
Andrew Vasanth Royappa

Report Number:
93-014
SYMBOLIC METHODS IN COMPUTER GRAPHICS
AND GEOMETRIC MODELING

A Thesis
Submitted to the Faculty

of

Purdue University

by

Andrew Vasanth Royappa

In Partial Fulfillment of the
Requirements for the Degree

of

Doctor of Philosophy

December 1992
To my beloved wife Sudha.
ACKNOWLEDGMENTS

This thesis is made possible by my advisor, Professor Chaudrajit Bajaj. He always challenged me to broaden my horizon just a little bit more, and I learned much about the process of scientific research from him. In addition, he provided me with an outstanding research environment and long-time financial support. To him I am deeply indebted.

I thank Professors Christoph M. Hoffmann, Shreeram M. Abhyankar and Greg N. Frederickson for serving on my advisory committee. At various times they have all rendered me valuable assistance. For their time and trouble, I am grateful. Professor Hoffmann's help is particularly appreciated; his careful reading of the manuscript and extensive comments have helped me a great deal.

Special thanks go to my friends Dr. Insung Ihm, Bill Bouma, Steve Swanson, and Dr. Guoliang Xu for a variety of technical assistance, including reviewing this manuscript, and many productive discussions. To the many other friends I made many during my stay at Purdue: thanks for all the good times.

I will always appreciate the emotional and material sacrifices my parents made on my behalf, to give me the opportunity for higher studies. This work was made possible through their commitment.

And finally, to my wife Sudha, for her love and endless patience, I dedicate this thesis.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>LIST OF FIGURES</th>
<th>vii</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>ix</td>
</tr>
<tr>
<td>1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Parameterization in Finite Precision</td>
<td>2</td>
</tr>
<tr>
<td>1.2 Accurately Displaying Parametric Curves and Surfaces</td>
<td>4</td>
</tr>
<tr>
<td>1.3 Original Results</td>
<td>4</td>
</tr>
<tr>
<td>1.4 Organization and Summary</td>
<td>6</td>
</tr>
<tr>
<td>2. APPROXIMATE PARAMETERIZATION</td>
<td>7</td>
</tr>
<tr>
<td>2.1 Introduction</td>
<td>7</td>
</tr>
<tr>
<td>2.1.1 Arithmetic Domains</td>
<td>7</td>
</tr>
<tr>
<td>2.1.2 Computational Approach</td>
<td>9</td>
</tr>
<tr>
<td>2.1.3 Structure</td>
<td>12</td>
</tr>
<tr>
<td>2.2 Conic Curves</td>
<td>12</td>
</tr>
<tr>
<td>2.2.1 Algorithm</td>
<td>13</td>
</tr>
<tr>
<td>2.2.2 Error Analysis</td>
<td>17</td>
</tr>
<tr>
<td>2.3 Quadric Surfaces</td>
<td>21</td>
</tr>
<tr>
<td>2.3.1 Algorithm</td>
<td>22</td>
</tr>
<tr>
<td>2.3.2 Error Analysis</td>
<td>25</td>
</tr>
<tr>
<td>2.4 Singular Cubic Curves</td>
<td>27</td>
</tr>
<tr>
<td>2.4.1 Algorithm</td>
<td>28</td>
</tr>
<tr>
<td>2.4.2 Error Analysis</td>
<td>30</td>
</tr>
<tr>
<td>2.5 Planar Monoid Curves</td>
<td>32</td>
</tr>
<tr>
<td>2.5.1 Algorithm</td>
<td>33</td>
</tr>
<tr>
<td>2.5.2 Error Analysis</td>
<td>35</td>
</tr>
<tr>
<td>2.6 Monoidal Surfaces</td>
<td>36</td>
</tr>
<tr>
<td>2.6.1 Algorithm</td>
<td>36</td>
</tr>
<tr>
<td>2.6.2 Error Analysis</td>
<td>37</td>
</tr>
</tbody>
</table>
2.7 Summary .................................................. 38

3. APPLICATIONS OF ERROR ANALYSIS .................. 39

3.1 Introduction ............................................. 39
3.2 Geometric Interpretation of Error .................... 39
3.2.1 Conics ............................................... 40
3.2.2 Quadrics ............................................ 41
3.2.3 Monoid Curves ...................................... 41
3.2.4 Monoidal Surfaces .................................. 46
3.2.5 Geometric Interpretation of Error: Summary .... 46
3.3 Geometric Error Bounds: Conics ..................... 46
3.3.1 Properties of Conics ................................ 47
3.3.2 Bounding the Orthogonal Distance ............... 48
3.4 Quadric Surface Parameterization .................... 53
3.4.1 Properties of Quadrics ............................. 54
3.4.2 Bounding the Orthogonal Distance ............... 55
3.5 Diophantine Solutions .................................. 56
3.5.1 Singular Cubics .................................... 56
3.5.2 Monoid Plane Curves ............................... 58
3.6 Summary .................................................. 59

4. PROJECTIVE REPARAMETERIZATIONS ..................... 63

4.1 Introduction ............................................. 63
4.2 Projective Linear Reparameterizations ............... 64
4.2.1 Total Mapping: Curves ............................ 65
4.2.2 The Complementary Segment ....................... 67
4.2.3 Parameter Compression ............................. 69
4.2.4 Total Mapping: Surfaces .......................... 71
4.3 Piecewise Finite Representations of Parametric Varieties ........................................ 74
4.4 Orientation of the Mappings ............................ 77
4.5 Projective Quadratic Reparameterizations .......... 79
4.5.1 Definitions .......................................... 79
4.5.2 Missing Points ..................................... 81
4.5.3 Normal Parameterizations ......................... 86
4.6 Single Finite Representation of Parametric Varieties ........................................ 90
4.7 Summary .................................................. 92

5. ROBUST DISPLAY OF ARBITRARY RATIONAL PARAMETRICS .... 100

5.1 Introduction ............................................. 100
5.2 Domain and Range Space Approaches ................ 102
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.3</td>
<td>Domain Sampling Using Rational Maps</td>
<td>103</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Domain Poles</td>
<td>104</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Domain Base Points</td>
<td>105</td>
</tr>
<tr>
<td>5.4</td>
<td>Accurate Display of Rational Parametric Curves</td>
<td>106</td>
</tr>
<tr>
<td>5.5</td>
<td>Accurate Display of Rational Parametric Surfaces</td>
<td>107</td>
</tr>
<tr>
<td>5.5.1</td>
<td>Surface Trimming</td>
<td>107</td>
</tr>
<tr>
<td>5.5.2</td>
<td>Domain Partitioning</td>
<td>109</td>
</tr>
<tr>
<td>5.5.3</td>
<td>Base Points and Seam Curves</td>
<td>113</td>
</tr>
<tr>
<td>5.5.4</td>
<td>Drawbacks and Future Work</td>
<td>116</td>
</tr>
<tr>
<td>5.5.5</td>
<td>Implementation Details</td>
<td>116</td>
</tr>
<tr>
<td>5.6</td>
<td>Summary</td>
<td>117</td>
</tr>
<tr>
<td>6.</td>
<td>CONCLUSIONS AND FUTURE WORK</td>
<td>127</td>
</tr>
<tr>
<td>6.1</td>
<td>Approximations in Exact Algebraic Algorithms</td>
<td>127</td>
</tr>
<tr>
<td>6.1.1</td>
<td>Parameterization Algorithms for Other Classes</td>
<td>127</td>
</tr>
<tr>
<td>6.2</td>
<td>Display of Rational Parametric Curves and Surfaces</td>
<td>128</td>
</tr>
<tr>
<td>6.2.1</td>
<td>Complex Parameter Values</td>
<td>129</td>
</tr>
<tr>
<td>6.2.2</td>
<td>Computing Triangulations on Surfaces</td>
<td>131</td>
</tr>
<tr>
<td>BIBLIOGRAPHY</td>
<td></td>
<td>134</td>
</tr>
<tr>
<td>VITA</td>
<td></td>
<td>143</td>
</tr>
</tbody>
</table>
### LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Numerical stability in monoid parameterizations</td>
<td>42</td>
</tr>
<tr>
<td>3.2</td>
<td>Perturbing the constant term in conic equations</td>
<td>60</td>
</tr>
<tr>
<td>3.3</td>
<td>Perturbing squared term in the conic equations</td>
<td>61</td>
</tr>
<tr>
<td>3.4</td>
<td>Orthogonal distance between concentric, coaxial conics</td>
<td>62</td>
</tr>
<tr>
<td>4.1</td>
<td>Total mapping of rational parametric curve (circle)</td>
<td>92</td>
</tr>
<tr>
<td>4.2</td>
<td>Total mapping of rational parametric curve (quartic)</td>
<td>93</td>
</tr>
<tr>
<td>4.3</td>
<td>An arc of a circle and its complementary segment</td>
<td>93</td>
</tr>
<tr>
<td>4.4</td>
<td>An arc of a quartic curve and its complementary segment</td>
<td>94</td>
</tr>
<tr>
<td>4.5</td>
<td>Discretization of rational parametric curve, with parameter compression</td>
<td>94</td>
</tr>
<tr>
<td>4.6</td>
<td>Discretization of polynomial parametric curve ($s^3, -s^2$)</td>
<td>95</td>
</tr>
<tr>
<td>4.7</td>
<td>Discretization of polynomial parametric curve ($s^2 - 1, s^2 - s^3$)</td>
<td>95</td>
</tr>
<tr>
<td>4.8</td>
<td>Total mapping of sphere using three rectangular patches</td>
<td>96</td>
</tr>
<tr>
<td>4.9</td>
<td>Total mapping of rational parametric surface (sphere)</td>
<td>96</td>
</tr>
<tr>
<td>4.10</td>
<td>Total mapping of rational parametric surface (steiner surface)</td>
<td>97</td>
</tr>
<tr>
<td>4.11</td>
<td>Use of projective quadratic transformation (circle)</td>
<td>97</td>
</tr>
<tr>
<td>4.12</td>
<td>Use of projective quadratic transformation (quartic)</td>
<td>98</td>
</tr>
<tr>
<td>4.13</td>
<td>Use of projective quadratic transformation (sphere)</td>
<td>98</td>
</tr>
<tr>
<td>4.14</td>
<td>Use of projective quadratic transformation (steiner surface)</td>
<td>99</td>
</tr>
<tr>
<td>5.1</td>
<td>Hyperboloid of 1 sheet with seam curve gaps</td>
<td>119</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>5.2</td>
<td>Partition of domain triangulation by pole curve</td>
<td>120</td>
</tr>
<tr>
<td>5.3</td>
<td>Image patches of domain triangles with pole vertices</td>
<td>121</td>
</tr>
<tr>
<td>5.4</td>
<td>Image patch of a domain triangle with a base point</td>
<td>122</td>
</tr>
<tr>
<td>5.5</td>
<td>Disjoint branches being wrongly connected (Mathematica)</td>
<td>123</td>
</tr>
<tr>
<td>5.6</td>
<td>Single-sheeted surface with domain poles (Maple)</td>
<td>123</td>
</tr>
<tr>
<td>5.7</td>
<td>Trimmed rational parametric sphere, and trim curves in domain</td>
<td>124</td>
</tr>
<tr>
<td>5.8</td>
<td>Trimmed hyperboloid of 2 sheets, and trim curves in domain</td>
<td>124</td>
</tr>
<tr>
<td>5.9</td>
<td>Rational parametric surface: cubic node</td>
<td>125</td>
</tr>
<tr>
<td>5.10</td>
<td>Rational parametric surface (two sheets)</td>
<td>125</td>
</tr>
<tr>
<td>5.11</td>
<td>Rational parametric surface (nine sheets)</td>
<td>126</td>
</tr>
<tr>
<td>6.1</td>
<td>Triangulation of a Steiner surface along curve singularities</td>
<td>133</td>
</tr>
<tr>
<td>6.2</td>
<td>Triangulation of parametric surface with point singularity</td>
<td>133</td>
</tr>
</tbody>
</table>
ABSTRACT


Certain restricted classes of algebraic curves and surfaces admit both parametric and implicit representations. Such dual forms are useful in computer graphics and geometric modeling since they combine the strengths of the two representations. We consider the problem of computing the rational parameterization of an implicit curve or surface in a finite precision domain. Current algorithms for this problem are based on classical algebraic geometry, and assume exact arithmetic involving algebraic numbers. After applying a careful analysis of the use of algebraic numbers in current algorithms, we develop new versions of these algorithms that are more efficient. Over a certain finite precision domain we can derive succinct algebraic and geometric error characterizations, from which we conclude that our versions of the algorithms are numerically robust.

A companion problem to parameterization is the accurate display of rational parametric curves and surfaces; we show how visualizing an arbitrary and possibly multiple-sheeted parametric surface is non-trivial. Such surfaces can have pole curves in their domain, where the denominators of the parameter functions vanish, and domain base points that correspond to entire curves on the surface. These are ubiquitous problems occurring even among the natural quadrics. Ordinary display techniques based on domain sampling often fail to visualize the true shape of the curve or surface. We first develop two ways of handling infinite parameter values, by using projective domain transformations. These results are then applied to the
display problem. We give algorithms for parametric curves and surfaces and discuss our implementation efforts. As an implementation vehicle we developed the graphical symbolic algebra system GANITH which allows rapid prototyping of algorithms that require a blend of symbolic computation, numerical computation, and three-dimensional graphics facilities.
1. INTRODUCTION

Algebraic curves and surfaces are widely used in geometric design, geometric modeling, and computer graphics. Such curves and surfaces are most often represented in parametric form, but the work of [97] has brought the implicit representation to the attention of researchers as a promising alternate. For some restricted families of curves and surfaces, both representations are possible. Such "dual form" curves and surfaces could be of great value in computer aided geometric design (CAGD) and related fields. It thus becomes important to convert between the forms when possible.

Briefly, an algebraic curve or surface can be defined implicitly by a polynomial equation in two or three variables, or in terms of rational parametric equations in one or two parameters. For instance, a surface can be defined by a single polynomial equation:

\[ f(x, y, z) = 0 \]

or by rational parametric equations with a common denominator:

\[
\begin{align*}
    x(s, t) &= \frac{X(s, t)}{W(s, t)} \\
    y(s, t) &= \frac{Y(s, t)}{W(s, t)} \\
    z(s, t) &= \frac{Z(s, t)}{W(s, t)}
\end{align*}
\]

where \( X(s, t), Y(s, t), Z(s, t), W(s, t) \) are polynomials.

Mathematical techniques from algebraic geometry have been applied to the problem of converting between the two forms. Converting from parametric to implicit form (implicitization), is always possible, using concepts from elimination theory such as resultants and Gröbner bases [74, 75, 97, 5, 6, 49, 23, 30, 27, 46, 78].

Converting from implicit to parametric form (parameterization) is possible for certain classes of curves and surfaces; algorithms for several parameterizable classes
have appeared in the recent literature [71, 1, 2, 3, 52, 56, 99, 93, 102, 103]. Conversion methods for both directions are generally expensive and involve symbolic computation.

In this thesis, we will first investigate some parameterization algorithms. After analyzing the behavior of these algorithms in the context of approximate arithmetic, we propose ways of formulating these algorithms to make them more stable and efficient.

We then turn to the problem of displaying rational parametric curves and surfaces. Although one of the chief strengths of parametric curves and surfaces is the ease of generating a piecewise-linear approximation by domain sampling, this is appropriate only for continuous portions of such curves and surfaces. In general, parametric curves and surfaces, including those that are the output of parameterization algorithms, exhibit a range of behavior that make their accurate and complete visualization non-trivial. For instance, rational parametric curves and surfaces can have domain poles (points where the denominators of the parametric functions vanish), real points on the curve or surface that correspond to complex parameter values, and unfaithful parameterizations. Rational parametric surfaces can have domain base points, which are points at which the numerators and denominator vanish simultaneously. We shall provide examples of simple, low-degree curves and surfaces possessing these properties, and that are not accurately viewable with conventional domain sampling techniques, e.g., those implemented in modern symbolic algebra systems or graphing programs.

In this thesis, we explain the shortcomings of the current methods and then provide solutions to these problems.

We now explain the problems to be addressed in some detail.

1.1 Parameterization in Finite Precision

Algebraic curves and surfaces are the most common vehicles for representing curved objects in geometric modeling. The coordinates of the points of such curves
and surfaces satisfy polynomial equations. The coefficients of the polynomials are chosen from some field; the points of the curve or surface then lie in the algebraic closure of this field. An algebraic curve or surface is called rational, if its points can be described by rational functions in some parameters. If the defining equation of a curve or surface is given, it is said to be in implicit form, and if (rational) parametric equations are given, it is said to be in parametric form.

Functionally, rational parameterization takes one implicit equation in \( n \) variables, and for each implicit variable produces a rational function in \( n - 1 \) parameters. Since the rational functions have a common denominator, the output can be coded as \( n + 1 \) polynomials.

Parameterization algorithms generally assume mathematically exact computations with numbers. In our work, we shall assume that the input consists of a polynomial with rational coefficients, although the algorithms themselves are specified over more general coefficient fields. Even with this restriction, in their intermediate steps the algorithms may need a root of some polynomial with rational coefficients. We shall henceforth refer to a root of a polynomial with rational coefficients as an algebraic number (strictly speaking, it is algebraic over the field of rational numbers) [51]. In general, it is rare for such a root to be as simple as an integer or a rational number, which are readily represented inside a computer (the latter as a pair of integers). For instance, the polynomial \( x^2 - 2 \) has as its only two roots the numbers \( \pm \sqrt{2} = \pm 1.414 \ldots \), where the decimal expansion is non-repeating and non-terminating. While symbolic methods exist to perform arithmetic on numbers such as these [34], they are expensive.

We therefore investigate the behavior of certain parameterization algorithms when rational numbers are used to approximate algebraic numbers. In this case, the output parametric equations will not exactly satisfy the implicit equation, and we seek meaningful connections between the two.
1.2 Accurately Displaying Parametric Curves and Surfaces

A well-known strength of the parametric representation of a curve or surface is the ease by which points on the curve or surface are generated. This is certainly true in geometric design, where only smooth segments or patches are considered. Visualizing an arbitrary rational parametric curve or surface is rather more difficult, especially if one applies the standard domain sampling techniques.

Rational parametric surfaces can have pole curves in the parameter domain, which are points at which the denominators of the rational parameter functions vanish, causing numerical problems and possibly surface discontinuities; base points in the domain, that correspond to entire curves on the surface; fundamental curves in the domain, that correspond to a single point on the surface; unfaithful parameterizations; real portions of the surface that are reachable only by complex-valued parameters. In addition, portions of a parametric curve or surface at finite distances may correspond to infinite parameter values. Ordinary domain sampling techniques (e.g. those implemented in commercial utilities such as Mathematica, MapleV or Gnuplot) do not always produce "good" piecewise-linear approximations to surfaces that have such features.

We do not address all the problems listed above, but propose methods that address the problems of infinite parameter values, domain discontinuities, and domain base points. Domain discontinuities and base points are ubiquitous, occurring even among simple surfaces such as the natural quadrics.

Most of the techniques we develop generalize in a straightforward way to curves and hypersurfaces of any dimension. We will also discuss in detail our experience with implementing some of our methods for rational parametric surfaces. Our implementations can display very complicated parametric surfaces.

1.3 Original Results

The major new results in this thesis are:
1. We analyze algorithms that parameterize implicits by taking families of lines through a selected point on the curve or surface and then intersecting these lines with the curve or surface. We derive precise error characterizations of the algorithms when operating in finite precision. The numerical stability of another type of parameterization algorithm for conics/quadrics, based on matrix methods, is briefly discussed in [56], wherein it is also remarked that the behavior of parameterization algorithms in general is not clearly understood when numerical approximations are used. Using the idea of backward error analysis [115], a method from numerical analysis, we compute algebraic error formulae. These are then applied to derive geometric error bounds and make geometrically meaningful statements about the relationship between the ideal output (when exact arithmetic is used) and the actual output (when finite precision is used). These results could lead to more numerically stable implementations of parameterization algorithms.

2. Algorithms for displaying parametric curves and surfaces that address the problems of infinite parameter values, using projective linear transformations of the parameter domain. We show how to represent an entire rational parametric curve or surface (whose parameters usually vary over infinite ranges) using only a finite number of bounded ranges. The idea of using projective linear transformations of the parameter domain is extended to projective quadratic transformations; using such transformations we can compute the so-called normal parameterizations of certain curves and surfaces, solving an open problem posed in [28].

3. Applying the above results on projective linear transformations of the parameter domain, we address the problem of computing good piecewise-linear approximations to rational parametric curves and surfaces, even when domain poles, discontinuities, and base points are present. We develop a new domain sampling technique and report on our implementation.
1.4 Organization and Summary

This thesis is organized as follows. In Chapter 2, we introduce the parameterization problem and explain our model of arithmetic. We develop new parameterization algorithms and give algebraic error formulae for the algorithms operating in this model. In Chapter 3, the error formulae are applied to derive interesting geometric properties and error bounds. In Chapter 4, we discuss ways of representing entire curves and surfaces over finite parameter regions, first multiple regions using projective linear domain transformations, and then a single region using projective quadratic transformations. Chapter 5 discusses robust methods of constructing piecewise-linear approximations to curves and surfaces, using some results of Chapter 4. Concluding remarks and a discussion of potential problems for future research are found in Chapter 6.

In this thesis we address certain problems in CAGD that are solvable using mathematical techniques from algebraic geometry. Our analysis and reformulation of parameterization algorithms add to a young field of research that could be called "numerical algebraic geometry." We hope that the careful analysis of error due to numerical approximations will prove useful to those attempting to make practical use of algorithms based on algebraic geometry. Furthermore, our new display algorithms make possible the efficient visualization of a larger variety of rational parametric curves and surfaces than hereto possible using standard techniques.

Parts of this work appear in [17, 18, 19]. We have implemented some of the results in the Ganith algebraic geometry toolkit [16].
2. APPROXIMATE PARAMETERIZATION

2.1 Introduction

In this chapter, we analyze algorithms for parameterizing low-degree curves and surfaces, when finite precision arithmetic is used. Mathematical techniques for parameterizing various classes of curves and surfaces based on algebraic geometry have existed for decades, and the recent parameterization algorithms are their computational analogues. In our work we pay particular attention to numerical error caused by approximations, and reformulate such algorithms to stably operate using bounded precision rational arithmetic.

We now discuss our assumptions and lay out some goals.

2.1.1 Arithmetic Domains

All parameterization algorithms share this characteristic: they take as input one polynomial, and produce as output a set of polynomials (equivalently, a set of rational functions with a common denominator). Mathematically, these algorithms require the coefficients of the input and output polynomials to lie in some field. The common choices for fields are the complex numbers, the reals, and the rationals. Of these three, the first two share the attractive property that the coefficients of the output polynomials can be made to lie in the same field as the coefficients of the input polynomial. However, the high complexity of coefficient arithmetic over these domains limits their applicability (complex numbers have other disadvantages).

The field of rational numbers is attractive for implementation because it is simple and arithmetic is closed over it. However, parameterizations may require irrational numbers, and it is therefore not possible to guarantee exactness of the output.
Most applications don't even use a field for computations. The most common coefficient domain in applications is a finite set, namely the set of floating point numbers in some base $\beta$ with $k$-digit mantissas and $e$-digit exponents. Arithmetic over this domain is not closed.

For our domain of coefficients, we will use the field of rational numbers. (In any implementation, the size of the rational numbers is grossly bounded by the size of the main memory, so this is not technically speaking a field).

It is to be understood that each input and output coefficient will be a rational number, and algebraic numbers will be approximated by rationals to some precision. All computations will take place in rational arithmetic. In the next chapter, we will develop methods that will in some cases allow us to meaningfully decide how much precision to use in computing a rational approximation to an algebraic number (as opposed to using some arbitrary user-specified precision).

As a result of our analysis, we can reformulate the algorithms so that they are stable and efficient (and in some cases simply formulas). Thus given the size of the input coefficients, and the size of the rational approximants, we can compute a bound on the size of numerators and denominators of all rational numbers that need to be used in any computation.

In fact, since our formulas only involve integral operations ($+, -, \times$), we might even question the use of rational numbers in an implementation (some separate "black-box" technique is assumed to be available for computing approximations of any given precision to an algebraic number). Rational number arithmetic involves integer gcd operations to put rationals in canonical form. This is inefficient for large integers. A cheaper alternative is to use fixed point computation, where the size of all numbers is fixed, as is the (decimal or binary) point. If all input (fixed point) coefficients are taken to be exact, the error analyses will stand. Once again, the size of the fixed point numbers can be bounded a-priori using the parameterization formulas.

There are thus various tradeoffs between the coefficient domains, which could form the basis for future investigations. We will from now on consider our algorithms as
operating in a finite precision arithmetic domain, in which all operations are carried out without error or overflow, once the input coefficient precision and the precision of algebraic number approximations is given.

2.1.2 Computational Approach

Over a finite precision arithmetic domain, one can expect the output of the parameterization algorithms to be almost always inexact. In a purely symbolic setting, such a computation is incorrect and therefore useless. In a numerical setting, guidelines are necessary to determine whether such incorrect output is still acceptable. The following characteristics of a numerical algorithm are desirable:

1. In the limit case, when using unbounded precision, the output must be exact.

2. Small perturbations in intermediate computations due to numerical approximations should only lead to small perturbations in the output.

3. The perturbations should not lead to global changes in the shape or properties of the output curve or surface.

The last characteristic may be of more or less value depending on the context. For instance, some parameterization algorithms have a choice of several mathematically equivalent transformations to apply. When numerical approximations are used, some choices lead to bigger topological changes than others. If parameterization is being applied only because some small piece of the curve or surface is of interest, and the output doesn’t change much in the locality of this piece, then the global changes will not matter. On the other hand, there might be situations where global properties are important and must be preserved.

We reformulate parameterization algorithms to satisfy these goals, based on the approach described below. A by-product of this approach are succinct error formulas, which are useful both for implementation purposes and for gaining geometric understanding of the effects of errors.
The parameterization algorithms discussed here generate algebraic numbers from the input coefficients and use them in subsequent computations. An algebraic number is (roughly) a root $\alpha$ of some polynomial $p(x)$ (with rational coefficients), so $p(\alpha) = 0$. In a finite precision domain, some approximation $\tilde{\alpha} \approx \alpha$ is used. In general, one can expect that $p(\tilde{\alpha}) \neq 0$.

Now, if an algorithm makes no assumptions about the value of a certain expression, a small perturbation in its value will generally not be harmful. However, if an algorithm assumes an expression to be identically zero, a slight perturbation in its value will invalidate this assumption. (This problem is endemic in geometric computation and discussed at length in [34, 33, 40, 58, 60, 80, 84, 107, 116, 117]. Although our work relates to geometric computation, the specific processes employed here are primarily algebraic in nature, and thus a different approach is required).

For instance, an algorithm may need to translate a point on a plane curve to the origin. Algebraically, this is done by computing the coordinates of a point on the curve, and applying a linear coordinate transformation to the algebraic equation of the curve. Mathematically, the equation of the curve in the new coordinate system must not contain a constant term since the equation must be satisfied by $(0,0)$. If the coordinates of the point were calculated inexactly, however, the transformed curve equation will still have a constant term, albeit small in magnitude.

As another example, suppose an algorithm applies a transformation to cancel the highest order term of an equation, but due to finite precision calculations, the new equation actually only has a minute but non-zero highest order term. Then a symbolic operation such as “calculate degree of polynomial,” which scans for the highest degree non-zero term of a polynomial, will return a wrong answer.

A reasonable approach to ensure correctness would be to eliminate coefficients that are “very small” with respect to the precision used. This requires some judgement by the user or programmer: a wrong notion of “small” might eliminate terms that are in fact valuable.
With this discussion in mind, we therefore settle on the following procedure which basically amounts to symbolically deleting from a calculation certain quantities that must vanish.

1. Symbolically eliminate from all intermediate calculations (and output) all expressions that are known to vanish if exact arithmetic was used.

2. Whenever possible, derive closed form expressions for the output, in terms of the input polynomial coefficients and certain numerical values calculated from them. This results in simple, efficient algorithms.

We focus throughout on the use of algebraic numbers, because they are the sole source of error in our model. Due to the choice of arithmetic domain, once a finite-precision rational approximation is made to an algebraic number (by truncating the algebraic number at some precision), all subsequent arithmetic will be performed exactly.

An algorithm is reformulated by considering it as a “pipeline” of steps: each step reads an input expression from a previous step, and sends an output expression to a subsequent one. Treating algebraic numbers as indeterminate quantities, we examine the input of each step and identify subexpressions that must mathematically vanish in these indeterminates. These subexpressions are symbolically eliminated, leaving a reformulated output expression. The next step is then examined, with this reformulated expression as input.

Using this analysis, we also compute, whenever possible, closed form expressions for the entire output of the algorithm, in terms of the input coefficients, and certain numbers computed directly from them.

We can apply this approach to several parameterization algorithms.

The expressions that are symbolically eliminated in the reformulation are interpreted using the technique of backward error analysis, to derive very simple algebraic characterizations of the error in the output. Later on it will be shown that these
algebraic characterizations have equally simple geometric meanings, and some useful applications.

Our goals are summarized below.

1. Reformulate parameterization algorithms so they can be efficiently and correctly implemented in finite precision arithmetic.

2. Avoid symbolic computation whenever possible.

3. Derive symbolic expressions for the error in the finite precision versions, and investigate the uses of such formulas.

2.1.3 Structure

In the rest of this chapter, we consider various parameterizable classes of algebraic curves and surfaces: conic curves, quadric surfaces, rational cubic curves, and higher-degree monoidal curves and surfaces. For each class, a parameterization algorithm is first described, and then analyzed for error.

The input curve or surface is always assumed to be irreducible.

2.2 Conic Curves

We consider the algorithm from [1] for parameterizing plane curves of degree two, i.e., conic sections. It begins by computing a point on the curve. In general, a line that passes through this point will intersect the curve at one other point, since a conic has two intersections with a line. Consider the one-parameter family or pencil of lines through this point: each parameter value corresponds to one line, which in turn intersects the conic at two points, one of which is the fixed point whose coordinates are known. The coordinates of the remaining point can be found as rational functions in the parameter whose coefficients involve the coordinates of the fixed point: this gives the rational parameterization of the conic. This algorithm is basically reformulated to satisfy our goals.
2.2.1 Algorithm

We consider conics in homogeneous form. This allows the use of both projective and affine transformations. Given the equation of a conic plane curve, parameter functions for the curve are derived. The parameter functions are given as closed form formulas in a parameter $t$, the coefficients of the curve, and the coordinates of a point on the curve.

INPUT. An irreducible conic curve given by the quadratic equation

$$f(x, y) = a_{20}y^2 + a_{11}xy + a_{02}x^2 + a_{10}y + a_{01}x + a_{00} = 0$$

OUTPUT. Rational functions $(x(t), y(t))$ of degree at most two, with $f(x(t), y(t)) = 0$.

METHOD.

1. Homogenize the conic. This yields the homogeneous equation

$$F(X, Y, W) = a_{20}Y^2 + a_{11}XY + a_{02}X^2 + a_{10}YW + a_{01}XW + a_{00}W^2 = 0$$

If the $X^2$, $Y^2$ or $W^2$ term is missing from the conic’s equation, then it will be linear in the corresponding variable, and can be immediately parameterized. Compute quadratic polynomials $X(t), Y(t), W(t)$ such that $F(X(t), Y(t), W(t)) = 0$, and go to step 4.

2. If all squared terms are present, apply a linear transformation that cancels one of these terms. The transformations are described below.

3. Parameterize the conic in the new coordinate system. This will give the coordinates of a general point of the transformed curve as rational functions of a single parameter. The inverse transformation is then applied to this parameterization. Three quadratic polynomials $X(t), Y(t), W(t)$ are derived such that $F(X(t), Y(t), W(t)) = 0$. Thus in general, a point on the curve will have projective coordinates $(X(t), Y(t), W(t))$ for some value of the parameter $t$. 
4. Returning to the affine domain, the parameterization for the affine conic is then
given by the ratios of the above polynomials.

\[
\begin{align*}
    x(t) &= \frac{x(t)}{w(t)} \\
y(t) &= \frac{y(t)}{w(t)}
\end{align*}
\]  

(2.1)

TRANSFORMATIONS. If all three squared terms are present, then any one of the
following three transformations may be used in step 2 of the conic parameterization
algorithm. Given a point on the conic, a transformation is computed that maps
this point to a known point in another coordinate system. In the new coordinate
system, the conic has a fixed parameterization that can be computed once and for all.
Applying the inverse transformation to this fixed parameterization yields the desired
parameter functions. In this reformulation of the algorithm, there is no computation
other than that of finding the coordinates of some point on the conic.

• To cancel the \( X^2 \) term, use the transformation

\[
\begin{align*}
    X &= bX_1 \\
y &= cX_1 + Y_1 \\
W &= dX_1 + W_1
\end{align*}
\]

(2.2)

where \((b, c, d)\) are the homogeneous coordinates of some point on the curve. This
transformation takes \((b, c, d)\) to the point \((1, 0, 0)\) in the new coordinate system.

For the transformation to be well-defined, \( b \) must be non-zero. Then, if \( d \neq 0 \), the
transformation is affine; otherwise it is projective. Since proportional homogeneous
coordinates represent the same point, the restriction \( d = 0 \) or \( d = 1 \) is made. If
\( d = 0 \), then we also make a restriction \( b = 1 \) or \( c = 1 \); since \( b \neq 0 \) is required for the
transformation to be well-defined, we will restrict \( b = 1 \) in this case.

Transforming \( F \) yields a new conic curve with equation \( F_1(X_1, Y_1, W_1) = 0 \), where

\[
F_1(X_1, Y_1, W_1) = F(bX_1, cX_1 + Y_1, dX_1 + W_1)
\]

\[
= F(b, c, d)X_1^2 + F_2(X_1, Y_1, W_1)
\]

In the latter, the subexpression \( F(b, c, d)X_1^2 \) must vanish identically, so we only need
consider the parameterization of \( F_2 \). This is a conic in \( X_1, Y_1 \) and \( W_1 \) with no \( X_1^2 \)
term:

$$F_2(X_1, Y_1, W_1) = (a_{10}d + 2a_{20}c + a_{11}b)X_1Y_1 + (2a_{00}d + a_{10}c + a_{01}b)Y_1W_1 + a_{20}Y_1^2 + a_{10}Y_1W_1 + a_{00}W_1^2$$

(2.3)

The point $(1,0,0)$ lies on the conic $F_2 = 0$. A one-parameter family of lines through this point is given by the equation $Y_1 = tW_1$. To intersect this family of lines with $F_2$, we substitute this equation into $F_2(X_1, Y_1, W_1)$:

$$F_2(X_1, tW_1, W_1) = (a_{10}d + 2a_{20}c + a_{11}b)X_1(tW_1) + (2a_{00}d + a_{10}c + a_{01}b)Y_1W_1 + a_{20}(tW_1)^2 + a_{10}(tW_1)W_1 + a_{00}W_1^2$$

$$= W_1(((a_{10}d + 2a_{20}c + a_{11}b)t + (2a_{00}d + a_{10}c + a_{01}b))X_1 + (a_{20}t^2 + a_{10}t + a_{00})W_1)$$

$$= 0$$

The intersection $W_1 = 0$ corresponds to the fixed point $(1,0,0)$; hence the other factor corresponds to a general point of the curve. The other factor is a homogeneous linear polynomial in $X_1, W_1$ and hence $F_2$ is immediately parameterized by the following formulas:

$$X_1(t) = a_{20}t^2 + a_{10}t + a_{00}$$
$$Y_1(t) = -(a_{10}d + a_{11}b + 2a_{20}c)t^2 - (2a_{00}d + a_{01}b + a_{10}c)t$$
$$W_1(t) = -(a_{10}d + a_{11}b + 2a_{20}c)t - (2a_{00}d + a_{01}b + a_{10}c)$$

(2.4)

In this parameterization for $F_2$, $b, c, d$ are indeterminate: the parameterization is correct regardless of their specific values. That is, even if $F(b, c, d) \neq 0$ as was assumed, this parameterization is still an exact parameterization for $F_2$. This is a crucial point, and will allow us to calculate error expressions since we can always assume $F_2(X_1, Y_1, W_1) = 0$ exactly, even in finite precision.

Now, recall that $(b, c, d)$ is a point on the curve $F(X, Y, W) = 0$. Then $F(b, c, d) = 0$, and $F_1(X_1, Y_1, W_1) = F_2(X_1, Y_1, W_1)$. Hence the parameterization (2.4) is also one
for $F_1(X_1, Y_1, W_1) = 0$. Applying the inverse linear transformation to this parameterization immediately yields one for the original conic:

\[
\begin{align*}
X(t) &= b(a_{20}t^2 + a_{10}t + a_{00}) \\
Y(t) &= -(a_{10}d + a_{11}b + a_{20}c)t^2 - (2a_{00}d + a_{01}b)t + a_{00}c \\
W(t) &= a_{20}dt^2 - (a_{11}b + 2a_{20}c)t - (a_{00}d + a_{01}b + a_{10}c)
\end{align*}
\]

To cancel the $Y^2$ term, the following transformation is applied, that maps the point $(b, c, d)$ to $(0, 1, 0)$.

\[
\begin{align*}
X &= X_1 + bY_1 \\
Y &= cY_1 \\
W &= dY_1 + W_1
\end{align*}
\]

Again, we take $(b, c, d)$ be the homogeneous coordinates of a point on the curve.

For the transformation to be well-defined, $c$ must be non-zero. As before, we make the restriction that $d = 0$ or $d = 1$. If $d = 0$, then we further restrict $c = 1$.

Lines through the point $(0, 1, 0)$ are given by the equation $X_1 = tW_1$. The computations are symmetric to the first case. The final parameter formulas are given below.

\[
\begin{align*}
X(t) &= -(a_{01}d + a_{11}c + a_{02}b)t^2 - (2a_{00}d + a_{10}c)t + a_{00}b \\
Y(t) &= c(a_{02}t^2 + a_{01}t + a_{00}) \\
W(t) &= a_{02}dt^2 - (a_{11}c + 2a_{02}b)t - (a_{00}d + a_{10}c + a_{01}b)
\end{align*}
\]

To cancel the $W^2$ term, the transformation below is applied.

\[
\begin{align*}
X &= X_1 + bW_1 \\
Y &= Y_1 + cW_1 \\
W &= dW_1
\end{align*}
\]

Let $(b, c, d)$ be the homogeneous coordinates of a point on the curve. In this case $d \neq 0$ is required for the transformation to be well-defined; hence, we restrict $d = 1$. Therefore this transformation is always affine: it is a translation. It translates the affine point $(b, c)$ to the affine origin $(0, 0)$. 
Lines through the origin are given by $Y_1 = tX_1$. The parameterization formulas are as below:

\begin{align*}
X(t) &= -(a_{10}d + a_{20}c + a_{11}b)t^2 - (a_{01}d + 2a_{02}b)t + a_{02}c \\
Y(t) &= a_{20}bt^2 - (a_{10}d + 2a_{20}c)t + (a_{01}d + a_{11}c + a_{02}b) \\
W(t) &= d(a_{20}t^2 + a_{11}t + a_{02})
\end{align*}

We have now completed in some detail the description of an algorithm to compute the rational parameterization of conic curves. If exact arithmetic is to be used, it does not matter greatly which of the above three transformations is applied. In practice, that would be determined by the computation of a point on the curve: a point at infinity would lead to one of the first two transformations, a finite point would lead to the third. However, in the context of finite precision arithmetic, we will later establish a definite preference for the third.

Finally, we note that scaling the input polynomial $f(x, y)$ by a constant doesn't affect the correctness of the output parameter functions, since all the coefficients $a_{ij}$ appear linearly in both numerator and denominator.

2.2.2 Error Analysis

The only computation in the algorithm given above is to derive the coordinates of a point on the input conic curve. Once these coordinates are found, the parameterization is given as a closed form formula in terms of those numbers and the coefficients of the input curve.

The coordinates of the point satisfy a polynomial equation (the implicit equation of the curve), and hence are algebraic numbers.

The algorithm takes as input a curve given by the affine equation $f(x, y) = 0$, and produces as output two rational functions $(x(t), y(t))$ that satisfy $f$. The functions $x(t)$ and $y(t)$ are specified in terms of algebraic numbers $b$ and $c$ (the value $d$ is always either 0 or 1 exactly). When approximations $\tilde{b}$ and $\tilde{c}$ are used in place of $b$ and $c$, the output of the algorithm will be rational parameter functions $\tilde{x}(t)$ and $\tilde{y}(t)$ such that $f(\tilde{x}(t), \tilde{y}(t)) \neq 0$. These rational parameter functions also correspond to
some algebraic curve. We would like to find the implicit equation of this new curve and compare it to the original input curve. This is the approach of backward error analysis.

We will state each such result as a lemma.

LEMMA 2.1 Let the first transformation above be used in computing the parameterization. Then the output parametric curve exactly satisfies the implicit equation

\[ f(x, y) = a_{20}y^2 + a_{11}xy + (a_{02} - \delta)x^2 + a_{10}y + a_{01}x + a_{00} = 0 \]

where the value \( \delta \) is given by

\[ \delta = \begin{cases} \frac{f(\hat{b}, \hat{c})}{\hat{b}^2} & \text{if } d = 1 \\ a_{20}\hat{c}^2 + a_{11}\hat{c} + a_{02} & \text{if } d = 0 \end{cases} \]

PROOF. The analysis begins by computing the value \( f(\hat{x}(t), \hat{y}(t)) \). Note that this value must vanish when exact arithmetic is used, since every point on the output (parametric) curve must be on the input (implicit) curve. However, in the presence of numerical approximations, it will be non-zero, and can be found symbolically. This value depends on which of the three coordinate transformations above was used. Let \( f(x, y) = 0 \) be the equation of a conic curve, as before, and let \( F(X, Y, W) \) be the homogeneous form of \( f \). The following relation between a polynomial \( f(X_1, \ldots, X_n) \) and its homogeneous form \( F(X_1, \ldots, X_n, W) \) will be useful:

\[ f\left(\frac{X_1}{W}, \ldots, \frac{X_n}{W}\right) = \frac{F(X_1, \ldots, X_n, W)}{W^2} \]

where \( W \) is the homogenizing variable.
Suppose the algorithm computed \((\tilde{b}, \tilde{c}, d)\) as an approximation to a point on the curve, and output approximate rational functions \((\tilde{x}(t), \tilde{y}(t))\). Compute \(f(\tilde{x}(t), \tilde{y}(t))\):

\[
f(\tilde{x}(t), \tilde{y}(t)) = \frac{f(\tilde{X}(t), \tilde{Y}(t))}{\tilde{W}(t)} = \frac{F(\tilde{X}(t), \tilde{Y}(t), \tilde{W}(t))}{\tilde{W}^2(t)}
\]

\[
= \frac{F(b\tilde{X}_1(t), c\tilde{X}_1(t) + \tilde{Y}_1(t), d\tilde{X}_1(t) + \tilde{W}_1(t))}{\tilde{W}^2(t)}
\]

\[
= \frac{F_1(\tilde{X}_1(t), \tilde{Y}_1(t), \tilde{W}_1(t))}{\tilde{W}^2(t)} \frac{F(\tilde{b}, \tilde{c}, d)\tilde{X}_1^2(t) + F_2(\tilde{X}_1(t), \tilde{Y}_1(t), \tilde{W}_1(t))}{\tilde{W}^2(t)}
\]

\[
= \frac{F(\tilde{b}, \tilde{c}, d)\tilde{X}_1^2(t)}{\tilde{W}^2(t)} \frac{F(\tilde{b}, \tilde{c}, d)\tilde{X}_1^2(t)}{\tilde{W}^2(t)}
\]

\[
= \frac{F(\tilde{b}, \tilde{c}, d)\tilde{x}(t)}{\tilde{b}^2}
\]

When \(d = 1\), \(F(\tilde{b}, \tilde{c}, d) = f(\tilde{b}, \tilde{c})\). When \(d = 0\), \(\tilde{b} = 1\), and \(F(\tilde{b}, \tilde{c}, d) = a_{20}\tilde{c}^2 + a_{11}\tilde{c} + a_{02}\).

Thus, when the algorithm uses the first transformation, \(f(\tilde{x}(t), \tilde{y}(t))\) equals

\[
\frac{f(\tilde{b}, \tilde{c})\tilde{x^2}(t)}{\tilde{b}^2} \text{ if } d = 1
\]

\[
(a_{20}\tilde{c}^2 + a_{11}\tilde{c} + a_{02})\tilde{x^2}(t) \text{ if } d = 0
\]

(2.8)

With the above in hand, we can now describe exactly what happens to the original input. The algorithm starts with the implicit equation \(f(x, y) = 0\) of a conic curve \(C\), and produces as output the parametric equations \((\tilde{x}(t), \tilde{y}(t))\) of another conic curve \(\tilde{C}\). \(\tilde{C}\) is an algebraic curve, and its implicit equation is found below.

Let \(\delta = \frac{f(\tilde{b}, \tilde{c})}{\tilde{b}^2}\) or \(\delta = a_{20}\tilde{c}^2 + a_{11}\tilde{c} + a_{02}\), depending on whether the transformation was affine or projective. Then by (2.8), \(f(\tilde{x}(t), \tilde{y}(t)) = \delta \tilde{x}(t)^2\). This proves the lemma.

\(\square\)

In the lemma, it was stated only that the output parameterization satisfied a certain implicit equation. This does not imply that the two curves are identical. Since every parametric curve corresponds to exactly one irreducible algebraic curve, \(\tilde{C}\) is
either identical to the conic curve defined by the equation \( \tilde{f}(x, y) = f(x, y) - \delta x^2 = 0 \), or shares one component with it. There is a polynomial in the coefficients of any conic (its discriminant) whose vanishing implies the conic degenerates to a pair of lines. The coefficients \( a_{20}, a_{02}, a_{00} \) appear linearly in the discriminant of \( \tilde{f}(x, y) \), so there is only one value of the real number \( \delta \) that could cause this degeneracy, and it is easily calculated.

Thus, the implicit equation \( \tilde{f}(x, y) = 0 \) given above will almost always (with probability one) be the implicit equation of the output parametric curve.

Similar computations yield the following results, stated without proof.

**Lemma 2.2** Let the second transformation above be used in computing the parameterization. Then the output parametric curve exactly satisfies the implicit equation

\[
\tilde{f}(x, y) = (a_{20} - \delta)y^2 + a_{11}xy + a_{02}x^2 + a_{10}y + a_{01}x + a_{00} = 0
\]

where the value \( \delta \) is given by

\[
\delta = \begin{cases} 
\frac{f(\tilde{b}, \tilde{c})}{\tilde{c}^2} & \text{if } d = 1 \\
20 + a_{11} \tilde{b} + a_{02} \tilde{b}^2 & \text{if } d = 0
\end{cases}
\]

\( \square \)

**Lemma 2.3** Let the third transformation above be used in computing the parameterization. Then \( d = 1 \) always, and the output parametric curve exactly satisfies the implicit equation

\[
\tilde{f}(x, y) = a_{20}y^2 + a_{11}xy + a_{02}x^2 + a_{10}y + a_{01}x + (a_{00} - \delta) = 0
\]

where the value \( \delta \) is given by

\[
\delta = f(\tilde{b}, \tilde{c})
\]

\( \square \)

Thus the algorithm computes exactly the parameterization of a perturbed input curve. The input curve is perturbed in precisely one of the coefficients \( a_{20}, a_{02}, \) or
depending on the transformation used. In each case a symbolic expression was found for the perturbation $\delta$.

We have assumed that all computations are carried out in rational arithmetic, and rational approximations were computed to certain algebraic numbers. The error due to the approximation is localized, as the value of the perturbation. The magnitude of this perturbation approaches zero as the precision of the approximation is increased so that the approximations approach the actual algebraic numbers.

Finally, we note that the above results are independent of any rescaling of the input equation by a non-zero constant. Such a scaling does not change the equation of the curve. A glance at the formulas for the perturbation $\delta$ shows that any rescaling of the input equation will be accounted for in $\delta$ and hence in the perturbed output equation.

2.3 Quadric Surfaces

The method of parameterizing a conic surface by intersecting it with lines through a fixed point also carries over to quadric surfaces, i.e., algebraic surfaces of degree two [1]. In fact, the method generalizes to degree two hypersurfaces of any dimension, since each line in a family of lines through a fixed point of the hypersurface will intersect the hypersurface at precisely one other point [14].

While the parameterization algorithm generalizes in a straightforward way, one subtlety arises in the analysis. In the case of curves, the output parameter functions were of degree two. Even if some error was incurred due to the use of approximations, we could be confident that the corresponding algebraic curve was a conic, since a second degree parametric curve corresponds to a second degree implicit curve.

Not so with parametric surfaces. A surface given by rational parameter functions of degree $n$ corresponds to an algebraic surface whose implicit equation can be of a degree as high as $n^2$. The output of the quadric parameterization algorithm is a rational surface given by (bivariate) parameter functions of degree two. If the output
is not exact, one might reasonably question whether the corresponding algebraic surface is actually cubic or quartic.

Our analysis enables us to prove that even when the output is approximate, the corresponding algebraic surface is always a quadric. This information would not be valuable if only a small piece of the surface was of interest, and the actual output surface doesn't differ much locally compared to the input surface. However, in some applications it may be important for the output surface to remain quadric, since the application may wish to take advantage of special properties of quadrics.

2.3.1 Algorithm

The exposition here is brief due to the similarity to the conic case. Homogeneous quadrics are considered as before, to allow affine or projective transformations. Given the equation of a quadric surface, parameter functions of degree at most two are derived, as closed form formulas in a parameter $t$, the coefficients of the curve, and the coordinates of a point on the curve.

INPUT. An irreducible quadric surface given by the quadratic equation

$$f(x, y, z) = a_{200}z^2 + a_{020}y^2 + a_{002}x^2 + a_{110}zx + a_{101}yz + a_{011}yx + a_{100}z + a_{010}y + a_{001}x + a_{000}$$

OUTPUT. Quadratic bivariate rational functions $(x(s, t), y(s, t), z(s, t))$ that satisfy $f(x(s, t), y(s, t), z(s, t)) = 0$.

METHOD.

1. Homogenize the quadric. This yields the homogeneous equation

$$F(X, Y, W) = a_{200}Z^2 + a_{020}Y^2 + a_{002}X^2 + a_{110}ZX + a_{101}ZY + a_{011}XY + a_{100}Z + a_{010}Y + a_{001}X + a_{000}W^2$$

$$= 0$$
If the $X^2$, $Y^2$, $Z^2$ or $W^2$ term is missing from the conic's equation, then it will be linear in the corresponding variable, and can be immediately parameterized. Compute quadratic polynomials $X(s, t), Y(s, t), Z(s, t), W(s, t)$ such that $F(X(s, t), Y(s, t), Z(s, t), W(s, t)) = 0$.

2. If all squared terms are present, apply a linear transformation that cancels one of these terms. The transformations are described below.

3. Dehomogenize the parameterization by $x(s, t) = \frac{X(s, t)}{W(s, t)}$, etc.

TRANSFORMATIONS. A coordinate transformation is applied that cancels one of the squared terms. As in the conic case, a quadric without a squared term in some variable must contain the point at infinity along the axis corresponding to that variable. For instance, a curve without the $X^2$ term will contain the point at infinity along the $X$ axis, namely $(1, 0, 0, 0)$. Given a point on the curve, a transformation can be constructed that will map it to a special point, namely one of $(1, 0, 0, 0), (0, 1, 0, 0), (0, 0, 1, 0),$ or $(0, 0, 0, 1)$.

To cancel the $X^2$ term, use the transformation

\[
\begin{align*}
X &= aX_1 \\
Y &= bX_1 + Y_1 \\
Y &= cX_1 + Z_1 \\
W &= dX_1 + W_1
\end{align*}
\]

where now $(a, b, c, d)$ are the homogeneous coordinates of some point on the curve. This transformation takes $(a, b, c, d)$ to the point $(1, 0, 0, 0)$ in the new coordinate system.

For the transformation to be well-defined, $a$ must be non-zero. As before, we make additional restrictions to assure unique values for the coordinates of a point. First, restrict either $d = 0$ or $d = 1$. If $d = 0$, restrict $a = 1$ so the transformation is well-defined.
Transforming \( F \) yields a new quadric surface with equation \( F_1(X_1, Y_1, Z_1, W_1) = 0 \), where

\[
F_1(X_1, Y_1, Z_1, W_1) = F(aX_1, bX_1 + Y_1, cX_1 + Z_1, dX_1 + W_1)
\]

\[
= F(a, b, c, d)X_1^2 + F_2(X_1, Y_1, Z_1, W_1)
\]

(2.10)

and \( F_2 \) is a conic in \( X_1, Y_1 \) and \( W_1 \) with no \( X_1^2 \) term. In the latter equation, the subexpression \( F(a, b, c, d)X_1^2 \) must vanish identically, so only the quadric \( F_2 \) needs to be parameterized.

Now \( F_2 = 0 \) is a quadric containing the point \((1,0,0,0)\); intersect the quadric with lines from a two-parameter family through this point. Such lines are given by equations \( Y_1 = sW_1, Z_1 = tW_1 \). These equations can be substituted into the equation \( F_2 = 0 \), to yield an equation with two factors: one corresponding to \( W_1 = 0 \) and the other a homogeneous linear polynomial in \( X_1, W_1 \), which can be solved for as polynomials in \( t \). This, coupled with the line equations above, yield a parameterization for a general point on \( F_2(X_1, Y_1, Z_1, W_1) = 0 \). Since \( F(a, b, c, d) = 0 \), by (2.10) it follows this is also a parameterization for the transformed curve \( F_1 = 0 \). Then the parameterization is transformed by the coordinate transformation (2.9) to yield one for the original quadric. The parameter formulas for the original surface are:

\[
X(s,t) = a(a_{200}t^2 + a_{110}st + a_{100}t + a_{020}s^2 + a_{010}s + a_{000})
\]

\[
Y(s,t) = a_{200}bt^2 - (a_{100}d + 2a_{200}c + aa_{101})st + a_{100}bt - (a_{010}d + a_{110}c + a_{020}b + aa_{011})s^2 - (2a_{000}d + a_{100}c + aa_{001})s + a_{000}b
\]

\[
Z(s,t) = -(a_{100}d + a_{200}c + a_{110}b + aa_{101})t^2 - (a_{010}d + 2a_{020}b + aa_{011})st - (2a_{000}d + a_{010}b + aa_{001})t + a_{020}cs^2 + a_{010}cs + aa_{001}
\]

\[
W(s,t) = a_{200}dt^2 + a_{020}ds^2 + (a_{110}ds - 2a_{200}c - a_{110}b - aa_{101})t + -(a_{110}c + 2a_{020}b + aa_{011})s - (a_{000}d + a_{100}c + a_{010}b + aa_{001})
\]

(2.11)

Similar formulas result when one of the other squared terms is eliminated instead of \( X_1^2 \). When the \( W^2 \) term is eliminated, the transformation is always affine, so \( d = 1 \) is always true. It corresponds to translating the affine point \((a,b,c)\) to the origin. Since the parameterization in this latter case has certain attractive features, as will
be shown later, we give the formulas for this case also:

\[ X(s, t) = a_{200}t^2 + a_{020}st + a_{110}st - (a_{100}d + 2a_{200}c + a_{110}b)t - (a_{010}d + a_{110}c + 2a_{020}b)st - (a_{001}d + a_{101}c + a_{011}b + a_{002}) \]

\[ Y(s, t) = a_{200}bt^2 - (a_{010}d + a_{110}c + a_{020}b + a_{001})s^2 - (a_{100}d + 2a_{100}c + a_{001})st + a_{101}bt - (a_{001}d + a_{101}c + 2a_{002})st + a_{002}b \]

\[ Z(s, t) = -(a_{100}d + a_{200}c + a_{110}b + a_{010})t^2 + a_{020}cs^2 - (a_{010}d + 2a_{020}b + a_{001})st - (a_{001}d + a_{011}b + 2a_{002})st + a_{011}cs + a_{002}c \]

\[ W(s, t) = d(a_{200}t^2 + a_{020}st + a_{110}st + a_{101}t + a_{011}s + a_{002}) \]

From the polynomial parameterization of the projective quadric, the rational parameterization \((x(s, t), y(s, t), w(s, t))\) of the affine quadric is achieved by dividing \(X(s, t), Y(s, t)\) and \(Z(s, t)\) by the common denominator \(W(s, t)\).

### 2.3.2 Error Analysis

As before, the only computation in the algorithm given above is to derive the coordinates of a point on the input quadric: once this is known, closed form formulas give a rational parameterization of the quadric. The coordinates of the point are algebraic numbers that may have to be approximated.

When an approximation \((\tilde{a}, \tilde{b}, \tilde{c}, \tilde{d})\) is used for the point on the quadric, three approximate rational functions \((\tilde{x}(s, t), \tilde{y}(s, t), \tilde{z}(s, t))\) are output (computed from the corresponding projective parameterization). We would like to find what algebraic surface these rational functions correspond to, and compare it to the original algebraic surface.

**LEMMA 2.4** Suppose the transformation cancelling the \(X^2\) term was applied. Then the output parametric surface exactly satisfies the implicit equation

\[ f(x, y, z) = a_{200}z^2 + a_{020}y^2 + (a_{002} - \delta)x^2 + a_{110}zy + a_{101}zx + a_{011}yx + a_{000} \]

\[ = a_{100}x + a_{010}y + a_{001}z + a_{000} \]

\[ = 0 \]
where the value $\delta$ is given by

$\delta = \begin{cases} \frac{f(\tilde{a}, \tilde{b}, \tilde{c})}{\tilde{a}^2} & \text{if } d = 1 \\ c^2a_{200} + bca_{110} + ca_{101} + b^2a_{020} + ba_{011} + ao_{02} & \text{if } d = 0 \end{cases}$

PROOF. We begin by computing the value $f(\tilde{x}(s, t), \tilde{y}(s, t), \tilde{z}(s, t))$. This depends on the specific coordinate transformation used. Suppose the transformation cancelling the $X^2$ term was used. Then the computation is as before:

$$f(\tilde{x}(s, t), \tilde{y}(s, t), \tilde{z}(s, t)) = \frac{\tilde{X}(s, t) \tilde{Y}(s, t) \tilde{Z}(s, t)}{\tilde{W}(s, t)}$$

$$= \frac{F'(\tilde{X}(s, t), \tilde{Y}(s, t), \tilde{Z}(s, t), \tilde{W}(s, t))}{\tilde{W}^2(s, t)}$$

$$= \frac{F(\tilde{a}\tilde{X}_1(s, t), \tilde{b}\tilde{Y}_1(s, t) + \tilde{Y}_1(s, t), \tilde{c}\tilde{X}_1(s, t) + \tilde{Z}_1(s, t), \tilde{d}\tilde{X}_1(s, t) + \tilde{W}_1(s, t))}{\tilde{W}^2(s, t)}$$

$$= \frac{F(\tilde{a}, \tilde{b}, \tilde{c}, \tilde{d})\tilde{X}_1^2(s, t) + F_2(\tilde{X}_1(s, t), \tilde{Y}_1(s, t), \tilde{Z}_1(s, t), \tilde{W}_1(s, t))}{\tilde{W}^2(s, t)}$$

When $d = 1$, $F(\tilde{a}, \tilde{b}, \tilde{c}, \tilde{d}) = f(\tilde{a}, \tilde{b}, \tilde{c})$. When $d = 0$, $\tilde{a} = 1$, and $F(\tilde{a}, \tilde{b}, \tilde{c}, \tilde{d}) = c^2a_{200} + bca_{110} + ca_{101} + b^2a_{020} + ba_{011} + ao_{02}$. Thus, when the algorithm uses the first transformation, $f(\tilde{x}(s, t), \tilde{y}(s, t), \tilde{z}(s, t))$ equals

$$f(\tilde{a}, \tilde{b}, \tilde{c})\tilde{x}^2(s, t) \quad \text{if } d = 1$$

$$(c^2a_{200} + bca_{110} + ca_{101} + b^2a_{020} + ba_{011} + ao_{02})\tilde{x}^2(s, t) \quad \text{if } d = 0 \quad (2.12)$$

As in the conic case, we can now see that the output parametric surface corresponds to an algebraic surface that is quadratic. Suppose the transformation cancelling $X^2$ was used. Then every point $(\tilde{x}(s, t), \tilde{y}(s, t), \tilde{z}(s, t))$ of the output parameterization satisfies the equation

$$f(\tilde{x}(s, t), \tilde{y}(s, t), \tilde{z}(s, t)) = \delta\tilde{x}^2(s, t)$$

where the number $\delta$ is the appropriate coefficient of $\tilde{x}^2(s, t)$ in the equations (2.12). □
As for conics, there is one value of \( \delta \) for which the above quadric degenerates into a pair of planes, in which case the output parameterization will parameterize one of those planes.

The results are similar for the transformations eliminating \( Y^2 \) and \( Z^2 \). When \( W^2 \) is cancelled, the result is

\[
 f(\tilde{x}(s,t), \tilde{y}(s,t), \tilde{z}(s,t)) = f(\tilde{a}, \tilde{b}, \tilde{c})
\]  

(2.13)

Each of the other transformations will yield an output quadric that corresponds to perturbation of the original quadric. The perturbation will depend on the transformation used, and will be in the coefficient of the variable eliminated. In the case of eliminating \( W^2 \), the perturbation will be in the \( a_{000} \) coefficient by \( \delta = f(\tilde{a}, \tilde{b}, \tilde{c}) \).

Once again the results are unchanged if the input equation is scaled by a non-zero constant.

2.4 Singular Cubic Curves

Given the equation of a cubic plane curve, parameter functions for the curve are derived [2]. Singular cubics are also addressed in the section on monoid curves. In this section, we adapt the algorithm to our method, and analyse its use of algebraic numbers. The analysis enables us to point out a deficiency of this technique for singular cubics, especially if numerical approximations to algebraic numbers are used. We will conclude by prescribing the monoid parameterization algorithm for this case.

The parameter functions could be given as closed form formulas in the parameter \( t \), the coefficients of the curve, and certain other numbers. These expressions are unwieldy, and instead we show how they can be derived. The derivation assumes exact arithmetic, and a subsequent analysis reveals the error in the output when approximations are made. Our description of the algorithm is tailored to suit the error analysis, but it follows [2] in the essentials.
2.4.1 Algorithm

INPUT. A cubic plane curve given by the cubic equation
\[ f(x, y) = a_3 y^3 + a_2 x y^2 + a_1 y^2 + a_0 y + a_3 x^2 + a_2 x y + a_1 x y + a_0 x + a_0 \]
\[ = 0 \]

OUTPUT. Rational functions \((x(t), y(t))\) of degree at most four, such that
\[ f(x(t), y(t)) = 0. \]

METHOD. As in the conic case, the curve is transformed into a birationally equivalent one that is readily parameterizable. Several birational transformations are used. The steps are detailed below. If the cubic has a zero \(x^3\) or \(y^3\) term, the first step can be omitted. If both are present, then the first step below cancels \(y^3\), and the steps that follow assume that. The \(x^3\) term could just as well be cancelled: the computation is symmetric.

1. Apply a transformation that removes the \(y^3\) term of \(f\). This can be done via the linear transformation
\[
\begin{align*}
  x &= x_1 + q y_1 \\
  y &= y_1
\end{align*}
\]  
(2.14)

When applied to the cubic equation \(f(x, y) = 0\), this yields a new cubic curve with equation \(f_1(x_1, y_1) = 0\), in which
\[
\begin{align*}
  f_1(x_1, y_1) &= f(x_1 + q y_1, y_1) \\
  &= L(q) y_1^2 + f_2(x_1, y_1)
\end{align*}
\]  
(2.15)

where \(L(q) = a_3 q^3 + a_1 q^2 + a_2 q + a_0\). Choose \(q\) to be a root of \(L\), i.e. \(L(q) = 0\). Then \(f_1(x_1, y_1) = f_2(x_1, y_1)\) because the subexpression \(L(q) y_1^3\) must vanish; we only consider parameterizing the curve with equation \(f_2(x_1, y_1) = 0\).

2. Parameterize the cubic curve with equation \(f_2(x_1, y_1) = 0\), which has no \(y_1^3\) term. In order to do this, \(f_2\) is transformed into a quadratic. \(f_2\) is of the form
\[
\begin{align*}
  f_2(x_1, y_1) &= g_1(x_1) y_1^2 + g_2(x_1) y_1 + g_3(x_1)
\end{align*}
\]  
(2.16)
where \( g_1, g_2, g_3 \) have degrees equal to their subscripts. The discriminant of \( f_2 \) (with respect to \( y_1 \)) is simply \( g_4(x_1) = g_2(x_1)^2 - 4g_1(x_1)g_3(x_1) \). The roots of the discriminant are projections of the extreme and singular points of the curve, in the direction of the \( y_1 \)-axis, hence if the curve has a singular point, the discriminant will have a multiple root. By performing the following substitution

\[ y_2 = 2g_1 y_1 + g_2 \]  

we have

\[ 4g_1 f_2 = 4g_1^2 y_1^2 + 4g_1 g_2 y_1 + 4g_1 g_3 = (2g_1 y_1 + g_2)^2 - (g_2^2 - 4g_1 g_3) \]  

\[ = y_2^2 - g_4 \]

Note that \( g_4(x_1) \) is a polynomial in \( x_1 \) of degree at most four. The curve is singular (and hence rational) if and only if \( g_4(x_1) \) has a multiple root. This repeated root can be real or complex; only the real case is considered here.

Now for any number \( r \), expand the polynomial \( g_4(x_1) \) in a Taylor series at \( r \). Then

\[ g_4(x_1) = g_4(r) + g_4'(r)(x_1 - r) + \frac{g_4''(r)}{2}(x_1 - r)^2 + \frac{g_4^{(3)}(r)}{3!}(x_1 - r)^3 + \frac{g_4^{(4)}(r)}{4!}(x_1 - r)^4 \]

The terms of order higher than 4 are identically zero, \( g_4 \) being a polynomial of degree 4. Collecting coefficients of \( (x_1 - r)^2 \) yields

\[ g_4(x_1) = q_2(x_1)(x_1 - r)^2 + g_4'(r)(x_1 - r) + g_4(r) \]  

(2.19)

where \( q_2(x_1) \) is of degree two. Now apply the substitution

\[ y_3 = y_2 / (x_1 - r) \]  

(2.20)

together with (2.19) into the right-hand side of (2.16); this leads to

\[ 4g_1 f_1 = y_3^2 - g_4(x_1) \]  

\[ = (y_3^2 - q_2(x_1))(x_1 - r)^2 + g_4'(r)(x_1 - r) + g_4(r) \]  

(2.21)

\[ = f_3(x_1, y_3) \]
Choose \( r \) to be a multiple root of \( g_3(x_1) \): then \( g_4(r) = g_4'(r) = 0 \), and the subexpression \( g_4'(r)(x_1 - r) + g_4(r) \) must vanish. Therefore \( f_3(x_1, y_3) = (y_3^2 - q_2(x_1))(x_1 - r)^2 \). This suggests that to parameterize \( f_3(x_1, y_3) \), we can simply parameterize the curve with equation \( C(x_1, y_3) = y_3^2 - q_2(x_1) = 0 \). This curve is a conic since \( q_2(x_1) \) is of degree two.

3. Parameterize the conic with equation \( C(x_1, y_3) = 0 \) using the methods of the previous section. This yields a pair of rational functions \((x_1(t), y_3(t))\) such that \( C(x_1(t), y_3(t)) = 0 \). Then apply transformations (2.20), (2.17), (2.14) in reverse to find a pair of rational functions \((x(t), y(t))\) that parameterize the input curve. These rational functions will have a common denominator, with degree of numerators and denominator not exceeding four.

2.4.2 Error Analysis

By carefully counting the degrees of various polynomials in the algorithm and performing some simple calculations, we can verify that the numerator and denominator of the output rational functions can be of degree four. However, if exact arithmetic is used, the output curve still corresponds to the input cubic, and not to a quartic curve. Therefore, the numerators and common denominator of the rational functions must have a root in common. This root must be eliminated by dividing out the corresponding linear factor, but computing this factor exactly is problematic (see [56] for a supporting example). However, it might be possible to approximate this factor cheaply.

The cubic parameterization calls for computing a root \( q \) of the cubic polynomial \( L(q) \), a multiple root \( r \) of the quartic polynomial \( g_4(x_1) \), and a parameterization \((x_1(t), y_3(t))\) of the conic with equation \( C(x_1, y_3) = 0 \). Assuming the third conic transformation of the previous section was used, a pair of algebraic numbers \((b, c)\) need to be computed.
If all computations were exact, i.e. \( L(q) = g_4(r) = C(x_1(t), y_3(t)) = 0 \), then the output will be correct. However, one may need to use approximations \( \tilde{q}, \tilde{r} \) and \((\tilde{x}_1(t), \tilde{y}_3(t))\), which will lead to an approximate output parameterization \((\tilde{x}(t), \tilde{y}(t))\). In this case one must measure the error incurred. Once again, a backward error analysis will be performed, beginning with back-substitution.

**Lemma 2.5** The output parameterization will satisfy the implicit equation

\[
\tilde{f}(x, y) = f(x, y) - \left( L(\tilde{q})y^3 + \frac{C(\tilde{b}, \tilde{c})(x - \tilde{q}y - \tilde{r})^2 + g_4'\tilde{r}(x - \tilde{q}y - \tilde{r}) + g_4(\tilde{r})}{4g_1(x - \tilde{q}y)} \right)
\]

\[= 0\]

**Proof.** Given the approximate output parameterization \((\tilde{x}(t), \tilde{y}(t))\), we compute

\[f(\tilde{x}(t), \tilde{y}(t))\]. The subscript \((t)\) is dropped for convenience.

\[
f(\tilde{x}, \tilde{y}) = f(x_1 + \tilde{q}y_1, y_1) \quad \text{by (2.14)}
\]

\[= f_1(x_1, y_1)\]

\[= L(\tilde{q})y_1^3 + f_2(x_1, y_1) \quad \text{by (2.15)}\]

Continuing,

\[
f_2(\tilde{x}_1, \tilde{y}_1) = f_2(\tilde{x}_1, \tilde{y}_2 - g_2(\tilde{x}_1)) \quad \text{by (2.17)}
\]

\[= g_1(\tilde{x}_1) \left( \frac{\tilde{y}_2 - g_2(\tilde{x}_1)}{2g_1(\tilde{x}_1)} \right)^2 + g_2(\tilde{x}_1) \left( \frac{\tilde{y}_1 - g_2(\tilde{x}_1)}{2g_1(\tilde{x}_1)} \right) + g_3(\tilde{x}_1) \quad \text{by (2.16)}
\]

\[= \frac{\tilde{y}_2^2 - g_4(\tilde{x}_1)}{4g_1(\tilde{x}_1)} \quad \text{by (2.17)}
\]

\[= \frac{(\tilde{y}_3^2 - q_2(\tilde{x}_1))(\tilde{x}_1 - \tilde{r})^2 + g_4'(\tilde{r})(\tilde{x}_1 - \tilde{r}) + g_4(\tilde{r})}{4g_1(\tilde{x}_1)} \quad \text{by (2.19)}
\]

Now \( C(\tilde{x}_1(t), \tilde{y}_3(t)) = \tilde{y}_3^2 - q_2(\tilde{x}_1) \), and since we assumed that the third conic transformation was used to parameterize \( C \), it follows that there is a point \((\tilde{b}, \tilde{c})\) such that \( C(\tilde{x}_1(t), \tilde{y}_3(t)) = C(\tilde{b}, \tilde{c}) \). Therefore,

\[
f(\tilde{x}, \tilde{y}) = L(\tilde{q})\tilde{y}_1^3 + \frac{(\tilde{y}_3^2 - q_2(\tilde{x}_1))(\tilde{x}_1 - \tilde{r})^2 + g_4'(\tilde{r})(\tilde{x}_1 - \tilde{r}) + g_4(\tilde{r})}{4g_1(\tilde{x}_1)}
\]

\[= L(\tilde{q})\tilde{y}_1^3 + \frac{C(\tilde{b}, \tilde{c})(\tilde{x}_1 - \tilde{r})^2 + g_4'(\tilde{r})(\tilde{x}_1 - \tilde{r}) + g_4(\tilde{r})}{4g_1(\tilde{x}_1)}
\]

\[= L(\tilde{q})\tilde{y}_1^3 + \frac{C(\tilde{b}, \tilde{c})(\tilde{x} - \tilde{q}\tilde{y} - \tilde{r})^2 + g_4'(\tilde{r})(\tilde{x} - \tilde{q}\tilde{y} - \tilde{r}) + g_4(\tilde{r})}{4g_1(\tilde{x} - \tilde{q}\tilde{y})}
\]
This proves the lemma. □

If the values \( \bar{q}, \bar{r}, \bar{b}, \bar{c} \) are exact, then \( L(\bar{q}) = g_{4}(\bar{r}) = g'_{4}(\bar{r}) = \mathcal{G}(\bar{b}, \bar{c}) = 0 \), and it is clear that the parametric output curve coincides with the implicit input curve.

However, if the values are not exact, the output curve differs from the input curve. The coefficient perturbations are now present in many terms, not just one. The factor \( g_{1}(x - \bar{q}y) \) in the denominator may be linear, so according to the analysis the output curve may even be quartic. In this case, numerical perturbations will cause the numerators and denominator of the rational functions not to have a common linear factor, as explained earlier, and the curve will indeed have degree four parametric equations and thus be quartic.

This may or may not be desirable, depending on whether an approximation is interested only in a small piece of a curve, or whether the entire curve is to be used in some future computation. If it is important, one could try to eliminate the "approximate" common factor by numerically computing roots of numerators and denominators and then choosing those that match best. This approach is reasonable because it is known beforehand that this "approximate" common factor exists.

2.5 Planar Monoid Curves

The method of parameterizing a curve by intersecting it with lines through a fixed point is applicable to any curve which has a point such that most lines through this point intersect the curve at only one other point. Such curves are called monoids. A rational cubic curve is the lowest-degree non-trivial monoid, since it has a double point (conics are trivially monoids by the above definition). Lines through the double point will intersect the cubic at one other point, whose coordinates can be found, giving a rational parameterization of the cubic.

A monoid curve given by an equation \( f(x, y) = 0 \) of degree \( n \) will always have a point of multiplicity \( n - 1 \) [111]. This will be the only multiple point: if there was another (of any multiplicity > 1), a line through these two points will intersect the
curve in more than \( n \) points, counting intersection multiplicities, and thus violate Bezout's theorem.

2.5.1 Algorithm

The multiple point on the monoid curve can be found by equating to zero all partial derivatives of \( f(x, y) \) up to the \((n - 2)\)'th order, since they must all vanish at this point. The multiple point can be either at finite distances or at infinity, so the system of equations above must be homogenized with a variable \( W \), a solution with \( W = 0 \) corresponding to a multiple point at infinity. All the details below are straightforward in the projective case, so we only consider the affine curve.

For the rest of this section, in all sums with indices \( i, j \), we assume \( i, j \geq 0 \).

INPUT. A plane monoid curve of degree \( n \) given by the equation

\[
f(x, y) = \sum_{i+j=0}^{n} a_{ij}x^iy^j = 0
\]

OUTPUT. Rational functions \((x(t), y(t))\) of degree at most \( n \), with \( f(x(t), y(t)) = 0 \).

METHOD.

1. Compute the coordinates of the multiple point.

2. Translate the curve so this point is at the origin.

3. Parameterize the curve in the new coordinate system, and translate back to the original coordinate system.

Let \((b, c)\) be the coordinates of the multiple point computed in the first step. The translation taking this point to the origin is

\[
x = x_1 + b \\
y = y_1 + c
\] (2.22)
In the new coordinate system, the curve has equation \( f_1(x_1, y_1) = f(x_1 + b, y_1 + c) = 0 \). Expanding \( f_1(x_1, y_1) \) in a Taylor series at \((x_1, y_1) = (0,0)\) and using the chain rule yields

\[
\begin{align*}
f_1(x_1, y_1) &= \sum_{i+j=0}^{n} \frac{1}{i!j!} \frac{\partial^{i+j} f_1}{\partial x^i \partial y^j}(0,0) x_1^i y_1^j \\
&= \sum_{i+j=0}^{n} \frac{1}{i!j!} \frac{\partial^{i+j} f}{\partial x^i \partial y^j}(b, c) x_1^i y_1^j \\
&= \sum_{i+j=0}^{n-2} \frac{1}{i!j!} \frac{\partial^{i+j} f}{\partial x^i \partial y^j}(b, c) x_1^i y_1^j + f_2(x_1, y_1)
\end{align*}
\]

where \( f_2(x_1, y_1) \) is a polynomial with no terms of degree lower than \( n - 1 \). The first expression of the right hand side of the last equation above must vanish, since all the partials of \( f \) up to the \((n - 2)\)'nd vanish at \((b, c)\).

Hence we seek a parameterization of the curve with equation \( f_2(x_1, y_1) = 0 \). This curve is of degree \( n \) with a point of multiplicity \( n - 1 \) at the origin, and so lines through the origin intersect it at exactly one other point. Intersecting it with lines \( y_1 = tx_1 \) through the origin, we find that

\[
\begin{align*}
f_2(x_1, ty_1) &= \sum_{i+j=n-1}^{n} \frac{1}{i!j!} \frac{\partial^{i+j} f}{\partial x^i \partial y^j}(b, c) x_1^i (tx_1)^j \\
&= x_1^{n-1} \left\{ \sum_{i+j=n-1}^{n} \frac{1}{i!j!} \frac{\partial^{i+j} f}{\partial x^i \partial y^j}(b, c) t^j + x_1 \sum_{i+j=n}^{n} \frac{1}{i!j!} \frac{\partial^{i+j} f}{\partial x^i \partial y^j}(b, c) t^j \right\}
\end{align*}
\]

The factor \( x_1^{n-1} \) corresponds to the intersection at the origin. The other can be solved for as

\[
x_1 = -\sum_{i+j=n-1}^{n} \frac{1}{i!j!} \frac{\partial^{i+j} f}{\partial x^i \partial y^j}(b, c) t^j
\]

and of course, \( y_1 = tx_1 \). At this point, \( x_1 \) and \( y_1 \) are rational functions with common denominator and degree at most \( n \). They are an exact rational parameterization of the curve \( f_2(x_1, y_1) = 0 \), and hence also of the curve \( f_1(x_1, y_1) = 0 \).
To find the parameterization in the original coordinate system, we transform back using (2.22):

\[
x = x_1 + b - \sum_{i+j=n-1} \frac{1}{i!j!} \frac{\partial^{i+j} f}{\partial^i x \partial^j y}(b,c)t^i + b \sum_{i+j=n} \frac{1}{i!j!} \frac{\partial^{i+j} f}{\partial^i x \partial^j y}(b,c)t^i \\
y = y_1 + c - t \sum_{i+j=n-1} \frac{1}{i!j!} \frac{\partial^{i+j} f}{\partial^i x \partial^j y}(b,c)t^i + c \sum_{i+j=n} \frac{1}{i!j!} \frac{\partial^{i+j} f}{\partial^i x \partial^j y}(b,c)t^i
\]

(2.23)

Therefore, rational functions \((x(t), y(t))\) are found that rationally parameterize the monoid \(f(x, y) = 0\). While the expressions are not in closed form, they could be easily so written for any given \(n\). Their degree in numerator and denominator is at most \(n\).

2.5.2 Error Analysis

Suppose approximations \((\tilde{b}, \tilde{c})\) are used for the singular point \((b, c)\). Then the output parameterization will be approximate rational functions \((\tilde{x}(t), \tilde{y}(t))\).

**Lemma 2.6** The approximate output curve satisfies the implicit equation

\[
\tilde{f}(x, y) = f(x, y) - \sum_{i+j=0}^{n-2} \frac{1}{i!j!} \frac{\partial^{i+j} f}{\partial^i x \partial^j y}(\tilde{b}, \tilde{c})(x - \tilde{b})^i(y - \tilde{c})^j = 0
\]

**Proof.** We compute the implicit equation of the approximate output curve, in the usual manner.

\[
f(\tilde{x}(t), \tilde{y}(t)) = f(\tilde{x}_1(t) + \tilde{b}, \tilde{y}_1(t) + \tilde{c}) = \sum_{i+j=0}^{n-2} \frac{1}{i!j!} \frac{\partial^{i+j} f}{\partial^i x \partial^j y}(\tilde{b}, \tilde{c})\tilde{x}_1^i\tilde{y}_1^j + f_1(\tilde{x}_1, \tilde{y}_1)
\]
Again the crucial step is the second: we have used the fact that the parameterization formulas for \( f_2(x_1, y_1) \) are exact for all \((\tilde{b}, \tilde{c})\). This proves the lemma. \( \square \)

2.6 Monoidal Surfaces

A monoidal surface of degree \( n \) has on it a point of multiplicity \( n - 1 \). That is, most lines through this point intersect the surface \( n - 1 \) times at that point, and once elsewhere. Unlike the curve case, a surface can have more than one such multiple point, but a bounded number. For instance, a cubic surface or cubicoid may have up to four double points. A line through two of these four points will then lie entirely on the surface.

2.6.1 Algorithm

Monoidal surfaces are obviously rational since their rational parameterization can be derived in a manner similar to that for monoid curves, by intersecting them with a two-parameter family of lines through the special point. Once again, a degree \( n \) surface will intersect each line at one other point, whose coordinates can then be calculated.

For the rest of this section, in all sums with indices \( i, j, k \), we assume \( i, j, k \geq 0 \). We omit the intermediate calculations and show the output parameterization expressions.

INPUT. A monoidal surface of degree \( n \) given by the equation

\[
f(x, y, z) = \sum_{i+j+k=0}^{n} a_{ijk} x^i y^j z^k = 0
\]

OUTPUT. Rational functions \((x(s, t), y(s, t), z(s, t))\) of degree at most \( n \), such that \( f(x(s, t), y(s, t), z(s, t)) = 0 \).

METHOD.
1. Compute the coordinates of the multiple point.

2. Translate the surface so this point is at the origin.

3. Parameterize the surface in the new coordinate system, and translate back to the original coordinate system.

Let the coordinates of the multiple point be \((a, b, c)\). To parameterize the surface in the new coordinate system \(x_1, y_1, z_1\) where this multiple point is at the origin, it is intersected with the two-parameter family of lines given by the equations \(y_1 = sx_1, z_1 = tx_1\). The parameterizations are then given by

\[
x_1(s, t) = \frac{X_1(s, t)}{W_1(s, t)} = -\sum_{i+j+k=n-1}^{1} \frac{1}{i! j! k!} \frac{\partial^{i+j+k} f}{\partial x^i \partial y^j \partial z^k}(a, b, c) s^i t^j k^k
\]

Then \((x(s, t), y(s, t), z(s, t)) = \left(\frac{X(s, t)}{W(s, t)}, \frac{Y(s, t)}{W(s, t)}, \frac{Z(s, t)}{W(s, t)}\right)\) where \(X, Y, Z, W\) are the polynomials given below.

\[
X(s, t) = X_1(s, t) + aW_1(s, t)
\]
\[
Y(s, t) = sX_1(s, t) + bW_1(s, t)
\]
\[
Z(s, t) = tX_1(s, t) + cW_1(s, t)
\]
\[
W(s, t) = W_1(s, t)
\]

These rational functions \((x(s, t), y(s, t), z(s, t))\) are of degree \(n\) and rationally parameterize the monoidal surface \(f(x, y, z) = 0\). While the expressions are not in closed form, they could be easily so written for any given \(n\).

2.6.2 Error Analysis

Suppose approximations \((\tilde{a}, \tilde{b}, \tilde{c})\) are used for the singular point \((a, b, c)\); then the output parameterization will be approximate. We state the following without proof:
LEMMA 2.7 The approximate output surface satisfies the implicit equation

$$\hat{f}(x, y, z) = f(x, y, z) - \sum_{i+j+k=0}^{n-2} \frac{1}{i!j!k!} \frac{\partial^{i+j+k} f}{\partial x^i \partial y^j \partial z^k} (a, b, c)(x - \hat{a})^i(y - \hat{b})^j(z - \hat{c})^k = 0$$

□

2.7 Summary

In this chapter, the problem of rational parameterization of low-degree implicit curves and surfaces was examined. We investigated these algorithms, that assume exact arithmetic, when they operate over a practical finite precision domain. Then we showed how to structure the algorithms so that they can be efficiently implemented in finite precision arithmetic. Using a step-by-step analysis, the algorithms were systematically reformulated. The restructured algorithms are highly efficient in that the only calculation they require are polynomial root-solving, followed by integral operations (i.e. +, x, -) in the finite numerical domain. No polynomial or rational function manipulation is required, as in the original algorithms.

Using backward error analysis, we derived precise algebraic characterizations of the error in the approximate output. These error formulas isolate the error due to the approximation of algebraic numbers by rationals.

One issue we have not discussed so far is how to compute monoidal points. The direct way by solving a system of non-linear equations is not an easy problem, although some progress is being made [40, 77]. In the next chapter, we show that monoidal points of curves have rational coordinates and give a way of computing them exactly.
3. APPLICATIONS OF ERROR ANALYSIS

3.1 Introduction

In the previous chapter, we formulated parameterization algorithms to work in finite precision domains, and also derived in each case simple algebraic characterizations of the error due to the use of rational approximations to algebraic numbers. A major difference between our parameterization algorithms and the previous approaches is the amenability to error analysis.

We now consider how the algebraic error formulae might be put to use. First, the error analysis is applied to describe the perturbed output geometrically. Next, it is used to derive geometric error bounds for conic and quadric parameterizations.

Finally, a natural question that arises in the context of error analysis is, "when can error be avoided?" That is, are there classes of curves and surfaces which can, in theory, be exactly parameterized using only integral or rational operations on the coefficients? While avoiding a long detour into the theory of Diophantine equations, we answer the latter question in the affirmative.

3.2 Geometric Interpretation of Error

The parameterization algorithms must produce an output curve (or surface) identical to the input. When approximations are used, this will not happen. Ordinarily it would be hard to geometrically compare the output (parametric) curve with the input (implicit) curve.

However, we have shown how to calculate the implicit equation of the output parametric. In this section, this information is used to derive some simple and elegant
geometric properties of the approximate output, that will provide some insight into the parameterization process.

Recall that in all the parameterization algorithms above except the one for singular cubics, the only computation was finding a point on the input curve or surface. The basic result of this section is that the approximate point computed lies exactly on the approximate output curve or surface.

For monoids, the point satisfied additional conditions. We shall show that the approximate output curves are also monoids, whose monoidal point is exactly the approximate point.

Besides providing geometric insight into parameterization, these results clarify one important issue. In recent literature addressing the numerical behavior of implicitly defined algebraic curves and surfaces [39], the authors show that slight perturbations in the coefficients of an implicit curve can destroy any singularities it may have. Noting that rational parameterization algorithms (such as the monoid parameterizations) explicitly compute such singularities, they warn that this procedure is "fraught with danger in the context of imprecise arithmetic."

The results of this section, particularly regarding monoids, will show that our reformulated algorithms are numerically stable in this aspect.

3.2.1 Conics

FACT 3.1 Let \( F(X, Y, W) = 0 \) be the homogeneous implicit equation of a conic. Let \((b, c, d)\) be a point on the curve, which is approximated by a point \((\tilde{b}, \tilde{c}, d)\) in the conic parameterization algorithm, giving as output a parametric curve whose homogeneous implicit equation is \( \tilde{F}(X, Y, W) = 0 \). Then

\[
\tilde{F}(\tilde{b}, \tilde{c}, d) = 0
\]

Thus, \( F(b, c, d) = 0 \) and \( \tilde{F}(\tilde{b}, \tilde{c}, d) = 0 \).

PROOF. We use the lemmas proved in the algebraic error analysis of the previous chapter. Although the fact above is stated to allow projective transformations, we
only prove it for the affine case with curves given by \( f(x, y) = 0, \tilde{f}(x, y) = 0 \). The projective case is similar. Suppose the first transformation was used in the conic parameterization algorithm, and \( d = 1 \) (i.e. it was affine). Then by lemma (2.1), the output parametric curve is given by

\[
\tilde{f}(x, y) = a_{20}y^2 + a_{11}xy + \left(a_{02} - \frac{f(b, c)}{b^2}\right)x^2 + a_{10}y + a_{01}x + a_{00} = 0
\]

Then

\[
\begin{align*}
\tilde{f}(x, y) &= f(x, y) - \frac{f(b, c)}{b^2}x^2 \\
\tilde{f}(b, c) &= f(b, c) - \frac{f(b, c)}{b^2}b^2 \\
&= 0
\end{align*}
\]

The other cases of the algorithm can be enumerated, affine and projective. The fact follows from applying the appropriate lemmas of the last chapter. 

3.2.2 Quadrics

We state the above result for quadrics, without proof.

**FACT 3.2** Let \( F(X, Y, Z, W) = 0 \) be the homogeneous implicit equation of a quadric. Let \((a, b, c, d)\) be a point on the surface, which is approximated by a point \((\tilde{a}, \tilde{b}, \tilde{c}, d)\) in the quadric parameterization algorithm, giving as output a parametric surface whose homogeneous implicit equation is \( \tilde{F}(X, Y, Z, W) = 0 \). Then

\[
\tilde{F}(\tilde{a}, \tilde{b}, \tilde{c}, d) = 0
\]

Thus, \( F(a, b, c, d) = 0 \) and \( \tilde{F}(\tilde{a}, \tilde{b}, \tilde{c}, d) = 0 \).

3.2.3 Monoid Curves

For monoid curves, the point chosen by the parameterization algorithm is a special one, satisfying more equations than just the curve's. We shall show that the
Figure 3.1  Numerical stability in monoid parameterizations
output parametric curve also has such a special point, and then discuss the geometric situation in some detail, using the special case of singular cubics as an illustration.

**FACT 3.3** Let \( f(x, y) = 0 \) be the implicit equation of a monoid of degree \( n \), and \((b, c)\) the coordinates of its \((n - 1)\)-fold multiple point. Let this point be approximated by \((\hat{b}, \hat{c})\) in the monoid parameterization algorithm, giving as output a parametric curve whose implicit equation is \( \hat{f}(x, y) = 0 \). Then \((\hat{b}, \hat{c})\) is an \((n - 1)\)-fold point of the curve \( f(x, y) = 0 \).

**PROOF.** To prove that a point is a \( k \)-fold point of a curve, we must show that most lines through the point intersect the curve \( k \) times at that point. By lemma (2.6), the output curve equation is \( \hat{f}(x, y) = 0 \) where

\[
\hat{f}(x, y) = f(x, y) - \sum_{i+j=0}^{n-2} \frac{1}{i!j!} \frac{\partial^{i+j} f}{\partial x^i \partial y^j} (\hat{b}, \hat{c})(x - \hat{b})^i(y - \hat{c})^j
\]

Expand \( f(x, y) \) in a Taylor around \((x, y) = (\hat{b}, \hat{c})\): then

\[
f(x, y) = \sum_{i+j=0}^{n} \frac{1}{i!j!} \frac{\partial^{i+j} f}{\partial x^i \partial y^j} (\hat{b}, \hat{c})(x - \hat{b})^i(y - \hat{c})^j
\]

\[
\Rightarrow \hat{f}(x, y) = \sum_{i+j=n-1}^{n} \frac{1}{i!j!} \frac{\partial^{i+j} f}{\partial x^i \partial y^j} (\hat{b}, \hat{c})(x - \hat{b})^i(y - \hat{c})^j
\]

Now lines through \((\hat{b}, \hat{c})\) are given in parametric form by \( x = t + \hat{b}, y = mt + \hat{c} \), for all \( m \). Substituting in \( \hat{f}(x, y) = 0 \) yields a polynomial in \( t \) of degree \( n \); the roots of this polynomial correspond to intersections of a line with the curve.

\[
\hat{f}(t + \hat{b}, mt + \hat{c}) = \sum_{i+j=n-1}^{n} \frac{1}{i!j!} \frac{\partial^{i+j} f}{\partial x^i \partial y^j} (\hat{b}, \hat{c})((1 + \hat{b}) - \hat{b})^i((mt + \hat{c}) - \hat{c})^j
\]

\[
= \sum_{i+j=n-1}^{n} \frac{1}{i!j!} \frac{\partial^{i+j} f}{\partial x^i \partial y^j} (\hat{b}, \hat{c})t^i(ji)^j
\]

\[
= t^{n-1} \left( \sum_{i+j=n-1}^{n} \frac{1}{i!j!} \frac{\partial^{i+j} f}{\partial x^i \partial y^j} (\hat{b}, \hat{c})m^j + \sum_{i+j=n} \frac{1}{i!j!} \frac{\partial^{i+j} f}{\partial x^i \partial y^j} (\hat{b}, \hat{c})n^j t \right)
\]

\[
= 0
\]

This polynomial has a root of multiplicity \( n - 1 \) at \( t = 0 \), due to the factor \( t^{n-1} \). The value \( t = 0 \) corresponds to \((\hat{b}, \hat{c})\), so this is a point of multiplicity \((n - 1)\) on the curve.

\[\square\]
Thus the monoid parameterization algorithm can be reformulated to output a monoid, even with finite precision computations. This is important, for the following reason. A slight perturbation in the coefficients of the input implicit curve can destroy the singularity. Such perturbations can occur in the rational to floating-point conversion, if for example a numerical method is used to locate the singularity, by solving a system of polynomial equations. Unless it is very lucky (e.g., the coefficients are exactly representable in floating-point binary), the numerical algorithm will actually be operating on a curve that has no monoid point. While it will be unable to locate an exact vanishing point of the equations, it can be instructed to compute a point at which all the polynomials of the system will evaluate to a "small" value. Such a point is likely to be displaced from the true singularity by a small distance. Our analysis then shows that the output parameterization will have a monoidal point at a predictable location. Figure 3.1 shows two cubic monoids. The curve drawn using a solid line was graphed using the input implicit equation directly. An approximation to the monoidal point was used to compute the parametric equations, which were then used to graph it (using a dashed line). As predicted, the parametric curve is a monoid with the singular point slightly displaced from its true location.

Unfortunately, one cannot yet conclude that the output mirrors the input in every significant way, modulo small geometric perturbations; let us discuss why.

Algebraic curves can have a varied structure at singular points. One way to classify this structure is to consider the tangents at the singular point (there will be more than one). The tangents may be all distinct, some distinct and some coincident, or all imaginary in which case the singular point is an isolated point.

In the first and third cases, the singularity is termed ordinary. The second case depends on the identical vanishing of a certain polynomial. If the input curve happened to have such a configuration at the monoid point, while the output curve would still have a singularity of the same multiplicity, the tangent configuration would likely have been perturbed into one of the others. For instance, suppose a cusp singularity
is perturbed into a node singularity. This may not matter since the "loop" is likely to be small, but some applications may need to handle the new case differently.

Let \((b, c)\) be an \(r\)-fold point of the curve, that is, all derivatives of the curve up to the \((r - 2)\)th order vanish at \((b, c)\), but not all those of the \((r - 1)\)th order. The tangent configurations are related to the roots of the polynomial

\[
\phi(\lambda) = \sum_{i=0}^{r} \binom{r}{i} \frac{\partial^i f}{\partial x^i} \partial^r_i y(b, c) \lambda^i
\]

(from [111]; we only consider affine roots for simplicity). Each root of this polynomial gives rise to a tangent curve; hence, a multiple root gives rise to a multiple tangent. The singularity is ordinary if all the roots are distinct (real or complex).

The condition for a polynomial to have multiple roots is for its discriminant to vanish identically. The discriminant is the resultant of the polynomial and its derivative; the condition is then expressed as

\[
\text{resultant}(\phi(\lambda), \phi'(\lambda)) = 0
\]

This is a polynomial condition in the coefficients of the original curve, and the coordinates \((b, c)\) of the original point.

This equality is unlikely to be preserved if \((b, c)\) is replaced by an approximation \((\tilde{b}, \tilde{c})\).

Consider a singular cubic plane curve. The discriminant is \(D = f_{x}^2_y - f_{xx}f_{yy}\). If \(D > 0\), the cubic will have an ordinary singularity with two real, distinct tangents; if \(D < 0\), it will be an isolated point with complex tangents. Hence, unless the perturbation in \((\tilde{b}, \tilde{c})\) is large enough to change the sign of \(D\), the output parameterization will have the same type of singularity as the input curve. The case \(D = 0\) is rare in the sense that it is highly unlikely to happen by accident, with random input coefficients. If it does happen, however, the output of the parameterization is likely to be perturbed into one of the other two cases.
3.2.4 Monoidal Surfaces

We state without proof the following result for monoid surfaces.

FACT 3.4 Let \( f(x, y, z) = 0 \) be the implicit equation of a monoid surface of degree \( n \), and \((a, b, c)\) the coordinates of its \((n - 1)\)-fold point. Let this point be approximated by \((\tilde{a}, \tilde{b}, \tilde{c})\) in the monoid surface parameterization algorithm, giving as output a parametric surface whose implicit equation is \( f(x, y, z) = 0 \). Then \((\tilde{a}, \tilde{b}, \tilde{c})\) is an \((n - 1)\)-fold point of the surface \( \tilde{f}(x, y, z) = 0 \).

\[ \square \]

3.2.5 Geometric Interpretation of Error: Summary

The algorithms shown above share this simple property: they compute a point on the curve or surface, and the coordinates of this point appear in the output. If one must compute an approximate point, then the above results show that our reformulated algorithms possess this (desirable) feature: the geometric relationship between the exact point and the input curve or surface is mimicked by that between the approximate point and the output curve or surface.

3.3 Geometric Error Bounds: Conics

The algebraic analysis tells us that the output parametric curve corresponds to the input implicit curve, perturbed in one coefficient. This information can be used to derive geometric error bounds, i.e., to bound the maximum distance between the input implicit curve and output parametric curve. Such geometric bounds can be applied towards the accurate evaluation and display of curves and surfaces.

In [39], general bounds are given for geometric perturbations at a point on a curve due to small random perturbations in the coefficients of its equation. These bounds are local: for each point of the curve, a condition number is given, which measures how much that point is displaced, in a direction orthogonal to the curve at that point.
In our model, however, we have shown that the perturbations generated by the parameterization process are not random but have a definite structure. This structure can be exploited to derive global displacement bounds for the special cases of conics and quadrics.

We investigate the geometric effects of perturbing a single coefficient in the equation of a conic curve. Such a perturbation yields an entire family of conics. In particular, the effect of perturbing the constant coefficient of a conic is shown to be less detrimental than perturbing the coefficients of the quadratic terms. It will be shown that perturbing the constant coefficient gives rise to a conic very similar to the original conic. We then bound the maximum orthogonal distance between the original and perturbed conic.

3.3.1 Properties of Conics

Much is known about conics; we cite some relevant facts from [105] and [91]. Consider the affine quadratic equation of a conic curve $C$, in the form

$$f(x, y) = ax^2 + by^2 + 2hxy + 2gx + 2fy + c = 0$$

The discriminant of $C$ is

$$\Delta = \begin{vmatrix} a & h & g \\ h & b & f \\ g & f & c \end{vmatrix} = abc + 2fg \cos \alpha - a f^2 - bg^2 - ch^2$$

The following facts about conics are known:

1. $C$ degenerates to a pair of lines when $\Delta = 0$
2. $C$ is a parabola when $ab - h^2 = 0$, an ellipse when $ab - h^2 < 0$, and a hyperbola when $ab - h^2 > 0$.
3. When $C$ is not a parabola, its center is given by $(\frac{hf-bg}{ab-h^2}, \frac{bh-ag}{ab-h^2})$.
4. The axes of the conic are given by the equation $h(x^2 - y^2) - (a-b)xy = 0$. 
6. The conic can be translated to have its center at the origin, and in this coordinate system its equation is

\[ f(x, y) = ax^2 + 2hxy + by^2 + \frac{\Delta}{ab - h^2} = 0 \]

7. A conic that has been translated to the center can further have its axes be rotated to become the principal axes, and in this coordinate system its equation is

\[ f(x, y) = (a + b + R)x^2 + (a + b - R)y^2 + \frac{2\Delta}{ab - h^2} = 0 \quad (3.1) \]

where \( R^2 = (a - b)^2 + 4h^2 \).

### 3.3.2 Bounding the Orthogonal Distance

It is now clear from the above that perturbing the constant term \( c \) in the equation of a conic will leave \( a, b, h \) unchanged and so produce a perturbed conic of the same type that is concentric and coaxial with the original (see also [22]). Figure 3.2 a family of conics perturbed in the constant coefficient. Perturbing the coefficients of \( x^2 \) or \( y^2 \), on the other hand, can change all these quantities: Figure 3.3 shows a family of conics perturbed only in the coefficient of \( x^2 \); they vary in type, center, and axis. We will therefore only consider the third transformation of the conic algorithm, which is always affine. However, it is likely that bounds can also be derived for the other two cases, if necessary.

Even when only the constant coefficient is perturbed, the conic could still degenerate into a pair of lines, and a large enough perturbation could turn a hyperbola into one that is concentric and coaxial to the original, but with transverse and conjugate axes reversed. Hence, an upper bound must be imposed on the perturbation. Since the constant coefficient \( c \) appears linearly in the discriminant \( \Delta \), so will the perturbed coefficient \( c + \delta \), and hence one can immediately bound \( |\delta| \) to avoid this case. If this bound is very small the conic will already be close to degenerate. We will henceforth only consider perturbations smaller than this bound, to simplify the analysis.

For perturbations smaller than this bound, then, we wish to describe the error geometrically. Define the (orthogonal) distance from a point \( p \) on one conic to the
other conic as the shortest distance along the normal vector at \( p \) to the other conic. Then the maximum orthogonal distance from a point on one conic to the other will occur at one of the extreme points of the conic along its major axis, if ellipse, or transverse axis, if hyperbola. Figure 3.4 shows the situation for concentric, coaxial ellipses with their axes aligned along the major axes.

For simplicity, we only consider central conics from now on, i.e., with \( ab - h^2 \neq 0 \) (slightly different results can be derived for the case \( ab - h^2 = 0 \)).

Now suppose one is given two conics \( C, \tilde{C} \), where the second conic is derived by perturbing the constant coefficient in the equation of the first. Their equations will take the form

\[
\begin{align*}
ax^2 + by^2 + 2hxy + 2gx + 2fy + c_1 &= 0 \\
ax^2 + by^2 + 2hxy + 2gx + 2fy + c_2 &= 0
\end{align*}
\]

(3.2)

They will be concentric and coaxial, and we can consider their equations in a coordinate system where their center is at the origin and their axes are aligned with the primary axes. In this coordinate system their equations will take the form

\[
\begin{align*}
f(x,y) &= Ax^2 + By^2 + C_1 = 0 \\
\tilde{f}(x,y) &= Ax^2 + By^2 + C_2 = 0
\end{align*}
\]

where \( A, B, C_1, C_2 \) are as in (3.1).

Referring to Figure 3.4, let \( d_x, \tilde{d}_x \) be the distances along the \( x \)-axis from the origin to \( C, \tilde{C} \) respectively. Likewise, let \( d_y, \tilde{d}_y \) be the distances along the \( y \)-axis. (In the case of a hyperbola, only one of these distances is finite). Then

\[
\begin{align*}
\tilde{d}_x &= d_x + p_x \\
\tilde{d}_y &= d_y + p_y
\end{align*}
\]

One of \( p_x \) and \( p_y \) will be the maximum orthogonal distance between the two curves; \( p_x \) and \( p_y \) can be solved for directly.

**LEMMA 3.1** Let \( \delta = c_1 - c_2 \) be the coefficient perturbation between the two conics (3.2). The corresponding geometric perturbations \( p_x \) and \( p_y \) along their axes are given
by
\[ p_x = -d_x + \sqrt{d_z^2 + \left( \frac{2}{a + b + R} \right) \delta} \]  
(3.3)
and
\[ p_y = -d_y + \sqrt{d_y^2 + \left( \frac{2}{a + b - R} \right) \delta} \]  
(3.4)

The maximum of \( p_x, p_y \) will give a geometric error corresponding to some algebraic error \( \delta \).

**Proof.** First put \( y = 0 \) in the curve equations. Then
\[ d_z^2 = -\frac{C_1}{A} \]
\[ d_x^2 = -\frac{C_2}{A} \]

Hence
\[ d_z^2 - d_x^2 = (d_x + p_x)^2 - d_x^2 = \frac{C_1 - C_2}{A} \]

and
\[ p_z^2 + 2d_zp_x = \frac{C_1 - C_2}{A} \]

Now find \( p_x \) by the quadratic formula: since \( p_x = 0 \) when \( C_1 - C_2 = 0 \), we choose the positive sign for the radical and find that
\[ p_x = -d_x + \sqrt{d_z^2 + \frac{(C_1 - C_2)}{A}} \]  
(3.5)

Revert to the original coordinate system of the conics. Using the coordinate transformations of the previous section,
\[ C_1 - C_2 = \frac{2\Delta_1}{ab - h^2} - \frac{2\Delta_2}{ab - h^2} \]
\[ = \frac{2}{ab - h^2}(\Delta_1 - \Delta_2) \]

By the definition of the discriminant,
\[ \Delta_1 - \Delta_2 = \begin{vmatrix} a & h & g \\ h & b & f \\ g & f & c_1 \end{vmatrix} - \begin{vmatrix} a & h & g \\ h & b & f \\ g & f & c_2 \end{vmatrix} = \begin{vmatrix} a & h & (c_1 - c_2) = (ab - h^2)(c_1 - c_2) \end{vmatrix} \]
So
\[ C_1 - C_2 = 2(c_1 - c_2) \]  

(3.6)

Substitute the definitions of \( A, B, \) and \( R \) given in the previous section, and relation (3.6) into (3.5). This gives the first half of the lemma. The derivation for \( p_y \) is similar.

At this point, we have expressed the geometric perturbations along the axes of the conics terms of their coefficients, and the value \( \delta \) which is the perturbation in their coefficients. These give global bounds on the orthogonal displacement between a conic and its perturbed counterpart.

Note that a scaling of the input conic equation by a constant will appear in the quantities \( a, b, \) and \( R. \) If the value \( \delta \) is defined as in the previous chapter as the value of the conic equation at some point, then it will include this scale factor. So the expressions above are independent of such scalings.

In a practical setting, the value \( \delta \) depends on how well certain approximations to algebraic numbers satisfy a polynomial equation, and these approximations are computed iteratively. It would be pleasant to bound the value \( \delta \) in terms of a specified geometric error; this would lead to termination criteria for these iterative processes, and also be useful towards bounding the actual precision (bit-length of the rational approximations).

Thus we bound \( \delta, \) given a geometric error bound. The results of lemma (3.1) are used.

**Lemma 3.2** Given a number \( \epsilon > 0 \) that also satisfies \( \epsilon < \min(d_x, d_y), \) and two conics (3.2) that differ in their constant coefficients by a quantity \( \delta \) as defined above, if the geometric perturbations \( p_x, p_y \) are to satisfy
\[ \max(|p_x|, |p_y|) < \epsilon \]

then it suffices to choose \( \delta \) such that
\[ |\delta| < \epsilon \cdot \frac{\min(d_x \cdot |a + b + R|, d_y \cdot |a + b - R|)}{2} \]
If one of \( d_x \) or \( d_y \) is complex, then the corresponding perturbation value is not considered, and the \( \min(\cdot \cdot) \) is unnecessary.

**PROOF.** To have \( |p_x| < \epsilon \), it is necessary that \( p_x < \epsilon \) and \( p_x > -\epsilon \). Considering each case separately,

1. \( p_x < \epsilon \) means

\[
-d_x + \sqrt{d_x^2 + \left( \frac{2}{a+b+R} \right) \delta} < \epsilon
\]

\[
\sqrt{d_x^2 + \left( \frac{2}{a+b+R} \right) \delta} < d_x + \epsilon
\]

\[
d_x^2 + \left( \frac{2}{a+b+R} \right) \delta < (d_x + \epsilon)^2
\]

\[
\left( \frac{2}{a+b+R} \right) \delta < \epsilon (2d_x + \epsilon)
\]

Now there are two cases, depending on the sign of \( a + b + R \), and both can be satisfied by choosing

\[
|\delta| < \epsilon (2d_x + \epsilon) \left| \frac{a + b + R}{2} \right| \tag{3.7}
\]

2. Similarly, \( p_x > -\epsilon \) means

\[
-d_x + \sqrt{d_x^2 + \left( \frac{2}{a+b+R} \right) \delta} > -\epsilon
\]

\[
\sqrt{d_x^2 + \left( \frac{2}{a+b+R} \right) \delta} > d_x - \epsilon
\]

\[
d_x^2 + \left( \frac{2}{a+b+R} \right) \delta > (d_x - \epsilon)^2
\]

\[
\left( \frac{2}{a+b+R} \right) \delta < \epsilon (2d_x - \epsilon)
\]

after multiplying by \(-1\) in the last step. Again there are two cases, depending on the sign of \( a + b + R \), and both can be satisfied by choosing

\[
|\delta| < \epsilon (2d_x - \epsilon) \left| \frac{a + b + R}{2} \right| \tag{3.8}
\]

Finally, recalling that \( \epsilon < d_x \), inequalities (3.7) and (3.8) can be simultaneously satisfied by choosing

\[
|\delta| < \epsilon \cdot d_x \cdot \left| \frac{a + b + R}{2} \right|
\]
This is the only simplification made in the calculation, and at most a factor of two of accuracy (one bit) is lost. The restriction \( \epsilon < d_x \) is reasonable: it states that the geometric perturbation should be smaller than the "size" of the conic, where by "size" is meant here the length of an axis.

The error \( p_y \) is bounded in an identical way. It is therefore sufficient to compute \( \delta \) such that

\[
|\delta| < \epsilon \cdot \frac{\min(d_x \cdot |a + b + R|, d_y \cdot |a + b - R|)}{2}
\]

This proves the lemma. \( \Box \)

The quantities \( d_x \) and \( d_y \) are independent of any scaling of the coefficients of the original conic by a constant, but the scale factor will be linearly present in the quantities \( a + b + R \) and \( a + b - R \), as they should be.

Lower bounds are similarly calculated.

Thus, given a geometric bound \( \epsilon \) on the maximum orthogonal distance \( |p| \) between the original and perturbed conics, an upper bound on the parameterization error \( |\delta| \) can be derived in terms of \( \epsilon \) and the coefficients of the conic. So the geometric constraint \( \epsilon \) can be used to decide how small \( \delta \) must be chosen, that is, how closely to approximate the required algebraic numbers.

3.4 Quadric Surface Parameterization

In quadric parameterization, one can cancel any of the \( X^2, Y^2, Z^2, W^2 \) terms, at the cost of a perturbation in the corresponding coefficient. The error analysis for quadrics can be applied to deriving geometric error bounds. As elsewhere, the procedure for conics pleasantly generalizes to quadrics.

The geometry of quadric surfaces is also well-known. The discriminant of a quadric is a \( 4 \times 4 \) determinant in its coefficients, and its sign, along with certain other quantities, distinguishes among the various types of quadrics (its sign is invariant under a constant scaling of the quadric equation, since each term in the discriminant is a product of four coefficients).
3.4.1 Properties of Quadrics

Let the affine quadric equation be of a quadric: $Q$ be

$$f(x, y, z) = ax^2 + by^2 + cz^2 + 2fyz + 2gzx + 2hxy + 2px + 2qy + 2rz + d = 0$$

Its discriminant is

$$\Delta = \begin{vmatrix} a & h & g & p \\ h & b & f & q \\ g & f & c & r \\ p & q & r & d \end{vmatrix}$$

The following facts are known (gathered from [106], chapter 8, after slight algebraic manipulation):

1. $Q$ degenerates into planes when $\Delta = 0$
2. A central quadric is one whose center is not infinitely distant; the condition for a quadric to be central is $D \neq 0$, where

$$D = \begin{vmatrix} a & h & g \\ h & b & f \\ g & f & c \end{vmatrix}$$

3. A central quadric can be put in a standard form in which its center is at the origin and its axes lie along the principal axes. In this form its equation is

$$f(x, y, z) = \lambda_1 x^2 + \lambda_2 y^2 + \lambda_3 z^2 + \frac{\Delta}{D} = 0 \quad (3.9)$$

where $\lambda_i$ are the roots of the polynomial

$$\phi(\lambda) = \lambda^3 - (a + b + c)\lambda^2 + (b f + a g + a h)\lambda + a h g$$

The three roots of $\phi(\lambda)$ are always real. Isolating intervals for the three roots are

$[-\infty, \alpha_1], [\alpha_1, \alpha_2], [\alpha_2, +\infty]$ where $\alpha_1 < \alpha_2$ are the roots of the quadratic polynomial

$$\psi(\alpha) = \alpha^2 - (b + c)\alpha + (f^2 - bc)$$
If \( \alpha_1 = \alpha_2 \) then the discriminant of this quadratic is zero, which implies \( b = c \) and \( f = 0 \). In this case, the roots \( \{ \lambda_i \} \) are

\[
\left\{ \frac{(a + b) \pm \sqrt{(a - b)^2 + 4(g^2 + h^2)}}{2} \right\}
\]

### 3.4.2 Bounding the Orthogonal Distance

If a perturbation \( \delta \) in the constant coefficient is chosen small enough to preserve the sign of the discriminant and hence avoid degeneracies (this bound is easily computed as in the conic case, since \( \delta \) once again appears linearly in the quadric discriminant), one can again consider families of concentric, coaxial quadrics of the same type, and derive geometric error bounds similar to the conic case. The process is similar to the conic case, so we only state the results.

Let two central quadrics that differ only in their constant coefficient be given by

\[
\begin{align*}
ax^2 + by^2 + cz^2 + 2fyz + 2gxz + 2hxy + 2px + 2qy + 2rz + d_1 &= 0 \\
ax^2 + by^2 + cz^2 + 2fyz + 2gxz + 2hxy + 2px + 2qy + 2rz + d_2 &= 0
\end{align*}
\]

(3.10)

Then as in the conic case, we can define \( d_x, d_y, d_z, d_{x}, d_{y}, d_{z} \), the lengths of the semi-axes of the quadrics, and \( p_x, p_y, p_z \), the displacements along the axes between the two quadrics.

**LEMMA 3.3** Let \( \delta = d_1 - d_2 \) be the coefficient difference between the quadrics, and the values \( \Delta, D, \lambda_i \) be as defined in the previous section. The corresponding geometric perturbations along the axes are given by

\[
\begin{align*}
p_x &= -d_x + \sqrt{d_x^2 + \frac{\Delta}{\lambda_1}} \\
p_y &= -d_y + \sqrt{d_y^2 + \frac{\Delta}{\lambda_2}} \\
p_z &= -d_z + \sqrt{d_z^2 + \frac{\Delta}{\lambda_3}}
\end{align*}
\]

The quantities \( d_x, d_y, d_z \) are given by

\[
\begin{align*}
d_x^2 &= -\frac{\Delta}{D\lambda_1}, d_y^2 = -\frac{\Delta}{D\lambda_2}, d_z^2 = -\frac{\Delta}{D\lambda_3}
\end{align*}
\]
The maximum of $p_x, p_y, p_z$ will then give the maximum displacement between the original and perturbed quadrics.

To determine $\delta$ in terms of a specified geometric precision, we state the following.

**Lemma 3.4** Given a number $\epsilon > 0$ that also satisfies $\epsilon < \min(d_x, d_y, d_z)$, and two quadrics (3.10) that differ in their constant coefficients by a quantity $\delta$ as defined above, if the geometric perturbations $p_x, p_y, p_z$ are to satisfy

$$\max(|p_x|, |p_y|, |p_z|) < \epsilon$$

then it suffices to choose $\delta$ such that

$$|\delta| < \epsilon \cdot \min(d_x \cdot |\lambda_1|, d_y \cdot |\lambda_2|, d_z \cdot |\lambda_3|)$$

where expressions for $d_x, d_y, d_z, \lambda_i$ have been given previously. □

If one of $d_x, d_y, d_z$ is complex, then the corresponding perturbation value is not considered, and the $\min(\cdot, \cdot, \cdot)$ is unnecessary.

The bounds correct for scaling, as discussed in the section on quadrics.

### 3.5 Diophantine Solutions

In all our work to this point, we have assumed that error is inevitable in rational parameterization, due to the introduction of irrational numbers. A point with rational coordinates may not always exist on a conic curve or quadric surface with rational coefficients. Even if it does, its computation is not trivial.

Interestingly, if a curve of degree $n$ with rational coefficients is a monoid, then the monoidal point must have rational coordinates. We first discuss this for cubics, where a stronger statement holds.

#### 3.5.1 Singular Cubics

It is known in Diophantine analysis is that a rational cubic curve with rational coefficients has a rational double point [82]. This was apparently not widely known.
in the geometric modeling community. We are grateful to Allan Adler for alerting us to this fact and sketching the following proof.

**LEMMA 3.5** Let \( f(x, y) = 0 \) be the equation of an irreducible, singular cubic curve with rational number coefficients. Then the coordinates \((b, c)\) of the double point of the cubic are rational numbers.

**PROOF.**

1. An irreducible cubic curve has at most one singular point; if it had more, it would have more than three intersections with some line and hence would not be cubic. Thus the rational cubic has exactly one singular point.

2. Let \( A \) be an automorphism of the field of complex numbers, that is, \( A(p + q) = A(p) + A(q) \), and \( A(pq) = A(p)A(q) \), where \( p, q \) are any complex numbers. If \( h(x) \) is a polynomial with rational coefficients and \( h(b) = 0 \) for some number \( b \), then also \( h(A(b)) = 0 \). This generalizes to a system of multivariate polynomial equations with rational coefficients. Since the coordinates \((b, c)\) of a singular point of a curve with rational coefficients satisfy such a system, \((A(b), A(c))\) will also satisfy the system, and will also be a singular point.

3. Since the cubic has exactly one singular point, \((A(b), A(c)) = (b, c)\) for any automorphism \( A \) of the complexes. That is, \( A \) fixes \( b \) and \( c \).

4. Any complex number fixed by all automorphisms must be rational; see, e.g., [55].

\(\square\)

Thus rational cubics have a rational double point. Since the parameterization for a monoid involves only integral operations on the rational point and the coefficients of the curve, by equations (2.23), all rational cubics have a parameterization involving only rational coefficients.
Therefore, one can theoretically parameterize an irreducible rational cubic curve without error, by computing the singular point exactly. One way is as follows.

An affine singular point is found as a solution to the system of equations \( f(x, y) = f_x(x, y) = f_y(x, y) = 0 \). The \( x \)-coordinate of this solution will be a multiple rational root of the degree six polynomial \( p(x) = \text{resultant}(f(x, y), f_x(x, y), y) \). The rational roots of a polynomial can be computed by applying the algorithm in [73]. This algorithm requires a polynomial without multiple roots, however, so we must first isolate a factor of \( p(x) \) that contains each multiple root, exactly once. This factor, say \( g_2(x) \), can be computed as follows:

\[
\begin{align*}
g_1(x) &= \gcd(p(x), p'(x)) \\
g_2(x) &= \frac{g_1(x)}{\gcd(g_1(x), g_1'(x))}
\end{align*}
\]

The polynomial \( g_1(x) \) will exclude all non-repeated roots of \( p(x) \), and to derive \( g_2(x) \) we divide out all repeated factors of \( g_1(x) \). Thus each multiple root of \( p(x) \) will have exactly one representative in \( g_2(x) \), which will have rational coefficients and can be at most cubic in degree. The rational root-finder can then be applied to find a rational root of \( g_2(x) \), which will also be a root of \( p(x) \). This yields the \( x \)-coordinate of the singular point.

The resultant is computed using a subresultant remainder sequence; this may then be used to compute the \( y \)-coordinate [4]. Each \((x, y)\) pair found this way can be tested whether it additionally satisfies \( f_x(x, y) = 0 \); only one pair will satisfy the test. The tests will be error-free, and there can be at most two such tests, since there are at most three distinct multiple roots of any degree six polynomial.

### 3.5.2 Monoid Plane Curves

In the previous section, it was shown that all rational cubics have rational coefficient parameterizations. For curves of degree \( n > 3 \), we can generalize the previous result only for monoids, which are only a subset of the rational curves of degree \( n \). The key to the cubic case was the fact that the singular point was the sole solution of
some system of equations, and hence the coordinates of the singular point are fixed by all automorphisms. Note that the result does not automatically follow for monoidal surfaces. For instance, a cubic surface can have as many as four singular points [106]. (One might consider this impossible since a line through two singularities would intersect the cubic four times - but this merely implies that such a line lies entirely on the cubic surface).

In general, a similar line of reasoning can be used to show that if there is only one solution to a system of polynomial equations with rational coefficients, that solution must be rational. We note that if the system is non-linear, the number of equations must exceed the number of variables, otherwise there will always be more than one solution (this proceeds from Bezout's theorem).

For monoids, the monoidal point is the sole solution of a system of polynomial equations (all the partials of up to the $(n - 2)$'nd order equated to zero). Hence, the monoidal point must be rational, and rational coefficient parameterizations exist for monoids of any degree. The monoidal point will be a rational solution of any two equation subsystem of the above system of equations; by computing it using the method just given and backsubstituting, its exact rational coordinates may be found.

3.6 Summary

In this chapter, we have further investigated the error caused by parameterization algorithms. The analysis of the previous chapter was applied to derive geometric properties of the erroneous output, and to derive geometric error bounds. We also investigated the feasibility of rational coefficient (exact) parameterizations.

To summarize, in the presence of approximations, our reformulations of the parameterization algorithms generally satisfy these properties:

1. The degrees of the output expressions are bounded.
Figure 3.2 Perturbing the constant term in conic equations

2. If an approximate point on the input is used, the output and the approximate point are in a geometric relationship similar to that of the input and the exact point.

Furthermore, for the conic and quadric cases it was shown that the parametric output has the same algebraic degree as the implicit input.

Based on these results we conclude that rational parameterization algorithms can in general be formulated to operate in a numerically stable manner.
Figure 3.3 Perturbing squared term in the conic equations
Figure 3.4 Orthogonal distance between concentric, coaxial conics
4. PROJECTIVE REPARAMETERIZATIONS

4.1 Introduction

In this chapter we explore some applications of projective transformations of the parameter domain (also called projective reparameterizations). Reparameterizations do not affect the curve or surface, but can be used to control which portion of a curve or surface corresponds to a fixed portion of the parameter domain. Affine reparameterizations only affect finite portions of the curve or surface; projective reparameterizations can be used to move points at infinity in the domain to finite distances, to achieve certain effects. One problem amenable to this approach is what we term the total mapping problem: how to display the entire parametric curve or surface, when the parametric domain is infinite and the curve or surface itself has finite area.

Some of the literature discussing curves and surfaces over projective domains includes [85] and more recently, [35] and [32]. Solutions to the total mapping problem, for curves and surfaces in rational Bezier form are given in the latter work, using a technique called "homogeneous sampling" of the parameter domain. We propose an alternate method using projective reparameterizations, by which the problem of mapping an entire surface is reduced to mapping a fixed finite portion of the parameter domain several times. This fixed portion can be sampled in the normal way, allowing existing display algorithms to be applied without modification. The framework developed here allows us to derive some related results regarding parametric curves and surfaces, and forms the basis for the robust display algorithms of Chapter 5.

This chapter is organized as follows. First, we investigate various uses of projective linear reparameterizations of rational parametric curves, including total mapping of curves, finding the "complementary segment" of a curve segment, and total mapping of surfaces. Then projective quadratic reparameterizations are developed, as an
alternate and simpler way of solving a problem recently addressed in the literature, namely that of computing normal parameterizations of curves and surfaces.

4.2 Projective Linear Reparameterizations

We now show some uses of projective linear transformations of the parameters. These are equivalent to affine fractional linear transformations, and using them we can achieve effects that cannot be achieved using ordinary linear transformations of the parameter.

Consider the total mapping problem for curves, that is, it is desired to display an entire rational curve. The first solution that comes to mind is to allow the parameter to vary over the entire real line. This approach is considered in [32] and rejected due to the phenomenon of “parameter compression.” We will discuss this phenomenon in detail, which is peculiar to rational curves as opposed to polynomial ones, and provide an analytic explanation of why it occurs. This problem can be overcome to a certain extent using special parameter sampling techniques, e.g. [67], [86], whose generalizations to surfaces are expensive.

The approach taken in [32] is to consider the parameter domain to be a projective space. This allows finite and infinite values to be treated equally. In this method, a rational Bezier curve in a single parameter $u$ varying over the unit interval $[0,1]$ is converted into homogeneous form. In homogeneous form, the curve is given in a pair of parameters $(u_1,u_2)$, which are then set to vary over a piecewise linear curve and are related by $u_2 = 1 - |u_1|, u_1 \in [-1,1]$.

In our approach for curves in the monomial basis, we reduce the problem of mapping a curve over the entire parameter domain to that of mapping two curves over the unit interval $[0,1]$. The union of the two curves will give the original. The interval $[0,1]$ can then be sampled in the usual way; no separate sampling method is required. Our approach facilitates the robust display methods of Chapter 5 and the derivation of normal parameterizations, discussed in the next section.
To derive these results, we first solve the problem using projective transformations, but return to the affine domain at the end. The problem is approached in two ways. First, given a curve over \([-\infty, +\infty]\), find two curves over \([0,1]\) that are together equivalent to the first. Second, given a curve and some interval \([a,b]\), find another curve such that the latter curve over \([0,1]\) is the "complementary segment" of the first, i.e. that part of the first curve corresponding to the portion \([-\infty, +\infty]\ \setminus [a,b]\) of the parameter domain. In [70], it is shown how to find the complementary segment of a conic in rational Bezier form, and in [32] another solution is given.

Generalizing the treatment for curves, we give one of many possible solutions for surfaces.

4.2.1 Total Mapping: Curves

Given a rational parametric curve in a parameter that varies over the entire real line, we produce two rational curves, derived by reparameterizing the original, that together map to the original when their parameters are individually allowed to vary in \([0,1]\). The reparameterization functions are given as linear rational functions, in which form their action can be easily understood by inspection. It is to be understood however that they are actually projective linear transformations, and an implementation of such a reparameterization would substitute linear polynomials into a homogeneous polynomial, avoiding rational function manipulation. Our results are stated as lemmas.

**LEMMA 4.1** Let an algebraic space curve \(C(s)\) be given by rational parameter functions \(C(s) = (x_1(s), \ldots, x_n(s))\) in a parameter \(s \in [-\infty, +\infty]\). If \(p_1(t), p_2(t)\) are two parameter transformation functions defined by

\[
\begin{align*}
  s &= p_1(t) &= \frac{t}{1 - t} \quad (4.1) \\
  s &= p_2(t) &= \frac{-t}{1 - t} \quad (4.2)
\end{align*}
\]

then the curves \(C(p_1(t)), C(p_2(t))\) over \(t \in [0,1]\) together will contain the same set of points as \(C(s)\) over \(s \in [-\infty, +\infty]\).
PROOF. To accommodate infinite parameter values, we consider the curve \( C(s) \) over a projective parameter domain. Points in this domain are pairs \((S, U) \neq (0, 0)\), with proportional tuples representing the same point. Points with \( U \neq 0 \) represent finite points and tuples \((S, 0)\) represent the point at infinity. The homogeneous curve \( C(S, U) \) is derived by homogenizing the individual parameter functions to yield \((x_1(S, U), \ldots, x_n(S, U))\). Two projective linear transformations are enough for our purposes. Such transformations between a projective domain \((S, U)\) and a projective domain \((T, V)\) take the form

\[
\begin{pmatrix}
a_{11} & a_{12} \\
a_{21} & a_{22}
\end{pmatrix}
\begin{pmatrix}
T \\
V
\end{pmatrix}
= k
\begin{pmatrix}
S \\
U
\end{pmatrix}
\]  

(4.3)

where \( k \) is a non-zero constant of proportionality.

First we arrange for the interval \([0, +\infty]\) of the \( s \) domain to correspond with the interval \([0, 1]\) of the \( t \) domain. In projective coordinates, this translates to two conditions: the point \((T, V) = (0, 1)\) maps to \((S, U) = (0, 1)\), and \((1, 0)\) maps to \((1, 1)\). These conditions are stated as follows:

\[
\begin{pmatrix}
a_{11} & a_{12} \\
a_{21} & a_{22}
\end{pmatrix}
\begin{pmatrix}
0 \\
1
\end{pmatrix}
= k_1
\begin{pmatrix}
0 \\
1
\end{pmatrix}
\]

\[
\begin{pmatrix}
a_{11} & a_{12} \\
a_{21} & a_{22}
\end{pmatrix}
\begin{pmatrix}
1 \\
0
\end{pmatrix}
= k_2
\begin{pmatrix}
1 \\
1
\end{pmatrix}
\]

We can now derive a system of equations for the unknowns \( a_{ij} \) and \( k_i \):

\[
\begin{align*}
a_{12} &= 0 \\
a_{22} &= k_1 \\
a_{11} + a_{22} &= k_2 \\
a_{21} + a_{22} &= 0
\end{align*}
\]

which yields the solution

\[
\begin{pmatrix}
a_{11} & a_{12} \\
a_{21} & a_{22}
\end{pmatrix}
= \begin{pmatrix}
k_2 & 0 \\
-k_1 & k_1
\end{pmatrix}
\]
Recall that we must choose $k_1 k_2 \neq 0$, and hence the transformation is $1 - 1$. For any such $k_1, k_2$, the transformation maps $t = 0$ to $s = 0$, and $t = 1$ to $s = \infty$. What happens to $0 < t < 1$? Such points are given in the projective domain by $(T, 1)$, $0 < T < 1$, after dividing by the (non-zero) $V$ coordinate. Then

$$
\begin{pmatrix}
  k_2 & 0 \\
  -k_1 & k_1 
\end{pmatrix}
\begin{pmatrix}
  T \\
  1 
\end{pmatrix} =
\begin{pmatrix}
  k_2 T \\
  k_1 (1 - T) 
\end{pmatrix}
$$

Thus $0 < t < 1$ maps to $s = \frac{k_2 t}{k_1 1 - t}$. Since $\frac{t}{1 - t} > 0$ when $0 < t < 1$, we must choose $k_1 k_2 > 0$ to map into $s > 0$, and $k_1 k_2 < 0$ to map into $s < 0$. Taking $k_1 = k_2 = 1$ and dehomogenizing, we get the first parameter transformation function $p_1(t)$, and by setting $k_1 = -1, k_2 = 1$ we get function $p_2(t)$. □

Figure 4.1 and 4.2 are examples of closed plane curves that are entirely mapped in two pieces, using only parameter values in $[0, 1]$.

4.2.2 The Complementary Segment

A parametric curve and some interval in the parameter domain together define a segment of the curve. The rest of the (infinite) parameter domain also defines a segment of the curve, called the complementary segment. Using projective reparameterizations, we can compute the complementary segment. The union of the original and complementary segments gives the entire curve.

We can assume without loss of generality that the original interval given is $[0, 1]$. If it is some $[a, b]$, an ordinary linear transformation $s = a + t (b - a)$ suffices to bring it to this form (as before, $s$ is the original parameter and $t$ is the new).

**Lemma 4.2** Let an algebraic space curve $C(s)$ be given by rational parameter functions $C(s) = (x_1(s), \ldots, x_n(s))$. The curve segment complementary to the segment defined by $s \in [0, 1]$ is given by $C(p(t)), p \in [0, 1]$, where

$$s = p(t) = \frac{t}{2t - 1} \quad (4.4)$$
PROOF. As before, we consider the curves over a projective parameter domain, and compute a projective linear reparameterization to achieve our ends. The parameter transformation function \( s = p(t) \) must map \( t \in [0, 1] \) to \( s \in [-\infty, +\infty] \setminus [0, 1] \). In this context, it is useful to understand that the topology of the projective line is that of a circle (see Figure 4.5, and also [85]), and the interval \([-\infty, +\infty] \setminus [0, 1] \) also has endpoints \([0, 1]\), going in the opposite direction in the circle. Thus the parameter transformation function must fix the values 0 and 1. Projectively, the transformation takes the form 4.3, giving us the two conditions

\[
\begin{pmatrix}
  a_{11} & a_{12} \\
  a_{21} & a_{22}
\end{pmatrix}
\begin{pmatrix}
  0 \\
  1
\end{pmatrix}
= 
\begin{pmatrix}
  k_1 \\
  1
\end{pmatrix}
\]

\[
\begin{pmatrix}
  a_{11} & a_{12} \\
  a_{21} & a_{22}
\end{pmatrix}
\begin{pmatrix}
  1 \\
  0
\end{pmatrix}
= 
\begin{pmatrix}
  k_2 \\
  1
\end{pmatrix}
\]

This yields a set of linear equations which can be easily solved to yield

\[
\begin{pmatrix}
  a_{11} & a_{12} \\
  a_{21} & a_{22}
\end{pmatrix}
= 
\begin{pmatrix}
  k_2 & 0 \\
  k_2 - k_1 & k_1
\end{pmatrix}
\]

Once again, we must choose \( k_1, k_2 \neq 0 \), so the transformation is 1-1.

Now set \( k_1 = -1, k_2 = 1 \). Then the transformation maps the point \( t = \frac{1}{2} \) to \( s = \infty \), because

\[
\begin{pmatrix}
  k_2 & 0 \\
  k_2 - k_1 & k_1
\end{pmatrix}
\begin{pmatrix}
  \frac{1}{2} \\
  1
\end{pmatrix}
= 
\begin{pmatrix}
  1 & 0 \\
  2 & -1
\end{pmatrix}
\begin{pmatrix}
  \frac{1}{2} \\
  1
\end{pmatrix}
= 
\begin{pmatrix}
  \frac{1}{2} \\
  0
\end{pmatrix}
\]

Now suppose \( 0 < t < 1, t \neq \frac{1}{2} \). Then let \((T, 1)\) be the corresponding projective point, after scaling:

\[
\begin{pmatrix}
  1 & 0 \\
  2 & -1
\end{pmatrix}
\begin{pmatrix}
  T \\
  1
\end{pmatrix}
= 
\begin{pmatrix}
  T \\
  2T - 1
\end{pmatrix}
= 
\begin{pmatrix}
  \frac{T}{2T - 1} \\
  1
\end{pmatrix}
\]

We can then show using elementary arguments that

\[
0 \leq T < \frac{1}{2} \quad \Rightarrow \quad -\infty < \frac{T}{2T - 1} < 0
\]
and
\[
\frac{1}{2} < T \leq 1 \Rightarrow 1 < \frac{T}{2T - 1} < +\infty
\]

Coupled with the fact that the transformation is $1 - I$, dehomogenizing yields the parameter transformation function $p(t)$ described in the lemma. □

Complementary segment pairs of a circle and a rational quartic curve are given in Figure 4.3, 4.4.

4.2.3 Parameter Compression

We now discuss the problem of parameter compression. Consider the usual rational parameterization of the unit circle, \( (\frac{s^2 - 1}{s^2 + 1}, \frac{2s}{s^2 + 1}) \). Allowing $s$ to take evenly spaced values over an interval $[-a, a], a >> 1$, we find that the points tend to cluster together as the parameter takes on larger values (see Figure 4.5). This picture appears in [32], and we have also noticed the phenomenon. Since the circle has the same curvature everywhere, this is definitely not a desirable property. However, we have been told in private communication ([87]) is that polynomial curves do not suffer from this problem. In fact, in polynomial curves, evenly spaced parameter values lead to a discretization of the curve in which the arc length of a segment seems to be inversely proportional to the curvature, a welcome feature since it produces smoother displays. Some examples of polynomial curves are given in Figures 4.6 and 4.7, where constant stepping of the parameter is used. Curves where the degree of the numerator exceeds that of the denominator also appear to have this desirable feature.

Thus we empirically observe that the relative degrees of the numerator and denominators of the rational parameter functions appear to have some effect on the speed at which the curve is traversed by a parameter taking steps of fixed length in the domain. The following theorem about the limits at infinity of rational functions helps clear up the matter. It can be found in introductory calculus books, e.g. [24].

THEOREM 4.1 For the rational function $\frac{f(t)}{g(t)}$, where

\[
f(t) = a_nt^n + a_{n-1}t^{n-1} + \cdots + a_0 \quad \text{and}
\]

\[
g(t) = b_ms^m + b_{m-1}s^{m-1} + \cdots + b_0 \quad \text{and}
\]

we have

\[
\lim_{t \to \infty} \frac{f(t)}{g(t)} = \begin{cases} 0 & \text{if } m > n, \\ \frac{a_n}{b_m} & \text{if } m = n, \\ \infty & \text{if } m < n. \end{cases}
\]
We have
\[
\lim_{t \to \pm\infty} \frac{f(t)}{g(t)} = \begin{cases} 
0, & \text{if } n < m \\
\frac{a_n}{b_m}, & \text{if } n = m \\
\pm\infty, & \text{if } n > m
\end{cases}
\]

Now let \(C : (x(t), y(t)) = t \left(\frac{f_1(t)}{g(t)}, \frac{f_2(t)}{g(t)}\right)\) be a rational parametric curve, where the degree of the polynomial \(g(t)\) equals or exceeds the degree of \(f_1(t), f_2(t)\). Then
\[
\frac{d}{dt} \left[ \frac{f_1(t)}{g(t)} \right] = \frac{g(t)f_1'(t) - f_1(t)g'(t)}{g(t)^2}
\]
and similarly for \(f_2(t)\). Thus in the derivatives of \(x(t), y(t)\), the numerators have strictly lower degree than the denominators. By the theorem,
\[
\lim_{t \to \pm\infty} \frac{dx}{dt} = \lim_{t \to \pm\infty} \frac{dy}{dt} = 0 \quad (4.5)
\]

The curve \(C\) was given in terms of an arbitrary parameter \(t\). Let \(s\) denote the arc-length parameter and consider the curve in \(s\). The relationship between the arc-length parameter and the original one is
\[
ds = \sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2} \, dt
\]

(The arc-length parameterization is never rational unless the curve is a line \([40]\), but this is not important to the analysis).

By equation (4.5), we now see that as the magnitude of \(t\) grows, a constant step \(\Delta t\) will give rise to an arc length step \(\Delta s\) of vanishingly smaller magnitude.

Thus, rational curves given by parameter functions whose degree of numerator does not exceed the degree of the denominator (this category includes all closed curves like the circle) will suffer from parameter compression. In general, since a rational function whose numerator is of degree \(n\) and denominator is of degree \(m\) behaves like a polynomial of degree \(\left\lfloor \frac{n}{m} \right\rfloor\) (as the function argument approaches infinity), we have
an explanation of the parameter compression phenomenon. It would be useful to show conclusively the converse, that polynomial curves are “good” parameterizations in the sense outlined above.

Thus in visualizing a curve, if we apply the lemma derived in the section on total mapping, we avoid the parameter compression problem since the parameters always vary over the unit interval.

4.2.4 Total Mapping: Surfaces

We now consider the problem of displaying an entire surface, using only finite parameter values. Solutions are given for rational Bezier surfaces in tensor-product and triangular patches in [32], using homogeneous sampling. We will give here a solution for surfaces in the monomial basis, for which ordinary sampling techniques can be used, in the same framework as that for curves, i.e. in terms of projective linear reparameterizations.

The problem is to map the entire domain using only a finite number of finite areas of the parameter domain. The topology of the domain (the real projective plane) is rather more complicated than in the case of curves, and there is more than one way to achieve this. For instance, Figure 4.8 shows a rational parametric sphere completely mapped by three rectangular patches, two of which are degenerate and together “fill the hole” left by the third.

The approach chosen here is a direct generalization of the method for curves. Each quadrant of the parameter plane is mapped to the triangle spanned by [0, 0], [1, 0] and [0, 1] (i.e. the two-dimensional unit simplex). The method generalizes to hypersurfaces of dimensions $n \geq 1$: the $n$-dimensional unit simplex in the parameter domain is made to correspond with each of the $2^n$ $n$-dimensional “quadrants” of infinite volume. This distributes the parameter domain in symmetric portions among the reparameterizations.
To map the unit triangle onto each quadrant in turn, we need to derive four projective reparameterizations. We give the parameter transformation functions as linear rational functions of the (two) new parameters.

**Lemma 4.3** Let an algebraic surface $S(s,t)$ be given by rational parameter functions $S(s,t) = (x(s,t), y(s,t), z(s,t))$ in two parameters $(s,t) \in [-\infty, +\infty] \times [-\infty, +\infty]$. A quadrant containing a point $(i,j) \in \{-1,1\}^2$ is given the label $(i,j)$, which yields a unique labeling of the quadrants. Then the parameter transformation $p_{ij}(u,v)$ mapping the unit triangle in $(u,v)$ space to quadrant $(i,j)$ in $(s,t)$ space is given by

$$
\begin{align*}
s &= i \cdot \frac{u}{1-u-v} \\
t &= j \cdot \frac{v}{1-u-v}
\end{align*}
$$

Hence the entire parametric surface can be mapped by considering in turn the surfaces $S(p_{ij}(u,v))$ for $(i,j) \in \{-1,1\}^2$.

**Proof.** Consider the first quadrant, i.e. $(i,j) = (1,1)$. The form of a projective linear transformation of the domain is similar to (4.3), using the variable $R$ to homogenize the $(s,t)$ domain and $W$ to homogenize $(u,v)$:

$$
\left( \begin{array}{ccc}
a_{11} & a_{12} & a_{13} \\
a_{21} & a_{22} & a_{23} \\
a_{31} & a_{32} & a_{33} \\
\end{array} \right) \left( \begin{array}{c} U \\
V \\
W \\
\end{array} \right) = k \left( \begin{array}{c} S \\
T \\
R \\
\end{array} \right)
$$

(4.7)

where $k$ is a non-zero constant of proportionality.

We will fix $(0,0)$, and make $(1,0),(0,1)$ in $(u,v)$ correspond to $(\infty,0),(0,\infty)$ respectively. This gives us three conditions, projectively:

$$
\left( \begin{array}{ccc}
a_{11} & a_{12} & a_{13} \\
a_{21} & a_{22} & a_{23} \\
a_{31} & a_{32} & a_{33} \\
\end{array} \right) \left( \begin{array}{c} 0 \\
0 \\
1 \\
\end{array} \right) = k_1 \left( \begin{array}{c} 0 \\
0 \\
1 \\
\end{array} \right)
$$

$$
\left( \begin{array}{ccc}
a_{11} & a_{12} & a_{13} \\
a_{21} & a_{22} & a_{23} \\
a_{31} & a_{32} & a_{33} \\
\end{array} \right) \left( \begin{array}{c} 1 \\
1 \\
0 \\
\end{array} \right) = k_2 \left( \begin{array}{c} 1 \\
1 \\
0 \\
\end{array} \right)
$$

$$
\left( \begin{array}{ccc}
a_{11} & a_{12} & a_{13} \\
a_{21} & a_{22} & a_{23} \\
a_{31} & a_{32} & a_{33} \\
\end{array} \right) \left( \begin{array}{c} 1 \\
0 \\
0 \\
\end{array} \right) = k_3 \left( \begin{array}{c} 1 \\
0 \\
0 \\
\end{array} \right)
$$
Solving the resulting system of linear equations yields the solution

\[
\begin{pmatrix}
a_{11} & a_{12} & a_{13} \\
a_{21} & a_{22} & a_{23} \\
a_{31} & a_{32} & a_{33}
\end{pmatrix}
\begin{pmatrix}
0 \\
1 \\
1
\end{pmatrix}
= k_3
\begin{pmatrix}
0 \\
1 \\
0
\end{pmatrix}
\]

We must take \( k_1k_2k_3 \neq 0 \) for the transformation to be well-defined, and we only need consider what happens to finite points (with \( W = 1 \)) of the new parameter domain. After scaling by any common factor, we find that

\[
\begin{pmatrix}
k_2 & 0 & 0 \\
0 & k_3 & 0 \\
-k_1 & -k_1 & k_1
\end{pmatrix}
\begin{pmatrix}
U \\
V \\
1
\end{pmatrix}
= \begin{pmatrix}
k_2U \\
k_3V \\
k_1(1 - U - V)
\end{pmatrix}
\]

Thus all points on the line \( u + v = 1 \) (the diagonal of the unit triangle) map to the line at infinity in the \((s, t)\) plane as they should. For all other points,

\[
\begin{pmatrix}
k_2 & 0 & 0 \\
0 & k_3 & 0 \\
-k_1 & -k_1 & k_1
\end{pmatrix}
\begin{pmatrix}
U \\
V \\
1
\end{pmatrix}
= \begin{pmatrix}
\frac{U}{k_1} \\
\frac{1 - U - V}{k_3} \\
\frac{V}{k_1}
\end{pmatrix}
\]

Set \( k_1 = k_2 = k_3 = 1 \); then all points in the unit simplex of \((u, v)\) map onto the first quadrant of \((s, t)\), those on the line \( u + v = 1 \) mapping onto the line at infinity connecting \((\infty, 0)\) and \((0, \infty)\).

It now is easy to see that all the quadrants are mapped by varying \((k_2, k_3) \in \{-1, 1\}^2\). This proves the lemma. \( \Box \)

Examples of surfaces mapped using equations (4.6) are given in Figures 4.9 and 4.10.
4.3 Piecewise Finite Representations of Parametric Varieties

We now generalize the results of the previous sections to parametric varieties of any dimension, by showing how to replace a parameterization over an infinite parameter domain with a finite number of parameterizations, each over a fixed, bounded parameter domain.

Let \( \mathcal{R} \) denote the set of reals and \( \mathcal{RP} \) the set of reals augmented with the point at infinity (i.e. the projective line). Suppose we are given a parameterization \( V(s) : \mathcal{RP}^n \rightarrow \mathcal{R}^m \) of a variety \( V \). We wish to compute maps \( Q_1, \ldots, Q_k \), with \( Q_i : \mathcal{R}^n \rightarrow \mathcal{R}^m \), such that \( \bigcup_{i=1}^k Q_i(\mathcal{R}^n) = V(\mathcal{RP}^n) \). That is, the new maps restricted to finite values together yield the same set of points that the given one does, even though the latter maps both finite and infinite domain values. To derive a finite representation we also find a bounded region \( D \subset \mathcal{R}^n \) to which the \( Q_i \) can be be restricted, i.e., \( \bigcup_{i=1}^k Q_i(D) = V(\mathcal{RP}^n) \).

Specifically, we compute \( 2^n \) parameterizations, each restricted to the unit simplex of the parameter domain \( \mathcal{R}^n \), that together generate all the points that \( V(s) \) does for \( s \in \mathcal{RP}^n \).

We use linear projective domain transformations (reparameterizations) to map, in turn, the unit simplex \( D \) of the new parameter domain space onto an entire octant of the original parameter domain space. The reparameterizations are specified in affine fractional form for convenience, but in practice they would be applied by homogenizing a parameterization and then substituting polynomials.

**Theorem 4.2** Consider a real parametric variety \( V \) in \( \mathcal{R}^m \) of dimension \( n \), \( n < m \), which is parameterized by the equations

\[
V(s) = \begin{pmatrix}
  x_1(s_1, \ldots, s_n) \\
  \vdots \\
  x_m(s_1, \ldots, s_n)
\end{pmatrix}, \quad s_i \in (-\infty, +\infty)
\]

Let the \( 2^n \) octant cells in the parameter domain \( \mathcal{R}^n \) be labeled by the tuples \( < \sigma_1, \ldots, \sigma_n > \) with \( \sigma_i \in \{-1, 1\} \). Then the \( 2^n \) projective reparameterizations \( V(t_{<\sigma_1, \ldots, \sigma_n>}) \)
given by

\[ s_i = \sigma_i \frac{t_i}{1 - t_1 - t_2 - \ldots - t_n}, \quad i = 1, \ldots, n \]  \hspace{1cm} (4.8)

together map all the points of the variety \( V(s) \), \( s_i \in (-\infty, +\infty) \), using only parameter values satisfying \( t_i \geq 0 \) and \( t_1 + t_2 + \ldots + t_n \leq 1 \).

**PROOF.** We must show that every point in the old domain \( \mathbb{R}P^n \) is the image of some point in the new domain \( \mathbb{R}^n \). In particular, we show that the hyperplane \( t_1 + \ldots + t_n = 1 \) bordering the unit simplex in \( \mathbb{R}^n \) maps onto the hyperplane at infinity in \( \mathbb{R}P^n \), and the rest of the points of the unit simplex are mapped onto a particular octant of the original domain space, depending on the signs of the \( \sigma_i \).

Let \( s = (c s_1, \ldots, c s_n, c s_{n+1}) \in \mathbb{R}P^n \), where \( c \in \mathbb{R} \) is a non-zero constant of proportionality and \( s_{n+1} = 0 \) is the equation of the hyperplane at infinity in \( \mathbb{R}P^n \). Let \( t = (t_1, \ldots, t_n) \in \mathbb{R}^n \). Since (4.8) is a map from \( \mathbb{R}^n \to \mathbb{R}P^n \), the following relationship holds between the \( s_i \) and \( t_j \), under one of the \( 2^n \) transformations \( < \sigma_1, \ldots, \sigma_n > \):

\[ c s_1 = \sigma_1 t_1 \]

\[ \vdots \]

\[ c s_n = \sigma_n t_n \]

\[ c s_{n+1} = 1 - (t_1 + \ldots + t_n) \]

Let \( \text{sign}(\alpha) = -1 \) or \( +1 \) according to whether \( \alpha < 0 \) or \( \alpha \geq 0 \), respectively.

First we show that every \( s \in \mathbb{R}P^n \) on the hyperplane at infinity is the image of some point \( t = (t_1, \ldots, t_n) \in \mathbb{R}^n \) under one of the transformations, and additionally that \( t_i \geq 0 \) and \( t_1 + \ldots + t_n = 1 \).

Since \( s \) is on the hyperplane at infinity, \( s_{n+1} = 0 \), and hence

\[ c s_i = \sigma_i t_i, \quad i = 1, \ldots, n \]

\[ 0 = 1 - (t_1 + \ldots + t_n) \]
Then a solution \((t_1, \ldots, t_n)\) is derived by setting

\[
\sigma_i = \text{sign}(s_i) \\
c = \frac{1}{\sum_{i=1}^{n} \sigma_i s_i} \\
t_i = \frac{\sigma_i s_i}{\sum_{i=1}^{n} \sigma_i s_i}
\]

Noting that \(\sigma_i s_i \geq 0\) and not all of the \(s_i, i = 1, \ldots, n\) can be zero, it follows that \(t_i \geq 0\) and \(\sum_{i=1}^{n} t_i = 1\).

Second, let \(s \in \mathbb{R}^n \subseteq \mathbb{R}P^n\), i.e. \(s_{n+1} \neq 0\). We show that \(s\) is the image of some \(t \in \mathbb{R}^n\) under one of the transformations, and additionally \(t\) lies in the unit simplex of \(\mathbb{R}^n\).

We can set \(s_{n+1} = 1\) w.l.o.g. and the following system of equations for the \(t_i\) is derived:

\[
s_i t_i = \sigma_i t_i \\
c = 1 - (t_1 + \ldots + t_n)
\]

We can solve this linear system by setting

\[
\sigma_i = \text{sign}(s_i) \\
c = \frac{1}{1 + \sum_{i=1}^{n} \sigma_i s_i} \\
t_i = \frac{\sigma_i s_i}{1 + \sum_{i=1}^{n} \sigma_i s_i}
\]

and since \(\sigma_i s_i \geq 0\) it follows that \(t_i \geq 0\) and \(t_1 + \ldots + t_n < 1\), hence this point \(t\) is in the unit simplex in \(\mathbb{R}^n\), but not on the hyperplane \(t_1 + \ldots + t_n = 1\).

We have thus proved that all of \(\mathbb{R}P^n\) is mapped by the transformations (4.8), restricting each to the unit simplex of \(\mathbb{R}^n\). \(\square\)

**COROLLARY 4.1** Rational parametric curves \(C(s) = (x_1(s), \ldots, x_m(s))^T\) with \(s \in (-\infty, +\infty)\) can be finitely represented by \(C\left(\frac{t}{1-t}\right), C\left(\frac{-t}{1-t}\right)\), using only \(0 \leq t \leq 1\).
COROLLARY 4.2 Rational parametric surfaces $S(s_1, s_2) = (x_1(s_1, s_2), \ldots, x_m(s_1, s_2))^T$ with $s_1, s_2 \in (-\infty, +\infty)$ can be finitely represented by

$$S\left(\frac{t_1}{1-t_1-t_2}, \frac{t_2}{1-t_1-t_2}\right), \quad S\left(\frac{-t_1}{1-t_1-t_2}, \frac{t_2}{1-t_1-t_2}\right)$$
$$S\left(\frac{-t_1}{1-t_1-t_2}, \frac{-t_2}{1-t_1-t_2}\right), \quad S\left(\frac{t_1}{1-t_1-t_2}, \frac{-t_2}{1-t_1-t_2}\right)$$

using only $t_1, t_2 \geq 0 \land t_1 + t_2 \leq 1$.

4.4 Orientation of the Mappings

An interesting interplay between mathematics and computer graphics arises when the surface total mapping equations (4.6) are used to render a surface on a high-performance graphics workstation. Such computers commonly support advanced surface rendering techniques incorporating a variety of surface materials and lighting models. When a surface is rendered as a shiny material, the phenomenon of "specular highlighting" occurs (see, e.g., [44]). The specular highlight is the bright white spot on the shiny surface that corresponds to light that reflects off the surface directly at the viewer, and the intensity fades smoothly around the highlight. The phenomenon adds realism to a scene.

The surfaces in Figures 4.9 and 4.10 were originally displayed in such a manner. What is noticeable when they are viewed on a workstation screen, however, is that two of the four pieces (corresponding to two of the four parametric domain quadrants) appear matte rather than glossy, as they should due to the specular reflection.

In addition to the surface parametric equations, the surface normals are used in calculating the reflected light intensities. The direction of the normal vector at a point on the surface is generally considered to point towards the "outside" of the surface (the side that reflects light). In this case, we find that that two of the four pieces have normal vectors that are pointing in directions that are inconsistent with the normal vectors on the other two pieces. We appeal to some elementary differential geometry to explain and correct this detail. To do this we will consider the domain
transformations (4.6) as affine maps, defined on the open triangle \( u > 0, v > 0, u + v < 1 \).

Parametric equations of a surface are a map between a plane denoted by \((s, t)\) and space denoted by \((x, y, z)\); when we use the total mapping equations we are composing this map with another one from a new domain plane in \((u, v)\) to the \((s, t)\) plane. The direction of the normal at a point on the surface depends on whether the map from \((u, v)\) space to \((s, t)\) space is orientation-preserving or orientation-reversing [66]. In our case, we show that two of the maps (4.6) are orientation-preserving and two are orientation-reversing.

To do this we will consider the transformations (4.6) as functions from the \((u, v)\) plane to the \((s, t)\) plane, where the domain is restricted to the open triangle \( u > 0, v > 0, u + v < 1 \) (instead of considering them as projective linear transformations between projective spaces). That is,

\[
\begin{align*}
s &= F_1(u, v) = i \frac{u}{1 - u - v} \\
t &= F_2(u, v) = j \frac{v}{1 - u - v}
\end{align*}
\]

A map between \( \mathbb{R}^n \) and \( \mathbb{R}^m \) is orientation-preserving if the determinant of the Jacobian matrix is positive, and orientation-reversing if the determinant is negative. This determinant is given by

\[
J = \begin{vmatrix}
\frac{\partial F_1}{\partial u} & \frac{\partial F_1}{\partial v} \\
\frac{\partial F_2}{\partial u} & \frac{\partial F_2}{\partial v}
\end{vmatrix}
\]

Calculating \( J \) involves some tedious but simple arithmetic, which we omit here. Its value is

\[
J = \frac{ij}{(1 - u - v)^3}
\]

Now on the domain of interest \( u > 0, v > 0, u + v < 1 \), the denominator of \( J \) is always positive, and hence the sign of \( J \) depends entirely on the signs of \( i, j \). Thus we conclude that the maps (4.6) are orientation-preserving when the domain is to map onto the first and third quadrant of the \((s, t)\) domain, and orientation-reversing when the domain must map to the second and fourth quadrants.
These calculations agree with our observations in practice. A trivial correction is to simply reverse the normal directions in vertices of polygons generated to approximate the surface, for the maps of the second and fourth quadrants.

4.5 Projective Quadratic Reparameterizations

There are other benefits of considering rational parametrics over a projective domain. The problem of computing the so-called normal parameterizations and missing points of rational parametrics is discussed in [28]. The authors consider curves and surfaces over affine domains, and present general techniques to solve these problems. However, their algorithms are expensive and non-trivial to implement. They present solutions for conics and quadrics derived from lengthy machine computations using their implementation; for quadrics, they report that they were unable to compute normal parameterizations using their current implementation, and pose it as an open problem.

By considering rational parametrics over projective domains, we derive key results of their work regarding conics and quadrics, and solve the problem for quadrics that they left open. Our method does not resemble theirs in any way, and does not lead to a general solution. However, it must be noted that our results for these special cases require no machine computation for their derivation.

4.5.1 Definitions

In this section, the problems are introduced using terminology and definitions from [28].

**DEFINITION 4.1** Let $K$ be a field, and $E$ an algebraically closed extension field of $K$ that contains $t_1, \ldots, t_m$, indeterminates that are algebraically independent over $K$. If $P_i, Q_i \in K[t_1, \ldots, t_m]$ are polynomials in these symbols with $Q_i \neq 0$, then

$$y_1 = \frac{P_1}{Q_1}, \ldots, y_n = \frac{P_n}{Q_n}$$

(4.9)

with $m, n \geq 1$ are called a set of (affine) parametric equations.
It will always be assumed that gcd\( (P_i, Q_i) = 1 \). The degree of the set of parametric equations is the maximum degree of its constituent polynomials \( P_i, Q_i \).

**DEFINITION 4.2** The image of a set of parametric equations (4.9) is denoted by \( \text{IM}(P, Q) \), and defined by

\[
\text{IM}(P_1, \ldots, P_n, Q_1, \ldots, Q_n) = \{(y_1, \ldots, y_n) | \exists t \in E^m (y_i = P_i(t)/Q_i(t))\}
\]  

An irreducible algebraic curve or surface is a set of points that satisfy a system of polynomial equations, i.e. it is an irreducible variety. A set of parametric equations for a curve or surface may not describe all the points of its variety, that is, the image of the parametric equations is only a subset of the variety. The remaining points of the variety are called the missing points of the parameterization.

**DEFINITION 4.3** A set of parametric equations (4.9) is termed **normal**, if \( \text{IM}(P, Q) \) is an irreducible variety.

Informally, an affine parameterization of a curve or surface is termed normal if all the points of the curve or surface can be given by it.

These definitions are in terms of an algebraically closed extension field \( E \). To lend the problem more practical applicability, especially in computer graphics, Gao and Chou give a stronger definition using only the field of reals. With this stronger definition, the problem of finding normal parametric equations is harder, but the results once found are of more use. By taking infinite parameter values into account, we can compute normal parameterizations and in some cases real field normal parameterizations.

We now define real field normal parameterizations:

**DEFINITION 4.4** A set of parametric equations of the form (4.9) with \( K = \mathbb{R} \) is called **normal in the real number field** if there is an irreducible variety \( V \) in \( \mathbb{R}^n \) such that
1. For any \( t \in \mathbb{R}^m \), if \( Q_1(t) \ldots Q_n(t) \neq 0 \), then \((P_1(t)/Q_1(t), \ldots, P_n(t)/Q_n(t)) \in V \).

2. For any \( p \in V \) there is a \( t \in \mathbb{R}^m \) with \( p = (P_1(t)/Q_1(t), \ldots, P_n(t)/Q_n(t)) \).

Note that Gao and Chou's method does not in general find parametric equations that are normal in the real number field; however, they do prove that the normal parameterizations that they produce for conics are actually normal in the real number field. We will also compute parameterizations for conics that are normal in the real number field, as they do, and also find parameterizations for three important quadrics that are normal in the real number field.

4.5.2 Missing Points

If a set of parametric equations are not normal, then some points on the curve or surface are not given by any parameter value. Given a parameterization that is not normal, one would like to compute the points that are missing, and also a normal parameterization.

We consider four cases of conics and quadrics: ellipses, ellipsoids, hyperboloids of one sheet, and hyperboloids of two sheets. In [28], missing points are given for all of these, and a real field normal parameterization is given for the ellipse only. Their program was used to derive the results. These are the most important cases; the remaining conics and quadrics are not of as much interest since real field normal parameterizations for them are readily found, or they are degenerate. The conics and quadrics are assumed to be in standard form, since they can be easily transformed to such, as mentioned in [28] and earlier in this thesis.

Using some algebraic geometry, we will compute real missing points and real field normal parameterizations for all four cases. The key insight behind our methods is this: the only real missing points in the standard parameterizations are those corresponding to real, infinite parameter values. Once we show this, we can use projective reparameterizations to map finite parameter values in one domain to infinite values in another domain. In the previous section, we used linear transformations, in which
case more than one is necessary. Using quadratic transformations, we can show that only one is necessary; hence, after reparameterizing, we will have real field normal parameterizations.

In this section, the real missing points are computed for each standard parameterization of the four cases, and shown to correspond to infinite parameter values. In all cases, we only consider real, non-degenerate conics and quadrics, and compute only the real missing points.

CASE 1 Let ellipses be given in the standard form
\[ \frac{x^2}{a^2} + \frac{y^2}{b^2} - 1 = 0 \] (4.11)
where \( a, b \) are real numbers.

Then, the standard parameterization is
\[ (x, y) = \left( \frac{a(s^2 - 1)}{s^2 + 1}, \frac{2bs}{s^2 + 1} \right) \] (4.12)

**Lemma 4.4** The only missing point of the parameterization (4.12) is \((a, 0)\).

**Proof.** Since the real portion of the ellipse is closed, all its real points lie at finite distances, including the image of the domain point \( s = \infty \). Thus, this gives us one missing point. We compute its coordinates, and show it is the only missing point. By considering the parameterization over a projective domain \((S, U)\), we derive the parametric equations
\[ (x, y) = \left( \frac{a(S^2 - U^2)}{S^2 + U^2}, \frac{2bSU}{S^2 + U^2} \right) \]
Substituting \((S, U) = (S, 0)\), the projective representation of \( s = \infty \), we find \((x, y) = (a, 0)\). Thus \((a, 0)\) is a missing point. To see that it is unique, first compute an inversion of the parameterization, i.e., a function that gives the parameter \( s \) corresponding to a point \((x, y)\). Inversions for rational algebraic curves and surfaces can be computed using techniques developed recently using resultants [49],[50], or Gröbner-basis
Without specifying details, we will simply present the inversions that we computed. For curves, subresultant remainder sequence mechanisms in [16] were used. For surfaces, we used utilities from the symbolic algebra system Maple V to calculate Gröbner bases of polynomial ideals using a lexicographic (or elimination) ordering.

For the ellipse in standard form, an inversion is given by

\[ s = \frac{ay}{b(a - x)} \]

The inversion is defined for \( x \neq a \). The only point on the ellipse with \( x = a \) is \((a, 0)\). Hence every point in the real variety of the ellipse is contained in the image of the parameterization, except \((a, 0)\). Thus the latter is the only missing point of the standard ellipse parameterization (4.12).

CASE 2 The standard equation of an ellipsoid is

\[ \frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} - 1 = 0 \]  

and its standard parameterization is given by

\[
(x, y, z) = \left( \frac{2au}{u^2 + v^2 + 1}, \frac{2bv}{u^2 + v^2 + 1}, \frac{c(u^2 + v^2 - 1)}{u^2 + v^2 + 1} \right)
\]  

LEMMA 4.5 The only real missing point of the parameterization (4.14) is \((0, 0, c)\).

Gao and Chou derive this result from their computer program. However, they do not include the qualifying term "real" as we do; it is therefore not clear if they are implying that the parameterization only has one missing point. Surface parameterizations can in fact have entire missing curves, which may be real or complex. For instance, the ellipsoid parameterization specialized to spheres is shown to have two missing complex lines in [27], that lie on the complex sphere. The point \((0, 0, c)\) is the only real point lying on these missing lines.

PROOF. We know that the image points of the line at infinity in the domain of the parameterization must be missing from the surface. To calculate these image points,
we first homogenize the domain of the parameterization (4.14), to be over a projective plane whose points have coordinates \((U, V, W)\):

\[
(x, y, z) = \left( \frac{2aUW}{U^2 + V^2 + W^2}, \frac{2bVW}{U^2 + V^2 + W^2}, \frac{c(U^2 + V^2 - W^2)}{U^2 + V^2 + W^2} \right) \quad (4.15)
\]

All points on the line at infinity have coordinates \((U, V, 0)\) where one of \(U, V\) is non-zero. Substituting these points in the above parameterization yields \((x, y, z) = (0, 0, c)\). Hence the entire line at infinity maps to this point on the ellipsoid, which is therefore a missing point.

To show that it is the only missing point, we use an inversion of the parameterization (4.14):

\[
u = \frac{cx}{a(c - z)}, \quad v = \frac{cy}{b(c - z)}
\]

These are defined for all points \((x, y, z)\) with \(z \neq c\). The only point on the real ellipsoid with \(z = c\) is \((0, 0, c)\). Hence, every other real point of the ellipsoid is the image of a real, finite point in the parameter domain. Thus \((0, 0, c)\) is the only real missing point of (4.14). 

Note that if we were considering the complex numbers also, the entire intersection curve of the plane \(z = c\) with the ellipsoid is missing. It is easy to show that this missing curve is composed of two complex lines in space, agreeing with our earlier remarks about the work in [27].

CASE 3 The standard equation of a hyperboloid of one sheet is

\[
\frac{x^2}{a^2} + \frac{y^2}{b^2} - \frac{z^2}{c^2} - 1 = 0 \quad (4.16)
\]

and its standard parameterization is given by

\[
(x, y, z) = \left( \frac{a(u^2 - v^2 + 1)}{u^2 + v^2 - 1}, \frac{2buv}{u^2 + v^2 - 1}, \frac{2cu}{u^2 + v^2 - 1} \right) \quad (4.17)
\]
LEMMA 4.6 The real missing points of (4.17) lie on a space conic on the implicit surface, which can be described by the intersection of a plane and a cylinder. However, one point on this conic is not missing. The set of missing points can be described by

\[\{z = 0\} \cap \left\{\frac{x^2}{a^2} + \frac{y^2}{b^2} - 1 = 0\right\} \setminus \{x + a = 0\}\]

PROOF. First, consider the image of the line at infinity in the domain of the parameterization. If these map to finite points on the surface, then they are missing points. Again, we homogenize the domain and substitute the coordinates \((U, V, 0)\) of points on the line at infinity in the domain. Homogenizing,

\[
(x, y, z) = \left(\frac{a(U^2 - V^2 + W^2)}{U^2 + V^2 - W^2}, \frac{2bVU}{U^2 + V^2 - W^2}, \frac{2cUW}{U^2 + V^2 - W^2}\right)
\]

(4.18)

After substitution, we find these points corresponding to the domain line at infinity:

\[
(x, y, z) = \left(\frac{a(U^2 - V^2)}{U^2 + V^2}, \frac{2bVU}{U^2 + V^2}, 0\right)
\]

(4.19)

where \((U, V) \neq (0, 0)\). These are easily recognizable as the projective parametric equations of the ellipse \(E\) in the \(z = 0\) plane given by

\[
\{z = 0\} \cap \left\{\frac{x^2}{a^2} + \frac{y^2}{b^2} - 1 = 0\right\}
\]

(4.20)

which is the curve of intersection of the hyperboloid of one sheet with the \(z = 0\) plane.

An inversion for (4.17) is

\[
u = \frac{c(x + a)}{az}
\]

(4.21)

\[
v = \frac{cy}{bz}
\]

(4.22)

showing that all points on the hyperboloid except those in the \(z = 0\) plane are the image of some real, finite point in the parameter domain. The intersection of the hyperboloid and the \(z = 0\) plane gives us the ellipse \(E\), which is the image of the line at infinity of the domain. Consider a point of \(E\); it is the image of some parameter point \(P\) on the line at infinity. Suppose also that it is the image of some other point \(Q\) in the parameter domain. Then the line \(L\) joining \(P, Q\) maps to a conic, a line, or
a point on the surface, depending on whether it passes through zero, one, or two base points, respectively (base points are discussed in Chapter 5, giving references). Since a line and a conic are non-singular curves, the only way \( P \) and \( Q \) can map to the same surface point is if the entire line maps to the same point, i.e., the line passes through two base points. By direct calculation, we find that the quadric parameterization (4.17) has the two base points \((0, \pm 1)\), and hence there is only one such line, \( u = 0 \). The image of this line \( u = 0 \) is the surface point \((-a, 0, 0)\) lying on the ellipse \( E \). Hence this point is not missing, and the lemma follows. \( \square \)

CASE 4 The standard equation of a hyperboloid of two sheets is

\[
\frac{x^2}{a^2} + \frac{y^2}{b^2} - \frac{z^2}{c^2} + 1 = 0
\]

and its standard parameterization is given by

\[
(x, y, z) = \left( \frac{2au}{u^2 + v^2 - 1} - \frac{2bv}{u^2 + v^2 - 1} - \frac{c(u^2 + v^2 + 1)}{u^2 + v^2 - 1} \right)
\]

LEMMA 4.7 The sole real missing point of (4.24) is \((0, 0, c)\).

\( \square \)

The proof is similar to the one for ellipsoids, and therefore omitted.

4.5.3 Normal Parameterizations

We are now ready to compute, for the four cases described above, parameterizations that are normal in the real number field. We will actually derive slightly more general results, first for curves, then for surfaces. In the ensuing discussion, only real values are considered for polynomial and rational function coefficients.

THEOREM 4.3 Let an affine plane algebraic curve be given by parametric equations \( C(s) = (x(s), y(s)) \) over an affine domain. Suppose the only missing point in the real affine variety of the curve is the image of the point at infinity in the domain. Then a parameterization of \( C(s) \) that is normal in the real number field, according
to definition (4.4), is derived by applying to $C(s)$ the following projective quadratic domain transformation, expressed in fractional form as

$$ s = \frac{t}{1 - t^2} \quad (4.25) $$

In fact, $\text{IM}(x(s(t)), y(s(t)))$ equals the set of points of the curve's real variety even when restricting $t$ to the interval $[-1, 1]$.

**PROOF.** It is quite easy to see that the point $t = \pm 1$ of the new parameter domain maps to the point at infinity of the original. Viewing the transformation above as a homogeneous quadratic transformation between the new projective domain $(T, V)$ and the original projective domain $(S, U)$, it can be expressed as:

$$ S = TV $$
$$ U = V^2 - T^2 $$

Then the points $(\pm 1, 1)$ of the $(T, V)$ domain are transformed to $(1, 0)$ of the $(S, U)$ domain, which is the point at infinity.

Going back to the affine transformation, let $s$ be any real, finite value. For any such $s$, we must show that there exists a real value of $t$ that gives rise to this $s$. That is, the equation

$$ s = \frac{t}{1 - t^2} $$

in $t$ must have a real solution for every real $s$. This equation can be written as

$$ s(1 - t^2) - t = 0 $$

Then when $s = 0$, $t = 0$. If $s \neq 0$, there are two solutions for $t$, both real. Choose the solution

$$ t_- = \frac{\sqrt{1 + 4s^2} - 1}{2s} $$

Then

$$ t_-^2 - 1^2 = \frac{2(1 - \sqrt{1 + (2s)^2})}{(2s)^2} < 0 $$
for all $s \neq 0$, and hence $|t_{-}| < 1$. Thus for every $s \neq 0$, there is a $t \in [-1, 1]$ that maps to $s$. □

**COROLLARY 4.3** A normal parameterization for the ellipse (4.11) is given by

$$ (x, y) = \left( \frac{-a(t^4 - 3t^2 + 1)}{t^4 - t^2 + 1}, \frac{2b(t - t^3)}{t^4 - t^2 + 1} \right) \quad (4.26) $$

Figure 4.11 shows a graph of this parameterization specialized as $a = 1, b = 1$ (yielding a circle) for $t \in [-1, 1]$. Note that the entire real curve is covered; compare especially to figure 4.5. The parameter values are “small” and hence the parametric compression is moderate. Figure 4.12 shows the application of theorem 4.3 to a rational quartic ellipse.

**THEOREM 4.4** Let an affine algebraic surface be given by parametric equations $C(s, t) = (x(s, t), y(s, t), z(s, t))$ over an affine domain. Suppose the only missing points in the real affine variety of the surface is the image of the real line at infinity in the domain. Then a parameterization of $C(s, t)$ that is normal in the real number field, according to definition (4.4), is derived by applying to $C(s, t)$ the following projective quadratic domain transformation, expressed in fractional form as

$$ s = \frac{u}{1 - u^2 - v^2} \quad (4.27) $$

$$ t = \frac{v}{1 - u^2 - v^2} $$

In fact, the image of new parameterization in $(u, v)$ equals the set of points of the surface’s real variety even when restricting $u^2 + v^2 \leq 1$.

**PROOF.** In this case, the curve $1 - u^2 - v^2 = 0$, i.e. the unit circle of the new domain, maps to the line at infinity in the $(s, t)$ domain. Viewing transformation above as a homogeneous quadratic transformation between the new projective domain $(U, V, W)$ and the original projective domain $(S, T, R)$, it can be expressed as:

$$ S = UW $$

$$ T = VW $$

$$ R = W^2 - U^2 - V^2 $$
Then any point \((U, V, W)\) such that \(W^2 - U^2 - V^2 = 0, W \neq 0\) is mapped to a point \((S, T, 0)\), i.e. to a point on the line at infinity.

Now let \((s, t)\) be any real point. We will show that for every \((s, t)\) there is a \((u, v)\) satisfying (4.27) such that \(u^2 + v^2 \leq 1\). Consider the equations (4.27) in the variables \((u, v)\) alone. Then if \((s, t) = (0, 0)\), there is a solution \((u, v) = (0, 0)\). Otherwise, we have the equivalent system of equations

\[
s(1 - u^2 - v^2) - u = 0
\]
\[
t(1 - u^2 - v^2) - v = 0
\]

We can homogenize this system of equations with a variable \(W\); then, using Sylvester's resultant to eliminate either \(u\) or \(v\). For any \((s, t) \neq 0\), there are four solutions. Two are complex solutions at infinity, given in homogeneous coordinates \((U, V, W)\) as \((1, \pm i, 0)\). The other two are real and finite:

\[
(-s(\sqrt{1 + 4(s^2 + t^2)} + 1), \quad -t(\sqrt{1 + 4(s^2 + t^2)} + 1))
\]
\[
(\frac{s(\sqrt{1 + 4(s^2 + t^2)} - 1)}{2(s^2 + t^2)}, \quad \frac{t(\sqrt{1 + 4(s^2 + t^2)} - 1)}{2(s^2 + t^2)})
\]

Let \((u_-, v_-)\) denote the latter; then

\[
u_-^2 + v_-^2 - 1 = \frac{1 - \sqrt{1 + 4(s^2 + t^2)}}{2(s^2 + t^2)} < 0
\]

This proves the theorem. \( \Box \).

**COROLLARY 4.4** Real field normal parameterizations for the ellipsoid and hyperboloid of one and two sheets (4.13, 4.16, 4.23) can all be derived from the standard ones by applying (4.27). Then the image of the closed unit disk of in the new parameter domain equals the real variety of the original surface.

Figure 4.13 shows graphically the normal parameterization of a sphere; the image is derived by applying theorem 4.4 to a sphere parameterization (derived from equations (4.14) by setting \(a = b = c = 1\), and then mapping the new parameterization restricted to the unit disk \(u^2 + v^2 \leq 1\) of the parameter domain. Figure 4.14 shows a whole steiner quartic using this method.
4.6 Single Finite Representation of Parametric Varieties

The above results generalize directly to parametric varieties of any dimension. In general, we can cover an $n$-dimensional parametric variety using a single parameterization of twice the degree, where the parameter domain is the $n$-dimensional hypersphere.

**THEOREM 4.5** Consider a real parametric variety of dimension $n$ in $\mathbb{R}^n$, $n < m$, which is parameterized by the equations

$$V(s) = \begin{pmatrix} x_1(s_1, \ldots, s_n) \\ \vdots \\ x_m(s_1, \ldots, s_n) \end{pmatrix}, \quad s_i \in (-\infty, +\infty)$$

The single projective quadratic reparameterization given in fractional affine form as

$$s_i = \frac{t_i}{1 - t_1^2 - t_2^2 - \ldots - t_n^2}, \quad i = 1, \ldots, n \quad (4.28)$$

yields a finite representation $V(t)$ of the rational variety $V(s)$, restricting $t_1^2 + \ldots + t_n^2 \leq 1$.

**PROOF.** In this case, the proof consists of showing every point in the old domain $\mathcal{RP}^n$ is the image of some point in the new domain $\mathbb{R}^n$, using only the single transformation (4.28). We will show that the unit hypersphere in the new domain space maps onto the hyperplane at infinity of the old domain space, and every other point in the old parameter domain space is the image of a corresponding point in the new domain, which lies in the interior of the unit hypersphere.

Once again let $s = (cs_1, \ldots, cs_{n+1}) \in \mathcal{RP}^n$, $c \in \mathbb{R}$, $c \neq 0$ and we fix $s_{n+1} = 0$ as the hyperplane at infinity. Let $t \in \mathbb{R}^n$. The equations (4.28) are a map from $\mathbb{R}^n \to \mathcal{RP}^n$:

$$cs_i = t_i$$

$$\vdots$$

$$cs_n = t_n$$

$$cs_{n+1} = 1 - (t_1^2 + \ldots + t_n^2)$$
First, consider points \( s \) on the hyperplane at infinity, i.e. \( s_{n+1} = 0 \). Then (4.28) yields a system of equations

\[
\begin{align*}
    cs_i &= t_i & i &= 1, \ldots, n \\
    0 &= 1 - (t_1^2 + \ldots + t_n^2)
\end{align*}
\]

which has two real solutions, given below:

\[
\begin{align*}
    c &= \pm \frac{1}{\sqrt{\sum_{i=1}^{n} s_i^2}} \\
    t_i &= cs_i = \pm \frac{s_i}{\sqrt{\sum_{i=1}^{n} s_i^2}}
\end{align*}
\]

For either solution, \( t_1^2 + \ldots + t_n^2 = 1 \), showing that \( t \) lies on the unit hypersphere in \( \mathbb{R}^n \).

Second, consider affine points \( s \in \mathbb{R}^n \subset \mathbb{R}P^n \). We can set \( s_{n+1} = 1 \), w.l.o.g., and then (4.28) yields the system of equations

\[
\begin{align*}
    cs_i &= t_i & i &= 1, \ldots, n \\
    c &= 1 - (t_1^2 + \ldots + t_n^2)
\end{align*}
\]

This system also has two real solutions, given by

\[
\begin{align*}
    c &= \frac{-1 \pm \sqrt{1 + 4 \sum_{i=1}^{n} s_i^2}}{2 \sum_{i=1}^{n} s_i^2} \\
    t_i &= cs_i
\end{align*}
\]

Choosing \( c = \frac{-1 + \sqrt{1 + 4 \sum_{i=1}^{n} s_i^2}}{2 \sum_{i=1}^{n} s_i^2} \), some simple algebra shows that \( t_1^2 + \ldots + t_n^2 < 1 \).

Thus if \( s \) is on the hyperplane at infinity, there is a point \( t \) on the unit hypersphere that maps it; otherwise, there is a point \( t \) in the interior of the unit hypersphere that maps it. Only the single map (4.28) is necessary. \( \Box \)

**COROLLARY 4.5** Using theorem 4.5, we can compute normal parameterizations for the ellipse, ellipsoid, hyperboloid of one sheet, and hyperboloid of two sheets.
4.7 Summary

In this chapter we examined various applications of projective transformations of the parameter domain. Projective linear reparameterizations have applications such as enabling the complete display of curves and surfaces despite graphing only a finite number of bounded portions of the parameter domain. The problem of parametric compression that occurs in displaying parametric curves and surfaces was explained, and we discussed some computer graphics implications of such reparameterizations, namely the orientation of surface normals for shading computations.

Furthermore, projective quadratic reparameterizations were used to solve the open problem of finding normal parameterizations of the natural quadrics, or parameterizations that do not have any missing points.
Figure 4.2 Total mapping of rational parametric curve (quartic)

Figure 4.3 An arc of a circle and its complementary segment
Figure 4.4 An arc of a quartic curve and its complementary segment

Figure 4.5 Discretization of rational parametric curve, with parameter compression
Figure 4.6 Discretization of polynomial parametric curve \((s^3, -s^3)\)

Figure 4.7 Discretization of polynomial parametric curve \((s^2 - 1, s^2 - s^3)\)
Figure 4.8 Total mapping of sphere using three rectangular patches

Figure 4.9 Total mapping of rational parametric surface (sphere)
Figure 4.10  Total mapping of rational parametric surface (steiner surface)

Figure 4.11  Use of projective quadratic transformation (circle)
Figure 4.12 Use of projective quadratic transformation (quartic)

Figure 4.13 Use of projective quadratic transformation (sphere)
Figure 4.14 Use of projective quadratic transformation (steiner surface)
5. ROBUST DISPLAY OF ARBITRARY RATIONAL PARAMETRICS

5.1 Introduction

We now turn our attention to the problem of visualizing parametric curves and surfaces. In this chapter we give an algorithm to accurately display the entire real part of a rational parametric surface.

Points on a parametric surface can be generated by sampling the parametric functions over the parameter domain. Because of this, the display of patches of parametric surfaces is well-understood [26, 96, 68, 44]. More recent methods address in detail the problem of generating a polygonal mesh on a surface that is sensitive to variations in surface curvature: view-dependent methods [109] as well as view-independent [67, 86, 16]. All these methods assume continuity of the parametric functions over the portion of the domain that is being sampled.

The rational functions that define a surface can be viewed as a map from \( \mathbb{R}^2 \) into \( \mathbb{R}^3 \). We shall investigate the problem of visualizing the shape of a parametric surface, even if the map generating it is undefined at some domain points. Many surfaces (including simple ones such as some quadrics) are given by maps which are not defined everywhere.

In this formulation the problem is of more interest to mathematicians, than to CAD designers who work with patches of rational parametric surfaces. The latter usually express the rational functions defining the surface in terms of the Bezier or B-spline basis [20]; the rational functions are restricted to a standard part of the domain, and the weights are chosen such that the functions are defined at all points of the standard domain. Our primary motivation to investigate this problem grew out of unsuccessful attempts to visualize the output of cubic surface parameterization algorithms (using our own implementation of domain sampling techniques, as
well as commercial symbolic algebra and graphing programs e.g. Maple V, Mathematica). The inadequacy of domain-sampling techniques for displaying many interesting rational parametric surfaces led to a search for a better method.

A similar problem exists for parametric curves, and we shall investigate that also. We now formulate the problem more precisely. Let a parametric curve be given by two rational functions:

\[ x(s) = \frac{F_1(s)}{F_3(s)}, \quad y(s) = \frac{F_2(s)}{F_3(s)} \]  

(5.1)

and a parametric surface by three:

\[ x(s, t) = \frac{F_1(s, t)}{F_4(s, t)}, \quad y(s, t) = \frac{F_2(s, t)}{F_4(s, t)}, \quad z(s, t) = \frac{F_3(s, t)}{F_4(s, t)} \]  

(5.2)

In each case we assume that the polynomials \( F_i \) have real coefficients and have no common factor. Then:

1. Given real numbers \( x_{\text{min}} \leq x_{\text{max}}, y_{\text{min}} \leq y_{\text{max}}, \) compute the set \( C \) of all points \( (x, y) \) on the curve that satisfy \( (x, y) \in [x_{\text{min}}, x_{\text{max}}] \times [y_{\text{min}}, y_{\text{max}}]. \)

2. Given real numbers \( x_{\text{min}} \leq x_{\text{max}}, y_{\text{min}} \leq y_{\text{max}}, z_{\text{min}} \leq z_{\text{max}}, \) compute the set \( S \) of all points \( (x, y, z) \) on the surface that satisfy \( (x, y, z) \in [x_{\text{min}}, x_{\text{max}}] \times [y_{\text{min}}, y_{\text{max}}] \times [z_{\text{min}}, z_{\text{max}}]. \)

In practice, we shall be satisfied with piecewise-linear approximations to the sets \( C \) and \( S. \) However, we shall also be interested in approximations that represent the shape of the curve or surface accurately (some approximations that suffice for display may not meet this criterion).

The problem can be extended to include rational varieties, but we shall not discuss this here. The general flavor of the methods discussed will still apply, although implementing higher-dimensional methods would require more tools.

The rest of this chapter is organized as follows. To start with, two approaches are discussed: either directly approximating the curve or surface in the range space of the parametric functions, or computing those portions of the domain that map
onto the desired portions of the range. We argue that the domain-space approach is preferable in this context. Then, the specific difficulties that arise in any domain-space approach (domain poles and base points) are explained in detail. After this, we describe an algorithm for displaying curves, and one for surfaces. We explain the surface algorithm in detail, and then discuss situations in which it can fail and how it can be improved, indicating where more work is needed. Finally, we give some details of our implementation.

5.2 Domain and Range Space Approaches

One way to construct a piecewise-linear approximation to a parametric surface is to evaluate the parametric functions at various points on the parameter domain, and link together the resulting surface points to form an approximating mesh. When considering arbitrary rational parametric surfaces, the parametric functions may not be defined at some points, since rational functions are not defined at points where the denominator vanishes. Such points are called poles, and usually correspond to surface points at infinity. The exception occurs when all the polynomials \( F_i(s, t) \) vanish there (an event that can happen only finitely many times since \( F_i \) have no common divisor, by assumption). In this case the parameter point is a domain base point.

In the next section we explore poles and base points in detail, showing examples of how they can cause domain sampling techniques to fail.

Another way to approach the problem is to work directly in the range space of the rational function map. Since we are only interested in portions of a surface inside a bounding box, and poles correspond to surface points at infinity, a range-space method can avoid explicitly evaluating the rational functions at poles (base points still cause problems).

If a single-view display of the surface is all that is needed, a ray-tracing methods such as [63] can generate excellent images. However, once a surface is polygonized, modern graphics workstations can be used to quickly view the surface from many
viewpoints, which is preferable. Furthermore, piecewise-linear approximations provide a geometric data structure representation of a surface, and are convenient in other aspects. Hence we focus on them.

The following system of equations is equivalent to (5.2):

\[
\begin{align*}
F_3(s, t)x - F_1(s, t) &= 0 \\
F_3(s, t)y - F_2(s, t) &= 0 \\
F_3(s, t)z - F_3(s, t) &= 0
\end{align*}
\]

One can implicitize the parametric surface by eliminating \( s, t \) from this system using one of the many available methods [74, 75, 23, 30, 27, 46, 78] and then approximate the resulting implicit surface using a space-subdivision method [21] or approximation techniques [9, 10]. Note that a parametric surface of degree \( n \) could have an implicit equation of degree \( n^2 \).

A better alternative is to use higher-dimensional versions of the marching method described in [15, 57], e.g. [30, 59]. These techniques have been applied with success to approximate surfaces defined by large systems of polynomial equations. However, marching methods suffer from difficulties such as tracing through singularities, and they need a starting point on each sheet of the surface. The space-subdivision methods also have trouble handling singularities.

Since we would like to display surfaces with complicated singularities and several real sheets, we avoid the range-space approach. We show instead that a careful evaluation of the domain is sufficient to generate an accurate piecewise-linear approximation to the surfaces.

5.3 Domain Sampling Using Rational Maps

In this section we explain why domain base points and poles sometimes cause sampling techniques to fail, and give simple examples that are representative of the kinds of failures that occur. The main problem is that domain sampling techniques which
don't take poles and base points into account can generate surface approximations
which do not accurately represent the topology of the surface.

5.3.1 Domain Poles

Inability to evaluate a rational function at a pole (i.e., generating a divide by
zero exception in a numerical program) is not the main reason that domain sampling
methods fail when poles are present. Even if a domain sampling method avoids
evaluating a rational map at a pole, it may construct an approximation that does
not reflect the actual shape of the curve or surface. This happens when a part of the
domain that contains a pole is mapped onto the curve or surface.

When a parameterization contains poles, the curve or surface may have multiple
branches or sheets, e.g. the hyperbola

\[ x(s) = s, \quad y(s) = \frac{1}{s} \]

has a pole at \( s = 0 \). The curve consists of two branches.

A simple domain sampling algorithm for approximating this hyperbola might se-
lect a closed interval \([a, b]\) in the parameter domain, generate \( n \) equally-spaced param-
eter values \( s_i = a + i\frac{b-a}{n-1}, i = 0, \ldots, n \), and then connect the points \((x(s_{i-1}), y(s_{i-1}))\),
\((x(s_i), y(s_i))\) with a straight-line segment. In this example, a line segment could be
drawn between points whose parameter values lie on opposite sides of a pole. As a
result, the approximation does not accurately represent the shape of the curve. Fig-
ure 5.5 shows the output of the program Mathematica for plotting the hyperbola over
the domain interval \( s \in [-\frac{1}{2}, \frac{1}{2}] \).

Poles can cause this problem even when there aren't multiple branches or sheets.
The problem is particularly acute for surfaces. For instance, a hyperboloid of one
sheet with implicit equation \( x^2 + y^2 - z^2 - 1 = 0 \) is a surface whose real part is
connected. However, if we work from the equivalent parametric representation

\[
\begin{align*}
x(s, t) &= \frac{t^2 - s^2 + 1}{s^2 + t^2 - 1}, \quad y(s, t) = \frac{2st}{s^2 + t^2 - 1}, \quad z(s, t) = \frac{2t}{s^2 + t^2 - 1}
\end{align*}
\] (5.3)
then problems arise because of the pole curve described by \( s^2 + t^2 - 1 = 0 \) in the parameter domain. The right picture in Figure 1 shows the output produced by Maple \( V \) for this surface with \((s, t) \in [-2, 2] \times [-2, 2] \) (a domain region containing the pole curve).

A small digression is in order about the programs Mathematica and Maple \( V \). Both programs use sophisticated strategies for graphing curves and surfaces. For instance, Maple \( V \) uses adaptive stepping combined with cubic spline approximation and generally approximates curves and surfaces very well. However, both programs use domain sampling techniques which are not equipped to handle poles and base points, and hence fail for simple examples such as the above.

5.3.2 Domain Base Points

We assumed that the numerators and common denominator of the rational maps (5.1,5.2) have no common factor. For a curve, this means that there is no parameter value that causes both numerator and denominator of a parametric function to vanish. For surfaces, the situation is different, since it is still possible that there are a finite number of points \((a, b)\) such that \( F_1(a, b) = F_2(a, b) = F_3(a, b) = F_4(a, b) = 0 \). Each such point is called a base point of the parametric surface. Information about base points can be found in books on algebraic geometry such as [53, 101, 118]. Interesting material on base points in the context of CAGD appears in [27, 77, 100, 113]. In particular, [113] shows how to represent patches with up to six sides in the triangular rational Bezier patch form, by a clever use of domain base points.

Base points are problematic since there is no one surface point for the corresponding domain point. To each base point there actually corresponds a rational curve on the surface [101]. Approaching the base point along different directions leads to different points on the surface; the points corresponding to all directions form a space curve that lies on the surface. Since there is no parameter value for points on this curve (at which the surface map is defined), the entire curve will be missing from the parametric surface. Such a curve is called a seam curve. Even if poles are taken care
of in some way, the seam curves can show up as gaps on the surface, as in Figure 5.1. This figure shows the hyperboloid of one sheet given by (5.3). This parameterization has the two base points \((s, t) = (\pm 1, 0)\). The corresponding seam curves can be parameterized in parameters \(u, v\), giving the lines \((x(u), y(u), z(u)) = (-1, u, u)\) and \((x(v), y(v), z(v)) = (-1, v, -v)\) on the surface.

If base points are not taken into account, the domain sampling density may need to be unnecessarily dense (with respect to surface curvature) in order for the gaps to be narrow. Furthermore, even if the gap is narrow enough to suffice for display, the surface approximation will not correctly represent the surface's topology because of the gap.

5.4 Accurate Display of Rational Parametric Curves

We now give an algorithm for generating a piecewise-linear approximation to a rational parametric curve (5.1) given by a map that might contain poles. First, apply the projective linear transformations of Chapter 4 to map the entire curve in two parts, where each piece is parameterized over the closed unit interval \([0, 1]\). This is computationally trivial and simplifies the interval we must consider to always be \([0, 1]\). More importantly, it eliminates the possibility of parameter compression as discussed in Chapter 4.

Now perform the following procedure for each of the two curve pieces. First, intersect each edge of the bounding region with the curve. The edges are implicitly specified by the equations

\[
x = x_{\min}, \quad x = x_{\max}, \quad y = y_{\min}, \quad y = y_{\max}
\]

The intersection is performed by substituting \(x(s), y(s)\) into these equations and equating to zero the resulting polynomials in \(s\). Those roots that correspond to points on the boundary are sorted and duplicates eliminated to derive a set of distinct parameter values \(s_1 < s_2 \ldots < s_m\). These numbers partition \([0, 1]\) into closed intervals. Each interval corresponds to a portion of the curve that is either inside or
outside the bounding region. To test an interval, pick a value \( r \) at random from the interior of each interval, and compute the point \( p = (x(r), y(r)) \). If \( F_2(r) = 0 \), pick another value (note that with probability one, \( r \) will not be a pole since there are only a finite number of poles). If \( p \) is inside the bounding region, then that closed interval must be mapped, otherwise it must be excluded. An interval that is chosen for mapping cannot contain a pole, and \( x(s), y(s) \) can therefore be sampled safely over it.

The procedure terminates when all intervals are processed in this fashion. Testing each interval for being inside or out of the (convex) bounding region is quite efficient. For a little more efficiency, the root multiplicities can be used to avoid such tests, after checking if the first interval is inside or out (because the intervals will be alternately inside or outside). Multiple roots can occur, for instance, when a curve deflects tangentially off a boundary edge. If each interval is not checked, extra care must be taken to handle special cases.

5.5 Accurate Display of Rational Parametric Surfaces

After explaining why a direct generalization of the curve algorithm to surfaces is not a good idea, we present an algorithm for surfaces. This algorithm works by using the pole curve to partition the domain into regions that are "safe" to map. These regions are mapped onto the surface and clipped against the bounding box. Base points (which also lie on the pole curve) are handled in a special way.

5.5.1 Surface Trimming

Extending the curve algorithm to surfaces eventually reduces the problem to surface trimming. However, the reduction to a trimming problem is sufficiently complicated to avoid this approach, and we now explain why.

The parametric surface is intersected with the faces of the bounding region, whose plane equations are \( x = x_{\text{max}}, y = y_{\text{max}}, \text{etc.} \) The intersection is performed by substituting the parameter functions into the equations and equating them to zero, leading
to a set of bivariate equations. Each equation implicitly defines a curve in the parameter domain.

These implicit curves jointly determine inside-outside regions of the surface with respect to the bounding box. If we extract those segments of these curves that explicitly bound the "inside" regions, we are left with a trimmed surface patch. Each curve segment must be given a clockwise/counter-clockwise orientation to complete the trimmed surface representation, and segments that combine to form a closed loop must be grouped together. The oriented curve segments together form the trim curves. Trimmed surface patches can be approximated using the methods of [37, 89].

We now show two examples of this approach. In the first example, we wish to trim the unit sphere given homogeneously by

\[
X(s, t) = 2s \\
Y(s, t) = 2t \\
Z(s, t) = 1 - s^2 - t^2 \\
W(s, t) = s^2 + t^2 + 1
\] (5.4)

against the region \(|x|, |y|, |z| \leq \frac{3}{4}\). The result of such a trim is shown in Figure 5.7, on the left. The domain curves are shown alongside; the intersection region is that which is inside the largest circle but outside the smaller ones. In this case, the six domain curves themselves form the trim curve segments, when appropriately oriented.

Now consider another quadric surface, a hyperboloid of two sheets, whose homogeneous parameterization is

\[
X(s, t) = 4s \\
Y(s, t) = 4t \\
Z(s, t) = 5t^2 + 5st - 2t + 5s^2 - 2s + 1 \\
W(s, t) = 5t^2 + 6st + 5s^2 - 1
\]
The trimmed surface is given in Figure 5.8, and once again the trim curves are shown alongside. In this case the bounding box intersection curves are all ellipses, and some segments of these curves must be extracted to construct the trim curves.

In this case the trim region is quite complicated, even though the surface in this case is only of degree two.

The situation is further complicated by the fact that the trim curves are implicitly defined; trimming algorithms need trim curves to be given either by parametric equations, or as a sorted list of points along the curve. Techniques for sorting points on algebraic curves exist, but are expensive [62].

Thus extracting the bounding segments from the six implicit curves appears to unnecessarily complicate the problem. We shall instead focus on a single curve, the pole curve of the rational map.

5.5.2 Domain Partitioning

We now present an algorithm for displaying rational parametric surfaces. As is common in domain sampling techniques, a triangulation of the parametric domain is mapped onto the surface, yielding a piecewise-linear approximation to it. Triangular surface elements have several advantages, described in detail in [109].

The main idea of the algorithm is simple. Since the rational map of the surface may not be defined at some points, we construct the triangulation in a special way. The pole curve partitions the parameter domain into several regions. The rational functions of the map are continuous inside these open regions, and therefore each region maps to a possibly infinite but single-sheeted surface patch. By approximating each patch independently of the others, we avoid generating a topologically incorrect approximation.

We construct a domain triangulation whose edges do not cross the pole curve. Furthermore, each domain triangle is allowed to have up to two vertices that are on the pole curve, and up to three vertices that are base points. Each domain triangle then corresponds to a single-sheeted surface patch. The rational functions can be
sampled over the interior of each domain triangle safely, generating an approximation to the surface patch, which is then clipped against the bounding box.

We first show the steps of the algorithm, and then explain the steps in detail.

1. (RESTRICT TO FINITE DOMAIN) Perform the projective reparameterization of Chapter 4 so that the entire surface is mapped in four pieces, each over the “unit” triangle spanned by (0,0), (0,1), (1,0). Treat the four new mappings independently, and for each mapping perform the following steps.

2. (GENERATE POLE POINTS) Compute a piecewise-linear approximation to the pole curve of the current mapping inside the unit triangle.

3. (GENERATE DOMAIN POINTS) Generate points in the rest of the unit triangle according to some fixed or adaptive scheme.

4. (GENERATE BASE POINTS) Compute all the base points of the current mapping that lie inside the unit triangle.

The three kinds of points (ordinary domain points, pole points, and base points) are labeled differently.

5. (TRIANGULATE) Compute a triangulation of the points thus generated. If the edge of any triangle crosses the pole curve, insert the intersection points; if any triangle has three pole vertices, insert its midpoint.

6. (MAP TRIANGLES) Every triangle can now have up to 2 pole vertices and up to 3 base point vertices. Map each triangle onto a surface patch and clip it against the bounding box. Various types of patches result depending on the labels of a domain triangle. They are as follows:

   - All vertices are ordinary. The image of the triangle is a finite triangular patch.

   - One vertex is a pole. The image is an infinite triangular patch with one corner at infinity (Figure 5.3).
• Two vertices are poles. The image is an infinite triangular patch with two corners at infinity (Figure 5.3).

• One vertex is a base point. The base point blows up to a curve on the surface. Approaching the base point vertex along each of its incident edges leads to a different surface point on this curve. Thus, the image is a finite rectangular patch (Figure 5.4).

• Combinations of ordinary, base points, and pole points. The resulting patch can be finite or infinite, with up to six sides.

Mapping each domain triangle is accomplished by walking along its boundary and checking the vertex labels. For clipping, an iteration must be used to locate the intersection(s) of each edge of the surface patch with the bounding box. For each base point vertex a parameterization of the corresponding seam curve is necessary, taking as parameter the slope of lines through the base point.

This concludes the description of the algorithm; we now discuss some of the steps in more detail.

In step 2, the curve approximation must be sufficiently linear, otherwise there is a risk that some parts of the surface that lie inside the bounding box will be missed.

In step 3, points on the unit triangle can be generated either uniformly or adaptively spaced. Points that are uniformly spaced in the \( s \) and \( t \) directions are easily generated. For instance we can generate \( n(n + 1)/2 \) points by taking \( i + 1 \) equally spaced points on each line \( s + t = i/(n - 1), i = 0, \ldots, n - 1 \). The points can also be selected based on local surface curvature. However, the methods of [109, 67, 86] are applicable only for maps which are defined over all the domain. It would be better to apply adaptive sampling techniques in step 6, when it is known that the current triangle maps onto a single-sheeted surface.

In step 4 we must find the base points by solving the equations \( F_i = 0, i = 1, \ldots, 4 \). This could be done by picking two of the equations, finding their common solutions, and then checking whether these are solutions of the other two equations. In [27] a
method based on resultants is given for finding all base points and their multiplicities directly.

For step 5, any triangulation method [88] can be used, although some triangulations may have convenient properties.

Step 6 is complicated not only because of the many cases involved, but because we only know that the current domain triangle maps onto a single-sheeted patch. The patch can have up to six sides, and could twist in and out of the bounding box in a complicated way. The domain triangle should be further subdivided if necessary, using an adaptive domain sampling technique [109, 86, 67]. For instance, in [109] estimates of Lipschitz constants are used to decide when a portion of a surface is sufficiently linear to be approximated by a triangular facet.

Finally, base point vertices need special treatment. A domain triangle with $b$ base point vertices maps onto a patch with $b + 3$ sides. Three sides of the patch are the images of the domain triangle's three edges, and therefore tracing these sides (for clipping) is not a problem. How to trace the other $b$ sides of the patch is not obvious, since their points can't be generated by evaluating the rational map at some domain points.

Consider a triangle with a base point vertex $p$. Suppose $p$ is incident to the edges $e_1$ and $e_2$. Let the slope of the edges $e_1$ and $e_2$ be $m_1$ and $m_2$ respectively. Approaching $p$ along the line of slope $m_1$ leads to one point on the surface, and approaching it along the line of slope $m_2$ leads to another point on the surface. Both these points lie on the seam curve corresponding to the base point. Parameterize the seam curve in terms of $m$, the slope of lines through the base point. Then the side of the patch that corresponds to the base point vertex can be traced by evaluating the seam curve parameterization at values between $m_1$ and $m_2$. We discuss how to compute this parameterization in the next subsection.
5.5.3 Base Points and Seam Curves

In this section, we use projective coordinates to describe points on a rational parametric surface, for notational convenience. Thus a surface is given as follows:

\[ \rho X = X(s, t), \quad \rho Y = Y(s, t), \quad \rho Z = Z(s, t), \quad \rho W = W(s, t) \]

where \( \rho \) is a non-zero constant of proportionality (we still use an affine domain, since step 1 of the surface display algorithm allows us to restrict our attention to affine parameter points).

Then, let \( O \) be a common solution of the curves \( X = 0, \ldots, W = 0 \). Furthermore, let us suppose that \( O \) is a point of multiplicity \( q \) on each of the curves \( X = 0, \ldots, W = 0 \), and that the curves have no common tangent at \( O \). Then the image of the base point \( O \) is a rational curve of degree \( q \) on the surface [101].

In [27], a method is given to find the parametric equations of this curve. The basic idea is to pass a pencil of lines through the base point and then use the slope of these lines as a parameter, since approaching the base point from each direction leads to a different point on the seam curve. The seam curve equations are not given explicitly, but as quotients of certain polynomials. The algorithm fails when the curves \( X = 0, \ldots, W = 0 \) have common tangents at \( O \); in this case the parametric equations given by this algorithm generate only a single point of the seam curve.

In [77] a method is given for parameterizing seam curves that works for all cases (i.e., even when the tangents are equal). However, it is much more expensive than the previous method: multivariate resultants are used to compute a projection onto a plane of all the seam curves simultaneously, yielding a bivariate equation. Along with the projection, a rational map \( R \) is computed between the projection and the curves on the surface. A bivariate factorization algorithm (over the complexes) such as [8, 65] must first be applied to separate out the the projections of the individual curves. Each projected seam curve is then parameterized using a general curve parameterization technique [3], and finally mapped onto the surface using the rational map \( M \).
The method of [27] is much simpler than that of [77], and could be implemented as part of the surface display algorithm. However, we present a further simplification of [27] based on the same idea, which is found in algebraic geometry textbooks such as [101] (and hence it also fails when the tangents at $O$ are all equal). This simplification makes the method easier to implement numerically, since we find an explicit formula for the parametric equations of the seam curve. Furthermore, the formula clearly shows how the number of common tangents of $X = 0, \ldots, W = 0$ at the base point affects the seam curve, explaining why this method breaks down when the tangents at the base point are all equal.

**THEOREM 5.1** Let $(a, b)$ be a base point of multiplicity $q$. Then for any $m \in \mathbb{R}$, the image of a domain point approaching $(a, b)$ along a line of slope $m$ is given by $(X(m), Y(m), Z(m), W(m)) =\) 

\[
\left(\sum_{i=0}^{q} \left(\frac{\partial^n X}{\partial s^{q-i} \partial t^{i}}(a, b)\right) m^i, \ldots, \sum_{i=0}^{q} \left(\frac{\partial^n W}{\partial s^{q-i} \partial t^{i}}(a, b)\right) m^i\right) \tag{5.5}
\]

**PROOF.** Consider the image of a point $(s, t)$ as it approaches $(a, b)$ along the line of slope $m$ through $(a, b)$. Expressing the line as $t = m(s - a) + b$, this yields the point

\[
\lim_{s \to a} (X(s, m(s - a) + b), Y(s, m(s - a) + b), Z(s, m(s - a) + b), W(s, m(s - a) + b)) \tag{5.6}
\]

Expanding $X(s, t)$ in a Taylor series at $(a, b)$ yields

\[
X(s, t) = \sum_{k=0}^{p} \sum_{i=0}^{k} \frac{(s - a)^i (t - b)^{k-i}}{k!} \frac{\partial^k X}{\partial s^i \partial t^{k-i}}(a, b) \tag{5.7}
\]

Substituting $t = m(s - a) + b$ in (5.7) yields

\[
X(s) = \sum_{k=0}^{p} \sum_{i=0}^{k} \frac{(s - a)^k m^{k-i}}{k!} \frac{\partial^k X}{\partial s^i \partial t^{k-i}}(a, b)
\]

\[
= (s - a)^q \sum_{k=q}^{p} \sum_{i=0}^{k} \frac{(s - a)^{k-q} m^{k-i}}{k!} \frac{\partial^k X}{\partial s^i \partial t^{k-i}}(a, b)
\]

where $q$ is the multiplicity of the base point $(a, b)$, which implies that all derivatives of $X(s, t)$ up to order $q - 1$ vanish at $(a, b)$. 
Substituting $t = m(s-a)+b$ into the Taylor expansions of $Y(s,t), Z(s,t), W(s,t)$ yields $(X(s), \ldots, W(s)) = (s-a)^q \left( \sum_{k=0}^{p} \sum_{i=0}^{k} \frac{(s-a)^{k-i}m^{k-i}}{i!} \frac{\partial^k X}{\partial s^i \partial t^{k-i}}(a,b) \sum_{k=0}^{q} \sum_{i=0}^{k} \frac{(s-a)^{k-i}m^{k-i}}{i!} \frac{\partial^k W}{\partial s^i \partial t^{k-i}}(a,b) \right)$.

We drop the factor of proportionality $(s-a)^q$ and compute the limit (5.6):

$$\lim_{t \to a} (X(s), Y(s), Z(s), W(s)) = \frac{1}{q!} \left( \sum_{i=0}^{q} m^{s-i} \frac{\partial^q X}{\partial s^i \partial t^{q-i}}(a,b), \ldots, \sum_{i=0}^{q} m^{s-i} \frac{\partial^q W}{\partial s^i \partial t^{q-i}}(a,b) \right) = \left( \sum_{i=0}^{q} \left( \frac{\partial^q X}{\partial s^q \partial t^{q-i}}(a,b) \right) s^{q-i}, \ldots, \sum_{i=0}^{q} \left( \frac{\partial^q W}{\partial s^q \partial t^{q-i}}(a,b) \right) s^{q-i} \right)$$

Thus for each $m \in \mathbb{R}$ there is a corresponding point (5.5) on the parametric surface. These points collectively form a one-dimensional family or curve on the surface. □

**COROLLARY 5.1** If the curves $X(s,t) = 0, \ldots, W(s,t) = 0$ share $t$ tangent lines at $(a,b)$, then the seam curve $(X(m), Y(m), Z(m), W(m))$ has degree $q-t$. In particular, if $X(s,t) = 0, \ldots, W(s,t) = 0$ have identical tangents at $(a,b)$, then for all $m \in \mathbb{R}$ the coordinates $(X(m), \ldots, W(m))$ represent a single point.

**PROOF.** The equations of the tangent lines to the curve $X(s,t) = 0$ at $(a,b)$ are given by equating to zero the factors of the following curve, which are all linear (since it is homogeneous):

$$\sum_{i=0}^{q} \left( \frac{\partial^q X}{\partial s^q \partial t^{q-i}}(a,b) \right) s^{q-i} t^i = 0 \quad (5.8)$$

and similarly for $Y(s,t) = 0$ etc. Moreover, there is a 1-1 correspondence between the linear factors of this curve and the roots of the polynomial $X(m)$ in (5.5). Thus each common tangent of $X(s,t) = 0, \ldots, W(s,t) = 0$ at $(a,b)$ leads to a common root, and hence a common factor, among $X(m), \ldots, W(m)$. If there are $t$ common tangents there will be a common factor of degree $t$, which can be divided out of the seam curve parameterization $(X(m), \ldots, W(m))$ since proportional homogeneous coordinates represent the same point. Thus the seam curve is of degree $q-t$. □
5.5.4 Drawbacks and Future Work

The algorithm may fail to compute an approximation that accurately represents the surface, if the pole curve subdivision is not sufficiently linear. In particular, surface features inside the bounding box may be missed. The precision to which the pole curve is approximated is at present specified by the user; more research is needed to determine how to calculate this precision automatically.

Triangulations with special properties such as the Delaunay triangulation [88] could be used to speed up step 5, which is expensive. In particular, we hope to use Delaunay properties to avoid testing edges for intersection with the pole curve, that are far away from the pole curve.

In step 6, the seam curve parameterization algorithm is necessary; however, it fails when the domain curves defined by the numerators and denominator of the rational map have identical tangents at a base point. One could handle this case by using the seam curve parameterization of [77], but a simpler and cheaper technique is necessary. One approach would be to use a quadratic domain transformation to derive a new parameterization of the surface where the domain curves have separate tangents at the base points.

5.5.5 Implementation Details

The above algorithm has been implemented in the Ganith [16] system, with a few shortcuts. We now give some of the details of the implementation. Some pictures of the algorithms output on rational parametric surfaces are shown in Figures 5.9, 5.10, 5.11.

Step 1 is implemented by homogenizing the parametric form and using polynomial substitution.

In [47], several methods of approximating implicit curves are compared, with the author arguing in favor of recursive subdivision. An implementation of the subdivision-based algorithm from [47] is used for approximating the pole curve in...
step 2. This method has certain drawbacks. For instance, the algorithm subdivides the curve to the limit at its extreme points (corresponding to points where the derivative is zero), generating a proliferation of points there. If the pole curve equation happens to contain a repeated factor, then the derivative is zero everywhere along the curve component corresponding to that factor. Then the subdivision proceeds to the limit along that entire component, and the program runs very slowly. For this reason, we use the symbolic facilities of Gallith to find and divide out any repeated factors from the pole curve equation (but this is not a desirable step when floating-point coefficients are used). In its favor, the subdivision algorithm handles singularities, isolated points, and multiple curve branches. An alternative would be to use a marching technique such as [15] for tracing the pole curve.

Uniform subdivision is used to generate ordinary domain points in step 3, using a user-supplied subdivision limit to control the smoothness of the final approximation.

A point-insertion technique [108] is used in step 5 to construct the domain triangulation. While its worst case running time is quadratic in the number of points, its average running time is linear, and furthermore it is very robust.

Some shortcuts are used in the handling of base points, and hence the implementation still fails on some examples with base points. At present step 4 is omitted in the implementation; instead, each pole point is checked to see if it is a base point. Furthermore, instead of using theorem (5.1) for the seam curve, we simply generate rough approximations to seam curve points by taking the images of domain points close to the base point, along lines of desired slope.

While the implementation works well in many examples we tested, it still fails when the pole curve is not approximated sufficiently closely, or when base points are present but not detected. There is plenty of room for improvement.

5.6 Summary

In this chapter we gave algorithms for computing piecewise-linear approximations to parametric curves and surfaces using rational maps. These algorithms compute
approximations that accurately represent the shape of the curve or surface, even when
the maps had domain poles or base points (for surfaces).

The curve algorithm first intersects the curve with the edges of a bounding region.
This partitions the parameter domain into intervals that map onto curve segments
lying inside or outside the surface. The algorithm then classifies intervals as mapping
onto "in" or "out" segments, and finally approximates the "in" segments by domain
sampling.

We use a different approach in the surface algorithm, since the computation of
the "in" and "out" domain regions appears complicated for surfaces. The algorithm
instead partitions the domain by the pole curve into regions where the map is defined.
The algorithm then triangulates the domain such that domain triangles never cross
regions. Each domain triangle is mapped onto the surface and clipped against the
bounding box. If base points are present, they are inserted into the triangulation.
Triangles with base point vertices may map onto patches with more than three sides.
If this happens, the curves corresponding to the base points are parameterized, and
segments of them are extracted to construct sides of the surface patch.

We discussed an implementation of the algorithm, and showed examples of its
output.
Figure 5.1 Hyperboloid of 1 sheet with seam curve gaps
Figure 5.2 Partition of domain triangulation by pole curve
Figure 5.3 Image patches of domain triangles with pole vertices
Figure 5.4 Image patch of a domain triangle with a base point
Figure 5.5 Disjoint branches being wrongly connected (Mathematica)

Figure 5.6 Single-sheeted surface with domain poles (Maple)
Figure 5.7 Trimmed rational parametric sphere, and trim curves in domain

Figure 5.8 Trimmed hyperboloid of 2 sheets, and trim curves in domain
Figure 5.9 Rational parametric surface: cubic node

Figure 5.10 Rational parametric surface (two sheets)
Figure 5.11 Rational parametric surface (nine sheets)
6. CONCLUSIONS AND FUTURE WORK

6.1 Approximations in Exact Algebraic Algorithms

We have developed systematic techniques for using numerical approximations in some algorithms that originally used exact arithmetic. Our techniques led to efficient versions of parameterization algorithms for certain classes of rational curves and surfaces. We also analyzed the algorithms using the standard technique of backward error analysis and developed geometric characterizations of the error.

From our error analysis we conclude that our versions of the algorithms are numerically stable. That is, small perturbations in the computations due to rational approximation of algebraic numbers are reflected by small geometric perturbations in the output. Parameterization algorithms were previously held in suspicion in this regard [39], especially when singularities of the input curve/surface had to be approximated.

We briefly outline an area for future research, related to this topic.

6.1.1 Parameterization Algorithms for Other Classes

As the degree of a curve or surface increases, the chances of it being rational grow less. Thus algorithms for classes of higher degree curves and surfaces are scarce. A notable exception is [3], which can parameterize rational algebraic plane curves of any degree. The use of algebraic numbers in this algorithm is rather more complicated than in the parameterization algorithms discussed here. It would be interesting to see if the approximation methods developed here can be applied to this algorithm also. If so, it would make the seam curve parameterization algorithm of [77] more
practical, and in Chapter 5 we showed how seam curve parameterizations are useful in parametric surface display.

Algorithms exist for cubic surface parameterization [99, 2]; an error analysis should be developed for these algorithms, which also compute with algebraic numbers. However, the algebraic numbers in this case are coordinates of lines or curves on the surface, not of points. Thus the error analysis is likely to be more complicated than the ones developed here. Similarly, the cubic hypersurface parameterization algorithm of [14] is also a candidate for error analysis.

Although we have only discussed global parameterization of a curve or surface, there are techniques for parameterizing a small portion of a curve or surface around a point, using power series. This procedure can be used in generating curve and surface approximations. When the point in question is singular, such local parameterization techniques [11] require algebraic number computation. We could try to estimate the error if rational approximations to algebraic numbers are used in this algorithm.

As more exact algorithms from algebraic geometry are developed to solve problems in computer graphics and geometric modeling, we hope that our techniques can simplify the implementation of these algorithms and make them easier to analyze for error.

6.2 Display of Rational Parametric Curves and Surfaces

We have highlighted some problems in the display of arbitrary rational parametric curves and surfaces. These problems appear largely neglected in the literature, probably since they arose from mathematical visualization rather than a geometric design or modeling application.

Problems that arise in curve and surface display are: mapping an entire curve or surface using only finite parameter values, handling domain poles, and handling domain base points (for surfaces).
We showed how projective transformations in the parametric domain can be used to map an entire curve or surface using a finite number of pieces if linear transformations are used, or a single piece if quadratic transformations are used. Quadratic domain transformations yielded normal parameterizations of the natural quadrics, i.e., parameterizations that do not have any missing points.

We discussed differences between polynomial and rational parameterizations from the perspective of generating curvature-sensitive approximations, and explained why rational parameterizations behave poorly in this regard.

In Chapter 5, we used results from Chapter 4 to develop robust algorithms for approximating parametric curves and surfaces, taking care of infinite parameter values, domain poles, and domain base points. Solutions were given for both curves and surfaces. The graphical computer algebra system Ganith [16] was used for implementing the surface algorithm.

It would be interesting to experiment with other classes of parametric surfaces, e.g., those involving transcendental functions. Much of the surface algorithm of Chapter 5 will remain valid, although subprograms for approximating algebraic pole curves will need replacement, and projective reparameterizations won't apply.

We now explore two areas for future work regarding rational parametric curves and surfaces.

6.2.1 Complex Parameter Values

While the parameterization (5.2) defines a map from \( \mathbb{R}^2 \) into \( \mathbb{R}^3 \), it also defines a unique algebraic surface in \( \mathbb{C}^3 \) which can be given by a single equation in three variables, with real coefficients. This algebraic surface may contain real points which are not mapped by any real parameter values. If we want to view the entire real part of the algebraic surface defined by the map, and not just the image of \( \mathbb{R}^2 \), additional computations are needed.
For instance, consider a Steiner quartic surface (Figure 4.10) given implicitly by
\[ F(x, y, z) = x^2y^2 + y^2z^2 + x^2z^2 - 2xyz = 0, \]
or parametrically by
\[
\begin{align*}
  x(s, t) &= \frac{2s}{s^2 + t^2 + 1}, \\
  y(s, t) &= \frac{2t}{s^2 + t^2 + 1}, \\
  z(s, t) &= \frac{2st}{s^2 + t^2 + 1}.
\end{align*}
\]

Note that the \( x, y \) and \( z \) axes lie entirely on the algebraic surface \( F(x, y, z) = 0 \).

Let us consider the parametric map to see which parameter values give rise to the \( x \) axis, which is described by \( y = z = 0 \). Setting \( y(s, t) = z(s, t) = 0 \) and solving for \( s, t \) yields \( t = 0 \). Thus \((x(s,0),0,0) = (2s/(s^2 + 1),0,0), s \in \mathbb{R}, \) are the points on the \( x \) axis that are given by the map. This shows that any parameter value \( s \in \mathbb{R} \) yields a surface point \((x,0,0)\) with \( |x| \leq 1 \).

To find parameter values giving rise to the remaining surface points on the \( x \)-axis we must extend the parameter domain to \( \mathbb{C}^2 \). We now show one way to compute the complex parameter values that map onto these points.

Let the parameters \( s, t \) denote complex numbers given as \( s = a + bi, t = c + di \), where \( a, b, c, d \in \mathbb{R} \) and \( i = \sqrt{-1} \).

Then the parametric map from \( \mathbb{C}^2 \rightarrow \mathbb{R}^3 \) can be expressed as
\[
\begin{align*}
  x(s, t) &= x(a + bi, c + di) = X_R(a, b, c, d) + X_I(a, b, c, d) \cdot i, \\
  y(s, t) &= y(a + bi, c + di) = Y_R(a, b, c, d) + Y_I(a, b, c, d) \cdot i, \\
  z(s, t) &= z(a + bi, c + di) = Z_R(a, b, c, d) + Z_I(a, b, c, d) \cdot i,
\end{align*}
\]

where \( X_R \) denotes the real part of \( x(a + bi, c + di) \) and \( X_I \) denotes its imaginary part, etc.

Then \( X_I(a, b, c, d) = 0, Y_I(a, b, c, d) = 0, Z_I(a, b, c, d) = 0 \) form a system of three equations in four unknowns whose solutions give parameter values that map to real surface points. In general, such a system has a one-dimensional solution set which can be traced in 4-space by a marching method such as [15]. Note that this particular system has the trivial two-dimensional solution \( b = d = 0 \) which must be excluded.

Further work is needed along these lines to develop an efficient procedure for computing the entire real part of the algebraic surface defined by a rational map, including real points given by complex parameter values.
6.2.2 Computing Triangulations on Surfaces

The surface display algorithm of Chapter 5 generated a piecewise-linear mesh of triangles approximating a surface. This mesh was derived by constructing a planar triangulation in the domain and mapping it onto the surface. However, when a planar domain triangulation is mapped onto a curved surface, the resulting triangles in space may not possess the properties of a planar triangulation. That is, an edge of the surface triangulation may intersect a surface triangle at a point that is not one of its vertices.

There are two reasons for this. First, if the domain sampling density is not fine enough with respect to the surface curvature, two surface triangles may overlap each other. Second, if the surface actually crosses itself, some surface triangles near the crossing may cross each other.

Constructing triangulations on surfaces is useful, especially in mesh generation for finite-element analysis. Even for display, a surface triangulation is preferable. This is because scanline-rendering algorithms suffer from aliasing effects triangle intersections; this causes what should appear as a sharp edge on the screen to appear wavy.

Figure 6.1 shows a triangular mesh approximating the a Steiner quartic surface (Figure 4.10). The mesh was constructed using the surface display algorithm of Chapter 5. The surface crosses itself along the x, y and z axes, which are also drawn in Figure 6.1. In this case, the mesh is actually a surface triangulation. This is a coincidence, and happens because the four quadrants of the parameter domain happen to map onto four pieces of the surface that meet exactly along the singular lines. Since the surface display algorithm maps each of the four domain quadrants separately, the resulting triangles on the mesh also meet along the singular lines.

Suppose we apply a random linear reparameterization to get another map for the same Steiner surface, and apply the display algorithm to generate a mesh for the new map. In general, the new mesh will not be a surface triangulation.
We propose a way to construct a surface triangulation on a parametric surface that crosses itself, i.e., on a singular parametric surface. The idea is to compute the curves and points in the parametric domain that map onto singular points on the surface, and then partition the domain by these curves (as well as by the pole curves). If this is done, no domain triangle will overlap domain points that map onto a surface singularity. Hence triangles on the surface will meet only along their edges or at their vertices, even if the surface is singular.

The points and curves in the real parameter domain that map onto surface singularities can be found by formulating a system of polynomial equations, extending the procedure of [4] for singularities of parametric plane curves.

For example, consider the surface given by the the following equations, taking $x(s,t) = X(s,t)/W(s,t)$, etc.

\[
\begin{align*}
X(s,t) & = s^3 + st^2 - 3s \\
Y(s,t) & = (s^2 + t^2)^2 - 3(s^2 + t^2) \\
Z(s,t) & = s^2t + t^3 - 3t \\
W(s,t) & = (s^2 + t^2)^2 + 2(s^2 + t^2) + 1
\end{align*}
\]

This is a surface of revolution (see Figure 6.2); it has a point singularity at the origin. It can be shown that the circle of radius $\sqrt{3}$ in the parameter domain, and the point $(0,0)$ in the parameter domain both map onto this singular point.

In this case we found the image of the surface singularity by hand. Further work is needed to find efficient procedures to solve this problem for surfaces with both point singularities (Figure 6.2) and curve singularities (Figure 6.1).
Figure 6.1  Triangulation of a Steiner surface along curve singularities

Figure 6.2  Triangulation of parametric surface with point singularity
BIBLIOGRAPHY


[87] Peters, J. (private communication)


VITA
VITA

Andrew Vasanth Royappa was born in Madurai, India on October 7, 1966. In 1985 he graduated from Purdue University in West Lafayette, Indiana with a Bachelor of Science degree in Computer Science. He entered graduate school at Purdue University in 1985 and began his doctoral work in 1987 under Professor Chandrakirti Bajaj. At present his research interests include computer graphics and computer aided geometric design. Throughout his doctoral work, he was supported by research assistantships from Professor Bajaj.