Diss. ETH No. 14531

Bernstein–Bézier Representations for Facial Surgery Simulation

A dissertation submitted to the
Swiss Federal Institute of Technology, ETH, Zurich

for the degree of
Doctor of Technical Sciences

presented by
SAMUEL HANS MARTIN ROTH
Dipl. Informatik–Ing. ETH
Swiss Federal Institute of Technology, ETH, Zurich
born April 1, 1969
citizen of Niederbipp, Berne, Switzerland

accepted on the recommendation of
Prof. M. H. Gross, examiner
Prof. W. Gander, co-examiner

2002

“In theory, there is no difference between theory and practice;
In practice, there is.”

Chuck Reid

“All models are wrong but some are useful.”

George Box

ABSTRACT

The finite element method and its application to the simulation of static linear elasticity has a long research history. The same applies for Bernstein–Bézier representations of curves and surfaces in computer aided geometric design. However, the combination of both to build tetrahedral Bernstein–Bézier finite elements presents an inspiring and fruitful challenge. The theory and implementation of these elements and their application in the context of facial surgery simulation is the main focus of this thesis.

Both for patients and surgeons, thorough planning is an absolute prerequisite for successful surgical procedures. Therefore, attention is turning to computer–assisted planning systems. The three–dimensional physically–based simulation of facial surgery is envisioned to replace or complement on current surgical planning techniques.

After a short motivation and overview of existing deformable models in computer graphics and surgery simulation, we give an introduction to the finite element method. Its application in the context of static elasticity is one of the main building blocks of the envisioned tissue model for surgery simulation. Besides classical linear elasticity, incompressibility and nonlinear stress–strain relations are taken into account.

The representation of surfaces and volumes by means of Bernstein–Bézier patches is revisited. Emphasis is put on barycentric representations and on the construction of smooth patch transitions. Further, multivariate hermite interpolants are investigated and evaluated with respect to their suitability for finite element modeling. The construction of a globally C^1 trivariate tetrahedral interpolant based on a multi–dimensional generalization of the well–known Clough–Tocher split is presented.

As a next step, Bernstein–Bézier techniques are put into the context of finite element analysis of static elastomechanics. A C^1–continuous tetrahedral finite element is derived from the trivariate Clough–Tocher construction. The complex assembly procedure resulting from the construction is given special emphasis. In a thorough test series, C^0–continuous tetrahedral elements are compared with the C^1 Clough–Tocher element. Degree elevation and mesh refinement are opposed to the effect of imposing higher level continuity constraints. The C^1 construction scheme is shown to invalidate neither the approximation properties nor the locality of the Bernstein finite element basis. At the same time it preserves the integral nature of the basis and therefore allows for analytical integration.

Aiming at the evaluation of the physical tissue model and its finite element solution, we describe the implementation of a highly automatic surgery simulation prototype designed to post–simulate actual surgery. We propose methods and solutions needed in the model build–up and we describe an automatic computation of surgery displacement fields corresponding to real surgical procedures. The presentation of results achieved on the example of a test patient concludes the thesis.

ZUSAMMENFASSUNG

Sowohl die Analyse statischer Elastizität mit der Methode der finiten Elemente als auch die Bernstein–Bézier Repräsentation von Kurven und Flächen sind Forschungsgebiete mit einer langen Geschichte. Die Kombination von Erkenntnissen aus beiden Gebieten zu einem finiten Tetraederelement bietet allerdings eine fruchtbare Herausforderung. Die Theorie und Implementation solcher Elemente sowie deren Anwendung im Gebiet der Simulation kieferchirurgischer Eingriffe ist der Schwerpunkt dieser Dissertation.

Eine gründliche Chirurgieplanung ist sowohl für den Patienten als auch für den Chirurgen unabdingbar. Daher gewinnen computerunterstützte Planungsmethoden zunehmend an Bedeutung. Es ist vorstellbar, dass die dreidimensionale physikalisch basierte Planung kieferchirurgischer Eingriffe die traditionelle Planung mittelfristig verdrängt oder zumindest ergänzt.

Nach einer Motivation und einem Überblick über bestehende deformierbare Modelle in der Computergraphik und im Gebiet der Chirurgiesimulation folgt eine Einführung in die Methode der finiten Elemente. Ihre Anwendung im Kontext der statischen Elastizität ist ein Hauptbestandteil des angestrebten Gewebemodells. Neben klassischer linearer Elastizität werden Inkompressibilität und nichtlineares Materialverhalten behandelt.

Anschliessend wird die Darstellung und Manipulation von Bernstein–Bézier Flächen eingeführt. Der Schwerpunkt liegt auf baryzentrischen Repräsentationen und der Konstruktion glatter Übergänge zwischen Flächenstücken. Multivariate Hermite Interpolanten werden auf ihre Eignung für den Einsatz in der Methode der finiten Elemente untersucht. Ausgehend von einer mehrdimensionalen Verallgemeinerung des bekannten Clough–Tocher Splits wird die Konstruktion eines global C^1–stetigen Interpolanten auf Tetraedergittern beschrieben.

In einem nächsten Schritt wird die Bernstein–Bézier Technik im Hinblick auf die Analyse statischer Elastizität in den Kontext der Methode der finiten Elemente gesetzt. Ausgehend von der trivariaten Clough–Tocher Konstruktion wird ein C^1–stetiges finites Element vorgestellt. Der komplexen Assemblierung solcher Elemente wird spezielles Augenmerk gewidmet. In einer Testserie werden C^0–stetige Elemente mit dem C^1 Element verglichen. Graderhöhung und Gitterverfeinerung werden dem Ansatz mit erhöhter Stetigkeit gegenübergestellt. Dabei erweist es sich, dass die C^1–Konstruktion weder die Approximationseigenschaften verschlechtert noch die Lokalität der Bernsteinbasis invalidiert. Gleichzeitig bleibt die Repräsentation nichtrational und damit geeignet für die analytische Integration.

Im Hinblick auf die Evaluierung sowohl des physikalischen Modells als auch der finite Elemente Lösung wird die Prototyp–Implementation eines weitgehend automatischen Chirurgiesimulators zur Nachsimulation echter Operationen beschrieben. Methoden und Ansätze für den Aufbau des Modells werden vorgeschlagen, und die automatische Bestimmung von Verschiebungsfeldern zur Repräsentation echter Eingriffe wird beschrieben. Die Präsentation von Resultaten am Beispiel eines Testpatienten schliesst die Arbeit ab.

ACKNOWLEDGEMENTS

Many people helped me in completing this thesis. First of all, my thanks go to Prof. Markus Gross for establishing computer graphics at ETH and letting me be a part of it as one of his PhD students. His enthusiasm for the facial surgery simulation project has been a driving force for my PhD project. Further, he never lacked creative ideas and many helpful discussions revealed his enormously broad overview of graphics and simulation.

Similarly, I would like to express my gratitude to my co–advisor, Prof. Walter Gander. His excellent introductory lectures to numerical mathematics opened me the door to the world of scientific computing. It is a great pleasure to have him now as my co–advisor, and his fruitful comments made a clear difference in this work.

In addition, many thanks go to all my fellow PhD students at the Computer Graphics Laboratory for helping me along the way. Special thanks go to Lars Lippert, Oliver Staadt, Rolf Koch, Thomas Sprenger, Reto Lütolf, Daniel Bielser, Matthias Zwicker, Andreas Hubeli and Stephan Würmlin. Further, I want to thank my fellow students who also entered the PhD program, especially Erwin Achermann, Renato Pajarola, and Erich Oswald. Last but not least, I would like to thank Oscar Chinellato for many inspiring discussions.

Further, I would like to express my gratitude to all the diploma and semester students who contributed to this work: Christoph Zehnder, Silvio Turello, Fabian Honegger, Matthias Zwicker, Marcel Lattmann, Patrick Diezi, Martin Spengler, and Gregory Bleiker. Without their enthusiasm, I most probably would not have succeeded in finishing this thesis. A very special thank–you goes to Matthias Zwicker for his persistence and excellence in the algorithmic realization and implementation of the tetrahedral Clough–Tocher finite element.

My warmest thanks go to my parents, as well as to my sister and brother, for providing me with the greatest possible support during my years at ETH. To my parents, providing me with the best possible education has always been a matter of course, a fact that I deeply appreciate.

Many thanks go to Prof. Dr. Hermann Sailer, Dr. Friedrich Carls, and Dr. Axel Zimmermann at the University Hospital of Zurich for their valuable medical consulting and collaboration in the project. In addition, we thank our example patient Dario Dobranic for his patience and willingness to participate in the study.

Last but not least, my most special thanks go to Monique Sailer for being the most important person in my life. Her support and motivation helped me along the way and she always had an open ear in times of stress and trouble. Thank you, Monique!

This work was partially supported by the Swiss National Science Foundation (SNSF) under grant no. 2153–049247.96/1.

CONTENTS

1 Introduction . . . 1

1.1 Motivation and Goals . . . 1
  1.1.1 Medical Planning of Facial Surgical Procedures . . . 2
  1.1.2 Envisioned Planning System . . . 3
  1.1.3 Bernstein–Bézier Deformable Models . . . 4

1.2 Contribution . . . 5

1.3 Outline and Organization . . . 6

2 Deformable Models in Computer Graphics . . . 7

2.1 Geometric Models . . . 8

2.2 Physically–Based Models . . . 8
  2.2.1 Mass-Spring Models . . . 9
  2.2.2 Finite Differences . . . 10
  2.2.3 Finite Element Models . . . 11
  2.2.4 Boundary Element Method . . . 12

2.3 Facial Models and Surgery Simulation Systems . . . 13
  2.3.1 Facial Models . . . 13
  2.3.2 Surgery Simulation Systems . . . 14

2.4 Thesis Focus . . . 15

3 Finite Element Modeling of Elastic Materials . . . 17

3.1 Introductory Concepts . . . 17
  3.1.1 Notations and Definitions . . . 18
  3.1.2 Questions of Differentiability . . . 18

3.2 Continuous–System Mechanical Models . . . 23
  3.2.1 Differential Formulation . . . 23
  3.2.2 Variational Formulation . . . 24
  3.2.3 Equivalence of Strong and Weak Form . . . 25

3.3 Galerkin Approximation and Matrix Formulation . . . 27
  3.3.1 Galerkin Projection . . . 27
  3.3.2 Matrix Formulation . . . 30
  3.3.3 Global versus Local Point of View . . . 33

3.4 Static Elastomechanics for Soft Tissue Modeling . . . 34
  3.4.1 Linear Elasticity . . . 35
  3.4.2 Incompressibility . . . 38
  3.4.3 Nonlinear Extensions . . . 40

3.5 Finite Element Discretization 42
  3.5.1 Subdivision of Domain into Finite Elements 42
  3.5.2 Galerkin Projection 43
  3.5.3 Assembly 44
  3.5.4 Discretization of Mixed Formulation 45
  3.5.5 Discretization of Material–only Nonlinear Simulation 47

4 Bernstein Polynomials and Bézier Patches 49

4.1 Introductory Concepts 50
  4.1.1 Local Coordinate Systems 50
  4.1.2 Bernstein Polynomials 52
  4.1.3 The Notion of Continuity 53

4.2 Bézier Curves 54
  4.2.1 Definition and Properties 54
  4.2.2 De Casteljau Algorithm 55
  4.2.3 Derivatives of Bézier Curves 56
  4.2.4 Subdivision of Bézier Curves 57
  4.2.5 Additional Bézier Curve Topics 58
  4.2.6 Piecewise Bézier Curves 58
  4.2.7 Interpolation using Bézier Curves 59

4.3 Tensor Product Extension to N Dimensions 61
  4.3.1 Patch Construction 62
  4.3.2 Analogy to Univariate Case 62

4.4 Triangular Bernstein–Bézier Patches 63
  4.4.1 Barycentric Formulation 63
  4.4.2 Triangular Bézier Patch Topics 65
  4.4.3 C^m Continuity Conditions across Patch Boundaries 70
  4.4.4 Triangular Hermite Interpolation 72
  4.4.5 Split–Triangle Interpolants 75

4.5 Multivariate Barycentric Bézier Patches 77
  4.5.1 Extension to N Dimensions 77
  4.5.2 Generalized Clough–Tocher Split in N dimensions 80
  4.5.3 A Three–dimensional Clough–Tocher Split Interpolant 81

5 A Tetrahedral C^1 Bernstein–Bézier Finite Element 89

5.1 Introduction 90
  5.1.1 Bernstein–Polynomials as Shape Functions 90
  5.1.2 A One–Dimensional Example 91

5.2 Tetrahedral Bernstein–Bézier Element 97
  5.2.1 Interpolation of Geometry 97
  5.2.2 Definition of Bernstein–Bézier Finite Element 98
  5.2.3 Properties 99

5.3 Local Stiffness Matrices and Force Vectors 99

5.4 C^0–continuous Finite Element 101
- 5.4.1 *The Structure and Representation of the Global Stiffness Matrix* *101*
- 5.4.2 *Assembly and Disassembly* *102*

5.5 C^1–continuous Finite Element 103
- 5.5.1 *Terminology and Definitions* *104*
- 5.5.2 *The Structure of the Global Stiffness Matrix* *105*
- 5.5.3 *Clough–Tocher Splitting Tetrahedral Tesselations* *105*
- 5.5.4 *Assembly of a Macro–Element* *106*
- 5.5.5 *Disassembly* *111*

5.6 The Global System of Equations 111
- 5.6.1 *Properties* *111*
- 5.6.2 *Solving Strategies* *112*

5.7 Results on a Synthetic Block of Tissue 114
- 5.7.1 *C^0–continuous Simulation* *114*
- 5.7.2 *C^1–continuous Simulation* *122*

6 Ray Tracing Triangular Bézier Patches 127

6.1 Related Work 128
- 6.1.1 *Nested Bounding Volumes* *128*
- 6.1.2 *Parameter Interval Iteration* *129*
- 6.1.3 *Ray Tracing Triangular Patches* *129*

6.2 Triangular Bézier Clipping 129
- 6.2.1 *Ray–Patch Intersection Problem* *130*
- 6.2.2 *Reduction to Two Dimensions* *131*
- 6.2.3 *Finding Intersections* *132*
- 6.2.4 *Convex Hull Determination* *136*

6.3 Reporting Wrong Intersections 137

6.4 Results 139

7 A Prototype for Surgery Simulation 143

7.1 Overview of the Prototype 143
- 7.1.1 *Prerequisites* *144*
- 7.1.2 *Model Build–Up* *145*
- 7.1.3 *Jaw Cutting and Displacement Fields* *145*

7.2 Surface Registration 147
- 7.2.1 *Absolute Orientation using Unit Quaternions* *148*
- 7.2.2 *Iterative Closest Points Algorithm* *155*

7.3 Discretization of Tissue into Finite Elements 158
- 7.3.1 *Skull Extraction* *158*
- 7.3.2 *Tissue Meshing* *159*
- 7.3.3 *Imposition of Boundary Conditions* *161*

7.4 Computation of Displacement Fields 161
    7.4.1 Registration of Volume Data Sets 161
    7.4.2 Jaw Cutting and Movement 167

7.5 Results and Evaluation 167
    7.5.1 Evaluation Procedure 167
    7.5.2 The Example Patient 168
    7.5.3 Simulation Results 170

8 Conclusions and Future Work 177

8.1 Summary and Conclusions 177

8.2 Directions for Future Work 180

A Nomenclature 183

B References 191

C Color Plates 203

D Curriculum Vitae 209

FIGURES

FIG. 1.1: Example of a facial disharmony and its correction by maxillofacial surgery. . . 2
FIG. 1.2: Conventional planning 3
FIG. 2.1: Mass–spring model as spring interconnected masses on a regular grid 9
FIG. 2.2: Layered synthetic tissue model 13
FIG. 3.1: Generalized functions. 20
FIG. 3.2: Idealized bar of unit length 23
FIG. 3.3: Shape functions for one–dimensional piecewise linear finite element space 31
FIG. 3.4: A typical member of the finite element space 32
FIG. 3.5: General three–dimensional elastic body 34
FIG. 3.6: Linear approximation of volumetric changes 40
FIG. 3.7: Nonlinear stress–strain relationship 41
FIG. 4.1: Patchwise definition of a tensor product surface 50
FIG. 4.2: Patchwise definition of a triangular surface 51
FIG. 4.3: Barycentric coordinates. 52
FIG. 4.4: Quadratic, cubic and quartic Bernstein polynomials 53
FIG. 4.5: A cubic Bézier curve and its control polygon and convex hull 54
FIG. 4.6: De Casteljau evaluation of a cubic Bézier 55
FIG. 4.7: Subdivision of a cubic Bézier curve. 57
FIG. 4.8: C^1 condition on the example of cubic Bézier curves 60
FIG. 4.9: Cubic hermite interpolation using Bézier curves 61
FIG. 4.10: Cubic Hermite polynomials 62
FIG. 4.11: Tensor product Bézier patch of degree and its control net. 62
FIG. 4.12: Tensor product de Casteljau algorithm 63
FIG. 4.13: Cubic barycentric Bernstein polynomials. 64
FIG. 4.14: Schematic view of linear, quadratic and cubic triangular Bézier control nets 65
FIG. 4.15: Cubic triangular Bézier patch and its defining control net 65
FIG. 4.16: De Casteljau algorithm for triangular patches 66
FIG. 4.17: Tangent planes at the corners of a Bézier patch. 67
FIG. 4.18: Triangle in –parameter space 68
FIG. 4.19: Subdivision of barycentric patches at an internal point 69
FIG. 4.20: Subdivision of barycentric patches with respect to three internal point 70
FIG. 4.21: Construction of a C^∞–continuous cubic patch extension. 71
FIG. 4.22: Construction of a C^1–continuous cubic patch transition 72
FIG. 4.23: Numbering of control points for the quintic triangular interpolant 74
FIG. 4.24: Triangular Clough–Tocher interpolant. 76
FIG. 4.25: Cubic trivariate Bézier patch and de Casteljau algorithm 78
FIG. 4.26: Construction of a C^1–continuous patch transition between two tetrahedra . 79
FIG. 4.27: Hermite data stencil 82
FIG. 4.28: Generalized Clough–Tocher split on a tetrahedron 82
FIG. 4.29: Definition of a global quadratic on a Clough–Tocher split triangle 83
FIG. 4.30: Hyperplane condition at corners of Clough–Tocher split tetrahedron 84

FIG. 4.31: Hyperplane condition around edge of Clough–Tocher split tetrahedron. . . .84
FIG. 4.32: Micro–tetrahedra within macro–tetrahedron after peeling away outer shell .85
FIG. 4.33: Collinear splitting points in adjacent tetrahedra 86
FIG. 4.34: Quadratic bivariate control nets for evaluation of directional derivative 86
FIG. 5.1: Finite element solution using cubic Hermite shape functions 94
FIG. 5.2: Finite element solution using cubic Bernstein shape functions 95
FIG. 5.3: Structure of local stiffness matrices. 100
FIG. 5.4: Correlation between local and global stiffness matrices 102
FIG. 5.5: Structure of global stiffness matrix for cubic Bernstein–Bézier example. . . .103
FIG. 5.6: FEM data stencil and hyperplanes on macro–element 104
FIG. 5.7: Establishing linear dependencies by means of a virtual triangle. 107
FIG. 5.8: Four shells of control points in macro–element 107
FIG. 5.9: C^1 hyperplane condition degenerating to a plane condition 108
FIG. 5.10: Elimination matrix and intermediate result. 109
FIG. 5.11: Structure of a global stiffness matrix for an example problem 110
FIG. 5.12: Pull and push configuration for test simulations 114
FIG. 5.13: Four discretizations resulting from mesh refinement 115
FIG. 5.14: Test series with linear, quadratic and cubic interpolation 115
FIG. 5.15: Discretization used for the push configuration 117
FIG. 5.16: Comparison of pure displacement and mixed formulation (C^0 linear) 118
FIG. 5.17: Comparison of pure displacement and mixed formulation (C^0 quadratic) .119
FIG. 5.18: Comparison of pure displacement and mixed formulation(C^0 cubic). 120
FIG. 5.19: C^1 test series compared against the cubic C^0 element 122
FIG. 5.20: Comparison of pure displacement and mixed formulation (C^1) 125
FIG. 5.21: Pull configuration with prescribed displacement centered at top 126
FIG. 6.1: Representation of a ray by two orthogonal planes 130
FIG. 6.2: Reduction of the ray–patch intersection problem to two dimensions 131
FIG. 6.3: Bézier clipping in the parametric domain 132
FIG. 6.4: Line L_r and corresponding distances dr_{ij} of control points. 133
FIG. 6.5: Functional distance patch 134
FIG. 6.6: Projection of functional L_r and L_s distance patch 134
FIG. 6.7: Subdivision of patch according to clipping minima r_{min}, s_{min} and t_{min} 135
FIG. 6.8: Computation of upper and lower convex hull 136
FIG. 6.9: Potential reporting of wrong intersections (tensor product). 137
FIG. 6.10: Potential reporting of wrong intersections (triangular) 138
FIG. 6.11: Subdivision of a patch for hierarchy build–up 140
FIG. 6.12: Control net of Utah teapot and corresponding ray traced image 140
FIG. 6.13: Close–up of control net at knob and corresponding ray traced image. 140
FIG. 6.14: Textured teapot. 141
FIG. 7.1: CT scanner, data slices, skull isosurface, facial isosurface 144
FIG. 7.2: Laser range scanner, scanning process, resulting surface and texture 144
FIG. 7.3: Flow chart of model build–up 146
FIG. 7.4: Flow chart of jaw cutting and determination of displacement fields 147
FIG. 7.5: Surface registration of two identical geometries .157
FIG. 7.6: Registration of laser range surface to the facial isosurface 157
FIG. 7.7: Skull extraction 159
FIG. 7.8: Tissue Meshing 159
FIG. 7.9: Imposition of boundary conditions 160
FIG. 7.10: Multiresolution pyramid and 2D image registration result 166

FIG. 7.11: 3D image registration result 166
FIG. 7.12: Jaw cutting and Jaw alignment 167
FIG. 7.13: Evaluation procedure 168
FIG. 7.14: Example patient 168
FIG. 7.15: Lefort I osteotomy and sagittal split 169
FIG. 7.16: Conventional surgery planning 169
FIG. 7.17: Surface changes and displacement fields due to real world surgery 170
FIG. 7.18: Low and high resolution mesh and model 170
FIG. 7.19: Profile view comparison of low resolution simulation results 171
FIG. 7.20: Frontal view comparison of low resolution simulation results 172
FIG. 7.21: Profile view comparison of high resolution simulation results 174
FIG. 7.22: Frontal view comparison of high resolution simulation results 175

TABLES

TABLE 5.1: Timings and matrix statistics for the test series of Figure 5.14. 116
TABLE 5.2: Timings and statistics for the test series of Figure 5.16. 118
TABLE 5.3: Timings and statistics for the test series of Figure 5.17. 119
TABLE 5.4: Timings and statistics for the test series of Figure 5.18. 120
TABLE 5.5: Timings and statistics for the test series of Figure 5.19. 123
TABLE 5.6: Timings and statistics for the test series of Figure 5.20. 125
TABLE 7.1: Timings and statistics for the test series in Figures 7.19 and 7.20. 173
TABLE 7.2: Timings and statistics for the test series in Figures 7.21 and 7.22. 176

CHAPTER 1

INTRODUCTION

In facial surgery, planning is of paramount importance. Both for patients and surgeons thorough planning is an absolute prerequisite for successful surgical procedures. Therefore, attention is turning to computer–assisted planning systems. Physically–based three–dimensional simulation should enable a much more accurate and reliable prediction of surgery results. Questions arise as how to model the behavior of tissue, which method to choose in order to solve the inherent partial differential equations, and finally how to implement such a surgery simulator.

The theory of elasticity is a promising starting point in the quest for a physically–based deformable model of facial tissue. Although being an approximation, it turns out to be a good compromise between accuracy and computational feasibility. The finite element method has become a very general and powerful instrument for solving simulation problems in a great variety of disciplines. Having its roots in engineering disciplines, the method has been given a solid mathematical foundation over the past decades. The development of a finite element for elasticity is therefore a core topic in the implementation of the envisioned surgery simulator.

1.1 MOTIVATION AND GOALS

Maxillofacial surgery and craniofacial surgery take care of a great variety of diseases of the whole face and skull, i.e. fractures, tumors, infections and malformations. Fractures of craniofacial bones have to be repositioned and fixed and a wide spectrum of facial and craniofacial malformations (e.g. Figure 1.1) have to be treated. Some of the patients show only minor asymmetries, e.g. of the mandible and chin and hence seek for treatment. Others show more deformed faces due to inherited syndromes or as a result of congenital diseases without inheritance. Other malformations comprise acquired diseases during childhood or adolescence (e.g. reduced growth of a jaw after trauma to the temporomandibular joint, i.e. the jaw–joint).

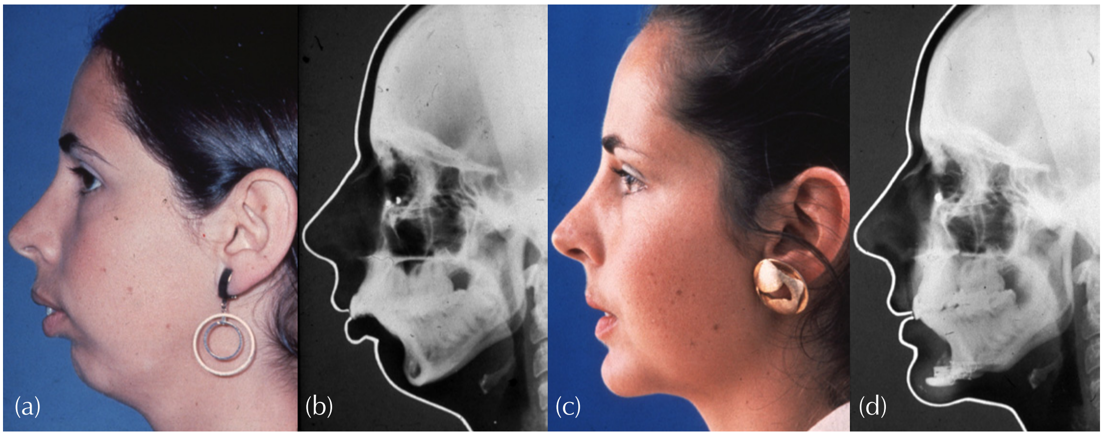


FIGURE 1.1 Example of a facial disharmony and its correction by maxillofacial surgery:
(a) Pre–surgical facial shape contour (profile)
(b) Pre–surgical lateral X–ray image
(c) Post–surgical appearance after maxillofacial surgery
(d) Post–surgical lateral X–ray image
(Data courtesy of Prof. H. F. Sailer, University Hospital of Zurich, Switzerland)

All these categories of diseases result in facial asymmetry or disfigurement. In addition to the general impairment of their health, patients with these diseases suffer a great deal from their facial deformities. Since the human face plays a key role in interpersonal relationships it is essential not only to cure the underlying disease but also to predict the post–surgical morphology and appearance of the face. It is obvious that this is a critical issue for patients with facial deformities. Moreover, cranio–maxillofacial surgery has to strive for the reconstruction of a balanced face. Even very subtle malformations of facial proportions can strongly affect the appearance of a face and determine on aesthetic aspects such as individual beauty [44].

1.1.1 Medical Planning of Facial Surgical Procedures

Therefore, surgeons often face the problem of predicting a fair facial surface before the actual surgery is carried out. Figure 1.1 illustrates a typical malformation of a female's face and its correction by surgery.

In a conventional set–up, the planning of a maxillofacial surgical procedure is done by means of lateral X–ray images thereby predicting the two–dimensional appearance of the post–surgical profile. Lateral X–ray images illustrating the actual and post–surgical profiles are presented in figures Figures 1.1b and 1.1d, respectively.

Medical artists sketch the desired post–surgical profile and discuss it intensively with both the patient and the surgeon (see Figure 1.2a and b). Together with computed tomography (CT) scans and plaster models (Figure 1.2c), these profile sketches are the main ingredients to facial surgery planning. In addition to the manual sketching of the post–surgical profile, first commercial computer–assisted planning systems are at disposal. Being based on image morphing, they allow for the computation of predictive images of the post–surgical appearance but clearly lack a physical foundation. Further, both the image morphing and the profile sketch approach are purely two–dimensional. Therefore, many aspects such as the frontal appearance of the lips and the nose are not taken into account but left to the surgeon's experience.

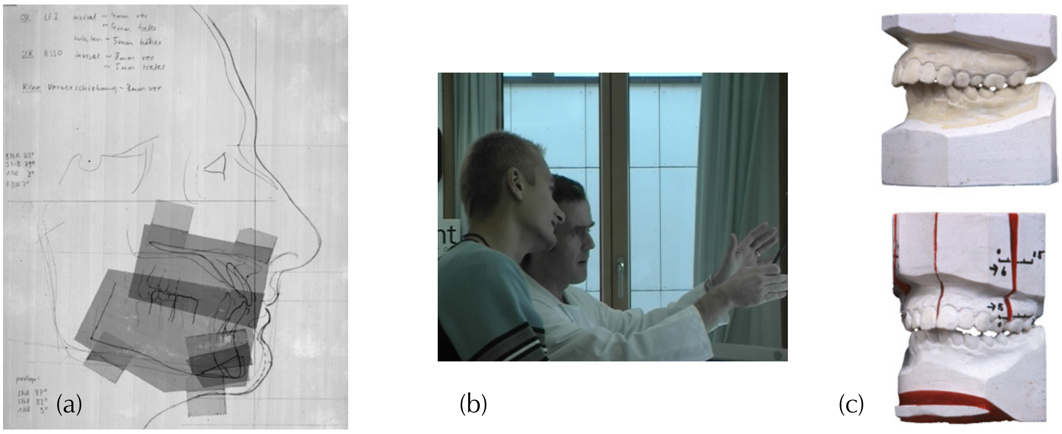


FIGURE 1.2 Conventional planning: (a) profile sketches, (b) patient consulting, (c) plaster models.

1.1.2 Envisioned Planning System

It is clear that both surgeons and their patients have a strong demand for a more accurate, three–dimensional planning system. It is in the interest of both not only to cure malformations and to restore masticatory capabilities, but also to achieve an aesthetically pleasing appearance. Prior to surgery, such a system would allow for the computation of highly realistic three–dimensional pictures of the post–surgical shape. Moreover, given a suited representation, ray tracing the resulting surface would facilitate the computation even of photo–realistic images. Any computation should be based on data available, or at least easy to obtain from the patient, e.g. computed tomography (CT) or magnetic resonance imaging (MRI) scans in combination with laser range (LR) scans of the surface. Obviously, such a physical model has to capture the most important anatomical and mechanical parameters of the face in order to predict the facial shape accurately.

Various approaches to this and similar problems can be found in literature and will be reviewed in Section 2.3. We believe that a model providing the accuracy needed in surgery planning must be based on sophisticated finite element modeling. To this aim, truely volumetric soft tissue models have to be combined with accurate geometric models of the facial surface and the individual skull.

The benefits of a three–dimensional computer–assisted planning system based on a physical model of the patient can be regarded as being threefold:

- *Planning of surgical procedures*
  The surgeon is able to plan surgery by means of the model and he or she can simulate the outcome of several variants of the procedure the best of which is then to be chosen.

- *Patient care and consulting*
  Prior to surgery, the patient can be given an idea of his or her post–surgical appearance. Consequently, he or she can get acquainted with a more or less significantly changed facial appearance, a process which sometimes turns out to be troublesome.

- *Surgery training*
  In future, physically–based models with an emphasis on real–time simulation will enable surgery training for educational purposes.

Focussing on the first two of the above issues, the thesis aims at the three–dimensional simulation of cranio–maxillofacial surgery. As opposed to the two–dimensional planning based on image morphing, the computation has to build upon a physical–based deformable tissue model reflecting the elastic properties of facial tissue. The finite element method will allow for the accurate computation of simulation results. Chapter 3 therefore is dedicated to the finite element modeling of elastic materials.

Besides the design of the physical model and the development of a suitable finite element solution, the implementation of a surgery prototype is a major issue of the thesis. With model evaluation in mind, emphasis is put on the post–simulation of actual surgery. Therefore, Chapter 7 summarizes the design of the prototype, including model build–up and the computation of displacement fields corresponding to actual surgery.

1.1.3 Bernstein–Bézier Deformable Models

In computer graphics, deformable models have been a major research issue during the last decades. Deformable models divide into two categories: purely geometric approaches and physically–based approaches. The two approaches will be discussed in Sections 2.1 and 2.2, respectively.

In computer aided geometric design (CAGD), Bernstein–Bézier representations have been investigated thoroughly over a long time. Both in the context of curves and surfaces, and even for volumes, they allow for a compact and intuitive representation of shapes in the form of Bézier patches. These include tensor–product as well as triangular representations. Similar to triangular patches in the bivariate case, tetrahedral patches clearly offer the most in geometric and topological flexibility and therefore are the primitives of choice for the trivariate representation of volumes.

The purely geometric approach which most often is inherent to computer aided geometric design has limited the application of Bernstein–Bézier representations to modeling in design and manufacturing. However, being a parametric representation, Bernstein–Bézier approaches are equally well suited as a physically–based deformable model. In two or three dimensions, Bernstein–Bézier patches basically can be regarded as a representation of shape in the basis of Bernstein polynomials weighted by their respective control points. Much as any other polynomial basis, such a representation lends itself well for application in the finite element method, taking the Bernstein polynomials as the finite element shape functions.

A second primary goal of this thesis consequently lies in the unification of the elegance of Bernstein–Bézier representations with the finite element method in the context of deformable volumes. The construction of smooth, i.e. C^1–continuous, patch transitions is given special emphasis. While global C^1 continuity is trivial to achieve for curves as well as for tensor–product surfaces and volumes, it is a challenging task for barycentric primitives such as triangles and tetrahedra. Chapter 4 gives an introduction to the relevant theory of Bernstein–Bézier representations, whereas Chapter 5 describes the implementation of a tetrahedral C^1 Bernstein–Bézier finite element. In Chapter 6, we propose a new method for the computation of ray intersections with triangular Bézier patches. This enables us to ray trace the envisioned finite elements directly and thus allows for highest quality rendering without reverting to the polygonalization of the resulting surface.

1.2 CONTRIBUTION

In correspondence with the motivation given above, the contributions of the thesis can be summarized as follows:

- *Volumetric finite element modeling for facial surgery simulation*
  In contrast to previous approaches, the simulation of facial tissue is truly volumetric and based on the theory of static elastomechanics. Besides classical linear elasticity, incompressibility and nonlinear stress–strain relations are taken into account. The finite element method is employed in order to solve the partial differential equations of static elastomechanics. For the sake of geometrical and topological flexibility, tetrahedral finite elements are used in the representation of the deformable model. We focus on Bernstein–Bézier representations and oppose the effect of degree elevation and mesh refinement to the imposition of higher level continuity constraints.
- *C^1–continuous tetrahedral Bernstein–Bézier finite element*
  As a consequence, in addition to the C^0–continuous tetrahedral elements of linear, quadratic and cubic degree, a C^1–continuous tetrahedral finite element is proposed. It is adapted from a theoretical n–dimensional interpolant known from approximation and interpolation theory. In doing so, the theory of Bernstein–Bézier representations is combined with the finite element method. Being based on an integral Bernstein–Bézier representation, the finite element basis functions are integral, too. In spite of providing C^1–continuous patch transitions, there is no need to revert to rational basis functions the like of which are used in triangular Hermite approaches. Consequently, the integrations inherent to the finite element method can be done analytically albeit at the price of a more complex assembly procedure. Although linear elasticity is known to be a C^0 problem, the effect of imposing C^1 continuity constraints is investigated.
- *Design of a highly automatic prototype for facial surgery simulation*
  Aiming at the evaluation of the physical model and its finite element solution, we describe the implementation of a highly automatic surgery simulation prototype, thus putting the above–mentioned finite elements in the context of facial surgery. For evaluation purposes we focus on the post–simulation of actual surgery. We describe methods and solutions needed in the model build–up and we propose an automatic computation of surgery displacement fields corresponding to real surgical procedures. Both the model build–up and the determination of bone displacements of course take into account the individual anatomy of the patient.
- *Raytracing of triangular Bézier patches by means of triangular Bézier clipping*
  The exploitation of the properties of the Bézier representation is shown to provide a means to ray trace the envisioned finite elements directly. Motivated from the triangular surface patches bounding the tetrahedral Bernstein–Bézier elements, triangular Bézier clipping is proposed in order to compute intersections between triangular Bézier patches and rays. This approach allows for the computation of ray–patch intersections at arbitrary accuracy, in reasonable time, and with few memory requirements. It is used in the implementation of a prototype raytracer for triangular Bézier patches. Thus the Bernstein–Bézier representation not only offers a means to achieve higher order continuity but also lends itself well for high quality rendering.

It is important to note that the thesis does not cover subjects such as the real–time simulation of surgical procedures and the development of biomechanical models representing the exact properties of facial tissue. However, directions for further readings, e.g. concerning the simulation of real–time cutting and biomechanics, are indicated at corresponding points in the text.

Further, we restrict ourselves to the simulation of more common procedures in cranio–maxillofacial surgery which do not result in large displacements. Consequently, the effects of geometrical nonlinearity are considered negligible and will not be taken into account.

1.3 OUTLINE AND ORGANIZATION

The remainder of the thesis is organized as follows:

- *Chapter 2* gives a survey of deformable models in computer graphics with an emphasis on facial surgery simulation systems. Geometric as well as physically–based deformable models are shortly revisited.
- *Chapter 3* introduces the finite element method. The first, more formal part gives an introduction the theory, whereas the second part is devoted to finite element procedures with respect to the static simulation of elastic materials. Both the simulation of incompressibility and the incorporation of a nonlinear stress–strain relationship are paid attention to.
- *Chapter 4* reviews the theory of Bernstein–Bézier representations. Emphasis is put on barycentric patches, i.e. patches over triangular and tetrahedral domains, as well as on the construction of C^1–continuous patch transitions. Bivariate and trivariate interpolants are revisited and evaluated with respect to their suitability for finite element modeling.
- *Chapter 5* puts the Bernstein–Bézier technique into the context of finite element analysis. While this is relatively obvious for a one–dimensional example it is extremely demanding for tetrahedral elements. Taking a C^1 trivariate Bernstein–Bézier interpolant as the starting point, a tetrahedral C^1 finite element is developed. Extensive tests on synthetic examples and a comparison to C^0 elements of various degrees conclude the chapter.
- *Chapter 6* introduces the concept of triangular Bézier clipping as a means of computing intersections between rays and triangular Bézier patches. A survey over related work is given and a prototype raytracer for triangular Bézier patches is presented.
- *Chapter 7* summarizes the implementation of the prototype simulator. Both the facial model build–up adapted to individual anatomy and the computation of displacement fields representing the bone movements of actual surgery are paid attention to. The registration of data sets in a common coordinate frame proves to be a prerequisite for the model build–up as well as for the computation of displacement fields. Therefore, registration constitutes a major part of the chapter. The presentation of results achieved on the example of a test patient concludes the chapter.
- *Chapter 8* concludes the thesis with a summary, the presentation of findings and directions for future work.
- The *Appendix* includes references, nomenclature and symbols used in the preceding chapters, as well as the color plates.

CHAPTER 2

DEFORMABLE MODELS IN COMPUTER GRAPHICS

Deformable models have been a major research interest in computer graphics for decades. Approaches for modeling of deformations vary considerably: purely geometric approaches, mainly used in computer aided design and modeling, attempt to edit the shape of an object by moving control points and optionally adjusting weights. In contrast, physical models are based on continuum mechanics and account for external and internal forces, changing material properties, and prescribed displacements. Physically–based models divide into several categories with respect to their computational realization: mass–spring models, finite differences, finite elements, and boundary elements to mention only the most important.

Deformable models have found a wide area of application. In computer aided design and computer drawing, geometric models are used to create and edit curves, surfaces, and even solids. In computer vision, deformable models are applied in the context of image analysis and segmentation. Image features exhibit attracting or repulsive forces on physically–based models such as snakes or balloons for edge detection and image segmentation [72]. Deformable models have also found applications in animation, particularly of clothes (e.g., [5]), as well as in synthesizing facial expressions (e.g., [78, 84]). Last but not least, surgery simulation demands for ever more sophisticated models and solving schemes, both in the context of real–time surgery training [15, 14, 22, 139] and for surgery planning [35, 106, 73, 119, 77, 143].

In the first two sections of the chapter, geometric and physically–based models will be shortly revisited. For a more detailed survey, the reader is referred to [51] and the references therein. In a subsequent section, surgery simulation systems with an emphasis on facial surgery simulation will be looked into.

2.1 GEOMETRIC MODELS

Geometric models do not capture physical properties, but mainly rely on user interaction in order to define the deformations. In the field of computer aided geometric design and in animation many representations of curves and surfaces have been developed, thus offering engineers and designers a means to define and manipulate geometric structures.

Parametric representations. *Parametric representations* define structures as a weighted sum of basis functions of one or more parameters. These representations include Bézier splines, B–splines, and NURBS, to mention only the most important [43]. In one dimension, the resulting structure is referred to as a spline, whereas in higher dimensions it is called a patch. In this thesis, Bernstein–Bézier representations will play an important role in the definition of a new kind of finite element. Therefore we give an extensive introduction to the subject in Chapter 4. In Chapter 5, we will then introduce a Bernstein–Bézier finite element, and thus in some way combine geometric and physically–based models.

Such parametric representations are computationally very efficient and allow for interactive editing. The flexibility of geometric models increases with the number of control points and patches used to represent the model. However, although offering a very fine–grained level of control, a large number of control points is sometimes very tedious to manipulate and thus editing parametric curves and surfaces in an exact manner demands experience and skill. As a consequence, even in shape design, there have been efforts to facilitate editing by introducing physics based interaction metaphors, see e.g. [26].

Free–form deformation. Instead of reverting to physics based editing, *free–form deformation* offers a higher level of control by defining a smooth deformation on the space including the objects embedded within that space. The original algorithm utilizes a trivariate tensor–product parametric Bézier solid defined by a lattice of control points [127]. Deforming an object now consists of calculating its local coordinates within the unit cube and subsequent deformation using the Bézier representation of the deformed lattice. Trivariate B–splines [70] and lattices of arbitrary topology [88] have later been used to define the deformation.

Implicit modeling. Implicit modeling defines geometric objects as contours, i.e. isosurfaces, through some scalar field in three–dimensional space. In contrast to parametric surfaces, implicit surfaces can easily describe smooth shapes, such as blobs and metaballs. Therefore, implicit surfaces have become a powerful tool in a growing number of graphics applications such as animation, graphic design, computer aided design and even visualization [131]. For an introduction to implicit surfaces, see e.g. [17].

Mesh–based modeling. In recent years, mesh–based representation and manipulation schemes based on subdivision surfaces have emerged. Subdivision schemes can be regarded as an algorithmic generalization of classical spline techniques. They provide globally smooth surfaces of arbitrary shape by iteratively applying simple refinement rules to the given control mesh (e.g. [76]). Further, they implicitly provide a multi–resolution representation.

2.2 PHYSICALLY–BASED MODELS

As a consequence of the somewhat mutually exclusive demands for accuracy and interactivity, physically–based models can be divided into two categories: interactive models mainly focussing on speed and low latency, as opposed to more accurate models with emphasis on physical correctness.

Models fitting the first category must provide interactive feedback to the user. In surgery simulation with haptic feedback, deformations have to be computed in real time. The same holds for the response to user input in virtual environments. As a consequence, efficient schemes such as mass–spring models are used in real–time simulations.

In contrast to the first category, more accurate models most often are simulated off–line. The models are based on the constitutive laws of the material and solve the resulting partial differential equations numerically. This is accomplished by means of the finite element method or, more recently, the boundary element method.

A good starting point for studying physically–based deformable models in computer graphics are the pioneering papers by Terzopoulos et al. [140, 141]. Their models incorporate elastic and inelastic deformation including viscoelasticity, plasticity and even fracture.

In the following sections, we will revisit the most important mathematical and computational techniques used for modeling deformable objects in computer graphics.

2.2.1 Mass-Spring Models

Mass–spring models represent objects as a collection of point masses connected by springs, often structured as a regular grid (see Figure 2.1).

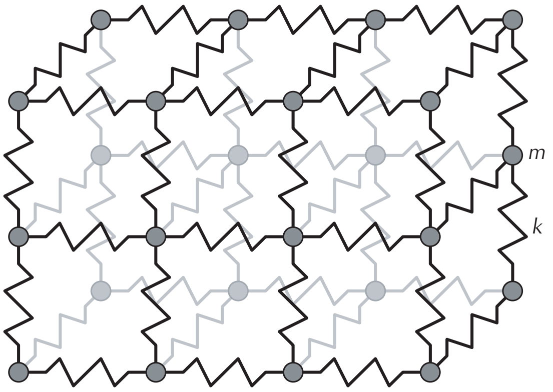


FIGURE 2.1 Mass–spring model as spring interconnected masses on a regular grid [51].

Theory. In a dynamic system, the masses m_i at positions $\mathbf{x}_i \in \mathbb{R}^3$ at time t are governed by Newton's second law of motion

$$m_i\ddot{\mathbf{x}}_i(t) + \gamma\dot{\mathbf{x}}_i(t) + \mathbf{f}_i^{\text{int}}(t) = -\mathbf{f}_i^{\text{ext}}(t) \tag{2.1}$$

where γ denotes a damping factor, $\mathbf{f}_i^{\text{int}}(t)$ refers to the internal forces resulting from spring interconnections and $\mathbf{f}_i^{\text{ext}}(t)$ represents the sum of external forces applied by the user or due to gravity or collision. The equations of motion for the entire system result from assembling the equations of all masses m_i in the lattice. Writing the positions of all n

masses component–wise into a position vector $\mathbf{x}$ of size $3n$, we can state a matrix equation corresponding to Equation 2.1 for the entire mass–spring system as

$$\mathbf{M}\ddot{\mathbf{x}} + \mathbf{D}\dot{\mathbf{x}} + \mathbf{K}\mathbf{x} = -\mathbf{f} \tag{2.2}$$

where $\mathbf{M}$, $\mathbf{D}$, and $\mathbf{K}$ are $3n \times 3n$ matrices representing mass, damping and stiffness, respectively. Although possibly large, these matrices are very sparse. $\mathbf{M}$ and $\mathbf{D}$ are diagonal, where $\mathbf{K}$ in a regular lattice is banded according to adjacency, i.e. interconnecting springs, between masses.

The system of equations in Equation 2.2 is numerically integrated through time by reducing it to two coupled systems of first–order differential equations as

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{v} \\ \dot{\mathbf{v}} &= \mathbf{M}^{-1}(-\mathbf{D}\mathbf{v} - \mathbf{K}\mathbf{x} - \mathbf{f}) \end{aligned}$$

Several numerical integration schemes are at disposal for computing the positions $\mathbf{x}$ and speeds $\mathbf{v}$ as functions of time given initial values for $\mathbf{x}$, $\mathbf{v}$, $\mathbf{f}_i^{\text{int}}(t)$ and $\mathbf{f}_i^{\text{ext}}(t)$. The simplest scheme is Euler's method. More sophisticated schemes include the midpoint method and higher order Runge Kutta methods, see e.g.[112].

Advantages and Limitations. Mass–spring systems have been applied in a variety of applications. They are simple to construct and very efficient. Thus, they allow for real–time computations which makes them attractive both for animation and simulation, e.g. in the context of surgery training [15, 14]. In addition, the technique has been extended to model non–linear materials and even liquids.

In theory, mass–spring systems are applicable for static and dynamic deformations. For static deformations, i.e. computing the rest position of a system after applying deforming forces, they are not well suited due to slow convergence and numerical instability. However, Teschner gives an alternative direct way of estimating deformation which improves upon these shortcomings and is both fast and robust [143].

Further, mass–spring lattices offer a rather coarse approximation of true physics. Spring constants must be adjusted to deliver plausible material behavior which often turns out to be a difficult task. In addition, certain constraints and material properties, such as incompressibility or the behavior of thin surfaces cannot be modeled at all. Last but not least, in the simulation of nearly rigid objects or in modeling hard constraints, mass–spring systems can behave stiff. This is due to large spring constants which result in poor stability and numerical condition.

As a consequence, for credible simulation of more complex objects, such as are encountered in facial animation or surgery simulation, it is advantageous to revert to continuous mathematical models such as the finite differences or the finite element technique which are shortly described in the following.

2.2.2 Finite Differences

Closely related to the mass–spring systems and an intermediate step on the way to finite element models is the method of finite differences. It is applicable for regular, i.e. rectilinear or curvilinear grids, in one ore more dimensions. The method of finite differences can be applied to the solution of either the differential or variational formulation of the mathematical model (compare Section 3.2, [8]).

The basic ingredients of a finite difference model are the discretization of the object into a finite number of points in combination with finite difference approximations of the derivatives at the points. It is general practice to approximate derivatives by means of central differences

$$\left.\frac{dw}{dx}\right|_i = \frac{w_{i+1} - w_{i-1}}{2h} \qquad \left.\frac{d^2w}{dx^2}\right|_i = \frac{w_{i+1} - 2w_i + w_{i-1}}{h^2}$$

where *h* denotes the distance between two adjacent points. Forward and backward differences can be used, too, but bear the disadvantage of not being symmetrical.

Finite differences schemes are simple and efficient. Matrix operators very similar to those with mass–spring system are easily generated and the resulting sparse systems of equations in general are easy to solve [140]. However, the inherent linear approximation and mass lumping at the nodes similar to mass–spring models is a disadvantage which renders finite differences far less general and powerful a method than the finite element approach described below.

2.2.3 Finite Element Models

In this section, we will oppose the finite element method to other physically–based deformable models. For a detailed, more formal introduction to the finite element method, the reader is referred to Chapter 3 where the method is theoretically motivated and put into the context of static elastomechanics.

Theory. Put in very simple words, the finite element method finds an approximation for a continuous function that satisfies an equilibrium condition which follows from the variational or weak formulation of the problem (compare Section 3.2). The discretization of the problem consists of decomposing its domain into a mesh of carefully selected elements, joined at discrete nodes. The solution of the variational equations is expanded as a weighted sum of finite element basis or shape functions on each element. Continuity across element boundaries is achieved by sharing discrete nodes and thus finite element weights. As a next step, the contributions of each element are assembled into a global system of equations which then can be solved for the shape function weights.

For a static elasticity problem, assembling the contribution of each element yields the familiar system of equations

$$\mathbf{KU} = \mathbf{F}$$

where **K** is the so–called stiffness matrix, **U** is the unknown vector of shape function weights and **F** is the load vector which represents the external forces. **K** is computed from the problem and results basically from integrating the weak form of the problem over the elements. Similar to the situation with mass–spring systems, **K** is sparse which speeds up solving.

For a dynamic simulation, the resulting system of equations is of the form

$$\mathbf{M\ddot{U}} + \mathbf{D\dot{U}} + \mathbf{KU} = \mathbf{F}$$

where again **M**, **D**, and **K** represent mass, damping and stiffness matrix, respectively, **U** is the weight vector, and **F** is the load vector.

Unlike mass–spring models or finite difference schemes, where the differential equations only hold at discrete points, the finite element method provides a continuous solution throughout the problem domain. Of course, the choice of elements decides on the degree of continuity. For a general introduction to the finite element method, the reader is referred to [8].

Advantages and Limitations. The finite element method has had a tremendous history in traditional engineering disciplines such as structural analysis, heat conduction and flow simulation.

The limitation of the finite element method in computer graphics lies primarily in its computational costs. The evaluation of the matrices $\mathbf{M}$, $\mathbf{D}$, and $\mathbf{K}$ generally involves numerical integration over each element and thus is computationally expensive. Further, real–time object deformation in theory necessitates ongoing re–evaluation of these matrices as the object deforms. As a consequence, in order to avoid re–evaluation, only small deformations of are taken into account often. In addition to the costs of integration, solving the resulting system of equations normally necessitates an iterative solving approach which is expensive, too.

Regardless of the limitations mentioned above and its rather recent history in computer graphics, the finite element method has seen many applications in facial models and surgery simulation some of which will be reviewed in Section 2.3. Further, the finite element method has successfully been applied for modeling [26] and muscle animation [27]. The issue of computational complexity has been addressed by means of pre–processing and algorithmic optimizations, such as condensation [22] and mass lumping [157], as well as through massive parallelization [139].

2.2.4 Boundary Element Method

It is only very recently that the boundary element method has been introduced to the field of computer graphics [69]. The authors present the method as a means to overcome the trade–off between accuracy and computational efficiency.

The boundary element method is applicable whenever we can come up with a boundary integral formulation for the boundary value problem under consideration. In the case of linearly deformable models this corresponds to applying integration by parts to the Navier equations of linear elasticity which produces boundary integrals form volume integrals. From this we see that in contrast to the finite element simulation of volumetric bodies, the unknowns are only on the boundary of the body under consideration. While this is a serious disadvantage for the simulation of inhomogeneities, stress, and deformations within the body, it is advantageous for computer graphics and haptics, since it is primarily the surface many applications in the field are interested in.

The boundary element method leads to a small set of equations represented as a dense system matrix. This again is in contrast to the finite element method which in general produces large sparse matrices. Although in general well–conditioned, solving these dense system prevents the method from application in real–time simulations when boundary conditions are allowed to change, thus changing the system matrix. Nevertheless, in [69], interactive deformation rates for linearly elastic materials are achieved by making use of the Sherman–Morrison–Woodbury formula [56]. This formula computes the inverse of a slightly changed matrix given the inverse of the original matrix. These optimizations together with precomputing a database used by the run–time solver eliminate the need of ongoing inversions from scratch of the system matrix and thus allow for real–time application.

2.3 FACIAL MODELS AND SURGERY SIMULATION SYSTEMS

Facial modeling and later surgery simulation systems have adopted more and more sophisticated deformable modeling techniques. This section first gives an overview of pioneering work done in facial modeling before focussing on surgery simulation systems, specifically on the research done in the field of facial surgery simulation.

2.3.1 Facial Models

The field of facial modeling has been an area of growing research efforts for nearly two decades. First approaches such as [104] were based on geometric deformations using parametric surfaces and aimed primarily at facial animation. Muscle models for facial expression modeling were introduced in [108] and improved upon in [153]. Both models included only little physical foundation and used springs to represent both muscles and the skin.

Later, physically–based simulation paradigms were adopted in order to model more accurately the physical properties of elastic materials (see e.g. [142]). Lee et al. [84] presented a promising approach to facial animation where they introduced a layered synthetic tissue model based on masses and springs connected to form prism–shaped elements (see Figure 2.2). The facial model is adapted from a template face, takes into account various anatomical aspects, and aims at facial animation. This layered tissue model was also adopted in facial surgery simulation systems, such as [73, 74].

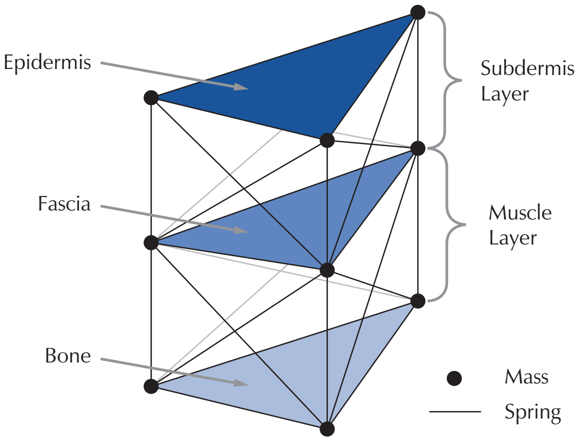


FIGURE 2.2 Layered synthetic tissue model according to [84].

A major motivation of introducing physically–based models to facial animation is the reduction of the parameters one would have to operate upon on a purely geometry based model. However, other approaches aiming at reducing the number of parameters have seen successful application in facial modeling, e.g. the morphable models in [107, 16]. A complete treatment of facial models and animation is beyond the scope of this thesis. For a survey of facial animation, see e.g. [103].

2.3.2 Surgery Simulation Systems

Parallel to and complementing on the research done on facial modeling and animation, facial surgery simulation systems and biomechanical models of facial tissue have been developed. As early as 1983, the idea of estimating facial soft tissue deformation due to bone realignments was first stated [150].

Back in 1986, Larrabee [82, 83] came up with a biomechanical finite element model of skin deformation. This work was followed by Deng's Ph.D thesis [35]. She developed a nonlinear thick shell model including a skin layer, a sliding layer, and a muscle layer, in order to simulate the closure of skin excisions on facial tissue in an analysis of plastic surgery. Following [8], she gave an incremental Lagrangian finite element formulation in order to obtain approximate solutions to the nonlinear equations representing the properties of skin and muscles. In 1991, Pieper [106] summed up his efforts to provide a system for computer–aided plastic surgery in his Ph.D thesis. Although being similar to Deng's work, he was the first to combine facial simulation with finite element modeling aiming at the planning of surgical procedures. However, his emphasis was on plastic surgery and therefore he concentrated on cutting and stretching of skin and epidermis rather than on repositioning bones. In contrast to Deng and with computational efficiency in mind, he restricted himself to the modeling of linear elasticity. However, his as well as Deng's model lacked the resolution required for a reliable simulation of very subtle changes in the appearance of a face and did neither provide C^1–continuous displacement fields nor smooth surfaces.

In 1994, Morten Bro–Nielsen introduced the modeling of elastic solids using active cubes [21]. The approach combined snake–like feature extraction in CT data sets with energy minimization, but did not achieve the precision required for surgery simulation. In 1996, he presented a promising approach designed for real–time application, which modeled both linear elasticity and dynamic behavior using finite elements [22]. Aiming at real–time simulation, he had to restrict himself to linear interpolation within the elements.

In 1996, Koch et al. [79] proposed a surface based finite element approach which adopted a method for physically–based sculpturing [26]. The facial tissue was represented as a C^1–continuous, thin plate finite element surface connected to the skull by springs. The model is generated directly from individual patient data sets. Applied to Visible Human Data Set [95], the model provided very promising results both in the field of surgery simulation and emotion editing [78]. Although taking into account anatomy, the model lacks true volumetric physics and therefore was unable to account for effects such as volume preservation, i.e. incompressibility.

Therefore, in the field of facial surgery simulation, researchers paid attention mainly to volumetric models in combination with more and more sophisticated finite element solution schemes. Keeve et al. [73] introduced Lee's layered tissue model to the field of facial surgery simulation. They combined the layered model with a mass–spring approach in order to solve the partial differential equations representing the tissue properties. Later, prism elements featuring linear interpolation functions were used in the C^0 finite element simulation of linear elasticity [74]. Individual patient models are built by adapting a generic facial model to the individual facial geometry.

As early as 1997, the use of the boundary element method was proposed for endoscopic simulation [93]. For models of low resolution, the authors achieved interactive simulation rates by means of pre–calculating displacements for unit tractions in all three spatial directions at each node of the surface discretization. The authors found their model to be superior to a mass–spring and spline model [92]. However, as a consequence of using the boundary element method, the model is not well suited to model inhomogeneities or volumetric effects within the body.

In 1998, we presented a versatile framework for the finite element simulation of soft tissue using tetrahedral Bernstein–Bézier elements [119]. For the first time, tetrahedral elements in combination with higher order interpolation were used for the representation of deformable tissue. In addition, the model incorporated incompressibility and allowed the simulation of a nonlinear stress–strain relationship. As did Keeve, we restricted ourselves to C^0–continuous interpolation across element boundaries.

Apart from the limitations mentioned above, all approaches discussed so far lacked an elaborate validation and error analysis with respect to real craniofacial surgery. In 1999, a technical report by Koch et al. overcame this shortcoming [80]. They described a volumetric finite element representation built on prism elements and featuring linear elasticity in combination with volume preservation. The simulation was C^0 in the interior combined with a C^1 surface similar to [79]. Four patient test cases were computed and compared against the real world surgery result. This and the preceding work [79, 78] are summed up in Koch's Ph.D. thesis [77].

In 2001, Teschner presented a means for the direct computation of soft tissue deformation in the context of cranio–maxillofacial surgery [143]. He used a multi–layered mass–spring representation of tissue, but instead of simulating the dynamic behavior by integration through time, he proposed a more robust method that directly determines a stable equilibrium. The tissue model considers nonlinear elasto–mechanical properties of soft tissue as well as gravity and turgor. However, due to the mass–spring approach, volumetric effects such as incompressibility are not taken into account. The tissue model was incorporated into a prototype system that allows for bone and tissue cutting.

Also in 2001, Gladilin et al. presented a validation of the model of linear elasticity for soft tissue simulation in craniofacial surgery [52]. They focussed on the simulation of large deformations and came up with a geometrically nonlinear finite element formulation similar to the one in [35]. They showed that the error due to geometrical linearization is substantial for large deformations in the range of 2 to 4cm. However, for small deformations up to 1cm the errors are negligible. Further, even for large displacements, the error is concentrated at the relocated bony structures, i.e. at the inside of the head, and therefore hardly visible on the facial surface. For efficiency reasons, the authors restricted themselves to linear interpolation within the elements.

2.4 THESIS FOCUS

Aiming at physical correctness and reliability, the physically–based simulation using finite elements has proved to be the method of choice for facial surgery simulation. We therefore prefer it over mass–spring or finite difference models.

The use of linear interpolation calls for a higher subdivision in order to reach the accuracy required in surgery simulation. This is a consequence of the fact, that convergence of finite element solutions can be achieved either by successive refinement of the mesh, or by increasing the polynomial degree of the interpolation on a fixed mesh. Furthermore, the use of prism elements restricts the geometry of the volume to shell–shaped structures.

While this is well suited for craniofacial surgery, its applicability is limited in situations of a more general geometrical or topological nature.

As a consequence, this thesis investigates the effect of higher order interpolation on tetrahedral finite elements and compares the results to pure mesh refinement in combination with linear interpolation. Both approaches allow for the construction of discrete analogues of arbitrarily high accuracy by introducing additional degrees of freedom. However, additional accuracy is only due to enlarging the dimension of the solution space. In two and three dimensions, the problem space grows enormously fast and thus may become a limiting factor in the computation. Therefore, an additional aim of the thesis is to reduce the solution space by imposing further constraints on the solutions, thereby reducing the problem dimension. The key idea is to increase the level of continuity of the solutions without destroying neither the approximation properties nor the simplicity of a local finite element basis. Consequently, in addition to C^0 finite elements, simulations using C^1–continuous elements are looked into and opposed to C^0 results.

Although the model of linear elasticity will be shown to be a C^0 problem in the next chapter, C^1 solutions may offer advantages. Besides from providing smooth surfaces given a smooth initial geometry, C^1 solutions may be preferred for stress and strain analysis and visualization. This will be made obvious in the next chapter where strain and stress will be defined in terms of partial derivatives of the resulting displacement function.

CHAPTER 3

FINITE ELEMENT MODELING OF ELASTIC MATERIALS

With increasing computing power, the finite element method has become one of the most important methods for the numerical solution of partial differential equations. Originally "invented" by engineering disciplines, the method has been given a thorough mathematical foundation over the past decades.

In the first part of this chapter, we will give an introduction to the ideas and mathematical concepts of the finite element method. To this aim, we will revisit the formulation of a mathematical model and its finite element discretization on the basis of a one–dimensional example problem. In the second part, we will propose a finite element procedure tailored for soft tissue modeling which is based on the simulation of elastic materials. Both incompressibility and the nonlinear response of elastic materials will be paid attention to. For an in–depth coverage of finite element theory and practice the reader is referred to [8, 68, 125, 160, 159].

The main ingredients of a finite element solution of a boundary value problem are the following:

- The variational or weak statement of the problem;
- The discretization of the domain of interest;
- The approximate solution of the variational equations by an expansion of the solution as a weighted sum of "finite element functions".

3.1 INTRODUCTORY CONCEPTS

Throughout the following chapter mathematical terminologies will be introduced on a gradual, as–needed basis. In contrast thereof, for the more theoretical first part of this chapter, some important concepts will be presented in this section.

3.1.1 Notations and Definitions

In the procedure employed to derive variational formulations from *functionals* – or *potentials* – of a given problem we will use the variational symbol δ. This symbol refers to the *variation* of a functional Π with respect to the function $u(t)$ and its derivatives.

Definition. The first **variation of** Π is defined as

$$\delta\Pi = \frac{\partial\Pi}{\partial u}\delta u + \frac{\partial\Pi}{\partial(du/dt)}\delta\left(\frac{du}{dt}\right) + \ldots + \frac{\partial\Pi}{\partial(d^k u/dt^k)}\delta\left(\frac{d^k u}{dt^k}\right). \tag{3.1}$$

Analogous definitions can be given for multivariate functions *u* by substituting partial derivatives for the “simple” derivatives. Looking at Equation 3.1, the similarity of $\delta\Pi$ to the total differential dF is easy to see. We can say that the variational operator δ acts like the differential operator with respect to the functions $u, du/dt, \ldots, du^k/dt^k$. Consequently, the variational operator complies to laws analogous to those of differentiation:

$$\delta(\Pi + \Phi) = \delta\Pi + \delta\Phi; \quad \delta(\Pi\Phi) = (\delta\Pi)\Phi + \Pi(\delta\Phi); \quad \delta(\Pi)^k = k(\Pi)^{k-1}\delta\Pi$$

$$\frac{d^k}{dt^k}\delta u = \delta\left(\frac{d^k u}{dt^k}\right); \quad \delta\int\Pi(t)\,dt = \int\delta\Pi(t)\,dt$$

In elasticity theory, especially in the derivation of the mixed formulation for incompressible materials (see Section 3.4.2) we will make use of tensors and indicial notation.

Definition. An ***n*–th rank tensor of order *m*** is a mathematical object in *m*–dimensional space which has *n* indices and m^n components and obeys certain transformation rules [8]. Each index of a tensor ranges over the number of dimensions in space. In tensor analysis, a tensor of rank 0 is also termed a **scalar**, whereas a tensor of rank 1 is referred to as a **vector v** written as v_i where $i = 1, \ldots, m$.

The notation of a tensor is similar to that of a matrix, $\mathbf{A} = (a_{ij})$. Tensor notation allows a very concise way of writing vector and more general identities. For example, the dot product can be written as

$$\mathbf{u} \cdot \mathbf{v} = u_i v_i. \tag{3.2}$$

On the right hand side of Equation 3.2, the summation sign has been omitted and repeated indices are summed over. This notation is referred to as the *summation convention* of indicial notation and is also termed *Einstein summation*. The multiplication of a matrix **m** and a vector **v**, e.g., can be written as $m_{ik}v_k$. We will use tensor notation in the constitutive relation that relates the stress to the strain tensor of linear elasticity (Section 3.4.1):

$$\tau_{ij} = c_{ijkl}\varepsilon_{kl}$$

3.1.2 Questions of Differentiability

Solving problems involving partial differential equations necessitates the consideration of differentiability properties of both the allowed initial or boundary data and the corresponding solutions. In general, for all input data with a given regularity, i.e. given differentiability properties, solutions with a certain type of regularity are to be obtained. The correspondence between the regularity of data and that of solutions is determined by the partial differential equations themselves. In order to gain more insight into the issue, questions of differentiability with respect to the finite element analysis of partial differential equations are discussed in the following.

Definition. We distinguish **open** and **closed unit intervals:**

$$\Omega = \,]0, 1[\qquad \text{(open)}$$

$$\overline{\Omega} = [0, 1] \qquad \text{(closed)}$$

For the development of the theory and a one–dimensional example we restrict ourselves to the unit domain. This can be done without loss of generality as it can be achieved by variable substitution.

Definition. A function $f:\Omega \to \mathbb{R}$ is said to be ***k*–times continuously differentiable**, or of class $C^k(\Omega)$ if its derivatives of order *j*, $0 \le j \le k$, exist and are continuous and bounded functions.

In general, finite element functions are smooth on element interiors but of lower continuity across element boundaries. The treatment of cross–boundary or inter–element continuity of the derivatives in geometric modeling and finite element procedures, respectively, demands special attention and will be treated in detail for our choice of basis functions in Chapter 4 and Chapter 5.

Mathematically, such piecewise differentiable functions may be treated by way of formal calculations based on the theory of distributions of functional analysis. The treatment of distributions is beyond the scope of this thesis. For further information, the reader is referred to [23, 154, 135, 109]. From this theory, we adopt the notion of *generalized functions* and *weak* or *distributional derivatives* which will be introduced in the following.

In order to calculate the derivatives of piecewise defined functions without regard to their differentiability, we need to introduce three special functions, the *unit step* or *Heavyside* function, the *box impulse,* and the *Dirac impulse* function:

Definition. The **Heavyside** or unit step function at $t = t_0$, $t_0 > 0$ is defined as

$$u(t-t_0) = \begin{cases} 1 & t > t_0 \\ 0 & t < t_0 \end{cases}.$$

Definition. The **box impulse** of height 1 and width *T* results form superimposition of two unit step functions as

$$u_T(t-t_0) = u(t-t_0) - u(t-t_0-T) = \begin{cases} 0 & t < t_0 \\ 1 & t_0 < t < t_0 + T \\ 0 & t > t_0 + T \end{cases}.$$

Definition. The **Dirac impulse function** can be represented as the limit of a box impulse of width $T \to 0$ and height $1/T$ at $t = t_0$

$$\delta(t-t_0) = \lim_{T \to 0} \frac{1}{T}[u(t-t_0) - u(t-t_0-T)]. \tag{3.3}$$

For continuous functions $h(t)$ the following identity holds:

$$\int_0^1 h(t)\delta(t-t_0)dt = h(t_0) \qquad 0 \le t_0 \le 1$$

This identity is of paramount importance in signal processing. If at t_0 the function h were discontinuous its value would be ambiguous. From Equation 3.3 it follows that

$$\delta(t-t_0) = \frac{d}{dt}u(t-t_0). \tag{3.4}$$

Using Equation 3.4 we find an identity for the first derivative of a piecewise differentiable function $f(t)$ with discontinuities of amplitude a_ν at t_ν, $\nu = 1, 2, \ldots, n$, as

$$\frac{d}{dt}f(t) = f'(t) + a_1\delta(t-t_1) + a_2\delta(t-t_2) + \ldots + a_n\delta(t-t_n) \tag{3.5}$$

with $f'(t)$ referring to the ordinary derivative of $f(t)$ where it is differentiable.

We refer to Equation 3.5 as a *generalized derivative* [68], which is a special case of a weak derivative (see below). In the context of finite elements, a generalized derivative eventually suffers from discontinuities at element boundaries but is well–defined on the interior. The value of the derivative at the boundary is irrelevant in the context of finite element theory as it has no effect on integrals and thus on properties like square–integrability which will prove important in the following. For example, the generalized first derivative of a piecewise linear function is a generalized step function (i.e. piecewise constant), the second derivative a generalized Dirac delta function (i.e., delta functions, of various amplitudes at the element boundaries). Figure 3.1 gives an example.

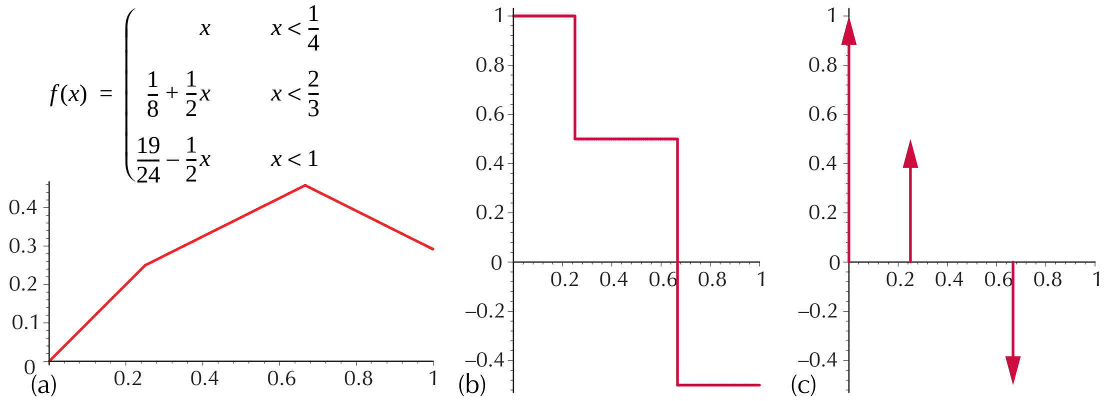


FIGURE 3.1 (a) Piecewise linear function
(b) Its generalized first derivative (a generalized step function)
(c) Its generalized second derivative (generalized Dirac delta function)

A function is called *smooth*, if it is of class C^1. There is a relation between the spaces of differentiable (smooth) functions and *Sobolev* spaces, which are introduced in the following. Before defining those spaces we need to introduce the concept of weak derivatives. The notion of weak derivatives is of paramount importance in the variational formulation of partial differential equations which serves as the basis of finite element solutions of such problems.

Definition. Let $u \in C^0(\overline{\Omega})$. Then v is **weak derivative** of u if

$$\int_\Omega v\varphi dt = -\int_\Omega u\varphi' dt \qquad \forall \varphi \in C_0^1(\overline{\Omega}) \tag{3.6}$$

with $C_0^1(\overline{\Omega}) := \{\varphi \in C^1(\overline{\Omega}),\ \ \varphi(0) = \varphi(1) = 0\}$, i.e. the space of C^1–continuous functions which evaluate to zero on the boundary.

The restriction $\varphi(\partial\Omega) = 0$ is necessary because Equation 3.6 corresponds to the rule of partial integration with vanishing terms at the borders:

$$\int_\Omega v\varphi dt = u\varphi|_{\partial\Omega} - \int_\Omega u\varphi' dt$$

In the context of finite element analysis, the functions $\varphi \in C_0^1(\overline{\Omega})$ are often termed *test functions.*

In analogy to Equation 3.6,

$$\int_\Omega v\varphi\, dt = (-1)^k \int_\Omega u\varphi^{(k)}\, dt \qquad \forall \varphi \in C_0^k(\overline{\Omega})$$

defines v as ***k***$^{\text{th}}$ **weak derivative** of u. It is possible to define weak derivatives in multiple dimensions [23, 154].

If a function $u(t)$ is differentiable in the strong, classical sense, i.e.

$$\frac{d}{dt}u(t_0) = \lim_{|h| \to 0} \frac{u(t_0 + h) - u(t_0)}{h}$$

for all $t_0 \in \overline{\Omega}$, then du/dt is equal to the weak derivative. If $u(t)$ is continuous but only piecewise differentiable, then the weak derivative is the discontinuous function equal to $u'(t)$ between, but undefined at the discontinuities of $u'(t)$. Strong derivatives can therefore be regarded as a special case of weak derivatives. As a convention, from now on, all derivatives are to be seen as weak derivatives.

We are now ready to introduce the Sobolev spaces as a generalization of the spaces of classically continuously differentiable functions.

Definition. A **Sobolev space of functions** is defined as follows:

$$H^k = H^k(\Omega) = \{u |\ u \in L^2;\ u' \in L^2;\ u'' \in L^2; \ldots ;\ u^{(k)} \in L^2\}$$

where

$$L^2 = L^2(\Omega) = \left\{ u |\ \ \|u\|^2 = \int_\Omega |u|^2\, dt < \infty \right\}. \tag{3.7}$$

A Sobolev space of functions of degree k consists of all functions with *square–integrable* weak derivatives through order k. Equation 3.7 gives the definition of a square–integrable function which is also called an L^2–function, or a function of *finite energy*. It is easy to see, that $H^0 = L^2$ and that $H^{k+1} \subset H^k$. In n dimensions, the Sobolev space is the space of functions, all of whose partial derivatives up to order k are square–integrable. More formally, the Sobolev space in n dimensions $H^k(\Omega)^n$ is composed of the functions u for which the quantity

$$\|u\|_k^2 = \sum_{i=0}^{k} \int_{\Omega^n} \left|D^i u\right|^2(\mathbf{t})\, d\mathbf{t} \tag{3.8}$$

is finite, with $|D^i u|^2$ denoting the sum of squares of all partial derivatives of u of order i, and $d\mathbf{t}$ denoting the n–dimensional differential with respect to the parameter vector $\mathbf{t}$.

A square–integrable function is in general defined only almost everywhere and, as stated above, the derivatives in Equation 3.8 must be interpreted as weak derivatives. The square integrability of weak derivatives is a prerequisite for the weak or variational formulation of boundary value problems which forms the basis of the finite element method (see Section 3.2.2).

However, the solution of a continuous problem should meet certain differentiability requirements with respect to the problem under consideration. For this reason one is interested in a connection between Sobolev spaces and functions whose derivatives exist pointwise in the classical sense. This connection is provided by the Sobolev imbedding theorem:

> ***Theorem.*** The **Sobolev imbedding theorem** states that, in one dimension, if a function is of class $H^{k+1}(\Omega)$ then it is a $C^k(\overline{\Omega})$ function:
>
> $$H^{k+1}(\Omega) \subset C^k(\overline{\Omega}) \tag{3.9}$$

In general, Equation 3.9 only holds for one–dimensional problems. In n dimensions, the theorem is as follows

$$H^m(\Omega)^n \subset C^k(\overline{\Omega})^n \qquad m > k + \frac{n}{2}. \tag{3.10}$$

Roughly speaking, the theorem states that m–times weakly differentiable functions loose one order of differentiability per half a space dimension compared to classically differentiable functions [154]. In other words, if a function u on $\mathbb{R}^n$ belongs to the Sobolev space H^m and if $k < m - n/2$ then there is a k–times continuously differentiable function which agrees with u everywhere except on a finite set of points.

From Equation 3.10 it follows that for solutions over three–dimensional domains to be continuous, the square–integrability of derivatives up to order two is needed [68]. However, the very restrictive formulation of the imbedding theorem in n dimensions addresses troublesome functions which cannot be piecewise polynomials the like of which are used in the context of finite element analysis [136]. Therefore, for practical considerations with respect to finite elements, Equation 3.9 is often applicable for higher dimensional problems and gives reason for the success of polynomial C^0 finite elements in a wide range of two– and three–dimensional problems such as linear elasticity (Section 3.4).

The theory of Sobolev spaces is of high importance in the analysis of existence and uniqueness of solutions as well as in the analysis of convergence of finite element solutions [8, 68, 136]. From this vast mathematical field, we will discuss only the question of admissibility of finite element basis functions in the context of the variational or weak form of the problem (see Sections 3.2.2 and 3.2.3).

For a more complete but rather formal treatment of Sobolev spaces and their impact to the formulation of the finite element method the reader is referred to [90].

3.2 CONTINUOUS–SYSTEM MECHANICAL MODELS

The analysis of soft tissue requires the idealization of tissue into a form that can be solved, the formulation of the mathematical and physical model, and the interpretation of the results. In general, such models encompass differential equations which are required to hold throughout the domain of the system. In addition, the definition of boundary conditions is a prerequisite for solving the problem. The exact solution of the differential equations is possible only for relatively simple problems, and numerical procedures must be employed in general.

We can distinguish between two different techniques which can be applied in formulating the governing differential equations: the *direct* or *differential* formulation and the *variational* or *weak* method. The latter finds its equivalent in the *principle of virtual displacements* which can be regarded as the basis of the finite element method.

In the remainder of the section we will revisit the basics of these formulations by means of the one–dimensional Poisson problem

$$\frac{d^2u}{dt^2} + f = 0 \qquad \text{on } \Omega. \tag{3.11}$$

This problem can be interpreted as an idealization of a bar of unit length subjected to a distributed load $f(t)$ within the bar which results in a longitudinal displacement $u(t)$. Figure 3.2 illustrates the situation.

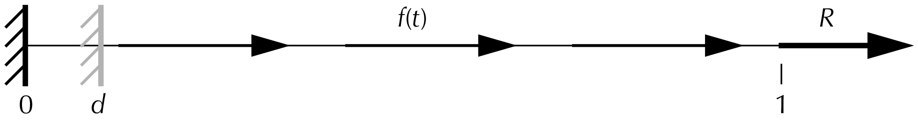


FIGURE 3.2 Idealized bar of unit length subjected to a distributed body load *f*(*t*) and a point load *R* at the end.

3.2.1 Differential Formulation

A boundary value problem for Equation 3.11 involves the imposition of boundary conditions on the solution function *u*. In this example we will require $u(0) = d$ (the beginning of the bar is shifted to the right) and $u'(1) = R$ (a concentrated load *R* is applied at the end). These two boundary conditions together with Equation 3.11 form the so–called *strong* or *differential formulation* of the problem:

$$(S)\begin{cases} \text{Given } f:\overline{\Omega} \to \mathbb{R}\text{, find } u:\overline{\Omega} \to \mathbb{R}\text{, such that} \\ \dfrac{d^2u}{dt^2} + f = 0 \qquad \text{on } \Omega & (3.12) \\ u(0) = d & (3.13) \\ u'(1) = R & (3.14) \end{cases}$$

Equation 3.12 is a statement of equilibrium at any point *t* within the bar, Equation 3.13 is called an *essential* or *geometric* boundary condition, and Equation 3.14 is a *natural* or *force* boundary condition. By means of these two different types of boundary conditions we will illustrate certain key features of variational formulations.

It is worth noting that Equation 3.12 is a second order differential equation and that the solution to the equation is required to be two times continuously differentiable.

3.2.2 Variational Formulation

To derive the variational or weak formulation of (S) we proceed similarly to the concept of weak derivatives. Instead of requiring the equilibrium of Equation 3.12 pointwise, everywhere in the bar, we only require it "in the average". To this aim, we need to characterize two classes of functions.

The first is to be composed of the *trial solutions*. These functions are required to meet the essential boundary conditions, i.e. in our example to be equal d at the beginning of the bar. The force boundary condition will not enter the definition of the problem. Further, we in this example need the trial solutions to be of class H^1, i.e. their first derivative must be in L^2. Thus, the set U of trial solutions is chosen as:

$$U = \{u |\ u \in H^1, u(0) = d\} \tag{3.15}$$

The second class of functions are the *variations* or *test functions* that have already been introduced in Equation 3.6 in the context of weak derivatives. This collection is very similar to the set U except that we require the homogeneous counterpart of the essential boundary conditions, in the example $v(0) = 0$. Thus, the set of variations V is given by:

$$V = \{v |\ v \in H^1, v(0) = 0\} \tag{3.16}$$

We may now state the *weak* or *variational form* of the example problem by multiplication of Equation 3.12 in (S) with a test function $v \in V$ and subsequent integration by parts (or more generally in *n* dimensions, by application of the divergence theorem). This yields:

$$\text{(W)} \begin{cases} \text{Given } f : \bar{\Omega} \to \mathbb{R}\text{, find } u \in U \text{ with} \\ \displaystyle\int_0^1 \frac{du}{dt}\frac{dv}{dt}dt = \int_0^1 fv dt + Rv(1) \\ \text{for all } v \in V. \end{cases} \tag{3.17}$$

For the details of the derivation of the weak form in Equation 3.17, the reader is referred to the proof of proposition (a) in Section 3.2.3.

In mechanics, the variations *v* are often written as δu or $\bar{u}$, and termed *virtual displacements*. Equation 3.17 is then referred to as the *principle of virtual displacements*, or the *principle of virtual work*. This principle states that the equilibrium of the body in the problem requires that for any small virtual displacement δu (which is zero at the points and surfaces of the body and corresponds to the essential boundary conditions) imposed on the body in its state of equilibrium, the total internal work is equal to the total external work [8]. In Equation 3.17, the left hand side corresponds to the internal work, whereas the right hand side represents the total external work done by the distributed force within the bar and the tip force at the end.

There is a second way in which to arrive at the weak problem statement (W). If we know a *potential* or *functional* Π of a problem we can invoke the stationarity of Π, i.e. $\delta\Pi = 0$, which yields the weak statement (W). It should be noted that appropriate functionals to a problem are not unique in general. Often they are found based on the observation of physical laws, such as the principle of minimal potential energy in mechanics. Such a potential for the above example problem would be of the form:

$$\Pi = \int_0^1 \frac{1}{2}\left(\frac{du}{dt}\right)^2 dt - \int_0^1 uf\,dt - Ru(1) \tag{3.18}$$

Using Equation 3.1, it is easy to see that invoking $\delta\Pi = 0$ for the potential in Equation 3.18 yields Equation 3.17. Bathe in [8] classifies problems with respect to the highest order of a derivative of a state variable in the functional of the problem. If the operator contains at most derivatives of order *m*, then the problem is termed a C^{m-1} *variational problem*. This is in agreement with the Sobolev imbedding of H^m in C^{m-1} according to Equation 3.9. We can observe, that the order of derivatives in the essential boundary conditions in a C^{m-1} problem is at most $m-1$.

In the finite element literature, it is common practice to write Equation 3.17 in the following short form:

$$a(u, v) = (f, v) + Rv(1) \tag{3.19}$$

The left hand side of Equation 3.19 is a *symmetric bilinear form* $a(\cdot,\ \cdot)$ and the right hand side is called a *linear functional* $(f,\ \cdot)$, often written as the inner product $\langle \cdot,\ \cdot \rangle$. Of course, these forms depend on the problem. As before, *u* denotes the exact displacements, the trial solutions, and *v* refers to any admissible virtual displacement (i.e. zero at the points and surfaces of the body and corresponding to the essential boundary conditions).

Let $u_1, u_2, v \in C^2(\overline{\Omega})$ and $\lambda_1, \lambda_2 \in \mathbb{R}$. Then the following properties hold:

$$\left.\begin{aligned} a(\lambda_1 u_1 + \lambda_2 u_2, v) &= \lambda_1 a(u_1, v) + \lambda_2 a(u_2, v) \\ a(u, \lambda_1 v_1 + \lambda_2 v_2) &= \lambda_1 a(u, v_1) + \lambda_2 a(u, v_2) \end{aligned}\right. \quad \text{bilinearity of } a(\cdot,\ \cdot) \tag{3.20}$$

$$a(u_1, u_2) = a(u_2, u_1) \quad \text{symmetry of } a(\cdot,\ \cdot) \tag{3.21}$$

$$(f, \lambda_1 v_1 + \lambda_2 v_2) = \lambda_1 (f, v_1) + \lambda_2 (f, v_2) \quad \text{linearity of } (f,\ \cdot) \tag{3.22}$$

In applications, $a(\cdot,\ \cdot)$ stands for the internal energy of a system, whereas $(f,\ \cdot)$ refers to external work.

3.2.3 Equivalence of Strong and Weak Form

Clearly there must be a relation between the weak and the strong formulation of a problem, as it would not make sense to introduce the weak formulation otherwise. It turns out that the solutions are identical.

Consider the following proposition.

Proposition. (a) Let *u* be a solution of (S). Then *u* is also a solution of (W).
(b) Let *u* be a solution of (W). Then *u* is also a solution of (S).

Similar to [68], we in the following give a short and rather informal proof of this proposition by means of our example problem.

(a) If *u* is a solution of (S), then $u'' + f = 0$ on Ω (Equation 3.12). Then

$$
\begin{aligned}
0 &= -\int_0^1 v(u'' + f)dt && \forall v \in V \\
&= -\int_0^1 vu''dt - \int_0^1 vf\,dt && |\ \text{partial integration} \\
&= \int_0^1 u'v'dt - vu'\big|_0^1 - \int_0^1 vf\,dt && |\ v(0) = 0, u'(1) = R \\
&= \int_0^1 u'v'dt - Rv(1) - \int_0^1 vf\,dt \\
\Rightarrow & \int_0^1 u'v'dt = \int_0^1 vf\,dt + Rv(1) && (*)
\end{aligned}
$$

As *u* is a solution of (S), then $u(0) = d$ and therefore *u* is in *U*. Since *u* also satisfies (*) for all $v \in V$, *u* is a solution of the weak statement.

(b) If *u* is a solution of (W), then $\int_0^1 u'v'dt = \int_0^1 fvdt + Rv(1)$ (Equation 3.17) for all $v \in V$. Further, $u \in U$ and consequently $u(0) = d$. Then integrating by parts together with $v(0) = 0$ yields

$$
\begin{aligned}
vu'\big|_0^1 - \int_0^1 vu''dt &= \int_0^1 vf\,dt + Rv(1) \\
v(1)u'(1) &= \int_0^1 v\ (u'' + f)dt + Rv(1) \\
0 &= \int_0^1 v\ (u'' + f)dt - v(1)[u'(1) - R] \qquad (*)
\end{aligned}
$$

It remains to show that (*) implies the following

(i) $u'' + f = 0$ on Ω
(ii) $u'(1) - R = 0$

(i) We define

$$v = \phi\ (u'' + f) \qquad (**)$$

with $\phi(t) \in V$ and, in addition, $\phi(t) > 0$ on $\Omega =]0, 1[$ as well as $\phi(0) = \phi(1) = 0$. Substituting (**) into (*) yields

$$0 = \int_0^1 \phi\ (u'' + f)^2 dt - 0 .$$

This implies (i) on $\Omega =]0, 1[$.

(ii) Using (i) we conclude from (*)

$$0 = v(1)[u'(1) - R] \qquad (***)$$

This $v \in V$ is arbitrary at $t = 1$. Assuming $v(1) \neq 0$, (***) implies (ii). Thus *u* is a solution of the strong form, too.

At this time, it is illustrative to make the following observation. Looking at the weak formulation, we realize that the force boundary conditions are not stated explicitly but included as a potential in the expression of Π. In contrast thereof, trial solutions are explicitly required to satisfy the essential boundary conditions. It is this feature of weak formulations that makes them a particularly powerful mechanism for the analysis of continuous systems using the finite element method.

Of course, the space of functions in which we look for solutions of the weak problem statement contains the space of functions in which we find solutions of the strong formulation. Thus, and as a consequence of the equivalence of the two formulations, we observe that if a strong problem can be solved, it can also be solved exactly using the weak formulation. Comparing the second order differential Equation 3.12 in the strong form (S) to Equation 3.17 in (W) we see that the weak expression only involves first order derivatives, due to the integration by parts in part (a) of the above proof. It is this fact which allows for enlarging the space of admissible functions from $H^2(\Omega)$ to $H^1(\Omega)$.

In general, we can state the following condition for admissibility of trial solutions: in order to solve a problem involving a differential equation of order $2m$, the trial functions and their first $m-1$ derivatives should be continuous (see e.g. [136]). Thus, the solution and its derivatives up to the *m*-th have finite energy and consequently are in the admissible Sobolev space $H^m(\Omega)$. This is in accordance with Bathe's C^{m-1} variational problems mentioned before.

In addition, due to the weak problem statement and due to the fact that force boundary conditions are built into the functional, the weak formulation can be applied to problems where discontinuities e.g. of material properties are present or in case of concentrated loads being applied within the bar. In these cases the first derivative of *u* is discontinuous, a fact which in a strong problem formulation would have to be treated in a piecewise manner with each continuous section connected to the adjoining section by means of corresponding boundary conditions.

3.3 GALERKIN APPROXIMATION AND MATRIX FORMULATION

In a next step of the derivation of a finite element solution for boundary value problems we have to come up with a method of obtaining an approximate solution based on the weak formulation. This is achieved by means of a process referred to as *Galerkin projection* which leads to the matrix formulation of the problem.

3.3.1 Galerkin Projection

The key idea is to project the solution into a finite–dimensional functional space. To this aim, we construct finite–dimensional subspaces of the spaces U and V in Equation 3.15 and Equation 3.16, respectively. We will denote these subspaces by U^h and V^h

$$U^h \subset U \qquad V^h \subset V \tag{3.23}$$

where the superscript *h* refers to the fact, that the subspaces are associated with a discretization of the domain Ω. Most often, *h* is taken to be the largest subinterval in the mesh which defines the discretization. Section 3.3.2 will give a more intuitive meaning to the value of *h*. From Equation 3.23 we see that functions in U^h and V^h must be in agreement with the properties of *U* and *V*, respectively, i.e. for the example problem $u(0) = d$ and $v(0) = 0$.

In finite element analysis and starting out from a given function space V^h of "discrete" variations or test functions, we now for each $w^h \in V^h$ define the corresponding trial solution $u^h \in U^h$ as

$$u^h = w^h + b^h \tag{3.24}$$

where b^h refers to a given function satisfying the essential boundary conditions, e.g. in our example problem $b^h(0) = d$. With the definition of *V* in Equation 3.16 it is easy to see that such trial solutions u^h are in agreement with the essential boundary conditions, too: $u^h(0) = w^h(0) + b^h(0) = 0 + d$. Thus, all functions satisfying Equation 3.24 constitute the space U^h. Using Equation 3.24, we see that the spaces U^h and V^h are identical up to the function b^h, which "represents the essential boundary conditions".

The variational problem statement in Equation 3.19 is now of the form

$$a(u^h, v^h) = (f, v^h) + Rv^h(1) \tag{3.25}$$

and defines an approximate solution u^h in the finite–dimensional space of trial solutions U^k. Substituting Equation 3.24 into Equation 3.25 together with the bilinearity of $a(\cdot, \cdot)$ (Equation 3.20) yields the *Galerkin formulation* of the problem:

$$(\mathrm{G}) \begin{cases} \text{Given } f, b{:}\bar{\Omega} \to \mathbb{R}\text{, find } u^h = w^h + b^h \\ \text{with } w^h \in V^h \text{ such that} \\ a(w^h, v^h) = (f, v^h) + Rv^h(1) - a(b^h, v^h) \\ \text{for all } v^h \in V^h. \end{cases} \tag{3.26}$$

The right hand side of Equation 3.26 shows that the imposition of essential boundary conditions now is represented in the form $a(b^h, v^h)$ as a part of the external work. This reflects the elimination of degrees of freedom corresponding to prescribed displacements. Further, the symmetric bilinear form $a(w^h, v^h)$ on the left hand side now operates on functions *out of the same function space* of variations V^h.

In order to arrive at the matrix formulation, we will have to define this space together with the function b^h. The basic idea herein is to define functions in V^h as linear combinations of *n* linearly independent functions $N_i{:}\bar{\Omega} \to \mathbb{R}$:

$$v^h = \sum_{i=1}^{n} \hat{v}_i N_i \tag{3.27}$$

In finite element terminology, the functions N_i are referred to as *basis* or *shape functions*, whereas the $\hat{v}_i$ are referred to as *weights*. In our example problem, these functions, of course, must satisfy $N_i(0) = 0$ as they are supposed to span the space of variations V^h which are zero at the essential boundary conditions.

In order to define the space of trial solutions analogously, we need to specify b^h in terms of an additional shape function $N_0 : \overline{\Omega} \to \mathbb{R}$ which, for the example problem, has the property $N_0(0) = 1$. Thus we have for b^h

$$b^h = dN_0 . \tag{3.28}$$

Substituting Equation 3.27 and Equation 3.28 into Equation 3.24 yields

$$u^h = w^h + b^h = \sum_{j=1}^{n} \hat{w}_j N_j + dN_0$$

as a typical solution of (G).

Next, we substitute Equation 3.27 and Equation 3.28 into the Galerkin Equation 3.26 and find

$$a\left(\sum_{j=1}^{n} \hat{w}_j N_j, \sum_{i=1}^{n} \hat{v}_i N_i\right) = \left(f, \sum_{i=1}^{n} \hat{v}_i N_i\right) + R \sum_{i=1}^{n} \hat{v}_i N_i(1) - a\left(dN_0, \sum_{i=1}^{n} \hat{v}_i N_i\right). \tag{3.29}$$

Using the properties of $a(\cdot, \cdot)$ and $(f, \cdot)$ (Equation 3.20 – Equation 3.22) we find

$$\begin{aligned}
\sum_{i=1}^{n} \hat{v}_i \underbrace{a(\sum_{j=1}^{n} \hat{w}_j N_j, N_i)}_{= \sum_{j=1}^{n} \hat{w}_j a(N_i, N_j)} &= \sum_{i=1}^{n} \hat{v}_i (f, N_i) + \sum_{i=1}^{n} \hat{v}_i N_i(1) R - \sum_{i=1}^{n} \hat{v}_i \underbrace{a(dN_0, N_i)}_{= da(N_i, N_0)} \\
0 &= \sum_{i=1}^{n} \hat{v}_i \left[\sum_{j=1}^{n} \hat{w}_j a(N_i, N_j) - (f, N_i) - N_i(1) R + da(N_i, N_0) \right] \\
0 &= \sum_{i=1}^{n} \hat{v}_i g_i \qquad (3.30)
\end{aligned}$$

with $$g_i = \sum_{j=1}^{n} \hat{w}_j a(N_i, N_j) - (f, N_i) - N_i(1) R + da(N_i, N_0) . \tag{3.31}$$

As the Galerkin equation must hold for every $v^h \in V^h$ it must also hold for all $\hat{v}_i, i = 1, \ldots, n$, due to Equation 3.27. Since the $\hat{v}_i$ are arbitrary in Equation 3.30 we can conclude that the $g_i, i = 1, \ldots, n$, in Equation 3.31 must be equal zero. Thus we find:

$$\sum_{j=1}^{n} \hat{w}_j a(N_i, N_j) = (f, N_i) + N_i(1) R - da(N_i, N_0) \qquad \text{for } i = 1, \ldots, n . \tag{3.32}$$

Equation 3.32 constitutes a system of n equations for n unknowns, i.e. $\hat{w}_i, i = 1, \ldots, n$.

An other, maybe more intuitive but less elegant way of arriving at this $n \times n$ system of equations is given by substituting *n* unit virtual displacements for $k = 1, \ldots, n$

$$\hat{v}_i = \begin{cases} 1 & k = i \\ 0 & \text{otherwise} \end{cases}$$

into Equation 3.29. Thus we arrive at *n* equations which can be solved for the $\hat{w}_i$. Substituting unit virtual displacements is legitimate since the variations are arbitrary.

3.3.2 Matrix Formulation

In order to arrive at the matrix formulation of the problem we adopt the following short notation:

$$\begin{aligned} K_{ij} &= a(N_i, N_j) \\ F_i &= (f, N_i) + N_i(1)R - da(N_i, N_n) \end{aligned}$$

Equation 3.32 consequently becomes

$$\sum_{j=1}^{n} K_{ij}\hat{w}_j = F_i \qquad \text{for } i = 1, \ldots, n.$$

Writing all *n* equations simultaneously yields the matrix notation

$$\mathbf{Kw} = \mathbf{F}$$

with

$$\mathbf{K} = \left[K_{ij}\right] = \begin{bmatrix} K_{11} & K_{12} & \cdots & K_{1n} \\ K_{21} & K_{22} & \cdots & K_{2n} \\ \vdots & \vdots & & \vdots \\ K_{n1} & K_{n2} & \cdots & K_{nn} \end{bmatrix} \tag{3.33}$$

$$\mathbf{F} = \left[F_i\right] = \left[F_1\ F_2\ \ldots\ F_n\right]^T$$

$$\mathbf{w} = \left[\hat{w}_j\right] = \left[\hat{w}_1\ \hat{w}_2\ \ldots\ \hat{w}_n\right]^T$$

In the above equations, **K** is termed the *stiffness matrix*, **F** is referred to as the *force* or *load vector* and **w** is the *weight* or *displacement vector*.

The matrix equivalent of the Galerkin problem statement (G) now reads

(M) Given the stiffness matrix **K** and the force vector **F**, find **w** such that

$$\mathbf{Kw} = \mathbf{F}.$$

It is easy to see, that the stiffness matrix **K** must be *symmetric*. This follows directly from the symmetry of the bilinear form since $K_{ij} = a(N_i, N_j) = a(N_j, N_i) = K_{ji}$. In order to observe more properties of **K** we have to point out to some additional features of typical shape functions. These additional properties are "unnecessary" from the Galerkin point of view but enable fast schemes to be applied for the stiffness matrix computation and subsequent solving of the resulting system of linear equations.

At this time, we should recall that the evaluation of the entries of the stiffness matrix and force vector involves the computation of integrals which have been hidden behind the notations $a(\cdot, \cdot)$ and $(f, \cdot)$. Since the computation of integrals, it be done analytically or numerically, is time consuming, the amount of such computation should be kept at its possible minimum. It is obvious, that a compact support of shape functions will keep the matrix sparse, since many of the integrals K_{ij} will trivially evaluate to zero. In addition to fast and efficient computation, sparsity of the stiffness matrix will allow for memory saving data structures and fast solving schemes.

In order to demonstrate the influence of basis functions with compact support on the stiffness matrix we employ the simplest finite element space in the context of our example problem – the linear finite element space in one dimension. Assume a partition of the problem domain $\overline{\Omega} = [0, 1]$ into n non–overlapping intervals $[t_i, t_{i+1}]$ with $\{t_0 = 0, t_1, \ldots, t_n = 1\}$. The intervals are not required to be of equal length $h_i = t_{i+1} - t_i$. As mentioned before, the h in the definition of the Galerkin finite–dimensional spaces refers to the largest element in the partition. The points t_i are often called the *nodes* of a finite element discretization whereas the intervals $[t_i, t_{i+1}]$ are referred to as *(finite) elements*. Now consider the following definition of shape functions

$$N_i(t) = \begin{cases} \dfrac{t - t_{i-1}}{h_{i-1}} & t_{i-1} \le t \le t_i \\ \dfrac{t_{i+1} - t}{h_i} & t_i \le t \le t_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{3.34}$$

For the boundary nodes t_0 and t_n, analogous definitions of N_0 and N_n restricted to the domain of interest $[0, 1]$ apply (see Figure 3.3). As a special property of shape functions it is worthwhile to note, that a shape function N_i takes on the value 1 at node t_i and is zero at all other nodes.

A typical member v^h of the Galerkin finite–dimensional space of variations V^h according to Equation 3.27 is depicted in Figure 3.4. Corresponding to the linear character of the shape functions, v^h is continuous but has discontinuous slopes at the element boundaries. As we have seen in Chapter 3.1.2, the generalized derivative of v^h is a piecewise constant (see also Figure 3.1).

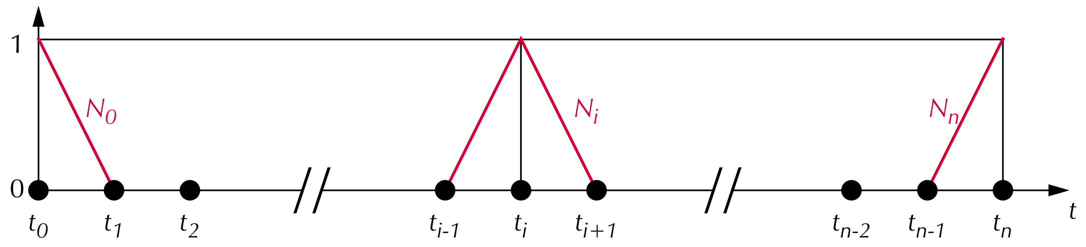


FIGURE 3.3 Shape functions for the one–dimensional piecewise linear finite element space.

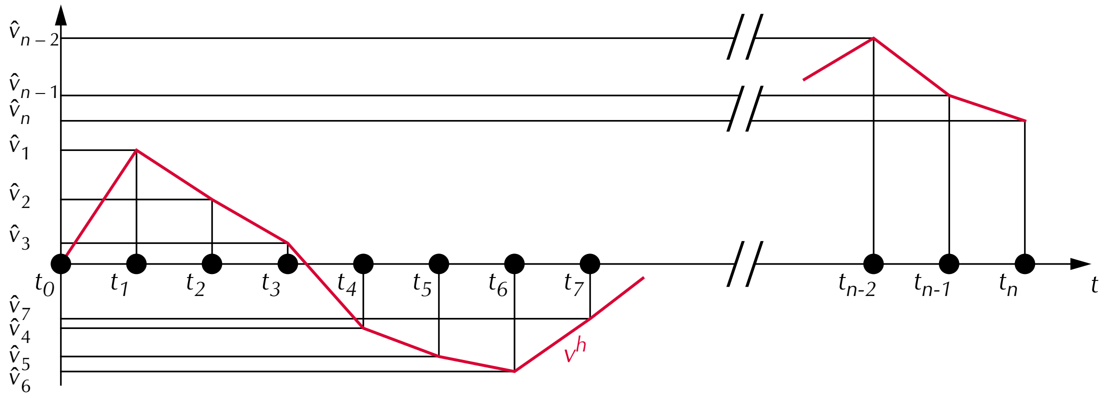


FIGURE 3.4 A typical member of V^h as a weighted sum of piecewise linear shape functions.

For our example problem, typical solutions u^h in U^h are found by adding $b^h = dN_0$ to any member of V^h thus ensuring the essential boundary condition $u(0) = d$.

Having this knowledge about the nature of shape functions we can now observe additional properties of the stiffness matrix.

Firstly, we can state a very important property of the stiffness matrix with respect to the applicability of efficient solving schemes.

Definition. A $n \times n$ matrix **A** is **positive definite** if

(a) $\mathbf{c}^T\mathbf{A}\mathbf{c} \geq 0$ for all *n*–vectors **c**.
(b) $\mathbf{c}^T\mathbf{A}\mathbf{c} = 0$ implies $\mathbf{c} = 0$.

Theorem. The stiffness matrix **K** (Equation 3.33) is positive definite.

The proof of this theorem follows from the definition of the entries in the stiffness matrix, the properties of $a(\cdot, \cdot)$ and the definition of the space of variations V^h:

(a) Let **c** be a vector having arbitrary entries c_i, $i = 1, \ldots, n$. Using the c_i we can construct a member of V^h as $v^h = \sum_{i=1}^{n} c_i N_i$ (Equation 3.27). Then

$$\begin{aligned}
\mathbf{c}^T\mathbf{K}\mathbf{c} &= \sum_{i,j=1}^{n} c_i K_{ij} c_j \\
&= \sum_{i,j=1}^{n} c_i a(N_i, N_j) c_j && |\text{ definition of } K_{ij} \\
&= a\Big(\sum_{i=1}^{n} c_i N_i, \sum_{j=1}^{n} c_j N_j\Big) && |\text{ bilinearity of } a(\cdot, \cdot) \\
&= a(v^h, v^h) && |\text{ definition of } v^h \\
&= \int_0^1 \left(\frac{dv^h}{dt}\right)^2 dt && |\text{ definition of } a(\cdot, \cdot) \\
&\geq 0
\end{aligned}$$

(b) Assume $\mathbf{c}^T\mathbf{K}\mathbf{c} = 0$. Then by the proof of (a)

$$\int_0^1 \left(\frac{dv^h}{dt}\right)^2 dt = 0$$

which implies that v^h must be constant. Since members of V^h are zero at essential boundary conditions, we conclude that $v^h(t) = 0$ for $t \in [0, 1]$ which is true only for $c_i = 0, i = 1, \ldots, n$, as required.

A conclusion of the positive definiteness of **K** is the existence of a unique inverse. Further, positive definite symmetric matrices lend themselves well for conjugate gradient solving schemes, which are known to be both efficient and memory saving [56].

A second important property of the stiffness matrix of a one–dimensional problem is its bandedness. This follows directly from the compact support of the shape functions, since $K_{ij} = 0$ if $i > j + 1$ or $j > i + 1$. In other words, in addition to the entries K_{ii} only entries of the stiffness matrix which correspond to neighboring nodes are not equal to zero. In the case of a piecewise linear approach for the one–dimensional example problem, this implies the stiffness matrix to be tri–diagonal. In multi–dimensional settings, the compact support of basis functions does not necessarily imply bandedness but only sparsity of the stiffness matrix. Most often, the resulting matrix is diagonally dominant, but this heavily depends on a smart numbering of nodes, a problem which is not trivial to solve in more than one dimension.

3.3.3 Global versus Local Point of View

So far we have viewed the finite element method simply as a particular Galerkin approximation procedure. Particular mainly in that the selected basis functions feature both piecewise smoothness and local support. This way of view can be called the *mathematical* or *global point of view*, since the basis functions are regarded as being defined everywhere on the domain of the boundary–value problem. This mathematical point of view is most useful in deriving the mathematical properties and foundations of the finite element method.

In terms of finite element procedures there is a second, local viewpoint which must be regarded the traditional engineering view. This *local* or *element point of view* is useful in the computer implementation of the finite element method and in the development of finite elements.

From an element point of view, one often thinks of the shape functions as being restricted to the element. Therefore, in Figure 3.3, on each interval or element there are two shape functions defined, one of which takes on the value 1 at the beginning of the interval and decreases to 0, whereas the second increases from 0 to 1 within the interval. Further, an element is defined by the nodal weights to the shape functions as well as its part of the domain, the element subdomain. In addition, it is most useful to define both the domain and the shape functions in terms of a local coordinate system, which will be used in the evaluation of the integrals restricted to the element under consideration. We now have to distinguish between the global stiffness matrix and force vector and their local counterparts.

The important observation to make is that **K** and **F** can be constructed by summing up the contributions of element matrices and vectors, respectively. This procedure is referred to as the *assembly* of the global stiffness matrix and is also termed the *direct stiffness method* [68]. As a consequence of the element formulation in a local coordinate system which most often is of unit length, the "local" integrations involve a transformation from the local to the global coordinate system.

The above remarks conclude the first part of this chapter. In the subsequent sections, we will propose a finite element formulation for static elastomechanics which will be employed in the simulation of soft tissue [119]. The notions and concepts introduced by means of the one–dimensional example problem will find their equivalents in the three–dimensional setting for an arbitrary elastic solid. In Chapter 5 we will take a closer look at both the computation of local stiffness matrices and force vectors and the assembly into their global counterparts for C^0 and C^1 tetrahedral elements. In addition to the direct stiffness method applicable to the C^0 assembly, we will introduce an assembly procedure which is needed in our approach to guarantee higher order C^1 continuity across element boundaries.

3.4 STATIC ELASTOMECHANICS FOR SOFT TISSUE MODELING

In our approach to soft tissue simulation we think of tissue as being elastic and therefore base on the theory of *static elastomechanics*. In the following sections, we will revisit the finite element formulations of the partial differential equations specific to static elastomechanics. We restrict ourselves to the isotropic case, i.e. the material properties are independent of direction, and the elastic behavior is characterized by only two independent parameters, elasticity and compressibility, as opposed to 21 parameters in the anisotropic case. We will focus on a pure linear setting, an incompressible setting as well as on a setting which, to a certain extent, is nonlinear.

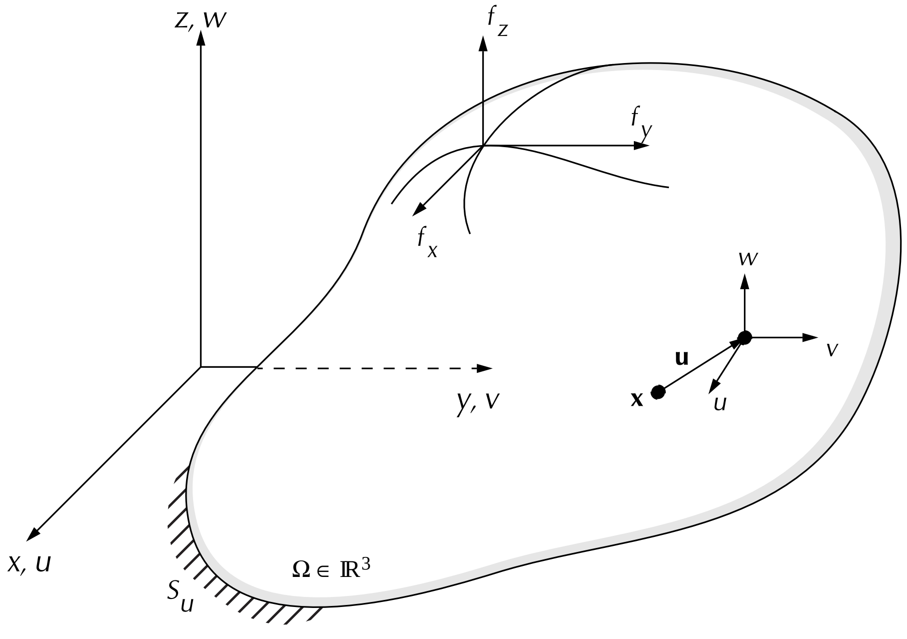


FIGURE 3.5 General three–dimensional elastic body.

In order to derive the differential equations, we follow the *variational approach*, starting out from the total potential Π of the system – the *functional* of the problem – and then invoke the *stationarity* of Π, i.e. $\delta\Pi = 0$ (see Section 3.2.2). As mentioned before, this is equivalent to finding the configuration of minimal potential energy. In elastomechanics the potential Π can be regarded as the elastic energy of the body, minus the work done by externally applied forces.

In the following sections, we will first introduce the weak problem statement for linear elasticity. We will see that the case of total incompressibility cannot be handled properly and thus derive the mixed formulation which enables the simulation of totally incompressible materials by the introduction of pressure as an additional variable. Last but not least, we will sketch an iterative approach for the simulation of a nonlinear response of elastic materials.

3.4.1 Linear Elasticity

We think of an elastic body as the domain $\Omega \in \mathbb{R}^3$ in the fixed coordinate system *x, y, z* (see Figure 3.5). The surface of the body is partly supported on the area S_u with prescribed displacements $\mathbf{u}^{S_u}$ as essential boundary conditions. In addition, the body is subjected to externally applied forces $\mathbf{f}(x, y, z) = [f_x(x, y, z), f_y(x, y, z), f_z(x, y, z)]^T$. The forces together with the prescribed displacements yield a displacement field within the body which we denote by

$$\mathbf{u}(x, y, z) = \begin{bmatrix} u(x, y, z) \\ v(x, y, z) \\ w(x, y, z) \end{bmatrix}$$

with $\mathbf{u} = \mathbf{u}^{S_u}$ on S_u. The displacement field itself yields corresponding *strains* ε [8, 147]

$$\varepsilon = \begin{bmatrix} \varepsilon_{xx} \; \varepsilon_{yy} \; \varepsilon_{zz} \; \gamma_{xy} \; \gamma_{yz} \; \gamma_{zx} \end{bmatrix}^T \tag{3.35}$$

where

$$\begin{array}{lll} \varepsilon_{xx} = \dfrac{\partial u}{\partial x}, & \varepsilon_{yy} = \dfrac{\partial v}{\partial y}, & \varepsilon_{zz} = \dfrac{\partial w}{\partial z} \\ \gamma_{xy} = \dfrac{\partial u}{\partial y} + \dfrac{\partial v}{\partial x}, & \gamma_{yz} = \dfrac{\partial v}{\partial z} + \dfrac{\partial w}{\partial y}, & \gamma_{zx} = \dfrac{\partial w}{\partial x} + \dfrac{\partial u}{\partial z} \end{array} \tag{3.36}$$

with ε_{xx}, ε_{yy}, and ε_{zz} as *volumetric strain* components and γ_{xy}, γ_{yz}, and γ_{zx} as *deviatoric* or *shear strain* components. Equation 3.35 represents the *engineering strains* as a "vector notation" of the symmetric strain tensor $\tilde{\varepsilon}$ which is a tensor of rank 2. The strain tensor's entries ε_{ij} are defined as

$$\varepsilon_{ij} = u_{(i,j)} \overset{\text{def.}}{=} \frac{u_{i,j} + u_{j,i}}{2} \tag{3.37}$$

with u_i denoting the *i*–th coordinate axis ($u_1 \equiv u$, $u_2 \equiv v$, $u_3 \equiv w$) and a comma denoting differentiation with respect to the corresponding axis. The deviatoric components of Equation 3.35 result from adding the symmetric shear strain components, e.g. $\gamma_{xy} = \varepsilon_{12} + \varepsilon_{21}$, whereas the volumetric components correspond directly to the diagonal entries, e.g. $\varepsilon_{xx} = \varepsilon_{11}$.

The state of *stress* τ at an arbitrary point within Ω can be described correspondingly as

$$\tau = \begin{bmatrix}\tau_{xx} & \tau_{yy} & \tau_{zz} & \tau_{xy} & \tau_{yz} & \tau_{zx}\end{bmatrix}^T .$$

In analogy to the strain components, τ_{xx}, τ_{yy}, and τ_{zz} describe the *normal stresses* in the coordinate directions whereas τ_{xy}, τ_{yz}, and τ_{zx} denote *shear stresses*. As before, the components of τ are referred to as *engineering stresses* and are related to the entries of the symmetric stress tensor which we denote by $\tilde{\tau}$.

In order to find the relationship between stress and strain, it is illustrative to revisit *Hooke's law* in one dimension. According to Hooke's law, there is a linear relationship between stress and strain which is described as

$$\tau = E\varepsilon \tag{3.38}$$

with E as *Young's modulus*. E depends on the elastic material under consideration and, in a linear analysis, may vary only as a function of the one–dimensional space parameter t but is constant otherwise (i.e., E does not depend on the stress state). Further, Hooke's law only holds for small displacements. For large displacements and corresponding large strains there is a nonlinear relationship between stress and strain.

For three–dimensional and *isotropic* bodies which do not feature direction depend elastic properties we can find linear relationships between the components of stress and strain using the *generalized Hooke's law* [115, 125, 147]:

$$\begin{aligned}
\varepsilon_{xx} &= \frac{1}{E}(\tau_{xx} - \nu\tau_{yy} - \nu\tau_{zz}) & \gamma_{xy} &= \frac{2(1+\nu)}{E}\tau_{xy} \\
\varepsilon_{yy} &= \frac{1}{E}(-\nu\tau_{xx} + \tau_{yy} - \nu\tau_{zz}) & \gamma_{yz} &= \frac{2(1+\nu)}{E}\tau_{yz} \\
\varepsilon_{zz} &= \frac{1}{E}(-\nu\tau_{xx} - \nu\tau_{yy} + \tau_{zz}) & \gamma_{zx} &= \frac{2(1+\nu)}{E}\tau_{zx}
\end{aligned} \tag{3.39}$$

Again, E denotes *Young's modulus* whereas ν is referred to as *Poisson's ratio*. Solving the linear relations in Equation 3.39 for the stress components yields the matrix equation

$$\tau = \mathbf{C}\varepsilon . \tag{3.40}$$

This equation is often referred to as the *constitutive relation* between stress and strain, where $\mathbf{C}$ denotes the stress–strain material matrix

$$\mathbf{C} = \frac{E}{(1+\nu)(1-2\nu)}\begin{bmatrix}
1-\nu & \nu & \nu & & & \\
\nu & 1-\nu & \nu & & \mathbf{0} & \\
\nu & \nu & 1-\nu & & & \\
 & & & \frac{1}{2}-\nu & & \\
 & \mathbf{0} & & & \frac{1}{2}-\nu & \\
 & & & & & \frac{1}{2}-\nu
\end{bmatrix} . \tag{3.41}$$

The magnitude of E in Equation 3.41 controls the stiffness of the material whereas $0 \leq \nu \leq 0.5$ measures its incompressibility. For $\nu = 0.5$ we obtain total incompressibility which obviously cannot be handled by Equation 3.41. The corresponding reformulation of the problem will be discussed in Section 3.4.2.

Again, for a linear analysis to be feasible, we require **C** to vary only as a function of *x*, *y* and *z*, but to be constant with respect to the stress state. For *homogeneous materials*, E and ν are constant throughout the body. In Section 3.4.3, we will investigate an iterative formulation which allows for the simulation of materials which harden with increasing stress.

It is important to note that the symmetry of **C** in Equation 3.41 assures the symmetry of the stiffness matrix which will result from discretization and subsequent matrix formulation of the problem. Further, Equation 3.40 can be written in tensor notation as

$$\tau_{ij} = c_{ijkl}\varepsilon_{kl} \tag{3.42}$$

where ε_{kl} is defined as in Equation 3.37 and the entries of the 4–th rank tensor **c** are defined in accordance with Equation 3.41 as

$$c_{ijkl} = \lambda\delta_{ij}\delta_{kl} + \mu(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}) \tag{3.43}$$

where δ_{ij} is the *Kronecker delta* and λ and μ are the *Lamé constants*

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)} \qquad \mu = \frac{E}{2(1+\nu)}. \tag{3.44}$$

The equivalence of Equation 3.40 and Equation 3.42 can easily be seen by means of Equation 3.41, Equation 3.43, and Equation 3.44. It is worthwhile to note that in contrast to the strain tensor, all entries of the symmetric stress tensor correspond directly to the entries of the vector of engineering stress, i.e. $\tau_{xx} = \tau_{11}$ and $\tau_{xy} = \tau_{12} = \tau_{21}$. This is pointed out by using the same symbol τ for both the normal and shearing components.

In essence, the problem can now be formulated as follows: given the geometry of the body, the loads **f**, the boundary or support conditions $\mathbf{u}^{S_u}$, and the stress–strain relation of the material, calculate the displacement field $\mathbf{u}(x, y, z)$ within the body.

It is most intuitive to follow the variational approach in order to solve the problem. Then, the solution is to be found as the configuration **u** which minimizes the potential energy of the system. For linearly elastic materials we obtain the potential Π as the difference of the elastic energy within the body and the work done by externally applied forces:

$$\Pi = \frac{1}{2}\int_V \varepsilon^T \tau dV - \int_V \mathbf{u}^T \mathbf{f} dV \tag{3.45}$$

As we have seen in Section 3.2.2, the *principle of virtual work*, or in terms of the finite element method the *principle of virtual displacements,* states that the equilibrium of the body requires the equivalence of the total internal and external virtual work done by any imposed small virtual displacement satisfying the boundary conditions. Consequently, we find an equilibrium condition by invoking the stationarity of Equation 3.45, $\delta\Pi = 0$, which yields [8]

$$\int_V \bar{\varepsilon}^T \tau \, dV = \int_V \bar{\mathbf{u}}^T \mathbf{f} \, dV . \tag{3.46}$$

The left hand side of the above equation represents the internal virtual work whereas the right hand side is the external virtual work. The overbars on $\mathbf{u}$ and ε denote virtual displacements and corresponding virtual strains, respectively.

The equilibrium Equation 3.46 is the weak formulation of the problem for linear elasticity. The correspondence to the weak formulation for the one–dimensional example problem in Section 3.2.2 can be seen if we state the problem formally as

$$(\mathrm{W}) \left\langle \begin{array}{l} \text{Given } \mathbf{f}:\overline{\Omega} \rightarrow \mathbb{R}^3 \text{, find } \mathbf{u} \in U^3 \text{ with} \\ \int\limits_V \bar{\varepsilon}^T \tau \, dV = \int\limits_V \bar{\mathbf{u}}^T \mathbf{f} \, dV \\ \text{for all } \bar{\mathbf{u}} \in V^3 . \end{array} \right. \tag{3.47}$$

where U^3 refers to the space of trial solutions in three–dimensional space which meet the essential boundary conditions $\mathbf{u} = \mathbf{u}^{S_u}$ on S_u (see Figure 3.5) and V^3 refers to the space of variations satisfying $\mathbf{v} = 0$ on S_u. In Equation 3.47 we omitted the term for point loads corresponding to the tip force in the example problem. This is due to the fact that the definition of point loads is not in correspondence with real physical situations where a point load would result in infinite strain [136].

Similar to the considerations at the end of Section 3.2.3, Equation 3.46 gives insight into the class of admissible functions for the problem of linear elasticity: due to the weak formulation, only the first order derivative terms hidden in the material strain ε appear in the equilibrium condition and therefore polynomial functions in $U^3 = H^1(\overline{\Omega})^3$ are admissible solution candidates. Their first derivatives have at worst a jump at element boundaries, and thus their energy remains finite, as required. As a consequence, C^0 finite element shape functions are admissible for the discretization of linear elasticity [77, 125, 136]. Nevertheless, in Chapter 5 we will motivate the use of both C^0 and C^1 tetrahedral finite elements and give a comparison of their respective performance.

3.4.2 Incompressibility

It is reasonable to think of human tissue as being almost incompressible because of its high water content. For the analysis of such media the pure displacement–based approach described in the previous section is not sufficient. We therefore employ a so–called *mixed formulation* which is far more efficient and can be thought of as a special case of the *Hu–Washizu* variational principle (see [8, 152]).

In the analysis of nearly incompressible materials the pressure proves difficult to predict accurately. Depending on how close the material is to being totally incompressible, the displacement–based finite element method may still provide an accurate solution, albeit at the cost of a very fine discretization compared to a similar problem involving a compressible material. A basic observation in nearly or totally incompressible simulations using a pure displacement formulation is the so–called *locking* effect. This term refers to experiences in the analysis of plates, shells and beams where such a formulation yields displacements much smaller than those intuitively expected [8]. The locking of the pure displacement formulation with Poisson's ratio ν approaching 0.5 makes it desirable to have a formulation which yields the same accuracy for a given mesh, irrespective of the material properties.

The key idea of the mixed formulation is to introduce pressure as an additional variable. Assuming almost incompressible behavior, we can think of the volumetric strains being small compared to the deviatoric strains, and therefore reformulate the constitutive relation using tensor notation in the form

$$\tau_{ij} = \kappa \varepsilon_V \delta_{ij} + 2G\varepsilon'_{ij} \tag{3.48}$$

where δ_{ij} denotes the Kronecker delta, κ is the *bulk modulus*

$$\kappa = \frac{E}{3(1-2\nu)},$$

and G is the *shear modulus*

$$G = \frac{E}{2(1+\nu)}.$$

We thereby separate the volumetric strain ε_V

$$\varepsilon_V = \varepsilon_{kk} = \varepsilon_{xx} + \varepsilon_{yy} + \varepsilon_{zz} \approx \frac{\Delta V}{V}, \tag{3.49}$$

and the deviatoric strain components ε'_{ij}

$$\varepsilon'_{ij} = \varepsilon_{ij} - \frac{\varepsilon_V}{3}\delta_{ij}. \tag{3.50}$$

For the pressure in the body we have

$$p = -\kappa\varepsilon_V \tag{3.51}$$

where

$$p = -\frac{\tau_{kk}}{3} = -\frac{\tau_{xx} + \tau_{yy} + \tau_{zz}}{3}.$$

By gradually increasing κ (which means that the Poisson ratio ν approaches 0.5) the volumetric strain drops to zero. Using Equation 3.48 and Equation 3.51, the stress components become

$$\tau_{ij} = -p\delta_{ij} + 2G\varepsilon'_{ij}. \tag{3.52}$$

For totally incompressible media, the displacement boundary conditions $\mathbf{u}^{S_u}$ must be compatible with zero volumetric strain throughout the body. Further, in order to compute a unique solution, the pressure must be defined at some point in the body.

We consequently have to work with the unknown displacements $\mathbf{u}$ and the pressure p as solution variables when using a mixed formulation. In vector notation, the principle of virtual work consequently converts to

$$\int_V \bar{\varepsilon}'^T \tau' \, dV - \int_V \bar{\varepsilon}_V p \, dV = \int_V \bar{\mathbf{u}}^T \mathbf{f} \, dV \tag{3.53}$$

with the deviatoric stress vector τ'

$$\tau' = \tau + p\delta$$

and the deviatoric strain vector ε'

$$\varepsilon' = \varepsilon - \frac{1}{3}\varepsilon_V\delta$$

where δ is a vector of the Kronecker delta symbol $\delta = [\ 1\ 1\ 1\ 0\ 0\ 0\ ]^T$.

In Equation 3.53 we have separated the volumetric and deviatoric strain energies. The connection between the independent variables p and **u** is provided by Equation 3.51 written in integral form

$$\int_V \left(\frac{p}{\kappa} + \varepsilon_V\right)\bar{p}\,dV = 0 \tag{3.54}$$

which states that every change in volume will be compensated by a corresponding change in pressure. For the case of total incompressibility, i.e. $\kappa \to \infty$, Equation 3.54 consequently becomes

$$\int_V \varepsilon_V \bar{p}\,dV = 0 \tag{3.55}$$

which allows no change in volume.

3.4.3 Nonlinear Extensions

The above–mentioned procedures apply only to linear problems where the response **u** is a linear function of the applied loads. This is a consequence of the following three assumptions:

- *Small displacements/small strains*
  The displacements **u** and the resulting strains must be small because all integrations are performed over the original volume. Further, in Equation 3.49, we approximated the volumetric strain linearly which ignores the geometrically nonlinear terms in the exact formula for the volume strain (see Figure 3.6):

$$\varepsilon_V = \varepsilon_{xx} + \varepsilon_{yy} + \varepsilon_{zz} + \overbrace{\varepsilon_{xx}\varepsilon_{yy} + \varepsilon_{yy}\varepsilon_{zz} + \varepsilon_{zz}\varepsilon_{xx} + \varepsilon_{xx}\varepsilon_{yy}\varepsilon_{zz}}^{\text{nonlinear terms}} = \frac{\Delta V}{V}$$

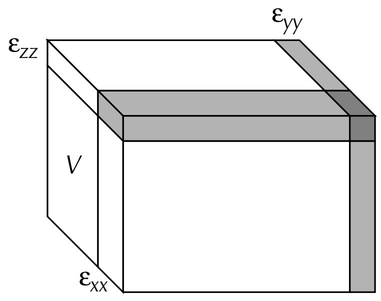


FIGURE 3.6 Shaded parts are ignored in a linear approximation of volumetric changes.

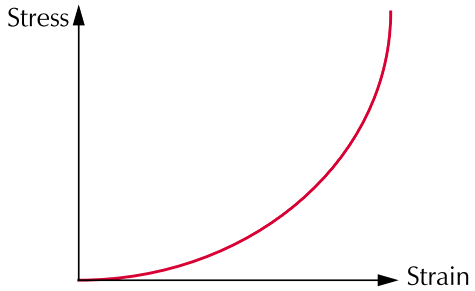


FIGURE 3.7 Nonlinear stress–strain relationship.

- *Linearly elastic material*
  The constitutive relations derived above only reflect a linear relationship between stress and strain.
- *Boundary conditions remain unchanged*

In order to solve problems where at least one of these conditions is not met, we have to elaborate on nonlinear schemes, which are still subject to extensive research activities in applied mechanics and material science.

Biomechanical studies have shown a highly nonlinear elastic response and hardening effects of facial tissue [18, 48]. We restrict ourselves to the static analysis of a nonlinear stress–strain relationship, still assuming small displacements and small strains (see Figure 3.7). This enables us to go on working with the engineering strains defined by Equation 3.35 and Equation 3.36 instead of using a total or updated Lagrangian formulations together with appropriate strain measures and the corresponding stress tensors (see [8, 35]).

In order to derive the formulation of incremental nonlinear analysis we restate the principle of virtual work in Equation 3.46. Using indicial notation and writing virtual displacements and corresponding strains as variations δu_i and $\delta\varepsilon_{ij}$, we have for the equilibrium condition of the body at time $t + \Delta t$

$$\int_V {}^{t+\Delta t}\tau_{ij}\delta\varepsilon_{ij}dV = \int_V {}^{t+\Delta t}f_i\delta u_i dV. \tag{3.56}$$

In a static time independent analysis, the step–by–step incremental solution reduces to a one–step analysis if the total load is applied at once. However, for computational reasons, the analysis of such problems frequently requires an incremental solution where the variable t denotes a number of loading steps to reach the total applied load.

As

$$ {}^{t+\Delta t}\tau_{ij} = {}^{t+\Delta t}c_{ijkl}\ {}^{t+\Delta t}\varepsilon_{kl} $$

we are left with the problem of defining the entries of the stress–strain material matrix **C**, ${}^{t+\Delta t}c_{ijkl}$. In order to achieve the desired nonlinear behavior of increasing stiffness with increasing strain we state Young's modulus as a function of the strain

$$E = E_0 e^{A\varepsilon^2} \tag{3.57}$$

with E_0 denoting Young's modulus for the relaxed material and A serving as a measure of nonlinearity. For vanishing strains and for $A = 0$, Equation 3.57 is equivalent to the linear case.

Note, that Equation 3.57 relates to one–dimensional problems, that is, we need a scalar ε representing the actual strain. To this aim we define ε in Equation 3.57 either using the Frobenius norm ε_F of the strain tensor $\tilde{\varepsilon}$ (Equation 3.37) or the arithmetic mean ε_A of the absolute values of the tensor's entries:

$$\varepsilon_F = \sqrt{\sum_{i,j} \varepsilon_{ij}^2} \qquad \varepsilon_A = \frac{1}{9}\sum_{i,j} |\varepsilon_{ij}|$$

Note furthermore, that volumetric formulations of Equation 3.57 are extremely difficult and open research issues in biomechanics [18].

Due to its nonlinearity, Equation 3.56 will have to be solved approximately by referring all variables to a previously calculated known equilibrium configuration and subsequent linearization of the resulting equation. From Equation 3.56, this results in

$$\int_V {}^t c_{ijkl} \varepsilon_{kl} \delta\varepsilon_{ij} dV = \int_V {}^{t+\Delta t} f_i \delta u_i dV - \int_V {}^t \tau_{ij} \delta\varepsilon_{ij} dV . \tag{3.58}$$

The integral term $\int_V {}^t\tau_{ij}\delta\varepsilon_{ij} dV$ denotes the *internal virtual work* which results from the actual physical stresses ${}^t\tau_{ij}$ at time *t*. Assuming that the approximate displacements and corresponding strains and stresses at time $t + \Delta t$ have been calculated according to Equation 3.58, we can now define the error due to linearization as the *out–of–balance virtual work*,

$$\text{Error} = \int_V {}^{t+\Delta t} f_i \delta u_i dV - \int_V {}^{t+\Delta t} \tau_{ij} \delta\varepsilon_{ij} dV . \tag{3.59}$$

Therefore, in general, an iteration of Equation 3.58 will be necessary to minimize the error given by Equation 3.59.

3.5 FINITE ELEMENT DISCRETIZATION

As mentioned at the beginning of the chapter, the discretization in FEM consists of two main aspects: Firstly, the domain of interest is subdivided into a finite and disjoint set of primitives, so–called *finite elements*, and secondly, we expand the solution within an element as a weighted sum of basis or *shape functions.*

3.5.1 Subdivision of Domain into Finite Elements

The task of mesh generation can be formulated as follows: Given a domain, the objective is to partition it into simple elements meeting in well–defined ways. On the one hand, there should be as few as possible elements since an increasing number of elements increases the number of degrees of freedoms and thus the size of the resulting system of equations. On the other hand, some portions of the domain may need smaller elements in order that the computation is more accurate there. Further, all elements should be *well shaped* which puts restrictions on the angles and aspect ratios of the elements. One distinguishes between *structured* and *unstructured* meshes by the way the elements meet; a structured mesh is one in which the elements have the topology of a regular grid. Structured meshes are typically easier worked upon in that they allow for simpler implementation and meshing techniques, but they in general require more or worse–shaped elements. Unstructured meshes are often computed by means of Delaunay triangulation of point sets; however there are quite varied approaches for selecting the points to be triangulated.

Considerable theoretical work has been done on these problems, complementing and building on practical work in numerical analysis. Theoretically, the preferred type of mesh is the simplex mesh, i.e. in two or three dimensions the triangulation or tetrahedralization, respectively. Tetrahedral elements offer the most in flexibility both geometrically and topologically which is of great importance for the application in tissue simulation for human faces. Unfortunately, in practice, quadrilaterals or higher dimensional cubical elements are easier to handle because they allow for regular or structured meshes and in addition tentatively have better numerical properties. Nevertheless, we in this work represent tissue as a tetrahedral mesh (see Section 7.3).

The core research aspects of finite element meshing are beyond the scope of this thesis. For a survey of meshing and further references the reader is referred to [9, 146]. Remaining problem areas in the theory of meshing include triangulations in dimensions higher than two, meshes with cubical elements, mesh smoothing, mesh decimation and multigrid methods, as well as data structures for the efficient implementation of meshing algorithms.

3.5.2 Galerkin Projection

Complementary to the more formal treatment of Galerkin projection in Section 3.3.1, we in the following treat the subject from the element point of view, which is more closely related to the actual situation in implementing finite element procedures. To this aim, we similar to Equation 3.27 expand the solution in a finite–dimensional subspace within an element m as a weighted sum of n basis or shape functions N_i:

$$\tilde{\mathbf{u}}^{(m)}(x, y, z) = \sum_{i=0}^{n-1} \hat{\mathbf{u}}_i N_i(x, y, z)$$

In contrast to the one–dimensional setting in the introductional part of this chapter, the weights $\hat{\mathbf{u}}_i$ are now three–dimensional vectors. We can therefore state the displacement field within an element m as

$$\mathbf{u}^{(m)}(x, y, z) = \mathbf{H}^{(m)}(x, y, z)\hat{\mathbf{u}}^{(m)} \tag{3.60}$$

where $\mathbf{H}^{(m)}$ denotes the *displacement interpolation matrix* and $\hat{\mathbf{u}}^{(m)}$ contains the $3n$ *nodal weights* to the basis functions. The three spatial components of the displacement field are interpolated according to the structure of $\mathbf{H}^{(m)}$ and $\hat{\mathbf{u}}^{(m)}$:

$$\mathbf{H}^{(m)} = \begin{bmatrix} N_0, \ldots, N_{n-1} & & 0 \\ & N_0, \ldots, N_{n-1} & \\ 0 & & N_0, \ldots, N_{n-1} \end{bmatrix} \in \mathbb{R}^{3 \times 3n}$$

$$\hat{\mathbf{u}}^{(m)} = \left[\hat{u}_{x_0}, \ldots, \hat{u}_{x_{n-1}}, \hat{u}_{y_0}, \ldots, \hat{u}_{y_{n-1}}, \hat{u}_{z_0}, \ldots, \hat{u}_{z_{n-1}}\right]^T \in \mathbb{R}^{3n}$$

Consequently, we have for the interpolation of the element strains

$$\varepsilon^{(m)}(x, y, z) = \mathbf{B}^{(m)}(x, y, z)\hat{\mathbf{u}}^{(m)} \tag{3.61}$$

with the *strain interpolation matrix* $\mathbf{B}^{(m)}$ corresponding to Equation 3.36

$$\mathbf{B}^{(m)} = \begin{bmatrix} \frac{\partial}{\partial x}[N_0, \ldots, N_{n-1}] & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \frac{\partial}{\partial y}[N_0, \ldots, N_{n-1}] & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \frac{\partial}{\partial z}[N_0, \ldots, N_{n-1}] \\ \frac{\partial}{\partial y}[N_0, \ldots, N_{n-1}] & \frac{\partial}{\partial x}[N_0, \ldots, N_{n-1}] & \mathbf{0} \\ \mathbf{0} & \frac{\partial}{\partial z}[N_0, \ldots, N_{n-1}] & \frac{\partial}{\partial y}[N_0, \ldots, N_{n-1}] \\ \frac{\partial}{\partial z}[N_0, \ldots, N_{n-1}] & \mathbf{0} & \frac{\partial}{\partial x}[N_0, \ldots, N_{n-1}] \end{bmatrix} \in \mathbb{R}^{6 \times 3n}. \tag{3.62}$$

Note specifically that $\mathbf{B}^{(m)}$ contains first order derivatives of the shape functions.

3.5.3 Assembly

Using Equation 3.60 and Equation 3.61 for the interpolation of the virtual displacement and virtual strains respectively, we can rewrite the principle of virtual displacements (Equation 3.46) as

$$\bar{\hat{\mathbf{U}}}^T\left[\sum_m \int_{V^{(m)}} \mathbf{B}^{(m)T}\mathbf{C}^{(m)}\mathbf{B}^{(m)}\, dV^{(m)}\right]\hat{\mathbf{U}} = \bar{\hat{\mathbf{U}}}^T\left[\sum_m \int_{V^{(m)}} \mathbf{H}^{(m)T}\mathbf{f}^{(m)}\, dV^{(m)}\right] \tag{3.63}$$

where $\hat{\mathbf{U}}$ and $\bar{\hat{\mathbf{U}}}$ are vectors containing all *3N* nodal weights of an element assemblage with a total of *N* nodes and $\mathbf{H}^{(m)}$, $\mathbf{B}^{(m)}$, and $\mathbf{C}^{(m)}$ are matrices of corresponding size which are zero everywhere except at the places which correspond to the weights of element *m*. The integrations are performed with respect to the element domain $V^{(m)}$.

Corresponding to the remark at the end of Section 3.3.1, we apply the principle of virtual displacement *3N* times using unit virtual displacements for all components of $\bar{\hat{\mathbf{U}}}$ which yields the governing system of linear equations

$$\mathbf{KU} = \mathbf{R} \tag{3.64}$$

where $\mathbf{U} \equiv \hat{\mathbf{U}}$. In Equation 3.64, $\mathbf{K}$ denotes the symmetric positive–definite *global stiffness matrix*

$$\mathbf{K} = \sum\nolimits_m \int_{V^{(m)}} \mathbf{B}^{(m)T}\mathbf{C}^{(m)}\mathbf{B}^{(m)}\, dV^{(m)}, \tag{3.65}$$

$\mathbf{R}$ the *global force vector*

$$\mathbf{R} = \sum\nolimits_m \int_{V^{(m)}} \mathbf{H}^{(m)T}\mathbf{f}^{(m)}\, dV^{(m)}, \tag{3.66}$$

and $\mathbf{U}$ the unknown *displacement vector*. The summation signs in Equation 3.65 and Equation 3.66 can be regarded as representing the *assembly* of the local stiffness matrices $\int_{V^{(m)}} \mathbf{B}^{(m)T}\mathbf{C}^{(m)}\mathbf{B}^{(m)}\, dV^{(m)}$ and the local force vectors $\int_{V^{(m)}} \mathbf{H}^{(m)T}\mathbf{f}^{(m)}\, dV^{(m)}$ into one global matrix and vector respectively.

3.5.4 Discretization of Mixed Formulation

The interpolation of the displacements remains the same in the case of incompressibility, whereas the interpolation of strain again has to be separated into its volumetric and deviatoric part

$$\begin{aligned} \varepsilon_V^{(m)}(x, y, z) &= \mathbf{B}_V^{(m)}(x, y, z)\hat{\mathbf{u}}^{(m)} \\ \varepsilon'^{(m)}(x, y, z) &= \mathbf{B}_D^{(m)}(x, y, z)\hat{\mathbf{u}}^{(m)} \end{aligned} \tag{3.67}$$

The interpolation matrices $\mathbf{B}_V^{(m)}$ and $\mathbf{B}_D^{(m)}$ follow from Equations 3.49, 3.50, 3.61 and 3.62 to

$$\mathbf{B}_V^{(m)} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix} \mathbf{B}^{(m)}$$

$$\mathbf{B}_D^{(m)} = \frac{1}{3} \begin{bmatrix} 2 & -1 & -1 & & & \\ -1 & 2 & -1 & & 0 & \\ -1 & -1 & 2 & & & \\ & & & 3 & & \\ & 0 & & & 3 & \\ & & & & & 3 \end{bmatrix} \mathbf{B}^{(m)}$$

In addition, the pressure is interpolated according to

$$p^{(m)}(x, y, z) = \mathbf{H}_p^{(m)}(x, y, z)\hat{\mathbf{p}}^{(m)} \tag{3.68}$$

where $\mathbf{H}_p^{(m)}$ and $\hat{\mathbf{p}}^{(m)}$ denote the pressure interpolation matrix and corresponding weight vector, respectively:

$$\begin{aligned} \mathbf{H}_p^{(m)} &= \begin{bmatrix} M_0, \ldots, M_{k-1} \end{bmatrix} \in \mathbb{R}^{1 \times k} \\ \hat{\mathbf{p}}^{(m)} &= \begin{bmatrix} \hat{p}_0, \ldots, \hat{p}_{k-1} \end{bmatrix}^T \in \mathbb{R}^k \end{aligned} \tag{3.69}$$

As can be seen from Equation 3.69, the k basis functions $M_0, \ldots, M_{k-1}$ for the interpolation of pressure do not necessarily have to be the same as for the interpolation of the displacements. Moreover, in order to arrive at an efficient finite element solution, the degree of the interpolation functions for displacement and pressure has to be chosen very carefully. When higher order displacement interpolation is employed, the choice of the appropriate pressure interpolation is not obvious and rather difficult. If the pressure is interpolated at too high a degree, the element again would behave like the displacement–based element and *lock* [8]. If too low a degree of interpolation is employed, we will face a poor pressure prediction which will not be very accurate.

Many finite elements have been analyzed theoretically and numerically. The *inf–sup condition* gives a basic mathematical criterion which determines whether a mixed finite element discretization is stable and convergent. The condition was introduced by Babuska and Brezzi. For a short introduction and further references, the reader is referred to [8]. In summary, it can be stated that the number of displacement degrees of freedom must always exceed the number of pressure degrees of freedom. Therefore, pressure is to be interpolated at lower a degree than the displacements [160].

Similar to Equation 3.63, we can rewrite Equation 3.53 using the interpolations in Equations 3.67 and 3.68 which yields

$$\bar{\hat{\mathbf{U}}}^T\left[\sum_m \int_{V^{(m)}} \mathbf{B}_D^{(m)T}\mathbf{C}'^{(m)}\mathbf{B}_D^{(m)}\,dV^{(m)}\right]\hat{\mathbf{U}} - \bar{\hat{\mathbf{U}}}^T\left[\sum_m \int_{V^{(m)}} \mathbf{B}_V^{(m)T}\mathbf{H}_p^{(m)}\,dV^{(m)}\right]\hat{\mathbf{P}} = \bar{\hat{\mathbf{U}}}^T\left[\sum_m \int_{V^{(m)}} \mathbf{H}^{(m)T}\mathbf{f}^{(m)}\,dV^{(m)}\right] \tag{3.70}$$

where $\mathbf{C}'^{(m)}$ denotes the deviatoric stress–strain relational matrix. From Equation 3.52 and taking into account that in the vector notation of Equation 3.70 the engineering deviatoric strains equal twice the tensorial deviatoric strains, $\mathbf{C}'^{(m)}$ evaluates to

$$\mathbf{C}'^{(m)} = \begin{bmatrix} 2G & & & & & \\ & 2G & & & 0 & \\ & & 2G & & & \\ & & & G & & \\ & 0 & & & G & \\ & & & & & G \end{bmatrix}. \tag{3.71}$$

After multiplication with -1, Equation 3.54 is interpolated in a similar manner to Equation 3.70 which results in

$$-\bar{\hat{\mathbf{P}}}^T\left[\sum_m \int_{V^{(m)}} \mathbf{H}_p^{(m)T}\mathbf{B}_V^{(m)}\,dV^{(m)}\right]\hat{\mathbf{U}} - \bar{\hat{\mathbf{P}}}^T\left[\sum_m \int_{V^{(m)}} \mathbf{H}_p^{(m)T}\frac{1}{\kappa}\mathbf{H}_p^{(m)}\,dV^{(m)}\right]\hat{\mathbf{P}} = 0. \tag{3.72}$$

The structure of the resulting system of linear equations consequently becomes

$$\begin{bmatrix} \mathbf{K}_{uu} & \mathbf{K}_{up} \\ \mathbf{K}_{pu} & \mathbf{K}_{pp} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{u}} \\ \hat{\mathbf{p}} \end{bmatrix} = \begin{bmatrix} \mathbf{R} \\ \mathbf{0} \end{bmatrix} \tag{3.73}$$

with the load $\mathbf{R}$ equal to the compressible case (Equation 3.66) and the partial stiffness matrices

$$\begin{aligned} \mathbf{K}_{uu} &= \sum_m \int_{V^{(m)}} \mathbf{B}_D^{(m)T}\mathbf{C}'^{(m)}\mathbf{B}_D^{(m)}\,dV^{(m)} \\ \mathbf{K}_{up} = \mathbf{K}_{pu}^T &= -\sum_m \int_{V^{(m)}} \mathbf{B}_V^{(m)T}\mathbf{H}_p^{(m)}\,dV^{(m)} \\ \mathbf{K}_{pp} &= -\sum_m \int_{V^{(m)}} \mathbf{H}_p^{(m)T}\frac{1}{\kappa}\mathbf{H}_p^{(m)}\,dV^{(m)} \end{aligned} \tag{3.74}$$

In accordance with Equation 3.55, for the case of total incompressibility the lower right submatrix of Equation 3.73 evaluates to zero, $\mathbf{K}_{pp} = \mathbf{0}$, and Equation 3.73 reads

$$\begin{bmatrix} \mathbf{K}_{uu} & \mathbf{K}_{up} \\ \mathbf{K}_{pu} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{U}} \\ \hat{\mathbf{P}} \end{bmatrix} = \begin{bmatrix} \mathbf{R} \\ \mathbf{0} \end{bmatrix} .$$

3.5.5 Discretization of Material–only Nonlinear Simulation

For the incremental step–by–step solution in the nonlinear case we are left with the problem of finding the state of equilibrium of a body at time $t + \Delta t$ assuming a known configuration at time *t*. The equilibrium at time *t* is equivalent to a vanishing out–of–balance virtual work in Equation 3.59 and can be expressed as

$$^{t + \Delta t}\mathbf{R} - {}^{t + \Delta t}\mathbf{F} = \mathbf{0} \tag{3.75}$$

where the vector $^{t + \Delta t}\mathbf{R}$ lists the externally applied forces at time $t + \Delta t$,

$$^{t + \Delta t}\mathbf{R} = \sum_m \int_{V^{(m)}} \mathbf{H}^{(m)T}\, {}^{t + \Delta t}\mathbf{f}^{(m)}\, dV^{(m)} ,$$

and the *stress load vector* $^{t + \Delta t}\mathbf{F}$ lists the nodal point forces that correspond to the element stresses at time $t + \Delta t$,

$$^{t + \Delta t}\mathbf{F} = \sum_m \int_{V^{(m)}} \mathbf{B}^{(m)T}\, {}^{t + \Delta t}\boldsymbol{\tau}^{(m)}\, dV^{(m)} .$$

Since the solution at time *t* is known, we can calculate the increment to the solution after a suitably small time increment Δt using

$$^{t + \Delta t}\mathbf{F} = {}^{t}\mathbf{F} + \mathbf{F} \tag{3.76}$$

where $\mathbf{F}$ is the increment to the stress load vector corresponding to the increment in element displacements and stresses from time *t* to time $t + \Delta t$. This vector can be approximated using a *tangent stiffness matrix* $^{t}\mathbf{K}$ which corresponds to the material conditions at time *t*,

$$\mathbf{F} \approx {}^{t}\mathbf{K}\mathbf{U} \tag{3.77}$$

where $\mathbf{U}$ is an incremental vector to the displacements at time *t* and

$$^{t}\mathbf{K} = \frac{\partial\, {}^{t}\mathbf{F}}{\partial\, {}^{t}\mathbf{U}} .$$

Substituting Equation 3.77 and Equation 3.76 into Equation 3.75 yields the matrix equation corresponding to Equation 3.58

$$^{t}\mathbf{K}\mathbf{U} = {}^{t + \Delta t}\mathbf{R} - {}^{t}\mathbf{F} , \tag{3.78}$$

and by solving for $\mathbf{U}$ we can calculate an approximation to the displacements at time $t + \Delta t$,

$$^{t + \Delta t}\mathbf{U} \approx {}^{t}\mathbf{U} + \mathbf{U} . \tag{3.79}$$

Due to the approximation in Equation 3.77, we cannot use the result of Equation 3.79 for the computation of the strains and stresses at time $t + \Delta t$ and proceed to the next time step. Depending on the size of time steps, such a solution might be very unstable and lead to significant errors. We therefore have to iterate the solution of Equation 3.78 until the condition in Equation 3.75 is met to sufficient accuracy.

The widely used iteration methods in finite element analysis are based on the classical *Newton–Raphson* technique [8]. That is, having calculated an increment to the total displacement vector **U**, we can use this vector instead of the vector known from time t to calculate the stress load vector and the tangent stiffness matrix. This results in the following nested iteration, for $i = 1, 2, 3, \ldots$,

$$\begin{aligned} {}^{t+\Delta t}\mathbf{K}^{(i-1)}\Delta\mathbf{U}^{(i)} &= {}^{t+\Delta t}\mathbf{R} - {}^{t+\Delta t}\mathbf{F}^{(i-1)} \\ {}^{t+\Delta t}\mathbf{U}^{(i)} &= {}^{t+\Delta t}\mathbf{U}^{(i-1)} + \Delta\mathbf{U}^{(i)} \\ {}^{t+\Delta t}\mathbf{F}^{(i)} &= {}^{t+\Delta t}\mathbf{K}^{(i)}\,{}^{t+\Delta t}\mathbf{U}^{(i)} \end{aligned} \tag{3.80}$$

with the initial conditions

$${}^{t+\Delta t}\mathbf{U}^{(0)} = {}^{t}\mathbf{U}, \quad {}^{t+\Delta t}\mathbf{K}^{(0)} = {}^{t}\mathbf{K}, \quad {}^{t+\Delta t}\mathbf{F}^{(0)} = {}^{t}\mathbf{F}.$$

The iteration of Equation 3.80 is continued until the out–of–balance load vector ${}^{t+\Delta t}\mathbf{R} - {}^{t+\Delta t}\mathbf{F}^{(i-1)}$ gets very small and therefore the increment in the nodal point displacements $\Delta\mathbf{U}^{(i)}$ meets a convergence criterion as for instance

$$\frac{\left\|\Delta\mathbf{U}^{(i)}\right\|_2}{\left\|{}^{t+\Delta t}\mathbf{U}^{(i)}\right\|_2} \le \varepsilon .$$

As it is very expensive to evaluate the tangent matrix ${}^{t+\Delta t}\mathbf{K}^{(i-1)}$ at every step of the iteration, it may be more efficient to evaluate a new tangent stiffness matrix only at the beginning of each loading step, a method which in the literature is referred to as *modified Newton–Raphson* iteration [8].

CHAPTER 4

BERNSTEIN POLYNOMIALS AND BÉZIER PATCHES

The exploration of parametric curves and surfaces for the use in CAD (computer aided design) can be regarded as the origin of computer aided geometric design (CAGD). A major breakthrough in CAGD was the theory of Bézier patches. Bézier curves and surfaces were independently developed by Paul de Casteljau at Citroën and Pierre Bézier at Renault. Although de Casteljau's work was slightly earlier than Bézier's, the field now bears Bézier's name since the work of de Casteljau was never published but only laid down in two internal reports at Citroën [31, 32, 12, 13]. In spite of many alternative representations, Bézier representations still play a central role in CAGD since they are numerically the most stable among all polygonal bases currently used in CAD [43, 45]. In addition, the theory of Bézier curves and surfaces gives a very intuitive understanding of many important concepts in CAGD. For a comprehensive introduction to the use of parametric curves and surfaces in CAGD the reader is referred to [43, 67].

The rapidly growing field of CAGD has been dominated by the theory of rectangular patches. However, it is interesting that triangular patches were already considered in the late fifties by de Casteljau. He had then realized that a formulation of Bézier curves in a local or barycentric coordinate system admitted a very elegant generalization to surfaces yielding barycentric triangular patches. Historically, triangular patches were considered before tensor product surfaces which is reflected in the mathematical observation that they are a more natural generalization than tensor product patches [41].

A main attraction of Bernstein–Bézier patches is that they lend themselves easily to a geometric understanding of the underlying mathematical concepts. After an introduction to Bézier curves, we in this chapter will focus mainly on triangular and tetrahedral patches. We will investigate the construction of higher order continuity across element boundaries and address the problem of scattered data interpolation using a Clough–Tocher split which ensures a globally C^1–continuous interpolation.

4.1 INTRODUCTORY CONCEPTS

4.1.1 Local Coordinate Systems

A parametric curve $\mathbf{x}(u)$ can be represented as a mapping of the one–dimensional parameter space u into two– or three–dimensional space

$$\mathbf{x}(u) = \begin{bmatrix} x(u) \\ y(u) \\ z(u) \end{bmatrix}.$$

A surface can be represented analogically as a mapping of the two–dimensional parameter space (u, v) into 2D or 3D

$$\mathbf{x}(u, v) = \begin{bmatrix} x(u, v) \\ y(u, v) \\ z(u, v) \end{bmatrix}.$$

Most often, curves are represented patchwise, in that the parametric domain is partitioned into multiple disjoint segments $[u_i, u_{i+1}]$ with $\{u_0, u_1, \ldots, u_n\}$. Over each interval, the curve is defined patchwise. In a tensor product approach, patchwise surfaces are defined analogously by partitioning both parameter directions u and v into $\{u_0, u_1, \ldots, u_n\}$ and $\{v_0, v_1, \ldots, v_n\}$, respectively. Figure 4.1a gives an example of such a tensor product surface as well as its parametric domain (b).

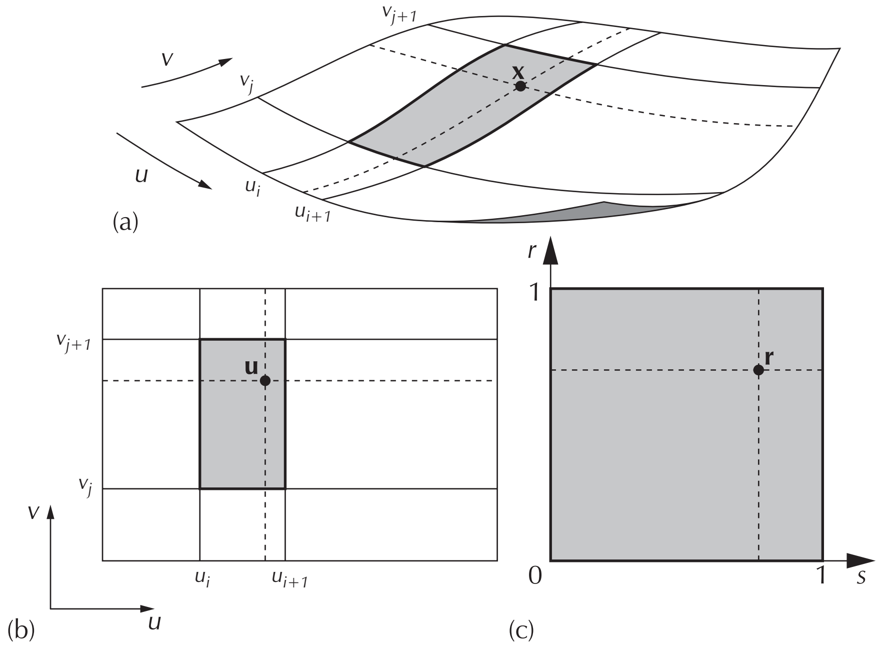


FIGURE 4.1 Patchwise definition of a tensor product surface [20]:
(a) Patchwise surface in three–dimensional space
(b) Corresponding parameter domain (u,v)
(c) Local coordinate system for each patch

For the representation of a curve or surface within a patch, it is convenient to use *local coordinates*. The parameter interval $[u_i, u_{i+1}]$ within a curve segment is described as:

$$t = u_i(1-t) + u_{i+1}t \tag{4.1}$$

which results in a mapping to the unit interval $[0, 1]$. A tensor product patch $u \in [u_i, u_{i+1}], v \in [v_i, v_{i+1}]$ is mapped analogously to the unit square $r, s \in [0, 1]$ by

$$u = u_i(1-r) + u_{i+1}r$$
$$v = v_j(1-s) + v_{j+1}s$$

(see Figure 4.1c).

Due to additional topological and geometric flexibility it is often advantageous to represent surfaces using triangular patches. In order to describe coordinates within a triangular domain we employ *barycentric coordinates*. To this aim, three local coordinates $r, s, t \in [0, 1]$ are introduced which satisfy the additional constraint $r + s + t = 1$. Every point in the parametric domain can now be described as

$$\mathbf{u} = \begin{bmatrix} u \\ v \end{bmatrix} = \mathbf{a}r + \mathbf{b}s + \mathbf{c}t \qquad \mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{R}^2$$

with **a**, **b** and **c** as the parametric corner coordinates of the triangle. Figure 4.2 gives an illustration of the situation.

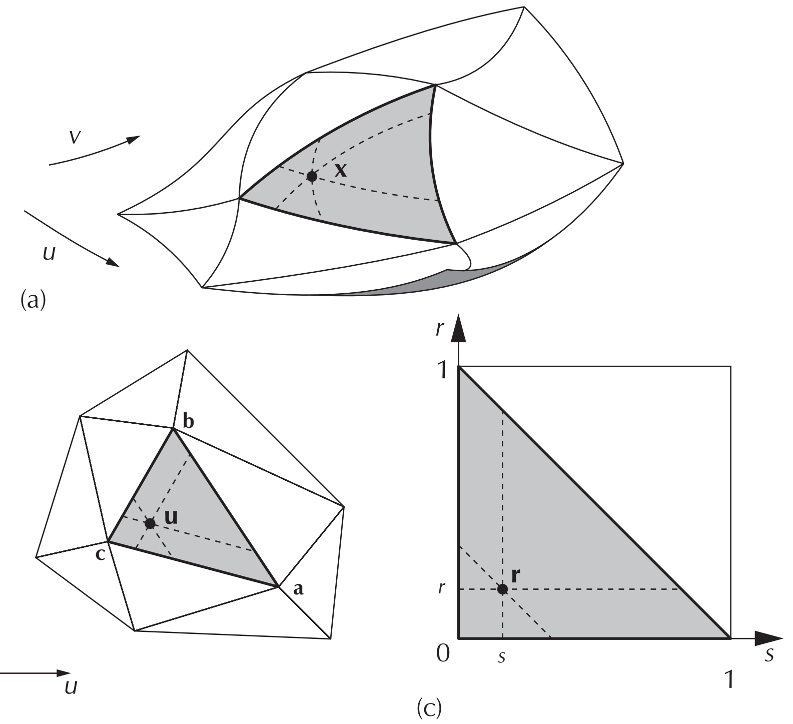


FIGURE 4.2 Patchwise definition of a triangular surface [20]:
(a) Patchwise surface in three–dimensional space
(b) Corresponding parameter domain (u,v); **a**, **b**, **c** define the triangular patch in (u,v)
(c) Barycentric coordinate system for each patch

Barycentric coordinates feature the following properties:

- Every point within the triangle is described by positive coordinates between 0 and 1.
- Barycentric coordinates are invariant to affine transformations. Therefore, the barycentric coordinate system is often sketched as a equilateral triangle (see Figure 4.3)
- The coordinates are proportional to the area of the corresponding sub–triangles:

$$r = \frac{\Delta_{ubc}}{\Delta_{abc}} \qquad s = \frac{\Delta_{uac}}{\Delta_{abc}} \qquad t = \frac{\Delta_{uab}}{\Delta_{abc}} \tag{4.2}$$

with Δ_{abc} as the signed area of the triangle defined by **a**, **b** and **c**.

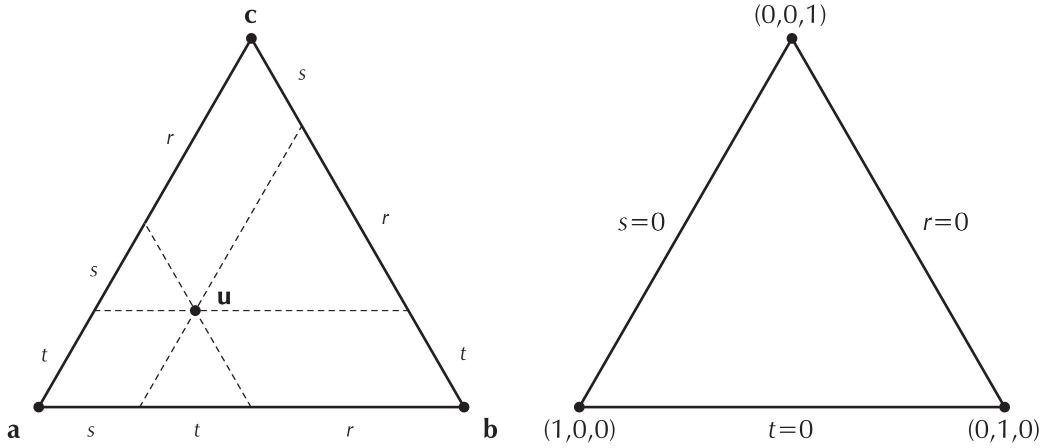


FIGURE 4.3 Barycentric coordinates.

For curves as well as for surfaces, local coordinates define at the same time the linear interpolation between associated points, i.e. for a triangle

$$\mathbf{u}(r, s, t) = \mathbf{t}_0 r + \mathbf{t}_1 s + \mathbf{t}_2 t$$

is the linear interpolation of the triangle vertices $\mathbf{t}_0$, $\mathbf{t}_1$, $\mathbf{t}_2$.

It is equally well possible to develop higher dimensional barycentric coordinates by introducing additional local coordinates. For the tetrahedral case this implies the introduction of an additional local coordinate q. The above–mentioned properties of the coordinates generalize to higher dimensions in an obvious manner, e.g. the areas of sub–triangles correspond to the volumes of sub–tetrahedra in the tetrahedral setting. We will need tetrahedral barycentric coordinates in Section 4.5.

4.1.2 Bernstein Polynomials

Bézier curves are expressed in terms of Bernstein polynomials which were invented by Sergei Bernstein in order to formulate a constructive proof of the Weierstrass approximation theorem [10, 43]. Bernstein polynomials of degree n are defined by

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i} \tag{4.3}$$

with the binomial coefficients given by

$$\binom{n}{i} = \begin{cases} \dfrac{n!}{i!(n-i)!} & 0 \le i \le n \\ 0 & \text{otherwise} \end{cases}$$

Bernstein polynomials have several important properties:

- *Partition of unity*
  All Bernstein polynomials $B_i^n(t)$ sum up to one everywhere in $[0, 1]$:

$$\sum_{i=0}^{n} B_i^n(t) \equiv 1 \tag{4.4}$$

- *Positivity*
  Bernstein polynomials are positive everywhere in $[0, 1]$.
- *Symmetry*
  Bernstein polynomials $B_i^n(t)$ are symmetric (see Figure 4.4)

$$B_i^n(t) = B_{n-i}^n(1-t). \tag{4.5}$$

- *Recursion*
  Bernstein polynomials satisfy the following recursion

$$B_i^n(t) = (1-t)B_i^{n-1}(t) + tB_{i-1}^{n-1}(t)$$

with $B_0^0(t) \equiv 1$ and $B_i^n(t) \equiv 0$ for $i \notin \{0, \ldots, n\}$.

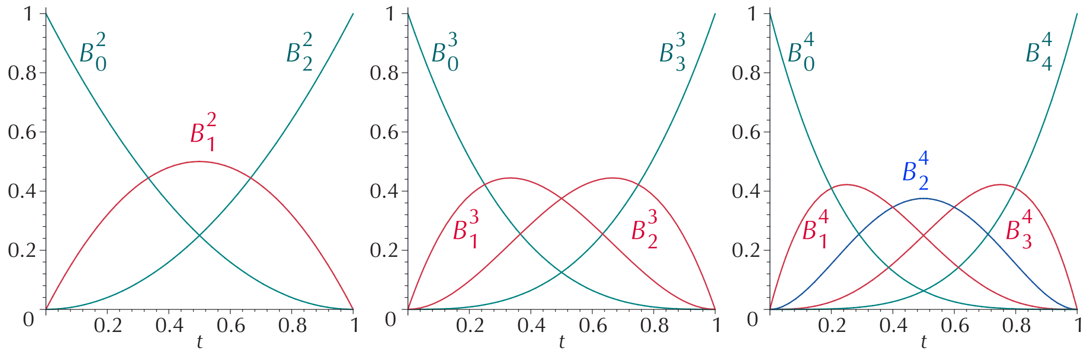


FIGURE 4.4 Quadratic, cubic and quartic Bernstein polynomials.

4.1.3 The Notion of Continuity

Similar to Section 3.1.2 we will need a notion of continuity in this chapter. The main concern of continuity in the context of piecewise patches is across patch boundaries. To this aim we call a piecewise curve or surface C^r*–continuous* if the first r derivatives exist and are in agreement at adjoining patch boundaries. Within the patch, the curve or surface is C^∞–continuous. A weaker statement than the parametric C^r continuity is given by the *geometric continuity* G^r. Geometric continuity only requires the tangents to vary continuously across patch boundaries.

4.2 BÉZIER CURVES

4.2.1 Definition and Properties

A Bézier curve of degree *n* is an expansion of the Bernstein polynomials $B_i^n(t)$ as

$$\mathbf{b}^n(t) = \sum_{i=0}^{n} \mathbf{b}_i B_i^n(t) \tag{4.6}$$

with the *control points* $\mathbf{b}_i$. Note that for two– or three–dimensional curves the control points are to be seen as vector quantities whereas in a functional setting they are scalars.

As illustrated in Figure 4.5, Bézier curves feature the following important properties:

- *Affine invariance*
  A Bézier curve and its control points are invariant under affine transformations. As a consequence, an affine transformation of a Bézier curve can easily be achieved by applying the corresponding affine map to its control points.
- *Convex hull property*
  The resulting Bézier curve always lies within the convex hull of its control points. This fact has several practical consequences, e.g. in intersection testing for collision detection or in raytracing and will be made heavy use of in Chapter 6.
- *Design property*
  The control polygon of a Bézier curve gives a rough impression of its shape. By subdivision or degree elevation of a Bézier curve, the control polygon more and more approximates the curve.
- *Endpoint interpolation*
  As a consequence of the partition of unity of Bernstein polynomials (Equation 4.4) together with the identities

  $$B_i^n(0) = B_i^n(1) = 1 \tag{4.7}$$

  a Bézier curve always interpolates its end control points $\mathbf{b}_0$ and $\mathbf{b}_n$.
- *Variation diminishing property*
  A straight line can have only as much intersections with the curve as it has intersections with its control polygon.

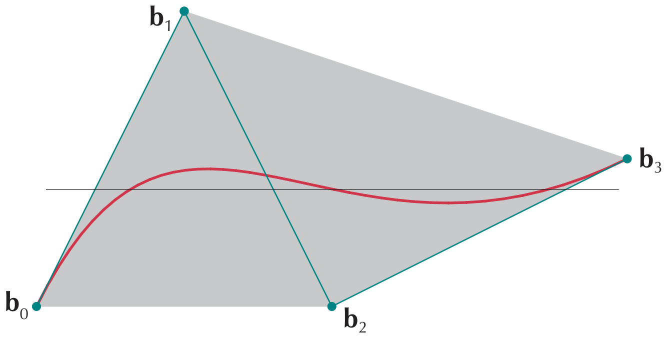


FIGURE 4.5 A cubic Bézier curve, its control polygon, convex hull, and variation diminishing property.

4.2.2 De Casteljau Algorithm

The de Casteljau algorithm is probably the most fundamental algorithm in the field of curve and surface design [43]. It is this algorithm that Paul de Casteljau started out from in the late fifties without considering the Bernstein basis. The connection between the geometrical construction scheme the algorithm gives and the Bernstein basis is easily seen by means of the following example:

Given any three points $\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2$ in $\mathbb{R}^2$ or $\mathbb{R}^3$ we construct

$$\begin{aligned} \mathbf{b}_0^1(t) &= (1-t)\mathbf{b}_0 + t\mathbf{b}_1 \\ \mathbf{b}_1^1(t) &= (1-t)\mathbf{b}_1 + t\mathbf{b}_2 \end{aligned}$$

$$\mathbf{b}_0^2(t) = (1-t)\mathbf{b}_0^1(t) + t\mathbf{b}_1^1(t)$$

Inserting the first two equations into the third yields

$$\mathbf{b}_0^2(t) = (1-t)^2\mathbf{b}_0 + 2t(1-t)\mathbf{b}_1 + t^2\mathbf{b}_2 \tag{4.8}$$

which is a quadratic expression in t describing a parabola when t varies from $-\infty$ to $+\infty$. The construction of Equation 4.8 in essence is a repeated linear interpolation. It can easily be verified that Equation 4.8 is equal to a quadratic Bézier curve using Equation 4.3 and Equation 4.6. It is possible to generalize the above construction scheme to polynomial curves of arbitrary degree n:

De Casteljau algorithm. Given the control points $\mathbf{b}_0, \mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{R}^3$ and $t \in \mathbb{R}$ set

$$\mathbf{b}_i^r(t) = (1-t)\mathbf{b}_i^{r-1}(t) + t\mathbf{b}_{i+1}^{r-1}(t) \qquad \begin{cases} r = 1, \ldots, n \\ i = 0, \ldots, n-r \end{cases}$$

and $\mathbf{b}_i^0(t) \equiv \mathbf{b}_i$. Then $\mathbf{b}_0^n(t)$ is the point with parameter value t on the Bézier curve $\mathbf{b}^n(t)$ according to Equation 4.6 [43].

Figure 4.6 illustrates the de Casteljau evaluation of a point on a cubic Bézier curve $\mathbf{b}^3(t)$ at $t = 0.4$.

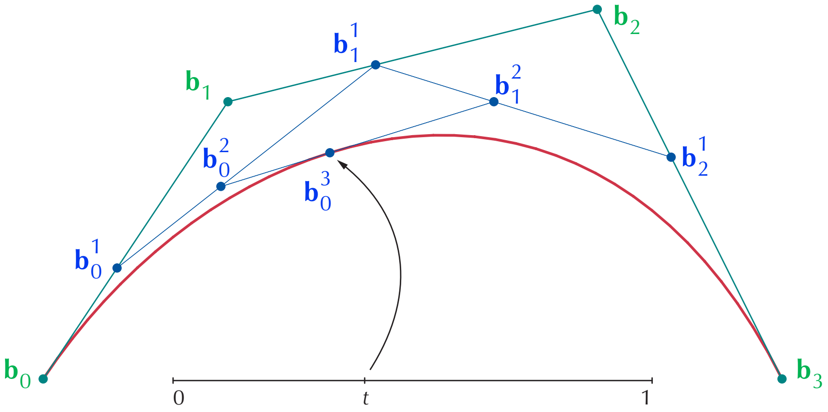


FIGURE 4.6 De Casteljau evaluation of a cubic Bézier curve at $t = 0.4$.

4.2.3 Derivatives of Bézier Curves

Derivatives of Bézier curves follow directly from the derivatives of the Bernstein polynomials (Equation 4.3)

$$\frac{d}{dt}B_i^n(t) = n[B_{i-1}^{n-1}(t) - B_i^{n-1}(t)]$$

to

$$\frac{d}{dt}\mathbf{b}^n(t) = n \cdot \sum_{i=0}^{n} [B_{i-1}^{n-1}(t) - B_i^{n-1}(t)]\mathbf{b}_i .$$

Since $B_i^n(t) \equiv 0$ for $i \notin \{0, \ldots, n\}$ this simplifies to

$$\frac{d}{dt}\mathbf{b}^n(t) = n \cdot \sum_{i=0}^{n-1} (\mathbf{b}_{i+1} - \mathbf{b}_i)B_i^{n-1}(t) . \tag{4.9}$$

As can be seen from this equation, the derivative of a Bézier curve of degree *n* is another Bézier curve of degree $n-1$. It is common practice to introduce a *forward differencing operator* Δ

$$\Delta\mathbf{b}_i = \mathbf{b}_{i+1} - \mathbf{b}_i$$

in order to simplify the notation of Equation 4.9 which now reads [43]

$$\frac{d}{dt}\mathbf{b}^n(t) = n \cdot \sum_{i=0}^{n-1} \Delta\mathbf{b}_i B_i^{n-1}(t) .$$

Higher order derivatives follow through generalization of the forward differencing operator to the iterated forward difference operator Δ^r

$$\Delta^r\mathbf{b}_i = \Delta^{r-1}\mathbf{b}_{i+1} - \Delta^{r-1}\mathbf{b}_j$$

to

$$\frac{d^r}{dt^r}\mathbf{b}^n(t) = \frac{n!}{(n-r)!} \cdot \sum_{i=0}^{n-r} \Delta^r\mathbf{b}_i B_i^{n-r}(t) .$$

Because of the endpoint interpolation of Bézier curves (Equation 4.7) the *r*–th derivatives at $t = 0$ and $t = 1$ of a curve of degree *n* convert to

$$\begin{aligned} \frac{d^r}{dt^r}\mathbf{b}^n(0) &= \frac{n!}{(n-r)!}\Delta^r\mathbf{b}_0 \\ \frac{d^r}{dt^r}\mathbf{b}^n(1) &= \frac{n!}{(n-r)!}\Delta^r\mathbf{b}_{n-r} \end{aligned} \tag{4.10}$$

which states that at an endpoint of an interval the *r*–th derivative only depends on the *r* neighboring control points and the endpoint itself. This fact will prove important in the construction of continuously differentiable piecewise Bézier curves in forthcoming sections.

Derivatives of Bézier curves may also be expressed in terms of the intermediate points generated by the de Casteljau algorithm:

$$\frac{d^r}{dt^r}\mathbf{b}^n(t) = \frac{n!}{(n-r)!}\Delta^r\mathbf{b}_0^{n-r}(t) \tag{4.11}$$

The case $r = 1$ is important in the following and therefore warrants special attention. Equation 4.11 together with the definition of the forward differencing operator yields

$$\frac{d}{dt}\mathbf{b}^n(t) = n[\mathbf{b}_1^{n-1}(t) - \mathbf{b}_0^{n-1}(t)]. \tag{4.12}$$

The intermediate points $\mathbf{b}_0^{n-1}$ and $\mathbf{b}_1^{n-1}$ thus determine the tangent vector at $\mathbf{b}^n(t)$ (see Figure 4.6).

4.2.4 Subdivision of Bézier Curves

It is often desirable to split a curve $\mathbf{b}^n(t)$ into several intervals which describe the same curve. This can be useful, e.g. if additional degrees of freedom are required or in the case of intersection testing for raytracing (see Chapter 6). An algorithm for intersection testing can be sketched as follows. Intersect the ray against the bounding box of the curve. If an intersection is reported, split the curve in two at $t = 0.5$ and repeat the box test recursively. As soon as the curve segment is close to a line, report the intersection with the line.

Assume a curve of degree *n* over $t \in [0, 1]$. Then the control points of the curve in the sub–interval $t \in [0, c]$ can be defined by means of the de Casteljau algorithm. The new *n* control points $\mathbf{c}_i$ follow after $i = 0, \ldots, n$ de Casteljau steps with $t = c$ to

$$\mathbf{c}_i = \mathbf{b}_0^i(c)$$

In other words, by calculating the point on the curve $\mathbf{b}^n(c)$, the de Casteljau algorithm not only computes the curve point at *c* but also determines the control points $\mathbf{c}_0$, $\mathbf{c}_1$, $\mathbf{c}_2$ and $\mathbf{c}_3$ at intermediate steps. Figure 4.7 illustrates the subdivision of a cubic curve at $c = 0.5$.

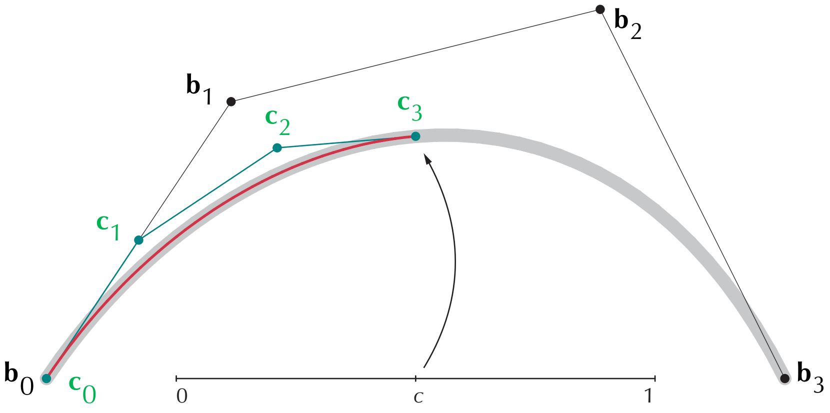


FIGURE 4.7 Subdivision of a cubic Bézier curve at $c = 0.5$.

From the symmetry property of Bernstein polynomials (Equation 4.5), it follows that the control points of the segment $[c, 1]$ are given by

$$\mathbf{c}_i = \mathbf{b}_i^{n-i}.$$

Sometimes one might be interested in an extrapolation of a Bézier curve beyond the interval $[0, 1]$. In complete analogy to subdivision, extrapolation to the interval $[1, d]$ yields the control points of the adjoining segment as

$$\mathbf{d}_i = \mathbf{b}_{n-i}^{i}(d). \tag{4.13}$$

4.2.5 Additional Bézier Curve Topics

Subdivision and derivatives of Bézier curves will be used in subsequent sections of the thesis. However, there are more topics in the context of Bézier curves which will not be covered in here, e.g. *degree elevation* and the more complex *degree reduction*. For a complete treatment, the reader is referred to the excellent text of Farin [43].

4.2.6 Piecewise Bézier Curves

As mentioned in Section 4.1.1 and Section 4.2.4, it is often required that the curve be described piecewise. The use of piecewise definitions offers additional flexibility in that additional control points are at disposal. Without piecewise formulations, the only way in which to increase the number of degrees of freedom would be to elevate the degree of the Bernstein basis functions. Unfortunately, for numerical reasons, this is impractical for degrees above about ten. However, piecewise definitions bear the disadvantage that one has to care for the continuity at patch boundaries.

In the following, we think of a piecewise curve as a mapping of the global coordinate u to the local coordinate t within each segment $[u_i, u_{i+1}]$ with $\{u_0, u_1, \ldots, u_n\}$ as

$$t = \frac{u - u_i}{u_{i+1} - u_i} = \frac{u - u_i}{h_i} \tag{4.14}$$

similar to Equation 4.1.

It is obvious, that C^0 continuity at patch boundaries is easy to achieve due to the endpoint interpolation property of Bézier curves. For higher order continuity conditions, let us look at two segments $\mathbf{b}_0, \ldots, \mathbf{b}_n$ in $[u_0, u_1]$ and $\mathbf{b}_{n+1}, \ldots, \mathbf{b}_{2n}$ in $[u_1, u_2]$. If we were to have C^∞ continuity, the two Bézier curves would have to be part of the same polynomial curve $\mathbf{b}^n(u)$ defined over $[u_0, u_2]$ and thus be a result of a subdivision process as described in Section 4.2.4. According to the extrapolation in Equation 4.13, the control points of the two segments must be related by

$$\mathbf{b}_{n+i} = \mathbf{b}_{n-i}^{i}(t) \qquad i = 0, \ldots, n$$

with $t = (u_2 - u_0)/(u_1 - u_0)$ being the local coordinate of u_2 with respect to the first interval $[u_0, u_1]$. If we were to change the control point b_{2n} arbitrarily, the derivatives up to order $n - 1$ would still agree at the patch boundary since they depend on the $n - 1$ neighboring points only (Equation 4.10).

We can thus state the C^r *condition for Bézier patches*: two Bézier curves defined over $u_0 \le u \le u_1$ and $u_1 \le u \le u_2$, by the control polygons $\mathbf{b}_0, \ldots, \mathbf{b}_n$ and $\mathbf{b}_n, \ldots, \mathbf{b}_{2n}$, respectively, are *r*–times continuously differentiable at $u = u_1$ if and only if

$$\mathbf{b}_{n+i} = \mathbf{b}^i_{n-i}(t) \qquad i = 0, \ldots, r \tag{4.15}$$

where $t = (u_2 - u_0)/(u_1 - u_0)$ is the local coordinate of u_2 with respect to the first interval $[u_0, u_1]$.

Another form of the above C^r condition can be found by equating the derivatives given by Equation 4.10 at the boundary. It should be kept in mind that the derivatives in Equation 4.10 are given in terms of the local coordinate *t* whereas for global piecewise continuity the derivatives with respect to the global parameter *u* are required to agree. Therefore, we must apply the chain rule corresponding to the variable substitution in Equation 4.14 which results in

$$\frac{d^r}{du^r}\mathbf{b}^n(u) = \left(\frac{1}{h_i}\right)^r \cdot \frac{d^r}{dt^r}\mathbf{b}^n(t) \tag{4.16}$$

where on the right hand side $\mathbf{b}^n(t)$ refers to the Bézier curve in segment *i* in the local coordinate *t*. Equating the derivatives according to Equation 4.10 at the boundary now yields the C^r condition

$$\frac{1}{h_0^i} \cdot \Delta^i \mathbf{b}_{n-i} = \frac{1}{h_1^i} \cdot \Delta^i \mathbf{b}_n \qquad i = 0, \ldots, r \tag{4.17}$$

between segment 0 and 1.

The case of C^1 continuity will prove important in subsequent sections. Therefore we will consider it in more detail. C^1 continuity requires the first derivative to agree at the boundary. Given the piecewise curve $\mathbf{b}(u)$ composed of $\mathbf{b}_0(u)$ and $\mathbf{b}_1(u)$ with local coordinates *s* and *t*, respectively. In order for $\mathbf{b}_0(s)$ and $\mathbf{b}_1(t)$ to blend smoothly at $s = 1$ and $t = 0$, respectively, it follows from Equation 4.16 and Equation 4.17 that

$$\begin{aligned} \frac{d}{du}\mathbf{b}(u) &= \frac{1}{h_0} \cdot \frac{d}{ds}\mathbf{b}_0(1) = \frac{1}{h_1} \cdot \frac{d}{dt}\mathbf{b}_1(0) = \\ &= h_1 \Delta \mathbf{b}_{n-1} = h_0 \Delta \mathbf{b}_n \end{aligned} \tag{4.18}$$

From Equation 4.10, we already know that only the three Bézier points $\mathbf{b}_{n-1}, \mathbf{b}_n, \mathbf{b}_{n+1}$ influence the first derivative at the junction point $\mathbf{b}_n$. From Equation 4.18 we further see that the three points must be collinear and must be in the ratio $h_0 : h_1 = (u_1 - u_0) : (u_2 - u_1)$. Figure 4.8 illustrates the situation.

4.2.7 Interpolation using Bézier Curves

Piecewise Bézier curves are ideally suited to interpolate to point data. This applies both to point data interpolation and Hermite interpolation. The term *Hermite interpolation* refers to the interpolation not only of points but also of derivatives at the points.

If one were to interpolate to point data only, the data would directly serve as the endpoints of Bézier segments leaving the remaining inner control points for guaranteeing smoothness constraints, a fact which will be useful in Chapter 5. In a Hermite setting, the

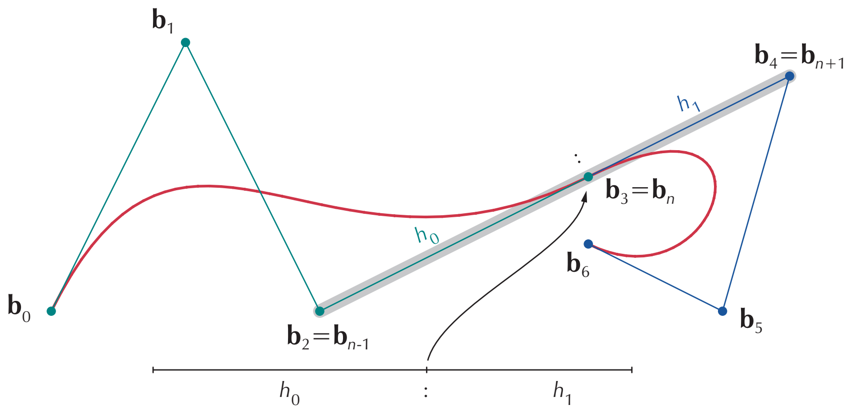

$\mathbf{b}_1$
$\mathbf{b}_4=\mathbf{b}_{n+1}$
h_1
$\mathbf{b}_3=\mathbf{b}_n$
h_0
$\mathbf{b}_6$
$\mathbf{b}_0$
$\mathbf{b}_5$
$\mathbf{b}_2=\mathbf{b}_{n-1}$
h_0 : h_1

FIGURE 4.8 C^1 condition on the example of cubic Bézier curves: the three Bézier points $\mathbf{b}_{n-1}, \mathbf{b}_n, \mathbf{b}_{n+1}$ must be collinear with ratio $h_0 : h_1$.

inner control points are determined such that the given derivatives at data points are satisfied.

From the previous sections, it is easy to see that a C^m–continuous interpolation in general requires Bézier curves of degree $n \geq 2m+1$. For the important case of C^1–continuous interpolation we therefore have to interpolate at a degree of $n = 3$. We will further investigate this case at the example of cubic Hermite interpolation given the point data $\mathbf{p}_0, \mathbf{p}_1$ and corresponding tangent vectors $\mathbf{v}_0, \mathbf{v}_1$. The objective is to find a cubic Bézier curve $\mathbf{b}(t)$ which interpolates to these pieces of data as

$$\mathbf{b}(0) = \mathbf{p}_0 \qquad \mathbf{b}'(0) = \mathbf{v}_0$$
$$\mathbf{b}(1) = \mathbf{p}_1 \qquad \mathbf{b}'(1) = \mathbf{v}_1$$

where the prime denotes differentiation. We are now left with the task of finding the Bézier control points which satisfy the above interpolation constraints. As mentioned before, the two control points at the end of the segment are easily identified due to the endpoint interpolation property of Bézier curves as

$$\mathbf{b}_0 = \mathbf{p}_0 \qquad \mathbf{b}_3 = \mathbf{p}_1 . \tag{4.19}$$

For the remaining two inner control points we recall the endpoint derivative for Bézier curves from Equation 4.10

$$\mathbf{b}'(0) = 3\Delta\mathbf{b}_0 \qquad \mathbf{b}'(1) = 3\Delta\mathbf{b}_2 . \tag{4.20}$$

Solving for $\mathbf{b}_1$ and $\mathbf{b}_2$ yields the control points as

$$\mathbf{b}_1 = \mathbf{p}_0 + \frac{1}{3}\mathbf{v}_0 \qquad \mathbf{b}_2 = \mathbf{p}_1 - \frac{1}{3}\mathbf{v}_1 .$$

Figure 4.9 gives an example of such an interpolation. It is obvious that the process may be repeated for additional Bézier segments. Continuity is either guaranteed by the appropriate interpolation of derivatives at the points or, in the case of a mere point interpolation, may result from an energy minimization scheme.

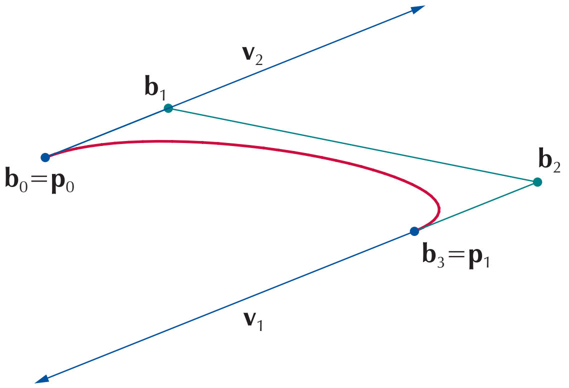


FIGURE 4.9 Cubic Hermite interpolation using Bézier curves.

It is illustrative to rearrange and sort the terms according to the given data. Starting out from the Bézier form of the interpolant we find successively

$$\begin{aligned}\mathbf{b}(t) &= \mathbf{p}_0 B_0^3(t) + \left(\mathbf{p}_0 + \frac{1}{3}\mathbf{v}_0\right) B_1^3(t) + \left(\mathbf{p}_1 - \frac{1}{3}\mathbf{v}_1\right) B_2^3(t) + \mathbf{p}_1 B_3^3(t)\\ &= \mathbf{p}_0 (B_0^3(t) + B_1^3(t)) + \mathbf{v}_0 \frac{1}{3} B_1^3(t) + \mathbf{v}_1 \left(-\frac{1}{3}\right) B_2^3(t) + \mathbf{p}_1 (B_2^3(t) + B_3^3(t))\\ &= \mathbf{p}_0 H_0^3(t) + \mathbf{v}_0 H_1^3(t) + \mathbf{v}_1 H_2^3(t) + \mathbf{p}_1 H_3^3(t)\end{aligned}$$

where the $H_i^3(t)$, $i = 0, \ldots, 3$ refer to the *cubic Hermite polynomials*. These polynomials are widely used for interpolation and finite element analysis because with respect to evaluation and differentiation at $t = 0$ and $t = 1$ each of the $H_i^3(t)$ equals 1 for one of these four operations and is zero for the remaining three:

$$\begin{array}{llll} H_0^3(0) = 1, & \frac{d}{dt}H_0^3(0) = 0, & \frac{d}{dt}H_0^3(1) = 0, & H_0^3(1) = 0\\ H_1^3(0) = 0, & \frac{d}{dt}H_1^3(0) = 1, & \frac{d}{dt}H_1^3(1) = 0, & H_1^3(1) = 0\\ H_2^3(0) = 0, & \frac{d}{dt}H_2^3(0) = 0, & \frac{d}{dt}H_2^3(1) = 1, & H_2^3(1) = 0\\ H_3^3(0) = 0, & \frac{d}{dt}H_3^3(0) = 0, & \frac{d}{dt}H_3^3(1) = 0, & H_3^3(1) = 1 \end{array} \tag{4.21}$$

Figure 4.10 exemplifies those properties.

4.3 TENSOR PRODUCT EXTENSION TO *N* DIMENSIONS

Obviously, it in CAGD is of much greater practical relevance to be able to handle surfaces in three–dimensional space than just curves, e.g. for the modeling of car bodies. The most obvious, though not most elegant and mathematically natural extension to surfaces of the above–presented univariate concepts is the so–called *tensor product* surface.

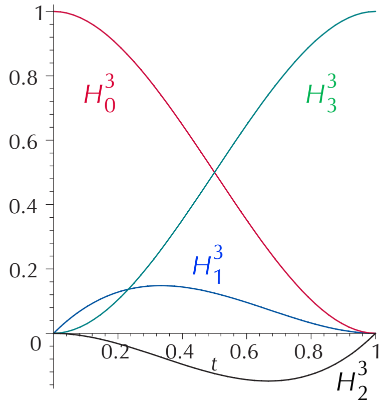


FIGURE 4.10 Cubic Hermite polynomials.

4.3.1 Patch Construction

The tensor product patch construction in essence multiplies two curve formulations for the *r*– and *s*–coordinate direction (see Figure 4.1). A Bézier surface of degree $m \times n$ can thus be formulated as

$$\mathbf{b}^{m,n}(r,s) = \sum_{i=0}^{m}\sum_{j=0}^{n} \mathbf{b}_{i,j} B_i^m(r) B_j^n(s)$$

with the $\mathbf{b}_{i,j}$ defining a control net of $m+1 \times n+1$ control points. Obviously, the degree does not necessarily have to be the same in the two parameter directions. Figure 4.11 gives an example of a tensor product Bézier surface of degree 3×2 and the corresponding control net.

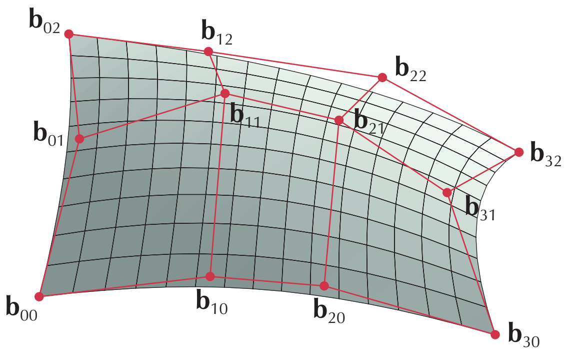


FIGURE 4.11 Tensor product Bézier patch of degree 3×2 and its control net.

4.3.2 Analogy to Univariate Case

A big advantage of tensor product surfaces lies in the complete analogy to the univariate formulation for curves.

On the one hand, tensor product Bézier patches share the same properties as do Bézier curves, i.e. invariance under affine transformations, convex hull property, variation diminishing property, design property and endpoint interpolation (see Section 4.2.1).

On the other hand, all concepts including the construction of smooth patch transitions follow analogously for tensor product patches. For partial derivatives in *r* and *s* analogous formulations apply [43]. Further, the de Casteljau algorithm easily generalizes to the tensor product case which follows from its construction as an iteration of linear interpolations to

$$\mathbf{b}_{i,j}^{k,k}(r,s) = \begin{bmatrix} 1-r & r \end{bmatrix} \begin{bmatrix} \mathbf{b}_{i,j}^{k-1,k-1} & \mathbf{b}_{i,j+1}^{k-1,k-1} \\ \mathbf{b}_{i+1,j}^{k-1,k-1} & \mathbf{b}_{i+1,j+1}^{k-1,k-1} \end{bmatrix} \begin{bmatrix} 1-s \\ s \end{bmatrix} \qquad \begin{aligned} k &= 1, \ldots, n \\ i, j &= 0, \ldots, n-k \end{aligned} \tag{4.22}$$

In the case of different degrees in *r* and *s* only $l = \min(m, n)$ steps can be taken according to Equation 4.22. At this point, the intermediate control points $\mathbf{b}_{i,j}^{l,l}$ form a curve control polygon which then has to be treated as in the univariate case. Figure 4.12 gives an example.

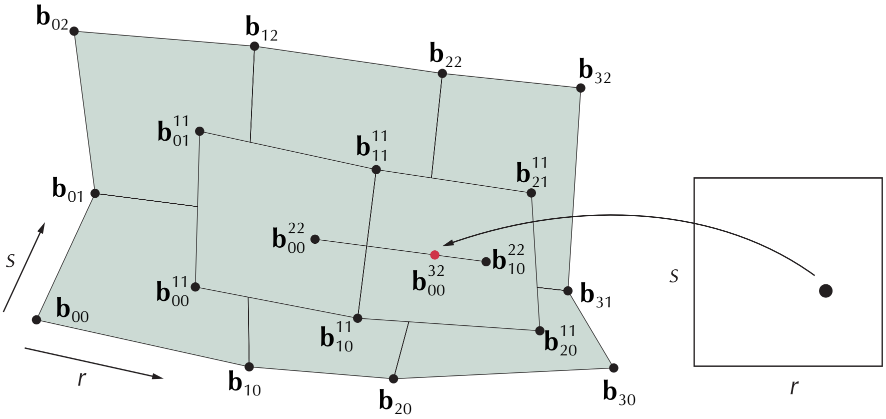


FIGURE 4.12 Tensor product de Casteljau algorithm on the example of a degree 3×2 patch.

The tensor product approach easily generalizes to three or even higher dimensions. Since we in this thesis will not employ this formulation, the reader is referred to the literature for more details [43].

4.4 TRIANGULAR BERNSTEIN–BÉZIER PATCHES

As mentioned in the introduction, triangular Bernstein–Bézier patches were the first representation of Bézier surfaces which were considered by de Casteljau. Although not as simple as the tensor product extension, their construction must be considered more elegant and mathematically more natural.

4.4.1 Barycentric Formulation

A barycentric formulation of the bivariate Bernstein polynomials of degree *n* follows to

$$B_{i,j,k}^{n}(r,s,t) = B_{\mathbf{i}}^{n}(\mathbf{r}) = \binom{n}{\mathbf{i}} r^i s^j t^k = \frac{n!}{i!\,j!\,k!} \cdot r^i s^j t^k \tag{4.23}$$

where **i** refers to the index triplet *i*, *j*, *k*, and **r** denotes the barycentric point *r*, *s*, *t* of evaluation. In analogy to the univariate polynomials $B_{\mathbf{i}}^{n}(\mathbf{r}) \equiv 0$ for one of (i, j, k) being negative or in the case that they sum up to more than *n*, $i + j + k > n$. In the following, the notation $|\mathbf{i}| = n$ refers to $i + j + k = n$ under the precondition that $i, j, k \geq 0$. Please note that although there are three barycentric coordinates *r, s, t* Equation 4.23 refers to the bivariate setting due to the linear dependence $r + s + t \equiv 1$. Figure 4.13 shows three of the $(n+1)(n+2)/2$ Bernstein polynomials for the degree $n = 3$.

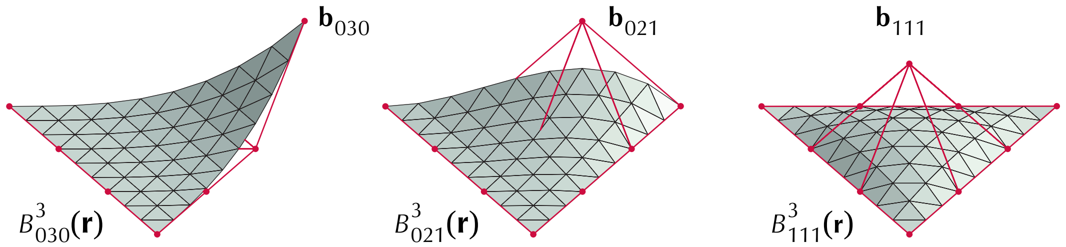

$\mathbf{b}_{030}$
$\mathbf{b}_{021}$
$\mathbf{b}_{111}$
$B_{030}^{3}(\mathbf{r})$
$B_{021}^{3}(\mathbf{r})$
$B_{111}^{3}(\mathbf{r})$

FIGURE 4.13 Cubic barycentric Bernstein polynomials (remaining polynomials follow from symmetry).

In correspondence with the univariate Bernstein polynomials (Section 4.1.2), their barycentric counterparts feature the same important properties:

- *Partition of unity*

$$\sum_{|\mathbf{i}| = n} B_{\mathbf{i}}^{n}(\mathbf{r}) \equiv 1$$

- *Positivity*

$$B_{\mathbf{i}}^{n}(\mathbf{r}) \geq 0, \qquad |\mathbf{i}| = n \quad \text{and} \quad r, s, t \in [0, 1]$$

- *Symmetry* (Figure 4.13)
- *Recursion*

$$B_{i,j,k}^{n}(\mathbf{r}) = r \cdot B_{i-1,j,k}^{n-1}(\mathbf{r}) + s \cdot B_{i,j-1,k}^{n-1}(\mathbf{r}) + t \cdot B_{i,j,k-1}^{n-1}(\mathbf{r}), \qquad |\mathbf{i}| = n$$

$$B_{0,0,0}^{0}(\mathbf{r}) \equiv 1$$

On the patch boundaries of the triangular domain we again have univariate Bernstein polynomials since

$$B_{0,j,k}^{n}(0, s, t) = B_{j}^{n}(s) = B_{k}^{n}(t).$$

Barycentric Bernstein polynomials are linearly independent and span the space of polynomials of total degree *n* defined over the barycentric domain [3]. The dimension of this space is equal $(n+1)(n+2)/2$ according to the number of Bernstein polynomials of degree *n*. Each element $\mathbf{b}^{n}(\mathbf{r})$ spanned by the $B_{\mathbf{i}}^{n}$ is of the form

$$\mathbf{b}^{n}(\mathbf{r}) = \sum_{|\mathbf{i}| = n} \mathbf{b}_{\mathbf{i}} B_{\mathbf{i}}^{n}(\mathbf{r})$$

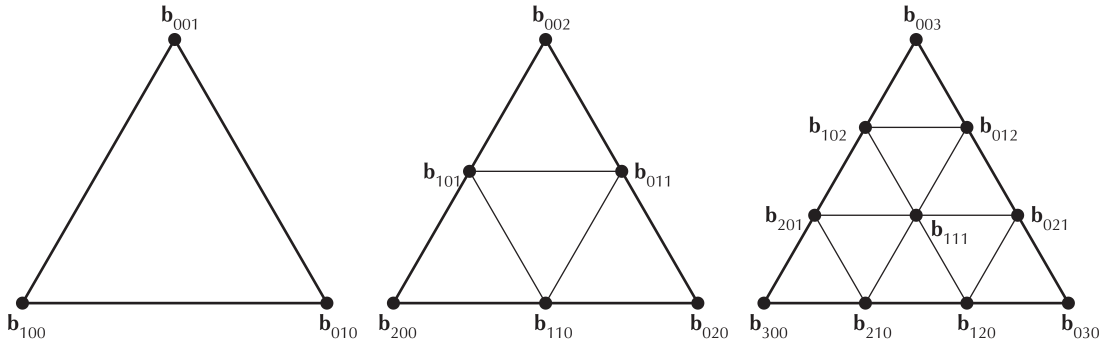


FIGURE 4.14 Schematic view of linear, quadratic and cubic triangular Bézier control nets.

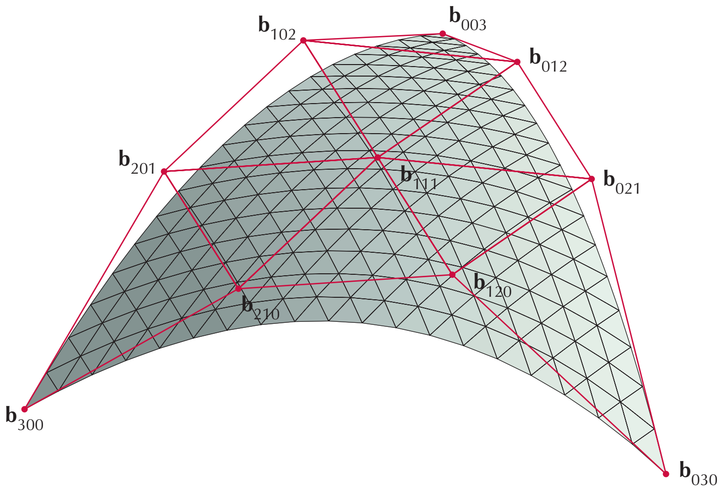


FIGURE 4.15 Cubic triangular Bézier patch and its defining control net.

which can be regarded as the parametric representation of a triangular Bézier patch of degree *n*. In analogy to the univariate case, the $\mathbf{b}_i$ are referred to as *Bézier control vertices* which form the *Bézier control net*. Figure 4.14 and Figure 4.15 illustrate the relationship of the degree of basis functions, the corresponding control nets and the surfaces spanned.

As a consequence of the properties of triangular Bernstein polynomials, the triangular Bézier patches share the same properties as Bézier curves: affine invariance, convex hull property, endpoint interpolation to mention only the most important. Many concepts of Bézier curves can easily be adapted to the triangular case. We will give the most important results in the following section.

4.4.2 Triangular Bézier Patch Topics

De Casteljau Algorithm. The de Casteljau algorithm is a generalization of the univariate case resulting from repeated barycentric linear interpolation. Given a triangular control net of points $\mathbf{b_i} \in \mathbb{R}^3$, $|\mathbf{i}| = n$ and a point in $\mathbb{R}^2$ with barycentric coordinates $\mathbf{r}$, we have

$$\mathbf{b}^l_{i,j,k}(\mathbf{r}) = r\cdot\mathbf{b}^{l-1}_{i+1,j,k}(\mathbf{r}) + s\cdot\mathbf{b}^{l-1}_{i,j+1,k}(\mathbf{r}) + t\cdot\mathbf{b}^{l-1}_{i,j,k+1}(\mathbf{r}) \qquad \begin{array}{c} |\mathbf{i}| = n-l \\ \mathbf{b}^0_{i,j,k} = \mathbf{b}_{i,j,k} \end{array} \tag{4.24}$$

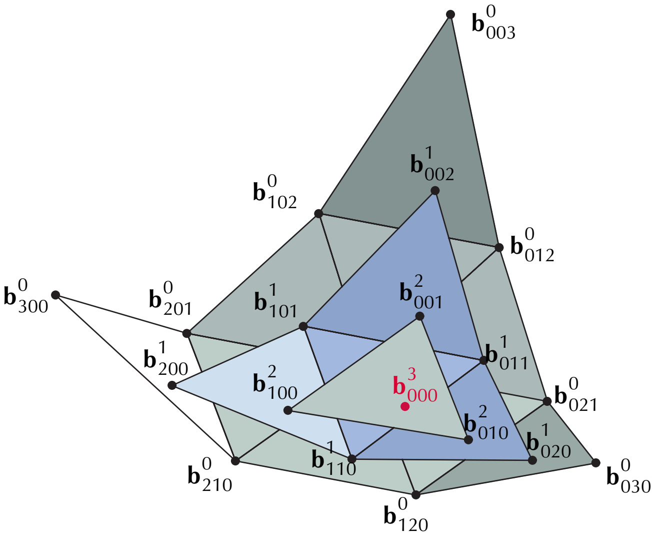


FIGURE 4.16 De Casteljau algorithm for triangular patches on the example of a cubic patch.

As before, the point $\mathbf{b}_{0,0,0}^n(\mathbf{r})$ is the point on the surface corresponding to the parameter value $\mathbf{r}$. Figure 4.16 illustrates the algorithm.

Derivatives. For the first partial derivative of $\mathbf{b}^n(\mathbf{r})$ we simply have

$$\mathbf{b}_r^n(\mathbf{r}) = n \cdot \sum_{|\mathbf{i}| = n-1} \mathbf{b}_{i+1,j,k} \cdot B_{ijk}^{n-1}(\mathbf{r}) \tag{4.25}$$

where the subscript r denotes differentiation with respect to the first barycentric parameter. Partial derivatives with respect to s and t follow analogously. Unfortunately, partial derivatives in the triangular setting do not correspond to directional derivatives with respect to the parameter lines due to the linear dependency of the third barycentric coordinate. The appropriate derivatives for triangular patches are directional derivatives which are given by

$$D_{\mathbf{d}}\mathbf{u}(\mathbf{r}) = d_r \cdot \mathbf{u}_r(\mathbf{r}) + d_s \cdot \mathbf{u}_s(\mathbf{r}) + d_t \cdot \mathbf{u}_t(\mathbf{r}) \tag{4.26}$$

where the subscript $\mathbf{d}$ refers to a barycentric vector $\mathbf{d} = (d_r, d_s, d_t)$ which results from subtracting two barycentric points $\mathbf{d} = \mathbf{p}_0 - \mathbf{p}_1$. In contrast to barycentric points the components of which are required to sum up to 1, barycentric vectors share the property that the components sum up to zero, $d_r + d_s + d_t \equiv 0$. A geometric interpretation of a directional derivative according to Equation 4.26 is as follows: the projection of a line which in parameter space passes through $\mathbf{r}$ and is parallel to $\mathbf{d}$ yields a curve on the surface. The tangent to this curve at $\mathbf{u}(\mathbf{r})$ is the directional derivative.

Using the partial derivatives according to Equation 4.25 in conjunction with Equation 4.26 yields

$$\begin{aligned} &D_{\mathbf{d}}\mathbf{b}^n(\mathbf{r}) = \\ &n \cdot \sum_{|\mathbf{i}| = n-1} [d_r \cdot \mathbf{b}_{i+1,j,k}(\mathbf{r}) + d_s \cdot \mathbf{b}_{i,j+1,k}(\mathbf{r}) + d_t \cdot \mathbf{b}_{i,j,k+1}(\mathbf{r})] B_{ijk}^{n-1}(\mathbf{r}) \end{aligned} \tag{4.27}$$

The expression in square brackets is easily identified as the result of the first step of the de Casteljau algorithm (Equation 4.24) which lets us rewrite Equation 4.27 as

$$D_{\mathbf{d}}\mathbf{b}^n(\mathbf{r}) = n \cdot \sum_{|\mathbf{i}| = n-1} \mathbf{b}^1_{ijk}(\mathbf{d}) \cdot B^{n-1}_{ijk}(\mathbf{r}) . \tag{4.28}$$

It is interesting to note that the expression in Equation 4.28 can be interpreted as taking one de Casteljau step with respect to the direction **d** followed by $n-1$ de Casteljau steps with respect to the parameter **r** at which we want to evaluate the directional derivative. Of course it is irrelevant in which order the steps are taken [38] which lets us rewrite Equation 4.28:

$$\begin{aligned} D_{\mathbf{d}}\mathbf{b}^n(\mathbf{r}) &= n \cdot \sum_{|\mathbf{i}| = 1} \mathbf{b}^{n-1}_{ijk}(\mathbf{r}) \cdot B^1_{ijk}(\mathbf{d}) \\ &= n(d_r \cdot \mathbf{b}^{n-1}_{100} + d_s \cdot \mathbf{b}^{n-1}_{010} + d_t \cdot \mathbf{b}^{n-1}_{001}) \end{aligned}$$

This result is obviously true for all directions $\mathbf{d} \in \mathbb{R}^2$ which leads to the conclusion that the $\mathbf{b}^{n-1}_{100}$, $\mathbf{b}^{n-1}_{010}$, $\mathbf{b}^{n-1}_{001}$ define the tangent plane at $\mathbf{b}^n(\mathbf{r})$. In particular, the three vertices $\mathbf{b}_{n,0,0}$, $\mathbf{b}_{n-1,1,0}$, $\mathbf{b}_{n-1,0,1}$ define the tangent plane at the triangle corner $r = 1$ which is a direct generalization of the univariate result of Equation 4.12. Similar results hold for the two remaining corners. The corner tangent planes are shown in Figure 4.17.

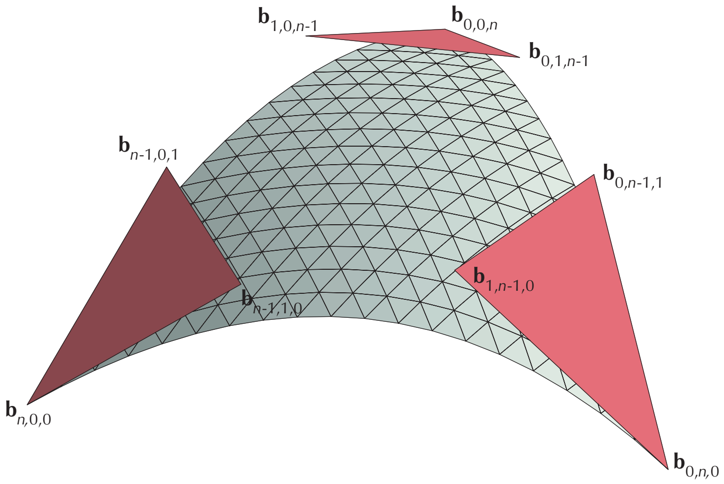


FIGURE 4.17 The control vertices at the corners define the tangent planes at the respective corners of a Bézier patch.

Similar expressions can be found for higher order derivatives [43]:

$$\begin{aligned} D^l_{\mathbf{d}}\mathbf{b}^n(\mathbf{r}) &= \frac{n!}{(n-l)!} \cdot \sum_{|\mathbf{i}| = n-l} \mathbf{b}^l_{ijk}(\mathbf{d}) \cdot B^{n-l}_{ijk}(\mathbf{r}) \\ &= \frac{n!}{(n-l)!} \cdot \sum_{|\mathbf{i}| = l} \mathbf{b}^{n-l}_{ijk}(\mathbf{r}) \cdot B^l_{ijk}(\mathbf{d}) \end{aligned}$$

It is also possible to compute mixed directional derivatives: taking two vectors $\mathbf{d}_1$ and $\mathbf{d}_2$, then corresponding mixed directional derivatives evaluate to

$$D^{l,m}_{\mathbf{d}_1,\mathbf{d}_2}\mathbf{b}^n(\mathbf{r}) = \frac{n!}{(n-l-m)!}\mathbf{b}[\mathbf{d}_1^{\langle l\rangle}, \mathbf{d}_2^{\langle m\rangle}, \mathbf{r}^{\langle n-l-m\rangle}] \tag{4.29}$$

where the notation in square brackets is the so–called *blossoming principle* [33, 113, 114]. It can be interpreted as taking l de Casteljau steps with respect to $\mathbf{d}_1$, followed by m steps with respect to $\mathbf{d}_2$ and $n-l-m$ steps with respect to the point of evaluation $\mathbf{r}$.

For the discussion of cross–boundary derivatives we consider the edge $r = 0$ and a direction $\mathbf{d}$ not parallel to it. For the first directional derivative with respect to $\mathbf{d}$ across the patch boundary $r = 0$ we find

$$D_{\mathbf{d}}\mathbf{b}^n(\mathbf{r})\Big|_{r=0} = n \cdot \sum_{|\mathbf{i}| = n-1} \mathbf{b}^1_{0jk}(\mathbf{d}) \cdot B^{n-1}_{0jk}(\mathbf{r})\Big|_{r=0} . \tag{4.30}$$

This is a univariate expression since $\mathbf{r} = (0, s, t) = (0, s, 1-s)$. From the fact that the $\mathbf{b}^1_{0jk}$ with $|\mathbf{i}| = n-1$ only depend on the control vertices on the boundary and the next inner row we can conclude that the first cross–boundary derivative only depends on the first two rows of control vertices at the boundary under consideration. Similar, k–th directional derivatives only depend on the first $k+1$ rows. Again, that result is a direct generalization of the univariate case (Equation 4.10) and will be useful in the construction of smooth patch transitions. Further, these facts prove important for the interpolation of a point set with triangular connectivity which in addition to the derivatives at the points also specifies derivatives across triangle boundaries [41, 43].

In addition to directional derivatives, it is often important to have derivatives with respect to the *u*– and *v*–parameter directions of a surface. Similar to Equation 4.16 these derivatives follow from the chain rule. Suppose a triangle in (u, v)–parameter space as illustrated in Figure 4.18. Substituting $t = 1-r-s$, we can write the transformation of local barycentric coordinates into global (u, v)–coordinates in matrix notation as

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} u_0 & u_1 & u_2 \\ v_0 & v_1 & v_2 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} r \\ s \\ 1-r-s \end{bmatrix} . \tag{4.31}$$

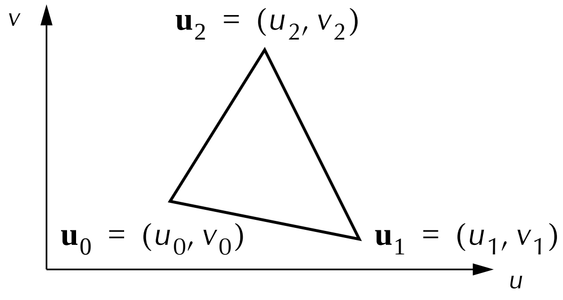

v
$\mathbf{u}_2 = (u_2, v_2)$
$\mathbf{u}_0 = (u_0, v_0)$
$\mathbf{u}_1 = (u_1, v_1)$
u

FIGURE 4.18 Triangle in (u, v)–parameter space.

An inverse transformation follows trivially from inverting the matrix to

$$\begin{bmatrix} r \\ s \\ t \end{bmatrix} = \begin{bmatrix} u_0 & u_1 & u_2 \\ v_0 & v_1 & v_2 \\ 1 & 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}.$$

Derivatives with respect to local barycentric coordinates follow from derivatives with respect to global coordinates to

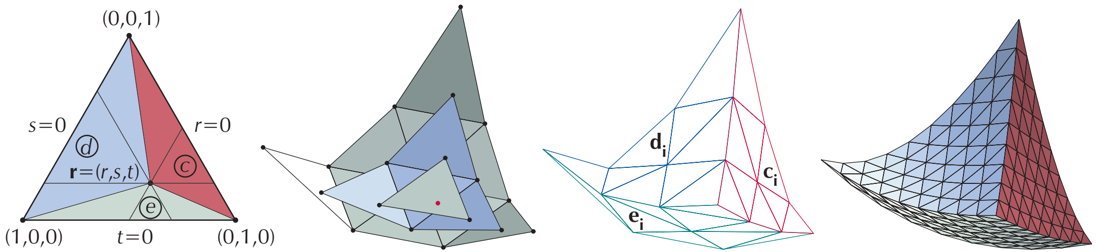
$$\begin{bmatrix} \frac{\partial}{\partial r} \\ \frac{\partial}{\partial s} \end{bmatrix} = \mathbf{J} \begin{bmatrix} \frac{\partial}{\partial u} \\ \frac{\partial}{\partial v} \end{bmatrix} \qquad \mathbf{J} = \begin{bmatrix} \frac{\partial u}{\partial r} & \frac{\partial v}{\partial r} \\ \frac{\partial v}{\partial s} & \frac{\partial v}{\partial s} \end{bmatrix} = \begin{bmatrix} u_0 - u_2 & v_0 - v_2 \\ u_1 - u_2 & v_1 - v_2 \end{bmatrix}$$

where **J** is the *Jacobi matrix,* or more precisely the Jacobi operator, and *u,v* are transformation functions of type $f(r, s)$ given by Equation 4.31. An inverse relation, expressing global derivatives in terms of partial barycentric derivatives is given by

$$\begin{bmatrix} \frac{\partial}{\partial u} \\ \frac{\partial}{\partial v} \end{bmatrix} = \mathbf{J}^{-1} \begin{bmatrix} \frac{\partial}{\partial r} \\ \frac{\partial}{\partial s} \end{bmatrix} \qquad \mathbf{J}^{-1} = \frac{1}{\det(\mathbf{J})} \begin{bmatrix} v_1 - v_2 & v_2 - v_0 \\ u_2 - u_1 & u_0 - u_2 \end{bmatrix}$$

where the determinant $\det(\mathbf{J}) = (u_0 - u_2)(v_1 - v_2) - (u_1 - u_2)(v_0 - v_2)$ is equal to twice the surface of the triangle in parameter space.

Subdivision. Subdivision of triangular Bézier patches can be formulated by means of the de Casteljau algorithm. Obviously, subdivision of a triangular patch at a barycentric point $\mathbf{r} = (r, s, t)$ yields three triangular sub–patches which we denote by c, d, e (Figure 4.19a). Analogously to the univariate situation, the control points $\mathbf{c}_i$, $\mathbf{d}_i$, $\mathbf{e}_i$ which



(a) (b) (c) (d)

FIGURE 4.19 Subdivision of barycentric patches at an internal point:
(a) Subdivision into three parametric sub–patches
(b) De Casteljau evaluation corresponding to point of subdivision $\mathbf{r} = (r, s, t)$
(c) Control nets of sub–patches resulting from intermediate de Casteljau points
(d) Resulting sub–patches

describe the original patch surface with respect to the corresponding triangular sub–domains c, d, e result from the intermediate steps of the de Casteljau algorithm employed to evaluate the point $\mathbf{b}(\mathbf{r})$

$$\begin{array}{llll} \mathbf{c}_{\mathbf{i}_{i=l}} = \mathbf{c}_{ljk} = \mathbf{b}^{l}_{\mathbf{i}_{i=0}}(\mathbf{r}) = \mathbf{b}^{l}_{0jk}(\mathbf{r}) & l & = & 0, \ldots, n \\ \mathbf{d}_{\mathbf{i}_{j=l}} = \mathbf{d}_{ilk} = \mathbf{b}^{l}_{\mathbf{i}_{j=0}}(\mathbf{r}) = \mathbf{b}^{l}_{i0k}(\mathbf{r}) & |\mathbf{i}_{*=0}| & = & n-l \\ \mathbf{e}_{\mathbf{i}_{k=l}} = \mathbf{e}_{ijl} = \mathbf{b}^{l}_{\mathbf{i}_{k=0}}(\mathbf{r}) = \mathbf{b}^{l}_{ij0}(\mathbf{r}) & |\mathbf{i}_{*=l}| & = & n \end{array} \qquad (4.32)$$

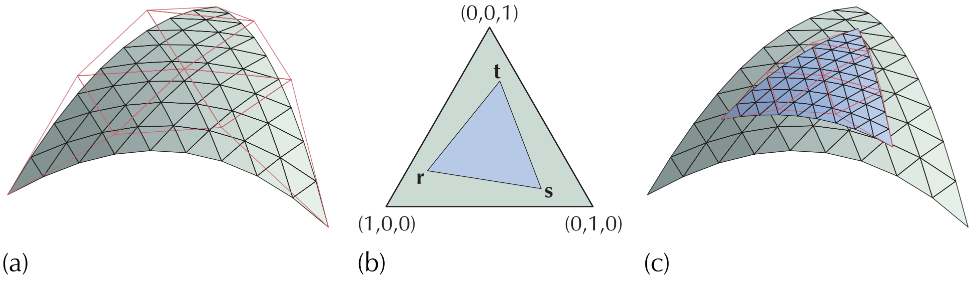
where n is the degree of the patch and the notations $* = 0$ and $* = l$ refer to one of the control vertex indices to equal 0 and l, respectively. See Figure 4.19 for an illustration of the situation.

In Chapter 6, we will need a more general subdivision with respect to three arbitrary internal barycentric points **r**, **s** and **t**. Such a subdivision was first introduced by Goldman [54] and later clarified by Boehm and Farin [19]. We follow the more elegant equivalent notation of Seidel [128] which is based on the blossoming principle: the control points $\mathbf{c}_{ijk}$ with respect to the triangular sub–domain (**r**, **s**, **t**) follow to

$$\mathbf{c}_{ijk} = \mathbf{b}[\mathbf{r}^{\langle i\rangle}, \mathbf{s}^{\langle j\rangle}, \mathbf{t}^{\langle k\rangle}] \qquad |\mathbf{i}| = n\,. \qquad (4.33)$$

Again, the blossom in Equation 4.33 is to be interpreted as taking i de Casteljau steps with respect to **r**, followed by j steps with respect to **s** and k steps with respect to **t**. See Figure 4.20 for an example.

Obviously, the subdivision according to Equation 4.32 can be regarded as a special case of Equation 4.33. This shows the very general nature of Seidel's formulation.



(a) (b) (c)

FIGURE 4.20 Subdivision of barycentric patches with respect to three internal points:
(a) Original patch and control net
(b) Parametric situation with triangular sub–domain spanned by **r**, **s** and **t**
(c) Sub–patch and control net corresponding to triangular domain (**r**, **s**, **t**)

4.4.3 C^m Continuity Conditions across Patch Boundaries

Before considering surfaces which consist of arbitrary many triangular patches forming an overall C^m surface it is illustrative to study the case of just two adjacent patches.

C^0 continuity across a patch boundary results from a common boundary curve, i.e. common control points on the boundary under consideration. As has been shown before, partial derivatives are not suited for the definition of higher order continuity conditions.

We therefore define C^m continuity at a patch boundary using the following generalization: assume two adjacent triangles in parameter space (see Figure 4.21a). Each line crossing the common edge of the two triangles is mapped onto a curve on each of the two triangular surfaces. If this piecewise curve is C^m–continuous for all crossing lines then the patch transition is called C^m–continuous [43].

Before deriving the conditions for C^m continuity, we first look at the situation of a C^∞–continuous continuation of a patch beyond its triangular domain. Analogously to the univariate case, this can be regarded as an extrapolation which is a special case of subdivision according to (Equation 4.32). Using Figure 4.21a we define the domain point $\hat{\mathbf{a}}$ in barycentric coordinates with respect to the triangle $(\mathbf{a}, \mathbf{b}, \mathbf{c})$

$$\hat{\mathbf{a}} = r\mathbf{a} + s\mathbf{b} + t\mathbf{c}$$

where r, s, t can be determined using Equation 4.2. Then we evaluate the point on the surface $\mathbf{b}(r, s, t)$ using the de Casteljau algorithm. The intermediate de Casteljau points according to Equation 4.32 are the control points of a patch which extends the surface defined over $(\mathbf{a}, \mathbf{b}, \mathbf{c})$ C^∞–continuously to the domain $(\hat{\mathbf{a}}, \mathbf{b}, \mathbf{c})$. The triangles of the resulting extended control net are an affine map of the corresponding triangular domain in parameter space. Figure 4.21a–d illustrate the construction of such a patch extension across the edge $r = 0$.

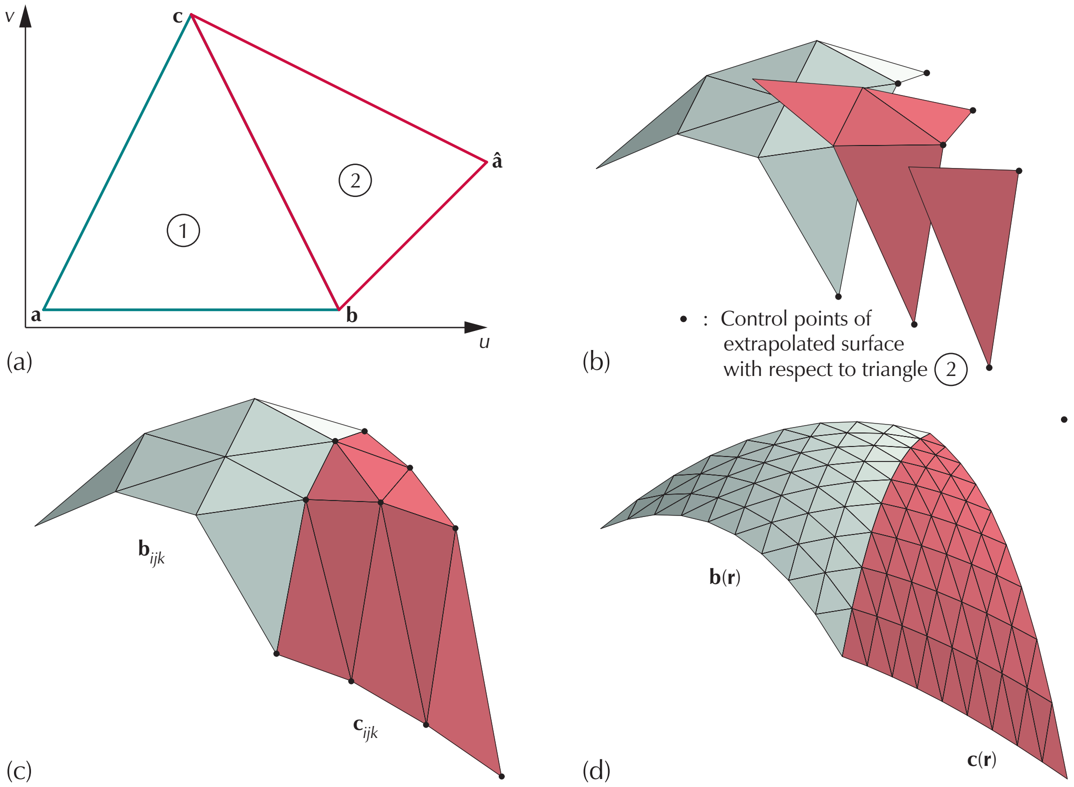

v
c
â
②
①
a
b
u
(a)
(b)
• : Control points of extrapolated surface with respect to triangle ②
b_ijk
c_ijk
(c)
b(r)
c(r)
(d)

FIGURE 4.21 Construction of a C^∞–continuous cubic patch extension:
(a) Situation in the parameter domain
(b) Construction of the control net (intermediate de Casteljau points)
(c) Control nets of both patches
(d) Resulting C^∞–continuous surface

From Equation 4.30 we know that for a patch transition between two patches of degree n to be C^m, $m \le n$ continuous, only $m+1$ rows of the adjoining control net are fixed. According to Equation 4.32, they follow to

$$\begin{aligned} \mathbf{c}_{\mathbf{i}_{i=l}} &= \mathbf{c}_{ljk} = \mathbf{b}^l_{\mathbf{i}_{i=0}}(\mathbf{r}) = \mathbf{b}^l_{0jk}(\mathbf{r}) & l &= 0, \ldots, m \\ \mathbf{d}_{\mathbf{i}_{j=l}} &= \mathbf{d}_{ilk} = \mathbf{b}^l_{\mathbf{i}_{j=0}}(\mathbf{r}) = \mathbf{b}^l_{i0k}(\mathbf{r}) & |\mathbf{i}_{*=0}| &= n-l \\ \mathbf{d}_{\mathbf{i}_{k=l}} &= \mathbf{e}_{ijl} = \mathbf{b}^l_{\mathbf{i}_{k=0}}(\mathbf{r}) = \mathbf{b}^l_{ij0}(\mathbf{r}) & |\mathbf{i}_{*=l}| &= n \end{aligned} \tag{4.34}$$

across the edges $r = 0, s = 0, t = 0$ with $\mathbf{r}$ being the barycentric coordinate of the triangle vertex $\hat{\mathbf{a}}$. The remaining control points may be chosen arbitrarily.

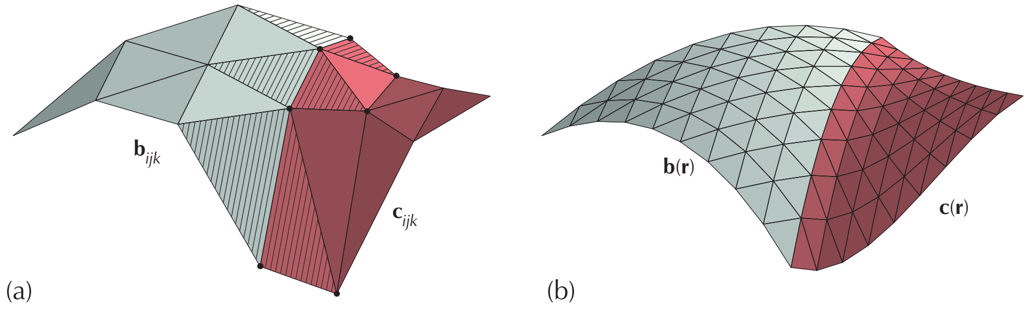


FIGURE 4.22 Construction of a C^1–continuous cubic patch transition:
(a) Control nets (black points fixed by C^1 condition, coplanar triangle pairs hatched)
(b) Resulting C^1–continuous patch transition

For $l = 0$, Equation 4.34 converts to the obvious C^0 condition which requires the control points on the patch boundary to coincide. Further, for C^1–continuous patch transitions, only the control points on the edge and one neighboring row are of importance. Figure 4.22 illustrates the C^1 condition across edge $r = 0$. In particular, each pair of hatched triangles is coplanar due to the linear interpolation inherent to the de Casteljau evaluation.

4.4.4 Triangular Hermite Interpolation

Having established the construction of smooth patch transitions for piecewise surfaces, we in this section will discuss interpolants to a function $\mathbf{f}(\mathbf{r})$ over a triangular domain. In particular, we will interpolate to position and derivative information at the vertices and across edges of a triangulation. In analogy to the univariate case, that type of interpolation is referred to as Hermite interpolation. We will focus on the main aspects with respect to this thesis only. For a good survey and ongoing references the reader is referred to [41].

C^0 Nine Parameter Interpolant. The given data are position and the derivatives along the edges at each triangle vertex. At the vertex $(1, 0, 0)$ we have

$$D^{r,s}_{\mathbf{e}_1, \mathbf{e}_2}\mathbf{f}(1, 0, 0) \qquad r + s \le 1$$

where $\mathbf{e}_1 = (0, 1, 0) - (1, 0, 0)$ and $\mathbf{e}_2 = (0, 0, 1) - (1, 0, 0)$ denote the barycentric directions along the adjoining edges and the operator D refers to the directional derivative according to Equation 4.29. At the two remaining corner vertices analogous data is provided. The determination of the nine boundary control points of a cubic triangular Bézier patch reduces to the univariate Hermite interpolation according to Equations 4.19 and 4.20. At the corner $r = 1$ and the neighboring control vertex on edge $t = 0$ we have

$$\mathbf{b}_{300} = f(1, 0, 0) \qquad \mathbf{b}_{210} = \frac{1}{3}D_{\mathbf{e}_1}\mathbf{f}(1, 0, 0) + \mathbf{b}_{300}. \tag{4.35}$$

The remaining control vertices on the boundary follow from symmetry. Only the center control point $\mathbf{b}_{111}$ is independent of the prescribed data and thus may be chosen arbitrarily. While this interpolant works well in the setting with only three points to interpolate at, it has a very serious disadvantage when we have to deal with data prescribed at several adjoining triangles: it requires C^1 data, but the resulting interpolated surface is only C^0 across the edges of the triangles. This is because in general it is impossible to choose the $\mathbf{b}_{111}$ such that the resulting surface is globally C^1.

C^1 Quintic Interpolant. In order to produce overall C^1–continuous surfaces both higher order interpolation functions and additional derivative information must be used. Derivative information now comprises cross–boundary derivatives at the midpoints of triangle edges as well as derivatives up to order two at the vertices. Quintic functions are needed in order to have enough degrees of freedom to satisfy cross–boundary smoothness. Second order derivatives at the vertices on the one hand determine additional boundary control points inherent to quintic Bézier patches. On the other hand, the mixed second order derivatives prevent the gray shaded control points in Figure 4.23 from overdetermination across neighboring edges due to potentially conflicting cross–boundary derivatives.

Consequently, at corner $(1, 0, 0)$ we now have to interpolate to the data

$$D^{r,s}_{\mathbf{e}_1,\mathbf{e}_2}\mathbf{f}(1, 0, 0) \qquad r + s \le 2$$

with analogous data at the other two corners of the triangle. In addition, we prescribe cross–boundary derivatives at the edge midpoints, e.g. at edge $r = 0$

$$D_{\mathbf{d}}\mathbf{f}(0, \tfrac{1}{2}, \tfrac{1}{2})$$

with $\mathbf{d} = (d_r, d_s, d_t)$ denoting a barycentric direction not parallel to the edge under consideration. Thus we have 21 pieces of prescribed data for the quintic patch determined by 21 control vertices. We now determine exemplarily the Bézier control vertices pertaining to the pieces of information given at corner $(1, 0, 0)$. Please refer to Figure 4.23 for the numbering of control vertices.

The points on edge $t = 0$ around the corner $(1, 0, 0)$ again follow from univariate relationships (Equation 4.10) to

$$\mathbf{b}_{500} = \mathbf{f}(1, 0, 0)$$

$$\mathbf{b}_{410} = \frac{1}{5}D_{\mathbf{e}_1}\mathbf{f}(1, 0, 0) + \mathbf{b}_{500}$$

$$\mathbf{b}_{320} = \frac{1}{20}D^2_{\mathbf{e}_1}\mathbf{f}(1, 0, 0) + 2\mathbf{b}_{410} - \mathbf{b}_{500}$$

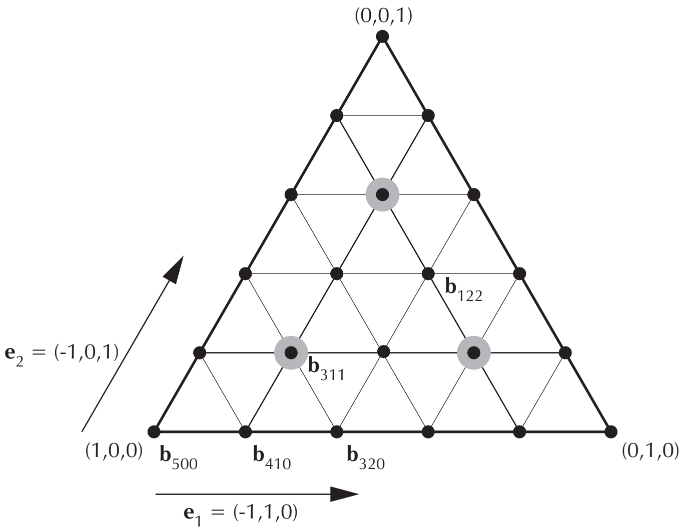


FIGURE 4.23 Numbering of control points for the quintic triangular interpolant: shaded points are prevented from overdetermination across neighboring edges by prescribing mixed second order derivatives at the corners.

For the unique determination of point $\mathbf{b}_{311}$ we now are in need of the mixed directional derivative $D^{1,1}_{\mathbf{e}_1,\mathbf{e}_2}\mathbf{f}(1,0,0)$. We from Equation 4.29 find

$$\mathbf{b}_{311} = \frac{1}{20}D^{1,1}_{\mathbf{e}_1,\mathbf{e}_2}\mathbf{f}(1,0,0) + \mathbf{b}_{410} + \mathbf{b}_{401} - \mathbf{b}_{500}.$$

In addition, the cross–boundary derivatives can now be used to determine the remaining control vertices $\mathbf{b}_{122}, \mathbf{b}_{221}, \mathbf{b}_{212}$. We illustrate the procedure for $\mathbf{b}_{122}$. From Equation 4.30 we know the univariate expression representing the first directional derivative with respect to **d** across the boundary $r = 0$. We may rewrite Equation 4.30 to

$$D_{\mathbf{d}}\mathbf{f}(\mathbf{m}_r) = 5 \cdot \sum_{|\mathbf{i}| = 4} \mathbf{b}^1_{0jk}(\mathbf{d}) \cdot B^4_{0jk}(\mathbf{m}_r) \tag{4.36}$$

where $\mathbf{m}_r = (0, 1/2, 1/2)$ is the barycentric coordinate of the midpoint on edge $r = 0$. Solving that expression for the unknown Bézier control point $\mathbf{b}_{122}$ yields

$$\mathbf{b}_{122} = \frac{1}{d_r}\left[\frac{1}{B^4_{022}(\mathbf{m}_r)}\left(\frac{1}{5}D_{\mathbf{d}}\mathbf{f}(\mathbf{m}_r) - \sum_{\substack{|\mathbf{i}| = 4\\ \mathbf{i} \neq 022}} \mathbf{b}^1_{0jk}(\mathbf{d}) \cdot B^4_{0jk}(\mathbf{m}_r)\right) - d_s\mathbf{b}_{032} - d_t\mathbf{b}_{023}\right]. \tag{4.37}$$

This expression is only meaningful for $d_r \neq 0$ which is equivalent to requiring the direction of the prescribed derivative not to be parallel to the edge. Normally, the directions are chosen to be perpendicular to the edges [41]. For the edge $r = 0$ this yields

$$\mathbf{d} = (1, \lambda, -1-\lambda) \qquad \lambda = \frac{(\mathbf{u}_1 - \mathbf{u}_2)^T(\mathbf{u}_0 - \mathbf{u}_1)}{\|\mathbf{u}_1 - \mathbf{u}_2\|} \tag{4.38}$$

where the $\mathbf{u}_i$ refer to the parametric coordinates of the triangle corners (see Figure 4.18).

The General Case. Farin in [41] states the following: A local piecewise polynomial surface interpolant that interpolates to all derivatives up to order *m* at the vertices of a triangulation and that is globally *m* times differentiable must be of degree $n \geq 4m + 1$. In order not to suffer from overdetermined control vertices similar to the shaded points in Figure 4.23, all derivatives up to order $2m$ have to be specified at each vertex of the triangulation. For a proof, see [41].

It is worth noting that this observation is only true when one restricts the interpolants to be *local*. This term reflects the situation where the interpolant over each triangle of the triangulation depends only on pieces of information belonging to the triangle under consideration. If we were to loosen that constraint, i.e. if we were to consider interpolants that are defined by *global* requirements (e.g. minimizing an energy functional), we can achieve higher order continuous surfaces at lower degrees. For example, at the cost of allowing global dependencies, one can construct a globally C^1–continuous surface which only interpolates to positions using cubic triangular Bézier functions. The non–interpolating control points of the patches are defined by minimizing a thin–plate functional and in addition are required to satisfy inter–patch continuity constraints. This approach leads to the solution of possibly large systems of linear equations needed for resolving global dependencies of control points. Unfortunately, those systems tend to be ill–conditioned. Further, the implementation of such an interpolation scheme is very tedious. Consequently, the approach has not been followed any further.

4.4.5 Split–Triangle Interpolants

All the interpolants we have seen so far share one common disadvantage: they require higher order derivative data at vertices than the desired order of overall continuity. This is due to the fact, that degree elevation only provides additional degrees of freedom but fails to deliver both a method in which to determine additional control points on the boundary and a means to prevent the shaded points in Figure 4.23 from overdetermination. This situation is uneconomical and thus the goal is to interpolate to position and derivative information which is of the same order as the required global surface continuity.

The solution to the problem lies in the consideration of piecewise interpolants over each triangle of the triangulation, i.e. in splitting up *macro–triangles* into several *micro–triangles.*

Clough–Tocher Split. The conceptually simplest such split–triangle interpolant is the *Clough–Tocher split* which originally stems from the finite element literature [28, 136]. The split is obtained by dividing each triangle in the triangulation about its centroid into three micro–triangles. On each micro–triangle the interpolant is cubic. Please note that the centroid is chosen as the splitting point for symmetry reasons only [41]. The split could take place at any inside point but the choice of the centroid simplifies subsequent computations.

The Clough–Tocher scheme interpolates to position and first order derivatives at the vertices and to first order cross–boundary derivatives at the midpoints of edges. Since adjoining triangles should share the same data along edges, it is advantageous to prescribe *cross–boundary normal derivatives* which are perpendicular to the edge (Equation 4.38).

We now give a description of the computation of the interpolant. For the numbering of control points please refer to Figure 4.24. The nine control vertices on the boundary of

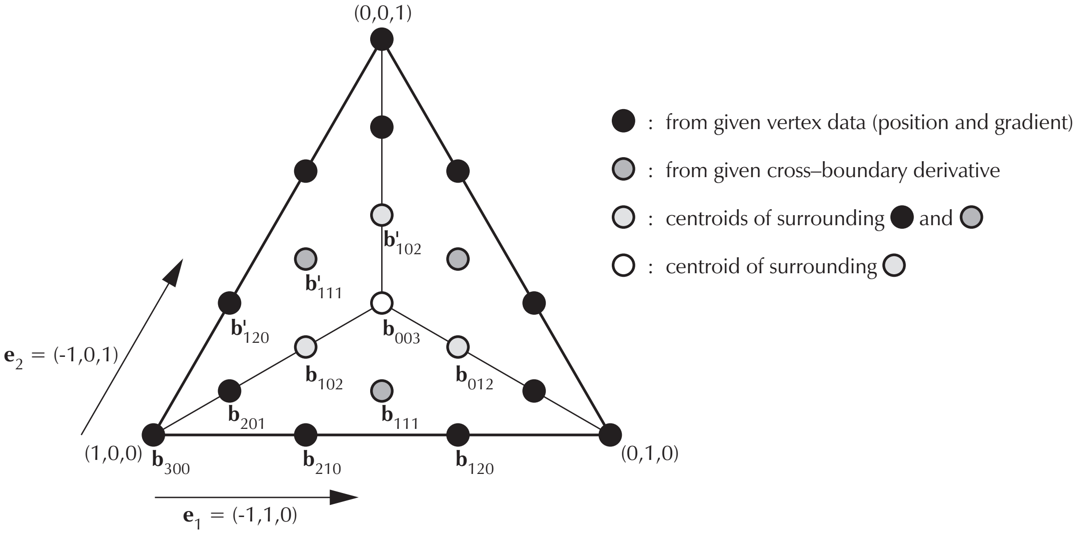


FIGURE 4.24 Clough–Tocher interpolant: numbering of control points is with respect to the lower micro–triangle.

the macro–triangle are determined exactly as for the nine parameter interpolant in Section 4.4.4. For example, $\mathbf{b}_{300}$ and $\mathbf{b}_{210}$ are computed according to Equation 4.35.

The point $\mathbf{b}_{201}$ may be determined by averaging $\mathbf{b}_{300}$, $\mathbf{b}_{210}$ and $\mathbf{b}'_{120}$. It evaluates to

$$\mathbf{b}_{201} = \frac{1}{3}(\mathbf{b}_{300} + \mathbf{b}_{210} + \mathbf{b}'_{120}) = \frac{1}{9} \cdot D_{\mathbf{e}_1}\mathbf{f}(1, 0, 0) + \frac{1}{9} \cdot D_{\mathbf{e}_1}\mathbf{f}(1, 0, 0) + \mathbf{b}_{300} \tag{4.39}$$

where we have taken advantage of choosing the centroid as the splitting point. As can be seen easily, this and the corresponding points are determined from vertex data only.

Using the cross–boundary derivative information $D_{\mathbf{d}}\mathbf{f}(\mathbf{m}_t)$ we can determine the point $\mathbf{b}_{111}$ similar to Equations 4.36 and 4.37 for the quintic C^1 interpolant. From

$$D_{\mathbf{d}}\mathbf{f}(\mathbf{m}_t) = 3 \cdot \sum_{|\mathbf{i}| = 2} \mathbf{b}^1_{ij0}(\mathbf{d}) \cdot B^2_{ij0}(\mathbf{m}_t)$$

with the normal derivative with respect to the direction $\mathbf{d}$ evaluated at the midpoint $\mathbf{m}_t$ of the edge $t = 0$ we find the control point $\mathbf{b}_{111}$ as

$$\mathbf{b}_{111} = \frac{1}{d_t}\left[\frac{1}{B^2_{011}(\mathbf{m}_t)}\left(\frac{1}{3}D_{\mathbf{d}}\mathbf{f}(\mathbf{m}_t) - \sum_{\substack{|\mathbf{i}| = 2 \\ \mathbf{i} \neq 011}} \mathbf{b}^1_{ij0}(\mathbf{d}) \cdot B^2_{ij0}(\mathbf{m}_t)\right) - d_r\mathbf{b}_{210} - d_s\mathbf{b}_{120}\right].$$

The remaining four control vertices follow from requiring C^1 continuity across the edges of micro–triangles. Again, splitting at the centroid facilitates their computation to simple averaging of three neighboring points, e.g. for $\mathbf{b}_{102}$ we have

$$\mathbf{b}_{102} = \frac{1}{3}(\mathbf{b}_{201} + \mathbf{b}_{111} + \mathbf{b}'_{111}). \tag{4.40}$$

The control point $\mathbf{b}_{003}$ finally follows from averaging its three neighbors to

$$\mathbf{b}_{003} = \frac{1}{3}(\mathbf{b}_{102} + \mathbf{b}_{012} + \mathbf{b}'_{102}). \tag{4.41}$$

In Equations 4.39, 4.40, and 4.41, we made use of the fact that the centroid was chosen as the splitting point. However, it will prove important in the following that the choice of an arbitrary splitting point does not invalidate the Clough–Tocher splitting scheme. Instead of simple averaging one has to use a weighted sum of control points with respect to the barycentric coordinate of the splitting point. It is easy to see, that averaging corresponds to a weighted sum with respect to the centroid having barycentric coordinates $(1/3, 1/3, 1/3)$.

If one is not given the cross–boundary derivative information, it is possible to estimate the derivatives using a process known as *condensation of parameters* [41]. At the endpoints of each edge of the macro–triangle one can easily compute the cross–boundary derivative. The corresponding derivative at the edge midpoint may now be set to be the average of those values. Visually smoother patch transitions can be achieved using the methods proposed in [39, 40].

Other Splits. Many other split–triangle interpolants have been proposed in the literature. An interesting split is the *Powell–Sabin split* which produces C^1 piecewise quadratic interpolants to C^1 data given at the vertices of a triangulation [110]. The macro–triangle is split into six micro–triangles by joining the center of the circumscribed circle to the edge midpoints. If one of the angles of the macro–triangle exceeds 75 degrees, a more complicated split into twelve micro–triangles is performed. The distinction between the two cases is heuristic and Farin in [41] states that it is unnecessary if the center of the inscribed circle, i.e. the incenter, is chosen as the splitting point. We will consider this argument in Section 4.5.3.

Alfeld gives a construction of a C^2–continuous interpolant to C^2 data [1]. By iterating the Clough–Tocher split, i.e. subjecting the micro–triangles of the split to another split of the same nature, it is possible to construct an interpolant which is piecewise quintic and overall C^2. Although the split satisfies the requirement of achieving the same degree of smoothness as the order of given data, the resulting surfaces often show unwanted oscillations.

4.5 MULTIVARIATE BARYCENTRIC BÉZIER PATCHES

So far we have been considering parametric curves and surfaces, i.e. mappings $\mathbb{R} \to \mathbb{R}^3$ and $\mathbb{R}^2 \to \mathbb{R}^3$. However, it is possible to generalize triangular patches to the case $\mathbb{R}^3 \to \mathbb{R}^3$ and even to N dimensions.

4.5.1 Extension to *N* Dimensions

Given the barycentric coordinates τ of a point $\mathbf{P}$ within an N–simplex P_N, the point evaluates to

$$\mathbf{P} = \sum_{i=0}^{N} \tau_i \cdot \mathbf{T}_i \qquad \mathbf{P}, \mathbf{T}_i \in \mathbb{R}^N$$

where the barycentric coordinates τ_i are required to sum up to 1, $|\tau| = 1$, and the $\mathbf{T}_i$ span an N–simplex in $\mathrm{I\!R}^N$. Bernstein polynomials scale equally well to higher dimensions. For *N–dimensional Bernstein polynomials* of degree *n* we find

$$B^n_\lambda(\tau) = \frac{n!}{\lambda!} \cdot \tau^\lambda \qquad |\lambda| = n \tag{4.42}$$

where $\lambda! = \lambda_0! \cdot \lambda_1! \cdot \ldots \cdot \lambda_N!$ and $\tau^\lambda = \tau_0^{\lambda_0} \cdot \tau_1^{\lambda_1} \cdot \ldots \cdot \tau_N^{\lambda_N}$. Consequently we have for an *N–dimensional Bézier patch* of degree *n*

$$\mathbf{b}^n(\tau) = \sum_{|\lambda| = n} \mathbf{b}_\lambda \cdot B^n_\lambda(\tau). \tag{4.43}$$

The repeated barycentric linear interpolation of the *de Casteljau algorithm* for degree *n* patches in *N* dimensions converts to

$$\mathbf{b}^l_\lambda(\tau) = \sum_{i=0}^{N} \tau_i \cdot \mathbf{b}^{l-1}_{\lambda + \mathbf{e}_i}(\tau) \qquad \begin{array}{l} |\lambda| = n - l \\ \mathbf{b}^0_\lambda = \mathbf{b}_\lambda \end{array}$$

where the $\mathbf{e}_i$ are unit vectors defined as $\mathbf{e}_i = (\delta_{i0}, \delta_{i1}, \ldots, \delta_{iN})$ with δ_{ij} being the Kronecker delta. A comparison of the bivariate de Casteljau iteration of Equation 4.24 with the above N–dimensional expression helps in understanding the introduced notation of indices. Analogous to the bivariate case, the intermediate points provide the control vertices of the subdivided patch. Further, in analogy to Equation 4.28, the algorithm again lends itself well to evaluate directional derivatives. Figure 4.25a illustrates the algorithm for the tetrahedral trivariate case.

It is interesting to notice a property of multivariate Bernstein–Bézier patches. Let *inner* Bézier control vertices denote control points $\mathbf{b}_\lambda$ with all $\lambda_i \neq 0$. The outer control points consequently denote points on the faces of the simplex. We can state that, for a given *N*, all Bézier patches $\mathbf{b}^n(\tau)$ of degree *n* have no inner control points for $n \le N$. This follows trivially, since $|\lambda| = n$ implies $\lambda_i = 0$ for some *i* if $n \le N$. Thus, for example, cubics have no inner control points for $N \ge 3$ (see Figure 4.25b).

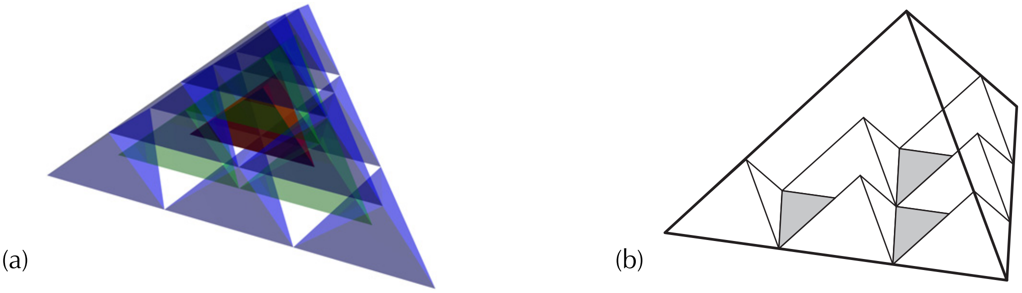


FIGURE 4.25 (a) Visualization of the trivariate de Casteljau algorithm on the example of a cubic patch. (b) Cubic Bézier patches have no inner control points in $N \ge 3$ dimensions.

In the following, we will focus on the properties of trivariate Bernstein–Bézier polynomials which are important for the construction of volumetric C^1–continuous interpolants. For a more complete and theoretical treatment of the representation of polynomials in N variables as Bernstein polynomials the reader is referred to [30].

For studying C^m, $m \le n$ continuous transitions between two N–dimensional simplices consider two simplices T_0 and T_1 sharing the common face S_{01} at $\tau_0 = 0$ which is an $N-1$–dimensional simplex. We in analogy to the bivariate case of Equation 4.34 find

$$\mathbf{c}_{\lambda_{\lambda_0 = l}} = \mathbf{c}_{l,\lambda_1,\ldots,\lambda_N} = \mathbf{b}^{l}_{\lambda_{\lambda_0 = 0}}(\sigma) = \mathbf{b}^{l}_{0,\lambda_1,\ldots,\lambda_N}(\sigma) \qquad \begin{aligned} l &= 0,\ldots,m \\ \left|\lambda_{\lambda_0 = 0}\right| &= n-l \\ \left|\lambda_{\lambda_0 = l}\right| &= n \end{aligned} \tag{4.44}$$

where σ corresponds to the barycentric coordinate of the vertex of T_1 opposite S_{01} with respect to T_0.

In order to illustrate Equation 4.44 more clearly, we demonstrate the construction of a C^1–continuous patch transition between two cubic tetrahedral patches. On the one hand, the case $N = 3$ is perhaps of greatest practical interest, on the other hand, it is this type of patch transition that will be used in subsequent parts of the thesis. Further, considering the three–dimensional case is particularly illuminating given that, of necessity, the figures may only cover that and the case $N = 2$. In the tetrahedral setting, we refer to the barycentric coordinates of a point as $\tau = (r, s, t, q)$ and we address the control vertices of a patch using $\lambda = (i, j, k, l)$.

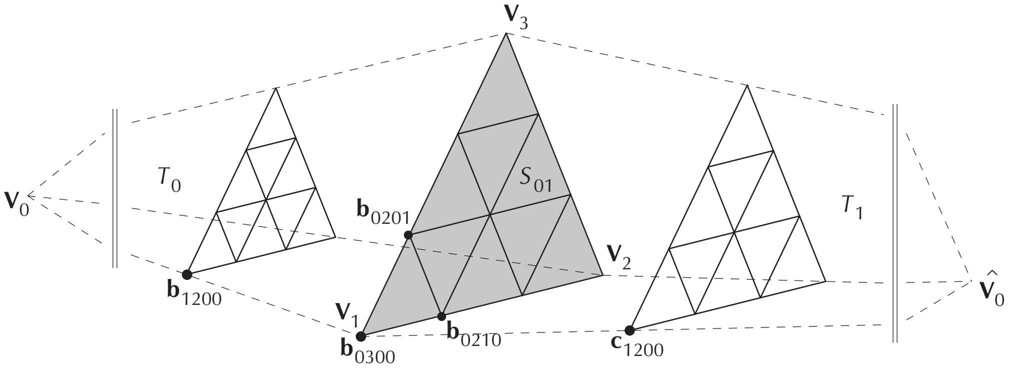


FIGURE 4.26 Construction of a C^1–continuous patch transition between two tetrahedral cubic patches $T_0 = (v_0, v_1, v_2, v_3)$ and $T_1 = (\hat{v}_0, v_1, v_2, v_3)$ sharing the common face $S_{01} = (v_1, v_2, v_3)$ at $r = 0$.

We again consider two simplices, the tetrahedra $T_0 = (v_0, v_1, v_2, v_3)$ and $T_1 = (\hat{v_0}, v_1, v_2, v_3)$, sharing a common face $S_{01} = (v_1, v_2, v_3)$ at $r = 0$. Please refer to Figure 4.26 for an illustration.

The barycentric coordinate $\hat{\sigma}$ of $\hat{v_0}$ follows analogously to Equation 4.2 to

$$\hat{r} = \frac{\Delta_{\hat{\mathbf{v}}_0,\mathbf{v}_1,\mathbf{v}_2,\mathbf{v}_3}}{\Delta_{\mathbf{v}_0,\mathbf{v}_1,\mathbf{v}_2,\mathbf{v}_3}}, \qquad \hat{s} = \frac{\Delta_{\mathbf{v}_0,\hat{\mathbf{v}}_0,\mathbf{v}_2,\mathbf{v}_3}}{\Delta_{\mathbf{v}_0,\mathbf{v}_1,\mathbf{v}_2,\mathbf{v}_3}}, \qquad \hat{t} = \frac{\Delta_{\mathbf{v}_0,\mathbf{v}_1,\hat{\mathbf{v}}_0,\mathbf{v}_3}}{\Delta_{\mathbf{v}_0,\mathbf{v}_1,\mathbf{v}_2,\mathbf{v}_3}}, \qquad \hat{q} = \frac{\Delta_{\mathbf{v}_0,\mathbf{v}_1,\mathbf{v}_2,\hat{\mathbf{v}}_0}}{\Delta_{\mathbf{v}_0,\mathbf{v}_1,\mathbf{v}_2,\mathbf{v}_3}} \tag{4.45}$$

where $\Delta_{a,b,c,d}$ refers to the signed volume of the tetrahedron spanned by the vertices *a*, *b*, *c*, and *d*. The volume of such a tetrahedron is positive if the points *b*, *c*, and *d* constitute a right–hand system with respect to *a*, and negative otherwise.

The C^1 condition for cubic tetrahedral patches follows from Equation 4.44 to

$$\mathbf{c}_{mjkl} = \mathbf{b}^m_{0jkl}(\hat{\sigma}) \qquad \begin{aligned} m &= 0, \ldots, 1 \\ 0 + j + k + l &= 3 - m \\ m + j + k + l &= 3 \end{aligned}$$

In accordance with the C^0 condition, this is equivalent to coinciding control vertices $\mathbf{c}_{0jkl}$ on the common face. In addition, the control vertices $\mathbf{c}_{1jkl}$ on the first layer from S_{01} inside T_1 follow as intermediate results from the first de Casteljau step in evaluating $\mathbf{b}^3(\hat{\sigma})$. This can more legibly be stated as

$$\mathbf{c}_{1,j,k,l} = \hat{r}\mathbf{b}_{1,j,k,l} + \hat{s}\mathbf{b}_{0,j+1,k,l} + \hat{t}\mathbf{b}_{0,j,k+1,l} + \hat{q}\mathbf{b}_{0,j,k,l+1}\,. \tag{4.46}$$

We will refer to Equation 4.46 as the C^1 or *hyperplane condition*. The term hyperplane is due to the analogy to the bivariate case: in Figure 4.22a, the control points across the C^1–continuous patch transition are required to lie on common tangent planes and thus form coplanar triangles. A *hyperplane* can thus be seen as a multidimensional tangent plane on which the adjacent control points around a corner or an edge must lie in order to form a C^1–continuous patch transition. In three dimensions, the four points on the right hand side of Equation 4.46 are said to *define* the corresponding hyperplane.

4.5.2 Generalized Clough–Tocher Split in *N* dimensions

In contrast to the more basic results above, the interpolants we have considered so far do not equally well generalize to *N* dimensions. Only the simplest interpolant, the cubic nine parameter interpolant, can be extended in an obvious recursive way. This can easily be seen by recalling the observation that cubics do not have inner control points in $N \geq 3$ dimensions. Thus a generalized nine parameter interpolant over an *N*–dimensional simplex P_N can be found recursively by constructing generalized nine parameter interpolants over its $N + 1$ $(N - 1)$–dimensional faces P_{N-1}. Since there are no inner control points we are already done when we arrive at the construction of the interpolants over the four two–dimensional triangular faces of a tetrahedron. In complete analogy to the bivariate case, this procedure results in only a C^0–continuous global interpolation, although C^1 data at the vertices was interpolated to. Consequently, for C^1 continuity one has to generalize the Clough–Tocher interpolant.

Alfeld extended the bivariate Clough–Tocher scheme to trivariate data by splitting each tetrahedron into four sub–tetrahedra. To this aim each vertex of the tetrahedron is connected to its centroid [2]. However, in order to obtain a globally C^1–continuous interpolant it is necessary to use piecewise quintic polynomials. Analogously to the quintic interpolant in Section 4.4.4, this approach bears the disadvantage that second order derivative information must be provided at the vertices of the tessellation. Furthermore, the given data does not uniquely determine the interpolant and a generalization to *N* dimensions results in interpolants of very high polynomial degree [41].

In [156], Worsey and Farin make the observation that the way in which the *N*–dimensional simplices are split is crucial to the interpolation process. They propose an inductive splitting scheme which is systematically built up from the two–dimensional Clough–

Tocher split. Using that splitting procedure they derive an *N–dimensional Clough–Tocher interpolant.* The interpolant is discussed in complete generality in *N* dimensions in [156]. Before presenting the application to three dimensions in Section 4.5.3, we in the following sketch the *generalized Clough–Tocher split* in *N* dimensions.

Worsey considers the problem of constructing a C^1 piecewise cubic interpolant to data defined over a tessellation of *N*–simplices in $\mathbb{R}^N$. The given data are position and gradient at every vertex and $N - 1$ directional derivatives at mid–edge points of the tessellation. The directions of the mid–edge derivatives are chosen such as to provide a basis for $\mathbb{R}^N$ together with the direction of the edge. Further, the derivative in the direction of the edge is fixed by the vertex data analogously to the Hermite interpolation along edges of the bivariate interpolants introduced in Sections 4.4.4 and 4.4.5.

Considering an element of the tessellation, an *N*–simplex P_N, the inductive splitting scheme proceeds systematically by increasing the dimension of the simplices to be split from 2 to *N*. Starting at triangular faces P_2 which are split at any interior point according to the Clough–Tocher split presented in Section 4.4.5, the scheme then moves to the next higher dimensional simplex P_3, a tetrahedron having four now split faces P_2. By connecting every point on the boundary (eight in total) to any interior point, the tetrahedron P_3 is split into twelve sub–tetrahedra.

In general, splitting an *M*–simplex P_M involves first splitting all its boundary faces P_{M-1} according to the lower dimensional scheme, and then joining every point on the boundary of P_M to any interior point. With $M = N$, the process subdivides P_N into $(N+1)!/2$ sub–simplices. We will see in Section 4.5.3, that the choice of the interior splitting points is crucial in the build–up of a globally C^1 interpolant.

4.5.3 A Three–dimensional Clough–Tocher Split Interpolant

In this section we will address the problem of finding a globally C^1–continuous trivariate interpolant to position and gradient information on a tessellation of tetrahedra. We will show that it is possible to find a unique piecewise cubic C^1 interpolant over the generalized Clough–Tocher split of a single tetrahedron. Further, we will investigate the restrictions imposed on the splitting points in order to solve the general interpolation problem where the tessellation consists of several tetrahedra, each of which is subdivided according to the generalized Clough–Tocher split.

Problem Statement. We consider the problem of constructing a C^1 piecewise cubic interpolant to data defined on a tetrahedralization in $\mathbb{R}^3$. The given data are position and derivatives along the three adjoining edges at every vertex as well as two mid–edge directional derivatives on every edge. The directions are chosen to be perpendicular to the edge and to lie in the respective adjoining faces. This data is such that it meets the above–mentioned criteria of Worsey for the *N*–dimensional Clough–Tocher interpolant. Figure 4.27 exemplifies the situation on one triangular surface of a tetrahedron. We will refer to these pieces of data as the *Hermite data stencil* as opposed to the *FEM data stencil* which will be introduced in Section 5.5.1.

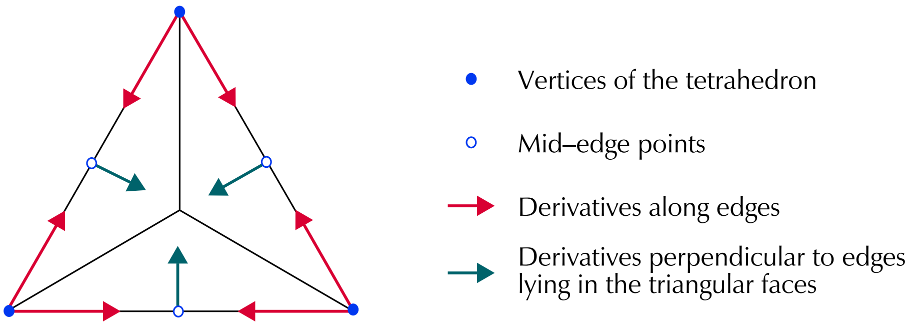

Vertices of the tetrahedron
Mid–edge points
Derivatives along edges
Derivatives perpendicular to edges lying in the triangular faces

FIGURE 4.27 Hermite data stencil: interpolated pieces of data comprise position at vertices as well as derivatives along edges and perpendicular to edges (only one triangular face of a tetrahedron is shown).

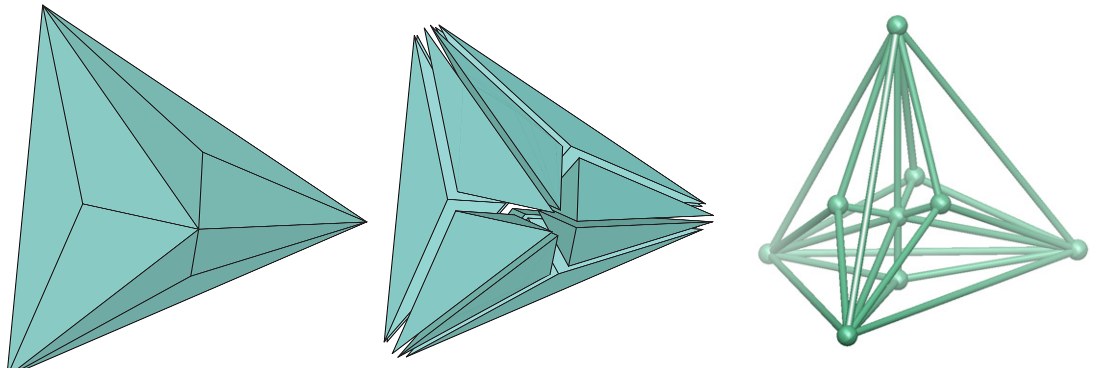

(a) (b) (c)

FIGURE 4.28 Generalized Clough–Tocher split on a tetrahedron: (a) surface, (b) micro–tetrahedra, (c) wireframe view.

C^1 Piecewise Cubic Interpolant on a Single Tetrahedron. As a first step, the macro–tetrahedron is split into twelve micro–tetrahedra according to the generalized Clough–Tocher split sketched above. Figure 4.28 shows the resulting subdivision of a tetrahedron. In a second step, one has to find piecewise cubic Bézier polynomials which interpolate to the given data. Similar to [156], we develop the local interpolant in a step–by–step manner.

First we can make the following observation:

> ***Theorem 4.1.*** Let $\mathbf{q}_i$ represent quadratic polynomials defined over the sub–simplices resulting from splitting a triangle (Clough–Tocher split) or a tetrahedron at any interior point. If these quadratics form a continuously differentiable function they reduce to a single quadratic polynomial over the original simplex.

A proof of the theorem follows from the continuity conditions between adjacent sub–simplices when we represent the quadratics in Bézier form. By prescribing the Bézier control points in any sub–simplex, we in the remaining sub–simplices fix both the control points on the common boundary and the control points on the first layer from it due to C^1 conditions across the boundary. In addition, the remaining control point opposite the

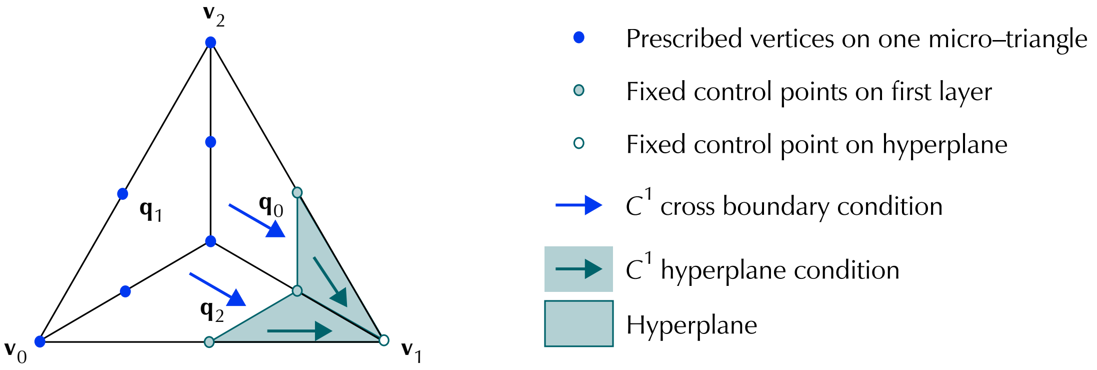

v2
q1
q0
q2
v0
v1
Prescribed vertices on one micro–triangle
Fixed control points on first layer
Fixed control point on hyperplane
C1 cross boundary condition
C1 hyperplane condition
Hyperplane

FIGURE 4.29 Prescribing the control vertices of a quadratic Bézier polynomial ($\mathbf{q}_1$) over one Clough–Tocher micro–triangle fixes the control vertices on the remaining two micro–triangles ($\mathbf{q}_0$, $\mathbf{q}_2$) if the quadratics are to evaluate to a continuously differentiable function over the macro–triangle.

first sub–simplex is fixed as it has to lie on the hyperplane spanned by the adjoining control vertices. Figure 4.29 illustrates this reasoning for the two–dimensional case.

Expanding the above reasoning to a tetrahedron which is split according to the generalized Clough–Tocher split yields:

> ***Theorem 4.2.*** Let T be a tetrahedron with corner vertices $(\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$. By splitting T according to the inductive buildup of the generalized Clough–Tocher split, we generate opposite the vertex $\mathbf{v}_i; i = 0, \ldots, 3$, the sub–tetrahedra $T_{i,j}; i = 0, \ldots, 3; j = 0, \ldots, 2$. Over each sub–simplex let $\mathbf{q}_{i,j}$ define a quadratic polynomial.
>
> If $\mathbf{q}_{i,j}; i = 0, \ldots, 3; j = 0, \ldots, 2$ form a continuously differentiable function on T, then they reduce to one globally defined quadratic polynomial $\tilde{\mathbf{q}}$.

In order to prove Theorem 4.2 we take an inductive approach corresponding to the nature of the splitting procedure: by splitting the triangular surfaces at any interior point and defining quadratic polynomials over the four surfaces, we on each surface are in the situation of Theorem 4.1. We now look at the quadratics $\mathbf{q}_{i,j}; j = 0, \ldots, 2$ defined over the sub–tetrahedra $T_{i,j}$ for any fixed i. The restriction of the now trivariate function to the surface opposite $\mathbf{v}_i$ obviously meets the criteria of Theorem 4.1. Moreover, prescribing the control points on the interior of $T_{i,j}$ for any j fixes the control points in the remaining two $T_{i,j}$ due to the continuity requirements imposed in both theorems. Therefore, we can state that

$$\mathbf{q}_{i,0} \equiv \mathbf{q}_{i,1} \equiv \mathbf{q}_{i,2} \equiv \mathbf{q}_i$$

for any fixed i.

Now, applying Theorem 4.1 to the polynomials $\mathbf{q}_i; i = 0, \ldots, 3$ yields

$$\mathbf{q}_0 \equiv \mathbf{q}_1 \equiv \mathbf{q}_2 \equiv \mathbf{q}_3 \equiv \tilde{\mathbf{q}}$$

as required.

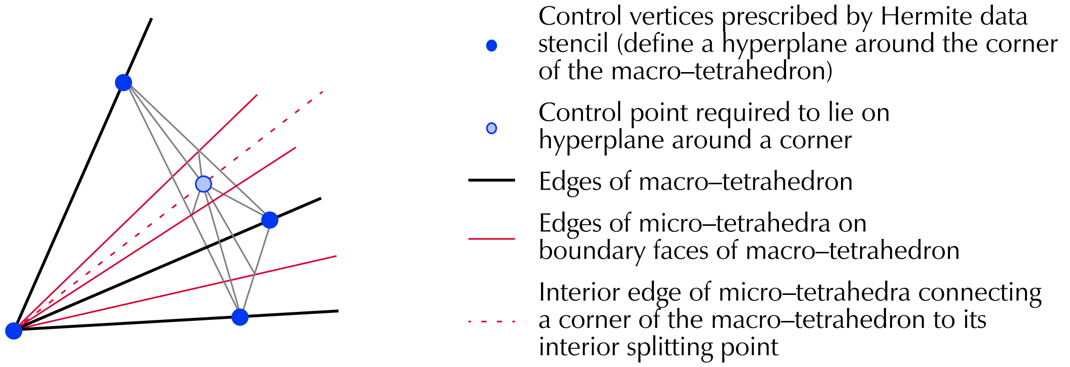

● Control vertices prescribed by Hermite data stencil (define a hyperplane around the corner of the macro–tetrahedron)
○ Control point required to lie on hyperplane around a corner
— Edges of macro–tetrahedron
— Edges of micro–tetrahedra on boundary faces of macro–tetrahedron
- - - Interior edge of micro–tetrahedra connecting a corner of the macro–tetrahedron to its interior splitting point

FIGURE 4.30 Hyperplane condition at the corners of a Clough–Tocher split macro–tetrahedron.

We now apply Theorem 4.2 in order to prove the following:

> ***Theorem 4.3.*** If a tetrahedron T is split according to the generalized Clough–Tocher split, we can define a cubic polynomial over each of the twelve micro–tetrahedra to obtain a unique C^1–interpolant to the Hermite data stencil given on T.

Again, the proof of the theorem follows the inductive nature of the construction. We know that the theorem holds for $N = 2$ because in the bivariate case the construction corresponds to the standard Clough–Tocher split (see Section 4.4.5). Further, the restriction of the Hermite data stencil to one triangular surface of the tetrahedron exactly corresponds to the data stencil in the bivariate setting. Taking the bivariate case as the inductive hypothesis, we only have to show one induction step for $N = 3$.

The cubic polynomials over the micro–tetrahedra are represented in Bernstein–Bézier form. Then, from the inductive hypothesis, all the control points on the surface of the split tetrahedron T are already fixed. Further, some of the control points on the interior of T are also determined by the data stencil. The outermost interior control vertices lying on the edges connecting the interior splitting point to any corner are uniquely determined by the gradient data, i.e. the hyperplane at the corresponding corner (see Figure 4.30). In addition, the control points on the interior of the triangular faces spanned by the edges of T and its interior splitting point are fixed, too: due to the derivative information at mid–edge points in the Hermite data stencil they are required to lie on corresponding hyperplanes around the edges (see Figure 4.31).

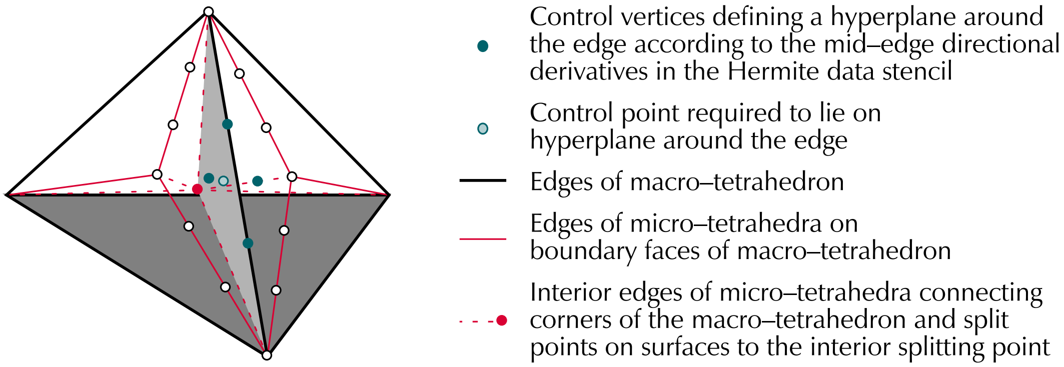

● Control vertices defining a hyperplane around the edge according to the mid–edge directional derivatives in the Hermite data stencil
○ Control point required to lie on hyperplane around the edge
— Edges of macro–tetrahedron
— Edges of micro–tetrahedra on boundary faces of macro–tetrahedron
- - -● Interior edges of micro–tetrahedra connecting corners of the macro–tetrahedron and split points on surfaces to the interior splitting point

FIGURE 4.31 Hyperplane condition around an edge of a Clough–Tocher split macro–tetrahedron.

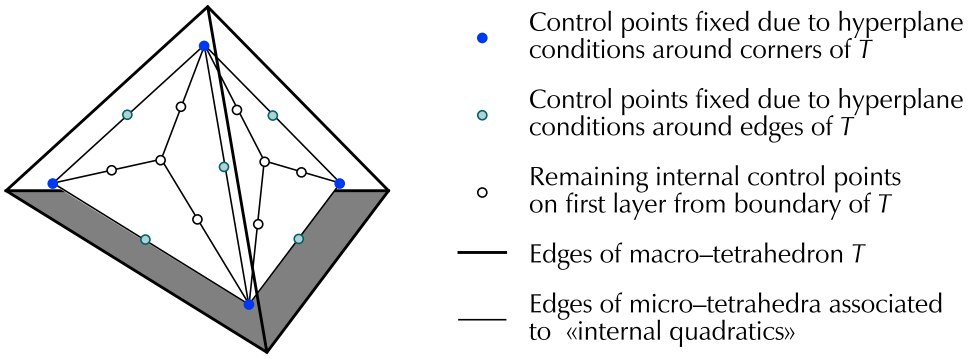

● Control points fixed due to hyperplane conditions around corners of T
○ Control points fixed due to hyperplane conditions around edges of T
○ Remaining internal control points on first layer from boundary of T
Edges of macro–tetrahedron T
Edges of micro–tetrahedra associated to «internal quadratics»

FIGURE 4.32 Micro–tetrahedra within macro–tetrahedron T after peeling away the outer shell of control points.

If we peel away the outer shell of control points on the boundary of T we can examine the trivariate quadratic polynomials associated with the remaining interior control points (see Figure 4.32). In order for the cubics associated with the micro–tetrahedra to be C^1 across boundary faces it is necessary that these quadratics are also [30, 156]. From Theorem 4.2, we know that these quadratics reduce to one global quadratic which in fact is determined by the control points already fixed due to hyperplane conditions either around corners or edges of T (see Figure 4.32). As a consequence, the unknown control points are easily and uniquely determined by subdividing the global quadratic with respect to the barycentric coordinate of the internal splitting point of the macro–tetrahedron T. This concludes the proof.

With the construction used to prove Theorem 4.3 we have found a unique C^1 interpolant $\mathbf{Q}$ to the Hermite data stencil on a single tetrahedron. We did not put any restriction on the choice of splitting points neither on the surface nor in the interior. In a next step we will investigate the construction of a C^1 interpolant to data given on an arbitrary tessellation of tetrahedra.

A Global C^1 Interpolant. Let T_0 and T_1 denote two adjacent tetrahedra sharing the common face S_{01}. From Theorem 4.3, we know how to define a piecewise C^1 interpolant to the Hermite data stencil on each of the tetrahedra. At the corners and along adjacent edges the interpolant is obviously C^1. In order to achieve a globally C^1 interpolant we in addition require the interpolant to be continuously differentiable across the common face S_{01}. It turns out that putting a restriction on the choice of splitting points suffices to comply with the requirement.

> ***Theorem 4.4.*** Let T_0 and T_1 be two adjacent tetrahedra split according to Theorem 4.3. Let further $\mathbf{p}_{01}$ denote the splitting point on the common face S_{01} and $\mathbf{p}_0, \mathbf{p}_1$ denote the interior splitting points of T_0 and T_1, respectively. Then the cubic interpolant $\mathbf{Q}$ is C^1 across S_{01} if $\mathbf{p}_0, \mathbf{p}_1$ and $\mathbf{p}_{01}$ are collinear (see Figure 4.33).

In order to prove the theorem, we first make the following observation: the face S_{01} is split according to the bivariate Clough–Tocher split. In each tetrahedron T_0 and T_1, it is composed of three triangular faces which are part of the micro–tetrahedra

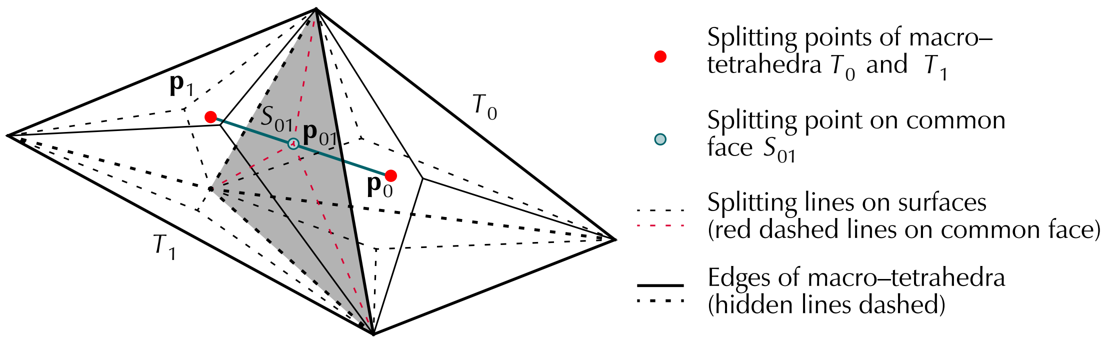

p1
S01
p01
p0
T0
T1
Splitting points of macro–tetrahedra T0 and T1
Splitting point on common face S01
Splitting lines on surfaces (red dashed lines on common face)
Edges of macro–tetrahedra (hidden lines dashed)

FIGURE 4.33 Collinear splitting points in adjacent tetrahedra are required for a C^1–continuous patch transition across a common face of two Clough–Tocher macro–tetrahedra.

$T_{i,j}$; $i = 0, \ldots, 3$; $j = 0, \ldots, 2$ opposite vertex $\mathbf{v}_i$. In order to evaluate the partial derivative

$$\left.\frac{\partial \mathbf{Q}}{\partial(\mathbf{p}_{01}-\mathbf{p}_0)}\right|_{S_{01}} \tag{4.47}$$

in T_0 restricted to the face S_{01}, one has to subtract two piecewise quadratic bivariate Bézier control nets. In order to see this we recall that the evaluation of a directional derivative of a Bézier polynomial of degree n involves taking one de Casteljau step with respect to the direction and $n-1$ steps with respect to the point of evaluation. It is due to the special direction under consideration that the first step simplifies to subtracting two quadratic control nets. The direction is along the common edge of the three micro–tetrahedra $T_{i,j}$; $j = 0, \ldots, 2$ for any fixed i. If we express this direction in barycentric coordinates of the micro–tetrahedra $T_{i,j}$, then the coordinates corresponding to $\mathbf{p}_{01}$ and $\mathbf{p}_0$ are 1 and -1, respectively, while the other two are 0. Thus the control points on the boundary of S_{01} are not involved in the directional de Casteljau step. As a consequence, we are left with quadratic sub–nets on S_{01} and the nets on the first layer from it, which, in fact, can be regarded as quadratic control nets, too. Figure 4.34 exemplifies these considerations.

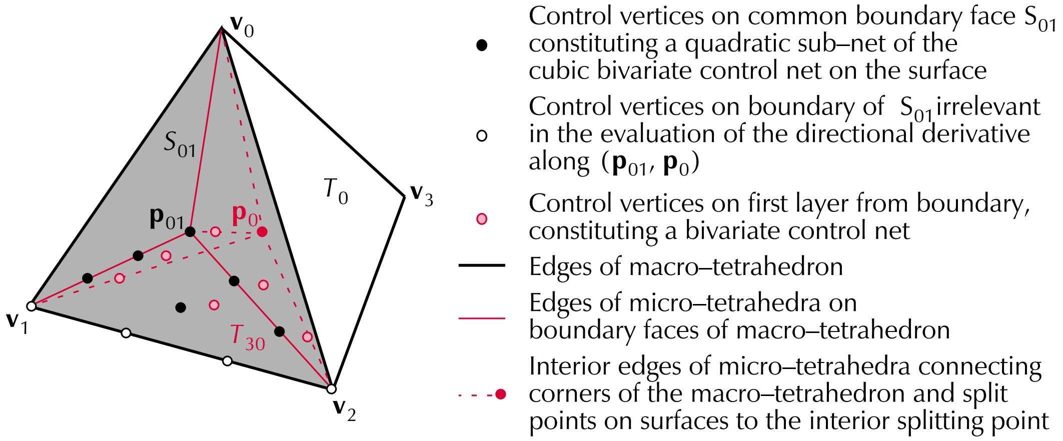

v0
S01
T0
v3
p01
p0
v1
T30
v2
Control vertices on common boundary face S01 constituting a quadratic sub–net of the cubic bivariate control net on the surface
Control vertices on boundary of S01 irrelevant in the evaluation of the directional derivative along (p01, p0)
Control vertices on first layer from boundary, constituting a bivariate control net
Edges of macro–tetrahedron
Edges of micro–tetrahedra on boundary faces of macro–tetrahedron
Interior edges of micro–tetrahedra connecting corners of the macro–tetrahedron and split points on surfaces to the interior splitting point

FIGURE 4.34 Quadratic bivariate control nets involved in the evaluation of the directional derivative along $(\mathbf{p}_{01}, \mathbf{p}_0)$.

The two piecewise quadratics defined by these sub–nets are C^1 since the interpolant $\mathbf{Q}$ is. Hence, from Theorem 4.1, they must be a global quadratic on $T_{i,j}$; $j = 0, \ldots, 2$ for any fixed i. Finally, Equation 4.47 being the difference of two global quadratics is a global quadratic, too. In addition, this bivariate polynomial is uniquely determined by the derivative data of the Hermite data stencil on S_{01}.

The statement of Theorem 4.4 now depends on the equivalence of the directional derivatives in T_0 and T_1 across S_{01}

$$\left.\frac{\partial \mathbf{Q}}{\partial(\mathbf{p}_0 - \mathbf{p}_{01})}\right|_{S_{01}} = \left.\frac{\partial \mathbf{Q}}{\partial(\mathbf{p}_{01} - \mathbf{p}_1)}\right|_{S_{01}} .$$

Since $\mathbf{p}_0$, $\mathbf{p}_1$ and $\mathbf{p}_{01}$ are chosen to be collinear, the directions are the same and the result follows.

Construction Scheme. With Theorem 4.4 we have found the solution to the problem of constructing a C^1–continuous global interpolant to the Hermite data stencil. Theorem 4.5 summarizes these findings:

> ***Theorem 4.5.*** Consider a tesselation of $\mathbb{R}^3$ into tetrahedra together with a consistent Hermite data stencil on each of the tetrahedra. Given that each tetrahedron is subdivided according to the generalized Clough–Tocher split, we can find a piecewise cubic unique C^1–continuous interpolant over the micro–tetrahedra of the tesselation provided that:
>
> 1. Common faces of adjacent macro–tetrahedra are split the same way.
> 2. The splitting points on such faces must be chosen such that they are collinear with the interior splitting points of the corresponding adjacent macro–tetrahedra.

The only remaining question is about the choice of internal splitting points in order to guarantee that the straight line connecting these points pierces the common face. Choosing centroids in general does not satisfy this constraint whereas the choice of incenters (centers of the inscribed spheres) does. The barycentric coordinates of a tetrahedron's incenter are given by

$$r = \frac{F_0}{F},\quad s = \frac{F_1}{F},\quad t = \frac{F_2}{F},\quad q = \frac{F_3}{F} \text{ with } F = F_0 + F_1 + F_2 + F_3$$

where F_i denotes the area of the triangular face across corner $\mathbf{v}_i$ of the tetrahedron.

CHAPTER

5

A TETRAHEDRAL C^1 BERNSTEIN–BÉZIER FINITE ELEMENT

In Chapter 3, we did not make any assumptions about the kind of elements and shapefunctions used in the discretization of elastic materials. However, in the interpolation of strains according to Equation 3.62 we noted that the corresponding matrix contained first order derivatives of the shape functions. We in this thesis consider tetrahedral C^1–continuous shapefunctions, although, from a mathematical point of view, C^0–continuous shape functions prove adequate in the solution of linear elasticity (see Section 3.4.1). On the one hand, the use of such higher–order differentiable functions provides for visually smoother results. On the other hand, raising the level of continuity imposes additional constraints on the solution and therefore reduces the overall problem dimension.

In a classical implementation of finite element procedures, the discretization of C^1 problems necessitates the use of Hermite type shape functions and thus the incorporation of derivatives as additional degrees of freedom [125, 159]. However, in triangular tessellations in two or more dimensions Hermite type shapefunctions providing overall C^1 continuity are inherently rational [26, 79, 159]. As a consequence, integrals cannot be computed analytically anymore and the evaluation of the result is computationally more expensive.

For that reason, we in this chapter motivate the use of integral Bernstein polynomials as shape functions. We believe that the Bernstein–Bézier representation allows for increasing the level of continuity without destroying neither the approximation properties nor the locality of the solution basis. After introducing a tetrahedral C^0–continuous element we propose a cubic Clough–Tocher split tetrahedron as a C^1 finite element. To this aim, we combine the theory of globally C^1–continuous trivariate interpolation (see Section 4.5.3) with the finite element method. We will focus on the algorithmic implementation of the trivariate Clough–Tocher split and consider the implications of C^1 conditions on the finite element assembly.

5.1 INTRODUCTION

As mentioned in Section 3.5.1, we in this thesis base on tetrahedral meshes. The tetrahedron, being the trivariate simplex as is the triangle in two dimensions, offers the most in flexibility both geometrically and topologically. Further, the use of tetrahedral elements allows for irregular meshes [133, 134] which proves important for soft tissue simulation in the context of human faces.

Finite element shape functions have to meet various requirements. First, for mathematical and convergence reasons, shape functions have to be *conforming* which means that they must be in the theoretical space of solutions in order to span a discrete subspace of it. In other words, they have to meet the physical continuity requirements of the problem under consideration [125]. We have seen this argument in Section 3.1.2.

Second, for practical and numerical reasons, shape functions should meet the following criteria:

- It is advantageous to use polynomial shape functions. Such functions speed up the computations of integrals and differentials as well as the evaluation of the function itself.
- Shape functions should feature a small support. This will result in a sparse global stiffness matrix and thus allow for the application of fast solvers for the linear system of equations (see Section 3.3.2). Normally, shape functions interpolate to the value of the corresponding nodal weight while being zero everywhere else.
- Shape functions should be symmetric. This eliminates the problem that the numbering of finite element nodes – the weights for corresponding basis functions – influences the interpolated shape. In other words, the shape functions are invariant under coordinate system transformations.

5.1.1 Bernstein–Polynomials as Shape Functions

As mentioned before, we in this thesis focus on C^1–continuous shape functions. Normally, Hermite polynomial functions are employed in order to guarantee higher order continuity. We tread a different path and develop a C^1–continuous finite element representation by employing a Bernstein–Bézier representation. On the one hand, it is obvious that Bernstein polynomials meet the above requirements with the exception that they do not interpolate to inner control points but instead always sum up to one (see Sections 4.1.2 and 4.4.1). On the other hand, such a representation offers a mean to achieve C^1 continuity without introducing rational blending functions [26] and thus enables us to compute the stiffness matrices analytically (see Section 5.3). With rational functions, one would have to resort to numerical integration [77]. Although a great deal of research has been spent on the development of integration formulae for triangular domains (see e.g. [60, 29]) we believe that the exact analytical integration is advantageous.

The first appearance of C^1 Bernstein–Bézier representations in the context of finite element analysis is in the thesis of Luscher [87]. He demonstrates the use of Bernstein polynomials for one– and two–dimensional finite elements and gives the theoretical foundations of C^1–continuous patch transitions. However, in the bivariate setting, only patch transitions across a single edge are considered. The problem of overall C^1 continuity is not taken into account. We will revisit the one–dimensional construction exemplarily in Section 5.1.2.

Nawotki employed a Bernstein–Bézier representation to prove the uniqueness of the solution of the polytropic (non–isotropic) thin plate problem. She used cubic tensor product quadrilateral patches interpolating to position and partial derivatives including mixed second derivatives at the vertices of the discretization [96, 97].

Zumbusch developed symmetric hierarchical polynomials for the *h–p*–version of finite elements [161, 162]. In a comparison of these specialized shape functions with various other polynomial bases used in the context of finite elements the Bernstein polynomials proved numerically most stable in three dimensions [161].

5.1.2 A One–Dimensional Example

In order to point out the differences between the Hermite and the Bernstein–Bézier approach we recall the Poisson problem of Section 3.2 and Section 3.3. We consider a bar of length L subjected to a constant distributed load $f(t)$ and a point load R at $t = l$. Further, we impose the boundary condition $u(0) = d$ at the beginning of the bar. In correspondence with Equation 3.12 – Equation 3.14, the strong formulation of the problem is as follows:

$$\begin{aligned} -u'' &= f \quad \text{on} \quad \Omega = 0\ldots l \\ u(0) &= d \\ u'(l) &= R \end{aligned}$$

The corresponding weak formulation, and thus the starting point for the finite element analysis, follows to (Equation 3.17)

$$\int_0^l u'^2\,dt = \int_0^l fu\,dt + Ru(l)\,. \tag{5.1}$$

In a first step, we derive the expression corresponding to Equation 5.1 on a single element.

A finite element in the global coordinate t is defined by its domain $[t_i, t_{i+1}]$ as well as corresponding nodes and shape functions on the domain. Together these paraphernalia constitute the global interpolation function $u(t)$ on the element. The length h_i of the element enters the definition of polynomial shape functions and thus of the global interpolation function. For example, for linear shape functions we find corresponding to Equation 3.34 from an element point of view

$$\begin{aligned} N_0(t) &= \frac{t_{i+1} - t}{h_i} = 1 - \frac{t - t_i}{h_i} \qquad N_1(t) = \frac{t - t_i}{h_i} \qquad t \in [t_i, t_{i+1}] \\ u(t) &= N_0(t) + N_1(t) \end{aligned}$$

Transformation of the element domain to the unit domain by substituting

$$t = h_i\xi + t_i \tag{5.2}$$

yields the shape functions and the corresponding interpolation function on the reference element as

$$\begin{aligned} N_0(\xi) &= 1 - \xi,\ N_1(\xi) = \xi \\ u(\xi) &= N_0(\xi) + N_1(\xi) \end{aligned} \qquad \xi \in [0, 1]$$

In analogous manner, Equation 5.1 for a single element $[t_i, t_{i+1}]$ transforms to the reference element $[0\ldots1]$. For the substitution $t = h_i\xi + t_i$ we trivially have

$$\frac{du}{d\xi} = \frac{du}{dt}\cdot\frac{dt}{d\xi} = h_i\cdot\frac{du}{dt} \tag{5.3}$$

which yields

$$\frac{du}{dt} = \frac{1}{h_i}\cdot\frac{du}{d\xi} \qquad dt = h_i\cdot d\xi\,. \tag{5.4}$$

Using the substitution in Equation 5.2 together with the identities in Equation 5.4 we find for the left hand side of Equation 5.1

$$\int_{t_i}^{t_{i+1}}\left(\frac{du}{dt}\right)^2 dt = \int_0^1\left(\frac{1}{h_i}\cdot\frac{du}{d\xi}\right)^2 h_i\cdot d\xi = \frac{1}{h_i}\cdot\int_0^1\left(\frac{du}{d\xi}\right)^2 d\xi\,.$$

Analogously, the integral expression on the right hand side transforms to

$$\int_{t_i}^{t_{i+1}} fu\,dt = h_i\cdot\int_0^1 fu\,d\xi\,.$$

Equation 5.1 on a single element thus reads

$$\frac{1}{h}\cdot I_S = h\cdot I_F \tag{5.5}$$

with the element length *h* and the integral expressions on the unit interval

$$I_S := \int_0^1 u'^2\,d\xi \qquad I_F := \int_0^1 fu\,d\xi\,. \tag{5.6}$$

Cubic Hermite Shape Functions. Following the procedure of Schwarz in [125] we expand the solution *u* on the reference element using the complete cubic monomial basis

$$u(\xi) = c_0 + c_1\xi + c_2\xi^2 + c_3\xi^3 \qquad \xi = 0\ldots1\,.$$

We now have for I_S and I_F

$$\begin{aligned} I_S &= c_1^2 + \frac{4}{3}c_2^2 + \frac{9}{5}c_3^2 + 2c_1c_2 + 2c_1c_3 + 3c_2c_3 &&= \mathbf{c}^T\tilde{\mathbf{S}}\mathbf{c} \\ I_F &= c_0 + \frac{1}{2}c_1 + \frac{1}{3}c_2 + \frac{1}{4}c_3 &&= \tilde{\mathbf{F}}^T\mathbf{c} \end{aligned} \tag{5.7}$$

with $\mathbf{c}^T = [\,c_0\ c_1\ c_2\ c_3]$ and

$$\tilde{\mathbf{S}} = \frac{1}{30}\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 30 & 30 & 30 \\ 0 & 30 & 40 & 45 \\ 0 & 30 & 45 & 54 \end{bmatrix} \qquad \tilde{\mathbf{F}} = \frac{1}{12}\begin{bmatrix} 12 \\ 6 \\ 4 \\ 3 \end{bmatrix}.$$

In order to arrive at the Hermite formulation we have to restate the problem in the local Hermite variables $\hat{\mathbf{h}}^T = [\ u_0\ u'_0\ u_1\ u'_1\ ]$ at the endpoints of the element. The linear relationships relating $\mathbf{c}$ to $\hat{\mathbf{h}}$ and the corresponding inverse matrix $\mathbf{A}$ result from the Hermite interpolation conditions according to Equation 4.21 to

$$\begin{aligned} u_0 &= u(0) = c_0 \\ u'_0 &= u'(0) = c_1 \\ u_1 &= u(1) = c_0 + c_1 + c_2 + c_3 \\ u'_1 &= u'(1) = c_1 + 2c_2 + 3c_3 \end{aligned} \qquad \mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -3 & -2 & 3 & -1 \\ 2 & 1 & -2 & 1 \end{bmatrix}$$

From the columns of matrix $\mathbf{A}$ the Hermite shape functions (see Figure 4.10) follow to

$$\begin{aligned} N_0(\xi) &= 1 - 3\xi^2 + 2\xi^3 = (1-\xi)^2(1+2\xi) \\ N_1(\xi) &= \xi - 2\xi^2 + \xi^3 = \xi(1-\xi)^2 \\ N_2(\xi) &= 3\xi^2 - 2\xi^3 = \xi^2(3-2\xi) \\ N_3(\xi) &= -\xi^2 + \xi^3 = -\xi^2(1-\xi) \end{aligned} \tag{5.8}$$

The matrix $\tilde{\mathbf{S}}$ and vector $\tilde{\mathbf{F}}$ now transform to $\hat{\mathbf{S}}$ and $\hat{\mathbf{F}}$, respectively,

$$\hat{\mathbf{S}} = \mathbf{A}^T\tilde{\mathbf{S}}\mathbf{A} = \frac{1}{30}\begin{bmatrix} 36 & 3 & -36 & 3 \\ 3 & 4 & -3 & -1 \\ -36 & -3 & 36 & -3 \\ 3 & -1 & -3 & 4 \end{bmatrix} \qquad \hat{\mathbf{F}} = \mathbf{A}^T\tilde{\mathbf{F}} = \frac{1}{12}\begin{bmatrix} 6 \\ 1 \\ 6 \\ -1 \end{bmatrix} \tag{5.9}$$

and we now have corresponding quadratic and linear forms in $\hat{\mathbf{h}}$

$$I_S = \hat{\mathbf{h}}^T\hat{\mathbf{S}}\hat{\mathbf{h}} \qquad I_F = \hat{\mathbf{F}}^T\hat{\mathbf{h}}.$$

In order to guarantee C^1–continuous transitions between adjacent elements the corresponding nodal weights must agree. Whereas this does not pose any problem for the displacement degrees of freedom u_0 and u_1, special care has to be taken for their derivative counterparts. In the definition of I_S in Equation 5.6 the derivative was taken with respect to the local variable ξ. For the assembly of adjacent elements of different length this is not appropriate. For the derivative nodal weights of adjacent elements to be meaningful, the derivatives have to be taken with respect to the global variable t.

From Equation 5.3 we see that the transition from the local Hermite nodal weights $\hat{\mathbf{h}}$ to their global counterparts $\mathbf{h}^T = [\ u_0\ du/dt|_0\ u_1\ du/dt|_1\ ]$ simply necessitates the multiplication of the second and fourth rows and columns of Equation 5.9 by h. We are now ready to state the local stiffness matrix and force vector of a Hermite finite element as

$$\mathbf{S} = \frac{1}{30h}\begin{bmatrix} 36 & 3h & -36 & 3h \\ 3h & 4h^2 & -3h & -h^2 \\ -36 & -3h & 36 & -3h \\ 3h & -h^2 & -3h & 4h^2 \end{bmatrix} \qquad \mathbf{F} = \frac{h}{12}\begin{bmatrix} 6 \\ h \\ 6 \\ -h \end{bmatrix} \tag{5.10}$$

where the factors $1/h$ and h of Equation 5.5 have been included again.

The assembly of two or more local stiffness matrices and force vectors can easily be achieved by means of the direct stiffness method, i.e. by summing up the local contributions into a global matrix and vector, respectively.

For the visualization of the resulting curve, the Hermite shape functions have to be scaled with respect to the element length. For the displacement shape functions $N_0(\xi)$, $N_2(\xi)$ this involves substituting $\xi = \xi/h$, whereas the derivative shape functions $N_1(\xi)$, $N_3(\xi)$ in addition have to be scaled by a factor of h [26]. From Equation 5.8, this scaling results in

$$\begin{aligned}
\tilde{N}_0(\xi) &= N_0(\xi/h) &&= (1-\xi/h)^2(1+2\xi/h)\\
\tilde{N}_1(\xi) &= h\cdot N_1(\xi/h) &&= \xi(1-\xi/h)^2\\
\tilde{N}_2(\xi) &= N_2(\xi/h) &&= (\xi/h)^2(3-2\xi/h)\\
\tilde{N}_3(\xi) &= h\cdot N_3(\xi/h) &&= -\xi^2/h(1-\xi/h)
\end{aligned}$$

Figure 5.1a shows the resulting displacement for the numerical values $l = 4$, $d = 1$, $f(t) = -1$, and $R = 1$. The bar is composed of three finite elements of length 1, 2, and 1, respectively. The bold gray curve depicts the exact solution of the problem. Figure 5.1b shows a schematic view of the direct stiffness assembly procedure. Dark gray entries stem from summing up entries of two local matrices and vectors, respectively. They are associated to nodal weights in adjacent elements. The first row and column correspond to the boundary condition $u(0) = d$. In order to account for homogeneous boundary conditions it is sufficient to delete corresponding rows and columns in the global system of equations. However, an inhomogeneous boundary condition causes a change of the force vector which has to be accounted for before deleting the associated row and column [125].

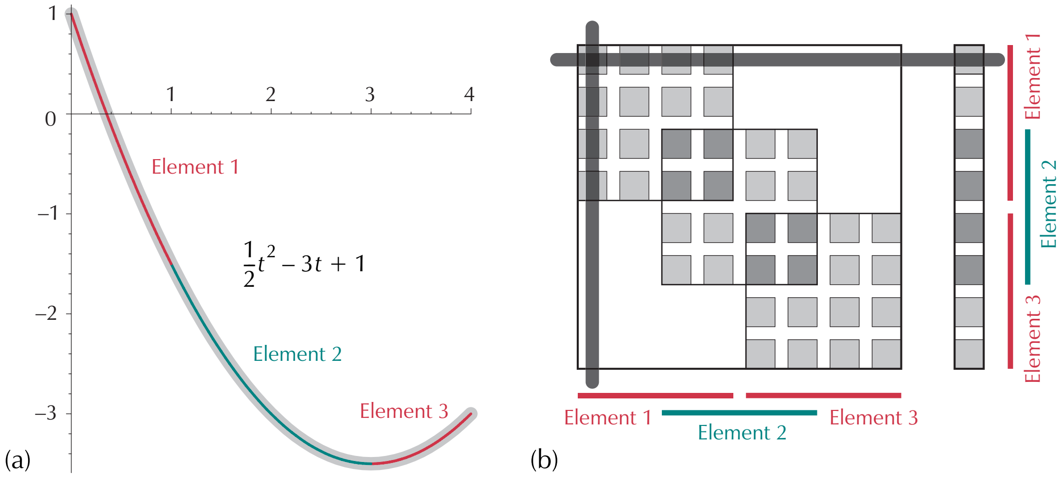


FIGURE 5.1 (a) Finite element solution of example Poisson problem using cubic Hermite shape functions.
(b) Schematic view of direct stiffness assembly and elimination of degrees of freedom corresponding to geometric boundary conditions.

Cubic Bernstein–Bézier Shape Functions. Instead of a Hermite or monomial basis we expand the solution in cubic Bernstein polynomials

$$u(\xi) = b_0(1-\xi)^3 + 3b_1(1-\xi)^2\xi + 3b_2(1-\xi)\xi^2 + b_3\xi^3 \qquad \xi = 0\dots1$$

We thus get for I_S and I_F

$$\begin{aligned} I_S &= \frac{3}{5}(3b_0^2 + 2b_1^2 + 2b_2^2 + 3b_3^2 \\ &\quad - 3b_0b_1 - 2b_0b_2 - b_0b_3 + b_1b_2 - 2b_1b_3 - 3b_2b_3) = \mathbf{b}^T\hat{\mathbf{S}}\mathbf{b} \\ I_F &= \frac{1}{4}b_0 + \frac{1}{4}b_1 + \frac{1}{4}b_2 + \frac{1}{4}b_3 \qquad = \hat{\mathbf{F}}^T\mathbf{b} \end{aligned}$$

with the finite element weights equal to the control points $\mathbf{b}^T = [\ b_0\ b_1\ b_2\ b_3\ ]$ and

$$\hat{\mathbf{S}} = \frac{1}{10}\begin{bmatrix} 18 & -9 & -6 & -3 \\ -9 & 12 & 3 & -6 \\ -6 & 3 & 12 & -9 \\ -3 & -6 & -9 & 18 \end{bmatrix} \qquad \hat{\mathbf{F}} = \frac{1}{4}\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}.$$

The local stiffness matrix and force vector corresponding to Equation 5.10 for a Bernstein–Bézier finite element follow consequently to

$$\mathbf{S} = \frac{1}{h}\cdot\hat{\mathbf{S}} \qquad \mathbf{F} = h\cdot\hat{\mathbf{F}}.$$

More effort has to be spent in the Bernstein–Bézier setting in order to arrive at a globally C^1–continuous solution. Figure 5.2a shows the Bernstein–Bézier solution of the same problem as in Figure 5.1. Further, Figure 5.2b gives a schematic view of the assembly procedure which will be discussed in the following.

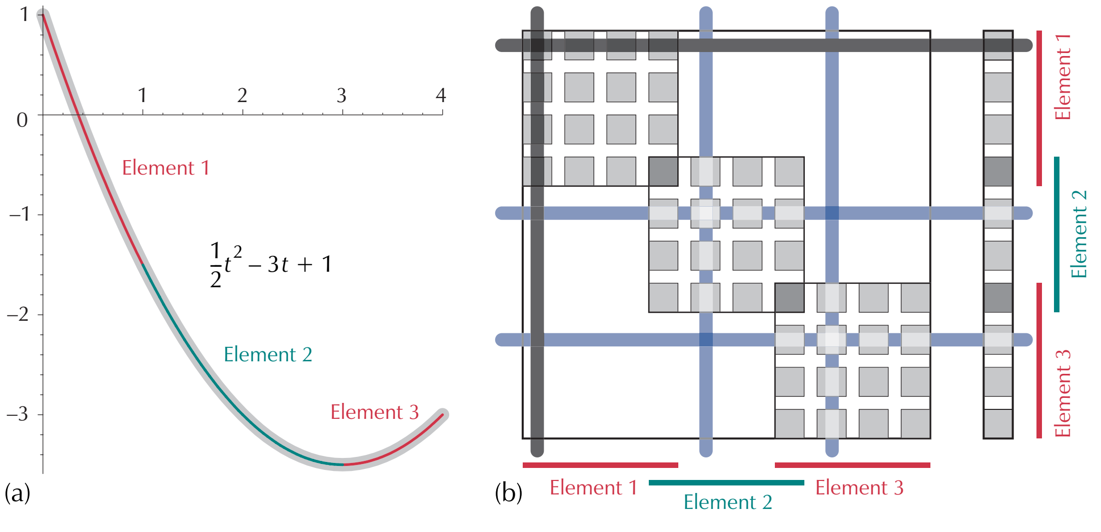


FIGURE 5.2 (a) Finite element solution of example Poisson problem using cubic Bernstein shape functions.
(b) Schematic view of C^1 assembly and elimination of degrees of freedom corresponding to geometric boundary conditions.

In a first step, by means of the direct stiffness method, a C^0 assembly is performed. Again, dark gray matrix and vector entries in Figure 5.2b are associated to degrees of freedom common to two adjacent elements. Thus their contributions have to be summed up.

In a second step, we have to enforce C^1 continuity. To this aim, we recall the C^1 condition for piecewise Bézier curves according to Equation 4.15. Two Bézier curves of respective parametric length h_0, h_1 and defined by their control polygons $\mathbf{b}_0, \ldots, \mathbf{b}_n$ and $\mathbf{b}_n, \ldots, \mathbf{b}_{2n}$, respectively, are continuously differentiable if and only if

$$\mathbf{b}_{n+1} = (1-\zeta) \cdot \mathbf{b}_{n-1} + \zeta \cdot \mathbf{b}_n \tag{5.11}$$

with

$$\zeta = \frac{h_0 + h_1}{h_0}. \tag{5.12}$$

Consequently, in the global stiffness matrix and force vector, one has to eliminate the rows and columns associated with linearly dependent control points. Equation 5.11 implies the elimination of the row and column associated to $\mathbf{b}_{n+1}$, but it is equally well possible to eliminate $\mathbf{b}_{n-1}$ or $\mathbf{b}_n$. However, the elimination of $\mathbf{b}_n$ bears the disadvantage that endpoints of intervals should be at disposal for the imposition of boundary conditions. Of course, for the elimination of $\mathbf{b}_{n+1}$ it is not sufficient to simply discard the corresponding row and column. Rather, their entries must be weighted according to Equation 5.11 and added onto the rows and columns associated to $\mathbf{b}_{n-1}$ and $\mathbf{b}_n$.

From Equation 5.11 and Equation 5.12 the elimination matrix for an assemblage of two cubic Bézier elements follows to the 7×6 matrix

$$\mathbf{M} = \begin{bmatrix} 1 & & & & & \\ & 1 & & & 0 & \\ & & 1 & & & \\ & & & 1 & & \\ & & 1-\zeta & \zeta & & \\ & 0 & & & 1 & \\ & & & & & 1 \end{bmatrix}.$$

Let $\mathbf{K}$ denote the global 7×7 stiffness matrix of a two–elements assemblage resulting from the C^0 direct stiffness method outlined above. Further, let $\mathbf{R}$ denote the corresponding global C^0 force vector. Then, the elimination of dependent Bézier weights follows to

$$\mathbf{K}_{red} = \mathbf{M}^T\mathbf{K}\mathbf{M} \qquad \mathbf{R}_{red} = \mathbf{M}^T\mathbf{R}. \tag{5.13}$$

The light blue bars in Figure 5.2b denote rows and columns associated to linearly dependent control points which have been eliminated from the global system of equations according to Equation 5.13. The imposition of boundary conditions in the Bernstein–Bézier approach is performed in the same manner as described for the Hermite setting. Consequently, the problem dimension for both approaches is the same.

In a disassembly step after solving the global system of equations, the eliminated control points have to be determined by means of Equation 5.11.

It is worth noting that the perfect match of Hermite and Bernstein solution is not accidental. In both cases, a cubic C^1 expansion, albeit in different bases, has been used [87].

5.2 TETRAHEDRAL BERNSTEIN–BÉZIER ELEMENT

We in Section 3.5 did not make any assumptions about the kind of elements used in the discretization. In this section we will introduce the notion of tetrahedral Bernstein–Bézier elements. For that purpose, we expand the solution of the trivariate elasticity problem of Section 3.4 in terms of Bernstein polynomials defined over the elements of an irregular tetrahedralization [119].

5.2.1 Interpolation of Geometry

The interpolation of the geometry of a finite element in trivariate Cartesian space essentially is given by

$$x = \sum_i x_i \cdot g_i \qquad y = \sum_i y_i \cdot g_i \qquad z = \sum_i z_i \cdot g_i$$

where the g_i are arbitrary interpolation functions and x_i, y_i and z_i describe the geometry of the element. In the tetrahedral setting, the interpolation functions are of the form

$$g_i = f(r, s, t, q)$$

with barycentric coordinates *r*, *s*, *t*, and *q*.

Depending on the type of interpolation, three different classes of elements can be distinguished.

- *Subparametric elements*
  The element geometry is interpolated to a lower degree than the displacement function.
- *Isoparametric elements*
  Element coordinates are interpolated using the same interpolation functions.
- *Superparametric elements*
  The element geometry is interpolated to a higher degree than the displacement.

We employ a subparametric formulation using a linear interpolation of element coordinates as

$$\begin{aligned} g_0(r, s, t, q) &= g_0(r, s, t) = r \\ g_1(r, s, t, q) &= g_1(r, s, t) = s \\ g_2(r, s, t, q) &= g_2(r, s, t) = t \\ g_3(r, s, t, q) &= g_3(r, s, t) = 1 - r - s - t \end{aligned}$$

which results in

$$\mathbf{v}(r, s, t) = \mathbf{v}_0 \cdot r + \mathbf{v}_1 \cdot s + \mathbf{v}_2 \cdot t + \mathbf{v}_3 \cdot (1 - r - s - t) \tag{5.14}$$

with the $\mathbf{v}_i$ representing the corners of the tetrahedral element in Cartesian coordinates.

The subparametric choice is justified both because of the assumption of small displacements and because of the linear tetrahedralization of the domain. In the linearization of geometrically nonlinear analysis where the new geometry of an element is obtained by adding the displacements to the previous geometry, such an approach would be inappropriate.

In order to evaluate the strain interpolation matrix according to Equation 3.62 we need derivatives with respect to the global Cartesian coordinates. Analogous to the bivariate triangular case discussed in Section 4.4.2, the relation of x, y, z to r, s, t, q derivatives with $q = 1 - r - s - t$ is given by the Jacobian operator **J** as

$$\begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} \end{bmatrix} = \mathbf{J}^{-1} \begin{bmatrix} \frac{\partial}{\partial r} \\ \frac{\partial}{\partial s} \\ \frac{\partial}{\partial t} \end{bmatrix} \text{ with } \mathbf{J} = \begin{bmatrix} \frac{\partial x}{\partial r} & \frac{\partial y}{\partial r} & \frac{\partial z}{\partial r} \\ \frac{\partial x}{\partial s} & \frac{\partial y}{\partial s} & \frac{\partial z}{\partial s} \\ \frac{\partial x}{\partial t} & \frac{\partial y}{\partial t} & \frac{\partial z}{\partial t} \end{bmatrix}. \tag{5.15}$$

The integration of a barycentric function $f(r, s, t)$ over an arbitrary tetrahedral domain follows after the transformation to the reference element to

$$\int_{V^{(m)}} f(r, s, t)\, dV = \int_0^1 \int_0^{(1-r)} \int_0^{(1-r-s)} f\, |\mathbf{J}|\, dt ds dr \tag{5.16}$$

with the Jacobi determinant $|\mathbf{J}|$ representing six times the volume of the tetrahedral domain of integration.

From Equation 5.15 and Equation 5.16 we see an additional advantage of the subparametric approach with linear interpolation of the geometry: the Jacobian and thus its inverse are constant. This fact will facilitate further computations.

Rewriting the linear interpolation of geometry according to Equation 5.14 in matrix form obviously yields the transformation of tetrahedral barycentric coordinates to Cartesian coordinates

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{v}_0 & \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} r \\ s \\ t \\ 1 - r - s - t \end{bmatrix}.$$

5.2.2 Definition of Bernstein–Bézier Finite Element

We are now ready to introduce the trivariate tetrahedral Bernstein–Bézier finite element. The trivariate barycentric Bernstein polynomials of degree *n* follow from Equation 4.42 to

$$B^n_{ijkl}(r, s, t, q) = \frac{n!}{i!j!k!l!} r^i s^j t^k q^l$$

and from Equation 4.43 we have the corresponding Bézier tetrahedron

$$\mathbf{b}^n(r, s, t, q) = \sum_{i, j, k, l \geq 0}^{i + j + k + l = n} \mathbf{b}_{ijkl} B^n_{ijkl}(r, s, t, q).$$

With respect to the interpolation of displacements, we will give examples for linear, quadratic and cubic Bézier elements. However, from Section 4.5.3 we know, that cubic tetrahedral elements in combination with a three–dimensional Clough–Tocher splitting scheme are required for a globally C^1–continuous displacement function.

5.2.3 Properties

In summary, the use of Bernstein-Bézier shape functions in finite element analysis bears the following advantages [119]:

- *Suitability for geometric modeling*
  The formulation of Bernstein–Bézier patches is well–studied and the inherent end-point interpolation lends itself well to finite element modeling. Further, cubic Bézier tetrahedra allow for the construction of trivariate globally C^1–continuous interpolants (Section 4.5.3).
- *Integral polynomial form of arbitrary degree*
- *Analytical derivatives and integrals*
  For integral polynomial functions $f(r, s, t, q)$ with $q = 1 - r - s - t$, the integral of Equation 5.16 can be evaluated analytically using the closed integration formula for monomials of arbitrary degree in *r*, *s* and *t*

  $$\int_0^1 \int_0^{(1-r)} \int_0^{(1-r-s)} r^i s^j t^k \, dt ds dr = \frac{i!j!k!}{(i+j+k+3)!}. \tag{5.17}$$

  This allows for an analytical evaluation of the integrals involved in the computation of local stiffness matrices and force vectors.
- *Fast subdivision*
  The De Casteljau algorithm can be used both for the visualization of the resulting surface and for fast subdivision and refinement schemes. It easily allows for a progressive refinement of the solution the like of which occurs in geometric multigrid approaches.
- *Numerically stable*
  In a comparison of several polynomial basis functions, the Bernstein polynomials performed best with respect to the condition number of local matrices for the Laplace operator, both for triangular and tetrahedral elements of different shapes [161].
- *Suitability for direct ray tracing*
  Triangular Bernstein–Bézier patches allow for direct ray tracing by means of an adapted version of the Bézier clipping algorithm [99, 117, 118] (see Chapter 6).

5.3 LOCAL STIFFNESS MATRICES AND FORCE VECTORS

Basically, the computation of the local stiffness matrices and force vectors for the problem of static elastomechanics (Section 3.4) results directly from the discretization discussed in Section 3.5. The stiffness matrices and force vectors for compressible analysis follow from Equations 3.65 and 3.66 whereas for the mixed formulation the matrices follow from the relations in Equation 3.74. The shape functions N_i and M_i in the interpolation matrices $\mathbf{B}^{(m)}, \mathbf{B}_V^{(m)}, \mathbf{B}_D^{(m)}$ and $\mathbf{H}_P^{(m)}$, respectively, are chosen to be the Bernstein polynomials $B^n_{ijkl}(r, s, t, q)$ with $q = 1 - r - s - t$.

We have implemented three different degrees of displacement interpolation: linear, quadratic and cubic. In compliance with the discussion in Section 3.5.4, the corresponding degree of pressure interpolation was chosen to be constant for the linear, and linear for the quadratic and cubic elements. The structure of the resulting stiffness matrices for the pure displacement and mixed setting is depicted in Figure 5.3.

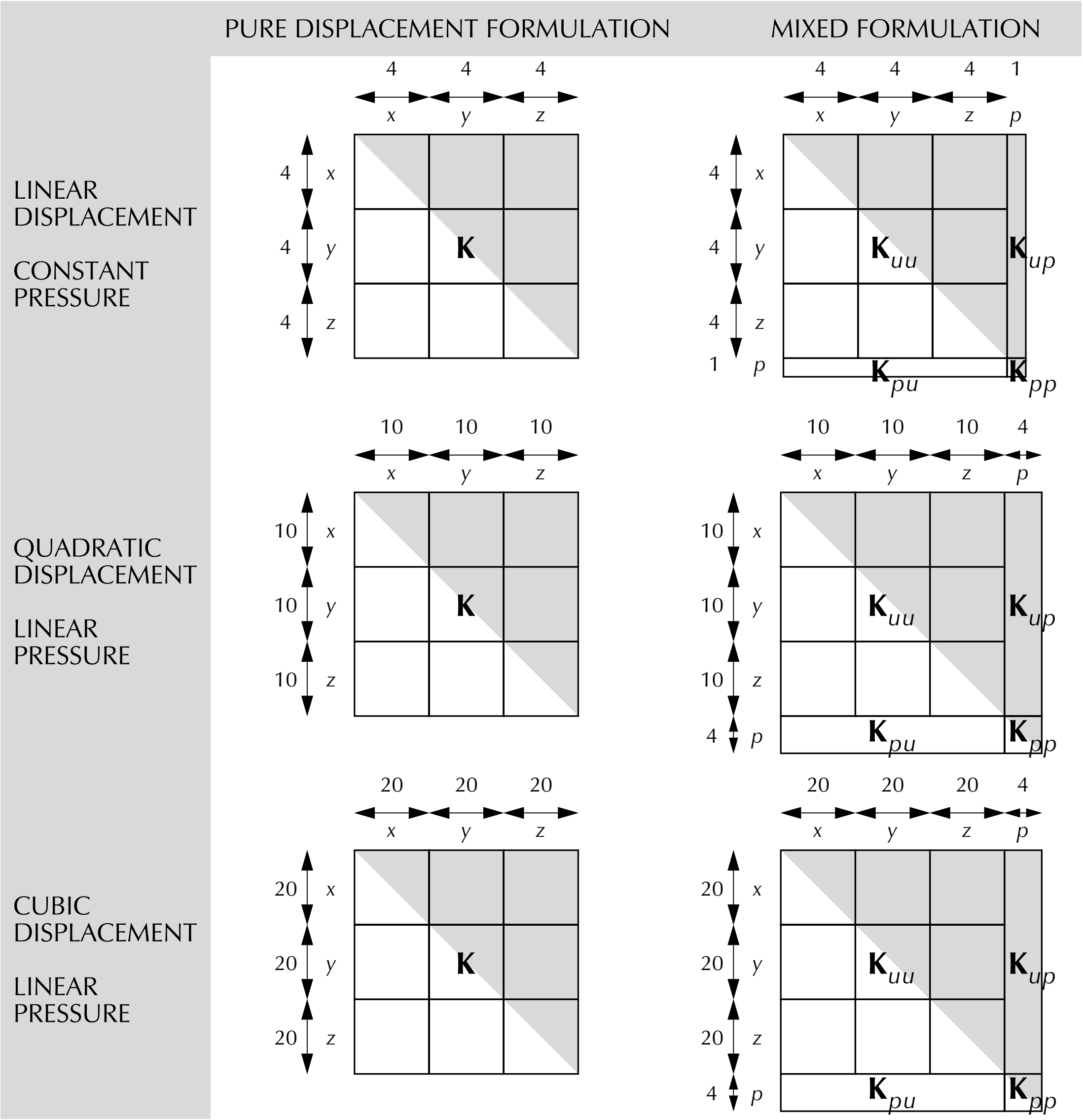


FIGURE 5.3 Structure of local stiffness matrices (due to the symmetry, only upper triangle parts need to be computed).

In contrast to scalar pressure degrees of freedom, a displacement degree of freedom is represented by its *x*–, *y*– and *z*–coordinate in the stiffness matrix and force vector.

For the special case of constant pressure in the linear element, the pressure degree of freedom pertains to only one element and thus can be eliminated from the local matrix using a process known as *static condensation* [8]. The condensed stiffness matrix follows from the Schur complement [56] to

$$\mathbf{K}_{upc} = \mathbf{K}_{uu} - \mathbf{K}_{up}\mathbf{K}_{pp}^{-1}\mathbf{K}_{pu}. \tag{5.18}$$

Hence, the degrees of freedom of the mixed element are only the displacement degrees of freedom and thus the same as in the pure displacement formulation. While static condensation does decrease the computational costs of such elements, the procedure bears the disadvantage that the case of total incompressibility cannot be modeled anymore. This is due to the fact that for $\nu = 0.5$ the submatrix $\mathbf{K}_{pp}$ is zero and cannot be inverted anymore. However, despite of static condensation, materials with ν very close to 0.5 can still be handled to sufficient accuracy and thus we chose static condensation for its efficiency.

The linear interpolation of pressure for the quadratic and cubic element allows for the calculation of a continuous pressure field and thus for the proper simulation of pressure flow between the elements of the body under consideration. Hereby, the pressure degrees of freedom pertain to the corner nodes of the tetrahedral element.

In complete analogy to the stiffness matrices, the force vectors are of length 3×4 for the linear element and of length $3 \times 10 + 4$ and $3 \times 20 + 4$ for the quadratic and cubic elements, respectively. The $+ 4$ stand for the pressure degrees of freedom in the mixed setting. The corresponding entries are zero in accordance with Equation 3.73.

The entries of the local stiffness matrices and force vectors can be computed with respect to the geometry of each element by means of Equations 5.15 and 5.16 and thanks to the Bernstein formulation, the resulting integrals can be evaluated analytically using the integration formula of Equation 5.17. As a consequence of the symmetry of the stiffness matrices, only the upper triangle parts in Figure 5.3 need to be computed.

5.4 C^0–CONTINUOUS FINITE ELEMENT

The tetrahedral Bernstein–Bézier element of Section 5.2 together with the matrices and force vectors of Section 5.3 essentially constitute the C^0–continuous finite element. In this section, we will discuss the structure and representation of the global stiffness matrix and sketch the procedure employed to compile this matrix.

The process referred to as the assembly of local stiffness matrices and force vectors into one global system of equations has already been discussed in Section 3.3.3. In essence, the contribution of every degree of freedom local to each element has to be summed up at the corresponding location in the global system. This location basically depends upon the numbering of degrees of freedom which strongly influences the structure of the resulting sparse matrix. Generally speaking, for efficient solving schemes to be applicable, the numbering ought to be such that the structure is as diagonally dominant as possible. In the one–dimensional setting of the example problem in Section 5.1.2 the best numbering is quite obvious and results in a banded global matrix. However, in a three–dimensional problem on a tetrahedral mesh of arbitrary connectivity an optimal numbering of degrees of freedom is hard to find.

5.4.1 The Structure and Representation of the Global Stiffness Matrix

We will sketch the process of assembly on the example of a mixed formulation without static condensation. The procedure for a pure displacement–based formulation follows analogously to the assembly of the submatrix $\mathbf{K}_{uu}$.

The structure of the global stiffness matrix is similar to the structure of the local matrices. Figure 5.4 illustrates the correlation. Lowercase letters stand for local entities whereas uppercase letters stand for their global counterparts.

As a basic principle, the global stiffness matrix contains *relational coefficients* $c_{I,J}$ between two degrees of freedom which we denote by their global indices *I*, *J*. The structure of the sparse global matrix is formally given by the following observation.

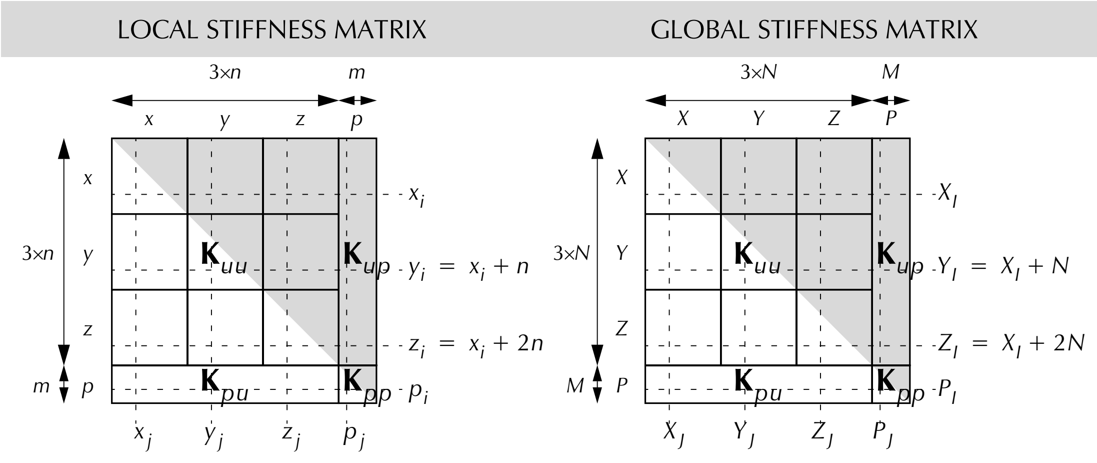


FIGURE 5.4 Correlation between local and global stiffness matrices:

n/N:	number of local/global displacement degree of freedom
m/M:	number of local/global pressure degrees of freedom
i/I:	local/global index of an example corner node
j/J:	local/global index of an other example corner node in the same element
$x_{i/j}, y_{i/j}, z_{i/j}$ / $X_{I/J}, Y_{I/J}, Z_{I/J}$:	indices of x–, y–, z–components in local/global stiffness matrix

Let T be a tetrahedral element. Further, let $A(T)$ denote the set of degrees of freedom pertaining to the element T.

The relational coefficient $c_{I,J}$ between two degrees of freedom I and J (it be pressure or displacement) is not equal to zero if and only if there exists an element T with $I \in A(T) \wedge J \in A(T)$.

This condition says that two nodes are interrelated whenever they are control point in a common element and thus have an entry in its stiffness matrix. Obviously, the condition is symmetrical, i.e. $c_{I,J} \neq 0 \Leftrightarrow c_{J,I} \neq 0$, which results in a symmetrical sparse global stiffness matrix.

We represent the sparse matrix in the *compressed row format*, a format which allows for a very small memory footprint at the cost of not allowing any modification of the matrix after its structure has been fixed. The format is very similar to the well–known column–based *Harwell–Boeing* sparse matrix format [37] and stores all non–zero entries row after row in a contiguous linear array. This array is complimented with additional structures allowing access to individual entries [6]. For symmetrical matrices the format is identical to the Harwell–Boeing format.

5.4.2 Assembly and Disassembly

The assembly in the C^0 setting is a two–step process. In a first step, the structure of the sparse global matrix is determined. For each node I, we traverse every element the node pertains to and prepare an entry for each node J in this element. Thus the structure of the matrix is built up in a row–by–row, i.e. node–by–node manner. The global indices I and J uniquely determine the position of the relational coefficient $c_{I,J}$ in the global matrix. However, we do not yet fill in the values of the $c_{I,J}$ since this would require that all local stiffness matrices be available at the same time. This, of course, for large systems is not feasible.

In a second step, by traversing all elements, the values of the $c_{I,J}$ can be inserted into the sparse matrix structure by adding up the corresponding local $c_{i,j}$. It is only now that the local stiffness matrices are computed and thus actually allocate memory.

In both steps, the symmetry of the local and global stiffness matrix can be taken advantage of. Further, the position of the entries for the y– and z–components follow from the index of the x–component (see Figure 5.4).

Displacement boundary conditions are taken into account in complete analogy to the procedure discussed in Section 5.1.2. Last but not least, the force vector is easily assembled using the correspondence between local and global indices.

Figure 5.5 shows the structure of the global stiffness matrix of an example problem using a discretization of 24 finite elements. Cubic Bernstein polynomials were used for the displacement interpolation, whereas pressure was interpolated linearly. The assembly resulted in 192 displacement and 18 pressure degrees of freedom, yielding a 594×594 stiffness matrix. The x–, y–, z–, and p–parts of the matrix structure according to Figure 5.4 are easy to identify.

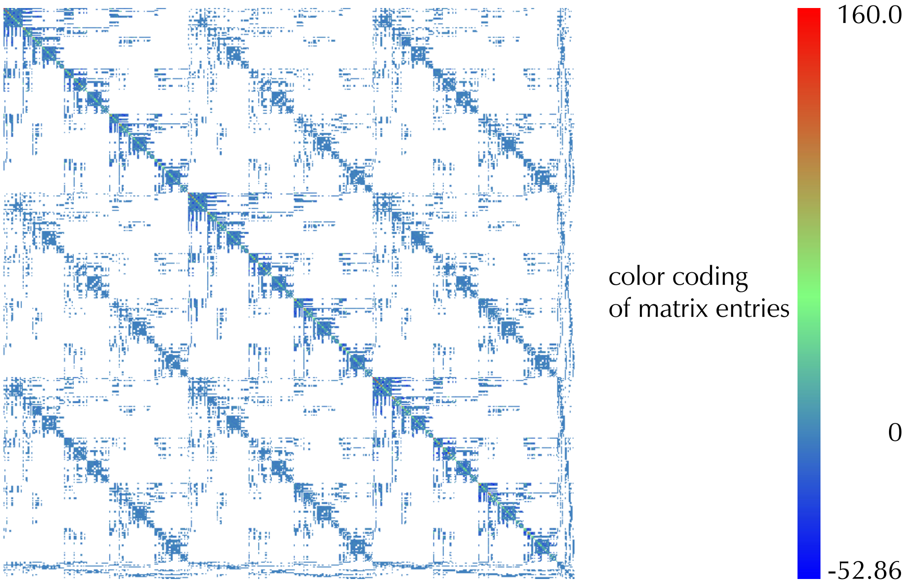


FIGURE 5.5 Structure of a global stiffness matrix for a cubic Bernstein–Bézier example problem.

The disassembly is very straightforward: for each degree of freedom the entry in the solution vector is attached to the corresponding node. Together with the boundary conditions these values determine the resulting trivariate displacement and scalar pressure function and thus the result of the simulation.

5.5 C^1–CONTINUOUS FINITE ELEMENT

In this section, we will motivate a Clough–Tocher split tetrahedron as a C^1 finite element. This involves both the definition of a globally C^1 Clough–Tocher interpolant on the input mesh (Section 5.5.3, compare Section 4.5.3) and an assembly procedure which takes into account the linear dependencies within the elements and across element boundaries (see Section 5.5.4). In the next section, we will first introduce some terminology and definitions which enter quite informally the forthcoming sections.

5.5.1 Terminology and Definitions

Micro–element, Macro–element and C^1–continuous finite element. In the following, we will distinguish between *micro–* and *macro–elements*. The term micro–element refers to a cubic Bernstein–Bézier tetrahedron, which in the C^0 setting directly corresponds to the element in the finite element sense. In contrast, a macro–element denotes the Clough–Tocher split input tetrahedron and consists of twelve micro–tetrahedra with a total of 91 control points (see Figure 4.28). The macro–element is also called the C^1–*continuous finite element* with 91 displacement degrees of freedom. In the mixed formulation, nine scalar pressure degrees of freedom are located at the corner nodes of the twelve micro–elements.

FEM Data Stencil. We introduce the *FEM data stencil* as opposed to the Hermite data stencil of Section 4.5.3. We refer to the FEM data stencil as the set of control points on a Clough–Tocher split cubic tetrahedron needed to satisfy the Hermite data stencil. We recall that the Hermite data stencil uniquely defines the macro–element. It is composed of position and derivatives along the edges at the four corner nodes of the macro–element as well as mid–edge directional derivatives on every edge of the macro–element (see Figure 4.27). As a consequence of the endpoint interpolation and derivative definition of Bézier splines, the FEM data stencil is thus composed of the four macro–element corner nodes, their neighbors along the edges and the mid–face nodes of the twelve micro–element faces on the surface of the macro–element (see Figure 5.6a).

Hyperplane. We recall the *hyperplane condition* of Section 4.5.1 given by Equation 4.46. The points $\mathbf{b}_{i,j,k,l} = \mathbf{b}_\mathbf{i}$ on the right hand side of Equation 4.46 are said to *define the hyperplane*. Let $\mathbf{b}_\mathbf{i}$ be one of the defining control points. At the same time, let $\mathbf{b}_\mathbf{i}$ be a corner control point of a macro–element. Then, the defining control points are referred to as a *hyperplane around the corner* $\mathbf{b}_\mathbf{i}$. Further, we call $\mathbf{b}_\mathbf{i}$ the *anchor* of the hyperplane.

In addition, let $\mathbf{b}_\mathbf{i}$ and $\mathbf{b}_\mathbf{j}$ be two defining control points. At the same time, let $\mathbf{b}_\mathbf{i}$ and $\mathbf{b}_\mathbf{j}$ be on an edge of a macro–element. Then, the defining control points are referred to as a *hyperplane around the edge* $(\mathbf{b}_\mathbf{i}, \mathbf{b}_\mathbf{j})$ and $(\mathbf{b}_\mathbf{i}, \mathbf{b}_\mathbf{j})$ is called the *anchor* of the hyperplane.

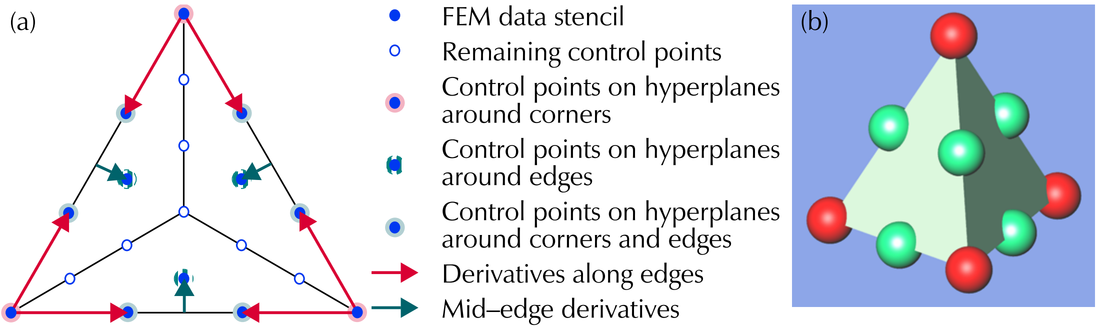

(a)
● FEM data stencil
○ Remaining control points
Control points on hyperplanes around corners
Control points on hyperplanes around edges
Control points on hyperplanes around corners and edges
Derivatives along edges
Mid–edge derivatives
(b)

FIGURE 5.6 (a) FEM data stencil: control points needed to interpolate to position at vertices as well as derivatives along edges and perpendicular to edges (only one triangular face of a Clough–Tocher split tetrahedron is shown).
(b) Hyperplanes on macro–element around corners (red) and edges (green).

A hyperplane *applies* to a control point $\mathbf{c_i}$ whenever we can state a hyperplane condition according to Equation 4.46 that contains both the point $\mathbf{c_i}$ and the anchor of the hyperplane. We also say that $\mathbf{c_i}$ *lies on* the corresponding hyperplane. Further, we say that a macro–element *owns* a hyperplane if the defining points $\mathbf{b_i}$ pertain to the element.

5.5.2 The Structure of the Global Stiffness Matrix

The structure and representation of the global stiffness matrix correspond to the situation introduced in Section 5.4.1.

Recall the relational coefficients $c_{I,\,J}$ between two degrees of freedom which we denote by their global indices *I*, *J*. In contrast to the C^0 setting, the structure of the sparse global matrix is now formally given by the following refined condition:

> Let *T* be a macro–element and let $A(T)$ denote the set of degrees of freedom pertaining to the element *T*. Further, let $B(T)$ denote the set of degrees of freedom which influence linearly dependent nodes on *T*.
>
> The relational coefficient $c_{I,\,J}$ between two degrees of freedom *I* and *J* (it be pressure or displacement) is not equal to zero if and only if there exists an element *T* with $(I \in A(T) \vee I \in B(T)) \wedge (J \in A(T) \vee J \in B(T))$.

Since we employ C^0 pressure interpolation, the set $B(T)$ obviously contains displacement degrees of freedom only. It is composed of all degrees of freedom which pertain to the hyperplanes around the corners and edges of the macro–element while not belonging to the element itself. Before assembling a macro–element we thus have to know these hyperplanes and their defining control points, possibly on adjoining elements. To this aim, we have to classify all control points in the Clough–Tocher split input mesh into free and linearly dependent ones (see Section 5.5.3). The free control points make up the set of finite element degrees of freedom and actually have an entry in the resulting global system of equations.

Analogously to the C^0 setting, the above condition is symmetrical which again results in a symmetrical sparse global stiffness matrix.

5.5.3 Clough–Tocher Splitting Tetrahedral Tesselations

In essence, the tetrahedral mesh is split according to the construction scheme of Theorem 4.5. All tetrahedra are traversed and split at their incenters. Surfaces of the macro–element are split at their centroids as long as the adjoining tetrahedron has not been split so far or in case that the surface belongs to the outer surface of the volume under consideration. As soon as both incenters of two neighboring macro–elements have been determined, the splitting point on the common surface is set to be the piercing point of the straight line connecting the two incenters.

Labelling of free and linearly dependent nodes. In the same traversal of elements, the control points pertaining to the FEM data stencil of each macro–element are labelled free or linearly dependent. The procedure is as follows: All four corner nodes of each macro–element are labelled free. In a next step, each corner is checked against a potential hyperplane already defined around this corner. In case of an existing hyperplane, the three neighboring control points on the edges emanating from this control point (see green shaded points in Figure 5.6a) are set to be linearly dependent on the hyperplane. Otherwise, these points are labelled free and together with the corner point set to define the hyperplane around the corner (red spheres in Figure 5.6b). This approach makes sure that hyperplanes are defined by four points pertaining to the same macro–element. A hyperplane is thus completely and uniquely defined as long as the macro–element is not degenerated.

After processing the hyperplanes around the corners of the macro–element, the hyperplanes around its six edges are investigated (green spheres in Figure 5.6b). The mid–face points of adjoining subtriangles (green hashed points in Figure 5.6a) are either set to be linearly dependent on an existing hyperplane around the edge or set to define this hyperplane together with the two mid–edge points (green shaded points in Figure 5.6a).

Establishing linear dependencies. After the first step, each linearly dependent control point knows the hyperplane it lies on. In other words, a linearly dependent control point is uniquely determined by the hyperplane condition of Equation 4.46 given that the weights according to Equation 4.45 are known. However, it is only after splitting all macro–elements that the weights can be evaluated. Consequently, these weights have to be computed in a second traversal of all macro–elements in order to establish the linear dependencies.

Unfortunately, Equations 4.46 and 4.45 give the hyperplane condition and corresponding weights for two *adjacent* tetrahedra. However, the sequence of traversal of elements does not guarantee that the macro–element comprising the linearly dependent point and the macro–element that owns the hyperplane have a common face. It is equally well possible that the two macro–elements share only a common corner or edge.

One way to cope with the problem is to propagate the hyperplane and thus the linear dependency through common faces to the macro–element under consideration. It turns out that this slow and complicated approach can be avoided. Instead, the linear dependency can be computed directly by means of a *virtual tetrahedron* comprising the linear dependent node and at the same time having a common face with the macro–element that owns the hyperplane.

Consider the two–dimensional example situation in Figure 5.7. The red control points $\mathbf{b}_{120}, \mathbf{b}_{030}, \mathbf{b}_{021}$ define the red shaded hyperplane owned by the triangle $(\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2)$. The blue control points around $\mathbf{v}_1$ are required to lie on this hyperplane and thus are linearly dependent. One way to determine the linear dependency of the control point $\mathbf{c}_{120}$ is to propagate the dependency stepwise across common edges (see Figure 5.7a). The result of the propagation is independent of the clockwise or counterclockwise sense of traversal and thus invariant of the geometry or number of traversed triangles. As a consequence, it is admissible to define a virtual triangle comprising $\mathbf{c}_{120}$ and sharing a common edge with the triangle that owns the hyperplane (light gray shaded triangle in Figure 5.7b). Now, the linear dependency of $\mathbf{c}_{120}$ follows directly from the hyperplane condition and the geometry of the virtual triangle with respect to the triangle $(\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2)$.

After establishing the linear dependencies, every macro–element knows all the control points which directly or by means of a hyperplane determine its FEM data stencil. The remaining control points on the surface and inside the element follow from internal C^1 conditions between micro–elements. These will be taken into consideration when assembling the element as described in the next section.

5.5.4 Assembly of a Macro–Element

Having a Clough–Tocher split tetrahedral tesselation according to the previous section, the assembly of a macro–element can take place locally. In short terms, the process consists

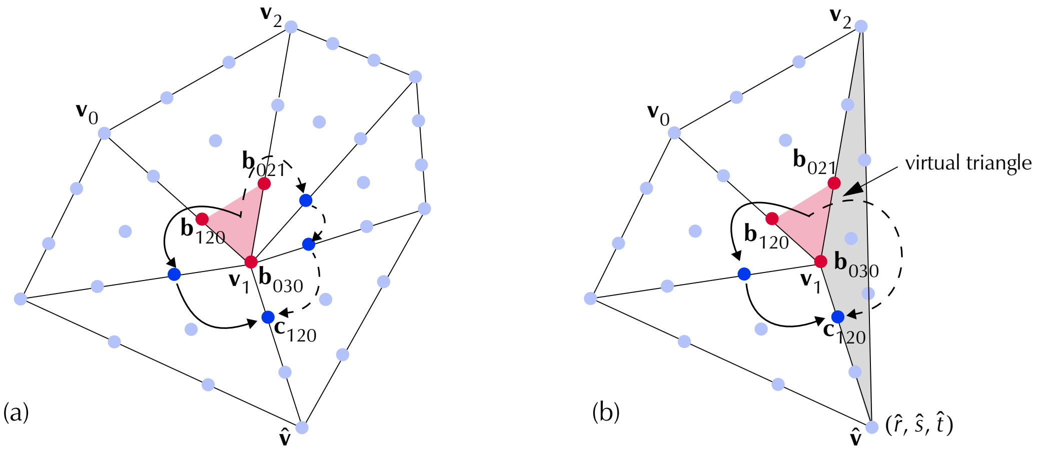


FIGURE 5.7 Establishing the linear dependency of $\mathbf{c}_{120}$ with respect to the three–dimensional hyperplane $(\mathbf{b}_{120}, \mathbf{b}_{030}, \mathbf{b}_{021})$ through propagation across common edges (a) or directly by means of a virtual triangle (b).

of summing up the contributions of the local stiffness matrices of the twelve micro–elements while at the same time eliminating linearly dependent control points from the global system of equations. Similar to the one–dimensional example in Section 5.1.2, this elimination also includes the distribution of the linearly dependent stiffness and force entries onto corresponding global degrees of freedom.

Resolving internal dependencies. Possible linear dependencies of control points pertaining to the FEM data stencil are already known. The determination of dependencies of the remaining surface control points as well as all internal control points of the macro–element is accomplished from the surface to the center by means of an iterative application of the hyperplane condition. As a consequence, for inner control points we have to determine multi–level dependencies.

We think of a macro–element as consisting of four shells of control points. The first shell corresponds to the 56 control points on the surface, the second shell comprises 26 control points, the third 8 and the fourth shell corresponds to the incenter splitting point (see Figure 5.8).

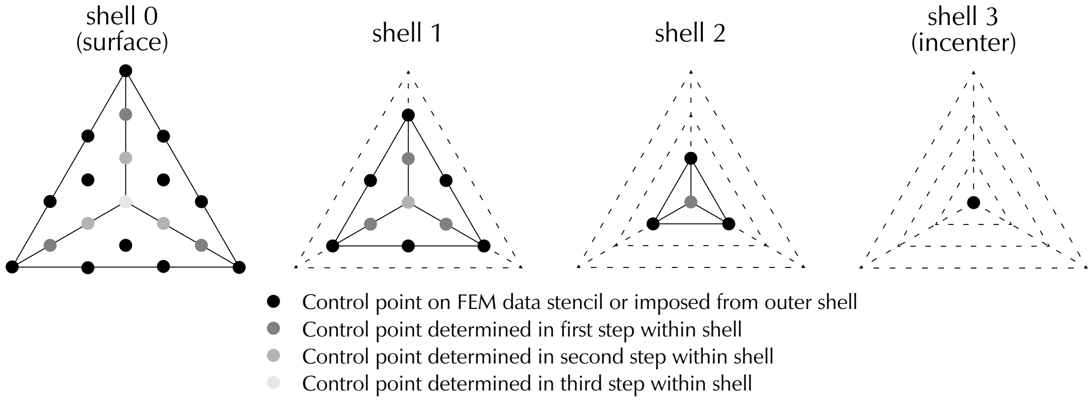

shell 0
(surface)
shell 1
shell 2
shell 3
(incenter)
● Control point on FEM data stencil or imposed from outer shell
● Control point determined in first step within shell
● Control point determined in second step within shell
● Control point determined in third step within shell

FIGURE 5.8 Four shells of control points in macro–element and order of determination of linearly dependent control points within each shell.

We now determine the dependencies shell to shell starting on the surface. By means of Equations 4.46 and 4.45, we for each node have to establish its linear dependency on four nodes which already must have been handled and thus uniquely define the corresponding local hyperplane. This condition imposes restrictions on the order of determination within each shell. The shading in Figure 5.8 depicts the resulting order on each shell. As a result, we for each linearly dependent control point arrive at a *dependency list* of its defining free control points and corresponding weights. For multi–level dependencies, the list results from a weighted combination of the dependency lists of the four control points defining the local hyperplane.

An interesting observation to make is that the hyperplane condition in the determination of linear dependencies within a shell converts to a plane condition. Due to the Clough–Tocher splitting process the three control points of the local hyperplane on the shell and the linearly dependent point happen to be coplanar. Thus, the weight of the fourth point evaluates to zero. This fact speeds up the determination of linear dependencies within the shells. Figure 5.9 gives an example where in the determination of c_{1200} the barycentric coordinates of $\hat{v}_0$ with respect to the tetrahedron (v_0, v_1, v_2, v_3) evaluate to $(\hat{r}, \hat{s}, \hat{t}, 0)$.

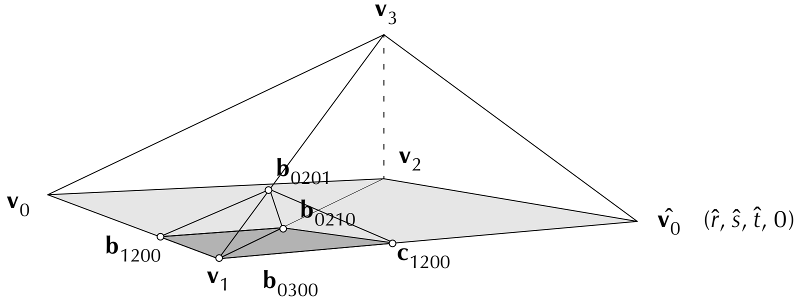


FIGURE 5.9 C^1 hyperplane condition degenerating to a plane condition.

Assembly of micro–elements. The cubic local stiffness matrices are exactly the same as in the cubic C^0 setting. The main difference is that we have to eliminate or in other words to compensate for linearly dependent control points in the global stiffness matrix. After compiling the dependency lists each linearly dependent control point $\mathbf{b}^{ld}$ is uniquely determined as a weighted sum of global degrees of freedom $\mathbf{b}^f$

$$\mathbf{b}^{ld} = \sum\nolimits_k \sigma_k \mathbf{b}_k^f . \tag{5.19}$$

Unlike the one–dimensional example in Section 5.1.2, we do not eliminate linear dependent control points after assembling the global system. This would not be memory efficient. Further, the structure of the global stiffness matrix has been determined for global degrees of freedom only in Section 5.5.2. Last but not least, only control points corresponding to global degrees are assigned a global index and only control points on the surface of macro–elements are permanently allocated. As a consequence, we have to find a means to take care of linearly dependent control points locally during the assembly of micro–elements.

Similar to the one–dimensional example in Equation 5.13 we can write the elimination multiplication as

$$\mathbf{K}_{red} = \mathbf{M}^T\mathbf{K}\mathbf{M} = \mathbf{M}^T(\sum\nolimits_m \mathbf{K}^{(m)})\mathbf{M} = \sum\nolimits_m \mathbf{M}^{(m)T}\mathbf{K}^{(m)}\mathbf{M}^{(m)}$$

where the superscript (m) denotes local entities. In order to perform the local elimination $\mathbf{M}^{(m)T}\mathbf{K}^{(m)}\mathbf{M}^{(m)}$ we have to elaborate upon a local elimination matrix $\mathbf{M}^{(m)}$. From Figure 5.4 we see that $\mathbf{M}^{(m)}$ will have a block structure corresponding to the *x*–, *y*–, *z*– and *p*–components of the local stiffness matrix. Further, we can state the following properties of $\mathbf{M}^{(m)}$ (compare Figure 5.10):

- $\mathbf{M}^{(m)}$ will have three columns with indices $c_x(k)$, $c_y(k)$, $c_z(k)$ for the element's global degrees of freedom $\mathbf{b}_k^f$ only. The index mappings $c_x(k)$, $c_y(k)$, $c_z(k)$ will result from subsequent considerations.
- $\mathbf{M}^{(m)}$ will have three rows for the *x*–, *y*– and *z*–component of each control point of the micro–element. The rows corresponding to the *x*–, *y*– and *z*–component of free control points will be zero except for a one in the columns $c_x(k)$, $c_y(k)$ and $c_z(k)$, respectively. In contrast, the rows corresponding to the *x*–, *y*– and *z*–component of linearly dependent points will be zero except for the weights σ_k in the respective columns $c_x(k)$, $c_y(k)$ and $c_z(k)$ of the global degrees of freedoms $\mathbf{b}_k^f$ defining the linear dependency. The pairs $(\mathbf{b}_k^f, \sigma_k)$ are given by the dependency list of each linearly dependent control point.
- In the mixed setting, $\mathbf{M}^{(m)}$ will have a unit submatrix corresponding to the local submatrix $\mathbf{K}_{pp}$.
- The dimension of $\mathbf{M}^{(m)}$ follows to $(20 + 20 + 20 + 4) \times (28 + 28 + 28 + 4) = 64 \times 88$. Each block $\mathbf{m}^{(m)}$ corresponding to the *x*–, *y*– or *z*–submatrix of the local stiffness matrix is identical and is of dimension 20×28. The number of rows obviously equals 20 because of the second property and corresponds to the number of control points of a trivariate cubic Bézier tetrahedron. The following reasoning yields the number of columns: the hyperplanes around the corners and edges of the macro–element uniquely define the FEM data stencil and thus the element. Each hyperplane has four defining control points. However, the six hyperplanes around the edges share two control points with hyperplanes around corners. Thus we have a total of $4 \times 4 + 6 \times 2 = 28$ free control points defining the macro–element.

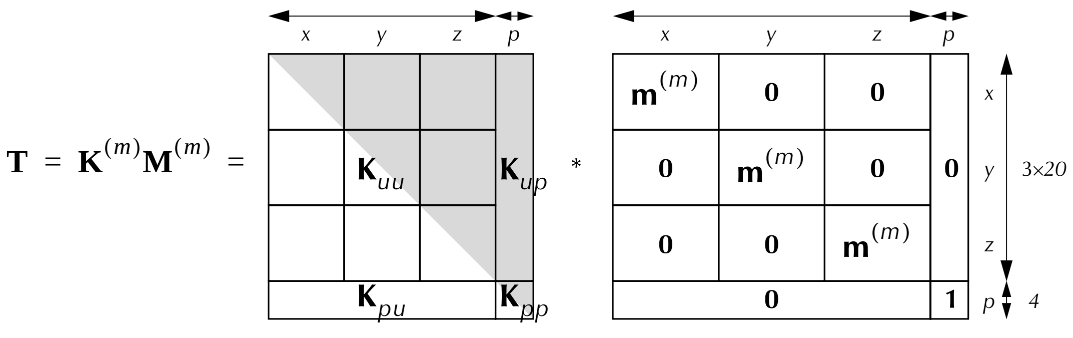


FIGURE 5.10 Elimination matrix and intermediate result in the local elimination of linearly dependent control points.

- Last but not least, the index mappings $c_x(k)$, $c_y(k)$, $c_z(k)$ follow from the dependency list of the incenter control point: obviously the incenter is part of all micro–elements and depends on all 28 control points defining the macro–element. Consequently, its dependency list may be used to define the following mapping: given that a pair $(\mathbf{b}_k^f, \sigma_k)$ is at position i in the list we chose $c_x(k) := i$, $c_y(k) := i + 28$ and $c_z(k) := i + 2 \times 28$.

Having found the elimination matrix, the assembly of a micro–element proceeds as follows: in a first step we compute the intermediate result $\mathbf{T} = \mathbf{K}^{(m)}\mathbf{M}^{(m)}$. This multiplication eliminates all columns pertaining to linearly dependent control points in $\mathbf{K}^{(m)}$. In a second step, we eliminate the rows by multiplying $\mathbf{M}^{(m)T}\mathbf{T}$. The result of the second multiplication is directly stored in the global stiffness matrix. The positions of the entries in the global matrix follow from the inverse mappings $c_x^{-1}(k)$, $c_y^{-1}(k)$, $c_z^{-1}(k)$ which yield the index k of a global degree of freedom.

Figure 5.11 shows the structure of the global C^1 stiffness matrix for the same example problem as in Figure 5.5. The Clough–Tocher split and the subsequent elimination of linearly dependent control points resulted in 182 displacement and 106 pressure degrees of freedom, yielding a 652×652 stiffness matrix. The *x*–, *y*–, *z*–, and *p*–parts of the matrix structure according to Figure 5.4 are again readily identified. The huge increase in displacement degrees of freedom induced by the split is more than compensated for by the elimination of linearly dependent nodes. This is in agreement with the expected reduction of the problem dimension due to an increased level of continuity. However, the number of pressure degrees of freedom is increased significantly. The C^1 matrix is more densely populated than its C^0 counterpart (23.5% as opposed to 13.5%). Further, the distribution of values is different due to the elimination of linearly dependent control points.

In Section 5.6, we will discuss properties of the resulting global stiffness matrices and oppose them to the C^0 case. Section 5.7 will present results and comparisons of C^0 and C^1 simulations on a synthetic block of tissue.

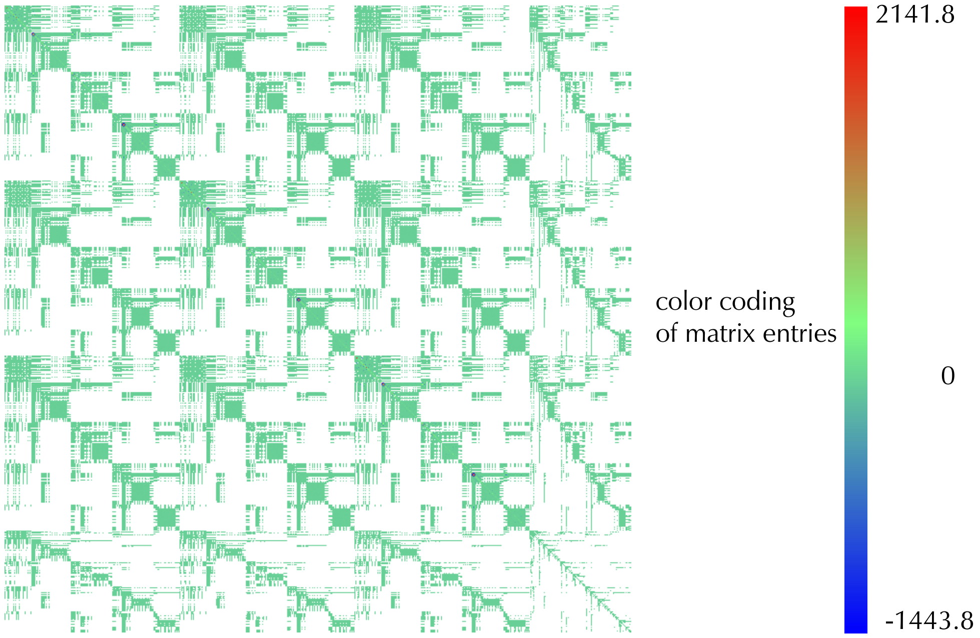


FIGURE 5.11 Structure of a global stiffness matrix for a C^1 Bernstein–Bézier example problem.

5.5.5 Disassembly

The disassembly of globally free nodes is equivalent to the C^0 setting: for each degree of freedom the entry in the solution vector is attached to the corresponding node. For linearly dependent nodes the solution follows from their dependency lists by means of Equation 5.19. Together with the boundary conditions these values determine the resulting trivariate displacement and scalar pressure function and thus the simulation result.

5.6 THE GLOBAL SYSTEM OF EQUATIONS

In order to decide on optimal solving strategies for the global system of equations it is expedient to know about the structure and properties of the global stiffness matrix. We therefore point out to some properties of the global system matrix before looking into solving strategies.

5.6.1 Properties

From Sections 5.4.1 and 5.5.2 we know that both the global matrix in the pure displacement– based formulation and the matrix of the mixed formulation are *symmetric*. Further, in Section 3.3.2, we have seen that for symmetric bilinear forms the global stiffness matrix is *positive definite* [68].

Unfortunately, only the pure displacement–based formulation yields a symmetric bilinear form. In contrast, the mixed formulation combines two bilinear forms the second of which is not symmetric due to the interconnection of displacement and pressure degrees of freedom (second term on the left hand side of Equations 3.53 and 3.70). The solution is then given in terms of a positive definite part for the displacement degrees of freedom and a negative definite part for the pressure degrees of freedom [160]. As a consequence, the resulting mixed stiffness matrix is *indefinite*, which is reflected in partly negative entries on the main diagonal. The only exception to this rule is the case of constant pressure interpolation where the pressure parts of the stiffness matrix are condensed out prior to the assembly (see Equation 5.18). This again yields a positive definite global matrix.

Looking at the local stiffness matrices we can observe the following properties. From a condition number point of view, all local matrices are singular, irrelevant of the degree of interpolation or the formulation used. In order to understand this, it is instructive to examine the eigenvalues of these matrices. As a consequence of the symmetry of local stiffness matrices, all eigenvalues will be real. Further we realize that in correspondence with the global matrices the local matrices are either *positive semidefinite* for the pure displacement–based formulation or *indefinite* in the mixed formulation. For the former, we among purely positive eigenvalues have six vanishing ones $\lambda_1 = \ldots = \lambda_6 = 0$, whereas for the latter, we arrive at four negative eigenvalues related to the linear pressure interpolation followed by positive and six vanishing ones. Vanishing eigenvalues correspond to the *rigid body modes* present, which in the case of three–dimensional linear elasticity are movements along the *x*–, *y*– and *z*–axis as well as the three rotations around the axes.

The same observation also holds for the global stiffness matrix as an assemblage of indefinite or positive semidefinite matrices: it is only after restraining the system by the imposition of boundary conditions that it will be solvable. To be more distinct, the imposition of a boundary condition decreases the number of eigenvalues of the stiffness matrix and a zero eigenvalue is lost whenever the restraint results in the elimination of a rigid body mode [8].

Last but not least, we have to consider the possible impact the elimination process of linearly dependent control points can have on the properties of the matrices. Obviously, symmetry is preserved under the elimination operation since it is applied symmetrically to rows and columns. The preservation of positive definiteness results from the following theorem [56]:

> ***Theorem 5.1.*** If $\mathbf{K} \in \mathbb{R}^{n \times n}$ is positive definite and $\mathbf{M} \in \mathbb{R}^{n \times k}$ has rank *k*, then $\mathbf{K}_{red} = \mathbf{M}^T\mathbf{K}\mathbf{M} \in \mathbb{R}^{k \times k}$ is positive definite, too.

If $\mathbf{z} \in \mathbb{R}^k$ then $\mathbf{z}^T\mathbf{K}_{red}\mathbf{z} = (\mathbf{M}\mathbf{z})^T\mathbf{K}(\mathbf{M}\mathbf{z}) \geq 0$. Further, $\mathbf{z}^T\mathbf{K}_{red}\mathbf{z} = 0$ implies $\mathbf{M}\mathbf{z} = \mathbf{0}$. Since **M** has full column rank, this implies $\mathbf{z} = \mathbf{0}$, which proofs the theorem.

What remains to be shown is that the elimination matrices of Sections 5.1.2 and 5.5.4 have full rank. We construct an $n \times n$ matrix $\mathbf{M}_\sigma^i$ which under $\mathbf{K} \cdot \mathbf{M}_\sigma^i$ does not eliminate the linearly dependent column *i* under consideration but only adds its contents weighted by the entries of the vector σ to arbitrary many other columns. Let $\mathbf{V}^{ij}$ be an $n \times n$ matrix having a one at (i, j) and zeros everywhere else and let **1** denote the identity matrix. Then the matrix $\mathbf{M}_\sigma^i$ follows to

$$\mathbf{M}_\sigma^i = \mathbf{1} + \sum_j \sigma_j \cdot \mathbf{V}^{ij}.$$

The rank of a matrix is invariant under the addition of multiples of a column to any other column. The repeated application of such column operations can be used to construct $\mathbf{M}_\sigma^i$ starting out at the identity matrix **1**. Consequently, $\mathbf{M}_\sigma^i$ is of rank *n* since the rank of the $n \times n$ identity matrix is *n*. In order to arrive at the elimination matrices of Sections 5.1.2 and 5.5.4 we only have to delete column *i*, which results in a full rank matrix **M** of rank $k = n - 1$. This completes the proof of the preservation of positive definiteness.

5.6.2 Solving Strategies

From the previous section we know that we have to look for solution strategies suited for symmetric sparse systems being either positive definite for the pure displacement–based formulation or indefinite for the mixed formulation.

In general, solution strategies divide into two categories: *direct solution* techniques and *iterative solution* methods. A predetermined number of steps and operations are executed in direct techniques whereas iterative methods work by repeatedly improving an approximate solution until it is accurate enough.

Most of the direct solution techniques used today basically are applications of *Gaussian elimination*. However, although the basic Gaussian solution scheme is applicable to almost any system of linear equations, its effectiveness in the context of finite element analysis depends on specific properties of the stiffness matrix, such as symmetry and, most important, positive definiteness. The process basically consists of repeatedly subtracting multiples of equations from other equations in order to bring the system into a form which is known as the *upper triangular form*. The solution then directly follows from back–substitution.

An important observation to make is that in every step this process assumes to have a corresponding nonzero diagonal element that makes it possible to reduce the elements below it to zero. These elements are again used during back–substitution. This property is assured for positive definite and indefinite matrices. For numerical reasons, this so–called *pivot element* should be as large as possible. This can be achieved by *pivoting*, a process which refers to reordering the rows and in complete pivoting even the columns such as to have the largest element of the remaining submatrix as pivot element.

However, Gaussian elimination is very memory–consuming. For dense $n \times n$ systems the required storage is proportional to n^2 and the number of operations sums up to $\frac{1}{3}n^3$. For sparse systems with half bandwidth m we arrive at a memory consumption proportional to $n \cdot m$ and the number of operations evaluates to roughly $\frac{1}{2} \cdot n \cdot m^2$ [8]. Assuming that the half bandwidth grows proportional to $\sqrt{n}$ we see that even for sparse matrices the storage consumption becomes a limiting factor for large systems.

Moreover, even without pivoting, the elimination of elements below the current diagonal entry alters all remaining equations and thus the structure of the matrix. As a consequence, it is difficult to use a memory–saving representation for sparse matrices which do not have a banded structure (see Figures 5.5 and 5.11). Further, special pivoting strategies that minimize fill-in have to be developed [56].

In contrast to direct methods, iterative solving schemes only access the linear system via the matrix–vector product $\mathbf{y} = \mathbf{Ax}$ [6]. This scheme allows for the exploitation of the sparse structure of $\mathbf{A}$ and is thus more memory efficient. A major disadvantage of iterative methods is that the time of solution can be estimated only roughly because the number of iterations required for convergence heavily depends on the *condition number* of the symmetric matrix $\mathbf{K}$

$$\mathrm{cond}(\mathbf{K}) = \|\mathbf{K}\| \cdot \|\mathbf{K}^{-1}\| \overset{\text{symm.}}{=} |\lambda_n| / |\lambda_1|$$

with λ_n and λ_1 referring to the largest and smallest eigenvalue, respectively.

The oldest, best known and most effective iterative method for symmetric positive definite systems is the *Conjugate Gradient* method, originally of Hestenes and Stiefel [63, 8, 56]. It generates a sequence of conjugate (or orthogonal) vectors. Basically these vectors are the residuals of the iterates and span the space of solutions. At each iteration the dimension of the spanned space is increased by one. It can be shown that in exact arithmetics convergence is reached in at most n iterations for an $n \times n$ system. Of course, in practice and by means of preconditioning, convergence is reached in much fewer iterations.

The algorithm is based on the principle that the solution of $\mathbf{KU} = \mathbf{R}$ minimizes the total potential $\Pi = \frac{1}{2}\mathbf{U}^T\mathbf{KU} - \mathbf{U}^T\mathbf{R}$. Looking this way, the conjugate vectors represent the gradients of the quadratic functional which gives the method its name. Although in theory this minimization is guaranteed to converge for positive definite $\mathbf{K}$ only, the conjugate gradient method is applicable to indefinite problems, too. However, despite of preconditioning, the rate of convergence decreases substantially and convergence cannot be expected to be monotonic anymore. There are alternatives for the solution of indefinite matrices such as *MINRES* and *SYMMLQ*. For further references as well as convergence estimations, the reader is referred to [56, 6].

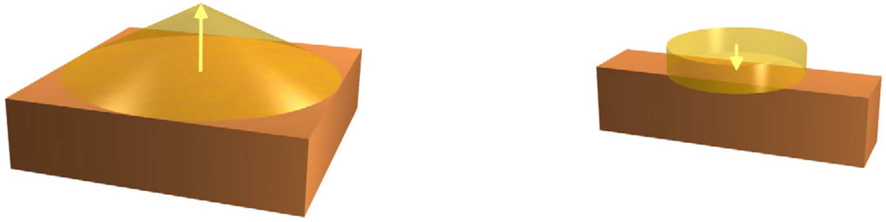
In the present implementation, and according to the suggestions in [6], we restricted ourself to the use of the corresponding publicly available conjugate gradient implementation in combination with three different preconditioners (diagonal, incomplete LU and incomplete Cholesky).

5.7 RESULTS ON A SYNTHETIC BLOCK OF TISSUE

In this section we will investigate the performance of the Bernstein–Bézier elements on a synthetic block of tissue.

In the case of C^0 elements we will oppose the influence of finite element mesh refinement to increasing the degree of interpolation functions. In addition, we will look at differences and properties of the mixed formulation. In a second section, we will oppose the C^1–continuous simulation to its cubic C^0 counterpart. As for the C^0 setting, we will investigate the differences between the displacement–based and the mixed formulation.

The simulation examples will be performed using two test configurations, hereinafter referred to as *pull* and *push* configurations, respectively. The pull configuration features a 20x20x5 block of elastic material subjected to a central loading force pointing upwards. The force is centered on the top surface and decreases linearly to the border which results in a conical force distribution. The base plane edges of the block are fixed in order to counterbalance the loading force. The push configuration consists of a 20x5x5 bar of elastic material which is subjected to a cylindrical force distribution pointing downwards at the center of the top. This time, the whole base plane is fixed. Please see Figure 5.12 for the two configurations.



pull configuration push configuration

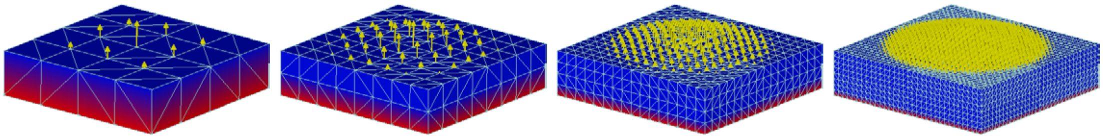
FIGURE 5.12 Pull and push configuration for test simulations.

5.7.1 C^0–continuous Simulation

Raising the degree of interpolation versus mesh refinement. The first test series attempts to reveal the influence of raising the degree of interpolation functions on a fixed mesh resolution as opposed to mesh refinement with fixed degree of interpolation. Both strategies obviously lead to an increase in the total number of degrees of freedom and thus to a more flexible response of the body under consideration.

The elastic material was chosen to have a Young modulus $E = 60$ and a Poisson ratio $\nu = 0.25$. For the simulation according to the pull configuration with a center force $f = 3$ we used four different discretizations resulting in 96, 768, 6144, and 49152 tetrahedra respectively. Figure 5.13 depicts the four discretizations: the red colored nodes at the base plane indicate zero displacement boundary conditions, whereas blue nodes stand for displacement degrees of freedom. Yellow arrows indicate the force distribution.

Besides the simulation outcome with respect to the degree of interpolation functions and the discretization, it will be interesting to investigate the interdependence of maximal displacement, number of degrees of freedom as well as the size and sparsity of the corresponding global stiffness matrix. Figure 5.14 shows the results of the test series. Runtime and matrix statistics are given in Table 5.1.



4x4x1 blocks,
96 tetrahedra

8x8x2 blocks,
768 tetrahedra

16x16x4 blocks,
6144 tetrahedra

32x32x8 blocks,
49152 tetrahedra

FIGURE 5.13 Four discretizations resulting from mesh refinement for the pull configuration.

There are three observation to make: firstly, a simulation with too coarse a discretization in combination with too low a degree of the polynomial expansion results in seemingly stiff materials (maximal displacement between 1 and 3). Increasing the number of degrees of freedoms (DOF), the simulation seems to converge to the exact solution which must be assumed to be at a maximal displacement of about 4.

The second observation is the close resemblance of convergence achieved by mesh refinement and degree elevation: either approach leads to an increase in the number of DOFs and thus to a more flexible elastic response. Consequently, one arrives at a more plausible simulation result. This is in correspondence with the finite element literature, which distinguishes between three different kinds of finite element analysis, all aiming at convergence to the real solution: the *h*–method, the *p*–method and the combination of both, the *hp*–method (see e.g. [8]). The *h*–method refers to mesh refinement using the same kind of elements, whereas the *p*–method denotes the increase of the degree of the polynomial expansion on a fixed mesh.

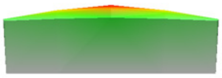
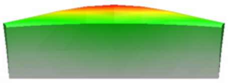
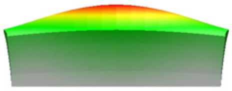
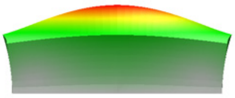
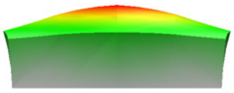
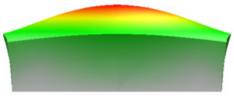
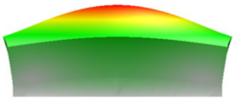
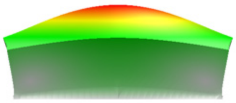
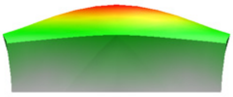
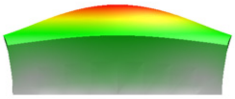
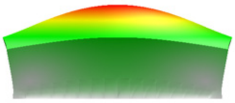
	4x4x1 / 96	8x8x2 / 768	16x16x4 / 6144	32x32x8 / 49152
LINEAR				
#DOFs	34	211	1381	9673
max. disp.	1.47	2.02	2.82	3.39
QUA-DRATIC				
#DOFs	207	1377	9669	71565
max. disp.	2.93	3.30	3.71	4.00
CUBIC				
#DOFs	616	4267	31009	
max. disp.	3.35	3.60	3.96	

FIGURE 5.14 C^0 test series with linear, quadratic and cubic interpolation according to the pull configuration for the four discretizations in Figure 5.13. White color indicates zero displacement while red stands for maximal displacement.

Thirdly, although the simulation is C^0, the results converge to a C^1 solution. This becomes evident in the very smooth appearance of the resulting displaced surface. As mentioned in the introduction of this chapter and in Section 3.4.1, this is in correspondence with theoretical results which state that a C^0 approach is sufficient for the simulation of linear elasticity [77, 125, 136]. In Section 5.7.2, we will investigate the degree of correspondence of the C^0 and C^1 approaches.

In addition to the above findings, Table 5.1 gives more detailed information about the performance of the elements under consideration. Matrix size and sparsity, the time spent in assembling the global stiffness matrix (including data structure build–up), as well as solving time and conjugent gradient iterations are stated. For all simulations, a diagonal preconditioner preceded the conjugent gradient solver, the runtime of which is included in the solving time.

Obviously, the problem and thus matrix size increases with the number of DOFs. At very fine discretizations, the matrix size becomes the limiting factor in the computation. Nevertheless, the biggest problem in the test series resulted in a 214695 by 214695 matrix which was solved in about 12 minutes after 426 conjugent gradient iterations. It is easy to see, that an increasing matrix size is accompanied by increasing sparsity (0.0281% for the above–mentioned example), a factor which considerably speeds up the vector–matrix products computed in the conjugent gradient iterations.

For C^0 problems and with increasing problem size, the assembly time becomes more and more negligible compared to the time spent in solving the system.

TABLE 5.1 Timings and matrix statistics for the test series of Figure 5.14.

degree	statistics	4x4x1	8x8x2	16x16x4	32x32x8
linear	matrix size	102	633	4143	29019
	sparsity (%)	17.6	4.35	0.845	0.136
	assembly time (s)	0.434	0.545	0.833	4.01
	solving time (s)	0.00988	0.136	2.46	47.5
	CG iterations	21	53	125	288
quadratic	matrix size	621	4131	29007	214695
	sparsity (%)	6.43	1.21	0.196	0.0281
	assembly time (s)	0.451	0.783	2.84	20.9
	solving time (s)	0.153	2.42	44.4	717
	CG iterations	47	95	197	426
cubic	matrix size	1848	12801	93027	—
	sparsity (%)	4.23	0.723	0.109	—
	assembly time (s)	0.579	1.95	13.6	—
	solving time (s)	1.13	18.9	311	—
	CG iterations	76	142	290	—

Pure displacement versus mixed formulation. For the comparison of the mixed and pure displacement based formulation we employ a push configuration. The material is chosen to be very elastic with Young modulus $E = 10$ and we let the Poisson ratio ν vary between 0.25 and 0.5, the case of total incompressibility. The cylindrical force centered at the top is chosen as $f = -2.5$. We restrict ourself to only one discretization, a 8x2x2 block resulting in 192 tetrahedra as is depicted in Figure 5.15.

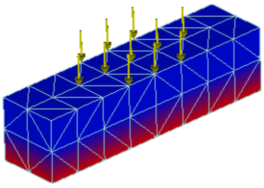


FIGURE 5.15 Discretization used for the push configuration.

For the following simulations we used the same conjugent gradient solver as in the previous test series but chose to have it preceded by an incomplete LU preconditioner. This choice is due to the fact that the diagonal preconditioner will fail when encountering a zero pivot element which for total incompressibility inherently is the case. The incomplete LU preconditioner performed best with respect to the ensuing number of conjugent gradient iterations. Unfortunately, it is computationally far more expensive than its diagonal counterpart. This is reflected by the fact that for some problems the running time of preconditioning becomes the predominant factor over the time spent in conjugent gradient iterations. Consequently, the solving times of Table 5.1 must not be compared directly to those in Tables 5.2–5.4.

The same test series was performed for the linear, quadratic and cubic element. For increasing Poisson ratio, we oppose the results of the pure displacement based to the mixed formulation. The results are given in Figure 5.16 and Table 5.2 for the linear element, Figure 5.17 and Table 5.3 for the quadratic element, and Figure 5.18 and Table 5.4 for the cubic element. The resulting deformation is depicted in a side and frontal view in order to emphasize both the resulting maximal displacement and the effects of volume preservation due to increasing incompressibility. A color ramp visualizes increasing displacement.

In addition, we compare the outcome of the pure displacement based and mixed formulation by means of pseudo colored difference images. The coloring is with respect to the *one–sided distance* between the respective surfaces. Given the distance $e(\mathbf{p}, S)$ between a point $\mathbf{p}$ and a surface S

$$e(\mathbf{p}, S) = \min_{\mathbf{p}' \in S} d(\mathbf{p}, \mathbf{p}')$$

where $d(\cdot, \cdot)$ denotes the Euclidean distance between two points, we find the one–sided distance between two surfaces S_1 and S_2 as

$$E(S_1, S_2) = \max_{\mathbf{p} \in S_1} e(\mathbf{p}, S_2). \tag{5.20}$$

Note, that this definition is not symmetric [102]. In addition, we give the mean square error as a percentage of the diameter of the axis-aligned bounding box.

The same color scale is used for the difference visualization in the three test series.

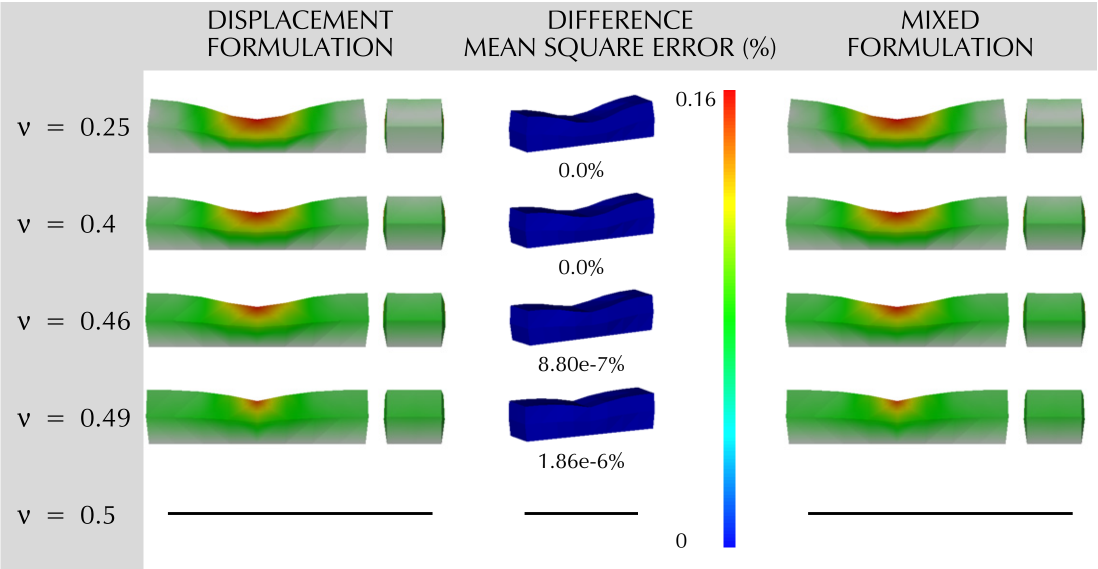
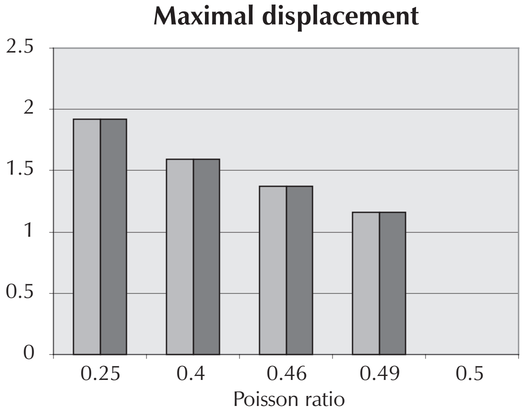


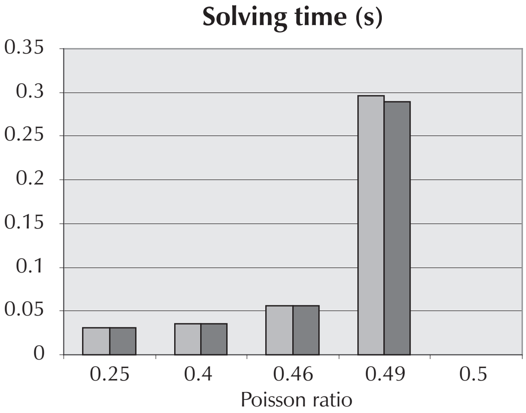
FIGURE 5.16 Comparison of pure displacement and mixed formulation for decreasing compressibility by means of the linear C^0 element using the push configuration according to Figure 5.15. White color indicates zero displacement, red maximal displacement. **[see Color Plate 2 on page 203]**

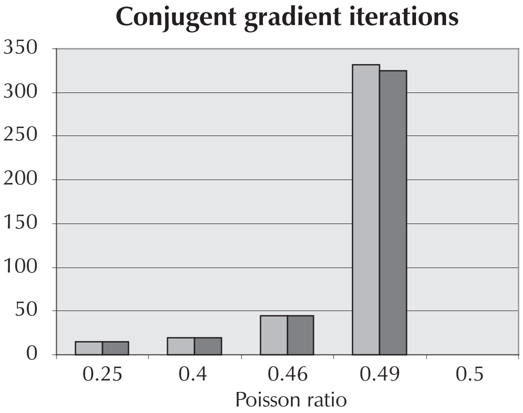
TABLE 5.2 Timings and statistics for the test series of Figure 5.16.

statistics	pure	mixed
# disp. DOFs	54	54
# pressure DOFs	–	–
matrix size	162	162
sparsity (%)	11.7	11.7
assembly time (s)	0.441	0.448



□ pure displacement formulation
■ mixed formulation





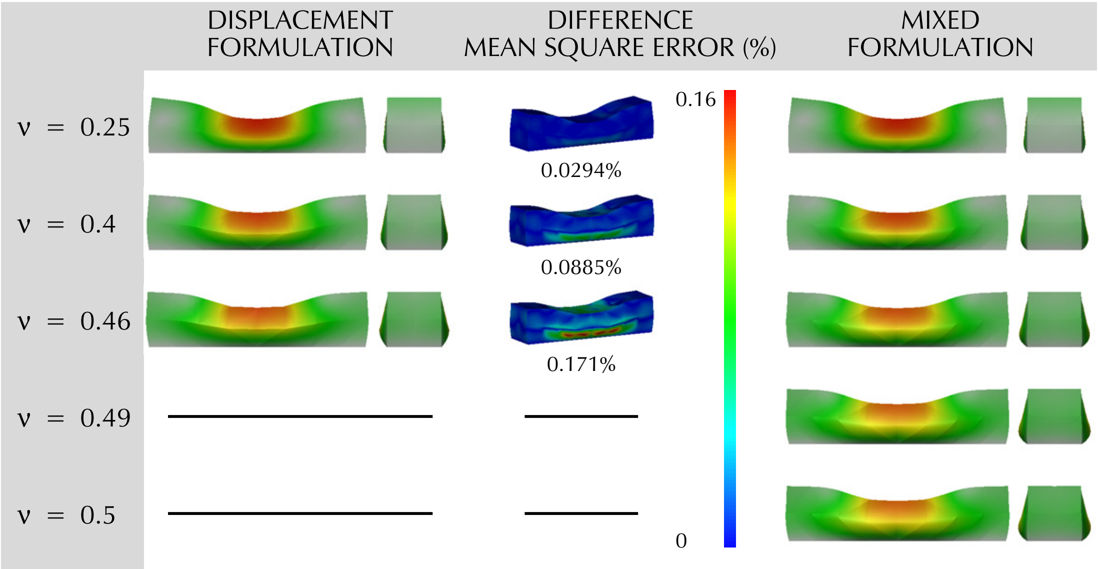
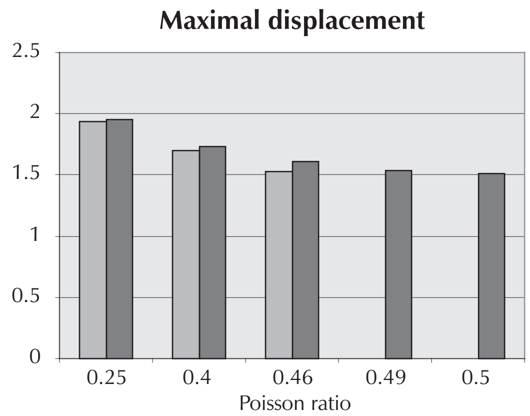


FIGURE 5.17 Comparison of pure displacement and mixed formulation for decreasing compressibility by means of the quadratic C^0 element using the push configuration according to Figure 5.15. White color indicates zero displacement, red maximal displacement. **[see Color Plate 3 on page 204]**

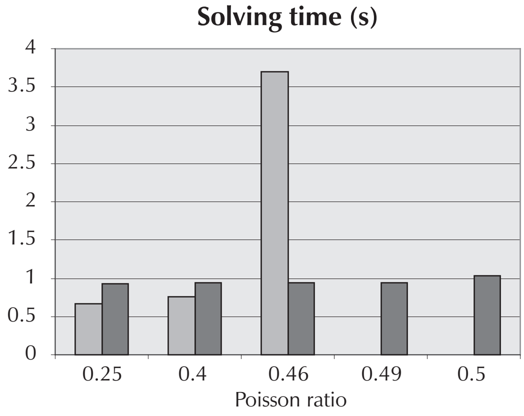
TABLE 5.3 Timings and statistics for the test series of Figure 5.17.

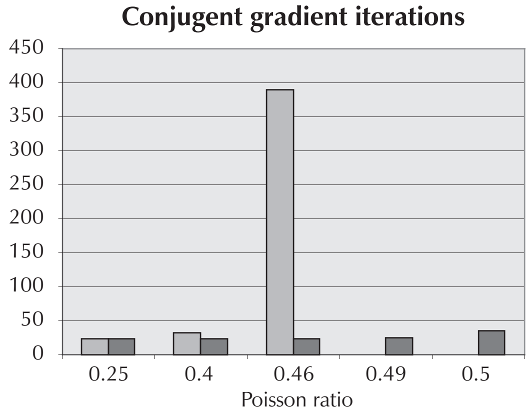
statistics	pure	mixed
# disp. DOFs	340	340
# pressure DOFs	–	81
matrix size	1020	1101
sparsity (%)	4.03	4.38
assembly time (s)	0.482	0.507



▨ pure displacement formulation

▨ mixed formulation





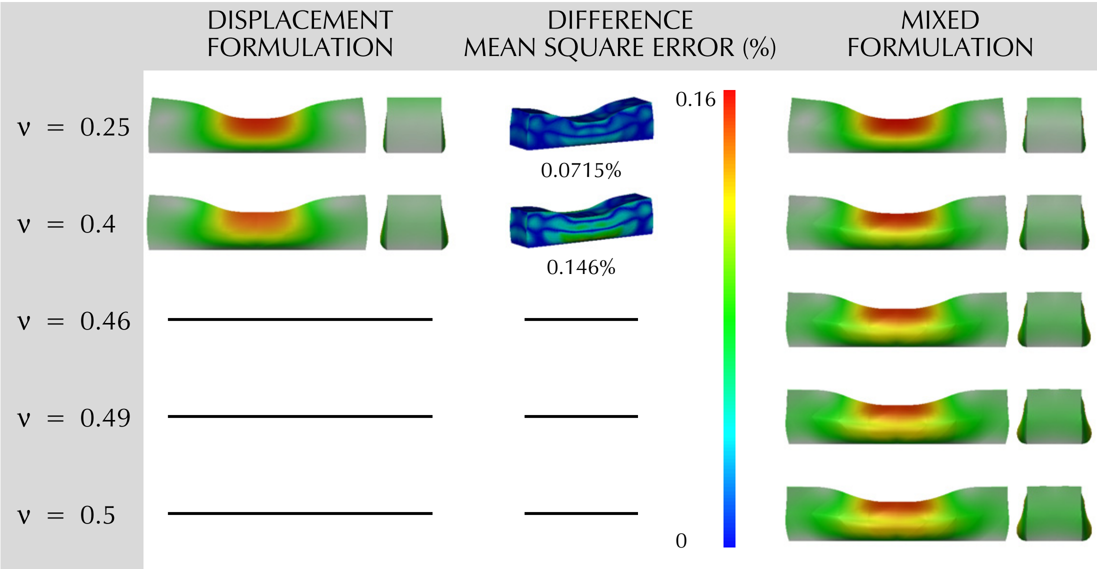
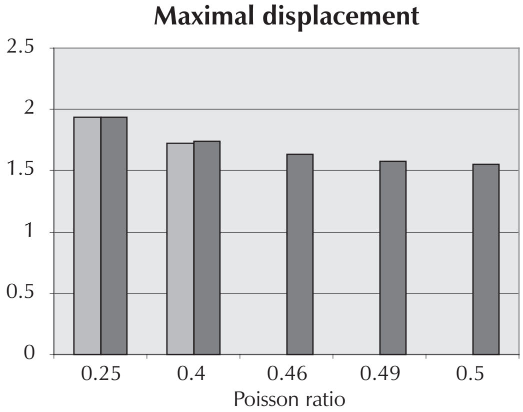


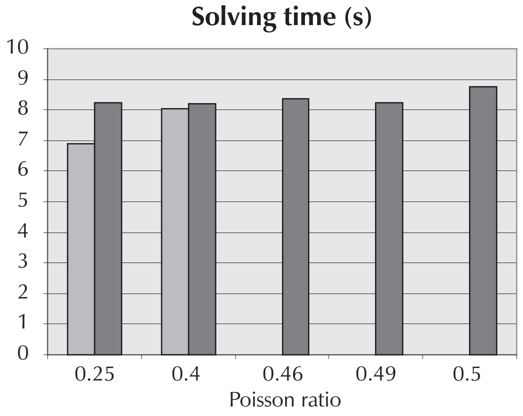
FIGURE 5.18 Comparison of pure displacement and mixed formulation for decreasing compressibility by means of the cubic C^0 element using the push configuration according to Figure 5.15. White color indicates zero displacement, red maximal displacement. **[see Color Plate 4 on page 204]**

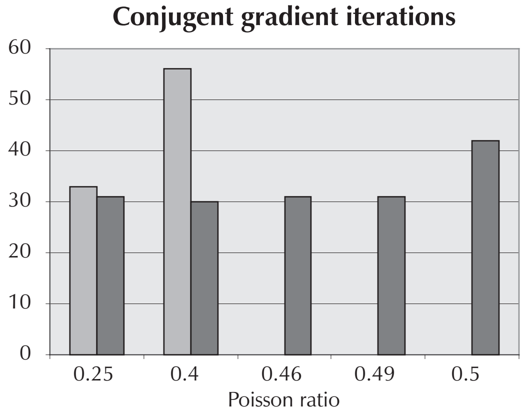
TABLE 5.4 Timings and statistics for the test series of Figure 5.18.

statistics	pure	mixed
# disp. DOFs	1050	1050
# pressure DOFs	–	81
matrix size	3150	3231
sparsity (%)	2.53	2.69
assembly time (s)	0.717	0.761



▨ pure displacement formulation
▨ mixed formulation





All three test series have in common that the assembly time and sparsity of the mixed and displacement based formulation are essentially the same. Further, and as expected, the simulation outcome for rather compressible materials at $\nu = 0.25$ is equivalent. Last but not least, the material seems to get stiffer with increasing ν which is reflected in a decreasing maximal displacement.

The linear test series in Figure 5.16 reveals very little overall deviation between the two formulations: for $\nu = 0.25$ and $\nu = 0.4$ the simulation outcome is identical whereas for $\nu = 0.46$ and $\nu = 0.49$ very minor differences result. The case $\nu = 0.5$ obviously cannot be simulated using the pure displacement based formulation. This also applies to the mixed formulation due to the zero $\mathbf{K}_{pp}$ submatrix in the static condensation of pressure degrees of freedom according to Equation 5.18.

Looking at the timings and statistics in Table 5.2 we see that both formulations perform very similarly: at $\nu = 0.49$ convergence is very slow compared to more compressible configurations. In comparison to the two higher order formulations discussed below, this behavior of the mixed formulation is surprisingly bad. This must be attributed to the constant interpolation of pressure which prevents pressure flow between the elements. However, constant interpolation of pressure avoids introducing additional pressure degrees of freedom into the global system of equation (see Table 5.2). In conclusion, the mixed and displacement based formulation for the linear element can be considered equivalent. Both formulations only allow for volume preservation by restricting the maximal displacement. Obviously, the linear interpolation in combination with such coarse a discretization does not offer enough degrees of freedom to allow for a lateral displacement which would compensate for the change of volume due to the compression.

In contrast to the linear setting, the quadratic and cubic setting reveal major differences between the two formulations for rather incompressible materials at $\nu \geq 0.4$. Whereas the differences in maximal displacement as well as the mean square errors stay remarkably low (see Figures 5.17 and 5.18), the rate of solver convergence in the purely displacement based setting tremendously decreases for more incompressible materials: both solving time and number of conjugent gradient iterations increase rapidly, if not exponentially (see Tables 5.3 and 5.4). For quadratic and cubic interpolation, the pure displacement based simulation did not converge anymore for materials with $\nu \geq 0.49$ and $\nu \geq 0.46$, respectively. In contrast thereof, when using the mixed formulation, neither the solving time nor the number of solver iterations are affected by increasing ν. With the exception of totally incompressible materials where a slight decrease can be observed, the rate of convergence for the test series is constant.

Further and in accordance with the locking phenomenon described in the finite element literature, we can observe the material being slightly stiffer in the pure displacement based setting. This is reflected in smaller maximal displacements and less lateral displacement. In contrast, the mixed formulation yields very decent volume preservation which is reflected in growing lateral displacement for increasing ν (see frontal view of mixed results as well as difference images in Figures 5.17 and 5.18). The slight concavity in the lateral bulge which mainly shows up for the cubic element and results in slightly spotted difference images (see Figure 5.18) is hard to explain. It could be due to the fact that finite element weights shared between several elements have their corresponding stiffness summed up in the global matrix. Such nodes tend to be globally stiffer than midface nodes at the surface. As a consequence and in combination with increasing volume preservation, such midface nodes seem to exhibit slightly excessive lateral displacement.

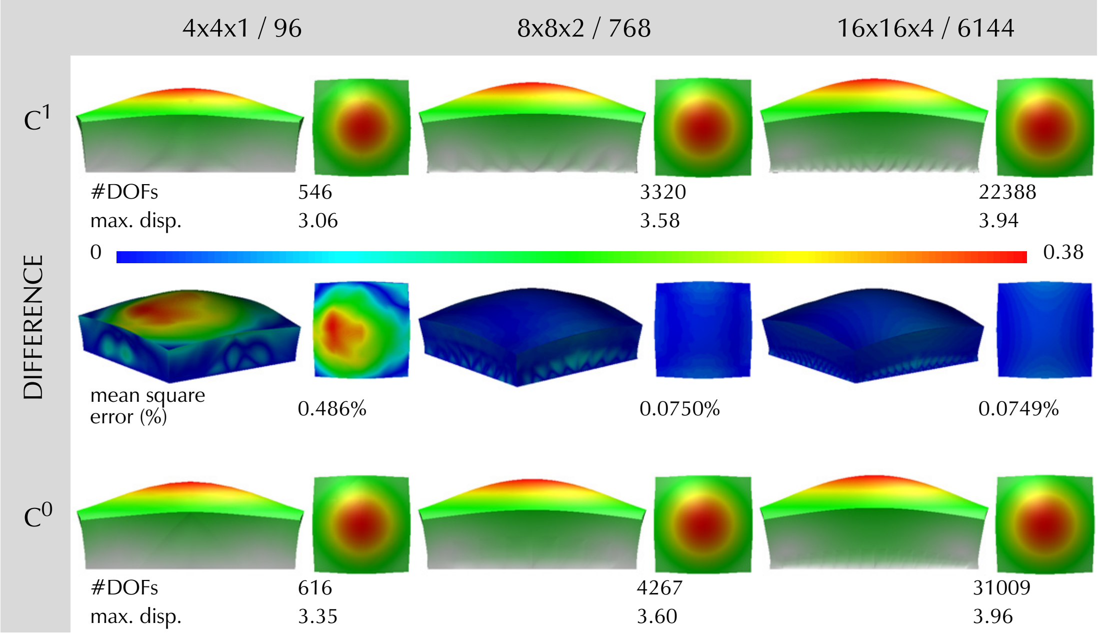


FIGURE 5.19 C^1 test series compared against the cubic C^0 element using the same configuration as in Figure 5.14. White color indicates zero displacement, red maximal displacement. **[see Color Plate 5 on page 205]**

In conclusion, the mixed formulation using linear pressure interpolation in combination with quadratic or cubic displacement interpolation can be regarded as numerically stable with respect to solver convergence, regardless of the compressibility of the simulated material. The number of additional pressure degrees of freedom is moderate as was made obvious already in Figure 5.5. However, the pure displacement based formulation performs well for materials with $\nu \leq 0.4$.

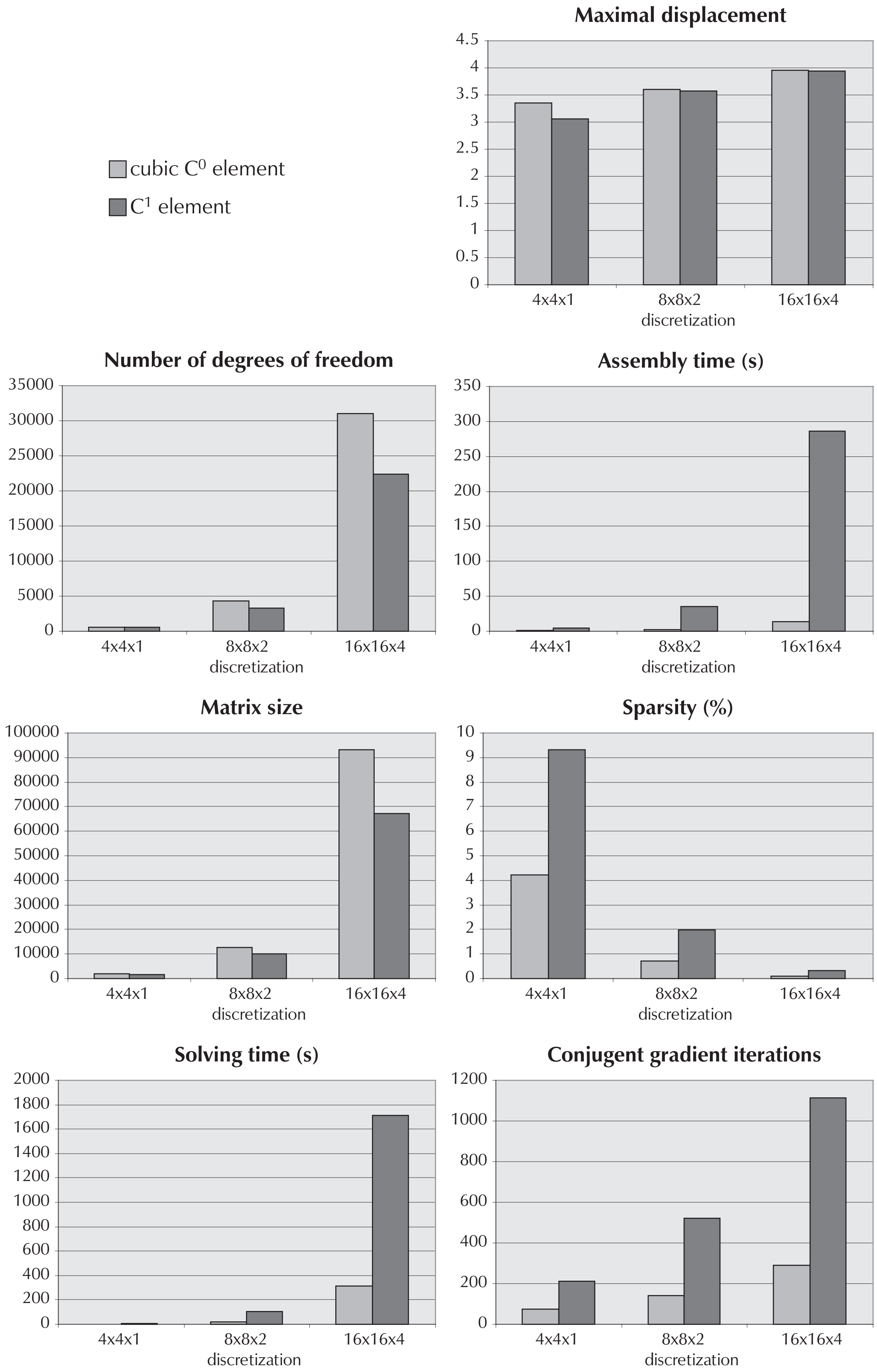
5.7.2 C^1–continuous Simulation

C^1 versus C^0 simulation. Using the same pull configuration as in Section 5.7.1, we will test the performance of the C^1 element compared to the cubic C^0 element. Figure 5.19 shows the results of the test series opposing C^1 to C^0 simulation for three different discretizations of the problem. Again pseudo colored difference images are used for error visualization. Timings and statistics are given in Table 5.5.

The first observation to make is the close resemblance of the simulation results. Even for the coarsest discretization, with only 96 tetrahedra the mean square error is below 0.5 percent with respect to the bounding box. For finer discretizations, the cubic C^0 and C^1 results are basically identical (see difference images in Figure 5.19 and the maximal displacements in Table 5.5). As stated in Section 5.7.1, this is in correspondence with theoretical results for the simulation of linear elasticity.

However, the difference image for the coarsest discretization is surprisingly asymmetrical although the top views in the upper and lower row reveal visually perfect rotational symmetry. On the one hand, this is due to the overemphasis in pseudo coloring of very slight differences. On the other hand, the effect must be attributed to numerical effects which render the propagation of linear dependencies in the Clough–Tocher C^1 construction not as symmetrical as it is in theory. As a consequence, asymmetries are to appear

TABLE 5.5 Timings and statistics for the test series of Figure 5.19.



which mainly depend on the tetrahedral mesh geometry and on the order of mesh traversal during data structure build–up (compare Sections 5.5.3 and 5.5.4). Similar asymmetries are to be seen in the C^1 test series opposing the pure displacement based to the mixed formulation (see Figure 5.20).

The first comment made for the C^0 elements in Section 5.7.1 also holds for the C^1 simulation: too coarse a discretization renders the material seemingly stiff and therefore counteracts the displacement in response to the applied force. In addition to the maximal displacement, it is very illustrative to compare the statistics of C^0 and C^1 simulation with respect to assembly time and matrix properties as well as solving time and iterations which are all given in Table 5.5.

Similar to the example in Section 5.5.4, we observe that in spite of the 1:12 split inherent to the Clough–Tocher construction the number of degrees of freedom is lower for a C^1 simulation: for the coarse, intermediate and fine discretization the number of C^1 DOFs amounts to only 89%, 78%, and 72% of C^0 DOFs, respectively. The same ratios are reflected in the size of the global system of equations. However, at the same time the C^1 matrices are more densely populated than their C^0 counterparts: the corresponding factors range from 2.2 to 3 for increasingly finer discretizations in the test series.

The performance of the C^1 element with respect to the assembly time is clearly worse: it takes 7.6, 17, and 21 times longer to assemble the C^1 system for the coarse, intermediate and fine discretization, respectively. As a matter of fact, this is not surprising at all given the complexity of the assembly procedure.

Bringing matrix size and solving time into correspondence, we see that the matrix size is not the only determining factor for the solving time. Both the sparsity and the condition of the matrices heavily influence the time spent in solving the C^1 systems. The sparsity influences the cost of the matrix–vector multiplications during the conjugent gradient iterations, whereas ill–conditioned systems necessitate a huge number of iterations to reach convergence. Both effects are clearly reflected in Table 5.5: although resulting in smaller systems, C^1 simulations clearly exceed the C^0 setting with respect to both the solving time and the iterations spent in the conjugate gradient solver. For the coarsest discretization, solving takes roughly 4 times longer for C^1 elements whereas the number of iterations needed to reach convergence exceeds the C^0 setting by a factor of about 2.5. The same factors rise to roughly 5.5 and 3.75 for finer discretizations.

In conclusion we can state that although smaller systems result in the C^1 setting, solving takes longer since the matrices are more densely populated and worse conditioned compared to the C^0 matrices. Further, due to its complexity, the C^1 assembly procedure is clearly more expensive than its C^0 counterpart. However, in accordance with the C^0 setting, solving is still the predominant factor for the overall computation time for larger problems.

Pure displacement versus mixed formulation. Similar to the C^0 test series, this section opposes the pure displacement and the mixed formulation using the C^1 element. The test series was performed using the same push configuration and discretization (see Figure 5.15). The results and the statistics are given in Figure 5.20 and Table 5.6, respectively. For the pseudo coloring of the difference images the same color scale as in the Figures 5.16, 5.17, and 5.18 was employed.

The first thing to mention is the close resemblance of the test series results with the results achieved for the cubic C^0 element. Again, the purely displacement based method

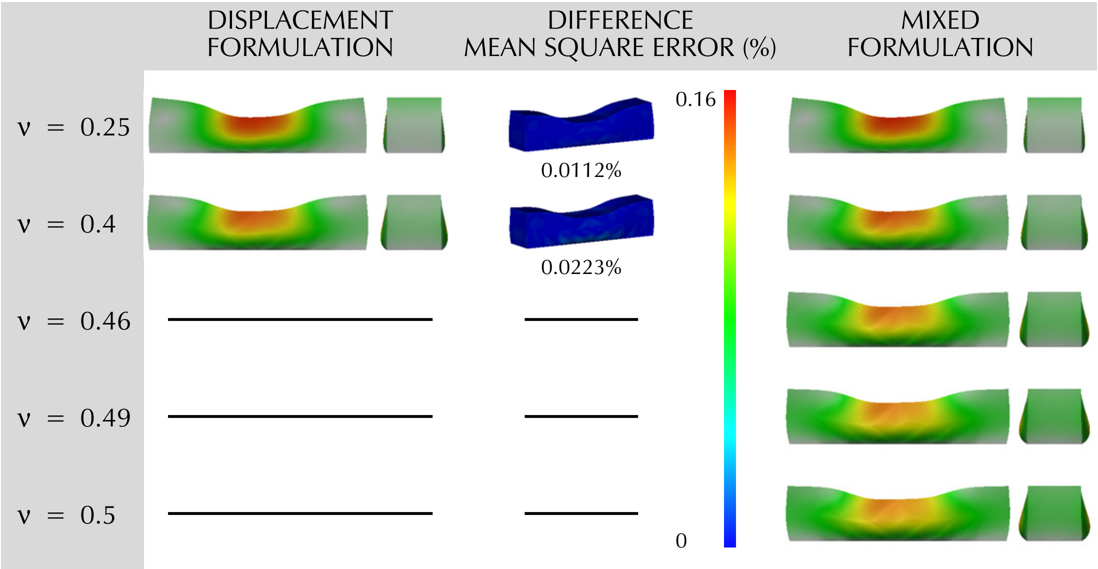
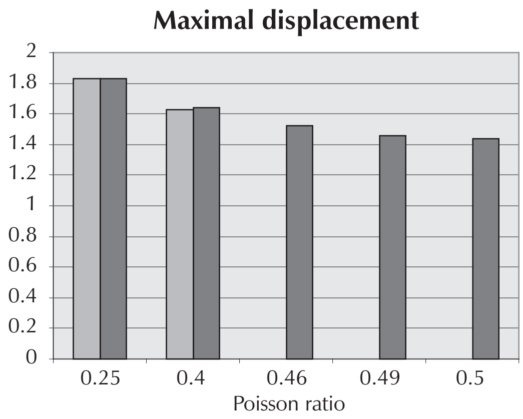


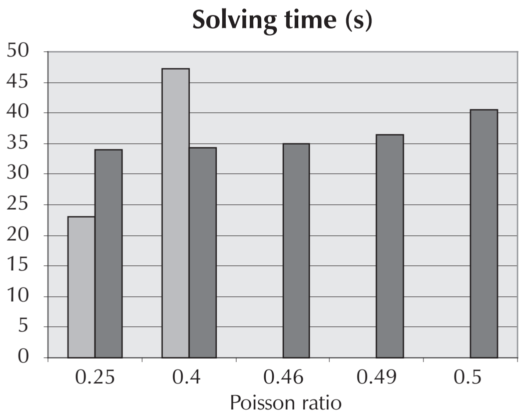
FIGURE 5.20 Comparison of pure displacement and mixed formulation for decreasing compressibility by means of the C^1 element using the push configuration according to Figure 5.15. White color indicates zero displacement, red maximal displacement.
[see Color Plate 6 on page 205]

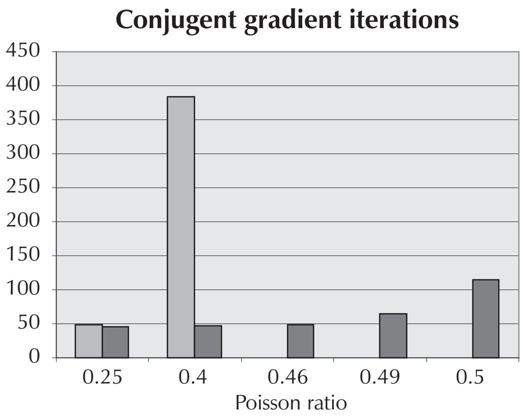
TABLE 5.6 Timings and statistics for the test series of Figure 5.20.

statistics	pure	mixed
# disp. DOFs	873	873
# pressure DOFs	–	729
matrix size	2619	3348
sparsity (%)	6.08	5.00
assembly time (s)	8.15	8.59



▨ pure displacement formulation
▨ mixed formulation





experiences problems for $\nu \geq 0.4$. For $\nu \geq 0.46$ the conjugate gradient solver fails to converge, whereas for $\nu = 0.4$ we have a noticeable decrease of the rate of convergence: the conjugate gradient iterations increase by a factor of 8 compared to $\nu = 0.25$. In contrast, the performance of the mixed formulation with respect to solving time and number of solver iterations is widely independent of Poisson's ratio: only for $\nu = 0.5$ we have a noticeable increase in the number of iterations.

Comparing the results quantitatively to the C^0 cubic test series (see Figure 5.18 and Table 5.4) we see that the C^1 element behaves slightly stiffer which results in smaller maximal displacements. Further, in accordance with the comparison in Table 5.5 the C^1 simulation is more expensive both with respect to assembly and solving time. Again, the size of the global matrix is smaller at the price of less sparsity. However, the increase of degrees of freedom induced by the mixed formulation is comparably huge for C^1 elements: while in the cubic C^0 setting the mixed formulation resulted in only a 7.7% increase of the overall number of degrees of freedom, the increase amounts to 84% in the mixed C^1 setting. This is due to the 1:12 split in the Clough–Tocher construction. Unlike the increase in displacement degrees of freedom, the increase in pressure degrees of freedom is not compensated for by a C^1 construction. As a consequence, the ratio between pressure and displacement degrees of freedom is clearly bigger in the C^1 setting (see also Figure 5.11).

In contrast to the C^0 setting, the lateral displacements counterbalancing the compression at the top of the block do not exhibit concavities when using C^1 interpolation (see the smooth convex bulges in the frontal images in Figure 5.20). Consequently, the corresponding difference images in Figure 5.20 show less patching artifacts than the difference images for the cubic C^0 element (Figure 5.18). Obviously, forcing the displacement field to be C^1–continuous reduces the problem of varying nodal stiffness with respect to mesh connectivity.

Conclusion. The C^1–continuous element yields results which are very similar to the C^0 simulations, especially when using the cubic element. This is in accordance with theoretical results for linear elasticity. In incompressible settings, the C^1 element performs slightly better but at the price of less computational efficiency. The test results make evident that the Bernstein–Bézier representation allows for increasing the level of continuity without destroying neither the approximation properties nor the locality of the solution basis. As has been expected, raising the level of continuity to C^1 imposes additional constraints on the solution and therefore reduces the overall problem dimension. However, the reduction is only with respect to the size of the global system of equations.

It is worth noting that the above examples were chosen such that the results are comparable. Especially when prescribing displacements instead of applying forces, major differences may result as depicted in Figure 5.21.

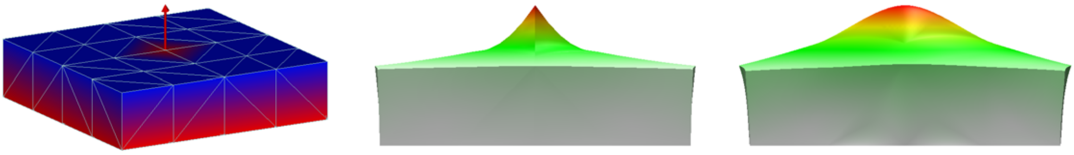


FIGURE 5.21 (a) Pull configuration with prescribed displacement centered at top.
(b) Cubic C^0 simulation.
(c) C^1 simulation.

CHAPTER 6

RAY TRACING TRIANGULAR BÉZIER PATCHES

Besides from representing the resulting surface of a finite element simulation using the elements of Chapter 5, triangular surfaces are more and more used to represent complex geometries [42]. Further, they lend themselves well to scattered data interpolation [4]. All those disciplines have a demand for an accurate visualization of the resulting surfaces.

Conceptually, there are two alternatives to render free–form surfaces: firstly, the conversion to a polygonal model and subsequent rendering of the resulting polygonal primitives or, secondly, direct ray tracing of the parametric surface description. The disadvantages of polygonalization are obvious. On the one hand, shading artifacts occur if the polygonalization is too coarse. On the other hand, visual effects like reflection and refraction as well as correct lighting is difficult or impossible to achieve. In contrast thereof, ray tracing, although being computationally more expensive, offers a means to accomplish all aforementioned effects. Further, ray tracing as a high quality rendering technique is well known and has been investigated thoroughly over the past decades (see e.g. [53]).

The basic requirement for ray tracing of objects, it be geometric primitives or a parametric surface, is the computation of intersections between a ray and the surface description. For free–form surfaces, this task is referred to as the *ray–patch intersection problem.* This chapter describes a new approach to this problem for triangular Bézier patches of arbitrary degree [41] based on the concept of Bézier clipping [99].

The chapter is organized as follows: after an overview of previous work, the extension of Bézier clipping to the triangular domain is presented. Thereafter, the problem of reporting wrong intersections inherent to the original algorithm for tensor product formulations is discussed, and differences and advantages of triangular Bézier clipping are investigated and opposed to the tensor product case. The presentation of results and a comparison to an approach based on nested bounding volumes concludes the chapter.

6.1 RELATED WORK

In general, the ray–patch intersection problem with free–form surfaces of arbitrary degree cannot be solved directly. As a consequence, one has to resort to an iterative computation of intersection points. Fournier and Buchanan [47] distinguish between two principle approaches: *geometric* and *parametric* intersection. Geometric intersection aims at finding the world coordinates of intersection points whereas parametric intersection determines their parametric coordinates. Most often, it is not only the intersection point itself one is interested in but also the local surface normal as well as shading and texture information. As far as free–form surfaces are concerned, all these requirements make it inevitable to know the world as well as the parametric location of an intersection. Further, as a matter of fact, knowing the parametric intersection coordinates automatically implies the corresponding point in 3D space whereas the contrary in general does not hold. As a consequence, with exception of early works [71, 126] emphasis in previous research as well as in this work has been put on parametric intersection methods.

Again, parametric intersection methods divide into two categories: *nested bounding volumes,* in general followed by a root finding scheme such as Newton iteration, and *parameter interval iteration*. These classes of algorithms will be revisited in the next two sections.

6.1.1 Nested Bounding Volumes

Approaches using nested bounding volumes basically compute a hierarchy of bounding volumes for every patch. In a preprocessing step each patch is hierarchically subdivided until the resulting sub–patches meet a certain stopping criterion, most often based on the flatness of the sub–patch. The computation of a ray–patch intersection now consists of a traversal of such a hierarchy guided by intersection tests between the ray and the bounding volumes on the respective hierarchy level. As soon as a leaf of the hierarchy is reached, the intersection between the ray and the leaf geometry is calculated. This is accomplished either by using Newton iteration, known to converge quickly on nearly planar surfaces, or by direct intersection of the ray with an approximating primitive.

One has to choose a suited bounding primitive providing for an optimal trade–off between tight enclosure and efficient intersection testing. Bounding spheres offering a very efficient intersection test [53] have been proposed as well as axes–aligned bounding boxes [138], oriented slabs [158], and parallelepipeds [7]. Especially the use of parallelepipeds is very suitable for Bézier patches as it takes advantage of their convex hull property. Another conceptually elegant and very efficient method, *Chebyshev boxing*, was introduced by Fournier and Buchanan in [47]. In order to find a hierarchy of enclosing bounding volumes they use the coefficients of the Chebyshev representation of bilinear patches approximating the sub–patches in the hierarchy. The actual point of intersection is finally found by intersecting interpolating bilinear patches in the leaves of the hierarchy with the ray. Obviously, such a procedure, as does any approach using an approximation of the surface in the leaves, requires solving the cracking problem between adjacent patches. Further, Chebyshev boxing can only deal with integral patches [25]. Therefore, Campagna et al. in [25] proposed an approach very similar to Chebyshev boxing which can handle rational patches, the *bounding volume hierarchy* (BVH). It computes nested bounding volumes using the Bézier representation. The actual intersection points are found either using Bézier clipping ([99], see section Section 6.1.2) on the sub–patches in the hierarchy leaves or by intersecting an interpolating bilinear sub–patch. In contrast to Chebyshev boxing, there is no need for calculating these sub–patches since the four corner control points of the corresponding Bézier sub–patch already form an interpolating bilinear patch.

6.1.2 Parameter Interval Iteration

In contrast to the fore–mentioned methods, parameter interval methods directly operate on the parametric domain by narrowing the candidate interval for an intersection. First approaches using multivariate Newton iteration employed interval arithmetic to overcome the inherent problem of finding a good starting point for the iteration [148]. More recently, Nishita et al. introduced the technique of Bézier clipping [99] which makes extensive use of the convex hull property of Bézier curves. Although being less computationally efficient than e.g. Chebyshev boxing, this approach is applicable to rational patches and, with slight improvements [24, 25], yields a robust and stable algorithm. In contrast to many nested bounding volume algorithms, Bézier clipping is guaranteed to find all intersections. Further, the memory usage of Bézier clipping is very low compared to nested bounding volume approaches, since only the control points of the patches have to be stored instead of hierarchical data structures and thousands of bilinear patches and bounding volumes.

In summary, in accordance with [25], we prefer Bézier clipping as a general choice for producing high–quality pictures without artifacts in reasonable time and with few memory requirements. Chebyshev boxing and BVH are still very useful for animation set–ups where the same scene may have to be rendered several times. A combination of a nested bounding volume approach together with Bézier clipping seems very promising. On the one hand, it eliminates the first clipping iterations which in general are less efficient due to little clipping of curved patches. On the other hand, it avoids potential approximation artifacts.

6.1.3 Ray Tracing Triangular Patches

Except for the early work on triangular Steiner patches in [126], only Stürzlinger addressed the problem of ray tracing triangular free–form surface patches [137]. Stürzlinger's approach must be attributed to the class of nested bounding volumes algorithms using *tripipeds* (skewed triangular prisms) as bounding primitives and Newton iteration similar to [7]. Chebyshev boxing unfortunately does not directly generalize to the triangular domain. Therefore, the next section presents an extension of Bézier clipping to the triangular domain.

6.2 TRIANGULAR BÉZIER CLIPPING

We recall that a triangular Bézier patch of degree *n* is given by

$$\mathbf{b}^n(r, s, t) = \sum_{|\boldsymbol{i}| = n} \mathbf{b}_{\mathbf{i}} B^n_{\mathbf{i}}(r, s, t)$$

where $|\boldsymbol{i}| = n$ stands for $i + j + k = n$ on the assumption that $i, j, k \geq 0$. Further, $B^n_{\mathbf{i}}(r, s, t)$ refers to the barycentric Bernstein polynomials of Equation 4.23 and $\mathbf{b}_{\mathbf{i}}$ represents the triangular control net (see Figures 4.14 and 4.15). Although there are three barycentric coordinates we are dealing with the bivariate case due to the linear dependency of the third barycentric coordinate $t = 1 - r - s$. As a consequence, and in correspondence with the notation in the original work of tensor product Bézier clipping [99], in the remainder of the chapter we will adhere to a notation using only *r* and *s* and omitting *t* and consequently only *i* and *j* while omitting $k = n - i - j$. A triangular patch therefore is of the form:

$$\mathbf{b}^n(r, s) = \sum_{j=0}^{n} \sum_{i=0}^{n-j} \mathbf{b}_{ij} B_{ij}^n(r, s) \tag{6.1}$$

After stating the ray–patch intersection problem and its reduction to two dimensions, we will present the Bézier clipping algorithm for triangular patches. We will point out to differences and analogies to the tensor product situation.

6.2.1 Ray–Patch Intersection Problem

The ray–patch intersection problem refers to the task of finding the intersections of a ray

$$\mathbf{r}(u) = \mathbf{r}_0 + u \cdot \mathbf{r}_d, \qquad u \in \mathrm{R}^+, \qquad \|\mathbf{r}_d\| = 1$$

with a Bézier patch according to Equation 6.1. As do [71, 99] we represent the ray as the intersection of two planes given by their normalized implicit equations

$$a_k x + b_k y + c_k z + e_k = 0, \qquad k \in \{0, 1\} \tag{6.2}$$

with $a_k^2 + b_k^2 + c_k^2 = 1$ (see Figure 6.1).

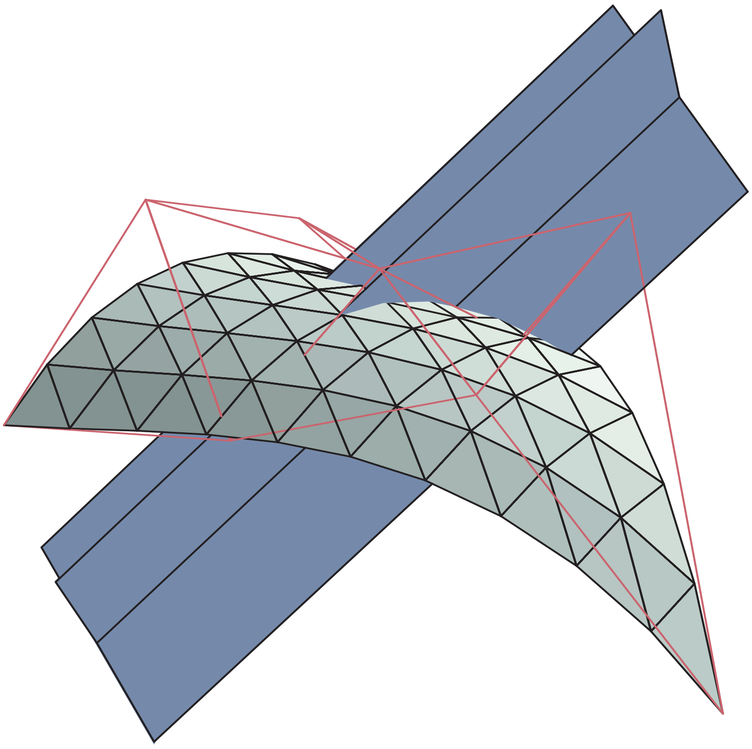


FIGURE 6.1 Representation of a ray by two orthogonal planes (control net of patch indicated in red).

In practice, the two planes are orthogonal. To this aim, we define the normals $\mathbf{n}_k = (a_k, b_k, c_k)^T$ and the distances to the origin e_k as

$$\mathbf{n}_0 = \frac{\mathbf{r}_0 \times \mathbf{r}_d}{\|\mathbf{r}_0 \times \mathbf{r}_d\|}, \qquad e_0 = -(\mathbf{n}_0 \cdot \mathbf{r}_0)$$

$$\mathbf{n}_1 = \frac{\mathbf{n}_0 \times \mathbf{r}_d}{\|\mathbf{n}_0 \times \mathbf{r}_d\|}, \qquad e_1 = -(\mathbf{n}_1 \cdot \mathbf{r}_0)$$

If $\mathbf{r}_0 \parallel \mathbf{r}_d$ we instead of $\mathbf{r}_0$ use a vector given by permuting the two biggest values in $\mathbf{r}_d$ in order to define $\mathbf{n}_0$.

6.2.2 Reduction to Two Dimensions

In complete analogy to [99, 25] the problem of finding an intersection

$$\mathbf{r}(u) = \mathbf{b}^n(r, s)$$

can be reduced from three to two dimensions even if the triangular patch $\mathbf{b}^n(r, s)$ is rational. This is accomplished by substituting Equation 6.1 into Equation 6.2 which yields

$$\mathbf{d}^n(r, s) = \sum_{j=0}^{n} \sum_{i=0}^{n-j} \mathbf{d}_{ij} B^n_{ij}(r, s) \overset{!}{=} \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{6.3}$$

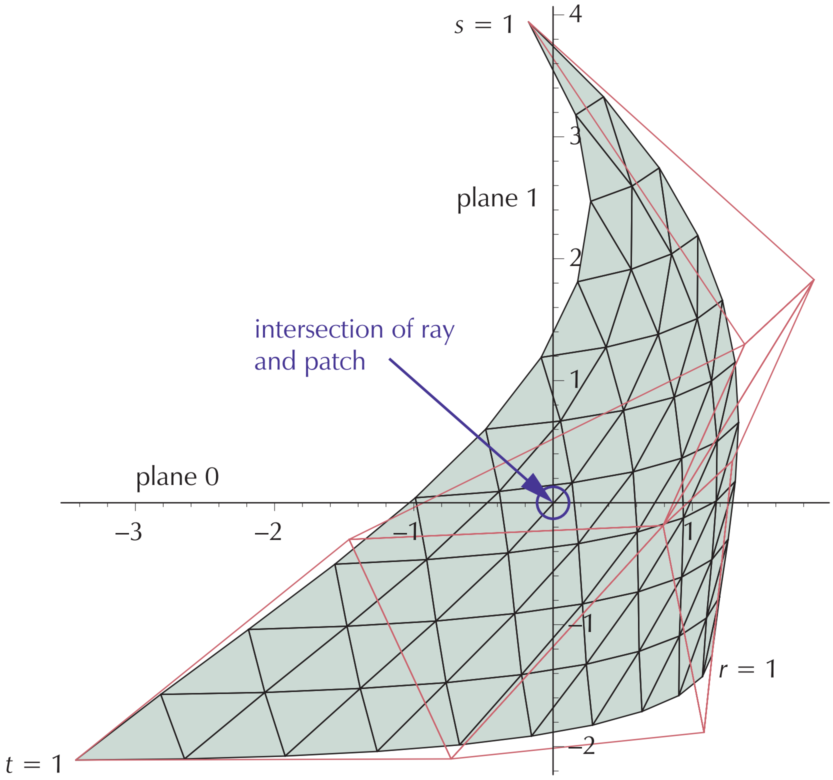


FIGURE 6.2 Reduction of the ray–patch intersection problem to two dimensions (control net indicated in red).

with
$$\mathbf{d}_{ij} = \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} x_{ij} + \begin{pmatrix} b_0 \\ b_1 \end{pmatrix} y_{ij} + \begin{pmatrix} c_0 \\ c_1 \end{pmatrix} z_{ij} + \begin{pmatrix} e_0 \\ e_1 \end{pmatrix} \tag{6.4}$$

and the coordinates x_{ij}, y_{ij}, and z_{ij} of the control points of the patch. The components 0 and 1 (in the remainder referred to as *x* and *y*) of $\mathbf{d}_{ij}$ geometrically represent the distance of the point to plane 0 and 1, respectively. For rational patches, these distances are scaled by the weights. The problem now reduces to finding the solution of Equation 6.3 (see Figure 6.2).

6.2.3 Finding Intersections

Bézier clipping basically clips away regions in the parametric domain which are known not to intersect the ray. For tensor product surfaces, Nishita et al. in [99] determine both (u_{min}, u_{max}) and (v_{min}, v_{max}) of the parametric candidate region for an intersection with the ray (see Figure 6.3a). Using these bounds, they subdivide the patch and iterate the procedure until the patch is small enough to satisfy a tolerance condition which assures sub–pixel accuracy.

A similar approach on the triangular domain of the barycentric coordinates *r, s* and *t* in general yields a complex non–triangular candidate region (see Figure 6.3b, red striped region). A triangular upper bound of this region can be found using only r_{min}, s_{min} and t_{min} (see Figure 6.3b, blue region). Subdivision of this triangular domain and iterating the procedure similar to the tensor product case determines the parametric intersection. In order to find potential multiple intersections, a patch will be subdivided into four sub–patches if in one clipping iteration in *r, s* and *t* too little of a patch was to be clipped away. In our implementation, the following subdivision criterion proved to be a good choice

$$\text{subdivide if } r_{min} + s_{min} + t_{min} < 0.5\,.$$

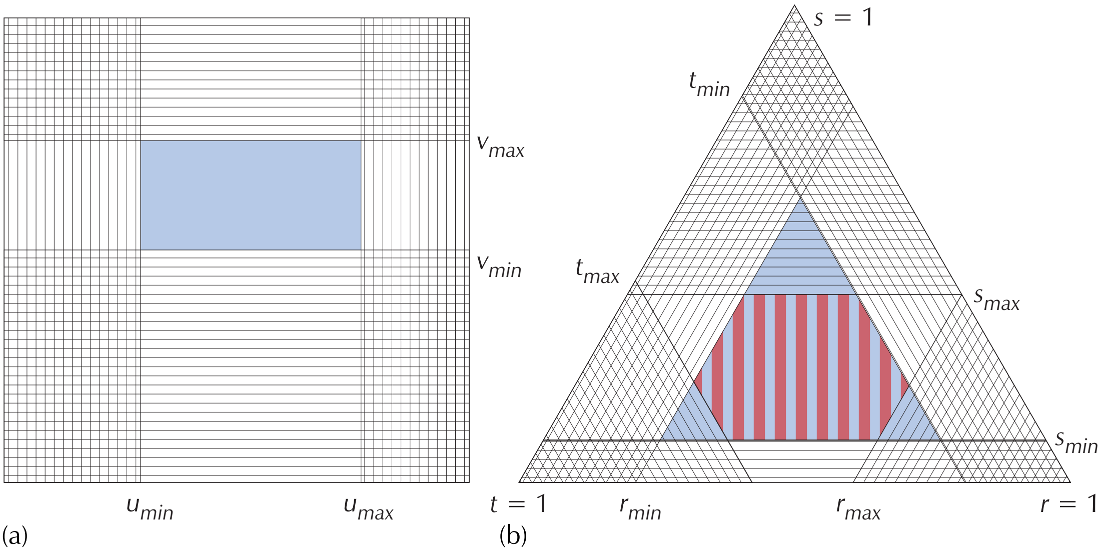


FIGURE 6.3 Bézier clipping in the parametric domain (candidate region highlighted in blue):
(a) Rectangular (tensor product) domain
(b) Triangular (barycentric) domain

In the following, the procedure of finding r_{min} on the example of a cubic patch will be illustrated. The steps for s_{min} and t_{min} follow from symmetry.

Firstly, we determine a line L_r parallel to the vector from $\mathbf{d}_{00}$ to $\mathbf{d}_{0n}$ through the origin. This line can be seen as a linear approximation of the curve of constant r through the origin. Expressing this line in its implicit form

$$ax + by + c = 0, \qquad a^2 + b^2 = 1$$

yields the distances dr_{ij} of the control points $\mathbf{d}_{ij} = (x_{ij}, y_{ij})$ to the line L_r as

$$dr_{ij} = ax_{ij} + by_{ij} + c, \qquad 0 \le i + j \le n$$

with $c = 0$ since the line passes through the origin. Figure 6.4 clarifies the situation.

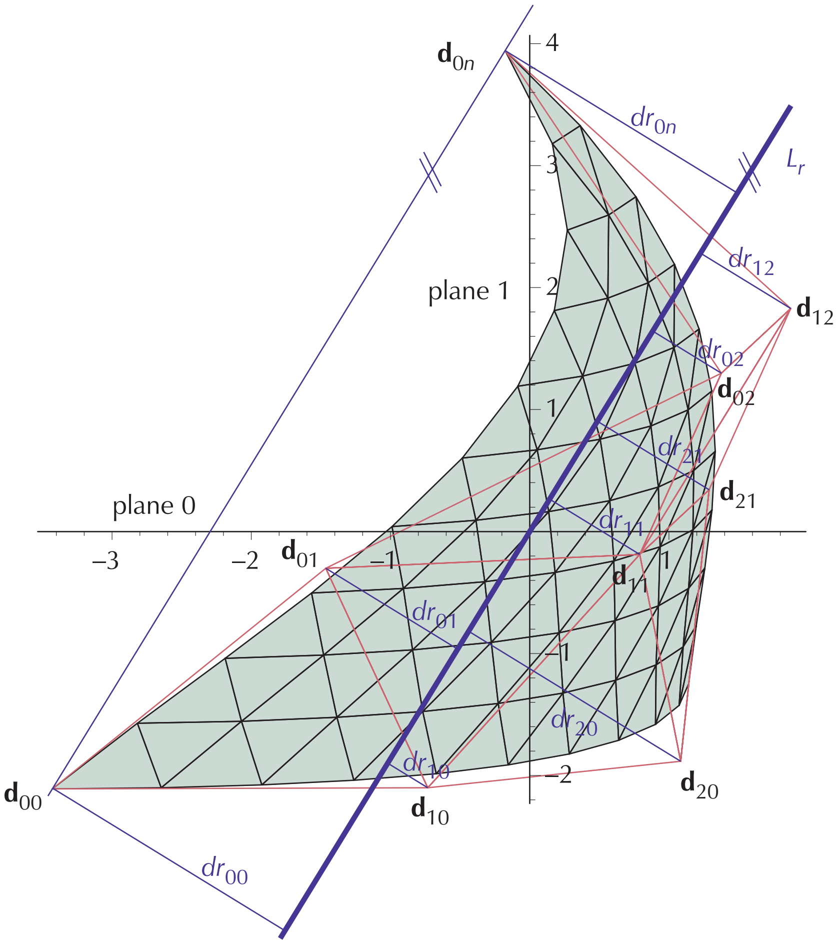


FIGURE 6.4 Line L_r and corresponding distances dr_{ij} of control points of a cubic patch (control point d_{n0} is hidden).

The distance $dr^n(r, s)$ of an arbitrary point to L_r consequently becomes

$$dr^n(r, s) = \sum_{j=0}^{n} \sum_{i=0}^{n-j} dr_{ij} B_{ij}^n(r, s).$$

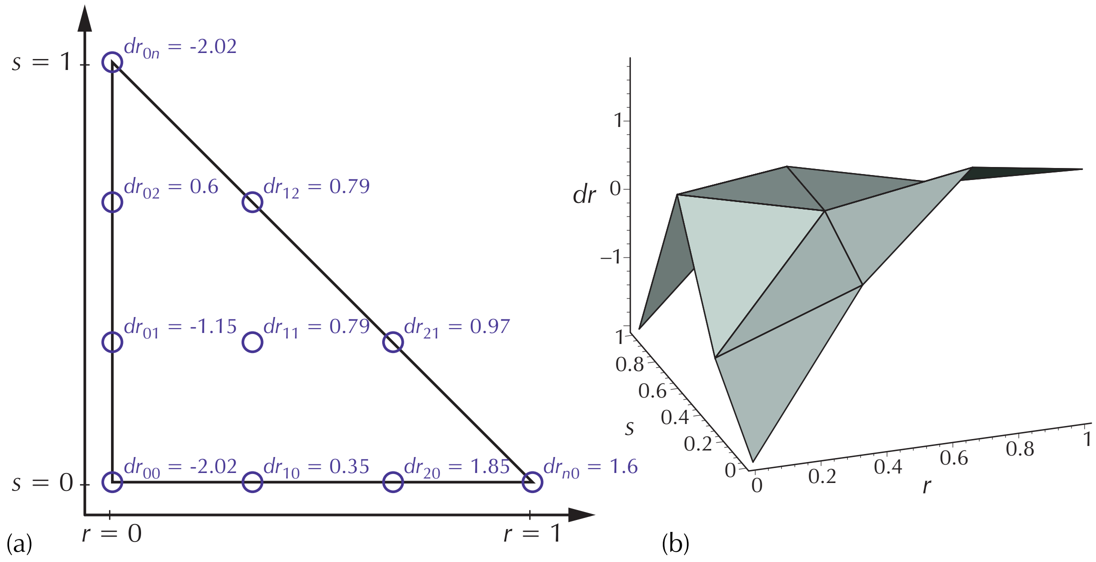


FIGURE 6.5 Functional distance patch: (a) top view, (b) 3D view.

The distance function $dr^n(r,s)$ can be regarded as a functional surface over the triangular domain as

$$\mathbf{dr}^n(r,s) = \sum_{j=0}^{n}\sum_{i=0}^{n-j}\mathbf{dr}_{ij}B_{ij}^n(r,s) = (r,s,dr^n(r,s))$$

with $\mathbf{dr}_{ij} = (r_i, s_j, dr_{ij})$ and equidistant $r_i = i/n$ and $s_j = j/n$ for $0 \le i + j \le n$. Figure 6.5 illustrates the functional distance patch and the corresponding distances.

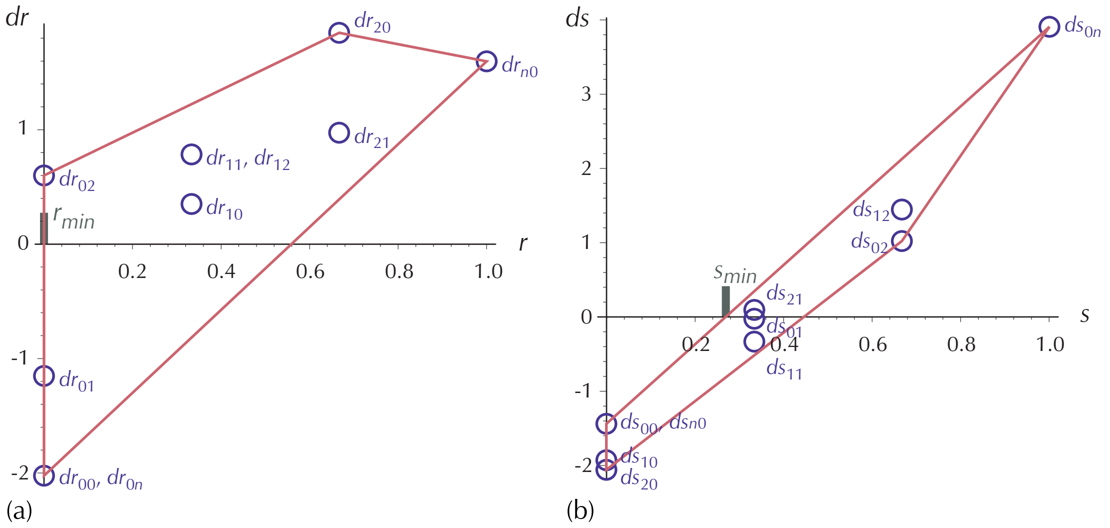


FIGURE 6.6 (a) Projection of functional L_r distance patch along the s direction, its convex hull and the resulting r_{min}.
(b) Projection of functional L_s distance patch along the r direction, its convex hull and the resulting s_{min}.

In a next step, the functional surface is projected along the L_r direction which corresponds to the $r = 0$ parametric line or *s* direction in Figure 6.5. Figure 6.6 shows the projected control points and their convex hull.

The clipping value r_{min} can now be found by intersecting the convex hull of the projected distances with the *r* coordinate axis (see Figure 6.6a). In this example r_{min} evaluates to zero. In the very same manner, a clipping value s_{min} can be found. Figure 6.6b shows the projected L_s distances and their convex hull as well as the resulting s_{min} value. For the determination of t_{min}, the convex hull projection direction is given by the line $t = 0$, which corresponds to the diagonal line in Figure 6.5a.

For parametric values below those minima there cannot be an intersection due to the convex hull property of Bézier patches. Further, the ray does not intersect the patch if

$$r_{min} + s_{min} + t_{min} > 1 .$$

If the parametric candidate domain is small enough to meet the stopping criterion, e.g. its projected size in screen space drops below one pixel, the centroid of r_{min}, s_{min} and t_{min} is taken as the parametric point of intersection.

Otherwise, the patch is subdivided according to the minima found (see Figure 6.7). According to Equation 4.33 in Section 4.4.2, subdivision of triangular Bézier patches with respect to three arbitrary internal points **r**, **s** and **t** yields the sub–patch control points $\mathbf{c}_{ij}$ corresponding to the triangular subdomain $(\mathbf{r}, \mathbf{s}, \mathbf{t})$ as

$$\mathbf{c}_{ij} = \mathbf{d}[\mathbf{r}^{\langle i\rangle}, \mathbf{s}^{\langle j\rangle}, \mathbf{t}^{\langle n-i-j\rangle}] \tag{6.5}$$

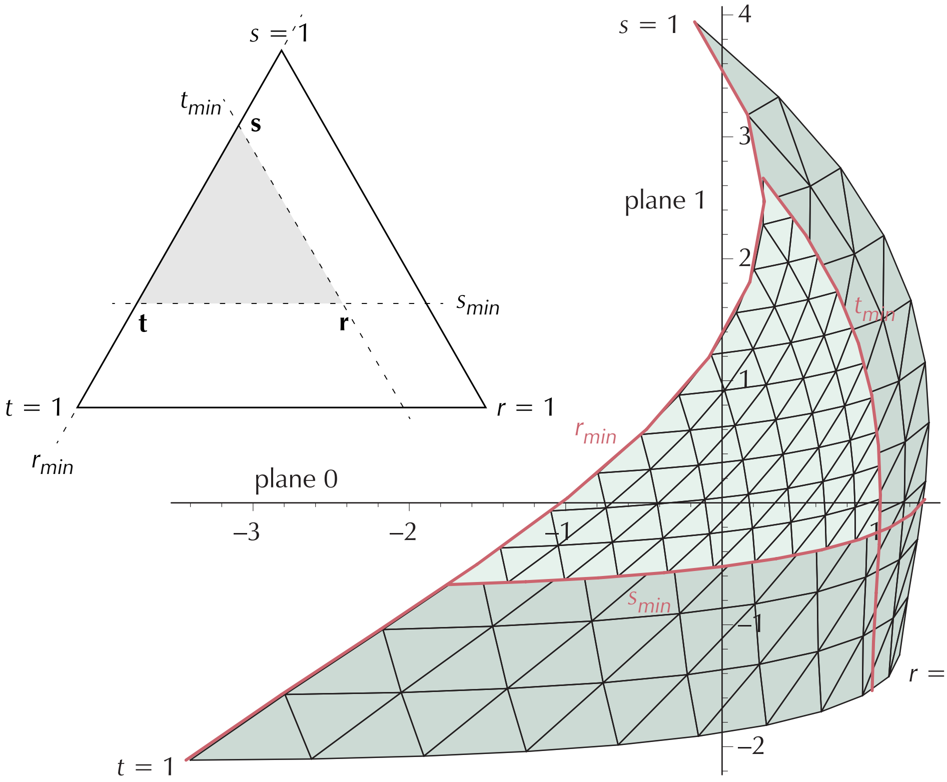


FIGURE 6.7 Subdivision of patch according to clipping minima r_{min}, s_{min} and t_{min}.

$$\begin{aligned} \mathbf{r} &= (1 - s_{min} - t_{min}, s_{min}, t_{min}) \\ \text{with} \qquad \mathbf{s} &= (r_{min}, 1 - r_{min} - t_{min}, t_{min}) \\ \mathbf{t} &= (r_{min}, s_{min}, 1 - r_{min} - s_{min}) \end{aligned}$$

and **d** referring to the control points of the projected patch according to Equation 6.4. Again, the blossom notation in Equation 6.5 can be interpreted as taking i de Casteljau steps with respect to **r**, followed by j steps with respect to **s** and $n - i - j$ steps with respect to **t**. The clipping procedure then continues on the resulting sub–patch as shown in Figure 6.7.

6.2.4 Convex Hull Determination

As we have seen in the previous section, each clipping iteration requires finding the convex hull of three side views of different distance patches. Using Figure 6.6b as an example, we will investigate the procedure of efficient determination of such a convex hull and the corresponding value s_{min}.

In a first step, the maxima h_i and minima l_i for projected points of equal s parameter values are determined. The h_i and l_i define two polylines, P_h and P_l respectively (see Figure 6.8). Finding the convex hull of all points can now be divided into finding the *upper convex hull* from P_h and the *lower convex hull* from P_l. This can be accomplished using an iterative approach. It turns out, however, that a complete determination in general is not required [25].

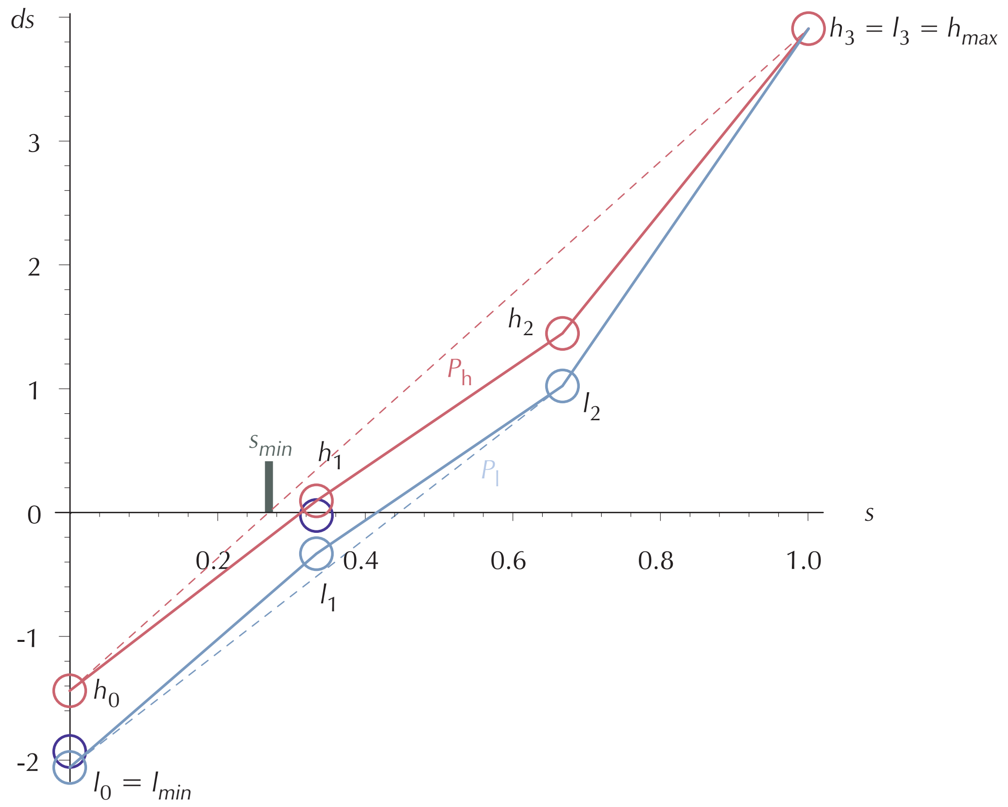


FIGURE 6.8 Computation of upper and lower convex hull.

Before proceeding to further computations one should decide whether the *s* axis intersects the convex hull at all. To this aim, we compute the maximum h_{max} of all h_i and the respective minimum l_{min}. If $h_{max} < 0$ or $l_{min} > 0$ there is no intersection of the convex hull and the ray will not intersect the patch.

For all other cases, three situations have to be dealt with:

$h_0 < 0$: Only the upper convex hull of the points h_0 to h_{max} has to be determined.

$l_0 > 0$: Only the lower convex hull of the points l_0 to l_{min} has to be computed.

$h_0 \geq 0$: $s_{min} = 0$ (as is the case for r_{min} in Figure 6.6a)

6.3 REPORTING WRONG INTERSECTIONS

The original Bézier clipping algorithm can report wrong intersections (see e.g. [24]). Nishita et al. in [99] proposed the following enlargement of the parametric candidate region in order to cope with what they considered numerical problems:

$$s_{min} = 0.99 \cdot s_{min}, \qquad s_{max} = 0.99 \cdot s_{max} + 0.01$$

Campagna and Slusallek in [24] showed that the problem is inherent to the algorithm and not due to numerical round–off. They proposed both a more subtle enlargement of the candidate region and an extension of the algorithm which ensures correct results, unfortunately at the cost of additional computations. After a short description of the problem in the tensor product setting, we will show that a comparable situation in the set–up of triangular patches is very improbable, nay close to impossible.

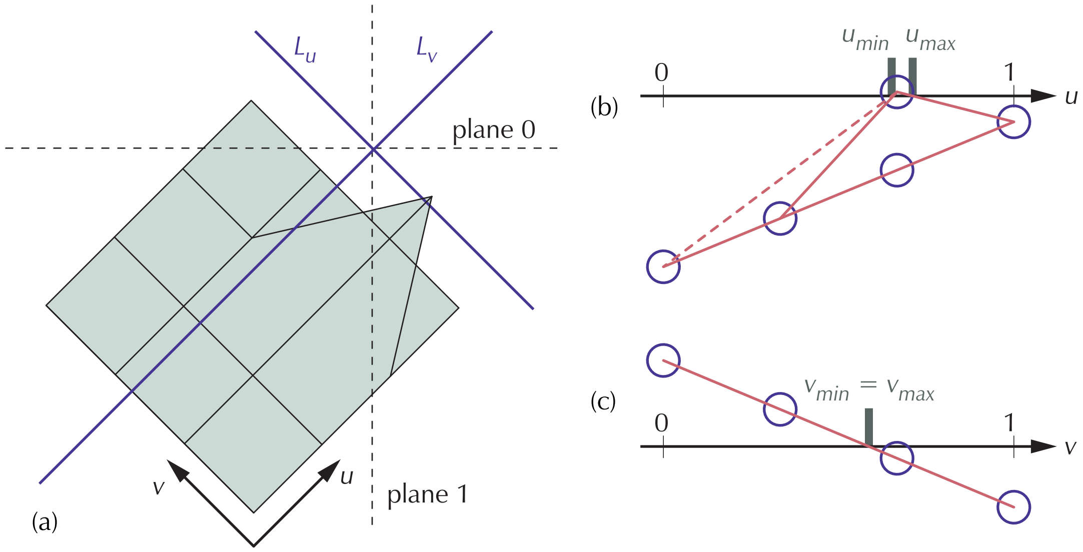


FIGURE 6.9 Potential reporting of wrong intersections for Bézier clipping of a cubic tensor product patch:
(a) Reduced problem and lines L_u, L_v perpendicular to *u*,*v* directions
(b) Projection of L_u distances along *v* and corresponding small candidate region between u_{min} and u_{max}
(c) Projection of L_v distances along *u* and corresponding collapsing candidate region $v_{min} = v_{max}$

In short terms, the error can occur whenever the convex hull of projected distances intersects the corresponding parameter axis even if the patch actually does not. If the candidate region computed due to this intersection happens to be very small, the iteration may terminate and report a wrong intersection if, by coincidence, the second parametric candidate region drops below the threshold, too.

Figure 6.9 illustrates the situation on the example of projected L_u distances. The slight intersection of the convex hull yields a small candidate region between u_{min} and u_{max}. At the same time, the convex hull of projected L_v distances converts to a line which results in a collapsing candidate domain in v. Thus, a potentially wrong intersection is reported. As we can see, the coincidence mentioned above is not very improbable as a collapsing domain results whenever the projection of the patch in one dimension is undistorted as in Figure 6.9 and the convex hull of projected distances consequently converts to a line. Obviously, the error can only occur for non–interpolating control points.

There are several reasons why a similar situation for triangular patches is difficult to construct. First and possibly most important, the algorithm only makes use of r_{min}, s_{min} and t_{min} but does not take into account the respective maxima. Thus, if the three minima in one iteration do not sum up to approximately 1 and therefore cause the iteration to stop, in the following clipping iteration steps the error is likely to disappear. This is due to the fact, that the control nets of subsequent subdivisions approximate the patch better and better. Second, distances are computed with respect to three coordinate directions, which

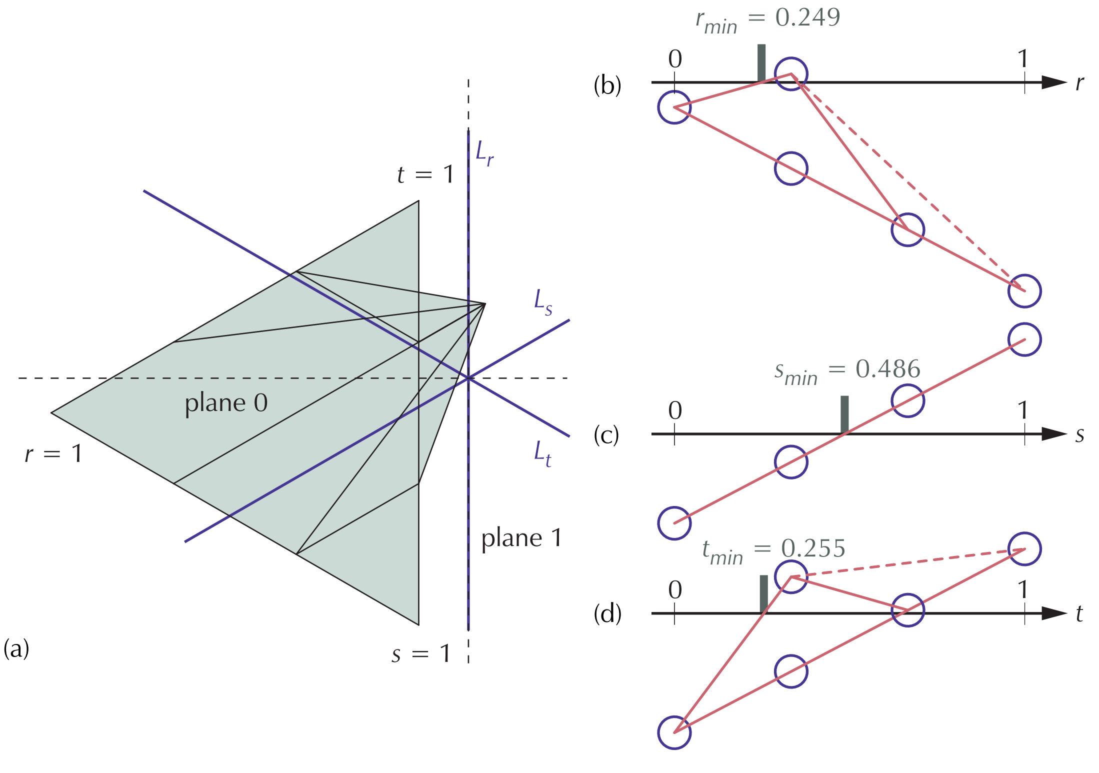


FIGURE 6.10 Potential reporting of wrong intersections for Bézier clipping of a cubic triangular patch:
(a) Reduced problem and lines L_r, L_s and L_t
(b) Projection of L_r distances and corresponding r_{min}
(c) Projection of L_s distances and corresponding s_{min}
(d) Projection of L_t distances and corresponding t_{min}

due to the barycentric setting are linearly dependent and thus to some respect redundant. As a consequence of these two facts, wrong intersections can only occur if both of the following conditions are met:

- In at least one projective view of distances, one non–interpolating control point lies above or below the respective axis and the others do not.
- The minima r_{min}, s_{min} and t_{min} accidentally sum up to approximately but not more than 1.

It is the second condition that makes triangular Bézier clipping far less error–prone. Figure 6.10 clarifies these conditions: The situation is very similar to the tensor product example in Figure 6.9. In the projective view of L_r distances one control point lies slightly above the *r* axis. Further, the convex hull of projected L_s distances converts to a line. The tensor product error conditions are hereby met. In the triangular case, due to the second of the above conditions, an erroneous intersection is reported only if in the third projective view t_{min} happens to evaluate accidentally to approximately $1 - r_{min} - s_{min}$.

The increased error tolerance of triangular Bézier clipping can thus be regarded as a direct consequence of the barycentric formulation. This again emphasizes the more natural nature of the barycentric generalization of Bézier curves compared to its tensor product counterpart.

6.4 RESULTS

As a proof of concept we extended the object oriented ray tracer (OORT) of Wilt [155] to handle triangular Bézier patches. The object oriented design of this ray tracer makes it easy to integrate new types of objects. Unfortunately, its shading capabilities are limited to some extent.

To speed up the Bézier clipping algorithm we additionally implemented a *bounding sphere hierarchy* (BSH) which eliminates the first clipping iterations. Although bounding spheres are far from optimal with respect to tight enclosure of the patch, we have chosen spheres for their simplicity and fast intersection testing. The BSH is built by subdividing the triangular Bézier patch into four sub–patches until a flatness criterion is met (see Figure 6.11). We define the flatness of a patch as its height *h*. The height is computed as the extent of the control points along the normal of the plane spanned by the patch's corner nodes

$$h = \max\left\{0, \max_{0 \le i+j \le n}(\mathbf{n} \cdot \mathbf{b}_{ij})\right\} - \min\left\{0, \min_{0 \le i+j \le n}(\mathbf{n} \cdot \mathbf{b}_{ij})\right\}$$

with the normalized plane normal $\mathbf{n} = \bar{\mathbf{n}} / \|\bar{\mathbf{n}}\|$ and $\bar{\mathbf{n}}$

$$\bar{\mathbf{n}} = (\mathbf{b}_{0n} - \mathbf{b}_{00}) \times (\mathbf{b}_{n0} - \mathbf{b}_{00}).$$

Finding intersections now consists of a BSH traversal and subsequent Bézier clipping on the leaf sub–patch.

In order to test the implementation, we converted the Utah teapot from 32 bicubic tensor product patches to 64 sextic triangular Bézier patches using the approach of Goldman and Filip [55]. On the one hand, patches of degree six are a demanding task for a ray

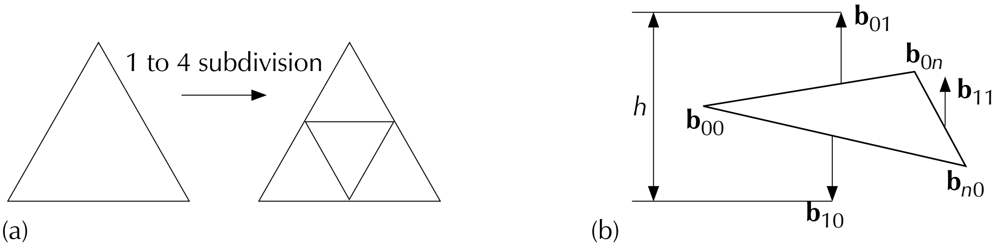


FIGURE 6.11 (a) Subdivision of a patch for hierarchy build–up.
(b) Computing the height of a quadratic patch.

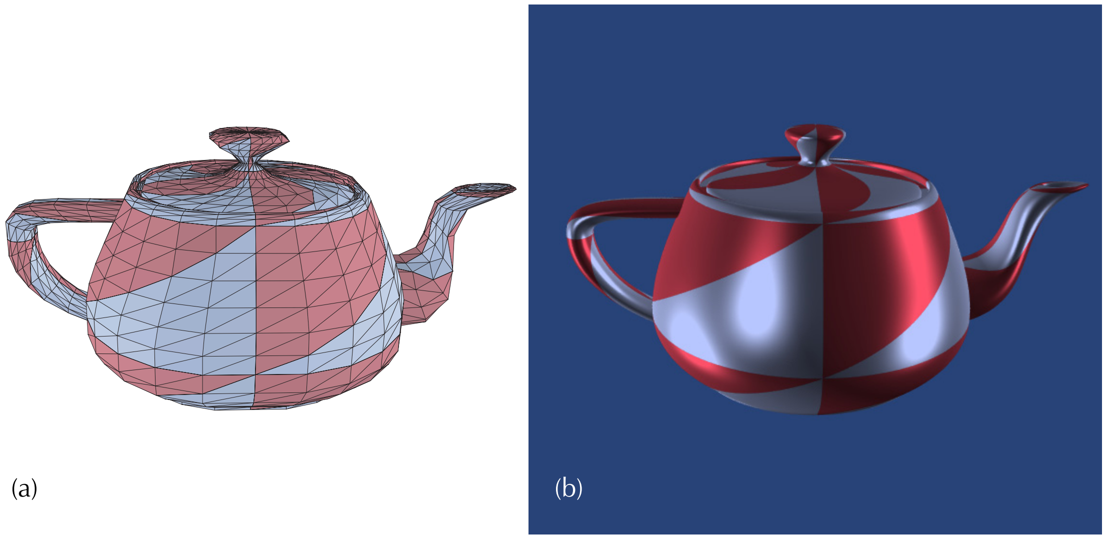


FIGURE 6.12 (a) Control net of Utah teapot made of 64 triangular sextic Bézier patches.
(b) Corresponding ray traced image.

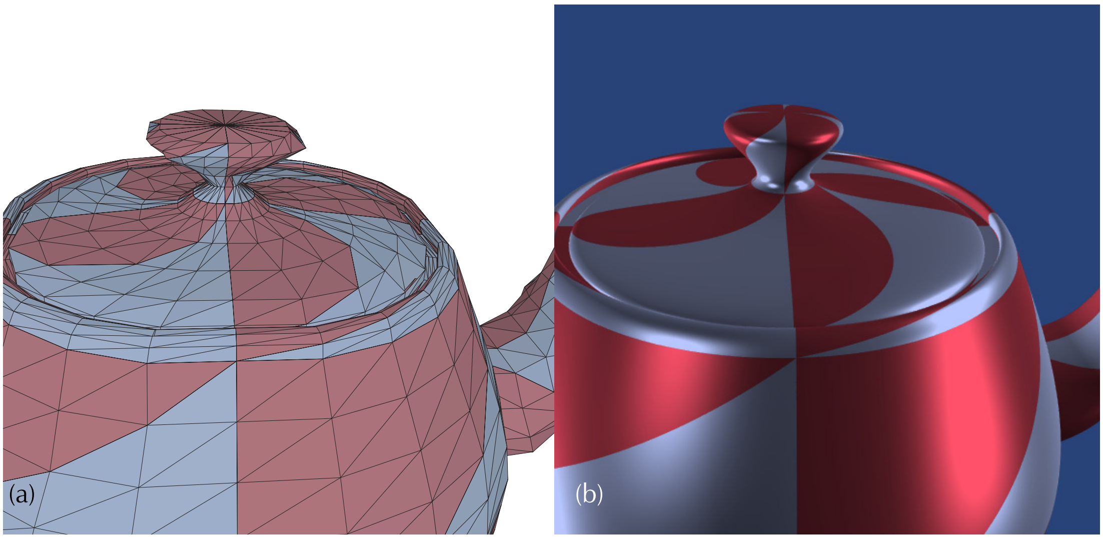


FIGURE 6.13 (a) Close–up of control net at knob.
(b) Corresponding ray traced image.

tracer. On the other hand, the teapot geometry features patches of different curvature. Figure 6.12 shows the sextic control net and the corresponding ray traced image. The pairs of red and blue triangular patches represent the original bicubic rectangular patches.

Figure 6.13 shows a close–up of the knob. As a consequence of degenerated patches in the original model, the four red patches at the knob suffer from a collapsing triangle edge (seven control points coincide). In general, unlike rectangular models, a model made of triangular patches would not require degenerated edges. Such degeneracies need special treatment in the Bézier clipping algorithm since the line L is not defined for collapsing edges. In fact, we in such a situation determine L using the two adjacent control points on the non–degenerated edges.

Due to the very straightforward implementation, it is difficult to make a quantitative performance analysis. What can be stated qualitatively is that Bézier clipping is clearly slower than a pure BSH based ray–patch intersection. Since the cost of the clipping operation grows with the cube of the patch's degree, ray tracing of sextic patches using Bézier clipping is roughly ten times slower than with pure BSH. Combining BSH and Bézier clipping, this ratio drops to about five. For patches of lower degree, the efficiency of Bézier clipping improves.

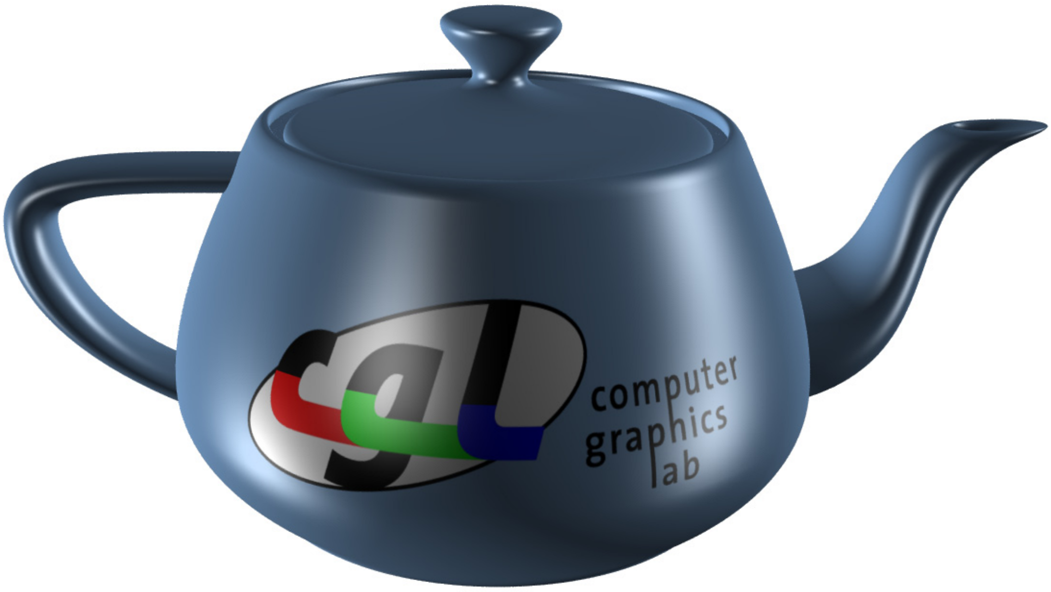


FIGURE 6.14 Textured teapot. **[see Color Plate 1 on page 203]**

In conclusion, triangular Bézier clipping is a valuable approach to ray tracing triangular Bézier patches. We have shown that the accuracy of triangular Bézier clipping equals its tensor product equivalent while its reliability even excels the original algorithm. In combination with a hierarchy of nested bounding volumes, triangular Bézier clipping yields results of highest quality in reasonable time.

CHAPTER 7

A PROTOTYPE FOR SURGERY SIMULATION

Similar to the work in [77], the mathematical model of Chapter 3 and the finite elements according to Chapter 5 are envisioned to be applied in the context of cranio–maxillofacial surgery simulation. In order to test the mathematical model and its discretization as well as the custom–built finite element implementation, we in this chapter describe the implementation of a prototype simulator. The suitability of different finite elements on coarser or finer meshes will be tested and compared by means of this prototype.

The post–simulation of actual surgery was found to be a good test bed since the simulation can be compared against the real surgery outcome. In contrast to previous work [77], we try to limit manual interaction to an absolute minimum by designing the simulator as automatic as possible, both with respect to the registration of data sets and to the determination of jaw movements in accordance with real surgery.

After an overview of the model build–up and the definition of jaw movements, we will focus on specific problems which arise in the implementation of the prototype. The topics include the registration and processing of data sets, skull reconstruction, facial tissue meshing as well as the definition of osteotomies, i.e. the cutting of jaw bones. The presentation of results computed using real patient data will conclude the chapter.

7.1 OVERVIEW OF THE PROTOTYPE

In this section, we first present the prerequisites of the prototype with respect to the data used. Then we describe the model build–up and the determination of jaw movements by means of a data flow diagram.

7.1.1 Prerequisites

Conventional cranio–maxillofacial surgery planning mainly relies on lateral X–ray images and profile sketches of medical artists, sometimes in combination with plaster models or computed tomography (CT) data.

In order to build up a facial model, in principle only a CT scan is required. The high contrast of bone structures easily allows for the extraction of the skull, and the facial surface could equally well be extracted by means of an isosurface generation such as marching cubes [86] (see Figure 7.1c and d). However, patients subject to cranio–maxillofacial surgery most often have metallic implants such as dental fillings and brackets which cause severe artifacts in the CT scan. The right slice in Figure 7.1b reveals the artifacts as star–shaped lines which emanate from the metallic brackets on the teeth. The artifacts basically stem from the very high X–ray absorption of metal which results in a blocking of rays. This on the one hand adversely affects the quality of measurements and on the other hand yields numerical problems in the CT image reconstruction. As a consequence, both the skull (Figure 7.1c) and the facial isosurface (Figure 7.1d) exhibit deficiencies in the jaw region.

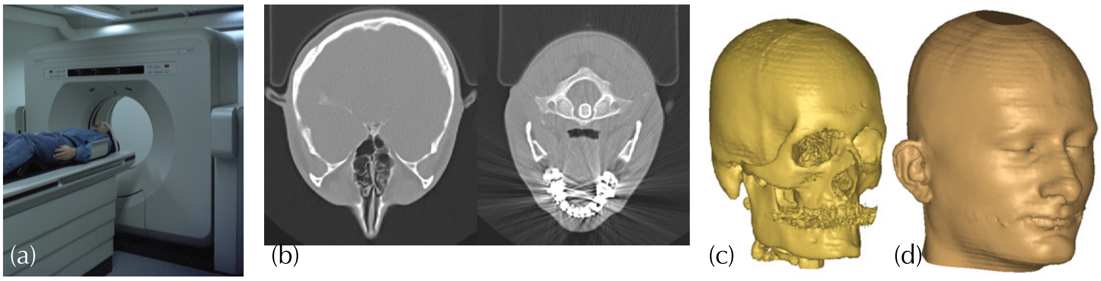


FIGURE 7.1 (a) CT scanner, (b) CT data slices, (c) Skull isosurface, (d) Facial isosurface.

One way to cope with the problem is attempting to reduce the artifacts. However, although yielding promising results, automatic approaches are very slow [105, 151]. Consequently, building up a high quality model necessitates an additional surface laser range scan (LR) as well as special care in extracting the skull surface.

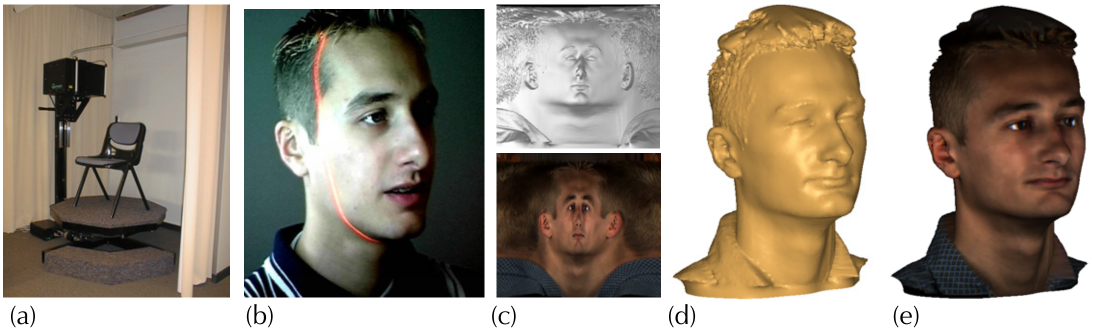


FIGURE 7.2 (a) Laser range scanner, (b) Scanning process, (c) Shaded cylindrical height field and corresponding texture, (d) Resulting surface, (e) Resulting textured surface.

The laser range scanner (Figure 7.2a) simultaneously acquires geometry and texture. The scanning head rotates once around the head of the patient which takes about 15 seconds. CCD–tracking of a vertical red line projected from a low intensity laser (Figure 7.2b) followed by optical triangulation yields the distance information as a cylindrical height field [57]. At the same time, texture is acquired by means of a second CCD chip. Figure 7.2c shows the texture and the corresponding cylindrical distance information as a shaded height field. Figure 7.2d and e show the resulting surface which is of much higher quality than the facial isosurface computed from the CT scan (Figure 7.1d).

As a consequence of post–simulation of real surgery, the prerequisites of the simulator are pre– *and* post–surgical CT in order to determine the jaw movements imposed by the surgical procedure. Of course, post–surgical CT data is not always available. However, the results presented at the end of the chapter are based on real data of a patient who had a post–surgical CT done for various reasons. In addition, for reasons of evaluation, a post–surgical laser range scan was done as well.

7.1.2 Model Build–Up

Figure 7.3 gives a flow chart of the model build–up. Gray shaded steps are explained in more detail in subsequent sections of the chapter. First, we read the patient data comprising both the laser range surface data and the CT volume data of the pre–surgical situation.

In a next step, we run a mesh decimation on the laser range data, which is of high importance in order to reduce the computational costs [123, 75]. The number of triangles used to represent the facial surface together with the imposed boundary conditions determines the number of finite elements in the assembly. The degree of interpolation on the elements decides on the number of local degrees of freedom. As a consequence, the size of the resulting system of equations is directly determined by both the number of finite element and the interpolation used.

In order to build up the model, laser range and CT data must be registered in a common coordinate frame. Whereas the scaling is known from the acquisition devices, rotation and translation have to be determined. The problem is best attacked as the registration of two surfaces: To this aim, we first compute the facial isosurface corresponding to the laser range surface in the CT scan and again run a mesh decimation on the resulting surface. The corresponding steps are depicted in the middle column of Figure 7.3.

Now the problem reduces to the registration of the laser range surface to the facial isosurface within the CT volume. This registration is achieved by means of an iterative closest points algorithm (ICP) which will be discussed in Section 7.2.

Given the exact location of the laser range surface within the volume data set, the skull surface is extracted from the CT scan by means of a cylindrical projection (Section 7.3.1).

Finally we are left with the task of meshing the tissue between the facial and the skull surface (Section 7.3.2).

7.1.3 Jaw Cutting and Displacement Fields

In order to simulate a surgical procedure we not only need a facial model but also need to represent the actual bone cutting and movements. As a consequence of the post–simulation of real surgery, the task is to determine the displacement fields corresponding to real surgery by comparing pre– and post–surgical CT scans. Figure 7.4 illustrates the procedure.

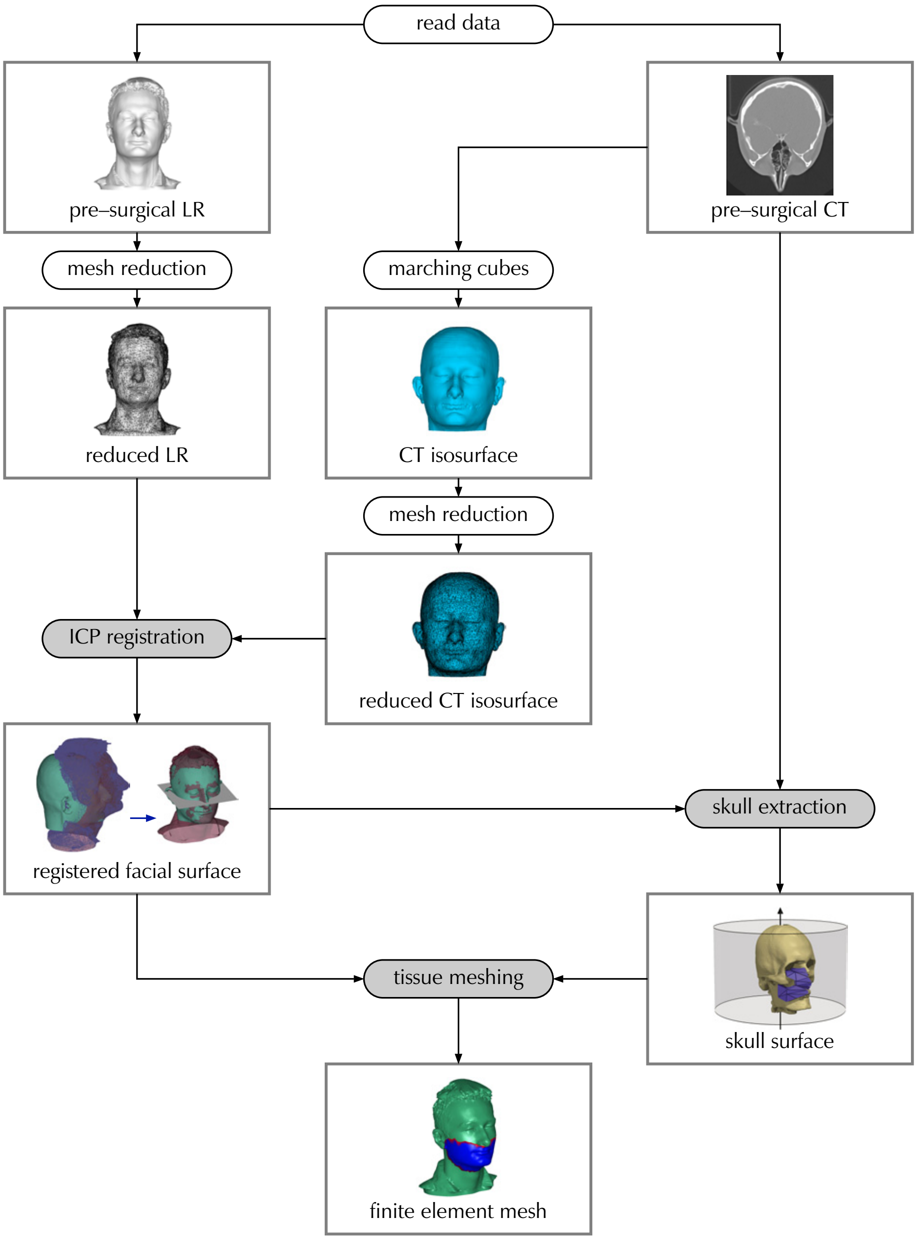


FIGURE 7.3 Flow chart of model build–up.

In a first step, we read the post–surgical CT scan. Of course, before determining the displacement fields, both data sets again have to be aligned in a common coordinate frame, a task which we refer to as CT registration. The image registration approach used to solve this task is outlined in Section 7.4.1.

Using the model's skull surface we now can cut the jaws according to the surgical procedure (see Section 7.4.2).

Finally, we register the cut parts onto the post–surgical skull. Again, this can be done by means of the ICP surface–to–surface registration algorithm already used to register the data sets in the model build–up (Section 7.2). Now, the surgery displacement fields result from the difference between the initial and registered position of the jaws.

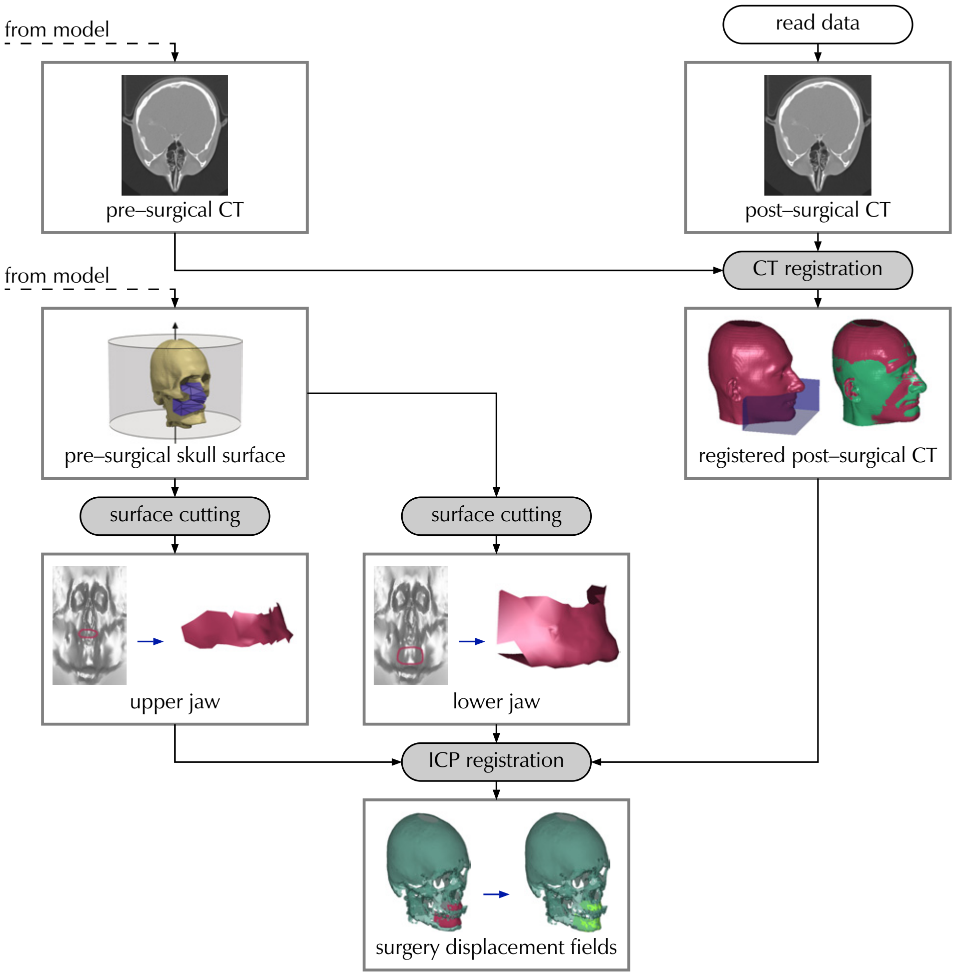


FIGURE 7.4 Flow chart of jaw cutting and determination of displacement fields.

7.2 SURFACE REGISTRATION

As we have seen in the preceding section, we are in need of a surface–to–surface registration procedure both for the model build–up and for the determination of surgery displacement fields. The objective is to find a transformation which brings two surfaces into correspondence in a common coordinate frame. The scaling is known from the acquisition devices and shearing is not permissible in the application context. Therefore we restrict the transformation to a rigid–body motion.

Two additional aspects are worth mentioning: firstly, the surfaces in general are not identical neither topologically nor geometrically such that we do not have a priori knowledge of corresponding points or landmarks. Secondly, the algorithm must be tolerant against outliers, i.e. points on either of the two surfaces which obviously do not have a correspondence. Both artifacts due to scanning deficiencies as well as non–corresponding regions from different clipping are referred to as outliers.

Various approaches have been followed in order to solve the surface registration problem. Besides truly manual approaches, many semi–automatic methods are based on landmarks — corresponding points on both surfaces — which must be set interactively by the user. Unfortunately, the accuracy of the registration heavily depends on the quality, i.e. the proper correspondence of the landmarks set. As a consequence, automatic approaches attempt to solve the problem without reverting to landmarks. Thus, the problem can be regarded as being twofold:

- Find the transformation given pairs of measurements, i.e. corresponding points.
- Find corresponding point sets and iteratively improve upon the quality of correspondence.

Three approaches tackling the entire problem are worth mentioning: Neugebauer proposed an automatic matching procedure for range images in the context of three–dimensional object localization [98]. He came up with a Levenberg–Marquardt iteration to solve the registration problem stated as a least–squares optimization task. Gander and Sourlier presented an automatic procedure for the best–fit of sculptured surfaces, i.e. parametric surfaces [50]. Besl and McKay introduced the *iterative closest points* algorithm which can be regarded as a general–purpose, representation independent method for the registration of three–dimensional shapes [11].

With various applications in mind, the first problem has widely and partly independently been investigated in a broad selection of research fields. In photogrammetry, the problem is referred to as that of *absolute orientation* [130, 124]. Given the coordinates of a number of points measured in two different Cartesian coordinate systems, the objective is to recover the transformation between the two coordinate systems. The problem arises typically in relating the stereo model from pairs of aerial photographs to a geodetic coordinate system. In machine vision and robotics, a similar problem is encountered when determining the *hand–eye transform* between measurements in a camera coordinate system and the coordinate system attached to a mechanical manipulator or to the world [64, 65, 66].

Except from solutions for specific configurations or a fixed number of measurements, the solutions mainly differ with respect to the way they attempt to solve the inherent least–squares problem. We can distinguish between quaternion–based approaches [46, 65] and methods which use the singular value decomposition [62, 61, 49] in order to find the orthogonal rotation matrix [56, 122]. The singular value approach, based on the cross–covariance matrix of two point distributions, does generalize easily to n dimensions. In three dimensions, we prefer the quaternion–based approach since reflections are not desired [11, 49].

In the next section we will describe the solution of Horn, who gave a closed–form solution to the problem of absolute orientation using unit quaternions [65]. He also pointed out to an equivalent closed–form solution using orthonormal matrices [66]. In order to find and improve on the quality of corresponding points we use the iterative closest points algorithm [11] which is sketched in Section 7.2.2.

7.2.1 Absolute Orientation using Unit Quaternions

Before presenting the solution to the problem of absolute orientation according to Horn [65], we will first restate some basic principles about quaternions and their use with respect to the representation of rotation. After stating the objective function to minimize, we will then be ready to give the solution both for the optimal translation and rotation.

Quaternions. Quaternions were discovered by William Rowan Hamilton in the nineteenth century [58, 59]. The obvious interpretation of real numbers lying on a line finds its two–dimensional equivalent in the representation of complex numbers in the complex plane. It is common practice to represent complex numbers as $a + ib$, where *a* and *b* are the real and imaginary part, respectively, and *i* denotes the square root of -1.

It is obvious to attempt introducing triples as an extended number system whose numbers represent points in three–dimensional space. These triples in addition to the real and imaginary part of complex numbers would feature a second distinct and independent square root of -1, *j*. However, in contrast to the apparent generalization of addition and subtraction as component–wise operations, Hamilton found it impossible to define multiplication on these triples analogously to that on complex numbers. This is best seen by recalling that the complex multiplication can be interpreted in polar form as a scaling and a rotation around the origin. In three dimensions, the interpretation of multiplication as a combination of scaling and rotation necessitates the indication of *two* additional parameters specifying the direction of the axis of rotation. Consequently, we need to introduce *four* parameters.

A solution to this dilemma can be found by attempting to extend the operation of conjugation from complex numbers to the triples mentioned above. The conjugate of a complex number $z = a + ib$ is given by $z^* = a - ib$. Multiplication of a complex number and its conjugate always yields a real number

$$zz^* = a^2 + b^2$$

which is the square of the length or *modulus* of the polar form, $r = \sqrt{a^2 + b^2}$. A similar approach on the triples yields conjugation as

$$z = a + ib + jc \qquad z^* = a - ib - jc$$

and consequently

$$zz^* = a^2 + b^2 + c^2 - 2ijbc.$$

Hamilton realized that the extra term $-2ijbc$ is actually more properly regarded as two terms, $-ijbc$ and $-jibc$. Now, breaking the commutative law of multiplication and assuming $ij = -ji$ forces the term to vanish. Applying the associative law of multiplication allows deeper insight into the value of ij:

$$ij \cdot ij = i(ji)j = -i(ij)j = -(i^2)(j^2) = -(-1)(-1) = -1$$

It turns out that ij is yet another root of -1, independent of both *i* and *j*, and therefore is to be regarded as a fourth imaginary unit *k* which turns the triples into the quadruples known as *quaternions:*

$$\dot{q} = q_0 + q_x i + q_y j + q_z k \qquad (7.1)$$

The quaternions are thus to be regarded as an extension of the complex numbers, with the exception that quaternion multiplication is noncommutative. From the above considerations it is easy to see that the following relationships hold:

$$i^2 = j^2 = k^2 = ijk = -1 \qquad \begin{matrix} ij = k & jk = i & ki = j \\ ji = -k & kj = -i & ik = -j \end{matrix} \qquad (7.2)$$

It is interesting to note from Equation 7.2 that the products of *i*, *j*, and *k* of -1 behave similarly to cross–products of the Cartesian three–dimensional unit base vectors.

Quaternions according to Equation 7.1 can also be represented as vectors with four components or as a composite of a scalar and an ordinary three–dimensional vector:

$$\begin{aligned} \dot{\mathbf{q}} &= [q_0, q_x, q_y, q_z] \\ &= [q_0, \mathbf{q}] \qquad \text{with} \qquad \mathbf{q} = [q_x, q_y, q_z] \end{aligned} \tag{7.3}$$

Using the quaternions, $\dot{p}$, $\dot{q}$ and $\dot{r}$ we will state the following important relationships and definitions of quaternions:

- *Addition, subtraction, multiplication by a real number*

$$\begin{aligned} \dot{q} + \dot{p} &= (q_0 + p_0) + i(q_x + p_x) + j(q_y + p_y) + k(q_z + p_z) \\ \dot{q} - \dot{p} &= (q_0 - p_0) + i(q_x - p_x) + j(q_y - p_y) + k(q_z - p_z) \\ r\dot{q} &= \dot{q}r = rq_0 + i(rq_x) + j(rq_y) + k(rq_z) \end{aligned}$$

- *Conjugate*

$$\dot{q}^* = q_0 - iq_x - jq_y - kq_z$$

- *Dot product*

$$\dot{q} \cdot \dot{p} = q_0 p_0 + q_x p_x + q_y p_y + q_z p_z$$

The square of the *magnitude* is the dot product of the quaternion with itself:

$$\|\dot{q}\|^2 = \dot{q} \cdot \dot{q}$$

A *unit quaternion* is a quaternion whose magnitude equals 1.

- *Product*

$$\begin{aligned} \dot{q}\dot{p} = {} & (q_0 p_0 - q_x p_x - q_y p_y - q_z p_z) \\ & + i(q_0 p_x + q_x p_0 + q_y p_z - q_z p_y) \\ & + j(q_0 p_y - q_x p_z + q_y p_0 + q_z p_x) \\ & + k(q_0 p_z + q_x p_y - q_y p_x + q_z p_0) \end{aligned} \tag{7.4}$$

Since quaternion multiplication is noncommutative, in general $\dot{q}\dot{p} \neq \dot{p}\dot{q}$. The product $\dot{p}\dot{q}$ is very similar, but six of the signs are changed, as readily can be verified.

Using the vector notation according to Equation 7.3, Equation 7.4 immediately can be rewritten as the product of an orthogonal 4×4 matrix and a vector with four components. Either of the quaternions in the product may be expanded into an orthogonal 4×4 matrix as follows:

$$\dot{q}\dot{p} = \begin{bmatrix} q_0 & -q_x & -q_y & -q_z \\ q_x & q_0 & -q_z & q_y \\ q_y & q_z & q_0 & -q_x \\ q_z & -q_y & q_x & q_0 \end{bmatrix} \dot{\mathbf{p}} = \mathbf{Q}\dot{\mathbf{p}} \text{ or } \dot{p}\dot{q} = \begin{bmatrix} q_0 & -q_x & -q_y & -q_z \\ q_x & q_0 & q_z & -q_y \\ q_y & -q_z & q_0 & q_x \\ q_z & q_y & -q_x & q_0 \end{bmatrix} \dot{\mathbf{p}} = \bar{\mathbf{Q}}\dot{\mathbf{p}} \tag{7.5}$$

$\overline{\mathbf{Q}}$ differs from $\mathbf{Q}$ in that the lower–right–hand 3×3 submatrix is transposed. This again emphasizes the noncommutative nature of quaternion multiplication.

The 4×4 matrices associated with the conjugate are just the transposes of the matrices associated with the quaternion itself. Since these matrices are orthogonal, the products $\mathbf{Q}\mathbf{Q}^T = (\dot{q} \cdot \dot{q})\mathbf{I}$ are diagonal with $\mathbf{I}$ the 4×4 identity matrix. Correspondingly the product of $\dot{q}$ and $\dot{q}^*$ is real, just as with the complex numbers,

$$\dot{q}\dot{q}^* = (q_0^2 + q_x^2 + q_y^2 + q_z^2) = \dot{q} \cdot \dot{q} = \|\dot{q}\|^2 .$$

An immediate conclusion is that a nonzero quaternion has an inverse

$$\dot{q}^{-1} = (1/\|\dot{q}\|^2)\dot{q}^* .$$

From this equation we see that in the case of a unit quaternion, the inverse is just the conjugate.

The inverse can be used to define division. However, it could be either defined as $\dot{q}/\dot{p} = \dot{q}\dot{p}^{-1}$ or as $\dot{q}/\dot{p} = \dot{p}^{-1}\dot{q}$.

Using the alternate composite quaternion notation of Equation 7.3, we can write the product of two quaternions as

$$\dot{q}\dot{p} = [q_0 p_0 - \mathbf{q} \cdot \mathbf{p}, q_0 \mathbf{p} + p_0 \mathbf{q} + \mathbf{q} \times \mathbf{p}]$$

which again illustrates that quaternion multiplication is not commutative, since the cross–product of two vectors is not commutative.

- *Useful properties of products*

Dot products are preserved, since the matrices associated with quaternions are orthogonal; thus we have

$$\begin{aligned}(\dot{q}\dot{p}) \cdot (\dot{q}\dot{r}) &= (\mathbf{Q}\dot{\mathbf{p}}) \cdot (\mathbf{Q}\dot{\mathbf{r}}) = (\mathbf{Q}\dot{\mathbf{p}})^T \cdot (\mathbf{Q}\dot{\mathbf{r}}) = \dot{\mathbf{p}}^T\mathbf{Q}^T\mathbf{Q}\dot{\mathbf{r}} \\ &= \dot{\mathbf{p}}^T(\dot{\mathbf{q}} \cdot \dot{\mathbf{q}})\mathbf{I}\dot{\mathbf{r}} = (\dot{q} \cdot \dot{q})(\dot{p} \cdot \dot{r})\end{aligned} \tag{7.6}$$

which, in the case when $\dot{q}$ is a unit quaternion, is just $\dot{p} \cdot \dot{r}$.

The following relationship holds for arbitrary quaternions, a result that will be used later:

$$(\dot{p}\dot{q}) \cdot \dot{r} = \dot{p} \cdot (\dot{r}\dot{q}^*) \tag{7.7}$$

- *Representation of vectors*

Vectors can be represented by purely imaginary quaternions: for $\mathbf{r} = [x, y, z]^T$

$$r = 0 + ix + jy + kz \tag{7.8}$$

can be used as its quaternion equivalent. Similarly, purely real quaternions can be used to represent scalars. The 4×4 matrices associated with purely imaginary quaternions are skew symmetric. Therefore, in this special case, we have

$$\mathbf{Q}^T = -\mathbf{Q} \text{ and } \overline{\mathbf{Q}}^T = -\overline{\mathbf{Q}} .$$

◗ *Unit quaternions and rotation*

Rotation does neither change the length of a vector nor does it change angles between vectors. Thus, dot products are preserved under rotation. In contrast to reflection, rotation does not alter cross–products, either.

From Equation 7.6 we know that for a unit quaternion $\dot{q}$ we have $(\dot{q}\dot{p}) \cdot (\dot{q}\dot{r}) = \dot{p} \cdot \dot{r}$. Consequently, we can represent rotation by using unit quaternions given that we can find a way of mapping vectors given as purely imaginary quaternions into purely imaginary quaternions (Equation 7.8) while preserving dot and cross–products. Simple multiplication cannot be used since in general the product of a unit quaternion and a purely imaginary quaternion is not purely imaginary anymore. However, the result of the composite product

$$\dot{r}' = \dot{q}\dot{r}\dot{q}^* \tag{7.9}$$

is purely imaginary. This can be shown by expanding

$$\dot{q}\dot{r}\dot{q}^* = (\mathbf{Q}\dot{\mathbf{r}})\dot{q}^* = \overline{\mathbf{Q}}^T(\mathbf{Q}\dot{\mathbf{r}}) = (\overline{\mathbf{Q}}^T\mathbf{Q})\dot{\mathbf{r}}$$

where transposing $\overline{\mathbf{Q}}$ corresponds to conjugation. Inspecting $\overline{\mathbf{Q}}^T\mathbf{Q}$

$$\overline{\mathbf{Q}}^T\mathbf{Q} = \begin{bmatrix} \dot{q} \cdot \dot{q} & 0 & 0 & 0 \\ 0 & (q_0^2 + q_x^2 - q_y^2 - q_z^2) & 2(q_xq_y - q_0q_z) & 2(q_xq_z + q_0q_y) \\ 0 & 2(q_yq_x + q_0q_z) & (q_0^2 - q_x^2 + q_y^2 - q_z^2) & 2(q_yq_z - q_0q_x) \\ 0 & 2(q_zq_x - q_0q_y) & 2(q_zq_y + q_0q_x) & (q_0^2 - q_x^2 - q_y^2 + q_z^2) \end{bmatrix}$$

reveals that $\dot{r}'$ in Equation 7.9 will be purely imaginary if $\dot{r}$ is. For a unit quaternion $\dot{q}$ both $\mathbf{Q}$ and $\overline{\mathbf{Q}}$ are orthonormal and by definition $\dot{q} \cdot \dot{q} = 1$. Consequently, the lower–right–hand 3×3 submatrix of $\overline{\mathbf{Q}}^T\mathbf{Q}$ will be orthonormal, too [65]. In fact, it is the familiar rotation matrix **R**, $\mathbf{r}' = \mathbf{R}\mathbf{r}$. Horn in [65] proves, that cross–products are also preserved by Equation 7.9.

It is interesting to note that $(-\dot{q})\dot{r}(-\dot{q}^*) = \dot{q}\dot{r}\dot{q}^*$, so that $-\dot{q}$ represents the same rotation as does $\dot{q}$.

Finally, it is illustrative to relate the rotation quaternion to a rotation by an angle ϕ about the axis defined by the unit vector $\omega = [\omega_x, \omega_y, \omega_z]$ as

$$\dot{q} = \cos\frac{\phi}{2} + \sin\frac{\phi}{2}(i\omega_x + j\omega_y + k\omega_z).$$

Notation and objective function. Recall that the objective is to solve the problem of absolute orientation by finding a transformation that maps a *data* surface P

$$P = \{\mathbf{p}_i, i = 1\ldots n\}$$

onto a *model* surface X

$$X = \{\mathbf{p}_i, i = 1\ldots m\}.$$

In addition to the input surfaces P and X we define the *closest points* surface as the point set

$$Y = \{\mathbf{y}_i, i = 1 \ldots n\}.$$

It is the set of corresponding points on the model surface X with respect to the points in P. In other words, each point $\mathbf{y}_i$ is the closest point to $\mathbf{p}_i$ on the model X. As a consequence, it is represented by the same number of points n as the data surface. It is this set of correspondences the iterative closest points algorithm in Section 7.2.2 will attempt to improve upon by transforming P onto X.

The sought–after transformation is parametrized by the *registration vector* with seven components

$$\mathbf{q} = [\dot{\mathbf{q}}_R | \mathbf{q}_T]$$

where $\dot{\mathbf{q}}_R$ and $\mathbf{q}_T$ represent the rotation quaternion and translation vector, respectively.

The *objective function* to minimize is given as the sum of squared differences between the closest points $\mathbf{y}_i$ and the rotated and translated data points $\mathbf{p}_i$

$$f(\mathbf{q}) = \sum_{i=1}^{n} \left\| \mathbf{y}_i - \mathbf{R}_{\dot{\mathbf{q}}_R}(\mathbf{p}_i) - \mathbf{q}_T \right\|^2 \tag{7.10}$$

where $\mathbf{R}_{\dot{\mathbf{q}}_R}(\cdot)$ denotes rotation according to the rotation quaternion $\dot{\mathbf{q}}_R$.

It is best to represent both surfaces Y and P with respect to their centroids. We refer to the centroid coordinates using the prime sign on $\mathbf{y}$ and $\mathbf{p}$ as

$$\mathbf{y}_i' = \mathbf{y}_i - \mu_Y \quad \text{with} \quad \mu_Y = \frac{1}{n}\sum_{i=1}^{n} \mathbf{y}_i$$

$$\mathbf{p}_i' = \mathbf{p}_i - \mu_P \quad \text{with} \quad \mu_P = \frac{1}{n}\sum_{i=1}^{n} \mathbf{p}_i$$

The objective function with respect to the centroid coordinates now includes two corrective terms which follow from the linearity of rotation and simple algebra to

$$f(\mathbf{q}) = \sum_{i=1}^{n} \left\| \mathbf{y}_i' - \mathbf{R}_{\dot{\mathbf{q}}_R}(\mathbf{p}_i') \underbrace{- \mathbf{q}_T + \mu_Y}_{\substack{\text{corrective term} \\ \text{for } \mathbf{y}_i'}} \overbrace{- \mathbf{R}_{\dot{\mathbf{q}}_R}(\mu_P)}^{\substack{\text{corrective term} \\ \text{for } \mathbf{p}_i'}} \right\|^2 . \tag{7.11}$$

Expanding the objective function in Equation 7.11 yields

$$f(\mathbf{q}) = \underbrace{\sum_{i=1}^{n} \left\| \mathbf{y}_i' - \mathbf{R}_{\dot{\mathbf{q}}_R}(\mathbf{p}_i') \right\|^2}_{A} \underbrace{- 2\mathbf{q}_T' \sum_{i=1}^{n} \mathbf{y}_i' - \mathbf{R}_{\dot{\mathbf{q}}_R}(\mathbf{p}_i')}_{B} + \underbrace{n \left\| \mathbf{q}_T' \right\|^2}_{C} \tag{7.12}$$

where $\mathbf{q}_T' = \mathbf{q}_T - \mu_Y + \mathbf{R}_{\dot{\mathbf{q}}_R}(\mu_P)$. The terms *A*, *B* and *C* in Equation 7.12 will be the starting point for the further optimizations.

Optimal translation. Looking for the optimal translation, from the three terms *A*, *B* and *C* in Equation 7.12 we see that *A* is independent of $\mathbf{q}_T'$ whereas *B* sums up to zero due to the centroid coordinates. Therefore, the only term at disposal for the minimization with respect to translation is the term *C*. Thus, the objective function will be minimal for $C = 0$ which is the case for $\mathbf{q}_T' = 0$. From the definition of $\mathbf{q}_T' = \mathbf{q}_T - \mu_Y + \mathbf{R}_{\dot{\mathbf{q}}_R}(\mu_P)$ we find that the optimal translation is the difference of the centroids after rotation of the data surface

$$\mathbf{q}_T = \mu_Y - \mathbf{R}_{\dot{\mathbf{q}}_R}(\mu_P).$$

Optimal rotation. The optimal rotation will have to minimize the terms *A* and *B* of the expanded objective function in Equation 7.12. Given that the surfaces have equal scaling and knowing that the centroids will coincide, *A* and *B* will be minimal for a maximal dot product of the $\mathbf{y}_i$ and the rotated $\mathbf{p}_i$

$$\max_{\dot{\mathbf{q}}_R} \sum_{i=1}^{n} \mathbf{y}_i' \cdot \mathbf{R}_{\dot{\mathbf{q}}_R}(\mathbf{p}_i').$$

We choose to represent rotation using unit quaternions instead of rotation matrices. This is advantageous since keeping the rotation orthonormal only requires the normalization of the quaternion instead of computing the closest orthonormal matrix. Alternatively, the orthogonal rotation matrix could be computed as the solution of the *Procrustes* problem [49]. However, due to the inherent singular value decomposition, reflections would have to be taken special care of. Thus, from Equation 7.9, we have to maximize

$$\max_{\dot{q}_R} \sum_{i=1}^{n} \dot{y}_i' \cdot \dot{q}_R \dot{p}_i' \dot{q}_R^*. \tag{7.13}$$

Using the relationship in Equation 7.7 we can rewrite Equation 7.13 to

$$\max_{\dot{q}_R} \sum_{i=1}^{n} \dot{q}_R \dot{p}_i' \cdot \dot{y}_i' \dot{q}_R.$$

Now, rewriting quaternion multiplication according to Equation 7.5 and reordering the dot product similar to Equation 7.6 yields

$$\begin{aligned} & \max_{\dot{q}_R} \sum_{i=1}^{n} \overline{\mathbf{P}}_i' \dot{\mathbf{q}}_R \cdot \mathbf{Y}_i' \dot{\mathbf{q}}_R \\ = {} & \max_{\dot{q}_R} \sum_{i=1}^{n} \dot{\mathbf{q}}_R^T \overline{\mathbf{P}}_i'^T \mathbf{Y}_i' \dot{\mathbf{q}}_R \\ = {} & \max_{\dot{q}_R} \dot{\mathbf{q}}_R^T \underbrace{\left(\sum_{i=1}^{n} \overline{\mathbf{P}}_i'^T \mathbf{Y}_i' \right)}_{\mathbf{N}} \dot{\mathbf{q}}_R \end{aligned} \tag{7.14}$$

We are now left with the problem of maximizing the quadratic form $\dot{\mathbf{q}}_R^T \mathbf{N} \dot{\mathbf{q}}_R$. To this aim it is expedient to know the properties of **N**. **N** is computed from the point–to–point correspondences between the data surface *P* and the closest points surface *Y*. It is a symmetric 4×4 matrix which consequently has four real eigenvalues λ_1, λ_2, λ_3, λ_4 and four corresponding orthogonal unit eigenvectors $\mathbf{e}_i$

$$\mathbf{N}\mathbf{e}_i = \lambda_i \mathbf{e}_i \qquad \text{for} \qquad i = 1, 2, \ldots 4 .$$

The $\mathbf{e}_i$ span the four–dimensional space of quaternions as

$$\dot{\mathbf{q}}_R = \alpha_1 \mathbf{e}_1 + \alpha_2 \mathbf{e}_2 + \alpha_3 \mathbf{e}_3 + \alpha_4 \mathbf{e}_4 . \tag{7.15}$$

Since the eigenvectors are orthogonal and knowing that $\dot{\mathbf{q}}_R$ is required to be of unit length we can state

$$\dot{\mathbf{q}}_R \cdot \dot{\mathbf{q}}_R = \alpha_1^2 + \alpha_2^2 + \alpha_3^2 + \alpha_4^2 = 1 . \tag{7.16}$$

Next, we can write

$$\mathbf{N}\dot{\mathbf{q}}_R = \alpha_1 \lambda_1 \mathbf{e}_1 + \alpha_2 \lambda_2 \mathbf{e}_2 + \alpha_3 \lambda_3 \mathbf{e}_3 + \alpha_4 \lambda_4 \mathbf{e}_4$$

since the $\mathbf{e}_i$ are the eigenvectors of **N**. Consequently, we have

$$\dot{\mathbf{q}}_R^T \mathbf{N} \dot{\mathbf{q}}_R = \dot{\mathbf{q}}_R \cdot \mathbf{N} \dot{\mathbf{q}}_R = \alpha_1^2 \lambda_1 + \alpha_2^2 \lambda_2 + \alpha_3^2 \lambda_3 + \alpha_4^2 \lambda_4 . \tag{7.17}$$

Suppose that the eigenvalues have been arranged in ascending order $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \lambda_4$. Then we from Equations 7.16 and 7.17 find that the quadratic form cannot become larger than the most positive eigenvalue

$$\dot{\mathbf{q}}_R^T \mathbf{N} \dot{\mathbf{q}}_R \leq \alpha_1^2 \lambda_1 + \alpha_2^2 \lambda_1 + \alpha_3^2 \lambda_1 + \alpha_4^2 \lambda_1 = \lambda_1 .$$

Further, we conclude that the maximum $\dot{\mathbf{q}}_R^T \mathbf{N} \dot{\mathbf{q}}_R = \lambda_1$ is attained for $\alpha_1 = 1$ and $\alpha_2 = \alpha_3 = \alpha_4 = 0$ and thus we from Equation 7.15 find the sought–after quaternion as

$$\dot{\mathbf{q}}_R = \mathbf{e}_1 .$$

In summary, the unit eigenvector corresponding to the most positive eigenvalue of **N** maximizes the quadratic form in Equation 7.14 and thus is to be chosen as the optimal rotation quaternion.

Horn gives a closed–form solution for the computation of the eigenvalues by taking advantage of special properties of the coefficients of the fourth–order characteristic polynomial $\det(\mathbf{N} - \lambda \mathbf{I}) = 0$ where **I** is the 4×4 identity matrix in [65]. However, in our implementation we use the Jacobi method which is very efficient for 4×4 matrices since only few Jacobi rotations are required to reach convergence [112].

7.2.2 Iterative Closest Points Algorithm

Now that we know how to compute the optimal translation and rotation given corresponding points on the input surfaces, we are ready to state the iterative closest points algorithm. In each iteration we will need to find the closest points surface *Y* which by definition consists of the closest points on the model corresponding to the data surface points.

We then compute the optimal translation and rotation, and transform the data surface accordingly. After computing the new total distance d_k between the surfaces according to Equation 7.10 we can either stop due to a convergence criterion or iterate the procedure.

Convergence is reached when the mean square error, the total distance, does not decrease between two iterations, $d_k \geq d_{k-1}$, or drops below a preset threshold. Besl and McKay in [11] state and prove a convergence theorem for the iterative closest points algorithm. The key idea is that the least squares registration generically reduces the average distance between corresponding points during each iteration, whereas the closest points determination generically reduces the distance for each point individually. Consequently, the total distance and thus average distance is reduced, too. However, convergence is only local in that the algorithm converges monotonically to a local minimum. Therefore, in order to reach the global minimum, a reasonably good initial alignment is required.

The iterative closest points algorithm can be accelerated by linearly or quadratically extrapolating in 7–space from the sequence of registration vectors generated in the course of the iteration. For details the reader is referred to [11].

In our implementation we additionally incorporated an outlier elimination mechanism which after convergence eliminates obvious outliers and thereafter restarts the registration until convergence. Outliers can be identified as corresponding points whose distance exceeds the average of point distances about a certain, user–adjustable factor.

Last but not least, the rather bad resemblance of the laser range surface and the facial isosurface computed from the CT scan necessitates an initial masking step. Similar to the outlier elimination, masking neglects regions of the data surface which are known not to match the model well or which do not have a correspondence at all.

The following pseudo–code sequence illustrates the algorithm:

```
read(model);         // read model surface
read(data);          // read data surface
read(data_mask);     // read data surface mask
mask(data);          // mask data
d_new = FLT_MAX;     // initialize distance
// start outlier elimination iteration
do {
  // start icp iteration
  do {
    d_old = d_new; // remember old total distance
    closest_points = closest_points(data, model);
    q = compute_optimal_xform(data, closest_points);
    xform(data, q);
    d_new = total_distance(data, model); // new distance
  }
  while (d_new < d_old); // loop for convergence
}
while (identify_outliers()); // loop outlier elimination
```

Computing the point correspondences, i.e. the closest points surface, is by far the most expensive part of the nested iteration. In a brute force approach, this step would require testing each data surface point against all the triangles of the model, and thus would be of $O(n \cdot 2m)$, given that each of roughly $2m$ triangles of the model has to be searched for the closest points to n data surface vertices. However, spatial data structures such as an

R^* – or *kd*–tree [121, 120] may be used to reduce the computational costs of such proximity problems to roughly $n \cdot \log(2m)$ [111]. However, we found it very efficient to use a hexahedral grid data structure which is very simple to build and allows for quick proximity queries.

Figure 7.5 shows the result of the iterative closest points algorithm applied to the same geometry at different position and orientation (see Figure 7.5a and b). The registration result is exceptional as is shown in Figure 7.5c. Figure 7.5d depicts the cross section corresponding to the gray plane in Figure 7.5c. Errors are mainly due to numerical inaccuracies during the initial transformation of the data surface.

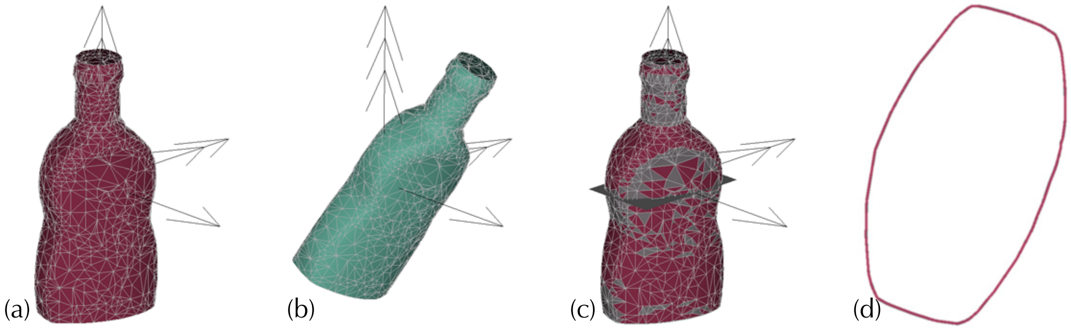


FIGURE 7.5 Surface registration of two identical geometries:
(a) First bottle (model surface)
(b) Second bottle (data surface)
(c) Result of registration
(d) Cross–section according to the gray plane in (c)

Figure 7.6 illustrates the application of the iterative closest points algorithm in the context of the surgery simulation model build–up as described in Section 7.1.2. As mentioned before, the laser range surface and the facial isosurface are clearly different (Figure 7.6a). Firstly, the laser range scanner covers a bigger area of the face. Secondly, details such as

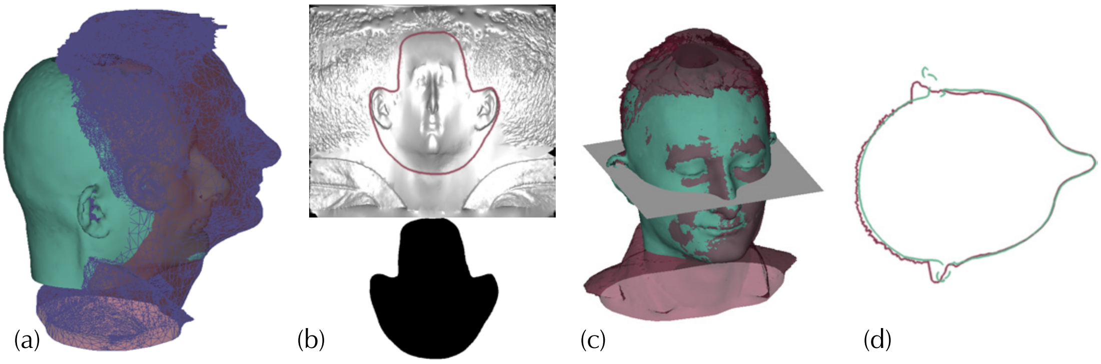


FIGURE 7.6 Registration of laser range surface to the facial isosurface:
(a) Initial alignment of laser range surface and CT isosurface
(b) Registration mask: sketching on cylindrical height field and resulting mask bitmap
(c) Result of registration
(d) Cross–section according to the gray plane in (c)

hair or eyebrows only appear in the laser scanner surface. As a consequence, prior to registration, the data surface has to be constrained to only the facial parts that do have a correspondence in the isosurface. Figure 7.6b illustrates this step and the resulting registration mask. Figure 7.6c and d show the accuracy of the registration: only at the eyes and around the ears major misalignments appear. This is due to the fact that in contrast to laser scanning, the eyes are closed during CT. Further, the ears are supported on a special pillow during CT whereas during laser scanning they are not. In summary, the region within the registration mask is registered to utmost accuracy.

7.3 DISCRETIZATION OF TISSUE INTO FINITE ELEMENTS

After the registration of the laser range surface within the CT volume data we can proceed to the discretization of the tissue. As a first step, we extract the skull surface from the CT scan. In a second step, we tile the tissue between the facial and skull surface into tetrahedra.

Much work has been done on mesh generation [146]. For tetrahedral, or more general unstructured meshing, the algorithms fit into three main categories: Octree–, Delaunay– and Advancing Front–based methods [101]. However, implementations meeting the needs of tissue meshing in the context of the surgery simulation prototype are not readily available. Both geometrically and from a data quality point of view, the setting with real patient data poses a lot of hard problems which do not arise when meshing structural parts or domains for fluid dynamic simulation. For the sake of simplicity we therefore chose to implement meshing in a rather simple way similar to [77] which is described in the following. For a survey of meshing and grid generation, including references to software packages, the reader is referred to [100].

7.3.1 Skull Extraction

Instead of extracting the skull surface by means of the marching cubes algorithm [86], the skull is extracted by projecting the facial vertices into the CT scan and recording bone isosurface crossings. Marching cubes surface extraction in the context is troublesome since the resulting surface will have holes. Further, both the isosurface crossing from tissue into bone and the crossing when leaving bone structures would be reported, thus rendering the skull surface twice. By projecting the vertices into the CT scan, we can overcome this shortcoming and additionally we can guarantee that the skull surface will have the same connectivity as the facial surface, a fact which will facilitate meshing of the tissue.

The direction of projection can be continuously adjusted between a spherical and a cylindrical projection. In the cylindrical set–up, the projection is normal to the center axis of the laser range scan. By scaling this axis continuously from its original size down to a single point, the projection can be adjusted to be more and more spherical. Best results are achieved when scaling the axis to about 80 percent, see Figure 7.7a. At 80 percent, the slightly upward direction of projection at the chin allows for better capture of the jaw bone structures. By limiting ray traversal to a user–adjustable percentage of the cylinder radius we can guarantee the skull surface not to have holes.

As mentioned in Section 7.1.1, the CT scans of patients subject to cranio–maxillofacial surgery often suffer from severe artifacts in the teeth region due to metal implants (see Figure 7.1b). We employ a simple artifact reduction scheme by using a different bone threshold around the teeth. Again, similar to the process of defining the registration mask

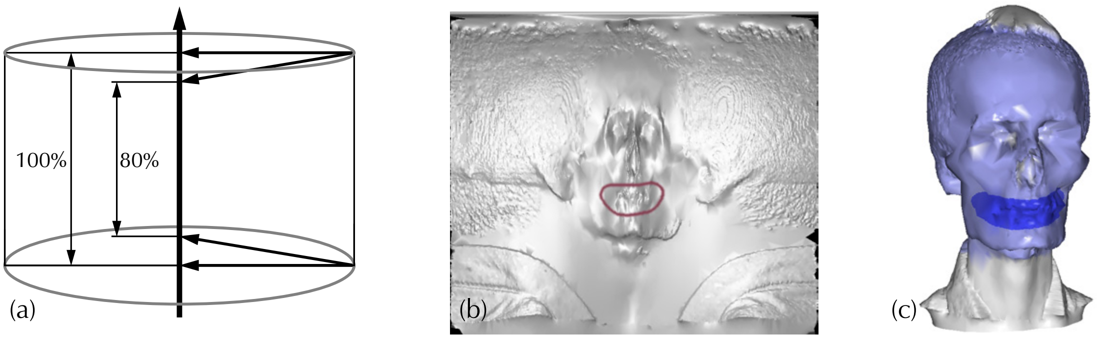


FIGURE 7.7 Skull extraction:
(a) Adjusting the direction of projection
(b) Sketching the region for artifact reduction
(b) Resulting skull surface (artifact region highlighted in darker blue)

described above, the region of influence of this different threshold is sketched overlaying the cylindrical height field representation, see Figure 7.7b.

7.3.2 Tissue Meshing

With the knowledge of both the facial and the skull surface we can span prisms between corresponding triangles since, as a consequence of the construction scheme, the two surfaces have identical connectivity (see Figure 7.8a).

In order to arrive at a tetrahedral mesh, we employ a 1:7–split of each prism into seven tetrahedra (see Figure 7.8b and c). We use this split instead of the simple 1:3–split in order to avoid cracking problems at adjacent prisms since all the four–sided prism surfaces are split the same way. With this split we have the choice of either splitting skull or facial edges.

The mesh quality of the above meshing scheme mainly depends on the quality of the mesh decimation used to simplify the laser range scan surface [123, 75]. For further tetrahedral mesh decimation or in order to increase mesh quality one could base upon the work of Staadt et al. [133, 134, 132].

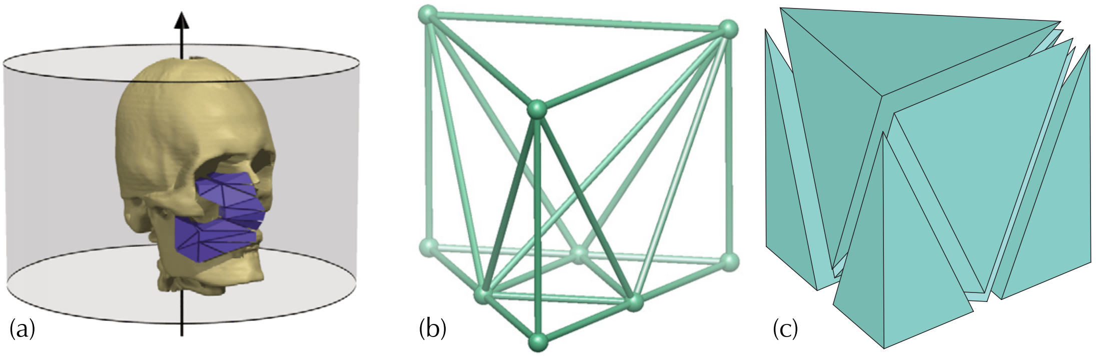


FIGURE 7.8 Tissue Meshing:
(a) Tiling the tissue between face and skull with prisms
(b) Wireframe representation of the 1:7 split of a prism
(c) Tetrahedra resulting from 1:7 split

The last issue of tissue meshing is the element–wise definition of the material properties of isotropic, linearly elastic materials, i.e. the Young modulus E and the Poisson ratio ν (see Section 3.4.1, Equation 3.41). For facial tissue, these parameters are extraordinarily difficult to determine for several reasons. First, determination in vivo is not practicable. Further, the properties of tissue change over time and as a function of patient age and pathology. Last but not least, living tissue has been shown to be neither homogeneous nor isotropic and to exhibit very complex viscoelastic behavior [18, 48]. Therefore, the model of linear elasticity must be considered an approximation.

However, Monserrat et al. showed in rheological experiments on a pig liver that tissue does exhibit a quasi linear behavior for small displacements [94]. Parameter values of $E = 150\text{MPa}$ and $\nu = 0.4$ have been found to model the elastic properties of the liver tissue well. Mechanical properties for various special kinds of tissues, such as arteries and lung tissue, can be found in the text books of Fung [48] and Duck [36]. In [48], Collagen, a basic structural element for soft and hard living tissues, has been assigned a Young modulus of $E = 10^3\text{MPa}$, whereas Elastin, the most linearly elastic biosolid material, exhibits a modulus of $E = 0.6\text{MPa}$.

In a prescribed displacement setting, the like of which is used in order to represent bone movements in the surgery simulation prototype, the order of magnitude of Young's modulus is irrelevant. This can be seen readily when recalling that prescribed displacements are accounted for by a change of the load vector. This change is proportional to the displacement and the corresponding column in the stiffness matrix, the entries of which are proportional to Young's modulus. As a consequence, it is only the inter–element variation of Young's modulus that influences the result.

In the current implementation, tissue has been assigned a Poisson ratio of $\nu = 0.4$ and either a constant elasticity modulus of about 140MPa or a modulus varying between 100 and 300MPa as a function of CT intensity at the element corner nodes. However, the results proved to be rather insensitive against variation of elastic properties.

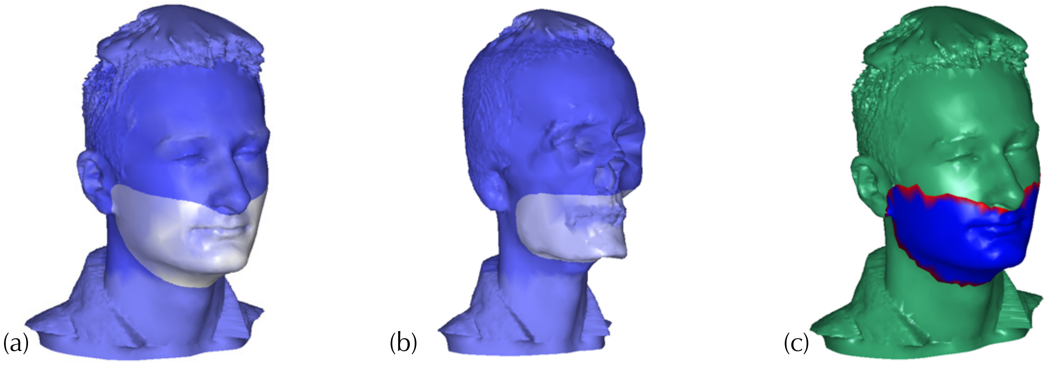


FIGURE 7.9 Imposition of boundary conditions:
(a) Boundary conditions on the face
(b) Boundary conditions on the skull
(c) Resulting model

7.3.3 Imposition of Boundary Conditions

The last step in the set–up of the model is the definition of zero–displacement boundary conditions both on the skull and on the face. Using the same sketching approach on the cylindrical height field representation described above, we mask out regions which are supposed not to have been altered during surgery. Figure 7.9a and b illustrate the definition of the boundary conditions on the face and on the skull, respectively.

Figure 7.9c shows the resulting model. Rigid parts are depicted in green whereas the lower facial region is depicted in blue with red zero–displacement boundary conditions at the boundary of the region.

7.4 COMPUTATION OF DISPLACEMENT FIELDS

On the prerequisite of having both pre– and post–surgical CT scans, we aim at an automatic determination of the displacement fields which represent the bone movements corresponding to a surgical procedure. In order for the displacement fields to be reliable, the two CT scans must be registered in a common coordinate frame. Although the variance of absolute orientation of a patient between two CT scans is rather small, there is still considerable room for improvements. After successful registration, the pre–surgical jaw bone surface will be cut in correspondence with surgery and the determination of the displacement field is achieved by means of the surface registration technique described in Section 7.2.

7.4.1 Registration of Volume Data Sets

The registration of volume data sets can be formulated as a nonlinear least–square optimization procedure. We follow the approach of Thévenaz, Unser et al. [145, 144, 149]. They describe an automatic subpixel registration algorithm that minimizes the least–square intensity difference between a reference and a test image. The image can either be two– or three–dimensional. It uses an explicit spline representation of the images and in addition is based on a multiresolution pyramid which allows for a coarse–to–fine registration procedure. The geometric deformation model is a global three–dimensional affine transformation that can be restricted to a homomorphic, rigid–body transformation (rotation and translation) in combination with isometric scaling. The minimization is performed by means of a modified Levenberg–Marquardt algorithm [145] which will be outlined in the following.

Problem statement. The objective of the registration is the minimization of the intensity differences between the test image $f_T(\mathbf{x})$ and the reference image $f_R(\mathbf{x})$ by means of a transformation of the test image. Thus the objective function to be minimized may be stated as

$$\varepsilon^2 = \int\limits_{\{\mathbf{x}\} \subset \mathbb{R}^q} (f_R(\mathbf{x}) - Q_{\mathbf{p}}\{f_T(\mathbf{x})\})^2 d\mathbf{x} = \|f_R(\mathbf{x}) - Q_{\mathbf{p}}\{f_T(\mathbf{x})\}\|^2 \tag{7.18}$$

with $Q_{\mathbf{p}}$ being the transformation which transforms the test into the reference image, $\mathbf{p}$ the parameter vector which describes the transformation, and q the dimension of the images. Now, the problem reduces to finding the parameter vector $\mathbf{p}$ which minimizes the intensity difference, thus to solve $\partial\varepsilon^2(\mathbf{p})/\partial\mathbf{p} = 0$.

Although the spline representation of the images can be regarded as being continuous, it is useful to approximate the objective function in Equation 7.18 by the finite sum over the number of pixels or voxels n

$$\varepsilon^2 \cong \chi^2(\mathbf{p}) = \frac{1}{n}\sum_{i=1}^{n}(f_R(\mathbf{x}_i) - Q_{\mathbf{p}}\{f_T(\mathbf{x}_i)\})^2 . \tag{7.19}$$

Due to the nonlinear dependencies imposed by the transformation, the minimization must proceed iteratively. Given a trial value for the transformation parameter vector $\mathbf{p}$, the objective is to find improvements to the actual vector until convergence is reached:

$$\mathbf{p}_{t+1} = \mathbf{p}_t + \delta\mathbf{p}_t$$

Among many others there are two methods which can be applied in order to find the correction vector $\delta\mathbf{p}_t$: the purely gradient based steepest descent and the (Gauss–)Newton method. A combination of both methods, the Levenberg–Marquardt method [89], has proven very powerful and has become the standard of nonlinear least–squares routines.

Steepest Descent. The *steepest descent* method defines the correction vector in the negative direction of the gradient of the objective function

$$\mathbf{p}_{t+1} = \mathbf{p}_t - h_t\nabla\chi^2(\mathbf{p}_t) \tag{7.20}$$

where the Nabla sign $\nabla\chi^2$ stands for the gradient of the objective function and h_t stands for the step size taken in the negative direction of the gradient.

Gauss–Newton. The *Newton* method is based on the assumption that sufficiently close to the minimum the objective function $\chi^2(\mathbf{p})$ may be approximated by a quadratic form

$$\chi^2(\mathbf{p}) \approx c - \mathbf{dp} + \frac{1}{2}(\mathbf{p}^T\mathbf{D}\mathbf{p}) . \tag{7.21}$$

In fact, the quadratic form stems from the Taylor expansion of χ^2 in parameter space around the actual parameter vector $\mathbf{p}_t$

$$\chi^2(\mathbf{p}_t + \mathbf{p}) = \chi^2(\mathbf{p}_t) + \mathbf{p}^T\nabla\chi^2(\mathbf{p}_t) + \frac{1}{2}\mathbf{p}^T\mathbf{H}(\mathbf{p}_t)\mathbf{p} + O(\|\mathbf{p}\|^3) .$$

Taking the point $\mathbf{p}_t$ as the origin, we see from this equation that the matrix $\mathbf{D}$ in Equation 7.21 in fact is the Hessian matrix of χ^2 at $\mathbf{p}_t$ and $\mathbf{d}$ corresponds to the negative gradient $\nabla\chi^2$ whereas c stands for $\chi^2(0)$.

At any point $\mathbf{p}$, the gradient of Equation 7.21 easily evaluates to

$$\nabla\chi^2(\mathbf{p}) = \mathbf{Dp} - \mathbf{d} .$$

At the minimum point $\mathbf{p}_{min}$, the gradient will vanish which gives the minimum condition

$$\mathbf{D}\mathbf{p}_{min} = \mathbf{d} \tag{7.22}$$

whereas at any other point $\mathbf{p}_t$ we have

$$\mathbf{D}\mathbf{p}_t = \nabla\chi^2(\mathbf{p}_t) + \mathbf{d}. \tag{7.23}$$

Subtracting Equations 7.22 and 7.23 and multiplying by the inverse matrix $\mathbf{D}^{-1}$ we find the correction step needed to come from $\mathbf{p}_t$ to the minimum $\mathbf{p}_{min}$ as

$$\mathbf{p}_{min} - \mathbf{p}_t = \mathbf{D}^{-1}[-\nabla\chi^2(\mathbf{p}_t)]. \tag{7.24}$$

From that, and with $\mathbf{D} \equiv \mathbf{H}_t$ we find the Newton correction step similar to its steepest descent counterpart in Equation 7.20 as

$$\mathbf{p}_{t+1} = \mathbf{p}_t + \mathbf{H}_t^{-1}[-\nabla\chi^2(\mathbf{p}_t)]. \tag{7.25}$$

In order to compute the correction vector $\delta\mathbf{p}_t = \mathbf{p}_{t+1} - \mathbf{p}_t = \mathbf{H}_t^{-1}[-\nabla\chi^2(\mathbf{p}_t)]$ we need both the Hessian and the gradient of χ^2 at $\mathbf{p}_t$, whereas for the steepest descent it is proportional to only the gradient. The Newton correction vector $\delta\mathbf{p}_t$ results from solving the system of equations which directly follows from Equation 7.25 to

$$\mathbf{H}_t\delta\mathbf{p}_t = -\nabla\chi^2(\mathbf{p}_t).$$

It is illustrative to explicitly state this system of equation with respect to the discrete optimization criterion in Equation 7.19

$$\sum_{l=1}^{m} \alpha_{kl}\delta p_l = \beta_k \tag{7.26}$$

with m being the number of transformation parameters for the affine (9 affine degrees of freedom + 3 translational degrees of freedom + 1 gray–level scaling factor) or homomorphic (3 Euler angles + 3 translational degrees of freedom + 1 isometric scaling factor + 1 gray–level scaling factor) case.

The m vector $[\beta_k]$ follows from the gradient to

$$\beta_k = -\frac{1}{2}\frac{\partial}{\partial p_k}\chi^2(\mathbf{p}) = \frac{1}{n}\sum_{i=1}^{n}(f_R(\mathbf{x}_i) - Q_{\mathbf{p}}\{f_T(\mathbf{x}_i)\})\frac{\partial}{\partial p_k}Q_{\mathbf{p}}\{f_T(\mathbf{x}_i)\}. \tag{7.27}$$

The $m \times m$ matrix $[\alpha_{kl}]$ follows from the Hessian to

$$\begin{aligned}\frac{1}{2}\frac{\partial^2}{\partial p_k \partial p_l}\chi^2(\mathbf{p}) = \frac{1}{n}\sum_{i=1}^{n}\Bigg(&\frac{\partial}{\partial p_k}Q_{\mathbf{p}}\{f_T(\mathbf{x}_i)\}\frac{\partial}{\partial p_l}Q_{\mathbf{p}}\{f_T(\mathbf{x}_i)\} \\ &-(f_R(\mathbf{x}_i) - Q_{\mathbf{p}}\{f_T(\mathbf{x}_i)\})\frac{\partial^2}{\partial p_k \partial p_l}Q_{\mathbf{p}}\{f_T(\mathbf{x}_i)\}\Bigg)\end{aligned} \tag{7.28}$$

For reasons given in [112], the second derivative terms in Equation 7.28 are usually ignored which actually transforms the Newton method into a scheme known as *Gauss–*

Newton and yields the matrix $[\alpha_{kl}]$ as

$$\alpha_{kl} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\partial}{\partial p_k}Q_{\mathbf{p}}\{f_T(\mathbf{x}_i)\}\frac{\partial}{\partial p_l}Q_{\mathbf{p}}\{f_T(\mathbf{x}_i)\}\right). \tag{7.29}$$

Omitting the second derivative terms slows down the quadratic convergence of the Newton to only linear convergence of the Gauss–Newton method. Further, although the iteration scheme still guarantees a downhill search direction, it often overshoots unless good starting values are available and thus the quadratic approximation of $\chi^2(\mathbf{p})$ is of good quality. However, the evaluation of the Gauss–Newton iterative scheme is far less computationally expensive and the combination of steepest descent and Gauss–Newton according to Marquardt overcomes most of the aforementioned disadvantages.

Levenberg–Marquardt. In order to understand the Levenberg–Marquardt algorithm [89] it is illustrative to rewrite the steepest descent Equation 7.20 similar to Equation 7.26 using the gradient according to Equation 7.27 as the correction vector $\delta\mathbf{p}_t = \mathbf{p}_{t+1} - \mathbf{p}_t$

$$\delta p_k = h_k\beta_k \qquad \text{or} \qquad \frac{1}{h_k}\delta p_k = \beta_k. \tag{7.30}$$

Marquardt's first insight was about the scaling factor h_k: whereas the quantity of χ^2 is non–dimensional, the β_k have the dimension of $1/p_k$. From Equation 7.30, we see that the scaling factor h_k therefore must be of dimension p_k^2. Marquardt found that the Hessian could give some information about the order of magnitude of the scale. The only components of $[\alpha_{kl}]$ which have the right dimension are $1/\alpha_{kk}$, so the diagonal elements are eligible to set the scale of the constant. In addition to fitting the dimension, $1/\alpha_{kk}$ is guaranteed to be positive by the definition of Equation 7.29. In order to cut down the step size and thus reduce the danger of overshooting, the scale is divided by a non–dimensional factor λ. Thus we from Equation 7.30 arrive at

$$\delta p_k = \frac{1}{\lambda\alpha_{kk}}\beta_k \qquad \text{or} \qquad \lambda\alpha_{kk}\delta p_k = \beta_k. \tag{7.31}$$

In addition to the scaling, Marquardt adopted Levenberg's suggestion of adding some scalar value to the diagonal elements α_{kk} in order to shrink the step size [85]. But unlike Levenberg who suggested adding the same constant to all diagonal elements, Marquardt took into account their vastly different magnitudes and thus found a way to combine the steepest descent and Gauss–Newton parameter vector update. He suggested the following alterations to the matrix $[\alpha_{kl}]$

$$\begin{cases} \alpha'_{kl} = \alpha_{kl}(1+\lambda) & k = l \\ \alpha'_{kl} = \alpha_{kl} & k \neq l \end{cases}$$

thus rewriting Equation 7.26 to

$$\sum_{l=1}^{m}\alpha'_{kl}\delta p_l = \beta_k. \tag{7.32}$$

The free parameter λ now determines the degree to which the update $\delta\mathbf{p}_t$ conforms to a Gauss–Newton or steepest descent step.

The characteristics of the Levenberg–Marquardt algorithm is to adapt λ in each iteration in such a way that the more successful the previous updates $\delta\mathbf{p}_t$ have been, the more Gauss–Newton like the next update will be $(\lambda \approx 0)$. Conversely, the less successful, the more gradient like the next update will be $(\lambda » 0)$. This is in concordance with the observation that for a good quadratic local approximation of $\chi^2(\mathbf{p})$ we know how to jump to the minimum by means of the Gauss–Newton Equation 7.24, but for a poor approximation the best update is a step down the gradient, as in the steepest descent method. From a matrix algebra point of view we see that the bigger λ becomes the more diagonally dominant the matrix will be and thus the closer Equation 7.32 will be to Equation 7.31.

Modified Levenberg–Marquardt. A major disadvantage of the original Levenberg–Marquardt procedure is the computational effort which in each iteration is involved in computing both the gradient vector $[\beta_k]$ and the Hessian $[\alpha_{kl}]$. In order to bypass these steps, the following change of strategy is suggested in [145]: instead of trying to find parameters $\mathbf{p}_1$ such that

$$\left\| f_R - Q_{\mathbf{p}_1}\{ f_T\}\right\|^2 < \left\| f_R - Q_{\mathbf{p}_0}\{ f_T\}\right\|^2,$$

it is favorable to compute $\mathbf{p}_2$ such that

$$\left\|Q^{-1}_{\mathbf{p}_0}\{ f_R\} - Q_{\mathbf{p}_2}\{ f_T\}\right\|^2 < \left\|Q^{-1}_{\mathbf{p}_0}\{ f_R\} - f_T\right\|^2.$$

Using this strategy, Thévenaz et al. update the inverse transformation $Q^{-1}_{\mathbf{p}_0}$ that is applied to f_R instead of the direct transformation $Q_{\mathbf{p}_2}$ that is applied to f_T. As a consequence, the gradient and Hessian of the criterion with respect to $\mathbf{p}_2$ are now independent of the initial guess $\mathbf{p}_0$ and are computed at a fixed point in parameter space, namely at $\mathbf{p}_2 = 0$. In other terms, the original Levenberg–Marquardt algorithm iteratively solves $(\partial\varepsilon^2(\mathbf{q})/\partial\mathbf{q})|_{\mathbf{q} = \mathbf{p}} = 0$, while the modified version solves $(\partial\varepsilon^2(\mathbf{p} + \mathbf{q})/\partial\mathbf{q})|_{\mathbf{q} = 0} = 0$. The former case involves a Taylor expansion of ε^2 around a different point at each iteration, whereas in the latter case this point is fixed.

Interpolation model and multiresolution pyramid. The advantages of combining the above optimization scheme with a multiresolution pyramid (see Figure 7.10a) complemented with spline interpolation at each level are manifold [145].

Firstly, starting the registration at a coarse level bears the following advantage: it is at this level that most of the iterations will take place since initially we do not have a fair transformation estimate. However, these iterations are computationally cheap since due to the decimation inherent to the pyramid, the number of pixels at the coarsest level is small. Secondly, the Gauss–Newton like optimization at finer levels will converge in very few steps given a good starting estimate from the previous level. So, the more pixels we have to process the better the estimate gets and the faster convergence will be reached, usually in only one step.

In addition, the smoothness imposed by the polynomial cubic splines used for the interpolation at each level tends to regularize the optimization problem. The surface $\varepsilon^2(\mathbf{p})$ becomes smoother at coarser levels since the downsampling and subsequent interpolation removes more and more detail and noise. As a consequence, the algorithm achieves first a registration with respect to only the large–scale features and then makes small corrections for progressively finer details. Such a procedure obviously minimizes the risk of getting trapped in a local minimum.

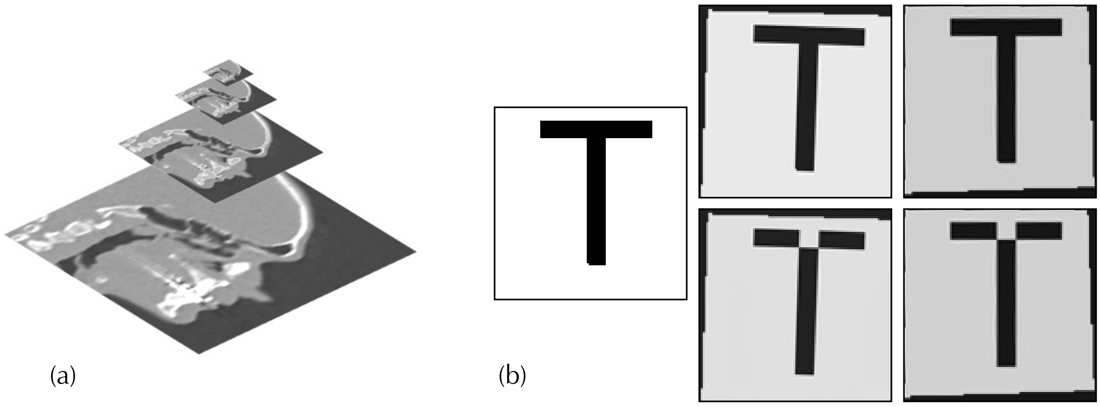


FIGURE 7.10 (a) Multiresolution pyramid used for the image registration.
(b) Registration results on a two–dimensional image data set.

Results. Figure 7.10b shows the results by means of a two–dimensional test image. In a first test, the original image T was transformed and registered again. The registration was successful with a translational accuracy of about 0.1 pixels and a rotational accuracy of 0.002 degrees. In a second test, the original image was altered and transformed identically. The registration proved to be stable enough to handle such alterations: the translational error was within the bounds of 0.15 pixels, whereas the rotational error still was below 0.01 degrees. This stability has to be attributed to the multiresolution set–up which renders the procedure robust against small deviations in the data set. Both registrations were performed without attempting to adjust the intensity differences. Doing so in such small an example would severely compromise the quality of registration since the misalignment could be cured either by transformation or by an adaptation of image intensity.

Figure 7.11 shows the results of a three–dimensional registration procedure: the postoperative CT scan was registered to the preoperative scan. In order not to have the registration suffering from deviations due to the surgical procedure, the jaw region was masked out and ignored during the registration. In Figure 7.11c, the exceptional quality of the

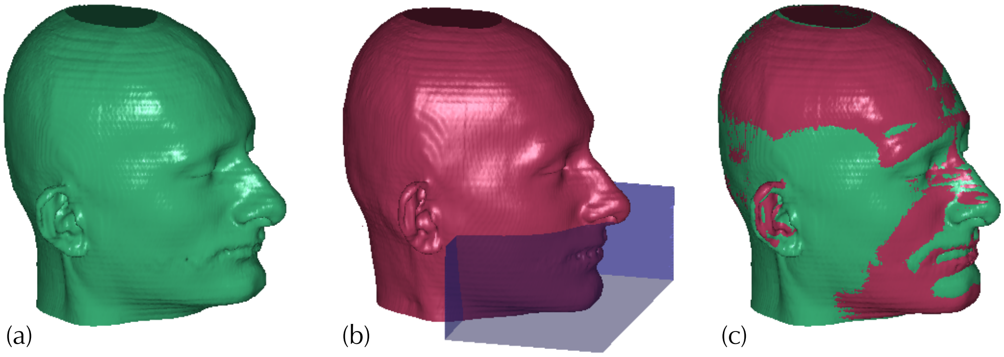


FIGURE 7.11 Registration results on a three–dimensional data set:
(a) Facial isosurface in preoperative CT scan
(b) Facial isosurface in postoperative CT scan (differing jaw regions masked out)
(c) Facial isosurface in registered postoperative CT scan overlaid to the surface in (a)

registration becomes obvious: major misalignments appear only at the jaw and around the cheeks. These regions clearly have been altered due to the cranio–maxillofacial surgery done on the patient.

Similar to the two–dimensional test of Figure 7.10b the registration in Figure 7.11 was done without taking into account intensity differences. We found that the registration process tends to trade rotational accuracy against intensity adjustment. In addition, the intensity differences in the two CT scans proved to be negligible. Therefore, it is also admissible to use the same threshold for both isosurfaces in Figure 7.11c.

7.4.2 Jaw Cutting and Movement

After the registration step, we are ready to simulate the surgical procedure. Using the cylindrical height field representation we cut the upper and lower jaw according to the surgeon (see Figure 7.12a).

Next we roughly align the cut jaw geometries within the post–surgical skull isosurface. Using the iterative closest points surface registration of Section 7.2.2, we fit the cut parts into their post–surgical positions and thus find the displacement fields as the difference of pre– and post–surgical position (see Figure 7.12a).

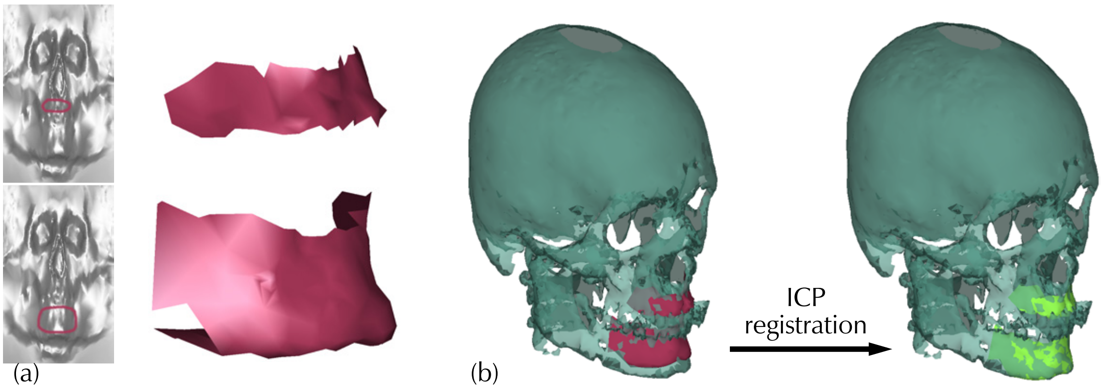


FIGURE 7.12 (a) Jaw cutting and cut geometry on upper and lower jaw.
(b) Jaw alignment within the post–surgical CT scan.

7.5 RESULTS AND EVALUATION

After a quick look at the evaluation procedure, we in this section will present results achieved with the prototype surgery simulator. We compare the simulation results with respect to the degree of interpolation, grid resolution and the kind of elements used.

7.5.1 Evaluation Procedure

In order to evaluate the simulation, a second laser range scan of the patient is taken after surgery. It is important to take the laser scan several months after surgery since swelling would otherwise degrade correspondence and thus affect the evaluation negatively.

Further we of course have to align the post–surgical scan with the simulation data set, i.e. the model used for the simulation. This is again achieved using the iterative closest points surface registration algorithm. However, we have to take into account that major parts of

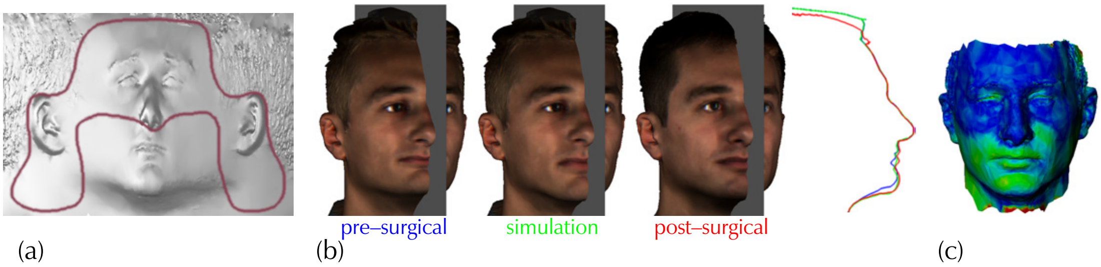


FIGURE 7.13 Evaluation procedure:
(a) Model to simulation registration region of interest
(b) Profile lines representing cross sections according to the gray plane
(c) Frontal error map: distance between post–surgical laser range scan and simulation

the face around the jaws have been altered in course of surgery. Therefore, we define a registration region of interest which spares not only hair but also the jaw region (see Figure 7.13a).

We will compare the pre–surgical situation to the simulated outcome, and, more important, simulation and real outcome. In addition to frontal and profile views, we give both profile lines and frontal error maps. The profile lines represent cross sections with respect to the gray plane in Figure 7.13b. The frontal error map depicts the distance between simulated and post–surgical surface (Figure 7.13c). The distance is calculated according to the one–sided distance in Equation 5.20 [102].

7.5.2 The Example Patient

The example patient is depicted in Figure 7.14. He suffers from a so–called *short face* with a deep bite caused by the retropositioned mandible, i.e. the lower jaw, as well as a reduced vertical facial height due to a maxilla (upper jaw) positioned too high. If an orthodontic treatment, i.e. braces, of such malformations proves inefficient, one resolves to orthognatic surgery. The upper and lower jaws are detached from the skull in order for correct realignment. To this aim, the upper jaw is cut according to a so–called *Lefort–1 osteotomy* whereas the lower jaw is detached by means of a *sagittal split* which preserves the neurovascular bundle inside the mandible (see Figure 7.15).

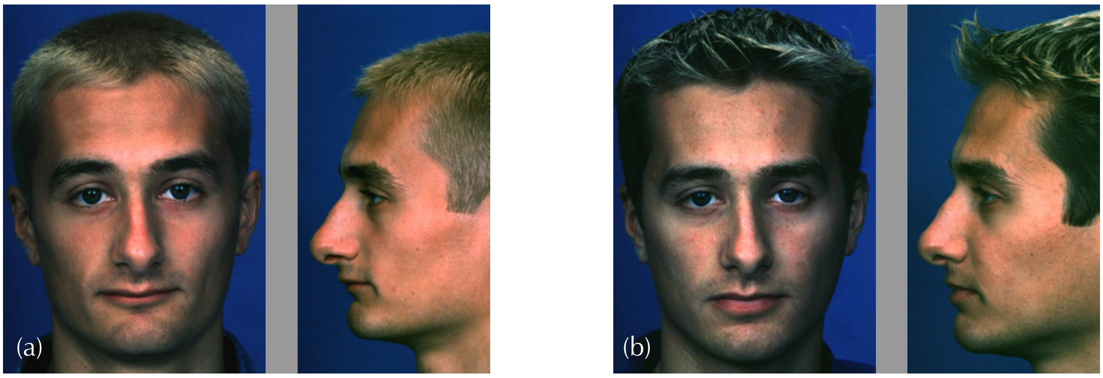


FIGURE 7.14 Example patient: (a) pre–surgical situation, (b) post–surgical situation.

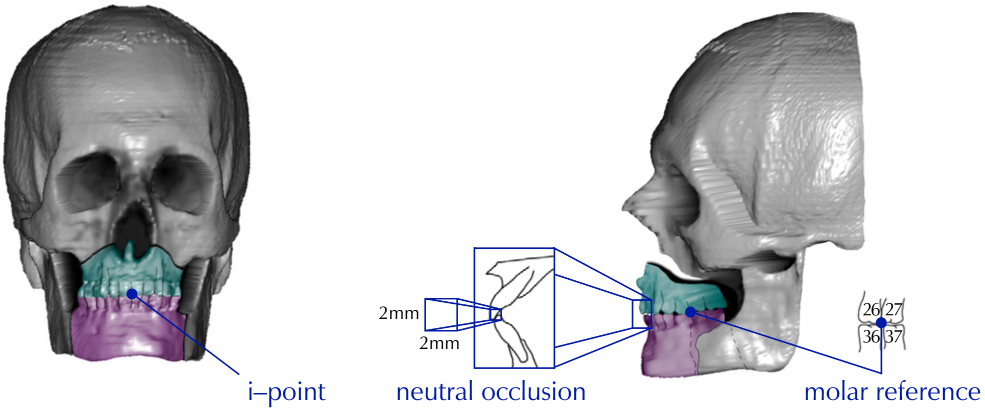


FIGURE 7.15 Lefort I osteotomy (maxilla) and sagittal split (mandible) with reference points (teeth numbering according to the World Dental Federation) [80, 77].

The repositioning of the upper jaw is done with respect to two reference points: the *i–point*, between the upper incisors, and either the tips of the upper canines or the *molar reference point* situated at the molars 17 and 27 on the right and left, respectively (see Figure 7.15). The lower jaw is then positioned with respect to the upper jaw in order to reach the *neutral occlusion* at the i–point as indicated in Figure 7.15. The main objective of repositioning of course lies in the correction of mandibular disharmonies and in the reconstruction of the masticatory system. However, aesthetic aspects are taken into account, too.

Figure 7.16a and b illustrate the traditional surgery planning procedure. Lateral X–ray images and a profile sketch are the main ingredients to the procedure. In addition, a plaster model of the jaw helps in finding the correct alignment of upper and lower jaw (Figure 7.16c). Figure 7.16d illustrates the troublesome presence of brackets and dental braces which render CT images prone to artifacts.

Figure 7.17a visualizes the displacements due to surgery. The pseudo coloring results from comparing the pre– and post–surgical laser range scanned situation. The pseudo coloring reveals surface displacements between 0 to 7 millimeters with maxima appearing at the lower lip and at the chin. Figure 7.17b illustrates the displacement field which was computed using the procedure described in Section 7.4.

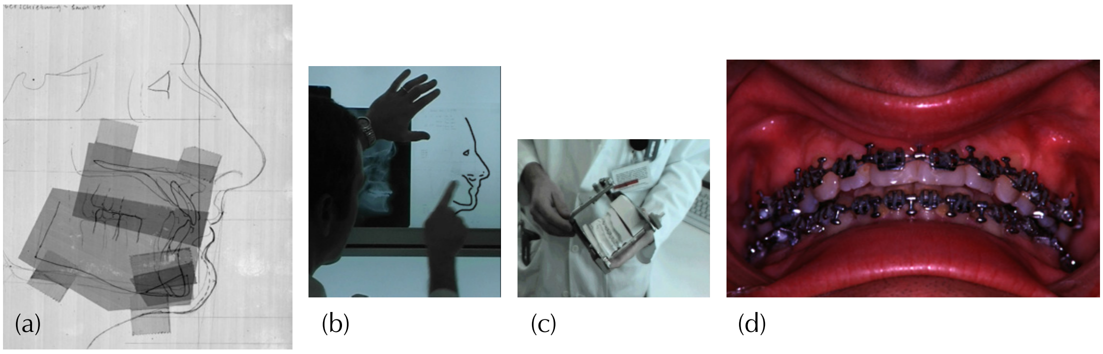


FIGURE 7.16 Conventional surgery planning: (a) profile sketch, (b) lateral X–ray image, (c) plaster model, (d) dental braces and brackets responsible for CT artifacts.

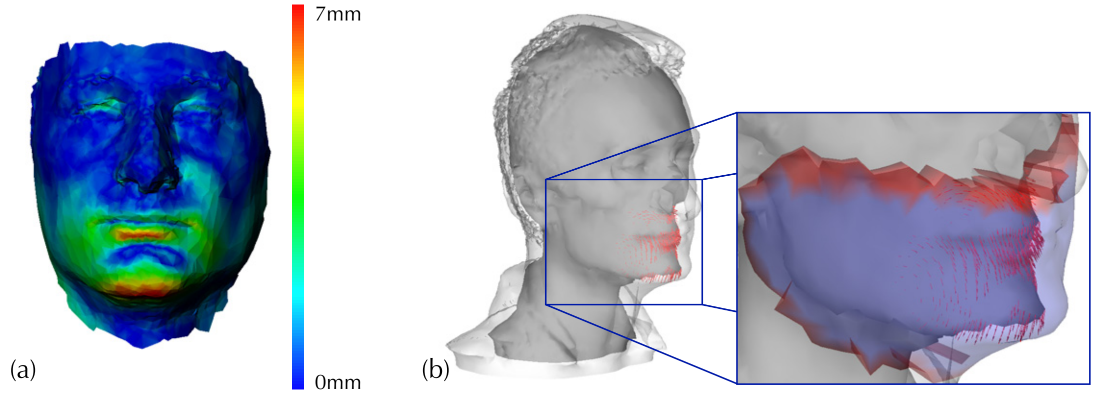


FIGURE 7.17 (a) Surface changes due to real world surgery.
(b) Displacement fields computed according to Section 7.4.

7.5.3 Simulation Results

In this section, the four finite elements of Chapter 5 are compared on two meshes of different resolution. The two mesh resolutions, the low resolution model, and an example displacement visualization are given in Figure 7.18. The low resolution of only 6769 elements is chosen such that the linear, quadratic and cubic C^0 as well as the C^1 simulations converged in reasonable time (Figures 7.19 and 7.20, Table 7.1). Linear and quadratic C^0 simulations were done on the high resolution model comprising 14888 elements (Figures 7.21 and 7.22, Table 7.2).

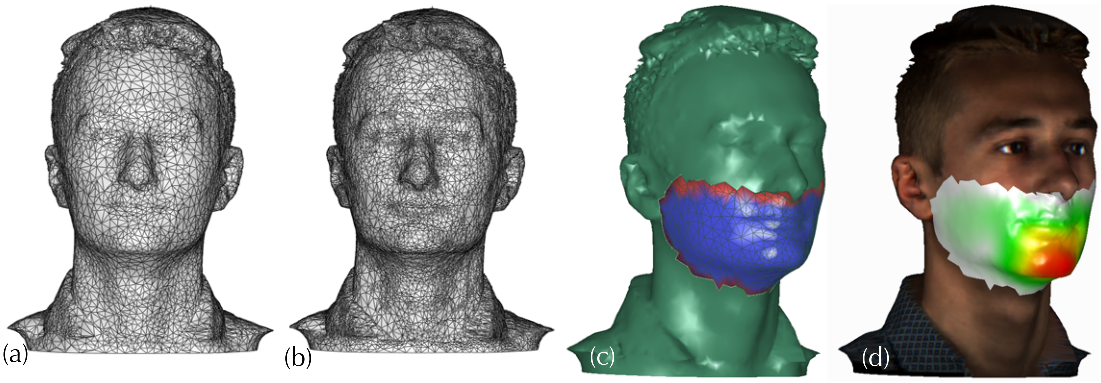


FIGURE 7.18 (a) Low resolution mesh.
(b) High resolution mesh.
(c) Low resolution model comprising 6769 tetrahedral elements.
(d) Surface displacement visualization resulting from a quadratic C^0 simulation.

The Poisson ratio was chosen to be $\nu = 0.4$ whereas Young's modulus varies between 100 and 300MPa as a function of CT intensity. The pure displacement based simulation performed surprisingly well and produced better results in less time then the mixed formulation. Therefore, the following results were produced using the pure displacement based formulation. A diagonal preconditioner preceded the conjugate gradient solver which was run to a residual accuracy of 10^{-5}.

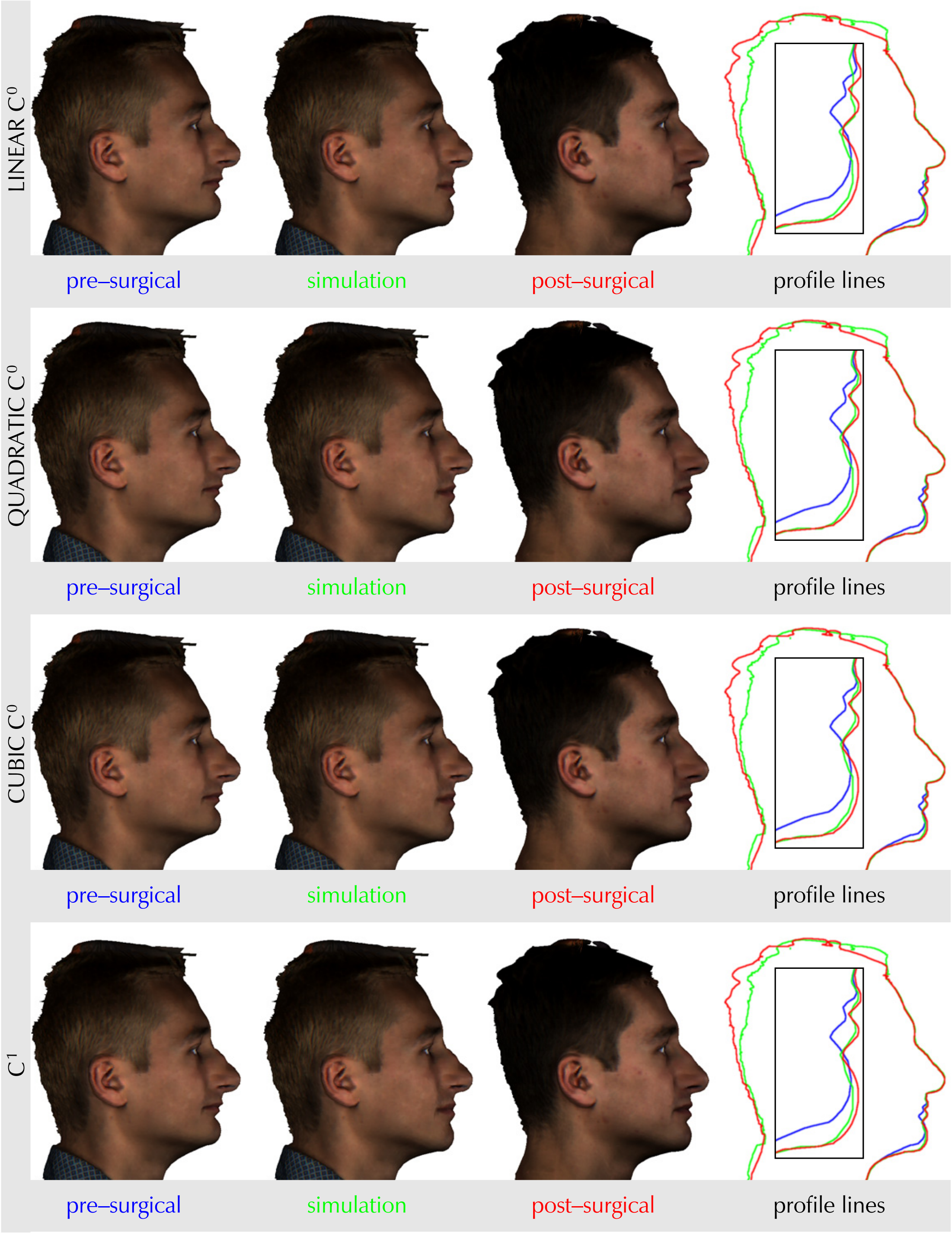


FIGURE 7.19 Profile view comparison of low resolution simulation results: C^0 linear, quadratic and cubic element opposed to the C^1 element. **[see Color Plate 7 on page 206]**

The first thing to notice in Figure 7.19 is the near equivalence of the simulation results: only the linear C^0 simulation reveals minor artifacts mainly at the upper lip which gives the simulation result a stiff and tense expression. The corresponding profile lines, especially in the magnified view, clearly illustrate the deviation. However, the profile views of the higher order C^0 as well as the result of the C^1 simulation show a near to perfect match of simulation and real outcome and must be considered largely equivalent.

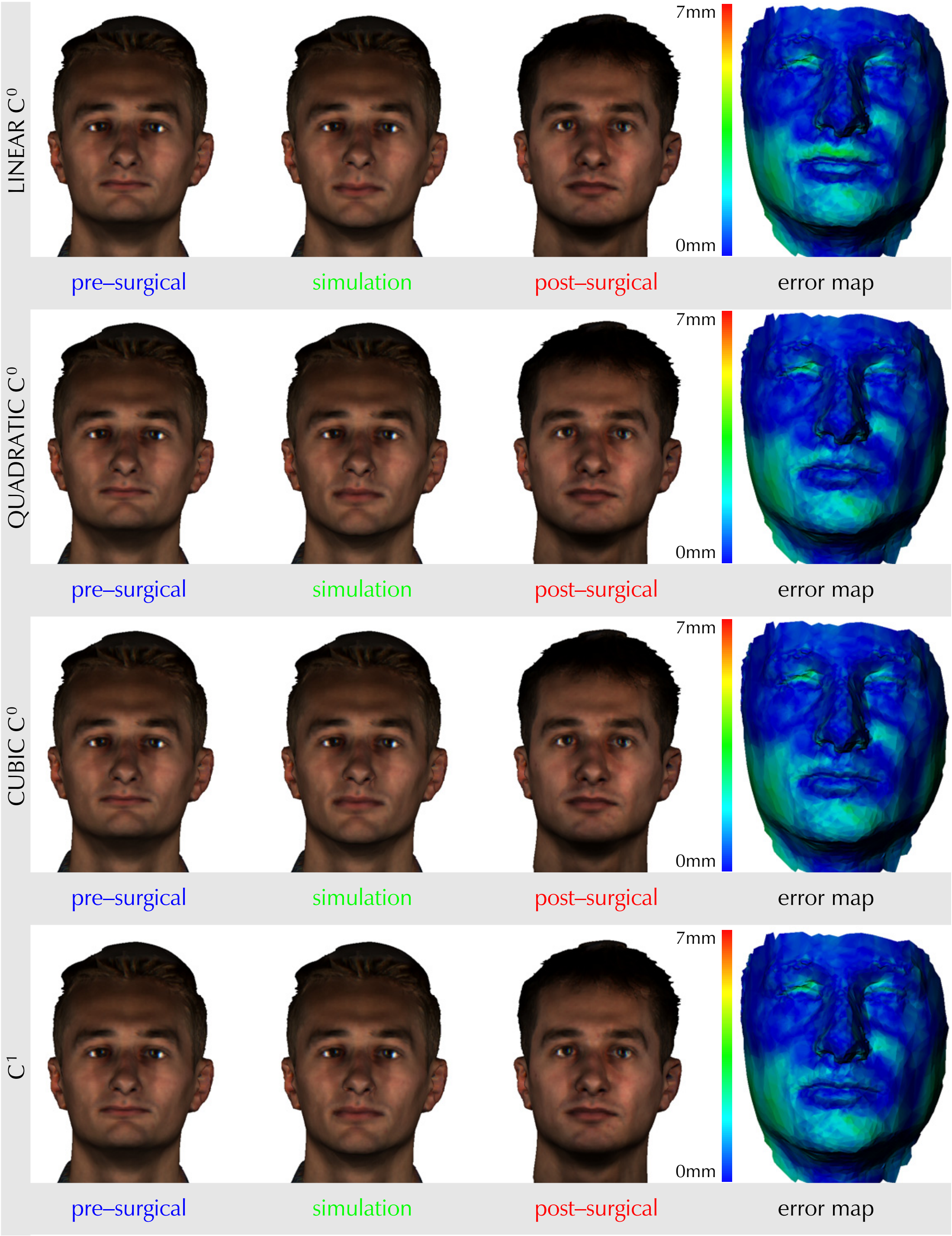
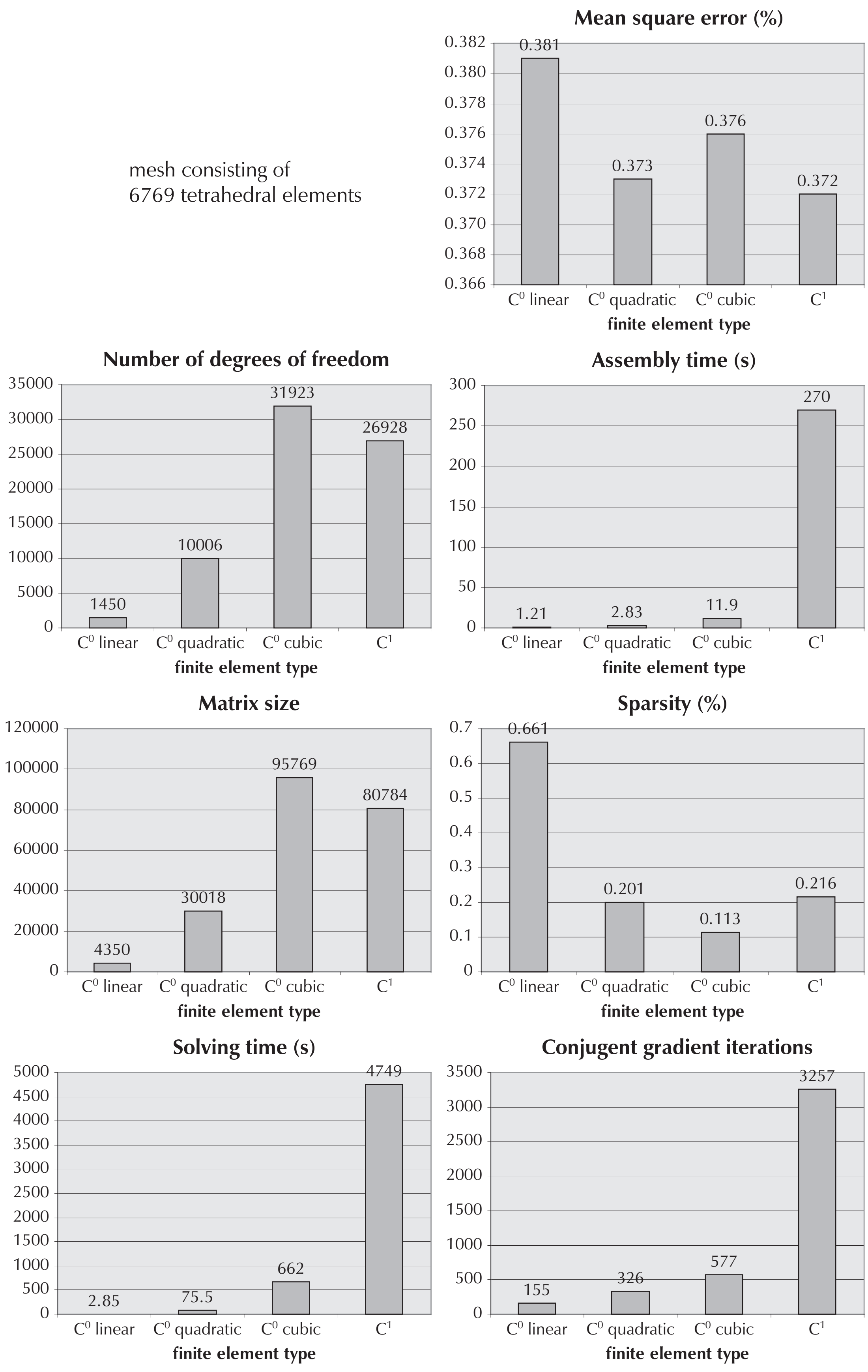


FIGURE 7.20 Frontal view comparison of low resolution simulation results: C^0 linear, quadratic and cubic element opposed to the C^1 element. **[see Color Plate 8 on page 207]**

The frontal view of the same simulation test series confirms what has been stated before: only the linear C^0 simulation shows minor deficits. Again, the corresponding facial expression is slightly tense and the error map reveals artifacts in the region of the upper lip (see top row of Figure 7.20). However, the errors in all the simulations are in the range of 0 to 3 millimeters and, what is most important, the visual appearance of the simulation results reflects the real post–surgical appearance to a great extent.

TABLE 7.1 Timings and statistics for the test series in Figures 7.19 and 7.20.



Similar to the results in [80], swelling artifacts appear in the left upper jaw region. In addition to swelling, the slightly asymmetric error distribution must be attributed to muscular activities and misalignments during scanning which could be compensated only partly in the registration of the data sets. As a consequence, both the swelling artifacts and comparable asymmetries are also to be seen in the difference visualization between pre– and post–surgical appearance in Figure 7.17a.

The mean square error statistics in Table 7.1 confirm the visual expression of near equivalence: the higher order C^0 and the C^1 simulation differ only slightly.

In contrast to the visual and geometric similarity, the statistics in Table 7.1 reveal major differences with respect to computational costs. As expected the linear C^0 simulation is by far the cheapest, whereas the computational costs of the C^1 simulation clearly exceed the C^0 simulations. In accordance with the observations made in Section 5.7, the C^1 construction not only results in less sparse but also in worse conditioned systems. This is reflected in both longer overall solving times and a noticeable increase in the number of conjugate gradient iterations needed to reach convergence. The bottom row of Table 7.1 makes evident that the solving costs of a C^1 simulation exceed even the cubic C^0 simulation costs by almost an order of magnitude. For the linear and quadratic interpolation the ratio gets even more noticeable. Moreover, the C^1 assembly procedure exceeds its C^0 counterpart by 1 to 2 orders of magnitude.

As far as matrix size, number of degrees of freedom and sparsity of C^0 simulations are concerned, similar remarks to the ones in Section 5.7.1 are to be made: increasing matrix size goes along with a more and more sparsely populated structure. Therefore, the increase in computational costs for solving is rather modest for the higher order C^0 simulations.

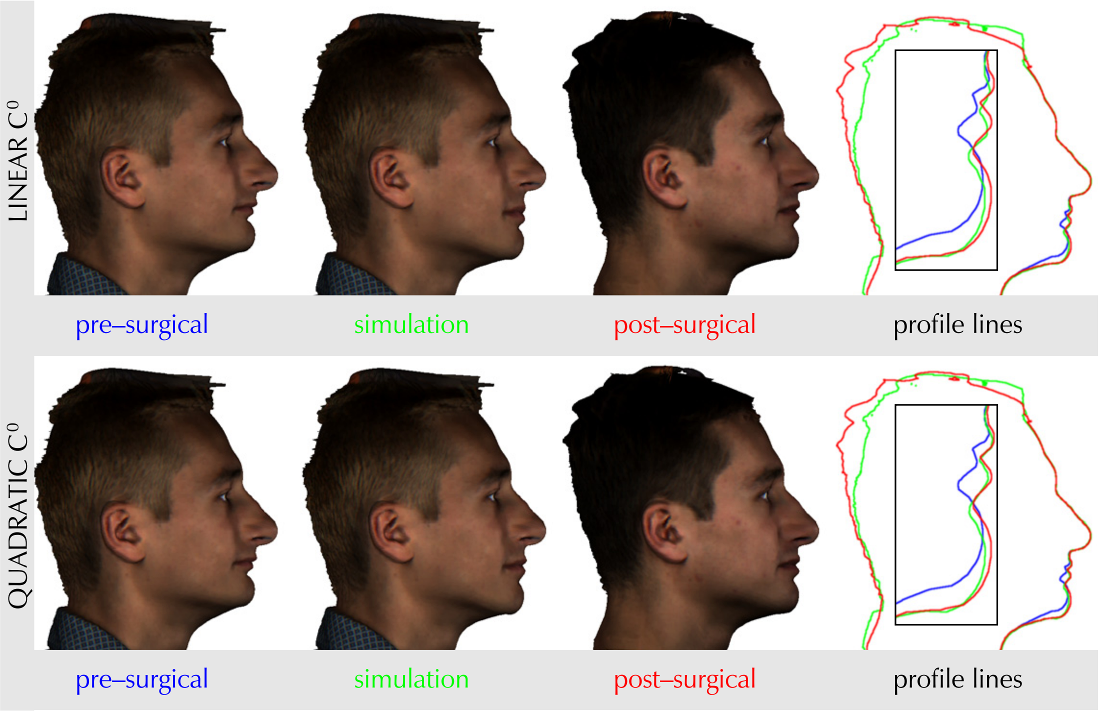


FIGURE 7.21 Profile view comparison of high resolution simulation results: C^0 linear and quadratic element. **[see Color Plate 9 on page 208]**

The above remarks lead to the following conclusion: although the C^1 simulation performs best with respect to the mean square error statistics and at least as visually convincing as the best C^0 simulation, it is questionable whether the increase in computational costs is justifiable. Given the very good performance at much lower cost of even the quadratic C^0 simulation it seems to be far more promising to increase the mesh resolution than to invest into higher degree of interpolation or even higher order of continuity. In addition to the obvious advantages of this approach, an increased mesh resolution allows for a more detailed capture of the face and skull geometry in the model build–up and consequently for a more exact computation and representation of the surgery displacement fields.

In addition, the C^1 simulation only bears advantages with respect to surface quality if the initial surface is C^1 continuous. This is the case in the examples in Section 5.7. In the surgery simulation example, the model build–up as described in Section 7.1.2 does not provide a smooth surface. An additional fairing step, preceding the simulation, would be needed to achieve a C^1 surface as in [80].

As a consequence of the above findings, the simulation of the same surgical procedure was done on a finer mesh comprising 14888 tetrahedra using the linear and quadratic C^0 elements only. The results are given in Figures 7.21 and 7.22 for the profile and frontal view, respectively. In Table 7.2, the simulation statistics are opposed to the results of the low resolution C^1 simulation. The results look visually similar but the statistics reveal a clear improvement with respect to the mean square error. At the same time, the computational costs both with respect to solving and assembly are still clearly below the C^1 reference used in the comparison. The simulation using the linear element still results in a slightly tense expression which renders the quadratic element the element of choice.

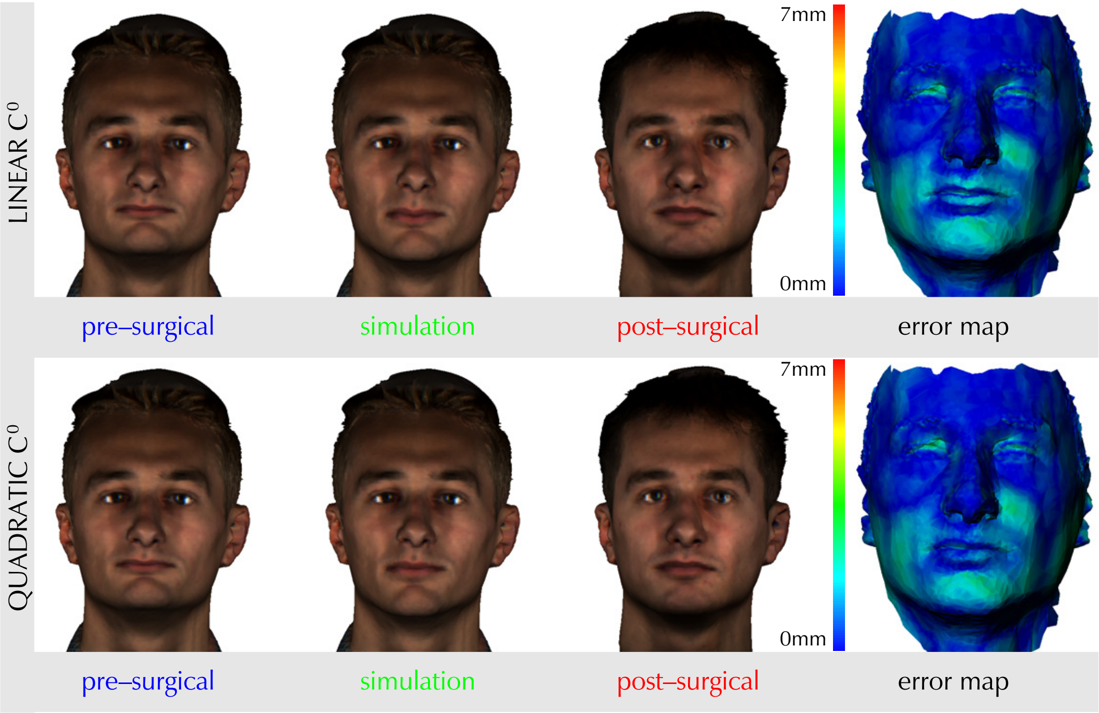
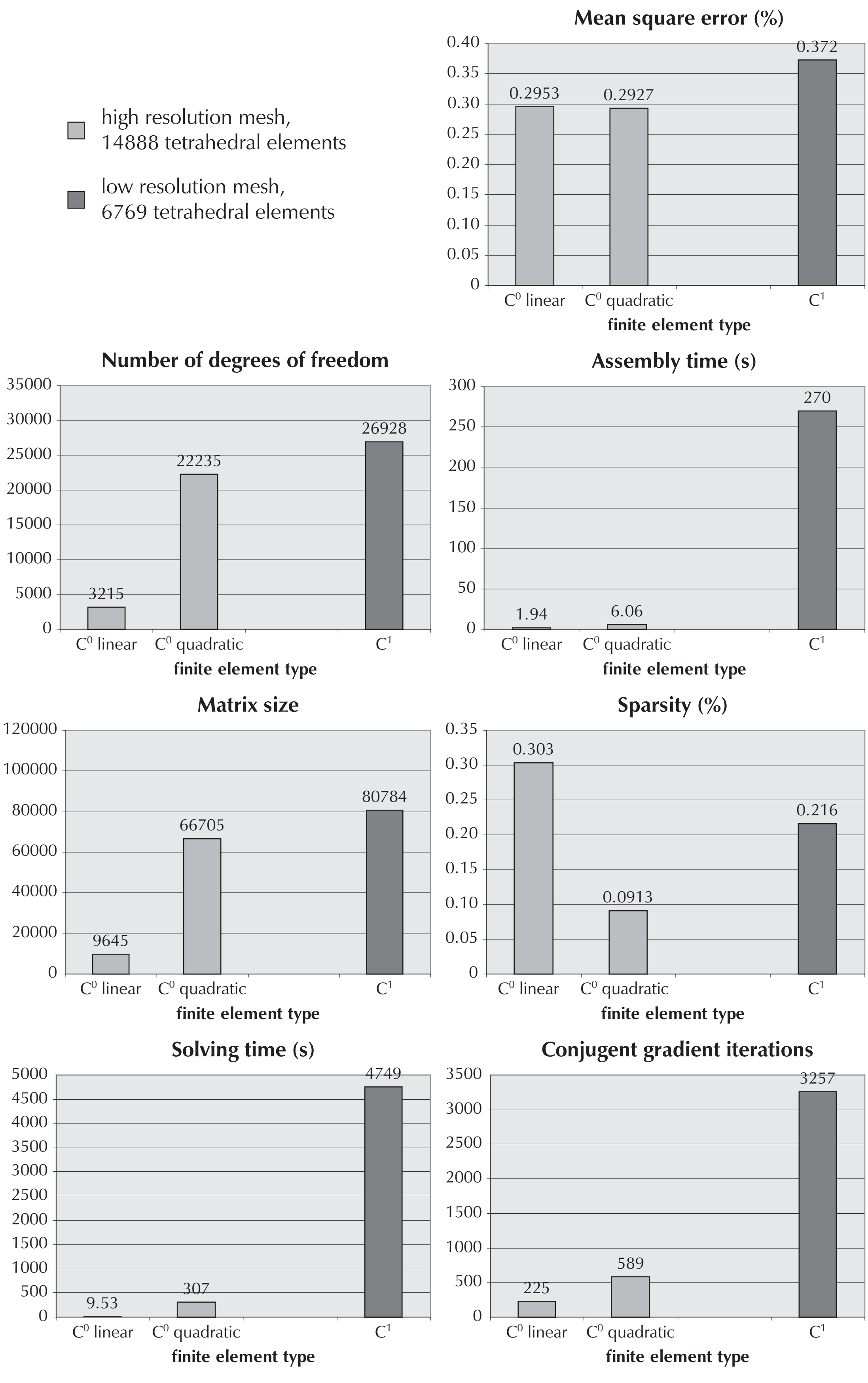


FIGURE 7.22 Frontal view comparison of high resolution simulation results: C^0 linear and quadratic element. **[see Color Plate 10 on page 208]**

TABLE 7.2 Timings and statistics for the test series in Figures 7.21 and 7.22.



CHAPTER 8

CONCLUSIONS AND FUTURE WORK

This chapter summarizes the results of the previous chapters of the thesis. The presentation of findings and conclusions as well as directions for future work complement on this summary.

8.1 SUMMARY AND CONCLUSIONS

We have described techniques and procedures needed in the implementation of a facial surgery simulator which is envisioned to replace or complement on traditional planning instruments in cranio–maxillofacial surgery. Based on data readily available from typical patients subject to cranio–maxillofacial surgery, the simulation allows for a highly accurate physically–based simulation of the procedure prior to actual surgery. The use of such a planning system not only raises current planning techniques from two to three dimensions but also excels current procedures in reliability. Surgeons are given the ability of simulating several variants of the same surgical procedure the best of which is then to be put into execution. Patients are given the option of getting acquainted to their post–surgical appearance and therefore should less often suffer the unpleasant surprise of an unfamiliar post–surgical look.

Tissue model. The facial tissue model has been chosen to be truly volumetric and based on the theory of static elastomechanics. Besides classical linear elasticity, incompressibility and nonlinear stress–strain relations have been looked into. The finite element method proved to be the numerical instrument of choice in order to solve the inherent partial differential equations. For the sake of geometric and topological flexibility, tetrahedral finite elements have been used in the discretization of facial tissue.

The thesis never aimed at the development of a tissue representation which models the biomechanical properties of tissue at utmost accuracy. However, although being a coarse approximation, the model of linear elasticity used in the surgery simulation prototype proved to be well suited for the type of surgical procedures considered in the test example.

Moreover, in accordance with [77], it was even the simplest, pure displacement–based approach which was found to yield the best results and therefore was chosen in the surgery examples. Modeling a nonlinear stress–strain relationship does not make sense in the set–up of the surgery simulation prototype which is based on prescribing displacements. It is only useful in cases when one is interested in the displacements resulting from presumably large forces applied to the body under consideration. Further, forces and consequently strains resulting from surgical procedures should be kept as low as possible in order to limit imposed stresses to an absolute minimum.

Tetrahedral elements have been chosen for their modeling flexibility. However, in the current simulator set–up, this flexibility is not taken advantage of. Only if one were to model muscles as well as different epidermic structures and layers explicitly, the geometric and topological advantages would pay off.

Tetrahedral Bernstein–Bézier finite elements. Bernstein–Bézier representations have been put into the context of finite element analysis. The effect of mesh refinement was opposed to raising the degree of finite element interpolation. Both approaches allow for increasingly accurate simulation results. In order to remedy the enormous growth of the problem dimension due to mesh refinement as well as resulting from higher order interpolation, further studies have been directed to raising the level of continuity of the finite element solution. Consequently, a tetrahedral C^1 Bernstein–Bézier finite element adapted from a theoretical *n*–dimensional interpolant has been proposed. The element was applied to the problem of linear elasticity, although linear elasticity is commonly known to be a C^0 problem. The tetrahedral C^1 construction scheme invalidates neither the approximation properties nor the locality of the Bernstein finite element basis. At the same time it preserves the integral nature of the Bernstein basis and therefore allows for analytical integration in the context of finite element analysis.

In a thorough test series, the C^1–continuous element yielded results which are very similar to the C^0 simulations, especially when using the cubic element. This is in accordance with theoretical results for linear elasticity. In nearly incompressible settings, the C^1 element performed slightly better, but at the price of far less computational efficiency.

The huge increase in displacement degrees of freedom induced by the Clough–Tocher split is more than compensated for by the elimination of linearly dependent nodes. This is in accordance with the theoretical consideration that rising the level of continuity to C^1 imposes additional constraints on the solution and therefore reduces the overall problem dimension. However, the reduction is at the cost of a very costly assembly procedure. Further, in a mixed setting with C^0 pressure interpolation, the number of pressure degrees of freedom is increased significantly due to the split.

The C^1 matrix is more densely populated than its C^0 counterpart and the distribution of values is different. This must be attributed to the elimination of linearly dependent control points. In addition, the condition of C^1 matrices is clearly worse compared to the matrices resulting from a C^0 assembly. As a consequence, the computational costs of solving the systems of equations resulting from C^1 simulations by far exceed the C^0 case. Therefore, the reduction of the problem dimension is only with respect to the size of the global system of equations and not with respect to overall simulation costs.

In conclusion, the Clough–Tocher construction proved to be feasible in the context of finite element analysis. Its application in the context of a C^0 problem such as linear elasticity is hardly justifiable due to the significant increase in computational costs induced by the construction itself and due to a clearly worse solver performance. Hence, the admissible space under minimum restrictions, i.e. the space of C^0 polynomial functions, is clearly to be preferred over the more restrictive space of C^1 functions. However, in fourth order problems such as the thin plate problem governed by the biharmonic equation, the application of the C^1 element would be promising.

Facial surgery simulation prototype. As a proof of concept and for evaluation purposes, a prototype simulator was implemented which is designed for the post–simulation of actual surgery. The prototype allows for the comparison of real and simulated surgery outcome, the equivalence of which is the ultimate goal to strive for. The simulator is designed to be as automatic as possible with respect to the model build–up as well as with regard to the determination of jaw movements. Due to the highly automatic nature of the prototype, user interaction is kept at a minimum and so are potential inconsistencies between the simulation and the real surgical procedure. Registration of data sets was found to be an integral part both of the model preparation and the computation of bone displacement fields. Cutting edge registration techniques therefore make up a major part of the simulator.

The prototype has successfully been applied to the simulation of the correction of a short face syndrome. In order to decide on the optimal finite element and mesh size for this kind of simulation, the real surgery result has been compared to the results of various simulations. Approximation quality as well as simulation efficiency have been taken into consideration. We found that for simulations on a coarse grid, the use of higher order interpolation or even of the C^1–continuous finite element is advantageous as far as approximation quality is concerned. However, cubic C^0 and C^1 simulations are computationally far more expensive. Therefore, linear and quadratic C^0 simulations on a finer grid have been compared to these more expensive cubic simulations. The results achieved on this finer mesh are clearly superior both with respect to the mean square error and with respect to the overall computation time. As a consequence, for this kind of simulation, mesh refinement is to be preferred over higher order or even C^1–continuous interpolation. In conclusion, we found the quadratic C^0 element to perform best with regard to the trade–off between accuracy and efficiency.

Remaining discrepancies between the simulation results and the real surgery outcome must be attributed to the following sources of errors (see also [77]):

- *Data acquisition*
  Data acquisition is a major source of error. Different muscular activities, changing head alignment, motion artifact and the removal of dental braces may influence the correspondence of laser range scans. Further, metallic implants heavily deteriorate the quality of CT scans by causing severe artifacts in the teeth region.

- *Tissue model*
  The assumption of small displacements and small strains may sometimes be violated to a certain extent. Traditional finite element analysis is applied to materials such as metals where the amount of deformation is limited to less than 1% of the object dimensions. In facial tissue simulation, deformations may amount to 10% or even more, depending on the surgical procedure.

- *Model build–up and determination of surgery displacement fields*
  Although the automatic registration procedures employed in the surgery simulation prototype provide maximal accuracy, problems such as the artifacts in CT scans influ-

ence the computation of surgery displacement fields negatively. The artifact reduction scheme employed in the determination of the pre–surgical skull does not provide for an exact capture of the skull geometry. Therefore, an approximation of the pre–surgical skull is registered to the isosurface of the post–surgical skull. Very often, dental braces are altered or removed immediately after surgery, which renders the post–surgical CT less prone to artifacts. As a consequence, minor jaw misalignments are inevitable in the current automatic set–up.

As mentioned above, the flexibility of tetrahedral meshes is not taken advantage of in the current meshing procedure. The cylindrical projection of facial vertices onto the skull and subsequent tetrahedralization of the resulting prism mesh has to be considered a rather straightforward meshing approach. Therefore, the mesh is of doubtful quality for application in finite element procedures. The Clough–Tocher split construction further accentuates these problems by introducing even more sliver–like tetrahedra. Such tetrahedra feature small inside angles which are numerically troublesome.

Triangular Bézier Clipping. Last but not least, Bézier Clipping was adapted to the triangular domain and incorporated as the core of a raytracer for triangular Bézier patches. Triangular Bézier clipping allows for the computation of arbitrarily accurate intersections between rays and triangular Bézier patches. In combination with a bounding volume hierarchy, its computational costs are reasonable both with respect to computation time and memory requirements. Further, the accuracy of triangular Bézier clipping equals its tensor product equivalent while its reliability even excels the original algorithm.

8.2 DIRECTIONS FOR FUTURE WORK

As a consequence of the above conclusions, directions for future work are to be seen in the following areas:

- *Data acquisition*
  The acquisition of both laser range and CT scans offers plenty of space for improvement. The quality of laser range scans, especially with respect to motion artifacts, would greatly benefit from shorter scanning times. However, it is questionable whether today's scanning techniques allow for a noticeable speed–up. Further, it is not only towards shorter scanning times research should be directed to. At the same time, future approaches to 3D scanning should be able to handle cavities the like of which cannot be captured in the cylindrical scanning set–up. Hence, new paradigms in the acquisition of three–dimensional objects have to be investigated. Image–based methods such as [91], voxel coloring or space carving approaches like [129, 81], structured light approaches similar to [116] or combinations thereof seem to be promising starting points for the development of at the same time high quality and low cost scanning techniques. In addition, texture quality can substantially be elaborated upon, e.g. by adding reflectance and scattering information [34].

  Further, the above–mentioned shortcomings with respect to CT scan quality, necessitate the development of new techniques for the reduction of artifacts due to metal implants. The methods proposed in [105, 151] point in that direction.

- *Nonlinear finite element simulation*
  For the simulation of more complex surgical procedures and tissue behavior, sophisticated tissue models will have to be investigated. On the one hand complex surgical

procedures resulting in large displacements necessitate the incorporation of geometrical nonlinearity in order to cope with changing volumes over which stiffness matrix and force vector integrations are to be performed [8, 35]. Lately, Gladilin and Zachow have been investigating such considerations [52].

On the other hand, with increasing computing power, more advanced mechanical models are eligible for the modeling of highly nonlinear biomechanical properties of tissue. In combination with high quality tissue segmentation and discretization, such models would allow for the representation of distinct properties of various anatomical tissue types. A facial model of unrivaled accuracy would result from such a proceeding, opening the door not only to highest reliability in surgery simulation but also to fields like anatomy–based facial animation and modeling of virtual characters.

Last but not least, effects like sliding of tissue on teeth, will necessitate the imposition of complex and even changing boundary conditions. Handling such contact problems also results in nonlinear analysis [8].

All the above–mentioned nonlinear effects greatly increase the computational costs of simulations. Therefore, more efficient methods, e.g. based on adaptive multigrid procedures, could be employed for speed–up.

- *Simulator design and user interfaces*
  In order to capture anatomical structures, like muscles, fat, fasciae, or various epidermic layers, a more sophisticated meshing of tissue is of high importance. Advancing Front–based methods seem to be a promising starting point [101]. Further, only such meshing procedures would truly benefit from the flexibility of tetrahedral meshes.

  For the evaluation of the benefits derived from more advanced finite element tissue modeling and meshing, new perception–based error metrics will have to be developed. The analysis of errors by means of profile lines and frontal error maps does not capture the subtlety of human perception which strongly weighs error as a function of the error location, especially around the eyes and in the region of the mouth.

  Last but not least, in order to establish simulation as a commodity planning instrument for cranio–maxillofacial surgery, more emphasis has to be put on simulator design and user interfaces. Consumer, i.e. surgeon, acceptance not only is based on convincing results but also on practical concerns like ease of use. Therefore, future research efforts have to be spent on the development of interaction metaphors that allow for the definition of osteotomies and the realignment of bony structures as well as for the automatic build–up of facial models.

APPENDIX A

NOMENCLATURE

CHAPTER 3 FINITE ELEMENT MODELING OF ELASTIC MATERIALS

Introductory concepts

Π	potential or functional of a problem
$\delta\Pi$	variation of a functional Π
$\mathbb{R}^n$	Euclidean n–space
Ω, $\overline{\Omega}$	open and closed domain
$\partial\Omega$	boundary of Ω
$C^k(\Omega)$	space of k–times continuously differentiable functions on Ω
$C_0^k(\Omega)$	space of $C^k(\Omega)$ functions which evaluate to zero on $\partial\Omega$
$u(t-t_0)$	Unit step of Heavyside function (generalized function)
$u_T(t-t_0)$	Box impulse (generalized function)
$\delta(t-t_0)$	Dirac impulse function (generalized function)
$H^k(\Omega)$	Sobolev space of functions of order k
$L^2(\Omega)$	space of square–integrable functions

Introduction to the finite element method

(S)	strong or differential problem statement
(W)	weak or variational problem statement, principle of virtual work
(G)	Galerkin problem statement
(M)	matrix equivalent of Galerkin problem statement

U	collection of trial solutions
V	collection of variations or test functions
h	measure of discretization, mesh parameter
U^h	finite–dimensional subspace of U
V^h	finite–dimensional subspace of V
δu , $\bar{u}$	virtual displacements, variations
$a(\cdot, \cdot)$	symmetric bilinear form, internal energy in weak problem statement
$(f, \cdot)$	linear functional, external energy in weak problem statement
b^h	special trial solution satisfying the essential boundary conditions
N_i	basis or shape function in Galerkin projection
N_0	shape function representing the essential boundary condition
$\hat{v}_i$, $\hat{w}_i$	Galerkin weights
$\mathbf{K}$, K_{ij}	stiffness matrix, stiffness matrix entries
$\mathbf{F}$, F_i	force or loading vector, force vector entries
$\mathbf{w}$, $\hat{w}_j$	weight or displacement vector, weight vector entries
t_i	finite element nodes in the discretization of the example problem
$[t_i, t_{i+1}]$	finite elements in the discretization of the example problem

Static elastomechanics

$\mathbf{u}(x, y, z)$, u_i	displacement field
$\mathbf{f}(f_x, f_y, f_z)$	externally applied forces
S_u	supported part of elastic body (prescribed displacements)
$\mathbf{u}^{S_u}$	prescribed displacements, essential boundary conditions on S_u
ε	engineering strain vector
ε_{xx}, ε_{yy}, and ε_{zz}	volumetric strain components in ε
γ_{xy}, γ_{yz}, and γ_{zx}	deviatoric or shear strain components in ε
$\tilde{\varepsilon}$	symmetric strain tensor
ε_{ij}	entries of symmetric strain tensor $\tilde{\varepsilon}$
τ	engineering stress vector
τ_{xx}, τ_{yy}, and τ_{zz}	normal stresses
τ_{xy}, τ_{yz}, and τ_{zx}	shear stresses
$\tilde{\tau}$	symmetric stress tensor
τ_{ij}	entries of symmetric stress tensor $\tilde{\tau}$
E	Young’s modulus
ν	Poisson’s ratio
$\mathbf{C}$	stress–strain material matrix
δ_{ij}	Kronecker delta

λ , μ	Lamé constants
$\bar{\mathbf{u}}$, $\bar{\varepsilon}$	virtual displacements, corresponding virtual strains

Incompressibility

κ	bulk modulus
G	shear modulus
ε_V , ε_{kk}	volumetric strain
ε'_{ij}	deviatoric strain components
p	pressure
τ'	deviatoric stress vector
ε'	deviatoric strain vector

Nonlinear extensions

E_0	Young's modulus for relaxed material
A	measure of material–only nonlinearity
ε_F	Frobenius norm of symmetric strain tensor $\tilde{\varepsilon}$
ε_A	arithmetic mean of symmetric strain tensor entries ε_{ij}

Finite element discretization of static elastomechanics

$\tilde{\mathbf{u}}^{(m)}(x, y, z)$	Galerkin approximation of displacement field within element *m*
$\mathbf{H}^{(m)}$	local displacement interpolation matrix on element *m*
N_i	shape functions for displacement interpolation
$\hat{\mathbf{u}}^{(m)}$	local nodal displacement weight vector on element *m*
$\mathbf{B}^{(m)}$	strain interpolation matrix on element *m*
$\mathbf{U}$, $\hat{\mathbf{U}}$	global nodal weight vector, displacement vector
$\mathbf{R}$	global force vector
$\mathbf{B}_V^{(m)}$	volumetric strain interpolation matrix on element *m*
$\mathbf{B}_D^{(m)}$	deviatoric strain interpolation matrix on element *m*
$p^{(m)}(x, y, z)$	Galerkin approximation of pressure within element *m*
$\mathbf{H}_p^{(m)}$	local pressure interpolation matrix on element *m*
M_i	shape functions for pressure interpolation
$\hat{\mathbf{p}}^{(m)}$	local nodal pressure weight vector on element *m*
$\mathbf{C}'^{(m)}$	deviatoric stress–strain material matrix on element *m*
$\mathbf{K}_{uu}$	displacement part of mixed stiffness matrix
$\mathbf{K}_{up}$, $\mathbf{K}_{pu}$	mixed parts of mixed stiffness matrix
$\mathbf{K}_{pp}$	pressure part of mixed stiffness matrix
${}^t\mathbf{K}$	tangent stiffness matrix
$\Delta\mathbf{U}^{(i)}$	increment in nodal weight vector in Newton–Raphson iteration

CHAPTER 4 BERNSTEIN POLYNOMIALS AND BÉZIER PATCHES

Introductory concepts

u, v, w	global parametric coordinates
q, r, s, t	local parametric coordinates (barycentric and tensor–product)

Bézier curves

$B_i^n(t)$	univariate Bernstein polynomials of degree n
$\mathbf{b}_i$	one–dimensional Bézier control points
$\mathbf{b}^n(t)$	univariate Bézier curve of degree n
$\mathbf{b}_i^r(t)$	intermediate Bézier control points from de Casteljau iteration r
Δ^r	(iterated) forward differencing operator
$H_i^3(t)$	cubic Hermite polynomials

Tensor product extension

$\mathbf{b}^{m,n}(r, s)$	bivariate tensor product Bézier patch of degree $m \times n$
$\mathbf{b}_{i,j}$	bivariate tensor product Bézier control points
$\mathbf{b}_{i,j}^{k,k}(r, s)$	intermediate Bézier control points from bivariate de Casteljau iteration k

Triangular Bernstein–Bézier patches

$\mathbf{r}$	barycentric point of coordinates r, s, t
$\mathbf{i}$	index triplet i, j, k of control points
$B_{\mathbf{i}}^n(\mathbf{r})$	triangular, barycentric bivariate Bernstein polynomials of degree n
$\mathbf{b}_i$	triangular bivariate Bézier control points
$\mathbf{b}^n(\mathbf{r})$	barycentric Bézier patch of degree n
$\mathbf{b}_{i,j,k}^l(\mathbf{r})$	intermediate Bézier control points from barycentric de Casteljau iteration l
$\mathbf{d}$	barycentric directional vector
$D_{\mathbf{d}}^l\mathbf{u}(\mathbf{r})$	l–th directional derivative of $\mathbf{u}(\mathbf{r})$ at $\mathbf{r}$ with respect to $\mathbf{d}$
$\mathbf{b}[\mathbf{r}^{\langle i\rangle}, \mathbf{s}^{\langle j\rangle}, \mathbf{t}^{\langle k\rangle}]$	blossoming principle

Multivariate barycentric Bernstein–Bézier patches

τ	N–dimensional barycentric point of coordinates τ_i
λ	N–dimensional index vector of indices λ_i of control points
P_N	N–dimensional simplex
T_i	corners of N-simplex in $\mathbb{R}^N$
$B_\lambda^n(\tau)$	N–dimensional barycentric Bernstein polynomials of degree n
$\mathbf{b}^n(\tau)$	N–dimensional barycentric Bézier patch of degree n

$\mathbf{b}^{l}_{\lambda}(\tau)$	intermediate Bézier control points from barycentric de Casteljau iteration *l* in *N* dimensions
$\tau = (r, s, t, q)$	trivariate tetrahedral barycentric coordinates
$\lambda = (i, j, k, l)$	trivariate tetrahedral barycentric indices of control points

CHAPTER 5 A TETRAHEDRAL C^1 BERNSTEIN–BÉZIER FINITE ELEMENT

One–dimensional example

$u(t)$	global interpolation function
t	global parameter
$[t_i, t_{i+1}]$	finite element
h_i	length of finite element
ξ	local parameter
I_S	internal energy
I_F	external energy
c_0, c_1, c_2, c_3	weight of cubic monomials
$\tilde{\mathbf{S}}$, $\tilde{\mathbf{F}}$	stiffness matrix and force vector w.r.t. cubic monomials
$\hat{\mathbf{h}}$	vector of local Hermite variables
$\hat{\mathbf{S}}$, $\hat{\mathbf{F}}$	stiffness matrix and force vector w.r.t. local Hermite variables and w.r.t. local Bézier variables
$\mathbf{h}$	vector of global Hermite variables
$\mathbf{S}$, $\mathbf{F}$	stiffness matrix and force vector w.r.t. global Hermite variables and w.r.t. global Bézier variables
$\mathbf{b}$	vector of Bézier weights

Tetrahedral Bernstein–Bézier elements

$\mathbf{J}$	Jacobian matrix or operator
$B^{n}_{ijkl}(r, s, t, q)$	trivariate barycentric Bernstein polynomials
$\mathbf{b}^{n}(r, s, t, q)$	tetrahedral Bézier element of degree *n*
T	tetrahedral element
$A(T)$	set of degrees of freedom pertaining to element T
$B(T)$	set of degrees of freedom influencing linearly dependent nodes on T
n, N	number of local/global displacement degrees of freedom
m, M	number of local/global pressure degrees of freedom
i, j, I, J	local/global index of example corner nodes
$x_{i/j}, y_{i/j}, z_{i/j}$	indices of *x*–, *y*–, and *z*–components in local stiffness matrix
$X_{I/J}, Y_{I/J}, Z_{I/J}$	indices of *x*–, *y*–, and *z*–components in global stiffness matrix

$c_{I,J}$, $c_{i,j}$	relational coefficient between two degrees of freedom, entry in stiffness matrix
$\mathbf{b}^f$	global Bézier degree of freedom
$\mathbf{b}^{ld}$	linearly dependent Bézier node
σ_k	weights of linear dependence
$\mathbf{M}^{(m)}$	local elimination matrix for linearly dependent nodes in element m
$c_x(k)$, $c_y(k)$, $c_z(k)$	index mappings used in building $\mathbf{M}^{(m)}$

CHAPTER 6 RAY TRACING TRIANGULAR BÉZIER PATCHES

Ray–patch intersection

$B^n_{ij}(r, s)$	triangular Bernstein polynomials omitting $t = 1 - r - s$
$\mathbf{b}_{ij}$	triangular Bézier control points omitting $k = n - i - j$
$\mathbf{b}^n(r, s)$	triangular Bézier patch omitting $t = 1 - r - s$
$\mathbf{r}(u)$	definition of the ray
$\mathbf{n}_k = (a_k, b_k, c_k)^T$	plane normals of ray representation as intersection of two planes
e_k	distance to origin of ray representation planes
$\mathbf{d}^n(r, s)$	two–dimensional triangular Bézier patch representation
$\mathbf{d}_{ij}$	control points of $\mathbf{d}^n(r, s)$

Bézier clipping

u_{min}, v_{min}	Bézier clipping minima (tensor product)
u_{max}, v_{max}	Bézier clipping maxima (tensor product)
r_{min}, s_{min}, t_{min}	Bézier clipping minima (barycentric)
L_r	distance line through origin parallel to $r = 0$
L_s	distance line through origin parallel to $s = 0$
L_t	distance line through origin parallel to $t = 0$
L_u	distance line through origin parallel to u parameter direction
L_v	distance line through origin parallel to v parameter direction
$dr^n(r, s)$	Bézier representation of distance to L_r
dr_{ij}	control points to $dr^n(r, s)$
$\mathbf{dr}^n(r, s)$	functional distance patch of $dr^n(r, s)$
$\mathbf{dr}_{ij}$	control points to $\mathbf{dr}^n(r, s)$
$\mathbf{c}_{ij}$	control points of clipped sub–patch
P_h, P_l	upper and lower convex hull polyline

CHAPTER 7 A PROTOTYPE FOR SURGERY SIMULATION

Surface registration

$\dot{q}$	quaternion
$\dot{\mathbf{q}}$	vector representation of a quaternion
i, j, k	imaginary units of quaternions
$\mathbf{Q}$, $\overline{\mathbf{Q}}$	matrix representation of the left and right quaternion in the product of two quaternions
$\dot{q}^{-1}$	inverse of a quaternion
$\dot{q}^*$	conjugate of a quaternion
P, $\mathbf{p}_i$	data surface, data surface points
X	model surface
Y, $\mathbf{y}_i$	closest points surface, closest points
$\dot{\mathbf{q}}_R$	sought–after rotation quaternion
$\mathbf{q}_T$	sought–after translation vector
$\mathbf{q} = [\dot{\mathbf{q}}_R \vert \mathbf{q}_T]$	registration vector
$f(\mathbf{q})$	objective function of registration
μ_P, μ_Y	centroids of data points and closest points
$\mathbf{p}_i'$, $\mathbf{y}_i'$	centroid coordinates of data points and closest points

Image registration

$f_R(\mathbf{x})$	reference image
$f_T(\mathbf{x})$	test image
ε^2	least–square error (objective function)
$Q_{\mathbf{p}}$	sought–after transformation according to the parameters $\mathbf{p}$
$\chi^2(\mathbf{p})$	discrete objective function of registration
$\mathbf{p}_t$	registration parameters in iteration t
$\delta\mathbf{p}_t$	registration parameter correction vector in iteration t
$\nabla\chi^2$	gradient of objective function
h_t	step size in iteration t
$\mathbf{D}$, $\mathbf{H}_t$	Hessian of $\chi^2(\mathbf{p}_t)$
$\mathbf{d}$	negative gradient of objective function
$[\alpha_{kl}]$	matrix resulting from Hessian $\mathbf{H}_t$
$[\beta_k]$	vector resulting from gradient $\nabla\chi^2$
$[\alpha_{kl}']$	adjusted Hessian matrix in Levenberg–Marquardt method
λ	Levenberg–Marquardt parameter

APPENDIX B

REFERENCES

[1] P. Alfeld. "A Bivariate C2 Clough-Tocher Scheme." *Computer Aided Geometric Design*, 1(3):257–267, 1984.

[2] P. Alfeld. "A Trivariate Clough-Tocher Scheme for Tetrahedral Data." *Computer Aided Geometric Design*, 1(2):169–181, 1984.

[3] P. Alfeld. "On the Dimension of Piecewise Polynomial Functions." In *Proceedings of the Biennal Dundee Conference on Numerical Analysis*. Pitman Publishers, Boston, 1985.

[4] C. L. Bajaj, F. Bernardini, and G. Xu. "Automatic Reconstruction of Surfaces and Scalar Fields from 3D Scans." In *SIGGRAPH'95 Conference Proceedings*, Annual Conference Series, pages 109–118. ACM SIGGRAPH, Addison Wesley, 1995.

[5] D. Baraff and A. Witkin. "Large Steps in Cloth Simulation." In *SIGGRAPH'98 Conference Proceedings*, Annual Conference Series, pages 43–54. ACM SIGGRAPH, Addison Wesley, 1998.

[6] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM Publications, Philadelphia, 1994.

[7] W. Barth and W. Stürzlinger. "Efficient Ray Tracing for Bézier and B-Spline Surfaces." *Computers & Graphics*, 17(4):423–430, 1993.

[8] K.-J. Bathe. *Finite Element Procedures*. Prentice Hall, Englewood Cliffs, New Jersey, 2nd edition, 1996. Rev. of: Finite Element Procedures in Engineering Analysis. 1982.

[9] M. W. Bern and D. Eppstein. "Mesh Generation and Optimal Triangulation." In D.-Z. Du and F. K. Hwang, editors, *Computing in Euclidean Geometry*, number 4 in Lecture Notes Series on Computing, pages 47–123. World Scientific, 2nd edition, 1995.

[10] S. Bernstein. “Démonstration du Théorème de Weierstrass fondeé sur le calcul des Probabilités.” *Harkov Soobs. Matem ob-va*, 13:1–2, 1912.

[11] P. J. Besl and N. D. McKay. “A Pyramid Approach to Subpixel Registration Based on Intensity.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, Feb. 1992.

[12] P. Bézier. “Définition Numérique des Courbes et Surfaces I.” *Automatisme*, XI:625–632, 1966.

[13] P. Bézier. “Définition Numérique des Courbes et Surfaces II.” *Automatisme*, XII:17–21, 1967.

[14] D. Bielser and M. H. Gross. “Interactive Simulation of Surgical Cuts.” In *Proceedings of Pacific Graphics 2000*, pages 116–125. IEEE Computer Society Press, 2000. held in Hong Kong, China, 2–5 October 2000.

[15] D. Bielser, V. A. Maiwald, and M. H. Gross. “Interactive Cuts through 3-Dimensional Soft Tissue.” In *COMPUTER GRAPHICS Forum*, Vol. 18, No. 3, pages C31–C38. Eurographics, Blackwell Publishers Ltd, 1999. EUROGRAPHICS’99 Conference issue.

[16] V. Blanz and T. Vetter. “A Morphable Model for the Synthesis of 3D Faces.” In *SIGGRAPH’99 Conference Proceedings*, Annual Conference Series, pages 187–194. ACM SIGGRAPH, Addison Wesley, 1999.

[17] J. Bloomenthal, editor. *Introduction to Implicit Surfaces*. Morgan-Kaufmann, 1997.

[18] S. R. Bodner and M. B. Rubin. “A Study of the Extensional Properties of Excised Facial Skin and SMAS.” In *Fifth Pan American Congress of Applied Mechanics – PACAM V*, 1997. held in San Juan, Puerto Rico, 2–4 January 1997.

[19] W. Böhm and G. Farin. “Letter to the Editor.” *Computer-Aided Design*, 15(5):260–261, 1983.

[20] W. Böhm and G. Farin. “Computer Aided Geometry Design: Geometric Principles and Applications.” Tutorial Note EG 91 TN 5, Eurographics, 1991.

[21] M. Bro-Nielsen. “Modelling Elasticity in Solids using Active Cubes – Application to Simulated Operations.” In *Proc. of Computer Vision, Virtual Reality, and Robotics in Medicine ’95 (CVRMed’95)*, 1995.

[22] M. Bro-Nielsen and S. Cotin. “Real-time Volumetric Deformable Models for Surgery Simulation using Finite Elements and Condensation.” In *COMPUTER GRAPHICS Forum*, Vol. 15, No. 3, pages C57–C66. Eurographics, Blackwell Publishers Ltd, 1996. EUROGRAPHICS’96 Conference issue.

[23] I. N. Bronstein, K. A. Semendjajew, G. Musiol, and H. Mühlig. *Taschenbuch der Mathematik*. Verlag Harri Deutsch, Frankfurt am Main, Thun, 4th edition, 1999. Rev. of: I. N. Bronstein, K. A. Semendjajew: Taschenbuch der Mathematik für Ingenieure und Studenten, FIZMATLIT, Moskau, 1977.

[24] S. Campagna and P. Slusallek. “Improving Bézier Clipping and Chebyshev Boxing for Ray Tracing Parametric Surfaces.” In B. Girod, H. Niemann, and H.-P. Seidel, editors, *Proceedings of 3D Image Analysis and Synthesis ’96*, pages 95–102, 1996.

[25] S. Campagna, P. Slusallek, and H.-P. Seidel. "Ray Tracing of Spline Surfaces: Bézier Clipping, Chebyshev Boxing, and Bounding Volume Hierarchy – A Critical Comparison with New Results." *The Visual Computer*, 13(6):265–282, 1997.

[26] G. Celniker and D. Gossard. "Deformable Curve and Surface Finite Elements for Free-Form Shape Design." In *SIGGRAPH'91 Conference Proceedings*, Annual Conference Series, pages 257–266. ACM SIGGRAPH, Addison Wesley, 1991.

[27] D. D. Chen and D. Zeltzer. "Pump It Up: Computer Animation Based Model of Muscle Using the Finite Element Method." In *SIGGRAPH'92 Conference Proceedings*, Annual Conference Series, pages 89–98. ACM SIGGRAPH, Addison Wesley, 1992.

[28] R. W. Clough and J. L. Tocher. "Finite Element Stiffness Matrices for Analysis of Plates in Bending." In *Proceedings of Conference on Matrix Methods in Structural Mechanics*, pages 515–545. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 1965.

[29] R. Cools. "Constructing Cubature Formulae: The Art Behind the Science." *Acta Numerica*, 6:1–54, 1997.

[30] C. de Boor. "B-form Basics." In G. E. Farin, editor, *Geometric Modelling: Algorithms and New Trends*, pages 131–148. SIAM Publications, Philadelphia, 1987.

[31] P. de Casteljau. "Outillage Méthodes Calcul." Technical report, André Citroën Automobiles SA, Paris, 1959.

[32] P. de Casteljau. "Courbes et Surfaces à Poles." Technical report, André Citroën Automobiles SA, Paris, 1963.

[33] P. de Casteljau. *Shape Mathematics and CAD*. Kogan Page, London, 1986.

[34] P. Debevec, T. Hawkins, C. Tchou, H.-P. Duiker, W. Sarokin, and M. Sagar. "Acquiring the Reflectance Field of a Human Face." In *SIGGRAPH 2000 Conference Proceedings*, Annual Conference Series, pages 145–156. ACM SIGGRAPH, Addison Wesley, 2000.

[35] X. Q. Deng. *A Finite Element Analysis of Surgery of the Human Facial Tissue*. PhD thesis, Columbia University, New York, 1988.

[36] F. A. Duck. *Physical Properties of Tissue – A Comprehensive Reference Book*. Academic Press, London, 1990.

[37] I. S. Duff, R. G. Grimes, and J. G. Lewis. "Sparse Matrix Test Problems." *ACM Transactions on Mathematical Software*, 15(1):1–14, 1989.

[38] G. Farin. *Subsplines über Dreiecken*. PhD thesis, Technical University Braunschweig, Germany, 1979.

[39] G. Farin. "Smooth Interpolation to Scattered 3D Data." In R. Barnhill and W. Boehm, editors, *Surfaces in CAGD*. North-Holland, Amsterdam, 1983.

[40] G. Farin. "A Modified Clough-Tocher Interpolant." *Computer Aided Geometric Design*, 2(1–3):19–27, 1985.

[41] G. Farin. "Triangular Bernstein-Bézier Patches." *Computer Aided Geometric Design*, 3(2):83–127, 1986.

[42] G. Farin. “The Use of Triangular Patches in CAD.” In M. Wozny, H. McLaughlin, and J. Encarnacao, editors, *Geometric Modeling for CAD Applications*, pages 191–194. North-Holland, Amsterdam, 1988.

[43] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design*. Academic Press, 2nd edition, 1990.

[44] L. G. Farkas. *Anthropometry of the Head and Face*. Raven Press, 2nd edition, 1994.

[45] R. Farouki and V. Rajan. “On the numerical Condition of Polynomials in Bernstein Form.” *Computer Aided Geometric Design*, 4(3):191–216, 1987.

[46] O. Faugeras and M. Hebert. “The Representation, Recognition, and Locating of 3-D Objects.” *Int. Journal of Robotics Research*, 5(3):27–52, 1989.

[47] A. Fournier and J. Buchanan. “Chebyshev Polynomials for Boxing and Intersections of Parametric Curves and Surfaces.” In *COMPUTER GRAPHICS Forum*, Vol. 13, No. 3, pages 127–142. Eurographics, Basil Blackwell Ltd, 1994. EUROGRAPHICS’94 Conference issue.

[48] Y. C. Fung. *Biomechanics: Mechanical Properties of Living Tissues*. Springer, 2nd edition, 1993.

[49] W. Gander and J. Hrebícek, editors. *Solving Problems in Scientific Computing using Maple and Matlab*, chapter 23, Least Squares Fit of Point Clouds, W. Gander, pages 339–349. Springer, 3rd, expanded and revised edition, 1997.

[50] W. Gander and D. Sourlier. “Best-Fit of Sculptured Surfaces.” Technical Report 307, Institute of Scientific Computing, Swiss Federal Institute of Technology (ETH), Zürich, October 1998.

[51] S. F. F. Gibson and B. Mirtich. “A Survey of Deformable Modeling in Computer Graphics.” Technical Report TR-97-19, Mitsubishi Electric Research Lab. (MERL), Cambridge, MA, Nov. 1997.

[52] E. Gladilin, S. Zachow, P. Deuflhard, and H.-C. Hege. “Validation of a Linear Elastic Model for Soft Tissue Simulation in Craniofacial Surgery.” In *Proceedings of Medical Imaging 2001: Visualization, Display, and Image-Guided Procedures*, volume 4319, pages 27–35. International Society for Optical Engineering, SPIE, May 2001.

[53] A. Glassner, editor. *An Introduction to Ray Tracing*. Academic Press, 1989.

[54] R. N. Goldman. “Subdivision Algorithms for Bézier Triangles.” *Computer-Aided Design*, 15(3):159–166, 1983.

[55] R. N. Goldman and D. J. Filip. “Conversion from Bézier Rectangles to Bézier Triangles.” *Computer-Aided Design*, 19(1):25–27, 1987.

[56] G. H. Golub and C. F. van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore and London, 3rd edition, 1996.

[57] M. H. Gross. *Visual Computing*. Springer, 1994.

[58] W. R. Hamilton. “On a New Species of Imaginary Quantities Connected with a Theory of Quaternions.” In *Proceedings of the Royal Irish Academy, vol. 2,* pages 424–434, 1844. read at the meeting of the Royal Irish Academy, November 13th, 1843.

[59] W. R. Hamilton. *Elements of Quaternions*, volume I–II. Chelsea Publishing Co., New York, 3rd edition, 1969. originally published by Longmans, Green & Co., London, 1866.

[60] P. C. Hammer, O. J. Marlowe, and A. H. Stroud. “Numerical Integration Over Simplexes and Cones.” *Mathematical Tables and Other Aids to Computation*, 10(55):130–137, 1956.

[61] R. J. Hanson and M. J. Norris. “Analysis of Measurements Based on the Singular Value Decomposition.” *SIAM J. on Sci. and Stat. Comp.*, 2(3):363–373, Sept. 1981.

[62] R. Haralick, H. Joo, C. Lee, X. Zhuang, V. Vaidya, and M. Kim. “Pose Estimation from Corresponding Point Data.” *IEEE Transactions on Systems, Man and Cybernetics*, 19(6):1426–1446, 1989.

[63] M. R. Hestenes and E. Stiefel. “Methods of Conjugate Gradients for Solving Linear Systems.” *Journal of Research of the National Bureau of Standards*, 49:409–436, 1952.

[64] B. K. Horn. *Robot Vision*. MIT Press, Cambridge Massachusetts & McGraw-Hill, New York, 1986.

[65] B. K. Horn. “Closed-Form Solution of Absolute Orientation using Unit Quaternions.” *Journal of the Optical Society of America A, Optics and Image Science*, 4(4):629–642, Apr. 1987.

[66] B. K. Horn, H. M. Hilden, and S. Negahdaripour. “Closed-Form Solution of Absolute Orientation using Orthonormal Matrices.” *Journal of the Optical Society of America A, Optics and Image Science*, 5(7):1127–1135, July 1988.

[67] J. Hoschek and D. Lasser. *Grundlagen der geometrischen Datenverarbeitung*. B.G. Teubner, Stuttgart, 1989. english translation: Fundamentals of Computer Aided Geometric Design, A.K. Peters, 1993.

[68] T. J. R. Hughes. *The Finite Element Method, Linear Static and Dynamic Finite Element Analysis*. Prentice Hall, Englewood Cliffs, New Jersey, 1987.

[69] D. L. James and D. K. Pai. “ARTDEFO – Accurate Real Time Deformable Objects.” In *SIGGRAPH’99 Conference Proceedings*, Annual Conference Series, pages 65–72. ACM SIGGRAPH, Addison Wesley, 1999.

[70] G. Josef and W. Purgathofer. “Deformation of Solids with Trivariate B-Splines.” In *COMPUTER GRAPHICS Forum*, Vol. 8, No. 3, pages C137–C148. Eurographics, Blackwell Publishers Ltd, 1989. EUROGRAPHICS’89 Conference issue.

[71] J. T. Kajiya. “Ray Tracing Parametric Patches.” In *SIGGRAPH’82 Conference Proceedings*, Annual Conference Series, pages 245–254. ACM SIGGRAPH, Addison Wesley, 1982.

[72] M. Kass, A. Witkin, and D. Terzopoulos. “Snakes: Active Contour Models.” In *Proc. of IEEE Conference on Computer Vision*, pages 259–268, London, England, 8-11 1987. also in: International Journal of Computer Vision 1:321–331.

[73] E. Keeve, S. Girod, and B. Girod. “Computer-Aided Craniofacial Surgery.” In *Proceedings of CARS’96*, Computer Assisted Radiology and Surgery, 10th International Congress and Exhibition, pages 757–762. Elsevier Publishers, 1996.

[74] E. Keeve, S. Girod, P. Pfeifle, and B. Girod. “Anatomy-Based Facial Tissue Modeling using the Finite Element Method.” In *Proc. of IEEE Visualization ’96*, pages 21–28, 1996.

[75] R. Klein, G. Liebich, and W. Strass}er. “Mesh Reduction with Error Control.” In R. Yagel and G. M. Nielson., editors, *IEEE Visualization ’96*, pages 311–318, 1996.

[76] L. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel. “Interactive Multi-Resolution Modeling on Arbitrary Meshes.” In *SIGGRAPH’96 Conference Proceedings*, Annual Conference Series, pages 105–114. ACM SIGGRAPH, Addison Wesley, 1998.

[77] R. M. Koch. *Methods for Physics Based Facial Surgery Prediction*. PhD thesis, Swiss Federal Institute of Technology (ETH), Zürich, Switzerland, 2000. No. 13912.

[78] R. M. Koch, M. H. Gross, and A. A. Bosshard. “Emotion Editing using Finite Elements.” In *COMPUTER GRAPHICS Forum*, Vol. 17, No. 3, pages C295–C302. Eurographics, Blackwell Publishers Ltd, 1998. EUROGRAPHICS’98 Conference issue.

[79] R. M. Koch, M. H. Gross, F. R. Carls, D. F. von Büren, G. Fankhauser, and Y. I. H. Parish. “Simulating Facial Surgery using Finite Element Models.” In *SIGGRAPH’96 Conference Proceedings*, Annual Conference Series, pages 421–428. ACM SIGGRAPH, Addison Wesley, 1996.

[80] R. M. Koch, S. H. M. Roth, M. H. Gross, A. P. Zimmermann, and H. F. Sailer. “A Framework for Facial Surgery Simulation.” Technical Report 326, Institute of Scientific Computing, Swiss Federal Institute of Technology (ETH), Zürich, 1998.

[81] K. N. Kutulakos and S. M. Seitz. “A Theory of Shape by Space Carving.” *International Journal of Computer Vision, Marr Prize Special Issue,*, 38(3):199–218, 2000.

[82] W. Larrabee. “A Finite Element Model of Skin Deformation. I Biomechanics of Skin and Soft Tissue: A Review.” In *Laryngoscope*, pages 399–405. Lippincott Williams & Wilkins, 1986.

[83] W. Larrabee. “A Finite Element Model of Skin Deformation. II An Experimental Model of Skin Deformation.” In *Laryngoscope*, pages 406–412. Lippincott Williams & Wilkins, 1986.

[84] Y. Lee, D. Terzopoulos, and K. Waters. “Realistic Face Modeling for Animation.” In *SIGGRAPH’95 Conference Proceedings*, Annual Conference Series, pages 55–62. ACM SIGGRAPH, Addison Wesley, 1995.

[85] K. Levenberg. “A Method for the Solution of Certain Nonlinear Problems in Least Squares.” *Quart. J. Appl. Math.*, 2:164–168, 1944.

[86] W. E. Lorensen and H. E. Cline. “Marching Cubes: A High Resolution 3D Surface Construction Algorithm.” In *SIGGRAPH’87 Conference Proceedings*, Annual Conference Series, pages 163–169. ACM SIGGRAPH, Addison Wesley, 1987.

[87] N. Luscher. *Die Bernstein-Bézier-Technik in der Methode der Finiten Elemente*. PhD thesis, Technische Universität Carolo-Wilhelmina zu Braunschweig, Germany, 1987.

[88] R. MacCracken and K. I. Joy. "Free-Form Deformations with Lattices of Arbitrary topology." In *SIGGRAPH'96 Conference Proceedings*, Annual Conference Series, pages 181–188. ACM SIGGRAPH, Addison Wesley, 1996.

[89] D. W. Marquardt. "An Algorithm for Least-Squares Estimation of Nonlinear Parameters." *J. Soc. Indust. Appl. Math.*, 11:431–441, 1963.

[90] J. T. Marti. *Introduction to Sobolev Spaces and Finite Element Solution of Elliptic Boundary Value Problems*. Computational Mathematics and Applications. Academic Press, Harcourt Brace Jovanovich, Publishers, 1986.

[91] W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan. "Image-Based Visual Hulls." In *SIGGRAPH 2000 Conference Proceedings*, Annual Conference Series, pages 369–374. ACM SIGGRAPH, Addison Wesley, 2000.

[92] C. Monserrat, V. Hernández, M. Alcañiz, M. C. Juan, and V. Grau. "Evaluation and Study of a New Deformable Model Based on Boundary Element Methods." In *Proceedings of CARS'99*, Computer Assisted Radiology and Surgery, 13th International Congress and Exhibition, pages 860–864. Elsevier Publishers, 1999.

[93] C. Monserrat, U. Meier, F. Chinesta, M. Alcañiz, and V. Grau. "A Fast Real Time Tissue Deformation Algorithm for Surgery Simulation." In *Proceedings of CARS'97*, Computer Assisted Radiology and Surgery, 11th International Congress and Exhibition, pages 812–817. Elsevier Publishers, 1997.

[94] C. Monserrat, U. Meier, M. C. Juan, M. Alcañiz, C. Knoll, V. Grau, and F. C. C. Duval. "A New Approach for the Real-time Simulation of Tissue Deformations." *Rhéologie des materiaux du vivant*, 16(3):290–297, 1999.

[95] National Library of Medicine. "The Visible Human Project," 1995. http://www.nlm.nih.gov/research/visible/visible_human.html.

[96] A. Nawotki. "Konstruktion von Bézierflächen aus der Plattengleichung mit variablen Materialparametern." Master's thesis, Fachbereich Mathematik, Universität Kaiserslautern, 1995.

[97] A. Nawotki. "Flächenmodifikation mit der Methode der finiten Eelemente." In *GI Seminar Effiziente Methoden der geometrischen Modellierung und der wissenschaftlichen Visualisierung*, Dagstuhl, 5.–7. May 1997.

[98] P. J. Neugebauer. "Hochgenaue Objektlokalisation in Tiefenbildern." In *Visualisierung – Rolle von Interaktivität und Echtzeit*, Sankt Augustin, Schloss Birlinghoven, June 1992. GMD – Gesellschaft f¸r Mathematik und Datenverarbeitung mbH.

[99] T. Nishita, T. W. Sederberg, and M. Kakimoto. "Ray Tracing Trimmed Rational Surface Patches." In *SIGGRAPH'90 Conference Proceedings*, Annual Conference Series, pages 337–345. ACM SIGGRAPH, Addison Wesley, 1990.

[100] S. J. Owen. "Meshing Research Corner." http://www.andrew.cmu.edu/user/sowen/mesh.html.

[101] S. J. Owen. "A Survey of Unstructured Mesh Generation Technology." In *Proceedings 7th International Meshing Roundtable*, Dearborn, MI, Oct. 1998. also available at: http://www.andrew.cmu.edu/user/sowen/survey/.

[102] C. R. P. Cignoni and R. Scopigno. “Metro: Measuring Error on Simplified Surfaces.” *COMPUTER GRAPHICS Forum*, 17(2):167–174, 1998.

[103] F. Parke and K. Waters. *Computer Facial Animation*. A K Peters, 1996.

[104] F. I. Parke. “Parameterized Models for Facial Animation.” *IEEE Computer Graphics and Applications*, 2(9):61–68, Nov. 1982.

[105] M. Path, C. P. Zollikofer, and P. Stucki. “New Approaches in CT Artifact Suppression – a Case Study in Maxillofacial Surgery.” In *Proceedings of CARS’98*, Computer Assisted Radiology and Surgery, 12th International Congress and Exhibition, pages 830–835. Elsevier Publishers, 1998.

[106] S. D. Pieper. *CAPS: Computer-Aided Plastic Surgery*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1992.

[107] F. Pighin, J. Hecker, D. Lischinski, R. Szeliski, and D. H. Salesin. “Synthesizing Realistic Facial Expressions from Photographs.” In *SIGGRAPH’98 Conference Proceedings*, Annual Conference Series, pages 75–84. ACM SIGGRAPH, Addison Wesley, 1998.

[108] S. M. Platt. “Animating Facial Expressions.” In *SIGGRAPH’81 Conference Proceedings*, Annual Conference Series, pages 245–252. ACM SIGGRAPH, Addison Wesley, 1981.

[109] E. P. Popov. *Introduction to Mechanics of Solids*. Prentice Hall, Englewood Cliffs, New Jersey, 1968.

[110] M. J. D. Powell and M. A. Sabin. “Piecewise Quadratic Approximations on Triangles.” *ACM Transactions on Mathematical Software*, 3(4):316–325, 1977.

[111] F. P. Preparata and M. I. Shamos. *Computational Geometry*. Springer, New York, 1985.

[112] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing (2nd ed.)*. Cambridge University Press, Cambridge, 1992.

[113] L. Ramshaw. “Blossoming: a Connect–the–Dots Approach to Splines.” Technical report, Digital Systems Research Center, Palo Alto, CA, 1987.

[114] L. Ramshaw. “Blossoms Are Polar Forms.” *Computer Aided Geometric Design*, 6(4):323–359, 1989.

[115] J. D. Renton. *Applied Elasticity: Matrix and Tensor Analysis of Elastic Continua*. Ellis Horwood series in mechanical engineering. Ellis Horwood Limited, 1987.

[116] C. Rocchini, P. Cignoni, C. Montani, P. Pingi, and R. Scopigno. “A Low Cost 3D Scanner Based on Structured Light.” In *COMPUTER GRAPHICS Forum*, Vol. 20, No. 3, pages C299–C308. Eurographics, Blackwell Publishers Ltd, 2001. EUROGRAPHICS’2001 Conference issue.

[117] S. H. M. Roth, P. Diezi, and M. H. Gross. “Triangular Bézier Clipping.” In *PACIFIC GRAPHICS 2000 Proceedings*, pages 413–414. IEEE, IEEE Computer Society, 2000.

[118] S. H. M. Roth, P. Diezi, and M. H. Gross. “Ray Tracing Triangular Bézier Patches.” In *COMPUTER GRAPHICS Forum*, Vol. 20, No. 3, pages C422–C430. Eurographics, Blackwell Publishers Ltd, 2001. EUROGRAPHICS’2001 Conference issue.

[119] S. H. M. Roth, M. H. Gross, S. Turello, and F. R. Carls. "A Bernstein-Bézier Based Approach to Soft Tissue Simulation." In *COMPUTER GRAPHICS Forum*, Vol. 17, No. 3, pages C285–C294. Eurographics, Blackwell Publishers Ltd, 1998. EUROGRAPHICS'98 Conference issue.

[120] H. Samet. *Applications of Spatial Data Structures*. Addison-Wesley, Reading, MA, 1990.

[121] H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, Reading, MA, 1990.

[122] P. H. Schönemann. "A Generalized Solution to the Orthogonal Procrustes Problem." *Psychometrika*, 31(1):1–10, 1966.

[123] W. J. Schroeder, J. A. Zarge, and W. E. Lorensen. "Decimation of Triangle Meshes." In *SIGGRAPH'92 Conference Proceedings*, Annual Conference Series, pages 65–70. ACM SIGGRAPH, Addison Wesley, 1992.

[124] G. H. Schut. "On Exact Linear Equations for the Computation of the Rotational Elements of Absolute Orientation." *Photogrammetria*, 16:34–37, 1960.

[125] H. R. Schwarz. *Methode der finiten Elemente*. B. G. Teubner, Stuttgart, 3rd edition, 1991.

[126] T. W. Sederberg and D. C. Anderson. "Ray Tracing of Steiner Patches." In *SIGGRAPH'84 Conference Proceedings*, Annual Conference Series, pages 159–164. ACM SIGGRAPH, Addison Wesley, 1984.

[127] T. W. Sederberg and S. R. Parry. "Free-Form Deformation of Solid Geometric Models." In *SIGGRAPH'86 Conference Proceedings*, Annual Conference Series, pages 151–160. ACM SIGGRAPH, Addison Wesley, 1986.

[128] H. Seidel. "A General Subdivision Theorem for Bézier Triangles." In T. Lyche and L. L. Schumaker, editors, *Mathematical Methods in Computer Aided Geometric Design*, pages 573–581. Academic Press, Inc., 1989.

[129] S. M. Seitz and C. R. Dyer. "Photorealistic Scene Reconstruction by Voxel Coloring." In *Proc. Computer Vision and Pattern Recognition Conf.*, pages 1067–1073, 1997.

[130] C. C. Slama, C. Theurer, and S. Hendrikson, editors. *Manual of Photogrammetry*. American Society of Photogrammetry, Falls Church, Virginia, 1980.

[131] T. C. Sprenger, R. Brunella, and M. H. Gross. "H-BLOB: A Hierarchical Visual Clustering Method Using Implicit Surfaces." In *Proceedings of IEEE Visualization 2000*, pages 61–68. IEEE, 2000.

[132] O. G. Staadt. *Multiresolution Representation and Compression of Surfaces and Volumes*. PhD thesis, Swiss Federal Institute of Technology (ETH), Zürich, Switzerland, 2001. No. 14013.

[133] O. G. Staadt and M. H. Gross. "Progressive Tetrahedralizations." In *Proceedings of IEEE Visualization 1998*, pages 397–402. IEEE, 1998.

[134] O. G. Staadt, M. H. Gross, and R. Weber. "Multiresolution Compression and Reconstruction." In *Proceedings of IEEE Visualization 1997*, pages 337–346. IEEE, 1997.

[135] I. Stakgold. *Boundary-Value Problems of Mathematical Physics*, volume I and II. Macmillan, New York, 1968.

[136] G. Strang and G. Fix. *An Analysis of the Finite Element Method*. Prentice Hall, Englewood Cliffs, New Jersey, 1973.

[137] W. Stürzlinger. "Ray Tracing Triangular Trimmed Free Form Surfaces." *IEEE Transactions on Visualization and Computer Graphics*, 4(3):202–214, 1998.

[138] M. Sweeney and R. Bartels. "Ray-Tracing Free-Form B-Spline Surfaces." *IEEE Computer Graphics and Applications*, 6(2):41–49, 1986.

[139] G. Székely, C. Brechbühler, R. Hutter, A. Rhomberg, N.Ironmonger, and P. Schmid. "Modelling of Soft Tissue Deformation for Laparoscopic Surgery Simulation." In W. M. Wells et al., editors, *Proceedings of MICCAI'98*, Lecture Notes in Computer Science, pages 550–561. Springer Verlag, 1998.

[140] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. "Elastically Deformable Models." In *SIGGRAPH'87 Conference Proceedings*, Annual Conference Series, pages 205–214. ACM SIGGRAPH, Addison Wesley, 1987.

[141] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. "Deformable Models." *The Visual Computer*, 4(6):306–331, 1988.

[142] D. Terzopoulos and K. Waters. "Physically-based Facial Modelling, Analysis, and Animation." *The Journal of Visualization and Computer Animation*, 1(2):73–80, Dec. 1990.

[143] M. Teschner. *Direct Computation of Soft-Tissue Deformation in Craniofacial Surgery Simulation*. PhD thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany, 2001. Shaker Verlag, Aachen, Germany, ISBN 3-8265-8317-5.

[144] P. Thévenaz, U. E. Ruttimann, , and M. Unser. "Iterative Multi-Scale Registration without Landmarks." In *Proc. IEEE Int. Conf. on Image Processing*, volume 3, pages 228–231. IEEE, Oct. 1995.

[145] P. Thévenaz and M. Unser. "A Pyramid Approach to Subpixel Registration Based on Intensity." *IEEE Transactions on Image Processing*, 7(1):27–41, Jan. 1998. http://big-www.epfl.ch/thevenaz/registration.

[146] J. F. Thompson, B. Soni, and N. P. Weatherrill. *Handbook of Grid Generation*. CRC Press, 1999.

[147] S. Timoshenko and J. N. Goodier. *Theory of Elasticity*. McGraw-Hill, New York, 3rd edition, 1970.

[148] D. L. Toth. "On Ray Tracing Parametric Surfaces." In *SIGGRAPH'85 Conference Proceedings*, Annual Conference Series, pages 171–179. ACM SIGGRAPH, Addison Wesley, 1985.

[149] M. Unser, A. Aldroubi, and C. Gerfen. "A Multiresolution Image Registration Procedure Using Spline Pyramids." In *Proceedings of SPIE, Wavelet Applications in Signal and Image Processing*, volume 2034, pages 160–170. International Society for Optical Engineering, SPIE, Nov. 1993.

[150] M. W. Vannier, J. L. Marsh, and J. O. Warren. “Three Dimensional Computer Graphics for Craniofacial Surgical Planning and Evaluation.” In *SIGGRAPH’83 Conference Proceedings*, Annual Conference Series, pages 263–273. ACM SIGGRAPH, Addison Wesley, 1983.

[151] T. Warnke, M. Path, C. P. Zollikofer, A. Zimmermann, F. Carls, P. Stucki, and H. Sailer. “Clinical Evaluation of a New Artifact Reduction Algorithm for CT Data Used for Building Stereolithography Models.” In *Proceedings of CARS’98*, Computer Assisted Radiology and Surgery, 12th International Congress and Exhibition. Elsevier Publishers, 1998.

[152] K. Washizu. *Variational Methods in Elasticity and Plasticity*. Pergamon Press, Elmsford, NY, 1975.

[153] K. Waters. “A Muscle Model for Animating Three-Dimensional Facial Expression.” In *SIGGRAPH’87 Conference Proceedings*, Annual Conference Series, pages 17–24. ACM SIGGRAPH, Addison Wesley, 1987.

[154] D. Werner. *Funktionalanalysis*. Springer Verlag, 2nd edition, 1997.

[155] N. Wilt. *Object-Oriented Ray Tracing*. Wiley, 1st edition, 1994.

[156] A. J. Worsey and G. Farin. “An n-Dimensional Clough-Tocher Interpolant.” *Constructive Approximation*, 3(2):99–110, 1987.

[157] X. Wu, M. S. Downes, T. Goktekin, and F. Tendick. “Adaptive Nonlinear Finite Elements for Deformable Body Simulation Using Dynamic Progressive Meshes.” In *COMPUTER GRAPHICS Forum*, Vol. 20, No. 3, pages C349–C358. Eurographics, Blackwell Publishers Ltd, 2001. EUROGRAPHICS’2001 Conference issue.

[158] J. Yen, S. Spach, M. Smith, and R. Pulleyblank. “Parallel Boxing in B-Spline Intersection.” *IEEE Computer Graphics and Applications*, 11(1):72–79, 1991.

[159] O. C. Zienkiewicz and R. L. Taylor. *The Finite Element Method: Solid and Fluid Mechanics, Dynamics and Non-Linearity*, volume 2. McGraw-Hill, 4th edition, 1991.

[160] O. C. Zienkiewicz and R. L. Taylor. *The Finite Element Method: Basic Formulation and Linear Problems*, volume 1. McGraw-Hill, 4th edition, 1994.

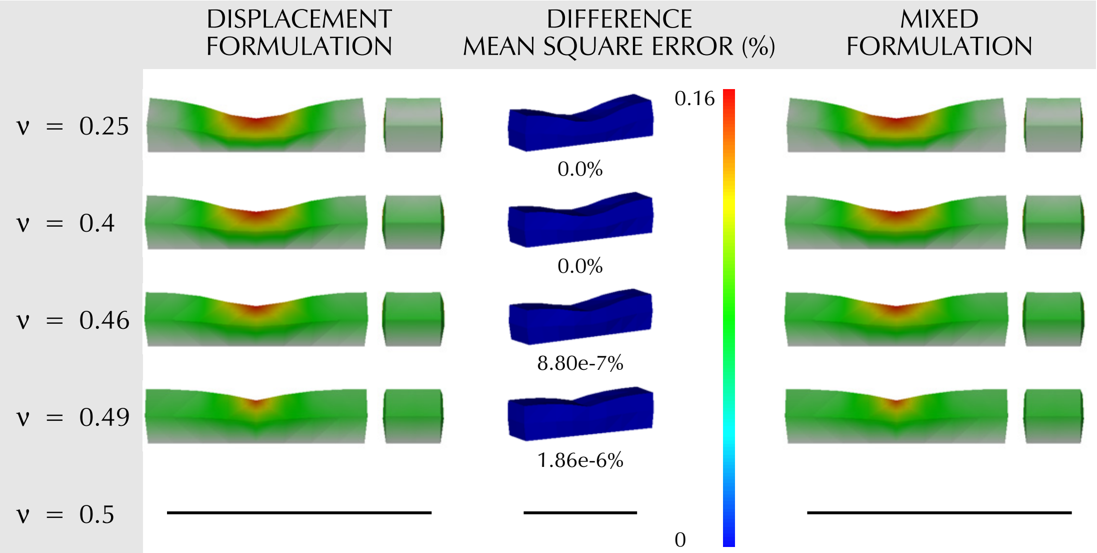
[161] G. W. Zumbusch. “Symmetric Hierarchical Polynomials for the h-p-Version of Finite Elements.” Technical Report SC-93-32, Konrad-Zuse-Zentrum, Berlin, Germany, 1993.

[162] G. W. Zumbusch. “Symmetric Hierarchical Polynomials and the Adaptive h-p-Version.” *Houston Journal of Mathematics*, pages 529–540, 1996. Proceedings of the Third International Conference on Spectral and High Order Methods, ICOSAHOM’95, also as report SC-95-18 ZIB, Berlin.

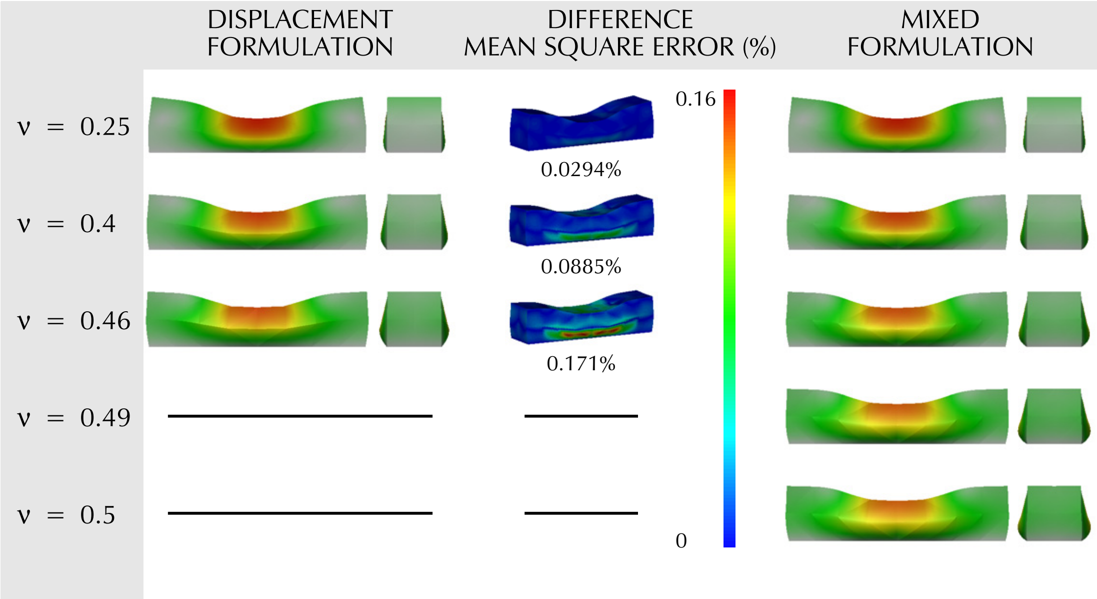
APPENDIX C

COLOR PLATES

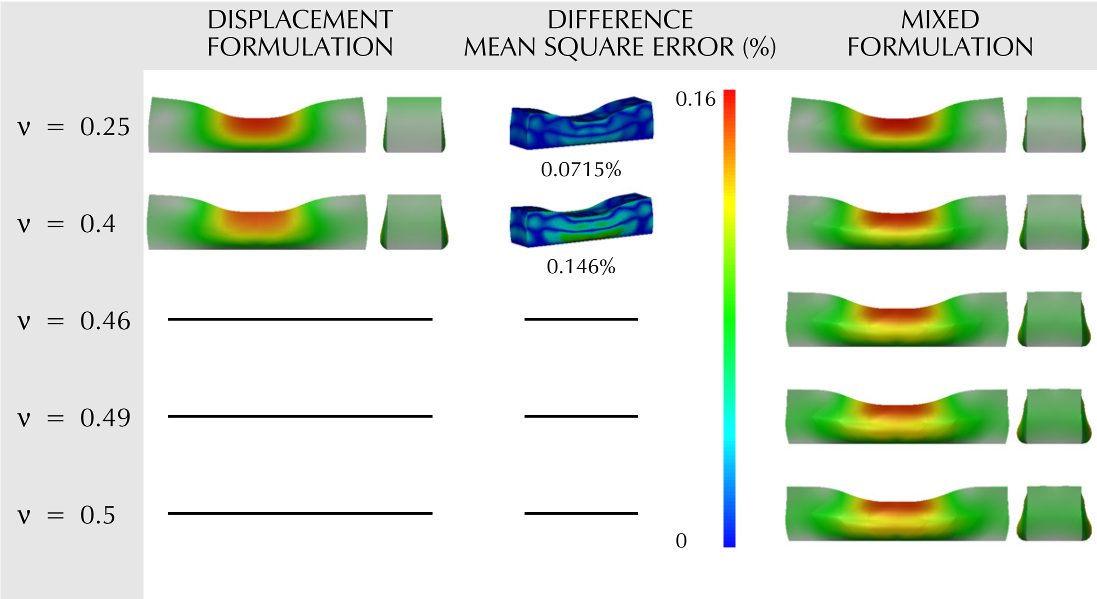
COLOR PLATE 1 Textured teapot. **[see Figure 6.14 on page 141]**



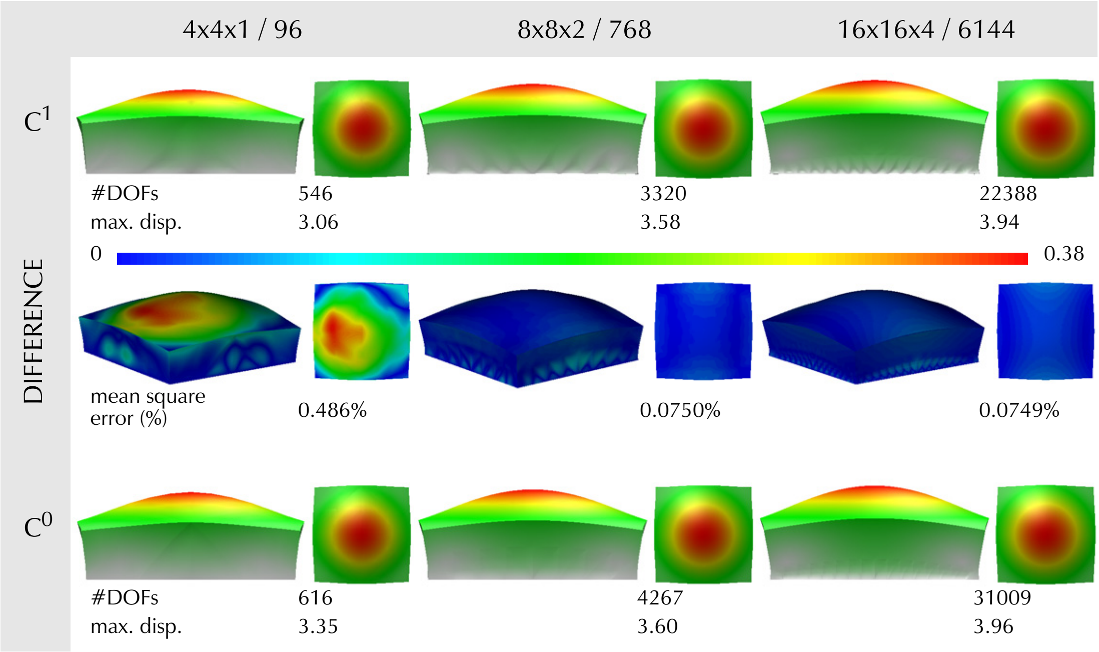
COLOR PLATE 2 Comparison of pure displacement and mixed formulation for decreasing compressibility by means of the linear C^0 element using the push configuration according to Figure 5.15. White color indicates zero displacement, red maximal displacement. **[see Figure 5.16 on page 118]**



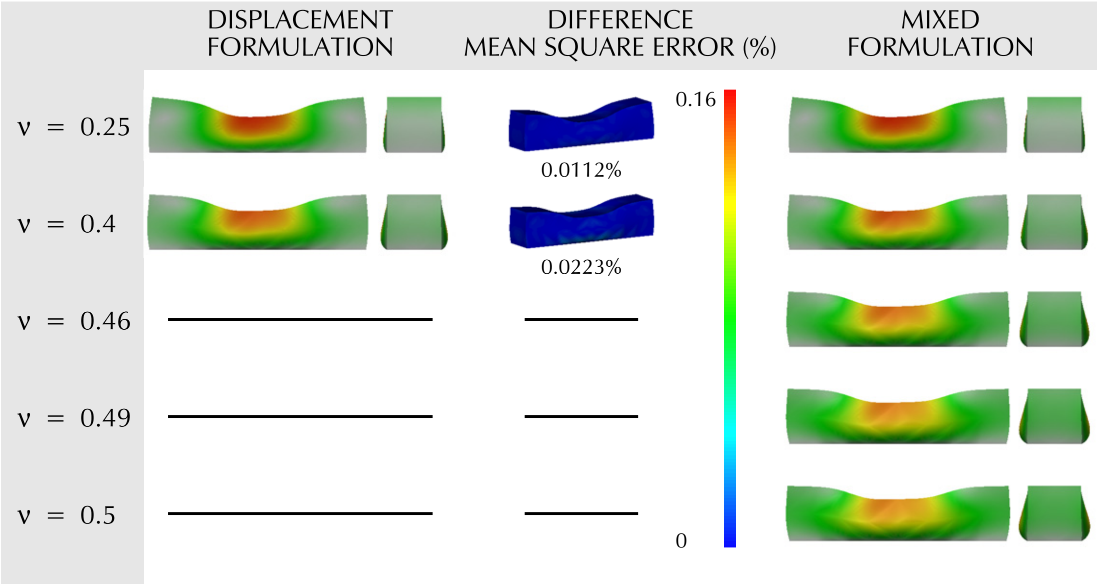
COLOR PLATE 3 Comparison of pure displacement and mixed formulation for decreasing compressibility by means of the quadratic C^0 element using the push configuration according to Figure 5.15. White color indicates zero displacement, red maximal displacement. **[see Figure 5.17 on page 119]**



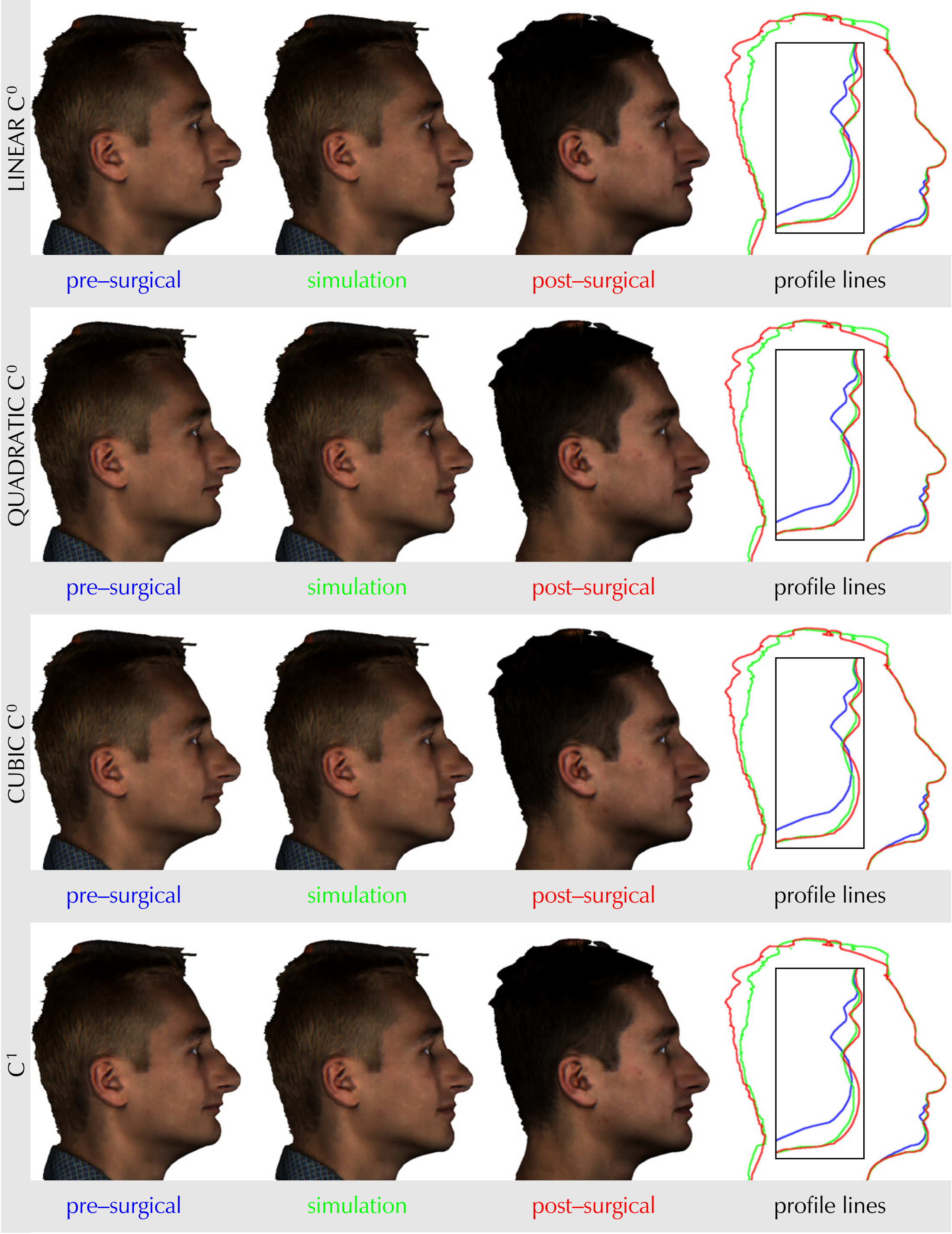
COLOR PLATE 4 Comparison of pure displacement and mixed formulation for decreasing compressibility by means of the cubic C^0 element using the push configuration according to Figure 5.15. White color indicates zero displacement, red maximal displacement. **[see Figure 5.18 on page 120]**



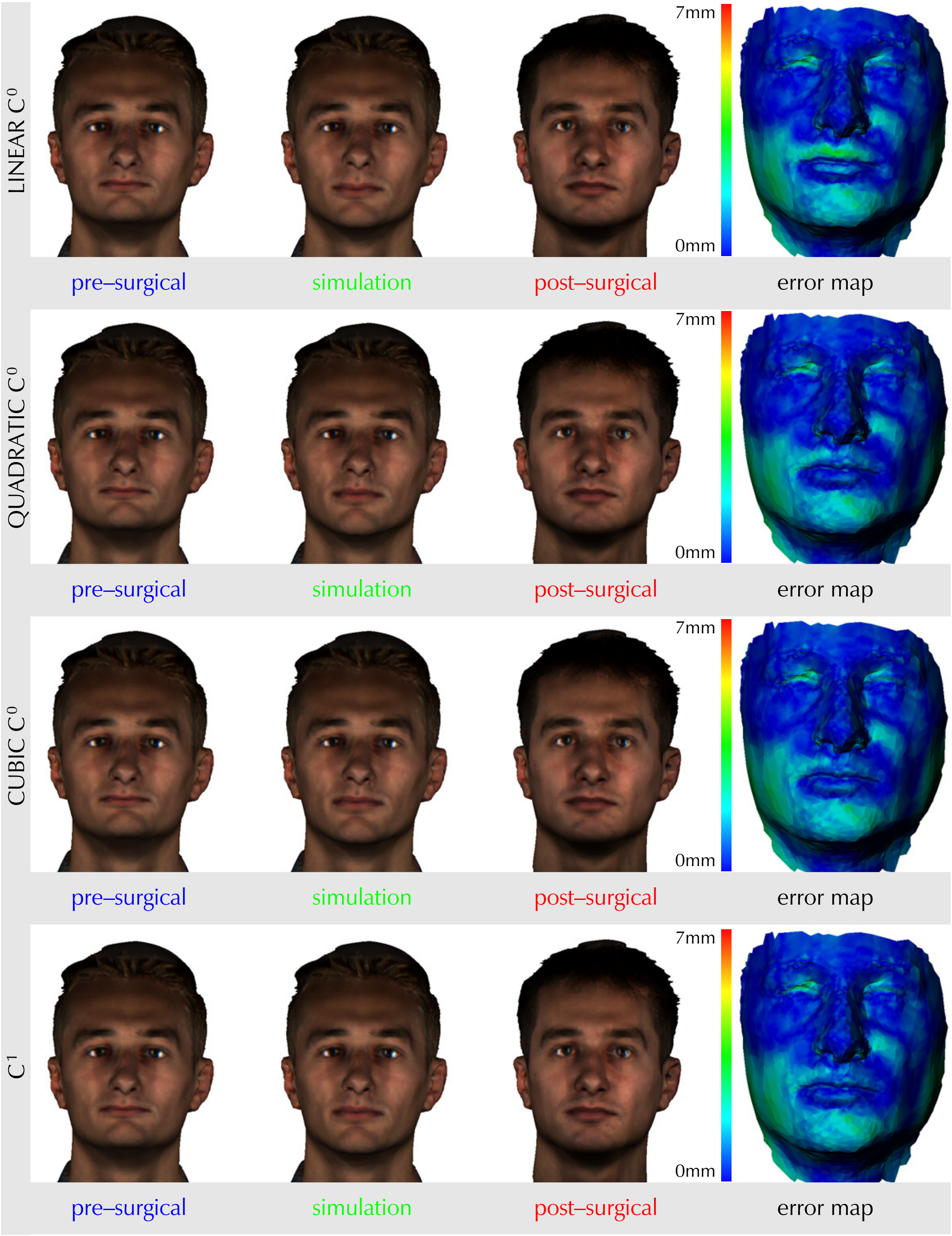
COLOR PLATE 5 C^1 test series compared against the cubic C^0 element using the same configuration as in Figure 5.14. White color indicates zero displacement, red maximal displacement.
[see Figure 5.19 on page 122]



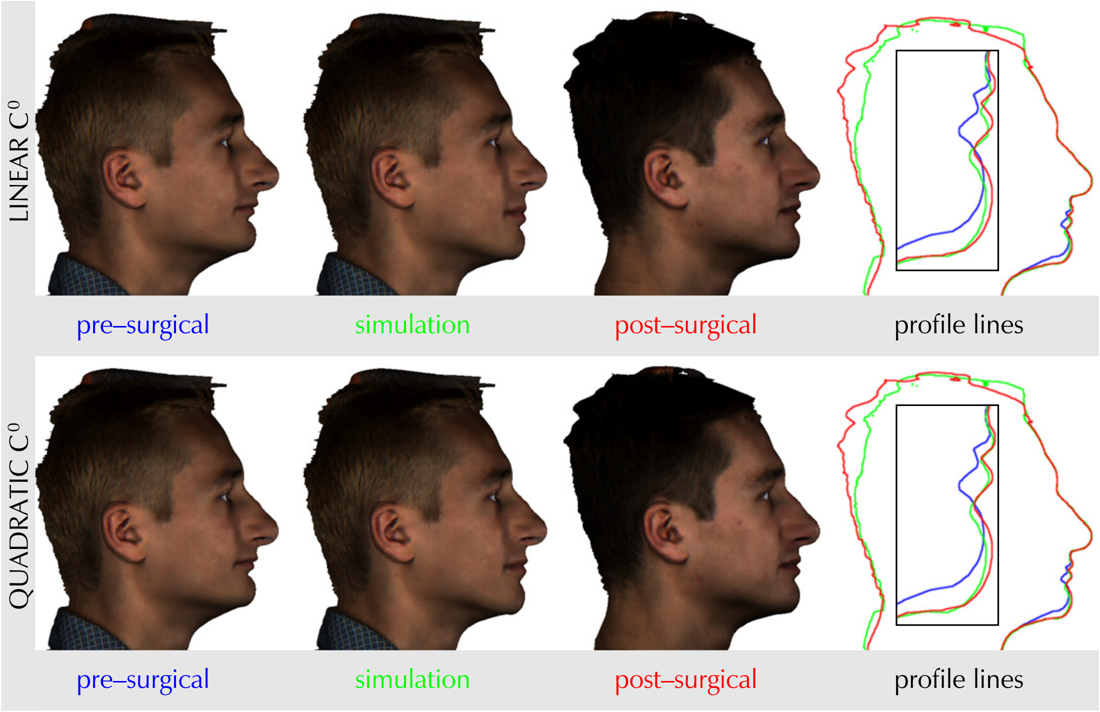
COLOR PLATE 6 Comparison of pure displacement and mixed formulation for decreasing compressibility by means of the C^1 element using the push configuration according to Figure 5.15. White color indicates zero displacement, red maximal displacement.
[see Figure 5.20 on page 125]



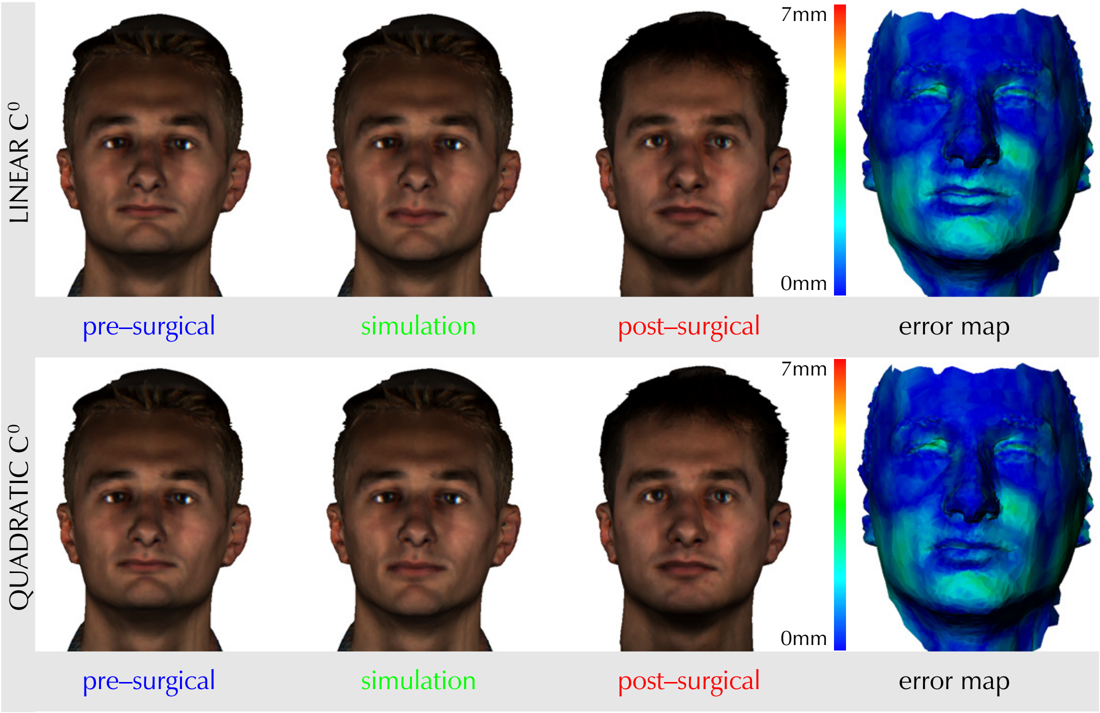
COLOR PLATE 7 Profile view comparison of low resolution simulation results: C^0 linear, quadratic and cubic element opposed to the C^1 element. **[see Figure 7.19 on page 171]**



COLOR PLATE 8 Frontal view comparison of low resolution simulation results: C^0 linear, quadratic and cubic element opposed to the C^1 element. **[see Figure 7.20 on page 172]**



COLOR PLATE 9 Profile view comparison of high resolution simulation results: C^0 linear and quadratic element. **[see Figure 7.21 on page 174]**



COLOR PLATE 10 Frontal view comparison of high resolution simulation results: C^0 linear and quadratic element. **[see Figure 7.22 on page 175]**

APPENDIX D

CURRICULUM VITAE

S. H. Martin Roth

April 1, 1969	Born in Berne, Switzerland, citizen of Niederbipp, Berne, Switzerland, son of Hans and Ursula Roth–Küng
1988	Matura, Gymnasium Köniz, Berne, Switzerland (Swiss high school examination)
1995	Diploma in Computer Science as Dipl. Informatik-Ing. ETH, Swiss Federal Institute of Technology, ETH, Zurich, Switzerland
1995 – 2001	Research and teaching assistant, Computer graphics laboratory, headed by Prof. M. Gross, Institute of Scientific Computing, Swiss Federal Institute of Technology, ETH, Zurich, Switzerland