An Efficient Active B-Spline/NURBS Model for Virtual Sculpting

Patricia Moore

Bachelor of Engineering in Electronic Engineering

September 2013

School of Electronic Engineering
Faculty of Engineering and Computing
Dublin City University

Supervised by Dr. Derek Molloy

This dissertation is submitted for the degree of Doctor of Philosophy

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Doctor of Philosophy is entirely my own work, that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed: _____ ID No.: _____ Date: _____

Abstract

This thesis presents an Efficient Active B-Spline/Nurbs Model for Virtual Sculpting. In spite of the on-going rapid development of computer graphics and computer-aided design tools, 3D graphics designers still rely on non-intuitive modelling procedures for the creation and manipulation of *freeform* virtual content. The 'Virtual Sculpting' paradigm is a well-established mechanism for shielding designers from the complex mathematics that underpin freeform shape design. The premise is to emulate familiar elements of traditional clay sculpting within the virtual design environment. Purely geometric techniques can mimic some physical properties. More exact energy-based approaches struggle to do so at interactive rates. This thesis establishes a unified approach for the representation of physically aware, energy-based, deformable models, across the domains of Computer Graphics, Computer-Aided Design and Computer Vision, and formalises the theoretical relationships between them. A novel reformulation of the computer vision approach of Active Contour Models (ACMs) is proposed for the domain of Virtual Sculpting. The proposed ACM-based model offers novel interaction behaviours and captures a compromise between purely geometric and more exact energy-based approaches, facilitating physically plausible results at interactive rates. Predefined shape primitives provide features of interest, acting like sculpting tools such that the overall deformation of an Active Surface Model is analogous to traditional clay modelling. The thesis develops a custom-approach to provide full support for B-Splines, the *de facto* standard industry representation of freeform surfaces, which have not previously benefited from the seamless embodiment of a true Virtual Sculpting metaphor. A novel generalised computationally efficient mathematical framework for the energy minimisation of an Active B-Spline Surface is established. The resulting algorithm is shown to significantly reduce computation times and has broader applications across the domains of Computer-Aided Design, Computer Graphics, and Computer Vision. A prototype 'Virtual Sculpting' environment encapsulating each of the outlined approaches is presented that demonstrates their effectiveness towards addressing the long-standing need for a computationally efficient and intuitive solution to the problem of interactive computer-based freeform shape design.

Acknowledgements

I wish to express my sincere gratitude to my supervisor Dr. Derek Molloy for his enthusiasm, support and guidance over the course of this research project. I would also like to thank Dr. Noel Murphy for his support and encouragement during this research. I have benefitted from many discussions with my friends and colleagues in the Vision Systems Group, who have always been eager to help out and share their knowledge. I thank them for their constructive input. I would like to extend my thanks to the staff of Dublin City University, particularly those in the School of Electronic Engineering, for their positive attitudes and hard work behind the scenes. I would like to take the opportunity to thank my family and friends outside of Dublin City University for their constant encouragement and guidance. Finally, I would also like to acknowledge the Irish Research Council for Science, Engineering and Technology for the financial support that made this research possible.

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

ACM Active Contour Model

API Application Programming Interface

BEM Boundary Element Method

CAD Computer-Aided Design

CAE Computer-Aided Engineering

CAGD Computer-Aided Geometric Design

CAM Computer-Aided Manufacturing

CSG Constructive Solid Geometry

CT Computerised Tomography

D-NURBS Dynamic Non-Uniform Rational B-Splines

EFFD Extended Free Form Deformation

FDM Finite Difference Method

FEM Finite Element Method

FFD Free Form Deformation

FVM Finite Volume Method

GVF Gradient Vector Flow

IgA Iso-geometric Analysis

MRI Magnetic Resonance Imaging

MSD Mass-Spring-Damper

MVC Model-View-Controller

NURBS Non-Uniform Rational B-Splines

PDE Partial Differential Equation

Introduction 1

Over the past five decades, the computer has become a seemingly indispensable modelling tool. Indeed, it is difficult to think of an area in which computer-based modelling cannot be used to some advantage. The extent to which we can make use of computer-based models is highly dependent on our ability to create and manipulate computerised representations of objects and phenomena of interest. For this reason, providing intuitive design metaphors and accompanying tools for computer-based shape modelling and deformation continues to be a key challenge across research communities in design, e.g., Computer-Aided Design (CAD), Computer-Aided Geometric Design (CAGD), Computer-Aided Manufacturing (CAM), Computer-Aided Engineering (CAE), and also in other disciplines such as Computer Graphics, Computer Vision, Visualisation and many more.

This thesis is primarily concerned with the interactive design of *freeform* surfaces. Freeform surface design encompasses those techniques for the design of smoothly varying surfaces, characterised by strict requirements on surface quality and aesthetics, with looser tolerances and dimensioning constraints than more functional forms of design. Initially developed for automotive and aerospace industries, these surfaces have become increasingly important in surface design. Freeform surfaces are now the standard in CAD for surface design and are used across various domains for creating purely aesthetic surfaces for artistic purposes, for creating aesthetic surfaces that also perform a function, and even for highly technical surfaces. Their application can be seen in areas as diverse as prototyping, engineering design, consumer product design, computer graphics, animations, special effects, medical visualisation, architecture, sculpture, etc.

Traditionally, modelling and graphics have been regarded as two related but quite separate fields of study. Modelling encompasses those geometric techniques for describing object shapes by means of mathematical and abstract relationships suitable for processing by a computer, while graphics has been seen to cover those techniques for handling the visual display and manipulation of such computer-based models. In recent years, computer-based modelling has become a largely interactive process, relying on computer graphics techniques for defining geometric specifications. For freeform modelling, where the aesthetics of the final shape are paramount, the 'Virtual Sculpting' paradigm has long been hailed as a natural and intuitive design metaphor. By emulating traditional clay sculpting in an interactive environment, the task of ma-

nipulating complex mathematical models can be hidden behind the familiar[1] physical action of moulding and manipulating inelastic substances such as clay.

The ability to intuitively create is core to supporting the design process and intuitive design tools should shield designers from complicated underlying mathematics and physics, removing the need for user expertise. Since its inception (Parent, 1977), Virtual Sculpting has been hailed as a means of doing just that. This thesis examines employing the Virtual Sculpting paradigm for intuitive interactive design of freeform surfaces.

1.1 Challenges in Freeform Surface Design and Analysis

There are two main processes involved in freeform surface design. Firstly, the creative design process, whereby aesthetics are the primary consideration and form is favoured over function. Secondly, for many applications, freeform surfaces must subsequently undergo physical analysis, whereby the physical behaviour of the surface is simulated and/or analysed, e.g., for production/manufacture or graphics/animation, etc. While largely separate, the two processes are highly inter-dependent and typically the overall design requires iterative cycles of the two processes until the desired design and behaviours are simultaneously achieved. For Virtual Sculpting, the creative design process incorporates analysis techniques. It is therefore necessary to consider both design and analysis simultaneously. This section outlines the main challenges facing both the design and analysis processes in current CAD workflows.

1.1.1 Challenges in Design

With sophisticated software facilitating the visual display of complex graphic content, and the wealth of geometric and deformable modelling techniques at our disposal, it might be reasonable to think that the creation of virtual content could be a straightforward process. However, where automation of content capture is not possible, describing complex objects in terms of geometric relationships, even for skilled designers, can be a daunting task.

Shapes can be specified in many ways, but a technique that has become popular in CAD is using a control mesh, specified by a series of connected control points, to govern a smoothly varying surface that approximates the mesh. For freeform surfaces designed in this way, B-Splines/Non-Uniform Rational B-Splines (NURBS) are the *de facto* standard representation in CAD. Such surfaces are built upon complex underlying mathematics such that traditional methods of generating and deforming such models require prerequisite knowledge or experience in order to utilise design tools effectively. Many computer graphics modelling and CAD environments require skilled labour and large time investments on the part of the designer, as current modelling tools regularly

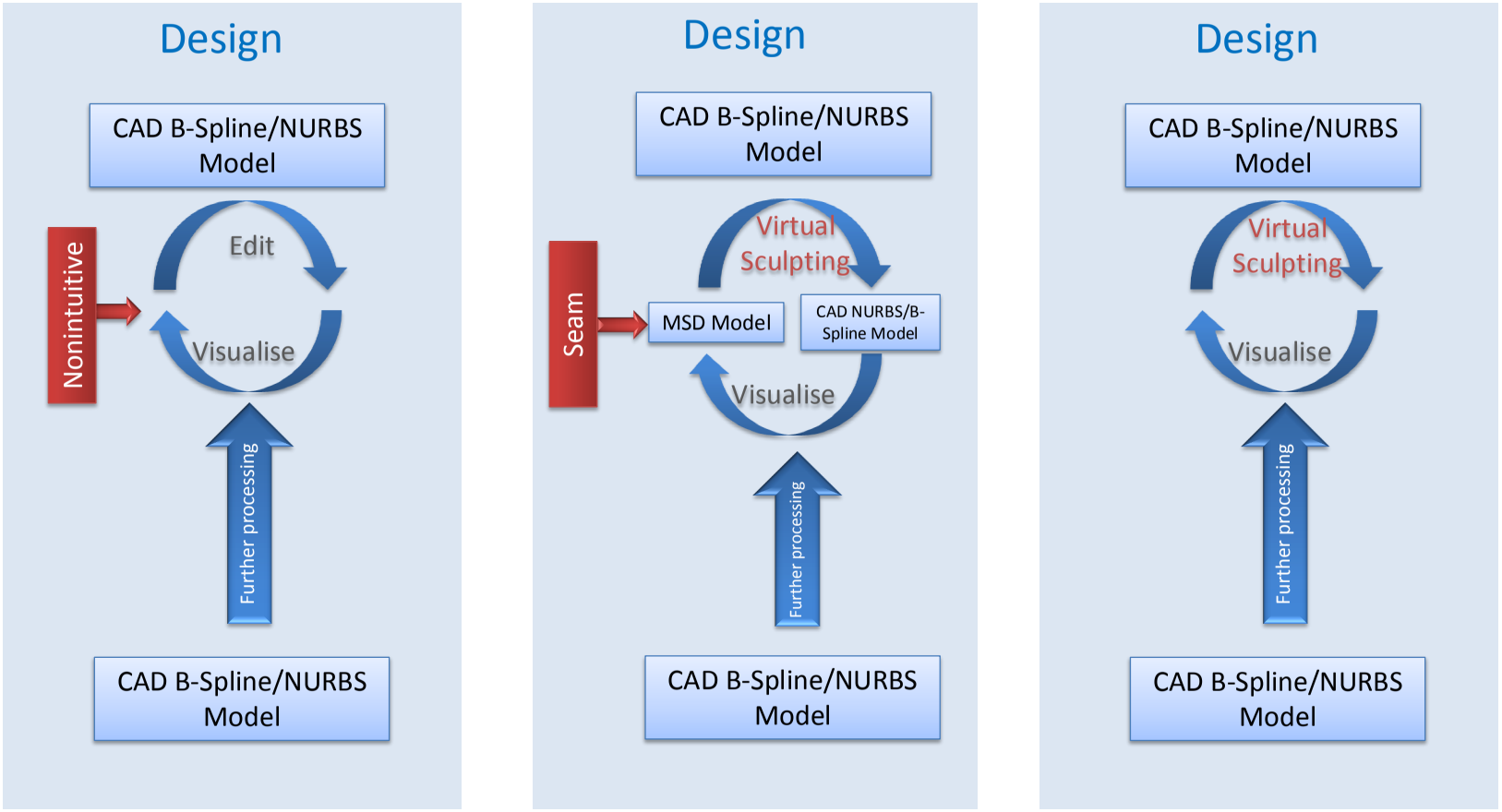
[1] Most people are introduced to physical clay modelling from an early age in the form of Play-Doh or plasticine

require the manual manipulation of numerous control points via a keyboard/mouse while monitoring modifications on a 2D visual display (Knopf and Igwe, 2005). While a competent designer can transfer their ideas onto paper within minutes, conveying the same idea via a CAD system may take hours, days or even weeks (Piegl, 2005). Figure 1.1(a) illustrates the traditional design pipeline used in current CAD environments.

Virtual Sculpting environments generally rely on combinations of various shape modelling and deformation techniques. Some physical deformation properties can be mimicked using purely geometric approaches. However, such approaches do not provide a mechanism for simulating the real mechanics of a physical deformation. More intuitive energy minimisation approaches incorporate the principles of continuum mechanics and account for the material properties of the objects being deformed, but struggle to do so at interactive rates. Several attempts have been made to incorporate Virtual Sculpting techniques within CAD environments. However the techniques proposed to date in the literature employ multiple disparate representation technologies (Dachille IX et al., 2001, Gao and Gibson, 2006, Pungotra et al., 2010). The introduction of disparate representation technologies constitutes what shall be referred to in the thesis as a 'seam' in the design pipeline. Conversions between representations at these seams typically create bottlenecks. Additionally, the alternative representations employed are only approximations of the CAD geometry, which shall be referred to in the thesis as 'exact'. Approaches to date do not facilitate a 'seamless' integration. Accordingly, there is little evidence to suggest that industry is adopting the representations. Figure 1.1(b) illustrates the current situation in the literature with regard to Virtual Sculpting of B-Spline/NURBS freeform surfaces in CAD.

Although computer graphics and CAD tools have evolved rapidly in recent years, the task of computer-based modelling is still beset with practical and conceptual usability difficulties (Gain and Marais, 2005), to such an extent that 3D graphics designers still rely on many non-intuitive modelling procedures for the creation of complex content (Knopf and Igwe, 2005). In spite of more than thirty years of research in the area of Virtual Sculpting, many designers still choose to create 3D content by digitising a preliminary clay maquette created in the physical world rather than the virtual world. In *Toy Story*, the movie touted as the first feature film entirely created by computer animation, NURBS representations were used and several of the more detailed characters were created in physical clay before digitisation (Massie, 1998).

A novel interactive approach built upon the Virtual Sculpting paradigm would facilitate intuitive design, free from reliance on prerequisite knowledge. In order to achieve this, the analysis pipeline must also be examined such that elements of the analysis pipeline can be incorporated within the design process. In order for industry to consider adopting Virtual Sculpting techniques, a seamless integration must be sought. Figure 1.1(c) shows a desirable workflow that would not only have the potential to greatly reduce the time investments currently made by designers, but may also open up the domain of 3D computer graphics design to the lay person.



(a) Traditional Nonintuitive Design Pipeline (b) Existing Virtual Sculpting (c) Seamless Virtual Sculpting

Figure 1.1: The Design Pipeline: Non-intuitive modelling procedures and divisions in representation technologies hinder the design process

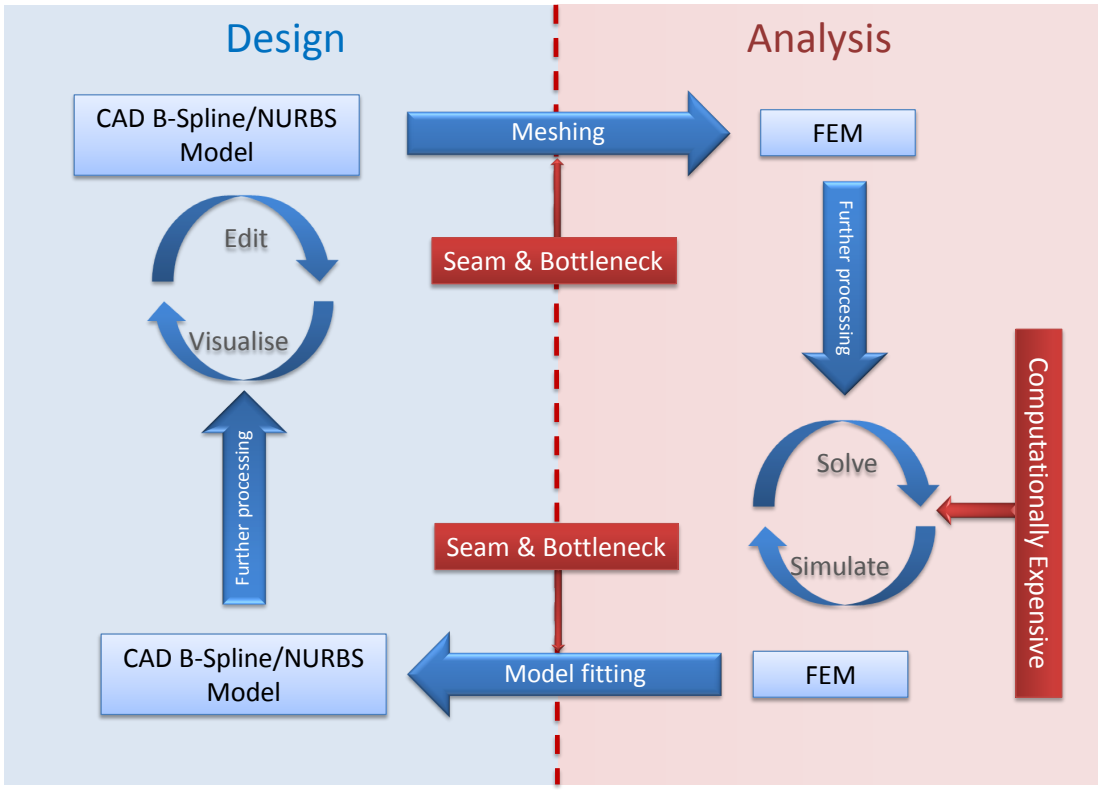
1.1.2 Challenges in Analysis

Since the advent of CAD, the design process and the physical analysis process have been largely separate. This has been mainly due to the divide in representation technologies. It is difficult to design in polygon representations, but these representations have been required for the analysis process where finite element meshes are needed by the Partial Differential Equation (PDE) solvers. Designers generate CAD files, typically using B-Spline/NURBS representations, and these must be translated into analysis-suitable Finite Element Method (FEM) geometries. This situation is depicted in Figure 1.2(a). The conversion between the two representations at this point introduces an additional seam in the design and analysis pipeline, and as discussed in Section 1.1.1, such a seam not only causes a large bottleneck, but also results in an approximation of the exact CAD geometry. The time spent on such conversions can constitute up to 80% of the total analysis time (Cottrell et al., 2009).

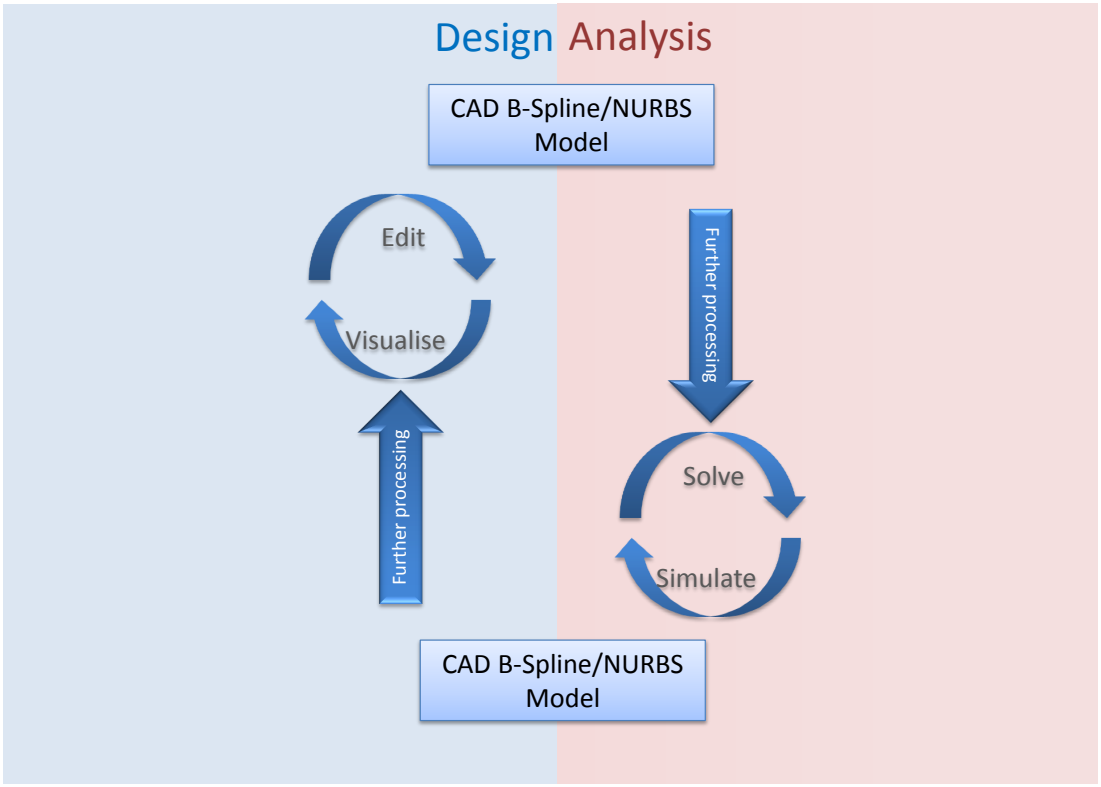
While numeric approaches are generic and therefore universally applicable, analytic solutions, where possible, can be significantly more efficient. Additionally, analytic solutions are generally superior for processes like manufacturing that place higher demands on geometric precision. When a surface is stored via an analytic representation instead of discrete geometry, an 'exact' mathematical definition is available. This makes accurate analyses possible. On the other hand, meshes, built from discrete data, are inherently disconnected from the CAD geometries from which they were created. Poor design decisions can be made as a result of analyses of discrete representations. Unless the designer/analyst has tight control over the discrete geometry's resolution, the discrete model may not accurately represent the original model. It is essential that designers/analysts ensure adequate resolution is achieved such that important topological information is preserved.

Recent research by Hughes et al. (2005), Cottrell et al. (2009), Hughes et al. (2010), and Auricchio et al. (2012) has advocated employing a CAD B-Spline/NURBS model right through the CAD pipeline from design to analysis. Iso-geometric Analysis (IgA) has provided a mechanism for maintaining a parametric representation for the analysis process. Although IgA is nonstandard in industry, there is growing evidence that it is making headway on the more established FEM techniques. While this removes the unnecessary bottlenecks in the pipeline (see Figure 1.2(b)), numerical methods are still employed to ultimately solve the system. The technique is computationally expensive as a result.

Figure 1.3(a) depicts the traditional analysis pipeline that adopts discrete FEM representations for numerical analyses. Figure 1.3(b) depicts the incorporation of IgA in the analysis pipeline. This facilitates the preservation of the original CAD representation. However, a computationally expensive numeric solver is ultimately used to perform the analyses. Finally, Figure 1.3(c) illustrates a more desirable situation that preserves the CAD representation throughout the analysis process.



(a) Traditional CAD Pipeline



(b) Seamless CAD Pipeline

Figure 1.2: Traditional CAD Pipeline vs Seamless CAD Pipeline: Preserving the CAD geometry removes bottlenecks in the CAD process and brings the Design and Analysis processes closer together

(a) Traditional Analysis Pipeline (b) Existing incorporation of IgA (c) Efficient Analytic Solution

Figure 1.3: The Analysis Pipline: Numeric solvers are computationally expensive

1.1.3 Historical Context

In order to address the issues discussed in Section 1.1.1 and 1.1.2, an appreciation of how the current state of the art developed is useful. Figure 1.4 depicts a summary of the key developments that underpin the research conducted in this thesis. Each of these developments and the subsequent research they inspired will be addressed in greater detail, where appropriate, throughout the thesis. Its presence at this point is to illustrate the development of the state of the art discussed in this chapter, and also to further introduce a context for the research presented in this thesis.

From Figure 1.4, it can be seen that FEM techniques were technically mature (Strang and Fix, 1973) before the fields of CAD and CAGD were even established (Cottrell et al., 2009), let alone before B-Spline/NURBS representations were standardised (Piegl and Tiller, 1997). The foundations of a Virtual Sculpting paradigm can be traced back to shortly before this standardisation occurred (Parent, 1977).

Physically aware Deformable Models began to develop shortly after. Figure 1.4 also shows that research efforts were conducted concurrently in various and largely disconnected research communities, e.g., Elastically Deformable models (Terzopoulos et al., 1987) in Computer Graphics, Active Contour Models (Snakes) (Kass et al., 1987) and B-Snakes (Menet et al., 1990) in Computer Vision and Visualisation, and Dynamic Nurbs (Terzopoulos and Qin, 1994) in Computer Graphics and CAD/CAGD.

Research activity in B-Spline/NURBS arguably underwent a decline in the late 1990s and early 2000s. The status quo was largely accepted in CAD and CAGD, where they were standard, while other domains less tied to B-Spline/NURBS representations focused on exploring alternatives.

In 2005, Hughes et al. (2005) suggested that FEM meshing of CAD/CAGD representations was unnecessary and that the analysis process should be adapted to work with the original CAD/CAGD representations, and coined the term IgA. Within a few years, it was evident that this research had triggered a new research drive in CAD/CAGD representations. Another key development came from the Computer Graphics domain, where Cashman (2010) successfully developed a unified model for B-Spline/NURBS models and their main competitor in Computer Graphics, Subdivision Surfaces.

With this research has come a renewed interest in facilitating a Virtual Sculpting paradigm in CAD/CAGD (Gao and Gibson, 2006, Pungotra et al., 2010). The Virtual Sculpting techniques suggested to date for CAD/CAGD representations have yet to embrace the idea of IgA. This is perhaps due to the computational overhead currently associated with IgA Analysis. Recent research has begun to explore analytic solutions to reduce this overhead (González-Hidalgo et al., 2013).

Figure 1.4: Timeline of significant events relating to this thesis

1.1.4 Discussion

As outlined in Section 1.1.1 and 1.1.2, the CAD design and analysis pipelines each come with their own specific set of challenges. In order to improve the intuitiveness of design process, the Virtual Sculpting paradigm may be employed. However, to achieve this, elements of the analysis process must be incorporated in the design process. The conversions between design and analysis representations in conjunction with the numerical techniques employed by the analysis process have thus far limited the utility of analysis in design. In the Automotive Industry a mesh for an entire vehicle takes approximately 4 months to create. Design decisions, in contrast, can be made on a daily basis (Cottrell et al., 2009). Additionally, once a mesh has been generated, refinement requires a link back to the CAD representation. This situation is highly undesirable. Cottrell et al. (2009) state that "The benefits of design optimization have been largely unavailable to industry. The bottleneck is that to do shape optimization the CAD geometry-to-mesh mapping needs to be automatic, differentiable, and tightly integrated with the solver and optimizer. This is simply not the case as meshes are disconnected from the CAD geometries from which they were generated". To achieve a Virtual Sculpting paradigm, each of these challenges must be met simultaneously.

1.2 Thesis

This thesis aims to address each of the challenges discussed in Section 1.1. The overall objective is depicted by Figure 1.5 that combines each of the presented desirable situations. This section presents a summary of the approaches taken.

Traditional Active Contour Models (ACMs), used extensively in computer vision, are energy minimising curves defined within the domain of an image that can deform to fit image features by latching onto potential minima generated by the appropriate processing of the image. Such models are physically motivated and approximate the physical properties of real materials, offering a compromise between purely geometric and more exact energy-based approaches. This thesis proposes a novel reformulation of ACMs for the domain of Virtual Sculpting. In the proposed technique, ACMs are employed within a Virtual Sculpting environment where predefined shape primitives provide the features of interest. These primitives act as sculpting tools, providing visual guides facilitating intuitive deformation results, such that the overall approach is analogous to traditional clay modelling. The compromise offered by an ACM-based approach facilitates physically-plausible results at interactive rates. An additional novelty of the proposed ACM-based approach is that it facilitates the design of proximity-based tools that offer different interaction behaviours to those offered by collision based sculpting techniques.

B-Spline/NURBS representations have become the *de facto* standard in industry for the representation of freeform surfaces. Where local control properties, flexibility and compact representation are desirable, B-Spline/NURBS are a natural choice of surface representation. However, these representations have not yet fully benefited from the

seamless embodiment of a true Virtual Sculpting paradigm. This thesis addresses this issue, and develops a customised ACM-based approach providing full support for B-Spline/NURBS representations. The analytic continuous mathematical description of B-Splines/NURBS makes them a particularly suitable choice as the underlying representation for the proposed ACM-based approach. The interactive Virtual Sculpting of Active B-Spline/NURBS Surfaces enabled by the customised approach offers a more intuitive alternative to complex and tedious manual control point manipulations and also facilitates data exchange between existing CAD systems.

The key to achieving the desired objectives, using the above approaches, is an analytic solver that can be used in both the design and analysis processes. The preservation of the model would remove the 80% bottleneck while the inherently tailored analytic solution would make the remaining 20% analysis time faster. Additionally, an analytic solution would facilitate the preservation of the "exact" description of the geometry. This would simultaneously remove the need for costly conversions of representation and the resulting approximations they introduce. Such a solution would offer a "seamless" design/analysis pipeline, facilitating accurate interactive design and both faster and more accurate analyses. Additionally, such a system would integrate directly with existing CAD systems.

The direct implementation of an Active B-Spline/NURBS Surface requires minimising its associated energy function. In this thesis, existing approaches to dealing with the energy minimisation of an Active B-Spline/NURBS Surface are examined. The complexity of B-Spline/NURBS mathematics has led to a preference for generic numeric techniques. The analysis conducted in this thesis leads to the development of a novel efficient mathematical framework based on an analytic evaluation of the associated stiffness matrix. This framework forms the basis of an efficient algorithm which is shown to reduce significantly the computational complexity, and thus computation time, of the stiffness matrix evaluation. The effectiveness of the proposed approach for the calculation of each individual energy term making up the stiffness matrix is demonstrated through a metric comparison with existing approaches. The approach is shown to fully extend to the remaining elements of the energy equation such that the force-balance equations can be solved analytically.

The mathematical framework developed not only enhances the interactive Virtual Sculpting approach presented in this thesis, but also has broader applications across the domains of Computer Vision and Computer Graphics.

A prototype Virtual Sculpting environment encapsulating each of the outlined approaches is presented that demonstrates their effectiveness towards addressing the long-standing need for an efficient and intuitive solution to the problem of interactive computer-based freeform shape design.

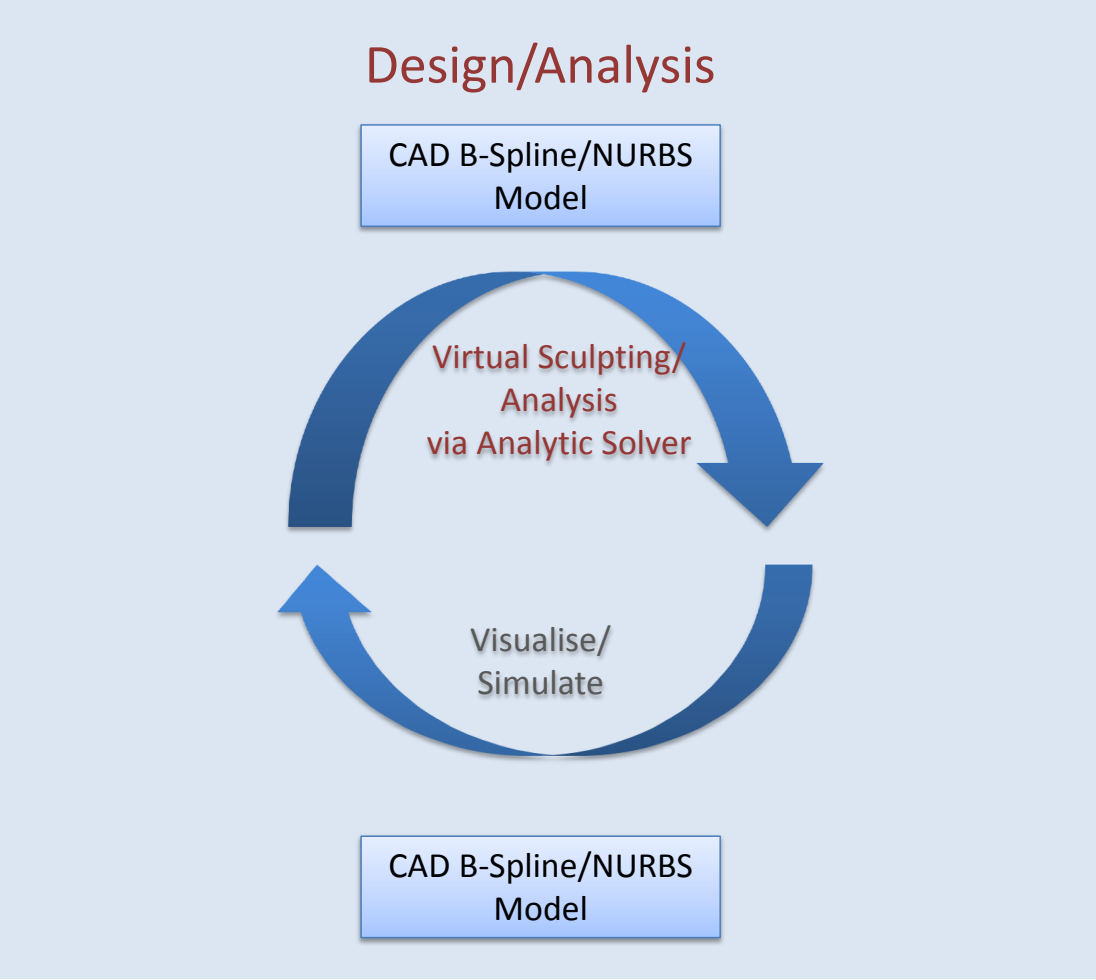


Figure 1.5: Proposed Seamless Virtual Sculpting Approach that preserves the analytic representation throughout the design and analysis processes

1.3 Contributions

This section details the contributions presented in this thesis. The section begins with a summary statement outlining the overall contribution and a list of core contributions. Details of each individual contribution are then provided, aligned to the outline structure of the document.

1.3.1 Contribution Summary

The immediate contribution of this work is the overall method that provides, for the first time, a seamless Virtual Sculpting design and analysis framework for freeform surfaces, that is fully compatible with existing CAD frameworks. The innovative Analytic solver developed facilitates the preservation of the model integrity throughout the design and analysis processes such that the 80% bottleneck in the traditional CAD pipeline is removed and the subsequent analyses are "exact". Additionally, the solver is shown to reduce the computational complexity of the analysis such that it can handle

physics-based deformations of the analytic surface up to 4 times faster than the current state of the art numeric solver. This effectively makes the remaining 20% of the analysis process 4 times faster. This solver not only allows the incorporation of the analysis process in the design process, thus facilitating Virtual Sculpting for intuitive design, but can also readily replace current numeric solvers for the analysis process.

The core contributions of the thesis are:

- A comprehensive cross-disciplinary Literature Review.
- A unified approach for the representation of cross-domain Deformable Models.
- A novel ACM based approach to Virtual Sculpting of Active B-Spline/NURBS Surface Models.
- A novel efficient mathematical framework for the seamless and exact analytic physical analysis of Active B-Spline/NURBS models.

Secondary contributions of the thesis are:

- An Efficient Gaussian Quadrature approach tailored for analysis of Active B-Spline/NURBS models, with accompanying algorithms.
- An Efficient Analytic approach for analysis of Active B-Spline/NURBS models, with accompanying algorithms.
- Full treatment of the Integral of a B-Spline basis function with accompanying algorithms.
- Augmented treatment of the Derivative of a B-Spline basis function with accompanying algorithms.
- Novel physical and graphical interpretations of the underlying mechanics of the analytic solution.
- The extension of the Analytic Approach to handling varying material properties.
- The extension of the Analytic Approach to handling varying mass, damping and forcing functions.
- The extension of the Analytic Approach to the important case of NURBS.
- Comprehensive theoretical and experimental stress testing and evaluation of the algorithms developed.
- A Prototype Virtual Sculpting Environment.
- A Java 3D Library for Active B-Spline/NURBS.

A list of publications associated with this research can be found in Chapter 7 Section 7.2.

1.3.2 Contribution Detail and Thesis Outline

The following is a breakdown of contributions presented in this thesis that correspond to the main thesis chapters:

- Due to the multidisciplinary nature of freeform shape design and analysis, there is a vast amount of literature in the area. This has served to confuse and overwhelm general practitioners. **In this thesis, the literature review clarifies the various available methods relating to shape representation, deformation, and the various combinations of the two**, such that practitioners may make appropriate decisions regarding the recited techniques by providing a systematic description of their assumptions and setup in each case. No other work was found in the literature that considers the full spectrum of techniques across different domains. (Chapter 2)

- The use of techniques based on energy minimisation for the implementation of physically aware deformable models is widespread across the domains of Computer Vision, Computer Graphics, and CAD. However, there is no common thread that generalises the disparate approaches for use across the different domains. **In this thesis, a unified approach for the representation of such models is developed, and the theoretical relationships between the different models are formalised**. It is also demonstrated that mathematical similarities between seemingly different models can be introduced as a consequence of the solution employed to perform the energy minimisation. (Chapter 3)

- **A novel ACM based approach to Virtual Sculpting is developed using Active B-Spline/NURBS Surface models**. In the proposed technique, predefined shape primitives provide the features of interest. These primitives act as sculpting tools, providing visual guides that facilitate intuitive deformation results, such that the overall approach is analogous to traditional clay modelling. The main benefit of the proposed technique is that it facilitates the seamless integration of a Virtual Sculpting methodology in a CAD environment. The Virtual Sculpting of Active B-Spline/NURBS Surfaces enabled by the customised approach offers a more intuitive alternative to complex and tedious manual control point manipulations. The Active B-Spline/NURBS formulation provides an exact mathematical description of the surface throughout the process, such that there is no division in representation. Additionally the ACM-based approach facilitates different interaction behaviours to those provided by existing sculpting techniques. The compromise offered by an ACM-based approach contributes to facilitating physically plausible results at interactive rates. (Chapter 4)

- **An innovative Efficient Mathematical Framework is developed, based on a novel analytic solution for the energy-based minimisation of a B-Spline/NURBS surface model, which is shown to handle the deformations of the analytic surface up to** 4 **times faster than the current state of the art numeric solver**. While

initially designed to facilitate the seamless integration of a Virtual Sculpting paradigm within the CAD design process, this solver can also readily replace current numeric solvers for the analysis process, such that the overall approach is fully integrated, efficient and truly seamless. The overall result is the removal of the 80% bottleneck, and exact analyses that are 4 times faster than the current state of the art. As part of the development of the mathematical framework, several distinct contributions make up the overall contribution. These contributions are as follows: (Chapter 5)

- Gaussian Quadrature is the current state of the art technique in IgA for the energy-minimisation of an Active B-Spline surface. **As part of the developments presented in this thesis, an enhanced Gaussian Quadrature approach is derived and developed**. Optimisations are achieved by tailoring the approach to take advantage of combined problem-specific efficiencies and derived approach-specific efficiencies. Additionally, algorithms for its implementation are developed and implemented. The optimisations are equally applicable to Active NURBS surfaces.
- **A novel fully Analytic approach to the energy-minimisation of an Active B-Spline Surface is derived and developed**. The resulting algorithm is shown both theoretically and experimentally to be up to 4 times faster than the tailored Gaussian Quadrature approach developed. The approach is shown to handle complex conditions such as mass, damping and varying material properties and forcing functions. It is also shown to be applicable to NURBS surfaces.
- **Full treatment of the Integral of a B-Spline basis function is developed and presented**. This is an area found to be lacking coverage in the literature. Several algorithms are provided for its evaluation in various situations.
- **Augmented treatment of the Derivative of a B-Spline basis function is developed and presented, covering situations neglected in the literature**. Accompanying algorithms are developed and presented.
- **Novel physical and graphical interpretations of the underlying mechanics of the analytic solution are developed and presented**.
- **Extension of the proposed Analytic Approach to handling varying material properties is developed and presented.**
- **Extension of the proposed Analytic Approach to handle mass, damping and forcing functions is developed and presented.**
- **Extension of the proposed Analytic Approach to the important case of NURBS is developed and presented.**
- **Comprehensive theoretical and experimental stress testing and evaluation of the algorithm is performed, and the algorithm is shown to perform**

well numerically for extreme cases. An in-depth analysis of the cause of small numerical round off inaccuracies in computing the algorithm is presented and techniques are discussed for controlling their already small impact.

- **A prototype Virtual Sculpting environment is developed, implementing each of the algorithms developed**. The environment is presented as further 'proof of concept' of the applicability of each of the outlined contributions to the domain of Virtual Sculpting for CAD. (Chapter 6)
- **An auxiliary contribution of this research is the development of a JAVA 3D library implementing Active B-Spline/NURBS surfaces**, partially based on customised versions of the algorithms presented in *The NURBS Book* by Piegl and Tiller (1997), and complete with the novel algorithms developed during the course of this research.

Literature Review 2

Due to the multidisciplinary nature of freeform shape design and analysis, there is a vast amount of literature in the area. This has served to confuse and overwhelm the general practitioner. An understanding of interactive freeform surface modelling requires a good grasp of the literature relating to shape representation, deformable modelling, and interactive design or Virtual Sculpting practices. This chapter presents a thorough exposition of the literature necessary to fully contextualise freeform surface modelling. For each of the three fundamental topics identified, the chapter presents the following: firstly, a background to the topic, secondly, a review of the literature in the area, and thirdly, a context for the outstanding challenges in the area. The pertinent mathematical constructs and models used to capture the various shape representations, to describe the deformation of these shapes, and to facilitate their interactive modification within a Virtual Sculpting paradigm, are presented at the appropriate stages. The main goal of the chapter is to set out the current state of the art in shape modelling and deformation. This is to enable appropriate decisions in later chapters, regarding the recited techniques, by providing a systematic description of their assumptions and setup in each case.

2.1 Shape Representation

Geometric modelling is an ancient field of study concerned with representing shape and spatial relation information through mathematical and abstract descriptions. Geometric modelling technologies play a critical role in computer graphics model creation by providing complete and accurate geometric data. The advent of computers and the subsequent advances in computational power have facilitated the implementation of complex mathematical shape representations in the digital domain. Today, geometric information is pervasive in computing and supports the creation, communication, visualisation, animation, and interrogation of digital models.

Approaches to computer-based geometric modelling for digital shape representation vary substantially in the literature relating to Computer Graphics, CAD and Computer Vision. This is because there is presently no single representation of shape that can be seen as a universally satisfactory solution to the disparate problems that arise across the host of different applications within each domain.

Drawings and sketches are often used as the primary media for the communication of an object's shape information. Using drawings or sketches, three-dimensional shape

information is typically presented using two-dimensional drawings of the silhouette of the object under a perspective projection. Shape information is also often communicated by describing an object's shape in terms of other well known shapes. These techniques are generally successful for conveying the shape information of simple or well-known objects where the three-dimensional details can be deduced from a-priori knowledge and/or intuition. For less familiar and more complex objects, such schemes often prove to be inadequate.

Early industry attempts to adopt a standard means of shape communication, or to represent objects in a standardised way, resulted in engineering drawings that described objects using a collection of planar orthogonal projections from different view points. These multiple views helped to further convey the three-dimensional information of the object. However, these drawings still relied heavily on a-priori information and intuition for successful shape communication. Early computer-based shape representations were heavily influenced by real world shape communication techniques. As processing power increased, these digital representations drew more and more from complex mathematical descriptions of shape and spatial relations studied in geometry.

This section presents an overview of geometric modelling techniques for digital shape representation, approached with an historical perspective to provide a sense of the evolution of shape representation and to provide a context for the use of shape representation for the interactive shape modelling discussed in later sections.

2.1.1 Wireframe Models

Developed in the early 1960s, wireframe representations were among the earliest geometric modelling techniques. It is natural to humans to describe three-dimensional objects by their outlines/edges. It is hardly surprising then that early computer-based models represented an object's shape with straight lines or curves connecting the constituent vertices on the surface of the object. Though simple and intuitive, wireframe schemes suffer from several deficiencies (Requicha and Voelcker, 1982), most notably the problem of ambiguity. This problem stems from the possibility of interpreting a single wireframe model several ways, with each interpretation representing a different object (see Figure 2.1). A simple solution to the ambiguity problem associated with wireframe representations was to add faces. This technique led to the now well-established boundary representations.

2.1.2 Boundary Representations/Polygon Models

Since one only sees the surface of solids, it is natural to expect a representation based solely on the boundary of an object (Agoston, 2004). The 1970s saw boundary representations emerge as a standard representation scheme in computer graphics modelling (Requicha and Voelcker, 1982). Boundary representations can be seen as a generalisation of wireframe models, expressed in terms of faces, thus solving the associated

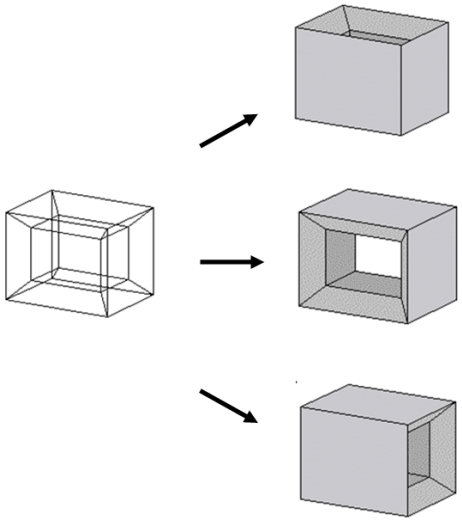


Figure 2.1: Wireframe Ambiguity

ambiguity problem. Early boundary representations were restricted to simple set operations to produce models based on a union of an object's bounding faces (see Figure 2.2). Today there are many different boundary representations available.

The polygonal representation is the classic representation scheme in computer graphics. A polygonal model approximates the boundary of an object with a mesh of planar polygonal facets. All that is required is a collection of sample vertices from the surface of the given object and a corresponding piecewise polygonal representation is generated by connecting these sample vertices using a polygonisation scheme (see Figure 2.3).

Polygons, particularly triangles, have several desirable properties that make them suitable for machine processing. They are flat structures and made up of straight edges. Triangles have the additional property that they cannot self-intersect. Polygons are planar, however, and can only approximate curved surfaces. This can lead to visual

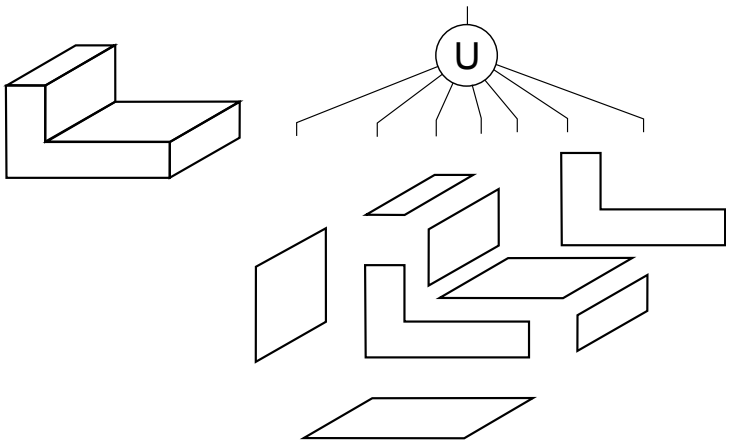


Figure 2.2: Early Boundary Representation

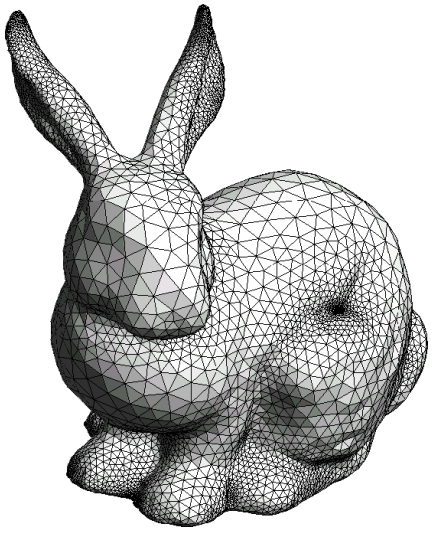


Figure 2.3: Polygonal Representation: Stanford Bunny (Rypl, 2003)

artefacts in the rendered model. However, the visual impact of the piecewise linearities can be greatly reduced by efficient shading algorithms. One of the key advantages of triangular mesh representations is that they are not restricted by geometric, topological, or connectivity constraints like many other representations. Today, the vast majority of 3D models at some stage of the graphics pipeline are represented as textured polygonal models. They have proved to be topologically flexible, generally straightforward to create and can be quickly rendered by a computer. The advances in laser range scanning technology has made polygonal representations even more popular in recent years.

2.1.3 Constructive Solid Geometry Models

Pure solid modelling representations emerged in the 1970s in the form of Constructive Solid Geometry (CSG). While boundary representations have many advantages for efficient rendering algorithms, object definitions can be complicated from a design perspective (Requicha and Voelcker, 1982). CSG arose from the idea of expressing an object's shape information in terms of other well known shapes. Engineers realised that many manufactured objects could be represented by combining geometric primitives into a tree of successive Boolean operations and linear transformations (Ricci, 1973) (Agoston, 2004) Cani et al. (2008).

Primitives and operators are arranged in a binary tree structure with operators at internal nodes and primitives at the leaves. Any node may have one 'parent' and two 'child' nodes. The root node of the tree has no parent and represents the complete model. The leaf nodes of the tree have no children and represent simple primitives. Internal nodes may be used to represent the linear operations such as translations, rotations, and scaling, or Boolean operations, such as union and intersection (see Figure 2.4).

CSG proved to be a more intuitive shape representation than the more machine driven polygonal representations. Using this technique, designers built up a shape

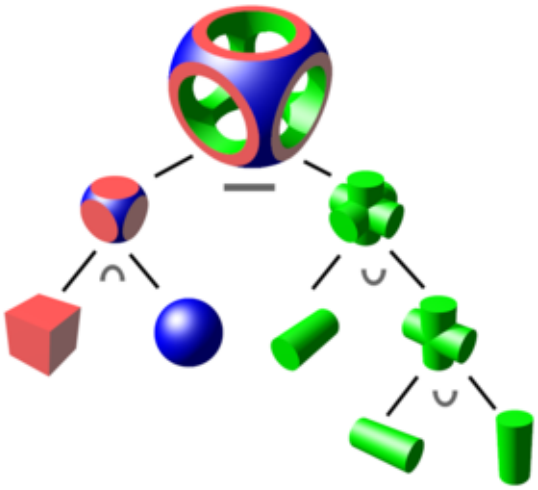


Figure 2.4: Constructive Solid Geometry Tree representing an object's shape using combinations of Boolean operations (*CSG Tree [Online]*, n.d.)

description using the metaphor of three-dimensional building blocks. Early schemes were very much limited to representing combinations of regular shapes. It is difficult to deal with freeform models using a CSG representation. An example implementation of a CSG scheme will be examined in Section 2.1.6.

2.1.4 Parametric Models

Although CSG worked well for describing exact and regular shapes, it had limited scope in the representation of free-form shapes. This problem precipitated the development of parametric representations known as splines which gained popularity rapidly in the 1980s. Surface design with parametric splines originated in the automobile industry, principally for car body design; in the shipbuilding industry, for the design of ship hulls; and in the aircraft industry, for the design of wings, fuselages, and so on.

Splines are composite, continuous curves formed with polynomial sections, satisfying specified continuity conditions at the boundaries of each section. There are several different kinds of spline, each with its own distinct set of properties, specified by the type of polynomials used and the boundary conditions enforced e.g. Bezier Spline curves, NURBS, etc. However, the procedure for constructing each type of spline curve is similar.

Curves or surfaces are represented by a set or mesh of control points. The curve or surface approximates the given control mesh, whilst guaranteeing a certain level of smoothness. This is achieved by using the control points to weight a linear sum of basis functions associated with each spline type.
The parametric curve $C(u)$ is defined by

$$C(u) = \sum_{i=0}^{n} P_i B_i(u) \tag{2.1}$$

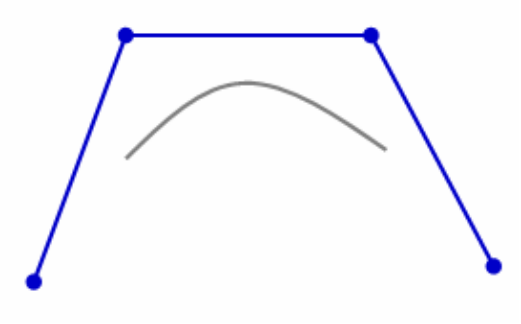


Figure 2.5: Parametric Spline Curve

where P_i are the control points and $B_i(u)$ are the basis functions (see Figure 2.5). The parametric surface $S(u, v)$ is defined by a related tensor-product patch. This is essentially a bidirectional curve scheme (see Figure 2.6). Here, the bases are bivariate functions of the parameters u and v, and are constructed as a tensor product of univariate basis functions. This is given by

$$S(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} P_{ij} B_j(v) B_i(u) \tag{2.2}$$

This parametric scheme can be further generalised to represent volumes.

NURBS, which define surfaces using ratios of polynomials, are worth a special mention as rational functions permit much better control over the derivatives of curves, and hence their curvature, than polynomials alone.

Spline models are explicit representations that have the advantage that their parameterisations enable the reduction of several three-dimensional problems on the surface to two dimensional problems in the parameter domain (see Equation 2.3).

$$f(x, y, z) \rightarrow f(u, v) \tag{2.3}$$

For instance, points on a surface S can be found by simple function evaluations of

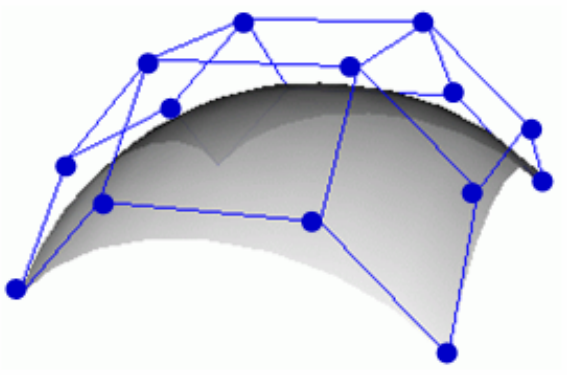


Figure 2.6: Parametric Surface

$f(u, v)$. Neighbourhoods on the surface can similarly be evaluated using neighbouring points in the parameter domain.

Using such representations, even complex curves can be described by a small number of vectors. In addition to this, several of the more popular spline representations inherently support local control of the model, facilitating interactive modification. However, the manual manipulation of large numbers of control points can be laborious and is sometimes necessary for even perceptually simple changes.

Although spline representations offer many advantages, there are difficulties associated with representing closed surfaces or branching structures. Their explicit parametric representation imposes the restriction that the parameter domain has to match the topological and metric structure of the surface it represents. The generalisation from spline curves to surfaces uses a tensor-product construction, resulting in a rectangular parameter space and a surface which is topologically equivalent to a plane. Closing or looping the parameter space makes it possible to create a surface that is topologically an open cylinder or a torus, but a single spline patch cannot represent a surface of any higher genus.

As a consequence of their topological constraints, typical CAD models use many surface patches in order to represent a high quality smooth surface. Where separate spline surfaces meet, they must be manually 'stitched' together, and the seams do not have the same continuity guarantees which splines provide for the rest of the surface. In the specific cases where the surfaces have compatible knot vectors, these guarantees can instead be achieved by careful positioning of the control points on either side of each seam. However, numerical inaccuracy means that the composite surfaces are often not even continuous at the seams, let alone smooth (Cashman, 2010). Recent developments have led to more complex structures known as T-splines that allow the formation of T-junctions within the spline control grid (Sederberg et al., 2003). These structures can be used in conjunction with B-Splines to overcome some of their associated limitations.

In spite of their limitations, their flexibility and computational efficiency has meant that NURBS curves and surfaces rapidly became the *de facto* standard for the representation of geometric information in computer processing. There are many textbooks available devoted to the coverage of curve and surface modelling with splines. Excellent examples are Bartels et al. (1987), Piegl and Tiller (1997), and Cohen et al. (2001).

2.1.5 Subdivision Models

Models generated using a subdivision scheme are known as subdivision models (Ma, 2005, Cani et al., 2008). Although developed in the 1970s, subdivision models did not become widely used until the late 1990s. Much like the spline models described in the previous section, a subdivision model is initially described by a set of control vertices. However, unlike splines, where, for surfaces, we are restricted to the use of a rectangular mesh structure, subdivision surfaces can be described by any arbitrary control mesh, which means that the resulting surface generated can be of arbitrary

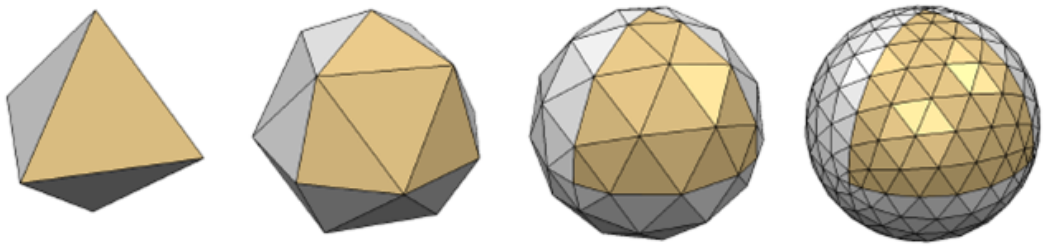


Figure 2.7: Smooth Subdivision Surface defined by an iterative subdivision applied to its control mesh: each subdivision step halves the edge lengths, increasing the number of faces by a factor of 4 (Botsch, 2006)

topology. It is this flexibility of topology that precipitated the growth in the popularity of subdivision models throughout the 1990s. Unlike the patchwork models required to represent arbitrary topologies with splines, a subdivision mesh produces a single surface, allowing us to manipulate the control net without worrying about seams or other continuity issues.

A subdivision surface generates a smooth approximation of a polygon mesh using the rules laid out by a subdivision scheme. Various schemes have been developed but most involve an iterative process whereby the original control vertices are kept, and new additional vertices are calculated as a function of the original vertices. The process then iterates until the desired level of subdivision is achieved. The limit of this subdivision process is a smooth limit model. Interestingly for many of the standard subdivision schemes, in the case of uniform subdivision, the limit of this process is in fact a spline model (Botsch, 2006) (see Figure 2.7).

The success of subdivision surfaces is evident by their use in the computer graphics industry. Pixar switched from the use of NURBS for *Toy Story 1* to the use of subdivision surfaces in *Toy Story 2* to avoid the discontinuity issues associated with spline patches.

In spite of the topological flexibilities introduced by subdivision surfaces, they still suffer from many of the problems associated with spline models. Like spline models, modifying the surface involves indirect editing via its control mesh, which generally requires skilled designers with a good understanding of the underlying mathematical representation. In addition to this, while the topology of the control mesh is flexible, its explicit storage makes it difficult to modify after it has been defined. Recent research by Cashman has facilitated a unified model for NURBS and subdivision surfaces, which has resulted in increased research interest in NURBS after an arguably quiet period (Cashman, 2010).

2.1.6 Implicit Models

Implicit surfaces were introduced in the 1980s as an alternative to splines to address their associated topology issues (Cani et al., 2008) (see Section 2.1.4). The basic concept is to characterise the whole embedding space of an object by classifying each point as lying either inside, outside or on the surface bounding the object.

Mathematically, an implicit function is a function in which the dependent variable has not been given "explicitly" in terms of the independent variables, as in Equation 2.4 (Agoston, 2004).

$$f(x, y, z) = 0 \tag{2.4}$$

Such functions can be used to represent a variety of shapes. A typical example of an implicit function for a sphere is given in Equation 2.5, where r is the radius of the sphere.

$$x^2 + y^2 + z^2 - r^2 = 0 \tag{2.5}$$

Implicit functions in this direct form are of limited usefulness for shape representation as there is a limit to the number of intuitive objects that can be represented in this way. However, more flexible uses of implicit representations in computer graphics appeared in the 1980s.

The implicit function for a given surface is not uniquely determined, and can be represented using continuous algebraic surfaces, radial basis functions, discrete voxilisations, etc. The surface generated in each case, $f(x, y, z) = 0$, is known as the zero-level iso-surface of the scalar field defined by the implicit function, $f(x, y, z)$. The most common and arguably the most intuitive representation of an implicit surface is a distance field function whose value at a point decreases with distance from a surface. Distance fields thus represent the surface by storing the scalar distance to the surface at a given point (Payne and Toga 1992). The distance field of the unit sphere is given by Equation 2.6.

$$f(x, y, z) = 1 - (x^2 + y^2 + z^2)^{\frac{1}{2}} \tag{2.6}$$

Here, f is the Euclidean signed distance from the unit sphere centred at the origin. By definition, the field $f(x, y, z)$ inherently characterises an in/out/on function where if $f(x, y, z) = 0$ then the point (x, y, z) is on a surface described by the implicit function, points satisfying $f < 0$ are on one side (nominally the 'inside') of the surface, while those points for which $f > 0$ are on the other side of the same surface. Here, f does not explicitly describe the surface, but implies its existence.

Scalar fields defined by implicit functions, like the distance field described above, can be blended together to produce a new global scalar field whose iso-surface can be used to represent a wide variety of organic shapes. The basic idea involves blending simple shapes described by implicit functions to create more complex shape representations. If you have two implicit surfaces $f(x, y, z) = 0$ and $g(x, y, z) = 0$ that are fairly continuous, with a common in/out/on sign convention then the implicit surface defined by $f + g = 0$, for example, is a blend of the shapes. Such blending techniques form the basis for 'Blobs' (Blinn, 1982*a*), 'Metaballs' (Nishimura et al., 1985), and 'Soft objects' (Wyvill et al., 1986). Figure 2.8 shows an example of the blending of two implicit surfaces.

Figure 2.8: Implicit Surface (Matthew Ward, n.d.)

Because implicit surfaces conveniently define volumes, they are used frequently in CSG-based solid modelers (Requicha and Voelcker, 1982). Boolean operations, such as union, $\cup$, and intersection, $\cap$, can be achieved by simple min and max combinations of the objects' implicit functions. In order to efficiently process implicit representations, the scalar field, F, is typically discretised in some bounding box around the object using a spatial subdivision/decomposition/voxelisation to sample the field on a regular grid (see Section 2.1.7).

Rendering implicit surfaces is also a difficult task. Implicit surfaces are often converted to an explicit boundary representation (see Section 2.1.2) for processing and rendering efficiencies. The *de facto* standard algorithm for this iso-surface extraction is 'Marching Cubes' (Lorensen and Cline, 1987). Marching Cubes samples the implicit function on a regular grid and processes each cell to check for intersections. For each cell that is intersected by the iso-surface, a surface patch is generated based on certain local criteria. The collection of all these small pieces eventually yields a triangle mesh approximation of the complete iso-surface.

An important property of implicit surfaces is that since they are defined as level-sets of a scalar field function, geometric self-intersections cannot occur.

Quadrics and Superquadrics

Quadrics and Superquadrics represent an interesting subset of implicit model descriptions. A quadric surface is a subset of points in R^3 that satisfies a general quadratic equation of the form

$$ax^2 + by^2 + cz^2 + dxy + exz + fyz + gx + hy + iz + j = 0 \tag{2.7}$$

Superquadrics are a special class of shape primitives derived from quadric surfaces. They are built on purely geometric foundations and provide a flexible family of 3D parametric objects for use in computer-based modelling (Montagnat et al., 2001).

Superquadric surfaces are derived from quadric surfaces. However, extra flexibility is achieved by raising each trigonometric term to an exponent. These exponents control the relative roundness and squareness in both the horizontal and vertical directions.

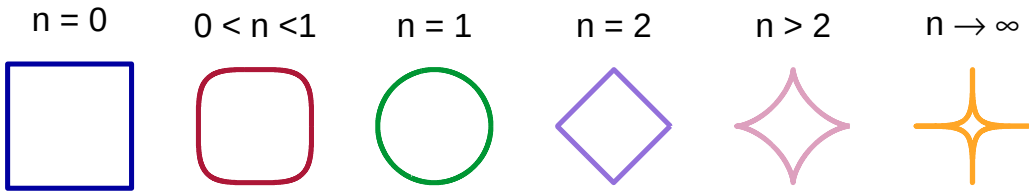


Figure 2.9: Superellipses with varying trigonometric exponents

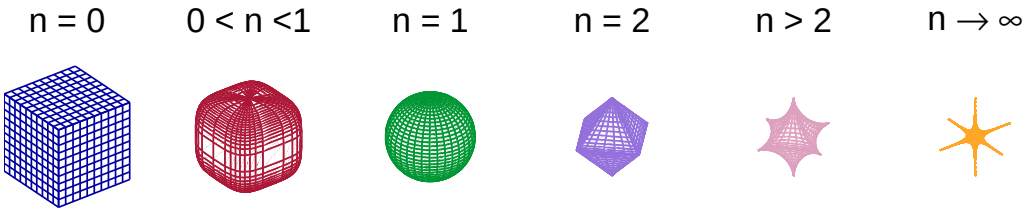


Figure 2.10: Superellipsoids with varying trigonometric exponents

An ellipse is a 2D example of a quadric. The parametric representation of an ellipse centred on the origin of the coordinate axes is

$$x(\theta) = r_x \cos(\theta) \tag{2.8}$$

$$y(\theta) = r_y \sin(\theta) \tag{2.9}$$

where r_x and r_y represent the span of the ellipse in the x and y directions, respectively. A superellipse is defined by the parametric representation

$$x(\theta) = r_x \cos^n(\theta) \qquad -\pi \leq \theta \leq \pi \tag{2.10}$$

$$y(\theta) = r_y \sin^n(\theta) \tag{2.11}$$

Various shapes can be generated by varying the 'n' parameter (see Figure 2.9). Various shapes generated by a superellipsoid with varying trigonometric exponents are shown in Figure 2.10.

There are similar exponential trigonometric functions available to represent all quadric surfaces. These include ellipsoids, cylinders, cones, paraboloids, and hyperboloids. The structures produced by these functions are known as superquadrics.

A particularly attractive feature of superquadrics is their simple mathematical representation. Superquadrics are generally used in model construction by combining different shapes to create complex models.

2.1.7 Spatial Decomposition

Sections 2.1.2 and 2.1.5 examine representing an object by decomposing it into smaller, simpler pieces. Space decomposition applies subdivision techniques in the spatial

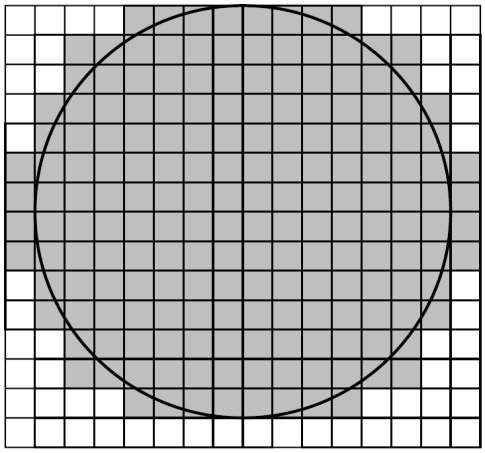


Figure 2.11: Spatial Subdivision: Circle represented by a Uniform Subdivision of Space

domain, representing an object by subdividing the space it occupies (Agoston, 2004). In three dimensions, volumes are divided into cubic cell elements called 'voxels' to form a 'voxmap'. This is analogous to the pixels that decompose two dimensional image data in a bitmap to form a pixmap.

The simplest decomposition scheme involves uniformly subdividing the space into a 3D grid of voxels, usually equally sized and spaced cubes, and marking all the cubes that intersect the object interior. The object would thus be represented by the collection of marked cubes. For such volumetric representations, topologies can be arbitrary and Boolean operations, such as those discussed in Section 2.1.3, are trivial to implement. However, this representation is approximate, and the size of the individual elements will determine the degree of accuracy. Another significant problem is the large memory required to store the voxel grid. See Figure 2.11 for a 2D example. Other metrics can also be used within such a decomposition scheme. Implicit surfaces (see Section 2.1.6) are often sampled into such a discrete grid of values for efficient processing.

The main issues with this technique are the tradeoffs between memory consumption and accuracy. When a high degree of accuracy is required, the number of cubes may be too great for this data structure to be considered convenient. Adaptive decomposition and structural organisation techniques such as octrees etc. alleviate storage costs, e.g., (Gibson et al., 2000). This technique has become more viable in recent years with gigabytes of memory becoming a reality. Voxel-based representations have become very popular in the visualisation of large datasets.

2.1.8 Meshfree Models

A meshfree model is essentially a model whose underlying representation does not contain any topology information or explicit connectivity information between vertices. In simple terms, it is a point-based representation rather than a mesh-based representation.

Point-based representations are not new to computer graphics. Indeed, there are

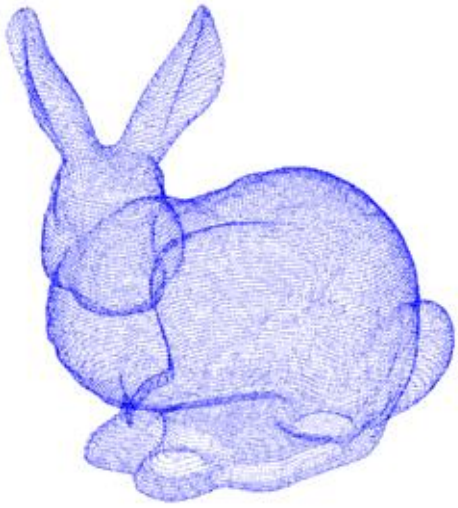


Figure 2.12: MeshFree Model (Stanford Bunny) (FarField Technology, n.d.)

many situations where points have long been considered inherently better model primitives than triangles. Objects like water, trees, clouds and smoke lend themselves better to point representations than mesh representations (Blinn, 1982*b*, Reeves, 1983).

Levoy M. (1985) suggested that as the visual complexity of computer generated scenes increases, the use of classical modelling primitives as display primitives becomes less appealing. They also put forward the notion of points as a universal meta-primitive. Although initially somewhat overlooked by the graphics community, over the past decade the ideas presented in this seminal paper have experienced a resurgence in computer graphics research. Early examples of this resurgence include Rusinkiewicz and Levoy (2000), Levoy et al. (2000), and Pfister et al. (2000).

The complexity of 3D model geometry has increased to a great extent. This is mainly due to the advancement and growing popularity of 3D scanning technologies. The concurrent increase in available memory and processing power have made it possible to capture and process the large data sets associated with high resolution geometric models captured using such scanning technologies.

As discussed in Section 2.1.2, mesh-based polygon models are ubiquitous in computer graphics. They have attracted interest as low level outputs for the higher level shape representations discussed in Sections 2.1.4 to 2.1.7, as an output representation for laser scanned objects, and have become the *de facto* standard shape representation for machine processing. Although triangles are currently the most popular display primitives, as geometry becomes more and more complex, the triangles become smaller and smaller, until a point where the associated overhead is no longer justified. With advances in screen resolution falling behind those in model resolution, triangles often only occupy sub-pixel screen regions. Processes such as generating and maintaining connectivity information of a triangle mesh become intractable with the huge point data sets output by modern 3D scanning devices. In such situations, it becomes easier

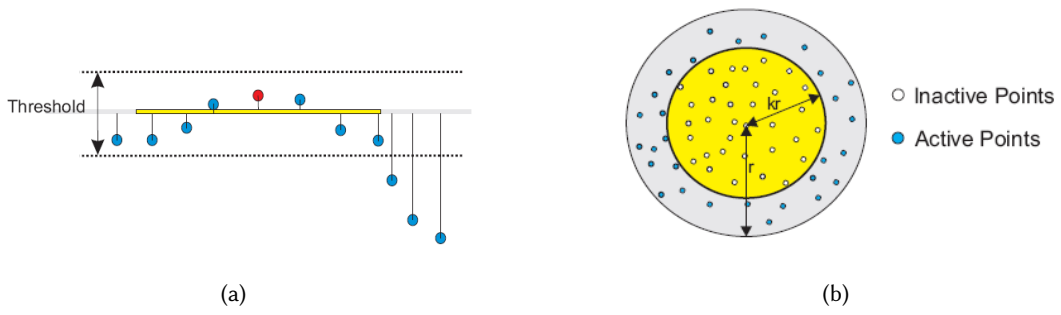
to deal with the points of the data set themselves. Meshfree models have emerged as an alternative to the traditional polygonal representation in such cases.

Point based representations can be visualised by rendering point primitives directly. This requires a very large point cloud to reasonably approximate the shape (see Figure 2.12). A more desirable solution and an important challenge for point rendering techniques is to reconstruct continuous surfaces from the irregularly spaced point samples while maintaining the fidelity of the scanned data. In order to appear as a continuous surface, points need extrapolation as the view is moved closer to the surface. Typically, a point sample also has a radius to define an area around it. Such samples are called 'surfels' and approximate the shape of the surface linearly in their neighborhood.

The most popular conversion of point sample data to surfels is called surface splatting (Zwicker et al., 2001, Kashyap, 2008). The surface splatting approach is based on computing a surface normal at each sample's surface point and a radial or elliptical expansion tangential to the surface. The generated discs are called 'splats'. These splats overlap in order to cover the entire surface of the scanned object resulting in a hole-free piecewise linear approximation of the input data.

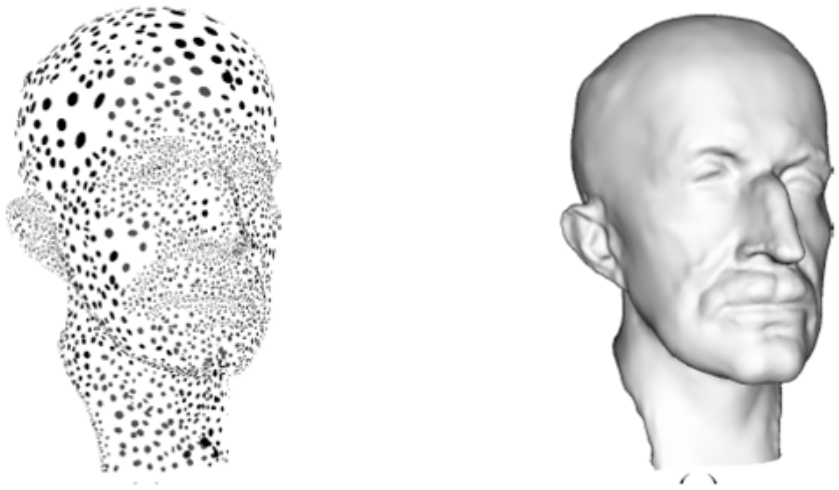
Starting with a seed point (shown in red in Figure 2.13(a)), the local normal direction is estimated by fitting a least squares plane to the seed point and its nearest neighbors. The splat is grown by adding neighboring sample points in the order of their projected distances to the seed point. For each new point added, its signed projected distance is computed and the growing stops as soon as the error becomes larger than a predefined threshold. The centre of the splat is then set as the seed point offset by the average of the minimum and maximum projected distances of points from the splat. Its radius is then the distance from the centre to the point where the thresholding fails (see Figure 2.13(a)).

To ensure a hole-free surface, a surface like this could be created for every point. However, this would be a highly inefficient scheme. The number of splats needed to cover the surface depends on the value of the threshold. General approaches are based on the relative distances of points to the splat centres. As points are determined to exist inside a splat they are deactivated and will no longer be used as a seed for a new splat (see Figure 2.13(b)).



(a) (b)

Figure 2.13: Meshfree Model Technique: (a) Moving least squares approach (b) Splatting Density (Kashyap, 2008)



(a) (b)

Figure 2.14: Meshfree Model Example: (a) splatting technique (b) continuous model (Pauly et al., 2003)

The naive rendering of inter-penetrating splats results in visual surface discontinuities. Therefore, final surface reconstruction is generally done by some averaging of contributions of all splats relating to a pixel, e.g., Zwicker et al. (2001) proposed a technique whereby, for each pixel, a weighted average is taken of the splat fragments with similar depth values. If surface splats are to represent sharp features, all splats that sample these features have to be clipped against clipping lines that are specified in their local tangent frames (Pauly et al., 2003).

Note that even though each splat individually represents a surface, we have no global surface information about the model. This is because each splat is independent of its neighbours, and is not connected in any way with other splats (see Figure 2.14(a)).

Another popular technique for surface reconstruction, the point set surface approach, is based on the moving least-squares surface definition (Alexa et al., 2001, 2003). The surface is locally reconstructed by fitting a polynomial to the sample points within a small neighborhood surrounding a given point. The given point is projected onto an implicitly defined polynomial surface. The point set surface is defined as the set of points that project onto themselves. The splatting approach is much simpler in its mathematical formulation and computational complexity. However, the point set surface approach implicitly generates continuous surface representations.

Point-based models provide us with a simple structure for dealing with large or complex data sets. The biggest advantage of point-based representations over polygonal meshes is that they can easily be restructured. Hence, applications requiring frequent geometry re-sampling will benefit most from point-based methods. However, point-based models basically lack an abstract governing structure which would help the user to quickly modify the shape in a meaningful way.

Meshfree modelling is still very much an active research area. Much of the cur-

rent research focuses on compression, improved surface reconstructions, visualisation techniques and the development of efficient algorithms to handle such geometric representations directly, without resorting to manual conversion to more conventional surface representations (Öztireli et al., 2009, Kashyap, 2008, Kashyap et al., 2010). An excellent survey of point-based techniques can be found in Kobbelt and Botsch (2004) and excellent tutorials in Alexa et al. (2002) and Gross (2009).

2.1.9 Discussion

Computer graphics shape representation for object modelling is very much an unsolved problem. Various shape representations have evolved under a variety of influences. Polygon/Faceted/B-rep models have several nice properties that make them a suitable choice for machine processing. However, although they have found a place in user/interface representations, they are not a natural choice from a user perspective. This need for intuitive representations led to CSG techniques. The lack of an intuitive description within a CSG framework for free form shape descriptions precipitated the use of parametric spline models in a digital modelling framework.

The generalisation from B-Spline/NURBS curves to surfaces uses a tensor-product construction, resulting in a rectangular parameter space and a surface that is topologically equivalent to a plane. Closing or looping the parameter space makes it possible to create a surface that is topologically an open cylinder or a torus, but a single patch cannot represent a surface of higher genus. Parametric topology restrictions fueled the development of subdivision surfaces. Topologically more flexible Subdivision Surfaces are often used as an alternative representation to B-Splines to combat this issue. However, they too suffer from the restriction that once the topology of a virtual object has been defined, it is difficult to modify.

The lack of a simple description for closed surfaces using parametric splines and subdivision surfaces led to Implicit Models. It is difficult to model freeform shapes with implicit models. Their ability to represent a shape in terms of an 'inside' or 'outside' relationship has led to implicit models being very useful in collision detections. Their collision detection qualities in conjunction with their ability to conveniently represent standard shapes (quadrics and superquadrics) makes them good candidates for the representation of sculpting tools.

Tesselated surfaces are the most widely used representation for general computer graphics shape modelling. Today, the majority of 3D models at some stage of the graphics pipeline are represented as polygon models. Their flexibility of representation and suitability for machine processing has always made them an attractive option for the later stages of the pipeline. Advances in automation of content capture via laser scanning technologies has fuelled a growth in their popularity throughout the graphics pipeline. In spite of their advantages, the use of tesselated surfaces has drawbacks within the domain of interactive shape modelling and deformation for organic freeform shape design. Where automation of content capture is not available, polygonal rep-

resentations can be difficult to define. Additionally, polygons are planar and can only approximate smooth surfaces. This can lead to artefacts in a rendered model. Moreover, the quality of the triangles deteriorates as the model deforms, introducing additional artefacts.

Recent advances in technology have led to larger data-sets being used for representing shape, spawning a new trend in digital shape representation, where the polygon mesh that has been popular since the very beginning has been set aside in many applications. However, this is another machine processing oriented representation.

B-Spline/NURBS representations remain the standard for CAD/CAGD applications. They are well suited to the task of modelling freeform shapes and have the benefit of providing an exact mathematical description of the freeform shape. Their piecewise representation makes local modifications possible. Their compact representation and stability of the methods that exist for processing them are additional advantages for CAD/CAGD.

Overall, there is no universal solution to satisfy the many problems that exist in digital shape representation. Rather, as outlined in this chapter, particular modelling methods have evolved for particular contexts.

2.2 Deformable Modelling Techniques

Until the 1980s, computer based modelling techniques had only allowed for the modelling of rigid bodies. In 1984, Barr (1984) introduced a series of geometric operators for deforming a solid object by transforming its co-ordinate space. These operators paved the way for a more generalised free-form deformation technique, known as Free Form Deformation (FFD), introduced by Sederberg and Parry (1986). This technique facilitated the deformation of an arbitrary object via a structural hyper-patch. In the following year, Kass et al. (1987) introduced explicit 2D Active Contours, which were soon generalised to the 3D case by Terzopoulos et al. (1987), who coined the term 'Deformable Models'. These seminal papers introduced models which allow for elastic deformations of objects by incorporating concepts from continuum mechanics. The vast scope of deformable models was quickly recognised, and these early techniques provided an extensible framework for the construction of 2D and 3D virtual models.

Current deformable modelling techniques combine elements from geometry, physics, advanced calculus, approximation theory, statistics, and numerical computation studies. The complexity of such problems has been the primary limiting factor in the development of deformable models, particularly for real time applications (Müller et al., 2002). Where accuracy is paramount, simplifications are often not acceptable. However, as computational power increases, complex models are processed at more reasonable rates. Today, deformable models are used for simulating car crashes (Varkonyi-Koczy et al., 2007), tracking motion in surveillance videos (Hu et al., 2004), intuitive Virtual Sculpting environments (Gao and Gibson, 2006), and countless other familiar applications. Highly sophisticated models are used for even complex medical applications,

such as medical image analysis (McInerney and Terzopoulos, 1996), real-time surgical simulations (Schein and Elber, 2004), and 3D reconstructions from Magnetic Resonance Imaging (MRI) or Computerised Tomography (CT) scans (McInerney and Terzopoulos, 1996).

Deformable models are now established tools in the areas of Image Analysis, Computer Graphics, and Visualisation. Accordingly, as many different construction problems have different requirements, various modelling approaches have been proposed. This section presents a survey of the research carried out to date in the area of deformable modelling. As this section deals with many disparate approaches tailored for various application domains, the section organises the various approaches by technique, and provides a description, critique, and an overview of applications for each. Finally, the 'state of the art' of deformable modelling is discussed through comparing and contrasting the overall schemes, with a focus on their suitability for interactive computer-based freeform modelling environments.

2.2.1 Non-Physics-Based Techniques

Non-physics-based techniques are purely geometric techniques used to deform virtual objects. In such cases, the system has no knowledge about the material of the object being deformed. These techniques are appropriate for applications where physical accuracy can be sacrificed for computational efficiency.

Model-Based Techniques

Section 2.1 presented an overview of the various geometric models used for digital shape representation. Model-based techniques for deformation rely on knowledge of the model's underlying shape representation.

Technique

Model-based deformations can be achieved by directly editing the variables or parameters of the original model, e.g., modifying the position of control points to edit a spline model (Foley, 1996, Hearn and Baker, 1994).

Critique

As discussed in Section 2.1.4, modifying a model by directly manipulating its control parameters can be laborious and far from intuitive. The manual editing of large numbers of parameters requires time and effort and often requires skilled designers with experience and understanding of the underlying shape representation.

Moreover, the control afforded is often indirect, e.g., modifying control points that do not lie on the surface of the object being modelled. Direct editing of lower level representations, such as polygon meshes, can also require effort and skill as smoothness can quickly deteriorate when manually modifying vertices of a mesh. In addition to

this, it is often difficult to maintain the original structure of the model for desirable results.

Research in this area has addressed some of these issues by providing specific interfaces for easing manipulation of geometric models. Section 2.3 on Virtual Sculpting will cover these techniques in detail.

Applications

Although direct editing of model parameters is rarely intuitive, it is often relied upon by designers as the search for more intuitive design mechanisms is still a very active area of research. However, a useful application of direct model editing lies in the kinematic modification of model parameters to facilitate aesthetically pleasing deformation simulations. Superquadrics (see Section 2.1.6) can be deformed very simply kinematically, i.e., without taking mass or inertia into account. For example, a sphere can be deformed as its radius length is changed.

Spatial Techniques: Free Form Deformation

FFD is a space-warping tool that has become very important in CAD/CAGD and animation. Although not strictly considered FFD techniques by today's standards, Barr (1984) developed some useful deformation operators (bending, tapering, twisting, and stretching), which were independent of control points (see Figure 2.15). Complex deformations, once achieved by skilled and laborious manipulation of numerous control points, could now be achieved by applying these operators to an object in a hierarchical fashion. However, Barr's operators are constrained to performing actions about a single axis and the control afforded to the user is not very intuitive.

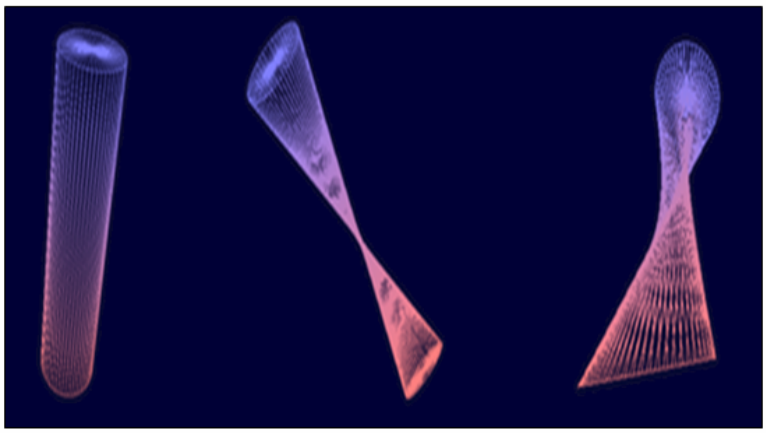


Figure 2.15: Examples of Barr's Operators (a) Undeformed (b) Tapered (c) Twisted

Technique

In 1986, Sederberg and Parry (1986) generalised Barr's technique and introduced the idea of FFD as we know it today. This technique involves deforming solid geometric

models in a free-form manner via a structural hyper-patch. For illustrative purposes, a simplified way to implement this technique is to impose a 4x4x4 parallelepiped lattice of control points, aligned with the x, y, and z coordinate axes, around the model to be deformed. Once this has been done, a relationship must be established between the control points and the model. Sederberg and Parry's FFD is defined in terms of a trivariate Bernstein polynomial given by

$$P(u,v,w) = \sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{k=0}^{3} p_{ijk} B_i^3(u) B_j^3(v) B_k^3(w) \tag{2.12}$$

where $P(u,v,w)$ are the model points that exist in the parameterised lattice-space, p_{ijk} are the lattice control points, and the remaining functions form a tensor product of univariate cubic Bernstein polynomials, e.g., B_i^3. For this example, cubic polynomials are chosen.

Each $P(x,y,z)$ point of the model must be re-expressed in its parametric form, $P(u,v,w)$, which in this case is simply a matter of normalising the x, y, and z coordinates. After any control point adjustment, every point of the model is simply mapped through this function to find its new position (see Figure 2.16). Animation can be achieved by interpolating over the passage of time between one set of control points and another.

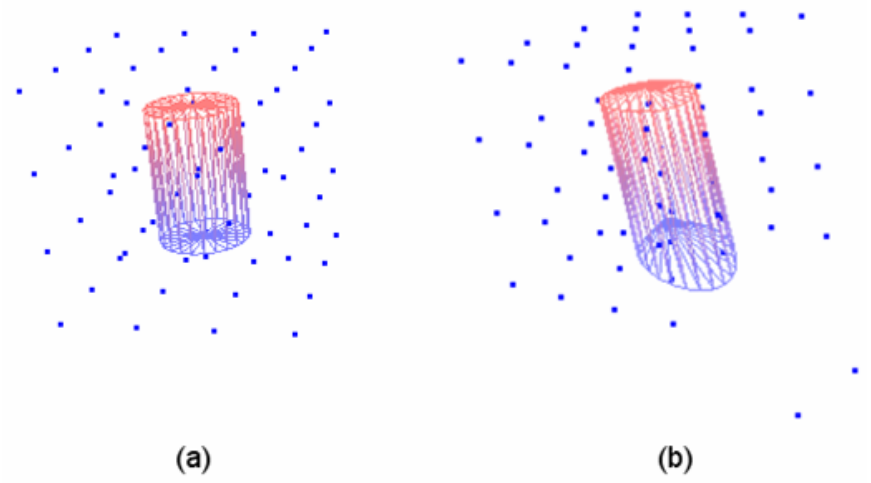


Figure 2.16: Example of FFD: (a) Non-deformed model (b) Deformed model

Critique

Sederberg and Parry's original technique does not impose any restrictions on the degree of polynomials used to construct the hyper-patch, or on the lattice size and orientation. Many researchers have taken advantage of different curve properties and have reformulated the technique using B-Spline curves, NURBS curves and subdivision volumes (Kalra et al., 1992, Lamousin and Waggenspack Jr., 1994, MacCracken and Joy, 1996). However, for all cases, the technique is the same.

FFD has the advantage of being a very general technique that is not limited to any one type of object. However this advantage is coupled with the disadvantage that the deformation takes no account of the geometry of the object. This can make it difficult to control the locality of deformations. FFD lattices must be applied in a piecewise manner to achieve local deformations. It has also been argued that the manipulation of the model space to achieve deformations is not as intuitive as direct manipulation of the model itself. Another disadvantage of Sederberg and Parry's FFD formulation is the restriction on the lattice shape to be parallelepiped. Structures like circular bumps are almost impossible to achieve using this technique and the number of control points to be moved to define a deformation depends on size rather than shape. However, subsequent research efforts, outlined below, have resolved some of these issues.

Coquillart (1990) developed the idea of Extended Free Form Deformation (EFFD), which solves the shape problem of FFD. The idea is to allow the user to define the shape of a lattice, which in turn determines/induces the shape of the deformation. MacCracken and Joy (1996) put forward the idea of Free-Form Deformation with Lattices of Arbitrary Topology. Hsu et al. (1992) described a way of manipulating the deformable object directly, leading to better control of the deformation, and a more intuitive interface than that of control point manipulation. There have been many other developments to address the issue of local deformation (Chang and Rockwood, 1994, Lazarus et al., 1994), to deal with distortions of polygon models Gain and Dodgson (1999), to examine alternative embedding spaces Hua and Qin (2003), to facilitate discontinuities Schein and Elber (2004), etc. An excellent survey of FFD techniques can be found in Gain and Bechmann (2008).

Many of the limitation issues associated with the original Sederberg and Parry formulation have now been resolved. Additionally, similar 2D, 1-D (Lazarus et al., 1994), and 0-D (Borrel, 1994) spatial deformation tools have been defined for surface-based, axial and point-based deformations respectively. Today, FFD facilitates smooth deformations of arbitrary structures, provides local control over deformations, and provides a computationally efficient algorithm that is easy to implement. For this reason, most CAD systems still support FFD. However, the fact that FFD takes no account of the geometry of the object being deformed remains its biggest disadvantage and it is largely used in conjunction with other deformation techniques.

Applications

FFD is primarily used as a geometric sculpting tool in CAD applications (Gain and Marais, 2005). However, it is still used for surprisingly complex modelling tasks where one might expect to see physical modelling techniques employed, e.g., deforming models of the human face and simulating complex medical procedures in real-time (Sela et al., 2006, Kalra et al., 1991, Schein and Elber, 2004) (see Figure 2.17).

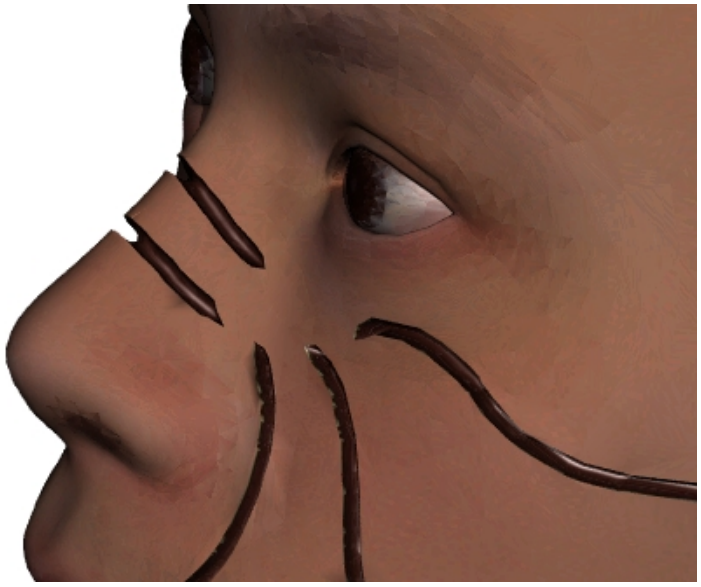


Figure 2.17: Discontinuous FFD - Modelling Surgical Simulations (Schein and Elber, 2004)

2.2.2 Physics-Based Techniques

As computational power has increased, so too have the capabilities of computer based modelling techniques. It is now possible to incorporate the principles of continuum mechanics within the geometric structure of a model for more convincing and realistic deformations.

Discrete Models: Mass-Spring-Damper Methods

As the title suggests, Mass-Spring-Damper (MSD) models represent physical bodies using a collection of point masses connected by springs in a lattice structure (Eberly and Shoemake, 2004).

Technique

MSD models are dynamic models. The spring forces acting on each node, F_s , are usually considered to be linear (Hookean). This means that the spring force has a magnitude directly proportional to the displacement of the spring from its rest position and acts to restore the spring to its rest position. Damping is then introduced to the system to account for the resistance to motion (due to friction, etc.). The damping force, F_d, on each node, represented schematically by a dashpot, has a magnitude proportional to that of the velocity of the node but opposite in direction to it. Without this force, a perturbed system would oscillate forever. The spring and damping forces are given by

$$F_s = -ku \tag{2.13}$$

$$F_d = -\gamma \dot{u} \tag{2.14}$$

where u is the displacement of the mass points, k is the spring constant chosen to approximate the material stiffness, and γ represents the damping constant (see Figure 2.18).

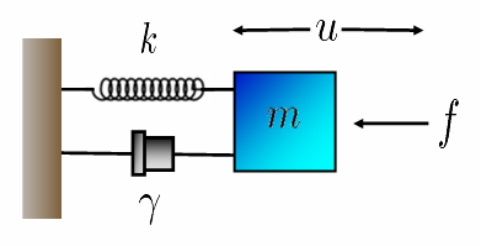


Figure 2.18: MSD Model

Combining the spring, damping, and external forces and applying Newton's second law of motion to an isolated MSD element gives

$$m\ddot{u} = -\gamma\dot{u} - ku + f \tag{2.15}$$

Thus, in an MSD system, where each node is part of a lattice of MSD elements (see Figure 2.19), the displacement, u_i, of a node i , due to its j neighbouring nodes and the external force, f_i, is governed by

$$m_i\ddot{u}_i = -\gamma_i\dot{u}_i - \sum_j k_{ij}u_{ij} + f_i \tag{2.16}$$

where m_i is the mass of the node, γ_i, its damping constant, and k_{ij} and u_{ij}, the spring constants and the spring displacements, respectively, between node i and its j neighbouring nodes.

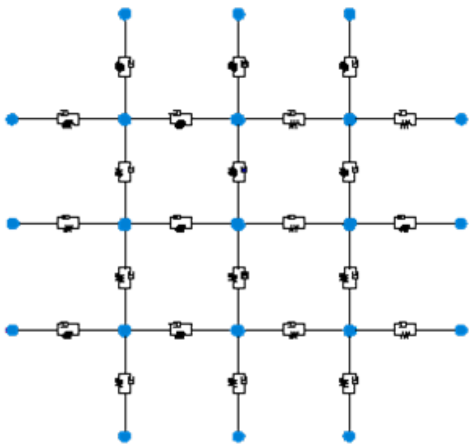


Figure 2.19: MSD Lattice

Assembling all of the individual masses in the lattice, the equations of motion for the entire system become

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{C}\dot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{f} \tag{2.17}$$

where **M**, **C**, and **K** are the mass, damping and stiffness matrices respectively for the deformable object, **u** is the vector of node displacements, and **f** is the vector of applied forces (Eberly and Shoemake, 2004).

In deformable modelling, objects are discretised by a network of mass-spring-damper elements, and deformations are simulated by the dynamics of the mass points and springs. There are various numerical techniques, e.g., Euler integration schemes, available for solving this system and evolving the deformation through time. Most comprehensive numerical methods textbooks cover these techniques.

Critique

MSD systems are intuitive, generally easy to implement, and are computationally efficient, making real time animations possible. Such systems handle even large topological deformations with relative ease. They are also well suited to parallel computation due to the local nature of interactions.

The main drawback associated with using MSD systems is that the discrete model imposes significant approximations of the true physics that would occur in a continuous body. This approximation of elasticity theory means that it can be difficult to define the spring parameters that result in convincing simulations. MSD systems also suffer from a problem known as 'stiffness', where large spring constants lead to poor system stability. To ensure stability, the system must be integrated over small time-steps, resulting in very slow simulations. On a more general note, the local nature of interactions means that, for global deformations, there is a delay in the propagation of force effects through a given system. Finally, MSD systems have a tendency to oscillate due to their iterative basis.

There has been much research carried out in recent years in an effort to mitigate the accuracy issues in MSD systems. However, most efforts have been application specific and have resulted in hybrid models where the best features of a number of deformable modelling techniques are combined (Etzmuß et al., 2003, Chadwick et al., 1989). Other research has focused on improving simulation times, while ensuring stability (Hauth and Etzmuß, 2001, Baraff and Witkin, 1998). In much of the literature, MSD systems are generalised as sets of point masses with fixed connectivity. Energy equations can then be defined for each point, in a similar fashion to that described below for Finite Element Methods, which can be minimised to find new point positions. These systems are known as Particle Systems, and are well suited for use with rectangular meshes (Etzmuß et al., 2003).

Applications

MSD models are used for a wide range of modelling applications. They are especially important for applications where there is a need for compromise between physical

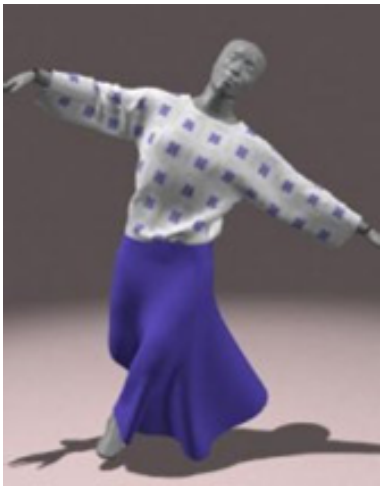


Figure 2.20: MSD Cloth Model (Baraff and Witkin, 1998)

accuracy and computation times. They have seen use in medical applications for soft tissue simulations and deformations (Kuhnapfel et al., 2000). However, they are more commonly associated with cloth simulation (Baraff and Witkin, 1998, Eberhardt et al., 1996, Etzmuß et al., 2003) (see Figure 2.20), and facial modelling and animation (Platt and Badler, 1981, Waters, 1987).

Continuum Models: Finite Element Methods

FEM are widely used mathematical techniques for finding approximate solutions to PDEs (Hunter and Pullan, 2005). Most physical phenomena can be modelled using differential and integral equations. However, the range of calculus equations that can be solved analytically is quite restrictive, making exact solutions very difficult to obtain. This problem was tackled by introducing numerical techniques where an approximate numerical solution is obtained at discrete values of the independent variables. These methods were once considered prohibitively time-consuming. However, modern computers have revolutionised the field of numerical methods and have facilitated the processing of large problems that once lay beyond reach (Logan, 2011).

The FEM is similar to the long established, and widely used, Finite Difference Method (FDM) (Mitchell and Griffiths, 1980). However, FEM is more suitable for dealing with complex irregular geometries. An excellent introductory text is provided in Hunter and Pullan (2005).

Technique

Traditionally, differential equations are solved numerically using FDM (Mitchell and Griffiths, 1980). Here, the domain of a given problem is subdivided, resulting in a finite mesh of nodes arranged in orthogonal rows and columns. Differential equations are approximated at the mesh nodes by replacing the continuous derivatives with finite difference approximations. These approximations are derived from a simple truncation

of a Taylor series expansion and reduce the problem to a set of algebraic equations made up of simple quotients of differences between node values.

The FDM approximation for a 2-variable system is given by

$$\left.\frac{\partial f(x,y)}{\partial x}\right|_{ij} \approx \frac{-f_{i-1,j} + f_{i+1,j}}{2h} \tag{2.18}$$

$$\left.\frac{\partial f(x,y)}{\partial y}\right|_{ij} \approx \frac{-f_{i,j-1} + f_{i,j+1}}{2h} \tag{2.19}$$

where f_i represents the magnitude of interest at the i^{th} node and is the characteristic dimension of discretisation, h. A global approximation then depends on a finite set of data specified at the boundary nodes. Using this data, a set of simultaneous equations in terms of the unknown nodal values can be generated and solved.

In the case of deformable modelling, continuum mechanics provide the differential equation and the deformable object becomes its domain. The PDE governing dynamic elastic materials is given by

$$\rho\ddot{u} = \nabla \cdot \sigma + f \tag{2.20}$$

where ρ is the density of the material, f is the externally applied forces, $\nabla \cdot \sigma$ is the internal force resulting from a deformed infinitesimal volume, and $\ddot{u}$ is the node acceleration function.

Although the FDM technique is simple to implement, it is difficult to approximate the boundary of irregular geometries with a regular mesh. This problem is addressed by the more computationally expensive FEM technique.

The FEM technique works by subdividing the domain of the problem, resulting in a discrete mesh of smaller parts known as finite elements (Hunter and Pullan, 2005). Both FEM and FDM techniques aim to reduce a complicated differential equation to a set of algebraic equations that can be solved numerically. The difference lies in how the sub-regions are used. While FDM approximates the differential equation at discrete nodes, FEM approximates the equation's solution.

With FEM, for the 1-D case with n nodes, a local approximation, $\tilde{f}_i(x)$, of the solution, $f(x)$, is defined over each element and is represented as a function of the values of the element nodes. Typically, these functions are a set of weighted polynomial basis functions, known as local shape functions. Assembling the local solutions gives a global approximation, $\tilde{f}(x)$, which can be substituted into the original differential equation. This results in a set of algebraic equations

$$\tilde{f}(x) = \sum_{i=0}^{n} a_i \tilde{f}_i(x) \tag{2.21}$$

The approximation, brought about by the discretisation of the domain, generates an error when substituted into the original differential equation. This error is known as the residual, and is denoted by $R(x)$. Rather than solve the differential equation directly for the unknown a_i values, the Galerkin method (Strang, 1986) of weighted residuals formulates an optimisation problem which aims to find the a_i values that minimise the residual

$$\int_D w(x)R(x) = 0 \tag{2.22}$$

where D is the domain of the problem, $R(x)$ is the error function or residual, and $w(x)$ is a set of weighting functions. To ensure that the minimum residue is achieved, the functions are carefully chosen to force $R(x)$ to be orthogonal to the space of functions used to approximate the solution.

This technique results in a set of simultaneous equations in terms of the unknown nodal values. With the application of boundary conditions, this set of equations can be solved. For simple linear and static cases, the resulting system of equations is of the form

$$\mathbf{Ku} = \mathbf{f} \tag{2.23}$$

where **u** is the vector describing the unknown magnitudes of interest, **f** is the vector describing all known values in the system, and **K** is the matrix relating the two. The overall technique is the same for higher dimensional cases.

For deformable modelling problems, a displacement-based FEM model is used. For the static case, the residual can be formulated using

$$R = \nabla \cdot \sigma + f \tag{2.24}$$

and the system can be reduced to

$$\mathbf{Ku} = \mathbf{f} \tag{2.25}$$

For dynamic simulations, inertia and damping effects must be considered in the system

$$M\ddot{u} + C\dot{u} + \boxed{Ku = f}$$

Original static system

where **M**, **C**, and **K** are the mass, damping and stiffness matrices, respectively for the deformable object, **u** is the vector of unknown displacements, and **f** is the vector of applied forces.

Alternative Technique

The method of weighted residuals applied to continuum mechanics problems can be interpreted as a variational problem of energy minimization, and many papers opt for this simpler form. Here, a solution to the differential equation is obtained by translating it into a minimisation problem for an energy function, E, which is a linear functional defined on a function space Ω. The solution is then found by solving for

$$u \in \Omega \text{ so that } E(u) \leq E(v) \text{for all functions } v \in \Omega$$

For displacement based FEM, the function v corresponds to the continuously changing displacements of the elastic body and E is an energy equilibrium equation, defined in terms of material displacement over the deformable object continuum.

The goal is to apply the FEM subdivision technique (as for the Galerkin method (Strang, 1986)), to the space, Ω. The energy function for each element is then considered, and the resulting system of equations is solved for the nodal displacements that minimise the total potential energy of the deformable object. This is achieved by setting all partial derivatives of the potential energy equal to zero. The energy formulations generally assume linear behaviour but this approach is only valid for small deformations.

The finite elements and interpolation functions chosen depend on the geometry and the various requirements with respect to accuracy, convergence, degrees of freedom, computation, etc. There are usually trade-offs to be considered. Most general FEM textbooks cover a range of possible element choices and suitable interpolation equations (Segerlind, 1984) (see Figure 2.21).

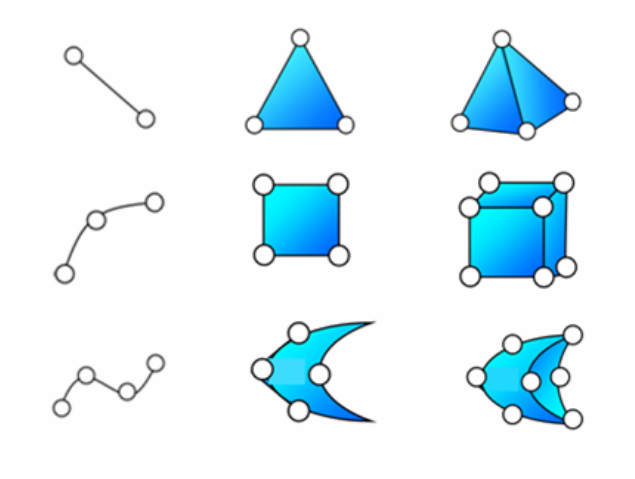


Figure 2.21: Finite Element Shapes

Critique

Unlike MSD techniques, rather than starting with a discrete model, FEM techniques start with the PDE and subsequently discretise it in space. Thus, FEM models produce

more physically realistic simulations and can be used to model even complex soft tissue deformations and non-linear material properties accurately. However, the use of these systems is severely limited by heavy computational requirements. Thus far, FEM techniques have proven to be impractical for real time applications. Because the force vectors, in addition to the mass and stiffness matrices, are computed by integrating over the object through time, they must be re-evaluated at each time-step, as the object deforms. This requires significant processing time. If small deformations are presupposed, and the topology over the time interval is constant, this processing time can be significantly reduced. In this case, linear elastic behaviour can be assumed and the mass and stiffness matrices can be assumed to remain constant. This 'small deformation' assumption, however, does not hold for 'soft' bodies.

FEM models are well established and are the state of the art when it comes to accurate modelling of complex physical deformations. Ongoing areas of research include exploring appropriate shape functions (N. Sukumar, 2006) and facilitating discontinuities across element boudnaries (P. Kaufmann, 2009). Most of the research effort has, however, been devoted to speeding up the FEM process in an effort to lift the restrictions on its use.

Finite Volume Method (FVM) (Teran et al., 2003), and Boundary Element Method (BEM) James and Pai (1999), have been developed to solve for relevant forces more directly. The FVM technique does this by solving for the nodal forces of an element by distributing the internal force per unit area, calculated from the stress tensor with respect to a given plane, evenly among the nodes of each element. Teran et al. (2003) use this technique for simulating skeletal muscle. For the BEM technique, the volume integral of the equation of motion is transformed to a surface integral using the Green-Gauss theorem. This procedure only works for structures with interiors made up of homogenous material. Although these techniques speed up the FEM process, they limit the types of deformations achievable.

Other techniques, including condensation (Bro-Nielsen and Cotin, 1996), preprocessing (Cotin et al., 1996), and modal analysis (James and Pai, 1999), have been used to simplify complex FEM problems. Condensation techniques involve simulating only those visible surface nodes in a similar fashion to BEM. Preprocessing techniques not only assume linearity, but also take advantage of superposition. Here, the reaction of each node for an infinitesimal force is pre-calculated. Applied forces are then expressed as sums of these infinitesimal forces. The stored results are then superimposed to estimate the global deformation. This is not suitable for large deformations and topological changes cannot be considered. Finally, modal analysis involves using only the most significant vibration modes of the object to compute the deformation field for applied forces. Much of the research effort in recent years has focused on hardware acceleration techniques (El-Kurdi et al., 2007, Liu et al., 2008)

In spite of these advances computationally expensive FEM techniques are still widely considered impractical for use in general modelling.

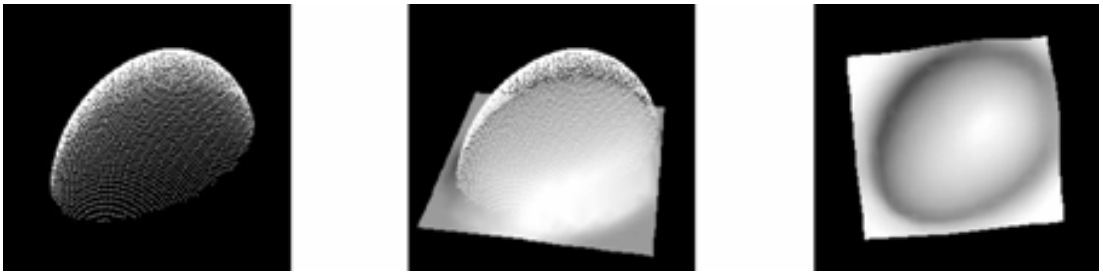


Figure 2.22: Model Fitting using FEM (McInerney, 1992)

Applications

Because of the computational overhead associated with FEM techniques, they are generally reserved for applications where physical realism is essential, e.g., FEM for medical applications. However, they have been used for a broad range of applications: shape editing in computer aided design (Celniker and Gossard, 1991), virtual cloth modelling (Collier et al., 1991), deformation of muscles and other objects (Chen and Zeltzer, 1992), shape fitting and tracking (Essa et al., 1993), simulating brittle and ductile fracture for elastoplastic materials (O'Brien and Hodgins, 1999), surgical simulation, fitting deformable models to 3D data (McInerney, 1992) (see Figure 2.22), and countless other applications.

IgA

IgA is a recent non-standard numerical method for solving partial differential equations involving parametric CAD representations such as splines (Hughes et al., 2005, Cottrell et al., 2009, Hughes et al., 2010). Unlike FEM techniques that discretise the geometry for the solver, the ultimate goal of IgA is to directly adopt the parameterised geometry description for the PDE analysis.

Technique

In IgA the parametric representation of a B-Spline/NURBS curve is divided into finite pieces[2]. The nature of B-Spline/NURBS representations, given their local control properties, means that an exact analytic description of each finite piece is available (See Section 2.1.4 for parametric spline function descriptions). Rather than approximate this function with a set of standard finite element shape functions, the premise of IgA is to use the available local analytic descriptions directly. The resulting PDE is solved numerically using Gaussian Quadrature techniques (Press et al., 2007).

Critique

Finite Element Analysis is very much the *de facto* standard in industry. However, the emergence of IgA has questioned its suitability for dealing with B-Spline/NURBS representations. Finite Elements existed long before B-Spline/NURBS representations and

[2]The word 'piece' is adopted rather than element to avoid confusion with FEM terminology

it may take some time before IgA becomes standard. There is much evidence of its growing popularity and its emergence has sparked a renewed interest in B-Spline/NURBS research after an arguably slow period. The main benefit of the technique is that it removes the need for computationally expensive conversions from CAD design representations to FEM analysis representations. Additionally, access to an "exact" mathematical description of the model is available throughout the deformation process.

Applications

IgA is a custom technique specific to dealing with the parametric geometries standard in CAD.

Mesh-free Methods

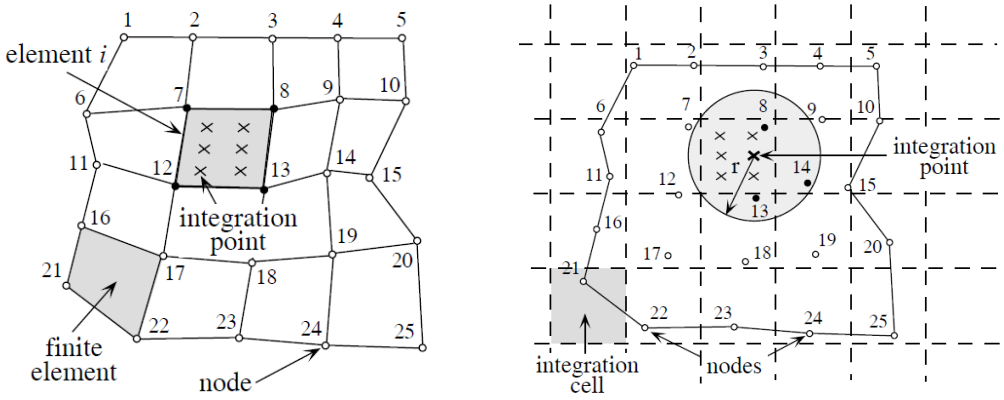
Relatively new to the field of computer graphics shape deformation, 'Mesh-free', also known as 'Meshless', Methods are mathematical techniques for finding approximate solutions to PDEs. The techniques of FEM, FVM, and BEM, discussed in Section 2.2.2, rely on a grid or a mesh. Meshfree methods in contrast use the geometry of the simulated object directly for calculations. As discussed in Section 2.1.8, Meshfree representations have become an important alternative to the traditional polygonal representation in certain cases involving large deformations of complex models.

Technique

Instead of using a pre-defined mesh, only point generation is used in mesh-free simulations. Inter-point connectivity is not defined. Since the mesh-free method does not describe the point topology explicitly, interactions depend on distance rather than on a specified graph of connections.

Just as the surface values of a mesh-free shape representation can be interpolated, so too can the values of the deformation function. For the mesh-free case, the approximation is based on nodes, not elements. For each point a shape function is created much like in FEM techniques. The value of a property at any given location requires interpolating values between neighbouring nodes. Each node represents a 'source' in a smoothed, localised field. The field represents the material properties, such as density or velocity. In order to find the value of the field at a given location or integration point, the contributions to that field due to all nodes in the vicinity of that location are combined. Usually, several nodes, each with a radius of influence, contribute to the field at each integration point (See Figure 2.23).

The underlying mathematics involves a Moving Least Squares approximation of the gradient of the displacement vector field and an element-free version of the Galerkin method described in Section 2.2.2 (Nayroles et al., 1992, Belytschko et al., 1994, Cingoski et al., 1998). A good introductory text can be found in Liu (2003).



(a) (b)

Figure 2.23: Meshfree Models: (a) FEM (b) Meshfree (Cingoski et al., 1998)

Critique

FEM techniques are the *de facto* standard in accurate physics based modelling of deformations. However, in circumstances where the cost and complexity of generating a finite element mesh from the sample points outweigh the benefits, the mesh-free methods can be more appropriate. In situations where mesh-free models representing water, trees, clouds or smoke etc. are to be deformed, mesh-free deformation techniques lend themselves better than FEM techniques (Blinn, 1982*a*, Reeves, 1983).

The topological flexibility of the mesh-free technique means that it is more appropriate than FEM techniques in situations where a deformation results in the underlying mesh becoming extremely skewed or compressed as adaptive remeshing and node refinement must take place throughout the deformation to prevent severe distortion of the elements. For such large three-dimensional problems which are becoming more common (see Section 2.1.8), the computational cost of re-meshing at each step often becomes prohibitively expensive.

A disadvantage of using the technique is that the underlying point-based subdivision of the domain of the partial differential equation means that values are calculated based on several neighbouring values. This results in a smoothing effect and restricts the modelling of sharp edges in models.

Applications

Early use of mesh-free techniques was in simulating the dynamics of fluid and smoke particle systems. However, as discussed, they are now finding use as an alternative to FEM techniques in large or complex data sets where triangulation and complex deformations that severely alter the geometry of the original model become prohibitively expensive for standard FEM techniques.

Active Contour Models/Snakes

An ACM is an energy minimising curve, often defined by splines, whose energy equation is structured in such a way as to achieve a desired deformation. These models are often called 'snakes', as they appear to slither across images. They are an example of the general technique of matching a deformable model to an image using energy minimisation. Snakes were originally introduced by Kass et al. (1987) for solving various computer vision problems, such as edge and contour detection, and motion tracking in images.

Technique

The deformation of Kass' snake is governed by an energy equilibrium equation which considers internal snake-forces that resist stretching and bending, and external snake-forces, including local image-generated forces and constraint forces. These constraint forces are usually supplied by the user. The snake's energy thus depends on its shape, location within the image, and any user input.

The snake is defined as a parametric contour, $v(s)$, where the curve parameter is typically normalised, i.e., $s \in [0, 1]$. The total energy of the snake, E_{snake}, is then given by the functional

$$E_{snake}(v(s)) = E_{int}(v(s)) + E_{im}(v(s)) + E_{con}(v(s)) \tag{2.26}$$

The internal energy is given by

$$E_{int}(v(s)) = \int_0^1 \alpha \left| \frac{dv}{ds}(s) \right|^2 + \beta \left| \frac{d^2v}{ds^2}(s) \right|^2 ds \tag{2.27}$$

The first term in this equation imposes tension on the curve and the second term imposes rigidity. These terms can be weighted using the α and β parameters to balance the two effects.

The image energy is formulated to attract the snake towards features of interest. Thus, a potential function is chosen, whose minima coincides with the interesting feature of the image. This function is of the form

$$E_{im}(v(s)) = \int_0^1 P(v(s))\, ds \tag{2.28}$$

The simplest potential energy is the image intensity

$$P(v(s)) = I(v(s)) \tag{2.29}$$

According to the sign of I, the snake will be attracted to either light or dark regions in the image. Optional external energies can be added at this point to impose constraints defined by the user.

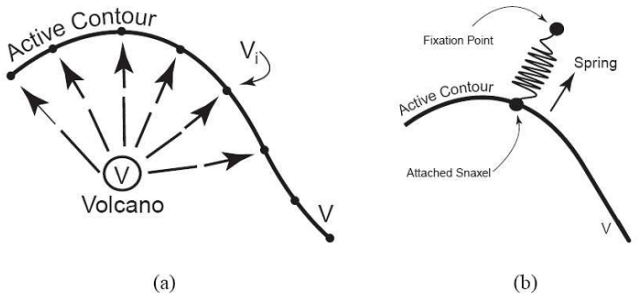


Figure 2.24: ACM Springs and Volcanoes (a) the volcano pushing the contour away, and (b) the spring force attached to a contour control point (Molloy and Whelan, 2000)

The original formulation of Kass et al. implemented user-imposed constraints as springs and volcanoes that provided forces that pushed or pulled the snake toward or away from certain features or regions respectively. This idea is illustrated in Figure 2.24.

The system is then solved to find the contour that minimises the total energy.

$$\begin{aligned} E_{snake}(v(s)) &= \int_0^1 \left[\alpha \left| \frac{dv}{ds}(s) \right|^2 + \beta \left| \frac{d^2v}{ds^2}(s) \right|^2 + P(v(s)) \right] ds \\ &= \int_0^1 F(v(s))\, ds \end{aligned} \tag{2.30}$$

From the study of calculus of variations, it is known that the solution to this problem must satisfy its Euler-Lagrange equation given by

$$\frac{dF}{dv} - \frac{d}{ds}\left(\frac{dF}{\delta \dot{v}}\right) + \frac{d^2}{ds^2}\left(\frac{dF}{d\ddot{v}}\right) = 0 \tag{2.31}$$

This becomes

$$-\alpha \frac{d^2v}{ds^2} + \beta \frac{d^4v}{ds^4} = -\nabla P(s) \tag{2.32}$$

which can be solved numerically using finite differences for a local solution. The global solution can be found by making the position of the snake a function of time and solving iteratively using implicit/explicit Euler integration schemes. A good introductory text can be found in Ivins and Porrill (2000).

Active Contours can be extended to two (surface) and three (volume) dimensions.

Critique

One of the chief virtues of snake representations is the ability to specify a wide range of snake properties through the energy function, by analogy with physical systems. 2D snakes have been generalised to active surfaces and active volumes, which can be used to smoothly fit surfaces or meshes to 3D image data (Cohen, 1991, Cohen and Cohen,

1993, Cohen, 1996, Ahlberg, 1996). Due to their reliance on calculus of variations, such models are often referred to as variational models. These techniques can be used to aid the creation of 3D models for visualisation purposes. The introduction of topologically adaptive snakes (McInerney and Terzopoulos, 2000), and the incorporation of a-priori image information for 'smart snakes' such as active templates and shape models (Cootes et al., 2001), also stand out in the literature.

The chief limitation of Kass' formulation of snakes was its reliance on the user to interactively initialise the snake by placing it somewhere near the desired feature. Cohen improved this situation by altering Kass' energy formulations (Cohen, 1991). This extension of the snake-concept, called balloons, added an inflation force to a closed contour to make the contour bypass irrelevant elements in the image. Xu and Prince (1997) introduced Gradient Vector Flow (GVF) snakes. These models define a force field computed from the spatial diffusion of the image edge map gradient. This computation causes diffuse forces to exist far from the feature of interest and crisp force vectors to exist near its edges. This allows the snake to start far away from the feature of interest and yet have the ability to draw it towards the object and force it into the boundary concavities.

A host of statistically based models exists in this category. However, these models require a-priori knowledge about the images for analysis and are therefore beyond the scope of this review. Further information can be found in McInerney and Terzopoulos (1996).

Applications

ACMs have been used for a wide variety of image processing tasks, including image segmentation, registration, model fitting, and motion tracking (Richens et al., 1992, Vieren et al., 1995, Ferrier et al., 1994, Derraz et al., 2004)(see Figure 2.25 for an example of image segmentation).

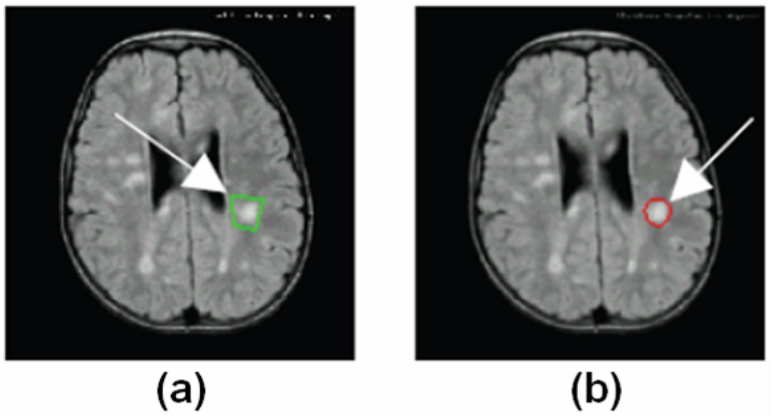


Figure 2.25: Active Contour Segmentation of a Tumour in MRI Brain Scan (a) Initial Placement (b) Final Position (Derraz et al., 2004)

Elastically Deformable Models

Terzopoulos et al. (1987) generalise Kass' energy minimising curves to 'Elastically Deformable Models' in the form of curves, surfaces and solids for use in 3D animations. Their model is regarded as seminal in computer graphics deformable modelling literature and they coined the term 'Deformable model'.

Technique

The technique generalises Kass' energy minimising curve to 2D and 3D. The internal energy of an elastically deformable surface can be specified by

$$E_{surface}(v(s,r)) = \int_0^1 \int_0^1 \alpha_s \left|\frac{\partial v}{\partial s}\right|^2 + \alpha_r \left|\frac{\partial v}{\partial r}\right|^2 + \beta_s \left|\frac{\partial^2 v}{\partial s^2}\right|^2 + \beta_r \left|\frac{\partial^2 v}{\partial r^2}\right|^2 + \beta_{sr} \left|\frac{\partial^2 v}{\partial s \partial r}\right|^2 + P(v) \quad dsdr \tag{2.33}$$

Here, α_s and α_r determine the elasticity along the s and r axes respectively, while β_s and β_r determine the corresponding rigidities. β_{sr} determines the resistance to twist. The energy is minimised by finding the surface, $v(s,r)$, that satisfies the corresponding Euler-Lagrange equation given by

$$-\alpha_s \frac{\partial^2 v}{\partial s^2} - \alpha_r \frac{\partial^2 v}{\partial r^2} + \beta_s \frac{\partial^4 v}{\partial s^4} + \beta_r \frac{\partial^4 v}{\partial r^4} + \beta_{sr} \frac{\partial^4 v}{\partial s^2 \partial r^2} = -\nabla P(v) \tag{2.34}$$

For use in general computer graphics applications, Terzopoulos et al. incorporate mass and damping terms. The difference between the deformation energy of the rest shape and the deformed shape, as calculated above, is then minimised. The dynamic equations can then be solved using Finite Difference or Finite Element Methods.

Critique

Much like ACM/Snake models, a wide range of properties can be specified through the energy function, by analogy with physical systems. Various energy minimisation functions have been employed to achieve such deformations. Botsch and Sorkine (2008) present a detailed survey of linear techniques employed. Additionally, Terzopoulos and Fleischer (1988) model non-linear effects through a variation of the model that can capture inelastic deformations.

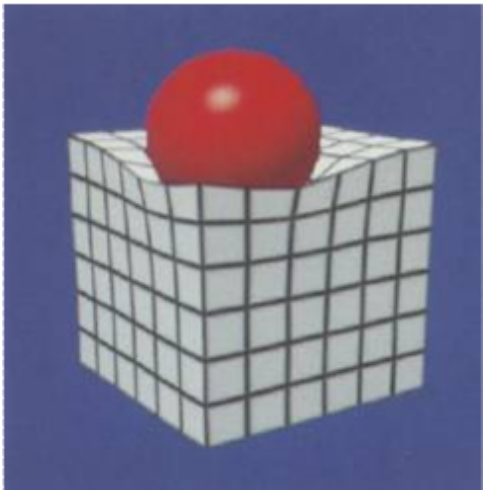


Figure 2.26: Elastically Deformable Solid (Terzopoulos et al., 1987)

Applications

By simulating properties such as tension and rigidity, in curves, surfaces and solids, Elastically Deformable Models capture the elastic behaviours exhibited by a wide range of deformable objects such as string, rubber, cloth, paper, and flexible materials. Furthermore, by including physical properties such as mass and damping, they simulate the dynamics of such objects. The work focuses on animation and simulation applications. Figure 2.26 shows an example of an elastically deformable solid.

2.2.3 Discussion

Purely geometric deformation techniques are faster and are generally simpler to implement than their physically based competitors. Although these models tend to be geometrically and computationally efficient, there is a heavy reliance on the skill of the designer as deformations are achieved by manually adjusting control points and shape parameters. Often, these adjustments are non-intuitive and a large number of control points must be adjusted in order to achieve perceptually simple deformations. In addition, they do not simulate the underlying mechanics of a given deformation. Despite these shortcomings, FFD techniques have remained popular in CAD and computer-based freeform modelling environments, mainly due to their generality and low computational overheads.

Sophisticated, physics-based models, although necessary for simulating the dynamics of realistic interactions, are not yet suitable for fully interactive real-time simulation of multiple objects in virtual environments due to the current limitations of computational power.

Research indicates that MSD systems are the most widely used deformable models at present. It is likely that their popularity originally stemmed from the simplicity of obtaining equations and programming them. MSD models thus evolved rapidly and implementations for nearly every conceivable type of interaction have been developed.

This, unsurprisingly, only increased their popularity. The main advantage of the MSD model over its competitors today is that it can approximate physical realism at real-time rates. Although the performance of modern computers and graphics hardware has made these physically-based animations possible in real-time, such techniques are not ideally suited to 3D shape modelling applications due to their complexity and computational overhead.

The Finite Element Methods are state of the art in physically based modelling and simulation and outperform all competitors in terms of accurate modelling of continuum mechanics in complex geometries. They also provide the most flexible platform with respect to overcoming limitations generally associated with modelling complex boundary conditions and material types. It is foreseeable that in coming years, the processing speed of desktop computers will continue to improve and the time will come when other deformable models with higher physical realism, such as FEM models, will gain in popularity. The shear amount of research being carried out with FEM models in areas such as surgical simulation and computer graphics, where real-time deformations are highly desirable, is evidence enough that researchers are working and progressing towards this time. IgA has made headway against FEM in the area of physics-based simulations involving B-Spline/NURBS based representations, where it is likely to become the new standard in CAD and CAGD.

ACMs fundamentally differ from the other deformable modelling techniques discussed in that they represent a generalised approach for matching a deformable model to an image, rather than a generic deformable modelling technique suitable for computer graphics modelling and deformation. These intuitive models offer the geometric and computational efficiency of the non physics-based models, while incorporating some of the realism achieved using those that are physics-based. Such a compromise would be very attractive in an interactive computer graphics free-form modelling environment.

2.3 Virtual Sculpting

'Virtual Sculpting' has become one of the most important tools for interactive 3D content creation for use in the film industry, special effects, computer games, etc. Introduced by Parent (1977), interactive shape design environments based on a clay sculpting metaphor have long been touted as a natural and intuitive solution to the complex task of freeform shape design.

Freeform modelling is a significant sub-discipline of computer graphics concerned with the computer-based design of 3D shapes. In contrast to functional modelling, freeform modelling is used in situations where the primary consideration is the aesthetics of the final shape, rather than functional aspects such as aerodynamics, volume or stress response. For this reason, freeform models are not generally subjected to the same plethora of constraints associated with functional modelling. A freeform design environment can take advantage of this freedom from constraints and afford a some-

what more blasé attitude towards the rigorous physics-based constraints associated with functional modelling. Virtual Sculpting is therefore primarily used in smooth organic modelling as opposed to hard surface modelling.

Virtual Sculpting environments based on the clay sculpting paradigm rely heavily on the shape modelling and deformation techniques discussed in Section 2.1 and Section 2.2. Such environments can generally be characterised in terms of the underlying shape representations and the laws governing shape deformations. Anticipating the effects of a deformation during a sculpting session requires not only experience and understanding of the underlying mathematics associated with the deformation, but also familiarity with the shape representations of the models being deformed. Thus, deformation and shape modelling techniques alone do not offer designers an intuitive means for the definition of freeform shapes. As can be seen in Section 2.2, such techniques are generally applied to the problems of editing existing shapes or creating animations, rather than to the problem of interactive design.

A true sculpting metaphor should allow a designer to create a model using a series of simple real-time modelling gestures. Overcoming the need for manually controlling a large number of variables has been the goal for researchers in the area of Virtual Sculpting. Much of the research towards achieving this goal has focused on combining the various techniques for shape representation and deformation with an intuitive interactive modelling interface, providing tools for automatic editing of model parameters in response to intuitive user interactions.

This section explores and compares current approaches to virtual sculpting.

2.3.1 Geometric Sculpting Tools

This section discusses geometric approaches to Virtual Sculpting that do not incorporate principles from Continuum Mechanics directly. Rather, these approaches typically seek to mimic physical properties using geometric techniques.

Direct Model Editing

Model-based methods for interactive shape editing rely on the model-based deformation techniques discussed in Section 2.2.1. Although designers used such approaches in early modelling systems, creating and editing complex models by directly defining and editing, interactively or otherwise, the available parameters associated with a chosen shape representation can require time and effort and be far from intuitive. Systems relying on such techniques generally require skilled designers with a full understanding of the underlying shape representation allowing them to pre-empt the effects of altering model parameters in order to 'sculpt' the model effectively.

Decay Functions as Sculpting Tools

In Parent's seminal paper (Parent, 1977), sculpting involves the application of early geometric algorithms such as intersection, warping, bending, etc. Parent also introduced the use of decay functions in surface modelling. Here, the user attaches the

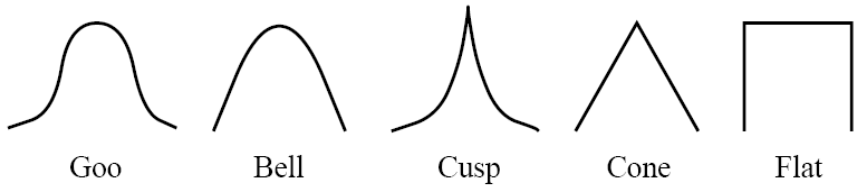


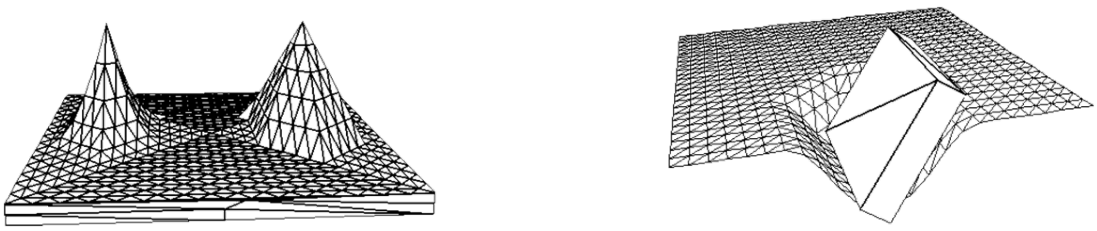
Figure 2.27: A sample set of decay functions (Bill and Lodha, 1994)

cursor to a surface point and repositions the point. A decreasing proportion of its movement is distributed across the mesh using a customisable interpolation routine. In this way, movements of vertices are propagated to surrounding vertices. In Parent's implementation, the displacement decays as a simple power function of the surface distance, which is calculated by counting the edges in the shortest path between a given vertex and the selected vertex.

Allan et al. (1989) extend decay functions by employing a Euclidean measure of surface distance and defining a variety of decay shapes (cone, bell, cusp, random, sine). They also allow vertices to be directly bound to the selected vertex movement or anchored in place.

In addition to providing decay functions (see Figure 2.27), Bill and Lodha (1994) devised shaped tools described by superquadrics, ranging from rounded boxes and cylinders to ellipses and spheres, which can simultaneously push or pull many vertices at once (see Figure 2.28). Collision detection between the mesh and the implicit surface tool is used to determine if points lie in, on, or outside the tool. In push mode, points 'in' the tool are translated in the direction of tool movement so that they are 'on' the tool, while in pull mode, points 'on' the tool are translated with the tool movement. Bill and Lodha (1994) also discuss adaptive smoothing and refinement operations for processing the sculpted mesh. Shadows of sculpting tool are generated and rendered on the mesh to enhance the sculpting experience.

While the basic operation of moving a point is central to interactive modelling, virtual tools, such as decay functions, offer a higher level of deformation control. They allow multiple primitive operations to be applied to a model simultaneously, in an intuitive fashion (Bill and Lodha, 1994).



(a) (b)

Figure 2.28: Bill and Lodha Virtual Sculpting: (a) Decay Function (b) Shaped Sculpting Tool: Box-shaped tool pushing downwards on flat mesh (Bill and Lodha, 1994)

Implicit Sculpting

Implicit surfaces are often used for Virtual Sculpting applications because of their ability to handle topological changes with relative ease. As discussed in Section 2.1, implicit surfaces are often used within CSG and blending environments where the visual effect of sculpting can be achieved with the successive Boolean combination or blending of implicit primitives. Using such techniques, the addition of each primitive causes the complexity of the shape to continually increase, to the point where real-time interaction can become untenable. This can occur even in situations where the designer is removing most of the material. An alternative option is to use a grid-based approach.

Introduced in the early 1990s by the seminal work of Galyean and Hughes (1991), grid-based sculpting allows the user to interactively add to, remove from, or smooth material represented by a discrete scalar field stored in a voxel grid or voxmap. This type of interaction is also conducive to copy-paste and cut-paste actions. As discussed in Section 2.1.7, a freeform solid can be approximately represented as a collection of discrete cube-shaped volume elements (voxels), called a voxmap. To continue the 3D pixel analogy, sculpting tools are applied to locally modify the voxmap and consequently the solid that it represents, much as 'paintbrush' tools affect a pixmap in traditional 2D paint programs.

In the approach of Galyean and Hughes (1991), the field values stored in the voxmap loosely correspond to the density of a virtual clay representing the sculpted object. In their implementation, a '1' indicates the presence of virtual material, while a '0' indicates an empty cell. The tool used to edit the object's field values, in this case a sphere primitive, is itself represented by the same form of density voxmap. The tool acts by combining its field contribution with the discrete field function that defines the sculpted object. Virtual material can be deposited or removed using additions and subtractions. Smoothing can be achieved by using a low pass filter which recomputes each field value covered by the tool as an average of itself and its closest neighbours. The object and tool surfaces used for display are defined as iso-surfaces of their respective density fields. These iso-surfaces can be converted into a triangular mesh using a marching cubes algorithm. An incremental marching cubes algorithm for the sculpted object, i.e., the marching process is only performed in the modified region.

Wang and Kaufman (1995) extended the technique of Galyean and Hughes (1991) to support more complex carving and sawing tools, which can be rotated with respect to the object . They also introduced voxels with attributes such as colour and texture. Ferley et al. (1999, 2000, 2001) extended Galyean and Hughes' toolset to include generic ellipsoidal primitives and resolution adaptive grids. Figure 2.29 shows the sculpting environment of Ferley et al.

In order to achieve a sculpting metaphor more akin to clay sculpting, Ferley et al. (2000) introduced local deformation tools with part-negative part-positive fields in order to compress the sculpted object in the area where the tool penetrates it, while

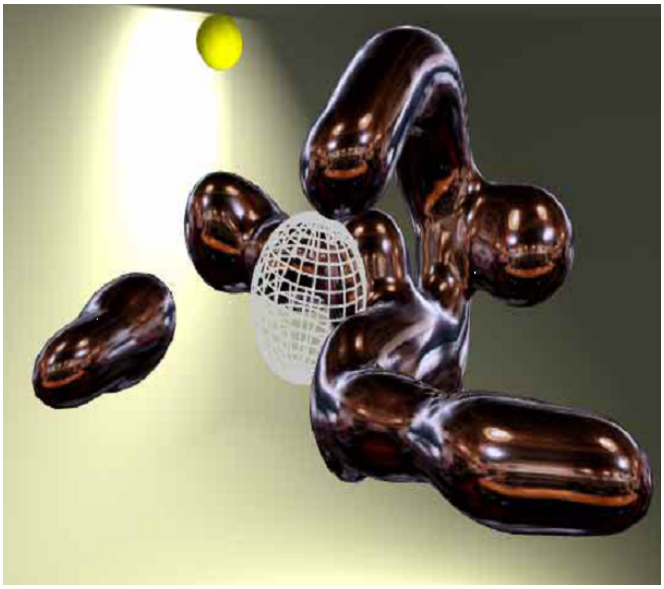


Figure 2.29: Implicit Sculpting: The sculpting tool is displayed in wireframe. (Ferley et al., 1999)

creating a bulge to imitate material displacement around the contact region (see Figure 2.30).

Dewaele and Cani (2004) extend the approaches of Galyean and Hughes and Ferley et al. by using a layered model that can provide 'Large-Scale Deformations', 'Volume Conservation' and 'Surface Tension' properties . As in Galyean and Hughes' approach, Dweaele et al. use a grid based representation for the material. In their implementation, the grid is populated with a density field that records the quantity of matter in each cell with values ranging from '0' (an empty cell) to '1' (a full cell). The surface of the material is defined as an iso-surface of the density field, usually chosen at a given threshold between '0' and '1'. Dewaele and Cani's 'Large-Scale Deformations' aim to provide support for global plastic deformations such as bending and twisting. This is achieved by defining suitable displacement fields to move matter along the material path, rather than Euclidean distances. Volume conservation is achieved by propagating any excess density in each cell to its nearest neighbours.

After several deformations using the two layers above, the model material has a

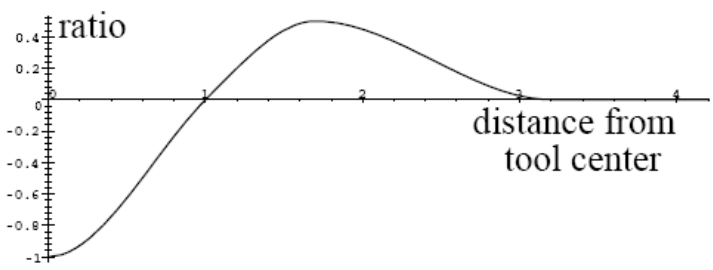


Figure 2.30: Shape of the deforming function. Horizontal axis is the distance from the tool center. Vertical axis is the ratio (Ferley et al., 1999).

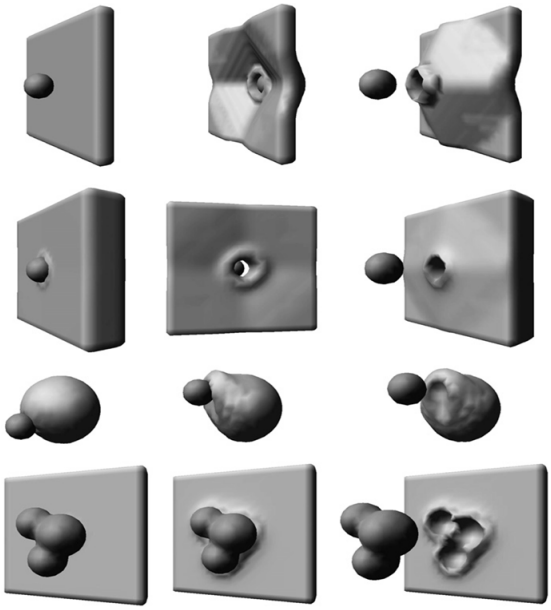


Figure 2.31: Holes, folds, and prints obtained with Dewaele and Cani's virtual clay model (Dewaele and Cani, 2004).

tendency to loosen and spread. Cells with low density appear as material is propagated. Those cells with density below the threshold iso-surface value are not visible. This can give the illusion that the material's volume is changing and can lead to strange effects of clay appearing seemingly from nowhere when a tool's action increases the cell density to above the threshold. The surface tension layer aims to limit these problems by keeping the gradient of density near the surface of the clay at an acceptable value. Matter in cells with very low densities is moved to nearby cells with higher densities along the gradient direction. This has the effect of mimicking surface tension and keeps the material more compact. Figure 2.31 shows some of their sculpting examples.

Much of the remaining literature involving grid-based sculpting is focused on providing suitable data structures for grid based representations that alleviate storage costs and facilitate field queries at interactive rates (Bærentzen, 1998).

Distance fields, as described in Section 2.3.1, have also been used within Virtual Sculpting applications (Perry and Gibson, 2001). However, a distance field requires global editing, as the value of closest distance may need to be updated in an unbounded region while editing the shape. In addition to this, distance fields are discontinuous in concave regions, so smooth approximations of them need to be computed to preserve a good level of continuity for the iso-surface (Cani et al., 2008). Density fields offer a more memory efficient option as empty cells do not need to store a value explicitly. The creation of intuitive editing tools is straightforward, and editing is local.

Other implicit representations such as the Blob Tree framework have also been successfully used in sculpting frameworks. Here, constructive solid geometry techniques facilitate the creation of heterogeneous objects with adaptive topology (Raviv and Elber, 1999, Schmitt et al., 2004). The disadvantage of these techniques is that everything is

stored in hierarchy which increases in depth with each modification. Local extraction of the surface can therefore become slow.

Unlike CSG, where primitives are stored individually in a binary tree, grid-based models based on spatial decomposition schemes are made up of a single set of cubic 'space' elements or voxels, each storing a scalar value such as intensity or distance. The immediate advantage is that the time taken to conduct each field query no longer increases with the length of the sculpting session (Cani et al., 2008).

The principal difficulties with grid-based sculpting are the aliasing artefacts arising from the discrete nature of voxels, which are typically reduced with low-pass filters. As the surface is extracted from the grid, triangles are recomputed each time a part of the model is modified. Over time, successive re-samplings can cause blurring. Moreover, since the triangles are not persistent, attaching properties to the surface such as bumps or texture is difficult (Stanculescu et al., 2011). The high cost of rendering can be alleviated with local incremental updates.

A significant advantage of this sculpting approach is that the user can directly interact with a model's surface without any knowledge of the model's underlying representation. While grid-based models require large computation and storage overheads, they support highly complex topologies within an intuitive interface (Gain and Bechmann, 2008).

Spatial Sculpting

Sculpting metaphors are often based on spatial deformation techniques. Most of these techniques use a tool to define a deformation, where interactive modifications of the tool are passed to the object being sculpted. These techniques can be used directly within a sculpting framework, as discussed in Section 2.2.1.

More recently, much research has focused on providing a dedicated framework built upon spatial deformations for the specific task of interactive shape modelling. As shall be seen, a direct result of this has been the emergence of hybrid spatial tools encoded by distance fields and decay functions, much like the tools described in Section 2.3.1. These tools provide an interface more akin to a clay sculpting metaphor.

Decaudin (1996) propose a technique that allows the artist to model a shape by iteratively adding the volume of simple 3D shapes. This technique inflates space by inserting a shaped tool . By inserting the tool inside the object to be sculpted, the volume of the tool will be added to the volume of the object. If the tool is applied outside the object, the object will be deformed but its volume will remain constant. A restriction imposed by the deformation function used is that the tool must be star-convex with respect to its center.

'Wires', introduced by Singh and Eugene (1998), are curve based spatial deformation tools which can easily achieve a very rich set of deformations. Their technique is inspired by the armatures used by sculptors. A wire is defined by a reference curve, a wire curve, a scaling factor that controls bulging around the curve, and a radius of influence. A set of reference curves describes the armature embedded in the initial

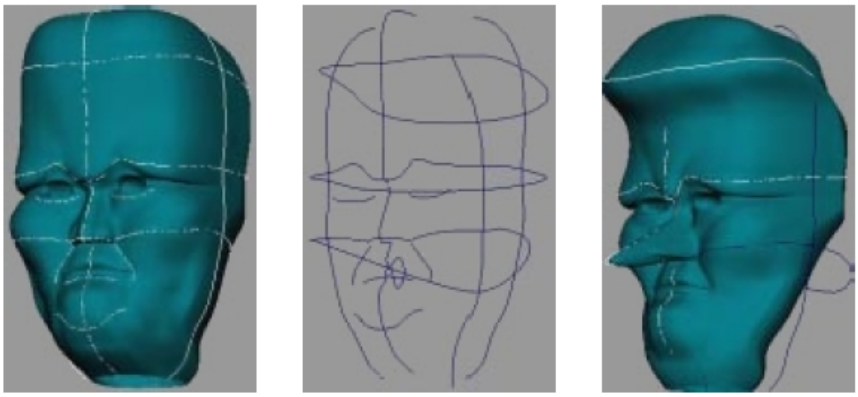


Figure 2.32: Wires: A Geometric Deformation Technique - deformation of a shape with multiple wires (Singh and Eugene, 1998). The first image shows the initial shape, the second shows the reference curves and the third shows the wire curves and the deformed shape.

shape, while a set of wire curves defines the new pose of the armature. This technique extends curve-based methods by enabling a network of 1-D curves to deform a model by blending the contributions of each wire (see Figure 2.32).

'Sweepers', introduced by Angelidis et al. (2004), are space deformations defined by gestures. The artist describes a basic deformation as a path through which a tool is moved. The tools in this case are simple shapes. When moved along the defined path, the tool causes a deformation of the sculpted object along the path. More complicated deformations can be achieved by applying several tools simultaneously using a blending formula.

Angelidis (2005) attributes the restriction of traditional freeform deformation techniques to the domain of editing and animation of existing shapes rather than the creation of new models to the lack of research into the provision of appropriate features. An important example of such a feature is the automatic prevention of self-intersections. This is particularly important for spatial deformation techniques as no spatial deformation can remove a self intersection in a surface once it has been created.

Angelidis et al. define a tool very similar to the tools used by Ferley et al. (2001) and Dewaele and Cani (2004). The tool is a 3D shape defined by a field function which smoothly decreases and vanishes at a distance. Sweeping involves weighting over space the transformation defined by the tool's motion and field of influence. Self intersections are avoided by breaking up the deformation into small steps, computed so that fold-over is prevented (Cani et al., 2008).

Sweepers in their seminal form compress and dilate space and thus provide similar functionality to implicit sculpting tools facilitating the addition and removal of material. Later, Angelidis et al. (2006) introduced 'Swirling Sweepers' for constant volume sweeping. A 'swirl' is defined as a rotation about an axis whose magnitude decreases away from its centre and smoothly vanishes at a distance, defining a divergence-free vector field. A swirl has the effect of twisting space locally around the axis without compression or dilation. Angelidis et al. convert this volume preserving rotation func-

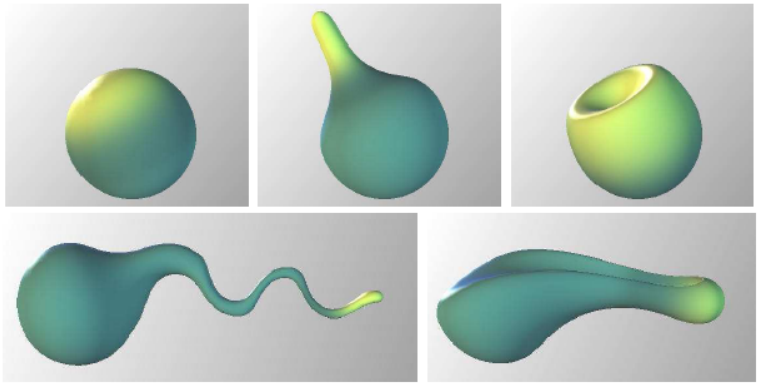


Figure 2.33: Swirling Sweepers: When pushed or pulled, a sphere will inflate or deflate elsewhere (Angelidis et al., 2006)

tion into a translation by using a ring of swirls positioned in a circle with axes along the local tangent direction to the circle. Each of them swirls matter through the inside of the circle. Summing their contributions has the effect of locally pulling space through the circle (Cani et al., 2008). To hide the implementation details from the user, the parameters of the swirls are reverse engineered from the user's sweep gesture. Figure 2.33 shows an example of sculpting operations with Sweepers.

Gain and Marais (2005) introduce a similar technique to sweepers called 'Warp Sculpting' which also uses rigid body transformations of shaped tools. Their shaped tools are also encoded by distance fields and decay functions and are also used to warp a sculpted object's ambient space. Their tools rely on a pre-computed distance field that can be applied after sampling the actual distances involved. This cuts down substantially on closest point operations and improves the interactivity of the sculpting process. Their technique was developed in parallel with Angelidis et al. The sculpting interface presented by Gain et al. is more tool-based than gesture-based (see Figure 2.34). Their approach also differs from Angelidis et al. in the representation of the tool's trajectory.

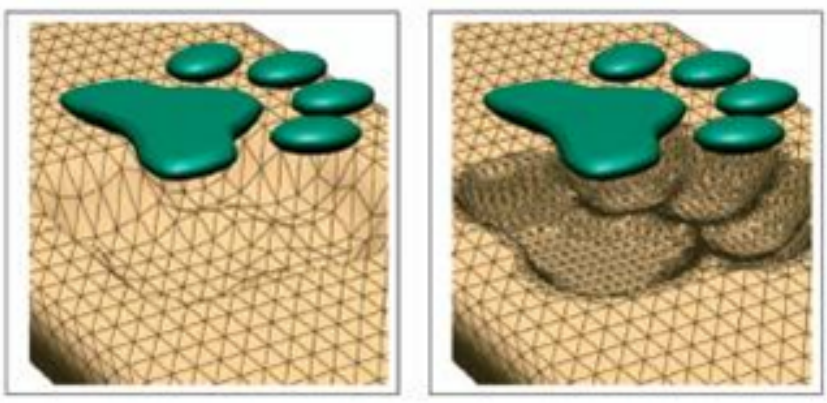


Figure 2.34: Warp Sculpting (Gain and Marais, 2005)

Funck et al. (2006) generalise swirls using divergence-free vector fields to produce different types of deformations including twists and bends. These types of deformations improve the functionality of sweepers but do not fit nicely into the sweeping gestural interface described by Angelidis et al.

The use of fold-over free space deformation to avoid self intersections makes it impossible to change an object's topology. Stanculescu et al. (2011) recently introduced 'Freestyle: sculpting meshes with self-adaptive topology' to combat this problem. In their implementation, sweepers are combined with a specially structured temporally coherent mesh, typically a machine vision tool, to track the sweep. The topology of the mesh is controlled from a single user-specified resolution threshold. Any part of the model thinner than this threshold automatically splits, while any two parts of the mesh that come closer than the threshold distance automatically merge. Complex models can be created using this technique (see Figure 2.35).

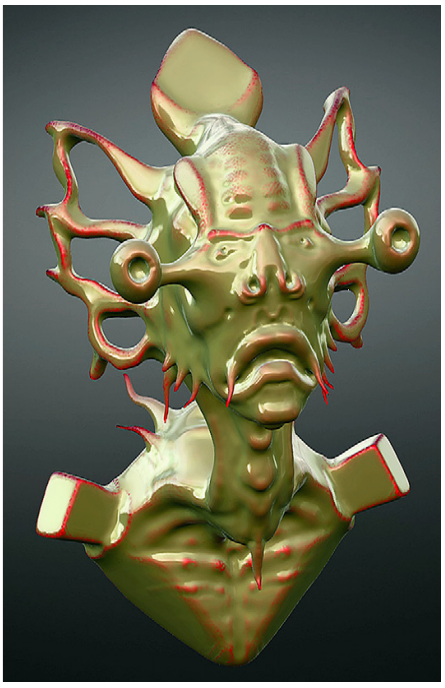


Figure 2.35: 'Freestyle': Sculpting Meshes with Self-Adaptive Topology (Stanculescu et al., 2011)

2.3.2 Physics-Based Sculpting Tools

While many of the approaches presented above mimic some physical properties using geometric techniques, the models did not generally provide any mechanism for physically deforming a model. Physics-based deformations have become established in computer graphics animation and simulation (see Section 2.2) and a number of these techniques have also appeared in physics-based sculpting environments.

Terzopoulos and Qin (1994) introduced Dynamic Non-Uniform Rational B-Splines (D-NURBS) surfaces as an extension of traditional NURBS that permit more natural control of the geometry of the surface. D-NURBS are physics based models that in-

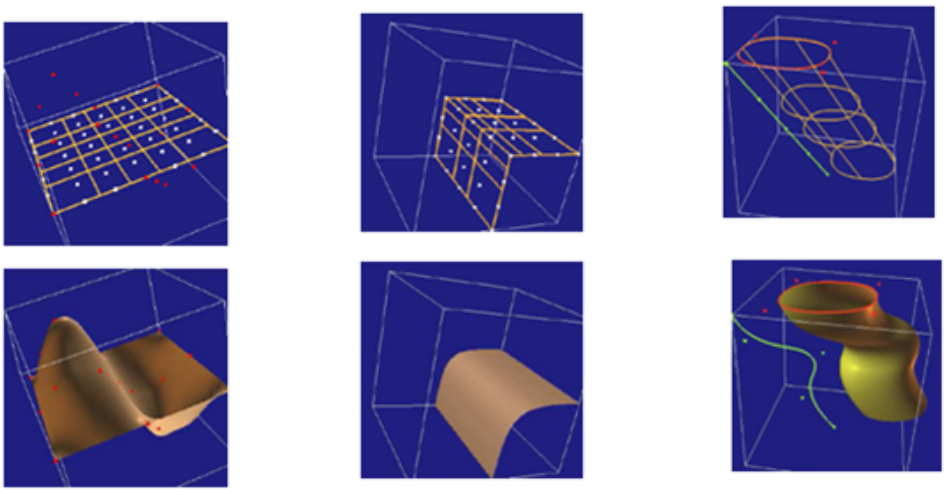


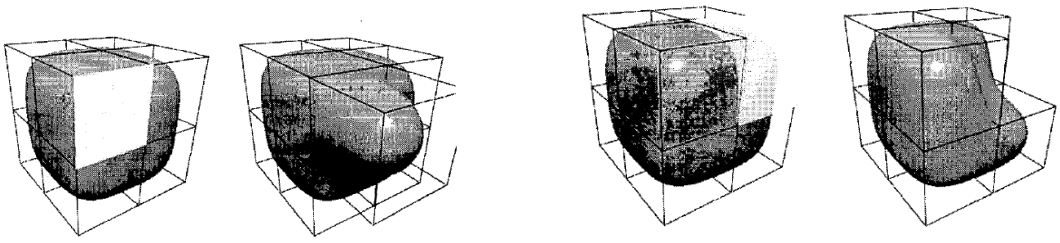
Figure 2.36: D-NURBS: Examples of D-NURBS Surface Fitting, Solid Rounding, and Deformation through Constraint Modification. (Terzopoulos and Qin, 1994)

corporate mass and damping distributions and internal deformation energies into the NURBS geometric substrate. These models are governed by dynamic differential equations which, when integrated numerically through time, continuously evolve the controls and weights in response to applied forces. Optimal shape design is achieved through energy minimization solved using Finite Element techniques.

The D-NURBS formulation supports interactive direct manipulation of NURBS surfaces through constraint modification, which results in physically meaningful and intuitively predictable motion and shape variation (Terzopoulos and Qin, 1994, Qin and Terzopoulos, 1994, 1996). Using D-NURBS, a modeler can interactively sculpt complex shapes by kinematically adjusting control points and weights, or dynamically by applying forces. Additional control over dynamic sculpting stems from the modification of physical parameters such as mass, damping, and elastic properties. Design applications discussed include solid rounding, surface fitting, dynamic FFD, and cross sectional interactive design. Figure 2.36 shows some examples of these applications.

Later, such ideas were extended to subdivision surfaces (Qin et al., 1998, McDonnell and Qin, 2000, McDonnell et al., 2001, McDonnell and Qin, 2002, McDonnell et al., 2005, McDonnell and Qin, 2007*a*,*b*). Sculpting is achieved by adding or removing subdivision control cells. See Figure 2.37 for sculpting examples. Du and Qin (2000*a*,*b*, 2001, 2005, 2007) examined incorporating such internal deformation energies into both surface and volumetric PDE geometric substrates for similar constraint-based modelling. Figure 2.38 shows a sculpting example. Hua and Qin (2004) examined incorporating physics into an implicit sculpting approach to provide haptic interaction.

Dachille IX et al. (1999, 2001) present a sculpting system utilising a dual representation for a B-spline surface in both mathematical and physical space. Their system is ultimately a discrete case of D-NURBS with fixed weights. A B-Spline surface is discretised using a MSD model. The system thus maintains two synchronised representations



(a) (b)

Figure 2.37: Dynamic Subdivision Surfaces: (a) Extrusion by adding a cell (b) Indent by removing a cell (McDonnell and Qin, 2000)

that permit surfaces to conform to B-Splines in the mathematical domain while exhibiting physical behaviour and satisfying material properties subject to intrinsic geometric constraints. The system allows for specification of tangent, normal, curvature and other constraints. Surface behaviour responds to the Lagrangian equations of motion subject to various geometric constraints. Optimal shape design is achieved through energy minimization solved using Finite Difference techniques. Finally, the system implements a haptic interface via the Phantom Stylus and provides a rope/spring like feature as the interaction tool (see Figure 2.39).

In contrast, Mullenhoffe (1998) employs a continuous model and discusses obstacle avoidance of a sphere as being akin to Virtual Sculpting. However, the obstacle avoidance is computed only after the surface has reached its minimum energy configuration subject to other constraints.

Knopf and Igwe (2005) developed a Virtual Sculpting framework based on mesh models represented by a self-organising feature map. The action of sculpting is simulated by moving the mesh according to a simple dynamic model based on a mass-spring system.

Gao and Gibson (2006) combine the idea of a shaped tool, by Bill and Lodha (1994), with the physics based B-Spline model of Dachille IX et al. (1999). The tool they adopt is a spherical implicit surface that is governed by the Phantom Stylus (see Figure 2.40).

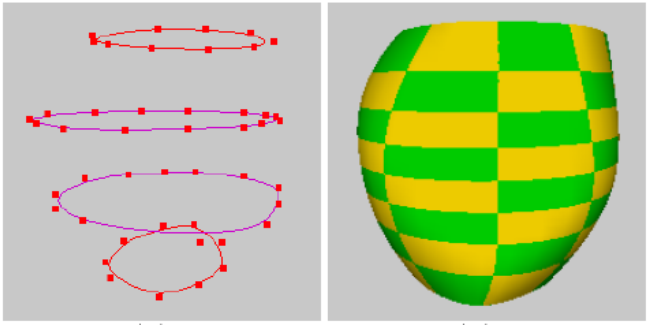
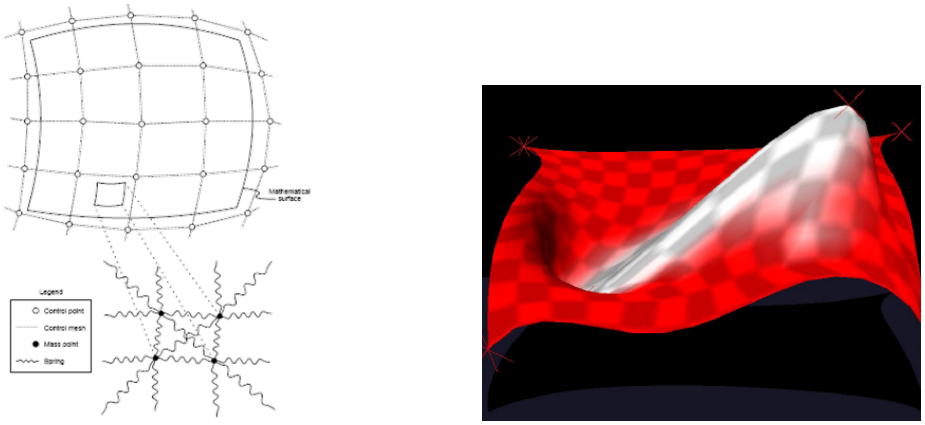


Figure 2.38: Physics Based PDE Surface Constraint Curves and Fitted Surface (Du and Qin, 2000*b*).



(a) (b)

Figure 2.39: Haptic Sculpting of Dynamic Surfaces: (a) MSD Discretisation (b) Rope Tool (Dachille IX et al., 1999)

Pungotra et al. (2010) present a similar approach to that presented in Gao and Gibson (2006) capable of dealing with B-Spline tools.

Smooth surface deformation functions with prescribed boundary conditions are often modelled using an energy minimisation principle as described in Section 2.2.2 (Moreton and Séquin, 1992, Celniker and Gossard, 1991, Welch and Witkin, 1992, 1994, Botsch and Kobbelt, 2004). In such cases, the surface is assumed to behave like a physical skin, which resists stretches and bends as forces act on it. Mathematically, this behaviour can be captured by an energy functional which penalises stretching and bending. In each case, the energy is minimised while satisfying the prescribed boundary constraints. This technique has been adapted in various ways for use in interactive modelers. Welch and Witkin (1992) specify a number of original and target vertices and compute the remaining positions using a variational approach. Celniker and Gossard (1991) define an object using a set of 3D character lines. The object is 'skinned', i.e., fitted with a deformable surface defined with character lines as geometric constraints. The

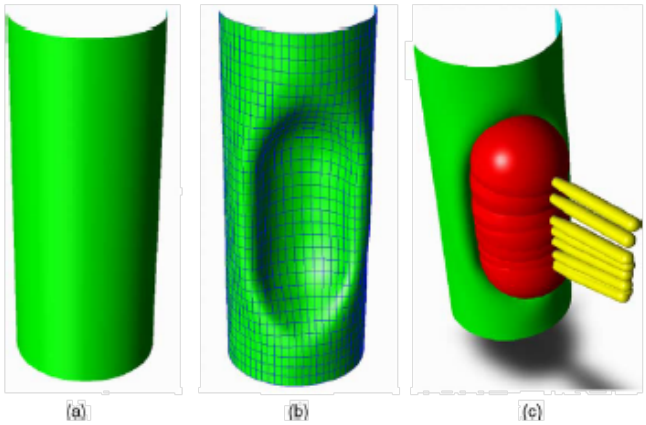
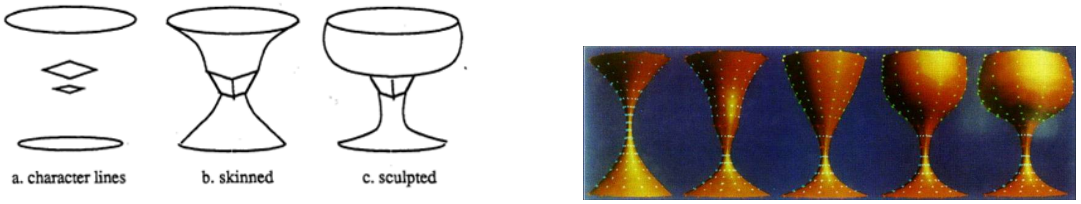


Figure 2.40: Shaped Tools for Haptic Sculpting (Gao and Gibson, 2006)



(a) (b)

Figure 2.41: 'ShapeWright' Sculpting Tool: (a) Character Lines (b) Interactive Sculpting with Inflation Force (Celniker and Gossard, 1991)

surface fitting is achieved using a Finite Element solution (see Section 2.2.2). Interactive sculpting with forces such as inflation can be used to further manipulate the object. An example is shown in Figure 2.41.

Botsch and Kobbelt (2004) use a similar technique to allow a customisable shape modification. The deformation is specified visually by drawing 'handles', which are regions of the surface that can be pushed or pulled. Regions of the surface can be marked 'fixed' or 'free'. As the handles are manipulated, the 'free' surface deforms to fit its new constraints. An example of their interface is shown in Figure 2.42.

2.3.3 Discussion

This section has described various approaches to interactive shape design based on a sculpting metaphor.

Virtual Sculpting is very much an active area of research. Geometric techniques seem to be favoured due to the levels of interactivity deemed necessary. While these techniques can be visually effective, they are not physically accurate and are typically presented as a "more suitable" alternative to physics-based sculpting where interactivity is paramount. While geometric techniques are seen to be computationally less expensive, they are generally seen as a less desirable alternative to physics-based techniques.

Although energy-based shape deformation techniques have been applied to the problem of Virtual Sculpting, they appear to be primarily targeted at solutions for surface fitting based on predefined constraints such that the definition of the surface, rather than interactive deformations akin to virtual sculpting, is the primary goal.

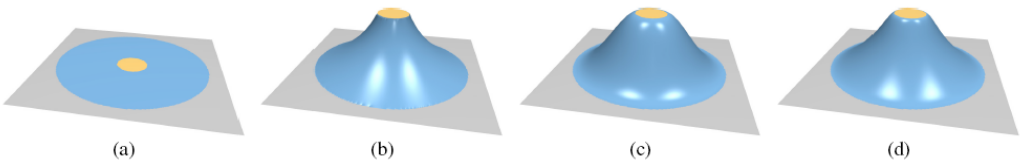


Figure 2.42: Constraint Shape Optimization: User can customise the surface fitting by modifying smoothness and stiffness parameters (Botsch and Kobbelt, 2004)

Those techniques that have incorporated shaped tools have relied on approximating discretisation techniques for interactivity or static obstacle avoidance algorithms. Some techniques seem to be hampered by a desire to use haptic interfaces. Many of the haptic interfaces available are prohibitively expensive which severely hampers general progressive development, while those more popular techniques that use the less expensive Phantom Stylus option are limited by the restrictions it imposes.

Energy-based techniques have not yet been fully utilised in the area of interactive Virtual Sculpting environments.

2.4 Conclusions

One of the major advantages in computer graphics and CAD is the ability to combine techniques in layers to achieve more complex and realistic results and tailor a solution for a specific set of requirements. However, this may also be one of its greatest disadvantages as practitioners must be familiar with a host of practices in order to make well-informed decisions. This chapter details and classifies the wealth of available methods relating to shape representation, deformation, and the various combinations of the two. By providing a systematic description of the assumptions and setup of each model, the practitioner can make more informed decisions regarding the appropriate selection of the shape representation, the method whereby it should be deformed, and their integration into a freeform modelling framework. No other work was found in the literature that considers the full spectrum of techniques across different domains that are necessary to understand the development and the current state of freeform design. Overall, no one technique offers all of the sought after qualities for a particular application and there are always trade-offs between realism, accuracy and computational efficiency.

Discrete shape representations are shown to be well suited to computer processing and manipulation, but do not generally lend themselves well to intuitive design. Continuous representations, such as CSG, provide a more intuitive solution. However, it is difficult to work with freeform models in a CSG representation. Despite drawbacks relating to the representation of higher genus surfaces, B-Spline/NURBS representations are the *de facto* standard for shape representation in CAD/CAGD applications. There is no one shape representation that stands out as being universally applicable, and it is the designer's prerogative to select a representation that meets their application-specific needs. For the freeform modelling application developed in this thesis, B-Spline/NURBS representations are the most suitable representation of the deformable model. They are piecewise polynomial facilitating simultaneously a compact representation and ease of local modification. They are numerically very stable. They can represent not only elementary curves and surfaces, such as circles and ellipses, but are ideal for smoothly varying freeform surfaces.

Geometric deformation techniques are computationally fast and easy to implement, but designer interaction with them typically occurs at a low-level, necessitating prac-

tice and skill on the designer's part. Additionally, such techniques are not physically motivated and therefore their deformations do not reflect real-world deformations. In contrast, physics-based deformation models can capture realistic deformation by imparting material properties to the object that interact with applied forces under the laws of continuum mechanics. These models are the state of the art for achieving accurate deformations. However, they are difficult to implement at interactive rates due to the computational overhead of the associated PDE solvers. Consequently, many physics-based models compromise on their physical fidelity in order to improve the computational efficiency. FEM is considered state of the art amongst these models.

Active Contour Models and Elastically Deformable models provide a mechanism for describing physically active curves and surfaces. These models lend themselves well to describing the energy of an Active B-Spline/NURBS surface. For the purposes of animation, where accurate simulations involving time are important, Elastically Deformable models that include mass and damping properties are suitable. For interactive sculpting, the modelling of time as a real property is less important such that realistic simulation of accelerations due to mass and damping become less important than the material properties of the model, namely; elasticity and rigidity. In this thesis, the deformable model takes its inspiration from the Active Contour Model. However, the mass and damping properties of the Elastically Deformable model are also considered for completeness.

The analytic descriptions for Active Contour Models and Elastically Deformable models are typically discretised and solved using FDM or FEM appraoches, FEM again being the state of the art. IgA represents a recent alternative to FEM for the specific case of analysing B-Splines/NURBS geometries. IgA removes the need for computationally expensive conversions between CAD/CAGD geometries and FEM geometries by incorporating the B-Spline/NURBS representations directly in the PDE solvers. Additionally, it facilitates more accurate results. Consequently, it is starting to make headway on FEM techniques for CAD/CAGD applications. IgA is particularly interesting to the work of this thesis as B-Spline/NURBS representations are the most suitable underlying shape representation for the freeform deformable model.

The selection of suitable shape representations and deformable modelling techniques are paramount to enabling a feasible Virtual Sculpting approach for freeform surface design. The pertinent desirable properties for the Virtual Sculpting paradigm are that the model interactions are interactive and exhibit physical realism. The proposed approach developed in this thesis is ACM-based Active B-Spline/NURBS Surfaces, with an IgA approach to solving the resulting PDE, that provides a balance between interactivity and physical realism.

Unification of Energy-Based Methods for Deformable Surface Modelling

3

This chapter is concerned with the mathematical modelling of energy-based deformations for computing. In Chapter 2, physics-based techniques were identified as the state of the art tools for simulating physical deformations in a body. The ever widening range of applications includes not only industrial and artistic design, but also a host of other applications that stem from multiple computing domains, e.g., Computer Vision, Visualisation, Computer Graphics, Computer-Aided Design, Computer-Aided Engineering, Computer-Aided Manufacturing, etc. Despite the very strong relationships that exist between the various fields, large overlaps are not being fully exploited due to divisions between the relevant research communities. Perhaps worse still, the similarities of the techniques employed have led to confusion in the literature. Each domain typically has its own set of requirements and constraints, defined by the specific applications. This means that domain-specific modifications are often made to the more general approach. The similarities of the techniques employed often lead to confusion resulting in material being ported from one field to another without a full appreciation of its origins or appropriate usage. This situation is exacerbated by the disparities that have arisen with regard to the terminology and notation employed within each domain. There is no common thread that generalises the disparate approaches for use across the different domains. In this chapter, a unified representation is sought. To achieve this, several apparent contrasting models are derived from first principles and shown to be largely equivalent. Where they differ, the theoretical relationships between them are formally defined.

3.1 Continuum Mechanics vs Differential Geometry

Continuum Mechanics and Differential Geometry provide the tools required to describe changes in the configuration of a deformable body. In order to fully capture such a change, two components must be considered: the absolute displacement of particles in the body, and the relative displacement of particles within the body. The former describes the rigid-body displacement that changes the absolute position of the body without changing its size or shape. The latter describes the deformation of the body and captures the change in the relative displacements between particles. The field of Continuum Mechanics is concerned with modelling the strain induced by stresses

within the deformable body as a result of the application of external forces. Differential Geometry facilitates the measurement of the induced displacements within the geometry of the deformable model. Figure 3.1 illustrates the change in configuration of a given section of an undeformed model that undergoes a deformation. The section between two points p and q on the model, where subscript 0 represents the original dimensions, is monitored as the model deforms. As might be expected, measuring the changes in strains and stresses or measuring the changes in the local displacements, both facilitate a quantification of deformation. This section describes the two prominent but alternative approaches to deriving an energy-based deformable model, one more heavily rooted in Continuum Mechanics and the other in Differential Geometry. The section will show that the two approaches are largely equivalent. While such equivalences are studied in the field of Continuum Mechanics, they are often neglected in the discussion of energy-based deformable models within the fields of CAD/CAGD, Computer Graphics, Machine Vision and Visualisation.

3.1.1 Continuum Mechanics Approach

The Continuum Mechanics approach to modelling deformation relies heavily on two tensors, namely the Deformation Gradient Tensor and the Displacement Gradient Tensor. These tensors are used to quantify the strains and stresses within a deformable body. This in turn facilitates modelling the energy of the deformation. This section describes the construction of such a model.

Deformation and Displacement Gradients

In general, deformations are specified in terms of a deformation and/or displacement gradient (Reddy, 2008). The deformation gradient is a measure of the derivatives of the coordinate locations of the deformed model, (x, y, z), with respect to those of the undeformed model, (x_0, y_0, z_0) (See Figure 3.1). The deformation gradient matrix is given by

$$\mathbf{F} = \begin{bmatrix} \frac{\partial x}{\partial x_0} & \frac{\partial x}{\partial y_0} & \frac{\partial x}{\partial z_0} \\ \frac{\partial y}{\partial x_0} & \frac{\partial y}{\partial y_0} & \frac{\partial y}{\partial z_0} \\ \frac{\partial z}{\partial x_0} & \frac{\partial z}{\partial y_0} & \frac{\partial z}{\partial z_0} \end{bmatrix} \tag{3.1}$$

Alternatively, the deformation can be specified by the gradient of the displacement function, $\nabla_{x_0} u(x_0)$, where $u(x_0) = x - x_0$. The displacement gradient matrix is thus given by

$$\mathbf{G} = \mathbf{F} - \mathbf{I} = \begin{bmatrix} \frac{\partial x}{\partial x_0} - 1 & \frac{\partial x}{\partial y_0} & \frac{\partial x}{\partial z_0} \\ \frac{\partial y}{\partial x_0} & \frac{\partial y}{\partial y_0} - 1 & \frac{\partial y}{\partial z_0} \\ \frac{\partial z}{\partial x_0} & \frac{\partial z}{\partial y_0} & \frac{\partial z}{\partial z_0} - 1 \end{bmatrix} \tag{3.2}$$

The tensors $\mathbf{F}$ and $\mathbf{G}$ are the foundations of Continuum Mechanics.

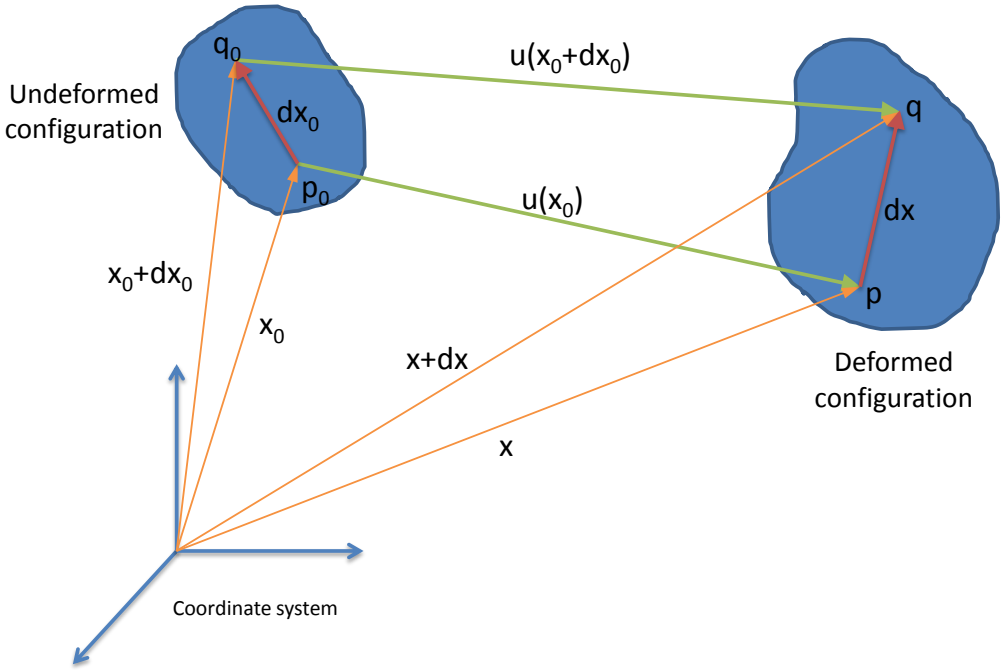


Figure 3.1: Model Deformation

Often the rotational component of the deformation tensor is removed by multiplying the tensor by its transpose. The product causes the rotational elements, R, to become the Identity matrix, $R^T R = I$. The Cauchy-Green rotation-independent deformation tensor, $\mathbf{C}$, is used most often in practice, and is given by

$$\mathbf{C} = \mathbf{F}^T \mathbf{F} \tag{3.3}$$

Consider a parameterised curve $\mathbf{x}$. A measure of the rest length, l_{x_0}, of the curve is given by

$$l_{x_0} = \int \left| \frac{d\mathbf{x}_0}{ds} \right| ds \tag{3.4}$$

Differentiation of the line element with respect to the parameter, and squared result is given by

$$\left(\frac{dl_{x_0}}{ds} \right)^2 = \frac{d\mathbf{x}_0}{ds} \cdot \mathbf{I} \cdot \frac{d\mathbf{x}_0}{ds} \tag{3.5}$$

Similarly, the corresponding equations for the deformed configuration, l_x, are given by

$$l_x = \int \left| \frac{d\mathbf{x}}{ds} \right| ds \tag{3.6}$$

and

$$\left(\frac{dl_x}{ds} \right)^2 = \left(\frac{d\mathbf{x}}{dx_0} \cdot \frac{d\mathbf{x}_0}{ds} \right) \cdot \left(\frac{d\mathbf{x}}{dx_0} \cdot \frac{d\mathbf{x}_0}{ds} \right)$$

$$= \frac{d\mathbf{x}_0}{ds} \cdot \left[\left(\frac{d\mathbf{x}}{d\mathbf{x}_0} \right)^T \frac{d\mathbf{x}}{d\mathbf{x}_0} \right] \cdot \frac{d\mathbf{x}_0}{ds}$$
$$= \frac{d\mathbf{x}_0}{ds} \cdot \mathbf{C} \cdot \frac{d\mathbf{x}_0}{ds} \tag{3.7}$$

As can be seen, the difference between the two lengths is captured in the relative change from **I** to **C**. In this way, a physical interpretation of the tensor is that it provides a measure of the square of the local change in distances due to the deformation, i.e., $d\mathbf{x}^2 = d\mathbf{x}_0 \cdot \mathbf{C} \cdot d\mathbf{x}_0$.

Strain

Strain describes the relative displacements between infinitesimal particles within a body. It is quite clear that **I** and **C**, in Equations 3.5 and 3.7, quantify the change in the lengths of the line element moving from the undeformed configuration to the deformed configuration. The relative displacement is captured by the difference between the undeformed and deformed configurations. The *Green-Lagrange strain tensor*, often referred to as *Green's tensor* or the *Green-St-Venant strain tensor* in the literature, is the most popular choice when computing for representing strain (Nealen et al., 2006). The tensor is given by

$$\boldsymbol{\varepsilon}_{G_{deformation}} = \frac{1}{2}(\mathbf{C} - \mathbf{I}) \tag{3.8}$$

Where the displacement gradient tensor, **G**, is used, the alternative strain measure is described by

$$\boldsymbol{\varepsilon}_{G_{displacement}} = \frac{1}{2}(\mathbf{G} + \mathbf{G}^T) + \mathbf{G}^T\mathbf{G} \tag{3.9}$$

Regardless of the tensor chosen, it is usually denoted in the literature (Shabana, 2011) by

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_{11} & \varepsilon_{12} & \varepsilon_{13} \\ \varepsilon_{21} & \varepsilon_{22} & \varepsilon_{23} \\ \varepsilon_{31} & \varepsilon_{32} & \varepsilon_{33} \end{bmatrix} \equiv \begin{bmatrix} \varepsilon_{xx} & \varepsilon_{xy} & \varepsilon_{xz} \\ \varepsilon_{yx} & \varepsilon_{yy} & \varepsilon_{yz} \\ \varepsilon_{zx} & \varepsilon_{zy} & \varepsilon_{zz} \end{bmatrix} \tag{3.10}$$

Stress

By the principal of virtual work, each strain measure has a corresponding stress measure that measures the internal forces acting within a deformable body. Quantitatively, it is a measure of the average force per unit area of a surface, within the body on which internal forces act. This is given by

$$\boldsymbol{\sigma} = \frac{\mathbf{F}}{A} \tag{3.11}$$

where σ is the stress, F is the external force and A is the area of the surface. The stress forces arise as a reaction to external forces being applied to the body. Because the

deformable body is assumed to behave as a continuum, the internal forces distribute continuously within the volume of the body, resulting in deformation of the body's shape. Stress is generally not uniformly distributed over the cross-section of a material body. Consequently, the stress at a given point differs from the average stress over the entire area. Therefore, it is necessary to define the stress at a specific point in the body. This can be expressed by the Cauchy stress tensor denoted by

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{bmatrix} \equiv \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{bmatrix} \tag{3.12}$$

Constitutive Equations

Constitutive laws define the relationship between the stress and the strain. The relationship can be non-linear but, in general a linear relationship is adopted. The most popular constitutive law in Computer Graphics applications is Hooke's Linear Material Law, which is denoted by

$$\boldsymbol{\sigma} = \mathbf{E} \cdot \boldsymbol{\varepsilon} \tag{3.13}$$

where $\mathbf{E}$ is a new tensor that linearly relates the coefficients of the stress tensor to those of the strain tensor. For isotropic materials, the coefficients of $\mathbf{E}$ only depend on Young's modulus and Poisson's ratio, that describe the material properties.

Deformation Energy and Force

The deformation energy over the deformable body can be computed by integrating the component-wise scalar-product of the stress and strain tensors over the body.

$$E = \int \boldsymbol{\varepsilon} \cdot \sigma d\mathbf{x} \tag{3.14}$$

The corresponding forces, $\mathbf{f}$, are the derivatives of the energy with respect to positional parameters, $\mathbf{x}$, of the model, where $\mathbf{f}$ is given by

$$\mathbf{f} = \mathbf{K}\mathbf{x} \tag{3.15}$$

or the displacement parameters, $\mathbf{u}$, of the model, where $\mathbf{f}$ is given by

$$\mathbf{f} = \mathbf{K}\mathbf{u} \tag{3.16}$$

The stiffness matrix, $\mathbf{K}$, relates the forces acting on a deformable body to its positional/displacement parameters respectively.

Finally, dynamics can be added to the model by introducing mass, damping, and time to the system as

$$\mathbf{M}\frac{d^2\mathbf{x}}{dt^2} + \mathbf{D}\frac{d\mathbf{x}}{dt} + \mathbf{K}\mathbf{x} = \mathbf{f} \tag{3.17}$$

$$\mathbf{M}\frac{d^2\mathbf{u}}{dt^2} + \mathbf{D}\frac{d\mathbf{u}}{dt} + \mathbf{K}\mathbf{u} = \mathbf{f} \tag{3.18}$$

where $\mathbf{M}$ and $\mathbf{D}$ represent the mass and damping of the system respectively.

3.1.2 Differential Geometry Approach

An alternative approach to the problem of quantifying deformation in a deformable body comes from the study of the Fundamental Forms of Differential Geometry. Much like the tensors described in the previous section, Fundamental Forms provide a means of quantifying local displacements on/in a deformable body.

In the literature, the approach is typically presented starting with parameterised versions of the dynamic force balance equations (See Equations 3.17 and 3.18) that concluded the treatment of Continuum Mechanics in the previous section (Terzopoulos and Fleischer, 1988, Christensen and Floren, 2004). The general equation is given by

$$\frac{\partial}{\partial t}\left(\mu\frac{\partial r}{\partial t}\right) + \gamma\frac{\partial r}{\partial t} + \frac{\delta\varepsilon(r)}{\delta r} = f(r,t) \tag{3.19}$$

where r is the parametric description of the body, μ and γ are the mass and damping parameters of the model, and $\frac{\delta\varepsilon(r)}{\delta r}$ represents the elasticity and bending properties of the model. The fundamental theorem of solids is applied to model the elastic and bending properties.

The fundamental theorem of solids states that two solids in space, described by a positional vector $\mathbf{x}(\mathbf{u})$, will have the same instantaneous shape if their metric tensors are identical functions of $\mathbf{u} = (u_1, u_2, u_3)$ at time t.

The First Fundamental Form is also known as the Metric Tensor, $\mathbf{G}$, and is usually given by index notation in the literature as

$$\mathbf{G}_{ij}(\mathbf{x}(u,t)) = \frac{\partial\mathbf{x}}{\partial\mathbf{u}_i}\cdot\frac{\partial\mathbf{x}}{\partial\mathbf{u}_j} \tag{3.20}$$

In similar notation as that used in Continuum Mechanics resources, a 0 (superscripted rather than subscripted) denotes the undeformed metric tensor.

$$\mathbf{G}^0_{ij}(\mathbf{x}^0(u,t)) = \frac{\partial\mathbf{x}^0}{\partial\mathbf{u}_i}\cdot\frac{\partial\mathbf{x}^0}{\partial\mathbf{u}_j} \tag{3.21}$$

Terzopoulos and Fleischer (1988) state that a reasonable energy for elastic bodies is a Euclidean norm of the difference between the fundamental tensors of the deformed body and the fundamental tensors of the body in its natural shape. Adopting their notation for the energy gives

$$\boldsymbol{\varepsilon}(\mathbf{x}) = \int_\Omega \left|\mathbf{G} - \mathbf{G}^0\right|^2_{w^1} du_1 du_2 du_3 \tag{3.22}$$

where w^1 is a weighting function that determines resistance to stretching and shearing in the model.

3.1.3 Discussion

The two alternative developments of the energy-based deformable models clearly demonstrate that the models are largely equivalent. In the case of the Continuum Mechanics description, the energy is given by

$$E = \int \boldsymbol{\varepsilon} \cdot \sigma d\mathbf{x} = \int \boldsymbol{\varepsilon} \cdot \mathbf{E} \cdot \boldsymbol{\varepsilon} \, d\mathbf{x} \tag{3.23}$$

while in the case of the Differential Geometry description, the energy is given by

$$\boldsymbol{\varepsilon}(\mathbf{x}) = \int_{\Omega} \left| \mathbf{G} - \mathbf{G}^0 \right|_{w^1}^2 \, du_1 du_2 du_3 \tag{3.24}$$

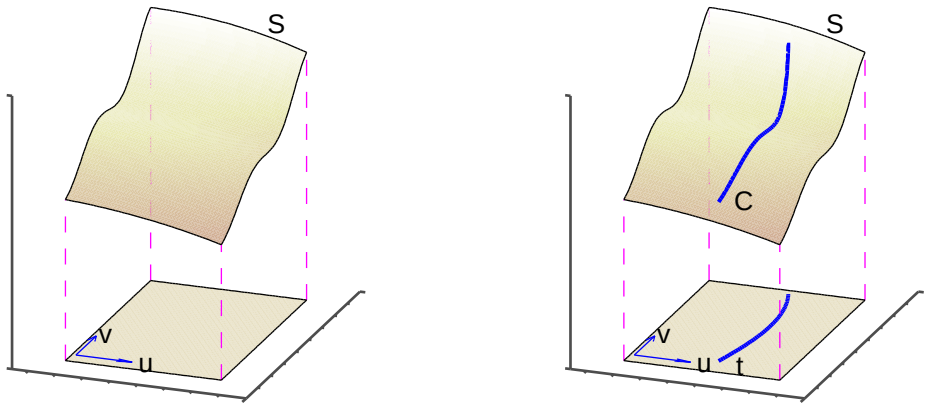
such that the difference in metric tensors provides the strain measurement and the weighting function w^1 provides the material properties, replacing $\mathbf{E}$ in the Continuum Mechanics description.

Similar equivalences can be shown between the Continuum Mechanics and Differential Geometry descriptions for surfaces and curves. In the case of a surface, Continuum Mechanics draws from 'Membrane', 'Shell' and 'Plate' theory that treats volumetric bodies whose thickness is small compared to their extents (Shabana, 2011). The reason for the distinction is that where the thickness is relatively small, the distances between nearby points is no longer a sufficient measure as the surface can be bent without perturbing local distances. For this reason additional tensors that measure local curvatures as well as displacements must be considered. For a 3D deformable curve, torsion must also be accounted for.

The equivalence between the two models is important and, while studied in the field of Continuum Mechanics, is often neglected in the literature relating to CAD/CAGD, Computer Graphics, Computer Vision and Visualisation such that assumed models are adopted without due consideration tof their origins. Apparently contrasting models, and resulting alternative interpretations, can lead to confusion. As an example, assigning a weight parameter to govern the material properties of the model is arguably simplified in the Differential Geometry description. However, without the reference to Continuum Mechanics, it is not necessarily apparent to a designer exactly how the weight parameter relates to the material properties. Designers are left to choose weights on an ad-hoc basis to achieve a desired result. Exposing the relationships between Differential Geometry and Continuum Mechanics in the derivation of the energy equation enables the designer to attribute model parameters in a more physically meaningful way.

3.2 Deformable Surface Model

As this thesis is primarily concerned with surface deformations, a model of an energy-based surface is now derived using the Differential Geometry approach so that meaningful comparisons can be made between this description and alternative surface de-



(a) Parameterised Surface (b) Embedded Parameterised Curve

Figure 3.2: Differential Geometry: (a) Parameterised Surface $S(u, v)$ (b) Parameterised Curve, $C(t)$, embedded in Surface, $S(u, v)$

scriptions described in the literature. The techniques can be typically generalised to higher dimensions. A good overview can be found in (Christensen and Floren, 2004).

3.2.1 Fundamental Forms

As introduced in Section 3.1.2, Fundamental Forms are units of measure that facilitate the determination and description of certain metric properties of curves and surfaces.

First Fundamental Form

Consider a curve, $\mathbf{C}$, on a surface, $\mathbf{S}$, defined by $\mathbf{S} = \mathbf{S}(u(t), v(t))$, as depicted in Figure 3.2. The arc length, s, of the curve on the surface is given by

$$s(t) = \int_{t_2}^{t_1} \left| \frac{d\mathbf{S}}{dt} \right| \, dt \tag{3.25}$$

Differentiating the arc length with respect to the parameter t, and squaring the result yields

$$\left(\frac{ds}{dt}\right)^2 = \left|\frac{d\mathbf{S}}{dt}\right|^2 = \frac{d\mathbf{S}}{dt} \cdot \frac{d\mathbf{S}}{dt} \tag{3.26}$$

Using the chain rule of differentiation,

$$\frac{d\mathbf{S}}{dt} = \frac{\partial \mathbf{S}}{\partial u}\frac{du}{dt} + \frac{\partial \mathbf{S}}{\partial v}\frac{dv}{dt} \tag{3.27}$$

Accordingly,

$$\left(\frac{ds}{dt}\right)^2 = E\left(\frac{du}{dt}\right)^2 + 2F\frac{du}{dt}\frac{dv}{dt} + G\left(\frac{dv}{dt}\right)^2 \tag{3.28}$$

where E, F and G are called coefficients of the First Fundamental Form.

$$E = \left(\frac{\partial \mathbf{S}}{\partial u}\right)^2 \tag{3.29}$$

$$F = \frac{\partial \mathbf{S}}{du} \cdot \frac{\partial \mathbf{S}}{dv} \tag{3.30}$$

$$G = \left(\frac{\partial \mathbf{S}}{dv}\right)^2 \tag{3.31}$$

The coefficients are assembled into a tensor, typically denoted $\mathbf{G}$ or $\mathbf{I}$ in the literature given by

$$\mathbf{G} = \begin{bmatrix} E & F \\ F & G \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathbf{S}}{\partial u} \cdot \frac{\partial \mathbf{S}}{\partial u} & \frac{\partial \mathbf{S}}{\partial u} \cdot \frac{\partial \mathbf{S}}{\partial v} \\ \frac{\partial \mathbf{S}}{\partial v} \cdot \frac{\partial \mathbf{S}}{\partial u} & \frac{\partial \mathbf{S}}{\partial v} \cdot \frac{\partial \mathbf{S}}{\partial v} \end{bmatrix} \tag{3.32}$$

This tensor is the Metric Tensor or First Fundamental Form of the parametric surface.

Second Fundamental Form

As discussed briefly in Section 3.1.3, for a surface, the First Fundamental Form is not sufficient to describe the deformation. The Second Fundamental Form facilitates the quantification of the curvatures of a surface $\mathbf{S}$, by considering the curvatures along the curve $\mathbf{C}$.
Starting with

$$\frac{d\mathbf{S}}{dt} = \frac{\partial \mathbf{S}}{\partial u}\frac{du}{dt} + \frac{\partial \mathbf{S}}{\partial v}\frac{dv}{dt} \tag{3.33}$$

the second derivative of the curve can be obtained by differentiating as follows

$$\frac{d^2\mathbf{S}}{dt^2} = \frac{\partial \mathbf{S}}{\partial u}\frac{d^2u}{dt^2} + \frac{\partial \mathbf{S}}{\partial v}\frac{d^2v}{dt^2} + \frac{\partial^2 \mathbf{S}}{\partial u^2}\left(\frac{du}{dt}\right)^2 + 2\frac{\partial^2 \mathbf{S}}{\partial u \partial v}\frac{du}{dt}\frac{dv}{dt} + \frac{\partial^2 \mathbf{S}}{\partial v^2}\left(\frac{dv}{dt}\right)^2 \tag{3.34}$$

The Second Fundamental Form is obtained by projection onto the normal, n,

$$n = \frac{\partial \mathbf{S}}{\partial u} \times \frac{\partial \mathbf{S}}{\partial v} \tag{3.35}$$

$$a = \frac{n}{|n|} \cdot \frac{d^2\mathbf{S}}{dt^2} \tag{3.36}$$

$$a = L\left(\frac{du}{dt}\right)^2 + 2M\frac{du}{dt}\frac{dv}{dt} + N\left(\frac{dv}{dt}\right)^2 \tag{3.37}$$

The coefficients L, M and N are the coefficients of the Second Fundamental Form

$$L = n \cdot \frac{\partial^2 \mathbf{S}}{\partial u^2} \tag{3.38}$$

$$M = n \cdot \frac{\partial^2 \mathbf{S}}{\partial u \partial v} \tag{3.39}$$

$$N = n \cdot \frac{\partial^2 \mathbf{S}}{\partial v^2} \tag{3.40}$$

and are gathered as

$$\mathbf{B} = \begin{bmatrix} L & M \\ M & N \end{bmatrix} = \begin{bmatrix} n \cdot \frac{\partial^2 \mathbf{S}}{\partial u^2} & n \cdot \frac{\partial^2 \mathbf{S}}{\partial u \partial v} \\ n \cdot \frac{\partial^2 \mathbf{S}}{\partial v \partial u} & n \cdot \frac{\partial^2 \mathbf{S}}{\partial v^2} \end{bmatrix} \tag{3.41}$$

where $\mathbf{B}$ is the Curvature Tensor or Second Fundamental Form of the parametric surface.

3.2.2 Energy Description

The energy of the deformation can be measured, as outlined in Section 3.1.2, using the differences of the Fundamental Forms as

$$\varepsilon(\mathbf{x}) = \int_\Omega \left| \mathbf{G} - \mathbf{G}^0 \right|_{w^1}^2 + \left| \mathbf{B} - \mathbf{B}^0 \right|_{w^2}^2 \, dudv \tag{3.42}$$

The Fundamental Forms in this way capture the deformation of the body as follows:

- The differences in the diagonal terms of the First Fundamental Form, $\mathbf{G} - \mathbf{G}^0$, measure scaling or stretching.
- The differences in the off-diagonal terms of the First Fundamental Form, $\mathbf{G} - \mathbf{G}^0$, measure shearing.
- The differences between the terms of the Second Fundamental Form, $\mathbf{B} - \mathbf{B}^0$, measure bending.

3.3 Non-Linear Model vs Linear Model

In the Continuum Mechanics treatment of deformation, presented in Section 3.1.1, the strain tensor is non-linear. Non-linear terms are often linearised/omitted, e.g., the displacement tensor reduces to Cauchy's linear strain tensor.

$$\varepsilon_C = \frac{1}{2}(\mathbf{G} + \mathbf{G}^T) \tag{3.43}$$

In the case of the Differential Geometry approach, the equations are often simplified by replacing the change of First and Second Fundamental Forms by first and second order partial derivatives of the displacement function, $\mathbf{d}$, such that the energy equation becomes

$$\varepsilon(\mathbf{x}) = \int_\Omega \alpha(\|\mathbf{d}_u\|^2 + \|\mathbf{d}_v\|^2) + \beta(\|\mathbf{d}_{uu}\|^2 + 2\|\mathbf{d}_{uv}\|^2 + \|\mathbf{d}_{vv}\|^2) \tag{3.44}$$

where u and v subscripts are partial derivatives.

The linear approximation is valid for small deformations that undergo linear elastic deformation. However, for highly flexible objects like cloth, such linear formulations can produce non-intuitive results, as the linear strain tensor does not remain invariant under rotations. Note that the premise of linearisation in this context is to achieve a quadratic energy functional such that the forces are linear (Botsch and Sorkine, 2008).

Various alternative functions are minimised in the literature to achieve different results. The positional function, the gradient function, and the Laplacian function have all been considered in the literature for various applications (Botsch and Sorkine, 2008). Such functions are commonplace in surface smoothing/fairing applications. The minimisation is achieved by applying variational calculus to yield the Euler-Lagrange Equation, which is then typically solved using numeric approaches.

3.4 ACM/Snake Model

ACM/Snake models were introduced in Chapter 2, Section 2.2.2. The underlying equation, describing the internal energy of the model, is repeated here for convenience.

$$E_{int}(v(s)) = \int_0^1 \alpha \left| \frac{dv}{ds}(s) \right|^2 + \beta \left| \frac{d^2v}{ds^2}(s) \right|^2 ds \tag{3.45}$$

Recall that $v(s)$ in ACM/Snake theory represents the positions of the particles. Consequently, by comparing Equations 3.44 and 3.45 it can be seen that the active contour model description is in fact the linearised version of a 2D deformable curve described using Differential Geometry. Coupled with the derivations of Section 3.1, it is clear that there is a direct equivalence between the ACM/Snake model and the Continuum Mechanics model. This can be generalised for higher dimensions.

3.5 Analytic Model vs Discrete Model

In Chapter 2, the differences between a discrete model and a continuous model were discussed. Although discrete geometry representations have their use, analytic representations are generally superior. This is especially true for design and manufacturing processes which place higher demands on geometric precision. When a surface is stored analytically instead of discretely, the geometry can be subsequently discretised to any level of precision, since a perfect mathematical definition is available. Unless the designer/analyst has tight control over the discrete geometry's resolution, design decisions made from analyses on the discrete representation may not accurately represent the manufactured component or system. Additionally, extra care must be taken when using discrete geometry to ensure important topological features are preserved in the mesh and that adequate geometric resolution is obtained.

In Chapter 2, Section 2.1, the recent advances in automation of content capture via laser scanning technologies was noted. Automatic capture of content typically

results in a discrete set of points or a triangulation of a discrete set of points. An analytic description of such a surface is generally not available. In such cases, it may be appropriate to employ a MSD model approach. However, where an analytic description of the model is available, in the form of a parametric curve or surface, tools from Continuum Mechanics and Differential Geometry facilitate a more accurate solution. The main difficulty with a MSD system is that the model itself is discrete, unlike finite differencing applied to analytic models where the level of discretisation can be chosen such that as the resolution is increased, the model converges to the continuous model.

3.6 Discrete Model vs Discrete Solver

This section identifies potential areas of confusion relating to the differences between a discrete model and a discrete solver. For simplicity, consider the linearised 2D equation associated with the ACM/Snake model. As described in Chapter 2, Section 2.2.2 the energy of the model can be minimised by solving its corresponding Euler-Lagrange equation given by

$$-\alpha\frac{d^2v(s)}{ds^2}+\beta\frac{d^4v(s)}{ds^4}=-\nabla P(s) \tag{3.46}$$

Applying finite differences to the derivative terms results in equations of the form

$$\frac{d^2v}{ds^2}\approx v(s_{n-1})-2v(s_n)+v(s_{n+1}) \tag{3.47}$$

which effectively places springs between each pair of finite nodes. The spring force in a MSD model is given by

$$F_s=-ku \tag{3.48}$$

For the ACM/Snake model α and β provide the spring constants, while the distance between the nodes provides the displacement u. Assembling the discrete equations gives

$$\mathbf{A}\mathbf{x}+\mathbf{P}_\mathbf{x}=0 \tag{3.49}$$

$$\mathbf{A}\mathbf{y}+\mathbf{P}_\mathbf{y}=0 \tag{3.50}$$

and thus finite differences applied to the ACM/Snake model resolves into a spring representation of the model.

An additional equivalence to consider is that in order to ensure a global minimum can be achieved rather than a local minimum, the ACM model is made a function of time and solved iteratively using an explicit/implicit solver such that the equation becomes

$$\mathbf{A}x_t+\mathbf{P}_{x_{t-1}}=-\lambda(x_t-x_{t-1}) \tag{3.51}$$

$$\mathbf{A}y_t + \mathbf{P}_{y_{t-1}} = -\lambda((y_t - y_{t-1}) \tag{3.52}$$

In this way the solver introduces damping to the model.

This theory generalises to the non-linear case where the Fundamental Forms become the units of measure and the spring force is defined by the weighting factor and the displacement of the Fundamental Forms.

$$\mathbf{F}_s = -w^1(\mathbf{G} - \mathbf{G}^0) \tag{3.53}$$

$$\mathbf{F}_s = -w^2(\mathbf{B} - \mathbf{B}^0) \tag{3.54}$$

Overall, it is clear that there is an equivalence between the MSD model and the finite difference discretisation of the continuous force equations. Additionally, by merit of the type of solver used, further MSD properties can be introduced. This theory generalises to the higher dimension case of the parametric surface or indeed a volume. The traditional MSD model is typically considered as being defined over a rectangular domain. In Van Gelder (1998), it is shown that it is not possible to model an isotropic material by a triangular spring mesh. Thus, the analogy presented is restricted to this case.

An assumed corollary of a finite discretisation of the continuous force function being equivalent to a MSD model is that a MSD model is a finite difference discretisation of a continuous force function. The problem with this is that it suffers the disadvantage of resolution not being tuneable, as the continuous analytic model description is lost and thus all a-priori knowledge of the system is lost. In the literature, the confusion is apparent. Dachille IX et al. (1999), Gao and Gibson (2006), and Pungotra et al. (2010) each present a Virtual Sculpting system based on B-Splines (an analytic representation) for free form surface design. However, each approach imparts physical properties on the model using MSD techniques. This approach is counterintuitive as an exact solution of the model with a rectangular parameterisation is available. Additionally, the discrete surface points of the model are evolved with no consideration to the control point positions. This means that a global interpolation technique must be employed to find a new 'best-fit' B-Spline surface. Such global interpolation schemes for converting back to an analytic model require the setting up of a linear system where each discrete sample point on the mesh maps to a single control point. Again, this approach is counterintuitive as one of the merits of B-Spline representations is its compact description.

This section highlights the importance of considering both the model and an appropriate solver in tandem as unexpected effects may be introduced to a system where the solver is not fully understood. In general, solvers are treated as 'black-box' solutions, but it is clear that practitioners should give due consideration to their choice of solver, particularly when modelling physical processes.

3.7 Analytic Solution vs Discrete Solution

This section highlights ambiguities and potential areas of confusion relating to the use of contrasting analytic and discrete solutions for the energy based minimisation of a deformable model.

In Computer Vision, Computer Graphics, and Visualisation, the literature on active contour models suffers from many inconsistencies in approach and ambiguous terminology. The original active contour model, introduced by Kass et al. (1987) used a finite difference discretisation in both space and time to evolve the active contour to its minimal energy state. Advances in active contour research have included the incorporation of B-Spline representations (Menet et al., 1990) and NURBS representations (Meegama and Rajapakse, 2003). These representations improve on convergence speed and facilitate local control and greater flexibility in the curve representation. An additional benefit of the analytic B-Spline and NURBS curve descriptions, that has not been addressed in the literature, is that their availability facilitates an analytic solution.

Ivins and Porill, well-cited in the area of active contour models (Ivins and Porrill, 1995),(Ivins, 1996), (Ivins and Porrill, 2000), provide a Technical Memo entitled "Everything you always wanted to know about Snakes (But were afraid to ask)". The report discusses B-Snakes (Menet et al., 1990) and their implementation. It appears that the authors adopt an analytic approach, at least where such an approach is available in the literature, and revert to a numerical approach otherwise. It is unclear from the original work by Menet et al. (1990) exactly what was intended. In earlier research by Ivins, the author asserts that due to the complexity of the variational equation, before dynamics are even considered, that an analytic solution is impossible (Ivins, 1996). This makes the general approach awkward and inconsistent. For the variational case, where the position of the surface points are optimised, an analytic solution using B-Spline representation has recently become available in the field of CAD (González-Hidalgo et al., 2013). The mathematical framework developed in this thesis (presented in Chapter 5), will address the more complex case of solving directly for the control points, where calculus of variations cannot be employed to simplify the equation.

It might be reasonable to argue that for the case of Snakes, where the data providing the external forces, i.e., an image, is ultimately discrete, that numerical approaches are well-suited. However, as demonstrated by Cohen and Cohen (1990), greater accuracy and stability can be achieved by first fitting the discrete elements of the data with a continuous function, and then solving the entire system using FEM techniques. This approach has also been extended to the case of a deformable surface (Cohen and Cohen, 1992, 1993, Cohen, 1996). Recent Computer-Aided Mechanical Engineering research by Hughes et al. (2010) has called into question the use of FEM techniques for dealing with deformable spline representations. The FEM discretisation of the model results in an approximation of the solution, albeit a much more accurate solution than finite differences. Hughes et al. (2010) advocate working directly with the underlying geometry, applying gaussian quadrature to achieve exact solutions for B-Splines and

highly accurate solutions for NURBS. The mathematical framework developed in this thesis (presented in Chapter 5) addresses this issue.

Another area of potential confusion in the literature stems from the varying use of the term 'control point'. It is common in the literature for authors to refer to the discrete points along the snake as control points, e.g., Tiilikainen (2007). The term control point has a very specific meaning when dealing with spline curves, which are typically used as the underlying representation of a snake model. These are the control points of the spline rather than the 'sample' or 'discrete' points. In some cases, the forces are applied directly to spline control points which are then evolved to a minimum configuration. This practice is not ideal as for most spline representations, the control points do not lie on the surface of the deformable body. For some applications, the subtlety may not be apparent, particularly if many 'control points' are employed.

The literature in the areas of design and analysis also suffer from the same ambiguities. The literature is somewhat more consistent in approach and it is clear that numerical methods are favoured in general, FEM techniques being dominant (González-Hidalgo et al., 2013, Botsch and Sorkine, 2008). However, in many cases authors are unclear in presenting the detail of their solution. In Qin and Terzopoulos (1996), a milestone paper in Computer-Aided Geometric Design, Gaussian Quadrature is discussed as the method of choice for evaluating the integrations in their solution. The authors make no reference to how the derivatives are dealt with. The approach of Qin et al. already considered seminal in the literature for imparting physics-based dynamics on a NURBS based geometric substrate, now appears to also have been somewhat ahead of its time in its departure from FEM techniques as advocated in more recent research by Hughes et al. (2010).

Where dynamics are involved, time must also be considered. For computer vision applications and design-type environments the spatial accuracy is typically more important than the accuracy relating to the model's dynamics. Where time is concerned, stability is more important than accuracy. In such cases, less accurate and more computationally efficient numerical approaches, such as finite differences, are often used to simulate the dynamics. For computer graphics applications where realistic responses are desirable, more accurate numerical approaches may be employed to solve for the dynamics of the model.

A big advantage of numerical approaches is that they are generic such that they can be applied to a host of problems without modification or tailoring. However, this is also one of their biggest disadvantages. They do not utilise a-priori knowledge of the system and are often inefficient as a result. To achieve the levels of accuracy often deemed necessary, large numbers of discrete data points must be processed, resulting in computationally expensive algorithms. Analytic solutions, where possible, can offer significant computational savings (González-Hidalgo et al., 2013). This will be addressed further in Chapter 5.

3.8 B-Spline/NURBS in FEM vs IgA

B-Spline/NURBS representations are a compact way of defining a curved surface analytically. Such representations provide a mathematically exact representation of freeform surfaces that can be exactly reproduced whenever technically needed. For solving energy-based systems with B-Spline/NURBS–based geometric substrates, Finite Element Analysis is only an approximation of the geometry. There are many pitfalls associated with using polygonal representations to approximate curved boundaries. IgA is a non-standard numerical method for solving partial differential equations, which was introduced by Hughes et al. (2005). In the IgA framework, the ultimate goal is to adopt the parameterised geometry description and use it for the analysis, that is, within the model solver.

The use of B-Splines/NURBS as the basis for a numerical solver may raise questions regarding the analytic nature of the curve description. In IgA, those basis functions that represent the geometry also form the basis of the solution. While B-Spline/NURBS provide an analytic description of a surface, the basis functions can be thought of as a type of quantisation. The term 'discretising' implies loss of data and thus is inappropriate in this case. Rather the quantised elements represent exact analytic descriptions of finite pieces of the geometry. This is a by-product of the local modification properties of the geometry.

3.9 Conclusions

This chapter is primarily concerned with the mathematical modelling of energy-based deformations for computing. The equivalences between the various models is important and yet it is neglected in much of the literature relating to CAD, Computer Graphics, Machine Vision, and Visualisation such that assumed models are adopted without consideration to their origins. Alternative interpretations of the traditionally used models lead to confusion in some cases. To resolve this issue, apparent contrasting models are derived from first principles and equivalences are identified. Differences in models stem from application-dependent approximations which are formalised in the chapter.

It is also demonstrated that apparently contrasting models can become equivalent depending on the choice of solver. The chapter identifies potential problem areas, due to the lack of clarity in the literature, and presents some examples of cases where assumed models and solvers have led to a degree of confusion. In particular, Discrete and Analytic Models and Solvers are compared and contrasted, highlighting subtleties in their usage that are often neglected.

Overall this chapter demonstrates that the energy-based deformable models used across the plethora of application domains within computing are largely equivalent. Novelties presented in the literature across these domains therefore stem from the applications or efficiencies of the approach adopted, rather than the actual models

themselves. What may appear to be different models for describing deformations, and which are commonly considered as such in the literature, are each demonstrated to be derived from the same foundations of Continuum Mechanics. As a result, several areas of potential confusion relating to the various descriptions of an energy-based deformable model across the host of application domains are highlighted and clarified.

The lack of a seamless energy-based model that is purely analytic, capable of forming the basis for both design and analysis, that removes scope for confusion is ever more apparent. This issue will be addressed in the remaining chapters of this thesis.

Virtual Sculpting of an Active B-Spline/NURBS Surface Model

4

This chapter presents a novel and seamless ACM-based technique for the interactive Virtual Sculpting of Active B-Spline/NURBS freeform surfaces. Chapter 2 introduced Virtual Sculpting, a well established mechanism in freeform modelling environments that shields designers from the complicated mathematics and physics of shape deformation. By emulating traditional clay sculpting in an interactive environment, the task of manipulating complex models can be hidden behind the familiar action of moulding and manipulating inelastic substances, such as clay. Virtual Sculpting environments generally rely on combinations of various shape modelling and deformation techniques. Some physical properties can be mimicked using purely geometric approaches. However, such approaches do not provide a mechanism for simulating the real mechanics of a physical deformation. More intuitive energy minimisation approaches incorporate the principles of continuum mechanics and account for the material properties of the objects being deformed, but struggle to do so at interactive rates.

Used extensively in computer vision, ACMs represent a generalised approach for matching a deformable model to an image through the minimisation of an associated energy functional. In Chapter 3 it was established that ACMs are equivalent to linear approximations of the equations derived from continuum mechanics that offer a compromise between purely geometric and more exact energy-based approaches. In the approach presented in this chapter, ACMs are removed from the image domain and Active Surface Models are interactively sculpted by deforming the model, wholly or in part, to primitives. This approach forms the basis of a shaped tool-set to facilitate intuitive real-time energy based Virtual Sculpting in 3D space, analogous to the traditional clay modelling of real sculptors.

B-Spline/NURBS representations are a compact way of defining a curved surface analytically. Such representations provide a mathematically precise representation of freeform surfaces that can be exactly reproduced whenever technically needed. B-Spline/NURBS representations have become the *de facto* standard in industry for the representation of freeform surfaces. However, as outlined in Chapter 1, these representations have not yet fully benefited from the seamless embodiment of a true Virtual Sculpting paradigm. The analytic continuous mathematical description of B-Splines/NURBS makes them a particularly suitable choice as the underlying representation for the proposed ACM-based approach. Active B-Spline/NURBS representations are thus chosen as the underlying surface representation in this chapter.

Chapter 2 described the various approaches to Virtual Sculpting that have been developed in recent years. Previous work has incorporated physics based principles into a B-Spline/NURBS geometric substrate, e.g., Qin and Terzopoulos (1994), Terzopoulos and Qin (1994), and Qin and Terzopoulos (1996). When such models have been used, it has typically been in the context of generating a surface by smoothly interpolating a set of skeletal constraint curves (Celniker and Gossard, 1991, Qin and Terzopoulos, 1994). The definition of the surface rather than interactive sculpting of same has been the primary goal. Several attempts have been made to incorporate shaped Virtual Sculpting tools within CAD environments. However, the techniques proposed to date in the literature have adopted discrete models and rely on iterative conversions between the original CAD model and the employed discrete model. The disparate model representations introduce unnecessary seams into the design process. Additionally, this does not facilitate a smooth integration with existing CAD environments (Dachille IX et al., 2001, Gao and Gibson, 2006, Pungotra et al., 2010). Mullenhoffe (1998) employs a continuous model and discusses obstacle avoidance of a sphere which is akin to Virtual Sculpting. However, the obstacle avoidance is computed only after the surface has reached its minimum energy configuration subject to other constraints.

The approach developed in this chapter differs substantially from existing techniques in its application of shape derived forces, in the model evolution, and in its preservation of the analytic model. An additional novelty of the proposed ACM-based approach is that it facilitates the design of proximity-based tools that offer different interaction behaviours to those offered by collision-based sculpting techniques. This chapter does not aim to show Virtual Sculpting results, but rather sets out the theoretical development of the proposed Virtual Sculpting Approach to pave the way for the remaining chapters of this thesis.

4.1 Technical Background

4.1.1 B-Splines

A B-Spline curve is a composite, continuous, parametric curve formed with polynomial sections. The curve is represented by a set of control points which are used to weight a linear sum of associated basis functions. The basis functions ensure that the curve satisfies specified continuity conditions at the boundaries of each section. The parametric curve $C(u)$ is defined by

$$C(u) = \sum_{i=0}^{n} N_{i,p}(u) P_i \tag{4.1}$$

where P_i represents the set of $n+1$ control points and $N_{i,p}(u)$ are the basis functions.

For a B-Spline curve, the basis functions are defined by the Cox-deBoor recursion (Deboor, 1978)

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u)$$

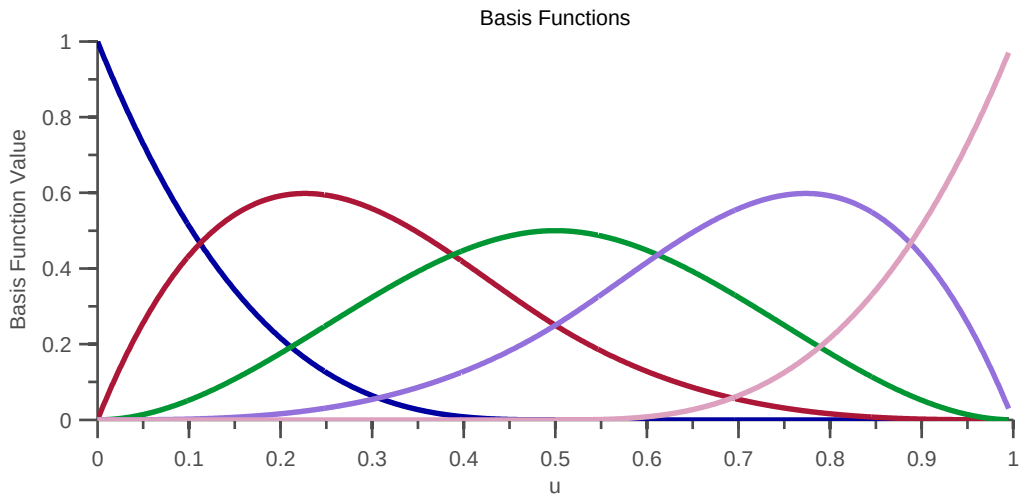


Figure 4.1: Cubic B-Spline Basis Functions

$$N_{i,0}(u) = \begin{cases} 1 & \text{if } u_i \leq u < u_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

$N_{i,p}(u)$ are the basis functions of degree p in the u parametric direction, defined over the knot vector U given by

$$U = \{\underbrace{0,\ldots,0}_{p+1}, u_{p+1},\ldots,u_{n-1}, \underbrace{1,\ldots,1}_{p+1}\} \tag{4.3}$$

The B-Spline basis functions are defined by a recursion that builds more complex high order basis functions from simpler lower order basis functions. The seed of the recursion is the 0^{th} order basis function $N_{i,0}$. The knot vector specifies the distribution of the parameter, u, along the curve. By convention, knots at the ends of the B-Spline are repeated $p+1$ times so that a B-Spline curve with $m+1$ knots will have $n+1$ control points where $m+1 = (n+1)+p+1$, and the range of u is limited to $u_p \leq u \leq u_{m-p}$.

Figure 4.1 shows an example set of cubic basis functions. Figure 4.2 shows the corresponding B-Spline curve as defined by the control points shown in red.

These equations can be generalised for higher dimensions. A surface model, $S(u,v)$, can be represented by

$$S(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} N_{i,p}(u)N_{j,q}(v)P_{i,j} \tag{4.4}$$

$$S(u,v) = \begin{bmatrix} N_{0,p}(u)\cdots N_{m,p}(u) \end{bmatrix} \begin{bmatrix} P_{0,0} & \cdots & P_{0,n} \\ \vdots & \ddots & \vdots \\ P_{m,0} & \cdots & P_{m,n} \end{bmatrix} \begin{bmatrix} N_{0,q}(v)\cdots N_{n,q}(v) \end{bmatrix}^T$$

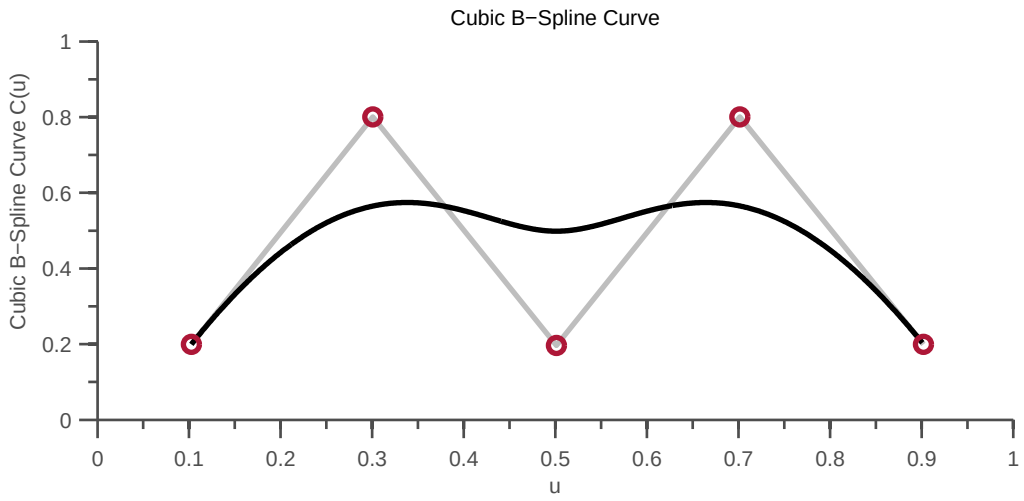


Figure 4.2: Cubic B-Spline Curve

$$= \begin{bmatrix} N_{0,p}(u)N_{0,q}(v) \dots N_{m,p}(u)N_{n,q}(v) \end{bmatrix} \begin{bmatrix} P_{0,0} \\ \vdots \\ P_{m,n} \end{bmatrix} \tag{4.5}$$

4.1.2 NURBS

A NURBS curve can be thought of as a projection onto n-dimensional space of a B-Spline curve defined in $(n+1)$-dimensional homogeneous co-ordinate space. The definition of a NURBS curve is much the same as that of the non-rational B-Spline curve outlined in Section 4.1.1. However, in the case of NURBS, the basis functions are rational such that a NURBS basis function is defined by

$$R_{i,p}(u) = \frac{N_{i,p}(u)w_i}{\sum_{i=0}^{n} N_{i,p}(u)w_i} \tag{4.6}$$

where the new parameter, w, represents a set of $n+1$ weights associated with the control points.

Analogous to the B-Spline equations developed in Section 4.1.1, the basis functions for a parametric NURBS surface are defined as

$$R_{i,j}^{p,q}(u,v) = \frac{N_{i,p}(u)N_{j,q}(v)w_{i,j}}{\sum_{i=0}^{m}\sum_{j=0}^{n} N_{i,p}(u)N_{j,q}(v)w_{i,j}} \tag{4.7}$$

In this way, NURBS can be thought of as a generalisation of B-Spline representations. The additional degrees of freedom afforded by the new weight parameter $w_{i,j}$ offer greater flexibility in design, such that a wider variety of shape descriptions are possible, e.g., unlike B-Splines, NURBS facilitate the representation of a perfect circle. Further information on NURBS can be found in Piegl and Tiller (1997).

4.2 Active B-Spline Surface Model

As outlined in Chapter 3, the energy-based deformable models used across the plethora of application domains within computing are largely equivalent. It was noted that true novelties presented in the literature across these domains therefore stem from the applications or efficiencies of the approaches adopted, rather than from the actual models themselves. In this section, an Active B-Spline Surface Model is presented. This section does not claim novelty in the idea of an Active B-Spline Surface model itself. It is quite clear from the literature that such concepts date back to the early 1980s. However, in the literature, only very high level descriptions are provided of the mathematical mechanics that form the basis for such models. As a consequence, it is difficult for practitioners to develop a deeper understanding of the mathematics and identify potential opportunities for improvement, without first developing the equations in full. This task is not trivial and is obfuscated in much of the literature by attempts to replace integrals with approximating summations without providing rationale, e.g., Ivins (1996). Therefore, full treatment of the equations is provided here. As the goal of this thesis is to preserve the continuous representation of the model without discretisation, the full analytic mechanics of the equations are required. The techniques discussed extend to the NURBS case, unless otherwise stated, by simply swapping in the NURBS basis functions in place of the B-Spline basis functions.

4.2.1 Energy Model

The energy model for the Active Surface adopted in this thesis is defined, using the Differential Geometry approach outlined in Chapter 3, as

$$E(S(u,v)) = \int_{\sigma} \alpha_{11}\left|\frac{\partial S}{\partial u}\right|^2 + 2\alpha_{12}\frac{\partial S}{\partial u}\bullet\frac{\partial S}{\partial v} + \alpha_{22}\left|\frac{\partial S}{\partial v}\right|^2 + \beta_{11}\left|\frac{\partial^2 S}{\partial u^2}\right|^2 + 2\beta_{12}\left|\frac{\partial^2 S}{\partial u\partial v}\right|^2 + \beta_{22}\left|\frac{\partial^2 S}{\partial v^2}\right|^2 dudv \quad (4.8)$$

where E is the internal energy equation associated with the parametric surface $S(u,v)$, and α and β are weights that control the material properties of the surface and σ represents the domain of the integration.

Adopting a B-Spline surface representation for $S(u,v)$ given by

$$S(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} N_{i,p}(u)N_{j,q}(v)P_{i,j} \quad (4.9)$$

and given the derivative of a B-Spline Surface as described in Piegl and Tiller (1997) as

$$\frac{\partial^{k+l}S(u,v)}{\partial^k u\partial^l v} = \sum_{i=0}^{m}\sum_{j=0}^{n} N_{i,p}^{(k)}(u)N_{j,q}^{(l)}(v)P_{i,j} \quad (4.10)$$

the B-Spline active surface equation becomes

$$E(S(u,v)) =$$

$$\int_\sigma \alpha_{11}\left|\sum_{i=0}^{m}\sum_{j=0}^{n} N_{i,p}^{(1)}(u)N_{j,q}(v)P_{i,j}\right|^2 + 2\alpha_{12}\left[\sum_{i=0}^{m}\sum_{j=0}^{n} N_{i,p}^{(1)}(u)N_{j,q}(v)P_{i,j}\right]\bullet\left[\sum_{i=0}^{m}\sum_{j=0}^{n} N_{i,p}(u)N_{j,q}^{(1)}(v)P_{i,j}\right] + \alpha_{22}\left|\sum_{i=0}^{m}\sum_{j=0}^{n} N_{i,p}(u)N_{j,q}^{(1)}(v)P_{i,j}\right|^2$$

$$+ \beta_{11}\left|\sum_{i=0}^{m}\sum_{j=0}^{n} N_{i,p}^{(2)}(u)N_{j,q}(v)P_{i,j}\right|^2 + 2\beta_{12}\left|\sum_{i=0}^{m}\sum_{j=0}^{n} N_{i,p}^{(1)}(u)N_{j,q}^{(1)}(v)P_{i,j}\right|^2 + \beta_{22}\left|\sum_{i=0}^{m}\sum_{j=0}^{n} N_{i,p}(u)N_{j,q}^{(2)}(v)P_{i,j}\right|^2 dudv \quad (4.11)$$

In matrix form, this gives

$$E(S(u,v)) =$$

$$\int_\sigma \begin{bmatrix} P_{0,0} \\ \vdots \\ P_{m,n} \end{bmatrix}^T \begin{bmatrix} N_{0,p}^{(1)}(u)N_{0,q}(v) & \cdots & N_{m,p}^{(1)}(u)N_{n,q}(v) \\ N_{0,p}(u)N_{0,q}^{(1)}(v) & \cdots & N_{m,p}(u)N_{n,q}^{(1)}(v) \end{bmatrix}^T \begin{bmatrix} \alpha_{11} & \alpha_{12} \\ \alpha_{21} & \alpha_{22} \end{bmatrix} \begin{bmatrix} N_{0,p}^{(1)}(u)N_{0,q}(v) & \cdots & N_{m,p}^{(1)}(u)N_{n,q}(v) \\ N_{0,p}(u)N_{0,q}^{(1)}(v) & \cdots & N_{m,p}(u)N_{n,q}^{(1)}(v) \end{bmatrix} \begin{bmatrix} P_{0,0} \\ \vdots \\ P_{m,n} \end{bmatrix}$$

$$+ \begin{bmatrix} P_{0,0} \\ \vdots \\ P_{m,n} \end{bmatrix}^T \begin{bmatrix} N_{0,p}^{(2)}(u)N_{0,q}(v) & \cdots & N_{m,p}^{(2)}(u)N_{n,q}(v) \\ N_{0,p}^{(1)}(u)N_{0,q}^{(1)}(v) & \cdots & N_{m,p}^{(1)}(u)N_{n,q}^{(1)}(v) \\ N_{0,p}(u)N_{0,q}^{(2)}(v) & \cdots & N_{m,p}(u)N_{n,q}^{(2)}(v) \end{bmatrix}^T \begin{bmatrix} \beta_{11} & 0 & 0 \\ 0 & \beta_{12} & 0 \\ 0 & 0 & \beta_{22} \end{bmatrix} \begin{bmatrix} N_{0,p}^{(2)}(u)N_{0,q}(v) & \cdots & N_{m,p}^{(2)}(u)N_{n,q}(v) \\ N_{0,p}^{(1)}(u)N_{0,q}^{(1)}(v) & \cdots & N_{m,p}^{(1)}(u)N_{n,q}^{(1)}(v) \\ N_{0,p}(u)N_{0,q}^{(2)}(v) & \cdots & N_{m,p}(u)N_{n,q}^{(2)}(v) \end{bmatrix} \begin{bmatrix} P_{0,0} \\ \vdots \\ P_{m,n} \end{bmatrix} dudv \quad (4.12)$$

or more compactly (the standard notation in the literature (Celniker and Welch, 1992)),

$$E(S(u,v)) = \int_{\sigma} P^T \, \Phi_s^T \, \overline{\alpha} \, \Phi_s P \, + P^T \, \Phi_b^T \, \overline{\beta} \, \Phi_b \, P \, dudv \tag{4.13}$$

where

$$\mathbf{P}^T = [P_{0,0} \quad \cdots \quad P_{m,n}]$$

$$\Phi = \begin{bmatrix} N_{0,p}(u)N_{0,q}(v) & \cdots & N_{m,p}(u)N_{n,q}(v) \end{bmatrix}$$

$$\mathbf{\Phi}_s = \begin{bmatrix} \mathbf{\Phi}_u \\ \mathbf{\Phi}_v \end{bmatrix} \quad \mathbf{\Phi}_b = \begin{bmatrix} \mathbf{\Phi}_{uu} \\ 2\mathbf{\Phi}_{uv} \\ \mathbf{\Phi}_{vv} \end{bmatrix}$$

$$\overline{\boldsymbol{\alpha}} = \begin{bmatrix} \alpha_{11} & \alpha_{12} \\ \alpha_{21} & \alpha_{22} \end{bmatrix} \quad \overline{\boldsymbol{\beta}} = \begin{bmatrix} \beta_{11} & 0 & 0 \\ 0 & \beta_{12} & 0 \\ 0 & 0 & \beta_{22} \end{bmatrix}$$

This can be written as

$$E(S(u,v)) = \mathbf{P}^T \, \mathbf{K}_\sigma \, \mathbf{P} \tag{4.14}$$

where

$$\mathbf{K}_\sigma = \int_{\sigma}^{'} \mathbf{\Phi_s}^T \, \overline{\boldsymbol{\alpha}} \, \mathbf{\Phi_s} + \, \mathbf{\Phi_b}^T \, \overline{\boldsymbol{\beta}} \, \mathbf{\Phi_b} \, dudv \tag{4.15}$$

Equations 4.8–4.15, inclusive, describe the internal energy of an Active B-Spline surface. The model evolution is driven by internal forces generated by an energy minimisation that will be discussed in the next section.

4.2.2 Internal Force Model

In order to minimise the energy, the set of control points $P_{a,b}$ must be found such that

$$\forall a \in (0, 1 \ldots m), b \in (0, 1 \ldots n) : \frac{\partial S}{\partial P_{a,b}} = 0 \tag{4.16}$$

Differentiating the energy function, E, with respect to the control points, P, gives

$$\begin{aligned}
&\frac{\partial E(S(u,v))}{\partial P_{a,b}} = \\
&\int_\sigma \alpha_{11} N^{(1)}_{a,p}(u) N_{b,q}(v) \sum_{i=0}^{m}\sum_{j=0}^{n} N^{(1)}_{i,p}(u) N_{j,q}(v) P_{i,j} + \alpha_{22} N_{a,p}(u) N^{(1)}_{b,q}(v) \sum_{i=0}^{m}\sum_{j=0}^{n} N_{i,p}(u) N^{(1)}_{j,q}(v) P_{i,j} \\
&+ 2\alpha_{12} \left(N_{a,p}(u) N^{(1)}_{b,q}(v) \sum_{i=0}^{m}\sum_{j=0}^{n} N^{(1)}_{a,p}(u) N_{b,q}(v) P_{i,j} + N^{(1)}_{a,p}(u) N_{b,q}(v) \sum_{i=0}^{m}\sum_{j=0}^{n} N_{i,p}(u) N^{(1)}_{j,q}(v) P_{i,j} \right) \\
&+ \beta_{11} N^{(2)}_{a,p}(u) N_{b,q}(v) \sum_{i=0}^{m}\sum_{j=0}^{n} N^{(2)}_{i,p}(u) N_{j,q}(v) P_{i,j} + \beta_{22} N_{a,p}(u) N^{(2)}_{b,q}(v) \sum_{i=0}^{m}\sum_{j=0}^{n} N_{i,p}(u) N^{(2)}_{j,q}(v) P_{i,j} \\
&+ 2\beta_{12} N^{(1)}_{a,p}(u) N^{(1)}_{b,q}(v) \sum_{i=0}^{m}\sum_{j=0}^{n} N^{(1)}_{i,p}(u) N^{(1)}_{j,q}(v) P_{i,j} du dv = 0
\end{aligned} \tag{4.17}$$

Or in matrix representation

$$\begin{aligned}
&\frac{\partial E(S(u,v))}{\partial P} = \\
&\int_\sigma \begin{bmatrix} N^{(1)}_{0,p}(u) N_{0,q}(v) & \cdots & N^{(1)}_{m,p}(u) N_{n,q}(v) \\ N_{0,p}(u) N^{(1)}_{0,q}(v) & \cdots & N_{m,p}(u) N^{(1)}_{n,q}(v) \end{bmatrix}^T \begin{bmatrix} \alpha_{11} & \alpha_{12} \\ \alpha_{21} & \alpha_{22} \end{bmatrix} \begin{bmatrix} N^{(1)}_{0,p}(u) N_{0,q}(v) & \cdots & N^{(1)}_{m,p}(u) N_{n,q}(v) \\ N_{0,p}(u) N^{(1)}_{0,q}(v) & \cdots & N_{m,p}(u) N^{(1)}_{n,q}(v) \end{bmatrix} \begin{bmatrix} P_{0,0} & \cdots & P_{m,n} \end{bmatrix}^T \\
&+ \begin{bmatrix} N^{(2)}_{0,p}(u) N_{0,q}(v) & \cdots & N^{(2)}_{m,p}(u) N_{n,q}(v) \\ N^{(1)}_{0,p}(u) N^{(1)}_{0,q}(v) & \cdots & N^{(1)}_{m,p}(u) N^{(1)}_{n,q}(v) \\ N_{0,p}(u) N^{(2)}_{0,q}(v) & \cdots & N_{m,p}(u) N^{(2)}_{n,q}(v) \end{bmatrix}^T \begin{bmatrix} \beta_{11} & 0 & 0 \\ 0 & \beta_{12} & 0 \\ 0 & 0 & \beta_{22} \end{bmatrix} \begin{bmatrix} N^{(2)}_{0,p}(u) N_{0,q}(v) & \cdots & N^{(2)}_{m,p}(u) N_{n,q}(v) \\ N^{(1)}_{0,p}(u) N^{(1)}_{0,q}(v) & \cdots & N^{(1)}_{m,p}(u) N^{(1)}_{n,q}(v) \\ N_{0,p}(u) N^{(2)}_{0,q}(v) & \cdots & N_{m,p}(u) N^{(2)}_{n,q}(v) \end{bmatrix} \begin{bmatrix} P_{0,0} & \cdots & P_{m,n} \end{bmatrix}^T du dv = 0
\end{aligned} \tag{4.18}$$

The expression can be represented more compactly by

$$\frac{\partial E(S(u,v))}{\partial P} = \int_{\sigma} \mathbf{\Phi_s}^T \, \overline{\boldsymbol{\alpha}} \, \mathbf{\Phi_s P} \; + \mathbf{\Phi_b}^T \, \overline{\boldsymbol{\beta}} \, \mathbf{\Phi_b} \, \mathbf{P} \, dudv = 0 \tag{4.19}$$

which can be gathered as follows

$$\mathbf{K}_{\sigma}\mathbf{P} = 0 \tag{4.20}$$

where $\mathbf{K}$ is an $(m \times n) \times (m \times n)$ matrix known as the stiffness matrix[3].

This system of mxn equations is gathered resulting in the following; a matrix $\mathbf{A}$, comprising the basis function information; column vector $\mathbf{P}$, comprising the x, y, and z coordinates of the mxn unknown control points positions; and matrix $\mathbf{F}$, comprising the external force information.

$$\mathbf{AP} + \mathbf{F} = 0 \tag{4.21}$$

$\mathbf{A}$ notation is adopted as this is the standard notation adopted in the literature and by numerical texts (Press et al., 2007) when dealing with the complete system to be solved. In the case presented here $\mathbf{A}$ simply comprises the $\mathbf{K_{\alpha}}$, the stiffness matrix. $\mathbf{A}$ is sparse due to the local control properties of the B-Spline basis functions.

4.2.3 Dynamics

Where dynamics are desired, to afford the user interactive control over the convergence, the system can be made a function of time, t. Using an ACM-based approach, the evolution of the model is controlled by using an implicit Euler integration scheme (Kass et al., 1987) as follows

$$\mathbf{P}_t = (\mathbf{A} + \lambda \mathbf{I})^{-1}(\lambda \mathbf{P}_{t-1} - \mathbf{F}_{t-1}) \tag{4.22}$$

The solution is then iterated, where λ is the user-definable step size, until the desired deformation is achieved.

4.3 Sculpting Tools

In the absence of external forces/constraints, the model described will collapse to a point under the influence of its own internal forces. Force effects can be generated by supplying the system with a force distribution of the form

[3]The $\frac{1}{2}$ and 2 constant multipliers found in energy and force equations are generally encompassed by the alpha and beta parameters.

$$\mathbf{F}_\sigma = \int_\sigma \Phi^T f \, dudv \tag{4.23}$$

where $\mathbf{F}$ is the forcing matrix, σ, the domain of integration, ϕ, as defined in Equation 4.13, and $\mathbf{f}$ the force function.

Calculating the integral exactly requires a unique calculation for every type of forcing function applied. A virtual tool is therefore the analytic description of the force distribution. An example of such a function is connecting a material point (u_0, v_0) of the surface to a point d_0 in space (Qin and Terzopoulos, 1994). This is equivalent to the spring and volcano type forces discussed in (Kass et al., 1987) for guiding an active contour to the desired rest configuration.

$$f(u,v) = \int\int k(d_0 - S(u,v))\delta(u - u_0, v - v_0)dudv \tag{4.24}$$

where δ is the unit delta function, k is the spring constant according to Hooke's law, and d_0 is the point in space. This can be generalised to a unit step or smooth distribution (Qin and Terzopoulos, 1994).

The sculpting tools proposed for the approach developed in this thesis are applied forces that are calculated based on the Euclidean distance between the Active B-Spline Surface model and a shaped primitive. Each shaped primitive/tool calculates its distance from the Active B-Spline Surface. The tool surface is then connected to the Active B-Spline surface via an ideal Hookean spring relationship, thus defining the external force distribution function, $f(u,v)$.

$$f(u,v) = k(T - S(u,v)) \tag{4.25}$$

where $S(u,v)$ represents the Active Surface and T represents the tool surface.

The response of the surface to these applied forces can be adjusted by modifying the values of the α and β parameters that control the surface's resistance to stretch and bend under the action of the tool.

4.4 Preserving the Model Integrity

The successful implementation of the continuous model presented in this chapter relies heavily on its computational efficiency. While the ACM approach itself contributes to a computationally efficient solution, it is essential that the solver is computationally efficient. Numeric techniques such as FDM/FEM could be used to numerically solve the system, or the more accurate approach of IgA with Gaussian Quadrature could be employed. However, numeric techniques are computationally expensive. This section

breaks Equation 4.13, presented in Section 4.2, into simpler terms so that an analytic solution may be sought.

The **K** matrix described in Section 4.2 can be separated into a sum of integrals. To exemplify the breakdown, the first term of the stiffness matrix (the stretch term), is analysed

$$\int_{\sigma} \Phi_u^T \, \overline{\alpha} \, \Phi_u \, dudv \tag{4.26}$$

As the function is integrated with respect to u, the basis functions of the v parameter can be treated as constants. Breaking this term down gives

$$\int_{\sigma} \alpha_{11}\Phi_u^2 + \alpha_{12}\Phi_u\Phi_v + \alpha_{22}\Phi_v^2 \, dudv \tag{4.27}$$

Taking the first term

$$\int_{\sigma} \alpha_{11} \, \Phi_u^2 \, dudv \tag{4.28}$$

and expanding gives

$$\int_{\sigma} \alpha_{11} \begin{bmatrix} N_{0,p}^{(1)}(u)N_{0,q}(v) & \cdots & N_{m,p}^{(1)}(u)N_{n,q}(v) \end{bmatrix}^T \begin{bmatrix} N_{0,p}^{(1)}(u)N_{0,q}(v) & \cdots & N_{m,p}^{(1)}(u)N_{n,q}(v) \end{bmatrix} \, dudv \tag{4.29}$$

A typical term in the resulting $(m \times n) \times (m \times n)$ matrix is

$$N_{w,p}^{(1)}(u)N_{x,q}(v)N_{y,p}^{(1)}(u)N_{z,q}(v) \tag{4.30}$$

where $w, y \in (0, 1, \ldots, m)$ and $x, z \in (0, 1, \ldots, n)$. To integrate the term over the surface, we can integrate with respect to each parameter separately. Each of the remaining integration terms will have a similar form. As the integral is separable, the first integration over the u parameter reduces to

$$\int N_{w,p}^{(1)}(u)N_{y,p}^{(1)}(u) \, du \tag{4.31}$$

Chapter 5 investigates an analytic solution to the stiffness matrix, based on this reduction. The practical implementation of the approach will be discussed in Chapter 6.

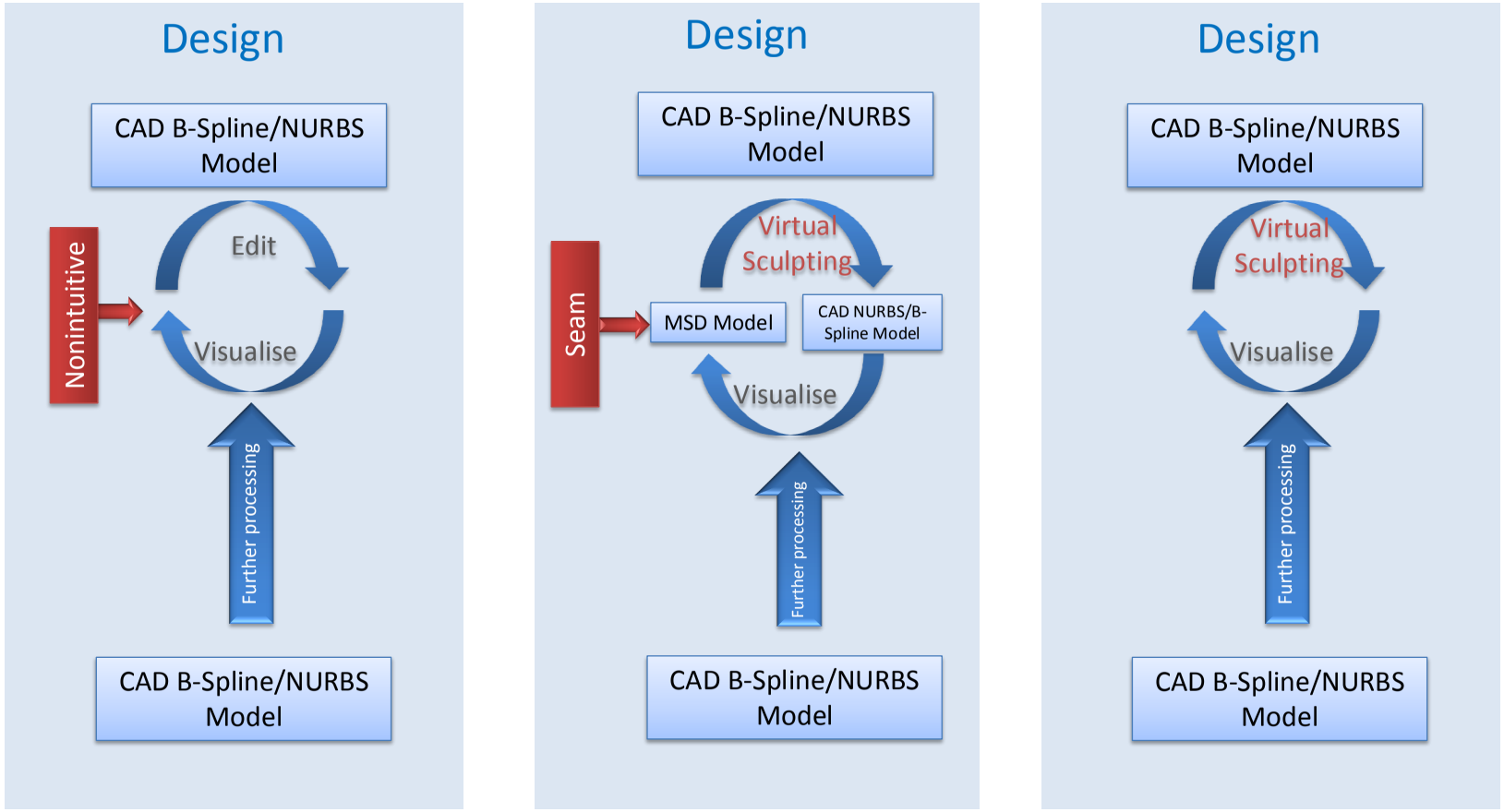
4.5 Conclusion

In this chapter, a novel approach to Virtual Sculpting, based on a reformulation of ACMs, is developed using Active B-Spline/NURBS Surface models. In the proposed technique, ACMs are removed from their traditional image domain and predefined shape primitives provide the features of interest. These primitives act as sculpting tools, providing visual guides that facilitate intuitive deformation results, such that the overall approach is analogous to traditional clay modelling. The interactive Virtual Sculpting of Active B-Spline/NURBS Surfaces enabled by the proposed approach offers a more intuitive alternative to complex and tedious manual control point manipulations and also facilitates data exchange between existing CAD systems. Unlike existing Virtual Sculpting approaches, the model takes its inspiration from the traditional ACM: The forces are derived based on proximity to shaped sculpting tools and the system is made a function of time to afford the user control over its convergence. The proximity-based approach facilitates the design of non-contact tools that offer different interaction behaviours to those offered by collision based sculpting techniques; for example, consider the contrasting behaviour of a contact and non-contact sculpting tool incorporating a concavity.

In Chapter 1, two challenges were identified relating to the creative design of free-form surfaces. Firstly, the traditional design process suffers from complex and unintuitive design mechanisms, necessitating prerequisite expertise to achieve even conceptually simple designs. Secondly, existing Virtual Sculpting approaches, developed to alleviate this problem, rely on contrasting representation technologies to the existing CAD models and thus preclude a seamless integration with the existing CAD workflow. These issues are illustrated in Figure 4.3(a) and Figure 4.3(b) respectively.

The approach presented in this chapter addresses these two issues by presenting, for the first time, a Virtual Sculpting framework that operates directly on the continuous analytic representation of existing CAD models. The main benefit of the proposed technique is that it facilitates the seamless integration of a Virtual Sculpting metaphor within existing CAD environments. The situation facilitated by the approach developed is depicted by Figure 4.3(c).

In order for the system to be feasible for interactive freeform shape design, the system must be solved in a computationally efficient manner. Chapter 5 investigates an analytic solution to the stiffness matrix, based on the element-wise decomposition of the stiffness matrix presented in Section 4.4. The practical implementation of the approach will be discussed in Chapter 6.



(a) Traditional Non-intuitive Design Pipeline (b) Existing Virtual Sculpting (c) Seamless Virtual Sculpting

Figure 4.3: The Design Pipeline: Non-intuitive modelling procedures and divisions in representation technologies hinder the design process

Generalised Computationally Efficient Mathematical Framework

5

The use of techniques based on energy minimisation for the implementation of physically aware deformable models is widespread across the domains of Computer Graphics, Computer Aided Design and Computer Vision. Due primarily to the myriad of applications, the study of deformable models is one of the most active research topics in computer graphics.

The key component to imparting physical properties on a geometric model is the construction of the stiffness matrix. The generation of the stiffness matrix is one of the more time-consuming operations in computer-based modelling applications. For interactive applications such as Virtual Sculpting or prototyping, performance is key, and waiting long periods of time for models to respond to changes made is highly undesirable. George Allen, Chief Technologist and Technical Fellow at Siemens PLM Software, recently addressed the academic research community, where he talked about what he saw as the big geometric modelling problems in industrial CAD/CAM/CAE software (Allen, 2007). Allen cited performance as one of the key issues, stating that 'Lack of real-time response makes some exploratory functions unusable, and this impairs user creativity'.

In CAD/CAGD/CAM/CAE, computationally efficient bi-parametric B-Spline representations, particularly NURBS, are the dominant technology. The study of efficient techniques for the construction of the stiffness matrix has been an active area of research since the introduction of computer-based deformable models in the early 1980s. The recent popularity gain experienced by IgA has instigated a spurt of new research activity for the specific problem of minimising the energy of an Active B-Spline/NURBS Surface (González-Hidalgo et al., 2013, Rypl and Patzák, 2012, Hughes et al., 2010).

There are several popular approaches ranging from indirect approaches that operate on the geometry rather than the controls to approaches that target the controls directly. Irrespective of the approach adopted, numerical methods (Press et al., 2007) are often favoured for their 'black-box' solution with universal applicability. A disadvantage of such generic numerical approaches is that they typically do not fully utilise a-priori knowledge of the system. Consequently, to achieve the levels of accuracy often deemed necessary, large numbers of discrete data points must be processed, resulting in computationally expensive algorithms. Analytic solutions, where possible, can offer significant computational savings (González-Hidalgo et al., 2013).

From the literature, Gaussian Quadrature, a numerical technique, is identified

as the 'state of the art'. While the general Gaussian Quadrature technique is well documented, the literature lacks implementation details for the specific problem of generating the stiffness matrix of a B-Spline surface. To fill this gap in the literature, and to facilitate meaningful comparisons of results, the general problem specific efficiencies are first considered. An efficient Gaussian Quadrature algorithm is then developed and presented that capitalises on both the problem specific efficiencies identified, and also further approach specific efficiencies that can be gained by tailoring the approach.

This chapter aims to further address the problem of computational expense and presents a novel, generalised efficient analytic mathematical framework complete with accompanying algorithms for the generation of the stiffness matrix associated with an Active B-Spline surface model. The approach is shown to generalise to the problems of computing mass, damping, and forcing matrices. It is also demonstrated that the approach can cope with variable mass, damping, and stiffness coefficients. No assumptions regarding the problem complexity, degree, or regularity of the knot vector are imposed on the solution. Further to this is its extension for the important special case of an Active NURBS surface using the approximation properties of B-Splines (Hughes et al., 2010, Schumaker, 2007). Detailed analysis of the integral and repeated integral of a B-Spline basis function, areas which are lacking coverage in the literature, is presented and algorithms for computing such integrals are developed. To illustrate the capabilities of the algorithms and verify their respective performances, several case studies are presented and detailed analysis of the computational efficiency, accuracy and stability is undertaken. Metric comparisons between Gaussian Quadrature results, Analytic results, and Exact 'ground-truth' results obtained using symbolic analysis are presented and discussed.

The main benefit of the technique is the resulting algorithm that takes advantage of the reduced computational complexity of the analytic approach leading to a significant reduction in the computation time required to evaluate the stiffness matrix. The algorithm developed is up to 4.3 times faster than the Gaussian Quadrature approach for the practical cases considered.

5.1 Overview of Existing Approaches

Various approaches are employed to evaluate the stiffness matrix in the context of minimising the energy of an Active B-Spline Surface. Indirect approaches work with points on the B-Spline surface, while more direct approaches target the minimisation of the actual degrees of freedom of the model, i.e., the control point positions. In either approach, numerical methods are typically used to evaluate the integrals and derivatives involved.

5.1.1 Indirect Approaches

For indirect approaches, where the surface is discretised so that discrete surface points are to be optimised, the Euler Lagrange equation can be employed to simplify the equation to one involving derivatives only.

The energy is minimised by finding the surface, $S(u, v)$, that satisfies the corresponding Euler-Lagrange equation given by

$$-\alpha_{11}\frac{\partial^2 S}{\partial u^2} - 2\alpha_{12}\frac{\partial^2 S}{\partial u \partial v} - \alpha_{22}\frac{\partial^2 S}{\partial v^2} + 2\beta_{12}\frac{\partial^4 S}{\partial u^2 \partial v^2} + \beta_{11}\frac{\partial^4 S}{\partial u^4} + \beta_{22}\frac{\partial^4 S}{\partial v^4} = 0 \qquad (5.1)$$

Introduced to the domain of computer vision by Kass et al. (1987) and generalised to the 3D case for computer graphics applications by Terzopoulos and Fleischer (1988), this technique is prominent in the literature and has been adopted by the computer-aided design communities in the area of B-Spline surface design (Celniker and Gossard, 1991, Welch and Witkin, 1992). Recent studies have examined analytic solutions to the partial differential equation for B-Splines (González-Hidalgo et al., 2013). However, numerical approaches such as finite differences and finite elements remain the most popular choice in solving for the minimum surface point configuration (González-Hidalgo et al., 2013, Botsch and Sorkine, 2008).

A similar discrete approach to the modelling of deformable B-Splines involves representing the surface using a mass spring damper model. Linear springs are attached between the discrete data points such that the minimal surface configuration can be found by evolving the data points under the influence of mass spring damper model forces. Typically, a B-Spline description is sought as the final representation of the deformed model and a new B-Spline surface is fitted to the new data points using a global interpolation scheme (Gao and Gibson, 2006, Pungotra et al., 2010).

5.1.2 Direct Approaches

Unlike the indirect, discrete approaches described above, more accurate direct approaches involve employing the continuous description of the B-Spline Surface representation to specify the geometry in the energy equations. In such cases, the control point positions are directly optimised and the Euler-Lagrange approach cannot be used to eliminate the integral. Numerical integration approaches are thus employed to solve for the minimal energy configuration.

A wide range of numerical approaches are available with varying degrees of accuracy. For B-Spline surfaces, given their polynomial nature, Gaussian Quadrature with Gauss-Legendre calculated weights and abscissae is typically cited as the appropriate choice (Hughes et al., 2010, Schumaker, 2007). In general, Gaussian quadrature evaluates the integral exactly with N weights and abscissae for polynomials of degree $2N - 1$.

5.1.3 Discussion

From the literature, Gaussian Quadrature is the 'state of the art' technique for dealing with the exact energy minimisation of Active B-spline surfaces (Auricchio et al., 2012, Hughes et al., 2010, Cottrell et al., 2009). However, where the number of points in a deformable model is large, integrating the energy equations using Gaussian Quadrature is a computationally intensive process. With the growing popularity of Isogeometric Analysis as an alternative to Finite Element Analysis, the study of direct approaches has regained momentum and has become an important research area as there is a growing need for more efficient solutions (Hughes et al., 2010). To this end, much research has been devoted to designing efficient quadrature rules for B-Spline representations (Cottrell et al., 2009, Hughes et al., 2010, Rypl and Patzák, 2012). Approaches adopted to alleviate the computational costs typically involve taking advantage of simplifications that come into play where the B-Spline functions have a high degree of regularity, or are defined over a uniform knot vector. In this case, the majority of the B-Spline basis functions are simply translated versions of a single function. Taking such simplifications into account, more efficient quadrature rules can be devised (Hughes et al., 2010), or 'look-up tables' can be employed storing only the unique results (Mullenhoffe, 1998). This chapter examines the more general case.

5.2 Problem Specific Efficiencies

As outlined in Chapter 4 Section 4.2, the assembly of the stiffness matrix involves the summation of integral terms representing different properties of the active surface. The tensor product construction of the B-Spline surface means that each integral can be decomposed into its constituent parametric elements such that each can be treated separately for the purpose of evaluating the integrals before being recombined in the overall stiffness matrix (Mullenhoffe, 1998, Hughes et al., 2010). The overall problem reduces to the calculation of integrals of the form

$$\int N_{i,p}(u) N_{j,p}(u)\, du \tag{5.2}$$

$$\int N^{(1)}_{i,p}(u) N^{(1)}_{j,p}(u)\, du \tag{5.3}$$

$$\int N^{(2)}_{i,p}(u) N^{(2)}_{j,p}(u)\, du \tag{5.4}$$

$$\int N^{(1)}_{i,p}(u) N_{j,p}(u)\, du \tag{5.5}$$

Equations 5.2–5.5, inclusive, represent four distinct terms that must be computed for each possible combination of i and j in both the u and v parametric directions (See Chapter 4 for further detail on their origins). In linear algebra, the resulting matrices are known as Gramian matrices (Schumaker, 2007).

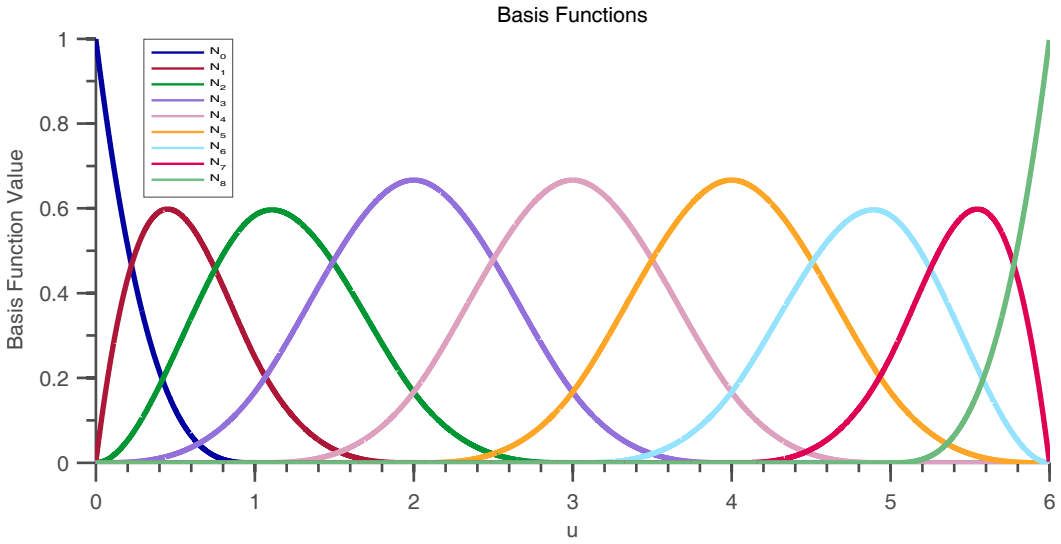


Figure 5.1: Cubic B-Spline Basis Functions on $U = \{0, 0, 0, 0, 1, 2, 3, 4, 5, 6, 6, 6, 6\}$

The Gramian matrix for each term exhibits certain properties such as symmetry, of which advantage can be taken to reduce the number of computations required. Additionally, by the support properties of B-Splines, the Gramian matrix for a B-Spline curve defined on a knot vector of degree p is $2(p+1)-1$ banded and thus, many of the elements are zero and do not need to be computed.

While the above properties are mentioned in Schumaker (2007), the literature lacks detail on how best to take advantage of these properties. This section addresses this gap in the literature and identifies efficiencies, based on these properties, that can be adopted regardless of the approach used to tackle the stiffness matrix generation.

B-Spline basis functions have limited support regions. This means that a basis function is non zero over a limited number of spans, i.e., $p+1$. Figure 5.1 shows an example set of cubic basis functions defined over knot vector $U = \{0, 0, 0, 0, 1, 2, 3, 4, 5, 6, 6, 6, 6\}$. From the diagram, it is clear that the product $N_{i,p}\ N_{j,p}$, where p in this case is 3, is only non zero where $i - p \leq j \leq i + p$. The local support feature means that a stencil [4] can be designed that starts at the first basis function and then moves diagonally along the Gramian matrix facilitating the computation of only non-zero products. Figure 5.2 shows a graphical representation of this stencil that incorporates every possible non-zero product combination for a particular basis function and its p higher index neighbours. As can be seen, the stencil terms are symmetric. By traversing the template across each Gramian matrix, as shown in Figure 5.3, the unique non-zero values of the Gramian matrix are identified.

Building on these problem specific efficiencies, the following sections present two different approaches to computing the four Gramian matrices that make up the overall stiffness matrix. Full detail of how best to take advantage of the properties and stencils described are presented for each approach.

[4] The term stencil is used to refer to a mask or template that represents a common process to be applied repeatedly at different locations.

5.3 Gaussian Quadrature Approach

The first approach considered is the state of the art Gaussian Quadrature approach. While the implementation of the Gaussian Quadrature algorithm was initially intended only to facilitate bench-marking the results of the Analytic approach presented later in the chapter against the Gaussian Quadrature approach, its practical implementation presented several opportunities for contribution to the domain.

The mathematics of the Gaussian Quadrature approach are well documented in the literature, and there is ubiquitous free and commercial software available providing 'black-box', universally-applicable solutions. To adopt such solutions, the user typically provides a description of a single function to be integrated as an input and the software provides the corresponding Gaussian Quadrature solution. However, for the computation of the Gramian matrices described in the previous section, this generic 'black-box' approach is not well-suited in terms of efficiency, as it is not tailored for the specific problem and generates unnecessary repetition of calculations.

Implementation detail on how best to employ a Gaussian Quadrature approach in computing the stiffness matrix of an Active B-Spline Surface is lacking in the literature. While some problem-specific optimisations are suggested (Schumaker, 2007), they are typically references to taking advantage of the banded nature of the Gramian matrices to avoid unnecessary function evaluations, without discussion of how best to take advantage of this. Additionally, such efficiencies are generic and can be adopted regardless of the approach taken. However, tailored efficiencies that can come into play when considering simultaneously a specific approach and its application to a specific problem seem to be neglected in the literature.

In the development of the Analytic algorithm, presented later in this chapter, several approach-specific efficiencies were identified. In order to present a fair comparison between the two algorithms it was deemed necessary to also tailor the Gaussian Quadrature algorithm to the specific problem. It also became apparent that with suitable tailoring, efficiencies identified for the Analytic algorithm could also be employed by a Gaussian Quadrature algorithm.

In essence, it is the two approaches, Gaussian Quadrature and Analytic, that are being compared. To compare like with like, it is important that the algorithms implementing the approaches adopt the same efficiencies and remove redundant or unnecessary repetition of calculations. An analytic approach is inherently tailored for a specific problem, while numeric Gaussian Quadrature is a generic approach. Its application to the specific problem of Active B-Spline surfaces if applied to each term

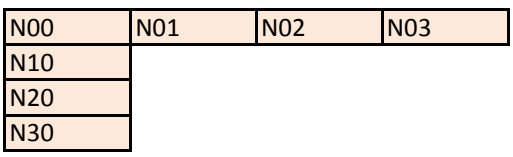


Figure 5.2: Symmetric Stencil to generate unique non-zero product terms

N00	N01	N02	N03	N04	N05	N06	N07	N08
N10	N11	N12	N13	N14	N15	N16	N17	N18
N20	N21	N22	N23	N24	N25	N26	N27	N28
N30	N31	N32	N33	N34	N35	N36	N37	N38
N40	N41	N42	N43	N44	N45	N46	N47	N48
N50	N51	N52	N53	N54	N55	N56	N57	N58
N60	N61	N62	N63	N64	N65	N66	N67	N68
N70	N71	N72	N73	N74	N75	N76	N77	N78
N80	N81	N82	N83	N84	N85	N86	N87	N88

Figure 5.3: Symmetric Stencil Traversal of Gramian Matrix. Note: Lower greyed values are not unique and therefore can be omitted from the Gramian computation

of the Gramian matrices separately leads to unnecessary repetition of computations. For this reason, in the algorithm developed in this thesis, the approach is applied in a piece-meal fashion to remove this redundancy. Both algorithms benefit in the same way from the remaining efficiencies identified.

In this section several algorithmic efficiencies are identified and discussed, and an efficient Gaussian Quadrature algorithm, tailored for the specific problem of computing the Gramian matrices associated with an Active B-Spline surface, is developed and presented. The section first provides the technical background required for the implementation of a generic Gaussian Quadrature algorithm, before breaking the algorithm up in such a way as to optimise it for the specific problem domain.

5.3.1 Gaussian Quadrature

Gaussian Quadrature is a class of numerical integration techniques based on the following theorem:

Theorem 5.1: *Let $q(x)$ be a polynomial of degree N, such that:*

$$\int_a^b w(x)x^k q(x)dx = 0 \tag{5.6}$$

where k is any integer on $[0, N-1]$, and $w(x)$ is some weighting function.
The Gaussian-quadrature computed integral is constructed as follows:

$$\int_a^b w(x)f(x)dx \approx \sum_{i=0}^{N} w_i f(x_i) \tag{5.7}$$

If $\{x_i\}$ are the N roots of $q(x)$, then there exists some set of n $\{w_i\}$ such that the approximation is exact if $f(x)$ is a polynomial of degree $< 2N$.

Much like other numerical integration techniques, the quadrature rule facilitates the approximation of a definite integral of a function through a weighted sum of values at specified points within the domain of the integration. In general, given an arbitrary function $f(x)$ and a weighting function $w(x)$, Gaussian Quadrature results in an approximation of the integral of their product. The approach will produce good

results if the function $f(x)$ can be reasonably approximated by a polynomial function. The theorem states that by carefully choosing N weights and abscissae, an exact solution can be obtained where $f(x)$ is a polynomial function of degree $< 2N$. The proof stems from the study of orthogonal polynomial systems and can be found in most numerical analysis textbooks (Stoer et al., 2002).

Gauss-Legendre

Gauss-Legendre quadrature is a special case of Gaussian Quadrature where the weighting function is simply $w(x) = 1$. This case is appropriate for dealing with polynomial data. The integration is typically defined on the interval $[-1, 1]$ such that the Gauss-Legendre quadrature rule becomes

$$\int_{-1}^{1} f(x) \approx \sum_{i=0}^{n} w_i f(x_i) \tag{5.8}$$

The N evaluation points , x_i, for a so called 'N-point' Gauss-Legendre quadrature rule are the roots of the N^{th} order Legendre polynomials, $P_n(x)$. To achieve an exact result, N must be chosen such that the degree of the integrand does not exceed $2N - 1$. The minimum number of points needed to achieve an exact result for a polynomial of degree p is given by

$$N = \left\lceil \frac{p+1}{2} \right\rceil \tag{5.9}$$

where $\lceil\ \rceil$ represents the ceiling value of the enclosed function.
The Legendre Polynomials are defined by the following recursive rule:

$$\begin{aligned} P_0(x) &= 1 \\ P_1(x) &= x \\ nP_n(x) &= (2n-1)xP_{n-1}(x) - (n-1)P_{n-2}(x) \end{aligned} \tag{5.10}$$

The roots of these polynomials are typically approximated numerically using the Newton-Raphson iteration scheme (Press et al., 2007).

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{5.11}$$

The initial seed for the i^{th} root using the Newton-Raphson method is chosen as

$$x_0 = \cos\left(\Pi \frac{i - 0.25}{n + 0.5}\right) \tag{5.12}$$

The derivative of the Legendre Polynomials, needed for the Newton-Raphson iteration, can also be expressed by a recursive relation as follows

$$P_n'(x) = \frac{n}{x^2 - 1}\Big(xP_n(x) - P_{n-1}(x)\Big) \tag{5.13}$$

Table 5.1: Gauss Quadrature Sample Abscissae

N	x_1	x_2	x_3	x_4	x_5
2	−0.5773502692	0.5773502692			
3	0.0000000000	−0.7745966692	0.7745966692		
4	−0.3399810436	0.3399810436	−0.8611363116	0.8611363116	
5	0.0000000000	−0.5384693101	0.5384693101	−0.9061798459	0.9061798459

Table 5.2: Gauss Quadrature Sample Weights

N	w_1	w_2	w_3	w_4	w_5
2	1.0000000000	1.0000000000			
3	0.8888888889	0.5555555556	0.5555555556		
4	0.6521451549	0.6521451549	0.3478548451	0.3478548451	
5	0.5688888889	0.4786286705	0.4786286705	0.2369268851	0.2369268851

The roots of the Legendre polynomials become the abscissae and act as the grid-points for the Gauss-Legendre quadrature. The corresponding weights can be calculated using

$$w_i = \frac{2}{(1 - x_i^2)\left[P_n'(x_i)\right]^2} \tag{5.14}$$

Once the N abscissae and weights over the range$[-1, 1]$ have been calculated, any integral range can then be mapped onto this interval by a linear transformation of variables

$$\int_a^b f(x)\, dx = \int_{-1}^{1} \frac{b-a}{2} f\Big(\frac{(b-a)x + (b+a)}{2}\Big)\, dx \tag{5.15}$$

The generic Gauss-Legendre quadrature rule then becomes

$$\int_a^b f(x)\, dx = \sum_{i=0}^{n} w_i \frac{b-a}{2} f\Big(\frac{(b-a)x_i + (b+a)}{2}\Big) \tag{5.16}$$

Sample values of Gauss abscissae and weights for low N are given in Table 5.1 and Table 5.2.

5.3.2 Efficient Gaussian Quadrature Algorithm

Section 5.3.1 summarises the relevant mechanics of generic Gaussian Quadrature. Further detail can be found in most numerical textbooks, e.g., (Press et al., 2007) and (Stoer et al., 2002). As discussed, this section considers the approach with respect to the specific problem of constructing the stiffness matrix of an Active B-Spline Surface. The section describes the development of an efficient algorithm for the computation of the four Gramian matrices required for the assembly of the stiffness matrix using a tailored Gaussian Quadrature technique. The efficiencies are algorithmic and stem from avoiding unnecessary repetition of calculations. To exemplify the overall approach, the technique is demonstrated for the calculation of the Gramian matrices associated with the cubic basis functions shown in Figure 5.1.

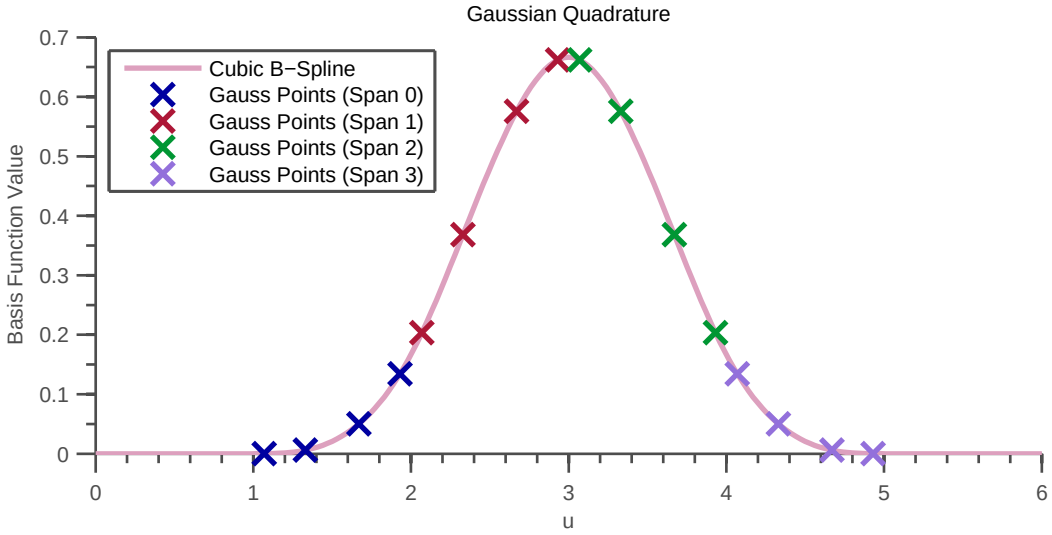


Figure 5.4: Gaussian points required for the exact integration of the product of two cubic B-Spline basis functions e.g., $\int N_{4,3}\ N_{4,3}\ du$ on the knot vector $U = \{0, 0, 0, 0, 1, 2, 3, 4, 5, 6, 6, 6, 6\}$. The degree of the product is 6 requiring a 4-point Gaussian Quadrature.

The first step of the process requires determining the number of Gaussian abscissae needed for exact computation of the integrals. As each term involves a product of two basis functions, N becomes

$$N = \left\lceil \frac{2p+1}{2} \right\rceil \tag{5.17}$$

where $\lceil\ \rceil$ represents the ceiling value of the enclosed function. Taking $\int N_{4,3}\ N_{4,3}\ du$ for instance, where the degree, p, of each basis function is 3, a 4-point Gaussian Quadrature is required to achieve accurate results. Therefore, the weights and abscissae for the 4-point Gaussian Quadrature are required.

As B-Spline basis functions are piecewise polynomial, to utilise the Gaussian Quadrature approach, the technique must be applied to each polynomial section of each product separately. Figure 5.4 shows the $p+1$ polynomial sections over $p+1$ knot spans for the problem $\int N_{4,3}\ N_{4,3}\ du$, where $N_{4,3}$ is the cubic basis function of index 4 on the knot vector $U = \{0, 0, 0, 0, 1, 2, 3, 4, 5, 6, 6, 6, 6\}$ (See Figure 5.9). Here, the Gaussian abscissae have been linearly transformed to lie between the end points of each knot span. The integral of the function over each span is the weighted summation of the function evaluations at the transformed Gaussian abscissae as outlined in Equation 5.16.

The knots of the knot-vector are the divisions of the polynomial sections, and several basis functions exist over a single knot span. This means that it is more computationally efficient to fully address each sample point within a span at each step, and update the relevant terms of the Gramian matrix, rather than address each Gramian term individually.

Over any given span, there are $p+1$ non-zero basis functions. This means that at each Gaussian abscissa on the span, $p+1$ basis functions must be evaluated, weighted, and summed into their corresponding elements in the Gramian matrix. Taking the first non-zero length knot span for the cubic case in Figure 5.1 as an example, there are four non-zero basis functions, namely, $N_{0,3}$, $N_{1,3}$, $N_{2,3}$ and $N_{3,3}$. Each of these basis functions must be evaluated at each Gaussian abscissa on the span. This process is illustrated in Figure 5.5.

Span3	N0	N1	N2	N3
w(i)	x(i)	x(i)	x(i)	x(i)
w(i+1)	x(i+1)	x(i+1)	x(i+1)	x(i+1)
w(i+2)	x(i+2)	x(i+2)	x(i+2)	x(i+2)
w(i+3)	x(i+3)	x(i+3)	x(i+3)	x(i+3)

Figure 5.5: Sub-Grid of Gaussian Samples

Every possible product combination of the basis functions at each abscissa on the span is evaluated, and weighted by the appropriate weight resulting in $N \times (p+1) \times (p+1)$ terms to be summed into the appropriate elements of the Gramian matrix. For the example case, this results in four 4×4 grids. This situation is depicted in Figure 5.6.

N00	N01	N02	N03
N10	N11	N12	N13
N20	N21	N22	N23
N30	N31	N32	N33

Figure 5.6: Assembly of Sub-Grid of Gaussian Samples

Taking advantage of the problem-specific efficiencies presented in Section 5.2, the grids of product combinations are symmetric. This means that there are $\binom{p+1}{2} + p + 1$ unique terms to be evaluated at each abscissa. The ten unique values for the example case are illustrated in Figure 5.6.

This process is repeated on a span-by-span basis until each span has been traversed and the Gramian matrix has been filled. In this way the $(p+1) \times (p+1)$ grid can be thought of as a stencil or template for each span. This idea is illustrated in Figure 5.7.

Apart from the advantage of re-using the linear transform results of each span for several Gramian matrix updates, another benefit of this approach stems from the basis function calculation. As can be seen from equation 4.2 (repeated here for convenience), the basis functions at a given parametric value depend on the span in which the value lies.

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u)$$

N00	N01	N02	N03	N04	N05	N06	N07	N08
N10	N11	N12	N13	N14	N15	N16	N17	N18
N20	N21	N22	N23	N24	N25	N26	N27	N28
N30	N31	N32	N33	N34	N35	N36	N37	N38
N40	N41	N42	N43	N44	N45	N46	N47	N48
N50	N51	N52	N53	N54	N55	N56	N57	N58
N60	N61	N62	N63	N64	N65	N66	N67	N68
N70	N71	N72	N73	N74	N75	N76	N77	N78
N80	N81	N82	N83	N84	N85	N86	N87	N88

Figure 5.7: Gramian Grid: Traversal of the sampling stencil in Figure 5.6 across the knot vector for Gramian Matrix construction

$$N_{i,0}(u) = \begin{cases} 1 & \text{if } u_i \leq u < u_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{5.18}$$

Figure 5.8 illustrates how the non-zero basis functions at any value on span i are built up from the 0^{th} degree basis function $N_{i,0}$ at that value. The terms outside the triangular structure are zero on the span. Computing the Gramian matrix in this way avoids repeated calculations of key values.

The Gramian matrices associated with the remaining three terms described by Equations 5.3–5.5, inclusive, can be computed simultaneously for maximum efficiency. While these terms are lower order, and therefore require fewer Gaussian sample points to achieve exact solutions, the re-use of key values computed for the case described for the computation of the Gramian matrix associated with Equation 5.2, makes simultaneous computation more efficient.

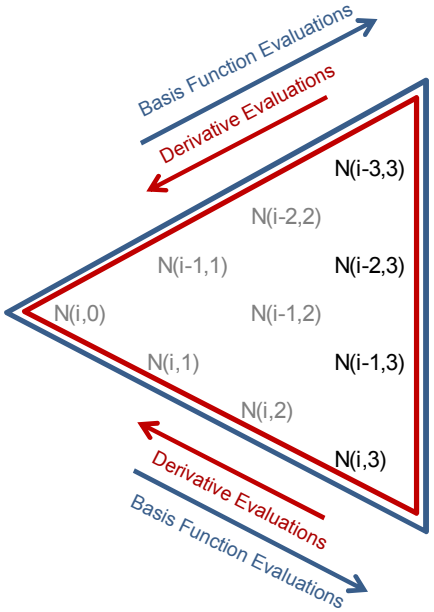


Figure 5.8: Efficient Basis Function and Derivative Calculation

The derivatives of the basis function, as needed by the remaining terms, can be calculated using repeated application of

$$N_{i,p}^{(k)}(u) = \frac{p}{u_{i+p} - u_i} N_{i,p-1}^{(k-1)}(u) - \frac{p}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}^{(k-1)}(u) \tag{5.19}$$

Equation 5.19 can be applied directly where $0 \leq k \leq p$. It is a standard formulation and can be found in most textbooks covering B-Spline curves and surfaces. For a detailed proof, see Piegl and Tiller (1997). Figure 5.8 illustrates how the application of this equation to a value on the knot span i involves the reverse traversal of the triangular structure.

The Gaussian Quadrature algorithm to compute all four Gramian matrices is summarised in Algorithm 5.1. The Gramian matrices are labeled according to their respective derivative components as follows $G00$, $G11$, $G22$, and $G10$.

Algorithm 5.1 Efficient Gaussian Quadrature Computed Stiffness Terms

Compute N
Compute N abscissae and weights
for $spanIndex = p + 1 \rightarrow endSpan - (p + 1)$
  Compute upper and lower limits of span, $spanIndex$
  for $n = 1 \rightarrow N$
    Perform linear transform of Gaussian abscissa x_n
    Compute $p + 1$ non-zero Basis Functions and required Derivatives at x_n
    for $i = 1 \rightarrow p + 1$
      for $j = i \rightarrow p + 1$
        $G00_{i+spanIndex,j+spanIndex} \mathrel{+}= w_n N_i(x_n) N_j(x_n)$
        $G11_{i+spanIndex,j+spanIndex} \mathrel{+}= w_n N_i^1(x_n) N_j^1(x_n)$
        $G22_{i+spanIndex,j+spanIndex} \mathrel{+}= w_n N_i^2(x_n) N_j^2(x_n)$
        $G10_{i+spanIndex,j+spanIndex} \mathrel{+}= w_n N_i^1(x_n) N_j(x_n)$

5.3.3 Discussion

The algorithm presented in this section is an efficient algorithm for generating the four Gramian matrices associated with the stiffness matrix of an Active B-Spline surface using the state of the art Gaussian Quadrature approach. By breaking up the traditional Gaussian Quadrature algorithm, the approach takes advantage of the re-use of linearly transformed Gaussian abscissae on a span-by-span basis. Further algorithmic efficiencies are gained by generating the B-Spline basis functions and their derivatives on a sample-by-sample basis before assembling into the final Gramian matrices. In this way, unnecessary repetition of computations, that would be incurred by applying the approach directly to each individual term in the Gramian matrices, is avoided. This is important for general Gaussian Quadrature for Active B-Spline/NURBS problems, and also to facilitate a fair comparison with the analytic algorithm developed in the next section. The algorithm makes no assumptions about the regularity of the knot vector. It is worth noting that recent research by Hughes et al. (2010) and Auricchio

et al. (2012) has proven that Gaussian Quadrature rules can be improved by taking into account a-priori knowledge of the continuity of a B-Spline/NURBS curve across each knot. In this research, the number of samples per span required for the computation of the integrals can be reduced. It is likely that this approach will replace existing Gaussian Quadrature rules as the state of the art Gaussian Quadrature approach. The algorithmic efficiencies presented in this section are equally applicable to the reduced Quadrature approach. The bench-marking implications of this research are further discussed in Section 5.4.3.

5.4 Proposed Analytic Approach

This section introduces a novel and more direct analytic solution to the problem of constructing the stiffness matrix of an Active B-Spline Surface. The section begins by developing the mathematics of the proposed Analytic Approach to solving the problem of constructing the Gramian matrix associated with each of the terms that make up the stiffness matrix. A special property of the problem is identified that can be applied for each term facilitating exact analytical solutions. Graphical illustrations of the underlying mathematical mechanics are developed and provided for each term.

A key component in evaluating the Analytic solution is the ability to determine the integral and repeated integrals of a B-Spline basis function. The literature in this area points mainly to numerical approaches and there is a lack of detail with regard to an analytic evaluation. Where the analytic approach of divided differences is discussed, it is dismissed as being highly unstable (Schumaker, 2007). Most of the key textbooks in the area omit a discussion on the integral of a B-Spline curve or basis function. While the integral of a B-Spline curve is discussed briefly in (Schumaker, 2007) and (Deboor, 1978), with the migration of B-Spline evaluation practices from traditional divided differencing techniques to those based on the more stable Cox-deBoor recursion, the notation in the literature varies greatly. For consistency, in Section 5.4.1, a formula for evaluating the integrals and repeated integrals is derived directly from the basis function and derivative formulae provided in Piegl and Tiller (1997), which appears to be the standard modern notation. Graphical representations are also provided. Efficient algorithms for computing the integrals and repeated integrals, based on the presented mathematics, are developed and presented.

An additional component required is the ability to determine the $(p+1)^{th}$ derivative of a basis function. The algorithms presented so far in this thesis and in the literature only cater for up to the p^{th} derivative. It is typically assumed that all higher derivatives are 0, which is not strictly the case. Section 5.4.1 fully addresses this gap in the literature and an efficient algorithm for the computation of the $(p+1)^{th}$ derivative is developed and presented.

Finally, an efficient algorithm is presented for the evaluation of the four Gramian matrices. In section 5.9 the computational complexity of the approach, and the performance, accuracy and stability of the algorithm are studied and discussed in detail

and bench-marked both against the Gaussian Quadrature approach, developed and presented in the previous section, and exact results obtained symbolically. The Analytic approach developed is shown to be computationally less complex and consequently up to 4.3 times faster than the Gaussian Quadrature approach for the practical cases considered, while preserving a high degree of accuracy and stability.

5.4.1 Analytic Approach

In mathematical analysis, Integration by Parts is a technique frequently used in an effort to simplify integrals involving products of functions. The relevant formula is given by

$$\int u(x)v'(x)\,dx = u(x)v(x) - \int u'(x)v(x)\,dx \tag{5.20}$$

To derive an analytic solution to the Gramian Matrix problem, Integration by Parts is applied to each element to simplify the terms involving the integral of a product of basis functions.

For dealing with the Active B-Spline surface equations, the notation $g(x)$ and $h(x)$ is adopted to replace the more traditional notation, $u(x)$ and $v(x)$, to avoid confusion with the parametric variables u and v. Thus, applying integration by parts, as per Equation 5.20, to the integral $\int N_{i,p}(u)N_{j,p}(u)\,du$,

Let $g = N_{i,p}(u)$ and $dh = N_{j,p}(u)\,du$ such that

$$\int N_{i,p}(u)N_{j,p}(u)\,du = N_{i,p}(u)\int N_{j,p}(u)\,du - \int \left(\int N_{j,p}(u)\,du\right) N^{(1)}_{i,p}(u)\,du \tag{5.21}$$

Repeating the process and applying Integration by Parts successively to each new integral term generated by the preceding Integration by Parts gives

Let $g = N^{(1)}_{i,p}(u)$ and $dh = \int N_{j,p}(u)\,du$

$$N^{(1)}_{i,p}(u)\iint N_{j,p}(u)\,du^2 - \int \left(\iint N_{j,p}(u)\,du^2\right) N^{(2)}_{i,p}(u)\,du \tag{5.22}$$

Let $g = N^{(2)}_{i,p}(u)$ and $dh = \iint N_{j,p}(u)\,du^2$

$$N^{(2)}_{i,p}(u)\iiint N_{j,p}(u)\,du^3 - \int \left(\iiint N_{j,p}(u)\,du^3\right) N^{(3)}_{i,p}(u)\,du \tag{5.23}$$

Let $g = N^{(3)}_{i,p}(u)$ and $dh = \iiint N_{j,p}(u)\,du^3$

$$N^{(3)}_{i,p}(u)\iiiint N_{j,p}(u)\,du^4 - \int \left(\iiiint N_{j,p}(u)\,du^4\right) N^{(4)}_{i,p}(u)\,du \tag{5.24}$$

The iteration must be repeated such that the final iteration results in a product of basis functions that includes the $(p+1)^{th}$ derivative. As before, we assume cubic basis functions such that the fourth derivative is required. Expanding Equation 5.2 with the

partial expressions from the repeated Integration by Parts, and applying the limits of integration for the parameter domain, e.g., $[0, 1]$, gives

$$\int_0^1 N_{i,p}(u)N_{j,p}(u)\,du = \left[N_{i,p}(u)\int_0^1 N_{j,p}(u)\,du\right]_0^1 - \left[N_{i,p}^{(1)}(u)\int_0^1\int_0^1 N_{j,p}(u)\,du^2\right]_0^1 + \left[N_{i,p}^{(2)}(u)\int_0^1\int_0^1\int_0^1 N_{j,p}(u)\,du^3\right]_0^1 - \left[N_{i,p}^{(3)}(u)\int_0^1\int_0^1\int_0^1\int_0^1 N_{j,p}(u)\,du^4\right]_0^1 + \int_0^1\left(\int_0^1\int_0^1\int_0^1\int_0^1 N_{j,p}(u)\,du^4\right)N_{i,p}^{(4)}(u)\,du \quad (5.25)$$

Similarly, Equations 5.3 through to 5.5 become

$$\int_0^1 N_{i,p}^{(1)}(u)N_{j,p}^{(1)}(u)\,du = \left[N_{i,p}^{(1)}(u)N_{j,p}(u)\right]_0^1 - \left[N_{i,p}^{(2)}(u)\int_0^1 N_{j,p}(u)\,du\right]_0^1 + \left[N_{i,p}^{(3)}(u)\int_0^1\int_0^1 N_{j,p}(u)\,du^2\right]_0^1 - \int_0^1\left(\int_0^1\int_0^1 N_{j,p}(u)\,du^2\right)N_{i,p}^{(4)}(u)\,du \quad (5.26)$$

$$\int_0^1 N_{i,p}^{(2)}(u)N_{j,p}^{(2)}(u)\,du = \left[N_{i,p}^{(2)}(u)N_{j,p}^{(1)}(u)\right]_0^1 - \left[N_{i,p}^{(3)}(u)N_{j,p}(u)\right]_0^1 + \int_0^1 N_{j,p}(u)N_{i,p}^{(4)}(u)\,du \quad (5.27)$$

$$\int_0^1 N_{i,p}^{(1)}(u)N_{j,p}(u)\,du = \left[N_{i,p}^{(1)}(u)\int N_{j,p}(u)\,du\right]_0^1 - \left[N_{i,p}^{(2)}(u)\int_0^1\int_0^1 N_{j,p}(u)\,du^2\right]_0^1 + \left[N_{i,p}^{(3)}(u)\int_0^1\int_0^1\int_0^1 N_{j,p}(u)\,du^3\right]_0^1 - \int_0^1\left(\int_0^1\int_0^1\int_0^1 N_{j,p}(u)\,du^3\right)N_{i,p}^{(4)}(u)\,du \quad (5.28)$$

Graphical representations of the process for each term are provided in Section 5.4.1.

At first it may not appear as though Integration by Parts has simplified the problem. However, the majority of the constant terms in the equations are 0, with the exception of those involving the basis functions at either end of the knot vector. Additionally, the $(p+1)^{th}$ derivative of the p^{th} degree curve presents an interesting special case that results in a simplification of the problem that will be dealt with in the next section. In order to evaluate these expressions analytically, the special case of the $(p+1)^{th}$ derivative must be dealt with. Additionally, the derivatives and integrals of the basis functions must be calculated.

Special Case

Basis functions are generally treated as only existing over a finite region of the knot vector, i.e., $N_{i,p}$ exists only where $u_i \leq u < u_{i+p+1}$. The piecewise nature of the equations means that separate equations define the basis functions over each individual knot span $u_i \leq u < u_{i+1}$. In the literature, the equations provided typically deal with defining values for basis functions within the limits of each span, e.g., Piegl and Tiller (1997) state that k in equation 5.19 should not exceed p as all higher derivatives are zero. If the basis functions are viewed as existing over the full range of the knot vector with zero value outside the range $u_i \leq u < u_{i+p+1}$, an interesting case develops at the junction of each span, i.e., the knots.

Figure 5.9 shows the basis functions defined for the example cubic basis functions defined over the knot vector $U = \{0, 0, 0, 0, 1, 2, 3, 4, 5, 6, 6, 6, 6\}$. Repeated differentiation of the basis functions leads to piecewise quadratic curves, piecewise linear curves, and as can be seen, if the full range of u, spanning the entire knot vector, is considered, the p^{th} derivative of each basis function becomes a sum of weighted and shifted Heaviside/Step functions.

The Heaviside function is the basic building block for functions with a discontinuity (Jordan and Smith, 2002) and is defined as

$$H(t) = \begin{cases} 0 & \text{when } t < 0, \\ 1 & \text{when } t \geq 0 \end{cases} \tag{5.29}$$

The Heaviside function can be shifted so that it can represent a discontinuity at the critical point.

$$H(t - t_0) = \begin{cases} 0 & \text{when } t < t_0, \\ 1 & \text{when } t \geq t_0 \end{cases} \tag{5.30}$$

The Heaviside function is shown in Figure 5.10.

The Analytic solution to the energy minimisation problem described in the previous section requires the evaluation of the $(p+1)^{th}$ derivative. As described in (Piegl and Tiller, 1997), the first p derivatives can be calculated using Equation 5.19. The $(p+1)^{th}$ derivative amounts to finding the derivative of each sum of weighted and shifted Heaviside/Step functions.

The Dirac delta function, $\delta(t)$, shown in Figure 5.11, can be regarded formally as the derivative of the unit function $H(t)$ (Jordan and Smith, 2002). The relationship between the Heaviside function and Dirac delta function is given in Equations 5.31 and 5.32.

$$\frac{dH(t)}{dt} = \delta(t) \tag{5.31}$$

$$H(t) = \int_{-\infty}^{t} \delta(t)dt \tag{5.32}$$

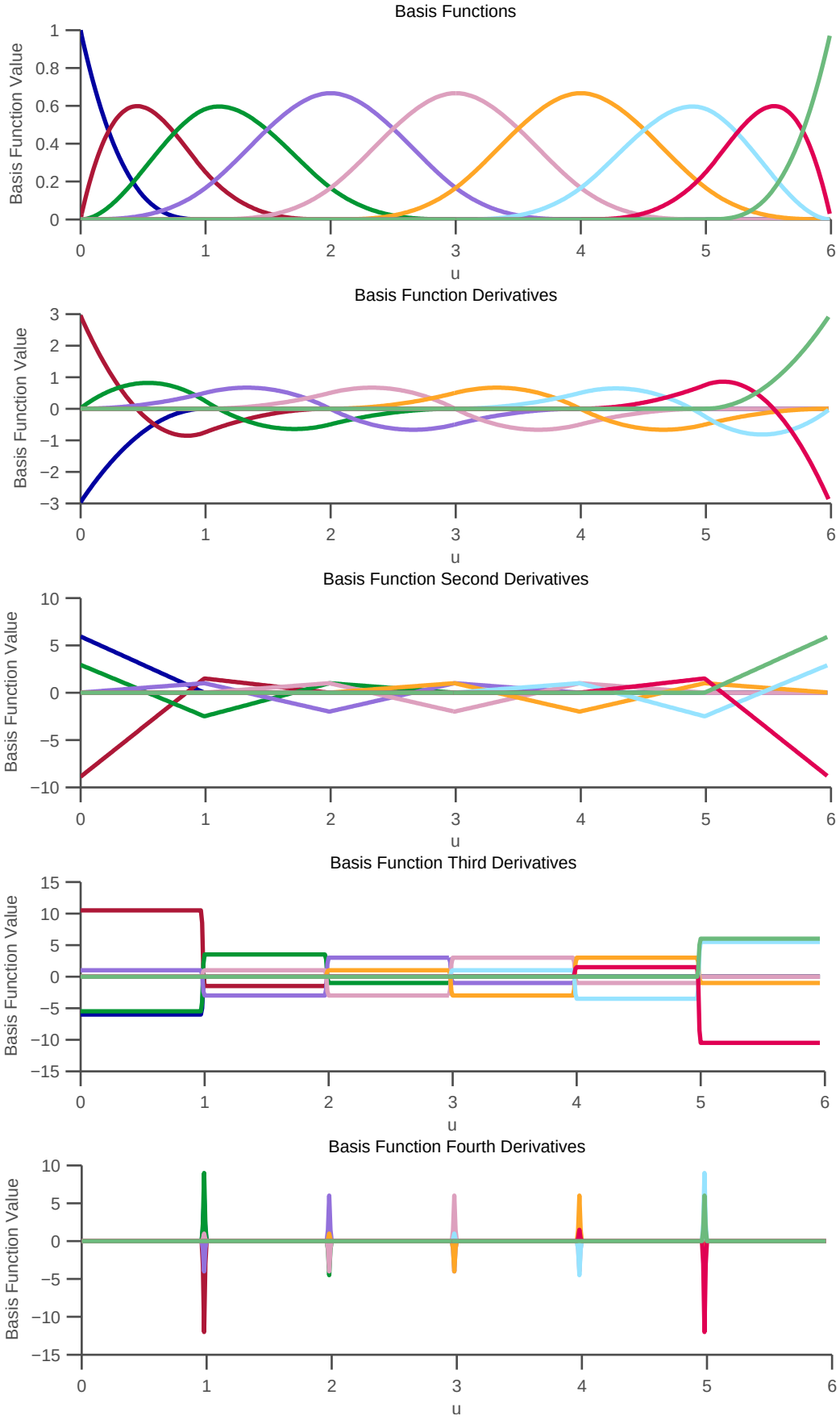


Figure 5.9: Cubic basis functions and repeated derivatives on the knot vector $U = \{0, 0, 0, 0, 1, 2, 3, 4, 5, 6, 6, 6, 6\}$

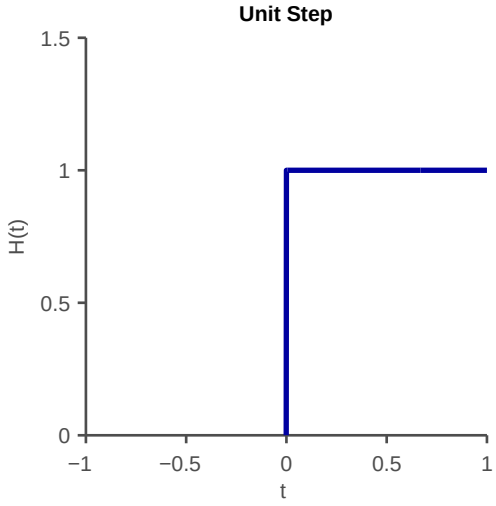


Figure 5.10: Heaviside Function

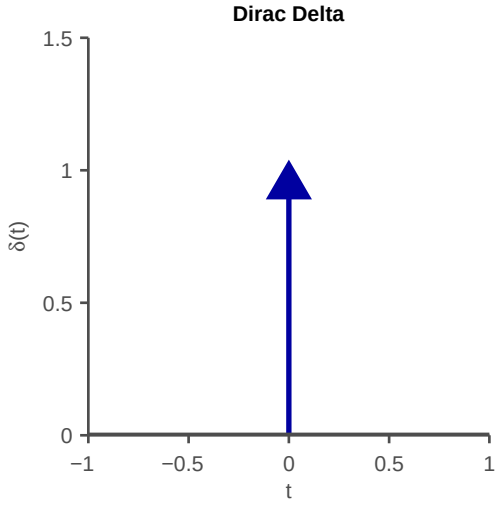


Figure 5.11: Dirac Delta Function

The derivative of a sum of weighted and shifted Heaviside functions equates to a sum of weighted and shifted Dirac delta functions. This situation is depicted for the example case in Figure 5.9.

The 'sifting' property of the Dirac delta, described by Equation 5.33, facilitates the evaluation of the special case referred to in the previous section.

$$\int_a^b f(t)\delta(t-t_0)dt = \begin{cases} f(t) & \text{for } a \leq t_0 < b \\ 0 & \text{otherwise} \end{cases} \tag{5.33}$$

Using the sifting property, the equation

$$\int_0^1 \left(\int_0^1 \int_0^1 \int_0^1 \int_0^1 N_{j,p}(u)\, du^4 \right) N_{i,p}^{(4)}(u)\, du \tag{5.34}$$

results in a sum of weighted and shifted Dirac delta functions, weighted by the magnitudes of the associated Heaviside changes, that sample the values of $\int_0^1 \int_0^1 \int_0^1 \int_0^1 N_{j,p}(u)\, du^4$ at the sites of the relevant Dirac deltas, i.e., the knots.

To calculate the magnitude of the Dirac delta weights, the p^{th} derivatives at either side of the site of the Dirac delta can be differenced. The direction of the Dirac delta is given by the sign of the weight.

Using these mathematical tools, an Analytic solution to the stiffness matrix problem can be found.

Graphical Representation

This section provides graphical illustrations of the mathematics presented in Section 5.4.

The premise of Integration by Parts is to express $\int u\ dv$ in terms of the related functions u, v, and an ideally simpler integral, $\int v\ du$ that can be evaluated [5]. Figure 5.12 shows a graphical interpretation of the Analytic approach with $u(x)$ plotted against $v(x)$. The integral terms $\int u\ dv$ and $\int v\ du$ are also marked on the diagram. As illustrated, if the functions u, v, and $\int v\ du$ can all be evaluated, then so too can $\int u\ dv$.

In the case of the energy minimisation problem, the graphical representation of the repeated Integration by Parts for each of the four terms given by Equations 5.2–5.5, inclusive, is shown by Figures 5.13 - 5.20, inclusive. For each case, the product of the relevant terms is shown, together with the required integral curve. The subsequent diagrams show the corresponding $u(x)$ vs $v(x)$ relationships relating to each repeated Integration by Parts. As can be seen, each Integration by Parts results in an alternative integral curve description, and systematically reduces the integral term to the special case depicted in Figure 5.21. The Dirac delta functions are scaled here to reflect the weight introduced as a result of the magnitude of the related Heaviside functions produced by repeated differentiation of the basis function. Figure 5.22 shows the basis function and its p repeated derivatives that bring about the Heaviside/Step functions.

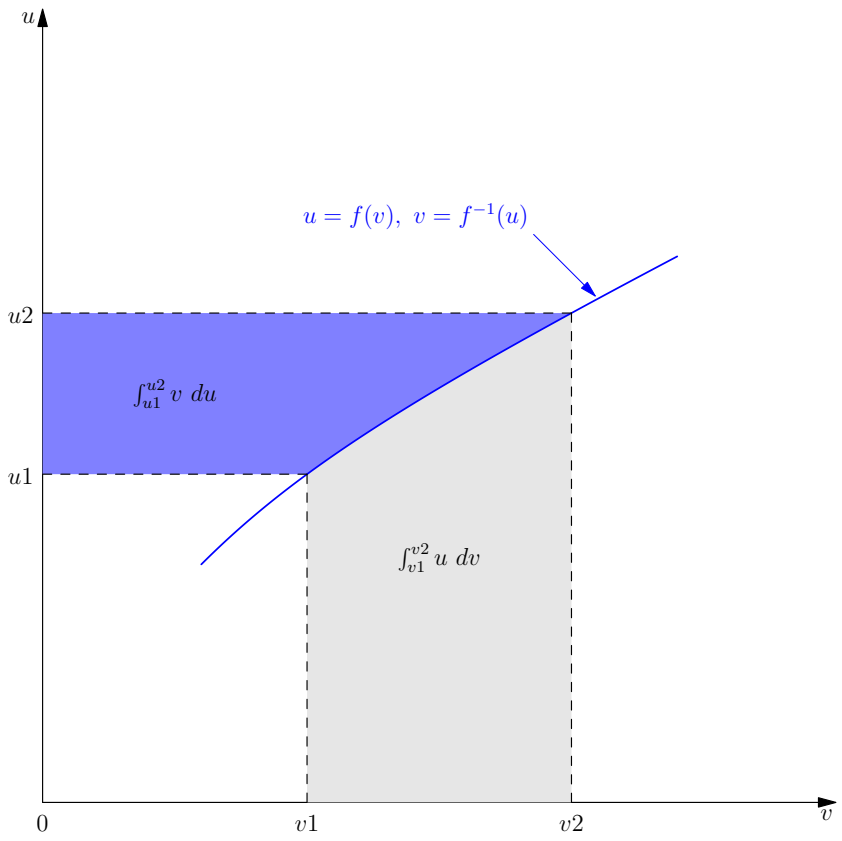


Figure 5.12: Graphical Representation of Integration by Parts $\int u\ dv = uv - \int v\ du$

[5] The notation here reverts to standard Integration by Parts notation, $\int u(x)v'(x)\ dx = u(x)v(x) - \int u'(x)v(x)\ dx$, not to be confused with the u and v parameters of the B-Spline surface

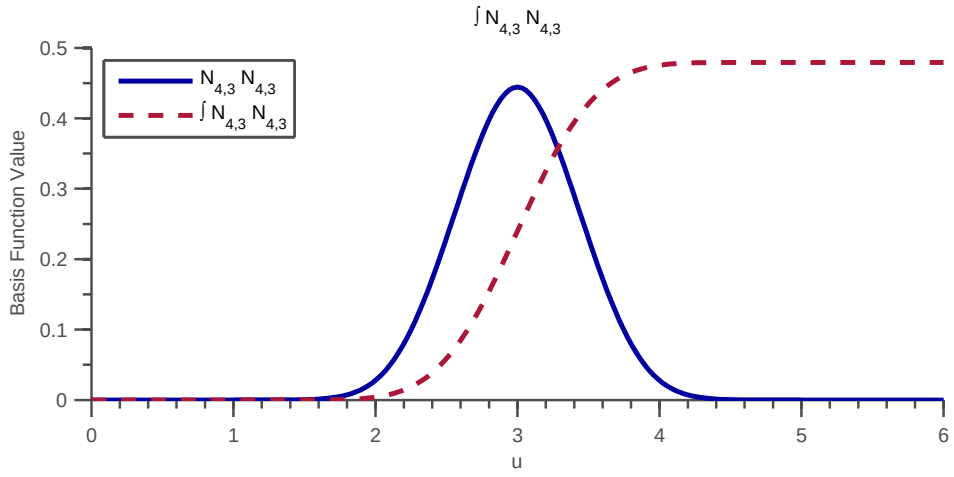


Figure 5.13: Basis Function Product and Product Integral for $\int N_{4,3}N_{4,3}$ on $U = \{0, 0, 0, 0, 1, 2, 3, 4, 5, 6, 6, 6, 6\}$

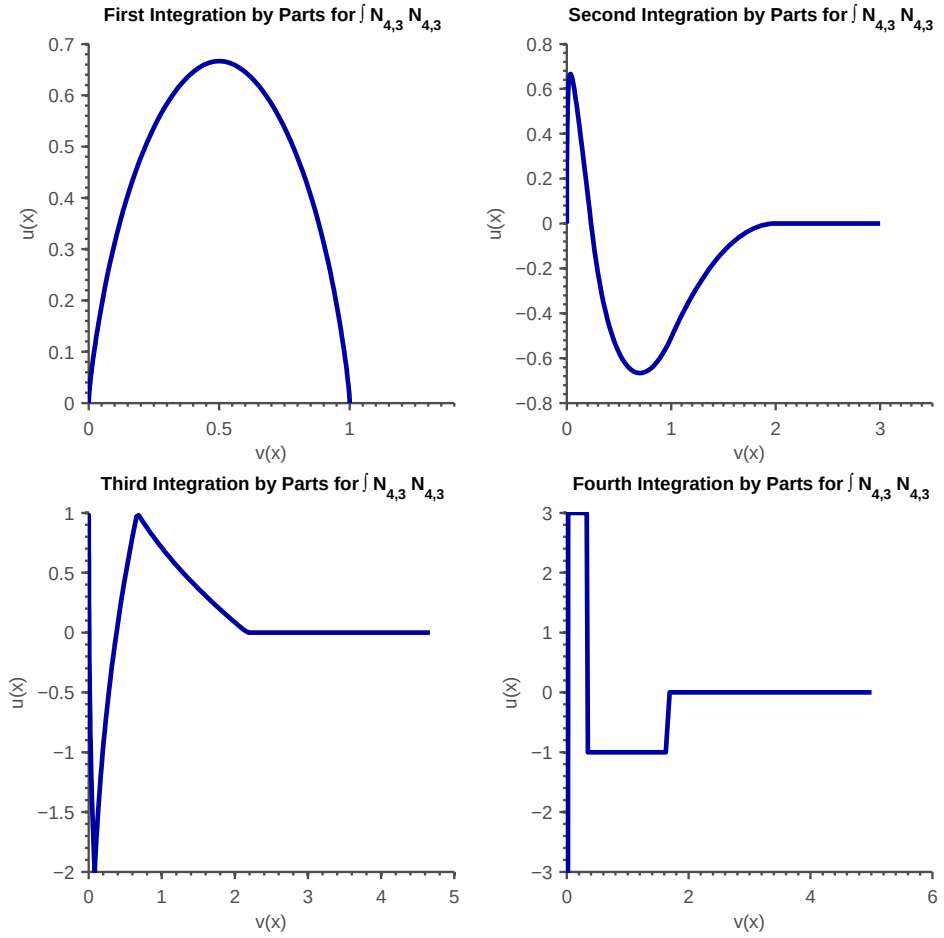


Figure 5.14: Graphical Illustration of Repeated Integration by Parts for $\int N_{4,3}N_{4,3}$, as applied to each new integral term introduced by the preceding application

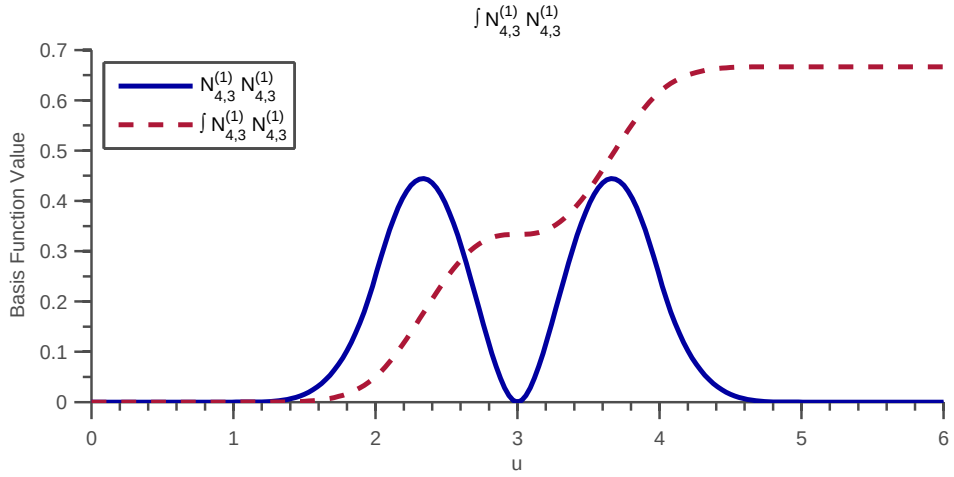


Figure 5.15: Basis Function Product and Product Integral for $\int N_{4,3}^{(1)} N_{4,3}^{(1)}$ on $U = \{0, 0, 0, 0, 1, 2, 3, 4, 5, 6, 6, 6, 6\}$

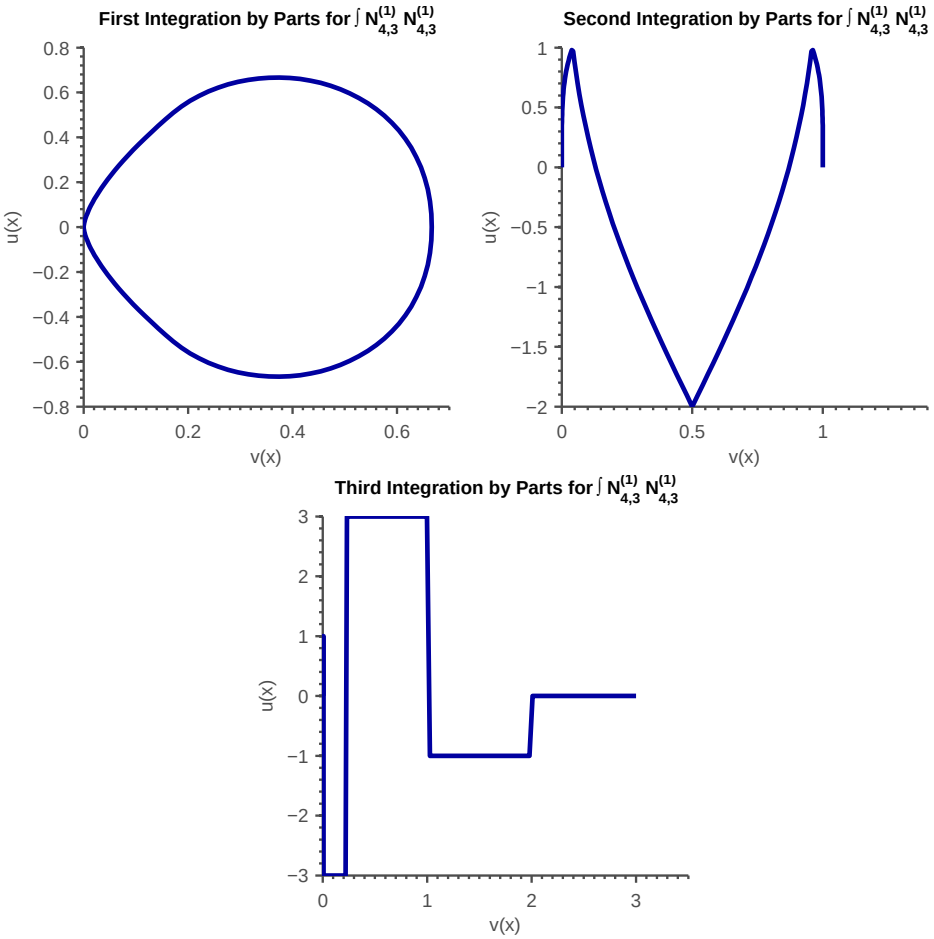


Figure 5.16: Graphical Illustration of Repeated Integration by Parts for $\int N_{4,3}^{(1)} N_{4,3}^{(1)}$, as applied to each new integral term introduced by the preceding application

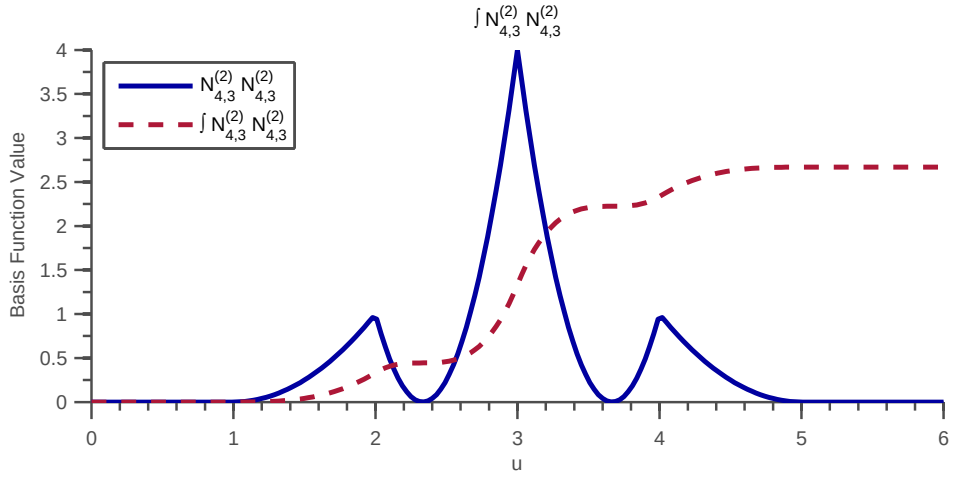


Figure 5.17: Basis Function Product and Product Integral for $\int N_{4,3}^{(2)} N_{4,3}^{(2)}$ on $U = \{0, 0, 0, 0, 1, 2, 3, 4, 5, 6, 6, 6, 6\}$

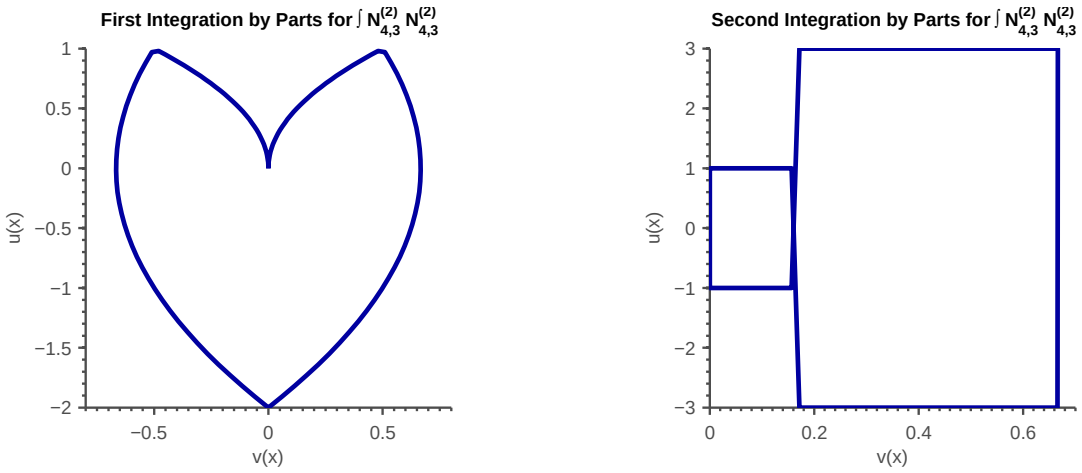


Figure 5.18: Graphical Illustration of Repeated Integration by Parts for $\int N_{4,3}^{(2)} N_{4,3}^{(2)}$, as applied to each new integral term introduced by the preceding application

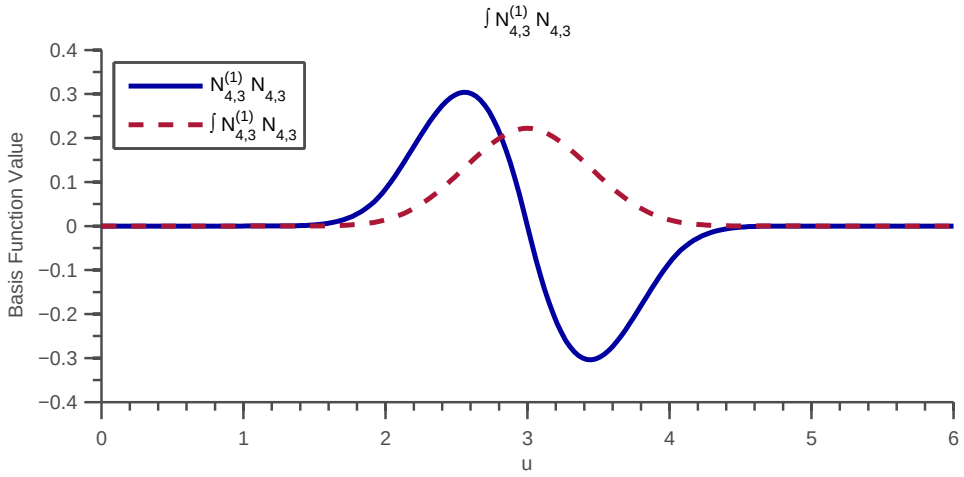


Figure 5.19: Basis Function Product and Product Integral for $\int N_{4,3}^{(1)} N_{4,3}^{(0)}$ on $U = \{0, 0, 0, 0, 1, 2, 3, 4, 5, 6, 6, 6, 6\}$

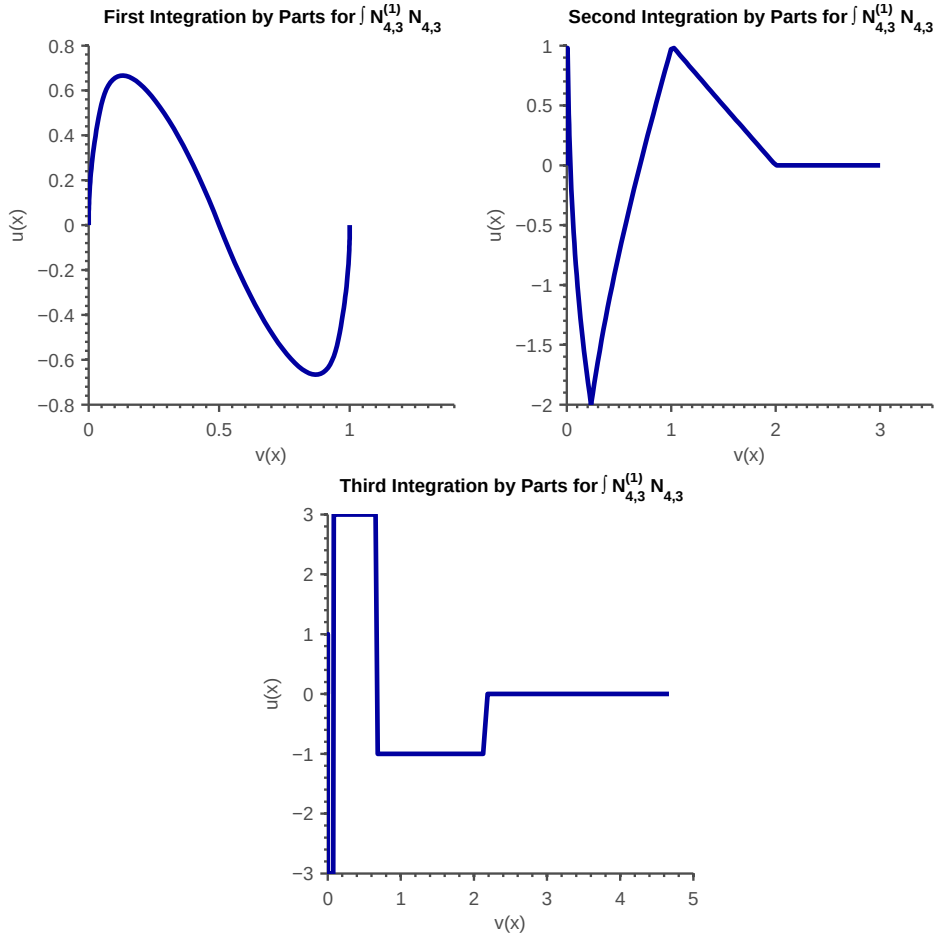


Figure 5.20: Graphical Illustration of Repeated Integration by Parts for $\int N_{4,3}^{(1)} N_{4,3}^{(0)}$, as applied to each new integral term introduced by the preceding application

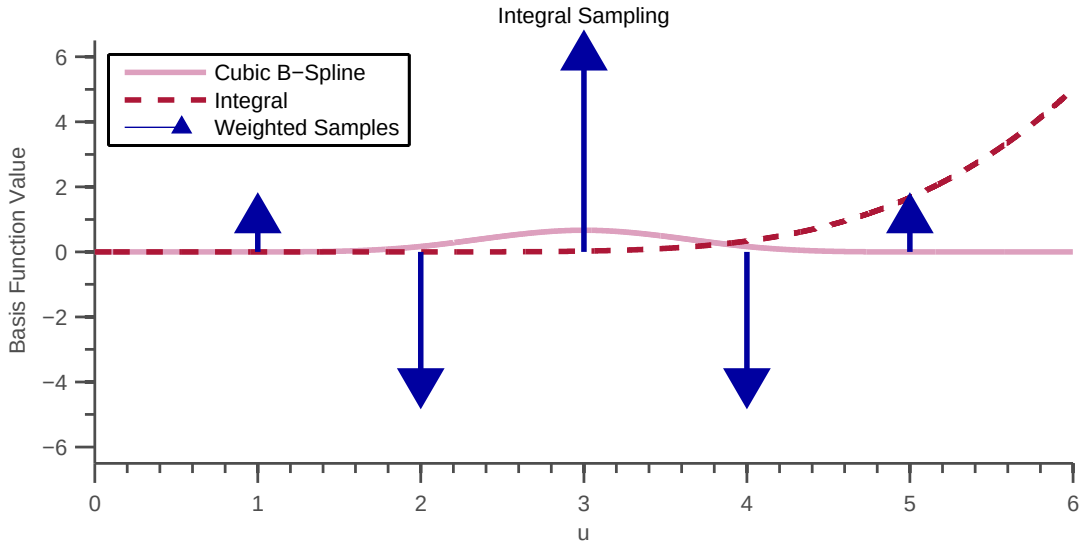


Figure 5.21: Special case: Weighted Sampling of Integral Values of the Basis Function $N_{4,3}$ on $U = \{0, 0, 0, 0, 1, 2, 3, 4, 5, 6, 6, 6, 6\}$

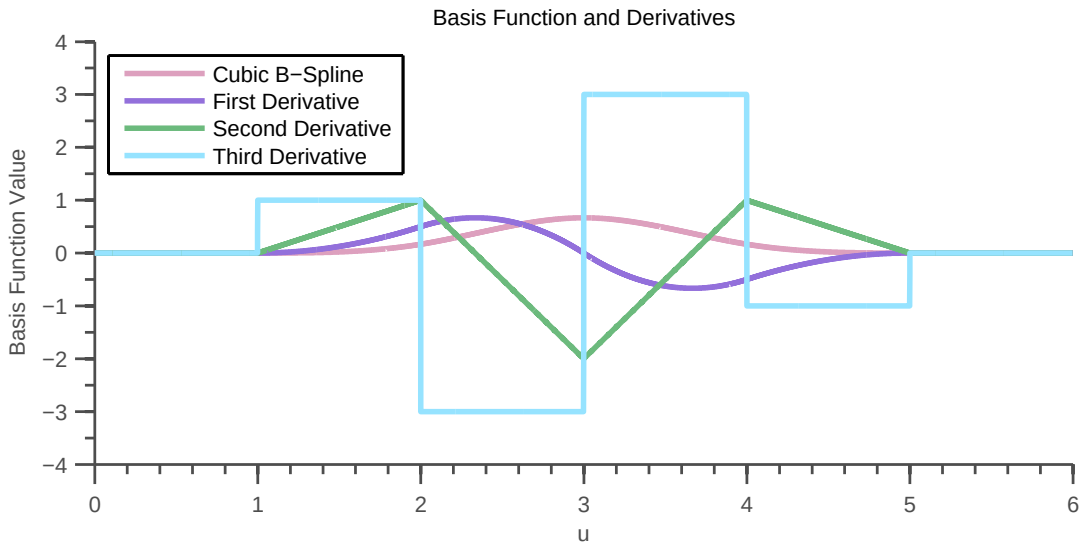


Figure 5.22: Basis Function $N_{4,3}$ and its derivatives on $U = \{0, 0, 0, 0, 1, 2, 3, 4, 5, 6, 6, 6, 6\}$

Integral of a B-Spline Basis Function

As introduced at the start of section 5.4, treatment of the integral of a B-Spline basis function does not appear to be standard in the literature (Piegl and Tiller, 1997, Farin, 2002, Cohen et al., 2001, Bartels et al., 1987). Integrals are briefly discussed by Deboor (1978) and by Schumaker (2007) and Kazinnik and Elber (1997). In this section an analytic formula for evaluating the integrals and repeated integrals is derived directly from the basis function and derivative formulae provided in (Piegl and Tiller, 1997) (A similar derivation can be found in Kazinnik and Elber (1997)). This results in the i^{th} basis function equating to a weighted sum of higher degree basis functions from i to m, the final knot, evaluated over the same knot vector. Any additional knots necessary to support the higher order functions can be achieved by extending the $p+1$ multiple knots at the end of the knot vector. Graphical illustrations of the process and its results are also provided in Figures 5.23–5.25.

Theorem 5.2:

$$\int N_{i,p} = \frac{u_{i+p+1} - u_i}{p+1} \sum_{j=i}^{m} N_{j,p+1}(u) \tag{5.35}$$

Proof. Starting with the recursive solution for finding the derivative of a B-Spline basis function

$$N_{i,p}^{(1)}(u) = \frac{p}{u_{i+p} - u_i} N_{i,p-1}(u) - \frac{p}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u)$$

Rearranging and integrating gives

$$\int N_{i,p-1}(u)\, du = \frac{u_{i+p} - u_i}{p} \left(\int N_{i,p}^{(1)}(u)\, du + \frac{p}{u_{i+p+1} - u_{i+1}} \int N_{i+1,p-1}(u)\, du \right)$$

Repeating this process for remaining integral term yields

$$\int N_{i+1,p-1}(u)\, du = \frac{u_{i+p+1} - u_{i+1}}{p} \left(\int N_{i+1,p}^{(1)}(u)\, du + \frac{p}{u_{i+p+2} - u_{i+2}} \int N_{i+2,p-1}(u)\, du \right)$$

Substituting for second term

$$\int N_{i,p-1}(u)\, du = \frac{u_{i+p} - u_i}{p} \left(\int N_{i,p}^{(1)}(u)\, du + \frac{p}{u_{i+p+1} - u_{i+1}} \right.$$
$$\left. \left(\frac{u_{i+p+1} - u_{i+1}}{p} \left(\int N_{i+1,p}^{(1)}(u)\, du + \frac{p}{u_{i+p+2} - u_{i+2}} \int N_{i+2,p-1}(u)\, du \right) \right) \right)$$

As can be seen, the additional constant multipliers cancel each other out. Repeating this process, all additional constant multipliers cancel out. Rearranging the indexing completes the proof.

$$\int N_{i,p}(u)\, du = \frac{u_{i+p+1} - u_i}{p+1} \left(N_{i,p+1}(u) + N_{i+1,p+1}(u) + \cdots + N_{m,p+1}(u) \right)$$

□

Figure 5.23 depicts the calculation of the integral of basis function $N_{4,3}$ on knot vector $U = \{0, 0, 0, 0, 1, 2, 3, 4, 5, 6, 6, 6, 6\}$. As can be seen, the summation of the higher order basis functions on the same knot vector equates to the integral of the curve. Figure 5.24 shows the integrals for all of the cubic basis functions on the knot vector $U = \{0, 0, 0, 0, 1, 2, 3, 4, 5, 6, 6, 6, 6\}$.

Typically the area under a basis function is 1 due to the partition of unity property of the basis functions. However, it can be observed from Figure 5.24 that this is not the case for those basis functions at either end of the knot vector because of the scaling associated with the range spanned. The basis functions at either end of the knot vector will still sum to 1. Repeated integrals can be calculated by repeated application of the integral formula as seen in Figure 5.25.

Figure 5.25 illustrates the repeated integration of the basis function. As shown, for each integration, the value will be 0 on its incident knot and piecewise polynomial over the supporting knot spans. When the basis function reaches its terminating knot, it will have reached a smooth polynomial condition (a non-piecewise function of u). While the formula presented in this section is valid across all the knots, it is only necessary for the piecewise descriptions of those spans supporting the basis function. If evaluations are needed at subsequent knot values, it may be more efficient to evaluate the continuous function in that region. For completeness, details are provided in Algorithm 5.2.

5.4.2 Efficient Analytic Algorithm

This section builds on the mathematical framework presented so far in Section 5.4, and develops an efficient algorithm for the computation of the four Gramian matrices required for the assembly of the stiffness matrix. The example case employed in Section 5.3.2 is reconsidered.

The main difficulty in computing the Gramian Matrices analytically lies in the evaluation of the weighted sums of integrals. As outlined in Section 5.4.1, this in turn amounts to an evaluation of weighted sum of higher order basis function evaluations.

In order to evaluate all of the necessary higher order basis functions, the knot vector must first be extended to provide support for the extended basis functions. This can be achieved by simply appending the necessary number of extra repeated knots to the end of the knot vector. For the computation of Equations 5.25 – 5.28, inclusive, the $(p+1)^{th}$ integral is the highest order of integral required. The knot vector is therefore extended by $p+1$ knots to $U = \{0, 0, 0, 0, 1, 2, 3, 4, 5, 6, 6, 6, 6, 6, 6\}$.

As is the case for Gaussian Quadrature in Section 5.3.2, due to the re-use of values possible, when all computations are evaluated on a single knot span, it is more efficient to address the Gramian matrix generation sample-by-sample on each span. The equivalent triangular structure for the Analytic solution is depicted in Figure 5.26. The columns shown in bold indicate the higher order basis functions that must be evaluated at each knot or sample point.

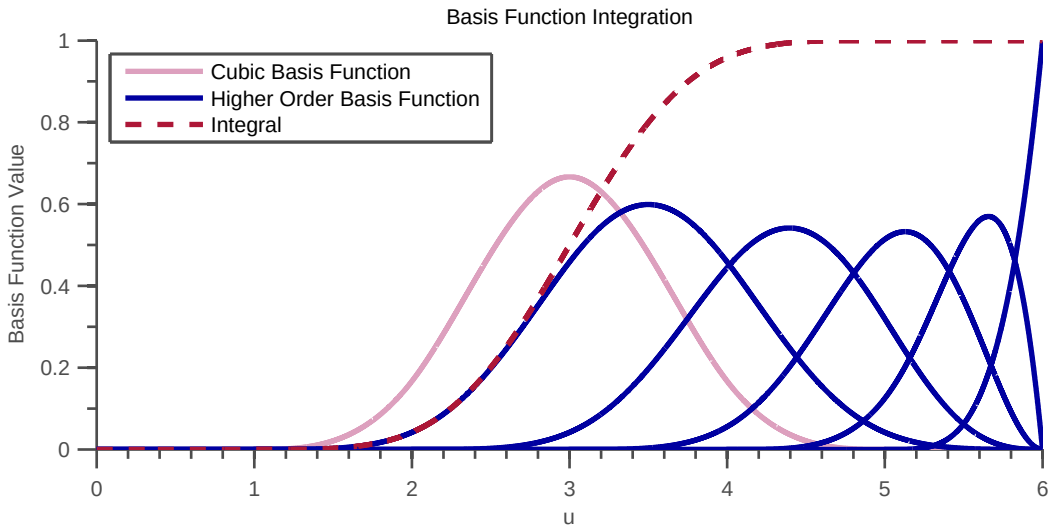


Figure 5.23: Integral of the Basis Function $N_{4,3}$ on $U = \{0, 0, 0, 0, 1, 2, 3, 4, 5, 6, 6, 6, 6\}$

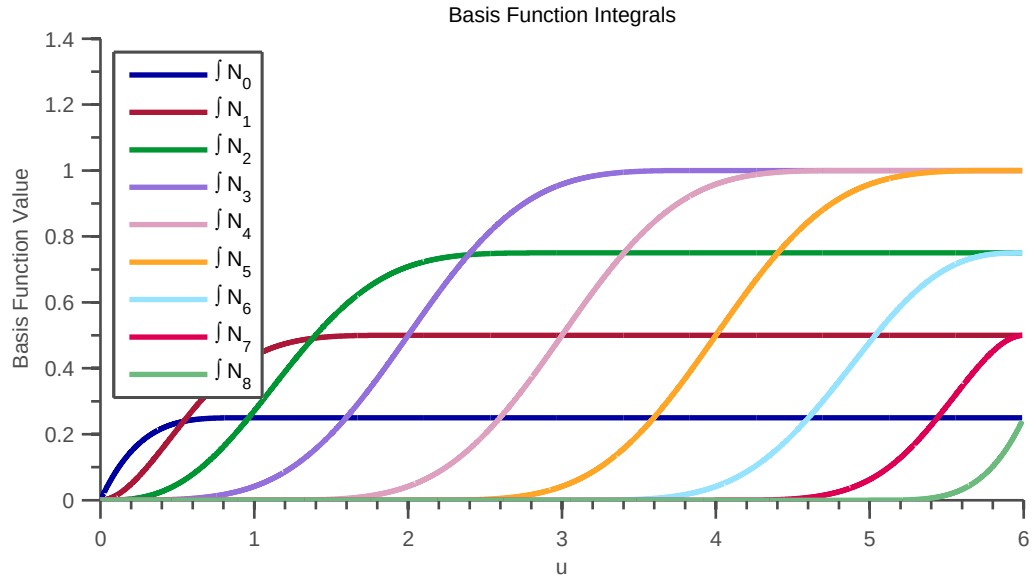


Figure 5.24: Integrals of the Basis Functions on $U = \{0, 0, 0, 0, 1, 2, 3, 4, 5, 6, 6, 6, 6\}$

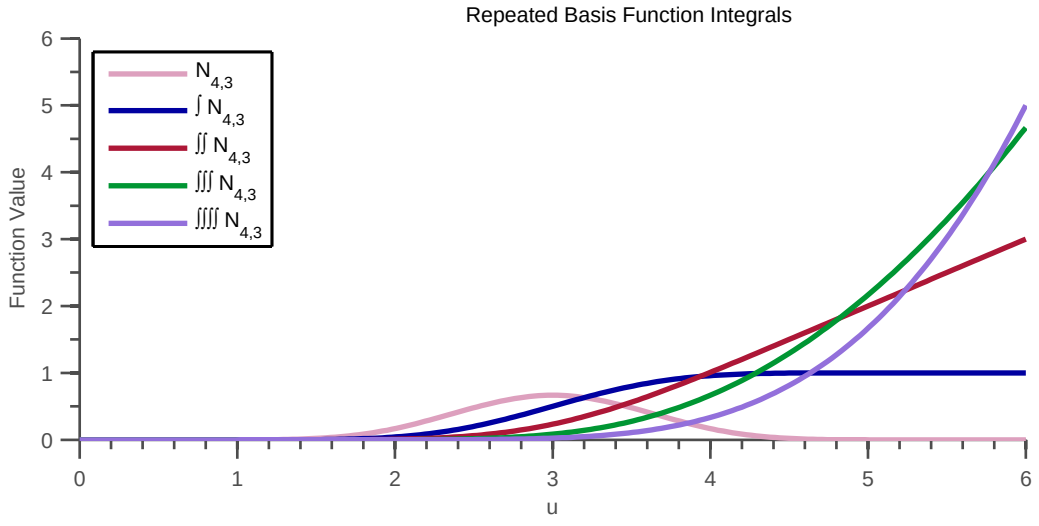


Figure 5.25: Repeated Integrals of the Basis Function $N_{4,3}$ on $U = \{0, 0, 0, 0, 1, 2, 3, 4, 5, 6, 6, 6, 6\}$

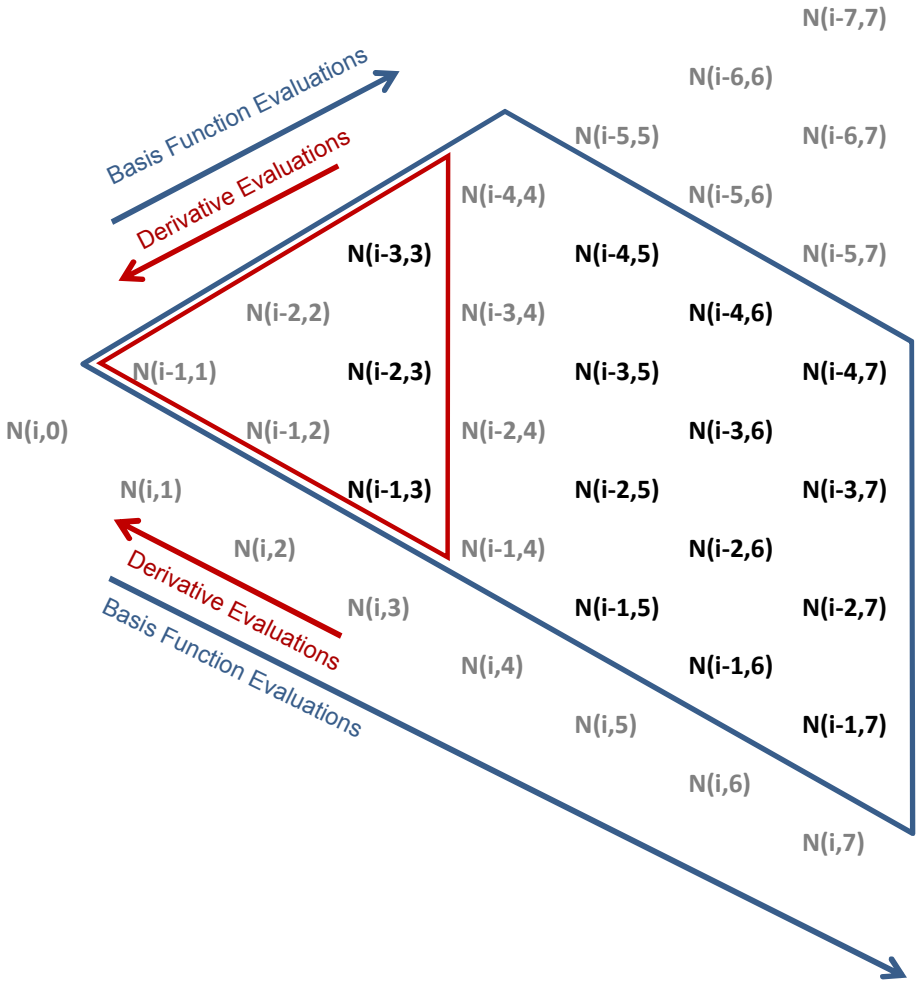


Figure 5.26: Higher Order Basis Function and Derivative Evaluations

Figure 5.27 depicts the cubic basis functions and the related higher order basis functions of degree 4 distributed over the same knot vector. It is clear from the diagram, that over each span there are $p+1$ non-zero basis functions, and that each basis function spans only $p+1$ knots.

The integral of the basis function will be 0 on its incident knot and will have reached a non-piecewise polynomial condition that is simply a function of u when the basis function reaches its terminating knot. Due to the nature of the products of basis functions being integrated, this function of u is not required for the evaluation of the Gramian matrices. However, for completeness, an efficient algorithm for computing the relevant function is developed and included in Algorithm 5.2.

At the knot spans of interest, where the integrals of all of the basis functions will have non-zero values, there is a maximum of $p+1$ non-zero higher order basis functions that can contribute to the integral samples of the $p+1$ cubic basis functions that are non-zero on the span. A sample at the terminating knot of a basis function will include $p+1$ higher order basis function contributions, while the knots internal to a basis function will require 1, 2, and up to p contributions. The number of contributions is

Algorithm 5.2 Integral of a basis function at u (for u beyond $basisEndKnot$)

function symbolicBasisFunctionIntegral
input : $u,\ integralGrade$
output : $integral$
$uproducts_0 = 1$
$uproducts_1 = u - basisEndKnot$
for $gradeIndex = 2 \rightarrow integralGrade - 1$
    $uproducts_{gradeIndex} = uproducts_{gradeIndex-1} * uproducts_1/(gradeIndex)$
for $gradeIndex = 1 \rightarrow integralGrade$
    $integral += basisEndIntegral_{gradeIndex-1} * uproducts_{integralGrade-gradeIndex}$

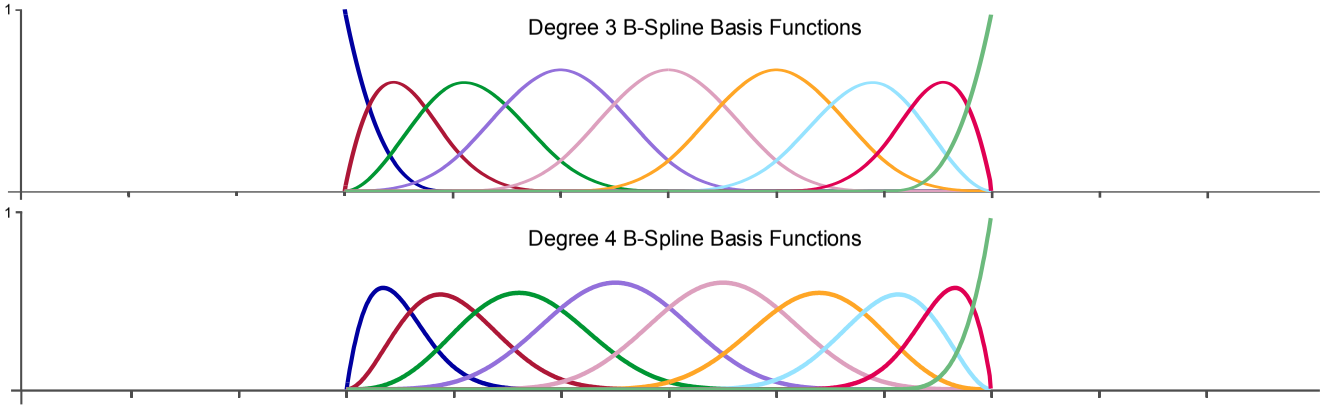
independent of the degree of the contributing higher order basis functions, and thus the conditions do not change with the grade of integral being calculated.

The trapezoidal bounding-box of Figure 5.26 illustrates the independence of the number of contributions. While there are more than $p + 1$ non-zero higher order basis functions on span i, only a maximum of $p + 1$ terms contribute to a given basis function integral evaluation. As the values outside the trapezoid are not needed for either the forward traversal evaluation of the basis functions or the reverse traversal evaluation of the derivatives, only the values inside the trapezoid need be calculated. Note that the extra repeated knots required for the higher order basis functions are omitted as they are simply an extension of the last $p + 1$ knots and no basis function values over the extra zero length knot spans are required.

For computing the necessary integral samples on a given knot span, it is computationally efficient to compute the single basis function and iteratively add one basis function at a time until all necessary sums for a given span have been computed. Each of the highlighted columns in Figure 5.26, and the terms therein, contribute to a summation.

Once the basis function summations have been computed, the weights must be evaluated. The generation of the weights produces an 'n-ary' tree. An example of this tree for the case of computing the necessary integrals of a single cubic basis function is depicted in Figure 5.28. Each level of the tree represents a level of integration while each branch represents the necessary summation of basis functions that is to be weighted and summed to produce the integral. Figure 5.29 compactly depicts the set of trees (one exists for each grade of integral) required for computing the individual integrals and includes the relevant summation terms.

The weight-tree and thus the integrals can be generated recursively. Algorithm 5.3 outlines a recursive algorithm that generates the necessary weights and summations to return the integral of a specified basis function at a given parameter value. All of the integrals can be calculated simultaneously by the algorithm by including the appropriate summation at the appropriate level.



Knots	0	0	0	0	1	2	3	4	5	6	6	6	6
KnotIndex	0	1	2	3	4	5	6	7	8	9	10	11	12
N(0,3)	0	0	0	0	N(0,4) N(1,4) N(2,4) N(3,4) 0	function of u	function of u	function of u	function of u	function of u			
N(1,3)	0	0	0	0	N(1,4) N(2,4) N(3,4) 0	N(1,4) N(2,4) N(3,4) N(4,4) 0	function of u	function of u	function of u	function of u			
N(2,3)	0	0	0	0	N(2,4) N(3,4) 0	N(2,4) N(3,4) N(4,4) 0	N(2,4) N(3,4) N(4,4) N(5,4) 0	function of u	function of u	function of u			
N(3,3)	0	0	0	0	N(3,4) 0	N(3,4) N(4,4) 0	N(3,4) N(4,4) N(5,4) 0	N(3,4) N(4,4) N(5,4) N(6,4) 0	function of u	function of u			
N(4,3)	0	0	0	0	0	N(4,4) 0	N(4,4) N(5,4) 0	N(4,4) N(5,4) N(6,4) 0	N(4,4) N(5,4) N(6,4) N(7,4) 0	function of u			
N(5,3)	0	0	0	0	0	0	N(5,4) 0	N(5,4) N(6,4) 0	N(5,4) N(6,4) N(7,4) 0	0 0 0 1			
N(6,3)	0	0	0	0	0	0	0	N(6,4) 0	N(6,4) N(7,4) 0	0 0 1			
N(7,3)	0	0	0	0	0	0	0	0	N(7,4) 0	0 1			
N(8,3)									0	1			

Figure 5.27: Analytic Integral Calculations over Knot Vector $U = \{0, 0, 0, 0, 1, 2, 3, 4, 5, 6, 6, 6, 6\}$

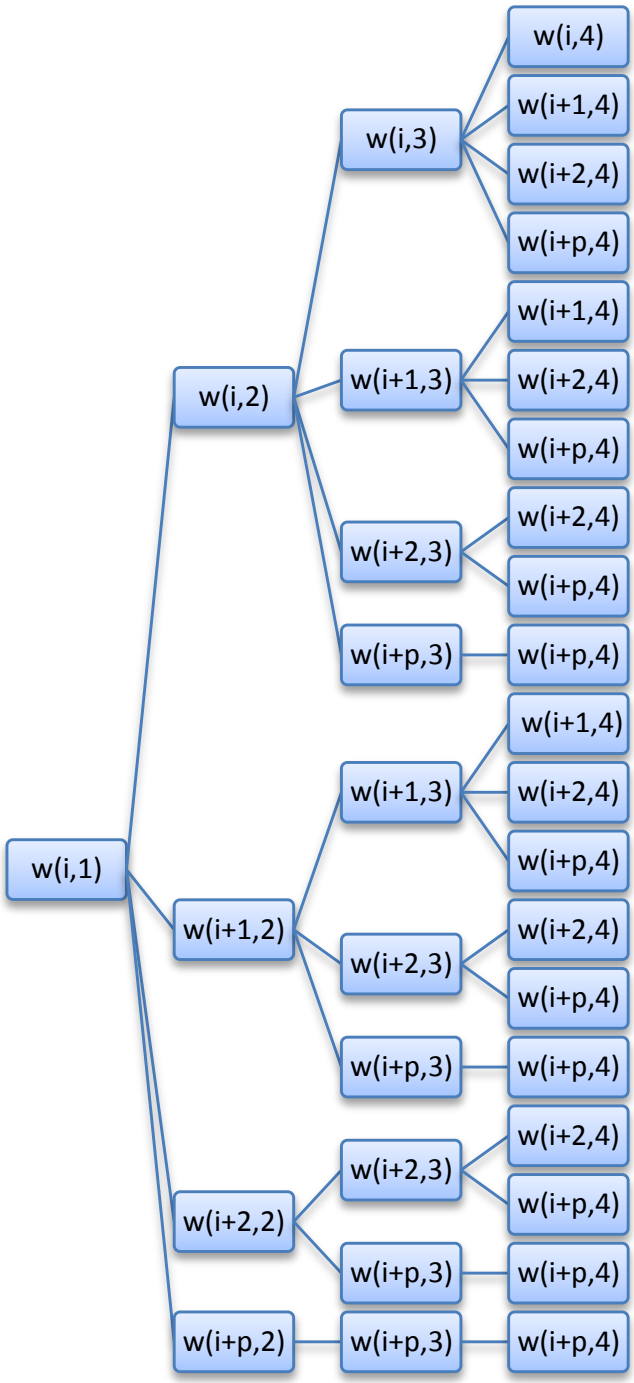


Figure 5.28: Tree of Weights for Integral Evaluation for $p = 3$

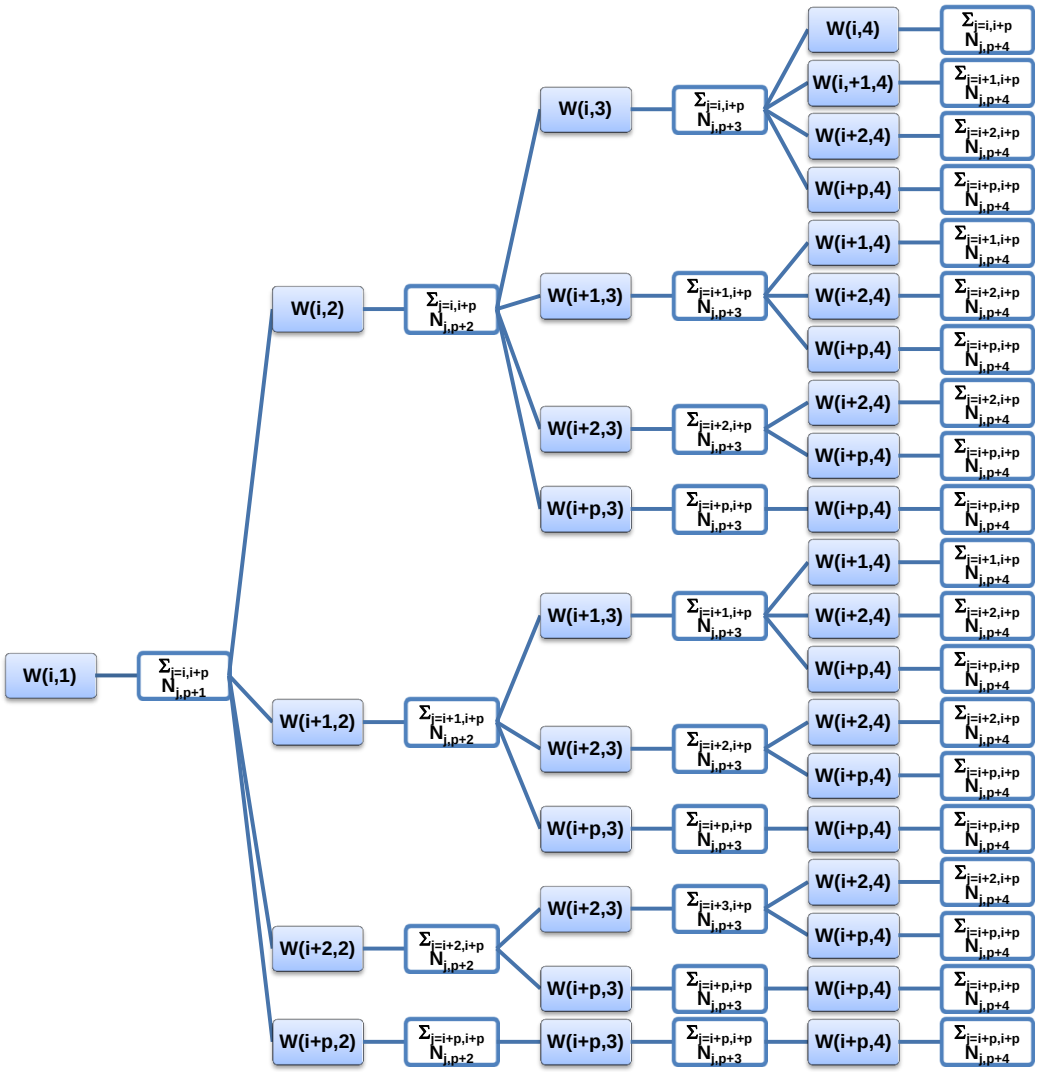


Figure 5.29: Tree of Weights with Summation Terms

For simplicity, only the most complex $(p+1)^{th}$ integral is shown in the algorithm. In the algorithm, *basisIndex* is the index of the basis function being integrated. The first integral is captured by the initial weight, *startWeight*, and therefore we start the recursion at $gradeIndex = 2$, where *grade* is the grade of integration sought.

Although the recursive algorithm generates each unique product term required for the integral at the cost of a single multiplication for the generation of each new term, and offers an intuitive approach that gives a valuable insight to the integral problem, it is more efficient to take a more direct 'depth-first' approach and traverse the tree structure in reverse. The distributive law can be applied so that the necessary summations can be grouped before applying the weights. Not only does this minimise the number of multiplications that are required at each level, it also means that only the very first branch of the tree need be considered as there is a large degree of repetition in the lower branches of the tree. Employing the same iterative technique for the addition of the basis functions to generate the $p+1$ summations needed, summations of weighted

Algorithm 5.3 Integral Recursion

function integralRecursion
input : $basisIndex,\ startWeight,\ gradeIndex\ \ grade$
output : $integral, I$
for $index = basisIndex \rightarrow endIndex$
  $currentWeight = startWeight * weight_{index,gradeIndex}$
  $I = I + weight * basisSum_{index,gradeIndex}$
  if $gradeIndex \neq grade$
    $integralRecursion(index,\ currentWeight,\ gradeIndex + 1)$

basis function sums can be evaluated. This cycle can be repeated iteratively so that only $p + 1$ multiplications and $p + 1$ additions are needed at each level. The process is graphically illustrated for the fourth integral required for the cubic case in Figure 5.30.

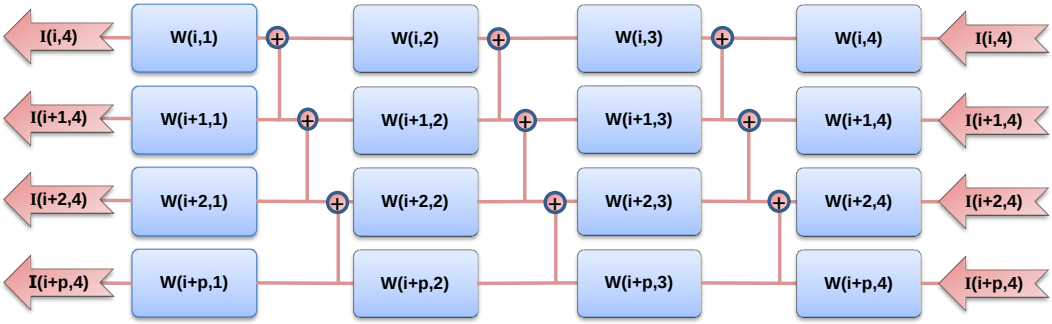


Figure 5.30: Efficient Tree of Weights with Summation Terms

The integral value, $I_{i,grade}$, where i is the basis function index and $grade$ is the grade of integral, is initialised with the appropriate sum term of higher order basis functions before passing through the iterative cycle of product summations, as depicted. Additional integrals can be calculated in the same fashion by starting the cycle at the appropriate level of the tree with an initial seed holding the sum corresponding higher order basis functions. An additional benefit of this approach is that all non-zero integral samples on the knot can be calculated simultaneously by simply choosing the appropriate weighting term at the last step of the algorithm $w(i, 1)$ to $w(i + p, 1)$. The overall algorithm, calculating each integral term up to a specified maximum, is summarised in Algorithm 5.4. As in Algorithm 5.3, $gradeIndex$ iterates over the necessary integrations up to a maximum of $grade$. $termIndex$ indexes the output integral terms being calculated and determines the initial sum of basis functions needed to initiate the algorithm.

Once the integrals have been sampled at each of the knots, they must be scaled by the weight introduced by the magnitude of the step change that leads to the formation of the Dirac delta at the knot. The term "Dirac Delta Weight" will be used to refer to this weight. The Dirac Delta Weight can be calculated by differencing the step functions on either side of the knot. The direction of the step change gives the sign of the weight. This process can be seen in Figures 5.21 and 5.9.

Algorithm 5.4 Efficient Integral Calculation

Initialise integral, I
for $gradeIndex = grade \rightarrow 1$
  for $termIndex = gradeIndex \rightarrow grade$
    $I_{p,\ termIndex} = weight_{p,\ gradeIndex} * I_{p,\ termIndex}$
    for $basisIndex = p - 1 \rightarrow 0$
      $I_{basisIndex,\ termIndex} = I_{basisIndex+1,\ termIndex} + weight_{basisIndex,gradeIndex} * I_{basisIndex,termIndex}$

The remaining terms to be evaluated at either end of the knot vector are typically of 0 value for the majority of basis functions, as they are most often composed of terms involving a product of a basis function integral and derivative, one of which is typically 0 at either end of the knot vector. Only the first and last $p + 1$ basis functions need special consideration.

The Analytic algorithm to compute all four Gramian matrices is summarised in Algorithm 5.5. As before, the Gramian matrices are labeled according to their respective derivative components as follows $G00$, $G11$, $G22$ and $G10$, while $I00$ and $D00$ represent the respective integrals and derivatives. The $spanIndex$ is the span on the knot vector being addressed, and finally, the $basisIndex$ indexes the non zero basis functions on the given span..

Algorithm 5.5 Efficient Analytic Stiffness Terms

Compute extended knot vector
Compute weights
Initialise Gramian matrices with constant terms resulting from multiple knots
$basisIndex = 0$
for $spanIndex = p \rightarrow endSpan - p$
  Compute $p + 1$ non-zero higher order Basis Functions up to degree $p + p + 1$
  Compute $p + 1$ non-zero Derivative Differences up to p
  Compute $(p + 1)^{th}$, p^{th}, $(p - 1)^{th}$ and $(p - 3)^{th}$ integrals (see Algorithm 5.4)
  for $i = 1 \rightarrow p + 1$
    for $j = i \rightarrow p + 1$
      $G00_{i+spanIndex,j+spanIndex} += I00_{basisIndex+j} * D00_{basisIndex+i}$
      $G11_{i+spanIndex,j+spanIndex} += I11_{basisIndex+j} * D11_{basisIndex+i}$
      $G22_{i+spanIndex,j+spanIndex} += I22_{basisIndex+j} * D22_{basisIndex+i}$
      $G10_{i+spanIndex,j+spanIndex} += I10_{basisIndex+j} * D10_{basisIndex+i}$
  $basisIndex++$

5.4.3 Discussion

This section has developed and presented novel algorithms implementing an analytic evaluation of the integral and repeated integrals of a B-Spline basis function. These algorithms form the backbone of the novel algorithm presented for the efficient analytic generation of the four Gramian matrices associated with the stiffness matrix of an Active B-Spline surface. Much like the algorithm presented in the previous section

concerning Gaussian Quadrature, the algorithm takes advantage of the algorithmic efficiencies that can be gained by generating the B-Spline basis functions and their derivatives on a sample-by-sample basis before assembling into the final Gramian matrices. The key mathematical benefit of this approach is that it requires only a single sampling of function values per span in contrast to the four samples required by the Gaussian Quadrature approach. Additionally, the approach makes no assumptions about the regularity of the knot vector in order to achieve this efficiency. Full testing and comparative results for both the Analytic approach and Gaussian Quadrature approaches developed are presented in Section 5.9.

5.5 Extension to varying material properties

In the literature, the material properties are usually treated as constants in order to simplify the mathematics and thus far, this approach has been adopted in this thesis. However, this section demonstrates that the special case presented in Section 5.4 also facilitates the evaluation of the problem where the material properties are varying over the domain of integration. The mathematical mechanics of the problem are essentially unchanged. However, the integration by parts formula must be extended to handle the extra term. This can be done by integrating the product rule for three terms such that

$$\int u(x)v(x)w'(x)dx = u(x)v(x)w(x) - \int u(x)w(x)v'(x)dx - \int v(x)w(x)u'(x)dx \quad (5.36)$$

Applying this to

$$\int_0^1 \alpha(u,v)N_{i,p}(u)N_{j,p}(u)du \quad (5.37)$$

gives

$$\begin{aligned}
\int_0^1 \alpha(u,v)N_{i,p}(u)N_{j,p}(u)du = {} & \left[N_{i,p}N_{j,p}\int_0^1 \alpha(u,v)du\right]_0^1 - \left[N_{i,p}N_{j,p}^{(1)}\int_0^1\int_0^1 \alpha(u,v)du^2\right]_0^1 \\
& - \left[N_{i,p}^{(1)}N_{j,p}\int_0^1\int_0^1 \alpha(u,v)du^2\right]_0^1 + \left[N_{i,p}N_{j,p}^{(2)}\int_0^1\int_0^1\int_0^1 \alpha(u,v)du^3\right]_0^1 \\
& + 2\left[N_{i,p}^{(1)}N_{j,p}^{(1)}\int_0^1\int_0^1\int_0^1 \alpha(u,v)du^3\right]_0^1 + \left[N_{i,p}^{(2)}N_{j,p}\int_0^1\int_0^1\int_0^1 \alpha(u,v)du^3\right]_0^1 \\
& - \left[N_{i,p}N_{j,p}^{(3)}\int_0^1\int_0^1\int_0^1\int_0^1 \alpha(u,v)du^4\right]_0^1 - 3\left[N_{i,p}^{(1)}N_{j,p}^{(2)}\int_0^1\int_0^1\int_0^1\int_0^1 \alpha(u,v)du^4\right]_0^1 \\
& - 3\left[N_{i,p}^{(2)}N_{j,p}^{(1)}\int_0^1\int_0^1\int_0^1\int_0^1 \alpha(u,v)du^4\right]_0^1 - \left[N_{i,p}^{(3)}N_{j,p}\int_0^1\int_0^1\int_0^1\int_0^1 \alpha(u,v)du^4\right]_0^1
\end{aligned}$$

$$
\begin{aligned}
&+4\left[N_{i,p}^{(1)}N_{j,p}^{(3)}\int_0^1\int_0^1\int_0^1\int_0^1\int_0^1\alpha(u,v)du^5\right]_0^1\\
&+6\left[N_{i,p}^{(2)}N_{j,p}^{(2)}\int_0^1\int_0^1\int_0^1\int_0^1\int_0^1\alpha(u,v)du^5\right]_0^1\\
&+4\left[N_{i,p}^{(3)}N_{j,p}^{(1)}\int_0^1\int_0^1\int_0^1\int_0^1\int_0^1\alpha(u,v)du^5\right]_0^1\\
&-10\left[N_{i,p}^{(2)}N_{j,p}^{(3)}\int_0^1\int_0^1\int_0^1\int_0^1\int_0^1\alpha(u,v)du^6\right]_0^1\\
&-10\left[N_{i,p}^{(3)}N_{j,p}^{(2)}\int_0^1\int_0^1\int_0^1\int_0^1\int_0^1\alpha(u,v)du^6\right]_0^1\\
&+20\left[N_{i,p}^{(3)}N_{j,p}^{(3)}\int_0^1\int_0^1\int_0^1\int_0^1\int_0^1\int_0^1\alpha(u,v)du^7\right]_0^1\\
&+\int_0^1 N_{i,p}N_{j,p}^{(4)}\int_0^1\int_0^1\int_0^1\int_0^1\alpha(u,v)du^4\,du\\
&+\int_0^1 N_{i,p}^{(4)}N_{j,p}\int_0^1\int_0^1\int_0^1\int_0^1\alpha(u,v)du^4\,du\\
&-4\int_0^1 N_{i,p}^{(1)}N_{j,p}^{(4)}\int_0^1\int_0^1\int_0^1\int_0^1\int_0^1\alpha(u,v)du^5\,du\\
&-4\int_0^1 N_{i,p}^{(4)}N_{j,p}^{(1)}\int_0^1\int_0^1\int_0^1\int_0^1\int_0^1\alpha(u,v)du^5\,du\\
&+10\int_0^1 N_{i,p}^{(2)}N_{j,p}^{(4)}\int_0^1\int_0^1\int_0^1\int_0^1\int_0^1\int_0^1\alpha(u,v)du^6\,du\\
&+10\int_0^1 N_{i,p}^{(4)}N_{j,p}^{(2)}\int_0^1\int_0^1\int_0^1\int_0^1\int_0^1\int_0^1\alpha(u,v)du^6\,du\\
&-20\int_0^1 N_{i,p}^{(3)}N_{j,p}^{(4)}\int_0^1\int_0^1\int_0^1\int_0^1\int_0^1\int_0^1\int_0^1\alpha(u,v)du^7\,du\\
&-20\int_0^1 N_{i,p}^{(4)}N_{j,p}^{(3)}\int_0^1\int_0^1\int_0^1\int_0^1\int_0^1\int_0^1\int_0^1\alpha(u,v)du^7\,du
\end{aligned}
\tag{5.38}
$$

From Equation 5.38, it can be seen that the equation reduces in the same way as those in Section 5.4.1 to a series of samples of quantities that can be evaluated, and in the same way, the constant terms generally are 0 at either end of the knot vector. The terms can be grouped in such a way that once the α or β function is differentiable or integrable to the point of achieving the Dirac Delta condition with the derivatives of either of the basis functions, an analytic solution can be obtained.

5.6 Extension to Mass, Damping and Forcing Function

Active B-Spline/NURBS surfaces are used to model the response, and often the dynamics, of a surface under the influence of applied forces according to the following equation (Terzopoulos and Qin, 1994).

$$M\ddot{p} + D\dot{p} + Kp = F \tag{5.39}$$

where M, D, and F are the mass, damping and forcing matrices respectively, and are given by

$$F = \int_{\sigma} \Phi^T f(u,v)\, dudv \tag{5.40}$$

$$M = \int_{\sigma} \mu(u,v)\ \Phi^T \Phi\ dudv \tag{5.41}$$

$$D = \int_{\sigma} \gamma(u,v)\ \Phi^T \Phi\ dudv \tag{5.42}$$

The forcing function that describes the external forces applied to the Active B-Spline Surface must be transformed such that the forces are mapped to the control points. This is achieved by a mapping that involves the product with the B-Spline basis functions. Thus, the forcing function can be handled in the same way as the varying material properties as outlined above. Equally, if mass and damping properties of the surface are to be considered, their distributions can be accommodated in a similar fashion (Terzopoulos and Qin, 1994).

5.7 Extension to Active Volumes and Higher Dimension Models

The energy of an Active B-Spline Volume (Doi et al., 2002) is typically given by

$$\begin{aligned} E(S(u,v,w)) = \int_{\Omega} \alpha \left(\left|\frac{\partial S}{\partial u}\right|^2 + \left|\frac{\partial S}{\partial v}\right|^2 + \left|\frac{\partial S}{\partial w}\right|^2 \right) \\ + \beta \left(\left|\frac{\partial^2 S}{\partial u^2}\right|^2 + 2\left|\frac{\partial^2 S}{\partial u \partial v}\right|^2 + \left|\frac{\partial^2 S}{\partial v^2}\right|^2 \right. \\ \left. + 2\left|\frac{\partial^2 S}{\partial u \partial w}\right|^2 + \left|\frac{\partial^2 S}{\partial w^2}\right|^2 + 2\left|\frac{\partial^2 S}{\partial v \partial w}\right|^2 \right) dudvdw \end{aligned} \tag{5.43}$$

It is clear that the equation is separable in the same way as the B-Spline Surface and that each parameter can be handled in the same way as outlined in Section 5.2 for the calculation of the corresponding Gramian matrices. The technique can also be generalised for higher dimensions.

5.8 Extension to NURBS

Although the B-Spline representation discussed so far facilitates the modelling of a wide range of shapes, there are a number of curve and surface types that cannot be exactly represented using polynomials alone, e.g., circles, ellipses, hyperbolas, cylinders, cones, spheres etc. These conics can however be represented using rational functions of polynomials. NURBS are the rational version of B-Spline representations that facilitate the precise representation of all the conics.

NURBS curves and surfaces were introduced in Chapter 4, Section 4.1.2. Their treatment is much the same as that of B-Spline curves and surfaces, only in the case of NURBS, the basis functions are rational. For convenience the relevant equations are repeated such that a NURBS basis function is defined by

$$R_{i,p}(u) = \frac{N_{i,p}(u)w_i}{\sum_{i=0}^{n} N_{i,p}(u)w_i} \tag{5.44}$$

Analogous to the B-Spline equations developed in Chapter 4, Section 4.1.1, the basis functions for a parametric NURBS surface are defined as

$$R_{i,j}^{p,q}(u,v) = \frac{N_{i,p}(u)N_{j,q}(v)w_{i,j}}{\sum_{i=0}^{m}\sum_{j=0}^{n} N_{i,p}(u)N_{j,q}(v)w_{i,j}} \tag{5.45}$$

Further information on NURBS can be found in (Piegl, 2005).

Much like B-Splines, Gaussian Quadrature is typically used to integrate energy functions associated with NURBS curves and surfaces (Qin et al., 1998). However, such schemes typically offer only approximations of the integral of rational functions, as the technique can only handle polynomials exactly (Rypl and Patzák, 2012). As discussed in Section 5.1, for the non-rational case, simplifications are often made where the B-Spline is defined over a uniform knot vector. Such approximations are equally applicable to the rational case.

An analytic solution for the integral of a NURBS product is currently unavailable. However, the Analytic approach presented in Section 5.4 can be applied to the NURBS problem if the following simplification is made.

Over the domain of integration, the denominator of the NURBS basis functions changes slowly when compared to the polynomial numerator. It is common practice to select quadrature rules that give exact integration when the denominator is constant (Auricchio et al., 2012, Hughes et al., 2010, Cottrell et al., 2009). In this case the NURBS integrals reduce to a weighted version of those integrals presented for B-Splines (see Equations 5.2–5.5, inclusive) and the Analytic solution presented here can be employed.

5.9 Results

Sections 5.3.2 and 5.4 presented two alternative approaches to computing the four Gramian matrices that make up the stiffness matrix of an Active B-Spline surface. The Analytic algorithm presented in 5.4 was developed in an effort to achieve a more elegant and efficient solution. This section presents a detailed comparison of the alternative approaches, and analyses results in terms of efficiency, accuracy and stability.

5.9.1 Computational Complexity

This section compares the computational complexity of the two approaches both theoretically and practically with real performance related results.

A Brief Note on Implementation

Sections 5.3.2 and 5.4 develop mathematical and algorithmic efficiencies without regard to programming efficiencies. A significant programming efficiency stems from the data structures used in both the Gaussian Quadrature implementation and the Analytic equivalent. These data structures are primarily symmetric in their nature and should be stored as such. This makes indexing and memory allocation efficient and also facilitates the representation of much larger data models. The algorithms were developed in Matlab and implemented in Java for proof of concept as part of the Virtual Sculpting Environment presented in Chapter 6 and all experiments were run on a Dell Inspiron 6000 Laptop with a single core centrino processor.

Theoretical Results

This section presents theoretical analysis of the computational complexities of both the Gaussian Quadrature approach and the Analytic approach presented in the chapter. Unless otherwise stated, where plots are shown for varying numbers of control points, the degree is fixed for the cubic case, and where the degree varies, the number of control points is fixed at 100. The figures shown are the computations required for the generation of all four Gramian matrices in each case.

Figure 5.31 shows the number of samples of each basis function that must be computed per span as the degree of basis function is increased. As can be seen from the graph, using the Analytic approach, only one sample point is required per span, regardless of the degree of the curve employed. As the product of two p degree curves is under consideration in each case, Gaussian Quadrature requires $\lceil (p+p+1)/2 \rceil$ samples per span. This saving comes directly from the mathematical approach adopted.

Figures 5.32 and 5.33 show the total number of basis function evaluations required for both Gaussian Quadrature and the Analytic approach varying over an increasing number of points and increasing degree, respectively. As could be expected from Figure 5.31, the Analytic approach requires significantly fewer basis function evaluations. The relationship between the number of basis function calculations and the number of

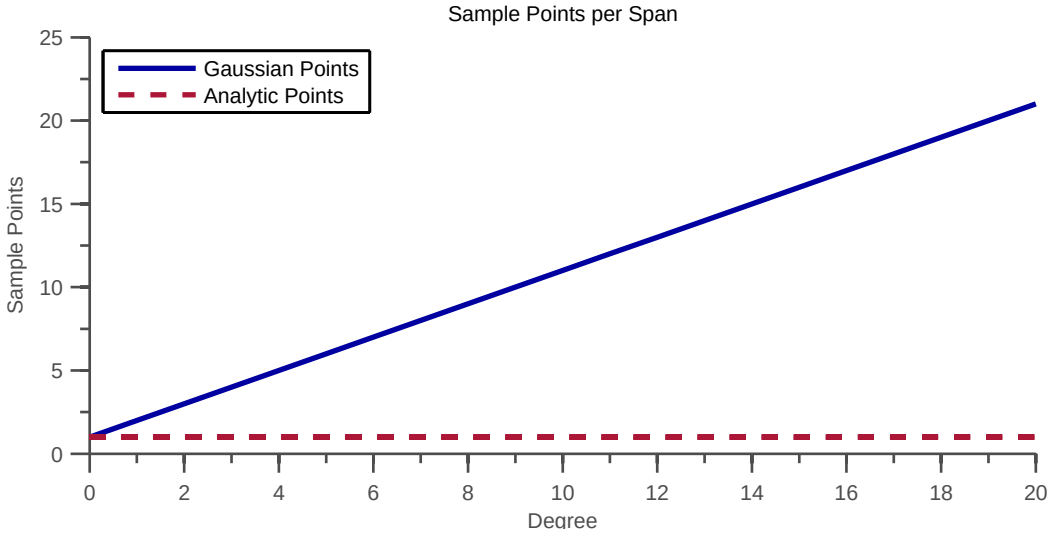


Figure 5.31: Number of Basis Function Samples Required per Span

points is $O(n)$ or linear for both approaches, as could be expected. The relationship between the number of basis function evaluations and the degree is $O(p^2)$ or quadratic for Gaussian Quadrature whereas it is $O(p)$ or linear for the Analytic approach. This is because the calculation is based on the number of samples required per span, the number of spans per basis function and the total number of basis functions. For the Gaussian Quadrature approach, this equates to $[\frac{2p+1}{2}][n+1][p+1]$ where n is the number of control points and p is the degree of the curve. Similarly, for the Analytic approach, this equates to $[1][n+1][p+1]$. In the Gaussian Quadrature case the first two of the terms are proportional to p, while for the Analytic case only the second term is proportional to p. This computational complexity refers directly to the mathematical approaches adopted. This process must be conducted for each Gramian matrix. Using the Gaussian Quadrature approach, terms of the basis function itself, the first derivative, and the second derivative (curves of degree p, $p-1$, and $p-2$, respectively) must be sampled. Using the Analytic approach, one derivative evaluation and four integral evaluations must be calculated.

Figures 5.34 and 5.35 show the number of basis function evaluations required for both approaches when the triangular structures presented in Figures 5.8 and 5.26 are used to iteratively calculate the required basis functions and derivatives for all four Gramian matrices in an 'all-in-one' fashion. As can be seen, this approach drastically reduces the number of computations required. This saving stems from the algorithmic efficiencies identified in the preceding sections and as can be seen from the figures, apply to both the Gaussian Quadrature algorithm and the Analytic Algorithm.

The heretofore evaluation of the computational complexity does not complete the analysis however, as the comparisons do not take into account that for the Analytic approach, higher order basis functions must be calculated at greater computational expense than the lower order basis functions required to satisfy the Gaussian Quadrature approach. The preceding figures showed the number of basis function evaluations

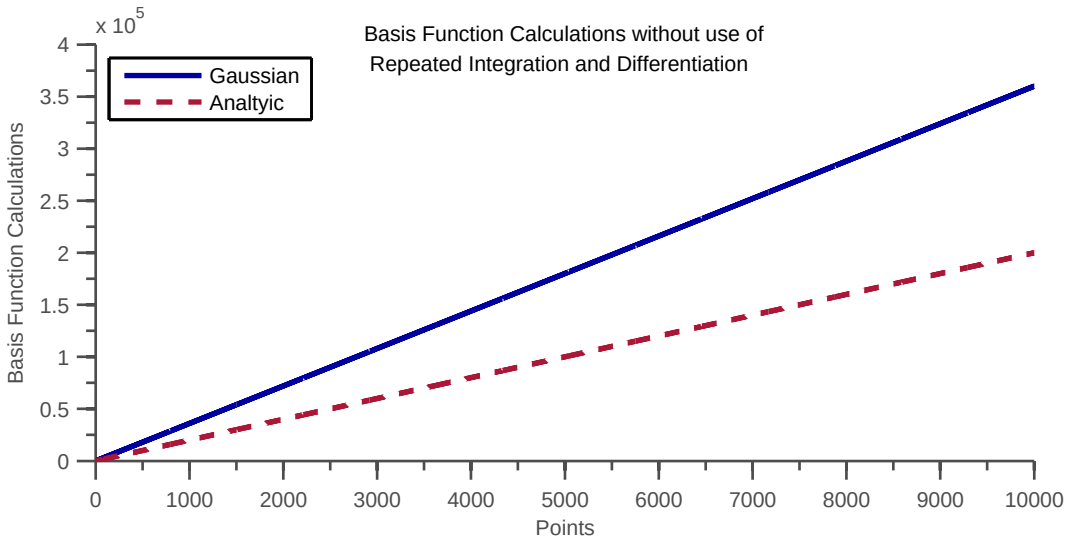


Figure 5.32: Number of Basis Function Evaluations Required vs Number of Points

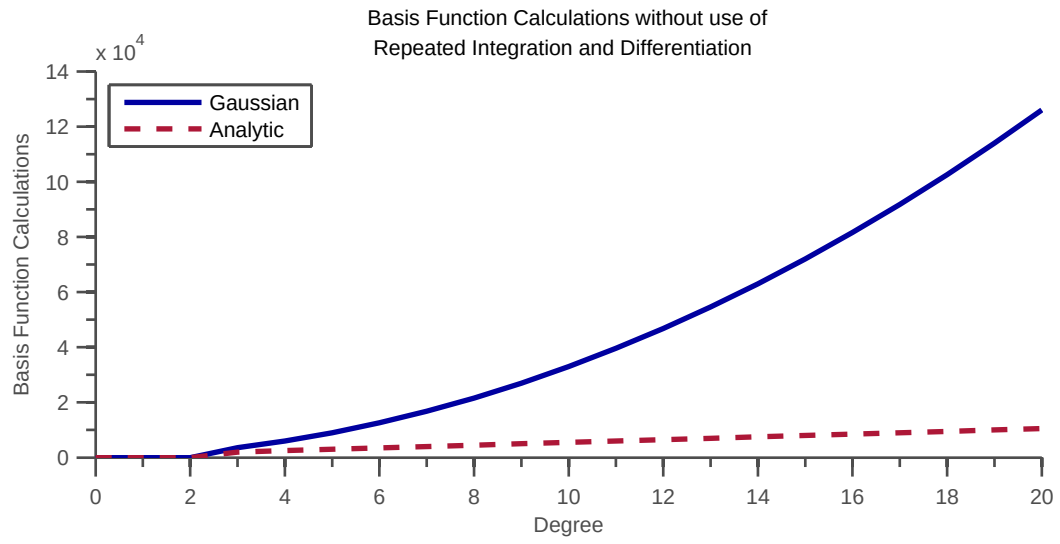


Figure 5.33: Number of Basis Function Evaluations Required vs Degree

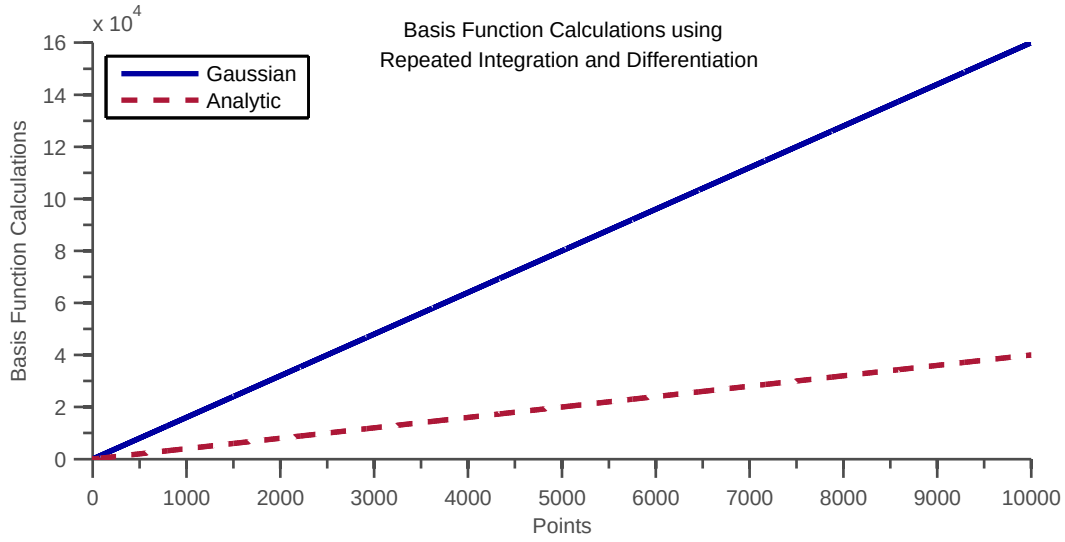


Figure 5.34: Number of Basis Function Evaluations Required vs Number of Points with Efficient Re-use of Values

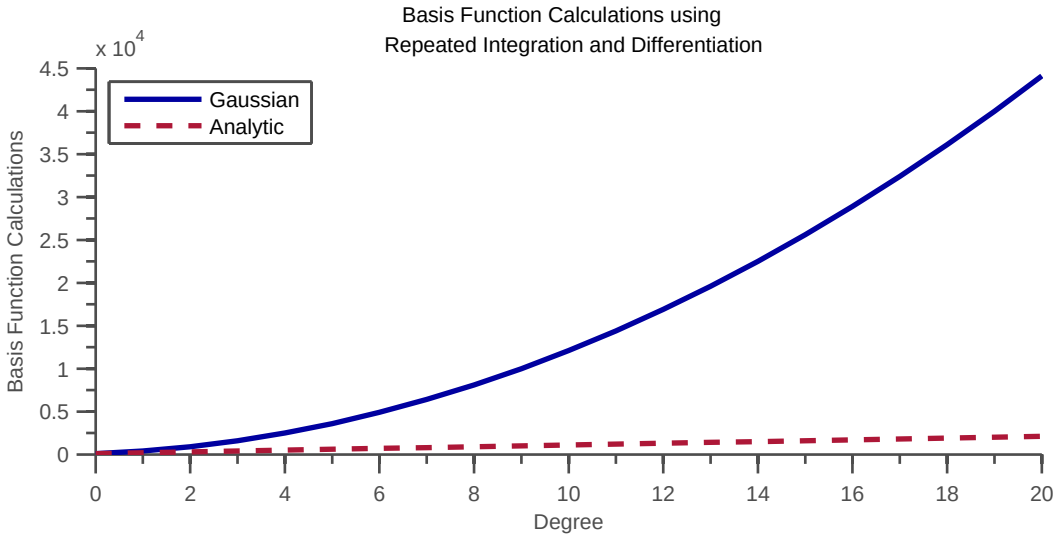


Figure 5.35: Number of Basis Function Evaluations Required vs Degree with Efficient Re-use of Values

needed. Figures 5.36 and 5.37 indicate the relative number of operations involved in these evaluations. The plots take into account the computational complexity of the basis function evaluations themselves by considering the number of iterations of the relevant recursions necessary for both approaches. As can be seen from the graphs, in spite of the additional overhead incurred by the Analytic approach in evaluating the higher order basis functions, it still offers a much less computationally intensive solution. In both cases, the relationship between the total number of operations required per individual evaluation and the degree is $O(p^2)$.

For clarity, Figure 5.38 shows the same results presented in Figure 5.37 over a more practical range of degrees, $[0, 5]$. It will be noted that the savings increase with higher degree and significant savings are achieved over the most commonly used lower degree curves.

Figures 5.39 and 5.40 show total mathematical operation counts, i.e., subtractions, additions, multiplications and divisions, for both approaches in computing all four Gramian matrices. For the Gaussian approach, the calculations include operation counts for shifting of limits, weight calculations, basis function evaluations, derivative evaluations, and finally the products and summations required for the assembly of the Gramian matrices. For the Analytic approach, the total operation count includes the weighted sums and products of the higher order basis functions, the derivative evaluations and differences, and finally the products and summations required for the assembly of the Gramian matrices. The actual mathematical operation counts were based on the preceding analysis of the number of basis function calculations and the number of branches required to create the relevant triangular structures as illustrated in Figure 5.8 and Figure 5.26. For the Gaussian approach, the forward traversal of the triangular structure equates to $([p+1][p+1])/2$ branching iterations, while for the higher order basis functions required by the Analytic approach, this equates to $([(2p+1+1)][(2p+1P+2])/2$ branching iterations. Similarly for the relevant derivatives, the backward traversal of the triangular structures equate to $([2+1][2+1])/2$ iterations to calculate the second derivative for Gaussian Quadrature and $([p][p+])/2$ to calculate the p^{th} derivative for the Analytic approach. The number of operations per iteration was counted according to the algorithms presented in Piegl and Tiller (1997). The products of the resulting terms are counted and their additions to the Gramian matrices, while taking into account that there are $\binom{p+1}{2} + p + 1$ unique values for each of the sampling grids for each of the four Gramian matrices.

Finally, Figure 5.41 shows a theoretical 3D plot depicting the relative computational complexity of the Gaussian Quadrature and Analytic approaches over both varying points and degree.

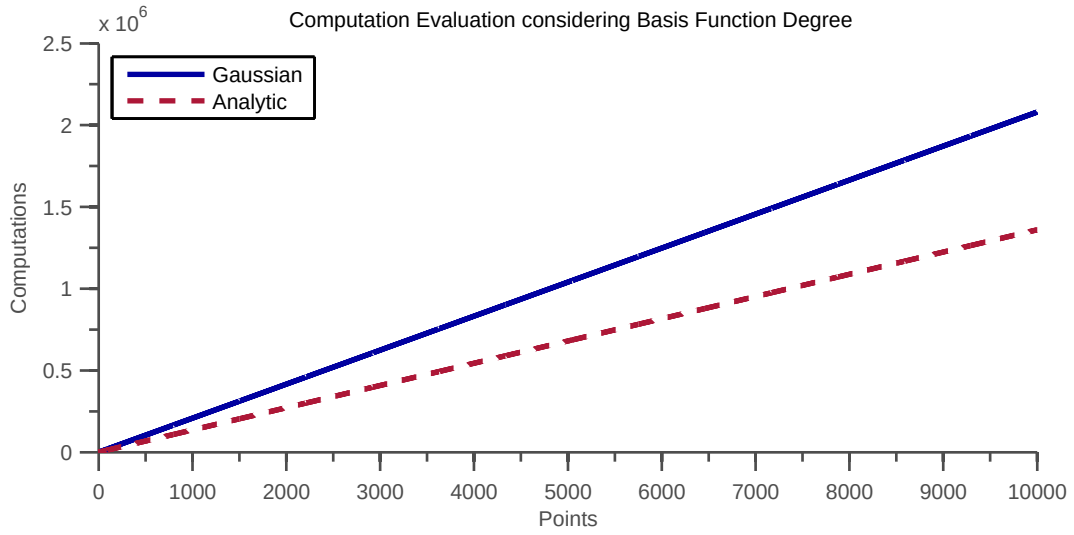


Figure 5.36: Relative Computations Required for Basis Function Evaluation Considering Degree vs Points

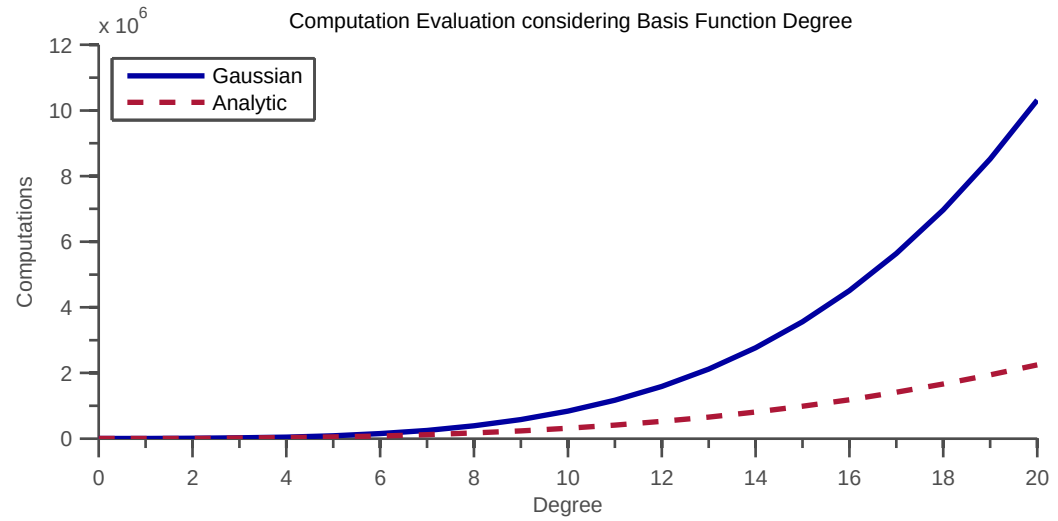


Figure 5.37: Relative Computations Required for Basis Function Evaluation Considering Degree vs Degree

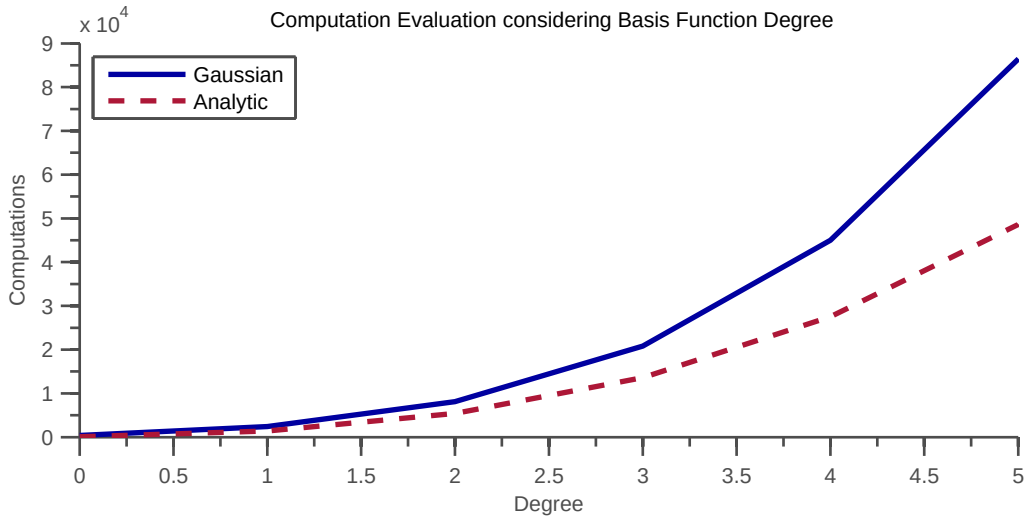


Figure 5.38: Zoomed Relative Computations Required for Basis Function Evaluation Considering Degree

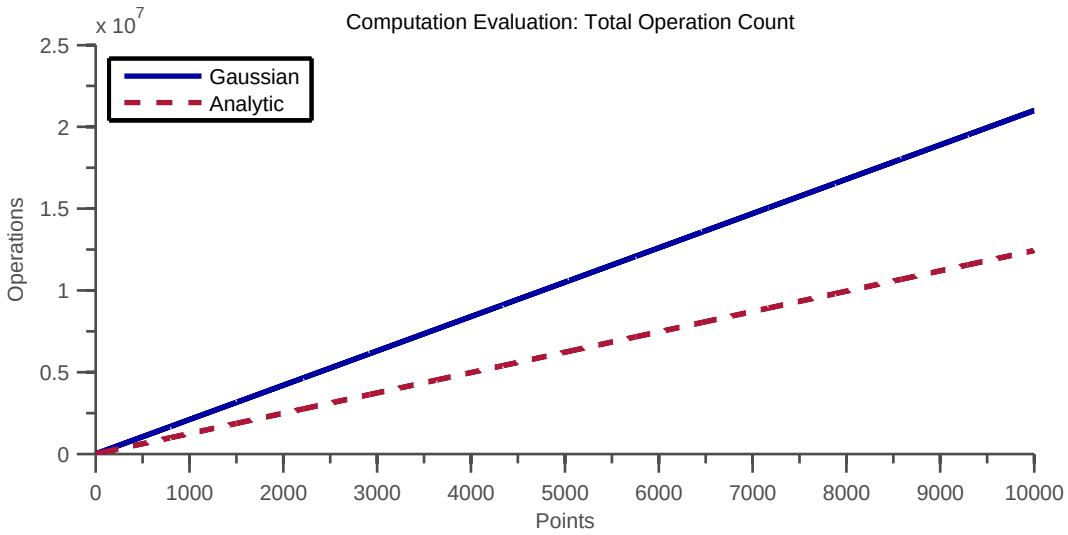


Figure 5.39: Total Mathematical Operation Count vs Points

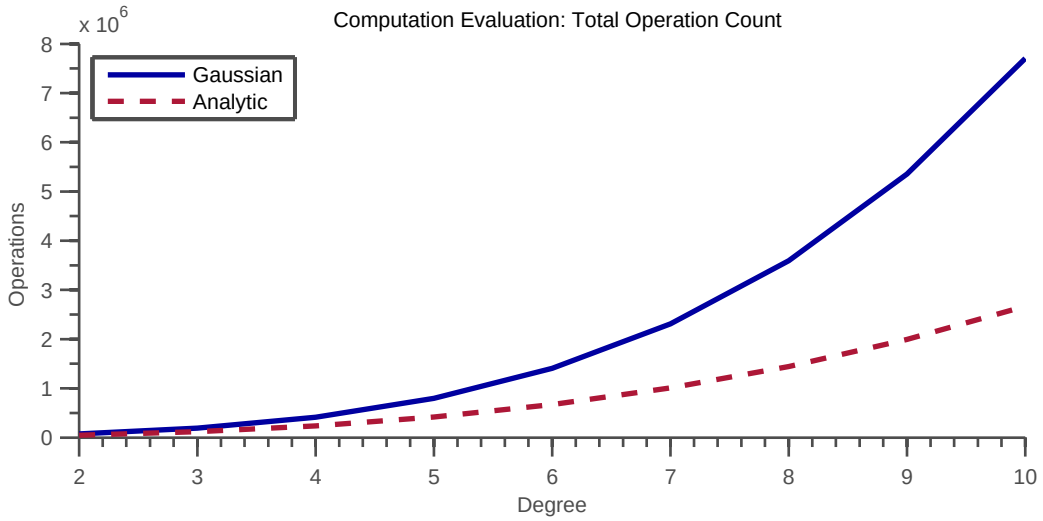


Figure 5.40: Total Mathematical Operation Count vs Degree

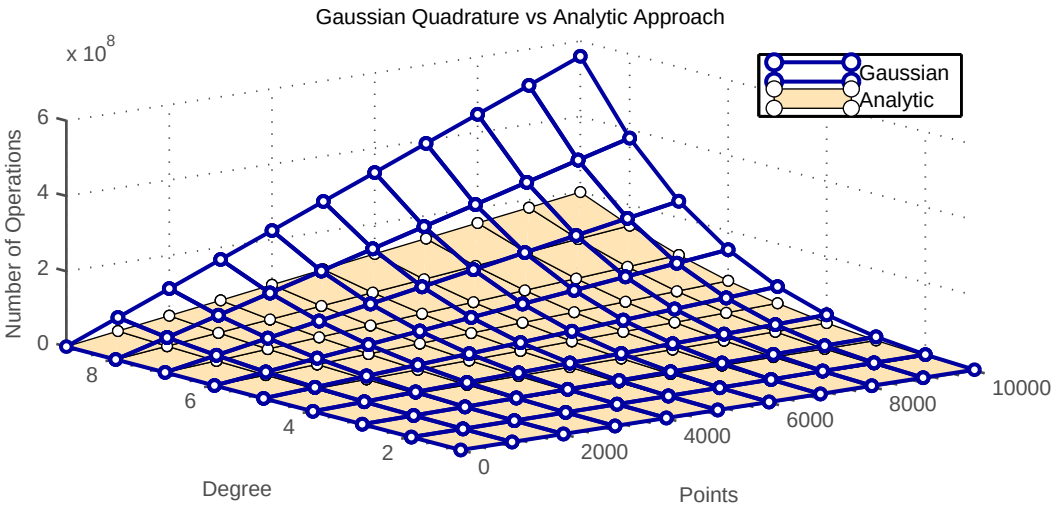


Figure 5.41: Total Mathematical Operation Count vs Points and Degree

Practical Results

This section presents real computation times obtained from the implementation of the algorithms presented in Sections 5.3 and 5.4.

Figures 5.42 – 5.44, inclusive, show the average total computation time required to compute all four Gramian matrices using both the Gaussian Quadrature and Analytic approaches.

The computation times reflect the theoretical results deduced in the previous section. There is an improvement in the expected performance with respect to an increasing number of points and this is mainly down to the nature of the computations for each case. The Gaussian Quadrature approach suffers from a significant increase in the number of multiplication operations with increasing number of points which impacts on its performance.

Figure 5.45 shows the overall improvement in computation times with varying degree and varying number of points. The percentage improvement is effectively constant for increasing number of points. The slight variation that can be observed at the lower range of points is due to the special case conditions at the last $p + 1$ knots at either end of the knot vector. Where there are very small numbers of points, these special conditions dominate the overall calculations and thus impact on the overall percentage improvement. The impact of these special cases becomes negligible for increasing numbers of points.

The percentage improvement/reduction in time can be seen to increase with increasing degree. The overall improvements range from 63% to 77% over the range of degrees presented. This means that the Analytic solution ranges from being 2.7 times to 4.3 times faster than Gaussian Quadrature. The practical relevance of these improvements is further discussed in Section 5.9.3

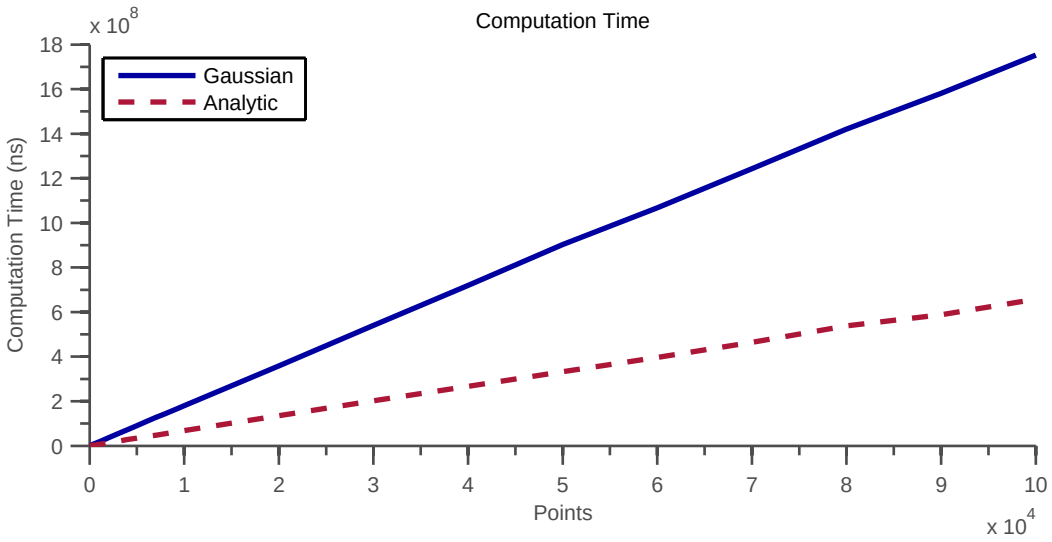


Figure 5.42: Computation Time for Full Computation of Gramian Matrices in Nanoseconds vs Points

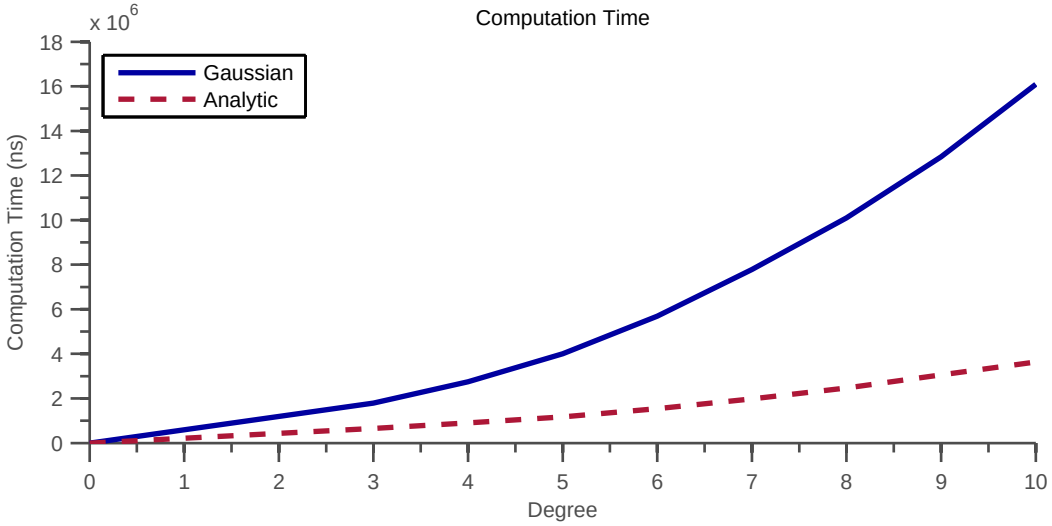


Figure 5.43: Computation Time for Full Computation of Gramian Matrices in Nanoseconds vs Degree

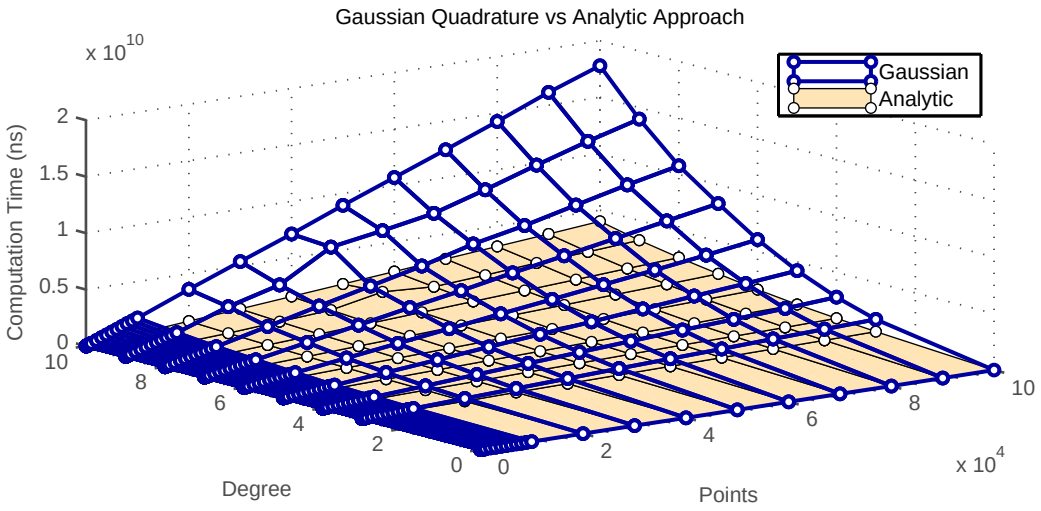


Figure 5.44: Computation Time for Full Computation of Gramian Matrices in Nanoseconds vs Points and Degree

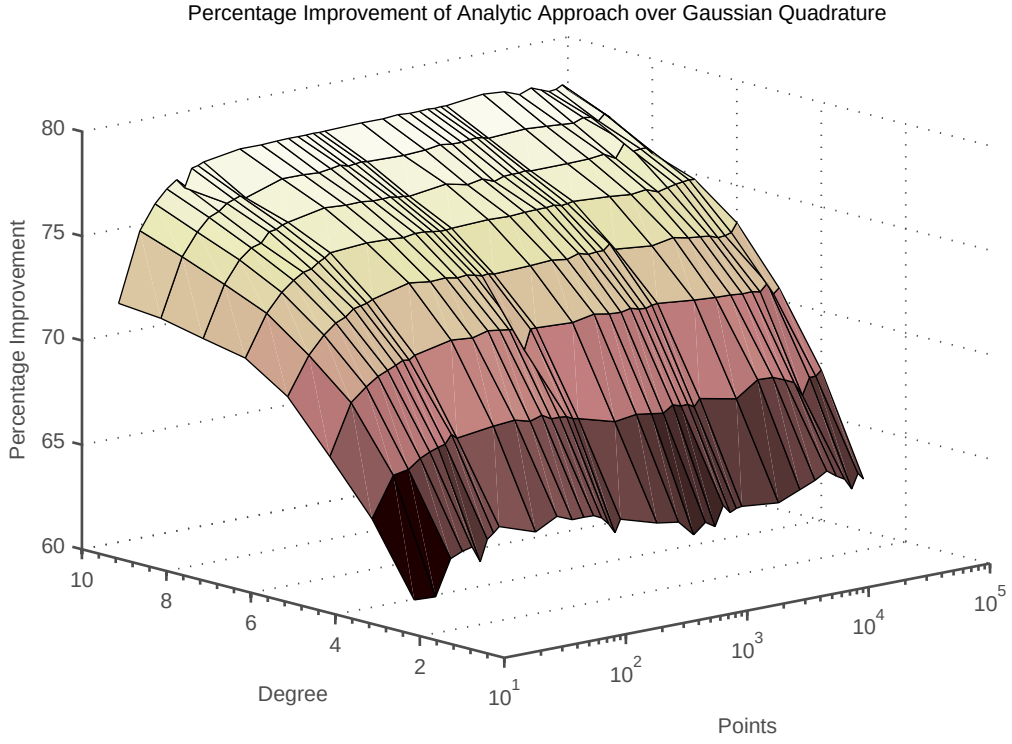


Figure 5.45: Percentage Improvement in Computation Time of Analytic Approach over Gaussian Quadrature

5.9.2 Accuracy and Stability

In this section, the accuracy and stability of the Gaussian Quadrature and Analytic approaches are compared and analysed. Respective performances are discussed in detail and deviations in the results are investigated and explained. Suggestions for mitigating deviations are made and investigated. Finally, a sample set of complete Gramian matrices computed using both the Gaussian Quadrature and the Analytic approach developed in this thesis are shown.

Comparison

In this section, the accuracy and stability of the Gaussian Quadrature and Analytic approaches are compared and analysed. Exact results obtained symbolically are also presented for comparison. Schumaker (2007) tests the numerical accuracy of Gaussian Quadrature against Divided Differences for the computation of the integral of a product of two B-Spline curves by testing the algorithms over an increasingly small knot span. Their experiment is adapted and extended here for examining the robustness of the approach in computing all four Gramian matrices associated with the various terms that contribute to the Stiffness matrix of an Active B-Spline Surface. In each experiment a knot vector is chosen that includes $p+1$ repeated knots at either end of an otherwise even knot spacing but for a single span that is iteratively reduced from 1 to 0.1 to 10^{-r}. Results for an element of each Gramian matrix that includes a basis function supported by the span are presented for comparison.

The results are organised as follows:

- Table 5.3 shows results for $\int N_{8,3}N_{8,3}$ on knot vector $U = \{0,0,0,0,1,2,3,4,5,6,6+10^{-r},7,8,9,10,11,12,13,13,13,13\}$. Similarly, Tables 5.4–5.6, inclusive, show results for the same product for the remaining Gramian Matrices.
- Tables 5.7–5.10, inclusive, show results for a degree 4 curve.
- Tables 5.11–5.14, inclusive, show results for the degree 5 curve.

Note: The numbers underlined in Tables 5.3–5.14, inclusive, represent the decimal place where the results deviate from the exact solution.

Table 5.3: Gaussian Quadrature vs Analytic $\int N_{8,3}N_{8,3}$ [a]

r	Exact	Gaussian	Analytic
0	0.479365079365079	0.479365079365079	0.479365079365079
1	0.464742598397771	0.464742598397771	0.464742598397770
2	0.461726816414083	0.461726816414083	0.461726816414082
3	0.461410918985698	0.461410918985698	0.461410918985697
4	0.461379188554965	0.461379188554965	0.461379188554959
5	0.461376014107772	0.461376014107772	0.461376014107772
6	0.461375696649014	0.461375696649015	0.461375696649668
7	0.461375664902998	0.461375664902998	0.461375664908268
8	0.461375661728395	0.461375661728395	0.461375661718229
9	0.461375661410935	0.461375661410935	0.461375661363573
10	0.461375661379189	0.461375661379189	0.461375663214036

Table 5.4: Gaussian Quadrature vs Analytic $\int N^1_{8,3}N^1_{8,3}$

r	Exact	Gaussian	Analytic
0	0.666666666666667	0.666666666666667	0.666666666666667
1	0.632810867293626	0.632810867293626	0.632810867293626
2	0.623393710608519	0.623393710608519	0.623393710608518
3	0.622340602600689	0.622340602600689	0.622340602600721
4	0.622234072691490	0.622234072691490	0.622234072691725
5	0.622223407393580	0.622223407393580	0.622223407393615
6	0.622222340740603	0.622222340740602	0.622222340720259
7	0.622222234074073	0.622222234074072	0.622222233945981
8	0.622222223407407	0.622222223407407	0.622222225476104
9	0.622222222340741	0.622222222340741	0.622222219530126
10	0.622222222234074	0.622222222234074	0.622222092461922

Table 5.5: Gaussian Quadrature vs Analytic $\int N^2_{8,3}N^2_{8,3}$

r	Exact	Gaussian	Analytic
0	2.66666666666667	2.66666666666667	2.66666666666667
1	3.34378265412748	3.34378265412748	3.34378265412748
2	3.53212578782962	3.53212578782962	3.53212578782962
3	3.55318794798621	3.55318794798621	3.55318794798578
4	3.55531854617021	3.55531854617021	3.55531854616891
5	3.55553185212839	3.55553185212839	3.55553185213185
6	3.55555318518795	3.55555318518795	3.55555318546162
7	3.55555531851855	3.55555531851854	3.55555531931451
8	3.55555553185185	3.55555553185184	3.55555552224570
9	3.55555555318519	3.55555555318519	3.55555568782514
10	3.55555555531852	3.55555555531852	3.55555640324754

Table 5.6: Gaussian Quadrature vs Analytic $\int N^1_{8,3}N^0_{8,3}$

r	Exact	Gaussian	Analytic
0	0	$-5.87348196621648e^{-18}$	0
1	0	$-2.33298964604716e^{-16}$	$6.66133814775094e^{-16}$
2	0	$3.12780746233492e^{-17}$	$4.44089209850063e^{-16}$
3	0	$9.25131006265147e^{-17}$	$0.03551744741647e^{-16}$
4	0	$-3.27582547725088e^{-18}$	$3.88578058618805e^{-14}$
5	0	$-2.72567266455637e^{-16}$	$5.15143483426073e^{-14}$
6	0	$-3.70841000106120e^{-16}$	$3.87245790989255e^{-13}$
7	0	$-4.67825049993792e^{-16}$	$5.89062132405616e^{-12}$
8	0	$-2.08983821753874e^{-16}$	$3.59985374842609e^{-11}$
9	0	$2.22925876532200e^{-16}$	$-3.84451359636273e^{-10}$
10	0	$1.11811161451650e^{-16}$	$-6.06099770372737e^{-9}$

[a] The underlined number in each case represents the decimal place where the results deviate from the exact solution.

Table 5.7: Gaussian Quadrature vs Analytic $\int N_{8,4}N_{8,4}$

r	Exact	Gaussian	Analytic
0	0.430417768959436	0.430417768959436	0.430417768959436
1	0.408862460283437	0.408862460283437	0.408862460283438
2	0.406114699442957	0.406114699442957	0.406114699442958
3	0.405835553494608	0.405835553494608	0.405835553494609
4	0.405807596778624	0.405807596778624	0.405807596778624
5	0.405804800687391	0.405804800687391	0.405804800687391
6	0.405804521074072	0.405804521074072	0.405804521074073
7	0.405804493112699	0.405804493112699	0.405804493112598
8	0.405804490316561	0.405804490316561	0.405804490317407
9	0.405804490036947	0.405804490036947	0.405804490036211
10	0.405804490008986	0.405804490008986	0.405804489942062

Table 5.8: Gaussian Quadrature vs Analytic $\int N^1_{8,4}N^1_{8,4}$

r	Exact	Gaussian	Analytic
0	0.486111111111111	0.486111111111111	0.486111111111110
1	0.410697731387387	0.410697731387387	0.410697731387384
2	0.402087934772304	0.402087934772304	0.402087934772303
3	0.401220755397126	0.401220755397127	0.401220755397126
4	0.401133980867521	0.401133980867521	0.401133980867518
5	0.401125302854310	0.401125302854310	0.401125302854331
6	0.401124435047392	0.401124435047392	0.401124435047404
7	0.401124348266645	0.401124348266645	0.401124348271150
8	0.401124339588569	0.401124339588569	0.401124339574327
9	0.401124338720762	0.401124338720762	0.401124338861805
10	0.401124338633981	0.401124338633981	0.401124337172381

Table 5.9: Gaussian Quadrature vs Analytic $\int N^2_{8,4}N^2_{8,4}$

r	Exact	Gaussian	Analytic
0	1.58333333333333	1.58333333333333	1.58333333333333
1	1.19684912788361	1.19684912788361	1.19684912788361
2	1.15726914557827	1.15726914557827	1.15726914557828
3	1.15322789822692	1.15322789822692	1.15322789822695
4	1.15282279981403	1.15282279981403	1.15282279981421
5	1.15278228008147	1.15278228008147	1.15278228008053
6	1.15277822800915	1.15277822800915	1.15277822801313
7	1.15277782280093	1.15277782280092	1.15277782266884
8	1.15277778228009	1.15277778228009	1.15277778183615
9	1.15277777822801	1.15277777822801	1.15277778535480
10	1.15277777782280	1.15277777782280	1.15277784410059

Table 5.10: Gaussian Quadrature vs Analytic $\int N^1_{8,4}N_{8,4}$

r	Exact	Gaussian Quadrature	Analytic
0	0	$-5.18001067771968e^{-19}$	0
1	0	$9.65518154811523e^{-17}$	$-2.22044604925031e^{-15}$
2	0	$-2.48796884034310e^{-17}$	$-1.33226762955019e^{-15}$
3	0	$-1.43423987543144e^{-16}$	$-4.44089209850063e^{-16}$
4	0	$2.07125912169394e^{-16}$	$-4.44089209850063e^{-16}$
5	0	$5.58278747727824e^{-17}$	$2.66453525910038e^{-15}$
6	0	$-1.99383351011882e^{-16}$	$-3.15303338993545e^{-14}$
7	0	$-1.77034855159170e^{-17}$	$8.86402062860725e^{-13}$
8	0	$-7.15922774645037e^{-17}$	$-6.18527451479167e^{-12}$
9	0	$1.13657654970488e^{-16}$	$1.33759670006839e^{-12}$
10	0	$1.07236852276289e^{-16}$	$2.68026489891327e^{-10}$

Table 5.11: Gaussian Quadrature vs Analytic $\int N_{8,5}N_{8,5}$

r	Exact	Gaussian	Analytic
0	0.393925565175565	0.393925565175565	0.393925565175565
1	0.374681473025774	0.374681473025774	0.374681473025774
2	0.372488474275695	0.372488474275695	0.372488474275694
3	0.372267326710112	0.372267326710112	0.372267326710112
4	0.372245194248786	0.372245194248786	0.372245194248785
5	0.372242980826370	0.372242980826370	0.372242980826369
6	0.372242759482366	0.372242759482367	0.372242759482365
7	0.372242737347948	0.372242737347948	0.372242737347948
8	0.372242735134506	0.372242735134507	0.372242735134508
9	0.372242734913162	0.372242734913162	0.372242734913034
10	0.372242734891028	0.372242734891028	0.372242734891082

Table 5.12: Gaussian Quadrature vs Analytic $\int N^1_{8,5}N^1_{8,5}$

r	Exact	Gaussian	Analytic
0	0.374537037037037	0.374537037037037	0.374537037037037
1	0.316973261575654	0.316973261575654	0.316973261575654
2	0.311093777296306	0.311093777296306	0.311093777296311
3	0.310507179378248	0.310507179378248	0.310507179378248
4	0.310448535237918	0.310448535237918	0.310448535237917
5	0.310442670982505	0.310442670982505	0.310442670982507
6	0.310442084558552	0.310442084558552	0.310442084558552
7	0.310442025916173	0.310442025916173	0.310442025916004
8	0.310442020051935	0.310442020051935	0.310442020051955
9	0.310442019465511	0.310442019465511	0.310442019467501
10	0.310442019406869	0.310442019406869	0.310442019400694

Table 5.13: Gaussian Quadrature vs Analytic $\int N^2_{8,5}N^2_{8,5}$

r	Exact	Gaussian	Analytic
0	1.03333333333333	1.03333333333333	1.03333333333333
1	0.761096281152580	0.761096281152580	0.761096281152579
2	0.736578488484438	0.736578488484439	0.736578488484437
3	0.734157533734545	0.734157533734546	0.734157533734544
4	0.733915750218338	0.733915750218339	0.733915750218326
5	0.733891574990279	0.733891574990279	0.733891574990252
6	0.733889157498712	0.733889157498712	0.733889157499523
7	0.733888915749868	0.733888915749869	0.733888915761653
8	0.733888891574987	0.733888891574988	0.733888891627649
9	0.733888889157499	0.733888889157499	0.733888889388507
10	0.733888888915750	0.733888888915750	0.733888890113376

Table 5.14: Gaussian Quadrature vs Analytic $\int N^1_{8,5}N_{8,5}$

r	Exact	Gaussian Quadrature	Analytic
0	0	$7.90589424200270e^{-17}$	$1.77635683940025e^{-15}$
1	0	$-2.48561945657092e^{-17}$	$8.88178419700125e^{-16}$
2	0	$-3.84438854050127e^{-17}$	$8.88178419700125e^{-16}$
3	0	$2.04158899892434e^{-17}$	0
4	0	$1.59314856120033e^{-17}$	0
5	0	$6.65712593676185e^{-17}$	$1.77635683940025e^{-15}$
6	0	$1.02374053415046e^{-16}$	0
7	0	$5.39669002320583e^{-17}$	$-6.21724893790088e^{-15}$
8	0	$-5.19608105634518e^{-17}$	$-6.48370246381091e^{-14}$
9	0	$7.44523263378936e^{-17}$	$7.27418125734403e^{-13}$
10	0	$6.15236799062335e^{-17}$	$-5.75184344597801e^{-12}$

Figure 5.46: Percentage Error vs Knot Spacing for varying Degree for $G00$ term

Figure 5.46 depicts the percentage error introduced by the Analytic approach for the $G00$ term. Note that as the scale is logarithmic, 0 values cannot be represented. Therefore the plot shows only the non-zero error. As can be seen for the cubic case the percentage error incurred ranges from 0% to $10^{-4}\%$ over the range of knot spacing from regular to extreme, respectively. The magnitude of the error reduces with increasing degree. The figure also demonstrates the stability of the approach.

Figure 5.47 depicts the percentage error introduced by the Analytic approach for the $G00$, $G11$, and $G22$ terms. As can be seen from the figure, the error incurred increases slightly for the terms involving the derivatives of the basis functions. The figure also shows that for the cubic case the percentage error incurred by all terms is captured by the

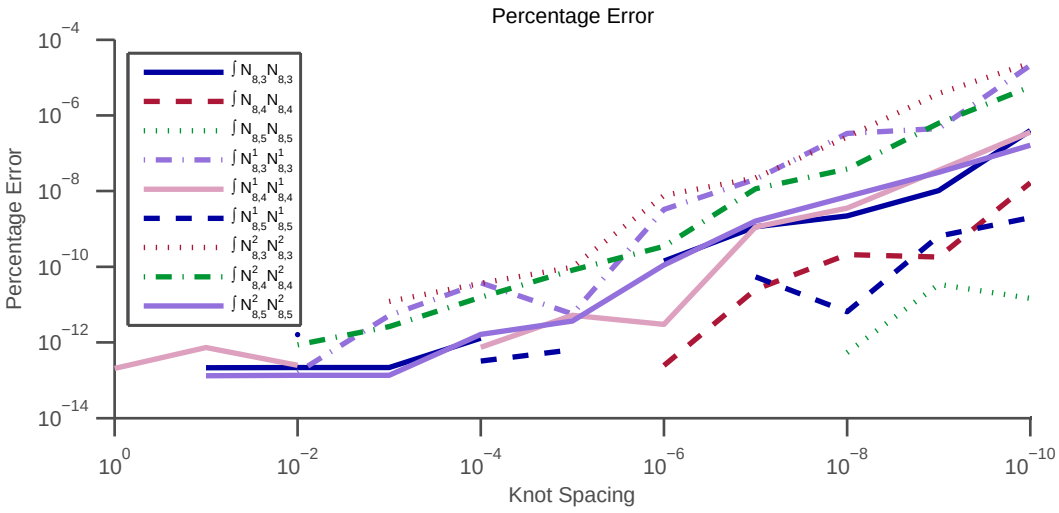


Figure 5.47: Percentage Error vs Knot Spacing for varying Degree for $G00$, $G11$, and $G22$ terms

range 0% to $10^{-4}\%$ over the range of knot spacing from regular to extreme respectively and in each case the magnitude of the error reduces with increasing degree.

While the Gaussian Quadrature approach exhibits some rounding errors in the least significant digit, the approach does not accumulate error as the ratios of knot spacings become more extreme. For this reason error plots are of little value and have been omitted.

Error Analysis

The results show that the Gaussian Quadrature approach presented is slightly more accurate than the Analytic Approach. The mathematical approach adopted is exact, in a sense that a perfect mathematical description of the result is available at all times. Therefore, the error is a result of rounding error, introduced during computation as a result of the floating-point arithmetic operations. The practical implications of this error will be discussed in Section 5.10. This section investigates the root cause of the error introduced.

It is well documented that the Cox-deBoor recursion (see Equation 4.2) used to calculate the basis functions for both the Gaussian Quadrature approach and the Analytic Approach is numerically stable and highly accurate (Deboor, 1978, Piegl and Tiller, 1997, Farin, 2002). The basis function and derivative algorithms employed in this thesis are largely based on the algorithms presented in (Piegl and Tiller, 1997) that are directly derived from the Cox-deBoor recursion. Piegl and Tiller (1997) do not comment on the stability or accuracy of their algorithms. As part of the research described in this thesis, the algorithms were tested and proved (with the exception of rounding error in the least significant digit) to match symbolic results to machine precision for each of the basis function and derivative evaluations performed for the cases presented in Tables 5.3–5.14, inclusive.

In this chapter, formulae and algorithms for the computation of the integral and repeated integrals are derived and developed directly from the Cox-deBoor recursion. These algorithms were tested and proven to match symbolic results to machine precision for each of the integrals required for the tests presented in Tables 5.3–5.14, inclusive. Table 5.15 exemplifies the results and shows the fourth integral of the cubic curve, as required by the Analytic approach, at the two extremes of the knot spacing tested, namely $r = 0$ and $r = 10$. As can be seen, the results shown match those obtained symbolically to machine precision.

While the derivative and integral algorithms are more prone to round off errors than the basis function algorithm, due to the higher ratios of orders of magnitudes involved in the calculations, in the experiments performed in this thesis the error incurred did not propagate beyond the least significant digit. As each of the component algorithms employed in the computation of the Gramian matrices individually produce results

Table 5.15: $\int\int\int\int N_{8,3}$ over knots $\{5, 6, 6+10^{-r}, 8, 9\}$

	Knot	Exact	Analytic Algorithm
$r = 0$	6	$1.98412698412698e^{-4}$	$1.98412698412698e^{-4}$
	$6 + 10^{-r}$	$2.46031746031746e^{-2}$	$2.46031746031746e^{-2}$
	8	$3.33531746031746e^{-1}$	$3.33531746031746e^{-1}$
	9	1.66666666666667	1.66666666666667
$r = 10$	6	$3.96825396785714e^{-4}$	$3.96825396785714e^{-4}$
	$6 + 10^{-r}$	$3.96825397063492e^{-4}$	$3.96825397063492e^{-4}$
	8	$4.95370370349899e^{-1}$	$4.95370370349899e^{-1}$
	9	2.16190476184762	2.16190476184762

accurate to machine precision, it is concluded that the assembly of the terms into the Gramian matrices is what results in some loss of accuracy as the knot spacing ratios become extreme.

The key component in assembling the Gramian matrices, using the Analytic approach, is the summation of integral values sampled at the knots, and scaled by the corresponding Dirac Delta weight. Adopting the same testing approach as the preceding section, a knot vector of type $U = \{\dots 5, 6, 6+10^{-r}, 8, 9, \dots\}$ is considered. Figure 5.48 illustrates the effect of varying ratios of knot spacing on the affected values.

Firstly, the values of the Dirac Delta weights are considered. Increasing the value of r results in an anomalous knot span, whereby the ratio of its size in relation to its neighbours becomes extreme. As a result, the derivative values over the affected span become very large relative to those of neighbouring spans.

The calculation of the weight associated with the Dirac delta function relies on the differencing of the p^{th} derivatives of the affected span with those of its two neighbouring spans. The large derivative introduced by the anomalous knot dominates the difference in both cases as r increases in value. This results in the Dirac Delta Weights at either side of the span being very large and very similar in magnitude.

Figure 5.48(a) shows the absolute value of the Dirac Delta Weights at the knots left and right of the affected span for curves of degrees 3, 6, and 10. The figure shows the increase in magnitude of the Dirac weight with decreasing knot spacing. More importantly, the figure also shows that the magnitudes, as dominated by the same anomalous knot span, are all but indistinguishable. Note that the direction and thus the sign of the Dirac delta alternates from one side to the other. However, the absolute value is shown here to illustrate the similarity in magnitudes. As can be seen in the figure, the plots for the magnitude of the Dirac Delta weight at either side of the affected span are overlayed.

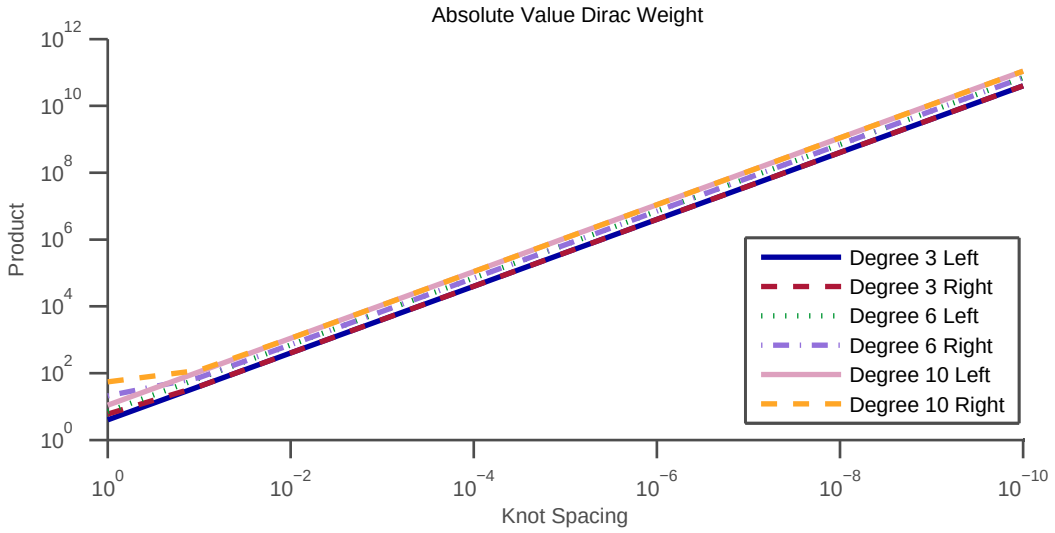
The magnitudes of the integral values, as shown in Figure 5.48(b), decrease rapidly

as the knot spacing decreases, as could be expected. As the knot span decreases in size, so too does the area under the curve segment supported by the span. This is especially the case for the higher order curves whose area is spread out over an increased number of knot spans. Another result of this, is that as the knot span decreases in size, the integral values at the knots on either side of the anomalous span become more and more similar.

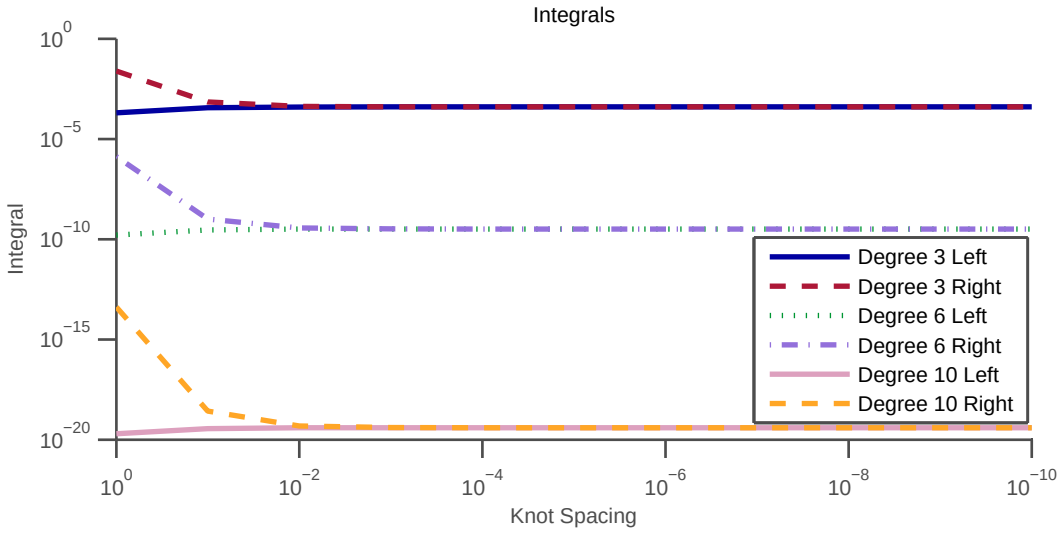
Finally, the products of the Dirac Delta weight and integral values at the knots are presented in Figure 5.48(c). As it is the case for both the integrals and Dirac Delta weights that their magnitudes are similar at either side of the anomalous knot, so too are their products. More importantly, the Dirac Delta weights at either side are opposite in sign. To compute the overall term for the Gramian matrices, the values must be summed. Because the magnitudes of the product values on either side of the affected knot span are so similar, the resulting difference between them leads to a cancellation of significant digits. This gives consequence to the otherwise insignificant roundoff errors stored in the least significant digits of the product.

To exacerbate the problem, as the magnitude of the Dirac Delta weights is so large relative to those at the neighbouring knots, the least significant digits of the products at either side of the affected span can be of similar order to the most significant digits of the products at the neighbouring knots. This means that the overall contribution of these error-prone digits to the overall answer grows in size with decreasing knot span.

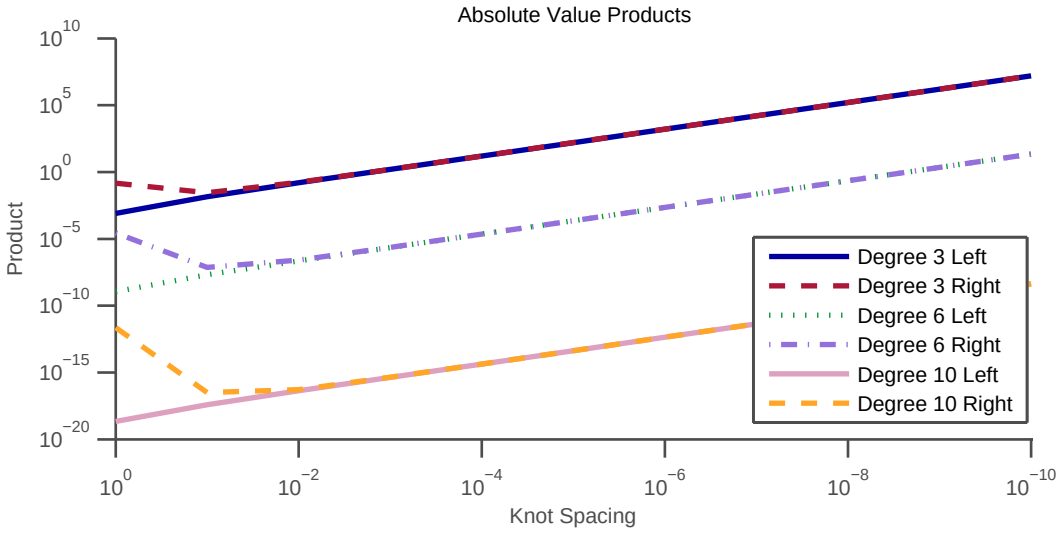
Fortunately, the integral values are so small over the affected span, that they act to offset the significance of the error incurred. Figure 5.48(b) shows how the integral values decrease with decreasing knot spacing. It is also evident from the graph that the integral values decrease rapidly with increasing order of curve, further offsetting the impact of the inaccuracy.



(a) Absolute Value of Dirac Delta Weights at either side of the modified knot span



(b) Integral Trend at either side of the modified knot span



(c) Absolute Value of the Product at either side of the modified knot span

Figure 5.48: Graphic illustration of error introduced in Analytic Approach

To further illustrate the effect of the integral mitigation, Table 5.16 shows the relevant numerical values. The problematic values are highlighted. It is clear from the table that for the degree 10 curve, the integral is so small that it offsets the values of the derivatives to such an extent that the erroneous values produced by the differencing of similar terms are reduced to insignificance in the final answer.

While the Gaussian Quadrature approach suffers from the same numerical rounding and precision errors associated with floating-point arithmetic, the error is not as prevalent. This is because the approach does not incur the differencing effect described above as a result of the anomalous knot. For the Gaussian Quadrature approach, the summations include several values over a span that are of the same sign and similar orders of magnitude, and as such they do not give consequence to erroneous data present in digits of lower significance.

Further information on the difficulties that can be incurred when dealing with floating-point arithmetic can be found in (Goldberg, 1991).

Error Mitigation

There are numerical techniques for handling summations and differences that can mitigate the impact of numerical error (Press et al., 2007, Goldberg, 1991). Such techniques usually impose a trade-off between computational expense and accuracy, e.g., there are various number formats available in the different programming languages that can accommodate higher precision. However, use of such techniques may impact on the speed of the computations. There are also several algorithms that act to actively compensate for numerical error as it is introduced, e.g., the Kahan Summation Algorithm (Goldberg, 2001).

A common practice in industry to alleviate the impact of floating point arithmetic errors on numerical accuracy is to normalise the knot vector to the range $[0, 1]$. Because of the higher density of floating point numbers on this interval, greater accuracy can often be achieved. Tables 5.17–5.20, inclusive, show results for the cubic case presented over a normalised knot vector.

Table 5.16: Analytic Approach terms for $N_{15,10}$ over its support knots $\{5, 6, 6+10^{-r}, 8, 9, 10, 11, 12, 13, 14, 15, 16\}$ including 11^{th} integral, 11^{th} derivative

Knot	Integral	Derivative	Dirac Delta Weight	Product
6	$3.91458821228679e^{-20}$	1.99999999980000	$-1.09999990898560e^{11}$	$-4.30604667723157e^{-9}$
$6+10^{-r}$	$3.91458822050743e^{-20}$	$-1.09999990896560e^{11}$	$1.09999990908779e^{11}$	$4.30604668667429e^{-9}$
8	$4.05158224375449e^{-10}$	$1.22186507947778e^{1}$	$-8.25000000041250e^{1}$	$-3.34255535126458e^{-8}$
9	$1.58492605091671e^{-7}$	$-7.02813492093472e^{1}$	$2.20000000007333e^{2}$	$3.48683731213299e^{-5}$
10	$1.45745806819173e^{-5}$	$1.49718650797986e^{2}$	$-3.46500000008663e^{2}$	$-5.05009220641060e^{-3}$
11	$5.22906670382717e^{-4}$	$-1.96781349210676e^{2}$	$3.69600000007392e^{2}$	$1.93266305377318e^{-1}$
12	$1.58492605091671e^{-7}$	$1.72818650796716e^{2}$	$-2.75000000004583e^{2}$	-2.67149649187859
13	$1.11251433921923e^{-1}$	$-1.02181349207868e^{2}$	$1.41428571430592e^{2}$	$1.57341313691834e^{1}$
14	$8.81900244082422e^{-1}$	$3.92472222227242e^{1}$	$-4.81250000006016e^{1}$	$-4.24414492469971e^{1}$
15	5.25095450579084	-8.87777777787742	9.77777777788642	$5.13426662794143e^{1}$
16	$2.49295942234626e^{1}$	$9.00000000009000e^{1}$	$-9.00000000009000e^{-1}$	$-2.24366348013407e^{1}$

Table 5.17: Gaussian Quadrature vs Analytic Normalised $\int N_{8,3}N_{8,3}$

r	Exact	Gaussian	Analytic
0	0.0368742368742369	0.0368742368742369	0.0368742368742369
1	0.0357494306459824	0.0357494306459824	0.0357494306459824
2	0.0355174474164679	0.0355174474164679	0.0355174474164679
3	0.0354931476142845	0.0354931476142845	0.0354931476142844
4	0.0354907068119204	0.0354907068119204	0.0354907068119201
5	0.0354904626236748	0.0354904626236748	0.0354904626236755
6	0.0354904382037703	0.0354904382037703	0.0354904382038676
7	0.0354904357617691	0.0354904357617691	0.0354904357621220
8	0.0354904355175689	0.0354904355175689	0.0354904355153877
9	0.0354904354931488	0.0354904354931488	0.0354904355096545
10	0.0354904354907068	0.0354904354907068	0.0354904358288595

Table 5.18: Gaussian Quadrature vs Analytic Normalised $\int N^1_{8,3}N^1_{8,3}$

r	Exact	Gaussian	Analytic
0	8.66666666666667	8.66666666666667	8.66666666666667
1	8.22654127481714	8.22654127481713	8.22654127481713
2	8.10411823791075	8.10411823791075	8.10411823791077
3	8.09042783380896	8.09042783380897	8.09042783380919
4	8.08904294498937	8.08904294498937	8.08904294499017
5	8.08890429611655	8.08890429611655	8.08890429612064
6	8.08889042962783	8.08889042962783	8.08889042899957
7	8.08888904296295	8.08888904296294	8.08888904292756
8	8.08888890429630	8.08888890429630	8.08888891442555
9	8.08888889042963	8.08888889042963	8.08888863782479
10	8.08888888904296	8.08888888904297	8.08888541386564

Table 5.19: Gaussian Quadrature vs Analytic Normalised $\int N^2_{8,3}N^2_{8,3}$

r	Exact	Gaussian	Analytic
0	5858.66666666667	5858.66666666667	5858.66666666667
1	7346.29049111808	7346.29049111808	7346.29049111808
2	7760.08035586167	7760.08035586168	7760.08035586168
3	7806.35392172571	7806.35392172572	7806.35392172479
4	7811.03484593594	7811.03484593595	7811.03484593884
5	7811.50347912608	7811.50347912609	7811.50347917864
6	7811.55034785793	7811.55034785793	7811.55034923149
7	7811.55503478525	7811.55503478525	7811.55503506622
8	7811.55550347852	7811.55550347853	7811.55542263504
9	7811.55555034785	7811.55555034785	7811.55609768343
10	7811.55555503479	7811.55555503480	7811.55512148709

Table 5.20: Gaussian Quadrature vs Analytic Normalised $\int N^1_{8,3}N_{8,3}$

r	Exact	Gaussian Quadrature	Analytic
0	0	$-1.39314000302920e^{-16}$	$1.33226762955019e^{-15}$
1	0	$-4.09373053309227e^{-16}$	$2.22044604925031e^{-16}$
2	0	$-3.83620229194738e^{-18}$	$-4.44089209850063e^{-16}$
3	0	$3.65276076360721e^{-17}$	$3.77475828372553e^{-15}$
4	0	$4.31740290042263e^{-16}$	$5.39568389967826e^{-14}$
5	0	$-1.12755548593918e^{-16}$	$5.12923037376822e^{-14}$
6	0	$3.43878953741364e^{-16}$	$-7.66320340517268e^{-12}$
7	0	$-1.45355896241852e^{-16}$	$-8.66040572589100e^{-12}$
8	0	$-5.16442909788313e^{-16}$	$2.68828959093526e^{-10}$
9	0	$-2.24605355923787e^{-16}$	$-2.24709606477802e^{-9}$
10	0	$7.88449567818682e^{-17}$	$-5.07644815073149e^{-8}$

The results show an improvement in the performance of the Analytic approach, which for the first term $\int N_{8,3}N_{8,3}$ now matches Gaussian Quadrature in precision up to 15 significant digits for the first four powers.

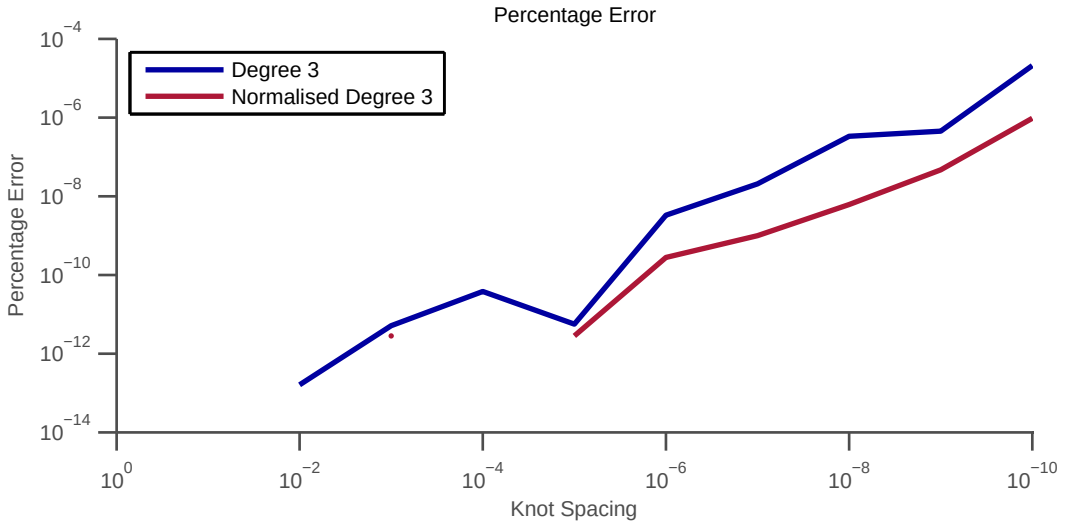


Figure 5.49: Normalised Error

Figure 5.49 shows the improvement in the percentage error introduced when the knot vector is normalised. The normalised curve begins to accumulate error only for knot spacings in the region of 10^{-5} and above. The percentage error also shows a reduction from a maximum error of $10^{-4}\%$ to $10^{-6}\%$.

Further information on mitigating the difficulties that can be incurred when dealing with floating-point arithmetic can be found in (Goldberg, 1991).

Gramian Matrices

The results discussed so far have been primarily to stress test the Analytic approach by pushing it to extreme conditions unlikely to occur in practice. Figures 5.50–5.57, inclusive, show screen grabs of each of the Gramian matrices generated by the Java implementation of the algorithms discussed for the more usual scenario of cubic curves with even knot spacing [6]. As the results show, with the exception of occasional variances in the least significant digits (a numerical effect that affects both the Gaussian Quadrature and the Analytic approaches presented), the results match in each case.

[6] For each figure "gram" stands for Gramian Matrix, labeled according to their respective derivative components as follows $G00$, $G11$, $G22$, and $G10$

Analytic gram00

	0	1	2	3
0	1.42857142857142e-01	8.75000000000000e-02	1.84523809523809e-02	1.19047619047619e-03
1	2.21428571428571e-01	1.56249999999999e-01	3.45238095238095e-02	2.97619047619047e-04
2	3.26785714285714e-01	2.24603174603174e-01	2.37103174603174e-02	1.98412698412698e-04
3	4.79365079365079e-01	2.36309523809523e-01	2.38095238095238e-02	1.98412698412698e-04
4	4.79365079365079e-01	2.36309523809523e-01	2.38095238095238e-02	1.98412698412698e-04
5	4.79365079365079e-01	2.36309523809523e-01	2.38095238095238e-02	1.98412698412698e-04
6	4.79365079365079e-01	2.36309523809523e-01	2.38095238095238e-02	1.98412698412698e-04
7	4.79365079365079e-01	2.36309523809523e-01	2.38095238095238e-02	1.98412698412698e-04
8	4.79365079365079e-01	2.36309523809523e-01	2.38095238095238e-02	1.98412698412698e-04
9	4.79365079365079e-01	2.36309523809523e-01	2.38095238095238e-02	1.98412698412698e-04
10	4.79365079365079e-01	2.36309523809523e-01	2.38095238095238e-02	1.98412698412698e-04
11	4.79365079365079e-01	2.36309523809523e-01	2.37103174603174e-02	2.97619047619047e-04
12	4.79365079365079e-01	2.24603174603174e-01	3.45238095238095e-02	1.19047619047619e-03
13	3.26785714285714e-01	1.56250000000000e-01	1.84523809523809e-02	0.00000000000000e+00
14	2.21428571428571e-01	8.75000000000000e-02	0.00000000000000e+00	0.00000000000000e+00
15	1.42857142857142e-01	0.00000000000000e+00	0.00000000000000e+00	0.00000000000000e+00

Figure 5.50: Analytic $G00$

Gaussian gram00

	0	1	2	3
0	1.42857142857142e-01	8.75000000000000e-02	1.84523809523809e-02	1.19047619047619e-03
1	2.21428571428571e-01	1.56250000000000e-01	3.45238095238095e-02	2.97619047619047e-04
2	3.26785714285714e-01	2.24603174603174e-01	2.37103174603174e-02	1.98412698412698e-04
3	4.79365079365079e-01	2.36309523809523e-01	2.38095238095238e-02	1.98412698412698e-04
4	4.79365079365079e-01	2.36309523809523e-01	2.38095238095238e-02	1.98412698412698e-04
5	4.79365079365079e-01	2.36309523809523e-01	2.38095238095238e-02	1.98412698412698e-04
6	4.79365079365079e-01	2.36309523809523e-01	2.38095238095238e-02	1.98412698412698e-04
7	4.79365079365079e-01	2.36309523809523e-01	2.38095238095238e-02	1.98412698412698e-04
8	4.79365079365079e-01	2.36309523809523e-01	2.38095238095238e-02	1.98412698412698e-04
9	4.79365079365079e-01	2.36309523809523e-01	2.38095238095238e-02	1.98412698412698e-04
10	4.79365079365079e-01	2.36309523809523e-01	2.38095238095238e-02	1.98412698412698e-04
11	4.79365079365079e-01	2.36309523809523e-01	2.37103174603174e-02	2.97619047619047e-04
12	4.79365079365079e-01	2.24603174603174e-01	3.45238095238095e-02	1.19047619047619e-03
13	3.26785714285714e-01	1.56250000000000e-01	1.84523809523809e-02	0.00000000000000e+00
14	2.21428571428571e-01	8.75000000000000e-02	0.00000000000000e+00	0.00000000000000e+00
15	1.42857142857142e-01	0.00000000000000e+00	0.00000000000000e+00	0.00000000000000e+00

Figure 5.51: Gaussian $G00$

Analytic gram11

	0	1	2	3
0	1.80000000000000e+00	-1.27500000000000e+00	-4.75000000000000e-01	-5.00000000000000e-02
1	1.50000000000000e+00	3.75000000000000e-02	.50000000000000e-01	-1.25000000000000e-02
2	6.75000000000000e-01	-3.33333333333334e-02	-1.95833333333333e-01	-8.33333333333333e-03
3	6.66666666666667e-01	-1.25000000000000e-01	-2.00000000000000e-01	-8.33333333333333e-03
4	6.66666666666667e-01	-1.25000000000000e-01	-2.00000000000000e-01	-8.33333333333333e-03
5	6.66666666666667e-01	-1.25000000000000e-01	-2.00000000000000e-01	-8.33333333333333e-03
6	6.66666666666667e-01	-1.25000000000000e-01	-2.00000000000000e-01	-8.33333333333333e-03
7	6.66666666666667e-01	-1.25000000000000e-01	-2.00000000000000e-01	-8.33333333333333e-03
8	6.66666666666667e-01	-1.25000000000000e-01	-2.00000000000000e-01	-8.33333333333333e-03
9	6.66666666666667e-01	-1.25000000000000e-01	-2.00000000000000e-01	-8.33333333333333e-03
10	6.66666666666667e-01	-1.25000000000000e-01	-2.00000000000000e-01	-8.33333333333334e-03
11	6.66666666666667e-01	-1.25000000000000e-01	-1.95833333333333e-01	-1.25000000000000e-02
12	6.66666666666667e-01	-3.33333333333334e-02	-2.50000000000000e-01	-5.00000000000000e-02
13	6.75000000000001e-01	3.75000000000003e-02	-4.75000000000000e-013	0.00000000000000e+00
14	1.50000000000000e+00	-1.27500000000000e+00	0.00000000000000e+00	0.00000000000000e+00
15	1.80000000000000e+00	0.00000000000000e+00	0.00000000000000e+00	0.00000000000000e+00

Figure 5.52: Analytic $G11$

Gaussian gram11

	0	1	2	3
0	1.80000000000000e+00	-1.27500000000000e+00	-4.75000000000000e-01	-5.00000000000000e-02
1	1.50000000000000e+00	3.75000000000000e-02	-2.50000000000000e-01	-1.25000000000000e-02
2	6.75000000000000e-01	-3.33333333333334e-02	-1.95833333333333e-01	-8.33333333333333e-03
3	6.66666666666667e-01	-1.25000000000000e-01	-2.00000000000000e-01	-8.33333333333333e-03
4	6.66666666666667e-01	-1.25000000000000e-01	-2.00000000000000e-01	-8.33333333333334e-03
5	6.66666666666667e-01	-1.25000000000000e-01	-2.00000000000000e-01	-8.33333333333334e-03
6	6.66666666666667e-01	-1.25000000000000e-01	-2.00000000000000e-01	-8.33333333333334e-03
7	6.66666666666667e-01	-1.25000000000000e-01	-2.00000000000000e-01	-8.33333333333334e-03
8	6.66666666666667e-01	-1.25000000000000e-01	-2.00000000000000e-01	-8.33333333333334e-03
9	6.66666666666667e-01	-1.25000000000000e-01	-2.00000000000000e-01	-8.33333333333334e-03
10	6.66666666666667e-01	-1.25000000000000e-01	-2.00000000000000e-01	-8.33333333333334e-03
11	6.66666666666667e-01	-1.25000000000000e-01	-1.95833333333333e-01	-1.25000000000000e-02
12	6.66666666666667e-01	-3.33333333333334e-02	-2.50000000000000e-01	-5.00000000000000e-02
13	6.75000000000000e-01	3.75000000000003e-02	-4.75000000000000e-01	0.00000000000000e+00
14	1.50000000000000e+00	-1.27500000000000e+00	0.00000000000000e+00	0.00000000000000e+00
15	1.80000000000000e+00	0.00000000000000e+00	0.00000000000000e+00	0.00000000000000e+00

Figure 5.53: Gaussian $G11$

Analytic gram22

	0	1	2	3
0	1.20000000000000e+01	-1.65000000000000e+01	3.50000000000000e+00	1.00000000000000e+00
1	2.40000000000000e+01	-6.75000000000000e+00	-1.00000000000000e+00	2.50000000000000e-01
2	4.50000000000000e+00	-1.33333333333333e+00	-8.33333333333334e-02	1.66666666666667e-01
3	2.66666666666667e+00	-1.50000000000000e+00	0.00000000000000e+00	1.66666666666667e-01
4	2.66666666666667e+00	-1.50000000000000e+00	0.00000000000000e+00	1.66666666666667e-01
5	2.66666666666667e+00	-1.50000000000000e+00	0.00000000000000e+00	1.66666666666667e-01
6	2.66666666666667e+00	-1.50000000000000e+00	0.00000000000000e+00	1.66666666666667e-01
7	2.66666666666667e+00	-1.50000000000000e+00	0.00000000000000e+00	1.66666666666667e-01
8	2.66666666666667e+00	-1.50000000000000e+00	0.00000000000000e+00	1.66666666666667e-01
9	2.66666666666667e+00	-1.50000000000000e+00	0.00000000000000e+00	1.66666666666667e-01
10	2.66666666666667e+00	-1.50000000000000e+00	0.00000000000000e+00	1.66666666666667e-01
11	2.66666666666667e+00	-1.50000000000000e+00	-8.33333333333334e-02	2.50000000000000e-01
12	2.66666666666667e+00	-1.33333333333333e+00	-1.00000000000000e+00	1.00000000000000e+00
13	4.50000000000000e+00	-6.75000000000000e+00	3.50000000000000e+00	0.00000000000000e+00
14	2.40000000000000e+01	-1.65000000000000e+01	0.00000000000000e+00	0.00000000000000e+00
15	1.20000000000000e+01	0.00000000000000e+00	0.00000000000000e+00	0.00000000000000e+00

Figure 5.54: Analytic $G22$

Gaussian gram22

	0	1	2	3
0	1.20000000000000e+01	-1.65000000000000e+01	3.50000000000000e+00	1.00000000000000e+00
1	2.40000000000000e+01	-6.75000000000000e+00	-1.00000000000000e+00	2.50000000000000e-01
2	4.50000000000000e+00	-1.33333333333333e+00	-8.33333333333333e-02	1.66666666666667e-01
3	2.66666666666667e+00	-1.50000000000000e+00	2.35922392732846e-16	1.66666666666667e-01
4	2.66666666666667e+00	-1.50000000000000e+00	-1.63064006741820e-16	1.66666666666667e-01
5	2.66666666666667e+00	-1.50000000000000e+00	-5.51642065360625e-16	1.66666666666667e-01
6	2.66666666666667e+00	-1.50000000000000e+00	-5.51642065360625e-16	1.66666666666667e-01
7	2.66666666666667e+00	-1.50000000000000e+00	-5.51642065360625e-16	1.66666666666667e-01
8	2.66666666666667e+00	-1.50000000000000e+00	-5.51642065360625e-16	1.66666666666667e-01
9	2.66666666666667e+00	-1.50000000000000e+00	-5.51642065360625e-16	1.66666666666667e-01
10	2.66666666666667e+00	-1.50000000000000e+00	-5.51642065360625e-16	1.66666666666667e-01
11	2.66666666666667e+00	-1.50000000000000e+00	-8.33333333333340e-02	2.50000000000000e-01
12	2.66666666666667e+00	-1.33333333333333e+00	-1.00000000000000e+00	1.00000000000000e+00
13	4.50000000000000e+00	-6.75000000000000e+00	3.50000000000000e+00	0.00000000000000e+00
14	2.40000000000000e+01	-1.65000000000000e+01	0.00000000000000e+00	0.00000000000000e+00
15	1.20000000000000e+01	0.00000000000000e+00	0.00000000000000e+00	0.00000000000000e+00

Figure 5.55: Gaussian $G22$

Analytic gram10

	0	1	2	3
0	-5.00000000000000e-01	-3.87500000000000e-01	-1.04166666666667e-01	-8.33333333333333e-03
1	2.22044604925031e-16	-2.77083333333333e-01	-1.08333333333333e-01	-2.08333333333333e-03
2	0.00000000000000e+00	-3.02777777777778e-01	-7.70833333333333e-02	-1.38888888888889e-03
3	0.00000000000000e+00	-3.40277777777778e-01	-7.77777777777778e-02	-1.38888888888889e-03
4	0.00000000000000e+00	-3.40277777777778e-01	-7.77777777777778e-02	-1.38888888888889e-03
5	0.00000000000000e+00	-3.40277777777778e-01	-7.77777777777778e-02	-1.38888888888889e-03
6	0.00000000000000e+00	-3.40277777777778e-01	-7.77777777777778e-02	-1.38888888888889e-03
7	0.00000000000000e+00	-3.40277777777778e-01	-7.77777777777778e-02	-1.38888888888889e-03
8	0.00000000000000e+00	-3.40277777777778e-01	-7.77777777777778e-02	-1.38888888888889e-03
9	0.00000000000000e+00	-3.40277777777778e-01	-7.77777777777778e-02	-1.38888888888889e-03
10	0.00000000000000e+00	-3.40277777777778e-01	-7.77777777777778e-02	-1.38888888888889e-03
11	0.00000000000000e+00	-3.40277777777778e-01	-7.70833333333333e-02	-2.08333333333333e-03
12	4.44089209850063e-16	-3.02777777777778e-01	-1.08333333333333e-01	-8.33333333333333e-03
13	0.00000000000000e+00	-2.77083333333333e-01	-1.04166666666667e-01	0.00000000000000e+00
14	0.00000000000000e+00	-3.87500000000000e-01	0.00000000000000e+00	0.00000000000000e+00
15	5.00000000000000e-01	0.00000000000000e+00	0.00000000000000e+00	0.00000000000000e+00

Figure 5.56: Analytic $G10$

Gaussian gram10

	0	1	2	3
0	-5.00000000000000e-01	-3.87500000000000e-01	-1.04166666666667e-01	-8.33333333333334e-03
1	7.00430072930302e-19	-2.77083333333333e-01	-1.08333333333333e-01	-2.08333333333333e-03
2	3.01603716360553e-17	-3.02777777777778e-01	-7.70833333333334e-02	-1.38888888888889e-03
3	5.09770533477770e-17	-3.40277777777778e-01	-7.77777777777778e-02	-1.38888888888889e-03
4	1.62818228220773e-17	-3.40277777777778e-01	-7.77777777777778e-02	-1.38888888888889e-03
5	-7.73932448806703e-17	-3.40277777777778e-01	-7.77777777777778e-02	-1.38888888888889e-03
6	5.87348196621648e-18	-3.40277777777778e-01	-7.77777777777778e-02	-1.38888888888889e-03
7	5.87348196621648e-18	-3.40277777777778e-01	-7.77777777777778e-02	-1.38888888888889e-03
8	5.87348196621648e-18	-3.40277777777778e-01	-7.77777777777778e-02	-1.38888888888889e-03
9	5.87348196621648e-18	-3.40277777777778e-01	-7.77777777777778e-02	-1.38888888888889e-03
10	5.87348196621648e-18	-3.40277777777778e-01	-7.77777777777778e-02	-1.38888888888889e-03
11	5.87348196621648e-18	-3.40277777777778e-01	-7.70833333333333e-02	-2.08333333333333e-03
12	5.87348196621648e-18	-3.02777777777778e-01	-1.08333333333333e-01	-8.33333333333334e-03
13	4.33680868994202e-19	-2.77083333333333e-01	-1.04166666666667e-01	0.00000000000000e+00
14	-2.49800180540660e-16	-3.87499999999999e-01	0.00000000000000e+00	0.00000000000000e+00
15	5.00000000000000e-01	0.00000000000000e+00	0.00000000000000e+00	0.00000000000000e+00

Figure 5.57: Gaussian $G10$

5.9.3 Discussion

The Analytic approach presented in this chapter reduces the computational complexity dramatically such that it is more than 3 times faster than the Gaussian Quadrature approach for the most practical cases discussed and achieves very high levels of accuracy. Additionally, both the accuracy, computational complexity, and performance times improve with increasing degree. This is of great importance for interactive applications such as prototyping and Virtual Sculpting but also for processing models that consist of large numbers of patches.

While the Gaussian Quadrature computation is slightly more accurate than the Analytic computation presented, for equally spaced knots, it can be seen that the Analytic approach performs to the same degree of precision as Gaussian Quadrature up to 15 significant digits. For reasonably spaced knots, the approach continues to perform very well and provides accuracy up to eight decimal places for even the most extreme case considered. Additionally, the accuracy of the approach improves with increasing degree and, as demonstrated in the previous section, for a degree 10 curve the error is all but eliminated.

It is important to note that the numerical differencing problem described above only becomes an issue where a very small knot spacing occurs between two larger ones such that the derivative over the anomalous knot span dominates the associated difference terms on both sides. Practical cases typically involve cubic curves with reasonably regular knot vectors. Where anomalous knots do occur, they are usually in the form of repeated knots to manipulate curve continuity, or several knots are drawn closer together to achieve directional effects in a curve.

Equally, it is important to note that while the error of the Analytic approach is all but removed for curves of high degree, in practice, cubic curves are the most commonly used curves. Higher order curves (degree 5 or greater) are used in industry for the design of highly smooth aesthetic surfaces such as those used for car body design, but cubic curves are the most prevalent for general design purposes. Autodesk's Maya supports curves up to degree 7 (AutoDesk, 2012) but notes that the inclusion of the higher degrees is predominantly to facilitate data exchange with other CAD packages.

Tolerance levels in industry, for the machine fabrication of products, are typically of the order of 0.0001mm-1mm and as models are typically composed of multiple patches, the error incurred using the Analytic approach would typically fall well inside these tolerances. Where greater accuracy is required for the most extreme cases, there are several numerical techniques for handling summations and differences that can reduce the impact of numerical error (Press et al., 2007, Goldberg, 1991). Such techniques range from ones of relatively small additional computational expense, e.g., normalisation to those of more significant computational expense, e.g., higher precision data. There are various number formats available in the different programming languages that can accommodate higher precision. However, use of such techniques may impact on the speed of the computations. There are also several algorithms that act to actively com-

pensate for numerical error as it is introduced, e.g., the Kahan Summation Algorithm (Goldberg, 2001).

In Section 5.3.1, it was noted that recent research by Hughes et al. (2010) has presented reduced quadrature rules for Gaussian Quadrature that take advantage of a-priori knowledge of the continuity of a B-Spline/NURBS curve across each knot. The impact of the approach improves with increasing numbers of control points. In respect of the cases presented in this chapter, for the best case scenario using the reduced quadrature rules, as the number of points tends towards infinity the number of sample points required tends asymptotically towards half those required using the standard Gaussian Quadrature approach presented in this chapter. It is likely that this approach will replace existing Gaussian Quadrature rules as the state of the art Gaussian Quadrature approach in time. However, the Analytic approach presented in this chapter still outperforms the approach of Hughes et al. (2010) on computational complexity. Additionally, no assumptions are made about the continuity of the Active Surface in the proposed Analytic approach.

5.10 Conclusions

In this chapter a summary of existing techniques for the construction and assembly of the stiffness matrix associated with an Active B-spline Surface, is presented, identifying Gaussian Quadrature as the state of the art approach. A gap in the literature is identified with regard to implementation details of the technique for the specific problem of constructing the stiffness matrix of an Active B-Spline Surface. Efficiencies that arise as a result of the specific problem domain are identified and an efficient tailored Gaussian Quadrature algorithm is then developed capitalising on further efficiencies specific to a Gaussian Quadrature solution of the B-Spline problem.

A more direct Analytic approach is then considered with the aim of achieving greater efficiency, resulting in a novel approach and efficient algorithm that is shown both theoretically and experimentally to significantly outperform the Gaussian Quadrature approach in terms of efficiency, while preserving high levels of accuracy and stability. The performance gain is mainly due to the mathematical efficiency introduced whereby only a single sample per span is required, regardless of the degree of the curve. Additionally, unlike other existing approaches, no assumptions about the regularity of the active surface are made in order to achieve the savings. Algorithmic efficiencies are also employed to optimise the necessary number of calculations. The technique is shown to be equally applicable to cases where the α and β parameters of the model are more complex functions of both u and v. An additional benefit of the approach is that it also facilitates an analytic evaluation of the mass, damping, and forcing functions.

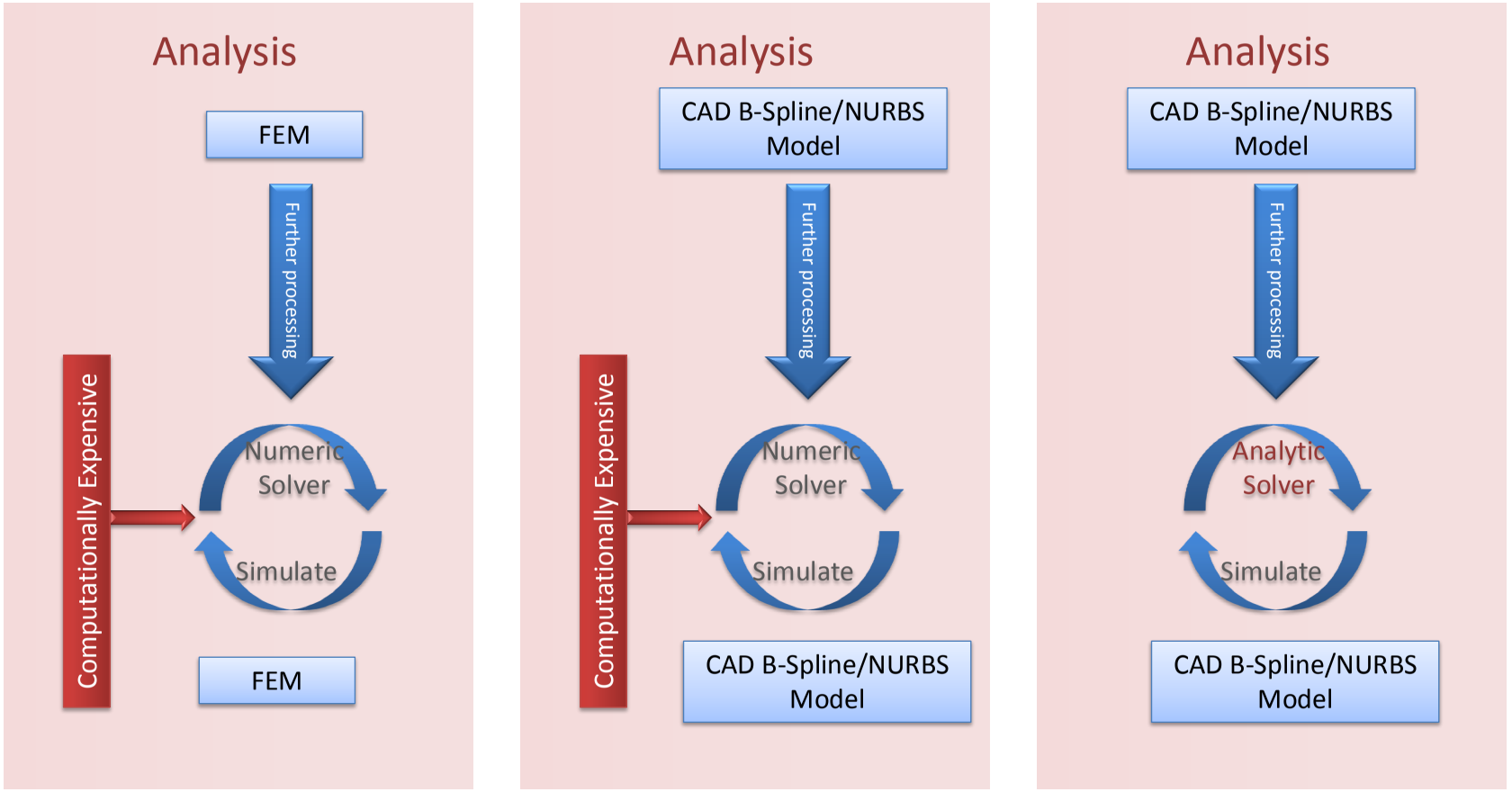
An extension to the important case of NURBS is presented and while a direct analytic solution is as yet unavailable for NURBS, it is demonstrated how the Analytic approach for a non-rational B-Spline can be modified to approximate a solution.

As part of the development of the proposed Analytic approach, theory relating to the integral and repeated integral of a B-Spline basis function is derived, and efficient algorithms for computing such integrals are developed and shown to be highly accurate and stable. An important special case relating to the repeated differentiation of a B-Spline basis function, neglected in the literature, is identified and novel algorithms for dealing with the special case are provided.

While the Analytic algorithm does not perform as accurately as Gaussian Quadrature in extreme cases, due to the propagation of floating-point errors, the algorithm is shown to achieve very high levels of accuracy up to and beyond most practical cases. Additionally, for interactive CAD prototyping or Virtual Sculpting applications, compromises on accuracy are often made in practice to achieve acceptable levels of interaction. An analysis of the small numerical error incurred by the Analytic approach identifies the root cause of the error and suggestions are offered as to ways of achieving greater accuracy.

Overall, the experiments show that the reduction in computational complexity achieved by the proposed Analytic approach presented in this chapter, facilitates significant computational savings of between 63% to 77%, making the algorithm up to 4.3 times faster for the cases presented, at a cost of as little as 0% – $10^{-4}\%$ drop in accuracy from regular to extreme cases of knot spacing respectively.

The solution presented in this chapter not only facilitates interactive intuitive design, but can also readily replace current numeric solvers for the analysis process, such that the issues caused by division in representations are fully resolved (See Figure 5.58). The presented approach not only removes the need for the costly conversion to an FEM representation that makes up to 80% of the total cost of analysis, but it also reduces the computational complexity of the Gaussian Quadrature such that the remaining analysis is up to 4.3 times faster for the cases considered.



(a) Traditional Analysis Pipeline (b) Existing incorporation of IgA (c) Efficient Analytic Solution

Figure 5.58: The Analysis Pipline: Numeric solvers are computationally expensive

Virtual Sculpting Environment

6

This chapter presents an overview of a Virtual Sculpting Environment that was developed over the course of this research as an experimental platform for proof of concept of the ideas and methodologies recited. A fully fledged Sculpting Environment is beyond the scope of this research. Rather, this chapter serves to illustrate how the ideas and techniques presented can be integrated into a Virtual Sculpting Environment.

The chapter begins by outlining the high-level structure of the Virtual Sculpting Environment. The architecture, the backend algorithms and structures, and its front end structure and presentation, are briefly discussed. Additionally, examples are given of the energy minimisation of an Active B-Spline Surface under the influence of its own internal forces; and under the influence of a sculpting tool, subject to constraints. Figure 6.1 shows a screen grab of the homescreen of the environment.

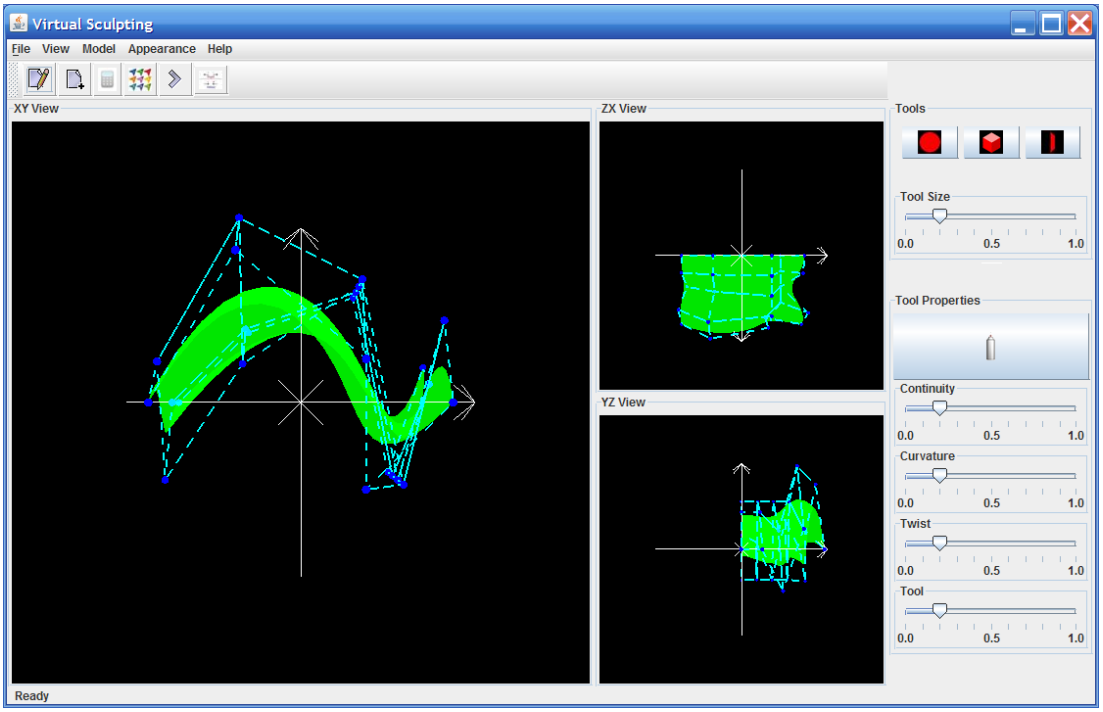


Figure 6.1: Virtual Sculpting Environment

6.1 Model-View-Controller (MVC) Architecture

A model view controller architecture was adopted as the framework for managing the Virtual Sculpting Environment. An MVC model is composed as follows:

- The model stores the data of the system
- The view creates the visual representation of the components from the data in the model
- The controller deals with user interaction with the system components and modifies the model and/or the view in response to user actions as necessary.

Figure 6.2 depicts the composition of the architecture.

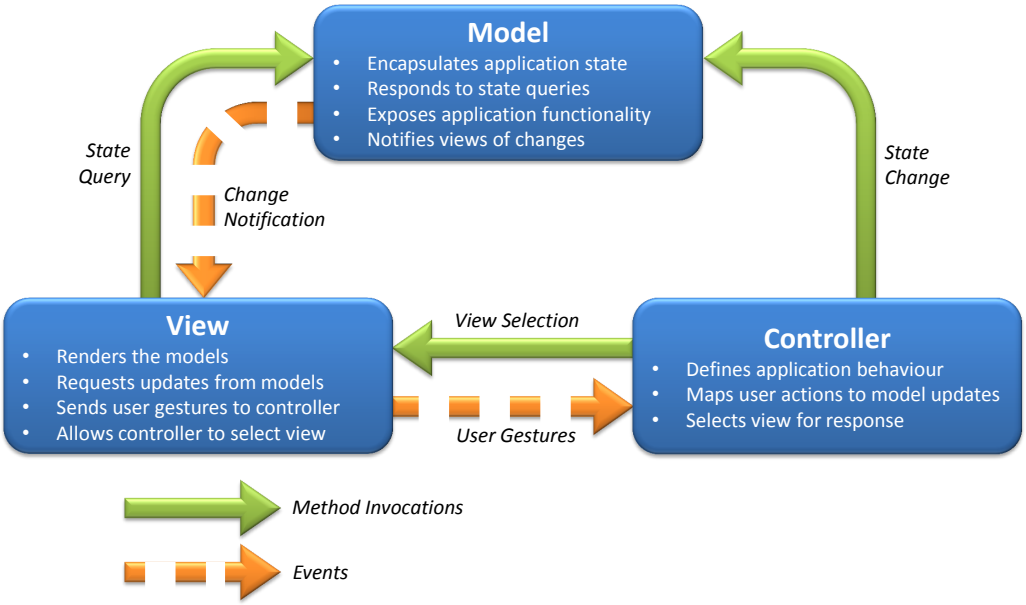


Figure 6.2: MVC Architecture

6.2 A note on Implementation

The JAVA 3D API, a scene graph Application Programming Interface (API) developed by Sun Microsystems, provides a collection of high level constructs for creating, rendering and manipulating a 3D scene graph. This scene graph API was chosen to manage and organise the 3D virtual environment presented. In addition to scene graph management, the Java and Java3D APIs provide a rich library of tools to aid in complex 3D content generation and interactive behaviours.

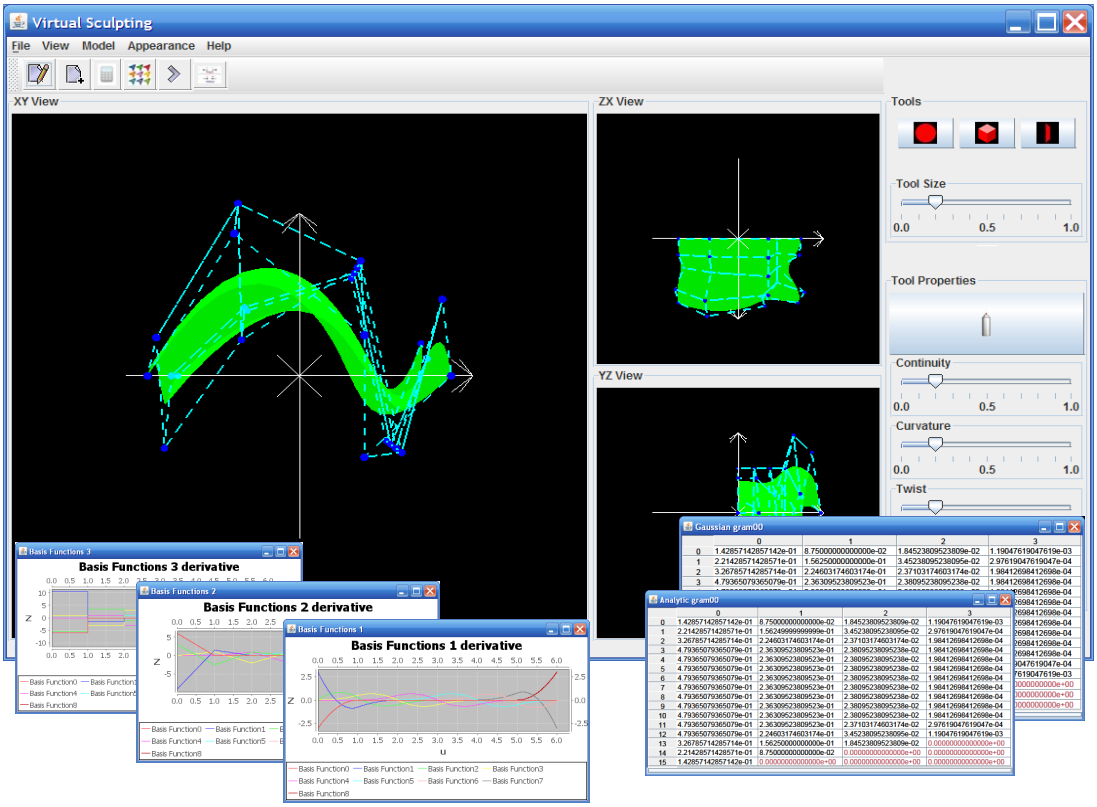


Figure 6.3: Virtual Sculpting System: Backend Visualisation

6.3 Backend

In Chapter 4, a novel ACM-based approach for the Virtual Sculpting of an Active B-Spline/NURBS Surface Model is proposed and the underlying mathematics developed. The backend of the Virtual Sculpting Environment includes the development of a JAVA 3D library implementing Active B-Spline/NURBS surfaces, built on the standard NURBS algorithms presented in *The NURBS Book* by Piegl and Tiller (1997). The library is extended to include the novel algorithms developed during the course of this research. Chapter 5 presented both an optimised Gaussian Quadrature and a novel Analytic solver for the set of energy equations derived. The chapter concluded with output from the Java/Java3D modelling environment. Additional functionality provided by the backend of the system is plotting of basis functions, derivatives and integrals. Figure 6.3 shows examples of plots and Gramian matrices illustrating the backend of the system.

6.4 Frontend

This section presents the user interface and functionality of the Virtual Environment. A subset of the Java3D SceneGraph of the system is briefly covered in order to illlustrate the pertinent frontend functionality. The system usage is then discussed before presenting some examples of the energy minimisation of an Active B-Spline Surface Model under the influence of its own internal forces and under the influence of a sculpting tool.

The Java3D SceneGraph is made up of two key components. The *ViewBranchGraph* and the *ContentBranchGraph*. Figure 6.4 illustrates what is called the *ViewBranchGraph* of the Java3D SceneGraph. This is where all of the functionality relating to viewing is described. Monitoring 3D deformations on a 2D display can be enhanced by adding additional views of the scene. The three *Views* presented are along the the xy, xz, and yz axes, respectively. These views are updated as the user navigates the 3D environment.

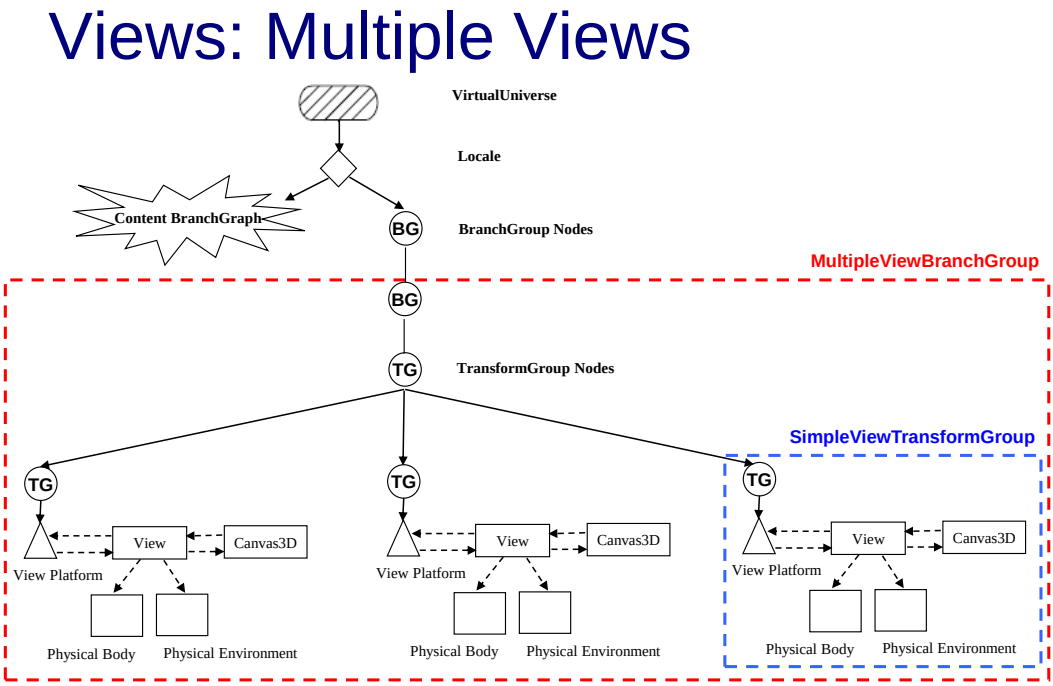


Figure 6.4: View BranchGraph: The dotted blue line depicts the standard SceneGraph View, while the dotted red line depicts the multiple view environment

Figure 6.5 illustrates the *ContentBranchGraph*. The figure shows the key components that interact with the backend in order to simulate the physics based deformation of the Active B-Spline Surfaces. Each 'BG' symbol represents a distinct branch of the scene graph known as a *BranchGraph*. Branches are used to organise related components in the system such that the related items are governed by the same Transform (*TransformGroup* or 'TG'). The main components of the sytem are the NURBS objects themselves which are made up by the three leaf node geometries denoted by 'S'. The three nodes represent the geometry of the NURBS surface itself, its Control Points and its Control Mesh. The PickSpheresBranchGroup enables the user to interactively pick and directly move control points in the system.

6.4.1 User Interface

Figure 6.1 illustrates the user interface of the Virtual Environment. There are a selection of tools ranging from view settings to tool settings. The user can also specify weight

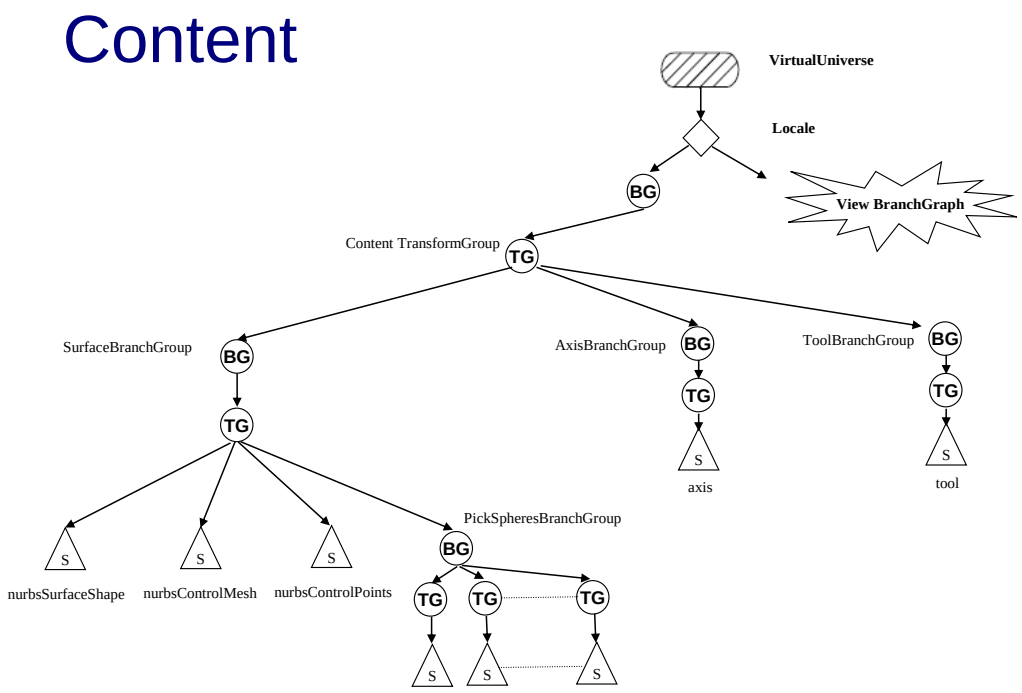


Figure 6.5: Content BranchGraph

parameters that govern the stretch and bend in the Active B-Spline/NURBS Surface Model.

6.4.2 System Usage

Using the system, a user generates a B-Spline/NURBS surface patch. The patch is activated by choosing suitable values for the material properties. The surface points can be enlarged to visually aid selection. Figure 6.6 illustrates the surface property controls, where the material properties can be modified.

External forces are then calculated based on the perpendicular distance between the Active B-Spline/NURBS Surface and the selected deformation tool. The deformation tool is represented by a shape primitive and can be moved and resized freely within the environment, as depicted in Figure 6.7.

The user can specify weight and step size parameters that affect how much and how fast the Active B-Spline Model deforms to the sculpting tool as the energy is minimised. Points/areas not to be affected by a given deformation can be marked up by the user prior to sculpting. The model can be evaluated at all times during the sculpting process from three orthogonal views framed by the xy, xz and yz axes, respectively.

Figure 6.8 (a) shows a frame captured from the environment, illustrating the specification of the Active B-Spline Surface. Figure 6.8 (b) shows the addition of the sculpting tool.

Figures 6.9 (a) and (b) each show a cubic Active B-Spline Surface Model deforming

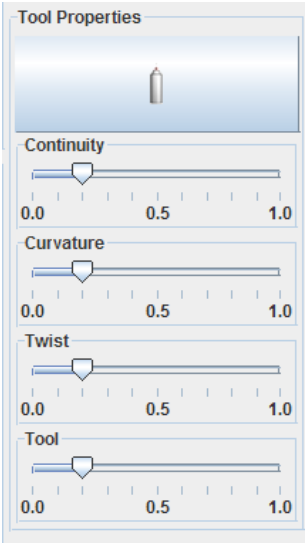
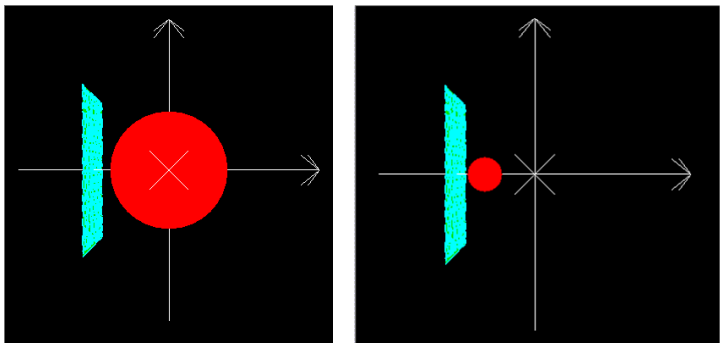


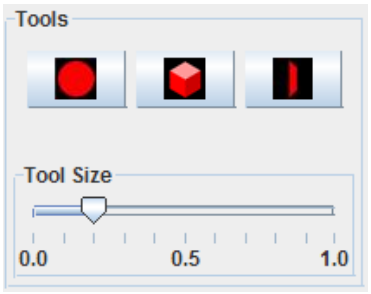
Figure 6.6: Property Application and Control: Energy Properties

under the influence of the shape of the selected tool. In each of the examples shown, the deformation tool is placed close to the Active B-Spline Surface Model which is then permitted to deform under the influence of the tool, while satisfying its internal constraints, until the user achieves the required amount of deformation.

Figure 6.10 illustrates an Active B-Spline Surface collapsing to a point under the

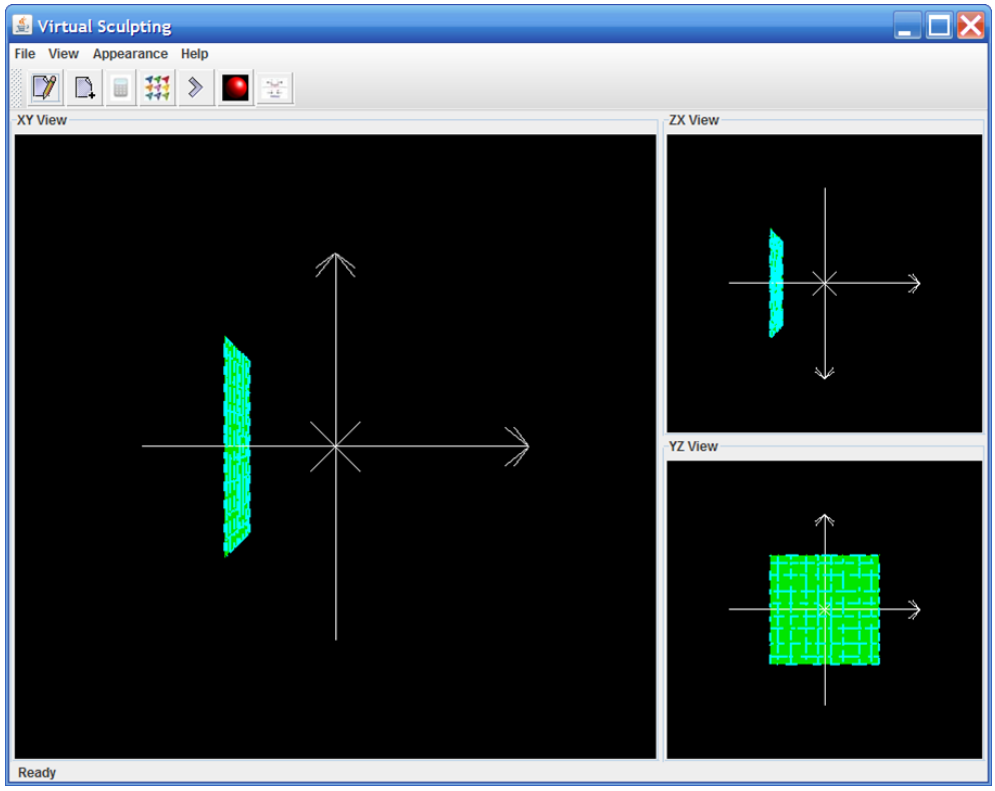


(a) Tool Sizing (b) Tool Resizing

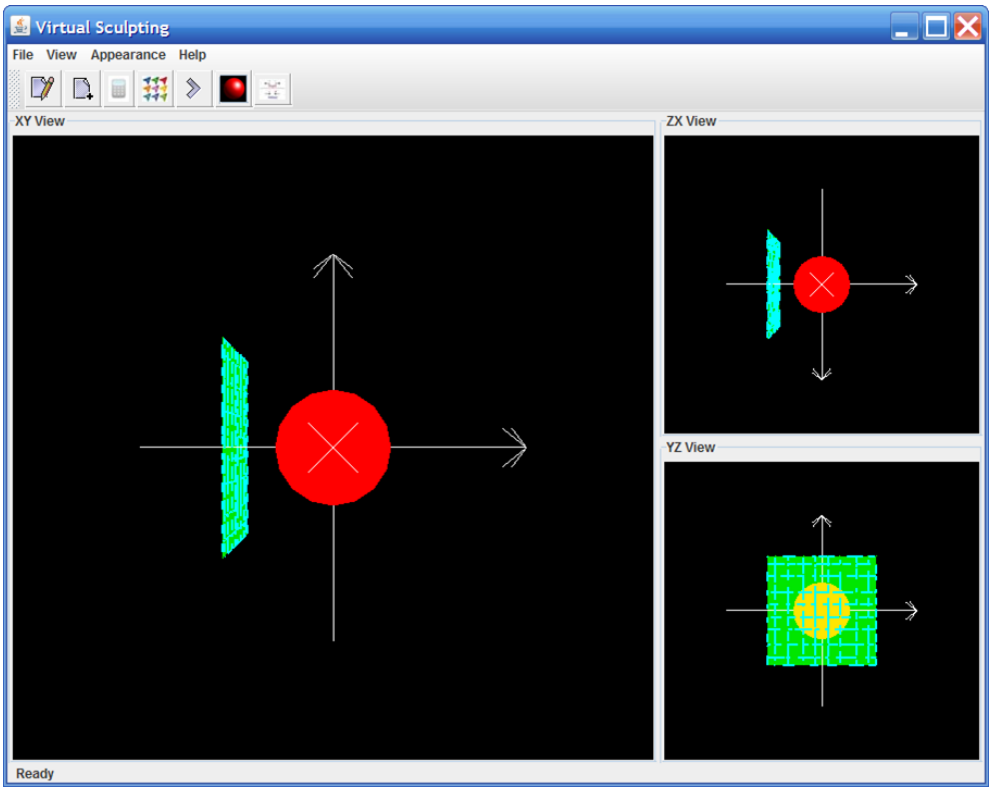


(c) Tool Property Controls

Figure 6.7: Property Application and Control: Tool Properties

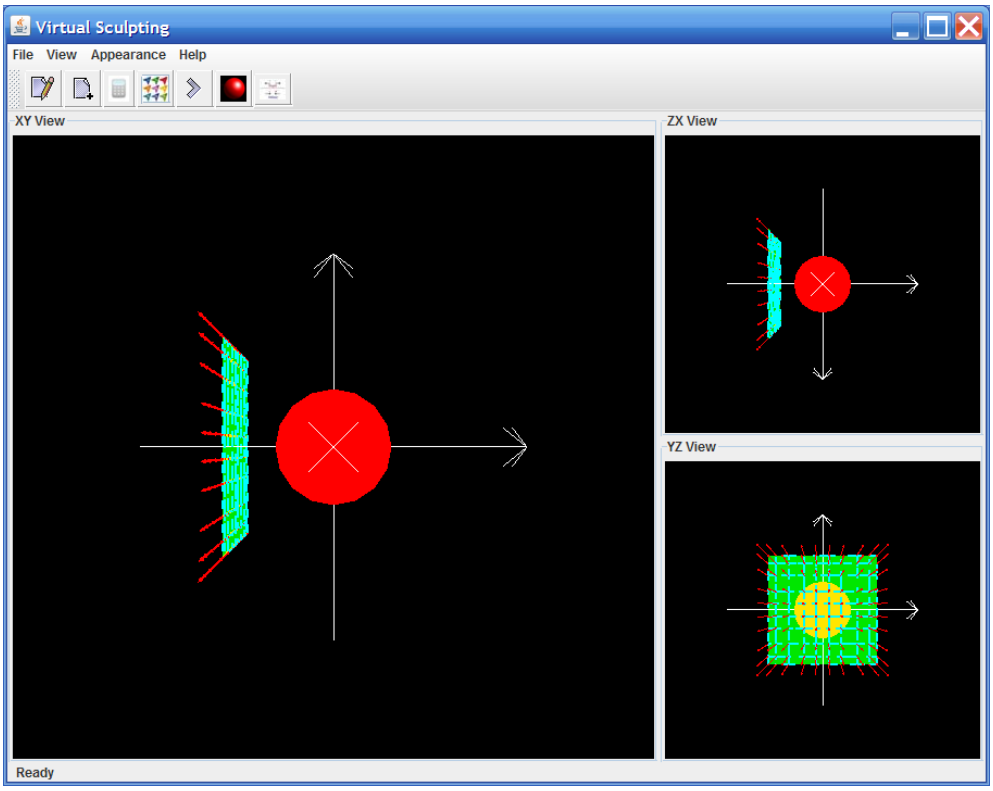


(a) Active Surface

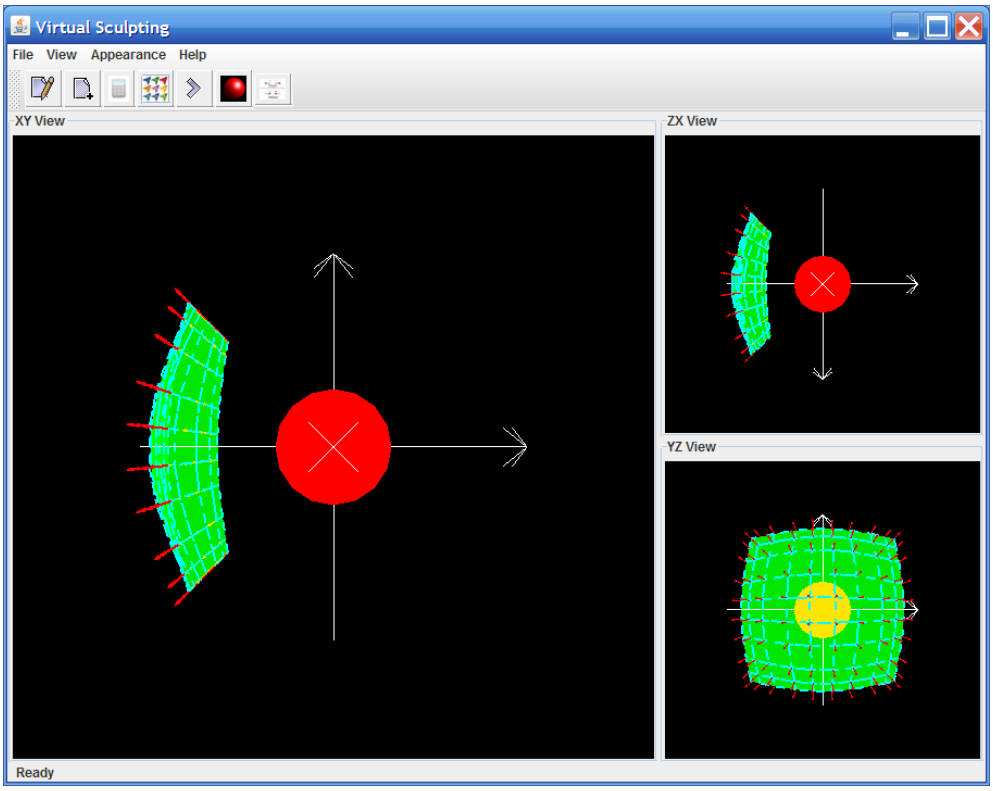


(b) Sculpting Tool

Figure 6.8: Virtual Sculpting Environment showing planar Active B-Spline Surface and Sculpting Tool



(a)



(b)

Figure 6.9: Virtual Sculpting Environment showing Active B-Spline planar surface with selected tool

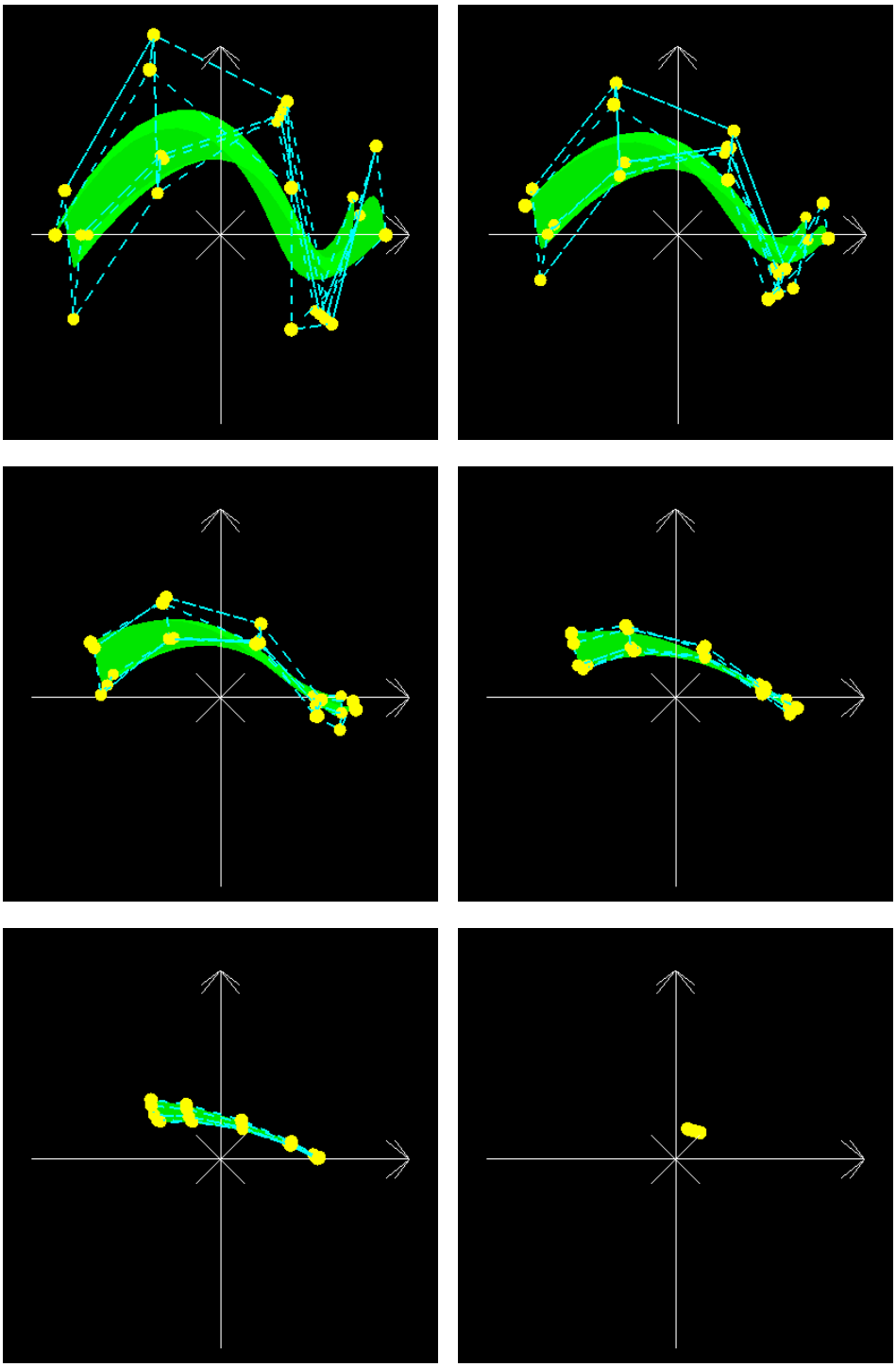
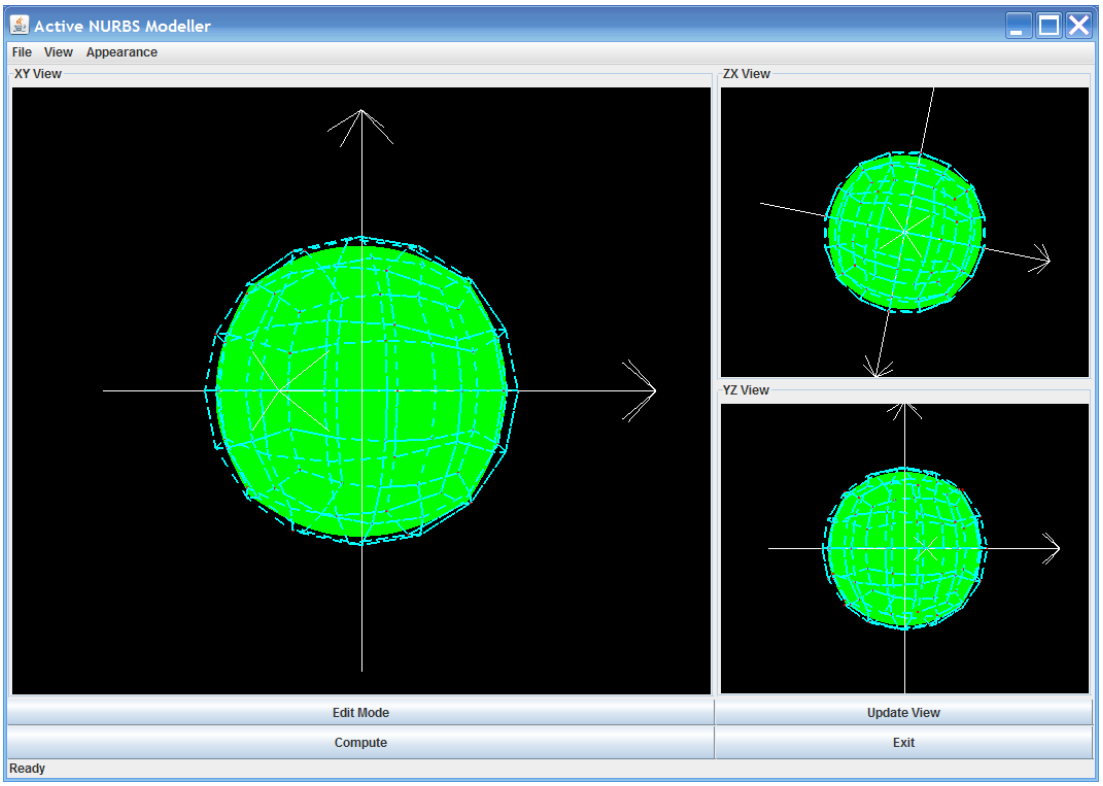


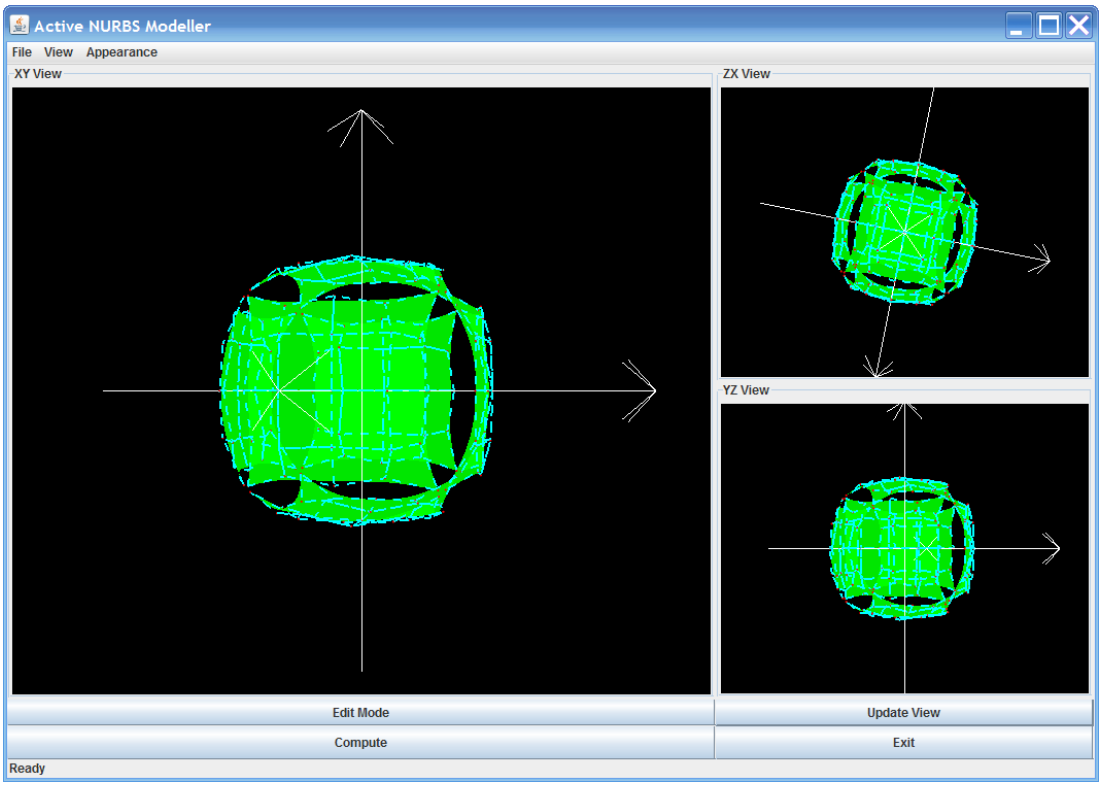
Figure 6.10: ACM surface deforming under internal forces.

influence of its own internal forces. As can be seen from the figure, the internal forces act to flatten and shrink the Active B-Spline Surface.

Figure 6.11 illustrates a multi-patch model coming apart at its seams under the influence of internal forces.

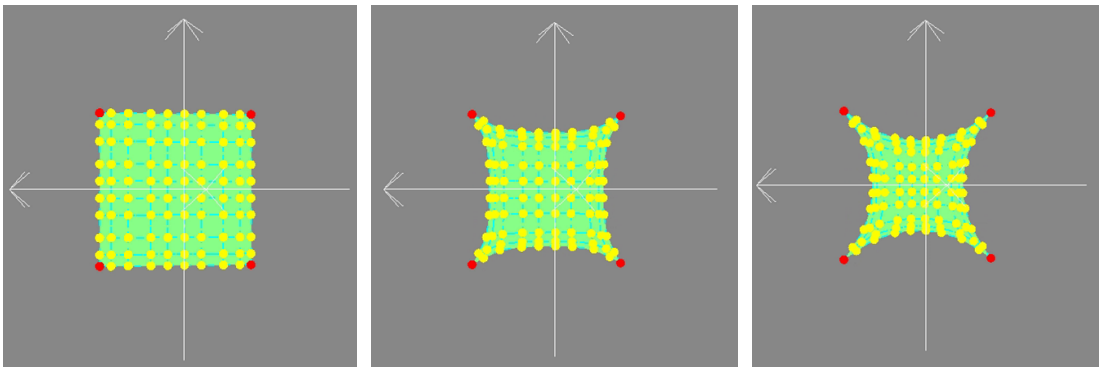


(a) Active Multi-Patch Model



(b) Patch Tearing

Figure 6.11: Virtual Sculpting Environment showing Active B-Spline patch tearing under influence of forces



(a) (b) (c)

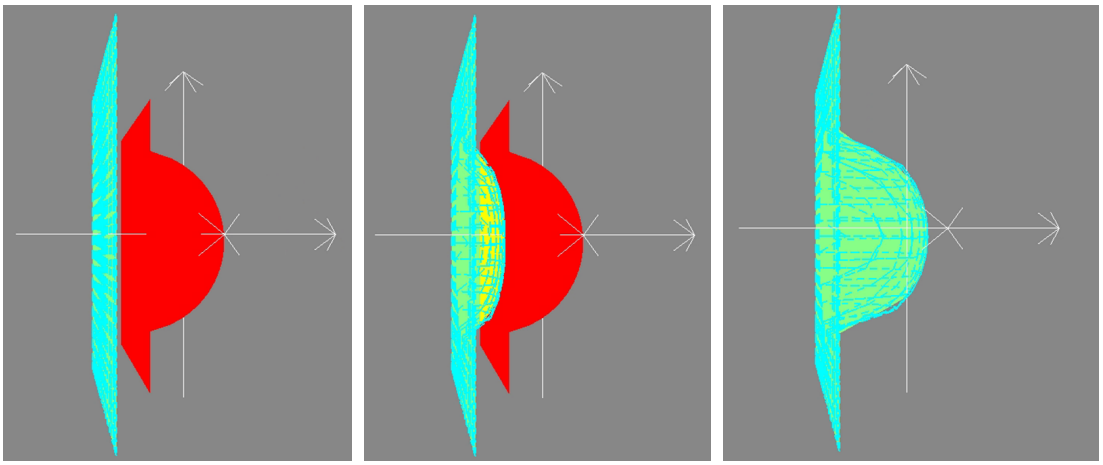
Figure 6.12: Virtual Sculpting Environment showing Active B-Spline patch deformation subject to point contraints

Figure 6.12 illustrates an Active B-Spline batch collapsing under its own internal forces subject to four point constraints. The points shown in red were fixed such that they were not free to move under the action of the internal forces. As can be seen from the figures, the sides of the patch become pinched as the model deforms.

Figure 6.13 shows an example of a proximity-based tool with a concavity. The tool creates a bump on the surface by pulling the surface towards the concavity.

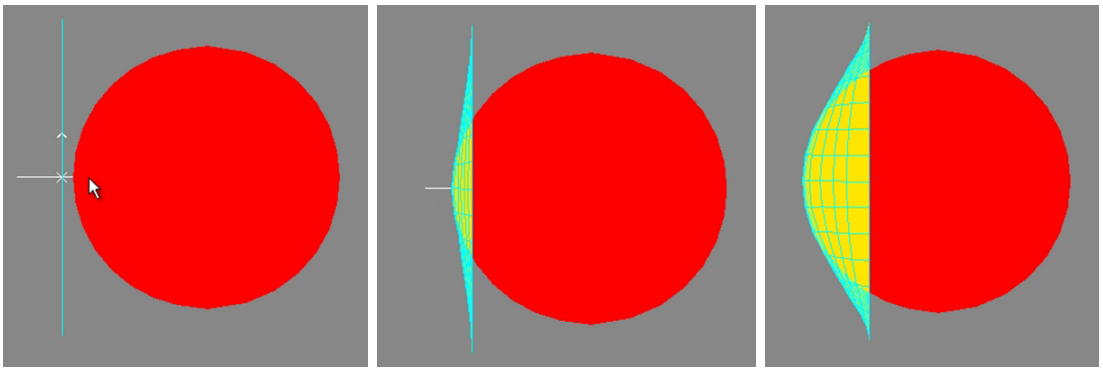
Figure 6.14 shows an example of a collision-based tool. The tool creates an indent in the surface.

Finally, Figure 6.15 illustrates the action of gravity on a falling surface patch. This model incorporates mass and damping properties such that the model animates under the application of the external force. On contact with the spherical tool, the patch drapes over the sphere.



(a) (b) (c)

Figure 6.13: Virtual Sculpting Environment showing proximity-based tool with a concavity operating on the Active B-Spline patch



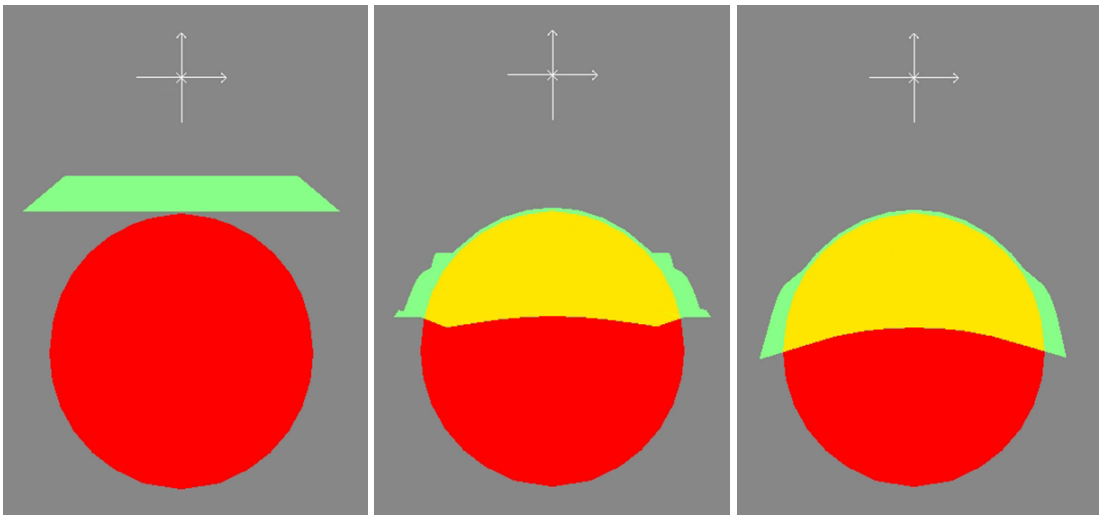
(a) (b) (c)

Figure 6.14: Virtual Sculpting Environment showing patch deforming under collision with spherical tool

6.5 Conclusions

This chapter has presented an overview of the proof of concept Virtual Sculpting Environment, developed over the course of this research. The outputs of most of the pertinent algorithms developed in this thesis were tested in Chapter 5. The Virtual Sculpting Environment was developed as a proof of concept for the feasibility of the proposed approach for the domain of Virtual Sculpting. The chapter presents examples of the energy minimisation of an Active B-Spline Surface under the influence of its own internal forces and under the influence of a sculpting tool, subject to constraints.

The initial implementation based upon this framework requires no more than the interaction capabilities of a standard personal computer. In addition, the system responds in real-time to applied deformations, allowing the designer to instantly evaluate the effect of applied modifications.



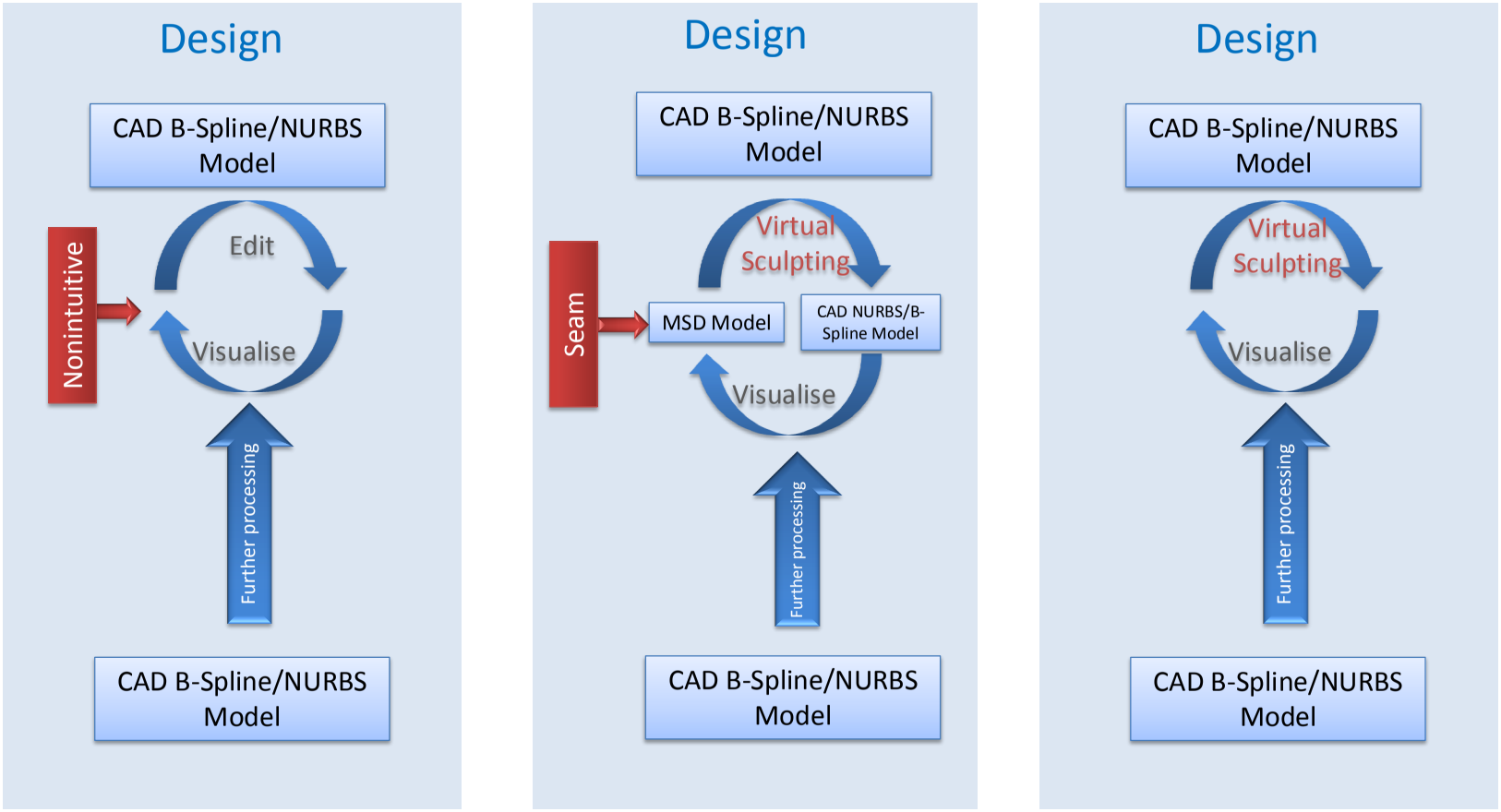
(a) (b) (c)

Figure 6.15: Virtual Sculpting Environment showing patch falling on obstacle under influence of gravity

An auxiliary contribution of this research is the development of a JAVA 3D library implementing Active B-Spline/NURBS surfaces, partially based on customised versions of the algorithms presented in *The NURBS Book* by Piegl and Tiller (1997), and complete with the novel algorithms developed during the course of this research.

This chapter demonstrates that the techniques presented in Chapters 4 and 5 can be fully integrated within a Virtual Sculpting Environment to provide real time, intuitive, and physically realistic interactions.

By facilitating a Virtual Sculpting paradigm within an interactive B-Spline/NURBS design environment, with real-time feedback on design decisions, the complexity of B-SplineNURBS mathematics is hidden behind the more familiar sculpting metaphor. The overall approach has the potential not only to greatly reduce the time investments currently made by computer graphics designers, but also facilitates a seamless integration with existing CAD frameworks (See Figure 6.16).



(a) Traditional Nonintuitive Design Pipeline (b) Existing Virtual Sculpting (c) Seamless Virtual Sculpting

Figure 6.16: The Design Pipeline: Nonintuitive modelling procedures and divisions in representation technologies hinder the design process

Conclusions and Future Work 7

This thesis has developed a novel framework for Efficient Analysis of Active B-Spline/NURBS surface models with application to Virtual Sculpting. The main objective of the thesis is to address design and analysis challenges in traditional CAD workflows relating to both divisions in representation resulting in conversion bottlenecks and the use of computationally expensive numeric techniques. This chapter presents an outline of the findings and contributions of this thesis. A list of the publications which have arisen as a result of this work is also included. Some extensions and suggested future directions of this research are then highlighted. Finally, a brief discussion is included comparing the stated objectives with the achievements of the research.

7.1 Summary of Contributions

The immediate contribution of this work is the overall method that provides, for the first time, a seamless Virtual Sculpting design and analysis methodology for freeform surfaces that is fully compatible with existing CAD frameworks. The innovative Analytic solver not only facilitates interactive intuitive design, but can also readily replace current numeric solvers for the analysis process, such that traditional issues caused by division in representations are fully resolved.

The main findings and contributions resulting from this thesis are identified and summarised below.

7.1.1 Literature Review

Freeform surface design and analysis has broad applications across the many disparate domains of computing. Consequently there is a vast body of literature that must be examined in order to fully appreciate the current state of the art. The extent of the literature has served to confuse and overwhelm general practitioners. In Chapter 2, a detailed exposition of the pertinent literature is presented. This exposition serves not only to illuminate the current state of the art, but also to provide a historical context such that the motivations of the contrasting techniques and their developments can be fully appreciated. The various methodologies are compared, contrasted and critiqued. The presented analysis facilitated an informed design decision for the purpose of this research, and also provides general practitioners with the necessary background to

enable informed design decisions. No other work was found in the literature that details the full spectrum of methods across the variety of domains that contribute to state of the art in freeform design.

7.1.2 Unification of Energy-Based Deformable Surface Modelling Methods

The use of energy-based techniques for the implementation of physically aware deformable models is widespread across a multitude of computing domains. As a consequence, a wide range of disparate approaches to representing and solving the associated energy minimisations have emerged in the literature. Both terminologies and, more importantly, model origins and derivations, vary across the different disciplines to the extent that their relationships are ambiguous in the literature relating to Computer Graphics, Machine Vision, and Visualisation. In Chapter 3, the ambiguities are addressed by showing how the most common energy minimisation approaches for deformable modelling are in fact all derived from the same foundations in continuum mechanics. Seemingly different models, from the fields of classical continuum mechanics and differential geometry, are shown to be largely similar in a thorough treatment of their theoretical derivations. It is also demonstrated that further ambiguities arise where subtle similarities between seemingly different models are introduced as a consequence of the solvers employed to perform the energy minimisations. By formalising the theoretical relationships between the different classes of energy-based models currently in use, a unified approach to their representation is developed, thus resolving the discussed ambiguities.

7.1.3 Novel ACM-Based Approach for the Virtual Sculpting of an Active B-Spline/NURBS Surface Model

In Chapter 4, a novel ACM-based approach for the Virtual Sculpting of an Active B-Spline/NURBS Surface Model is proposed. The novelties of this approach stem from several sources: firstly, the adoption of the ACM philosophy in its design approach via energy based minimisation under the influence of shaped features/tools, and also in its user controlled evolution; secondly, in its development as an *analytic* approach to energy-based minimisation; thirdly, in its application to the domain of Virtual Sculpting; and finally, its customisation for B-Spline/NURBS representations which heretofore have not had the benefit of the true embodiment of a Virtual Sculpting metaphor. This facilitates intuitive deformations of an Active B-Spline Surface through the application of forces derived from shape primitives. A major benefit of the proposed approach is the fully analytic description of the physics-based design metaphor. Unlike previous approaches, this allows for seamless integration with existing CAD workflows, without the need for conversions of representation. An additional contribution of the develop-

ment of the approach is the thorough treatment of the underlying mathematics of an Active B-Spline surface which could not be found in the literature.

7.1.4 An Efficient Mathematical Framework for Solving Energy-Based Deformations of B-Spline/NURBS Surface Models

A novel Efficient Mathematical Framework for solving Energy-Based Deformations of B-Spline/NURBS Surface Models was developed in Chapter 5 based on an analytic solution of the equations developed for the description of the surface and its energy in Chapter 4. The proposed Analytic approach is shown to significantly reduce the computational complexity of the energy problem. Algorithmic efficiencies are also identified and applied equally to both the Analytic and Gaussian Quadrature approaches such that there is no unnecessary repetition of calculations in either approach. The resulting algorithm for the Analytic approach is shown to be up to *4.3* times faster than the state of the art Gaussian Quadrature approach at computing the Stiffness Matrix of the Active B-Spline Surface, one of the more time consuming operations in computer-based deformable modelling applications. Previously, a direct analytic solution was not available for B-Spline/NURBS Surface Models. While initially designed to facilitate the seamless integration of a Virtual Sculpting paradigm within the CAD design process, the proposed Analytic solver can also readily replace current numeric solvers for the analysis process. The combination of the Computationally Efficient Mathematical Framework and the ACM-based approach for Virtual Sculpting of an Active B-Spline/NURBS Surface Model thus constitutes a fully integrated, seamless, and interactive freeform surface design and analysis methodology.

Several distinct contributions led to the design and analysis methodology: the proposed technique is benchmarked against Gaussian Quadrature, the state of the art numerical method for energy minimisation of a B-Spline/NURBS surface. Gaussian Quadrature is often treated as a black box numeric solver and standard algorithms for its implementation exist. However, as a result of thorough treatment of its operation, and study relating to its application to the energy minimisation problem discussed, numerous problem-specific and approach-specific efficiencies were identified and incorporated into a tailored Gaussian Quadrature algorithm that is presented for application to B-Spline/NURBS representations. The enhanced algorithm was used to benchmark the proposed Analytic approach which was shown through rigorous experimentation, in both simulated and real tests, to be more than *four* times faster than Gaussian Quadrature. Detailed numerical testing and analysis of the resulting algorithms demonstrates that for practical cases, the Analytic approach provides a very high degree of accuracy. A detailed analysis of the causes and effects of small roundoff errors in extreme cases is presented, and methods for minimising their impact are proposed.

As part of the development of the outlined approaches, properties of the integrals and derivatives of a B-Spline basis function, neglected by the literature, are exposed and presented. Physical and graphical interpretations of the underlying mechanics of the analytic solution are also developed and presented. The approach is shown to be fully extensible to the case of varying material parameters and forcing functions and also to the important case of NURBS, by virtue of the approximating powers of B-Splines.

7.1.5 Prototype Virtual Sculpting Environment

A Virtual Sculpting Environment is presented in Chapter 6, implementing the proposed interactive design technique. The outputs of most of the pertinent algorithms developed in this thesis were tested in Chapter 5. The Virtual Sculpting Environment was developed as a proof of concept for the feasibility of the proposed approach for the domain of Virtual Sculpting. The chapter presents examples of the energy minimisation of an Active B-Spline Surface under the influence of its own internal forces and under the influence of a sculpting tool, subject to constraints. An auxiliary contribution of this research is the development of a JAVA 3D library implementing Active B-Spline/NURBS surfaces, partially based on customised versions of the algorithms presented in *The NURBS Book* by Piegl and Tiller (1997), and complete with the novel algorithms developed during the course of this research.

7.2 Publications Arising

The following peer-reviewed publications stem directly from this research.

Efficient Energy Evaluations for Active B-Spline/NURBS Surfaces, Moore, P. and Molloy, D., *Computer-Aided Design*, In Press, Accepted Manuscript 2013 (Moore and Molloy, 2013).

Active B-Spline Surface Models for Intuitive 3D Virtual Sculpting, Moore, P. and Molloy, D., In Proceedings of the 28th Spring Conference on Computer Graphics 2012, Smolenice, Slovakia (Moore and Molloy, 2012).

Active Contour Models for Computer Graphics Shape Modelling and Deformation, Moore, P. and Molloy, D., In Proceedings of Vision, Graphics, and Visualisation 2008, Trinity College Dublin, Ireland (Moore and Molloy, 2008*a*).

Virtual Sculpting: Active Contour Models for Computer Graphics Shape Modelling and Deformation, Moore, P. and Molloy, D., In In Proceedings of EuroGraphics Ireland, 2007, University College Dublin, Ireland (Moore and Molloy, 2007*b*).

A Survey of Computer-Based Deformable Models, Moore, P. and Molloy, D., In Proceedings of the International Machine Vision and Image Processing Conference, 2007, NUI Maynooth, Ireland (Moore and Molloy, 2007*a*).

Active Surface Meshes for Intuitive 3D Computer Graphics Shape Deformation and Modelling, Moore, P. and Molloy, D., In Proceedings of the Irish Machine Vision and Image Processing Conference, 2006, Dublin City University, Ireland (Moore and Molloy, 2006).

Auxiliary publications

Automatic Construction of 3D Human Models from Only Two Orthographic Projections, Moore, P., Boyle, E. and Molloy, D., In Proceedings of EuroGraphics Ireland, 2005, Institute of Technology Blanchardstown, Ireland (Moore et al., 2005).

Seminars

Virtual Sculpting: Active Surface Meshes for Intuitive Computer Graphics Shape Modelling and Deformation, Moore, P. and Molloy, D., RINCE Research Seminar Series, Dublin, Ireland, 2008 (Moore and Molloy, 2008*b*).

7.3 Directions for Future Research

The generality of the Efficient Mathematical Framework presented in this thesis opens up a wide range of future directions for research. In this section, several extensions and related topics to this work worthy of further investigation are briefly described.

7.3.1 Enhanced Computational Efficiency

Farin (2002) and Vassilev (1997) note that the vectorisation of the control point matrix, a standard approach when dealing with B-Spline Surfaces, adds computational complexity to the solution of the problem. Where the Matrix structure of the control points can be preserved, the system is of type $\mathbf{AXB} = \mathbf{D}$ which can be solved in two passes, reducing the computational cost from $O(r^6)$, for the traditional case, to $O(r^4)$. Further investigation is merited to ascertain if this is possible within the analytic framework presented in this thesis.

7.3.2 Exact Extension to NURBS

The extension to NURBS presented in this thesis is an approximating extension, relevant to NURBS because of the approximation properties of B-splines (Hughes et al., 2010). An area worthy of further exploration is to ascertain if the analytic approach presented in this thesis may be used to obtain an exact analytic solution for the NURBS case. Integrals of rational functions are notoriously difficult to work with. Further work is needed to determine if the techniques presented in this thesis could be employed to

group the terms in such a way as to reduce the complex rational term to a summation of its samples.

7.3.3 Non-Linear Analysis

The Virtual Sculpting approach developed in this thesis is a linearised approximation of the true mechanics of a physical deformation as developed in Chapter 3. For CAD applications, linear models are good approximate solvers as the materials used within CAD environments generally operate within linear specifications. Non-linear systems are prevalent in areas such as medical visualisation of soft tissues and cloth modelling (Etzmuß et al., 2003, Nealen et al., 2006), where large deformations can occur and accuracy can be degraded where linear approximations are made. The significant savings achieved by the computationally efficient mathematical framework presented in this thesis may make the approach viable for the interactive simulation of non-linear systems.

7.3.4 Multi-Patch Models

The generalisation from B-Spline curves to surfaces uses a tensor-product construction, resulting in a rectangular parameter space and a surface that is topologically equivalent to a plane. Closing or looping the parameter space makes it possible to create a surface that is topologically an open cylinder or a torus, but a single patch cannot represent a surface of higher genus. To overcome this restriction, multiple patches are stitched together to facilitate more complex designs. For Active B-Spline Surfaces, this presents a problem of connecting the physics substrate across patches. This is very much an unsolved problem in the literature. Pungotra et al. (2010) attempt a solution by merging patches together to create a single larger patch. This is achieved by discretising the multiple patches to a discrete set of points and fitting a new B-Spline surface to the discrete points using a global interpolation technique. However, the approach does not solve the problem. By merging rectangular patches into a single larger rectangular patch, nothing is gained in terms of topological freedom. It may be more advantageous to store the individual patches as separate elements and implement the desired effects in software rather than mathematics. The Object Oriented approach adopted in the development of the application presented in this thesis may facilitate the stitching of multiple patches by virtue of what might be called a quirk in Object Oriented Programming. By sharing references to the control points governing a desired seam in the model, there may be scope to facilitate the merging of multiple patches, while ensuring the propagation of internal forces with no tearing at the seams. This would facilitate much greater flexibility in topology.

7.3.5 ACM/Snake Applications

The techniques presented in this thesis have broad scope and influence for many tasks outside of the Virtual Sculpting domain, e.g., within Machine Vision and Visualisa-

tion. The approach to Virtual Sculpting presented in this thesis drew its inspiration from the proven machine vision approach of ACMs. An area of future research, that now somewhat ironically holds much promise, is to explore the many opportunities for improvements that exist within Machine Vision and Visualisation. ACM curves and their generalisation to 3D systems have been used for a wide variety of image processing tasks including edge detection, image segmentation, model fitting, and motion tracking (Richens et al., 1992, Ferrier et al., 1994, Vieren et al., 1995, Cohen, 1996, Derraz et al., 2004). This list is by no means exhaustive as ACMs are used for a host of image processing and visualisation tasks. From the literature, numerical approaches such as FEM methods are the current state of the art for solving such systems. Where B-Spline/NURBS form the geometric substrate of the model, which in practice is often the case (Menet et al., 1990, Meegama and Rajapakse, 2003), the Analytic approach presented in this thesis, for solving for the energies, can be employed.

7.4 Concluding Remarks

This thesis set out to develop an approach to interactive freeform surface design to tighten the design/analysis loop in CAD applications by removing the bottlenecks created by model conversions, and facilitate the seamless integration of a true Virtual Sculpting paradigm within existing CAD workflows. The overall goal was to deliver the situation depicted by Figure 7.1. The overall ACM-based Virtual Sculpting approach to freeform surface design that is presented in this thesis achieves this goal by imposing the Virtual Sculpting metaphor upon a single fully analytic model and solver. The Analytic approach has the potential to free designers from making decisions based on analyses of incomplete, discrete descriptions of the geometric data. The model integrity is preserved right through the often iterative design and analysis processes. The presented technique has the potential to draw the CAD design and analysis communities closer together. This stems from the removal of the barriers in representation and bottlenecks in conversion that exist in traditional workflows. The significant savings in computation time, resulting from the analytic solution, have the potential to greatly improve interactivity. Finally, the techniques and algorithms developed in this thesis are quite general and have broad scope and influence for many tasks outside of the Virtual Sculpting domain.

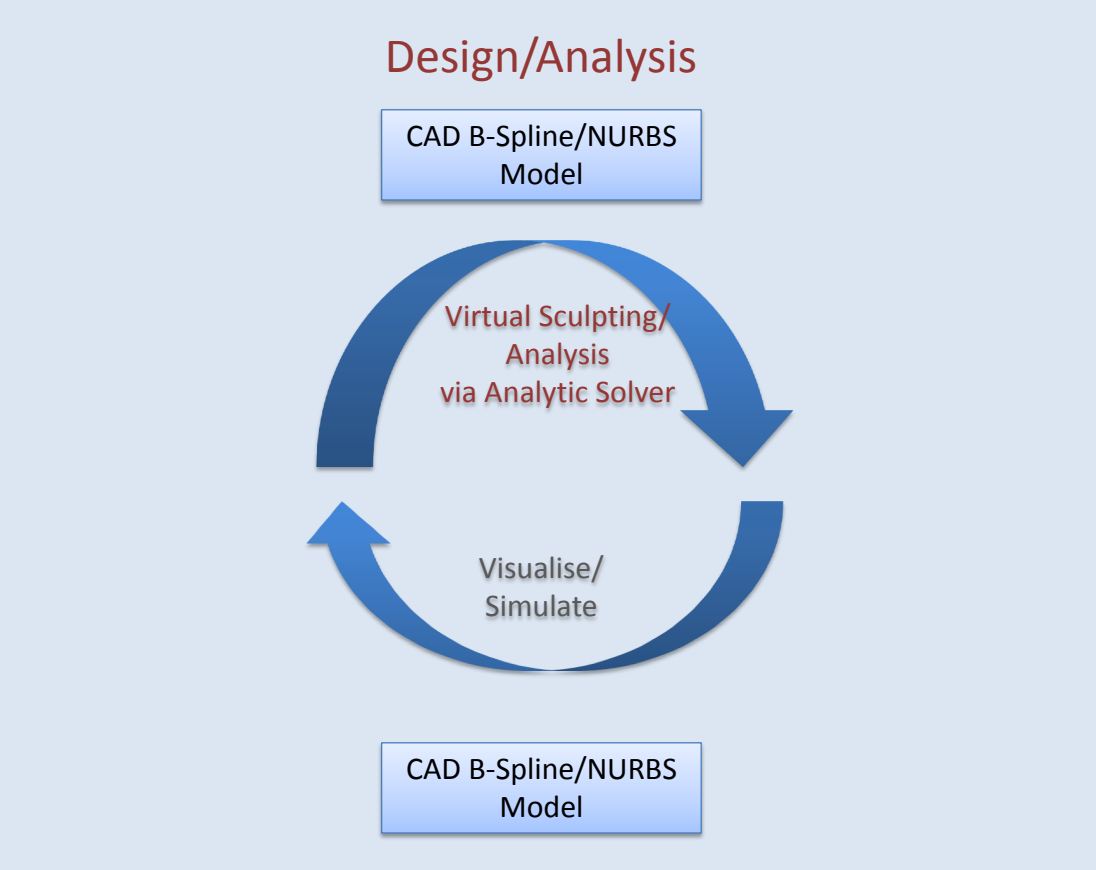


Figure 7.1: Proposed Seamless Virtual Sculpting Approach that preserves the analytic representation throughout the design and analysis process

Appendix A - Notation

A Note on Notation

The description of a deformable body requires dealing with vector and tensor fields such as displacements, strains, and stresses. However, across the different domains, the terminology and notations vary. While the terminology is difficult to group in a meaningful way, in general, the notation surrounding vector and tensor fields can be grouped under four main headings. Throughout the following discussions, the standard domain specific notation is adopted in each case. For this reason, the various notations are briefly outlined in this section. To illustrate the differences, the dot product between two vectors a and b is shown in each notation.

Full Notation

As the title implies, full notation delivers a complete description of the system, without relying on groupings, simplification, or abbreviations. There is no scope for ambiguity and thus, this representation is desirable when moving between domains and facilitates ease of understanding and a common frame of reference for meaningful comparisons between techniques. However, this notation is rarely delivered in practice due to time and space constraints.

$$\mathbf{a} \cdot \mathbf{b} = a_1 b_1 + a_2 b_2 + a_3 b_3 \tag{1}$$

Direct Notation/Algebraic Notation

With direct notation, vectors and tensors are represented by single symbols, usually bold letters. They are linked by the mathematical operators for dealing with such representations, e.g., ∇ for gradient.

$$\mathbf{a} \cdot \mathbf{b} \tag{2}$$

Matrix Notation

Matrix notation is similar to Direct Notation. However, the mathematical operators are dropped in favour of Matrix representations.

$$\mathbf{a}^T \mathbf{b} \tag{3}$$

Indices/Componentwise Notation

Subscripted indices are often used to denote/identify the individual components of a vector or tensor. It is often favoured for convenience as it is well suited to abbreviations using Einstein summations. Using this notation, commas can be used to imply partial derivatives.

$$\mathbf{a}_i \mathbf{b}_i \tag{4}$$

Bibliography

Agoston, M. 2004. *Implementation & algorithms ; 2. Mathematics*. Computer Graphics & Geometric Modeling. Springer.

Ahlberg, J. 1996. Active contours in three dimensions. Master's thesis, Linköping University.

Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D. and Silva, C. T. 2001. Point set surfaces. In *Proceedings of the IEEE Conference on Visualization*, VIS '01, IEEE Computer Society, pp. 21–28.

Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D. and T. Silva, C. 2003. Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics* 9(1), pp. 3–15.

Alexa, M., Darmstadt, T., Gross, M., Gross, M., Pauly, M., Zürich, E., Zürich, E., Zürich, E., Pfister, H., Pfister, H., Stamminger, M., universität Weimar, B. and Zwicker, M. 2002. Point-based computer graphics. In *SIGGRAPH 2004 Course Notes*.

Allan, J., Wyvill, B. and Witten, I. 1989. A methodology for direct manipulation of polygon meshes. In *New Advances in Computer Graphics*, Springer, pp. 451–469.

Allen, G. 2007. Geometric modeling problems in industrial CAD/CAM/CAE [Online]. Available from: `http://siag.project.ifi.uio.no/problems/allen.pdf`, [Accessed December 2012].

Angelidis, A. 2005. Shape Modeling by Swept Space Deformation. PhD thesis, University of Otago.

Angelidis, A., Cani, M.-P., Wyvill, G. and King, S. A. 2006. Swirling-sweepers: Constant-volume modeling. *Graphical Models* 68(4), pp. 324–332.

Angelidis, A., Wyvill, G. and Cani, M.-P. 2004. Sweepers: Swept user-defined tools for modeling by deformation. In *Proceedings of Shape Modeling International*, SMI '04, IEEE Computer Society, pp. 63–73.

Auricchio, F., Calabrò, F., Hughes, T., Reali, A. and Sangalli, G. 2012. A simple algorithm for obtaining nearly optimal quadrature rules for NURBS-based isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering* 249Ű252(0), pp. 15 – 27.

AutoDesk 2012. AutoDesk Maya 2012 User Guide [Online]. Available from: http://download.autodesk.com/global/docs/maya2012/en_us/, [Accessed December 2012].

Baraff, D. and Witkin, A. P. 1998. Large steps in cloth simulation. In *Proceedings of SIGGRAPH 98*, Annual Conference Series, ACM, pp. 43–54.

Barr, A. H. 1984. Global and local deformations of solid primitives. *Computer Graphics* 18(3), pp. 21–30.

Bartels, R. H., Beatty, J. C. and Barsky, B. A. 1987. *An introduction to splines for use in computer graphics & geometric modeling*. Morgan Kaufmann Publishers Inc.

Belytschko, T., Lu, Y. Y. and Gu, L. 1994. Element-free galerkin methods. *International Journal for Numerical Methods in Engineering* 37(2), pp. 229–256.

Bill, J. R. and Lodha, S. K. 1994. Computer sculpting of polygonal models using virtual tools. Technical Report UCSC-CRL-94-27, Baskin Centre for Computer Engineering & Information Sciences, University of California, Santa Cruz.

Blinn, J. F. 1982*a*. A generalization of algebraic surface drawing. *ACM Transactions on Graphics* 1, pp. 235–256.

Blinn, J. F. 1982*b*. Light reflection functions for simulation of clouds and dusty surfaces. *Computer Graphics* 16(3), pp. 21–29.

Borrel, P. 1994. Simple constrained deformations for geometric modeling and interactive design. *ACM Transactions on Graphics* 13(2), pp. 137–155.

Botsch, M. and Kobbelt, L. 2004. An intuitive framework for real-time freeform modeling. *ACM Transactions on Graphics* 23(3), pp. 630–634.

Botsch, M. Pauly, C. R. S. B. L. K. 2006. Geometric modeling based on triangle meshes. In *ACM SIGGRAPH 2006 Course Notes*.

Botsch, M. and Sorkine, O. 2008. On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics* 14(1), pp. 213–230.

Bærentzen, J. A. 1998. Octree-based volume sculpting. In *Proceedings of the IEEE Conference on Visualization*, VIS '98, IEEE Computer Society, pp. 9–12.

Bro-Nielsen, M. and Cotin, S. 1996. Real-time volumetric deformable models for surgery simulation using finite elements and condensation. *Computer Graphics Forum* 15(3), pp. 57–66.

Cani, M.-P., Igarashi, T. and Wyvill, G. 2008. *Interactive Shape Design*. Synthesis Lectures on Computer Graphics and Animation. Morgan & Claypool Publishers.

Cashman, T. J. 2010. NURBS-compatible subdivision surfaces. Technical Report UCAM-CL-TR-773, University of Cambridge.

Celniker, G. and Gossard, D. 1991. Deformable curve and surface finite-elements for free-form shape design. *Computer Graphics* 25(4), pp. 257–266.

Celniker, G. and Welch, W. 1992. Linear constraints for deformable non-uniform B-Spline surfaces. In *Proceedings of the Symposium on Interactive 3D Graphics*, I3D '92, ACM, pp. 165–170.

Chadwick, J. E., Haumann, D. R. and Parent, R. E. 1989. Layered construction for deformable animated characters. *Computer Graphics* 23(3), pp. 243–252.

Chang, Y.-K. and Rockwood, A. P. 1994. A generalized de casteljau approach to 3D free-form deformation. In *Proceedings of SIGGRAPH 94*, Annual Conference Series, ACM, pp. 257–260.

Chen, D. T. and Zeltzer, D. 1992. Pump it up: computer animation of a biomechanically based model of muscle using the finite element method. *Computer Graphics* 26(2), pp. 89–98.

Christensen, M. K. and Floren, A. 2004. Implementation of deformable objects. Technical report, Department of Computer Science, University of Copenhagen.

Cingoski, V., Miyamoto, N. and Yamashita, H. 1998. Element-free galerkin method for electromagnetic field computations. *Magnetics, IEEE Transactions on* 34(5), pp. 3236–3239.

Cohen, E., Riesenfeld, R. and Elber, G. 2001. *Geometric Modeling With Splines: An Introduction*. A K Peters.

Cohen, L. 1991. On active contour models and balloons. *CVGIP: Image Understanding* 53(2), pp. 211–218.

Cohen, L. 1996. Deformable surfaces and parametric models to fit and track 3D data. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, IEEE, pp. 2451–2456.

Cohen, L. and Cohen, I. 1990. A finite element method applied to new active contour models and 3d reconstruction from cross sections. In *Proceedings of the Third International Conference on Computer Vision*, ICCV '90, IEEE, pp. 587–591.

Cohen, L. and Cohen, I. 1992. Deformable models for 3D medical images using finite elements and balloons. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, CVPR '92, Soc. Press, pp. 592–598.

Cohen, L. and Cohen, I. 1993. Finite-element methods for active contour models and balloons for 2D and 3D images. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15(11), pp. 1131–1147.

Collier, J., Collier, B., OŠToole, G. and Sargand, S. 1991. Drape prediction by means of finite-element analysis. *Journal of the Textile Institute* 1(82), pp. 96–107.

Cootes, T. F., Edwards, G. J. and Taylor, C. J. 2001. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23(6), pp. 484–498.

Coquillart, S. 1990. Extended free-form deformation: A sculpturing tool for 3D geometric modeling. *Computer Graphics* 24(4), pp. 197–196.

Cotin, S., Delingette, H. and Ayache, N. 1996. Real time volumetric deformable models for surgery simulation. In *Visualization in Biomedical Computing*, Lecture Notes in Computer Science, Springer, pp. 535–540.

Cottrell, J., Hughes, T. and Bazilevs, Y. 2009. *Isogeometric Analysis: Toward Integration of CAD and FEA*. Wiley.

CSG Tree [Online] n.d.. Available From: `http://en.wikipedia.org/wiki/Constructive_solid_geometry`, [Accessed November 2012].

Dachille IX, F., Qin, H., Kaufman, A. and El-Sana 2001. A novel haptics-based interface and sculpting system for physics-based geometric design. *Computer-Aided Design* 33(5), pp. 403 – 420.

Dachille IX, F., Qin, H., Kaufman, A. and El-Sana, J. 1999. Haptic sculpting of dynamic surfaces. In *Proceedings of the Symposium on Interactive 3D Graphics*, I3D '99, ACM, pp. 103–110.

Deboor, C. 1978. *A Practical Guide to Splines*. Springer-Verlag Berlin and Heidelberg GmbH & Co. K.

Decaudin, P. 1996. Geometric deformation by merging a 3D object with a simple shape. In *Proceedings of Graphics Interface*, GI '96, Canadian Information Processing Society, pp. 55–60.

Derraz, F., Beladgham, M. and Khelif, M. 2004. Application of active contour models in medical image segmentation. In *Proceedings of the International Conference on Information Technology: Coding and Computing*, ITCC '04, IEEE Computer Society, pp. 675–681.

Dewaele, G. and Cani, M.-P. 2004. Interactive global and local deformations for virtual clay. *Graphical Models* 66(6), pp. 352–369.

Doi, A., Fujiwara, S., Matsuda, K. and Kameda, M. 2002. 3D volume extraction and mesh generation using energy minimization techniques. In *Proceedings of the First International Symposium on 3D Data Processing Visualization and Transmission*, 3DPVT '02, IEEE Computer Society, pp. 83–86.

Du, H. and Qin, H. 2000*a*. Direct manipulation and interactive sculpting of pde surfaces. *Computer Graphics Forum* 19(3), pp. 261–270.

Du, H. and Qin, H. 2000*b*. Dynamic pde surfaces with flexible and general geometric constraints. In *Proceedings of the 8th Pacific Conference on Computer Graphics and Applications*, PG '00, IEEE Computer Society, pp. 213–224.

Du, H. and Qin, H. 2001. Integrating physics-based modeling with PDE solids for geometric design. In *Proceedings of the 9th Pacific Conference on Computer Graphics and Applications*, PG '01, IEEE Computer Society, pp. 198–207.

Du, H. and Qin, H. 2005. A shape design system using volumetric implicit PDEs. In *ACM SIGGRAPH Courses*.

Du, H. and Qin, H. 2007. Free-form geometric modeling by integrating parametric and implicit PDEs. *IEEE Transactions on Visualization and Computer Graphics* 13(3), pp. 549–561.

Eberhardt, B., Weber, A. and Strasser, W. 1996. A fast, flexible, particle-system model for cloth draping. *IEEE Computer Graphics and Applications* 16(5), pp. 52–59.

Eberly, D. and Shoemake, K. 2004. *Game physics*. Elsevier/Morgan Kaufmann.

El-Kurdi, Y., Giannacopoulos, D. and Gross, W. 2007. Hardware acceleration for finite-element electromagnetics: Efficient sparse matrix floating-point computations with fpgas. *IEEE Transactions on Magnetics* 43(4), pp. 1525–1528.

Essa, I., Sclaroff, S. and Pentland, A. 1993. Physically-based modeling for graphics and vision. Directions in Geometric Computing, Edited by R Marin, Information Geometers.

Etzmuß, O., Gross, J. and Strasser, W. 2003. Deriving a particle system from continuum mechanics for the animation of deformable objects. *IEEE Transactions on Visualization and Computer Graphics* 9(4), pp. 538–550.

FarField Technology n.d.. Fitting surfaces to point clouds [Online]. Available from: `http://www.farfieldtechnology.com/products/toolbox/pointcloud/`, [Accessed November 2012].

Farin, G. 2002. *Curves and surfaces for CAGD: a practical guide*. Morgan Kaufmann Publishers Inc.

Ferley, E., Cani, M.-P. and Gascuel, J.-D. 2000. Practical volumetric sculpting. *The Visual Computer* 16(8), pp. 469–480.

Ferley, E., Cani, M.-P. and Gascuel, J.-D. 2001. Resolution adaptive volume sculpting. *Graphical Models* 63(6), pp. 459 – 478.

Ferley, E., paule Cani, M. and dominique Gascuel, J. 1999. Virtual sculpture. Short Papers and Demos Eurographics.

Ferrier, N., Rowe, S. and Blake, A. 1994. Real-time traffic monitoring. In *Proceedings of the Second IEEE Workshop on Applications of Computer Vision*, pp. 81–88.

Foley, J. 1996. *Computer Graphics: Principles and Practice, Second Edition in C*. Addison-Wesley Pub.

Funck, W. V., Theisel, H. and Seidel, H.-P. 2006. Vector field based shape deformations. *ACM Transactions on Graphics* 25(3), pp. 1118–1125.

Gain, J. and Bechmann, D. 2008. A survey of spatial deformation from a user-centered perspective. *ACM Transactions on Graphics* 27(4), pp. 1–21.

Gain, J. E. and Dodgson, N. A. 1999. Adaptive refinement and decimation under free-form deformation. In *Eurographics UK*, EGUK '99, pp. 13–15.

Gain, J. and Marais, P. 2005. Warp sculpting. *IEEE Transactions on Visualization and Computer Graphics* 11(2), pp. 217–227.

Galyean, T. A. and Hughes, J. F. 1991. Sculpting: An interactive volumetric modeling technique. *Computer Graphics* 25(4), pp. 267–274.

Gao, Z. and Gibson, I. 2006. Haptic sculpting of multi-resolution B-Spline surfaces with shaped tools. *Computer Aided Design* 38(6), pp. 661–676.

Gibson, S. F. F., Perry, R. N., Rockwood, A. P. and Jones, T. R. 2000. Adaptively sampled distance fields: a general representation of shape for computer graphics. In *Proceedings of SIGGRAPH 00*, Annual Conference Series, ACM Press/Addison-Wesley, pp. 249–254.

Goldberg, D. 1991. What every computer scientist should know about floating-point arithmetic. *ACM Computer Surveys* 23(1), pp. 5–48.

Goldberg, D. 2001. What every computer scientist should know about floating-point arithmetic [Online]. Available from: `http://docs.oracle.com/cd/E19957-01/806-3568/ncg_goldberg.html`, [Accessed December 2012].

González-Hidalgo, M., i Capó, A. J., Mir, A. and Nicolau-Bestard, G. 2013. A tool for analytical simulation of B-Splines surface deformation. *Computer-Aided Design* 45(2), pp. 168 – 179.

Gross, M. 2009. Point based graphics – state ofthe art and recent advances. In *SIGGRAPH 2009 Course Notes*.

Hauth, M. and Etzmuß, O. 2001. A high performance solver for the animation of deformable objects using advanced numerical methods. *Computer Graphics Forum* 20(3), pp. 319–328.

Hearn, D. and Baker, P. 1994. *Computer Graphics.* Prentice Hall PTR.

Hsu, W. M., Hughes, J. F. and Kaufman, H. 1992. Direct manipulation of free-form deformations. *Computer Graphics* 26(2), pp. 177–184.

Hu, W., Tan, T., Wang, L. and Maybank, S. 2004. A survey on visual surveillance of object motion and behaviors. *IEEE Transactions on Systems, Man and Cybernetics* 34(3), pp. 334–352.

Hua, J. and Qin, H. 2003. Free-form deformations via sketching and manipulating scalar fields. In *Proceedings of the Eighth ACM Symposium on Solid Modeling and Applications*, SM '03, ACM, pp. 328–333.

Hua, J. and Qin, H. 2004. Haptics-based dynamic implicit solid modeling. *IEEE Transactions on Visualization and Computer Graphics* 10(5), pp. 574 –586.

Hughes, T., Cottrell, J. and Bazilevs, Y. 2005. Isogeometric analysis: Cad, finite elements, nurbs, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering* 194(39Ű41), pp. 4135 – 4195.

Hughes, T., Reali, A. and Sangalli, G. 2010. Efficient quadrature for NURBS-based isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering* 199(5–8), pp. 301 – 313.

Hunter, P. and Pullan, A. 2005. FEM/BEM Notes [Online]. Available from: `http://www.cs.rutgers.edu/~suejung/fembemnotes.pdf`, [Accessed December 2012].

Ivins, J. P. 1996. Statistical Snakes: Active Region Models. PhD thesis, University of Sheffield.

Ivins, J. and Porrill, J. 1995. Active region models for segmenting textures and colours. *Image and Vision Computing* 13(5), pp. 431–438.

Ivins, J. and Porrill, J. 2000. Everything you always wanted to know about snakes (but were afraid to ask). Technical report, Artificial Intelligence Vision Research Unity, University of Sheffield.

James, D. L. and Pai, D. K. 1999. ArtDefo - accurate real time deformable objects. In *Proceedings of SIGGRAPH 99*, Annual Conference Series, ACM Press/Addison-Wesley, pp. 65–72.

Jordan, D. and Smith, P. 2002. *Mathematical Techniques: An Introduction for the Engineering, Physical, and Mathematical Sciences*. Oxford University Press, USA.

Kalra, P., Mangili, A., Thalmann, N. M. and Thalmann, D. 1991. 3D interactive free form deformations for facial expressions. In *Proceedings of COMPUGRAPHICS '91*, pp. 129–141.

Kalra, P., Mangili, A., Thalmann, N. M. and Thalmann, D. 1992. Simulation of facial muscle actions based on rational free form deformations. *Computer Graphics Forum* 11(3), pp. 59–69.

Kashyap, S. 2008. Splat based raytracing. Technical report, Department of Computer Science and Engineering, Indian Institute of Technology Bombay, Mumbai.

Kashyap, S., Goradia, R., Chaudhuri, P. and Chandran, S. 2010. Real time ray tracing of point-based models. In *Proceedings of the Symposium on Interactive 3D Graphics and Games*, I3D '10, ACM, pp. 4:1–4:1.

Kass, M., Witkin, A. and Terzopoulos, D. 1987. Snakes: Active Contour Models. *International Journal of Computer Vision* 1(4), pp. 321–331.

Kazinnik, R. and Elber, G. 1997. Orthogonal decomposition of non-uniform bspline spaces using wavelets. *Computer Graphics Forum* 16, pp. C27–C38.

Knopf, G. K. and Igwe, P. C. 2005. Deformable mesh for virtual shape sculpting. *Robotics and Computer-Integrated Manufacturing* 21(4–5), pp. 302 – 311.

Kobbelt, L. and Botsch, M. 2004. A survey of point-based techniques in computer graphics. *Computers & Graphics* 28(6), pp. 801–814.

Kuhnapfel, U., Cakmak, H. and Maass, H. 2000. Endoscopic surgery training using virtual reality and deformable tissue simulation. *Computers & Graphics* 24(5), pp. 671–682.

Lamousin, H. J. and Waggenspack Jr., W. N. 1994. NURBS-based free-form deformations. *IEEE Computer Graphics and Applications* 14(6), pp. 59–65.

Lazarus, F., Coquillart, S. and Jancene, P. 1994. Axial deformations: An intuitive deformation technique. *Computer-Aided Design* 26(8), pp. 607–613.

Levoy, M., Pulli, K., Curless, B., Rusinkiewicz, S., Koller, D., Pereira, L., Ginzton, M., Anderson, S., Davis, J., Ginsberg, J., Shade, J. and Fulk, D. 2000. The digital Michelangelo project: 3D scanning of large statues. In *Proceedings of SIGGRAPH 00*, Annual Conference Series, ACM Press/Addison-Wesley, pp. 131–144.

Levoy M., W. R. 1985. The use of points as a display primitive. Technical Report 85-022, Computer Science Department, UNC-Chapel Hill.

Liu, G. R. 2003. *Mesh free methods: Moving beyond the finite element method*. CRC Press.

Liu, Y., shaohui Jiao, Wu, W. and De, S. 2008. Gpu accelerated fast fem deformation simulation. In *IEEE Asia Pacific Conference on Circuits and Systems, 2008.*, APCCAS 2008.

Logan, D. 2011. *A First Course in the Finite Element Method: SI Edition*. Cengage Learning.

Lorensen, W. E. and Cline, H. E. 1987. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics* 21(4), pp. 163–169.

Ma, W. 2005. Subdivision surface for CAD - an overview. *Computer-Aided Design* 37(7), pp. 693–709.

MacCracken, R. and Joy, K. I. 1996. Free-form deformations with lattices of arbitrary topology. In *Proceedings of SIGGRAPH 96*, Annual Conference Series, ACM, pp. 181–188.

Massie, T. 1998. A tangible goal for 3D modeling. *IEEE Computer Graphics and Applications* 18(3), pp. 62–65.

Matthew Ward, W. C. D. n.d.. An overview of metaballs/blobby objects [Online]. Available From: `http://www.siggraph.org/education/materials/HyperGraph/modeling/metaballs/metaballs_mward.html`, [Accessed November 2012].

McDonnell, K. and Qin, H. 2000. Dynamic sculpting and animation of free-form subdivision solids. In *Proceedings of Computer Animation*, CA '00, IEEE Computer Society, pp. 126–133.

McDonnell, K. T., Chang, Y.-S. and Qin, H. 2005. Digitalsculpture: a subdivision-based approach to interactive implicit surface modeling. *Graphical Models* 67(4), pp. 347–369.

McDonnell, K. T. and Qin, H. 2002. Virtual clay: Haptics-based deformable solids of arbitrary topology. In *Proceedings of the Second International Workshop on Articulated Motion and Deformable Objects*, AMDO '02, Springer–Verlag, pp. 1–20.

McDonnell, K. T. and Qin, H. 2007*a*. A novel framework for physically based sculpting and animation of free-form solids. *The Visual Computer* 23(4), pp. 285–296.

McDonnell, K. T. and Qin, H. 2007*b*. PB-FFD: A point-based technique for free-form deformation. *Journal of Graphics Tools* 12(3), pp. 25–41.

McDonnell, K. T., Qin, H. and Wlodarczyk, R. A. 2001. Virtual clay: a real-time sculpting system with haptic toolkits. In *Proceedings of the Symposium on Interactive 3D Graphics*, I3D '01, ACM, pp. 179–190.

McInerney, T. 1992. Finite element techniques for fitting deformable models to 3D data. Master's thesis, Dept. of Computer Science, University of Toronto.

McInerney, T. and Terzopoulos, D. 1996. Deformable models in medical image analysis: A survey. *Medical Image Analysis* 1(2), pp. 91–108.

McInerney, T. and Terzopoulos, D. 2000. T-Snakes: Topology Adaptive Snakes. *Medical Image Analysis* 4(2), pp. 73–91.

Meegama, R. G. N. and Rajapakse, J. C. 2003. NURBS snakes. *Image and Vision Computing* 21, pp. 551–562.

Menet, S., Saint-Marc, P. and Medioni, G. 1990. B-Snakes: Implementation and application to stereo. In *Proceedings of DARPA*, pp. 720Ű–726.

Mitchell, A. and Griffiths, D. 1980. *The finite difference method in partial differential equations*. Wiley.

Molloy, D. and Whelan, P. F. 2000. Active-meshes. *Pattern Recognition Letters* 21(12), pp. 1071–1080.

Montagnat, J., Delingette, H. and Ayache, N. 2001. A review of deformable surfaces: Topology, geometry and deformation. *Image and Vision Computing* 19(14), pp. 1023–1040.

Moore, P., Boyle, E. and Molloy, D. 2005. Construction of 3D human models from two orthographic projections. In *Proceedings of EuroGraphics Ireland (EGIre)*.

Moore, P. and Molloy, D. 2006. Active surface meshes for intuitive 3-D computer graphics shape deformation and modelling. In *Proceedings of the Irish Machine Vision and Image Processing Conference (IMVIP)*.

Moore, P. and Molloy, D. 2007*a*. A survey of computer-based deformable models. In *Proceedings of the International Machine Vision and Image Processing Conference (IMVIP)*.

Moore, P. and Molloy, D. 2007*b*. Virtual sculpting: Active contour models for computer graphics shape modelling and deformation. In *Proceedings of EuroGraphics Ireland (EGIre)*.

Moore, P. and Molloy, D. 2008*a*. Active contour models for computer graphics shape modelling and deformation. In *Proceedings of Vision, Graphics, and Visualisation (VMV)*.

Moore, P. and Molloy, D. 2008*b*. Virtual sculpting: Active surface meshes for intuitive computer graphics shape modelling and deformation. RINCE Research Seminar Series, Dublin, Ireland, 2008.

Moore, P. and Molloy, D. 2012. Active B-Spline surface models for intuitive 3D virtual sculpting. In *Proceedings of the Spring Conference on Computer Graphics (SCCG)*.

Moore, P. and Molloy, D. 2013. Efficient energy evaluations for active B-Spline/NURBS surfaces. *Computer-Aided Design* , In Press, Accepted Manuscript.

Moreton, H. P. and Séquin, C. H. 1992. Functional optimization for fair surface design. *Computer Graphics* 26(2), pp. 167–176.

Mullenhoffe, C. J. 1998. Physically-based B-Spline surface sculpting. Master's thesis, The University of Utah.

Müller, M., Dorsey, J., McMillan, L., Jagnow, R. and Cutler, B. 2002. Stable real-time deformations. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '02, ACM, New York, NY, USA, pp. 49–54.

N. Sukumar, E. A. M. 2006. Recent advances in the construction of polygonal finite element interpolants. *Archives of Computational Methods in Engineering* 13, pp. 129–163.

Nayroles, B., Touzot, G. and Villon, P. 1992. Generalizing the finite element method: Diffuse approximation and diffuse elements. *Computational Mechanics* 10, pp. 307–318.

Nealen, A., Mueller, M., Keiser, R., Boxerman, E. and Carlson, M. 2006. Physically based deformable models in computer graphics. *Computer Graphics Forum* 25(4), pp. 809–836.

Nishimura, H., Hirai, M., Kawai, T., Kawata, T., Shirakawa, I. and Omura, K. 1985. Object modeling by distribution function and a method of image generation. *Transactions of the Institute of Electronics and Communcation Engineers of Japan* J68-D(4), pp. 718–725.

O'Brien, J. F. and Hodgins, J. K. 1999. Graphical modeling and animation of brittle fracture. In *Proceedings of SIGGRAPH 99*, Annual Conference Series, ACM Press / Addison-Wesley, pp. 137–146.

Öztireli, A. C., Guennebaud, G. and Gross, M. 2009. Feature preserving point set surfaces based on non-linear kernel regression. *Computer Graphics Forum* 28(2), pp. 493–501.

P. Kaufmann, S. Martin, M. B. M. G. 2009. Flexible simulation of deformable models using discontinuous galerkin fem. *Journal of Graphical Models* 71(4), pp. 123–167.

Parent, R. E. 1977. A system for sculpting 3D data. *Computer Graphics* 11(2), pp. 138–147.

Pauly, M., Keiser, R., Kobbelt, L. P. and Gross, M. 2003. Shape modeling with point-sampled geometry. *ACM Transactions on Graphics* 22(3), pp. 641–650.

Perry, R. N. and Gibson, S. F. F. 2001. Kizamu: a system for sculpting digital characters. In *Proceedings of SIGGRAPH 01*, Annual Conference Series, ACM Press / Addison-Wesley, pp. 47–56.

Pfister, H., Zwicker, M., van Baar, J. and Gross, M. 2000. Surfels: surface elements as rendering primitives. In *Proceedings of SIGGRAPH 00*, Annual Conference Series, ACM Press / Addison-Wesley, pp. 335–342.

Piegl, L. A. 2005. Ten challenges in computer-aided design. *Computer-Aided Design* 37(4), pp. 461 – 470.

Piegl, L. and Tiller, W. 1997. *The NURBS Book*. 2nd ed. Springer-Verlag New York, Inc., New York, NY, USA.

Platt, S. M. and Badler, N. I. 1981. Animating facial expressions. *Computer Graphics* 15(3), pp. 245–252.

Press, W. H., Teukolsky, S. A., Vetterling, W. T. and Flannery, B. P. 2007. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press.

Pungotra, H., Knopf, G. K. and Canas, R. 2010. Merging multiple B-Spline surface patches in a virtual reality environment. *Computer-Aided Design* 42(10), pp. 847–859.

Qin, H., Mandal, C. and Vemuri, B. 1998. Dynamic Catmull-Clark subdivision surfaces. *IEEE Transactions on Visualization and Computer Graphics* 4(3), pp. 215 –229.

Qin, H. and Terzopoulos, D. 1994. Physics-based NURBS Swung Surfaces. In *Proceedings of the IMA Conference on the Mathematics of Surfaces*, Clarendon Press, pp. 267–290.

Qin, H. and Terzopoulos, D. 1996. D-NURBS: A Physics-Based Framework for Geometric Design. *IEEE Transactions on Visualization and Computer Graphics* 2(1), pp. 85–96.

Raviv, A. and Elber, G. 1999. Three dimensional freeform sculpting via zero sets of scalar trivariate functions. In *Proceedings of the Fifth ACM Symposium on Solid Modeling and Applications*, SMA '99, ACM, pp. 246–257.

Reddy, J. 2008. *An Introduction to Continuum Mechanics*. Cambridge University Press.

Reeves, W. T. 1983. Particle systems: a technique for modeling a class of fuzzy objects. *ACM Transactions on Graphics* 2(2), pp. 91–108.

Requicha, A. and Voelcker, H. 1982. Solid modeling: A historical summary and contemporary assessment. *IEEE Computer Graphics and Applications* 2(2), pp. 9 –24.

Ricci, A. 1973. A constructive geometry for computer graphics. *The Computer Journal* 16(2), pp. 157–160.

Richens, D., Rougon, N., Bloch, I. and Mousseaux, E. 1992. Segmentation by deformable contours of MRI sequences of the left ventricle for quantitative analysis. In *Proceedings of the International Conference on Image Processing and its Applications*, pp. 393–396.

Rusinkiewicz, S. and Levoy, M. 2000. Qsplat: a multiresolution point rendering system for large meshes. In *Proceedings of SIGGRAPH 00*, Annual Conference Series, ACM Press/Addison-Wesley, pp. 343–352.

Rypl, D. 2003. Approaches to discretization of 3D surfaces. In *CTU Reports*, Vol. 7, CTU Publishing House, Prague, Czech Republic.

Rypl, D. and Patzák, B. 2012. Study of computational effinciency of numerical quadrature schemes in the isogeometric analysis. In *Proceedings of the 18th International Conference on Engineering Mechanics*, EM '12, pp. 1135–1143.

Schein, S. and Elber, G. 2004. Discontinuous Free Form Deformations. In *Proceedings of the 12th Pacific Conference on Computer Graphics and Applications*, PG '04, IEEE Computer Society, pp. 227–236.

Schmitt, B., Pasko, A. and Schlick, C. 2004. Constructive sculpting of heterogeneous volumetric objects using trivariate B-Splines. *The Visual Computer* 20(2), pp. 130–148.

Schumaker, L. 2007. *Spline Functions: Basic Theory*. Cambridge University Press.

Sederberg, T. W. and Parry, S. R. 1986. Free-form deformation of solid geometric models. *Computer Graphics* 20(4), pp. 151–160.

Sederberg, T. W., Zheng, J., Bakenov, A. and Nasri, A. 2003. T-splines and t-nurccs. *ACM Transactions on Graphics* 22(3), pp. 477–484.

Segerlind, L. 1984. *Applied Finite Element Analysis*. John Wiley and Sons, New York.

Sela, G., Subag, J., Lindblad, A., Albocher, D., Schein, S. and Elber, G. 2006. Real-time haptic incision simulation using FEM-based discontinuous free form deformation. In *Proceedings of the ACM Symposium on Solid and Physical Modeling*, SPM '06, ACM, pp. 75–84.

Shabana, A. 2011. *Computational Continuum Mechanics*. Cambridge University Press.

Singh, K. and Eugene, F. 1998. Wires: a geometric deformation technique. In *Proceedings of SIGGRAPH 98*, Annual Conference Series, ACM Press/Addison-Wesley, pp. 405–414.

Stanculescu, L., Chaine, R. and Cani, M.-P. 2011. Freestyle: Sculpting meshes with self-adaptive topology. *Computers & Graphics* 35(3), pp. 614–622.

Stoer, J., Bulirsch, R., Bartels, R., Gautschi, W. and Witzgall, C. 2002. *Introduction to Numerical Analysis*. Springer.

Strang, G. 1986. *Introduction to Applied Mathematics*. Wellesley-Cambridge Press.

Strang, G. and Fix, G. 1973. *An analysis of the finite element method*. Prentice Hall.

Teran, J., Blemker, S., Hing, V. N. T. and Fedkiw, R. 2003. Finite volume methods for the simulation of skeletal muscle. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '03, Eurographics Association, pp. 68–74.

Terzopoulos, D. and Fleischer, K. 1988. Deformable models. *The Visual Computer* 4(6), pp. 306–331.

Terzopoulos, D., Platt, J., Barr, A. and Fleischer, K. 1987. Elastically deformable models. *Computer Graphics* 21(4), pp. 205–214.

Terzopoulos, D. and Qin, H. 1994. Dynamic NURBS with geometric constraints for interactive sculpting. *ACM Transactions on Graphics* 13(2), pp. 103–136.

Tiilikainen, N. P. 2007. A comparative study of active contour snakes. Technical report, University of Copenhagen.

Van Gelder, A. 1998. Approximate simulation of elastic membranes by triangulated spring meshes. *Journal of Graphics Tools* 3(2), pp. 21–42.

Varkonyi-Koczy, A., Rovid, A. and Varkonyi, P. 2007. Intelligent 3d car-body deformation modeling. In *Proceedings of the Second International Conference on Systems*, ICONS '07, pp. 48–48.

Vassilev, T. I. 1997. Interactive sculpting with deformable nonuniform B-Splines. *Computer Graphics Forum* 16(4), pp. 191–199.

Vieren, C., Cabestaing, F. and Postaire, J.-G. 1995. Catching moving objects with snakes for motion tracking. *Pattern Recognition and Letters* 16(7), pp. 679–685.

Wang, S. W. and Kaufman, A. E. 1995. Volume sculpting. In *Proceedings of the 1995 Symposium on Interactive 3D Graphics*, I3D '95, ACM, pp. 151–ff.

Waters, K. 1987. A muscle model for animation three-dimensional facial expression. *Computer Graphics* 21(4), pp. 17–24.

Welch, W. and Witkin, A. 1992. Variational surface modeling. *Computer Graphics* 26(2), pp. 157–166.

Welch, W. and Witkin, A. 1994. Free-form shape design using triangulated surfaces. In *Proceedings of SIGGRAPH 94*, Annual Conference Series, ACM, pp. 247–256.

Wyvill, G., Mcpheeters, C. and Wyvill, B. 1986. Data structure for soft objects. *The Visual Computer* 2(4), pp. 227–234.

Xu, C. and Prince, J. 1997. Gradient vector flow: a new external force for snakes. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, CVPR '97, IEEE Computer Society, pp. 66–77.

Zwicker, M., Pfister, H., van Baar, J. and Gross, M. 2001. Surface splatting. In *Proceedings of SIGGRAPH 01*, Annual Conference Series, ACM Press/Addison-Wesley, pp. 371–378.