High-order adaptive methods for computing invariant manifolds of maps

Jacek K. Wrobel
New Jersey Institute of Technology

Follow this and additional works at: https://digitalcommons.njit.edu/dissertations

Part of the Mathematics Commons

Recommended Citation
https://digitalcommons.njit.edu/dissertations/267

This Dissertation is brought to you for free and open access by the Electronic Theses and Dissertations at Digital Commons @ NJIT. It has been accepted for inclusion in Dissertations by an authorized administrator of Digital Commons @ NJIT. For more information, please contact digitalcommons@njit.edu.
Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be “used for any purpose other than private study, scholarship, or research.” If a user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of “fair use” that user may be liable for copyright infringement.

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation.

Printing note: If you do not wish to print this page, then select “Pages from: first page # to: last page #” on the print dialog screen.
The Van Houten library has removed some of the personal information and all signatures from the approval page and biographical sketches of theses and dissertations in order to protect the identity of NJIT graduates and faculty.
ABSTRACT

HIGH-ORDER ADAPTIVE METHODS FOR COMPUTING INVARIANT MANIFOLDS OF MAPS

by

Jacek K. Wróbel

The author presents efficient and accurate numerical methods for computing invariant manifolds of maps which arise in the study of dynamical systems. In order to decrease the number of points needed to compute a given curve/surface, he proposes using higher-order interpolation/approximation techniques from geometric modeling. He uses Bézier curves/triangles, fundamental objects in curve/surface design, to create adaptive methods. The methods are based on tolerance conditions derived from properties of Bézier curves/triangles. The author develops and tests the methods for an ordinary parametric curve; then he adapts these methods to invariant manifolds of planar maps. Next, he develops and tests the method for parametric surfaces and then he adapts this method to invariant manifolds of three-dimensional maps.
HIGH-ORDER ADAPTIVE METHODS FOR COMPUTING IN Variant MANIFOLDS OF MAPS

by
Jacek K. Wróbel

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Mathematical Sciences

Department of Mathematical Sciences, NJIT

May 2011
HIGH-ORDER ADAPTIVE METHODS FOR COMPUTING INVARIANT MANIFOLDS OF MAPS

Jacek K. Wróbel

Roy H. Goodman, Dissertation Advisor
Associate Professor, Department of Mathematical Sciences, NJIT

Denis L. Blackmore, Committee Member
Professor, Department of Mathematical Sciences, NJIT

Amitabha K. Bose, Committee Member
Professor, Department of Mathematical Sciences, NJIT

Richard O. Moore, Committee Member
Associate Professor, Department of Mathematical Sciences, NJIT

Lee Mosher, Committee Member
Professor, Department of Mathematics & Computer Science, Rutgers University
BIOGRAPHICAL SKETCH

Author: Jacek K. Wróbel
Degree: Doctor of Philosophy
Date: May 2011

Undergraduate and Graduate Education:

- Doctor of Philosophy in Mathematical Sciences, New Jersey Institute of Technology, Newark, NJ, 2011
- Master of Science in Mathematics, AGH University of Science and Technology, Kraków, Poland, 2005

Major: Mathematical Sciences

Presentations and Publications:


To My Parents
ACKNOWLEDGMENT

First I would like to express my deepest appreciation to my dissertation advisor, Professor Roy H. Goodman, for his valuable guidance, support, encouragement, kindness and enthusiasm during the course of this work and my entire graduate study at NJIT. I am especially grateful for his inexhaustible patience. I have been greatly influenced by his organized and disciplined style of work. He has made a significant difference in my life and I will never forget it.

I am very grateful to Prof. Goodman and his wife, Sarah Berger, for their hospitality during my visit at the Technion-Israel Institute of Technology.

I wish to thank the committee members, Profs. Denis L. Blackmore, Amitabha K. Bose, and Richard O. Moore of the Department of Mathematical Sciences, NJIT, and Prof. Lee Mosher of the Department of Mathematics & Computer Science, Rutgers University.

For helpful discussions I would like to thank Denis Blackmore, Josh Carter, Gershon Elber, Martin Hering-Bertram, Rafael de la Llave, Hector Lomelí, Jason Mireles-James, James Meiss, Hinke Ozinga, and Denis Zorin. NJIT undergraduates Casayndra Basarab, Fatima Elgammal, Matthew Peragine and Priyanka Shah collaborated on a preliminary version of this work as part of the CSUMS program, supported by NSF DMS-0639270. The author received support from NSF DMS-0807284.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Iterated Maps</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Background</td>
<td>3</td>
</tr>
<tr>
<td>2 HIGH-ORDER ADAPTIVE METHOD FOR COMPUTING ONE-DIMENSIONAL INVARIANT MANIFOLDS OF MAPS</td>
<td>8</td>
</tr>
<tr>
<td>2.1 Model Curves</td>
<td>8</td>
</tr>
<tr>
<td>2.1.1 Parametric curves</td>
<td>8</td>
</tr>
<tr>
<td>2.1.2 Parameterization</td>
<td>10</td>
</tr>
<tr>
<td>2.2 2D Geometric Modeling Tools</td>
<td>10</td>
</tr>
<tr>
<td>2.2.1 Piecewise Linear Interpolation</td>
<td>10</td>
</tr>
<tr>
<td>2.2.2 Bézier Curves</td>
<td>11</td>
</tr>
<tr>
<td>2.2.3 Hermite Interpolating Polynomials</td>
<td>16</td>
</tr>
<tr>
<td>2.2.4 Catmull-Rom Splines</td>
<td>17</td>
</tr>
<tr>
<td>2.2.5 B-splines</td>
<td>20</td>
</tr>
<tr>
<td>2.2.6 Quasi-Interpolation Schemes</td>
<td>23</td>
</tr>
<tr>
<td>2.3 Adaptive Methods</td>
<td>25</td>
</tr>
<tr>
<td>2.3.1 Existing Methods</td>
<td>25</td>
</tr>
<tr>
<td>2.3.2 Proposed Methods</td>
<td>31</td>
</tr>
<tr>
<td>2.4 Numerical Tests of the Proposed Tools</td>
<td>35</td>
</tr>
<tr>
<td>2.4.1 A Model Curve</td>
<td>35</td>
</tr>
<tr>
<td>2.4.2 A Visual Test and Motivation for Improved Methods</td>
<td>36</td>
</tr>
<tr>
<td>2.4.3 Quantitative Comparison of Methods</td>
<td>38</td>
</tr>
<tr>
<td>2.5 Details of the Numerical Implementation: One-dimensional Invariant Manifold Calculation</td>
<td>40</td>
</tr>
<tr>
<td>2.5.1 Inductive Parameterization</td>
<td>40</td>
</tr>
<tr>
<td>2.5.2 Notation</td>
<td>41</td>
</tr>
</tbody>
</table>
# TABLE OF CONTENTS

(Continued)

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5.3</td>
<td>The Initial Primary Segment</td>
</tr>
<tr>
<td>2.5.4</td>
<td>Resolving a Simple Primary Segment</td>
</tr>
<tr>
<td>2.5.5</td>
<td>Kink Patching</td>
</tr>
<tr>
<td>2.6</td>
<td>Numerical Tests</td>
</tr>
<tr>
<td>2.6.1</td>
<td>Example 1: Hénon Map</td>
</tr>
<tr>
<td>2.6.2</td>
<td>Example 2: McMillan Map</td>
</tr>
<tr>
<td>2.6.3</td>
<td>More Direct Convergence Tests</td>
</tr>
<tr>
<td>2.6.4</td>
<td>Example 3: Map with an Explicit Manifold</td>
</tr>
<tr>
<td>2.7</td>
<td>Discussion</td>
</tr>
<tr>
<td>3</td>
<td>HIGH-ORDER ADAPTIVE METHOD FOR COMPUTING TWO-DIMENSIONAL INVARIANT MANIFOLDS OF MAPS</td>
</tr>
<tr>
<td>3.1</td>
<td>Existing Methods</td>
</tr>
<tr>
<td>3.1.1</td>
<td>Parameterization Method</td>
</tr>
<tr>
<td>3.1.2</td>
<td>The Krauskopf-Osinga Method</td>
</tr>
<tr>
<td>3.2</td>
<td>3D Geometric Modeling Tools</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Parametric Surface</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Triangulation</td>
</tr>
<tr>
<td>3.2.3</td>
<td>Barycentric Coordinates</td>
</tr>
<tr>
<td>3.2.4</td>
<td>Bernstein Polynomials</td>
</tr>
<tr>
<td>3.2.5</td>
<td>Triangular Bernstein-Bézier Patches</td>
</tr>
<tr>
<td>3.2.6</td>
<td>Surface Normal Estimation</td>
</tr>
<tr>
<td>3.3</td>
<td>Triangulation Based Surface Approximation Schemes</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Bivariate Linear Interpolation</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Nine-parameter Interpolant</td>
</tr>
<tr>
<td>3.3.3</td>
<td>The Clough-Tocher Method</td>
</tr>
<tr>
<td>3.3.4</td>
<td>The Shirman-Sequin Method</td>
</tr>
</tbody>
</table>
TABLE OF CONTENTS
(Continued)

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3.5 Quasi-Interpolation based on Quartic Bernstein-Bézier Patches</td>
<td>86</td>
</tr>
<tr>
<td>3.3.6 Adaptive Quasi-Interpolating Quartic Splines</td>
<td>92</td>
</tr>
<tr>
<td>3.4 Numerical Tests of Proposed Tools</td>
<td>96</td>
</tr>
<tr>
<td>3.4.1 Model Surfaces</td>
<td>96</td>
</tr>
<tr>
<td>3.4.2 Comparison Tests of Methods</td>
<td>98</td>
</tr>
<tr>
<td>3.4.3 Details of the Adaptive Quasi-Interpolating Quartic Splines</td>
<td>100</td>
</tr>
<tr>
<td>3.5 Details of the Numerical Implementation: Two-dimensional Invariant</td>
<td>106</td>
</tr>
<tr>
<td>Manifold Calculation</td>
<td></td>
</tr>
<tr>
<td>3.5.1 Parameterization</td>
<td>106</td>
</tr>
<tr>
<td>3.5.2 Notation</td>
<td>107</td>
</tr>
<tr>
<td>3.5.3 Initial Primary Annulus</td>
<td>108</td>
</tr>
<tr>
<td>3.5.4 Resolving a Primary Annulus</td>
<td>111</td>
</tr>
<tr>
<td>3.5.5 Joint Patching</td>
<td>112</td>
</tr>
<tr>
<td>3.5.6 Distance Control along the Manifold</td>
<td>113</td>
</tr>
<tr>
<td>3.6 Numerical Tests</td>
<td>113</td>
</tr>
<tr>
<td>3.6.1 Example 1: Volume-Preserving Hénon Map</td>
<td>114</td>
</tr>
<tr>
<td>3.6.2 Example 2: Arneodo-Coullet-Tresser Map</td>
<td>118</td>
</tr>
<tr>
<td>3.7 Discussion</td>
<td>121</td>
</tr>
<tr>
<td>4 SUMMARY AND FUTURE WORK</td>
<td>123</td>
</tr>
<tr>
<td>4.1 Recap</td>
<td>123</td>
</tr>
<tr>
<td>4.2 Improvement of the Two-dimensional Method</td>
<td>126</td>
</tr>
<tr>
<td>4.3 Application of the Adaptive Methods</td>
<td>126</td>
</tr>
<tr>
<td>4.4 Invariant Manifolds of Hyperbolic Tori</td>
<td>127</td>
</tr>
<tr>
<td>APPENDIX SOFTWARE</td>
<td>129</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>130</td>
</tr>
</tbody>
</table>
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Manifold calculation comparison using chord arc length of primary segments of the McMillan Map. The bold type values we can call the “exact” lengths of the segments. Values for Hobson’s and Carter’s method follows [4]. Values for <strong>ALI+</strong>, <strong>ACR3+</strong>, <strong>ACR3vs5</strong> are computed for 10 points at the initial primary segment.</td>
</tr>
<tr>
<td>2.2</td>
<td>Manifold calculation comparison using Gravesen’s arc length approximation of primary segments of McMillan Map. The values are computed for given value of the tolerance with 10 points at the initial primary segment.</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>(a) Schematic of stable/unstable manifolds (red/blue). (b) Stable/unstable</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>manifolds (red/blue) of a fixed point computed using the algorithm of Carter</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[4].</td>
<td></td>
</tr>
<tr>
<td>1.2</td>
<td>(Schematic) Union of primary segments forming a portion of the unstable</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>manifold.</td>
<td></td>
</tr>
<tr>
<td>1.3</td>
<td>(Schematic) Right: Two and one-dimensional manifolds of a hyperbolic fixed</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>point and the annular portion of the manifold between curves $\Gamma$ and $\Gamma'$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>on the right. Left: The annular domains for a parametrization method.</td>
<td></td>
</tr>
<tr>
<td>2.1</td>
<td>A cubic Bézier curve, together with its control points $p_0, p_1, p_2,$ and</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>their convex hull.</td>
<td></td>
</tr>
<tr>
<td>2.2</td>
<td>The de Casteljau Algorithm for evaluating/subdividing a cubic Bézier curve.</td>
<td>14</td>
</tr>
<tr>
<td>2.3</td>
<td>Degree elevation: both polygons define the same cubic curve.</td>
<td>14</td>
</tr>
<tr>
<td>2.4</td>
<td>Composite of two Bézier curves.</td>
<td>15</td>
</tr>
<tr>
<td>2.5</td>
<td>Cubic B-spline generated by the knot vector $[t_0,\ldots,t_7]$.</td>
<td>21</td>
</tr>
<tr>
<td>2.6</td>
<td>The de Boor construction algorithm for a point on a segment of a cubic B-spline.</td>
<td>23</td>
</tr>
<tr>
<td>2.7</td>
<td>The angle between consecutive segments in a discrete curve.</td>
<td>26</td>
</tr>
<tr>
<td>2.8</td>
<td>A situation where part of the curve might be missed.</td>
<td>27</td>
</tr>
<tr>
<td>2.9</td>
<td>The angle between consecutive linear segments and the bisection algorithm.</td>
<td>28</td>
</tr>
<tr>
<td>2.10</td>
<td>The control points and several characteristic quantities of a Bézier curve.</td>
<td>31</td>
</tr>
<tr>
<td>2.11</td>
<td>Two different Catmull-Rom approximations, $p(t)$ and $\tilde{p}(t)$, of the</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>curve $\gamma(t)$.</td>
<td></td>
</tr>
<tr>
<td>2.12</td>
<td>The test curve for $\epsilon = 10^{-2}$, $0 \leq t \leq 15$.</td>
<td>35</td>
</tr>
<tr>
<td>2.13</td>
<td>The exact curve (red dashed) and two different spline interpolants on the</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>knots from the adaptive Catmull-Rom method, part of the $\gamma_{\text{test}}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>curve with $\epsilon = 10^{-2}$.</td>
<td></td>
</tr>
<tr>
<td>2.14</td>
<td>Test on the model curve (2.16) with $\epsilon = 10^{-2}$, $0 \leq t \leq 15$,</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>(a) the error of the Adaptive Catmull-Rom methods, ACR3 green and ACR5 red,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(b) the difference between the ACR3 and ACR5 approximations.</td>
<td></td>
</tr>
</tbody>
</table>
2.15 (a) Convergence of the error for the adaptive methods with different refinement conditions changing as indicated, between the values at the opposite ends of each graph. The methods tested on the test curve with $0 \leq t \leq 10$ and $\epsilon = 10^{-3}$. Comparison of the ALI+ method (b), the ACR3+ method (c) and the ACR3vs5 method (d) on one sharply pointed segment of the curve for $t \in (7.57, 7.82)$, the error approximately at the same $10^{-8}$ level indicated by boxes in subfigure (a). The curvature at this point is about $1.1 \times 10^5$. ......................................................... 39

2.16 (a) The unstable manifold of the Henon Map, $b = 0.3$ and $a = 1.4$ with length about 419.6268, generated by the ACR3vs5 method; (b) box $A$; (c) box $B$ with with three approximations, $\text{tol} = 2^{-12}$–blue, $\text{tol} = 2^{-14}$–green, $\text{tol} = 2^{-16}$–red. Note subfigures (b) and (c) do not preserve the aspect ratio nor orientation of subfigure (a). In (b) $\text{height} = O(10^{-5})$, $\text{width} = O(10^{-4})$ and similarly for (c), $\text{height} = O(10^{-6})$, $\text{width} = O(10^{-4})$. ................. 47

2.17 (a) The unstable manifold of the McMillan Map, $\mu = 2.0$ and $\varepsilon = 0.05$ with length about 113.3335, generated by ACR3vs5; (b) zoom $A$; (c) zoom $B$ with with three approximations, $\text{tol} = 2^{-11}$–blue, $\text{tol} = 2^{-13}$–green, $\text{tol} = 2^{-15}$–red. Note (b) and (c) do not preserve the aspect ratio of (a), for (b) $\text{height} = O(10^{-3})$, $\text{width} = O(10^{-5})$ and for (c) $\text{height} = O(10^{-4})$, $\text{width} = O(10^{-4})$. 49

2.18 Convergence of the adaptive methods ACR3vs5 (blue), ACR3+ (red) and ALI+ (green) with decreasing refinement conditions changing as indicated, between the values at the opposite ends of each graph; 50 points on the initial primary segment. ............................................................... 51

2.19 (a) The explicit unstable manifold of the system (2.18) given by (2.20), (b) convergence of the error between the manifold and approximations given by adaptive methods ACR3vs5 (red), ACR3+ (green) and ALI+ (blue). The test is performed on the branch with length about 255, the initial primary segment starts at $x = 0.0011$ with 10 points and computes the first 11 primary segments. The decreasing refinement conditions are as indicated, between the values at the opposite ends of each graph. ..................................................... 55

3.1 (a) Initial setup for the Krauskopf-Osinga method, (b) the cases when the method fails, geodesic circles – black, primary fundamental annuli – red. 62

3.2 Delaunay triangulation in the plane with circumcircles shown. ............... 65

3.3 (a) Type-1 triangulation with one hexagon bolded, (b) The six triangles in $\Delta$ attached to $v_{i,j}$. ......................................................... 67

3.4 Cubic Bézier triangle with the control net and the domain triangle. ........ 69
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.5</td>
<td>The de Casteljau algorithm: the original cubic control net $b^0$ (black) together with the quadratic $b^1$ (red) and linear $b^2$ (blue) nets that determine the resulting point. All intermediate control points are generated in the ‘upright’ subtriangles. The blue linear net determines the tangent plane to the patch at the resulting point. The resulting subdivision into three smaller Bézier patches on right.</td>
</tr>
<tr>
<td>3.6</td>
<td>Two steps of 3-to-1 subdivision over a Bézier triangle.</td>
</tr>
<tr>
<td>3.7</td>
<td>Three steps of the 2-to-1 Subdivision over a Bézier triangle.</td>
</tr>
<tr>
<td>3.8</td>
<td>Schematic 4-to-1 Subdivision.</td>
</tr>
<tr>
<td>3.9</td>
<td>Steps of the 4-to-1 Subdivision of a Bézier triangle.</td>
</tr>
<tr>
<td>3.10</td>
<td>Schematic representation of a three-split of a cubic Bézier triangle; three mini-triangles and its 19 Bézier ordinates (control points).</td>
</tr>
<tr>
<td>3.11</td>
<td>Adjacent $C^1$ triangular patches defined over mini-triangles.</td>
</tr>
<tr>
<td>3.12</td>
<td>Schematic representation of the Shirman-Sequin three-split method.</td>
</tr>
<tr>
<td>3.13</td>
<td>Control points for the Shirman-Sequin method.</td>
</tr>
<tr>
<td>3.14</td>
<td>Relevant Bézier points of the two patches used by Chiyokura and Kimura’s construction. Note that construction uses both cubic and quartic control points.</td>
</tr>
<tr>
<td>3.15</td>
<td>Grids for Quartic Quasi-Interpolation Scheme.</td>
</tr>
<tr>
<td>3.16</td>
<td>Evaluation of the Bézier coefficient associated with (a) point $u$; (b) point $e$ shown by the open dots.</td>
</tr>
<tr>
<td>3.17</td>
<td>Evaluation of the Bézier coefficient associated with $z$ shown by the open dot.</td>
</tr>
<tr>
<td>3.18</td>
<td>A data set required by control points of a single Bézier triangle; data points, filled dots, on the left and resulting control points, open dots, on the right.</td>
</tr>
<tr>
<td>3.19</td>
<td>Bézier patches of different resolution, one level difference.</td>
</tr>
<tr>
<td>3.20</td>
<td>General and fully periodic parametric surfaces generated by the quasi-interpolation method described in Subsection 3.3.5; (a) plate surface given by the Equation (3.13), $r = 10$, $(u, v) \in (0, 2\pi) \times (0, 6)$ and sampled on the $48 \times 48$ grid; (b) pretzel given by the Equation (3.14), $a = .6$, $n = 1.5$, $r_1 = .4$, $r_2 = .4$, $(u, v) \in (0, 4\pi) \times (-2\pi, 2\pi)$ and sampled on the $18 \times 54$ grid.</td>
</tr>
<tr>
<td>3.21</td>
<td>Reconstruction of a torus given by Equation (3.15) using different schemes on the $15 \times 6$ uniform parametric data grid, $a = 2$, $r = 1$.</td>
</tr>
<tr>
<td>Figure</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>3.22</td>
<td>100</td>
</tr>
<tr>
<td>An annular test surface generated by the quasi-interpolation method described in Subsection 3.3.5; (a) plait surface given by the Equation (3.16), (a = 3, b = 2, (u, v) \in (0, 4) \times (0, 2\pi)) and sampled on the 9 \times 18 grid; (b) convergence introduced methods tested on the “plait”.</td>
<td></td>
</tr>
<tr>
<td>3.23</td>
<td>101</td>
</tr>
<tr>
<td>Points grid in a parametric space for Quartic Quasi-Interpolation Scheme – a surface of an annular topology periodic in the vertical direction; (a) domain of the data points, (b) domain of the generated control points, an approximated surface over the shaded region.</td>
<td></td>
</tr>
<tr>
<td>3.24</td>
<td>103</td>
</tr>
<tr>
<td>Cubic spline data extrapolation outside the domain (\Omega), given data points marked by the filled dots, extrapolated data by open dots; a schematic two-dimensional view.</td>
<td></td>
</tr>
<tr>
<td>3.25</td>
<td>105</td>
</tr>
<tr>
<td>General, annular and fully periodic parametric surfaces generated by the adaptive method described in Subsection 3.3.6, (\text{tol} = 10^{-2}, \text{tol} = 10^{-3}) and (\text{tol} = 10^{-3}), respectively; (a) plate surface given by the Equation (3.13), (r = 10, (u, v) \in (0, 2\pi) \times (0, 0.6)); (b) plait surface given by the Equation (3.16), (a = 3, b = 2, (u, v) \in (0, 4) \times (0, 2\pi)); (c) pretzel given by the Equation (3.14), (a = .6, n = 1.5, r_1 = .4, r_2 = .4, (u, v) \in (0, 4\pi) \times (−2\pi, 2\pi)).</td>
<td></td>
</tr>
<tr>
<td>3.26</td>
<td>107</td>
</tr>
<tr>
<td>Composite of primary annuli with the initial primary annulus (U_0) shaded, (a) for a real pair of eigenvalues with an anisotropic growth, (b) for a complex pair of eigenvalues with more an isotropic growth.</td>
<td></td>
</tr>
<tr>
<td>3.27</td>
<td>110</td>
</tr>
<tr>
<td>Schematic settings for the initial primary annulus (U_0), (a) five primary annuli in a physical space, dash line – the edges over a parallelogram grid with (\theta_0), solid bars – edges over a rectangular grid, (b) the triangulation of the parallelogram grid in the parametric space, the buffer dash triangles in the first and the last columns, (c) four primary annuli in a physical space, rotation of a primary annulus about (\theta_0) under the map (f) with a multiplicity (m = 3) and a correction (\omega_0); (N_T = 12) then every mapping the rotation about four triangles.</td>
<td></td>
</tr>
<tr>
<td>3.28</td>
<td>116</td>
</tr>
<tr>
<td>Two-dimensional stable and unstable manifolds of the Hénon map given by the Equation (3.18) for (a = .44, b = .21, c = .35, \alpha = -.25) and (\tau = -.3), eigenvalues s.t. (</td>
<td>\lambda_s</td>
</tr>
</tbody>
</table>
### LIST OF FIGURES
(Continued)

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.29 Two-dimensional stable and unstable manifolds of the Volume-Preserving Hénon map given by the Equation (3.18); (a) $a = c = .5$, $b = 0$, $\alpha = 0$ and $\tau = -.3$, eigenvalues s.t. $</td>
<td>\lambda_s</td>
</tr>
<tr>
<td>3.30 (a) Two-dimensional unstable manifold of the ACT map given by (3.19) with $a = .2$, $b = .5$, $c = .5$, $d = 1$, $e = 1$ $k = 2$ and $\omega = 4$ associated with nontrivial fixed point, eigenvalue s.t. $</td>
<td>\lambda_u</td>
</tr>
</tbody>
</table>
1.1 Iterated Maps

Iterated maps are ubiquitous in the study of dynamics, arising either as models of physical, biological, or economic systems themselves or as reductions of continuous-time dynamics, e.g., as Poincaré maps. Of fundamental importance in understanding these dynamical systems are invariant manifolds (stable or unstable) emanating from fixed points and periodic orbits. These manifolds act as barriers between different regions of the phase space and exert a significant influence on the dynamics through their topology. Except in some rare cases, such a manifold can not be expressed as a closed form parametric curve or surface, nor as the level set of some function, and therefore must be approximated numerically.

A one-dimensional invariant manifold of a two-dimensional map typically forms a so-called “tangle” in which the manifolds double back on themselves repeatedly, with segments of very high curvature connected to other segments where the curvature is more modest. The extreme stretching and folding of these curves gives rise to chaotic dynamics. For the well-known Hénon map [29], a dissipative and chaotic dynamical system, the strange attractor is equal to the closure of $W^u$, so constructing it provides a way of approximating the strange attractor and thus understanding the long-time behavior of the system. Similar reasoning applies to other dissipative chaotic systems possessing an attractor.

Two examples of one-dimensional invariant manifolds are shown in Figure 1.1. Such a manifold is generally infinite in extent, so any computation will approximate only a finite portion. In Figure 1.1a, we consider two regions defined as the areas above and below the curve $apb$. In a model of fluid mixing the four “lobes” $L_1, \ldots, L_4$, are known
as turnstiles, and can be used to quantify how fluid moves between the upper and lower regions, according to the theory of phase-space transport due to Rom-Kedar [51, 52]; see also the work by Meiss [42]. More recent work by Collins, and by Mitchell and Delos has shown how to use the information contained in the intersections of the stable and unstable manifolds to construct symbolic dynamics and obtain a fuller topological understanding of these systems [8, 46, 47]. Figure 1.1b shows the stable and unstable manifolds for the map defined by Goodman in [19] and demonstrates the intricacy of such curves in an area-preserving example.

![Figure 1.1](image.png)

**Figure 1.1** (a) Schematic of stable/unstable manifolds (red/blue). (b) Stable/unstable manifolds (red/blue) of a fixed point computed using the algorithm of Carter [4].

Next, we consider a family of three-dimensional volume-preserving maps. Such maps are useful in understanding the motion of passive tracers in fluids and magnetic field line configurations. They are also of interest since many phenomena in the two-dimensional case are not yet completely understood in higher dimensions. Such phenomena include transport, the breakup of heteroclinic connections, and the existence of invariant tori. These maps are also important as integrators for incompressible flows. The methods work equally well for conservative and dissipative systems and they are tools that can help understand either.
This dissertation is organized as follows: The remainder of this chapter introduces the basic terminology and the objects for which we develop approximation algorithms. Chapter 2 will focus on one-dimensional manifolds of two-dimensional maps and Chapter 3 on two-dimensional manifolds of three-dimensional maps. Chapter 4 provides a concluding discussion and points to directions for further research. Each part begins with an introduction to the CAGD tools used to construct our numerical methods, describes existing methods, introduces the methods we have constructed and presents details of their implementation and numerical tests.

The algorithms for computing the one-dimensional manifolds are adaptive in the sense that more points are placed in regions of high curvature than in those with modest curvature. Our tests find that existing methods place far more points than needed in these high-curvature regions, and this leads to slower than optimal convergence, and, in some cases, failure of the method to terminate. We test some of these methods against our new ones, using more stringent tests than we have seen previously applied, and find significant improvement.

1.2 Background

A discrete-time iterated map is a dynamical system $x_{j+1} = f(x_j)$ where, for simplicity, we assume $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is diffeomorphism, and as smooth as we need. For the present work, we set $n = 2, 3$. When considering the system simply as a diffeomorphism and not as a dynamical system, we write $x' = f(x)$. We assume that $f$ has a hyperbolic fixed point $x^*$, i.e., $f(x^*) = x^*$ such that the linearized matrix $Df(x^*)$ has:

- In the 2D case, two real eigenvalues $\lambda_s$ and $\lambda_u$ satisfying $0 < |\lambda_s| < 1 < |\lambda_u|$. For simplicity, we assume both are positive. This assures that the map is orientation-preserving and has one-dimensional stable and unstable manifolds of $x^*$ in $\mathbb{R}^2$, invariant under $f$. 

In the 3D case, three eigenvalues $\lambda_{s1}$, $\lambda_{s2}$ and $\lambda_u$ satisfying $0 < |\lambda_{s1}|, |\lambda_{s2}| < 1 < |\lambda_u|$ or three eigenvalues $\lambda_s$ and $\lambda_{u1}$, $\lambda_{u2}$ satisfying $0 < |\lambda_s| < 1 < |\lambda_{u1}|, |\lambda_{u2}|$. These assumptions assure the existence of two-dimensional stable and one-dimensional unstable manifolds or two-dimensional unstable and one-dimensional stable manifolds of $x^*$ in $\mathbb{R}^3$ which are invariant with respect to the map $f$; see Figure 1.3.

The stable manifold,

$$W^s(x^*) = \{x | \exists \mu > 0, c > 0 \text{ s.t. } \forall k \geq 0, \|f^k(x) - x^*\| < ce^{-\mu k}\}$$

is defined as the set of points which approach $x^*$ at an exponential rate in forward iterates of the map. The manifold is tangent to the stable eigenspace of the linearized system at $x^*$ and its global extension can be derived by applying the inverse mapping $f^{-1}$ to the local piece. The unstable manifold,

$$W^u(x^*) = \{x | \exists \mu > 0, c > 0 \text{ s.t. } \forall k \leq 0, \|f^k(x) - x^*\| < ce^{-\mu k}\}$$

is defined as the set of points which approach $x^*$ at an exponential rate in backward iterates of the map. The assumptions on the eigenvalues ensure this convergence is exponential. The manifold is tangent to the unstable eigenspace of the linearized system at $x^*$ and its global extension can be derived by applying the forward mapping to a local piece.

Figure 1.2 (Schematic) Union of primary segments forming a portion of the unstable manifold.
Most numerical methods constructed to compute invariant manifolds use the same basic idea: the global structure (of individual branches for a 1D manifold) of the unstable manifold is found by repeatedly applying the mapping to an existing portion of the manifold. The stable manifold can similarly be found by iterating $f^{-1}$. For two-dimensional maps, there also exist methods for computing $W^s$ when $f^{-1}$ is not available in closed form [33].

In the case of two-dimensional maps, consider two points $x, y \in W^u$. Let $W^u[x, y]$ denote the closed segment of $W^u$ connecting $x$ to $y$. For any given point $x_0 \in W^u$, the set of its images $\{x_k = f^k(x_0) : k \in \mathbb{Z}\}$ partitions $W^u$ into a family of finite curve segments disjoint except for their endpoints. We refer to the closed connected component of $W^u$ between $x_k$ and $x_{k+1}$ as the $k$th primary segment $U_k$, i.e.,

$$U_k = W^u[x_k, f(x_k)].$$

A single branch $U$ of the unstable manifold associated with the fixed point $x^*$ can be constructed as the union of the primary segments $U_k$; see Figure 1.2. Note that the same construction is valid for a one-dimensional unstable manifold of a three-dimensional map.

![Figure 1.3](image_url) (Schematic) Right: Two and one-dimensional manifolds of a hyperbolic fixed point and the annular portion of the manifold between curves $\Gamma$ and $\Gamma'$ on the right. Left: The annular domains for a parametrization method.

For a three-dimensional map, consider two simple, closed curves $\Gamma, \Gamma' \subset W^u$, which enclose the fixed point without intersecting each other. Let $W^u[\Gamma, \Gamma']$ denote the closed area of $W^u$ with $\Gamma$ and $\Gamma'$ at the boundary; see Figure 1.3. For any given simple, closed
curve $\Gamma_0 \in W^u$ that encloses $x^*$, the set of its images $\{\Gamma_k = f^k(\Gamma_0) : k \in \mathbb{Z}\}$ partition $W^u$ into a family of finite annular regions disjoint except for their boundary curves. We refer to the closed connected component of $W^u$ between $\Gamma_k$ and $\Gamma_{k+1}$ as the $k$th primary annulus $U_k$, i.e.,

$$U_k = W^u[\Gamma_k, f(\Gamma_k)].$$

The unstable manifold $W^u$ associated with the fixed point $x^*$ can be constructed as the union of the primary annuli $U_k$.

Therefore, in general we can write

$$U = \bigcup_{k=-\infty}^{\infty} U_n,$$

where, for a one-dimensional unstable manifold, the branch is determined by the choice of the initial primary segment $U_0$. We refer to the initial primary segment or annulus $U_0$ as a fundamental domain that is obtained by local analysis and it is usually taken very close to $x^*$. The small portion of $W^u$ between $x^*$ and the fundamental domain is represented by some local method (see Subsection 2.3.1.4 or Subsection 3.1.1) or it can be generated by backward iterates.

The problem of computing a finite portion of an unstable manifold of a two-dimensional map can be reduced to that of simply computing a parametric curve in the following manner. Given an already computed segment $U_n$ that has been endowed with parameterization $U_k = U_k(t), t \in [a, b]$, then the next segment is simply

$$U_{k+1}(t) = f(U_k(t)),$$

so that $U_{k+1}(t)$ is a parametric curve depending on the same parameter. We further discuss how to choose the parameterization in Subsection 2.5.1.

Similarly, the problem of computing a finite portion of an unstable manifold of a three-dimensional can be reduced to that of simply computing a parametric surface in
the following manner. Given an already computed annulus $U_k$ that has been endowed with parameterization $U_k = U_k(r, \theta)$, $(r, \theta) \in [r_k, r_{k+1}] \times [0, 2\pi]$, then the next segment is simply

$$U_{k+1}(r, \theta) = f(U_k(r, \theta)),$$

(1.2)

so that $U_{k+1}(r, \theta)$ is a parametric surface depending on the same parameter.

Clearly any numerical method employing this idea by using the same number of points to approximate the fundamental domain $U_0$ and all its forward images will have several disadvantages. First, the distribution of points along each piece is not controlled. A few iterations of the fundamental domain may produce very closely spaced points in some regions while large gaps between points occur elsewhere. Next, the size of a primary segment/annulus tends to increase rapidly with the number of iterations of the initial segment. The number of points required to resolve a later segment/annulus is generally much greater than that required to resolve the previous segments/annuli; see Figure 1.2. These arguments suggest that it is better to place the points adaptively along the manifold.

We focus our work on adaptive methods that incorporate ideas from computer-aided geometric design in the manner that have not been seen before in the field of dynamical systems. We propose using higher-order approximation techniques to construct the methods for computing invariant manifolds. These methods are able to adapt the distribution and number of points on each segment/annulus. This avoids large gaps between successive points on a segment without placing too many points on that segment/annulus. It can also avoid using far too many points in the smaller segments/annuli while still resolving the larger segments/annuli. Our goal is to generate a higher-order approximation of the manifold which is smoothly resolved with a minimum number of points and that is accurate even for manifolds with complicated folds.
CHAPTER 2

HIGH-ORDER ADAPTIVE METHOD FOR COMPUTING
ONE-DIMENSIONAL INVARIANT MANIFOLDS OF MAPS

This chapter is organized as follows. In Section 2.1, we introduce the model problem of drawing a parametric curve along with a few methods for parameterizing a curve. Section 2.2 gives a description of the tools from CAGD that are at the heart of the algorithms. Topics covered include piecewise linear interpolation, Bézier curves, and Catmull-Rom splines. Section 2.3 contains descriptions of adaptive methods for rendering parametric curves. We first show existing methods due to Hobson and Carter, next, briefly, a few other existing methods, and, at the end, the parameterization method. Then we describe two new methods. In Section 2.4, we perform numerical tests of the 2D geometric tools introduced earlier, especially Catmull-Rom splines, as well as tests of the proposed methods for rendering parametric curves. Section 2.5 contains a further description of the implementation of our proposed methods in the context of computing one-dimensional invariant manifolds. In Section 2.6, we perform various numerical tests showing convergence of the methods. Section 2.7 contains a summarizing discussion of the advantages of the algorithms for computing one-dimensional invariant manifolds presented here.

2.1 Model Curves

2.1.1 Parametric curves

Motivated by (1.1), we delay considering the problem of computing an unstable manifold and instead focus on the simpler problem of drawing a parametric curve

\[ \gamma = \{ \mathbf{x} \in \mathbb{R}^2 : \mathbf{x} = g(t), \ a \leq t \leq b \}. \]  

(2.1)
To discuss what “drawing a curve” means, it is useful to first discuss the idea of “geometric modeling.” Any computational algorithm must replace the continuous mathematical object $\gamma$ by a finite collection of data that can be stored and manipulated by a computer. These tools were developed as part of the field of CAGD and are at the heart of modern computer graphics, animation, and CAD programs.

Most existing methods for calculating invariant manifolds do not explicitly discuss the idea of geometric modeling, but all implicitly choose a model in order to construct an algorithm. In the cited references, the curve is modeled by a “discrete curve”, a sequence of points connected by a linear interpolant. The question addressed by the existing algorithms, then, is how best to generate a finite set of parameter values such that the model curve satisfies some analytic or aesthetic criteria. We discuss previous algorithms in Subsection 2.3.1. Similarly, existing methods for generating two-dimensional invariant manifolds are likewise based on linear (planar) interpolation of points on a surface. As a loose analogy, algorithms based on linear interpolation have local approximation error of $O(\Delta^2)$ where $\Delta$ is the distance between two consecutive interpolation points — the equivalent of the forward Euler schemes students first learn for the solution of ODE initial value problems. The proposed research would, stretching this metaphor, bring this to the level of second or third-order Runge-Kutta, with local errors scaling as $\Delta^3$ or $\Delta^4$ — this is made more precise in the next section.

In addition to drawing a curve that “looks nice” at screen resolution or upon “zooming in” to examine the features of a curve, there are analytic and topological criteria that our approximate or model curve should meet. For example the unstable manifold $W^u$ cannot have any self-intersections; this would violate the uniqueness of solutions. Such a point would have to have two preimages on the manifold and that, by the invertibility of the map, can not happen. The algorithms below would need to be modified to meet these additional criteria.
2.1.2 Parameterization

The spline interpolation problem in numerical analysis is usually stated for a given sequence of points \( \{x_0, \ldots, x_n\} \) and their corresponding parameter values \( \{t_0, \ldots, t_n\} \), called knots. In applications such as computer graphics, the parameter values are rarely provided and therefore must be chosen somehow. Chapter 9 of the textbook by Farin [14], dedicates a fair amount of discussion to how best to pick the parameter values. Two commonly used parameterizations are the uniform parameterization, \( t_k = k \) and the accumulated chord length parameterization: defined inductively by \( t_0 = 0 \) and \( t_k - t_{k-1} = ||x_k - x_{k-1}|| \), which gives a crude approximation to arc-length parameterization. The first method often works poorly because it ignores the geometry of the points. The second method spaces the knots proportionally to the distance between points and usually produces better results, although not in all cases.

Another parameterization which we have found to be very natural for our problem we have named the inductive parameterization. Details of this parameterization we will present in Subsection 2.5.1.

2.2 2D Geometric Modeling Tools

Here we introduce a few tools from CAGD that are used to construct the numerical method. All these tools are precisely defined, and a full explanation is given of their desirable properties, and the relations between them. We also discuss various notions of continuity, efficient algorithms for evaluating these curves, and the expected accuracy of the various approximations made.

2.2.1 Piecewise Linear Interpolation

For a given sequence of points \( x_0, \ldots, x_n \) and corresponding parameter values \( t_0, \ldots, t_n \), the line segment between the points \( x_{k-1} \) and \( x_k \) can be written as

\[
s_k(t) = \frac{t_k - t}{t_k - t_{k-1}} x_{k-1} + \frac{t - t_{k-1}}{t_k - t_{k-1}} x_k \quad \text{for} \quad t \in [t_{k-1}, t_k].
\]
Therefore, the whole interpolated curve can be written as

$$\gamma_{\text{approx}} = \bigcup_{k=1}^{n} s_k(t).$$

To accurately approximate a smooth curve using linear interpolation requires a large number of points, especially near regions of large curvature.

We can estimate the error of the linear interpolation. For any $C^2$ curve such that $\gamma : [t_{k-1}, t_k] \to \mathbb{R}^n$ and the linear interpolant given by Equation (2.2),

$$\sup_{t \in [t_k, t_{k-1})} ||\gamma(t) - s_k(t)|| \leq \frac{(t_k - t_{k-1})^2}{8} \sup_{t \in [t_k, t_{k-1})} ||\gamma''(t)||;$$

see the proof in [15].

### 2.2.2 Bézier Curves

The Bézier Curves are a family of degree $n$ curves, each defined in terms of $n + 1$ control points $\{p_0, p_1, \ldots, p_n\}$, with $n = 3$ being the most common choice, for a parameter interval $[0, 1]$. These curves interpolate the first and last points $p_0$ and $p_n$ at parameter values 0 and 1, respectively, while the other points merely influence the shape of the curve. An important property of these curves is the convex hull property, namely that the curve lies inside the convex hull of the control points for all parameter values between 0 and 1. We make use of this property to define the adaptivity condition.

This fundamental object in geometric modeling can be constructed using the Bernstein polynomials. The Bernstein polynomials are given by

$$B^n_k(t) = \binom{n}{k} t^k (1-t)^{n-k}; \ k = 0, \ldots, n. \quad (2.3)$$

Note that, by the binomial theorem,

$$1 = (t + (1-t))^n = \sum_{k=0}^{n} B^n_k(t). \quad (2.4)$$
A Bézier curve has parametric form given by the convex affine combination (i.e., with positive weights summing to one) of $n + 1$ control points $p_0, \ldots, p_n$:

$$
\beta(t) = \sum_{k=0}^{n} B^n_k(t)p_k.
$$

(2.5)

Note that $\beta(0) = p_0$ and $\beta(1) = p_n$, but that the other control points are not interpolated by the curve. We can see from Equation (2.4) that for $0 \leq t \leq 1$, the point $\beta(t)$ is a weighted average of the control points, and as the weights defined by the Bernstein polynomials are positive, this curve must lie inside the convex hull of the control points. The polygon $P$ formed by $p_0, \ldots, p_n$ is called the Bézier polygon or control polygon of the curve. An example of a cubic Bézier curve is shown in Figure 2.1. Note that in general the edge of a convex hull (the bounding box) is not the same as the control polygon; as in the curve shown in Figure 2.3.

Figure 2.1  A cubic Bézier curve, together with its control points $p_0, p_1, p_2,$ and $p_3$, and their convex hull.

Bézier curves are invariant under affine transformation of the independent variable $t$; setting

$$
t = \frac{u - a}{b - a},
$$

(2.6)

then the curve $\gamma(t(u))$ over the interval $[a, b]$ parameterizes the same curve as $\gamma(t)$ over $[0, 1]$. 

Finally, note from Equation (2.5) that the tangent vectors to the curve at the endpoints $p_0$ and $p_n$ are given by

$$T_0 = n(p_1 - p_0) \text{ and } T_n = n(p_n - p_{n-1}),$$  \hspace{1cm} (2.7)

respectively—this formula is modified slightly if the parameterization in Equation (2.6) is used. This can be observed in Figure 2.1.

Bézier curves are widely used in CAGD because, in addition to the above properties, there exist efficient algorithms for evaluating them (forward-differencing and the de Casteljau algorithm), and for performing other calculations such as finding their intersections [53, 55] or their arc length [22].

### 2.2.2.1 The de Casteljau Algorithm

The de Casteljau algorithm is a recursive method to evaluate a Bézier curve $P_{[a,b]}$ at an arbitrary parametric location $t \in (a, b)$. As an example, the geometric interpretation of the evaluation algorithm for a point on cubic Bézier curve is shown in Figure 2.2. We label the control points of a cubic Bézier curve $P_{[a,b]}$ with $p_0^0$, $p_1^0$, $p_2^0$, and $p_3^0$. It works by using iterated linear interpolation. Each line segment in the trellis is split in the ratio $(t - a)/(b - t)$, i.e., we define

$$p_{j+1}^i = ((t - a)p_j^i + (b - t)p_{j+1}^i)/(b - a).$$  \hspace{1cm} (2.8)

The value $P_{[a,b]}(t) = p_3^0$ is the value of $\beta(t)$ at $t$ as defined in Equation (2.5).

The de Casteljau algorithm can also be used to subdivide a Bézier curve $P_{[a,b]}$ into two shorter Bézier curves $P_{[a,t]}$ and $P_{[t,b]}$ whose union is equivalent to $P_{[a,b]}$. The control points for $P_{[a,t]}$ are $p_0^0$, $p_1^0$, $p_2^0$, and $p_3^0$ and the control points for $P_{[t,b]}$ are $p_3^0$, $p_2^1$, $p_1^2$, $p_0^3$. This is not the most efficient computational algorithm to compute $P_{[a,b]}$ but is very important in the mathematical theory of such curves.
2.2.2.2 Degree Elevation

Any Bézier curve of degree \( n \) is defined by its control polygon \( p_0, \ldots, p_n \). A degree \( n \) polynomial may be written as a degree \( n + 1 \) polynomial with leading coefficient 0, therefore a degree \( n \) Bézier curve may be represented as “degree elevated form” of the sum of Bernstein polynomials \( B_{n+1}^k(t) \). The control points \( p_0^1, \ldots, p_{n+1}^1 \) that describe the same curve are given by

\[
p_i^1 = \frac{i}{n+1} p_{i-1} + \left(1 - \frac{i}{n+1}\right) p_i; \quad i = 0, \ldots, n + 1.
\]

Thus new vertices \( p_i^1 \) are obtained from the old polygon by piecewise linear interpolation at the parameter value \( i/(n + 1) \). It follows that the new polygon lies in the convex hull of the old one; see Figure 2.3.

Figure 2.2 The de Casteljau Algorithm for evaluating/subdividing a cubic Bézier curve.

Figure 2.3 Degree elevation: both polygons define the same cubic curve.
2.2.2.3 Composite Bézier Curves

Several Bézier curves may be pieced together in order to generate shapes that are too complex for a single low-degree Bézier curve to handle. (High degree Bézier curves suffer from Runge’s phenomenon). In concatenating Bézier curves, we need to control the smoothness of the composite curve. Let \(p_0, \ldots, p_3\) and \(p_3, \ldots, p_6\) be the Bézier points of two cubic curve segments \(R_{[a,c]}\) and \(Q_{[c,b]}\); see Figure 2.4. Since they share the point \(p_3\), their union clearly forms a continuous, or \(C^0\), curve. With this minimal continuity requirement, the two curves may form a corner. To ensure that the two pieces meet smoothly, more care is called for.

\[
\begin{align*}
\text{Figure 2.4} & \quad \text{Composite of two Bézier curves.} \\
\end{align*}
\]

Two adjacent curve segments \(P\) and \(Q\) are said to be \(C^k\)-continuous at their common end point \(c\) (or, to have \(k\)th order parametric continuity) if

\[
P(c) = Q(c), \quad P'(c) = Q'(c), \ldots, \quad P^{(k)}(c) = Q^{(k)}(c).
\]

Thus, \(C^0\) means simply that the two adjacent curves share a common endpoint, \(p_3\) in our case. \(C^1\) means that the two curves not only share the same endpoint, but also that they have the same first order parametric derivatives. \(C^2\) means that two curves are \(C^1\) and in addition that they have the same second order parametric derivatives at
their shared endpoint. Equation (2.7) demonstrates that this curve is $C^1$ if and only if $(p_3 - p_2)/(c - a) = (p_4 - p_3)/(b - c)$. Similar condition exists for $C^2$ and higher continuity.

A slightly weaker notion of continuity of a piecewise curve, one which is independent of parameterization, is called geometric continuity and is denoted $G^k$. The two curves are $G^k$-continuous at $p_3$ if the $k$th derivative vectors from both sides point in the same direction. In practice this means that the two component curves may be reparameterized to make their union $C^k$. The conditions for geometric continuity (also known as visual continuity) are less strict than for parametric continuity.

$G^1$ continuity requires that the three points $p_2, p_3$ and $p_4$ are collinear; i.e., they have a common tangent line at their shared endpoint. $G^2$ (second order visual or geometric continuity) means that the two neighboring curves have the same tangent line and also the same curvature at their common boundary. Clearly, two curves which are $C^n$ are also $G^n$.

In many problems, one would like to construct a curve that interpolates (or comes very close to) the the points \{${x}_0, \ldots, {x}_n$\} at the parameter values \{${t}_0, \ldots, {t}_n$\}. We would like to create an interpolation method based on the composite Bézier curves, but they, by themselves, are insufficient to construct an interpolation scheme. There exists an interpolation method called Catmull-Rom splines, but before we introduce this method we need to briefly describe the idea of the Hermite Interpolating polynomials.

### 2.2.3 Hermite Interpolating Polynomials

Given two points, $x_1$ and $x_2$ and two vectors, $\vec{v}_1$ and $\vec{v}_2$, the cubic Hermite interpolating polynomial is the unique cubic polynomial $p(t)$ that interpolates the two points at $t = 0$ and $t = 1$, respectively, with tangent vectors $\vec{v}_1$ at $x_1$ and $\vec{v}_2$ at $x_2$. Using relation (2.7), the Hermite interpolating polynomial can be written as a cubic Bézier curve with

\[
p_0 = x_1, \quad p_1 = x_1 + \vec{v}_1/3, \quad p_2 = x_2 - \vec{v}_2/3, \quad \text{and} \quad p_3 = x_2.
\]  

(2.9)
It should be noted that, unlike Bézier curves, Hermite interpolating polynomials are not invariant under affine changes of the parameter $t$, given by Equation (2.6). Such a change of variables changes the length of the tangent vectors and thus produces a different curve. To preserve the shape of the curve, the tangent vectors $\vec{v}_1$ and $\vec{v}_2$ must be scaled appropriately.

If a curve $\gamma(t) = (x(t), y(t))$ is approximated by its Hermite interpolant, with $\vec{v}_1$ and $\vec{v}_2$ given by the exact tangent vectors at the endpoints, the error for $t \in [t_{k-1}, t_k]$ is given by

$$x(t) - p_1(t) = \frac{x^{(4)}(c)}{4!}(t - t_{k-1})^2(t - t_k)^2$$

for some $c \in [t_{k-1}, t_k]$,

the maximum of the right hand side is achieved for $t = (t_{k-1} + t_k)/2$, so we can bound the error in the first component for $t \in [t_{k-1}, t_k]$ with the expression

$$\sup_{t \in [t_{k-1}, t_k]} ||x(t) - p_1(t)|| \leq \frac{(t_k - t_{k-1})^4}{384} \sup_{t \in [t_{k-1}, t_k]} ||x^{(4)}(t)||.$$  

The error in the $y$ component has a similar bound. Then, we can estimate the error of the Hermite interpolant by

$$\sup_{t \in [t_{k-1}, t_k]} ||\gamma(t) - p(t)|| \leq \frac{(t_k - t_{k-1})^4}{384} \left( \left( \sup_{t \in [t_{k-1}, t_k]} |x^{(4)}(t)| \right)^2 + \left( \sup_{t \in [t_{k-1}, t_k]} |y^{(4)}(t)| \right)^2 \right)^{1/2}.$$  

This guarantees that refining the set of the parameter values results in a better approximation of the parametric curve.

### 2.2.4 Catmull-Rom Splines

One familiar way to interpolate $n$ points is with cubic splines – piecewise cubic curves chosen to satisfy matching conditions at the interpolated points so that the overall curve is $C^2$. This scheme has two disadvantages which have led to its rejection in CAD and computer animation systems. First, satisfying the matching conditions requires solving
a linear system over all the unknowns, so that if new points are added, the entire object must be recomputed. Second, and closely related, the method is non-local; changes to one interpolated point lead to changes in all the coefficients. Interpolation methods used in computer animation generally use piecewise-cubic splines, but abandon the $C^2$-matching requirement, which allows for far greater flexibility.

A degree $n$ Bézier curve does not solve this interpolation problem, so we turn to piecewise Bézier curves, namely Catmull-Rom splines. These satisfy certain continuity properties, while maintaining local control, i.e., a change to a given point $x_k$ will change the curve only on some restricted parameter interval $[t_{k-j}, t_{k+j}]$ rather than on the whole interval $[t_0, t_n]$. Recall that a cubic Hermite interpolant is defined to match both the function values and the tangent vectors at the endpoints of a curve segment. A Catmull-Rom spline is defined to be an Hermite curve, where the tangent vectors are determined by finite-difference approximations of the interpolation data.

In a general interpolation problem, the tangent of is not provided and must instead be approximated. To construct the segment connecting the points $x_k$ and $x_{k+1}$ approximate tangent vectors at those points are needed; call them $\vec{v}_k^+$ and $\vec{v}_{k+1}^-$. These, in turn, require an approximation $\tilde{v}_k$ to $\frac{dx}{dt}|_{t=t_k}$. We discuss two different approximations; the first based on three-point centered-differences and the second based on five-point centered-differences.

In the first case, under the uniform parameterization $t_k = k\Delta$ this is just

$$\tilde{v}_k = \frac{(x_{k+1} - x_{k-1})}{(2\Delta)}$$

but for general parameterization, it is

$$\tilde{v}_k = \frac{\Delta_k^2(x_{k+1} - x_k) + \Delta_{k+1}^2(x_k - x_{k-1})}{\Delta_{k+1}\Delta_k(\Delta_{k+1} + \Delta_k)}, \quad k = 1, \ldots, n-1$$

where $\Delta_k = t_k - t_{k-1}$. Since Hermite interpolating polynomials are not affine invariant with respect to $t$, this must be scaled by the length of the interval in order to give the right tangent vector to interpolate on $[t_{k-1}, t_k]$. This gives formulas

$$\vec{v}_k^+ = \Delta_k \tilde{v}_k \quad \text{and} \quad \vec{v}_{k+1}^- = \Delta_k \tilde{v}_{k+1}$$
for the tangent vectors at the left and right end points of the interior points \( k = 1, \ldots, n-1 \). In the formulas for the tangent vectors \( \vec{v}_0^+ \) and \( \vec{v}_n^- \) at the endpoints, the centered difference formula is replaced with a three-point one-sided difference formula, which will, on average, give twice the approximation error as the centered-difference formula.

The second approach, based on five-point centered-differences gives

\[
\tilde{v}_k = \frac{-x_{k+2} + 8x_{k+1} - 8x_{k-1} + x_{k-2}}{12\Delta_k}, \quad k = 2, \ldots, n-2,
\]

under the uniform parameterization. The formula for \( \tilde{v}_k \) under the non-uniform parameterization is of course more complex. Following [17] the approximation to the first derivative may be written as

\[
\alpha \tilde{v}_{k-1} + \tilde{v}_k + \beta \tilde{v}_{k+1} = Ax_{k+1} + Bx_{k-1} + Cx_{k+2} + Dx_{k-2} + Ex_k,
\]

where \( \alpha, \beta, A, B, C, D \) and \( E \) are functions of the non-uniform mesh spacings \( \Delta_k = t_k - t_{k-1} \). Relations between former coefficients can be derived by matching the Taylor series of various orders. The following relations are obtained

\[
A + B + C + D + E = 0,
\]

\[
\Delta_{k+1} A - \Delta_k B + (\Delta_{k+2} + \Delta_{k+1})C - (\Delta_k + \Delta_{k-1})D = 1 + \alpha + \beta,
\]

\[
\Delta_{k+1}^2 A + \Delta_k^2 B + (\Delta_{k+2} + \Delta_{k+1})^2 C + (\Delta_k + \Delta_{k-1})^2 D = \frac{2!}{1!}(\Delta_{k+1} \beta - \Delta_k \alpha), \quad (2.10)
\]

\[
\Delta_{k+1}^3 A - \Delta_k^3 B + (\Delta_{k+2} + \Delta_{k+1})^3 C - (\Delta_k + \Delta_{k-1})^3 D = \frac{3!}{2!}(\Delta_{k+1}^2 \beta + \Delta_k^2 \alpha),
\]

\[
\Delta_{k+1}^4 A + \Delta_k^4 B + (\Delta_{k+2} + \Delta_{k+1})^4 C + (\Delta_k + \Delta_{k-1})^4 D = \frac{4!}{3!}(\Delta_{k+1}^3 \beta - \Delta_k^3 \alpha),
\]

The solution of the linear system of Equations (2.10), where \( A, B, C, D \) and \( E \) are the unknowns, leads to a forth-order scheme. In these expressions, the parameters \( \alpha \) and \( \beta \) are considered equal constants, i.e., \( \alpha = \beta = 1/3 \).
To derive formulas for the tangent vectors $\vec{v}_0^+$ and $\vec{v}_n^-$ at the endpoints, the centered difference formulas are replaced by one-side differences. Similarly, to derive formulas for the tangent vectors $\vec{v}_1^-$ and $\vec{v}_1^+$ at the second point, and the tangent vectors $\vec{v}_{n-1}^-$ and $\vec{v}_{n-1}^+$ at the second point from the end, the centered difference formulas are replaced by asymmetric differences.

Using (2.9) to express Hermite interpolating polynomials as cubic Bézier curves, we consider Catmull-Rom interpolating splines as composite Bézier curves. The way that the control points of each Bézier segment are constructed guarantees the $C^1$ continuity of Catmull-Rom splines.

2.2.5 B-splines

B-splines are piecewise polynomial curves, offering much more versatility than Bézier curves. In practical terms, B-spline curves can be thought of as a degree $n$ composite Bézier curves that join automatically with $C^{n-1}$ continuity, regardless of where the control points are placed.

Assume that $n$ is the degree of each polynomial segment, $L + 1$ is the number of control points, and $K$ is the number of intervals. Then $\{t_0, \ldots, t_K\}$ is a list of parameter values, or knots and $p_0, \ldots, p_L$ are control points, with $L = K - n + 1$. The $k$th segment of a degree $n$ B-spline curve obeys the following:

- It is defined over the parameter interval $[t_k, t_{k+1}]$.
- It depends on the $n + 1$ control points $-p_{k-n}, \ldots, p_k$.
- It depends on $2n$ knots $-t_{k-n+1}, \ldots, t_{k+n}$.

The B-spline curve can be only defined over the interval $[t_{n-1}, \ldots, t_L]$. The above properties are not sufficient to define the curve. One way to define it is using the de Boor algorithm below.
Whereas a composite of \( m \) Bézier curves of degree \( n \) involve \( mn + 1 \) distinct control points (shared control points counted only once), that same string of Bézier curves can be expressed using only \( m + n \) B-spline control points (assuming all neighboring curves are \( C^{n-1} \)). A basic operation, extracting the constituent Bézier curves from B-splines, is called the Böhm algorithm; see [14].

Obviously, a knot vector must be a non-decreasing sequence of real numbers. If any knot value is repeated, it is referred to as a multiple knot. A B-spline curve whose knot vector is evenly spaced is known as a uniform B-spline. If the knot vector is not evenly spaced, the curve is called a non-uniform B-spline.

![Cubic B-spline generated by the knot vector \([t_0, \ldots, t_7]\).](image)

**Figure 2.5** Cubic B-spline generated by the knot vector \([t_0, \ldots, t_7]\).

The knot vectors are traditionally described as requiring \( n \) end-condition knots, and in the real world you will always find a meaningless additional knot at the beginning and end of a knot vector. For example, the values of \( t_0 \) and \( t_K \) have absolutely no effect on the curve. Therefore, we can ignore these dummy knot values, but be aware that they appear in B-spline literature and software; see Figure 2.5.
Basis splines

For all B-spline piecewise polynomials over the knot sequence \([t_{n-1}, t_L]\) there exist \(L + 1\) linearly independent piecewise polynomials \(N^n_i(t)\) that form a basis. Every B-spline piecewise parametric curve over \([t_{n-1}, t_L]\) may be written uniquely in the form

\[
\gamma(t) = \sum_{k=0}^{n} N^n_k(t) p_k.
\]

If we set all \(p_i = 1\), similarly as for Bernstein polynomials (2.4), the sum will be identically equal to 1, i.e., \(\sum_{k=0}^{n} N^n_k(t) = 1\), thus B-splines form a partition of unity. For more details about the basis splines see [14].

2.2.5.1 The de Boor Algorithm

The de Boor algorithm is a generalization of the de Casteljau algorithm. It provides a method for evaluating a B-spline curve at a given parameter value \(t\) in the domain, and can be thought of as a definition of the B-spline curve

A B-spline needs to be evaluated at given \(t \in [t_I, t_{I+1})\). Renumber the set of relevant control points \(p_{I-n+1}, \ldots, p_{I+1}\) as \(p_0, \ldots, p_n\). Next, split each line segment in the trellis in the ratio \(\alpha^k_j\), i.e., define

\[
p^k_j = (1 - \alpha^k_j) p^{k-1}_j + \alpha^k_j p^{k-1}_{j+1}.
\]

with

\[
\alpha^k_j = \frac{t - t_{I+j+1}}{t_{I-n+k+j} + t_{I+j+1}}
\]

for \(k = 1, \ldots, n\) and \(j = 0, \ldots, n - k\), see Figure 2.6. Here, we assume that \(t\) is not in the knot sequence.

The de Boor algorithm can be considered as the knot insertion process; a knot is added to the knot vector of a given B-spline. This results in an additional control point and a modification of a few existing control points. The end result is a new representation
of the existing curve, defined by a larger number of control points. One application is to provide a designer with the ability to add local details to a B-spline. Knot insertion provides more local control by isolating a region to be modified from the rest of the curve, which thereby becomes immune from subsequent local modification.

**Figure 2.6** The de Boor construction algorithm for a point on a segment of a cubic B-spline.

In CAGD literature there exists a good labeling scheme that can provide a lot of information about an algorithm. That labeling scheme leads to the notion of blossoming, an extremely powerful technique for analyzing the properties of curves and surfaces. Blossoming is a particularly effective tool for deriving change of basis algorithms. The idea was developed by de Casteljau and it is closely related to his algorithm. It presents a natural relation between algorithms such as subdivision (the de Casteljau algorithm) and degree elevation of Bézier curves, and the de Boor algorithm for B-splines. The same blossoming principle can be applied to surface patches. Because of its notational complexity we do not use blossoming in our description, however we point to its existence as a powerful tool. Very good introductions to blossoming are presented by Farin in [14] and by Goldman in [18].

### 2.2.6 Quasi-Interpolation Schemes

Quasi-interpolation is a method of constructing a curve from data that has attractive smoothness properties and local control, at the expense of sacrificing the interpolation
property. We consider the quasi-interpolation operator constructed by spline functions. A spline Quasi-Interpolant (QI) of \( f \) can be defined as operators of the form

\[
Qf = \sum_{j \in J} \mu_j(f)B_j
\]

where \( \{B_j : j \in J\} \) is a basis of some space of splines, say of degree \( m \), and \( \{\mu_j : j \in J\} \) is a family of linear functionals which are local in the sense that they only use values \( f \) in some neighborhood of \( \sum_j := \text{supp}(B_j) \). We denote by \( P_m \) the space of polynomials of total degree at most \( m \). Quasi-interpolation is defined by imposing the condition that \( Q \) is exact for functions \( f \) on the space \( P_m \), i.e., \( Qp = p \) for all \( p \in P_m \). As a consequence of this property, the approximation order is \( O(h^{m+1}) \) on smooth functions, \( h \) being the maximum steplength of the partition. The case when the coefficients \( \mu_j \) are linear combinations of discrete values of the function in some neighborhood of \( \sum_j \), is called a discrete Quasi-Interpolation.

The main advantage of QIs is that they have a direct construction without solving any system of linear equations. Moreover, they are local, in the sense that the value of \( Qf(x) \) depends only on values of \( f \) in a neighborhood of \( x \). Finally, \( Qf - f \) converges fairly rapidly as \( h \to 0 \).

For example, a single \( n \)-degree Bézier curve defined in terms of control points \( f \left( \frac{j}{n} \right), j = 0, \ldots, n \) may be thought of as a quasi-interpolant

\[
Qf(t) = \sum_{j=0}^{n} f \left( \frac{j}{n} \right) B_j^n(t), \text{ for } t \in [0, 1].
\]

### 2.2.6.1 Quasi-Interpolant based on Cubic B-splines

Consider the following cubic spline QI defined on \( \mathbb{R} \), endowed with a nonuniform partition, by

\[
Q_3f(t) := \sum_{i \in \mathbb{Z}} \mu_i(f)N_i^n(t),
\]
where $N_i^n$ is the cubic B-spline basis function with support $[t_{i-2}, t_{i+2}]$ centered at $t_i$ and

$$\mu_i(f) := a_i f_{i-1} + b_i f_i + c_i f_{i+1},$$

where $f_i := f(t_i)$, and the coefficients are given by

$$a_i = -\frac{1}{3} \frac{h_i^2}{h_{i-1}(h_{i-1} + h_i)}, \quad b_i = \frac{1}{3} \frac{(h_{i-1} + h_i)^2}{h_{i-1}h_i}, \quad c_i = -\frac{1}{3} \frac{h_{i-1}^2}{h_i(h_{i-1} + h_i)},$$

where $h_i := t_i - t_{i-1}$. Note that for a set of knots and function values $\{t_i, f_i\}_{i=0}^n$, the generated set of control points $\{\mu_j\}_{j=1}^{n-1}$ defines the spline function over the interval $[t_2, t_{n-2}]$. It is easy to verify that $Q_3$ is exact on $P_3$.

### 2.3 Adaptive Methods

In the standard interpolation problem of CAGD, as stated for example [14], one is given a sequence of points $x_0, \ldots, x_n$ and would like to draw a curve passing through those points in the given order. Our goal is somewhat different. We seek to choose the points $x_k = \gamma(t_k)$ as efficiently as possible in order to accurately render the curve.

#### 2.3.1 Existing Methods

Algorithms for efficiently computing unstable manifolds go back at least the 1980’s [50]. Two methods developed for drawing unstable manifolds of maps are due to Hobson [32] and Carter [4]. Both are essentially algorithms for computing a parametric curve $\gamma$ in Equation (2.1), although neither is framed this way. Suppose a model curve is defined by piecewise linear interpolation between $n+1$ points $\{x_0 = f(a), x_1 = f(t_1), \ldots, x_n = f(b)\}$. Define the lengths $l_k = ||x_k - x_{k+1}||$ and the the angle $\alpha_k$ between consecutive linear segments $x_{k-1}x_k$ and $x_kx_{k+1}$. Then in both methods, a model curve is considered acceptable
if it satisfies

\[ \alpha_k < \text{tol}_1, \quad (2.11a) \]
\[ \alpha_k l_k < \text{tol}_2, \quad \text{and} \quad \alpha_k l_{k-1} < \text{tol}_2, \quad (2.11b) \]

where \( \text{tol}_1 \) and \( \text{tol}_2 \) are user-specified tolerances. The first condition (2.11a) states that two consecutive segments should be in nearly the same direction, while the second and third (eq. (2.11b)) help to control the arc length. The first condition is scale-invariant while the second has an absolute scale. A schematic of such a linear interpolant, together with the associated angles and lengths, is shown in Figure 2.7.

2.3.1.1 The Hobson Method: Marching

Hobson’s method begins at \( t_0 = a \) and adds new points \( x(t_k) \) to a given list of points until \( t = b \) is reached. The method supposes that a sequence of points \( \{x_j = f(t_j), j = 0, \ldots, k\} \) has already been found and attempts to find the next point \( x_{k+1} = f(t_{k+1}) \). Let \( s \) be a short parameter increment and let \( t' = t_k + s, t'' = t_k + 2s, \)
\( x' = f(t') \) and \( x'' = f(t'') \). Define the vectors \( \vec{v}_1 = x_k x' \) and \( \vec{v}_2 = x' x'' \). Let \( d = ||\vec{v}_1|| \) and \( \alpha \) be the angle between \( \vec{v}_1 \) and \( \vec{v}_2 \). Hobson’s algorithm uses the conditions

\[ \alpha < \text{tol}_1, \quad (2.12a) \]
\[ \alpha d < \text{tol}_2 \quad (2.12b) \]
to determine whether to accept the point $x'$. If the tolerances are not met, the algorithm decreases $s$ and tries again. In either case, the point $x''$ is discarded. No explicit mention is made in [32] on a condition for choosing $s$, although the method for choosing how to adjust $s$ clearly can make a large difference in the efficiency of the algorithm. For example, one might take

$$s \rightarrow 0.95 \max \left\{ \frac{\text{tol}_1}{\alpha}, \frac{\text{tol}_2}{\alpha d} \right\} s$$

after each step, regardless of whether the current step is accepted.

There are two sources of inefficiency in this method. First, at each step, the computation of $x''$ does not contribute a point to the curve. In addition, each time a point $x'$ is rejected, two points are computed that do not contribute to the curve. Further note that the conditions (2.12) for accepting $x'$ are not equivalent to the conditions (2.11). It is possible, for the algorithm to satisfy conditions (2.12) while badly failing to satisfy conditions (2.11); see Figure 2.8. An obvious downside to this method is the need to compute and discard a point $x''$ at each step. A simpler method, given the curve up to $x_k$, would be to compute a candidate point $x'$ and test whether the three points $x_{k-1}, x_k$ and $x'$ satisfy conditions (2.11). Hobson demonstrates that this method can fail at instances that $x_k$ itself is too far from $x_{k-1}$ for any such $x'$ to produce an acceptable value of the angle $\alpha$, in particular if $x_{k-1}$ and $x_k$ are situated on opposite sides of a hairpin turn in the curve $\gamma$ as in Figure 2.8. This method is widely cited, as is a similar method [35].

![Figure 2.8](image-url) A situation where part of the curve might be missed.
2.3.1.2 The Carter Method: Bisection

Carter’s method, by contrast, is based on bisection. Given an approximation to the whole curve $\gamma$, the algorithm calculates all the angles $\alpha_k$ between the segments as shown schematically in Figure 2.9. For each point $x_k$ at which either condition of (2.11) is violated, the algorithm adds new points at $t^+_k = (t_k + t_{k+1})/2$ and $t^-_k = (t_{k-1} + t_k)/2$. A new approximate curve is constructed interpolating $\{x(t_k)\} \cup \{x(t^\pm_k)\}$.

Figure 2.9 The angle between consecutive linear segments and the bisection algorithm.

Carter compares his algorithm to Hobson’s (see Table 2.1) and finds that for strict tolerances, his method can draw the curve with as few as one third as many calls to the function $f$. He found that in practice it was sufficient to enforce only condition (2.11a) while ignoring (2.11b). Our experience is similar. This method has appeared only as a preprint and as an undergraduate honors thesis, and so has not been cited, although it is very similar to a method used, without description, to draw the tangents in [20, 21]. Although Carter was the first to propose this method in the context of invariant manifold calculations, it was earlier discussed in the computer graphics literature for parametric curves, although these papers are not well-cited. The method was proposed in [5] and subsequently in [9] and [60]. In these cases, the authors’ focus is on the rendering of parametric curves on computer screens or printers. Thus the refinement condition for the recursion is related to ensuring that any errors be smaller than one pixel. For dynamics problems we may be interested in resolving features of the curve that is finer than what is visible on the screen or on paper. We thus subject these methods to some exacting numerical diagnostics to test their effectiveness. In what follows, we refer to Carter’s method as Adaptive Linear Interpolation or ALI.
2.3.1.3 Other Existing Methods

Similar methods for computing the unstable manifold of planar maps, that can be seen as algorithms for computing a parametric curve, are developed by Krauskopf and Osinga in [11, 34, 35, 36, 37], and You et al. in [62]

Krauskopf and Osinga’s methods are based on similar conditions to Equations (2.12). The manifold is computed by adding one point after the other without using the idea of primary segments. This was inspired by difficulties in computing two-dimensional unstable manifolds as we will discuss in Subsection 3.1.2. However, the method is still based on linear interpolation.

You et al. present a slightly different approach. They do use the idea of primary segments. Their adaptive method, however, is controlled by the distance between resulting points. The procedure does not require an interpolation of any primary segment except the initial one, as long as the distance is maintained below a screen resolution.

2.3.1.4 Parameterization Method

Additionally, there exists an important rigorous, analytical method of computing invariant manifolds based on power series expansions that is suited just for this purpose. This method has been used both to prove analytical statements about invariant manifolds by Cabré et al. in [1, 2, 3] and to numerically calculate such manifolds [16]. In this method, a branch of the unstable manifold is written as \( \{(x(s), y(s))|0 < s < \infty\} \) where \( x \) and \( y \) are represented as power series in \( s \), and the invariance of the manifold under the map is used to derive equations for the coefficients in the series. If \( f \) is entire, this series has infinite radius of convergence, but, in practice, due to roundoff error, the numerical radius of convergence may be quite small, although there exist ways to increase this radius somewhat (without resorting to variable precision arithmetic). The method is also the basis for the recent computations of two-dimensional invariant manifolds, and their intersection by Mireles-James and Lomelí in [43, 45].
The goal of the parameterization method is to compute a smooth injection
\[ P : I = [-\tau, \tau] \subset \mathbb{R} \to \mathbb{R}^n, \]
such that
\[
P(0) = x^*,
\]
\[
P'(0) = \vec{v}_\lambda,
\]
\[
(f \circ P)(s) = P(\lambda s),
\]
where \( x^* \in \mathbb{R}^n \) is a fixed point of the map \( f \), \( \lambda \) is a stable or an unstable eigenvalue and \( \vec{v}_\lambda \in \mathbb{R}^n \) is an eigenvector of \( \lambda \). Note that for any \( P \) satisfying these conditions \( P(I) \) is an arc through the fixed point \( x^* \) tangent to eigenvector \( \vec{v}_\lambda \) with \( (f \circ P)(I) \subset P(I) \) for the stable manifold and \( (f^{-1} \circ P)(I) \subset P(I) \) for the unstable manifold. Then the image of \( P \) is a local manifold about \( x^* \).

In general \( P \) cannot be computed in closed form. Instead, we note that \( P \) satisfies a (functional) initial value problem with analytical data. Then it is natural to assume that \( P \) has power series expansion
\[
P(s) = \sum_{n=0}^{\infty} a_n s^n, \text{ where } a_n \in \mathbb{R}^n,
\]
with \( a_0 = x^* \) and \( a_1 = \vec{v}_\lambda \), and determine the unknown coefficients \( a_n \). The way to proceed is to insert Equation (2.14) into Equation (2.13), Taylor expand \( f \), and analytically compute recurrence relations for the coefficients of \( P \).

The method almost eliminates the need to compute the images of fundamental segments, and, thus, the propagation of error due to interpolation. Nonetheless, this method does not invalidate the work presented here. While this method provides an explicit formula to compute points on the manifold, it thus reduces the problem to that of drawing a parametric curve, i.e., choosing which points to plot and how to interpolate between them, which is exactly what our methods provide. Further, in the case that the map is given as a numerical routine and not by an analytical formula (e.g., as a numerical Poincaré map), this method is not applicable. Finally when the map is given
by an equation with complex singularities (e.g., poles) the power series may converge very slowly, giving a poor approximation, have finite radius of convergence or even fails to converge. In practice, even if \( f \) is an entire function the power series expansion has finite numerical radius of validity and it gives valid approximation only for a piece of the manifold. In order to compute a significant portion of the manifold, Mireles-James and Lomelí in [45] use this method to generate the initial segment, and then iterate.

The parameterization method can be seen as another motivation for the inductive parameterization that we describe in detail in Subsection 2.5.1. Note that substituting \( s = \lambda^t \) in (2.13) we obtain \(( f \circ P)(\lambda^t) = P(\lambda^{t+1})\).

### 2.3.2 Proposed Methods

We will demonstrate that ALI, while superior to Hobson’s method, does not compute invariant manifolds efficiently because it places too many points near hairpin turns. We propose to remedy this by using the Catmull-Rom spines described in Subsection 2.2.4.

### 2.3.2.1 Flatness Refinement Condition

We now discuss what the refinement condition, the analog to condition (2.11a), should be for a recursive method based on Catmull-Rom splines. The Catmull-Rom spline is by its definition \( C^1 \) at the interpolated points, so that the angle \( \alpha \) between two consecutive segments is identically zero. Because of this, our method depends on a condition applied within a single segment as opposed to the between-segments condition (2.11a). Geometrically, we require that each segment be sufficiently flat.

![Figure 2.10](image-url) The control points and several characteristic quantities of a Bézier curve.
Several possible flatness conditions are suggested by the equivalent Bézier form of the Catmull-Rom interpolant. Referring to the schematic of a Bézier control polygon given in Figure 2.10, several possible measurements of the flatness of the Bézier curve are given by

1. \( \max \{d_1, d_2\} \) (distance from interpolating line to the further control point);

2. \( |p_0p_1| + |p_1p_2| + |p_2p_3| - |p_0p_3| \);

3. \( \max \{d_1/d_0, d_2/d_0\} \) (aspect ratio of the bounding box);

4. the angle between the vectors \( \vec{v}_1 \) and \( \vec{v}_2 \);

5. \( (|p_0p_1| + |p_1p_2| + |p_2p_3| - |p_0p_3|)/|p_0p_3| \);

Each of the above conditions introduces a different approach to estimate the flatness. Moreover, conditions 1 and 2 have units of length whereas conditions 3, 4 and 5 are dimensionless. These conditions might seem arbitrary but are closely related to the geometry of Bézier curves. For example, the convex hull property guarantees that \( d_{\text{max}} \), the maximum distance between \( \gamma(t) \) and its interpolant is bounded by condition 1.\(^1\)

To motivate the flatness refinement condition we look at an application of the de Casteljau algorithm in Subsection 2.2.2.1, which is often used to adaptively render Bézier curves. The adaptive algorithm works by splitting a given cubic Bézier curve defined for \( t \in [a, b] \) into two equivalent Bézier curves, defined on the intervals \([a, (a + b)/2] \) and \([(a+b)/2, b], \) with the algorithm producing the value of the Bézier curve at the midpoint (choosing \( t = (a + b)/2 \) in Equation (2.8)). The algorithm chooses whether to split the interval based on a flatness condition like those on this list, often based on the resolution of the display device.

\(^1\)In fact, an elementary calculus exercise demonstrates that \( d_{\text{max}} \leq \frac{3}{4} \max \{d_1, d_2\} \), see [6].
Catmull-Rom splines based on three-point centered differences or five-point centered differences along with one of the above flatness refinement conditions give our next methods for adaptively resolving a parametric curve. We refer to these methods as Adaptive Catmull-Rom 3, \((\text{ACR3})\) and Adaptive Catmull-Rom 5, \((\text{ACR5})\), respectively.

### 2.3.2.2 Error Refinement Condition

We find in the numerical tests below that a slightly improved version of ALI (in which ALI is used to generate the points but the curve itself is defined as a cubic spline over these points) performs nearly as well as both these ACR methods. We note that each of these schemes features a refinement condition based on properties of a single model curve. Adaptive methods for other types of problems, for example adaptive quadratures and adaptive ODE initial value solvers, work by comparing two separate approximations in order to estimate an error and then refining near the locations where they disagree. We propose such a scheme here. At each iteration we compute both the 3-point and 5-point Catmull-Rom splines described in Subsection 2.2.4. The difference between them will give the refinement condition. As it is expensive to find the exact distance between two Bézier curves, we estimate it using their control polygons. To estimate the distance between two Catmull-Rom approximations we use the equivalent Bézier form of the Catmull-Rom interpolant. We refer to the schematic of a Bézier curve; see Figure 2.11.

![Figure 2.11](image)

(a) The two approximations \(\mathbf{p}\) and \(\mathbf{\tilde{p}}\).

(b) The error loop \(\mathbf{p} - \mathbf{\tilde{p}}\).

**Figure 2.11** Two different Catmull-Rom approximations, \(\mathbf{p}(t)\) and \(\mathbf{\tilde{p}}(t)\), of the curve \(\gamma(t)\).
This schematic depicts the difference between two different interpolants to the same sequence of points. Since the two curves interpolate the same data, the first and the last point of the difference curve is zero. Several possible estimations of the error between the two approximations are given by

1. \( \max\{|\vec{w}_1|, |\vec{w}_2|\} \); (maximum distance between the control points)

2. \( \max\left\{\frac{|\vec{w}_1|}{d_0}, \frac{|\vec{w}_2|}{d_0}\right\} \); (maximum relative distance between the control points)

Condition 1 has units of length whereas condition 2 is dimensionless. However, both these approximations are computationally less expensive than computing the maximum norm error directly; we can avoid “calculation” of the curve and estimate the error using the already-calculated control points. After experimenting with both the error conditions we chose to work with condition 1.

The adaptive method for rendering a parametric curve based on one of these error refinement conditions we call 3-point vs 5-point Adaptive Catmull-Rom or ACR3vs5.

### 2.3.2.3 Adaptive Quasi-Interpolation based on B-splines

This recursive method is based on the \(C^2\) cubic Quasi-Interpolation spline introduced in Subsection 2.2.6.1. To approximate a parametric curve \(\gamma\), over the interval \([a, b]\), method requires computation on the longer interval \([a - \delta_1, b + \delta_2]\). The algorithm starts by calculating the cubic QI splines based on a given data set \(D = \{(t_i, x_i), i = -2, \ldots, n+2\}\), where first two and last two buffer points are taken from outside the interval \([a, b]\), on each side. As the curve \(\gamma\) is approximated, not interpolated at the points \((x_0, \ldots, x_n)\), this suggests, that we may use the error in this approximation as a refinement condition. This method depends on a condition applied between segments, similar to condition (2.11a).

Geometrically, we require that the quasi-interpolant be sufficiently close to all data points.

Specifically, the method computes the error of the QI at each data point \(x_i\),

\[
\varepsilon_i := \|QI(t_i) - x_i\|,
\]

(2.15)
and splits the intervals \([t_{i-1}, t_i]\) and \([t_i, t_{i+1}]\) if the user-specified tolerance is exceeded.

The adaptive method for rendering a parametric curve based on the error of cubic Quasi-Interpolation spline we call Adaptive Quasi-Interpolation or AQI.

### 2.4 Numerical Tests of the Proposed Tools

In Section 2.6 we will test the algorithms using invariant manifolds, but by performing our initial numerical experiments on explicit parametric curves, we can gain a bit more control over the testing process and gain a clearer understanding of the behaviors of the methods.

#### 2.4.1 A Model Curve

We introduce a model curve with which we test our algorithms, given in polar coordinates as \(r = r(t)\) and \(\theta = \theta(t)\):

\[
\gamma_{\text{test}} = \{(r, \theta) : r = 1 + \epsilon(3t + \cos t + \cos \sqrt{2}t), \theta = \frac{\pi}{2}(\sin t + \sin \sqrt{2}t), a \leq t \leq b\},
\]

(2.16)

and is shown in Figure 2.12.

![Figure 2.12](image.png)

**Figure 2.12** The test curve for \(\epsilon = 10^{-2}, 0 \leq t \leq 15\).

The curve has portions that are nearly circular, connected by regions where the curve makes a sharp turn with large curvature. The curvature of the test curve at the hairpin turns may be increased as desired by adjusting the parameter \(\epsilon \ll 1\). We consider
the curve a “model” of unstable manifolds in the sense that it has regions of both large and small curvature, alternating somewhat unpredictably, and is non-self-intersecting. Since the nearly circular portions of this curve lie very close together, it can be used to test modifications to the algorithm to prevent self-intersection of the approximate curve.

2.4.2 A Visual Test and Motivation for Improved Methods

We start with a visual test of the flatness refinement condition with the Catmull-Rom spline based on three-point central differences on the test curve $\gamma_{\text{test}}$, (2.16). We tested all of the presented flatness conditions and decided to use condition 1 for further work. For various values of the flatness tolerance we visually checked the graphs containing both the exact and approximate curves to judge how ‘close’ the interpolated curve is to the exact curve. In our initial explorations, we noticed that the adaptive Catmull-Rom spline approximation performs well along the curve except near sharp tips, where the approximation has undesirable wiggles; see Figure 2.13a. The observed “wiggles” correspond to unwanted variation in the curvature of the $C^1$ Catmull-Rom spline and suggest an improvement. An improved method, which we call $\text{ACR3}+$, has two steps

1. use the $\text{ACR3}$ method to generate the points along the curve,

2. take the model curve to be the natural cubic spline\(^2\) interpolating these points.

The curvature of the natural cubic spline is a continuous function (the curve is $C^2$) in parametric space, whereas the curvature of the $C^1$ Catmull-Rom spline has jump discontinuities. Note the improvement in Figure 2.13b. We may apply this procedure to any of our methods, i.e., we may first use this method to generate the points, and then use cubic interpolation between the points. Such methods will be denoted by a “+” sign at the end of their names. We see in our quantitative comparison, to be discussed in

---

\(^2\)In all numerical tests we use MATLAB’s built-in `spline` command by default, which uses the not-a-knot condition at the endpoints and is more accurate than the natural spline. For convenience, we will refer to this as a natural cubic spline.
more detail below, that ACR3+ improves slightly over ACR3, but that ALI+ is nearly as good.

![Figure 2.13](image)

**Figure 2.13** The exact curve (red dashed) and two different spline interpolants on the knots from the adaptive Catmull-Rom method, part of the \( \gamma_{\text{test}} \) curve with \( \epsilon = 10^{-2} \).

Figure 2.15a gives a convenient summary of the performance of the various methods, but does not allow us to understand the cause of the disappointing performance. For this we must understand where large errors occur. Figure 2.14a shows the local error \( |\varepsilon(t)| \) in both the ACR3 (Catmull-Rom with 3-point centered differences) and ACR5 (Catmull-Rom with 5-point centered differences) methods after one run with fairly large tolerance. While both curves satisfy the flatness condition for all values of \( t \), the error in the approximation varies widely. Figure 2.14b shows the difference between the approximations. Note, that the error in both approximations is large exactly for values of \( t \) where the two approximations disagree with each other. These figures allow us to both diagnose and correct the problem with the ACR3 method. The flatness condition measures a property of the interpolating curve itself, one that clearly is not well-correlated with the interpolation error. The difference between the two approximations correlates better with the error and makes a more effective refinement condition. This motivates the ACR3vs5 method introduced in Subsection 2.3.2.2.
2.4.3 Quantitative Comparison of Methods

In this section we describe the initial numerical tests performed to demonstrate the superiority of the ACR3 and ACR3+ methods over ALI. Additionally, to make a fair comparison between them, we introduce one more method. In Figure 2.15a, we plot the maximum error $\varepsilon = \max_{t \in [0,10]} |\gamma_{\text{exact}}(t) - \gamma_{\text{approx}}(t)|$ as a function of the number of points used in each computation as the tolerance is decreased. The error is computed at one hundred points between every two interpolated points and the approximate maximum taken as the maximum over this finite set.

First, it is clear that ALI+ converges much faster than ALI and that ACR3+ provides a more modest improvement over ACR3. Our discussion of Figure 2.13 shows that although this improvement might be rather small quantitatively, it can provide significant qualitative improvement. A common technique used in evaluating CAGD methods is to plot the curvature as a function of the parameter; see [14, Ch. 9]. A “good” method would be one in which the computed curvature does not contain unnecessary oscillations. Although we do not provide such a plot here, Figure 2.13 makes it clear that ACR3+ would be seen to be better in this case.
Figure 2.15  (a) Convergence of the error for the adaptive methods with different refinement conditions changing as indicated, between the values at the opposite ends of each graph. The methods tested on the test curve with $0 \leq t \leq 10$ and $\epsilon = 10^{-3}$. Comparison of the ALI+ method (b), the ACR3+ method (c) and the ACR3vs5 method (d) on one sharply pointed segment of the curve for $t \in (7.57, 7.82)$, the error approximately at the same $10^{-8}$ level indicated by boxes in subfigure (a). The curvature at this point is about $1.1 \times 10^5$.

It is also clear from Figure 2.15 that ACR3 converges much more rapidly than ALI, but that the advantage of ACR3+ over ALI+ is far less dramatic. Nonetheless, the improvement is significant. On Figure 2.15a, we have drawn small boxes over the points corresponding to the largest tolerance for which the maximum error falls below $10^{-8}$, for the ALI+ and ACR3+ (and ACR3vs5) methods. The computation using ALI+ required 22125 total points while that using ACR3+ required only 5838 points. To illustrate the origin of this savings, we show in Figure 2.15b and 2.15c the points
computed by the two methods in the neighborhood of one hairpin turn. ALI+ has required 4756 points in this neighborhood, to 34 for ACR3+.

We test both methods ACR3vs5 and ACR3vs5+ on the test curve (2.16) along with the previous method ALI, ALI+ and ACR3 and ACR3+.

The two red lines in Figure 2.15 shows the convergence of the two error-refinement methods. The method ACR3vs5 converges faster than ALI, ALI+, ACR3 and ACR3+. This improvement leads us to expect that applying method ACR3vs5 to computing invariant manifolds should lead to similar results. Note that ACR3vs5+ performs no better than ACR3vs5.

We implemented the $C^2$ AQI methods of Subsection 2.3.2.3 hoping to achieve notable improvement over previous $C^1$ methods. Even though Figure 2.15a shows that performance of AQI is better than a few previous methods, it is just slightly better than the ACR3vs5, the method based on the error refinement condition. We decided not to use AQI for further work, because it did not show itself better than ACR3vs5.

2.5 Details of the Numerical Implementation: One-dimensional Invariant Manifold Calculation

2.5.1 Inductive Parameterization

Assuming the initial primary segment $U_0 = \{U_0(t)|0 \leq t \leq 1\}$ has been given some parameterization (which is discussed below), let $U_1(t) = f(U_0(t-1))$ for $1 \leq t \leq 2$. Then by mathematical induction, we can define $U_n(t)$ in terms of $U_{n-1}(t-1)$ and thus parameterize the whole curve.

We find the following example of the inductive parameterization both instructive and useful in the problem of computing unstable manifolds. Consider a linear map from $\mathbb{R}^2 \to \mathbb{R}^2$ in the form $\mathbf{x}' = A\mathbf{x}$ with a hyperbolic fixed point at the origin. Let $\lambda$ be the unstable eigenvalue and $\vec{v}$ be its associated eigenvector, and let $\mathbf{x}_0 = c\vec{v}$, $W^u$ is of course equal to $\text{span}\{\vec{v}\}$. The initial primary segment of the unstable manifold can be written as
\( U_0 = W^u[x_0, \lambda x_0] \). If this segment is parameterized by \( U_0(t) = \lambda^t x_0 \) for \( 0 \leq t \leq 1 \), then the algorithm described above extends this parameterization inductively to all segments \( U_n \) and thus to the whole branch of \( W^u \).

This example suggests a way to parameterize the initial primary segment \( U_0 \). Near the fixed point (which we can assume is \( x^* = 0 \)), the map \( f(x) \) is approximately given by \( x' = Df(0)x \) so we can approximate \( U_0 \) by an appropriate segment of the unstable eigenspace using the above parameterization. We discuss a nonlinear correction to this approach in Subsection 2.5.3.

Another motivation for the inductive parameterization can be found in the parameterization method, a rigorous analytical method of computing invariant manifolds of hyperbolic fixed points described in Subsection 2.3.1.4.

### 2.5.2 Notation

Let \( U_n \) be the numerical approximation to the \( n \)th true segment \( U_n \). This approximation is defined as the Catmull-Rom curve interpolating the points in the set

\[
\mathcal{X}_n = \{x^n_k\}_{k=0}^{N_n}
\]

at the corresponding parameter values in the set

\[
\mathcal{T}_n = \{t^n_k\}_{k=0}^{N_n}.
\]

In particular, we can define the operator \( \Gamma_{\text{CR}} \) that maps a set of parameter values and the associated interpolation points to their Catmull-Rom interpolant, i.e.,

\[
U_n = \Gamma_{\text{CR}}(\mathcal{T}_n, \mathcal{X}_n).
\]

Thus we may construct the approximate invariant manifold using an inductive procedure, with two obvious steps. First, to construct the initial segment \( U_0 = \Gamma_{\text{CR}}(\mathcal{T}_0, \mathcal{X}_0) \), which must satisfy the chosen refinement conditions. Second, given \( U_n \), to find suitable
sets $T_{n+1}$ and $X_{n+1}$, which define an approximation $U_{n+1} = \Gamma_{CR}(T_{n+1}, X_{n+1})$ which satisfies the same refinement condition. In order to make this construction, we introduce two new pieces of notation:

$$T_{n+1} = \{ t_k^n + 1 \}_{k=0}^{N_n} \text{ and } f(X_n) = \{ f(x_k^n) \}_{k=0}^{N_n}.$$ 

We call a segment resolved if it satisfies the adaptive refinement condition being used for the particular algorithm.

Consecutive primary segments are joined together so that the last point in $U_n$ is the first point in $U_{n+1}$, i.e., $x_n^1 = f^n(x_0^0)$. Subsequent primary segments can be found in the same way recursively. According to the inductive parameterization from Subection 2.5.1, all values $t_k^n \in T_n$ are drawn from the interval $[n, n+1]$.

### 2.5.3 The Initial Primary Segment

In order to initialize any adaptive method for computing an invariant manifold, one first has to determine the initial primary segment $U_0$ that approximates the true segment $U_0$, as is described in Section 1.2. The easiest and the most commonly used method is to choose the initial primary segment as a line segment from the unstable subspace. It is crucial to determine $U_0$ very accurately in order to avoid error propagation in the further computations. Therefore, this line segment must be taken very close to the fixed point. This approach is not practical when we are interested in computing a fairly long portion of the manifold. Setting the initial primary segment closer to the fixed point requires many more iterations to compute the same portion of the manifold. In order to be more efficient, the initial primary segment should be taken further from the fixed point, using higher-order approximation. Accordingly, $U_0$ is defined as a polynomial expansion of the manifold close to the hyperbolic fixed point.

We consider the map of the form $x' = f(x)$ as described in Section 1.2, where $x = (x, y), f : \mathbb{R}^2 \to \mathbb{R}^2$ and the prime denotes the forward mapping. The map can be
written as
\[ x' = f_1(x, y) \]
\[ y' = f_2(x, y). \]

Near the hyperbolic fixed point \((x^*, y^*)\) a branch of invariant manifold can be explicitly written as \(y = p(x) = \sum_{k=0}^{\infty} c_k (x - x^*)^k\). The invariance of the manifold under the map \(f\), yields the algebraic relation
\[ f_2(x, p(x)) = p(f_1(x, p(x))). \]

Solving this recurrence yields \(c_0 = y^*\) and two possible values for \(c_1\), corresponding to the stable, \(W^s\) and the unstable, \(W^u\) manifold. Once \(c_1\) is chosen, the remaining coefficients can be found uniquely in ascending order from relation (2.17). The single branch of the manifold near the hyperbolic fixed point \((x^*, y^*)\) can be approximated by \(y = p_N(x)\), the truncation of the series to the \((x - x^*)^N\) term. Taking \((x_0 - x^*)\) small enough and \(N\) large enough that
\[ |f_2(x_0, p_N(x_0)) - p_N(f_1(x_0, p_N(x_0)))| < \varepsilon \ll 1, \]
guarantees that the error between the polynomial approximation \(U_0\) and the true manifold \(U_0\) along this segment is less than \(\delta\). The same approach is used by Hobson in [32]. Moreover, our further numerical tests (not shown here) show that this is necessary in order to avoid error propagation and, especially, to smoothly join two segments \(U_{n-1}\) and \(U_n\).

We want to generate the sets \(X_0\) and \(T_0\) in order to construct an accurate approximation \(U_0\). Choosing \(x_0^0\), the \(x\)-coordinate of the first point of the initial segment, the \(y\)-coordinate \(y_0^0\) is determined by the expansion \(p(x)\) as \(y_0^0 = p(x_0^0)\). According to the definition of \(U_0\) the last point of the segment is determined by the choice of the first point and its image \((x_{N_0}^0, y_{N_0}^0) = f(x_0^0, y_0^0)\).
Having the first and the last point of the initial primary segment we generate the set of intermediate points. Let $I_0$ be the interval of the $x$-coordinates between $x^0_0$ and $x^0_{N_0}$. This is parameterized by the inductive parameterization introduced in Subsection 2.5.1 where the eigenvalue of $Df(0)$ is replaced by a nonlinear correction $\lambda = \frac{x^0_{N_0} - x^*}{x^0_0 - x^*}$. Then any point of the $U_0$ has the $x$-coordinate in a form $x^0_i = x^* + \lambda t_i (x^0_0 - x^*)$ where $t_i \in [0, 1]$ is a parameter value. The $y$-coordinate of the points are given by the manifold expansion as $y^0_i = p(x^0_i)$. The complete set of points with their parameter values determines a discrete representation of the initial primary segment. In this way we define the sets $T_0$ and $X_0$, which gives $U_0 = \Gamma_{CR}(T_0, X_0)$. In order to obtain an accurate interpolant at later steps, we found it necessary to use a large number of points, about 50.

### 2.5.4 Resolving a Simple Primary Segment

We restrict our attention the specific problem of finding the unknown primary segment $U_{n+1}$ which is the approximate image under the map $f$ of an already-resolved primary segment $U_n = \Gamma_{CR}(T_n, X_n)$, i.e., $U_{n+1} \approx f(U_n)$. Mapping the segment $U_n$ forward yields a parametric curve $\tilde{U}_{n+1} = \Gamma_{CR}(T_n + 1, f(X_n))$ which approximates the next primary segment $U_{n+1}$. As the dynamics is expanding along the manifold, we expect that a larger set of points should be needed to resolve the segment $U_{n+1}$ than was required for $U_n$. The curve $\tilde{U}_{n+1}$ will in general be unresolved, but will indicate the general shape of $U_{n+1}$.

Having already generated the initial guess $\tilde{U}_{n+1}$ we may apply any of the proposed adaptive method from Section 2.3.2 to produce a resolved approximation $U_{n+1}(t) = \Gamma_{CR}(T_{n+1}, X_{n+1})$ to the desired parametric curve $f(U_n)$. Note that by construction, the number of points on consecutive segments satisfy $N_{n+1} \geq N_n$.

### 2.5.5 Kink Patching

The proposed methods are quite capable of resolving a single primary segment of a manifold given a previously completed segment. Segments $U_n$ and $U_{n+1}$ may be individually
resolved but the composite curve formed by their union $U_n \cup U_{n+1}$ may be unresolved in the neighborhood of their common point. We call a point where this occurs a kink\(^3\). Therefore, some care must be taken to ensure that the composite curve formed by two successive primary segments is also well-resolved around that point. This can be accomplished by examining a small set of consecutive points centered about the joint point.

Consider the kink between already-resolved segments $U_n$ and $U_{n+1}$. The points before the joint are taken from the end of the discrete representation of $U_n$ and require preimages in $U_{n-1}(t)$, the points after are from the beginning of $U_{n+1}$ and require preimages in $U_n(t)$. The inductive parameterization presents a clear advantage in searching for preimages.

Assume that a subsegment of $U_n$ has to be refined between points $x^n_i$ and $x^n_{i+1}$. The parameter values corresponding to that pair are $t^n_i$ and $t^n_{i+1}$. Their preimages can be found by evaluating the previous segment approximation $U_{n-1}(t)$ at $t^n_i - 1$ and $t^n_{i+1} - 1$. In order to refine the subsegment of $U_n$ it is sufficient to sample the approximation $U_{n-1}(t)$ on $t^*_s = (t^n_i + t^n_{i+1})/2 - 1$ and map it forward, $x^n_s = f(U_{n-1}(t_s))$. The new point lies on $U_n$ between points $x^n_i$ and $x^n_{i+1}$, the new parameter value for this point is $t^n_s = t^*_s + 1$.

When the joint kink between segments $U_n$ and $U_{n+1}$ is resolved, then we can say that the segment $U_n$ is completed. Even though the segment $U_{n+1}$ is well resolved it may be necessary to refine it again close to the joint vertex with a segment $U_{n+2}$.

The procedure is only slightly different for patching the kink between segments $U_0$ and $U_1$. To refine the end of $U_0$ we just take more points from $U_0(t)$, the continuous approximation of the initial primary segment.

\(^3\)Note, this problem occurs in Carter’s method, as well, but not in Hobson’s.
2.6 Numerical Tests

2.6.1 Example 1: Hénon Map

It is one of the most studied examples of dynamical systems that exhibits chaotic behavior, this map is given by

\[ x' = 1 + y - ax^2 \]
\[ y' = bx. \]

The closure of one branch of the unstable manifold is a well-studied strange attractor that possesses fractal structure. We have chosen this example primarily for the purpose of visualization.

For the standard parameters \( a = 1.4 \) and \( b = 0.3 \) the map has a hyperbolic fixed point at \((x^*, y^*) = (-1.131354477089505, -0.339406343126851)\). In order to determine the initial primary segment \( U_0 \) we approximate the unstable manifold near this saddle point as \( y = p(x) \), where \( p(x) = \sum_{i=0}^{10} c_i (x - x^*)^i \) with \( c_i \) computed from the relation (2.17). The initial primary segment \( U_0 \) is generated by \( x_0 - x^* = 0.0001 \), then \( y_0 = p(x_0) = -0.339397140050439 \). The last point of the initial primary segment is determined by \((x_0^1, y_0^1) = f(x_0, y_0) = (-1.131028508759507, -0.339376343126851)\). The error between this polynomial approximation and the true manifold along this segment is less than machine precision.

Figure 2.16a shows part of the computed unstable manifold of the Hénon Map: the union of primary segments \( U_0 \) through \( U_{20} \). Figures 2.16b and 2.16c present a closeup view of the bounded boxes \( A \) and \( B \) from Figure 2.16a, respectively. They show more detail of three approximations of some sharp segments of the manifold, each approximation derived with a different tolerance condition. Additionally, the computed manifold is rotated in Figure 2.16b and 2.16c and the aspect ratio is not preserved. These show the behavior of the method close to slanted hairpin turns.
For computations of real unstable manifolds using methods based on linear interpolation, a concentration of points near maxima of curvature becomes a crippling fault. Near these hairpin turns the ALI method tends to place a very large number of points. For example, the unstable manifold of the Hénon Map develops folds where the curvature reaches $O(10^6)$. We found that ALI with such small tolerance places points so close together (e.g., see Figure 2.15b) that, in 16-digit arithmetic, there are not enough significant digits remaining to meaningfully compute the angle between successive segments of the computed curve, so that the computation just tries to bisect again, it enters an infinitely recursive loop and stalls. See also Figure 2.15 and 2.18b.
2.6.2 Example 2: McMillan Map

Next, we consider the McMillan Map defined by

\[
x' = y \\
y' = -x + 2y \left( \frac{\mu}{1+y^2} + \varepsilon \right)
\]

The map is completely integrable at \( \varepsilon = 0 \). For \( \mu = 2.0 \) and \( \varepsilon = 0.05 \) it has a saddle fixed point at the origin. We approximate the unstable manifold near the origin as

\[ p(x) = \sum_{i=0}^{10} c_i x^i \]

with \( c_i \) computed using the relation (2.17).

In order to make direct comparison between the proposed methods and those of both Hobson and Carter, we test our example of the McMillan map, choosing the same parameter values and initial segment. The initial primary segment \( U_0 \) is generated by \( x_0 = 0.001 \), so that \( y_0 = p(x_0) = 0.003839548998331 \). The last point of the initial primary segment is determined by \( (x_0^1, y_0^1) = f(x_0, y_0) = (0.003839548998331, 0.014741924483874) \). The error between this polynomial approximation and the true manifold along this segment is below machine precision.

Figure 2.17a shows part of the unstable manifold of the McMillan Map computed using the ACR3vs5 method, given by the union of primary segments \( U_0 \) through \( U_{15} \). Figure 2.17b and 2.17c present a closeup view of bounded boxes \( A \) and \( B \) from Figure 2.17a, respectively. They show more detail of three approximations of some sharp segments of the manifold, each approximation derived with different tolerance condition. These graphs show that the method appears to converge.

2.6.2.1 Proposed Methods vs Hobson and Carter Methods

In order to compare the proposed methods with existing methods we also perform computations with the ALI+ method which is based on the same type of refinement condition as in both Hobson and Carter. Our implementation of ALI is slightly different from Carter's and we put some effort into making sure that ALI performs at least as well as in his studies; see Tables 2.1.
(a) The unstable manifold of the McMillan Map,

(b) closeup of box A,  

(c) closeup of box B.

Figure 2.17  (a) The unstable manifold of the McMillan Map, $\mu = 2.0$ and $\varepsilon = 0.05$ with length about 113.3335, generated by **ACR3vs5**; (b) zoom $A$; (c) zoom $B$ with three approximations, $\text{tol} = 2^{-11}$—blue, $\text{tol} = 2^{-13}$—green, $\text{tol} = 2^{-15}$—red. Note (b) and (c) do not preserve the aspect ratio of (a), for (b) $\text{height} = O(10^{-3})$, $\text{width} = O(10^{-5})$ and for (c) $\text{height} = O(10^{-4})$, $\text{width} = O(10^{-4})$.

In order to test convergence, Hobson considered the convergence of the arc length of an individual primary segment, estimating the length as the sum of the chord lengths between adjacent discrete points. We perform the same test and show the results in Table 2.1, an expanded version of similar tables in [4] and [32]. This test indicates that **ACR3+** seems to converge fastest.

The sum of chord lengths between any neighboring points on the manifold does not converge to the length of the curve connecting them particularly fast. While the length of a cubic Bézier curve segment is not computable in closed form, a result due to Gravesen [22] shows that estimating the arc length by one half the perimeter of the
control polygon converges to the arc length faster than using chord length; at order four as opposed to order two using chord length. Using this method, it is clear that the arc length of our methods converges even more rapidly; see Table 2.2.

**Table 2.1** Manifold calculation comparison using chord arc length of primary segments of the McMillan Map. The bold type values we can call the “exact” lengths of the segments. Values for Hobson’s and Carter’s method follows [4]. Values for ALI+, ACR3+, ACR3vs5 are computed for 10 points at the initial primary segment.

<table>
<thead>
<tr>
<th></th>
<th>Hobson’s Method</th>
<th>Carter’s Method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>tol1</td>
<td>tol2</td>
</tr>
<tr>
<td></td>
<td>0.45 0.01</td>
<td>0.45 0.01</td>
</tr>
<tr>
<td>U₁₄</td>
<td>22.834 23.098</td>
<td>22.975 23.090</td>
</tr>
<tr>
<td>U₁₅</td>
<td>28.407 28.991</td>
<td>28.687 29.024</td>
</tr>
<tr>
<td>Calls</td>
<td>49.75 76.71</td>
<td>113.95 90.81</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Hobson’s Method</th>
<th>Carter’s Method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ALI+</td>
<td></td>
</tr>
<tr>
<td></td>
<td>tol</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.143 0.102</td>
<td>0.0737 0.0555</td>
</tr>
<tr>
<td>U₁₅</td>
<td>29.0154 29.0304</td>
<td>29.0350 29.0366</td>
</tr>
<tr>
<td>Calls</td>
<td>31.26 43.51</td>
<td>58.32 76.64</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Hobson’s Method</th>
<th>Carter’s Method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ACR3+</td>
<td>ACR3vs5</td>
</tr>
<tr>
<td></td>
<td>tol</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2(-9) 2(-12) 2(-14) 2(-16)</td>
<td>2(-13) 2(-18) 2(-20) 2(-23)</td>
</tr>
<tr>
<td>U₁₅</td>
<td>29.0210 29.0357 29.0375 29.0380</td>
<td>29.0234 29.0367 29.0375 29.0380</td>
</tr>
<tr>
<td>Calls</td>
<td>1.666 4276</td>
<td>8397 16649</td>
</tr>
</tbody>
</table>

**Table 2.2** Manifold calculation comparison using Gravesen’s arc length approximation of primary segments of McMillan Map. The values are computed for given value of the tolerance with 10 points at the initial primary segment.

<table>
<thead>
<tr>
<th></th>
<th>ACR3+</th>
<th>ACR3vs5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>tol</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2(-8) 2(-9) 2(-10) 2(-13)</td>
<td>2(-7) 2(-10) 2(-13) 2(-17)</td>
</tr>
<tr>
<td>U₁₅</td>
<td>29.0286 29.0359 29.0383 29.0381</td>
<td>29.0167 29.0400 29.0375 29.0381</td>
</tr>
<tr>
<td>Calls</td>
<td>1.190 1.666</td>
<td>2.219 6.246</td>
</tr>
</tbody>
</table>

We do not believe that the arc length convergence is the best way to show the convergence of the method. Approximations to individual segments of the curve may approach the same length without lying close together.
2.6.3 More Direct Convergence Tests

Figure 2.17b and 2.17c show that the \textbf{ACR3vs5} method appears to converge. However, to demonstrate this quantitatively we perform further tests.

In order to show convergence of the proposed methods we use high-order interpolation to generate continuous representations of the manifold for a very small value of the refinement condition; we call this an “exact” manifold. Next, we generate continuous representations of the manifold for several decreasing values of the refinement condition and check how the difference between the approximation and the “exact” manifold, measured using the maximum ($L^\infty$) norm, decays. Note that this is the same test as we performed in Section 2.4 for the model problem; see Figure 2.15.

The initial primary segment $U_0$ is generated as above and the first 15 primary segments are computed. Figure 2.18a shows the convergence of the two proposed methods, \textbf{ACR3vs5} and \textbf{ACR3+}. Additionally, as a comparison, the figure presents convergence of \textbf{ALI+}, described in previous paragraph, the method based on the angle refinement condition.

![Graph (a) Test on the McMillan Map. The “exact” manifolds for \textbf{ALI+}, \textbf{ACR3+} and \textbf{ACR3vs5} with tolerances $2^{-9}$, $2^{-22}$ and $2^{-32}$, respectively.]

![Graph (b) Test on the Hénon Map. The “exact” manifolds for \textbf{ALI+}, \textbf{ACR3+} and \textbf{ACR3vs5} with tolerances $2^{-8}$, $2^{-21}$ and $2^{-30}$, respectively, without reparameterization.]

\textbf{Figure 2.18} Convergence of the adaptive methods \textbf{ACR3vs5} (blue), \textbf{ACR3+} (red) and \textbf{ALI+} (green) with decreasing refinement conditions changing as indicated, between the values at the opposite ends of each graph; 50 points on the initial primary segment.
We perform the same test for the Hénon map. We use the same initial primary segment $U_0$ as above, and compute the first 16 primary segments, (Figure 2.16a contains 20 segments). Figure 2.18b shows that both methods converge very well and at a similar rate. The error between approximation and the “exact” manifold with about $10^5$ points is on the order of $10^{-8}$. The ALI method fails to converge for $\text{tol} < 2^{-8}$ with 16 iterates of the map. Using this tolerance, the same algorithm also failed to converge on the 17th iterate, as discussed in the final paragraph in the discussion of example 1.

Figures 2.18a and 2.18b show that both proposed methods ($\text{ACR3vs5}$ and $\text{ACR3+}$) converge faster than $\text{ALI+}$. The method $\text{ACR3vs5}$ seems to work slightly better than the $\text{ACR3+}$. However, for a few values of tolerance the second method outperform the first.

We encountered a difficulty in measuring the error in the unstable manifold calculations not present in the parametric curve convergence studies of Section 2.4. Since we do not know the true manifold $W^u$, we compute the error by calculating the distance between two approximations as $\sup_{t \in [t_k, t_{k+1}]} ||\gamma_{\text{approx}}(t) - \gamma_{\text{exact}}(t)||$, i.e., we monitor the difference between the two curves evaluated at the same “time” $t$. Because the map is expanding in the direction tangent to $W^u$, however, calculations with different values of the tolerance lead to slight changes in the parameterization of the curve. Plotting, say, the $x$-coordinate as a function of the parameter $t$, we find that the two graphs look the same but are shifted slightly along the $t$-axis. We call this phenomenon parameterization slip and, although it is the largest source of our computed error, it does not correspond to an actual error in the computation of the manifold. It is illustrated in Figure 2.17c: the points marked c and d correspond to the same parameter value, obtained using different values of the tolerance. The calculated error represents the distance between these two points, which is clearly much larger than the distance between the two curves in this neighborhood.
The results in Figure 2.18a and 2.18b are slightly different than the results which we see in Figure 2.15a for the model problem. However, computation of these manifolds is more complex than the simple problem of drawing a parametric curve. The accuracy of that computation depends only on the interpolation error, whereas the adaptive method for computing of invariant manifold has several sources of error. The convergence of the method can be affected by the interpolation error and by the amount by which this error and round-off errors are amplified by sensitive dependence on initial conditions. Much of the observed error is actually due to parameterization slip. The error in the normal direction often appears to be significantly smaller. In the last case, we can partially solve the problem of parameterization slip by chord length reparameterization. However, our numerical tests shows that this does not always help. The reparameterization improved the convergence for the McMillan map, whereas for the Hénon map it did not.

2.6.4 Example 3: Map with an Explicit Manifold

In this section, we test the methods on a map constructed explicitly to have an unstable manifold that is computable in closed form. \( W^u = \{(x, y) | y = h(x)\} \). First, since \( y \) is an explicit function of \( x \), we avoid the “parameterization slip” and its effect on the error. Second, we are able to compute the exact error between the true and computed manifolds, rather than the distance between two approximations. Note that the errors reported in this section refer to distance in the \( y \) direction rather than Euclidean distance. A similar test on an explicit manifold is performed in [49].

We consider example for which the unstable manifold can be found explicitly, namely

\[
\begin{align*}
\dot{x} &= x \\
\dot{y} &= -y + g(x),
\end{align*}
\]
with explicit solution
\[
\begin{align*}
x(t) &= c_1 e^t \\
y(t) &= c_2 e^{-t} + \frac{G(c_1 e^t)}{c_1 e^t},
\end{align*}
\]

where the function \( G(x) \) satisfies \( G'(x) = g(x) \). We assume that \( g(0) = 0 \), i.e., \( G'(0) = 0 \), this automatically ensures that the origin is a saddle point with associated stable and unstable manifolds tangent to the \( x \) and \( y \) axis, respectively. Note from (2.19) that trajectories of the system can be explicitly written in the form
\[
y = \frac{c_1 c_2 + G(x)}{x}.
\]

Note that assuming the explicit unstable manifold is tangent to the \( y \) axis at the origin requires \( y(0) = y'(0) = 0 \). Assuming \( c_1 c_2 = 1 \), the function \( G(x) = -\cos(x^2 e^{x^2}) \) satisfies all the conditions above and guarantees existence of the explicit unstable manifold of the form
\[
y = \frac{1 - \cos(x^2 e^{x^2})}{x};
\]

see Figure 2.19a. In addition, this manifold is significantly complex to provide a challenging test for the various algorithms.

In order to test the proposed methods we need to find the planar map associated with the system (2.18). For any flow \( \varphi^t(x, y) \) given by the solution of a dynamical system the planar map can be written as
\[
(x', y') = f(x, y) = \varphi^T(x, y),
\]

where \( T \) is a constant. Consider the flow \( \varphi^t(x, y) \) given by (2.19) with \( T = \ln 2 \) then the planar map takes the form
\[
\begin{align*}
x' &= 2x \\
y' &= \frac{xy + G(2x) - G(x)}{2x},
\end{align*}
\]
where the function $G(x)$ is as defined above.

We apply each of the proposed methods to the map (2.21) to generate the rightgoing branch of the unstable manifold; see Figure 2.19a. Next we investigate behavior of the maximum error between the exact manifold and each approximation for decreasing refinement condition.

![Figure 2.19](image)

(a) The unstable manifold. (b) Convergence of the three methods.

Figure 2.19 (a) The explicit unstable manifold of the system (2.18) given by the Equation (2.20), (b) convergence of the error between the manifold and approximations given by adaptive methods ACR3vs5 (red), ACR3+ (green) and ALI+ (blue). The test is performed on a portion of the branch with length about 255, the initial primary segment starts at $x = 0.0011$ with 10 points and computes the first 11 primary segments. The decreasing refinement conditions are as indicated, between the values at the opposite ends of each graph.

Figure 2.19b shows the convergence of the error for each method. Both methods, ACR3+ and ACR3vs5, perform very well (here, it appears ACR3vs5 converges slightly faster), whereas the ALI+ method does poorly. The test above confirms our previous results for the proposed methods. The question which method, ACR3+ or ACR3vs5, for computing unstable manifold is better in general remains open.

### 2.7 Discussion

The methods presented here for computing one-dimensional unstable manifolds incorporate ideas from computer aided geometric design and achieve significant improvements
in the accuracy of the calculation, and in reducing the number of calls to the function $f$. This is especially relevant if one wants to resolve all the sharp folds typically found on such manifolds. For the mere display of the manifold on a screen, resolving points where the curvature reaches such maxima would be excessive, but for computations involving the manifolds, such as estimating the dimension of the attractor dimension as suggested by Hobson in [32], or calculating their intersections in order to apply the theory of homotopic lobe dynamics [47], this would be necessary.

We should point out, however, that there are other costs involved with the implementation of this method, in comparison, say, with the simpler method of Carter in [4]. In particular, one segment of a piecewise linear interpolant can be plotted using only its values at the two endpoints, with high-level graphing software filling in the points in between. To effectively render a cubic interpolant, one first samples the curve at a finite number of points in between those interpolated, and then plots a linear interpolant through those points. By choosing these points adaptively, one can plot the piecewise-cubic curve to the desired degree of accuracy, but the overhead of this adaptive calculation makes the procedure more expensive than simply evaluating the interpolant a finite number of equally spaced parameter values. Here, too, one has the choice to either choose a large enough number of points to guarantee this will accurately compute the curve, or else to estimate the necessary number of points using a bound based on the derivatives of the interpolant; see, for example, references [6, 15].

The method has some additional advantages over methods based on linear interpolation. To analyze transport in a chaotic system, one needs to construct regions bounded by segments of the stable and unstable manifolds, and thus must determine intersections of $W^s$ and $W^u$. There are two steps to this process. Given piecewise interpolants to the two curves, one first compiles a list of pairs segments whose bounding boxes overlap and which might intersect. Second, one tests each of these pairs for an intersection. In the second step, it is simpler to detect intersections between linear interpolants (solving a
small system of linear equations), although many efficient methods exist for calculating the intersections of Bézier curves, for example [53, 55]. It is the first step that takes the most time however. Given $n$ segments, there are $n(n - 1)/2$ pairs of segments to check. Since our method greatly reduces $n$, finding the candidate pairs will take significantly less time.

While discussing manifold intersections, we should also point out that, as the map $f$ is invertible, $W^u$ is topologically forbidden from self intersecting. A more complete code should also check for self-intersections and refine the manifolds in their neighborhoods. Due to the fractal nature of the strange attractor, arbitrarily many additional segments of the manifold pass within any small neighborhood of any point on $W^u$. Because of the finite errors in the method, and the arbitrarily small spacing between segments, a computed manifold based on any refinement conditions of the type discussed will possess erroneous self-intersections.
CHAPTER 3

HIGH-ORDER ADAPTIVE METHOD FOR COMPUTING
TWO-DIMENSIONAL INVARIANT MANIFOLDS OF MAPS

This chapter is organized as follows. In Section 3.1, we show existing methods, the parameterization method, and the method due to Krauskopf and Osinga. Section 3.2 gives a description of the tools from CAGD that are crucial parts of the presented algorithm. Topics covered include triangulation and triangular Bernstein-Bézier patches, triangulation-based surface approximation schemes such as bivariate linear interpolation, nine-parameter interpolation, Clough-Tocher and Shirman-Sequin interpolating schemes, a quasi-interpolation method based on quartic Bernstein-Bézier patches and its adaptive version. In Section 3.4, we perform numerical tests of the 3D geometric tools introduced earlier, especially quasi-interpolation based on quartic Bernstein-Bézier patches and its adaptive version. Section 3.5 contains a description of the implementation of our introduced tools in the context of computing two-dimensional invariant manifolds. In Section 3.6, we perform various numerical tests showing convergence of the method. Section 3.7 contains a summarizing discussion of the advantages of the algorithms for computing one-dimensional invariant manifolds presented here.

3.1 Existing Methods

Before presenting our own methods, we review two existing, widely cited methods suited for computing two-dimensional manifolds of maps. The first one is the two-dimensional version of the parameterization method described in Subsection 2.3.1.4. The second method, by Krauskopf and Osinga, is a generalization of their method, briefly described in Subsection 2.3.1.3, to the two-dimensional unstable manifold computation.
3.1.1 Parameterization Method

A local approximation of the manifold close to the fixed point can be constructed based on the rigorous, analytical method of computing invariant manifolds of hyperbolic fixed points, e.g., [1, 2, 3, 16]. The parameterization method is the basis for the recent computations by Mireles-James and Lomelí [44, 45].

Assume that map \( f \) at a fixed point \( x^* \) has a pair of real, distinct eigenvalues \( \lambda_1, \lambda_2 \) of \( Df(x^*) \) with associated eigenvectors \( \vec{v}_1 \) and \( \vec{v}_2 \). The pair of the eigenvalues satisfies \( |\lambda_1|, |\lambda_2| < 1 \). The parameterization method for the two-dimensional manifolds of map is based on power series expansion in the form

\[
P(s, t) = \sum_{n=0}^{\infty} \sum_{m=0}^{\infty} a_{m,n} s^m t^n,
\]

where the coefficients \( a_{m,n} \in \mathbb{R}^3 \) satisfy

\[
a_{0,0} = x^*, \ a_{1,0} = \vec{v}_1 \ a_{0,1} = \vec{v}_2.
\]

The remaining coefficients are determined by substituting of the series (3.1) into the invariant equation

\[
(f \circ P)(s, t) = P(\lambda_1 s, \lambda_2 t),
\]

and solving the recurrence relation.

If a fixed point \( x^* \) of a map \( f \) has a complex conjugate pair of eigenvalues \( \lambda \) and \( \bar{\lambda} \) with associated pair of eigenvectors \( \vec{v} \) and \( \bar{\vec{v}} \) then the procedure is slightly different. It is useful to consider the power series for \( P \) in the form

\[
P(s, t) = Q(s + it),
\]

where \( Q \) is a complex series of the form

\[
Q(z) = \sum_{n=0}^{\infty} \sum_{m=0}^{\infty} z^m \bar{z}^n q_{m,n}
\]
and \( q_{m,n} \in \mathbb{C}^3 \). Assuming that \( q_{m,n} = \overline{q_{m,n}} \) guarantees that, if \((s, t) \in \mathbb{R}^2\) then the series

\[
P(s, t) = \sum_{n=0}^{\infty} \sum_{m=0}^{\infty} (s + it)^m (s - it)^n q_{m,n}
\]

or in polar coordinates for \( s = r \cos \theta, t = r \sin \theta \)

\[
P(r, \theta) = \sum_{n=0}^{\infty} \sum_{m=0}^{\infty} r^{m+n} e^{i(m-n)\theta} q_{m,n}.
\]

defines a map \( P : \mathbb{R}^2 \rightarrow \mathbb{R}^3 \). In a similar manner as before \( q_{0,0} = x^*, q_{1,0} = \mathbf{v} \) and the remaining coefficients \( q_{m,n} \) are determined by the invariance equation

\[
(f \circ P)(s, t) = P \circ E_{\lambda}(s, t),
\]

where \( E_{\lambda} \) is the \( 2 \times 2 \) Jordan block associated with the pair of stable or unstable eigenvalues.

This method provides an explicit formula to compute a parametric form of the manifold. The power series representation of the manifold has infinite radius of convergence, but, in practice, due to roundoff error, the numerical radius of convergence may be quite short. Even though the method is very accurate near the origin a complementary method is needed in order to compute larger portion of the manifolds. The parametric representation of the local manifold is used as an initial step of our proposed method.

### 3.1.2 The Krauskopf-Osinga Method

In [35] Krauskopf and Osinga present a method for computing the two-dimensional unstable manifolds of a hyperbolic fixed point of a discrete three-dimensional dynamical system given as a map. They drop the idea of iterating a fundamental domain and instead they “grow” the manifold at the same rate in each direction. The algorithm computes one-dimensional intersection curve of the unstable manifold with a finite number of two-dimensional leaves of a linear foliation. The linear foliation is a set of planes generated by the fixed point, the stable eigendirection (a one-dimensional vector) and
a set of points on a small circle in the unstable eigenspace (a two-dimensional plane) around the fixed point; see Figure 3.1a. Each leaf contains the fixed point, the stable eigendirection and one of these points from the unstable eigenspace. The foliation looks like an open book where the fixed point and the stable vector are contained in its spine, and the pages are the leaves of the foliation. The algorithm grows the manifold in circles starting with an initial circle on the unstable eigenspace. A growing portion of the manifold is represented by a planar triangular mesh.

Their attempt to grow uniformly is intended to alleviate the exponentially anisotropic growth if $1 < |\lambda_1| < |\lambda_2|$. It is designed to deal with the problem that fundamental annuli grow at different rates dependent on direction. However, it has some shortcomings which we describe below. Two aspects of this method can cause it to fail.

1. Its dependence on the foliation breaks down if the manifold is ever tangent to a leaf.

2. If the map has rotation, far points may map to near points. Points on a given ring that we intend to compute may depend on points outside that ring which are not yet computed; see Figure 3.1b.

Point 1 is acknowledged by Krauskopf and Osinga in [35]. Point 2 is not.

The main step of the algorithm is to add a new discrete circle at a suitable distance from the last computed discrete circle. For each mesh point on the outermost circle they determine a new point on the manifold in this plane at a given distance from the previous point. The portion of the manifold is defined by a sequence of discrete points at each of the leaves that have prescribed distance from each other. Connecting the points along the leaves and along the same discrete circle, the algorithm derives a quadrilateral mesh which is easily modified to a triangular by inserting diagonals.

The procedure of adding a new point on the leaf is fairly straightforward. The new point at distance $\Delta$ from the last point is found by finding its preimage in the already
computed manifold mesh. In practice the algorithm needs to find the triangle $T$ that contains this preimage and then find it by bisection in $T$. Note that the method still requires a triangulated approximation of the primary fundamental annulus as an initial step.

\[ E^{\alpha}(x^*) \]

\[ E^{\beta}(x^*) \]

\[ \begin{array}{c}
\text{(a)} \\
\text{(b)}
\end{array} \]

**Figure 3.1** (a) Initial setup for the Krauskopf-Osinga method, (b) the cases when the method fails, geodesic circles – black, primary fundamental annuli – red.

The algorithm is only capable of constructing a single connected component of the intersection of $W^u$ with a given leaf. If there exists an additional component, the algorithm will miss it. Such behavior must be preceded by a non-transverse intersect at which point the algorithm quits.

The method should work when the fixed point has eigenvalues with largely differing modulus, as well as in the case of complex conjugate eigenvalues. However, our experiments with this method show that this is not true in general and the method can fail or produce a manifold of poor quality. For example, it may happen that the preimage of a point the algorithm is trying to compute possesses geodesic distance to the fixed point greater than the point itself, and thus is not in the pre-computed set; see Figure 3.1b. This is a common behavior of the nonlinear maps and it can occur not only in the neighborhood of the fixed point. The behavior is easily seen in maps with a rotation component. For instance, this is one of the reasons why the method fails for maps on which we test our proposed method in Section 3.6. Moreover, their method is based on
a plane triangular surface interpolation which thus has only quadratic accuracy. Using higher-order interpolation should significantly improve computations.

### 3.2 3D Geometric Modeling Tools

Motivated by (1.2), again, we delay considering the problem of computing a two-dimensional manifold and instead focus on the simpler problem of approximating a parametric surface. Here we introduce a few tools from CAGD that we use to construct the numerical method.

#### 3.2.1 Parametric Surface

Consider a surfaces defined by a parametric representation

\[ S(s, t) = \{ \mathbf{x} \in \mathbb{R}^3 : \mathbf{x} = \mathbf{x}(s, t), (s, t) \in \Omega \subset \mathbb{R}^2 \}. \]

A one-dimensional parametric curve can be sampled on a parametric grid only in two ways, i.e., uniformly or non-uniformly. In case of a parametric surface, a two-dimensional parametric object, it is not so simple. We can recognize several families of tessellations of a two-dimensional parametric space. They can be based on a different regular polygons (regular tessellation), or of their combination (semi-regular tessellation). A tessellation can be also generated by less regular shapes. For example, given a scattered set of points Voronoi tessellation divides the domain into a set of polygonal regions with one point on the interior of each region.

The choice of tessellation depends on the problem which we are facing. It is tied to the kind of data that we are given or able to generate. Typically there are two possibilities: either there is given scattered data set that we need to tessellate, or else we are able to choose the data and then tessellate it. In the second case it seems that uniform sampling would be the most natural choice if the domain region allows it. However, for some of the problems the better choice is an adaptive tessellation, where distribution
of data points is semi-uniform. Normally, the data is tessellated into triangles, which is sometimes referred to as triangulation. In the problem of surface reconstruction, the approximation method strongly depends on the tessellation.

In our work we focus on surface approximation methods based on triangular patches. Modeling surfaces of high geometric complexity triangular-based methods offer greater flexibility, than any polygonal-based method. Because of their local nature they can be used in rendering applications.

3.2.2 Triangulation

Given a set of data points without a priori information on the topology, a triangulation algorithm constructs a triangular mesh using the given points as vertices with the restriction that each triangle side is entirely shared by two adjacent triangles.

3.2.2.1 Delaunay Triangulation

The Delaunay triangulation was invented by Boris Delaunay in 1934. Given a set $P$ of points in the plane, a triangulation $DT(P)$ with the property that no point in $P$ lies inside the circumcircle of any triangle in $DT(P)$ is called the Delaunay triangulation. It maximizes the minimum angle of all the angles of the triangles in the triangulation; it tends to avoid skinny triangles. The triangulation is obtained by connecting all the neighboring points in the Voronoi tessellation of the given point set.

Based on Delaunay’s definition, the circumcircle of a triangle formed by three points from the original point set is empty if it does not contain vertices other than the three that define it (other points are permitted only on the very perimeter, not inside); see Figure 3.2.

The Delaunay condition states that a triangular net is a Delaunay triangulation if all the circumcircles of all the triangles in the net are empty. This is the original definition
for two-dimensional spaces. It is possible to use it in three-dimensional spaces by using a circumscribed sphere in place of the circumscribed circle.

**Figure 3.2** Delaunay triangulation in the plane with circumcircles shown.

For a set of collinear points Delaunay triangulation is not defined. When four points lie on the same circle (e.g., the vertices of a rectangle) the Delaunay triangulation is not unique: clearly, the two possible triangulations that split a rectangle in two, both satisfy the Delaunay condition.

The triangulation is well-suited for a scattered parametric data set with underlying parameter values. In this case triangulation is constructed in the parameter space.

We are interested in adaptive rendering algorithms. The Delaunay triangulation is a very powerful tool for this problem. Adding more points to a existing point set and triangulation, the new triangulation can be recomputed quickly. There exists efficient methods for computing the new finer triangulation based on the existing one.

The main drawback of any surface approximating method based on the Delaunay triangulation is that for irregularly spaced data it is difficult to maintain continuity of approximated surface. It is hard to construct smooth surface from such data. Even if continuity is maintained, artifacts on a surface often appear.
There exists a variety of triangulations of regularly spaced data. We will use a standard type-1 triangulation $\Delta$ in the methods we construct. Given two linearly independent vectors $\vec{u}_1$ and $\vec{u}_2$, we define a two-dimensional lattice

$$V := \{v_{i,j} = i\vec{u}_1 + j\vec{u}_2\},$$

where we assume that both indices $i$ and $j$ take all values in $\mathbb{Z}$, unless we explicitly provide more specific ranges. The lattice consists of the vertices of tessellation whose faces are equal parallelograms. Splitting each parallelogram by drawing in the diagonals $v_{i,j}v_{i+1,j+1}$ yields two triangles

$$T_{i,j} := v_{i,j}v_{i+1,j+1}v_{i+1,j}, \quad \text{(top triangle)}$$

$$B_{i,j} := v_{i,j}v_{i,j+1}v_{i+1,j+1}. \quad \text{(bottom triangle)}$$

The union of these triangles

$$\Delta := \bigcup_{i,j}(\{T_{i,j}\} \cup \{B_{i,j}\})$$

is a planar triangulation, which is known as the type-1 triangulation and can be viewed as a collection of overlapping hexagons; see Figure 3.3.

When in the regular type-1 triangulation, we may split a single triangle at its midpoint edges to generate four triangles that produces a pseudo-regular triangulation. Such a triangulation can be used as a basis for an adaptive rendering method.

Although, a regular triangulation seems to have limited applications, there exists a number of higher order surface approximation schemes based on it. A surface can be smoothly reconstructed without artifacts.

The triangulation can also be applied for scattered parametric data in physical space. However, some higher-order surface approximation method must first be applied locally and sampled on the triangular grid.
Figure 3.3 (a) Type-1 triangulation with one hexagon bolded, (b) The six triangles in $\Delta$ attached to $v_{i,j}$.

3.2.3 Barycentric Coordinates

Consider a triangle $T$ with vertices $p_1, p_2, p_3$ (i.e., 3 non-collinear points) and a fourth point $p$, all in $\mathbb{E}^2$. Then $p$ may be written uniquely as a barycentric combination of these three vertices, i.e.,

$$p = u p_1 + v p_2 + w p_3,$$

where $u, v$ and $w$ are subjected to the constraint $u + v + w = 1$. The coefficients $u := (u, v, w)$ are called the barycentric coordinates of $p$ with respect to the triangle $T$.

The barycentric coordinates $(u, v, w)$ of $p$ can always be determined geometrically by the relations

$$u = \frac{\text{area}(pp_2p_3)}{\text{area}(p_1p_2p_3)}, \quad v = \frac{\text{area}(p_1pp_3)}{\text{area}(p_1p_2p_3)}, \quad w = \frac{\text{area}(p_1p_2p)}{\text{area}(p_1p_2p_3)},$$

where

$$\text{area}(p_ip_jp_k) = \begin{vmatrix} p_i & p_j & p_k \\ 1 & 1 & 1 \end{vmatrix}.$$
(\(p_i\) is a vector so this matrix is square) which gives signed area. If the coordinates \(u, v, w\) are all positive then the point \(p\) lies inside the triangle \(T\). A point outside the triangle has at least one negative coordinate.

### 3.2.4 Bernstein Polynomials

The Bernstein polynomial \(B^n_i\) of degree \(n\) over a triangle, given by the barycentric coordinates \(u := (u, v, w)\), is defined by

\[
B^n_i(u) = \binom{n}{i} u^i v^j w^k; \quad |i| = n.
\]

We define \(B^n_i(u) = 0\) if \(i, j, k < 0\) or \(i, j, k > n\). This follows the standard convention for the trinomial coefficients \(\binom{n}{i} = \frac{n!}{i! j! k!}\).

The Bernstein polynomials \(B^n_i\) are the natural generalization of univariate Bernstein polynomials \(B^n_i(t) = \binom{n}{i} t^i (1-t)^{n-i}\) which are defined over \([0, 1]\), and with the definitions \(u = t, v = 1 - t\), they can be written as

\[
B^n_{i,j}(u, v) = \frac{n!}{i! j!} u^i v^j; \quad i + j = n,
\]

where \(u, v\) are barycentric coordinates on the interval \([0, 1]\).

The Bernstein polynomials form a partition of unity, i.e.,

\[
\sum_{|i|=n} B^n_i(u) = 1,
\]

they are positive inside the triangles

\[
B^n_i(u) \geq 0,
\]

and satisfy the recursion relation

\[
B^n_{i,j,k} = uB^n_{i-1,j,k} + vB^n_{i,j-1,k} + wB^n_{i,j,k-1}. \quad (3.3)
\]
3.2.5 Triangular Bernstein-Bézier Patches

Triangular polynomial patches were first considered by de Casteljau in 1959. A Bernstein-Bézier triangular patch is a polynomial of the form

$$b^n(u) = \sum_{|i|=n} b_i B^n_i(u).$$

(3.4)

It is formed as the sum of \(p = \frac{1}{2} (n + 1) (n + 2)\) terms. The \(b_i\) are called the Bézier ordinates (control points) of \(b^n\). The piecewise linear interpolant of the points \((i/n, b_i)\) is called the Bézier control net of \(b^n\). The curved Bézier patch \(b^n\) itself is located inside the convex hull of its defining ordinates, the triangulated surface defined by the control net, just as a 1D Bézier curve lies inside the convex hull of its control polygon. This property is important for contouring and rendering of surfaces. The boundary curves of the patch \(b^n\) are (univariate) Bézier polynomials and their control polygons are the boundaries of the control net. Note that for the case of the control net being planar, the generalized patch is also planar.

**Figure 3.4** Cubic Bézier triangle with the control net and the domain triangle.

Triangular Bézier patches can also be used to build piecewise-defined complex surfaces, where each patch corresponds to a triangle of a tessellation of \((s, t)\)-plane. The main problem in the construction of such surfaces is to maintain their continuity. A factor which makes this harder than in the case of Bézier curves is that only corner points of Bézier patch are interpolated but continuity conditions have to be satisfied along the
When the surface is given by patches of different resolution, even $C^0$ continuity may be problematic. In order to maintain $C^1$ continuity between two patches, their control nets have to satisfy affinity conditions. We have better chance of solving the problem if we relax the requirement of $C^1$ continuity to that of $G^1$ continuity. Two patches are called $G^1$ if they have a continually varying tangent plane along that boundary curve. Note that it is not enough for $G^1$ continuity that the boundary subtriangles of two control nets are coplanar; the control net is not tangent to the patch at the boundary. In order to construct composite surface from parametric data satisfying continuity conditions we need to use higher order patches. For instance, as we will see in 3.3.4, to maintain $G^1$ continuity for a composite parametric surface we need to use quartic Bézier patches.

The theory of Bernstein-Bézier patches is well introduced by Farin in [13].

### 3.2.5.1 The de Casteljau Algorithm for Triangular Patches

The de Casteljau algorithm for triangular patches is a direct generalization of the corresponding algorithm for curves. The de Casteljau algorithm for curves uses repeated linear interpolation, and this process is also the key ingredient in the case of Bézier triangles.

The recursion formula (3.3) provides the de Casteljau algorithm for the evaluation of a Bézier triangle $b^n$ at given local coordinates $u := (u, v, w)$. Recursively defining auxiliary (de Casteljau) points

$$b^l_{i,j,k} = ub^l_{i-1,j,k} + vb^l_{i,j-1,k} + wb^l_{i,j,k-1},$$

where $i + j + k = n - l$ ($i, j, k \geq 0$), and $b^0_{i,j,k} = b_{i,j,k}$, it is easy to show that $b^n(u) = b^n_{0,0,0}(u,v,w)$. A schematic picture of the algorithm is shown in the left part of Figure 3.5.

The algorithm provides a numerically stable and efficient method for the evaluation of a Bézier triangle. It performs better – in terms of accuracy and speed – than simple evaluation of Bernstein-Bézier patches using formula (3.4).
Figure 3.5 The de Casteljau algorithm: the original cubic control net $b^0$ (black) together with the quadratic $b^1$ (red) and linear $b^2$ (blue) nets that determine the resulting point. All intermediate control points are generated in the ‘upright’ subtriangles. The blue linear net determines the tangent plane to the patch at the resulting point. The resulting subdivision into three smaller Bézier patches on right.

3.2.5.2 Subdividing Triangular Bézier Patches

Consider first the de Casteljau triangular algorithm for an $n$-degree triangular Bézier patch. The algorithm generates three subtriangles with edges pointing into the nodes from the three corners. Therefore, it follows immediately that we can subdivide a triangular Bézier patch into three patches by taking the values of intermediate control points of the de Casteljau algorithm; see Figure 3.5. This algorithm is the generalization to triangular patches of the de Casteljau subdivision algorithm for Bézier curves. As we will see below a 3-1 splitting does not generate a useful grid for an adaptive scheme, but it is used to generate both a 2-1 splitting and consequently the 4-1 splitting scheme that we eventually will use.

3-to-1 Subdivision is a straightforward application of the de Casteljau algorithm. The idea is that 3-to-1 subdivision is repeatedly performed until the control points are nearly coplanar. There are two problems with 3-to-1 subdivision. The first is that the aspect ratio of the polygons is bad. For a variety of reasons, it is preferable that the triangles are close to equilateral. However, 3-to-1 subdivision gives triangles that get longer and skinnier each time the splitting is performed; see Figure 3.6.
Figure 3.6 Two steps of 3-to-1 subdivision over a Bézier triangle.

The second flaw of 3-to-1 subdivision (for rendering purposes) is that it does not divide the edges of the patches, and would fail to produce image triangles that are nearly flat. Since the long edge is never refined, this method does not improve flatness if the image of the long edge has high curvature. For instance, consider a triangular patch wrapped around a cylinder. After repeat subdivision there still exists a long thin triangle wrapped around the cylinder whose ordinates, definitely, are not nearly flat. Additional problems can be caused by rapidly increasing valence (i.e., number of edges issuing from a vertex) of the vertices. The valence of a vertex is doubled each time the subdivision is applied, and an approximation near the high-valence points is problematic.

2-to-1 Subdivision is a more efficient way to subdivide a triangular Bézier patch and the initial step in creating a 4-1 split. Consider what happens when the de Casteljau algorithm is used to evaluate the Bézier patch on its boundary of the domain triangle. The barycentric coordinates of the domain point of evaluation are $u = (0, v, w)$. Essentially, it is accomplished by performing a sequence of degenerate 3-1 splits with the evaluation occurring at the midpoint of an edge. Thus, it really produces 2-to-1 subdivision.

Figure 3.7 Three steps of the 2-to-1 Subdivision over a Bézier triangle.

At first glance, a 2-to-1 subdivision appears to increase the aspect ratio; see Figure 3.7. But if 2-to-1 subdivision is repeatedly performed by connecting from the previous subdivision point to the two edges that had not been subdivided at the previous step, after three steps, all triangles have the same aspect ratio as the triangles produced after the
first step. Although there is a worsening of the aspect ratio in the first 2-to-1 subdivision step (and possibly in the second step), the aspect ratio gets no worse than this.

**4-to-1 Subdivision** is the subdivision technique that is crucial in our method. We saw earlier evaluation of a triangular Bézier patch with de Casteljau algorithm, produces a 3-to-1 subdivision of the patch. Note, however, that the edges of the patch (or of the subpatches) were never subdivided if the subdivision step was repeated. A nicer subdivision technique would be to perform a 4-to-1 subdivision by splitting all three edges of the domain and connecting them in the obvious fashion; see Figure 3.8.

![Figure 3.8 Schematic 4-to-1 Subdivision.](image)

Thus, the original domain is $\triangle P_0P_1P_2$ and it needs to be subdivided into four subtriangles, $T_0$, $T_1$, $T_2$, and $T_3$. This may be implemented by performing repeated 2-1 subdivision on the patch. Most of the subdivisions are 2-to-1 subdivisions, where 3-to-1 subdivision is performed on an edge and the degenerate patch is ignored; see Figure 3.9.

![Figure 3.9 Steps of the 4-to-1 Subdivision of a Bézier triangle.](image)
Here, the original patch is subdivided at $\tilde{P}_0$. Then patch $A$ is subdivided at $\tilde{P}_1$ giving patch $T_2$. Patch $B$ is subdivided at $\tilde{P}_2$ to get $T_1$. Next, either $T_2$ is subdivided at $\tilde{P}_2$ or $T_1$ at $\tilde{P}_1$ to get $T_3$. Finally, $T_3$ is subdivided at $P_0$ to get patch $T_0$.

### 3.2.6 Surface Normal Estimation

In the next section we will introduce a few triangular interpolation methods. Some schemes based on triangular Bézier patches require to a vector normal to the surface be given at each vertex. However, since an exact value of the normal is not always provided, in most cases it will have to be estimated from the given data values. This can be constructed using the data at a vertex and a few of its neighbors. The problem is similar to the construction of tangent vectors needed by Catmull-Rom splines for curves, described in Section 2.2.

A few possible methods for estimating the normals can be constructed. Some of these normal estimation methods can be considered as an additional preprocessing step, to be performed immediately after the triangulation step.

Analogously to the tangent vector of a curve, the normal vector to a surface at any vertex $\mathbf{x}_i$ can be estimated by weighting the unit normals of all the incident triangles, and normalizing the sum. Suppose a given vertex has valence $N$, i.e., it is the meeting place of $N$ edges and $N$ faces. A normal $\tilde{n}_i$ to each face is given by one-half of the cross product of the two edges which meet at the vertex, where the magnitude of $\tilde{n}_i$ is twice the area of the face. The approximate normal to the surface is constructed as some linear combination of $\tilde{n}_i$. For example:

1. weight by area. For each triangle, take the cross-product of two different vectors between its vertices. This vector is normal to the triangle, and its magnitude is twice the triangle’s area. Sum all the adjacent vectors, and normalize to unit length. The idea of this method is that the larger triangles correspond to a larger part of the surface, and should affect the orientation of the tangent plane moat;
2. weight uniformly. Divide the cross-products by twice the area of the triangle so as to give unit normals. Then normalize the sum;

3. weight by inverse area. Divide each unit normal by the area of the triangle, then normalize the sum. The reasoning behind this method is that a close neighboring vertex knows more about the local surface normal and should have a larger weight than faraway vertices.

All the above constructions provide second-order approximations to the normal. This will reduce the overall order of a higher-order interpolation method. An estimation method with higher accuracy could improve calculations. We performed numerical tests with each of above approaches and we did not find significant differences. Thus, for simplicity, in further numerical tests we use construction 2.

3.3 Triangulation Based Surface Approximation Schemes

3.3.1 Bivariate Linear Interpolation

Let the scattered points \((x_i, y_i)\) be located in the \((x, y)\)-plane. A continuous function \(f(x, y)\) can be visualized as altitudes \(z_i\) over this plane. Therefore, any triangulation of the scattered points in the \((x, y)\)-plane induces a piecewise triangular surface over the plane, whose nodes are the points \((x_i, y_i, z_i)\). This is a continuous surface made up of planar triangular pieces that are joined along edges which is an obvious generalization of broken line functions in one dimension. Such a surface is often called a triangulated irregular network. This surface represents, therefore, a piecewise interpolation scheme in which a bivariate linear interpolation is applied within each triangle.

Let points \(p_1, p_2,\) and \(p_3\), located at \((x_1, y_1)\), \((x_2, y_2)\) and \((x_3, y_3)\), be the three vertices of any of the underlying triangles in the \((x, y)\)-plane, and let their values be \(z_1\), \(z_2\), and \(z_3\). Then, the value \(z\) at any arbitrary point \(p\) located at \(x = (x, y)\) within this triangle can be found by barycentric coordinates. The point can be uniquely expressed
as \( z = uz_1 + vz_2 + wz_3 \), a weighted average of the locations of these three vertices

\[
P = up_1 + vp_2 + wp_3
\]

where \( u + v + w = 1 \) and all are non-negative. Note that this surface is \( C^0 \).

A similar approach can be easily implemented for parametric data by performing triangulation in the parametric space. Let points \( p_1, p_2, \) and \( p_3 \), located at \( (x_j, y_j, z_j) \), for \( (s_j, t_j) \), \( j = 1, 2, 3 \), be the three vertices of any of the underlying triangles in the \( s, t \) parametric space. Then, any arbitrary point \( p \) located at \( x = (x(s, t), y(s, t), z(s, t)) \) within this triangle can be found in terms of barycentric coordinates. As above, the point can be uniquely expressed as a weighted average of the locations of these three vertices by Equation (3.2).

### 3.3.2 Nine-parameter Interpolant

We now assume that we are given vertices and a normal at each of these vertices. The goal is to build a cubic Bézier patch over each triangle, such that the resulting collection of Bézier triangles forms a continuous surface. In general, this surface will not be \( G^1 \) — but is more accurate than a composite of planar patches.

Let us concentrate on one triangular facet. It has three vertices, denoted by \( b_{3,0,0}, b_{0,3,0}, b_{0,0,3} \) and normals at those points denoted \( n_{2,0,0}, n_{0,2,0}, n_{0,0,2} \). A nine-parameter interpolant will be cubic over each triangle, thus being determined by ten coefficients or Bézier ordinates. Nine of these are immediately determined by the given data. The desired Bézier patch needs three boundary curves; let’s concentrate on the one from \( b_{3,0,0} \) to \( b_{0,3,0} \). We need Bézier points \( b_{2,1,0} \) and \( b_{1,2,0} \) in order to have a boundary control polygon. These are found by using univariate cubic Hermite interpolation to each boundary edge. In order to find \( b_{2,1,0} \), we define \( b'_{2,1,0} = (2b_{3,0,0} + b_{0,3,0})/3 \). Then we project \( b'_{2,1,0} \) into the plane defined by point \( b_{3,0,0} \) and normal \( n_{3,0,0} \), and obtain

\[
b_{2,1,0} = b'_{2,1,0} - \frac{1}{3} (n_{2,0,0} ([b_{0,3,0} - b_{3,0,0}]) n_{2,0,0},
\]
and $b_{1,2,0}$ in a symmetric fashion. We repeat the process for the remaining two boundaries; see Figure 3.4.

The control point $b_{1,1,1}$ is still undetermined. It is independent of the prescribed data and can be assigned an arbitrary value. A reasonable choice is to select $b_{1,1,1}$ such that the quadratic precision of the interpolant is maintained. We express the coefficient $b_{1,1,1}$ as

$$b_{1,1,1} = \frac{1}{4}(b_{2,0,1} + b_{1,0,2} + b_{0,2,1} + b_{0,1,2} + b_{2,1,0} + b_{1,2,0}) - \frac{1}{6}(b_{3,0,0} + b_{0,3,0} + b_{0,0,3}).$$  (3.5)

At this point, we can build a cubic patch over each triangular facet; the resulting overall surface will not be smooth.

The method has a serious drawback: it requires $C^1$ data, but the produced overall surface is only $C^0$. However, there are also some advantages of using it. Most important is its simplicity; the method can be easily implemented and, moreover, works for both scalar and parametric 3D data. Additionally, there exists a modified display algorithm, called Point-Normal triangles, such that the resulting surface looks smooth.

### 3.3.3 The Clough-Tocher Method

As we have seen, one of the main drawbacks of piecewise linear interpolation is its low accuracy; it is a second-order method. The accuracy of nine-parameter interpolation is not better at all. Cubic patches improve visual appearance of a surface but in general the method is also second-order. The scheme is visually improved, but it is still only $C^0$. It is not smooth across the boundaries between triangles. $C^1$ continuity obviously requires piecewise interpolation schemes based on polynomials of order higher than one.

In order to guarantee $C^1$ continuity across boundaries, we need to have more information at each triangle’s vertices or edges. Given a $C^1$ function $f$ and set of points $\{(x_i, y_i)\}$, we need to know not only the function value $z_i = f(x_i, y_i)$ but the gradient $\{\nabla f|_{(x_i,y_i)}\}$. 
Like piecewise linear and nine-parameter interpolation, the Clough-Tocher interpolation method, is based on a triangulation of a scattered point set. It uses a cubic interpolation scheme within each triangle using bivariate cubic polynomials. In order to create a piecewise $C^1$-continuous interpolation surface, the Clough-Tocher method requires for each triangle the value and the gradient i.e., two partial derivatives at each of the three vertices, as well as the normal derivative at the midpoint of each of the three edges in order to ensure that the normal slope matches across triangle boundaries. The aim is to impose these twelve constraints (three per vertex + one per edge) on the cubic polynomial defined on our triangle. However, this is not possible since a bivariate cubic polynomial is determined by only ten coefficients and can therefore only satisfy ten constraints.

Nevertheless, these twelve constraints can all be satisfied by splitting each triangle of the original triangulation into three mini-triangles by joining each of the three vertices to the centroid and defining a bivariate cubic polynomial on each of the three mini-triangles. This construction gives more degrees of freedom for satisfying the constraints. The key idea behind the Clough-Tocher method is, therefore, to split each cubic polynomial patch into three cubic polynomial subpatches in order to satisfy the $C^1$ continuity constraints between neighbors. Specifically, a cubic Bézier patch is defined over each mini-triangle; see Figure 3.10. Because of this split we refer to the method as a splitting scheme.

![Figure 3.10](image-url) Schematic representation of a three-split of a cubic Bézier triangle; three mini-triangles and its 19 Bézier ordinates (control points).

Note that these cubic polynomials have exactly ten coefficients. The control points are not included in the scattered point set, and their values are not known a priori. They
are given by imposing constraints on the cross-boundary derivatives in order to impose 
\( C^1 \) smoothness across their segment boundaries.

Let us see now the geometric interpretation of the constraints that we need to impose in order to ensure \( C^1 \) continuity everywhere on the piecewise interpolating surface. The Bézier ordinates form within each mini-triangle a triangular net that divides each mini-triangle edge into three equal segments; this triangular net is known as the control net. We will call the nine small triangles that are defined by the control net within each mini-triangle micro-triangles. The constraints imposed in order to ensure \( C^1 \) continuity can be geometrically interpreted as a 2D generalization of the constraints used in the 1D case to impose a smooth behavior across segment boundaries in a piecewise Bézier curve. Just as in the 1D case the two control line segments to both sides of each boundary must be collinear, so in this 2D case every two neighboring micro-triangles that lie to both sides of a mini-triangle edge must be coplanar (note that there are three pairs of such coplanar micro-triangles along each mini-triangle edge). This also imposes the condition that all neighboring micro-triangles surrounding the vertex of a mini-triangle must be coplanar; see Figure 3.10.

**Figure 3.11** Adjacent \( C^1 \) triangular patches defined over mini-triangles.

There exist several variants and generalizations of the Clough-Tocher method; e.g., by Farin in [12] or by Mann in [40]. However, they all share the same main steps. Suppose
the triangle \( b_{3,0,0}, b_{0,3,0}, b_{0,0,3} \) is given, with normals \( n_{2,0,0}, n_{0,2,0}, n_{0,0,2} \). For \( i = \{i, j, k\} \) and \(|i| = 3\) the construction of the mini-triangle \( b_{3,0,0} b_{0,3,0} e \) follows

1. The nine boundary Bézier ordinates \( (b_i) \) are chosen as in the nine-parameter interpolant, determined from the given data by univariate cubic Hermite interpolation.

2. The point \( b_{1,1,1} \) is chosen as in the nine-parameter interpolant; Equation (3.5).

3. Set \( c_{0,1} = (b_{3,0,0} + b_{2,1,0} + b_{2,0,1})/3 \), \( c_{1,1} = (b_{0,3,0} + b_{1,2,0} + b_{0,2,1})/3 \) and \( c_{2,1} = (b_{0,0,3} + b_{0,1,2} + b_{0,2,1})/3 \). The \( c_{i,1} \) lie in tangent plane at the vertex.

4. Set \( d_0 \) to be coplanar with \( b_{2,1,0}, b_{1,2,0} \) and the corresponding point \( \tilde{d}_0 \) on the other side of the \( b_{3,0,0}, b_{0,0,3} \) boundary in order to satisfy the cross-boundary derivative condition. Similar conditions define \( d_1 \) and \( d_2 \).

5. Set \( c_{i,2} \) to lie at the center of the triangle defined by \( d_i, d_j \) and \( c_{i,1} \)

6. Set \( e = (c_{0,2} + c_{1,2} + c_{2,2})/3 \) as a replacement for \( b_{1,1,1} \).

The only freedom in this construction comes in step (4). Here, we only introduce the standard Clough-Tocher scheme, although variants have been proposed.

During the process of finding \( d_0 \) we can also find \( \tilde{d}_0 \), the center point of the neighboring mini-triangle. The other way to determine it would be to repeat the same procedure for the other mini-triangle; see Figure 3.11. Let \( e, b_{3,0,0}, b_{0,3,0} \) and \( \tilde{e}, b_{3,0,0}, b_{0,3,0} \) be the vertices of two adjacent mini-triangles and \( e', b'_{3,0,0}, b'_{0,3,0} \) and \( \tilde{e}', b'_{3,0,0}, b'_{0,3,0} \) be their projection into the \((x, y)\)-plane, respectively. We can express \( e' \) and \( \tilde{e}' \) in terms of barycentric coordinates of the opposite triangle

\[
\begin{align*}
    e'_x &= u'\tilde{e}'_x + v' b'_{0,3,0} + w' b'_{3,0,0} \\
    e'_y &= u'\tilde{e}'_y + v' b'_{0,3,0} + w' b'_{3,0,0} \\
    \tilde{e}'_x &= u e'_x + v b'_{3,0,0} + w b'_{0,3,0} \\
    \tilde{e}'_y &= u e'_y + v b'_{3,0,0} + w b'_{0,3,0}
\end{align*}
\] (3.6)
The $C^1$ continuity conditions are fulfilled by the subtriangle pair formed by $b_{3,0,0}\vec{c}_{0,1}b_{2,1,0}$ and $b_{3,0,0}b_{2,1,0}c_{0,1}$ and the pair $b_{0,3,0}b_{1,2,0}\vec{c}_{1,1}$ and $b_{0,3,0}c_{1,1}b_{1,2,0}$ since they lie in the same tangent plane. But $C^1$ continuity can be violated by the middle pair $\Delta b_{2,1,0}b_{1,2,0}d_0$ and $\Delta b_{2,1,0}\vec{d}_0b_{1,2,0}$. The $C^1$ condition that has to be met can be expressed as

$$d_0 = vb_{1,2,0} + wb_{2,1,0} + ud_0. \quad (3.7)$$

The standard way to guarantee (3.7) is as follows: let $L$ denote any direction not parallel to $b_{3,0,0}b_{0,3,0}$, (for our purposes $L = (c_{1,1} - \vec{c}_{1,1} + c_{0,1} - \vec{c}_{0,1})/4$). Then the $L$-directional derivative of the mini-patch defined over $\Delta b_{3,0,0}b_{0,3,0}e$ is a univariate quadratic Bézier polynomial with Bézier ordinates

$$3(l_1c_{1,1} + l_2b_{1,2,0} + l_3b_{0,3,0}), \quad 3(l_1d_0 + l_2b_{2,1,0} + l_3b_{1,2,0}), \quad 3(l_1c_{0,1} + l_2b_{3,0,0} + l_3b_{2,1,0}),$$

where $(l_1, l_2, l_3)$ denotes the barycentric representation of the direction $L$ in terms of the triangle $b_{3,0,0}b_{0,3,0}e$, i.e., $(l_1, l_2, l_3)$ is a solution of the system

$$L_x = l_1\vec{c}'_x + l_2b'_{0,3,0} + l_3b'_{3,0,0},$$

$$L_y = l_1\vec{c}'_y + l_2b'_{0,3,0} + l_3b'_{3,0,0},$$

$$l_1 + l_2 + l_3 = 0.$$ 

We can fix the unknown $d_0$ by imposing that the directional derivative varies linearly along the edge. The edge midpoint cross-boundary derivative is simply set to be the average of the directional derivatives at the edge endpoints. This choice can be expressed as

$$(l_1c_{1,1} + l_2b_{1,2,0} + l_3b_{0,3,0}) - 2(l_1d_0 + l_2b_{2,1,0} + l_3b_{1,2,0}) + (l_1c_{0,1} + l_2b_{3,0,0} + l_3b_{2,1,0}) = 0.$$ 

One approach to finding the remaining unknown $\vec{d}_0$ is to use relation (3.7). Another way to determine the remaining unknown $\vec{d}_0$ is to repeat the whole procedure for the
other mini-patch, defined over $\Delta b_{0,3,0} b_{3,0,0} \tilde{e}$. It is essential that the $L$ denotes the same direction in both neighboring mini-patches. After finding all three center points of the mini-triangle, we can compute the rest of the interior points by applying steps 5 and 6 of the procedure described above.

Note that the fact that makes this construction possible is that we do not need to impose restrictions at the third vertex of each mini-triangle (the center point): as we have just seen, the value and the slope there are automatically derived from the vertex and edge-midpoint data of the triangle. Hence, the subdivided surface now has enough degrees of freedom to allow $C^1$ continuity of the overall interpolant. By proceeding in a similar way for all the other triangles, we obtain a $C^1$-continuous piecewise interpolating surface that is defined over the triangulation of our given scattered point set.

Since the Clough-Tocher interpolation method is local, it has the advantage of speed: even large scattered point sets can be interpolated quite rapidly. Also, as for the linear triangulation-based interpolation methods, a correction in any of the given data points only influences the interpolated values within the triangles that have this data point as a vertex. The Clough-Tocher method gives a smooth $C^1$-continuous interpolation surface. However, it still depends on the quality of the underlying triangulation.

Preliminary numerical tests show that the method, while maintaining $C^1$ continuity, produces many surface ripples and artifacts. Moreover, the main disadvantage of the method is that the Clough-Tocher splitting scheme can be used only in functional settings. However, the method has been generalized to parametric data, where quartic patches must be used instead of cubic. This is the subject of the next section.

### 3.3.4 The Shirman-Sequin Method

If the surface to be constructed lies above the plane it can be described as $S(x, y) = (x, y, f(x, y))$. The data set is then referred to as scalar data as the surface
can be thought of as a scalar valued function over the plane. Such data can be interpolated with a $C^1$ surface, using, for instance, the Clough-Tocher method introduced above.

A parametric scheme, on the other hand, constructs a vector valued surface, $S(s, t) = (x(s, t), y(s, t), z(s, t))$ and is capable of representing a wider class of topologies. The parametric problem is generally considered to be more difficult than the scalar variant. It has been shown, for instance, that the data cannot always be interpolated with a parametrically continuous surface [31]. Instead, the continuity conditions have to be relaxed to $G^1$ (tangent plane) continuity; see Subsection 3.2.5.

![Figure 3.12 Schematic representation of the Shirman-Sequin three-split method.](image)

The Shirman-Sequin method, first introduced in [57, 58], is the splitting scheme. It builds a surface for each triangular face by first computing cubic boundary curves around the triangle and subsequently degree elevating them to quartics; see Subsection 2.2.2.2. Next, it constructs three quartic patches that match this boundary data; see Figure 3.12. The boundary curves are constructed independently of each other. Then a single Bézier patch cannot in general be used to interpolate the data, as the vertex consistency problem cannot in general be solved. Split domain schemes avoid this problem by constructing three patches per face, essentially splitting the domain triangle into three mini-triangles, as was done by Clough and Tocher for scalar valued data.

The scheme first computes cubic boundaries, so we need to refer to both the cubic control points $b_i$ where $|i| = 3$ and the quartic control points $b_j$ where $|j| = 4$ for these boundaries; see Figure 3.13. Control points $c_{i,1}$ lie in tangent plane at the vertex and they are computed as for the Clough-Tocher scheme; point (2) of the description. However, the control points $c_{i,2}$ are yet to be determined. Assume for the moment that points $l_i, m_i$
Farin’s $G^1$ conditions are used on the internal boundaries to ensure that the three patches meet each other with $G^1$ continuity. The final formulas for $c_{i,2}$ and $k_i$ are

- $c_{0,2} = (b_{4,0,0} - 3c_{0,1} + 4l_0 + 4m_2)/6$,
- $c_{1,2} = (b_{0,0,4} - 3c_{1,1} + 4l_1 + 4m_0)/6$,
- $c_{2,2} = (b_{0,4,0} - 3c_{2,1} + 4l_2 + 4m_1)/6$,
- $k_0 = (-c_{0,1} - c_{1,1} + 4c_{2,1} + 4c_{0,2} + 4c_{1,2} - 3c_{2,2})/4$,
- $k_1 = (-c_{1,1} - c_{2,1} + 4c_{0,1} + 4c_{1,2} + 4c_{2,2} - 3c_{0,2})/4$,
- $k_2 = (-c_{2,1} - c_{0,1} + 4c_{1,1} + 4c_{2,2} + 4c_{0,2} - 3c_{1,2})/4$,
- $e = (c_{0,2} + c_{1,2} + c_{2,2})/3$.

Note that the points $c_{i,j}$ are the cubic ordinates, so that to complete the set of quartic control points we have to degree elevate the cubic boundaries between subpatches. Note that points $d_i$ can be also derived independently as $d_0 = (b_{4,0,0} + b_{3,1,0} + b_{3,0,1})/3$, $d_1 = (b_{0,4,0} + b_{1,3,0} + b_{0,3,1})/3$ and $d_2 = (b_{0,0,4} + b_{0,1,3} + b_{0,3,1})/3$; i.e., $d_i$ lie in the tangent plane at the vertex.

**Figure 3.13** Control points for the Shirman-Sequin method.

The points $l_i, m_i$ are determined using Chiyokura and Kimura’s method [7]. The method uses the boundary data for two adjacent patches to construct the Bézier control points that influence the tangent plane behavior along their common boundary; see
Figure 3.14. Two quartic interior Bézier control points for each patch are set to match the tangent plane field.

For the Chiyokura and Kimura construction denote \( \vec{v}_0, \vec{v}_1 \) as two unit vectors that are perpendicular to the tangents of the boundary curve at the end points. Next, let us define vectors \( \vec{b}_0 = b_{2,1,0} - b_{3,0,0}, \vec{b}_1 = b_{1,2,0} - b_{2,1,0} \) and \( \vec{b}_2 = b_{0,3,0} - b_{1,2,0} \), and vectors \( \vec{d}_0 = \vec{d}_0 - \vec{d}_1, \vec{d}_1 = \vec{d}_1 - \vec{d}_0. \) The desired interior control points are given by

\[
\begin{align*}
\mathbf{l}_0 &= \left( \alpha_1 \vec{v}_0 + (\alpha_0 + \alpha_1) \vec{v}_1 + \beta_0 \vec{b}_1 + 2\beta_1 \vec{b}_0 \right) / 3 + b_{1,2,0}; \\
\mathbf{m}_0 &= \left( (\alpha_0 + \alpha_1) \vec{v}_0 + \alpha_0 \vec{v}_1 + 2\beta_0 \vec{b}_1 + \beta_1 \vec{b}_0 \right) / 3 + b_{1,2,0};
\end{align*}
\]

where the coefficients \( \alpha_0, \alpha_1, \beta_0 \) and \( \beta_1 \) are determined from relation

\[
\begin{align*}
\vec{d}_0 &= \alpha_0 \vec{v}_0 + \beta_0 \vec{b}_0; \\
\vec{d}_1 &= \alpha_1 \vec{v}_1 + \beta_1 \vec{b}_2.
\end{align*}
\]

This gives the construction for only one patch. The construction for the other patches, however, is identical. Now it is possible to compute the interior control points \( \mathbf{l}_i, \mathbf{m}_i \) easily. The resulting interior control points guarantee \( G^1 \) continuity between two patches. Further details of the scheme are described in [41, 57, 58].

In [41], Mann et al. survey triangular interpolation methods for parametric scattered data approximation problems. In conclusion, they recommend using the Shirman-
Sequin scheme. The use of polynomial patches, simplifies the calculation of derivatives, curvature, etc. of the interpolant. Although the domain split introduces extra artifacts in the surfaces, such as the creation of long, thin triangles, the major shape defects are common to all schemes.

3.3.5 Quasi-Interpolation based on Quartic Bernstein-Bézier Patches

After learning about the Shirman-Sequin method, we implemented it. Our tests, described below, showed several problems. The tangent continuity does not imply overall better visual impression: smooth interpolating schemes often generate extraneous creases. We therefore discuss a method that avoids these problems by avoiding precise interpolation.

In Subsection 2.2.6 we described an idea of quasi-interpolation for curves. The method described in this section was introduced and well motivated by Sorokina and Zeilfelder in [59]. This quasi-interpolation local method is based on bivariate $C^1$ splines and approximates data regularly distributed in the plane. The method uses the piecewise Bernstein-Bézier patches for its basis, and sets the Bézier coefficients (ordinates) directly so that the smoothness conditions are automatically satisfied. This is done by applying some natural and simple averaging rules to local portions of the given data, so that the spline is immediately available without solving any linear systems. Simultaneously, it ensures that the quasi-interpolation splines yield approximations of order four, which is best possible for the spaces of consideration.

Bernstein-Bézier Form of Quartic Splines. We consider the space of quartic $C^1$-splines on a type-1 triangulation $\triangle$, described in Subsection 3.2.2.2, defined by

**Definition 3.1**

$$S^1_4(\triangle) = \{ s \in C^1(\mathbb{R}^2) : s|_T \in \mathcal{P}_4, \forall T \in \triangle \}$$
where \( \mathcal{P}_4 = \text{span}\{x^iy^j : i + j = 0, \ldots, 4\} \) denotes the space of bivariate polynomials of total degree four.

We use the piecewise Bernstein-Bézier representation of the splines, i.e., for each spline \( s \in S_1^4(\Delta) \), the polynomial patch \( s|_T \in \mathcal{P}_4 \) on a triangle \( T = P_0P_1P_2 \in \Delta \) is given by

\[
s|_T = \sum_{i+j+k=4} c_{i,j,k}B_{i,j,k} = \sum_{i+j+k=4} c_{i,j,k} \frac{4!}{i!j!k!} u^i v^j w^k, \tag{3.8}
\]

where \( B_{i,j,k}, i + j + k = 4 \) are the quartic Bernstein polynomials associated with \( T \) and \( u, v, w \) are barycentric coordinates with respect to its vertices. We associate the Bézier ordinates \( c_{i,j,k} \) relative to \( T \) with the domain points \( \xi_{i,j,k} := (iP_0 + jP_1 + kP_2)/4 \) in \( T \).

We denote by \( \mathcal{D}_4 \) the union of the sets of domain points for the quartic polynomials associated with each triangle of \( \Delta \), where the domain points in neighboring triangles are not repeated. For the quasi-interpolation scheme, we also need the domain points associated with a cubic polynomial on \( T \): \( \eta_{i,j,k} := (iP_0 + jP_1 + kP_2)/3 \). We denote by \( \mathcal{D}_3 \) the union of the sets of domain points for the cubic polynomials associated with each triangle of \( \Delta \), where again the domain points in neighboring triangles are not repeated.

Due to the uniform structure of \( \Delta \), we can describe both \( \mathcal{D}_3 \) and \( \mathcal{D}_4 \) in convenient forms, which will simplify the description of the scheme. More precisely, setting

**Definition 3.2**

\[
\mathcal{D}^3_{i,j} = \{v_{i,j}, t_{i,j}, b_{i,j}, w_{i,j}^{k,m}, k, m \in \{-1, 0, 1\}, k + m \neq 0\}
\]

where

- \( t_{i,j} \) is the barycenter of \( T_{i,j} \),
- \( b_{i,j} \) is the barycenter of \( B_{i,j} \),
- \( w_{i,j}^{k,m} := (2v_{i,j} + v_{i+k,j+m})/3 \), \( k, m \in \{-1, 0, 1\}, k + m \neq 0 \),

we have

\[
\mathcal{D}_3 = \bigcup_{i,j} \mathcal{D}^3_{i,j}
\]
Each $D^3_{i,j}$ contains nine points depicted as black dots in Figure 3.15a.

![Figure 3.15 Grids for Quartic Quasi-Interpolation Scheme.](image)

Similarly, we can define

**Definition 3.3**

$$D^4_{i,j} = \{v_{i,j}, e_{i,j}^{l,n}, l, n \in \{0, 1\}, l + n \neq 0, v_{i,j}^{k,m}, z_{i,j}^{k,m}, k, m \in \{-1, 0, 1\}, k + m \neq 0\},$$

where

$$e_{i,j}^{l,n} := (v_{i,j} + v_{i+l,j+n})/2, \ l, n \in \{0, 1\}, \ l + n \neq 0,$$

$$v_{i,j}^{k,m} := (3v_{i,j} + v_{i+k,j+m})/4, \ k, m \in \{-1, 0, 1\}, \ k + m \neq 0,$$

$$z_{i,j}^{k,m} := (2v_{i,j} + v_{i+k,j+m} + v_{\xi,\eta})/3, \ k, m \in \{-1, 0, 1\}, \ k + m \neq 0,$$

such that $v_{\xi,\eta}$ is the third vertex of $v_{i,j}v_{i+k,j+m}v_{\xi,\eta} \in \triangle$ counting counterclockwise.

The set of all domain points $D_4$ for the quartic $C^1$ splines associated with $\triangle$ is given as

$$D_4 = \bigcup_{i,j} D^4_{i,j}.$$  \hspace{1cm} (3.9)$$

Each $D^4_{i,j}$ contains sixteen points depicted as black dots in Figure 3.15b.

**Quasi-Interpolation Scheme.** Let values $f(v), v \in D_3$, of a function $f \in C(\mathbb{R}^2)$ be given. In what follows we define a quartic spline $Qf$ on $\triangle$ by setting its Bézier coefficients on each triangle of $\triangle$. Due to the symmetry of $\triangle$, it suffices to describe the setting of
the Bézier ordinates corresponding to one of the domain points denoted as each letter in \( \{v, u, e, z\} \) in \( \mathcal{D}_4 \) defined in (3.9). This is done as follows.

**Definition 3.4** To determine the Bézier coefficient of \( Qf \) associated with a domain point

1. \( v \) in \( \mathcal{D}_4 \), identify the index of the vertex and set \( \mathbf{c}(v_{i,j}) := f(v_{i,j}) \).

2. \( u \) in \( \mathcal{D}_4 \), identify the edge where \( u \) is located and the closest vertex to \( u \) on that edge. Apply the mask shown in Figure 3.16a (or its rotated version) to \( \mathcal{D}_3 \), where \( u \) is marked as an open dot.

3. \( e \) in \( \mathcal{D}_4 \), identify the edge where \( e \) is located, and apply the mask shown in Figure 3.16b (or its rotated version) to \( \mathcal{D}_3 \), where \( e \) is marked as an open dot.

4. \( z \) in \( \mathcal{D}_4 \), identify the triangle where \( z \) is located and the vertex closest to \( z \) in that triangle. Apply the mask shown in Figure 3.17 (or its rotated version) to \( \mathcal{D}_3 \), where \( z \) is marked as an open dot.

In Definition 3.4, applying one of the masks shown in Figures 3.16–3.17 means taking the values of \( f \) at the black dots (i.e., points from \( \mathcal{D}_3 \)) and setting the Bézier coefficient of \( Qf \) associated with the white dot to be the corresponding linear combination of this local portion of data. To be more precise, we provide explicit formulae for particular sites of \( u, e \) and \( z \) shown as open dots in Figures 3.16–3.17, and located in the hexagon of
Figures 3.3–3.15. In the following, \( c(\xi) \) stands for the Bézier coefficient associated with the domain point \( \xi \in D_4 \).

**Figure 3.17** Evaluation of the Bézier coefficient associated with \( z \) shown by the open dot.

**Definition 3.5**

\[
c(u_{i,j}^{1,1}) := (48f(v_{i,j}) + 36[f(w_{i,j}^{1,1}) - f(w_{i,j}^{-1,-1})] + 4[f(v_{i+1,j+1}) - f(v_{i-1,j-1})] \\
+ 18[f(w_{i,j}^{1,0}) + f(w_{i,j}^{0,1}) + f(w_{i-1,j-1}^{1,1}) - f(w_{i,j}^{-1,0}) - f(w_{i,j}^{0,-1}) - f(w_{i+1,j+1}^{-1,1})] \\
+ 9[f(w_{i-1,j}^{1,0}) + f(w_{i,j}^{0,1}) - f(w_{i+1,j}^{1,0})f(w_{i,j+1}^{0,-1})] \\
+ 2[f(v_{i+1,j}) + f(v_{i,j+1}) - f(v_{i,j-1}) - f(v_{i-1,j})]/48.
\]

\[
c(e_{i-1,j-1}^{1,1}) := (54[f(t_{i-1,j-1}) + f(b_{i-1,j-1})] - 18[f(w_{i,j}^{-1,-1}) + f(w_{i-1,j-1}^{1,1})] \\
- 9[f(w_{i,j}^{-1,0}) + f(w_{i,j}^{0,-1}) + f(w_{i-1,j-1}^{1,0}) + f(w_{i-1,j-1}^{0,1})] \\
+ f(w_{i,j-1}^{-1,0}) + f(w_{i,j}^{0,1}) + f(w_{i,j}^{1,0}) + f(w_{i-1,j}^{0,-1})] \\
+ 8[f(v_{i,j}) + f(v_{i-1,j-1})] + 4[f(v_{i,j-1}) - f(v_{i-1,j})])]/24.
\]

\[
c(z_{i,j}^{1,0}) := (108f(b_{i,j}^{1,0}) + 32f(v_{i,j}) - 27[f(w_{i,j}^{0,1}) + f(w_{i,j}^{1,0})] \\
- 18[f(w_{i,j}^{-1,0}) + f(w_{i,j}^{0,1}) + f(w_{i+1,j}^{1,1}) + f(w_{i+1,j}^{-1,1})] + 10[f(v_{i+1,j}) + f(v_{i,j-1})] \\
+ 9[f(w_{i-1,j}^{1,0}) + f(w_{i,j+1}^{0,-1})] - 2[f(v_{i,j+1}) + f(v_{i-1,j})])]/48.
\]
The above setting uniquely determines every Bézier coefficient of the resulting quartic continuous spline \( Qf \) on \( \triangle \). Obviously, for each triangle \( T \) in \( \triangle \), the polynomial piece \( Qf|_T \in \mathcal{P}_4 \) is immediately obtained from a small local portion of the given data; see Figure 3.18. Moreover, this construction of the Bézier ordinates of \( Qf \) ensures \( C^1 \) continuity.

![Figure 3.18](image)

**Figure 3.18** A data set required by control points of a single Bézier triangle; data points, filled dots, on the left and resulting control points, open dots, on the right.

**Properties of The Quasi-Interpolation Operator.** We can now define a linear operator \( Q \) mapping \( C(\Omega) \) into the space \( S^1_4(\triangle) \). More precisely, for each \( f \in C(\Omega) \), we define \( Q(f) := Qf \), where \( Qf \in S^1_4(\triangle) \) is the quasi-interpolating spline of Definition 3.1. In the following, we refer to \( Q \) as the quasi-interpolation operator associated with the scheme described above.

**Theorem 3.6 ([59])** The quartic spline \( Qf \) is in \( C^1(\mathbb{R}^2) \).

**Lemma 3.7 ([59])** For any \( p \in \mathcal{P}_3 \), we have \( Q(p) = p \).

**Remark 1** The choice of masks in Definition 3.5 comes from utilizing the idea of setting all the Bézier coefficients of a spline in such a way that both the smoothness conditions of Theorem 3.6 and the reproduction properties of Theorem 3.7 are satisfied simultaneously.

**Lemma 3.8 ([59])** Let the operator \( Q \) mapping from \( C^1(\mathbb{R}^2) \) into \( S^1_4(\triangle) \), where both spaces are equipped with \( \infty \)-norm, be as in Definition 3.4. Then for any \( T \in \triangle \)

\[
||Q(f)||_T \leq 10||f||_{\Omega_T},
\]
where $\Omega_T$ is the union of the triangles in $\Delta$ having a non-empty intersection with $T$.

Now, we give error bounds for $f - Q(f)$ and its derivatives, which show that the quasi-interpolating spline $Q(f) := Qf$ as well as its piecewise derivatives yields nearly-optimal approximation order for smooth functions. Since our next result also includes an error bound for derivatives of order higher than one, and these derivatives of $Qf$ are not continuous across the edges of the triangles of $\Delta$ in general, we provide local estimates. For simplicity, all norms are $\infty$-norms, where for $f \in C^k(\mathbb{R}^2)$, we let

$$
||D^k f||_B := \max\{||D^\alpha_x D^\beta_y f||_B : \alpha + \beta = k\}
$$

for any compact subset $B$ of $\mathbb{R}^2$.

**Theorem 3.9 ([59])** Let $T$ be an arbitrary triangle in $\Delta$, and $\Omega_T$ be as in Lemma 3.8. Then, there exists an absolute constant $K$ such that for every $f \in C^{m+1}(\mathbb{R}^2)$, $0 \leq m \leq 3$,

$$
||D^\alpha_x D^\beta_y (f - Q(f))||_T \leq K||D^{m+1} f||_T h^{m+1-\alpha-\beta}, \quad \alpha + \beta = 0, 1, \ldots, m.
$$

For proofs of the above theorems see [59].

**3.3.6 Adaptive Quasi-Interpolating Quartic Splines**

The quasi-interpolation scheme based on quartic Bernstein-Bézier patches is a $C^1$-continuous, completely local, and fourth-order approximation method. It is very useful for approximating data of different types. However, for some spline spaces with complex structure it is difficult or sometimes even impossible to apply this concept. A local adaptation of resolution is crucial for modeling surfaces of varying geometric complexity. The adaptation should be driven by this geometrical complexity: regions of interest and visibility in a rendering or other computational application should be resolved more finely.
In [30], Hering-Bertram et al. present an adaptive quasi-interpolation quartic spline construction for regularly sampled surface. The method is based on the quartic spline scheme introduced above; see Subsection 3.3.5. This adaptive quasi-interpolation construction provides local control; it offers flexibility for local resolution adaptation. The entire surface is \(C^1\)-continuous. The construction can be applied to both parametric and scalar surfaces. The mesh is refined in a pseudo-regular manner. At \(T\)-nodes, a point where two meshes of different refinement meet, it is difficult to achieve continuity, much less smoothness; see Figure 3.19. This algorithm overcomes the problem by subdivision of quartic patches using the de Casteljau algorithm.

Given a set of regularly gridded data points \(\xi_{i,j,k} \in \mathcal{D}_4\), let \(s_0\) denote the full-resolution spline surface approximation on the \(\mathcal{D}_4\) grid. Analogously, we define a sequence of splines \(s_l, l = 1, 2, \ldots\) from the subset \(\xi_{2i,2j,2k}\). In this hierarchy, every triangular patch \(T\) in the domain of \(s_{l+1}\) overlaps with four patches of \(s_l\). Let \(T\) be the triangle \(P_0, P_1, P_2\). Denote middle point of the edges as

\[
\tilde{P}_0 = (P_1 + P_2)/2, \\
\tilde{P}_1 = (P_0 + P_2)/2, \\
\tilde{P}_2 = (P_0 + P_1)/2.
\]

Then the four triangles in the next finer representation \(s_{l+1}\) have the coordinates

\[
T_0 := (P_0, \tilde{P}_1, \tilde{P}_2) \\
T_1 := (\tilde{P}_0, P_1, \tilde{P}_2) \\
T_2 := (\tilde{P}_0, \tilde{P}_1, P_2) \\
T_3 := (\tilde{P}_0, \tilde{P}_1, \tilde{P}_2)
\]

The fourth sub-triangle is located in the middle of the others with flipped orientation in both principal axes; see Figure 3.9.
The algorithm starts by calculating the spline $s_{l_0}$ at a coarse base level $l_0$, where the finer representations are needed only for local evaluation, on demand. The preliminary strategy can be summarized as follows:

- Determine a subset of patches in $s_{l_0}$ to be subdivided, based on a refinement strategy.
- Recursively replace and subdivide these patches by locally evaluating the spline and the underlying function on the next finer levels.
- Enforce $C^1$ continuity at the boundaries between different resolutions.

The first two steps are rather simple. For its refinement condition, the method uses the maximum error of $s_{l}|_{\mathcal{T}}$ out of all data sites $\eta_{i,j} \in \mathcal{D}_3$ located inside $T$,

$$
\varepsilon(T) := \max_{\eta_{i,j} \in \mathcal{T}} \left\{ ||s_{l}(\eta_{i,j}) - f_{i,j}|| \right\} .
$$

All patches with an approximation error exceeding a prescribed tolerance are then split. For a large number of levels $l_0$, this strategy results in a computation time of $O(n \log n)$ in the number of data points at full resolution. We note that simpler alternative strategies, such as predefined regional subdivision levels, may result in linear time complexity, even if all levels are dense.

![Figure 3.19](image)

**Figure 3.19** Bézier patches of different resolution, one level difference.

The most difficult step is maintaining the $C^1$ continuity of this hierarchy of Bernstein-Bézier patches. A first idea would be to detect all edges with $T$-nodes, i.e., edges where different resolutions have been used on opposite sides; see Figure 3.19. Starting with the
finest subdivision level that both sides have in common, say level \( l \), one could calculate two rows of Bézier coefficients from \( s_l \) along the edge on the side to be subdivided. This means that the refined patch inherits the values and cross-boundary derivatives along this edge from \( s_l \). Unfortunately, this approach fails, since the altered two rows of control points violate the continuity condition inside the subdivided patch. In general, the sub-triangle \( T_3 \) in Equation (3.10) cannot be constructed that join \( C^1 \)-continuously with the remaining sub-triangles.

To solve this problem, the coarser side along edges with different resolutions on both sides needs to inherit the values and cross-boundary derivatives of the finer resolution. This requires the splitting of adjacent patches down to the same resolution. Again, constructing \( C^1 \) constraints by locally exchanging Bézier coefficients from different resolutions can require a sequence of other coefficients to be altered, causing undesirable oscillations in the surface. Hence, fixing continuity based on local operations on Bézier ordinates is generally not a good strategy.

What helps is the observation that the uniform spline construction is locally supported by a single stencil of patches for each coefficient. Hence, the most natural choice for blending between different resolutions is the use of the underlying basis functions of the uniform quasi-interpolating spline construction for blending. Therefore, we redefine the algorithm as follows:

- subtract the coarsest spline representation \( s_{l_0} \) from the given data,
  \[
  f_{i,j} := f_{i,j} - s_{l_0}(\eta_{i,j}).
  \]

- Recursively identify the set of patches to be split and set all data points \( f_{i,j} \) located outside this area to zero.

- Compute the next finer approximation level for the region influenced by the nonzero \( f_{i,j} \) and continue with the remaining approximation errors as data points.
This strategy results in a coarse approximation \( s_{l_0} \) and a sequence of locally supported correction functions \( \triangle s_{l_0-1}, \ldots, \triangle s_0 \), each of which decays to zero with \( C^1 \) continuity at the boundaries of its support. Hence, the final representation

\[
s_{\text{adaptive}} := s_{l_0} + \sum_{l=0}^{l_0} \triangle s_l
\]

is \( C^1 \)-continuous and can be locally evaluated. The summation is calculated automatically during the refinement process. For more details of the above method see [30].

3.4 Numerical Tests of Proposed Tools

As in Section 2.4 we perform our initial numerical experiments on curves with explicit parameterization. We implement and test algorithms and methods described in Sections 3.2 and 3.3. These tests allow us to understand behavior of the implemented methods and algorithms and to gain a bit more control over the testing processes. Finally, the test helps us to choose the most suitable methods for our purposes.

3.4.1 Model Surfaces

We introduce a few parametric model surfaces given by \( (x, y, z) = (x(u, v), y(u, v), z(u, v)) \).

The general case test surface, which we called “plate” is given in the parametric form

\[
S_{\text{plate}} = \{(x, y, z) : x = v \cos u, y = v \sin u, z = [\cos(r(v + u)) + 1]e^{-2v^2/r}\},
\]

where \( r = \text{const} \); see Figure 3.20a. The surface changes with a varying geometric complexity; its flat pieces at the outer regions and folded, curly behavior at the inner region are well suited to test the adaptive method.

Both numerical schemes, the quasi-interpolation based on the quartic triangles, Subsection 3.3.5, and its adaptive version, Subsection 3.3.6 require knowledge of the values on a slightly wider region in parameter space than that on which the approximate surface is generated. This is not a case if the surface is periodic in both \( u \) and \( v \).
Figure 3.20  General and fully periodic parametric surfaces generated by the quasi-interpolation method described in Subsection 3.3.5; (a) plate surface given by the Equation (3.13), $r = 10$, $(u, v) \in (0, 2\pi) \times (0, 6)$ and sampled on the 48 $\times$ 48 grid; (b) pretzel given by the Equation (3.14), $a = .6, n = 1.5, r_1 = .4, r_2 = .4$, $(u, v) \in (0, 4\pi) \times (-2\pi, 2\pi)$ and sampled on the 18 $\times$ 54 grid.

The fully periodic case is tested on two parametric surfaces. One with slightly more complex shape which we call “pretzel” given by

$S_{\text{pretzel}} = \{(x, y, z) : x = [r_2 \cos v \cos u + r_1 \cos u(1 + a \cos nu)]/r,$

$y = [r_2 \sin v + a \sin(nu)]/r,$

$z = [r_2 \cos v \sin u + r_1 \sin u(1 + a \cos nu)]/r\}$, 

(3.14)

where $r = r(x, y, z) = x^2 + y^2 + z^2$ and $a, n, r_1, r_2$ are constants that denote an amplitude of twist, the number of twists and the major and minor radius, respectively; see Figure 3.20b.

Next, is a torus given by the equation

$S_{\text{torus}} = \{(x, y, z) : x = a + r \cos v \cos u, y = a + r \cos v \sin u, z = r \sin(v)\}$, 

(3.15)

where the constants $a$ and $r$ denote the major and minor radius of torus, respectively.
In Section 1.2 we briefly described the idea of construction of a method for computing the two-dimensional invariant manifolds. The manifold can be seen as the union of surfaces with annular topology. In this manner each annulus can be considered as an annular parametric surface. Therefore, the final example represents a surface of an annular topology. The surface which we called “plait” is given by

\[
S_{\text{plait}} = \{(x, y, z) : x = a \cos u \cos v, y = a \sin u \cos v, z = a \sin v + bu\}, \tag{3.16}
\]

where \(a, b = \text{const}\); see Figure 3.22a.

### 3.4.2 Comparison Tests of Methods

We test our implementation of the previously introduced schemes using the model parametric surfaces to check their accuracy. First, we perform visual test applying all the schemes to generate a torus. Starting with uniform distribution of parametric data we generate type-1 triangulation in a parametric space and approximate the surface using the Bivariate Linear Interpolation, the Nine-parameter Interpolation, the Shirman-Sequin method and the Quartic Quasi-Interpolation, respectively; see Figure 3.21.

That simple test shows that the Quartic Quasi-Interpolation, see Figure 3.21, performs better than all other methods for a given set of data points, although it of course requires more work than Bivariate Interpolation. Although, the shape is not perfectly circular, the visual appearance shows the fewest artifacts.

Next, we perform a quantitative test. In Figure 3.22b, we plot the maximum error \(\varepsilon = \max_{(u,v) \in \Omega} |S_{\text{exact}}(u,v) - S_{\text{approx}}(u,v)|\) as a function of the number of points \(N\) used to generate \(N \times 2N\) mesh grid. The error is computed at a finite set of points and the maximum over this set is taken. The presented test is performed on a surface of an annular topology, Equation 3.16, however, the same tests for the general and the fully periodic parametric surface confirm these results, showing that the rate of convergence is unaffected by boundaries.
Figure 3.21  Reconstruction of a torus given by Equation (3.15) using different schemes on the $15 \times 6$ uniform parametric data grid, $a = 2, r = 1$.

First, it is clear that the Quartic Quasi-Interpolation method converges much faster than any other method; it demonstrates fourth order convergence. This result confirms what is stated in Theorem 3.9. The convergence of other methods is fairly slow, the rate of the convergence for Nine-parameter and Shirman-Sequin methods is the same order as for a simple interpolation with plane triangles. Note that the Nine-parameter and the Shirman-Sequin methods require data points with normal directions to be given at the initial step. In general, the normal direction is not given and needs to be approximated based on the triangulation. The poor performance of these methods is caused by the low order of the normal estimation. The normal estimation methods described in Subsection 3.2.6 are only second-order. The similar behavior of these methods is not surprising since they start with the same initial steps. It seems that the Shirman-Sequin method makes only a visual improvement over Nine-parameter Interpolation. Note that
in case of an explicit parametric surface we can modify the method by exactly specifying
the normal at each data point. This approach improves these methods to third-order.

![Image of a surface](image)

**Figure 3.22** An annular test surface generated by the quasi-interpolation method de-
scribed in Subsection 3.3.5; (a) plait surface given by the Equation (3.16), \(a = 3, b = 2,
(u, v) \in (0, 4) \times (0, 2\pi)\) and sampled on the \(9 \times 18\) grid; (b) convergence introduced
methods tested on the “plait”.

Disappointing visual and quantitative performance disqualify the Nine-parameter
and the Shirman-Sequin methods from further use. The only advantage of these method
is that they can be applied over arbitrary triangulations. We recommend the Nine-
parameter Interpolation as a tool for visualization of scattered data because of its straight-
forward implementation. The high accuracy and good visual performance of the Quartic
Quasi-Interpolation, see Figures 3.20, 3.21d and 3.22, suggest its further use. This
method is limited to type-1 triangulations. However, we can still easily combine it with
the Parameterization Method from Subsection 3.1.1.

### 3.4.3 Details of the Adaptive Quasi-Interpolating Quartic Splines

In this subsection we describe our implementation of the Adaptive Quasi-Interpolating
scheme introduced in Subsection 3.3.6. The method is based on the Quartic Quasi-Inter-
polation from Subsection 3.3.5. Preliminary visual and quantitative numerical tests above
show this quasi-interpolation scheme performs very well.
The goal is to adaptively approximate parametric surfaces of varying geometric complexity, where the local resolution is driven by the approximation error. The construction of high-quality spline surfaces is enhanced by the flexibility of adaptive pseudo-regular triangle meshes.

The first step of the method is to generate the Quartic Quasi-Interpolation approximation surface. Hence, for our purposes we define the type-1 triangulation on a rectangular parametric domain; see Figure 3.23. Remember that the method constructs approximating surface over any triangle $T$ based on data from $T$’s neighbors; see Figure 3.18. Thus if $T$ is an the boundary, it needs data from additional strip.

(a) Parametric coordinates of data points. (b) Parametric coordinates of control points.

**Figure 3.23** Points grid in a parametric space for Quartic Quasi-Interpolation Scheme – a surface of an annular topology periodic in the vertical direction; (a) domain of the data points, (b) domain of the generated control points, an approximated surface over the shaded region.

Note that in the case of an annular topology, a surface is approximated over inner triangles, shaded region in Figure 3.23b; there are strips of boundary data necessary for determining control points on the right and left edges where no approximation is computed. In the fully periodic case, the approximation of the surface is generated over the whole parameter domain; extra strips of triangles are unneeded. For a surface with boundaries we would get approximation of the surface only over the set of inner triangles and no approximation over the boundary triangles. However, if we need to define the
surface over the whole data domain we can extrapolate data over the larger domain used by the scheme. For instance, we can use cubic extrapolation along the lines of grid points.

We sample a test function on the mesh shown in Figure 3.23a, there are ten data points for each triangle. Next, according to the procedure described in Subsection 3.3.5, for each single triangle fifteen quartic Bézier control points are generated. The test surface is approximated by the $C^1$ composite quartic triangular surface. In order to draw the approximation surface, each Bézier triangle is sampled at uniformly distributed barycentric coordinates using the de Casteljau algorithm; see Subsection 3.2.5. Note that the points can also be evaluated using the Bernstein-Bézier form of a triangular patch; see Equation (3.8). However, the de Casteljau algorithm provides a stable and more efficient method of evaluation. Next, we use bivariate linear interpolation to connect the set of points on a single triangular Bézier patch. The resolution of each patch is controlled by the number of sampled points.

Next, we use relation (3.11) for each Bézier triangle and calculate the error between the approximation and the exact data points. Note that for each triangular patch we are given ten data points in its domain. A patch interpolates the three corner points, so we measure the error only at seven (six at boundaries and one at the center) points. The value of the approximation patch at these points is Bézier triangle evaluated at barycentric coordinates $(\frac{2}{3}, \frac{1}{3}, 0)$, $(\frac{1}{3}, \frac{2}{3}, 0)$, $(0, \frac{2}{3}, \frac{1}{3})$, $(0, \frac{1}{3}, \frac{2}{3})$, $(\frac{1}{3}, 0, \frac{2}{3})$, $(\frac{2}{3}, 0, \frac{1}{3})$ for the boundary and $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ for the center. In order to evaluate the approximated surface at these points we use the de Casteljau algorithm. This step determines set of the triangles which needs to be corrected, i.e., where the mesh has to be refined and the next level correction surface has to be computed.

Continuing, we refine the whole mesh in parametric space in a such way that any triangle from the previous triangulation is covered by four new triangles; see relation (3.10). The values on the new mesh are assumed to be zero everywhere outside the domain of the correction set and values inside the refinement set are computed as the difference
between the data and the previous approximation surfaces. Note that evaluation of a single correction triangle has to be done over four triangles from the finer mesh, i.e., over twenty five points. Next, based on these values, the correction surface is generated by Quartic Quasi-Interpolation. Note that this surface is defined on the whole domain, but its value is zero everywhere except the triangular correction domain and, crucially, one row of triangles around this domain. Even though the data values are zero, the newly created set of control points is based on a larger, nonzero stencil. In order to combine both the approximation surface and its correction at the same level of refinement we need to subdivide using the de Casteljau algorithm (4-to-1 subdivision from Section 3.2) all the triangles from the previous unrefined surface affected by the correction. After the correction is done, we repeat the error estimation procedure, treating the previous correction surface as the data. We proceed until the estimated error decreases below the chosen tolerance everywhere.

For the general and annular surface some additional work has to be done close the boundary. The approximation surface is created only over the inner triangles, so that for any correction surface, over the same domain (shaded region in Figure 3.23b), we need to maintain one external row of domain triangles (three extra rows of data). In order to form a single correction surface data has to be given over the extended triangulation. If no data exists, we need to estimate it or to assume that the previous approximation is exact outside of the original domain, i.e., the correction takes the value zero everywhere outside of the domain. We found that this approach failed: the error persists close to the boundary and it does not decay when the further adaptation steps are applied. Moreover, this error propagates into the domain.

![Figure 3.24](image.png)  
**Figure 3.24** Cubic spline data extrapolation outside the domain $\Omega$, given data points marked by the filled dots, extrapolated data by open dots; a schematic two-dimensional view.
Instead, we construct new data points that extend the data at from the domain out into the boundary triangles. We assume that the data matches smoothly the original data values at the boundary and connects to a zero with zero derivative at outer edge; see Figure 3.24. By that, we assume that the approximation is exact on the boundary and outside of the new, larger domain. This setup suggests to apply the cubic Hermite interpolation along each row. Derivatives at the boundary are approximated using finite differences. This approach works well, the error close to the boundary decays with the same rate as inside of the domain.

Note that this extrapolation is necessary only when correction is needed on a triangle close to the boundary of the domain. In any other case these data are still needed, but their value is zero. Note also that there is no need to do this for the fully periodic case: the correction is also periodic.

The adaptive method produces a $C^1$ approximation surface as the composition of Bézier triangles of different resolution. Now, we need to draw it in such a way that looks good, i.e., we want to avoid any cracks or pinholes. Each Bézier patch has to be perfectly matched with its neighbors. It means that more planar patches need to be used to draw a lower resolution patch. This situation is similar to that in Figure 3.19 but with planar triangles. The resolution needs to be tracked starting from the finest Bézier triangles, consequently doubling the mesh.

We test our implementation on all the previously introduced test surfaces. For instance, Figure 3.25a shows the parametric surface, given by the Equation (3.13), refined adaptively. We can easily notice at least three levels of the refinement, the flat region close to the boundary of the plait has a lower resolution, whereas the middle, curly and folded region has the highest. Next two examples, Figure 3.25b and 3.25c, adaptively present parametric surfaces given by Equations (3.16) and (3.14). In this case we can easily notice only two levels of the refinement, these surfaces do not have such a varying complexity as in the previous example. This test shows that the method itself and
its implementation perform well at refining surfaces where needed without creating any pinholes or artifacts.

Figure 3.25 General, annular and fully periodic parametric surfaces generated by the adaptive method described in Subsection 3.3.6, \( \text{tol} = 10^{-2} \), \( \text{tol} = 10^{-3} \) and \( \text{tol} = 10^{-3} \), respectively; (a) plate surface given by the Equation (3.13), \( r = 10, (u, v) \in (0, 2\pi) \times (0, 6) \); (b) plait surface given by the Equation (3.16), \( a = 3, b = 2, (u, v) \in (0, 4) \times (0, 2\pi) \); (c) pretzel given by the Equation (3.14), \( a = .6, n = 1.5, r_1 = .4, r_2 = .4, (u, v) \in (0, 4\pi) \times (-2\pi, 2\pi) \).

In our Adaptive Quasi-Interpolating implementation we use some ideas of “quadtree” data structures. At each step of refinement we need to subdivide the set of Bézier triangles (4-to-1 subdivision). Each patch for the set is replaced by the four new patches and only these are passed to the further calculations. The data structure is hierarchical with each subdivided patch pointing to its four “children”.
3.5 Details of the Numerical Implementation: Two-dimensional Invariant Manifold Calculation

3.5.1 Parameterization

Assuming the initial primary annulus $U_0 = U_0(r, \theta)$ has been given some parameterization (which is discussed above), let $U_1(r, \theta) = f(U_0(r, \theta))$ for some $r_0 \leq r \leq r_1$ and $0 \leq \theta \leq 2\pi$. Then considering this portion of the manifold as a topologically annular surface centered at the fixed point it is natural to use polar coordinates.

The choice of the initial parameterization is suggested by the problem of computing unstable manifolds. Consider a linear map of the form $x' = Ax$ with a hyperbolic fixed point at the origin with one stable and two unstable directions. Let $\lambda_1$ and $\lambda_2$ be the unstable eigenvalues and $\vec{v}_1, \vec{v}_2$ be its associated eigenvectors, respectively, i.e., $E^u$ is of course equal to span\{\vec{v}_1, \vec{v}_2\}. The initial primary annulus of the unstable manifold can be written as $U_0 = W^u[\Gamma_0, \Gamma_1]$, where

\[
\begin{align*}
\Gamma_0 &= r_0 \vec{v}_1 \cos \theta + r_0 \vec{v}_2 \sin \theta \\
\Gamma_1 &= A \Gamma_0 = r_0 |\lambda_1| \vec{v}_1 \cos(\theta + \theta_0) + r_0 |\lambda_2| \vec{v}_2 \sin(\theta + \theta_0)
\end{align*}
\]

with $r_0 = \text{const}$ and $\theta_0 = \text{arg}(\lambda_1/\lambda_2)$. The annulus creates the elliptical ring which can be written in polar coordinates as

\[
U_0(t, \theta) = (a(t)\vec{v}_1 \cos \theta + b(t)\vec{v}_2 \sin \theta)
\]

where $a(t) = r_0 |\lambda_1|^t$ and $b(t) = r_0 |\lambda_2|^t$ for $\theta \in [0, 2\pi]$ and $t \in [0, 1]$; see Figure 3.26.

For a complex conjugate pair of eigenvalues $\lambda_1 = \overline{\lambda_2}$ the growth of an annulus under the linear map is isotropic since $a(t) = b(t)$. Additionally $\theta_0 \neq 0$ and rotation about this angle is associated with the map.

This example suggests a way to parameterize the initial primary annulus $U_0$. Near the fixed point (which we can assume is $x^* = 0$), the map $f(x)$ is approximately given by $x' = Df(0)x$ so we can approximate $U_0$ by an appropriate disk of the unstable eigenspace.
using the above parameterization. If the initial primary annulus is parameterized by this algorithm then by mathematical induction, this parameterization can be used in all annuli $U_n$; we define $U_{n+1}(t, \theta) = f(U_n(t - 1, \theta - \theta_0))$.

**Figure 3.26** Composite of primary annuli with the initial primary annulus $U_0$ shaded, (a) for a real pair of eigenvalues with an anisotropic growth, (b) for a complex pair of eigenvalues with more an isotropic growth.

### 3.5.2 Notation

This notation differs slightly from that in Subsection 2.5.2 for computing one-dimensional invariant manifolds. Let $U_n$ be the approximation to the $n$th true annulus $U_n$. This approximation is defined adaptively as the composite quartic Bézier surface in two steps. Given is the initial set of points

$$X_n^0 = \{x_k^n\}_{k=0}^N$$

at the corresponding uniformly distributed parameter grid

$$T_n^0 = \{(t, \theta)_k^n\}_{k=0}^N$$

and the underlying type-1 triangulation. On each annulus the initial type-1 triangular mesh is chosen identically, consisting of three columns of triangles per annulus, where the first and the last column are the buffer triangles for the quasi-interpolation method; see Figure 3.23a. First, we define the operator $A$ that adaptively, based on a specified tolerance $\text{tol}$, adds points to the initial sets generating a pseudo-regular parameter grid
\( \mathcal{T}_n^A \) and the associated points \( \mathcal{X}_n^A \), i.e.,

\[
(\mathcal{T}_n^A, \mathcal{X}_n^A) = A(\mathcal{T}_n^0, \mathcal{X}_n^0, \text{tol}).
\]

Next, we define operator \( QI \) that constructs the composite quartic Bézier surface over the pseudo-regular parametric grid \( \mathcal{T}_n^A \) and the corresponding set of points \( \mathcal{X}_n^A \), i.e.,

\[
U_n = QI(\mathcal{T}_n^A, \mathcal{X}_n^A).
\]

A composite of these two operators we denote \( A-QI \). Additionally, we introduce two pieces of information

\[
\mathcal{T}_n^0 + 1 = \mathcal{T}_n^0 + (1, \theta_0) = \{(t^n_k + 1, \theta^n_k + \theta_0)_k\}^N_{k=0} \quad \text{and} \quad \mathcal{X}_n^0 = f(\mathcal{X}_n^0) = f^n(\mathcal{X}_0^0).
\] (3.17)

Thus we may construct the approximate invariant manifold using an inductive procedure. First, we use the rendering operator \( A \), \( (\mathcal{T}_0^A, \mathcal{X}_0^A) = A(\mathcal{T}_0^0, \mathcal{X}_0^0, \text{tol}) \), and the surface operator \( QI \) to construct the initial primary annulus \( U_0 = QI(\mathcal{T}_0^A, \mathcal{X}_0^A) \). Next, for given \( U_n \), we use the operator \( A \) to find \( \mathcal{T}_{n+1}^A \) and \( \mathcal{X}_{n+1}^A \) and the operator \( QI \) to construct \( U_{n+1} = QI(\mathcal{T}_{n+1}^A, \mathcal{X}_{n+1}^A) \).

3.5.3 Initial Primary Annulus

In order to initialize any adaptive method for computing an invariant manifold, one first has to determine the initial primary annulus \( U_0 \), an approximation to the true annulus \( U_0 \), as is described in Section 1.2.

Crucially, this method requires the initial primary annulus \( U_0 \) in a parametric form. Hence, the parameterization method described in Subsection 3.1.1 establishes the fundamentals of the method. The stable or unstable manifolds of map \( f \) at a fixed point \( \mathbf{x}^* \) is represented by a parametric function \( P(r, \theta) \). Even though the parameterization \( P \) is entire (see [45]), in practice, its computational radius of convergence is finite and small due to roundoff errors, and the truncated series only approximates the desired
parameterization. In order to make practical use of these truncated series, we have to set the parameter domain for the initial annulus on which the approximation is reasonable. It is crucial to determine $U_0$ very accurately in order to avoid error propagation in the ensuing computations. Therefore, the initial annulus must be taken close to the fixed point, where the truncated series of the higher-order approximation is valid.

Choosing $\gamma_0$ as described in Subsection 3.5.1, the very first ring around the origin in the parametric space, the closed curve $\Gamma_0$ at the unstable manifold (the interior boundary curve of the initial annulus $U_0$) is determined by the expansion $P$ as $\Gamma_0 = P(\gamma_0)$. Here, any $\gamma_i$ is a closed curve in the $(r, \theta)$-parameter space, whereas $\Gamma_i$ is a closed curve in the physical space and they are related by the expansion $P$, i.e., $\Gamma_i = P(\gamma_i)$; see Figure 1.3. The ring $\gamma_1$ in the parametric space is determined by the choice of the ring $\gamma_0$ by Equation 3.17 and its image $\Gamma_1 = P(\gamma_1)$ defines the other boundary of the initial annulus. Then the whole initial primary annulus $U_0 = W[\Gamma_0, \Gamma_1]$ is determined by the annulus in the parametric space and by the expansion $P$. Note, that the high accuracy of the expansion $P$ gives $\Gamma_1 = f(\Gamma_0)$. The annulus between $\gamma_0$ and $\gamma_1$ is defined for parameters $(r, \theta) \in [r_0, |\lambda| r_0] \times [0, 2\pi]$ or equivalently $(t, \theta) \in [0, 1] \times [0, 2\pi]$; see Subsection 3.5.1.

In order to construct an accurate approximation $U_0$ we sample our parametric domain $(t, \theta)$ on the uniform grid with underlying type-1 triangulation mesh; see Figure 3.23a. This grid defines the sets $T_0^0$ and $X_0^0 = P(T_0^0)$. The composite of the operators A and QI, $A-QI(T_0^0, X_0^0, \texttt{tol})$, generates the approximation $U_0$ as a composite Bézier surface.

In the case of complex eigenvalues $\lambda_1 = \overline{\lambda_2}$ the rotation about the angle $\theta_0 = \arg(\lambda_1/\lambda_2)$ is associated with the map. Any point from one boundary curve $\Gamma_0$ given by $P(r_0, \theta)$ for $\theta \in [0, 2\pi]$ is associated with its image point (under the map $f$) at the other boundary curve $\Gamma_1$ given by $P(r_1, \theta)$ for $\theta \in [\theta_0, \theta_0 + 2\pi]$. We want to construct a nice looking surface, without cracks and pinholes, so we need to match these points in the parametric space. Unfortunately, if we try to directly match a point with its image under the map $f$, a type-1 triangulation on the parallelogram grid will produce
long, narrow triangles in physical space. If we set the triangulation on the rectangular grid it will cause inconsistence between triangular patches on the consecutive annuli; see Figure 3.27a. In order to avoid this problem we need to carefully choose the parameter values for the initial primary annulus $U_0$.

Figure 3.27  Schematic settings for the initial primary annulus $U_0$, (a) five primary annuli in a physical space, dash line – the edges over a parallelogram grid with $\theta_0$, solid bars – edges over a rectangular grid, (b) the triangulation of the parallelogram grid in the parametric space, the buffer dash triangles in the first and the last columns, (c) four primary annuli in a physical space, rotation of a primary annulus about $\theta_0$ under the map $f$ with a multiplicity $m = 3$ and a correction $\omega_0$; $N_T = 12$ then every mapping the rotation about four triangles.

Let $m$ be the closest integer to $2\pi/\theta_0$, i.e., $m = \text{round}(2\pi/\theta_0)$, then $m$ is a multiplicity of the map $f$. There is a full rotation around the fixed point after $m$ mappings applied. To make that rotation matching points with their images exact we introduce a small correction $\omega_0$ to the rotation,

$$\omega_0 = (m\theta_0 - 2\pi)/m.$$

Next, sample the expansion $P(r, \theta)$ on the parallelogram grid, i.e., $\theta \in [0, 2\pi]$ for $\gamma_0$ and $\theta \in [\omega_0, \omega_0 + 2\pi]$ for $\gamma_1$; see Figure 3.27b. After $m$ mappings applied the points are on the same side of the fixed point as initially and, because of the correction, the initial points match with their images exactly; see Figure 3.27c.
Additionally, we need to make sure that the matching is valid for each triangle of the triangulation, thus the number of triangles in each column also depends on \( \theta_0 \) and must be a multiple of \( m \). For each application of the map, annular mesh is rotated about \( N_T/m \) triangles, where \( N_T \) is the number of triangles in a column.

Note that the proper choice of the parameter values is crucial only for the initial primary annulus \( U_0 \) and it guarantees a continuous strip of Bézier triangles on the whole manifold. During the process of creating further annuli, the algorithm uses only the parameters \( T \) in their barycentric coordinates with respect to the triangular mesh, so the phase term is irrelevant.

The portion of the manifold between the fixed point and \( \Gamma_0 \), the boundary edge of the initial annulus, is represented by the expansion \( P(r, \theta) \). Mireles-James and Lomelí in [45] showed that even for significant radius, the expansion can be accurate up to 15 digits. Then, the choice of the \( r_0 \), the radius for \( \Gamma_0 \), does not have to be “small.” This piece, which we call the “initial portion” of the manifold, can be simply represented by applying Adaptive Quasi-Interpolation on the surface given by \( P(r, \theta) \) with \( r \in (\varepsilon, r_0) \). In order to have this part of the manifold given in a manner consistent with the rest of the manifold we choose to set \( \varepsilon = r_0/|\lambda|^K \). Then, this part of the manifold can be seen as the composite of \( K \) primary annuli.

### 3.5.4 Resolving a Primary Annulus

We restrict our attention to the specific problem of finding the unknown primary annulus \( U_{n+1} \) which is the approximate image under the map \( f \) of an already-resolved primary annulus \( U_n = A\text{-QI}(T_n^0, X_n^0, \text{tol}) \), i.e., \( U_{n+1} \approx f(U_n) \). Mapping the segment \( U_n \) forward yields a parametric surface \( \tilde{U}_{n+1} = f(U_n) \) which approximates the next primary segment \( U_{n+1} \). At each primary annulus we start with a mesh \( (T_n^0, X_n^0) \) identical to that defined on the initial primary annulus. We apply operators \( A \) and \( \text{QI} \) to produce a resolved approximation \( U_{n+1} \) to the desired parametric surface \( f(U_n) \). The manifold is then approximated
by the composite of $C^1$ surfaces

$$U \approx \bigcup_{n=0}^{N} U_n.$$ 

### 3.5.5 Joint Patching

The proposed method is quite capable of resolving a single primary annulus of a manifold given a previously completed annulus. The annuli $U_n$ and $U_{n+1}$ may be individually resolved but the composite surface formed by their union $U_n \cup U_{n+1}$ may lose the continuity along the boundary. This happens when independently resolved annuli are represented with a different resolution close to the boundary; see Figure 3.19. The size of the gap is generally smaller than the tolerance which was used to generate these annuli. Therefore, some care must be taken to ensure that the composite surface formed by two successive primary annuli is at least continuous. This can be easily accomplished by refining the edges between annuli to the finest resolution on both sides and replacing the points at the inner edge of $U_{n+1}$ by points from outer edge of $U_n$. Note, these boundary curves are composite Bézier curves, so they can be refined using the one-dimensional de Casteljau algorithm of Subsection 2.2.2.1. This approach is straightforward and is computationally inexpensive. The manifold loses $C^1$ continuity on across the annular joints. We nonetheless decide to use this technique as a last step of our proposed method.

In the case of the map with rotation, this part has to be done carefully. Because of the rotation, matching of the boundary Bézier triangles is not automatic. At every annular joint we need to track the rotation angle, i.e., the number of triangles about which the consecutive annulus is rotated; see Figure 3.27b.

The way to maintain the $C^1$ continuity over the whole manifold would be to refine the annuli down to the finest resolution, satisfying $C^1$ continuity. Then, we may coarsen the joint surface, again. This approach is significantly more complex and computationally expensive than simply maintaining $C^0$ continuity on across the annular joints.
3.5.6 Distance Control along the Manifold

The method is based on simple forward mapping of a primary annulus. Such an annulus does not grow uniformly in all radial directions; see Figures 3.28a and 3.28b. Away from the fixed point, the iterates of the initial annulus may not look like a typical annulus at all. The shape of successive images could be very complicated. For example, the image of the initial primary annulus can grow a “finger” that stretches and that could wind around the computed portion of the manifold several times. We discuss such a behavior of the invariant manifold in case of the first map in the next section.

We partially solved this problem in our implementation A-QI. In order to accelerate the computation we can neglect the situation when errors above the tolerance occur at points sufficiently far from the fixed point. Using properties of Bézier triangles we can control the distance between each quartic triangle and the fixed point along the manifold; for example see Figure 3.29b. Note that the edges of Bézier triangles form composite Bézier curves situated on the manifold, and their length can be easily approximated using the Gravesen algorithm [22]. This distance is not the same as the geodesic distance; however it can be used to bound the distance between a point along the manifold and the fixed point. Similarly, for a drawing process we can avoid plotting parts that have drifted far away. An improvement of this step may significantly improve the proposed method.

3.6 Numerical Tests

Testing a method for computing two-dimensional invariant manifolds is much harder than testing in the one-dimensional case. The authors of the previous methods introduced in Section 3.1 provide some numerical tests of their methods.

Mireles and Lomelí in [45] well tested the parameterization method providing the quantitative error analysis by evaluating the residual. However, these tests show the validity of the method locally inside the series convergence region for $P(r, \theta)$. The accuracy
of the global portion of the manifold is assumed, based on high accuracy of the local approximation. Each new point on the manifold is generated by mapping new point from initial primary annulus. Krauskopf and Osinga do even less — their method is only tested by visualizing a few examples of invariant manifolds. They do not provide any numerical error analysis.

We performed a few numerical tests showing performance of the proposed method on the global portion of the invariant manifolds; the results are described below.

3.6.1 Example 1: Volume-Preserving Hénon Map

In order to demonstrate the utility of the proposed method, first we follow Mireles-James and Lomelí [45] and consider the volume-preserving Hénon family introduced in [38]. These maps are defined by

\[
\begin{pmatrix}
  f_1(x, y, z) \\
  f_2(x, y, z) \\
  f_2(x, y, z)
\end{pmatrix} =
\begin{pmatrix}
  x \\
  y
\end{pmatrix},
\]

where \(a + b + c = 1\) (to guarantee volume preservation). The relevance of the volume-preserving property is to guarantee that a generic fixed point has the features of interest: namely a two dimensional unstable manifold and one dimensional stable manifold or vice versa. The map is quadratic with an explicit quadratic inverse.

The Hénon map has hyperbolic fixed points \(f(x_\pm, x_\pm, x_\pm) = (x_\pm, x_\pm, x_\pm)\) where \(x_\pm = -\tau/2 \pm \sqrt{\tau^2 - 4\alpha}/2\). The Jacobian matrix is

\[
Df(x, y, z) = \begin{pmatrix}
  \tau + 2ax + by & 2cy + bx & 1 \\
  1 & 0 & 0 \\
  0 & 1 & 0
\end{pmatrix}.
\]

As the determinant of the Jacobian is identically one; generically it will have either three real eigenvalues, or one real eigenvalue and one complex conjugate pair. Since the
product of the three must be one, it has either a two dimensional stable eigenspace, or a
two dimensional unstable eigenspace.

We implement the parametrization method described in Subsection 3.1.1 to ap-
proximate the initial annulus of $W^n(x_+)$ and truncate the series $P(r, \theta)$ at order 60.
Since the inverse map may be found explicitly we can also use the method in order to
compute $W^s(x_-)$.

The complex conjugate pair of eigenvalues of the Hénon map suggests that the
map has a rotation component. Because of this, as described in Subsection 3.5.3, we
define a parallelogram parametric grid in order to generate the approximation $U_0$ that
guarantees consistency between annuli along the manifold. In order to match discrete
points of triangulation with their images under the map, the number of triangles in each
column of triangulation should be a multiple of 3; see Figure 3.27. Note that this step is
only necessary for the initial primary annulus $U_0$. To create further annuli, the algorithm
uses the parameters $T$ in their barycentric coordinates with respect to the triangulation.
Next, the set of primary annuli is generated as described above and the manifold is drawn
an annulus by the annulus.

The manifold looks like “an onion” where main part around the fixed point is a
bulb and the “fingers” create a stalk; see Figures 3.28a. If we unroll “the onion” we see
parts that grow fast and parts that grow slowly; see Figure 3.28c. During the calculation
process we notice that image of a single Bézier triangle under the map rotates around the
fixed point. Next, it hits one of the three “fingers”, at each further iterate, its image cycles
through the three fingers; see Figures 3.28b and 3.28c. This is caused by the influence of
nonlinear part of the map. In regions where the linear part of the map dominates, the
dynamics are dominated by rotation around the fixed point. In regions where nonlinear
part dominates, the “fingers” start to grow. After a few steps the primary annulus has a
complex shape with some parts fairly close to the fixed point but also with “fingers” far
away.
Figure 3.28  Two-dimensional stable and unstable manifolds of the Hénon map given by the Equation (3.18) for $a = .44$, $b = .21$, $c = .35$, $\alpha = -.25$ and $\tau = -.3$, eigenvalues s.t. $|\lambda_s| = 0.8482$, $|\lambda_u| = 1.1737$, $\text{tol} = 10^{-3}$ and the starting triangular mesh $1 \times 15$, (a) red-yellows triangles – stable, blue-green – unstable manifold, the initial portion and 15 primary annuli with $r_0 = .5$, (b) 38 primary annuli of the stable manifold for $r_0 = .01$, a single yellow Bézier triangle subject to the map; (c) schematic graph of the geodesic distance from the fixed point along one of the manifolds, (d) convergence of the proposed method tested on manifolds presented in (a) with tolerance from $\text{tol} = .1$ to $\text{tol} = .1 \times 2^{-16}$.

This phenomenon suggests it should be helpful to use some distance control to compute and to draw a manifold. We use the idea described in Subsection 3.5.6, which slightly improves the speed of the computation and its visualization; see Figure 3.29. Lomelí and Meiss in [38] proved that for the volume-preserving Hénon map all bounded orbits are contained in a cube and points outside this cube go to infinity as $t \to \infty$ or $t \to -\infty$. Because of this, we can neglect parts of the manifold that have drifted away outside of the cube.
Figure 3.29 Two-dimensional stable and unstable manifolds of the Volume-Preserving Hénon map given by the Equation (3.18); (a) $a = c = .5$, $b = 0$, $\alpha = 0$ and $\tau = -.3$, eigenvalues s.t. $|\lambda_s| = 0.9536$, $|\lambda_u| = 1.0487$, $r_0 = .3$ and 59 primary annuli are generated with $\text{tol} = 10^{-4}$, the initial triangular mesh $1 \times 18$, the distance cutoff along the manifold 2.1; (b) $a = .44$, $b = .21$, $c = .35$, $\alpha = -.25$ and $\tau = -.3$, eigenvalues s.t. $|\lambda_s| = 0.8482$, $|\lambda_u| = 1.1737$, $r_0 = 1.1$, 13 primary annuli generated with $\text{tol} = 10^{-3}$ and the triangular mesh $1 \times 36$, the distance cutoff along the manifold 5.

Note that due to the rotation, the orbit of a point on the primary annulus rotates around the manifold. At each step, its angle increases by about $2\pi/3$. It is possible that a point further away from the fixed point may be mapped back closer to the fixed point. This could happen close to the fixed point when the behavior of the manifold is driven by the linear part of the map, as well as, any further, where the dynamics of the manifold depends on the nonlinear part of the map. Our numerical test shows that is the case in this example. There exist points whose images under the map are closer to the fixed point than they themselves are, measured by geodesic distance; see point $p_1$ in Figure 3.28c. Because of this, the Krauskopf-Osinga method, described in Section 3.1, fails to compute this manifold. The situation looks similar like that in the schematic Figure 3.1b. This manifold can not be grown uniformly in radial direction, controlling geodesic distance of growth.
3.6.1.1 Convergence Test

In order to show the convergence of the proposed method we first perform a calculation using a very small value of the tolerance condition to generate the manifold which we call the “exact” manifold; \( \text{tol} \approx 10^{-6} \). Next, we generate composite surface representations of the manifold for several decreasing values of the refinement condition and check how the difference between the approximation and the “exact” manifold, measured using the maximum \((L^\infty)\) norm, decays. Note that this is the same test as we performed in Section 2.4 and in Subsection 2.6.3 for the 2D problem; see Figure 3.28d. It is clear that the proposed method converges well. The test confirms our expectation for the proposed method.

3.6.2 Example 2: Arneodo-Coullet-Tresser Map

This family of maps was introduced by Arneodo, Coullet and Tresser, motivated by the study of strange attractors in a family of differential equations on \( \mathbb{R}^3 \) with homoclinic points of Shilnikov type. Their numerical computations show some interesting phenomena in the dynamical behavior of these maps, such as a period doubling cascade and a strange attractor. More detailed bifurcation analysis is described by Du et al. in [10]. The ACT map is given by

\[
\begin{pmatrix}
  f_1(x, y, z) \\
  f_2(x, y, z) \\
  f_2(x, y, z)
\end{pmatrix} =
\begin{pmatrix}
  ax - \omega b(y - z) \\
  \frac{b}{\omega}x + a(y - z) \\
  cx - dx^2 + ez
\end{pmatrix},
\]

with fixed points at the origin, \( x_0 \), and at \( x^* = (x_1, \frac{a^2 + b^2 - a}{\omega b}x_1, \frac{(a-1)^2 + b^2}{\omega b}x_1) \) for \( x_1 = \frac{bc - (1-\epsilon)(a-1)^2 - b^2}{\omega bd} \). The local stability of the fixed points depends on the choice of parameters. Du et al. in [10], similarly to Lomelí and Meiss for the volume-preserving Hénon map, showed that the compact, nonwandering set as well as the set of bounded orbits, of the ACT map lies in a finite box.
Figure 3.30  (a) Two-dimensional unstable manifold of the ACT map given by the Equation (3.19) with $a = .2$, $b = .5$, $c = .5$, $d = 1$, $e = 1$, $k = 2$ and $\omega = 4$ associated with nontrivial fixed point, eigenvalue s.t. $|\lambda_u| = 1.2042$ and 30 primary annuli generated for $r_0 = .02$, $\text{tol} = 10^{-5}$ and the initial mesh $1 \times 36$, (b) sections of the manifold with planes containing stable direction, alternating colors for consecutive annuli, zoomed boxes does not preserve aspect ratio.

For certain parameter values, see caption of the Figure 3.30, the ACT map has a pair of hyperbolic fixed points and, associated with each, pair of complex eigenvalues; i.e., the map has a rotation component. As in the previous example, we use the parallelogram parametric grid in order to generate the approximation $U_0$. The unstable manifold of the ACT map associated with nontrivial fixed point $x^*$ is shown in Figure 3.30a, we notice that it does not grow far away from fixed point. It does not have growing “fingers”. Subsequent primary annuli stay fairly close while rotating around it. The unstable manifold seems to be bounded. The two-dimensional stable manifold of the other, trivial fixed point $x_0$ does not exhibit any more interesting behavior. Its dynamics is similar to that of the Hénon map.

Due to the rotation, stretching and folding the unstable manifold presents similar dynamics as the Rössler attractor famous for its chaotic behavior. The manifold displays the same type of banding as the attractor; it has a half-twist in it, which makes it look somewhat like a Möbius strip. Note also that since $W^u$ is bounded, but expanding due
the eigenvalue \( \lambda_u \) it folds back on itself. Points have large distance from \( x^* \) along the manifold but short Euclidean distance from \( x^* \). The manifolds shown in Figures 2.16 and 2.17, are seem to be composed of relatively flat sheets that fold back on themselves with very high curvature. These features are best visualized by examining the intersection of \( W^u \) with a plane as shown in Figure 3.30b.

Mireles-James and Lomeí in [45] show that for the Hénon map with specific parameter values (those of Figure 3.29) the parameterization method can be accurate up to 15 digits with radius above 1. The expansion can accurately represent a significant portion of the manifold without applying any other method. Similarly, for the ACT map with parameters value as we use we are able to derive such accuracy but only for a small radius (.02). The expansion can generate a small piece of the manifold. However, the interesting features (i.e., intersection of manifolds or folding) appear much further away. They can not be observed or computed by using only the parameterization method. In order to compute larger portion of the manifold we use the proposed method.

In this case of the unstable manifold of the ACT map, we do not have to use any distance control. Although, the growth is not necessarily uniform, the manifold is bounded and grows in a regular manner.

### 3.6.2.1 Cross-Section

In Figure 3.30a we see that the unstable manifold of the ACT map folds back and forth while rotating around the fixed point. Note that in certain places, the folds creates four layers of the manifold. Increasing the number of iterations in the computation, we can generate more such features. The folds of the manifold suggest that we should look closer at its intersection with a plane containing the stable direction of the fixed point; following the method described in Subsection 3.1.2, the set of such a planes is called foliation; a single plane is called a leaf of the foliation. The two sample section are presented in Figure 3.30b, along with closeup views. Sections of consecutive annuli are drawn using
alternating red and blue. The section curves fold back and forth making sharp turns. Our computations show that for both cross-sections curvature can takes value up to order $10^9$. In the boxes we magnify spots where the folds creates multiple layers of the manifold (five in the first closeup). This confirms the complexity of the manifold of the ACT map. Because of these many sharp turns the Krauskopf-Osinga method [35] for computing the unstable manifold fails. The method for computing the next discrete circle close to these sharp turn would have to use very short distance or it will miss the portion of the manifold by finding a point on a further layer. Such folding is likely to violate the transversality condition. This is another problem with the Krauskopf-Osinga method [35], beyond those presented in Subsection 3.1.2. It came to our attention while conducting these numerical tests, so we present it here.

3.7 Discussion

The method proposed here for computing unstable manifolds of the three-dimensional maps incorporates ideas from computer aided geometric design. The basic approach of our computation is forward mapping of a local portion of an invariant manifold. The strategy of this study has been to improve this technique.

We have subjected the method to more stringent tests than had been previously presented. The results show that the method achieves not only significant improvements in the accuracy of the calculation but can also handle the problems where the other methods have failed. High accuracy and near-$C^1$ continuity of the approximation, pleasant visual appearance, and reduction of the number of calls of the map $f$ allow us to see the method as the successor of these previously proposed. The method performs exceptionally well in the case of a bounded invariant manifold, even with highly varying complexity, smoothly resolving places of high curvature or exponential stretching. In the case of an unbounded, nonuniformly growing invariant manifold it also does well, however, during the process of computation it performs some unnecessary calculations:
the pieces of the manifold that are far away from the fixed point are refined to no real purpose. Distance control along the manifold reduces the number of computations, partially solving the problem, but not eliminating it completely and it significantly helps in the visualization process. The method requires at each step a parameter space with rectangular topology and this causes the problem. Finding an adaptive approximation method working over less topologically regular domain, for instance, a domain which is a composite of rectangles, might significantly improve the unstable manifold computation.

Similarly as for the one-dimensional methods, there are other costs involved with the implementation of this higher-order method, in comparison, with any method based on bivariate linear interpolation. In particular, one piece of a plane triangular interpolant can be plotted using only its values at the three corner points, with high-level graphing software filling in the points at the planar triangle. To render a quartic Bézier patch, one first samples the patch at a finite number of uniformly triangulated barycentric coordinates, and then plots a planar triangular interpolant through those points. Each of these points is evaluated by the de Casteljau algorithm for Bézier patches. The triangulation resolution is determined by the scale of the most-finely resolved Bézier triangles, so that the number of facets used to resolve a given Bézier triangle is inversely proportional to the square of the resolution level. This procedure constructs a continuous surface without cracks or pinholes; the plotted surface is uniformly triangulated in the parametric space.
CHAPTER 4

SUMMARY AND FUTURE WORK

4.1 Recap

The present study focuses on dynamical systems which model physical phenomena such as convective transport and mixing in fluid mechanics or the transport of charged particles in plasma fusion. Many applications give rise to mathematical models in the form of iterated maps or systems of ordinary differential equations. The goal is to understand the global dynamics of the system. In many cases this is not feasible analytically, so we have to use numerics. Unfortunately, some of these systems have such complex dynamics that we can only learn a small amount from direct numerical simulation. Instead, we try to understand the structure of families of solutions using combination of analysis and numerics.

To this end, one needs to find special invariant sets, such as equilibria, periodic orbits, and invariant tori. Furthermore, if these invariant sets are of saddle type, then they possess global stable and unstable manifolds. Knowing these manifolds is crucial as they organize the dynamics on a global scale. For example, stable manifolds may form the boundaries of basins of attraction, and it is well known that intersections of stable and unstable manifolds lead to complicated dynamics and chaos.

Considering a discrete-time iterated map (a discrete dynamical system), the stable/unstable manifold of a hyperbolic orbit is the set of points whose images approach the orbit under repeated forward/backward iteration of the map. The manifold is tangent to the stable/unstable eigenspace of the linearized system at origin and its global extension can be derived by applying the inverse/forward mapping to a local portion. Its dimension is equal to the dimension of the stable/unstable subspace of the linearized map.
Generally, global stable and unstable manifolds cannot be found analytically; they are not locally defined, meaning that it is not possible to find them as the zero-set of some function of the phase space variables. Hence, points on global invariant manifolds cannot be evaluated using any analytical formula. Instead, these manifolds must be “grown” from local knowledge, for example from the linearized dynamics in a neighborhood of the fixed point.

For a smooth system one uses the linearized map to gain information about the behavior near a hyperbolic fixed point. When the eigenvalues of the linearized map do not lie on the unit circle, the exponential growth of iterates near the fixed point persists for the nonlinear map. If the dynamics confines trajectories to a compact set, the expansion due to hyperbolicity and the recurrence due to compactness provides for complex and interesting global dynamics. The research in this area has been driven in part by many practical applications to experimental phenomena. Hyperbolic dynamics is central to the study of long term and stable behavior of evolving systems.

The computation of global invariant manifolds has seen renewed interest in recent years. An important survey of the most recent methods appeared in [37]. In order to improve the accuracy and efficiency of invariant manifold computations, we introduced numerical techniques from Computer Aided Geometrical Design (CAGD) such as Bézier curves and triangles, Catmull-Rom splines, spline quasi-interpolation schemes on triangulation and their adaptive versions.

In Chapter 2, we described an efficient and accurate numerical method for computing smooth approximations to invariant manifolds of planar maps, based on geometric modeling ideas from CAGD. We model the unstable manifold of a hyperbolic fixed point by a piecewise Bézier interpolant (a Catmull-Rom spline) and used properties of such curves to define a rule for adaptively adding points to ensure that the approximation resolves the manifold to within a specified tolerance. Numerical tests on a variety of
example mappings demonstrate that the new method produces a manifold of a given accuracy with far fewer calls to the map compared with previous methods.

In Chapter 3, we constructed a method for computing two-dimensional invariant manifolds of three-dimensional maps. We focused our study on maps in $\mathbb{R}^3$ especially on volume-preserving maps analyzed in [38]. Such maps are useful in understanding the motion of passive tracers in fluids and magnetic field line configurations. They are also of interest since many phenomena in the two-dimensional case are not yet completely understood in higher dimensions. Such phenomena include transport, the breakup of heteroclinic connections, and the existence of invariant tori. These maps are also important as integrators for incompressible flows. The methods work equally well for conservative and dissipative systems.

In order to decrease the number of points needed to compute a given surface, again we introduced higher-order approximation techniques from CAGD. We used a spline quasi-interpolation. This approximation method is based on locally supported B-spline basis functions. Next, we implement an adaptive quasi-interpolation based on Bézier triangles, a natural generalization of the Bézier curves, one of the fundamental objects in surface design. The method is based on tolerance conditions derived from properties of Bézier triangles. Stringent numerical tests on a few example mappings demonstrate that the new method produces a manifold of a higher accuracy with pleasant visual appearance.

We would like to make the point that we have not seen many applications of CAGD methods to dynamics problems in this manner before, and that we believe there is wide potential for their adoption. One exception is work by Henderson on numerical methods for invariant manifolds of continuous-time problems, although he uses a different set of tools than those described here [26, 27, 28]. CAGD methods, especially NURBS, have also been widely used in the finite elements literature. A large and very accessible literature exists on this subject. We hope that these techniques can prove useful to
other researchers in dynamical systems. Two excellent books are by Farin [14] and by Goldman [18].

4.2 Improvement of the Two-dimensional Method

The proposed method for computing two-dimensional invariant manifolds requires a parameter space with rectangular topology and this can cause some problems. Finding an adaptive approximation method working over less topologically regular domain might significantly improve the computation.

Possible improvement can be achieved by introducing T-splines. The main difference between a T-mesh, i.e., a T-spline control mesh, and a non-uniform B-spline (NURBS), topologically rectangular, control mesh is that T-splines allow a row of control points to terminate. The final control point in a partial row is called a T-junction. T-junctions allow T-splines to be locally refineable, i.e., control points can be inserted into the control grid without propagating an entire row or column of control points. Sederberg et al. in [54, 56] present algorithms for local refinement of T-splines. These algorithms should allow us to create a method that begins with the coarse approximation then performs a series of refinements by adding control points in regions where more detail needed, and then adjusting those control points. The control point insertion will be in the region being refined. Before constructing such an adaptive method for our problem, we would first have to construct a non-adaptive quasi-interpolation method based on non-uniform B-splines, a step already carried out by Sorokina [59] for the family of splines we chose to use.

4.3 Application of the Adaptive Methods

Our methods for computing unstable manifolds incorporate ideas from computer aided geometric design. To show some additional advantages over other methods, we propose to implement them to analyze transport in a chaotic system. In order to perform this
analysis, one needs to construct regions bounded (2D) or volumes trapped (3D) by segments of the stable and unstable manifolds and thus must determine intersections of these manifolds. Many efficient methods exist in the CAGD field, which may be used for these purposes [48, 55].

The tool may be applied to the study of vortex dynamics arising in the family of quadratic, volume-preserving, diffeomorphisms with quadratic inverse. By exhibiting transverse intersections between the stable and unstable manifolds of fixed points, we should be able to provide geometric evidence for the existence of chaotic motions. In addition, we can examine the qualitative features of homoclinic orbits and a homoclinic tangle in three dimensions. The performance of our method can be directly compared with the quasi-numerical results described in [44].

4.4 Invariant Manifolds of Hyperbolic Tori

Another natural continuation of our research would be the implementation of the method for computing two-dimensional invariant manifolds of hyperbolic invariant circles or tori. Dynamics on an invariant torus is typically quasi-periodic. Invariant manifolds of tori, called whiskers by Arnold, are characterized by being tangent to a chosen invariant space of the linearization around the torus. Invariant tori commonly occur in dynamical systems. The whiskers are a common feature of volume-preserving mappings [39], displaying type of behavior similar to the manifolds of the hyperbolic point.

Since invariant manifolds must be derived from local dynamics, we need to gain information about the behavior of the manifold in a neighborhood of hyperbolic tori. In [61], Wysham and Meiss provide first order approximations to invariant manifolds of the invariant torus; these methods can be used as an initial step for our adaptive method. Having implemented an adaptive method to compute invariant manifolds of the invariant tori, we can directly compare our results with several systems studied in [39, 61]. Additional
references could be found in [23, 24, 25], where more general computations of invariant tori and their whiskers, based on the parameterization method by de la Llave, are described.
APPENDIX

SOFTWARE

The author has written in collaboration with his dissertation advisor a small set of MATLAB programs that implement the methods described in this dissertation. These are available from the author’s website

REFERENCES


