Visual Simulation of Deformable Models

Dissertation
der Fakultät für Informations- und Kognitionswissenschaften
der Eberhard-Karls-Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
Dipl.-Inform. Dipl.-Math. Michael Hauth
aus Hechingen

Tübingen
2004

Tag der mündlichen Qualifikation:	21.7.2004
Dekan:	Prof. Dr. Ulrich Güntzer
1. Berichterstatter:	Prof. Dr.-Ing. Dr.-Ing. E.h. Wolfgang Straßer
2. Berichterstatter:	Prof. Dr. Nadia Magnenat-Thalmann
	(MIRALab, Université de Genève)
3. Berichterstatter:	Prof. Dr. Peter Hauck

To my grandparents
who taught me to
appreciate good handiwork.

Abstract

Computer animations are an essential part of today's visual production pipeline, for feature animated films and video games. By moving from static to dynamic scenes, immersion in virtual environments is greatly enhanced. Most of these animations however are concerning rigid or articulated bodies, and are generated manually by an artist or off-line. The interactive animation of deformable objects still is a challenging task, requiring high computational resources. Moreover, interactive environments for dressing up with virtual clothing or training surgeons on unaccustomed procedures pose higher requirements on fidelity as simple plausibility, which we denote by visual simulation.

This thesis presents a framework for the visual simulation of deformable objects, which is fast enough to be integrated into virtual environments. The modelling of these objects is based on continuum mechanics, yielding a better accuracy than the techniques commonly used in graphics. An important aspect of modelling are material laws. Measurements have shown viscoelasticity to be a distinctive characteristic of biological soft tissue, the hysteresis of cloth is a well known observation in standard Kawabata experiments. We introduce the modelling of viscoelasticity to graphics, which integrates smoothly into the continuum dynamical setting, and present the first interactive implementation of a viscoelastic model.

Chapter 2 introduces the basic concepts of this approach, which results in a partial differential equation; the following chapter discusses the numerical methods we employ for its solution. We use the method of lines to solve this equation. Hence, as a first step the equation is discretised in space using finite elements on a tetrahedral mesh of the object. This reduces the problem to the solution of an ordinary differential equation. The efficient time integration of this equation is vital for the performance of the application. The implicit methods we introduce for this task are well suited for two-dimensional structures such as cloth as well as for three-dimensional soft objects. Implicit integration requires the solution of large systems of equations. Hence, the efficiency of these schemes again critically depends on the linear solver; we propose to use several methods new to the field.

Two applications are presented in chapter 4 and 5 respectively, a core for cloth simulation using a simpler space discretization method, and a soft object simulator, assembling all the techniques we described in a set of flexible building blocks. The latter application allows the real-time simulation of soft objects composed of up to several thousand tetrahedral elements; the blocks can be configured to trade execution speed for approximation accuracy and hence to match the requirements of the actual area of usage.

Zusammenfassung

In der Produktion von computer-generierten Filmen und Videospielen spielen Animationen heutzutage eine tragende Rolle. Virtuelle Umgebungen gewinnen durch den Übergang von statischen zu dynamischen Szenarien an Tiefe, realistische Dynamik erleichtert dem Benutzer das Eintauchen in die virtuelle Realität. Animationen betreffen jedoch überwiegend starre Körper oder gekoppelte Starrkörpersysteme, Bewegungen werden häufig manuell erzeugt. Die interaktive Deformation weicher Objekte, zugeschnitten auf die Bedürfnisse der Computergraphik, ist noch immer eine Herausforderung, insbesondere aufgrund des hohen Rechenaufwandes, den sie benötigt. Zudem stellen spezielle Anwendungen wie beispielsweise eine virtuelle Schneiderstube oder ein virtuelles Operationsfeld zum Training minimal invasiver Eingriffe höhere Anforderungen an die Genauigkeit der Rechnung. Diese gehen über die simple Plausibilität von traditioneller Animation hinaus, die letztendlich einem Animator lediglich eine sinnvolle Steuerung der Bewegung ermöglichen muss. Dies prägte den Begriff visuelle Simulation. Visuelle Simulationen haben die optische, haptische oder akustische Ausgabe und die Interaktion mit einem Anwender als primäres Ziel, während die Simulationen der klassischen Ingenieurwissenschaften vor allem technische Entwurfsentscheidungen erleichtern sollen.

In dieser Arbeit stellen wir die wichtigsten Komponenten für die visuelle Simulation von deformierbaren Körpern vor. Die resultierende, interaktive Anwendung ist schnell genug, um eine Integration in eine virtuelle Umgebung zu erlauben. Deformierbare Materialien werden nach den Grundsätzen der Kontinuumsmechanik modelliert, dies ermöglicht, wie wir sehen werden, eine wesentlich bessere Näherung als die sonst in der Grafik üblichen Verfahren. Ein wichtiger Aspekt der Modellierung ist die Abbildung von viskoelastischen Parametern, da Messungen gezeigt haben, dass Viskoelastizität eine grundlegende Charakteristik von biologischem Weichgewebe ist. Ebenso sind Hysterese-Schleifen, Resultat einer Kombination aus viskosen und elastischen Eigenschaften, typisch für die Kawabata Messungen von Textilien. Viskoelastizität lässt sich nahtlos in die kontinuumsmechanische Beschreibung einfügen. Diese Arbeit führt die Modellierung von Viskoelastizität in die Computergraphik ein, und präsentiert die erste Implementierung eines viskoelastischen Materials im Rahmen einer interaktiven Anwendung.

Zunächst werden die wichtigsten physikalischen und mathematischen Grundlagen vorgestellt. Als Ergebnis der Modellierung erhalten wir eine partielle Differentialgleichung, die numerisch gelöst werden muss. Dazu wird die „method of lines“ verwendet, das heißt die Gleichung wird zunächst im Raum und anschließend in der Zeit diskretisiert. Zur Raumdiskretisierung wird die Methode der Finiten Elemente genutzt, die es erlaubt, die Lösung durch eine gewöhnliche Differentialgleichung auf einem Tetraedernetz anzunähern. Die effiziente numerische Lösung dieser Gleichung ist ein wichtiger Bestandteil eines schnellen Simulators. Zu diesem Zweck werden implizite Integrationsverfahren verwendet, die sich für die interaktive Simulation sowohl von zweidimensionalen Textilmodellen, als auch von deformierbaren, volumenbehafteten Objekten besonders eignen, allerdings vor allem für letztere Anwendung (noch) nicht verbreitet sind. Die Effizienz eines solchen Verfahrens wird vor allem durch einen schnellen Lösungsalgorithmus für lineare Gleichungen bestimmt. Aus diesem Grund adaptieren wir eine Reihe von direkten und it-

erativen Methoden, die innerhalb dieses Forschungsgebietes bisher noch nicht eingesetzt worden sind.

Die letzen beiden Kapitel stellen zwei Anwendungen vor, ein numerisches Kernmodul zur Simulation von Kleidung und einen interaktiven Simulator für weiche Objekte. Der Textilsimulator verwendet einen einfacheren Ansatz zur Raumdiskretisierung, die Applikation für dreidimensionale Modelle baut auf allen hier vorgestellten Techniken auf. Sie gestattet es, Netze mit bis zu mehren tausend Tetraedern in Echtzeit zu simulieren. Eine flexible Architektur erlaubt es, verschienende Module zu einem für den jeweiligen Anwendungsfall optimalen Kompromiss zwischen Schnelligkeit und Genauigkeit zu kombinieren.

Preface

The research work for this Ph.D. thesis was carried out within the DFG *ElastoMedTrain* project at the WSI/GRIS, University of Tübingen and during an occupation as a Research Assistant at the section de mathématiques, Faculté des sciences de l'Université de Genève. The ElastoMed-Train project includes a second research position, located at the Section for Minimally Invasive Surgery, within the Department of General Surgery at the University of Tübingen, which was responsible for building instruments for acquiring data on the visco-elastic behaviour of biological soft tissue. It has been held by Dr. J. Gross, Dipl.-Phys. N. Warncke and Dipl.-Ing. D. Kowalski.

The printed and electronic version of this document is augmented by several animations, which are reachable by selecting the icons displayed in the margin within the electronic version and directly from the media folder. A compressed archive of this folder is located at http://www.gris.uni-tuebingen.de/~mhauth. The clips have been encoded using Windows Media 9; player software is available for Windows, Mac OS, Linux, and BSD.

Acknowledgements

First, I would like to thank my advisor Prof. Wolfgang Strasser for his support and the encouragement I received for my research. I also would like to thank Prof. N. Magnenat-Thalmann and Prof. P. Hauck who agreed to be part of my graduation committee. I am especially grateful to Prof. C. Lubich for many discussions, as well as to Prof. E. Hairer and Prof. G. Wanner for their hospitality and warm reception in Geneva. I am thankful to Prof. H. Yserentant for his aid with continuum mechanics. In addition, I am indebted to Prof. B. Eberhardt for always providing me with valuable comments and advice.

Furthermore, I wish to thank all my colleagues at WSI/GRIS, especially M. Wand and S. Gumhold for discussing countless ideas, my roommates R. Sonntag and F. Hanisch for a pleasant environment, and M. Wacker for his help in proofreading. Moreover, I want to show my appreciation to J. Gross, N. Warncke, and D. Kowalski for a fruitful cooperation. Special thanks go to R. Kuchar and T. Schairer for modelling the liver and toy objects and their priceless artistic assistance in multimedia postproduction.

Finally yet importantly, I would like to express my gratitude for always backing me up to my beloved girlfriend Melanie Jesse who sacrificed numerous evenings and weekends, while I was working overtime or attending conferences. These acknowledgements would not be complete without thanking my family, always supporting me throughout my studies.

Contents

List of Figures

List of Tables

List of Animations

Chapter 1

Introduction

We hope that, since the endless complexity and variety of natural phenomenon are caused by few basic elements and laws, their visual simulation can be achieved using few basic primitives and algorithms. We are still far from that goal, but then again we do not have as many processors or as much time as nature does.

A. Fournier and W. T. Reeves,
Guest Editors' Introduction: Special Issue on the Modeling of Natural Phenomena, ACM Transactions on Graphics, July 1987.

1.1 Motivation and Problem Setting

Deformable materials are ubiquitous, being an inherent part of our body and our environment. Consequently, their simulation is a key aspect in the creation of a digital mock-up of our everyday world, one of the ultimate goals of computer graphics. Soft objects find their application in virtual textiles and biological materials for virtual surgery, in the animation of virtual creatures and facial expressions. The goal of visual simulation is to provide the basis for visual, haptic, or even sound rendering of the results, usually at interactive rates. This is a fundamental distinction from a technical simulation, which aims at providing information to aid constructions or to gain insight into processes.

Two of the applications will be considered in more detail: simulation of cloth and of biological soft tissue. The simulation of cloth, introduced to graphics in 1986 by Weil using a geometric method [Wei86], today serves for dressing up virtual characters [CYMTT92] and virtual customers for e-commerce and made-to-measure wear [CSMT03, WKK+04]. The real-time simulation of visco-elastic soft tissue is a central issue and a major difficulty in developing a training platform for endoscopic surgery [SBD+00]. Minimal invasive interventions pose high and unaccustomed demands to the hand-eye coordination of the surgeon, which is currently trained by Pelvi-trainers using plastic or animal tissues, and, due to ethical reasons very rarely, by animal experiments. These trainers fail to provide the realism of the real-world situation, which the surgeon faces, sooner or later. Hence, there is a considerable interest in virtual-reality training systems, which at the moment however do not satisfy the expectations of the professional user.

Computer graphics is not the only discipline that is concerned with the investigation of deformable materials. In fact, mathematics, physics, and engineering have a substantially longer tradition of examining and modelling the nature of deformation, reaching back to Galileo Galilei

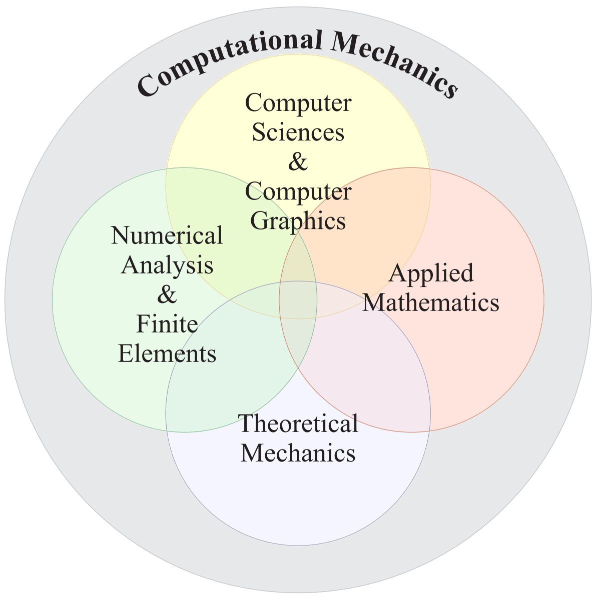

Computational Mechanics
Computer Sciences & Computer Graphics
Numerical Analysis & Finite Elements
Applied Mathematics
Theoretical Mechanics

Figure 1.1: Disciplines contributing to computational mechanics.

in the sixteenth century. Today computational mechanics is located at the intersection of computer sciences, applied mathematics, numerical analysis, finite element methods, and classical mechanics. In contrast to these disciplines, graphics has always put a special emphasis on interactivity and speed, willing to trade it for the fidelity of the simulation. Therefore, besides adopting the work of computational mechanics, computer graphics has developed its own fast and approximative techniques to deform solid models, i.e. geometrical and global deformations, finite automata, and mass-spring systems.

Based on the goal to develop a soft tissue kernel, which poses a stricter emphasis on fidelity than usual, a decision about the modelling of the soft objects had to be made at a very early stage. In computational mechanics, 'engineering accuracy' usually is defined to be 1% in displacements and 10% in stress. We rate the requirements of visual simulation at a 10% displacement error, which also seems to be acceptable for interactive medical training simulations. O'Sullivan et. al. [ODGK03] even proposed to use perceptual metrics to assess the quality of simulations and animations. While we essentially agree with the authors that animations are acceptable as long as the user does not notice a difference, for medical applications eventually liability will be questioned, hence medical simulations are explicitly excluded in their study and we rather keep a more classic error metric. From the traditional portfolio of graphics, these considerations leave mass-spring systems only, as they at least have a physical basis. Indeed several soft tissue kernels have been built upon masses and springs. After some experiments though, we made the experience, that mass-spring systems are not only quantitatively, but also qualitatively different to a continuum, which will be demonstrated, and hence could not meet our requirements. This is caused by the central weakness of mass-spring systems, that they provide only one constant to describe the material of a deformable solid. However even the most simple continuum mechanical description of a linear elastic, isotropic and homogeneous solid provides two constants, Young's modulus and Poisson's ratio[1]. For other animation tasks, this is less of an argument,

[1]Interestingly, exactly this point, the 'multi-constant' vs. 'rari-constant' question split the field for several decades (cf. [Lov27]). Advances in experiments and measurement devices slowly calmed the battle towards the multi-constant

as these are usually not focused on exact material properties but on meaningful parameters that allow an animator to control the result in a predictable manner. On the other hand, there are effects like volume preservation and visco-elasticity that mass-spring systems cannot model, so even for certain animation problems, they might prove to be insufficient. As a result, we chose to employ the toolbox provided by computational mechanics.

This choice is not without precedent. Some of the work on facial animation, also concerned with exact modelling, already used a continuum mechanical approach, though for non-interactive applications. In addition, interactive soft tissue modules based on linear, small strain elasticity have been presented, however this approach does not accommodate the deformations that arise during a typical operation. A notable exception is given by the framework developed in the Lasso research project [SBD+00], which yielded an interactive simulator based on non-linear finite elements. To achieve real-time performance however, a large parallel computer was needed, providing a sustained performance of 20 GFLOPS[2]. Nevertheless, this project proved the long-term feasibility of the method, though several more years will pass for this to be practicable on a single processor, assuming Moore's law will hold. In order to built a system for a single workstation, providing 1 GFLOP peak, different techniques had to be used.

The general structure of our approach is similar and typical for computational mechanics [Fun93]. In a first step, the object's mechanics is studied by measurements, which for biological tissue meant the development of new measurement devices, as done in the companion project ElastoMedTrain [GHEB01]. The focus of the project has been on the liver, because the tissue is comparably isotropic and cholecystectomy is the most common minimal invasive intervention. The continuum then is modelled as a partial differential equation. For its solution, we chose to apply the method of lines: to discretise the PDE in space using finite elements, and then to integrate the ordinary differential equation in time. Throughout this process, interactivity as a primary objective has always influenced our design decisions.

For cloth simulation, we directly started with an ordinary differential equation and only considered efficient time integration. Up to now, finite elements [EDC96] play only a minor role in this application, mass-spring systems, and particle systems dominate the field. As for soft tissue, fast and stable time integration plays a crucial role in building a fast simulator.

Why didn't we simply employ a commercial simulator for finite element mechanics? As the simulation should perform at interactive rates, the direct use of a commercial package is not feasible. We employ a commercial package for the computation of gold standards, which typically uses several hundred megabytes of memory, compared to a few dozen with our package, and is about one to two orders of magnitudes slower. The main reason for this is, that commercial packages are targeted at engineering requirements, providing a wide variety of easy exchangeable material laws and element types, whereas our application is optimised around a single element type and only a few materials. The implementation of a new material law takes a coding effort of several days; the change of the element type presumably would take several weeks.

The reason, finite elements are not so common in graphics, is the belief that they are more than an order of magnitude slower than mass-spring systems. We found, that with a proper implementation a mere factor of two to three remains, which should be weighted against the sound derivation that finite elements provide. Moreover, techniques like co-rotated strain allow even faster implementations of finite elements on the expense of a lower approximation order. Hence, we even consider the presented approach to be an alternative for animation tasks.

theory, however experiments concerning the issue were performed until the late 1960s.

[2]FLOP: floating point operations per second

1.2 Previous Work

Motivated by the goal to reproduce soft objects in a virtual environment, the subject has attracted considerable attention in computer graphics. Nowadays, the field is divided into several major working areas, among them are garment simulation, deformable tissue for facial animation and simulation, soft tissue for virtual surgery, and unspecific soft objects for animation purposes. As a result many of the components of our system, laid out in detail below, have some connections to previous published work that we will refer to in the appropriate sections. In this section, we describe the larger frame of prior and related work.

As our contribution concerns the modelling and simulation of deformable solids, applied in animation and virtual surgery, as well as efficient time integration for textile simulation, we will concentrate on these areas. For a survey addressing all aspects, we refer to the current books on cloth simulation by Volino and Thalmann [VMT00b], and Breen and House[HB00], as well as to the articles by Gibson and Mirtich [GM97], Delingette [Del98], Bro-Nielsen [BN98] and Szekely et al. [SBD$^+$00] covering animation and medical solid simulation.

1.2.1 Direct Modelling of Elastic Objects

The first step in the assembly of a soft object simulator is the choice of a representation for the elastic object. In the last two decades, several approaches have been considered.

Free Form Deformation (FFD) and Extensions Very early work, seeking for cheap and fast methods focused on geometric approaches. FFD is built upon free-form shapes, defined using Bezier-patches or splines. Barr [Bar84] presented the idea to apply transformation matrices like stretch, bend, or twist to geometric shapes. This idea has been extended to free form deformation models [SP86], now defining the transformation using a tensor product Bernstein basis. In its pure form the concept lacks a strict physical justification, which on the other hand is not necessary for applications like virtual sculpturing, where it received further attention [Coq90]. An important extension to unstructured lattices has been made by MacCracken and Joy [MJ96]. Faloutsos et al. [FvT97] provide more physical realism, introducing dynamics by employing a mass-spring network as a control mesh. Recently this has been generalised using a finite element model on the control lattice of objects generated by subdivision [CGC$^+$02].

Precomputed Global Response Another technique for fast global deformations is the use of a precomputed reduced representation of the dynamics of the object. This representation is commonly created from a modal analysis of the linear small strain problem, first presented by Pentland and Williams [PW89]. For special problems, where only small motions with very particular characteristics are present, this gives good results. It has been used for the animation of trees [Sta97], and can be assisted by modern graphics hardware [JP02]. Recently, the basic idea of precomputing dynamics has been enhanced considerably by employing precomputed impulse responses to a reduced phase space model [JF03]. This also allows nonlinear dynamics, but further restricts the possible interactions to the precomputed response tables.

Finite automata If the volumetric model is structured into voxel cells, it is possible to model deformations by cellular automata. The basic concept is to perform simple calculations on a very large number of elements, instead of complex computations on a small number of elements as in scientific computing. Examples are the sphere filled models used by Suzuki et al. [SHT$^+$98] or the 3D chainmail algorithm proposed by Gibson [FG99]. The latter algorithm adds an elastic relaxation step to smooth the output after each ’chain mail propagation’ of local deformation through the element grid. The algorithm can be extended to integrate inhomogeneous and anisotropic behaviour, proposed by Schill et al. [SGBM98] for a surgical intervention at the eye. Although these techniques allow the animation of a very high number of nodes, they are limited in their ability to reproduce physical material properties.

Mass-spring-damper systems The next step towards a more physically based approach are mass-spring-damper systems. To our best knowledge, they have been introduced by Platt and Badler [PB81] for facial animation and ever since are one of the most favoured modelling techniques in computer graphics. Numerous extensions have been proposed. Lee, Terzopoulos, and Waters [TW90] use multiple layers of meshes connected by springs to simulate the different tissue layers of the face. For cloth animation, Provot [Pro95] proposed to use bi-phasic springs and an iterative process to correct super-elongated edges to reduce the spring stiffness necessary for cloth-like objects. Kuhn, Kühnapfel, and Krumm [KKK96] used mass-spring-damper systems to build the KISMET simulator for virtual surgery. For the 'Spring'-framework [MBB$^+$02] used for example in the virtual rat trainer [BO02] 'nomen est omen'. Mass-spring networks allow the use of physical constants like spring-stiffness. The identification of spring-parameters to reflect the physical properties of a given material is a tedious task, and has been tackled by simulated annealing [DKT95], evolutionary algorithms [LPC95], and heuristics inspired by quantum mechanics [MBT03]. A complete solution to the problem cannot be expected, as van Gelder [VG98] showed the inability of mass-spring systems to model homogeneous materials. Also, incompressibility or transverse contraction can not be modelled without additional 'non-spring' penalty forces.

Particle systems A more general concept than mass-spring-systems are particle based approaches. They conserve the idea of discretising the object into mass points, but besides springs, allow arbitrarily shaped forces between these particles. Strictly spoken, the force should be a function of the distance between points only, but there are systems that define forces on other topological structures, sometimes referred to as generalised particle systems. For example Lee et al. [LTW95] add a volume preserving force to their mass-spring nets. Breen et al. [BHW94] used forces defined by polynomials fitted to Kawabata data to simulate woven cloth. The work has been extended by Eberhardt et al. [EWS96], now directly using measured forces from the Kawabata measurement system, and later to knitted textiles by the same authors [EW99]. Particle systems are not restricted to a fixed topology as mass-spring systems are, in fact their main application is for phenomena without a fixed neighbourhood, like fire and smoke, introduced by Reeves [Ree83]. They are popular, because of their direct geometrical interpretation, which is more accessible than the weak integral formulations of finite elements.

Particle systems are easy to implement and use. However, the defining forces have to be derived with special care, in order to obtain properties like independency of the discretisation and scaling of the object. Several approaches have been used in graphics: the smoothed-particle hydrodynamics (SPH) methodology, originating from astrophysics, has been employed by Desbrun and Cani [DG96]. As the name already states, it is restricted to highly deformable substances. Etzmuss et. al. [EGS03] proposed to use finite differences to derive particle system forces for cloth simulation. Finally, finite-volume methods, recently employed by Teran et al. [TBHF03] for muscle animation, can be used to derive well-defined forces between mass-points from a continuum mechanical description.

1.2.2 Modelling based on Continuum Mechanics

As soon as a continuum is considered as the underlying description of an elastic object, the field of scientific computing provides a large set of tools for the problem. The advantage of this approach is that it is based on a solid mathematical and physical foundation, and can be customised to most requirements. Therefore techniques from numerical engineering [ZT00, HW96], namely finite element and finite difference methods continue to be the most versatile and accurate methods. The drawback is, that they usually require high computational expenses, so some modifications are needed, trading interactivity for accuracy.

Finite Element Methods in Graphics The use of these techniques was pioneered by Terzopoulos et al. [TPBF87, TF88a] for simulations of elastic deformations, in this case off-line due to the high computational costs. The authors, deriving their concept from differential geometry, already used a nonlinear, rotationally invariant description for strain, namely the metric tensor or first fundamental form, which is a scalar multiple of Green's strain.

Linear Finite Elements Although Terzopoulos used the metric tensor, which is a large displacement measure, later on small displacement measures like Cauchy's tensor were common. Combined with a linear elastic material, this leads to a linear system of partial differential equations, which is easier to solve than a nonlinear one. This property coined the sloppy term 'linear finite elements', which does not necessarily imply linear shape functions. Commonly this is perceived as 'the finite element method', as for most engineering applications, where the technique dominates the field, materials like metal and concrete do not show large deformations, which qualifies the linear method as sufficient.

Perhaps the first appearance of linear finite elements in graphics was the simulation of a skinned human hand, grasping a squeezable ball by Gourret, N. Thalmann, and D. Thalmann [GTT89]. This paper also introduced the condensation technique, unnoticed by most and later advertised by Bro-Nielsen, into the field. Condensation reduces the numbers of variables by only considering surface nodes. The degrees of freedom corresponding to inner nodes are eliminated by applying block-substitution techniques to the linear system. The disadvantage of this approach is that the system matrix looses sparsity due to this block-substitution, so that we did not consider this trick in the present work. One of the next applications, now using higher order elements, but still with a geometrical- and material-linear model, was the animation of a biomechanical muscle model [CZ92].

An interactive medical application has been presented by Bro-Nielsen [BN98], who used linear tetrahedral finite-elements together with Cauchy's small strain tensor and condensation for interactive simulations. The work was continued by Cotin et al. [CDA99], who proposed a precomputation of elementary deformations exploiting the superposition principle. James and Pai [JP99] used boundary element techniques to formulate a linear system defined only on the surface. An update technique based on the one-dimensional Sherman-Morrison-Woodbury formula ensured a fast simulation. Later Pai et al. [PvdDJ$^+$01] presented an automatic measurement facility to acquire data for their model.

The drawback of using a linear elastic model is, that it is only accurate for small displacements of about ten percent of the mesh size, and, more seriously for animation tasks, produces ghost forces under rotations larger than a few degrees.

Nonlinear Finite Elements For this reason, the use of nonlinear elasticity has been considered already very early, however mainly for non-interactive applications. Several systems for computer-aided craniofacial surgery, more dependent on accuracy than facial animation, use nonlinear 2D and 3D elastic models discretised by finite elements for static and quasi-static simulations [KGC$^+$96, KGKG98, GZDH02]. Later, also facial animation took up finite elements [KGB98]. Large strain measures were brought back to non-medical animations by O'Brien and Hodgins [OH99], who apply a model based on Green's tensor to simulate brittle and fracture, computed off-line by an explicit integration method.

Real-time animation based on explicit finite elements, i.e. finite elements combined with explicit time integration, using Green's tensor has been presented by Debunne et al.[DDCB01]. To achieve interactive frame-rates they use a non-nested hierarchy of tetrahedral meshes. Similar to these publications one of the strain measures we will employ is Green's tensor as a nonlinear, large deformation measure.

Multi-resolution meshes recently became very popular in animation. Wu et al.[WDGT01] employ a precomputed hierarchy based on vertex-splitting combined with a forward Euler method. Capell et al. [CGC$^+$02] use a hierarchical basis on a hexahedral control lattice to animate an enclosed solid, which is then deformed with an FFD technique. They also use a linearised implicit Euler method. To avoid nonlinear systems, they linearise around a floating reference frame. This idea is also exploited in the work of Müller et al. [MMD+02]. Instead of linearising Green's tensor, they introduce a technique named warping to extrapolate this frame and use Cauchy's tensor around this warped coordinate system. This idea was previously used in engineering, introduced in the analysis of orbiting space and aircraft structures [dV76].

Instead of the heuristical warping technique we proposed [HS03] to use the polar decomposition for extracting rotations, an approach that now has been adopted by Müller in an upcoming publication [MG04]. The computer graphics community has been relatively unaware of this matrix decomposition technique. The only reference in graphics literature we were able to find was by Showmake and Duff [SD92], who used the polar decomposition for keyframe matrix animation. Their problem shows some similarities, as they also needed to separate strain and rigid body descriptions. A similar application than the one presented here, also using the polar decomposition, is currently evaluated for cloth simulations [EKS03]. The methods are related, although we discuss the general 3D case.

Hierarchical finite element based on nested meshes have been introduced into numerical analysis during the 80's [Ban96], the central idea itself has been traced back by Yserentant [Yse92] to the work of Faber [Fab09] at the beginning of the last century. CHARMS [GKS02, KGS01] extended the concept to a quasi-hierarchical basis and presented a rule-set guaranteeing consistency. We will use a similar rule-set, but only a standard hierarchical basis.

Elastic materials Concerning material laws for deformable solids, the linear Hooke law defines the standard, although it has been generalised to anisotropic linear elasticity [PDA01]. Viscous forces are usually treated by a lumped damping force proportional to the velocity. O'Brien et al. [OH99] use a better model for viscosity, based on the strain rate tensor. In addition, they propose a formulation for plasticity. Non-reversible deformations have also been considered by Terzopoulos and Fleischer [TF88b, TW88]. In this paper, the authors also discuss the importance of viscous forces and utilise a more complex viscoelastic material law. In fact, this model can be shown to be a special case of the Prony series model we employ, using only a single memory parameter.

1.2.3 Identification of Tissue Parameters

Finding stiffness and damping parameters that let medical simulations look realistic is also a problem that has been previously addressed. Maaß et al. [MK99] and Szekely et al. [SBHR98] were among the first to try and measure tissue stiffness with various methods in order to improve the realism of simulations. Ottensmeyer [Ott02] built a indentation measurement device that can be employed during a laparoscopic intervention. Brown et al. [BRM+02] equipped an endoscopic grasper with a motor and a force sensor. All those groups agree on the importance of viscous properties for soft tissue. Most of them use different constitutive laws, however. Neither of these groups implemented their viscoelastic model into a simulation. In the ElastoMed project, we chose to adopt the constant-Q hypothesis of Fung [Fun93] about the shape of the hysteresis loop for soft tissue. Ex-vivo measurements, fortifying this hypothesis, were performed by Gross [GHEB01]. The acquired data has been validated by comparative in-vivo measurements [OKG02, KOGD03]. We base the material model for soft tissue on these measurements.

In the future, non-invasive methods such as elastography based on NMR or ultrasound imaging introduced, among others, by Sinkus [SLS+00] are promising for the purpose of gathering viscoelastic data of human tissue in-vivo. A tentative ex-vivo test has already been made [KHM04]. Currently, a joint initiative to establish physical standards [KCO+03] to validate and improve in-vivo measurement devices has been started.

Measuring parameters for cloth is already standardized [Kaw80], but only elastic parameters are determined. Several current research initiatives include the measurement of viscous constants, but close to nothing has been published. Preliminary results [WG], performed by the Hohenstein Institute for Clothing Physiology, confirm the constant Q model as a good first order approximation.

1.2.4 Time Integration Methods

Concerning time integration methods, more advanced work has been presented in the publications about cloth simulation, which therefore is the primary focus of this section. We already mentioned some of the integration techniques used in prior work, from those only the most relevant contributions will be repeated.

For the ordinary differential equation resulting from their finite element formulation, Terzopoulos et. al. [TPBF87] used a simple semi-implicit Euler scheme. Later publications focused on explicit integration methods, e.g. Eberhardt et al. [EWS96] preferred RK4 and the Burlisch-Stoer extrapolation method as suggested in Press et al. [PTVF88]. Courshesnes, Volino and Thalmann [CVT95] used an explicit midpoint rule. The biphasic spring model of Provot [Pro95] has been introduced to allow weaker springs in order to alleviate the stability problem, and hence permit larger time steps with explicit time integration.

Implicit methods again became popular with the work of Baraff and Witkin [BW98]. They used a linearised implicit Euler method and achieved simulations about an order of magnitude faster than explicit methods. Although nonlinear constraints are formulated in the model, they only use a linear approximation to obtain a linear system of equations. This way the system that needs to be solved in each time step also becomes linear and can be solved efficiently by a conjugate gradient (cg) method. This method corresponds to the solution of a nonlinear system with only one Newton iteration. Because the nonlinear part is not integrated, with high stiffness one may encounter slowed down dynamics, observed by Volino et al. [VMT01] and Eberhardt et al. [EEH00]. In the latter work, we proposed to partition the differential equation into a stiff and non-stiff part, which permits the combination of an implicit method for the stiff and an explicit method for the non-stiff part yielding a more efficient time integrator.

Provot's spring model was combined with a linearised implicit method by Desbrun et. al. [DSB99]. Instead of linearising the whole system they split it in a linear and nonlinear part and use a precomputed inverse of the linearised system matrix for solving the linear part of the equations. They don't aim at solving the equation completely, as they don't integrate the nonlinear term explicitly. Instead, the angular momentum is corrected to account for the nonlinear part. With this algorithm one can neither change the time step size nor deal with a time dependent system.

Based on this work Kang et al. [KCC$^+$00] did some further simplification to avoid solving the linear system. In order to update the solution vector during a time step they divide by the diagonal entry of the matrix of the linear system. Therefore, they just do a single iteration of a Jacobi-like scheme for solving the linear equations. Again, this may lead to artificial slowdowns.

Recently, more advanced methods gain importance. In 2001, we proposed to use BDF (backward differentiation formulas) and the implicit midpoint rule. The BDF method has also been used by Choi and Ko [CK02].

For the simulation of deformable solids, the evolution of time integration shows some parallels. Most of the work concerning off-line simulation, e.g. for facial animation, used implicit integration, probably a result of the transition from static finite element calculations to dynamic ones. Picinbono, Lombardo, Delingette and Ayache [PLDA02] combined their anisotropic tensor-mass model with an explicit integration method. Because the incompressibility poses severe restrictions to the step size, Picinbono et al. [PDA00] used additional penalty forces, that permits less strict material properties and thus larger time steps. Debunne et al. [DDCB01] also advertised the use of explicit finite elements. We promoted the use of an explicit integration method, because of the simpler implementation [HGS03a], but chose a stabilised one, which is better suited for stiff equations. As our framework now is maturing, implicit methods are showing their benefits.

1.3 Overview and Contributions

This thesis achieved an efficient and accurate framework for the visual simulation of 2D and 3D deformable models. For 2D deformable models, we focus on the construction and implementation of an efficient time integration method for virtual textiles. 3D deformable models, finding their application in general animation and virtual surgery simulation, are less developed in graphics; here we propose to transfer the toolbox of scientific computing, in order to guarantee a valid and meaningful spatial model. For the implementation of an interactive simulator, special attention has to be paid to linear algebra, as for 3D topologies the dimension rises very fast, so that we propose and adopt several methods that have not been considered for this kind of application type yet.

In particular, we were among the first to customise and apply higher order implicit numerical time integration methods in cloth simulation [HE01, HES03], for example the second order BDF method, later also used by other authors. To our best knowledge, this thesis is the first one in this field to exploit the benefits of these methods for volumetric objects.

An implicit numerical time integrator is not a single algorithm, but a careful combination of an integration formula, a non-linear and a linear solver. For the latter, the method of conjugate gradients is preferred in graphics. To allow a unified treatment, we proposed the concept of a layered solver [HE01], with an inexact Newton method as a central part. This allows for the description and classification of several existing methods, for example the widely used semi-implicit or linearised implicit Euler method.

Implicit integration methods spend most of their time solving linear systems. We introduce an implementation of the preconditioned conjugate gradient method that provides preconditioning for free. In addition, we present direct methods for sparse linear systems that are superior for the ‘small’ dimensions typically used in interactive graphical applications. Both linear solvers need to be customised to be compatible with the collision handling methods used in graphics, which requires a new formulation of the problem to maintain efficiency.

For deformable solids, the field has not settled on a modelling technique. We employ non-linear finite elements, which until recently have been considered as not feasible for interactive applications, but now start to be more and more accepted. Whereas the first interactive implementation needed a large parallel computer, we achieve interactivity on a single processor workstation.

Medical simulations require a higher accuracy than the usual animations used in graphics. As measurements indicate, that viscosity is an important characteristic of biological soft tissue, we present the first interactive simulator using a viscoelastic material law [HGS03a] that permits a fit to measured data [HGS03b].

In order to use finite elements also with a larger number of elements, we improved the recently proposed technique by Müller et al. [MMD$^+$02] that allows the system to stay linear, by replacing a heuristic by the theoretically optimal technique [HS03, HS04]. This helps to reduce the error that is introduced by this approximation, as a comparison will show. To maximise the benefit of a limited number of elements we evaluated the use of a hierarchical basis that permits the addition of detail where necessary. Though the numerical basic technique proved valuable, the subdivision technique used to generate the adapted grid requires further attention.

We will evaluate the resulting application, to establish an agreement between the theoretical considerations and the practical problem, and to investigate the capabilities and also the limits of the approach. We compare to more traditional methods of computer graphics as mass-spring systems, and to a commercial finite element package.

The following chapters of the thesis pick up these issues as follows. The first chapter will introduce the most important concepts from the continuum mechanical modelling of deformable objects, as this is not standard in computer graphics. The next chapter presents the numerical techniques we selected to implement our solution of the problem. A short chapter will demonstrate the numerical core developed for clothing simulation. Finally, the last chapter discusses the interactive application for 3D solid dynamics.

Chapter 2

Physical and Mathematical Modelling

At the end of the year 1820 the fruit of all the ingenuity expended on elastic problems might be summed up as–an inadequate theory of flexure, an erroneous theory of torsion, an unproved theory of vibrations of bars and plates, and the definition of Young's modulus.

A Treatise on the Mathematical Theory of Elasticity,
A.E.H. Love, 1927.

In this chapter we will report briefly on relevant portions of continuum mechanics of solids, in order to provide the necessary tools for the following discussion. Partly, the concepts have been presented at the Symposium for Computer Animation [HGS03a] and at MICCAI [HGS03b]. For further details we refer to the textbooks of Brandt and Dahmen [BD96], Ciarlet [Cia92], Gurtin [Gur81], Atanackovic and Guran [AG00], or Bonet and Wood [BW00].

The overall structure of the approach is depicted in figure 2.1. In this chapter the mathematical and physical part is discussed, the numerical part will follow in the next chapter. The first section covers the mathematical description of a deformable solid, which leads to strain, describing the geometry of deformation. The following section covers stress, which characterises the force distribution in a body and allows the derivation of the equations of equilibrium. Elastic material laws link stress and strain, visco-elastic materials extend this concept to solids with memory. The only way to select a visco-elastic model for a special application is by measurements, which will be discussed next. A summary of the most important aspects concludes the chapter.

2.1 Deformable Objects

A common object in graphics is a parameterized surface (fig. 2.2(a)). Analogously, a rigid object with volumetric properties (fig. 2.2(b)) is described by its *configuration mapping* φ, now possessing a subset Ω of $\mathbb{R}^3$ as parameter domain. For a deformable solid D (fig. 2.2(c)) the mapping in addition depends on time,

$$\varphi : \Omega \times [0, \infty] \to \mathbb{R}^3.$$

It is convenient to treat this as a family of deformed rigids indexed by time and to write φ_t as an abbreviation, or if time does not play a role, even to drop the index t. We refer to the parameter

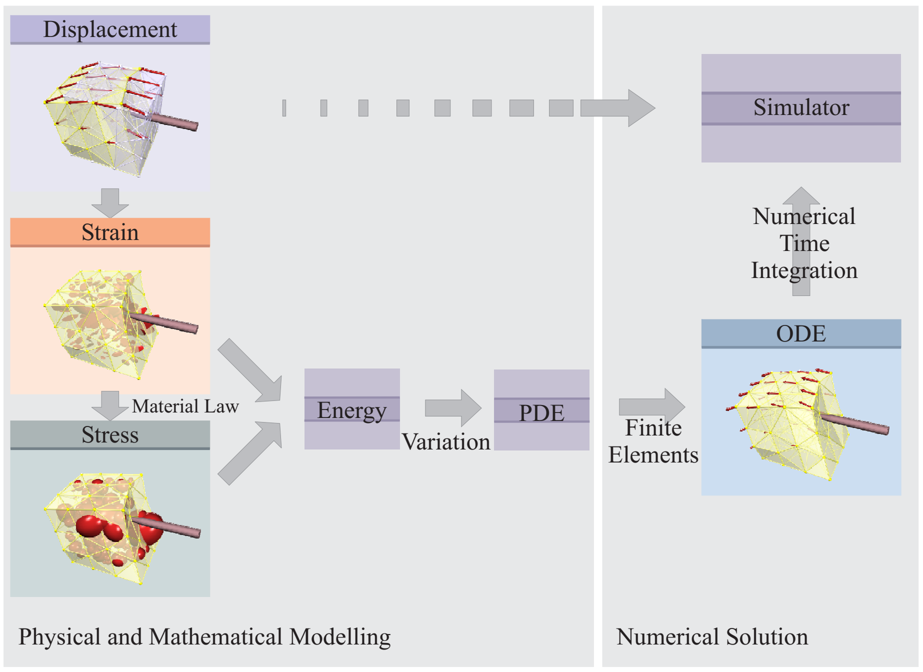


Figure 2.1: Modelling and solution of elastic problems.

domain Ω as the space of material coordinates. It is common to parameterize a deformable object over itself at rest, and to assume that the rest state is taken at $t = 0$, thus

$$\varphi_0 = \varphi(\cdot, 0) = \mathrm{id}.$$

Therefore $\Omega = \varphi_0(\Omega)$ and an alternative description for the state of D is given by the *displacement field*

$$u_t : \Omega \to \mathrm{I\!R}^3, \quad u_t = \varphi_t - \mathrm{id}.$$

Although both formulations are equivalent, the displacement field u is used predominantly.

2.2 Strain - A Measure of Deformation

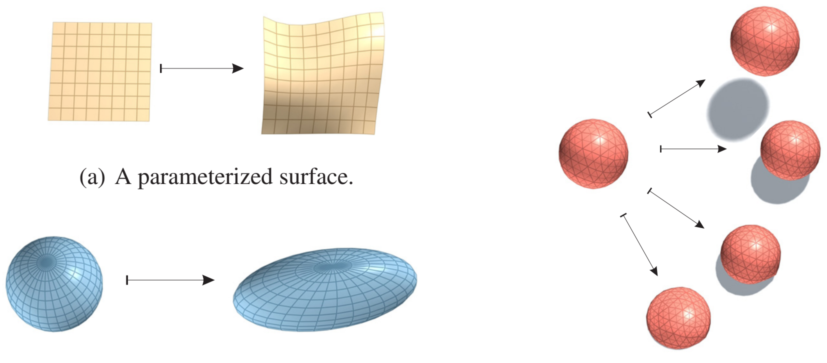
A rigid body movement of D induces no strain, but of course changes φ and u, hence the direct use of the configuration or displacement field is not suitable as a measure for strain.

To derive a measure, we consider the length changes of a segment dx in rest and deformed state (Figure 2.3). Using a Taylor expansion, it holds

$$\begin{aligned} dx_t &= \varphi_t(x + dx) - \varphi_t(x) = \varphi_t(x) + \nabla\varphi_t(x)dx + O(dx^2) - \varphi_t(x) \\ &\approx \nabla\varphi_t(x)dx = (\nabla u_t(x) - \mathrm{id})dx, \end{aligned}$$

for both above mentioned formulations. Within first order, the length change now becomes

$$\begin{aligned} \|dx_t\| - \|dx\| &= \langle dx_t, dx_t\rangle - \langle dx, dx\rangle = \langle \nabla\varphi_t dx, \nabla\varphi_t dx\rangle - \langle dx, dx\rangle \\ &= \langle (\nabla\varphi_t^T\nabla\varphi_t - \mathrm{id})dx, dx\rangle = \langle (\nabla u_t + \nabla u_t^T + \nabla u_t^T\nabla u_t)dx, dx\rangle. \end{aligned} \tag{2.2.1}$$


(a) A parameterized surface.

(b) A parameterized solid.

(c) A deformable object.

Figure 2.2: Parameterized objects.

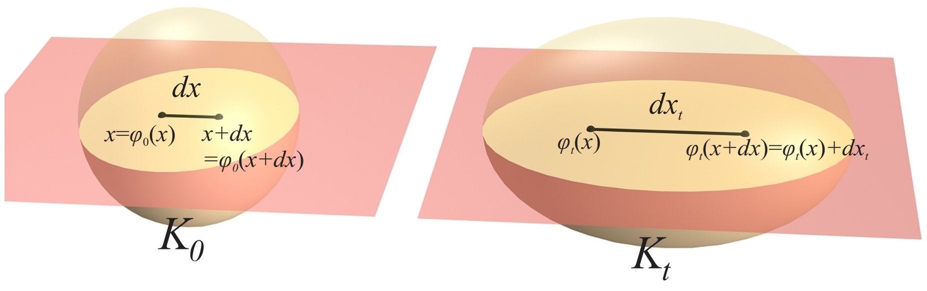


Figure 2.3: Length changes in a deforming body. Without loss of generality we assume $\varphi_0 = id$.

We therefore define the symmetric matrix[1], first introduced by Green and St.Venant,

$$\epsilon_t^G := \frac{1}{2}(\nabla\varphi_t^T \nabla\varphi_t - \mathrm{id}) = \frac{1}{2}(\nabla u_t + \nabla u_t^T + \nabla u_t \nabla u_t^T), \tag{2.2.2}$$

usually referred as Green's strain tensor and its linearisation

$$\epsilon_t^C := \frac{1}{2}(\nabla u_t + \nabla u_t^T),$$

Cauchy's strain tensor, which are governing the length change. The sub-term

$$C := \nabla\varphi_t^T \nabla\varphi_t \tag{2.2.3}$$

in the definition of Green's tensor is called Cauchy's deformation tensor. Cauchy's strain tensor is used for small deformations, e.g. for metal or concrete. Both strain tensors coincide in the small displacement limit. The individual components of Green's tensor are given by

$$\epsilon_{ij}^G = \frac{1}{2}\left(\frac{\partial\varphi_i}{\partial x_j}\frac{\partial\varphi_j}{\partial x_i} - \delta_{ij}\right) = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} + \sum_k \frac{\partial u_k}{\partial x_i}\frac{\partial u_k}{\partial x_j}\right).$$

[1]The factor $\frac{1}{2}$ takes care, that the diagonal entries reduce to $\partial u_i/\partial x_i$ in the small strain limit.

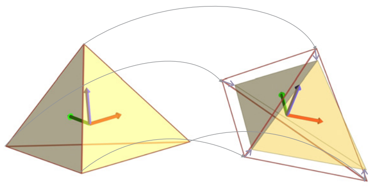


Figure 2.4: Separating rotation and deformation by a rotating reference frame.

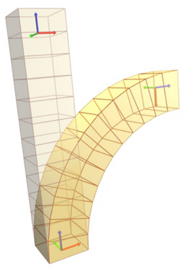


Figure 2.5: Non-moving bar with large rotations.

2.1 Using Cauchy's tensor leads to more efficient algorithms, as its evaluation is cheaper and it is a linear function of the displacements. However, it has the serious drawback that it is not invariant under rotations, which makes it inaccurate in the presence of finite rotations. Green's tensor is rotationally invariant, because for any rotation R applied to the configuration φ we obtain

$$\epsilon^G(R\varphi) = \frac{1}{2}\left(\nabla\varphi^T R^T R \nabla\varphi - \mathrm{id}\right) = \epsilon^G(\varphi),$$

exploiting the linearity of the gradient and the orthogonality of R.

The geometrical meaning of the entries of ϵ is as follows: the ϵ_{ii} determine the relative elongation of the line elements along the coordinate axes, the offdiagonal entries are related to the shear angles [AG00, pp.52-57].

Being symmetric, the strain tensor can be diagonalised over $\mathbb{R}$, yielding the principal directions of strain as eigenvectors. They determine the volume change. In this coordinate frame there are no shear deformations. The mean volume distortion $e := \frac{1}{3}\operatorname{tr}(\epsilon)$ gives this volume change and is invariant under a changing reference frame. This leads to a decomposition of ϵ as

$$\epsilon = e \cdot \mathrm{id} + \epsilon^{\mathrm{Dev}}, \tag{2.2.4}$$

where ϵ^{Dev} is called the deviatoric part of ϵ. It has a vanishing trace, thus inflicts no volume change. Therefore the strain is decomposed into a volume conserving distortion (or pure shear deformation) and a symmetrical dilatation (or stretch). This is the basis for two field formulations, where $e \cdot \mathrm{id}$ leads to the definition of a hydrostatic pressure.

The last strain measure, that will be presented, is the corotated Cauchy strain. The corotational formulation aims at the elimination of the geometrical nonlinearity in strain. The idea is to keep track of a rotated coordinate system of the body (fig. 2.4). It has first been employed with a single reference frame for the entire structure, when the analysis of orbiting space and aircraft structures made such techniques obviously necessary [dV76]. Unfortunately a single frame per body is not enough in the deformable case. Considering a long bar (figure 2.5) fixed at one end and bent at the other shows, that it is not sufficient to keep track of a single rigid body movement in general. The attached part is not rotated whereas the other end may undergo arbitrarily large rotations.

For the moment, we suppose that, at each point we know the local rigid body rotation $R(x)$. Then the corotational Cauchy stress

$$\epsilon^{\mathrm{CR}}(\varphi) := \epsilon^C(R^T\varphi)$$

is a trivially rotationally invariant strain description and still linear. So it unites two important features of ϵ^C and ϵ^G. Later the forces arising from this description need to be transformed back by R into their original frame.

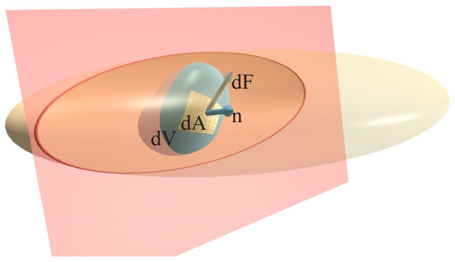


Figure 2.6: *Derivation of stress.*

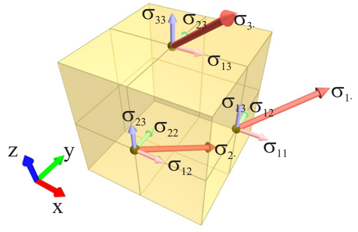


Figure 2.7: *Geometrical interpretation of stress as traction.*

2.3 Stress - A Description of Inner Forces

En développant cette idée, je suis parvenu à reconnaître que la pression ou tension excercée contre un plan quelconque en un point donné d'un corps solide se déduit très aisément, tant en grandeur qu'en direction, des pressions ou tensions exercées contre trois plans rectangulaires menés par le même point.

De la pression ou tension dans un corps solide, A.-L. Cauchy, 1827.

Although stress will appear as a dependent variable only, being the result of strain and a material law, it has a justification of its own and will allow the derivation of the equilibrium conditions of a deformable body. Note that all concepts depend on the current configuration of the body and are derived in its current frame, which is emphasized by the φ superscript.

To derive the concept of stress, we decompose a deformable body D^φ into small volume elements dV^φ (figure 2.6). Traction forces are acting on each differential surface element $n^\varphi|dA^\varphi|$, with normal n^φ and area $|dA^\varphi|$. The force magnitude $|dF^\varphi|$ is proportional to the area $|dA^\varphi|$ but dF^φ needs not to be aligned with n^φ. Therefore dF^φ can be written as

$$dF^\varphi = \sigma^\varphi(n^\varphi)n^\varphi|dA^\varphi|,$$

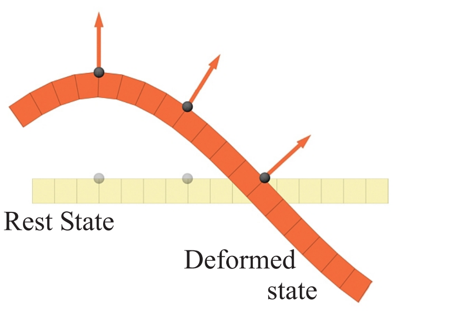
but $\sigma^\varphi(n^\varphi)$ must be a tensor, not a simple scalar. Cauchy's fundamental theorem of stress (Theorem A.1) states, that σ^φ does not depend on the normal direction n^φ, so it suffices to specify it as a 3 by 3 matrix per material point. Furthermore the resulting Cauchy stress tensor σ^φ is symmetric[2], a fact that can be derived from the axiom of angular moment balance for $dV^\varphi \to 0$ [AG00, Ch.1.3]. If we normalise the force vector dF^φ by $|dA^\varphi|$ we arrive at the traction or Cauchy stress vector

$$t^\varphi_{n^\varphi} = \sigma^\varphi n^\varphi, \tag{2.3.1}$$

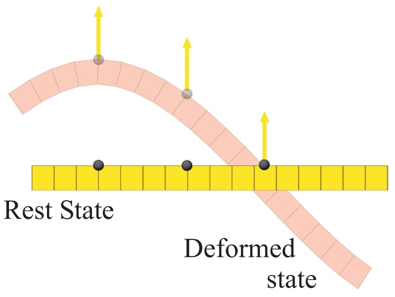
which has the unit of pressure, i.e. force per surface unit. Tractions give a geometrical meaning to σ^φ. The diagonal terms σ^φ_{ii} are called normal stresses, $\sigma^\varphi_{12}, \sigma^\varphi_{13}, \sigma^\varphi_{23}$ shearing stresses. Like strain, the tensor can be diagonalised, yielding the principal directions of stress. The diagonal tensor has normal components only, describing tension ($\sigma^\varphi_{ii} > 0$) or pressure ($\sigma^\varphi_{ii} < 0$). The mean pressure

$$p = -\frac{1}{3}\,\mathrm{tr}(\sigma^\varphi) \tag{2.3.2}$$

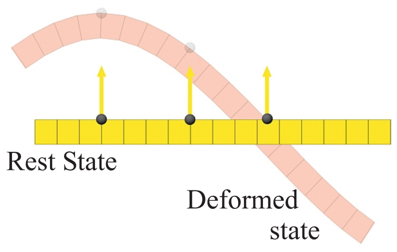
[2]This is sometimes called Boltzmann's axiom and does *not* hold for materials with assigned couples.



(a) Cauchy stress: Forces and reference configuration are in the deformed state.



(b) First Piola-Kirchhoff stress: Forces are in the deformed, and the reference configuration is the rest state.



(c) Second Piola-Kirchhoff stress: Forces and reference configuration are in the rest state.

Figure 2.8: Stresses and configurations.

is frame independent and in two-field, mixed formulations added as a dependent variable, coupled by Lagrangian multipliers. The remaining no-pressure part $\sigma^{\varphi\,\mathrm{Dev}} = \sigma^\varphi + p \cdot \mathrm{id}$ is again called the deviatoric part of stress.

2.4 Equilibrium and Elastic Energy

A body is in equilibrium if the forces acting on every volume element $V^\varphi \subset \Omega^\varphi$ cancel the imposed body forces f^φ and surface loads g^φ, i.e. with the Cauchy stress vector $t^\varphi_{n^\varphi}$

$$\int_{\partial V^\varphi} t^\varphi_{n^\varphi}(x^\varphi)\, dA^\varphi - \int_{V^\varphi} f^\varphi(x^\varphi)\, dx^\varphi - \int_{\partial V^\varphi} g^\varphi(x^\varphi)\, dA^\varphi = 0.$$

Using the definition of the Cauchy stress tensor and the divergence theorem, this transforms to

$$\int_{V^\varphi} \mathrm{div}^\varphi\, \sigma^\varphi(x^\varphi) - f^\varphi(x^\varphi)\; dx^\varphi - \int_{\partial V^\varphi} g^\varphi(x^\varphi)\, dA^\varphi = 0 \qquad \text{for all } V^\varphi \subset \Omega^\varphi, \tag{2.4.1}$$

and for $V^\varphi \to 0$ gives the famous equilibrium equations of continuum mechanics

$$\begin{aligned} \operatorname{div}^\varphi \sigma^\varphi(x^\varphi) - f^\varphi(x^\varphi) &= 0 \qquad \text{for all } x^\varphi \in \Omega^\varphi, \\ \sigma^\varphi(x^\varphi)n^\varphi - g^\varphi(x^\varphi) &= 0 \qquad \text{for all } x^\varphi \in \partial\Omega^\varphi. \end{aligned} \tag{2.4.2}$$

Using a variant of Green's formula, the equilibrium equation can be stated in a weak form

$$\int_{\Omega^\varphi} \sigma^\varphi(x^\varphi) : \nabla\theta^\varphi \, dx^\varphi = \int_{\Omega^\varphi} f^\varphi(x^\varphi)\theta^\varphi \, dx^\varphi + \int_{\partial\Omega^\varphi} g^\varphi(x^\varphi)\theta^\varphi \, dA^\varphi \tag{2.4.3}$$

for *all* smooth enough vector fields θ^φ that satisfy the boundary conditions. This equation constitutes the principle of virtual work in the *deformed* configuration.

Because the deformed configuration is not known, (2.4.3) should be transformed back to the reference configuration. This is achieved by transforming all integration domains and the tensor divergence operator by the Piola transformation [Cia92]

$$\int_\Omega (\det(\nabla\varphi)\, \sigma^\varphi \, \nabla\varphi^{-T}) : \nabla\theta \, dx =: \int_\Omega \sigma^1 : \nabla\theta \, dx = \int_\Omega f\theta \, dx + \int_{\partial\Omega} g\theta \, dA,$$

where $f(x) = \det(\nabla\varphi)\, f^\varphi(x^\varphi)$ with $x^\varphi = \varphi(x)$ and g is transformed likewise.

This leaves two issues open. First the first Piola-Kirchhoff stress σ^1 is not symmetric anymore and the external force densities f and g still depend on the configuration φ via the determinant and the identification of the arguments[3]. This is due to the fact, that the Cauchy stress vector t^φ is still balanced against the external body forces in the deformed geometry. Therefore we take an additional step and map the force dF^φ to the undeformed state obtaining $dF = \nabla\varphi^{-1} dF^\varphi$. The forces are now balanced in the reference configuration. Therefore the tractions (2.3.1) transform to $dF/|dA| = \nabla\varphi^{-1}\sigma^\varphi\nabla\varphi n$. After again applying the Piola push-forward/pull-back transformation, the resulting second Piola-Kirchhoff stress tensor, given by

$$\sigma = \det(\nabla\varphi)\nabla\varphi^{-1}\sigma^\varphi\nabla\varphi,$$

now is symmetric and the resulting equilibrium equations are

$$-\operatorname{div}\big(\nabla\varphi(x)\sigma(x)\big) = f(x) \qquad \text{and} \qquad \int_\Omega \nabla\varphi\sigma : \nabla\theta \, dx = \int_\Omega f\theta \, dx + \int_{\partial\Omega} g\theta \, dA \tag{2.4.4}$$

for all smooth enough vector fields θ, fulfilling the boundary conditions.

As so often in variational principles, there is an energy behind the weak equation. The first law of thermodynamics states that the rate of work done equals the change of kinematic and internal energy U per mass unit

$$\frac{d}{dt}\int_{\Omega^\varphi} \frac{1}{2}\rho^\varphi v^\varphi v^\varphi + \rho^\varphi U \, dx^\varphi = \int_{\Omega^\varphi} f^\varphi v^\varphi \, dx^\varphi + \int_{\partial\Omega^\varphi} g^\varphi v^\varphi \, dA^\varphi$$

Replacing the surface traction g^φ by the balanced surface stress $\sigma^\varphi n^\varphi$ and applying Gauss's theorem leaves

$$\frac{d}{dt}\int_{\Omega^\varphi} \frac{1}{2}\rho^\varphi v^\varphi v^\varphi + \rho^\varphi U \, dx^\varphi = \int_{\Omega^\varphi} (\operatorname{div}^\varphi \sigma^\varphi + f^\varphi)v^\varphi + \sigma^\varphi : \dot\epsilon^\varphi \, dx^\varphi$$

with the rate of deformation tensor $\dot\epsilon^\varphi := D := \frac{1}{2}(\nabla^\varphi v^\varphi + \nabla^\varphi v^{\varphi\,T})$ in spatial coordinates. Combining the equilibrium conditions (2.4.2) with Newton's third law of motion

$$\operatorname{div}^\varphi \sigma^\varphi + f^\varphi = \rho^\varphi \frac{d}{dt} v^\varphi$$

[3]Force densities that are, by their specification, independent of φ, i.e. $f = f^\varphi$, are called dead loads. Contrarily, for example pressure forces depend on the surface normal and therefore on φ.

allows the identification of the rate of the internal energy in spatial coordinates as

$$\rho^\varphi \frac{d}{dt} U = \sigma^\varphi : D. \tag{2.4.5}$$

To transform the equation into material coordinates, we exploit the fact, that the rate of Green's tensor can be written as (Appendix A, Prop. A.2)

$$\dot{\epsilon}^G = \frac{\partial}{\partial t} \nabla\varphi^T \nabla\varphi + \nabla\varphi^T \frac{\partial}{\partial t} \nabla\varphi = \nabla\varphi^T D \nabla\varphi$$

which gives

$$\rho^\varphi \frac{d}{dt} U = \sigma^\varphi : \nabla\varphi^{-T} \dot{\epsilon}^G \nabla\varphi^{-1} = \nabla\varphi^{-1} \sigma^\varphi \nabla\varphi^{-T} : \dot{\epsilon}^G = \sigma : \dot{\epsilon}^G. \tag{2.4.6}$$

Stress and strain measures fulfilling such an equation are called *energetically* or *work conjugate*. Hence, for small displacements the pair consisting of Cauchy stress and Cauchy strain, for large displacements the combination of the second Piola-Kirchhoff stress and the Green strain is preferred. Note that both the second Piola-Kirchhoff stress and the Green strain are objective tensors, i.e. they are indifferent under a rotation of the coordinate frame.

The total internal energy E_U is then given by

$$E_U = \int_\Omega \rho U \, dx = \int_0^t \int_\Omega \sigma : \dot{\epsilon}^G \, dx dt.$$

For small displacements, at the state of equilibrium the time integration can be carried out and

$$E_U = \int_\Omega \sigma : \epsilon \, dx.$$

The same equilibrium result is gained by replacing $v = \theta$ in the weak equation and employing $\sigma : \nabla v = \sigma : D$ (see Appendix A, Prop. A.3).

2.5 Material Laws

ceiiinosssttuv

Anagram, R. Hooke 1676.

Ut tensio sic vis.

De potentia restitutiva, R. Hooke, 1678.

Until now, we have considered the fundamental concepts of stress and strain separately, so something is still missing in the framework. Also a closer look at the three equations of equilibrium shows, that they form an underdetermined system, as they contain nine unknowns, namely the six components of stress and the three components of the configuration φ. The key to the remaining six equations is the material law that links, in our case, the geometric state to stress. Such a material is called elastic, viz. the stress is a function of the configuration only. For rheologies that are more complicated or materials of other nature, e.g. gas, further quantities like temperature, electric of magnetic field densities may influence the constitutive equations.

2.5.1 Hooke's Law

The simplest interdependence between stress and (small) strain is to assume a linear law

$$\sigma_{ij} = \sum_{k,l=1}^{3} \mathcal{C}_{ij,kl} \epsilon_{kl} \qquad \text{or shortly} \qquad \sigma = \mathcal{C}\epsilon. \tag{2.5.1}$$

In one dimension, for a spring with rest length l_0 and only one strain component, given by the relative elongation, this constitutes Hooke's law, and $\mathcal{C}/l_0 =: k$ is the spring stiffness. Therefore, schematically this relation is often symbolised by a spring. In three dimensions, for a deformable solid, $\mathcal{C}$ possesses theoretically 36 entries. Because Hooke's law can be derived from an elastic energy principle, only 21 distinct entries remain for an anisotropic material. For an orthotropic material, i.e. a material with orthogonal directions of anisotropy, the number of constants is reduced to 6; for an isotropic material only two constants λ and μ remain. Hooke's law then reads

$$\sigma_{ii} = \lambda \sum_{k=1}^{3} \epsilon_{kk} + 2\mu\epsilon_{ii} \qquad \text{and} \qquad \sigma_{ij} = 2\mu\epsilon_{ij} \quad (i \neq j).$$

2.5.2 General Isotropic Materials

The question that arises next is what other material laws should be taken into account. We usually want a material to be invariant under a change of the observer or *frame-indifferent*. Hence if the configuration φ is rotated by a matrix R we expect the stress tensor $\sigma^{R\varphi}(R\nabla\varphi)$ at every point to be the same with respect to the rotated system

$$\sigma^{R\varphi}(R\nabla\varphi) = R\sigma^{\varphi}(\nabla\varphi)R^T.$$

Related is the assumption of isotropy[4], which implies, that for a rotation of the material coordinates the stress stays the same

$$\sigma^{\varphi}(\nabla\varphi) = \sigma^{\varphi}(\nabla\varphi R).$$

These two properties together with the Rivelin-Erickson theorem for matrix mappings imply for the Cauchy and the second Piola-Kirchhoff stress, that

$$\sigma(\nabla\varphi) = \sigma(C) = \beta_0 \mathrm{id} + \beta_1 C + \beta_2 C^2$$

where $\beta_i = \beta_i(\iota_C)$ are real valued functions of the principle invariants[5] $\iota_C = \{\operatorname{tr} C, \frac{1}{2}[(\operatorname{tr} C)^2 - \operatorname{tr} C^2], \det C\}$ of the right Cauchy deformation tensor C (2.2.3). As an alternative to the second principle invariant often $\iota_{2a} = \operatorname{tr} C^2 = C : C$ is used, which is also a valid invariant because $\iota_1 = \operatorname{tr} C$ holds. This result means in particular, that the mapping is only a function of the rotationally invariant tensor C – and not of the structurally more complex deformation gradient – or, equivalently, of the Green strain and therefore can be formulated in these invariants.

A striking implication of this theorem is, that the material response function near the reference configuration, i.e. for $C \to \mathrm{id}$, takes the simple form

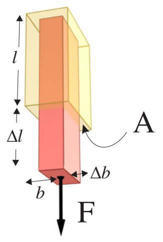
$$\sigma(\epsilon) = \lambda \operatorname{tr}(\epsilon)\mathrm{id} + 2\mu\epsilon + p_0\mathrm{id} + O(\|\epsilon\|^2), \tag{2.5.2}$$

which is exactly Hooke's law with an eventual prestress pressure term p_0. Hooke's law is therefore considered a good approximation for *any* isotropic material for small strains. The constants λ and μ are then called the Lamé constants. Note that this does not legitimate to use this approximation for large strains also and continuously adopt the Lamé constants. To exploit this, also the reference configuration would have to be adopted each time.

[4]Isotropy is a property depending on the reference configuration. For example, if an isotropic material is deformed non-uniformly and the deformed state is taken as a new reference configuration, it is no longer isotropic.

Anisotropy can be handled by an aligned material coordinate system and a transformation technique similar to the corotated strain concept.

[5]The principal invariants of a matrix C are the quantities that stay invariant under a change of basis $B^{-1}CB$. They can be identified as the coefficients of the characteristic polynomial of C.



Young's modulus:

$$\frac{\Delta l}{l} = \epsilon_{33} = \frac{\sigma_{33}}{E}$$

Poisson's ratio:

$$\frac{\Delta b}{b} = \epsilon_{11} = \epsilon_{22} = \frac{\nu}{E}\sigma_{33}$$

Figure 2.9: Standard experiment for the extraction of small strain material parameters. Only $\sigma_{33} = \frac{F}{A}$ is nonzero. Lamé's constants are related to Young's modulus E and Possion's ratio ν via the relations $\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}$ and $\mu = \frac{E}{2(1+\nu)}$.

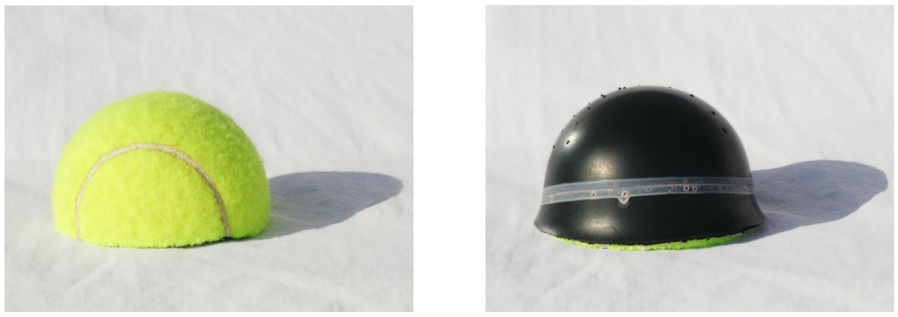


Figure 2.10: Sphere inversion. An ambiguous solution of a boundary value problem of elasticity, where vanishing traction boundary conditions were specified (pure traction problem).

2.5.3 St.Venant-Kirchhoff Elasticity

The combination of Hooke's law with Cauchy's linear strain tensor would lead to a linear partial differential equation with a well-defined solution. Easily accessible physical evidence contradicts this model (fig. 2.10). At first hand, this ambiguity can be blamed to the geometrical nonlinearity. St.Venant and Kirchhoff proposed to drop the small strain requirement and simply use Hooke's law in combination with Green's tensor for large strains. The resulting St.Venant-Kirchhoff material is the simplest among the nonlinear models[6] and quite popular in actual computations. We therefore choose it as one of the implemented models.

On the other hand, it has several shortcomings. First of all, large strain does physically not necessarily lead to large stress, as a linear law would imply. Secondly, the resulting hyperelastic energy is not polyconvex, limiting the available existence results. Last, no term in (2.5.1) prevents $\det(\nabla\varphi)$ to become 0, which of course is physically impossible. Despite all these shortcomings in practice St.Venant-Kirchhoff materials perform significantly better than linearised models [Cia92].

2.5.4 Hyperelasticity

The restriction to objective and isotropic materials already led to a substantial simplification of the material response function. A further, still very broad class are hyperelastic materials. A material is hyperelastic, if the work done during a deformation only depends on the initial and final strain. Motivated by the eventual use of a variational principle, it is conveniently defined by

[6]The mapping between φ and σ is nonlinear, though the constitutive law in terms of ϵ is linear.

its stored energy function $W(C)$. Stress is defined as the derivative of W

$$\sigma(C) = 2\frac{\partial W}{\partial C}(C) \qquad \text{or} \qquad \sigma(\epsilon) = \frac{\partial W}{\partial \epsilon}(\epsilon).$$

Interestingly, hyperelasticity can be linked to thermodynamics, more precisely to nonnegative work in closed processes. It can be shown [Gur81, Thm.28.2], that a material is hyperelastic and therefore a strain-energy density or stored energy function can be constructed, if and only if the work is nonnegative in any closed process.

As above, the energy of a hyperelastic, objective and isotropic material is a function of the invariants of C or ϵ only. The constitutive equation is given by[7]

$$\tfrac{1}{2}\sigma(C) = [\frac{\partial W}{\partial \iota_1} + \frac{\partial W}{\partial \iota_2}\iota_1 + \frac{\partial W}{\partial \iota_3}\iota_2]I - [\frac{\partial W}{\partial \iota_2} + \frac{\partial W}{\partial \iota_3}\iota_1]C + \frac{\partial W}{\partial \iota_3}C^2.$$

A St.Venant-Kirchhoff material is hyperelastic, with

$$W^{\mathrm{VK}}(\epsilon) = \frac{\lambda}{2}(\operatorname{tr}\epsilon)^2 + \mu \operatorname{tr}\epsilon^2.$$

As already mentioned $\iota_3 = \det \nabla\varphi$ is missing and W^{VK} is a convex function of ϵ.

To the stored energy function, physical requirements can be posed, such as $\det \nabla\varphi \to 0^+$ implies $W \to \infty$. The physical plausibility conditions can be summarised in a coerciveness inequality

$$\exists \alpha, p, q, r > 0, \text{and } \beta \quad \text{such that} \quad W(\nabla\varphi) \geq \alpha\big[\|\nabla\varphi\|^p + \|\operatorname{Cof}\nabla\varphi\|^q + (\det\nabla\varphi)^r\big] + \beta \quad (2.5.3)$$

for all $\det \nabla\varphi > 0$, which is the first ingredient for existence proofs. Here, we use $\operatorname{Cof}(A) = \det(A)A^{-T}$ as a notational abbreviation only. Generally existence proofs are very difficult, as for example the usual second ingredient, the convexity of W is incompatible with $W \to \infty$ for $\det\nabla\varphi \to 0^+$ [Cia92]. John Ball proposed to replace it by polyconvexity, which is convexity of W interpreted as a function of $(\nabla\varphi, \operatorname{Cof}\nabla\varphi, \det\nabla\varphi)$. Polyconvexity as a second ingredient allows some existence results. Finally a material with[8]

$$W(\nabla\varphi) = a_1 \operatorname{tr}(C) + a_2 \operatorname{tr}(C^2) + b \operatorname{tr}(\operatorname{Cof}(C)) + \Gamma(\det\nabla\varphi) + e$$

for any $a_i, b, c, d, e > 0$ and $\Gamma(x) = cx^2 - d\log(x)$ is polyconvex, satisfies the coerciveness equations (2.5.3) and locally corresponds to a linear material as in equation (2.5.2). Specialisations of this material are for example

$$\begin{aligned} W(\nabla\varphi) &= a\operatorname{tr}(C) + \Gamma(\det\nabla\varphi) && \text{(Compressible Neo-Hooke)} \\ W(\nabla\varphi) &= a\operatorname{tr}(C) + b\operatorname{tr}(\operatorname{Cof}(C)) + \Gamma(\det\nabla\varphi) && \text{(Compr. Mooney-Rivlin)}, \end{aligned}$$

and a generalisation is given by Odgen's material

$$W(F) = \sum_{i=1}^{M} a_i \operatorname{tr}(C)^{\gamma_i/2} + \sum_{i=1}^{N} \operatorname{tr}(\operatorname{Cof}(C))^{\delta_i/2} + \Gamma(\det\nabla\varphi)$$

for $M, N \in \mathbb{N}$ and $\delta_i, \gamma_i > 1$.

[7]This reveals a hidden constraint to W. If we assume a natural, unstressed rest state, for $C = \mathrm{id}$ the equation must evaluate to zero.

[8]Note that this is can also be written as $W(\nabla\varphi) = a_1\iota_1 + a_2\iota_{2a} + b\iota_2 + \Gamma(\iota_3)$.

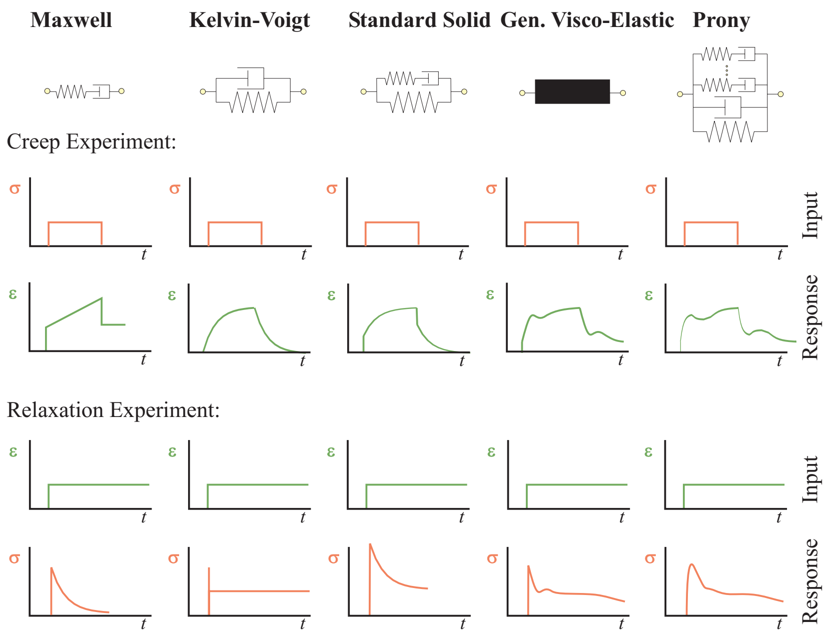


Figure 2.11: Overview over different viscoelastic laws and their behaviour for relaxation and creep experiments.

2.6 Viscoelasticity

Few biological tissues obey Hooke's law.
Biomechanics: Mechanical Properties of Living Tissues,
Y.C. Fung, 1993.

When a piece of tissue is strained, e.g. grabbed with a forceps, and held, the stresses will decrease with time. This phenomenon is called stress relaxation. If it is subjected to a constant traction or stress, the tissue will continue to deform, which is called creep. If it is repeatedly loaded and unloaded, the load and unload forces will have a phase lag, leading to dissipation of mechanical energy, which is called hysteresis. Simple elasticity fails to describe this behaviour, called *viscoelasticity*. For a viscoelastic material, the effective stiffness depends on the rate of deformation. On the other hand, for a dynamical simulation including inertia, viscous damping forces are required for the system to come to rest instead of performing a periodic motion.

Because the material seems to 'remember' the history of loading conditions for its response, stress needs to be a function of $(\epsilon_\tau)_{\tau<t}$. The most common approach to model this is by assuming linear viscoelasticity and summing up all step responses to differential changes in strain in a hereditary integral

$$\sigma(t) = \int_{-\infty}^{t} \hat{G}(t-\tau)\dot{\epsilon}(\tau)\, d\tau. \tag{2.6.1}$$

The function $\hat{G}$ is the response function to a unit step in stress, the relaxation function. Having a computer implementation in mind, this model is too complex, as it would be necessary to store the history of strain $(\epsilon_\tau)_{\tau<t}$ to evaluate the integral.

Therefore the relaxation function will be approximated by an exponential Prony series

$$\hat{G}(t) \approx G(t) := \mu_0 + \sum_{i=0}^{m} \mu_i e^{-t/\lambda_i}.$$

This approach has been developed in engineering [ZWK68]. The choice of a Prony series expansion is not at random. First, for a given $t_0 < t$, it easily allows to split the material law (2.6.1) into two terms: one representing the evolution of strain history beyond t_0 and the other one taking care of the strain during the interval $[t_0, t]$

$$\sigma(t) = \mu_0 \epsilon(t) + \sum_{i=0}^{m} \mu_i \Big[e^{-\frac{t-t_0}{\lambda_i}} \int_{-\infty}^{t_0} e^{-\frac{t_0-\tau}{\lambda_i}} \dot{\epsilon}(\tau)\, d\tau + \int_{t_0}^{t} e^{-\frac{t-\tau}{\lambda_i}} \dot{\epsilon}(\tau)\, d\tau \Big].$$

This leads to the definition of the ith memory parameter q_i as

$$q_i(t) := \int_{-\infty}^{t} e^{-\frac{t-\tau}{\lambda_i}} \dot{\epsilon}(\tau)\, d\tau.$$

Memory parameters are internal variables that store the strain history of the material. Finally approximating the strain rate tensor $\dot{\epsilon}(t) \approx (\epsilon(t) - \epsilon(t_0))/(t - t_0)$ linearly during a time step gives an update scheme for the memory parameters and ultimately strain, viz.

$$q_i(t) = e^{-\Delta t/\lambda_i} q_i(t_0) + \Delta q_i \tag{2.6.2}$$

$$\Delta q_i = (\lambda_i/\Delta t)\,(1 - e^{\Delta t/\lambda_i})\big(\epsilon(t) - \epsilon(t_0)\big) \tag{2.6.3}$$

$$\sigma(t) = \mu_0 \epsilon(t) + \sum_{i=0}^{m} \mu_i q_i(t), \tag{2.6.4}$$

where $\Delta t = t - t_n, t > t_n$. Inside a numerical time integration scheme, the memory parameter array q_i needs to be updated when advancing a time step and represents the history component of the strain.

The second reason, which motivated this scheme, is, that in 1D it represents the spring-damper combination shown in the last column of figure 2.11. Using only one memory parameter and no parallel damper this is the well known standard solid (3$^{\text{rd}}$ column of fig. 2.11). The memory parameters are the elongation of the dampers in figure 2.11. Note though, that a continuous tree-dimensional solid, given by a Prony series law and discretised by finite elements, is not equivalent to a mass-spring network.

2.7 Measurements

Sine experientia nihil sufficienter scrire potest.
Cited from [HW96], Inscription overlooking the Botanic Garden, Oxford.

...the response function which is essentially determined by experiments
Mathematical Elasticity, P.G.Ciarlet, 1988.

Now that we have laid out the basics of elasticity, we follow the principles formulated in the introduction and look at the physical properties of soft tissue by measurements. We explicitly state at this point, that all instruments, experiments, and measurements in the ElastoMedTrain project have been designed by our colleague J. Groß.

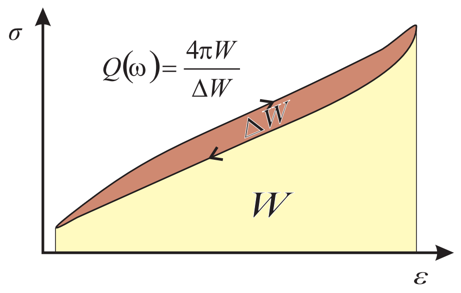


Figure 2.12: The definition of the mechanical quality factor Q.

The biomechanical reference work of Fung [Fun93, p. 281] states: ‘The hysteresis curves of most biological soft tissues have a salient feature: the hysteresis loop is almost independent of the strain rate within several decades of the rate variation’. Several experiments, first with bovine liver ex-vivo by Groß[GHE+00], then with ROSA-I [Sai01] and ROSA-II with pig liver in vivo, were performed to confirm this.

A simple description for a hysteresis loop is the mechanical quality factor Q. It is defined by

$$Q := \frac{4\pi W}{\Delta W},$$

where W denotes the mean stored elastic energy during a hysteresis loop, and ΔW is the energy loss (fig. 2.12). Consequently a material with $Q = 0$ absorbs all mechanical energy, like a fluid, and a high Q material possesses almost no hysteresis, like a steel spring. Q may depend on the speed the experiment is run, i.e. $Q = Q(\omega)$. An equivalent approach to define the quality factor is to employ Hook’s law in the frequency domain, with a nonconstant modulus matrix M with complex entries

$$\sigma(\omega) = M(\omega)\epsilon(\omega),$$

and set

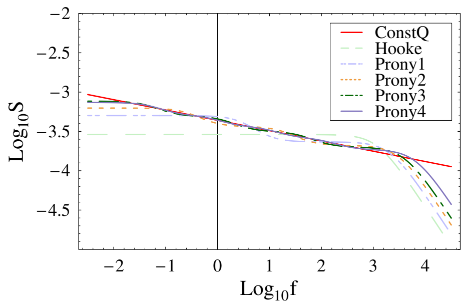
$$Q(\omega) = \frac{\operatorname{Re} M(\omega)}{\operatorname{Im} M(\omega)}.$$

Thus, a constant Q value means that the ratio between stiffness and damping constant is independent of how fast the hysteresis loop is traversed. The related relaxation function $\hat{G}$ of a material with constant Q is quite complicated [Kja79]. With $\gamma = 1/\tan(\pi Q)$ and M_0 being a reference modulus at t_0, we have

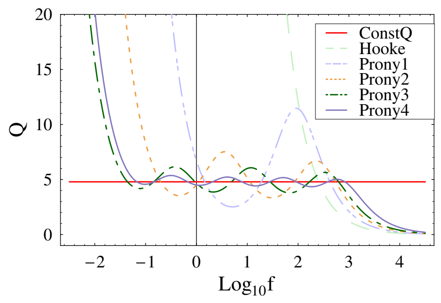
$$\hat{G}(t) = \frac{M_0}{\Gamma(1-2\gamma)}\left(\frac{t}{t_0}\right)^{-2\gamma}, \tag{2.7.1}$$

with the gamma function Γ.

To simulate a constant Q material, the parameters λ_i, μ_i of a Prony series element are fitted to the relaxation function. Instead of directly performing the nonlinear fit on (2.7.1) we rather match the compliance functions $S = M^{-1}$ of both models, minimising the relative L_2-error. This eases the numerical part of the fit and gives a good reproduction of relaxation *and* creep experiments simultaneously. A last observation eases this fit, as the constant Q model in general leads to a infinite, continuous hysteresis spectrum. The phenomena we are interested in for visual simulation are in the range of human perception, which is rated between 10 kHz (haptics) to 0.1 Hz. Therefore, it suffices to fit the constant Q behaviour in this range. Figure 2.13 shows some fits with 2 to 5 memory parameters. For comparison, a single KV-element (’Hooke’) has been added, where we note that its mechanical quality $Q(\omega) = k/(\omega d)$ depends on the frequency.



(a) Compliance S



(b) Mechanical Quality Q

Figure 2.13: Constant Q fits in the frequency domain.

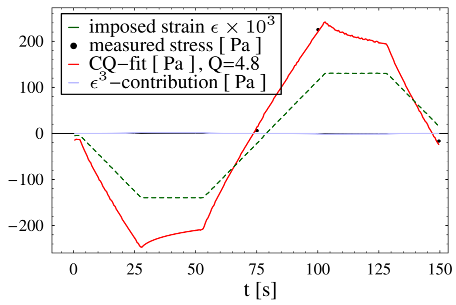


Figure 2.14: Fitting measured bovine-liver.

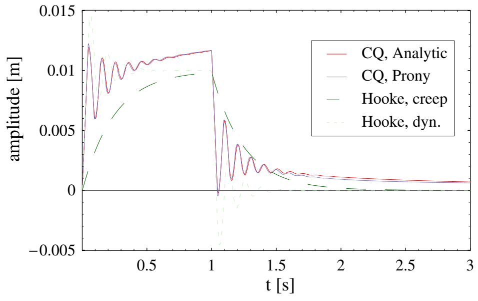


Figure 2.15: Creep experiments.

In figure 2.14, the quality of the constant Q-fit is compared to measured data. The data was obtained by Groß by imposing the shown shear *strain* on a piece of bovine liver, glued between two aluminium plates of a measurement facility. The red curve shows the results of fitting the linear constant Q model. To rate the quality of the linear hysteresis model, the next possible higher order is considered, adding a cubical dependency of the strain tensor ϵ, which did not improve the results, as the weight of this term is negligible.

In figure 2.15 the result of a one-dimensional relaxation and creep experiment is provided. The figure shows the response of a one dimensional probe of 0.1m length to a 1s unit step in *stress*. The modulus was set to 10 kPa. The analytical response of the constant Q material was computed using the FFT techniques from Kjartansson [Kja79]. The fit with a chain of 40 Prony series element with three memory parameters each comes very close.

For comparison with Hooke's law two fits with Kelvin-Voigt elements are added. The relaxation and creep function of these elements are given by exponentials, therefore the material law gives only an exponentially damped sinus response and is not able to show visual creep unless strongly overdamped. The first Hooke material shown was fitted to match the creep between 0.5 and 1s, the second one to match the initial slope.

The simulator however, is not limited to this material model. As an alternative model, we chose the one measured by Kauer, Vuskovic et al. [KVD$^+$01] and Nava at al. [NMK$^+$03]. They assume a hyperelastic material with a stored energy function in reduced polynomial form

$$W = \sum_{n=1}^{N} a_n(\iota_3^{-\frac{1}{3}}\iota_{2a})^n + \frac{1}{D}(\iota_1 - 1), \tag{2.7.2}$$

where the term $\iota_3^{-\frac{1}{3}}$ ensures, that the first part only affects the deviatoric part of strain, and $\frac{1}{D}$ admits a small material compressibility. Using the differentials of the invariants $\iota_3(C)$ and $\iota_{2a}(C)$ the stress tensor can be derived from the stored energy function (2.7.2), viz.

$$\sigma = \frac{\partial W}{\partial C} = \sum_{n=1}^{N} a_n n(\iota_{2a} - 3)^{n-1}\big\{ -\frac{1}{3}\iota_3^{-\frac{2}{3}}\iota_{2a}C^{-1} + 2\iota_3^{-\frac{2}{3}}C\big\} + \frac{1}{2D}\iota_3 C^{-1}.$$

To permit viscoelasticity, the coefficients a_i are given as a Prony series

$$a_n(t) = \mu_n^{\infty} - \sum_{i=0}^{m} \mu_{n,i} e^{-t/\lambda_i}.$$

Material properties are measured by an aspiration experiment and an inverse finite element computation is used to fit the parameters [KVD⁺01]. Currently Nava constructs a fit for $N = 5$ and $m = 4$.

2.8 Summary

In this section, we defined the rotational invariant, nonlinear Green strain and the linear Cauchy strain, which can both be computed from the gradient of the displacement field or from the configuration mapping. Combined with stress, strain allows the formulation of the equilibrium conditions of a deformable body, which are most conveniently stated by the weak equation (2.4.4) in material coordinates. For an elastic material, stress is a function of the invariants of strain only, and – in the viscoelastic case – of their history. This allows very often the restriction to hyperelastic materials, given by a stored energy function. Viscosity is in general handled by fitting a Prony series, which is well suited to be combined with numerical time integration. The actual form of the strain energy density and the coefficients are determined by measurements.

Chapter 3

Numerical Implementation

The last chapter sketched the physical and mathematical settings leading to a partial differential equation. This one will discuss the numerical basics of the implementation. First the partial differential equation will be discretised in space by finite elements [ZT00, Bat82], yielding an ordinary differential equation, which has to be solved by suitable numerical time integration methods [HW96]. Implicit integration methods require the solution of possibly large systems of equations that may be nonlinear. Therefore, efficient implementations of Newton's method will be discussed. Before concluding with a summary, some candidates for the embedded linear system solver will be presented.

The presentation of the implicit integration methods partly coincides with our tutorials given at Eurographics [HEE+02], Siggraph [KBF+03] and IEEE Visualisation/MICCAI [BHM03a, BHM03b]. Further details on the toy example can be found in [HES03].

3.1 Finite Elements

The limitations of the human mind are such that it cannot
grasp the behaviour of its complex surroundings and
creations in one operation. Thus the process of subdividing
all systems into their individual components or 'elements',
whose behaviour is readily understood,[...] is a natural way
The Finite Element Method, O.C. Zienkiewicz, 1967, 2000.

The finite element method seeks an approximate solution of a partial differential equation that minimises the error in a certain norm, usually defined by the function space of the solution. Because this space often has a non-finite dimension, it is necessary to project the solution into a finite dimensional subspace, defined by the finite element mesh and the shape functions. Consequently the error between the projection and a linear combination of these functions is minimised. The weights, given as the solution of a nonlinear system, can usually be geometrically interpreted as the position of the nodes. The norm is typically an integral norm, hence the solution is defined on the whole space and can be interpolated between the nodes.

Starting point for a finite element approach is the variational equation in spatial (2.4.3)

$$\int_{\Omega^\varphi} \delta D : \sigma^\varphi \, dx^\varphi = \int_{\Omega^\varphi} \delta v f^\varphi \, dx^\varphi + \int_{\partial\Omega^\varphi} \delta v g^\varphi \, dA^\varphi$$

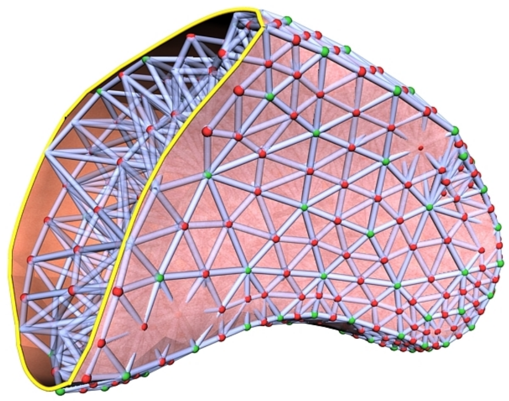


Figure 3.1: A finite element mesh of a liver shaped object.

or material coordinates (2.4.4)

$$\int_\Omega \delta\epsilon : \sigma \, dx = \int_\Omega \delta v f \, dx + \int_{\partial\Omega} \delta v g \, dA. \tag{3.1.1}$$

where σ and ϵ have to be work conjugate, like e.g. Green's strain and the second Piola-Kirchhoff stress (see section 2.4). For a total Lagrangian approach, which uses a single, fixed material coordinate system as reference, the second equation is suited best.

The functions u and δv are replaced by a locally supported, piecewise polynomial approximation over a decomposition of Ω into disjoint elements of simple shape, e.g. tetrahedra or hexahedra. The basis or shape functions are uniquely defined by the degree and the property

$$\phi_i(x_j) = \delta_{ij},$$

for the vertices x_j of the discretisation mesh. Hence u will be approximated by

$$u(x,t) = \sum_{i=0}^{N} \mu_i(t)\phi_i(x), \qquad \mu_i(t) = u(x_i,t),$$

and represented by the coordinate vector $[\mu_i(t)]_{i=1..N}$. In a Galerkin approach, which delivers the 'best' convergence and will be used here, the variation δv is taken from the same space and accordingly is represented by $[\zeta_i(t)]_{i=1..N}$. The finite element interpolation can be written in matrix form

$$u(x,t) = [\Phi(x)][\mu_i(t)].$$

Being the first variation of ϵ, $\delta\epsilon$ is also linear in $[\zeta_i]$ (section A.2)

$$\delta\epsilon = [B(x,[\mu_i])][\zeta_i].$$

For a more compact notation, we will again omit the dependence on t in this section. Inserting these equations in the principle of virtual work (3.1.1) gives

$$[\zeta_i] \int_\Omega [B(x,[\mu_i])] : \sigma \, dx = [\zeta_i] \Big\{ \int_\Omega [\Phi(x)]^T f \, dx + \int_{\partial\Omega} [\Phi(x)]^T g \, dA \Big\}.$$

Because the equation has to be satisfied for all variations $[\zeta_i]$, it is equivalent to the $3N$ equations

$$\int_\Omega [B(x,[\mu_i])] : \sigma \, dx = \int_\Omega [\Phi(x)]^T f \, dx + \int_{\partial\Omega} [\Phi(x)]^T g \, dA. \tag{3.1.2}$$

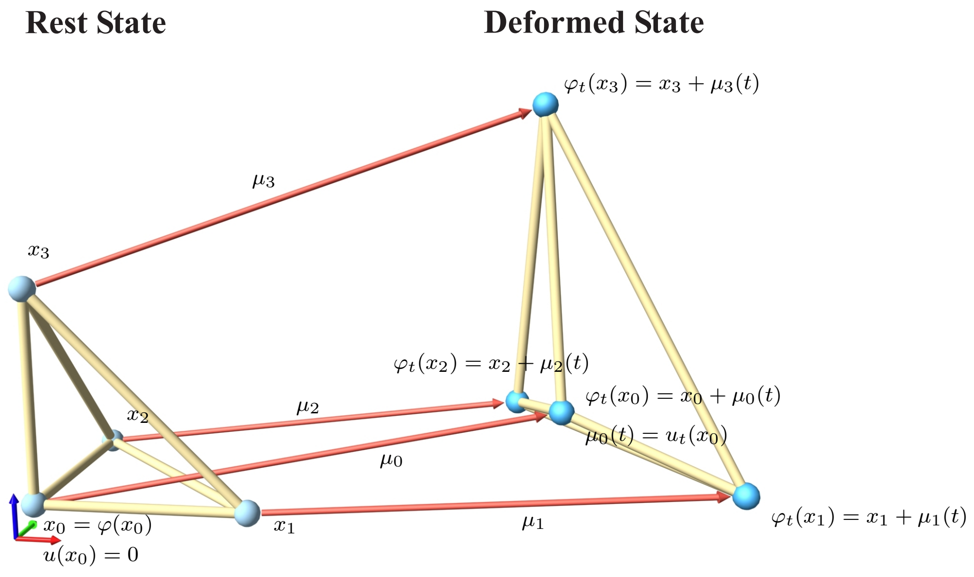

Rest State
Deformed State
$\varphi_t(x_3) = x_3 + \mu_3(t)$
μ_3
x_3
$\varphi_t(x_2) = x_2 + \mu_2(t)$
$\varphi_t(x_0) = x_0 + \mu_0(t)$
x_2
μ_2
$\mu_0(t) = u_t(x_0)$
μ_0
$x_0 = \varphi(x_0)$
x_1
μ_1
$\varphi_t(x_1) = x_1 + \mu_1(t)$
$u(x_0) = 0$

Figure 3.2: State coefficients μ of a tetrahedral element. The shape coefficients α can be computed from the rest coordinates (Appendix C.1).

All integrals are evaluated locally and piecewise over the individual elements, and then summed up.

3.1.1 Solid Tetrahedral Elements

Tetrahedra are usually the preferred elements in computer graphics because of their superior shape-fitting properties. Furthermore, the proposed discretisation employs linear shape functions ϕ_i, resulting in a piecewise affine approximation of the displacement field u. As a result, ϵ is constant over each tetrahedron, greatly reducing the computational cost. The disadvantage of this choice is the theoretical chance to suffer from locking effects (see section 3.1.4).

To make the discussion more concrete we provide the key equations for the selected element type. A common technique to simplify the derivation is to evaluate all integrals over a reference element, e.g. the unit tetrahedron and use affine or isoparametric transformations to map the elements. In our case, this means eight affine 3D-transformations, requiring a total of 144 floating-point operations. This gives an enhanced flexibility and ease of derivation, but in our opinion is too expensive for a specialised interactive application. Hence we write the interpolation already with general shape coefficients α_{nl}

$$u_j = \sum_{n=1}^{4} \alpha_{n0}\mu_{nj} + \sum_{n=1}^{4}\sum_{l=1}^{3} \alpha_{nl} x_l \mu_{nj} = \sum_{n=1}^{4} (\alpha_{n0} + \sum_{l=1}^{3} \alpha_{nl} x_l)\mu_{nj},$$

μ_{nj} being the jth coordinate of the coefficient of vertex n in a local vertex-numbering scheme. With $P(x) = [\alpha_{10} + \sum_{l=1}^{3} \alpha_{1l} x_l | \dots | \alpha_{40} + \sum_{l=1}^{3} \alpha_{4l} x_l]$ we have

$$\Phi(x) = [P(x) \otimes \mathrm{id}_{3\times 3}]\mu,$$

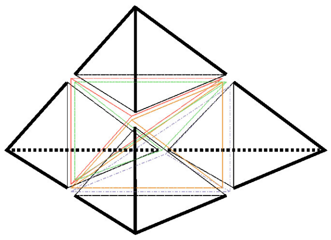


Figure 3.3: Octasection of a tetrahedron.

where $\otimes$ denotes the Kronecker tensor product. Hence, the deformation gradient $F := \nabla\varphi$ is

$$F := [f_{ij}] = \left[\frac{\partial \varphi_i}{\partial x_j}\right]_{i,j} = \left[\sum_{n=1}^{4} \alpha_{nj}\mu_{ni} + \delta_{ij}\right]_{i,j}$$

and consequently Green's strain tensor reads

$$\epsilon^G = \left[\sum_{k=1}^{3} f_{ki}f_{kj} - \delta_{ij}\right]_{i,j}.$$

Using one of the material laws proposed in the previous chapter and eventually the memory parameter array of the element, the second Piola-Kirchhoff stress σ can be computed.

The matrix $B := \delta\epsilon$ results from the variation of ϵ (Proposition A.5)

$$\delta\epsilon = \frac{1}{2}\left(\nabla\delta v^T\nabla\varphi + \nabla\varphi^T\nabla\delta v\right)$$

which gives

$$\delta\epsilon_{ij} = [\alpha_{ni}f_{kj} + \alpha_{nj}f_{ki}]^T_{(nk)=(11)..(43)}\left[\zeta_{(nk)}\right]_{(nk)=(11)..(43)}.$$

Therefore the contribution of the considered tetrahedron to the (nk)-th line of equation (3.1.2), which contributes to the kth force component of the current vertex x_n, reads

$$\sum_{i,j=1}^{3}\left\{\alpha_{ni}\sum_{l=1}^{4}\alpha_{lj}\mu_{lk} + \alpha_{nj}\sum_{l=1}^{4}\alpha_{li}\mu_{lk} + \alpha_{ni}\delta_{kj} + \delta_{ki}\alpha_{nj}\right\}\sigma_{ij}.$$

In appendix C.2 we provide some more details and show, how this computation can be carried out with 300 floating point operations per tetrahedron.

3.1.2 Hierarchical Approximations

A key concept in computer graphics is to make use of level-of-detail representations in order to add detail only where necessary. In the finite element setting this corresponds to the use of an adaptive family of basis functions. There are two different possibilities to refine a given basis: Either rise the degree of the shape functions (p-refinement) or refine the geometry (h-refinement). The geometry can be refined either by inserting points and re-meshing or by subdividing individual elements. The fastest technique is the subdivision h-refinement, which was therefore selected as most suitable. As a first step, a series of nested meshes is generated by subdivision of the initial mesh. In the second step, a hierarchy of nested function spaces is constructed on the subdivided meshes. This is most conveniently done by hierarchical approximations [Ban96].

```
for $k = l$ to 1
  for $x_i \in N_k \backslash N_{k-1}$
    $x_{p_{1,i}} += x_i/2$
    $x_{p_{2,i}} += x_i/2$
```

Algorithm 3.3: Transforming ϕ to ψ.

For a given coarse tetrahedral discretisation T_0 it is possible to construct a series of nested meshes

$$T_0 \subset T_1 \subset T_2 \subset \dots$$

by octasection of each tetrahedron (fig. 3.3). The inserted nodes are on the edges of the mother tetrahedron, i.e. each finer level node has exactly two parent nodes. If the diagonal of the inner octahedron is chosen properly, the resulting sequence is stable, i.e. the angles are uniformly bounded away from zero [Bey97]. We denote the node set on each level by N_l.

Let $\mathcal{S}_l$ be the subspace of piecewise linear functions on T_l. With the nestedness of T_l also the function spaces $\mathcal{S}_l$ are nested. For each hierarchy level l we have a nodal basis $\{\phi_i^l\}$ as before, and an approximation u^l defined by the values at the nodes x_i^l of the mesh

$$u^l(x,t) = \sum_{i=0}^{N} \mu_i^l(t)\phi_i^l(x), \qquad \mu_i^l(t) = u(t, x_i^l). \tag{3.1.3}$$

Using a hierarchical basis, the function is represented by its values at the nodes of the coarsest level and the difference between the nodal value at the next finer level and the linear interpolation of its two parent nodes (fig. 3.4(c)). The basis functions itself are defined by [Yse92]

$$\begin{aligned} \eta_i^1 &= \phi_i^1, \\ \eta_i^l &= \begin{cases} \eta_i^{l-1} & \text{node } i \in N_{l-1} \\ \phi_i^l & \text{node } i \in N_l \backslash N_{l-1} \end{cases} \qquad l > 1. \end{aligned} \tag{3.1.4}$$

The resulting basis is illustrated in figure 3.4(a). On the left we have the nodal basis $\{\phi_i^l\}$ on 3 levels, in wavelet theory these are the scaling functions. On the right we have the hierarchical wavelet basis $\{\psi_i^l\}$, spanning the same space. Now the coarse level basis functions are a subset of the finer level basis. Note that the wavelets are not orthogonal and cease to be a partition of unity.

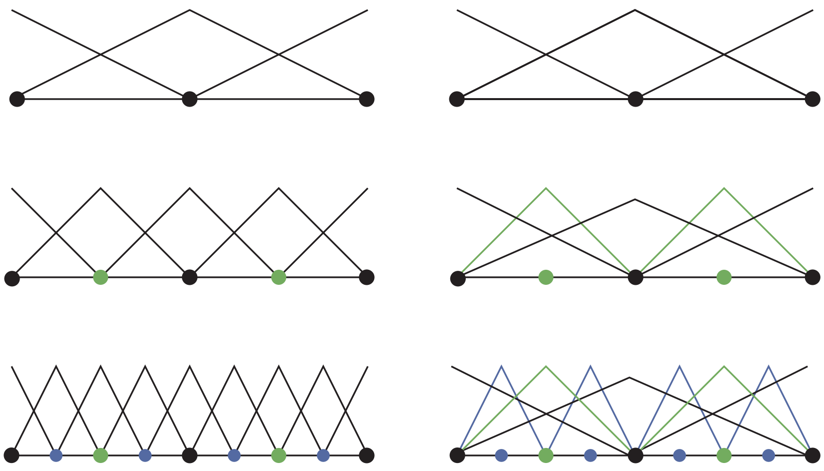
A key to the efficiency of algorithms linked to hierarchical representations are the fast basis transformations [Yse92]. For $u^l(x,t) = \sum_{i=0}^N \eta_i^l(t)\psi_i^l(x)$ we compute the coefficient vector μ of the nodal representation (3.1.3) by evaluating u^l at the nodes. The mapping $S : \eta^l \to \mu^l$ can be decomposed into transformations $S_1 \dots S_k$ iterating from the coarsest to the finest level, always evaluating the interpolating part from the two parents p_1 and p_2, then adding the detail coefficient (fig. 3.4). The resulting pseudo-code is given in algorithm 3.1. Quite obviously the inverse transformation S^{-1}, given in algorithm 3.2, can be deduced from this. Note that $S^{-1} = S_k^{-1} \dots S_1^{-1}$, so the loop is downwards.

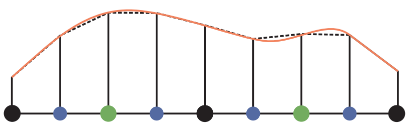
For the transformations between basis functions from $\{\psi_i\}$ to $\{\phi_i\}$ we use that

$$\sum_i \eta_i^l \psi_i^l = u^l = \sum_i \mu_i^l \phi_i^l = \sum_i [S\eta^l]|_i \phi_i^l,$$

Using vector-notation, this equation reads

$$\langle \eta^l, \psi^l \rangle = \langle \mu^l, \phi^l \rangle = \langle S\eta^l, \phi^l \rangle$$


(a) Function hierarchy. Standard nodal basis left, hierarchical right.


(b) Nodal basis, μ given by vertical lines.

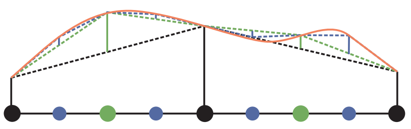

(c) Hierarchical basis, η given by vertical lines.

Figure 3.4: Standard and hierarchical function approximation in 1D.

```
$x = \eta$
**for** k=1 to $l$
    **for** $x_i \in N_k \backslash N_{k-1}$
        $x_i = x_i + \frac{1}{2}(x_{p_{1,i}} + x_{p_{2,i}})$
$\mu = x$
```

Algorithm 3.1: Transforming η to μ.

```
$x = \mu$
**for** k=$l$ downto 1
    **for** $x_i \in N_k \backslash N_{k-1}$
        $x_i = x_i - \frac{1}{2}(x_{p_{1,i}} + x_{p_{2,i}})$
$\eta = x$
```

Algorithm 3.2: Transforming μ to η.

Accordingly the transposed of S transforms the basis $\{\phi_i\}$ to $\{\psi_i\}$ by

$$S^T \phi^l = \psi^l,$$

and algorithm 3.3 follows. Thus we have the following diagrams:

$$\eta \underset{S^{-1}}{\overset{S}{\rightleftarrows}} \mu \qquad \text{and} \qquad \psi \underset{S^{T}}{\overset{S^{-T}}{\rightleftarrows}} \phi.$$

For the operator B (3.1), mapping displacements to the first variation of strain, we have with the chain rule

$$\begin{aligned}\frac{\partial \epsilon_\eta}{\partial \eta}(\eta) : \mathcal{C}(\epsilon_\eta(\eta)) = (S\frac{\partial \epsilon_\mu}{\partial \mu}(S\eta)) : \mathcal{C}(\epsilon_\mu(S\eta)) &= \big((S\frac{\partial \epsilon_\mu}{\partial \mu}(S\eta)\big) : \mathcal{C}(\epsilon_\mu(S\eta)) \\ &= S^T\big(\frac{\partial \epsilon_\mu}{\partial \mu} : \mathcal{C}(\epsilon_\mu)\big)(S\eta)\end{aligned}$$

and an analogous result for the viscous part. In each evaluation step we therefore perform an inverse wavelet transform, evaluate in the nodal basis and transform the result back by S^T. Similarly, the traction and body loads are computed in the nodal basis and then transformed back.

This makes an implementation straightforward. Once the code is able to solve the system in the nodal basis, one adds three lines for each of the transformations and applies them at the appropriate places.

3.1.3 Corotational Formulation

In the previous chapter, the corotated Cauchy strain was introduced as a rotational invariant strain measure that is still linear. However, the definition assumed that the rotation tensor field of the body is known. In this section, we describe how we embed this issue into the finite element setting. Like the strain tensor, we discretise the rotation tensor field $R(x)$ per element. This results in a finer representation of the tensor field than the per vertex discretisation of Müller et al. [MMD+02] and is more consistent with the finite element modelling paradigm. In addition, the resulting Jacobian matrix stays symmetric in contrast to a discretisation per vertex.

Another major difference to the work of Müller is the way the rotations are extracted. They 'found that the stability is not sensitive to the rotation field' and therefore use a heuristics called warping. It extracts rotations based on the deterministic selection of three edges per point of the mesh and tracks their rotations. This is sufficient for a rigid body movement superimposed on a deformed configuration, because the warping roughly compensates it and therefore does not produce large ghost forces.

However, accuracy suffers this approximation, as it is not directly clear, how the rigid body mode of an arbitrarily deformed element looks like. A way to extract a well defined rotation, is given by employing the polar decomposition of the deformation gradient $F := \nabla\varphi$. The polar decomposition solves the following problem [SD92, Cia92]

$$\text{Find an orthogonal } R \text{ minimizing } \|F - R\|_{\mathcal{F}}^2,$$

with the Frobenius norm $\|\cdot\|_{\mathcal{F}}$. It gives rise to a unique decomposition $F = RT$, with a symmetric matrix T and orthogonal R, and therefore reduces (2.2.2) to

$$\epsilon^G(\phi) = \frac{1}{2}(T^T T - \text{id}),$$

where T does not contain any rotational component. The factor R is the rotation closest to the deformation gradient in the space of matrices equipped by the Frobenius norm.

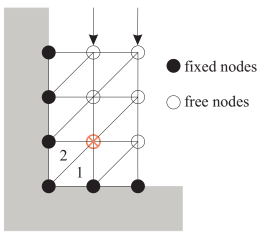
In two dimensions, this rotation is straightforward to compute. First, calculate

$$\hat{R} = F + \text{sign}(\det(F)) \left[\begin{pmatrix} f_{22} & -f_{21} \\ f_{12} & f_{11} \end{pmatrix} \right],$$

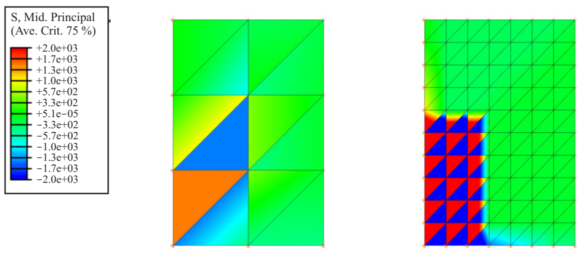
and a subsequent normalisation of the columns gives R. In higher dimensions, Higham [HS90] proposed an efficient, quadratically convergent iteration scheme

$$\begin{aligned} R^{(0)} &:= F \\ R^{(n+1)} &:= \frac{1}{2}\left(R^{(n)} + {R^{(n)}}^{-T}\right). \end{aligned}$$

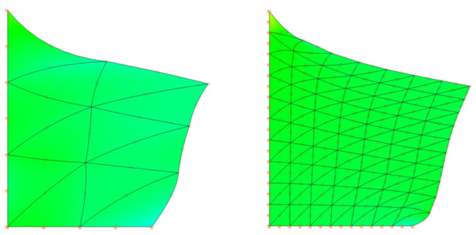
Because the deformation gradient may be singular, e.g. in case of a pure rotation around one of the coordinate axes, we use a QR decomposition ahead of the core algorithm as also proposed by Higham. This results in a very robust and fast algorithm. Seldom more than three iterations are required. A soon as R is known, we can compute the strain and proceed as before.



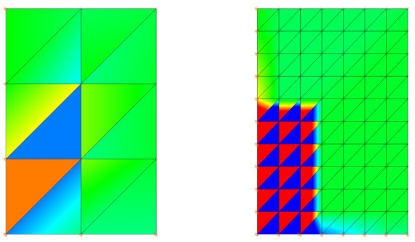
(a) A locking experiment.



(b) Linear triangles.



(c) Quadratic triangles with constant pressure.



(d) Linear triangles with decoupled pressure.

Figure 3.5: When linear triangles fail: (a) due to the incompressibility of Triangle 1 the red node may only move horizontally. Due to the incompressibility of triangle 2 and the wall, it may only move vertically. Hence, it will not move at all. The same argument can be propagated through the whole mesh. As a result, the whole structure will seem rigid, independent of the shear modulus. The simulations (b)-(d) have been performed using ABAQUS/Standard.

The last open issue for a corotational simulation is that the deformation gradient itself is one of the unknowns. In practice, this is solved by exploiting temporal coherence and use F from the previous time step or, in case of a hierarchical simulation, simulate the uppermost layer fully nonlinear and propagate the rotations successively to the finer levels.

3.1.4 Locking and Mixed Elements

As the example in figure 3.5 shows, in the incompressible case linear elements may fail (3.5(b)). Although we didn't implement any of the following issues, we report briefly on the problem. It can be explained by the Lemma of Céa [Bra97], which is part of the error analysis for finite elements and links the polynomial order of convergence to the coerciveness of the weak equation[1]. Céa's lemma hints that if ν approaches 0.5 the problem is arbitrary bad conditioned. In this case, the volumetric part (see equations (2.3.2) and (2.2.4)) is better formulated as a separate weak equation (cf. [ZT00]) and linked by Lagrangian multipliers, which can be identified as the

[1]That is, in the small displacement limit we have $a(u, \delta v) = \delta\epsilon : \sigma = \lambda\langle \operatorname{div} u, \operatorname{div} \delta v\rangle + 2\mu\langle D, \delta D\rangle$ and the constants of the coerciveness-inequality $\alpha\|v\| \leq a(v, v) \leq C\|v\|$ satisfy $\alpha \leq \mu$ and $C \geq \lambda + \mu$. The Lemma of Céa states, that the error is bounded by C/α times the optimal approximation error in the finite element space. Unfortunately for $\nu \longrightarrow 0.5$ the constant C/α tends to infinity.

pressure

$$\int_\Omega \delta\epsilon : \sigma^{Dev}\, dx + \int_\Omega \delta\epsilon_M : p\, dx = \int_\Omega \delta v f\, dx + \int_{\partial\Omega} \delta v g\, dA. \qquad \forall \delta\epsilon$$

$$\int_\Omega q(e - \frac{3p}{\lambda})\, dx = 0 \qquad \forall q \qquad (3.1.5)$$

with $e := \frac{1}{3}\operatorname{tr}(\epsilon)$ (see eq. (2.2.4)). This is called the two-field u/p-formulation and leads to a saddle point problem.

Due to the Ladyshenskaja-Babuška-Brezzi [Bra97] conditions not every combination of elements for p and u is admissible (fig. 3.5(d)), especially linear T4 tetrahedra with decoupled linear pressure do not improve the results. The lowest order elements that work are quadratic tetrahedra with a piecewise constant pressure (fig. 3.5(c)). For T10 elements $\nabla\varphi$ is linear in the material coordinates and ϵ is quadratic, therefore we need a 11-point integration formula which is fourth order accurate but expensive to compute. The saddle-point problem itself requires an adequate numerical method as for example the Uzawa-algorithm, so that the system to solve does not double in dimension. Then again, the quadratic displacement elements will allow a reduction of the number of nodes.

An second approach to tackle the problem is selected reduced integration of the pressure term, which of course then also requires quadratic elements for the displacement but omits the saddle point problem. In the case of discontinuous pressure it can be shown equivalent to the mixed formulation [ZT00], but is computationally much more efficient. In this case, the mixed formulation itself serves as a vehicle to select the weights of the reduced quadrature formula.

The last way to deal with locking effects would be to introduce additional degrees of freedom by using bubble functions as in the MINI-element [Bra97], or to use other elements, e.g. hexahedra, with their drawbacks in shape fitting and higher computational cost. The usual trick of reduced integration for faster computation brings other disadvantages with him, which then require unphysical cures, like hourglass control [Hut99].

In practise, locking effects are often not so drastic as demonstrated in the example. Slightly changing the node positions often decreases their impact. They disturb the convergence of the space discretisation, but, as for graphical applications often coarse meshes are used, this is only a minor issue in this case. Even sophisticated, non-interactive simulations of soft tissue still employ linear tetrahedra [ZGHD00]. On the other hand, a careful parameter study and an analysis of the impact would deserve further attention.

3.1.5 Dynamics: Mass and Inertia

Although the first step to a true dynamical simulation has been made with the inclusion of viscosity, the effects of mass have still been neglected in this presentation. The addition itself is straightforward, combining the equations of equilibrium with Newton's third law – or equivalently adding a kinematic term to the energy balance – gives the spatial weak equation

$$\int_{\Omega^\varphi} \rho^\varphi \delta v \ddot{\varphi} + \int_{\Omega^\varphi} \delta D : \sigma^\varphi\, dx^\varphi = \int_{\Omega^\varphi} \delta v f^\varphi\, dx^\varphi + \int_{\partial\Omega^\varphi} \delta v g^\varphi\, dA^\varphi$$

which as an extension of (3.1.2) gives the second order ordinary differential equation

$$\int_{\Omega^\varphi} \Phi^T \rho^\varphi \Phi\, dx^\varphi \ddot{\mu} + \int_\Omega [B(x, [\mu_i])] : \sigma\, dx = \int_\Omega [\Phi(x)]^T f\, dx + \int_{\partial\Omega} [\Phi(x)]^T g\, dA. \qquad (3.1.6)$$

The matrix $M^\varphi := \int_{\Omega^\varphi} \Phi^T \rho^\varphi \Phi\, dx^\varphi$ is called the mass matrix. Equation (3.1.6) has the abstract form of an implicit ODE $M\ddot{u} = f(u, \dot{u})$. A substantial part of the numerical solution methods for ODEs, e.g. all explicit methods, are only able to solve systems of the form $\ddot{u} = f(u, \dot{u})$, therefore the mass matrix has to be inverted and the resulting ODE reads

$$\ddot{\mu} = M^{\varphi -1}\Big\{ - \int_\Omega [B(x, [\mu_i])] : \sigma \, dx + \int_\Omega [\Phi(x)]^T f \, dx + \int_{\partial\Omega} [\Phi(x)]^T g \, dA \Big\}.$$

In practise, the inversion of M^φ may be computationally very expensive, much more expensive than the time integration itself. First M^φ depends on φ, thus needs to be re-inverted when φ changes. To save costs, often the approximation $M = M^0$ is made, or M is only recomputed, when a substantial amount of elements change their volume above some limit. The approximation is exact for an incompressible material, with a constant density. A second issue is, that the multiplication with a factorised matrix still may be costly due to the fill-in (section 3.4.2). A usual procedure is mass lumping, that is to concentrate the mass at the nodes to render the matrix diagonal. In contrast, the full mass matrix using the original elements is called consistent or compatible. When using a lumped mass we employ a row summation scheme, replacing M by the diagonal matrix

$$\hat{M}^\varphi = \mathrm{diag}\Big(\sum_{j=1}^{N} m_{ij}^\varphi\Big) = \mathrm{diag}(M^\varphi \mathbf{1}), \tag{3.1.7}$$

where $\mathbf{1}$ is the vector of all ones. In the hierarchical case, the second equality of (3.1.7) leads to an efficient computation of the lumped matrix. The price of mass lumping is a slight artificial increase of viscosity [ZT00] depending on the mesh.

3.2 Numerical Time Integration

Around 1960, things became completely different and everyone became aware that the world was full of stiff problems.

In Aiken: Stiff computation. Cited from [HW96],
G. Dahlquist, 1985.

The finite element method transforms the partial differential equation of elasto-dynamics into an initial value problem of second order,

$$M\mu''(t) = f_v\big(t, \mu(t), \mu'(t)\big), \quad \text{and } \mu(t_0) = \mu_0, \mu'(t_0) = v_0. \tag{3.2.1}$$

This differential equation can be transformed into a first order system by introducing velocities as a separate variable:

$$\begin{bmatrix} \mathrm{id} & 0 \\ 0 & M \end{bmatrix} \begin{bmatrix} \mu(t) \\ \nu(t) \end{bmatrix}' = \begin{bmatrix} \nu(t) \\ f_v\big(t, \mu(t), \nu(t)\big) \end{bmatrix}, \quad \text{and} \begin{bmatrix} \mu(t_0) \\ \nu(t_0) \end{bmatrix} = \begin{bmatrix} \mu_0 \\ \nu_0 \end{bmatrix}. \tag{3.2.2}$$

For the next few sections, it will be convenient to write this equation in the more abstract form

$$My'(t) = f\big(t, y(t)\big), \quad \text{and } y(t_0) = y_0, \tag{3.2.3}$$

or, because some methods cannot handle an implicit ODE, in explicit form

$$y'(t) = f_M\big(t, y(t)\big) = M^{-1} f\big(t, y(t)\big), \quad y(t_0) = y_0, \tag{3.2.4}$$

Later we will come back to the special setting (3.2.2) for eventually gaining computational advantages.

This section reports on the numerical methods used to solve the equation and which one is suited 'best'. Such a method will again produce a discretisation of y, this time approximating $y(nh)$ by Y_n at some instants $t_n := nh$ of time. There are several criteria for evaluating an integration method: convergence, accuracy, stability, and efficiency. Convergence means that for $h \to 0$ the numerical solutions Y_n meets the analytical. All useful methods must be convergent, so we won't discuss non-convergent methods or criteria for convergence. More interesting is the accuracy. By this we mean how fast a method converges for $h \to 0$. This is measured by the order. A method is defined to possess order p if $\|y(t_n) - Y_n\| = O(h^p)$. Stability will be an issue in the next subsections, and finally efficiency, the most interesting question can only be decided by experiment. Fortunately, there are some clues how to select the most promising candidates.

3.2.1 Explicit Methods

The oldest and most simple method of integration is the so called forward or explicit Euler method. To get a formula for advancing a time step h, the differential quotient on the left hand side of (3.2.4) is replaced by the forward difference quotient

$$\frac{y(t+h) - y(t)}{h} \approx y'(t) = f_M(t, y(t)).$$

Thus we obtain the integration formula for advancing a single time step

$$y(t+h) = y(t) + hf_M(t, y(t)).$$

Iterating this scheme gives a sequence of numerical approximations Y_n. Geometrically the method can be interpreted as straightly following the tangent of the solution at time t_n and then recalculating the slope for the next step.

By using a Taylor expansion for the exact solution after a single time step

$$y(t+h) = y(t) + hy'(t) + \frac{1}{2}h^2 y''(t) + O(h^3)$$

we find that the error of the numerical approximation Y_1 is $O(h^2)$. If we continue the method using the *numerical solution* Y_1 as a starting value for the next time step we lose [HNW93] a power of h for the global error, hence the explicit Euler method converges linearly or has *order* 1.

As a next step, we introduce methods of higher order. For this a centered difference estimation for $y'(t + h/2)$ in equation (3.2.4) is used

$$\frac{y(t+h) - y(t)}{h} \approx y'(t + h/2) = f(t + h/2, y(t + h/2)).$$

resulting in the iteration scheme

$$Y_{n+1} = Y_n + hf(t_n + h/2, y(t_n + h/2)). \tag{3.2.5}$$

But how do we find $y(t_n + h/2)$? For an estimation k_1 we use an explicit Euler step to get

$$\begin{aligned} k_1 &= \quad Y_n + \tfrac{h}{2} f(t_n, Y_n) && (3.2.6)\\ Y_{n+1} &= Y_n + hf(t_n + h/2, k_1), && (3.2.7)\end{aligned}$$

the so called *explicit midpoint* rule. The estimation by forward Euler, although not very accurate, is good enough, as the function evaluation is multiplied by the time step to advance to the next approximation. So by a Taylor expansion we find a local error of $O(h^3)$ leading to a global error of $O(h^2)$ for the explicit midpoint rule.

Generalizing the idea of using function evaluations at s intermediate points $t + c_j h$ leads to Runge-Kutta methods. They are defined by a Runge-Kutta matrix (a_{ij}), weights b_i, abscissae c_j, and the equations

$$\begin{aligned} k_i &= Y_n + h\sum_{j=1}^{s} a_{ij}k'_j \quad \text{with } k'_i = f(t_n + c_i h, k_i) \text{ for } \quad i = 1, \ldots, s, \\ Y_{n+1} &= Y_n + h\sum_{i=1}^{s} b_i k'_i. \end{aligned} \tag{3.2.8}$$

The coefficient set can comfortably be specified as shown in table 3.1(a). If the matrix (a_{ij})

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{12} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array}$$

(a) General Runge–Kutta method.

$$\begin{array}{c|cc} 0 & & \\ \frac{1}{2} & \frac{1}{2} & \\ \hline & 0 & 1 \end{array}$$

(b) Explicit Midpoint

$$\begin{array}{c|cccc} 0 & & & & \\ 1/2 & 1/2 & & & \\ 1/2 & 0 & 1/2 & & \\ 1 & 0 & 0 & 1 & \\ \hline & 1/6 & 2/6 & 2/6 & 1/6 \end{array}$$

(c) "The" Runge-Kutta method RK4.

Table 3.1: Runge-Kutta methods.

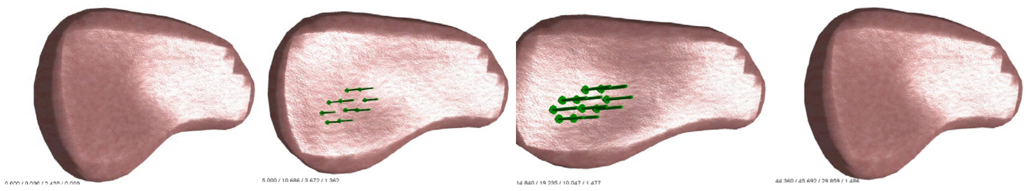
is strictly lower triangular, all inner stages k_i only depend on k_j with $j < i$ and thus can be computed one after the other. Table 3.1(b) shows the explicit midpoint rule interpreted as a Runge-Kutta method. The most famous scheme is the method by Runge and Kutta given in table 3.1(c), which possesses order 4.

By using algebraic relations for the coefficients, it is possible to construct explicit Runge-Kutta methods of arbitrary high order resulting in many inner stages with numerous evaluations. For most practical applications, order 4 is sufficient.

3.2.2 The Stability Problem

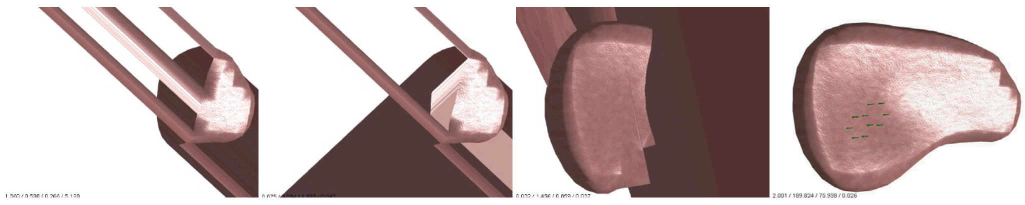
[Wir] werden bei dem Anfangswertproblem hyperbolischer Gleichungen erkennen, dass die Konvergenz allgemein nur dann vorhanden ist, wenn die Verhältnisse der Gittermaschen in verschiedenen Richtungen gewissen Ungleichungen genügen.
U. d. partiellen Differentialgleichungen der math. Physik,
Courant, Friedrichs, and Lewy,1928.

The stability problem, first recognised early in the 1900 in numerical fluid dynamics, yielded the CFL-condition [CFL28] for the explicit Euler method, restricting the time step for the numerical method to be able to capture the characteristics of the equation. Basically this means that the time step has to be small enough for a wavefront to traverse only one grid node per time step. This implies a severe increase of work, when the grid resolution is increased. Its systematic analysis evolved in the 50s and 60s [GL61], finally leading to the stability theory as used today [HW96]. The problem is illustrated in figure 3.6, where we compare the results of a simulation done with the explicit Euler, the method of Runge and Kutta, and the implicit Euler method, which will be



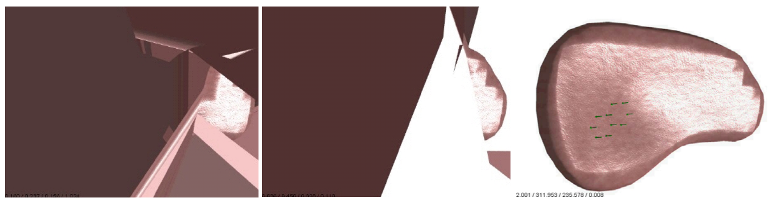
h=40ms, t=0s h=40ms, t=5.0s h=40ms, t=14s h=40ms, t=44s

(a) Several frames from a solution computed with the backward Euler method with h=0.04s.



h=40ms (unstable) h=0.4ms (unstable) h=0.3ms (unstable) h=0.2ms, t=2.0s (stable)

(b) Several frames from 'solutions' computed with the forward Euler method with different time steps. Only the smallest h=0.0002s allows a stable simulation.



h=40ms (unstable) h=0.4ms (unstable) h=0.3ms, t=2.0s (stable)

(c) Several frames from 'solutions' computed with the RK4 method with different time steps. Only the smallest h=0.0003s allows a stable simulation.

Figure 3.6: Several simulations with different methods and time steps to demonstrate the stability problem of explicit methods.

discussed below. For both explicit methods, there is a critical time step. Above this sharp limit, the simulation suddenly diverges. The larger limit of the higher order method only pretends to be a gain of efficiency: due to its four stages, the RK4 method takes 7078 seconds to simulate 60s, compared to the explicit Euler with 2511s. The implicit method, benefiting from the large time step, solves the problem in a visually pleasing manner in 40.1s.

The mathematical tool used to explain this phenomenon is *Dahlquist's test equation*

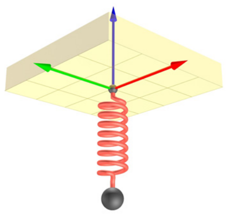
$$y' = \lambda y, \qquad \lambda \in \mathbb{C}. \tag{3.2.9}$$

Its exact solution for an initial value $y(0) = y_0$ is given by

$$y(t) = e^{\lambda t} y_0.$$

To make the problem more real-world like, we created the toy example [HES03] given in figure 3.7, which is equivalent to the equation

$$\begin{bmatrix} z \\ v_z \end{bmatrix}' = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{d}{m} \end{bmatrix} \begin{bmatrix} z \\ v_z \end{bmatrix} + \begin{bmatrix} 0 \\ -\frac{kl_0}{m} + g_z \end{bmatrix}$$



Equation: $\frac{d^2z}{dt^2} = \frac{k\,(l_0 - z)}{m} - \frac{d}{m}\frac{dz}{dt} + g_z.$

Parameter: $m = 0.1,\ k = 100,\ l_0 = -1,$
$d = 1,\ g = -10\ v_{0,z} = -5$

Analytical Solution:

$$z\,(t) = -\frac{33\sqrt{39}}{1300}\,e^{-5\,t}\sin\left(5\,\sqrt{39}t\right) + \frac{1}{100}\,e^{-5\,t}\cos\left(5\,\sqrt{39}t\right) - \frac{101}{100}$$

Figure 3.7: A simple mechanical system: A particle p with mass m connected to the origin using a spring with stiffness k, damping d and rest length l_0, is pulled down by gravity.

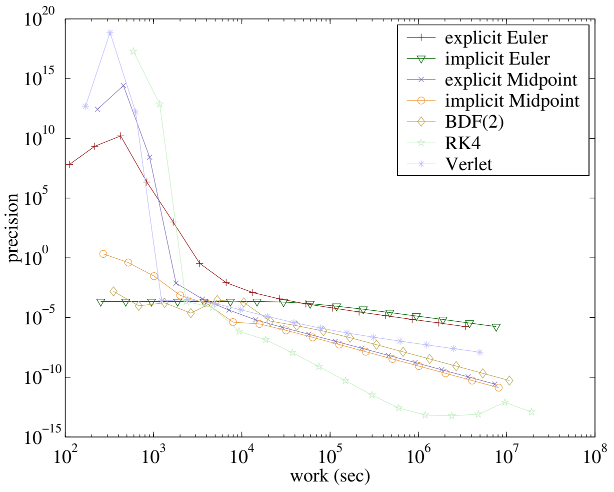


Figure 3.8: By varying the time step, solutions of different accuracy are produced. The work-precision diagram plots the flops spent against the error $\|Y(t_{end}) - y(t_{end})\|$. The stability limit is marked by a sudden loss of accuracy. In the convergent region, the order of the method matches the slope in the double logarithmic plot.

The real parts of the eigenvalues of the Jacobian, viz. the linearisation, of this equation are non-positive and for $d > 0$ negative. This is a characteristic of *every* damped mechanical system and a necessary condition for reaching a rest state. Therefore Dahlquists equation with $\operatorname{Re}\lambda < 0$ is considered to be a good model for a dissipative system. In this case, since the exponent is negative, the analytical solution is bounded for $t \to \infty$. Therefore, one expects from a meaningful numerical method to deliver a bounded solution. An integration scheme that yields a bounded solution is called *stable*.

If we apply the forward Euler method with a fixed step size h to (3.2.9), the n-th point of the numerical solution is given by:

$$Y_n = (1 + h\lambda)^n y_0 \tag{3.2.10}$$

For large $|\lambda|$, it is bounded if and only if $|1 + h\lambda| < 1$, i.e. for $h\lambda$ in the unit sphere around -1. A similar analysis can be carried out for the other methods and results in restrictions of the admissible step size too.

This analysis explains the steep rises in the work precision diagram in figures 3.8. Only when the step size drops below a certain limit dictated by λ, i.e. by $h < 2|\lambda|^{-1}$ in case of the forward Euler method, the numerical solutions can converge. If the damping is increased, i.e.

$\operatorname{Re}\lambda \to -\infty$, then for the explicit Euler necessarily h must tend to 0 for the solution to be stable. This means the step size is artificially limited and cannot be increased beyond the stability limit. This limits the flexibility of balancing work against accuracy.

3.2.3 The Implicit Euler Method

There are at least two ways to combat stiffness. One is to design a better computer, the other, to design a better algorithm.
In Aiken: Stiff computation. Cited from [HW96], H. Lomax, 1985.

To construct a method that better suits our needs we go back to (3.2.3) and substitute the differential quotient by a *backward* difference quotient

$$M\frac{y(t+h)-y(t)}{h} \approx My'(t+h) = f(t+h, y(t+h)).$$

This results in the integration formula

$$MY_{n+1} = MY_n + hf(t+h, Y_{n+1}),$$

the so called backward or implicit Euler method. As its explicit variant, this method can be shown to have order 1. Now the numerical solution only is given implicitly by the solution of the possibly nonlinear equation

$$MY_{n+1} - hf(t+h, Y_{n+1}) - MY_n = 0.$$

This time the method is able to cope with an implicit ODE, without explicitly inverting M, as this can be integrated into the nonlinear solution process. If we apply this method to the Dahlquist equation, we get the recurrence formula

$$Y_n = (1-h\lambda)^{-n} y_0. \tag{3.2.11}$$

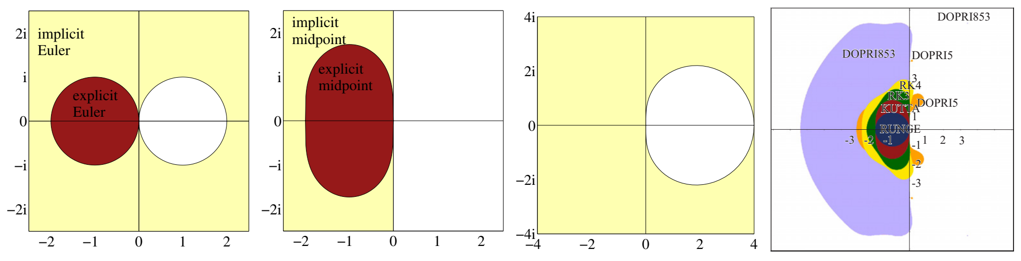
The numerical solution Y_n remains bounded for $|(1-h\lambda)^{-1}| < 1$. If we assume $\lambda < 0$, this holds for arbitrary $h > 0$. Thus, there is no restriction on step size. The method is *unconditionally stable*. Figure 3.8 shows the work-precision curve for the implicit Euler method. We observe that we never loose stability and we especially do not miss the solution by several orders of magnitude compared to the explicit methods. Of course, the method looses accuracy when the time steps become large.

As a useful tool for visualising the stability properties of a method we define the *stability region* $\mathcal{S}$ to be the set of parameters, for which the integration method yields a bounded solution:

$$\mathcal{S} := \{z := h\lambda \in \mathbb{C} : \text{ the numerical integration of equation (3.2.9)} \\ \text{with step size } h \text{ and parameter } \lambda \text{ is stable}\}.$$

If $\mathcal{S}$ contains the complete left half-plane the method is called *unconditionally stable* or A*-stable*. Such methods are well suited for the stable integration of stiff[2] equations. Obviously, the implicit Euler scheme is A–stable, whereas its explicit counterpart is not. The stability regions of some presented methods are shown in figure 3.9.

[2]A stiff equation, is by 'definition' an equation, where explicit methods suddenly loose stability. There is no precise definition of 'stiff' (cf.[HW96]), but usually the equation has some large negative eigenvalues.



(a) Euler methods (b) midpoint rules (c) BDF(2) (d) RK methods

Figure 3.9: Stability regions (shaded) of the methods.

After reviewing the process that led us to the definition of the stability region, we can outline a more general idea that will allow us to determine the stability of more complex methods. The idea for analysing both Euler methods applied to (3.2.9) was to find a closed expression describing the *stability function* $\mathcal{R}$. This function maps the initial value y_0 to the value Y_1, performing a single step of the method

$$\mathcal{R} : y_0 \mapsto Y_1.$$

Thus $Y_n = \mathcal{R}^n(h\lambda)y_0$. For the explicit Euler method we found in (3.2.10)

$$\mathcal{R}(z) = 1 + z,$$

for the implicit version in (3.2.11)

$$\mathcal{R}(z) = \frac{1}{1-z}.$$

The definition for the stability region now reads

$$\mathcal{S} = \{z \in \mathbb{C} : |\mathcal{R}(z)| \leq 1\}.$$

To compute $\mathcal{R}$ for an explicit Runge-Kutta method, we apply the definition (3.2.8) to (3.2.9):

$$\begin{aligned} g_1 &= y_0 && = p_1(h\lambda)y_0 \\ g_2 &= y_0 + a_{21}h\lambda g_1 && = p_2(h\lambda)y_0 \\ &\dots \end{aligned}$$

for the polynomials $p_1(h\lambda) = 1$ and $p_2(h\lambda) = a_{21}h\lambda + 1$. By induction, we have $Y_1 = p_{s+1}(h\lambda)y_0$ with a polynomial of degree $\leq s$. Therefore, the stability function of an explicit method is a polynomial. Hence, the stability region is bounded and the method cannot be unconditionally stable.

3.2.4 Implicit Runge-Kutta Methods

To find a higher order method, we go back to equation (3.2.5) and insert a linear interpolation term for $y(t + h/2)$. The resulting formula is taken as an implicit definition of $y(t + h)$. We get the *implicit midpoint rule*

$$Y_1 = Y_0 + hf\Big(t + h/2, \frac{Y_1 + Y_0}{2}\Big),$$

using a simplified notation for advancing one step, i.e. writing Y_0 and Y_1 instead of Y_n and Y_{n+1}. Again a mass matrix can simply be added by multiplying the defining formula by M.

Alternatively the midpoint rule can be derived as a *collocation method* [HW96] with $s = 1$ internal nodes, i.e. by constructing a polynomial interpolating the particle trajectories at a given, fixed set of s nodes [HW96]. This idea allows the construction of implicit Runge-Kutta methods with arbitrary order. In contrast to explicit methods the matrix (a_{ij}) ceases to be strictly lower triangular. These methods are computationally more expensive, so we just stick to the midpoint rule. Its stability function is given by

$$\mathcal{R} = \frac{1 + z/2}{1 - z/2}.$$

As $\mathcal{R} \leq 1$ for any $\operatorname{Re} z < 0$ the implicit midpoint rule is A-stable.

3.2.5 Multistep Methods

As another possible choice, we now introduce *multistep methods*. They are computationally inexpensive because they have no inner stages and some of them are A-stable. A multistep method with k steps is of the general form

$$\sum_{j=0}^{k} \alpha_j Y_{1-j} = h \sum_{j=0}^{k} \beta_j f_{1-j},$$

with $f_j := f(t_j, Y_j)$. Here we also have 'history points' with negative indices. The lowest coefficient α_0 is required to be nonzero; for variable time step sizes the coefficients depend on the last step sizes, which we have omitted here for the ease of demonstration. Important special cases are the class of *Adams methods* where $\alpha_0 = \cdots = \alpha_{k-2} = 0$:

$$Y_1 = Y_0 + h \sum_{j=0}^{k} \beta_j f_{1-j}$$

and the class of *BDF–methods* (**b**ackward **d**ifferentiation **f**ormulas) with $\beta_0 = \cdots = \beta_{k-1} = 0$:

$$\sum_{j=0}^{k} \alpha_j Y_{1-j} = h \beta_k f_1.$$

If the formula involves the right-hand side f_1 at the new approximation point Y_1 the method is said to be implicit. BDF-methods are always implicit. The stability regions of implicit and explicit Adams methods are bounded and located around the origin, thus they are not interesting for large time steps, and we won't discuss them further.

Implicit multistep methods are able to handle implicit differential equations in the same way as demonstrated for the backward Euler method, so we dropped the index M for a clearer notation. The coefficients for the methods can again be constructed by a collocation approach. For the BDF-methods, this is feasible up to order 6, higher order methods loose consistency for any choice of coefficients [HW96]. BDF-methods were the first to be developed to deal with stiff equations and possess an unbounded stability region covering a sector within the negative complex half-plane. Therefore, they are among the most widely used methods today. For $k+1$ points, these methods possess order $k + 1$ and only one nonlinear system has to be solved, whereas s coupled systems have to be solved for an s-stage implicit Runge-Kutta method.

The BDF-method for $k = 1$ is just the implicit Euler method, for k=2 the method, also known as trapezoidal rule, is given as

$$Y_1 = \frac{4}{3}Y_0 - \frac{1}{3}Y_{-1} + \frac{2}{3}hf(t+h, Y_1)$$

The coefficients for higher order methods are given in table 3.2. The stability region of BDF(2) is shown in figure 3.9. In [HES03] BDF with variable coefficients is discussed.

α_0	α_1	α_2	α_3	α_4	α_5	α_6	β
$\frac{3}{2}$	-2	$\frac{1}{2}$					1
$\frac{11}{6}$	-3	$\frac{3}{2}$	$-\frac{1}{3}$				1
$\frac{25}{12}$	-4	3	$-\frac{4}{3}$	$\frac{1}{4}$			1
$\frac{137}{60}$	-5	5	$-\frac{10}{3}$	$\frac{5}{4}$	$-\frac{1}{5}$		1
$\frac{147}{60}$	-6	$\frac{15}{2}$	$-\frac{20}{3}$	$\frac{15}{4}$	$-\frac{6}{5}$	$\frac{1}{6}$	1

Table 3.2: BDF Methods

3.2.6 Rosenbrock Methods

An idea employed by most other authors [BW98, DDBC99] is to ignore the non-linearitiy of the equations that arise from the application of an implicit solution method. This is equivalent to performing only a single Newton iteration and is sometimes called 'semi-implicit'. Unfortunately this may sacrifice A-stability. If on the other hand, we put this requirement already into the definition of the method, we arrive at Rosenbrock methods

$$(M - h\gamma_{ii}J)k_i = hf\left(Y_n + \sum_{j=1}^{i-1} a_{ij}k'_j\right) + hJ\sum_{j=1}^{i-1}\gamma_{ij}k_j \quad \text{with } J = \frac{\partial f}{\partial y}(Y_n)$$
$$Y_{n+1} = Y_n + h\sum_{i=1}^{s} b_i k_i.$$

Now the coefficient set $\alpha_{ij}, \gamma_{ij}, b_i$ can be specially constructed to maximize order and stability. Again, whole families arise, some of which are A-stable or include at least a sector of angle α around the negative real axis in their stability region ($A(\alpha)$-stable). Rosenbrock methods are recommended by Press et al. [PTVF88] for stiff equations, although the discussed siblings are not even A-stable. Furthermore Rosenbrock methods suffer from the necessity to recompute the Jacobian very often, per definitionem at each integration step. Therefore Steihaug and Wolfbrandt [SW79] attempted to construct methods that only require an approximation to J, thus relaxing the update burden. The resulting methods are called W-methods. Using the framework developed for the other implicit methods, it is trivial to implement them, so we added several methods of order 4 [HW96] (SHAMP, GRK4A, GRK4T, VELDA, VELDB, LSTAB), a second order Rosenbrock-W method due to Verwer et al. [VSBH99], and the third order Rosenbrock-W method AMF-RK32 [GV02] to our library.

3.2.7 The Verlet Method

The leapfrog or Stoermer-Verlet method is especially efficient if (3.2.1) is given as the second order system

$$x''(t) = f_v\big(t, x(t)\big), \tag{3.2.12}$$

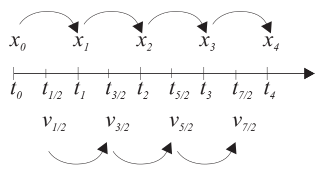


Figure 3.10: Staggered grids for the Verlet method.

i.e. $f_v\big(t, x(t), x'(t)\big) = f_v\big(t, x(t)\big)$. It is not applicable to general first order systems of the form (3.2.4).

To derive it, we use centered differences at a staggered grid (figure 3.10), i.e. we now approximate v at $t + (2i + 1)h/2$ and x at $t + ih$ by centered differences

$$\frac{v_{n+1/2} - v_{n-1/2}}{h} = f(x_n)$$
$$\frac{x_{n+1} - x_n}{h} = v_{n+1/2}$$

thus

$$v_{n+1/2} = v_{n-1/2} + hf(x_n) \tag{3.2.13}$$
$$x_{n+1} = x_n + hv_{n+1/2}. \tag{3.2.14}$$

The method possesses order 2 as one can see by substituting (3.2.13) into (3.2.14) resulting in the second order centered difference

$$\frac{x_{n+1} - 2x_n + x_{n-1}}{h^2} = f(x_n).$$

From this equation an alternative formulation of the Verlet scheme as a multistep method can be derived

$$v_n - v_{n-1} = hf(x_n),$$
$$x_{n+1} - x_n = hv_n,$$

which omits the half steps and staggered grids from above. Now for second order equations which do not possess the form of (3.2.12) one may replace $f(x_n)$ by $f(x_n, v_{n-1})$ at the expense of some stability. Correctly the replacement had to be with $f(x_n, v_n)$ but this would result in a implicit method. The work precision diagram 3.8 shows the Verlet method applied to our toy example.

Up to now, we have omitted a discussion of the stability properties of the leapfrog method. From the example, it can be observed that the method is not unconditionally stable. Indeed, some more delicate computations show that the method only delivers a bounded solution for arbitrary $h > 0$ for purely oscillatoric equations, i.e. for second order ordinary differential equations of the form (3.2.12) (with no damping term) and a contractive right hand side. Nevertheless, the method remains well behaved in the presence of low damping. This explains its importance in molecular and celestial dynamics. In the tutorial [KBF+03] we discuss an application to the undamped wave equation.

3.2.8 ROCK - A Stabilised Explicit Integration Method

The classical explicit methods were constructed by fixing a number of stages and maximising the order under some additional constraints, e.g. a cheap embedded method. Unfortunately,

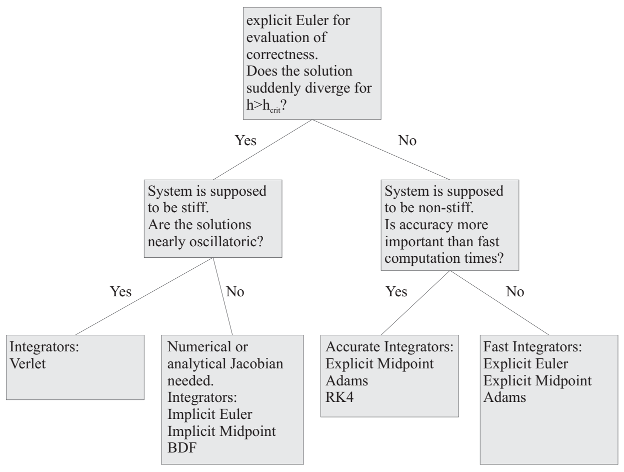

explicit Euler for evaluation of correctness. Does the solution suddenly diverge for h>h_crit?
Yes
No
System is supposed to be stiff. Are the solutions nearly oscillatoric?
System is supposed to be non-stiff. Is accuracy more important than fast computation times?
Yes
No
Yes
No
Integrators: Verlet
Numerical or analytical Jacobian needed. Integrators: Implicit Euler Implicit Midpoint BDF
Accurate Integrators: Explicit Midpoint Adams RK4
Fast Integrators: Explicit Euler Explicit Midpoint Adams

Figure 3.11: Selecting a method.

the stability region, characterised by the polynomial $\mathcal{R}$, is usually small. Fixing the number of stages s, one can also try to find the polynomials which give the largest stability region and coincide with the first terms of the exponential function to guarantee convergence. These are given by shifted and scaled Chebychev polynomials. Optimized to a tube along the real axis for parabolic equations, the lower bound can be pushed nearly to $-s^2$. The next problem is to find a corresponding Runge-Kutta method that can be implemented in a stable manner. Several techniques have been proposed, but they either require numerically computed roots or have low order. Recently Abdulle et al. [AM01] proposed a stable and efficient implementation based on orthogonal and nearly optimal stability polynomials. The Orthogonal-Runge-Kutta-Chebychev methods (ROCK) are a family of methods with a large stability domain along the negative real axis. The lower bound grows with s^2, thus the methods are efficient for stiff problems, because they are able to use time steps usually about 2 to 10 times larger than other explicit methods. ROCK2, which we use in our implementation adapts and balances s against the time step h based on a user-supplied estimate of the spectral bound. For details we refer to the thesis of Abdulle [Abd01].

3.2.9 Selecting an Efficient Method

Which method is best for a certain application? This question is nearly impossible to answer a priori. The only choice is to try a set of methods and to evaluate which one performs best. Choosing the methods to try though can be done based on theoretical considerations and observations of the problem at hand. A possible strategy is shown in figure 3.11.

The same statement holds for predicting the efficiency of a method. Generally, implicit methods require more work per step. On the other hand, one may be able to use time steps that are several magnitudes larger than the ones explicit methods would allow. Although accuracy will suffer, the integration won't be unstable. If evaluations of the right hand side function are cheap, a step with RK4 is faster than an implicit step with BDF(4). Then again, if it is cheap to compute a good sparse approximation to the Jacobian, it may be more efficient to solve the linear system with a few cg iterations than to perform four full function evaluations.

```
for $k = 1, 2, \dots$ until convergence do
  Compute $G(Y^{(k)})$.
  Compute $J^{(k)} = \frac{\partial}{\partial Y} G(Y^{(k)})$.
  Solve $J^{(k)} s^{(k)} = -G(Y^{(k)})$.
  $Y^{(k+1)} := Y^{(k)} + s^{(k)}$
end
```

Algorithm 3.4: Newton's Method

3.3 Solving Nonlinear Systems

Most implicit methods require the solution of a nonlinear system. The implicit Euler method for example reduces the integration problem to the solution of the nonlinear system

$$MY_1 - hf(Y_1) - MY_0 = 0.$$

The other methods yield a system of similar structure, for example

$$MY_1 - hf(\frac{1}{2}(Y_1 + Y_0)) - MY_0 = 0$$
$$MY_1 - \frac{2}{3}f(Y_1) + M\left(-\frac{4}{3}Y_0 + \frac{1}{3}Y_{-1}\right) = 0$$

for the midpoint and BDF(2) rule, respectively. This is a nonlinear system of dimension $6N$, from now on abbreviated as $G(Y) = 0$, where N is the number of nodes in the mesh. Due to the special structure of the system we have

$$\frac{\partial G}{\partial Y} = \begin{bmatrix} \text{id} & 0 \\ 0 & M \end{bmatrix} - \alpha h \begin{bmatrix} 0 & \text{id} \\ \frac{\partial f}{\partial \mu} & \frac{\partial f}{\partial \nu} \end{bmatrix},$$

which can later be exploited to speed up the linear solver. The system must be solved with Newton's method to allow arbitrary step sizes independent of λ. Simpler methods for nonlinear systems, like fixed-point iteration, would compensate the advantage of A-stability because the number of iterations would increase proportionally to the stiffness parameter $|\lambda|$. We now will work out an approach for implementing Newton's method efficiently.

3.3.1 Newton's Method

For the nonlinear system $G(Y) = 0$ we compute a numerical solution by algorithm 3.4. This reduces the problem to the successive solution of linear systems. In a classical Newton method these systems are solved by Gaussian elimination, which, for sparse systems, introduces a lot of additional non-zero elements into the factors. Therefore special sparse solvers need to be used, which will be discussed below.

The majority of authors [BW98, VMT00a, HE01, CK02] take another approach and use iterative methods to solve the linear system. As of the exchangeable solvers, we provide a preconditioned conjugate gradient method to solve the linear systems in each Newton step. Unfortunately this changes the convergence behaviour of the outer Newton method, which is referred to as an inexact Newton method [Rhe98], given by algorithm 3.5.

3.3.2 Convergence and Residual Control

The error of the iterative solution of the linear system in line 4 of algorithm 3.5 is formulated in terms of the residual, which is easily computationally accessible, whereas the actual error

```
for $k = 1, 2, \dots$ until convergence do
    Compute $G(Y^{(k)})$.
    Compute $J^{(k)} = \frac{\partial}{\partial y}(Y^{(k)})$.
    Find $s^{(k)}$ with $J^{(k)}s^{(k)} = -G(Y^{(k)})) + r^{(k)}$, such that $\|r^{(k)}\| \le \eta_k \|G(Y^{(k)})\|$.
    $Y^{(k+1)} := Y^{(k)} + s^{(k)}$
end
```

Algorithm 3.5: Inexact Newton Method

cannot be computed. The tolerance of the linear iteration is decreased proportionally to the monotonically decreasing residual of the nonlinear iteration.

An analysis of this method [Rhe98] shows that it converges under rather weak additional assumptions. If the classical Newton method converges and the scalar tolerances $\eta^{(k)}$ are uniformly bounded by an $\eta < 1$, the inexact method converges. In literature the $\eta^{(k)}$ are referred to as *forcing terms*. Note that this additional assumption is also necessary: For $\eta = 1$, $s^{(k)} = 0$ would be admissible and the iteration would stagnate. The inexact method then at least converges linearly, whereas Newton converges superlinearly. By choosing the η_k to converge to zero sufficiently fast, the convergence of the inexact Newton method can be forced to have an order > 1.

On the other hand, smaller $\eta^{(k)}$ mean tighter tolerances for the iterative linear solver, which imply more work to spend there. Hence, advanced codes carefully balance $\eta^{(k)}$ against the convergence of the method, which of course depends on G. Several choices of forcing terms have been evaluated in our thesis [Hau99a].

3.3.3 Inexact Simplified Newton Methods

The efficiency of the Newton method can be further improved by another approximation. In the simplified version of Newton's method the Jacobian $J^{(k)}$ is approximated by $J^{(0)}$. Such a scheme can be rewritten in the form of an inexact Newton method, if the linear system is written as follows and J is chosen as approximation to $J^{(k)}$

$$\begin{aligned} Js^{(k)} &= -G(Y^{(k)}) + (J - J^{(k)})s^{(k)} + r^{(k)} \\ &=: -G(Y^{(k)}) + \tilde{r}^{(k)} \end{aligned}$$

The residual $r^{(k)}$ is replaced by the larger $\tilde{r}^{(k)}$, which can be bounded if $J \approx J^{(k)}$. By choosing $\tilde{\eta}^{(k)}$ appropriately, the method still converges. In fact, we trade some accuracy approximating $J^{(k)}$ for accuracy in solving the linear system and up to a certain limit the method still behaves as before. If only a single Jacobian $J^{(0)}$ is used, a constant term is introduced into $\tilde{\eta}^{(k)}$, therefore the order of convergence drops to linear, no matter how fast the forcing terms $\eta^{(k)}$ converge to 0. In the simplified case it suffices to specify $\eta^{(k)} = \eta \equiv const$, which is a user parameter. Usually $\eta = 0.01$ gives good results. More important for a quick convergence are good initial values, which we obtain by extrapolating the solution of the previous time step.

3.3.4 Monitoring Convergence

Since the convergence is linear, we have for the step length $\|s^{(k)}\| = \|Y^{(k+1)} - Y^{(k)}\|$

$$\|s^{(k+1)}\| \le \Theta \|s^{(k)}\|$$

and therefore for the exact solution Y^*

$$\|Y^{(k+1)} - Y^*\| \le \frac{\Theta}{1-\Theta}\|s^{(k)}\|.$$

```
Compute $J \approx \frac{\partial}{\partial Y} G(Y^{(0)})$.
for  $k = 1, 2, \ldots$ until convergence do
  Compute $G(Y^{(k)})$.
  Find  $s^{(k)}$ with $Js^{(k)} = -G(Y^{(k)})) + r^{(k)}$, such that $\|r^{(k)}\| \leq \tilde{\eta}_k \|G(Y^{(k)})\|$.
  Update $Y^{(k+1)} := Y^{(k)} + s^{(k)}$
end
```

Algorithm 3.6: Inexact Simplified Newton's Method

Hence $\Theta/(1-\Theta)\|s^{(k)}\|$ provides a meaningful stopping criterion. The convergence rate Θ can easily be estimated by

$$\Theta_k \leq \frac{\|s^{(k)}\|}{\|s^{(k-1)}\|}.$$

In addition a small Θ_k or a low number of Newton iterations are a hint for a good convergence, thus we adopt the trick of Hairer [HW96] and save Jacobian updates, whenever less than 4 iterations were necessary or $\Theta_k < 10^{-3}$.

3.3.5 Global Convergence

As stated in the introduction an important issue in interactive simulations is to keep the simulator running. This means, that we have to take precautions against convergence failures of the Newton method. The usual procedure is to reject the step and restart it with a smaller time step and an updated Jacobian. This is too costly, as the current time quantum may already be nearly spent. In this case we reluctantly give up accuracy, but at least stability has to be insured. Convergence failures, although very rare, because the trajectory is usually smooth and the temporal coherence provides good starting values, may cause instability. Monitoring convergence and updating the Jacobian is an effective precaution. Large discontinuities however, caused for example by massive collisions, may occur without warning and the Newton iteration may overshoot. In this case, if the average decrease is not sufficiently fast, viz. $\|G(Y^{(k+1)})\| > \|G(^{(k)})\| + 10^{-4}\langle \nabla G, s^{(k)}\rangle$, a line-search technique is employed to find an admissible step length [Rhe98]. First $G(Y^{(k)})$, $G(Y^{(k+1)})$ and $\nabla G s^{(k)}$ is used to build a quadratic model for $g(\omega) = G(Y^{(k)} + \omega s^{(k)})$. The tentative step length ω_{ten} is then chosen to minimize the model. If this again fails, we build a safeguarded cubic interpolation [DS96], with the additional $g(\omega_{\text{ten}})$, and iterate till convergence or a maximum iteration count. For all tested cases, this guaranteed convergence.

3.4 Solving Linear Systems

Most codes for solving stiff systems spend most of their time solving systems of linear equations.
Cited from [HW96], Watkins and Hanson-Smith, 1983.

Employing Newton's method reduces the implicit integration problem to the solution of a linear system of equations of possibly large dimension. Rosenbrock and W-methods directly lead to a linear system. Besides the function and Jacobian evaluation, this part of the algorithm is the most time consuming. Here efficiency is preserved or can be lost. A characteristic property of this system is sparsity, which means only 3-by-3 blocks corresponding to nodes with a common edge have non-zero entries. To illustrate this, the sparsity pattern of three Jacobian resulting from the sample problems in figure 3.12 are shown in figure 3.13.

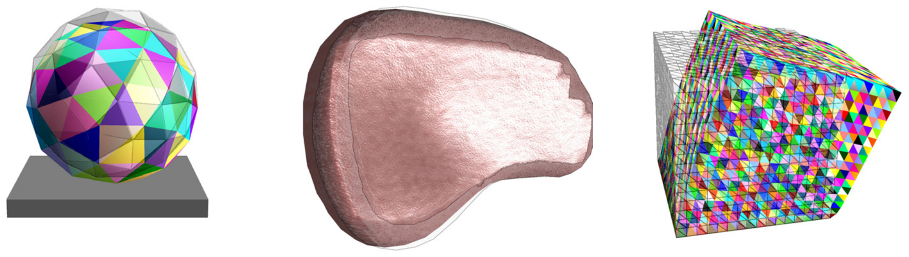


Figure 3.12: The sample scenes, directly grabbed from the application, showing an overlay of rest and deformed state. The smallest scene consists of a sphere composed of 175 tetrahedra and 70 nodes that bounces on a table. The medium example was taken from a liver model with 2616 tetrahedra and 739 nodes. Shearing a cube with 4913 nodes and 24576 tetrahedra gives the matrix of the largest example.

Currently three different techniques dominate in the solution of such systems: multi-grid methods, non-stationary iterative schemes [Saa96] or direct solvers [DER90]. For very large systems with millions of nodes, combinations of the underlying principles are not unusual, for example conjugate gradients with a multi-level BPX-preconditioner. For the problems, that are interactively solvable on a single computer, we restrict ourselves to two of the solution methods only: iterative schemes and sparse direct solvers. The application of multi-level methods, for example in the form of the Newton-Cascade algorithm, could be an interesting point for future work.

3.4.1 Reduction

The matrix A arising from Newton's method has a very special structure due to the second order nature of the differential equation,

$$A = \begin{bmatrix} \text{id} & 0 \\ 0 & M \end{bmatrix} - h\alpha \begin{bmatrix} 0 & \text{id} \\ \frac{\partial f}{\partial \mu} & \frac{\partial f}{\partial \nu} \end{bmatrix}, \tag{3.4.1}$$

where α depends on the integration method used. By using block sustitution, the linear system can be reduced to two systems of halved dimension

$$\begin{aligned} [M - (h\alpha)^2 \frac{\partial f}{\partial \mu} - h\alpha \frac{\partial f}{\partial \nu}]\nu &= b_\nu + \frac{\partial f}{\partial \mu} b_\mu \\ \mu - h\alpha\nu &= b_\mu . \end{aligned} \tag{3.4.2}$$

The second one requires only a scaled addition, after the first one is solved. Because the computational expense grows superlinearly with the dimension, this greatly improves performance. Moreover, it can be implemented without changing the surrounding logic.

3.4.2 Sequential and Parallel Direct Methods for Sparse Systems

The factorisation of a $m \times n$ matrix requires $O(mn^2)$ floating point operations, too quickly growing with the dimension to be feasible for larger problems. For sparse matrices, where only cn entries – with c typically less than a hundred – are different from zero, most work is wasted on the empty parts. Unfortunately, the direct application of Gaussian elimination tends to fill the empty regions, as the examples in figure 3.13 show. The problem can be tackled using graph-theory and the fill-in can be worked out. As it turns out, that it depends on the numbering of the unkowns, which induces the elimination order, it is possible to permute the unknowns

$$\begin{aligned} AP\hat{x} &= b, \\ x &= P^{-1}\hat{x} \end{aligned}$$

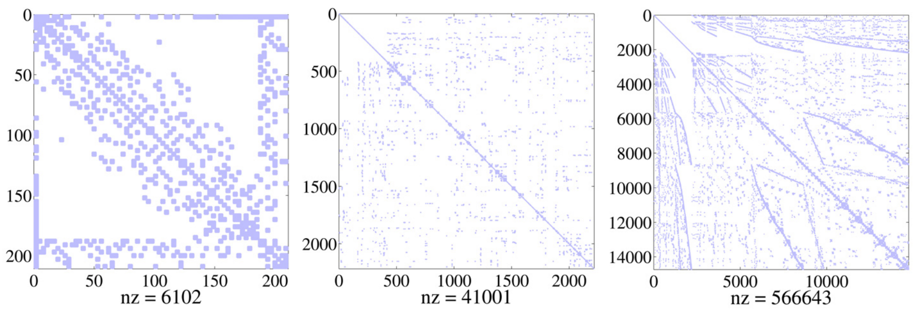
to reduce the fill-in. The permutation P is commonly known as ordering for sparsity. For symmetric matrices, the system is in addition multiplied by P^T from the left in order to keep symmetry. Unfortunately, the problem of computing the optimal permutation can be shown to be NP-complete [Yan81].

Typically, a direct solver proceeds in four steps: First an analysis of the graph structure to compute a fill-reducing permutation. Second a symbolic factorisation step that determines the non-zero entries and sets up data structures and memory. The third step is the numerical factorisation with pivoting. To solve for a given right-hand side, in the last step, forward and backward substitutions on the triangular factors are performed. If only the right hand side changes, like in the simplified Newton method, the factors can be re-used. Also, if the topology of the mesh remains fixed, the ordering for sparsity and the symbolic factorisation need to be executed only once. The efficiency of the algorithm depends on the sophisticated selection of the permutation and efficient data structures and kernels for the numerical solve. For example, linked lists – used in the first codes – would be convenient for adding elements, but will perform bad in step three and four. For the evaluation of permutation strategies as discussed below, we use MATLAB, version 5.3 and 6.1 [GMS92], in the actual C++-implementation, we employ SuperLU[DEG+99] and UMFPACK[DD99].

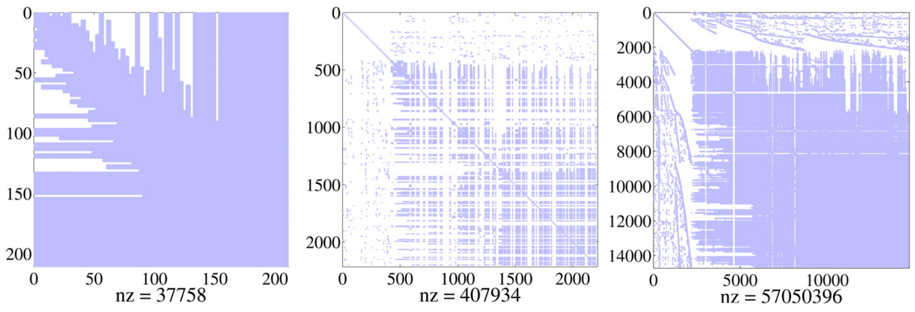
During the second step of the algorithm, a symbolic elimination tree is computed, that represents the order the variables (columns) are eliminated. If, after including the fill-in, column j has no entry in row i, they can be treated independently, as they do not update each other. Therefore, independent subtrees can be computed in parallel, which is the first source of coarse level parallelism. As soon, as there are only dependent subtrees left, pipelining allows to further exploit parallelism, that is a column can be processed, until a dependant variable k is found. If the complex scheduling is done in a sophisticated manner, a fine level parallelism is exposed, which of course then requires a fast communication facility. This approach is well suited for a shared memory architecture. We use SuperLU_MT, developed by Demmel et al. [DGL99]. For very large systems, also a cluster-based implementation (SuperLU_DIST) is available, which is not used by our application.

We performed a few experiments on the sample matrices, to determine the optimal permutation strategy. The results are summarized in table 3.3. We compare the best available native permutations in MATLAB6.1[3], the MATLABalgorithms combined with AMD [ADD96], and the mex-variants of UMFPACK4.1 and SuperLU, which is comparable to lu but significantly faster. Although using the different AMD variants with the built in lu gives the lowest fill-in, UMFPACKwith the built in variant of AMD is fastest.

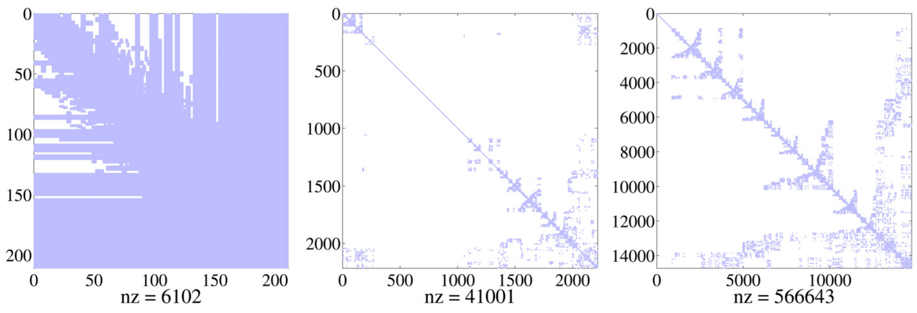
[3]Starting with MATLAB6.5 (2003), UMFPACK4.0 replaces lu, and AMD is made available by default.



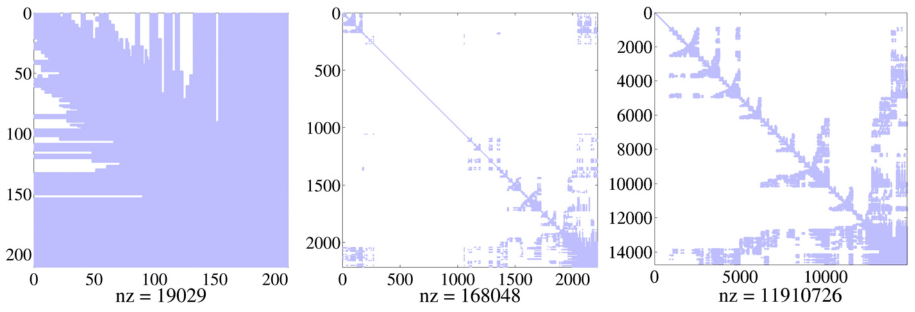
(a) Matrices Nonzero pattern



(b) Classical factorisation (L+U) with large fill-in



(c) Permuted matrices



(d) smart factorisation (L+U) with reduced fill-in

Figure 3.13: Test matrices

Problem (dimension)	Sphere (210)		Liver (2217)		Cube (14 739)	
	flops	nnz(L+U)	flops	nnz(L+U)	flops	nnz(L+U)
direct lu	4.15	37 758	643	407 934	199 994	57 015 396
lu/SYMMMD	.63	14 892	53	211 320	10 978	12 599 580
chol/SYMAMD	.27	13 962	7	170 304	5 096	11 910 726
lu/ best P:	.54	13 962	30	168 048	10 192	11 910 726
UMFPACK:	.56	17 804	15	182 718	14 366	13 759 224
source matrix		6 102		41 001		566 643

Table 3.3: Comparison of direct solvers. Operation count are $\times 10^6$ flops. For the smallest example, SYMAMD performed best, the medium showed a small favor for AMD , in the largest case SYMAMD was most effective.

3.4.3 Iterative Methods

Ich empfehle Ihnen diesen Modus zur Nachahmung.
Schwerlich werden Sie je wieder direct eliminieren,
wenigstens nicht, wenn Sie mehr als 2 Unbekannte haben.
Das indirecte Verfahren läßt sich halb im Schlafe ausführen,
oder man kann während desselben an andere Dinge denken.

Taken from a letter to Gerling, C.F. Gauß, citation from [Hac93], 1823.

Employing the inexact Newton formulation allows the use of iterative methods for an inner iteration. Using only a single grid resolution, non-stationary Krylov-methods are usually preferred over stationary methods, like Jacobi or SSOR. These classical methods then often are valuable as preconditioners. In the symmetric, positive definite case[4], arising from Ritz-Galerkin finite element methods, the conjugate gradient method, with a stable and fast three-term recursion requiring only matrix-vector products is the method of choice. It needs a constant amount of vector storage, no transposed multiplication routine, only a single matrix-vector product per iteration and converges in the A-norm, which corresponds naturally to the discretised energy norm of the underlying problem. Because the matrix A is sparse, a single iteration takes $O(n)$ work.

The number of iterations to reach a given accuracy is proportional to the square root of the condition number of A. Therefore, preconditioning techniques are used. Preconditioning means that instead of $As = g$, the modified system $(M^{-1}AM)M^{-1}s = M^{-1}g$, with $A := I - h\alpha J$, is solved, where $M^{-1}AM$ hopefully gives a smoother convergence. A suitable preconditioner M consists of an easily invertible matrix that approximates A. We will briefly compare some inexpensively accessible preconditoners.

The *diagonal* and *block diagonal preconditioner* is defined by using the diagonal or the diagonal three-by-three block-part of A as M. This is the only choice considered in other systems [BW98, VMT00a], also called diagonal scaling. The diagonal preconditioner is easily available and inexpensive to apply, as each application is just a division of each vector-component by the corresponding diagonal entry, resulting in N flops per application. In case of the block diagonal preconditioner, each block is factored, and the application is performed by backsubstitution, requiring $18N$ flops.

The *incomplete Cholesky factorisation* [Saa96] (chol) is computed by carrying out an approximate Cholesky factorisation of A, i.e. formally factorising A and dropping all intermediate

[4]The approximated Jacobian J is symmetric and negative definite due to Newton's third law and the passivity of the system.

Problem	Sphere (210)				Liver (2217)				Cube (14 739)			
	#iter	M	cg	total	#iter	M	cg	total	#iter	M	cg	total
diag	75	.000	2.14	2.14	42	.00	14.80	14.80	90	.0	243.4	243.4
blockDiag	72	.002	2.16	2.16	39	.01	14.20	14.21	81	.1	227.2	227.4
chol(0)	27	.071	1.11	1.18	14	.91	6.22	7.13	27	8.5	104.9	113.5
chol(1e-2)	22	.684	.91	1.59	7	6.67	3.25	9.92	14	499.2	58.6	557.9
ssor	37	.006	1.52	1.52	17	.04	8.12	8.16	33	.5	129.1	129.7
cgeis	38	.000	.71	.71	23	.00	3.74	3.74	51	.0	81.4	81.4

Table 3.4: This table compares the iteration count, the setup cost, and the cost of the cg-method for the different preconditoners. All numbers besides the iteration count are $\times 10^6$ floating point operations.

values that are either not fitting the sparsity pattern of A (chol(0)), or smaller than a given tolerance, e.g. 0.01. It is more expensive to compute and to apply and costs at least $2m$ flops per application, where m denotes the number of nonzero entries of A.

The *successive-symmetric over-relaxation method* [Saa96] (SSOR) is another iterative solving scheme for linear equations. It is common to use a classical iteration scheme as SSOR for preconditioning the *cg* method. The matrix formulation of one SSOR-step is given by the multiplication with

$$M := (D - L)^{-1} D (D - L^T), \tag{3.4.3}$$

where $A = D - L - L^T$ and $D = \operatorname{diag}(A)$, L strictly lower tridiagonal. Note that the inversion is carried out by inverting two triangular systems, as in practice one product with D is precomputed. Thus it is approximately as expensive to apply but cheaper to initialize as chol.

To compare the efficiency of the preconditioners we again solved the systems with MATLABup to a relative residual tolerance of 10^{-6}. The measurements in table 3.4 show the characteristics of the different preconditioners in terms of resulting iteration count, setup, and iteration cost. The SSOR-Preconditoner gives a good balance between both, and always scores second best after the incomplete factorisation. In the next chapter, we will employ a subset of these preconditioners directly in a cloth draping simulation, with similar results.

Eisenstat [Eis81, EOV90] proposed an implementation of the cg method, that already incorporates preconditioning. The algorithm combines A multiplications and preconditioning in an effective way. Using SSOR-preconditioning, the problem is restated[5] as

$$\{(D - L)^{-1} A (D - L^T)^{-1}\}(D - L^T)x = (D - L)^{-1} b,$$

which, when diagonally preconditioned, is equivalent to the original system. A clever reuse of partial result vectors reduces the iteration count from $6 \dim(A) + 2 \operatorname{nnz}(A)$ to $8 \dim(A) + \operatorname{nnz}(A)$. The last row of table 3.4 shows our MATLABimplementation of this algorithm and proves it to be up to twice as fast.

3.4.4 Numerical Evaluation

In a last experiment, we compared the C/C++-Implementation of the direct and iterative methods against each other, using the benchmark. The relative residual tolerance for the cg method was set to $10^{-4} - 10^{-7}$ in 4 steps, decreasing by an order of magnitude, reflecting the embedding in an outer Newton iteration. Table 3.5 summarizes the results. Clearly, the direct solvers benefit from their very fast backsubstitution. Also it can be seen, that the higher iteration count of the blockdiagonal preconditioner is partially compensated by its cheaper application.

[5] In European publications this is also known as the Niethammer trick.

Problem	Sphere (210)			Liver (2 217)			Box (14 739)		
	setup [ms]	solve [ms]	#iter	setup [ms]	solve [ms]	#iter	setup [s]	solve [s]	#iter
UMFPACK	6.76	.58	-	114	11	-	17.807	.370	-
SuperLU	6.12	.37	-	199	8	-	54.125	.307	-
SuperLU_MT	4.37*	.24*	-	110*	7*	-	24.805*	.280*	-
cg/blockDiag	.01	2.19	54/67/80/84	.1	16	28/34/39/45	.002	.339	56/68/80/92
cgeis	.01	.98	20/29/32/37	.2	11	13/16/19/21	.002	.225	29/33/38/43

Table 3.5: Comparison of direct and iterative solvers. The setup time reflects the factorisation or preconditioner computation step, the solve column gives the time for a single solve. For the iterative solvers a mean value for a residual tolerance of $10^{-4} - 10^{-7}$ is given. The parallel benchmark was made on a dual Xeon 2.2GHz, the sequential on a 2.0GHz Pentium 4. For a fair comparison, the timings should be increased by about 10%.*

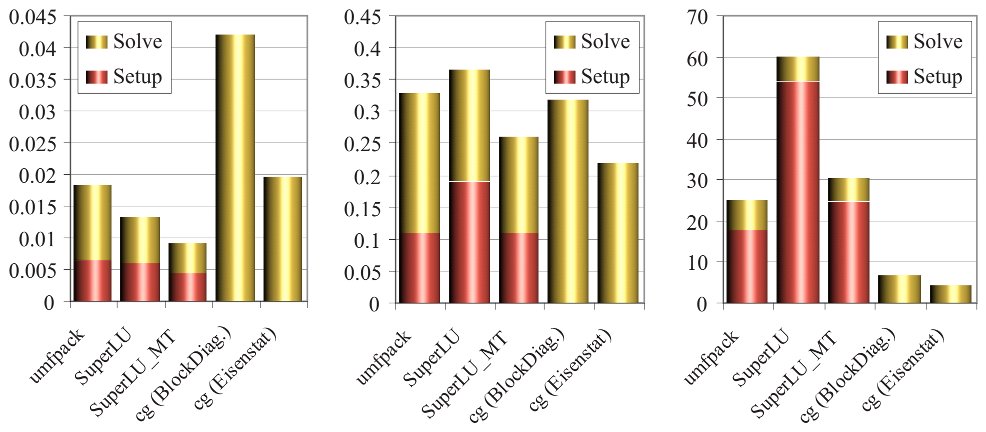


Figure 3.14: Comparison of solver efficiency for a single factorisation or preconditioner setup and 20 solves. The sample-matrices are described above, dimensions are 210, 2217, and 14793, from left to right.

If the solver is embedded in a Newton method inside a time integrator, a convergence monitor of the nonlinear solver often allows the code to keep the Jacobian constant over several time steps. This situation is compared in figure 3.14, which is based on the assumption, that the Jacobian is held constant for 5 time steps, each requiring 4 Newton iterations. Despite their higher setup costs, the direct solvers are competitive or even faster than the plain cg method, except for the largest example, which corresponds to 4913 nodes and 24576 tetrahedra.

3.5 Summary

In the first section of this chapter, we proposed to use finite elements to discretise the variational equations of elasticity, including mass and dynamics. A co-rotated strain measure provides a technique to keep the ordinary differential equation linear. Neglecting the possibility of locking, we employ linear tetrahedral shape functions yielding a reasonably fast implementation that suits our needs and can be extended to a hierarchical approximation.

The arising ordinary differential equation can be solved by explicit or implicit time integration. Implicit methods are preferable, as they require no mass lumping and allow large time steps due to their unconditional stability. As the equation is stiff and the accuracy requirements are comparably low, stable implicit integration methods will proof superior for the application. Their drawback is the necessity to provide a Jacobian, needed for the Newton-type nonlinear solver. The simplified Newton iteration is the method of choice inside a time integrator, for our needs enhanced for global convergence and eventually in an inexact form.

Even though the dimension can be reduced, the linear system solver finally is the most time consuming component. A flexible architecture permits the use of sparse direct solvers and iterative preconditioned Krylov methods, the latter starting to be faster only for larger systems with thousands of nodes. In both cases, techniques for enhancing efficiency as ordering for sparsity or Eisenstat's implementation have been discussed.

Chapter 4

Application 1: A Numerical Core for Clothing Simulation

The first application we developed – based on a part of the techniques presented in the last chapters – is a numerical core for the simulation of garments, implemented together with O. Etzmuß. It continued the work presented in the thesis 'Numerische Verfahren zur Simulation von Textilien' [Hau99b], now replacing adapted standard codes by custom-tailored building blocks.

The physical model is given by a simple particle system based on finite differences, covered in the first section. This space discretisation scheme was further analysed and refined by Etzmuß[EGS03]. The next section describes the assembly of the time integrator, consisting of an implicit integration method, a non-linear, and a linear solver. A module for handling interactions with the environment completes the system. To demonstrate the soundness of the approach, we present some experiments. The chapter is closed by a summary and an outlook on the continuation of the project by the Virtual-TryOn group.

4.1 Spatial Discretisation

When using the particle system paradigm, the discretisation is already a part of the physical modelling process, as the continuous object is immediately represented as a set of discrete points with finite masses. Physical properties are specified by directly defining forces between these mass points. Typical representatives of this approach, that is very popular in cloth simulations, are mass-spring-damper systems [Pro95, VSC01] and particle systems with forces defined directly by measured curves [EWS96] or low order polynomial fits of this data [BHW94].

As already demonstrated, a more ambitious physical modelling process yields a partial differential equation. A geometrically inspired discretisation technique, simpler than finite elements, is the finite difference method. Finite differences are less flexible, as they are restricted on the shape of the domain and the elements that can be used; also, convergence is usually inferior to finite elements. On the other hand, they result in less complex equations, which was the reason we chose them in the first place. The method proceeds by replacing the spatial derivatives with finite differences, i.e.

$$\frac{\partial f}{\partial x} \to \frac{f(x+\delta x)-f(x-\delta x)}{2\delta x} \qquad \text{and} \qquad \frac{\partial^2 f}{\partial x^2} \to \frac{f(x+\delta x)-2f(x)+f(x-\delta x)}{\delta x^2}. \tag{4.1.1}$$

This replacement is easily accomplished in one dimension or on structured grids in any dimension. It becomes harder to define finite difference approximations on unstructured meshes

[MDSB02], often finite element techniques are used for deriving appropriate schemes [Pol02]. An important trait of finite difference schemes is, that the terms on the right hand side of the ODE are given for a point and its neighbours, i.e. are specified on the *edges* of the discretisation. The resulting equations are structured very similar to these from particle systems.

Indeed finite difference techniques can be used to derive a particle system from a 2D continuum formulation employing a St.Venant-Kirchhoff material(Etzmuss et al. [EGS03]). This plan of attack combines a sound derivation with an easy implementation. In the model, used in several of our publications, the tension forces coincide with damped linear springs. Shear and bend forces from the finite difference discretisation, magnitudes smaller than the stiff in-plane springs, do not map to spring-type laws. We denote them by placeholders and refer to Etzmuss [Etz02] for details. The final second order equation that has to be solved for the particle system with positions x and velocities v reads

$$\rho \frac{d^2 x_i}{dt^2} = \sum_{j|(i,j)\in E} \left[\frac{k_{ij}}{l_{ij}^2} (\|x_i - x_j\| - l_{ij}) \frac{x_i - x_j}{\|x_i - x_j\|} + \frac{d_{ij}}{l_{ij}^2} (v_i - v_j) \right] + F_{\text{shear/bend}} + F_{\text{environment}},$$

where E is the set of all edges in the particle system, k_{ij}, and d_{ij} are the stiffness and damping constants determining the spring properties, and l_{ij} the rest lengths between the N particles. The system is nonlinear because of the appearance of the norm $\|x_i - x_j\|$.

4.2 Time Integration

The blueprint given by the method-of-lines now leaves us with an ordinary differential equation to solve. A first analysis of the structural problems identified the stiff in-plane forces as the limiting factor for using large time steps, which are necessary for a fast solution. Therefore, after the evaluation of standard codes, we used IMEX methods [EEH00, Etz02], that allowed to split off the stiff part of the system – essentially the stiff tension springs – to be treated implicitly, whereas the remaining part is treated explicitly. The combination of explicit and implicit methods is not allowed to be arbitrary; we used the implicit and explicit midpoint and the Euler methods.

Hairer and Wanner [HW96] call such methods ‘partitioning methods’. Their criticism is, that it is not always straightforward to identify stiff parts of the equation. Indeed, the first try with IMEX methods used a complete linearisation, as also done by Baraff and Witkin [BW98], that is the stiff part was assumed to be the linearised spring forces. This gives rise to a very efficient implementation, because only a single linear system has to be solved per time step. Although allowing larger time steps than explicit methods, the arising scheme is not unconditionally stable. The achievable time step sizes, which depend on the material parameters[1], are around 1ms. Therefore, an update and re-evaluation of the ‘non-stiff’ part during a single time step [EEH00] was used, which allows larger steps at the expense of solving a linear system per additional update.

A much more elegant solution is not to adapt the underlying numerical integration method to the partitioning but the linear algebra only [HE01, HES03]. This inevitably leads to a numerical integration method combined with an inexact Newton method, using an approximate Jacobian only. In some special cases, this is just the ‘updated IMEX’ idea, but now with a sound basis. In addition, the simple fix-point iteration of the update IMEX [Etz02] is replaced by the superior Newton method. This allows us to include techniques for convergence monitoring, adaptive termination of the Newton iteration and the use of forcing terms for balancing inner and outer iterations, as described in the previous chapter. In this case it is convenient to construct the solver

[1]As for example visible in the CFL condition [CFL28, DDCB01] (see also Sec. 3.2.2).

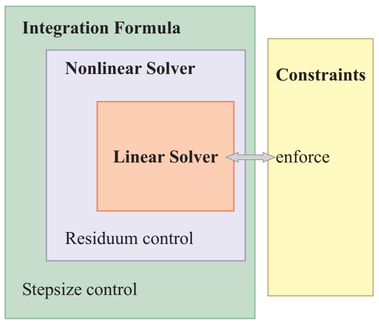


Figure 4.1: Layered structure of the numerical solver core for clothing simulation.

as a layered architecture as depicted in figure 4.1. To be more concrete, the following techniques were implemented in the clothing simulator core.

Integration methods The previously used Midpoint and Euler methods also have been implemented in the new framework. In addition a multistep BDF(2) method including a variable stepsize formulation has been added. For BDF(2), the complete integration algorithm has been published in 'The Visual Computer' [HES03].

Nonlinear solver As a nonlinear solver, the inexact Newton method has been applied. The preconditioned conjugate gradient method has been used as an inner, approximate solution method. As an additional degree of freedom, the Jacobian of the system has only been approximated. This significantly speeds up the computation. As an approximation the linearized tension part has been used

$$F_l = \sum_{j|(i,j)\in E} \left(\frac{k_{ij}}{l_{ij}^2}(x_i - x_j) + \frac{d_{ij}}{l_{ij}^2}(v_i - v_j) \right),$$

which gives the following form for the approximated Jacobian

$$J := \begin{bmatrix} 0 & \mathrm{id} \\ K & D \end{bmatrix},$$

where K and D possess a Laplacian-like structure, with local coefficients of diffusion. This choice of the Jacobian has two major advantages over the full Jacobian. First, J is inexpensive to compute and only changes when either the material constants or the step size changes. Second, we reduce the entries in the Jacobian to approximately a third of the entries in the sparsity pattern of the full Jacobian. Hence, an iteration of the linear solver only requires a third of the original time. The same matrix has already been used in a different technique for clothing simulation by Debunne [DDBC99]. In fact, this operator is the Jacobian associated to the Cauchy tensor of linear small strain elasticity, such that the small strain problem with respect to the initial state is solved in each Newton iteration.

Linear solver As already mentioned, the preconditioned cg method has been employed as a linear system solver. Of course, the system was reduced by exploiting the second order structure, as described in section 3.4.1. Because of their eminent importance, several preconditioners have been implemented, namely a diagonal preconditioner, a chol(0)-preconditioner and a ssor-precondioner, the latter two being a novelty for visual simulations. Etzmuss [Etz02] only reports on chol(0). In the next section, we will give a small example of their performance.

Collsions Collisions can be elegantly handled inside the cg method, a trick that was first employed by Baraff for the simulation of deformable objects. Later it was justified and analysed by Ascher and Boxerman [AB03]. It operates by restricting the solution increments computed

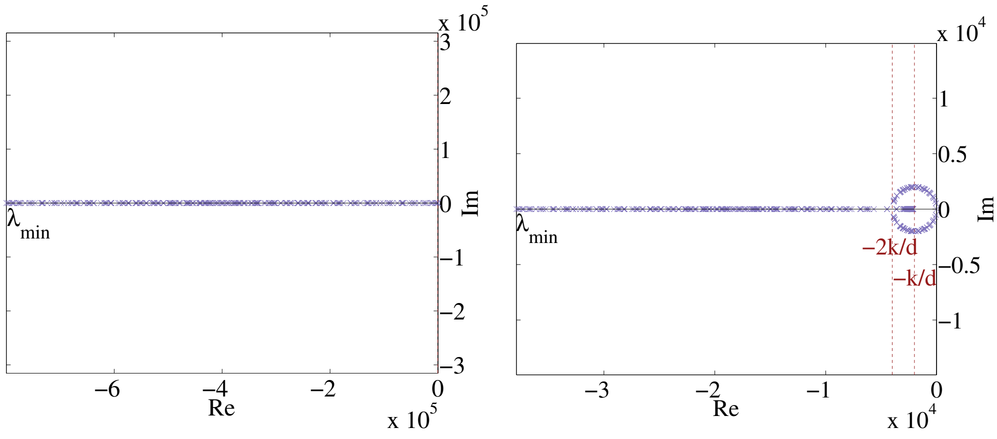


Figure 4.2: Eigenvalues of (4.3.1) for $N = 20$, $k = 5000Nm/m^2$, $l = 1m$, $\rho = 0.2kg/m^2$. The left picture shows $d = 50Ns/m$, the right $d = 2.5Ns/m$.

by the cg-update to a space orthogonal to a set of given constraint directions per particle and therefore prevents any movements in these directions. In section 5.4.2, when we describe a more complete collision handling concept, we will give some more detail.

4.3 Experiments and Results

In addition to some results, this section will present some real world experiments that back up the observations and conclusions made in the theory chapter. We start with an eigenvalue analysis of the approximated Jacobian, which describes the damped 2D wave equation

$$\rho\frac{d^2x_i}{dt^2} = k\nabla^2 x + d\nabla^2\frac{dx_i}{dt}. \tag{4.3.1}$$

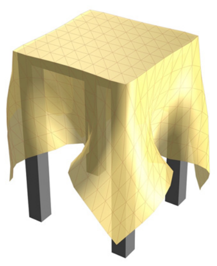
The eigenvalues of the Laplacian ∇^2 in the Poisson equation on a square of length l discretised by finite differences at $N \times N$ points are bounded by $-8N^2/l^2$ [HW96]. Therefore for $k = 0$ all eigenvalues of the complete second order system are aligned along the negative real axis in the interval $[-8dN^2/(\rho l^2), 0]$. For $d = 0$ only imaginary components in the interval $[-\sqrt{8kN^2/(\rho l^2)}i, \sqrt{8kN^2/(\rho l^2)}i]$ are present. For $k > 0$ and $d > 0$ the eigenvalues are distributed as shown in fig. 4.2. Analytical considerations allows the computation of the smallest eigenvalue, responsible for the maximal stiffness

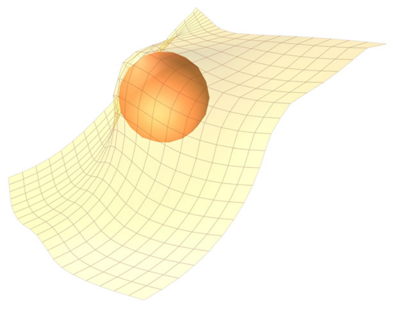
$$\lambda_{\text{min}} = 4\frac{dN^2}{\rho l^2}\Big(-1-\sqrt{1-\frac{k\rho l^2}{2d^2N^2}}\Big). \tag{4.3.2}$$

For a square piece of linen with $k = 5000Nm/m^2$, $l = 1m$, $\rho = 0.2kg/m^2$, and $d = 50Ns/m$, discretised by 20x20 points this gives $\lambda_{\text{min}} \approx -8 \cdot 10^5$, the resulting spectrum is shown in 4.2. Thus the time step for the explicit Euler and midpoint rules would be limited to 2.5μs. Equation (4.3.2) is a more accurate formulation of the quantity called‘*numerical hardness*’ (already including the time step h)

$$\frac{kN^2}{\rho l^2}h^2$$

defined and empirically confirmed by Volino and Magnenat-Thalmann [VMT01].


(a) A tablecloth


(b) A draping sheet hitting a ball.

(c) Dressing up a virtual mannequin.

(d) A walking mannequin.

Figure 4.3: Examples of clothing simulation

	#Newton	#cg	total execution time	construction time
diag	210	774	1.27s	0.04s
chol(0)	183	319	1.10s	0.11s
ssor	182	330	1.09s	0.08s

Table 4.1: Performance of preconditioners, using $h = 0.02$ for 400 particles on SGI R10k/250Mhz.

To compare the real world efficiency of the preconditioners we made a simple second experiment, draping a piece of cloth with a stiffness $\frac{k}{\rho l^2} = 10^6 \frac{N}{cm}$ and damping $\frac{d}{\rho l^2} = 1600 \frac{N \cdot s}{cm}$ over a square table with constant step size for 1s (fig. 4.3(a)). The preconditioner was recomputed in every integration step to measure the costs of its construction. A recomputation is necessary when A (in equation (3.4.1)) is changing, either because of a nonlinear stress or strain law or because h changes.

The results given in table 4.1 confirm the previous ones from the theory chapter. The ssor and chol(0) methods perform almost equally well. The diagonal preconditioner performs clearly worse. First, it requires nearly three *cg* iterations per Newton step, compared to less than two for the competitors. Second, it also requires more Newton iterations. This is explained by the fact, that the good convergence using ssor or chol(0) often leads to a more accurate solution than the residual control requires. This again results in a faster convergence of Newton's method. In the execution time, this is only partly visible, thanks to the low costs of the diagonal preconditioner.

4.1

4.2

Even the slow R10k-system this already was nearly real time, of course using simplified collision detection for the table top and no self collision, which for complex scenes can be one of the major bottlenecks. Figures 4.3 show some additional results produced with the simulator.

4.4 Discussion and Outlook

This chapter presented a numerical solver core for the simulation of textiles. Although the universal pattern will be the same for the following soft tissue simulator, the design will be considerably refined. Up to now, several of the concepts presented in the previous chapters are missing: the faster cg implementation, direct solvers, and stabilisation of the Newton method, as well as many of the integration methods, e.g. any explicit method, for comparison and testing. The concrete implementation lacked easily refinability, abstract basis classes, and flexibly combinable components.

Nevertheless, it provides the numerical kernel, upon which the Virtual-TryOn-Project has been started upon [EKK$^+$01] and furnished a solid basis for several publications and tutorials. Despite of the fact that the integrator is a vital part for simulation, other issues emerged with comparable importance. This comprises the detection and handling of collisions – especially self-collisions and collisions with a dynamic environment, which have been extensively improved [MKE03] and the physical model which was restricted to quadrilateral meshes. The most promising candidates for a generalisation are finite volume methods [TBHF03] that preserve the geometric simplicity of a particle system, or a finite element approach, that has recently been realised [EKS03]. Upon this, more complex applications could be realised, for example a virtual environment for garment assembly [KSFS03].

Chapter 5

Application 2: A Soft Object Simulator

The main application of the presented solid dynamics toolbox is the simulation of volumetric soft objects, which will be discussed in this chapter. The assembly of the framework was governed by the goal to be able to map material properties, which is important for an application in virtual medicine, hence the name VirTis (virtual tissue) has been chosen. Nevertheless, the framework provides the flexibility to cover animation tasks also, where speed is at least as significant as accuracy, by simply changing some components like the strain representation or the material law. If speed matters, stable time integration allows the time step to be increased up to the visual rate of 25Hz or beyond, admitting to animate several thousand elements in real time. Therefore time integration, space discretisation, and material model have been arranged as separate blocks. The architecture of the system will be described in the first section.

The next section will provide a set of examples to investigate the behaviour of the time integration methods. These and the following examples are designed to isolate specific components of the system, to demonstrate their strength and to analyse their weaknesses.

Convergence and its rate, a central issue in numerical analysis, is also a point of interest in visual simulation, perhaps of slightly less emphasis. It states, how much precision is gained when the time step is decreased, influencing the decision how to balance the workload. Stability is a very important issue, as an instable simulation is useless. Both can be studied by computing work-precision diagrams, as we did for the toy example to illustrate the theory discussed in section 3.2.2. Convergence for decreasing time steps is also important for *large* steps tuned for speed, as it proves the implementation to be valid. Some small glitch or wrong design decision in the implementation of a higher order method may reduce the order of convergence to one - or worse give a constant error - and thus the user won't notice any advantage over a simple Euler method. Convergence is seldom considered in graphics literature, the only notable exception is by Volino and Magnenat-Thalmann [VMT01]. The authors compare the number of cg-iterations needed for static equilibrium and the draping speed of a piece of cloth as an indicator for an accurate simulation. Using scripted benchmarks for three-dimensional objects, we are now able to close this gap between the theoretical analysis and real word examples in a more ambitious and satisfying way, than our toy example could provide. As a result, the magnitude of acceptable time steps for a given accuracy can be estimated.

The following section discusses issues of the spatial semi-discretisation and the material models. Some examples show the capabilities and the limit of the finite element method. As analytical solutions for elastic equations can only be obtained for very special cases, e.g. for the ones used by Gladilin [Gla03], we compare the results to a commercial finite element package. Furthermore, we repeat these examples using mass-spring-damper networks, which provide no alternative as soon as material properties need to be mapped. Next, the co-rotated strain tensor will considered and will prove to be an important tool to increase the number of elements while still achieving interactive rates. These experiments follow Hauth and Straßer [HS04]. Besides in this section and in the marked examples in section 5.4.4 we always use Green's strain.

Also several of the examples illustrating visco-elasticity have been taken from the papers published at SCA [HGS03a] and MICCAI [HGS03b]. For the majority, we show the actual surface of the tetrahedral mesh. The illusion of richer surface detail can be added by interpolating the displacement field defined on the finite element discretisation to a finer surface mesh, e.g. also used by Debunne et al. [DDCB01] or Müller et al. [MMD$^+$02]. We employ this technique when animating a mesh that possesses texture coordinates, necessary for advanced vertex and pixel shaders.

Collisions and user interaction are described in the succeeding section. We briefly describe the available methods, focusing on constraint based techniques, which need assistance from the integration method. Some effort will be necessary to combine them with the linear solvers that we introduced. Haptic output is achieved by customising the general idea of a virtual proxy. The last part of this section demonstrates the potential for animation, setting loose parameters for a fast execution, but still staying inside the well-grounded toolbox.

All screen captures and measurements have been performed on a single processor Pentium 4/Willamette Processor with 2.0 GHz.[1] We use double precision floating point numbers, the Microsoft Visual Studio 6 compiler and no special SSE enhancements besides the ones provided by the vendor BLAS implementation from Intel. Switching to single precision floats and SSE/SSE2 could enhance performance considerably, especially for the finite element loops. Single precision halves the memory requirements, thus doubles the available bandwidth, which is a bottleneck for inexpensive loops, e.g. for vector additions. Likewise, single precision SSE instructions increase peak performance from 1 to 4 GFLOPS. Unfortunately, SSE still requires a part of the low-level routines to be manually coded using these extensions.

5.1 System Architecture

The system is organised into several libraries, each providing a specific feature set. The component structure is given in figure 5.1. The bold-faced modules are developed at the WSI/GRIS. The core-libraries for the application are *odesolver, leana,* and *VirTis*, exclusively developed for this project. On top, two applications are provided, VirTisInteractive, which allows the interactive manipulation of a deformable object, e.g. by a Phantom, and VirTisScript, which uses XML to perform off-line computations. These three libraries, together with the two applications contain about sixty thousand lines of C++-code. The GUI is programmed using QT, assisted by the use of the DaVis system developed by S. Gumhold, which provides a reflection mechanism for easy user interface generation.

Two dimensional triangle-meshes and graphical primitives like 3D points are provided by the core-library, collision detection between triangle meshes is performed using the kdop- and collision-library [Mez01, MKE03] developed in the Virtual TryOn project. The rendering is performed using OpenGL. Because soft tissue is usually wet, specular highlights are important, and should be computed per pixel [NHS02]. We therefore implemented hardware accelerated bump mapping, using per pixel normals. The approach follows the work of Kilgard [Kil00] and

[1]Using the SPEC benchmark suite, which includes FEM and ODE solvers, the processor rates at approximately 750 SPECfp2000. On current leading-edge hardware, like a P4EE@3.4GHz (∼1550 SPECfp2000), an Opteron 148 (∼1650 SPECfp2000) or the Itanium2@1.5GHz (∼2100 SPECfp2000), the benchmarks would be faster accordingly.

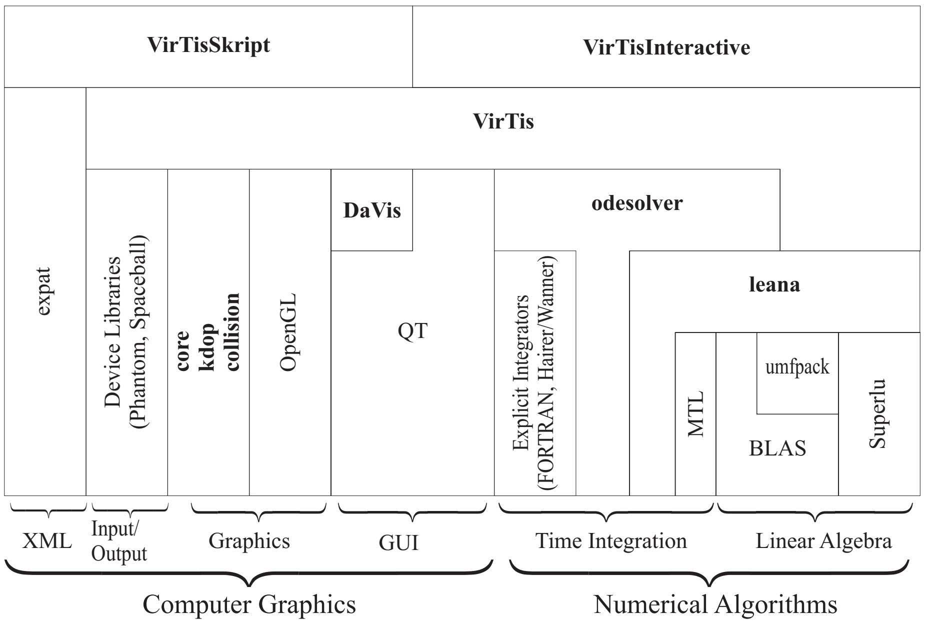


Figure 5.1: Overview over the VirTis system structure.

uses a normal map and a decal texture. For the metallic tools, we employ a reflection shader, using a static environment map. An example using DirectX 8 compliant vertex and pixel shaders is shown in figure 5.2.

leana: The leana-module (Lean Linear Algebra) provides a thin object-oriented wrapper around the standard BLAS library, and adds missing functionality. This is for example the summation of three vectors, which is considerably faster than using two of BLAS's axpys[2]. In addition, it provides compressed sparse matrix and block matrix classes, and implements the interface to UMFPACK, SuperLU, and SuperLU_MT. These freely available third party libraries are used for sparse Gaussian elimination. A legacy interface to MTL [SL99], which was previously used for the cloth-application, is offered. leana was primarily designed to replace MTL.

odesolver: At the moment, the odesolver library provides more than a dozen methods for the time integration of ordinary differential equations, some of them in different flavours. They are summarised in table 5.1. We included some third party code, namely by Hairer and Wanner, to establish an independent 'ground truth'.

The basic design of the library is shown in figure 5.3. An abstract 'ODESolver' base class allows an easy handling through a common interface. It uses an abstract 'ODE' base class, which is implemented by the finite element or mass spring mesh. In addition, the ODESolver class provides a callback after each time step, for collision detection, per-step output, coordinate updates and so on. From this class, explicit and implicit methods are derived, not all of whom are shown. For the third party code, thin wrapper classes are implemented to fit it into the framework. All implicit methods share a polymorphic 'LinearSolver' member pointer that is specialised to the different linear solver types, which are included in the leana module. At this point, which is not shown, also the polymorphism to the different solvers for collision handling, which will be discussed in section 5.4.2, takes place. The implicit Runge-Kutta methods and the multistep methods share a common Newton solver that is stabilised by backtracking.

[2]axpy is the BLAS function call for $\alpha x + y$

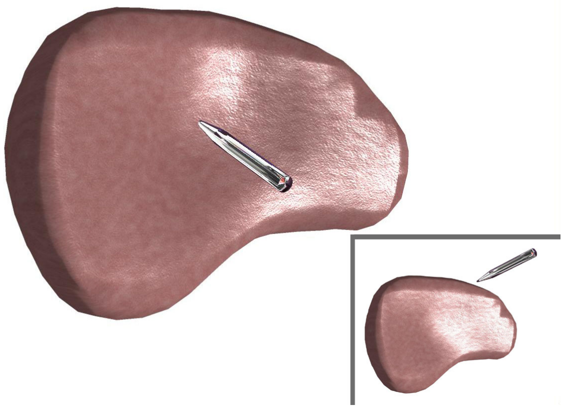


Figure 5.2: A per-pixel bump mapped liver with a reflective tool. The local deformation is revealed by the highlight.

Method	Type	Order	Remark
Forward Euler	Explicit (Runge-Kutta)	1	
Explicit Midpoint	Explicit Runge-Kutta	2	
RK4	Explicit Runge-Kutta	4	
3/8-rule	Explicit Runge-Kutta	4	alternative to RK4
Dopri5	Explicit Runge-Kutta	5	variable timestep, by Hairer and Wanner
Dopri853	Explicit Runge-Kutta	8(5,3)	variable order, variable time step, by Hairer and Wanner
ROCK2	Stabilised Runge-Kutta	2	by Abdulle, time stepping modified.
Stoermer-Verlet	Explicit Runge-Kutta	2	order 2 only for purely oscillatory problems.
Backward Euler	Implicit (Runge-Kutta)	1	
Implicit midpoint	Implicit Runge-Kutta	2	
BDF(2)	Multistep	2	
BDF(3)	Multistep	3	
Rosenbrock2	W-method	2	Jacobian recomputed and factored in every step.
AMF-RK32	W-method	3	Jacobian recomputed and factored in every step.
Rosenbrock4	Rosenbrock	4	Flavours: Shamp, Grk4A, Grk4T, Velda, Veldb, Lstab. Lstab is used for tests.

Table 5.1: Time integration methods provided by odesolver.

The design allows for an almost redundant free implementation of the methods. Especially the abstraction of Newton's method and the linear solver has proven its value. It allowed for the implementation of BDF(3) and the Rosenbrock methods in only a couple of hours each. The abstraction of the callback class permits the comfortable realisation of different kinds of simulators: the multi-threaded VirTisInteractive and the command-line VirTisScript.

VirTis: The VirTis library provides the glue between the different components, which are all almost independent of each other. It offers the functionality upon which higher-level applications are built. The most important components are the tetrahedral meshes, which are specialised to finite element meshes with different strain measures and constitutive laws, and mass-spring meshes. They implement the interface of an ODE, and therefore ODESolver is able to handle these active meshes. This is also the reason, why no separate FEM-module is constructed, as these meshes fuse graphical and numerical concepts, and we do not want to have derivations across several libraries.

For a versatile usage, the design is organised in a model-view-controller fashion. We gener-

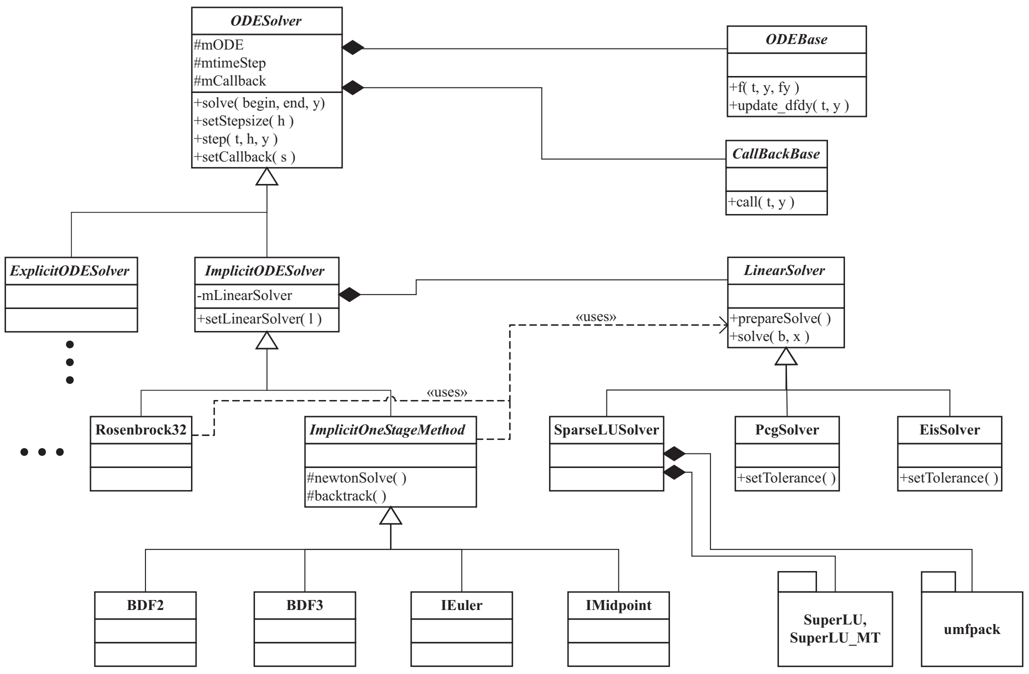


Figure 5.3: UML class diagram of the main components of the odesolver library. Only the most important members are shown.

alise the controller concept to physical and geometrical controller objects, which can be applied to a model given as an active mesh, similar for example to the 'SpaceWarps' of 3DStudioMAX. As an example of this concept, a fragment of the class hierarchy is shown in figure 5.4. Forces, derived from an abstract base class can be used to manipulate a TetraFEMMesh, and may be restricted to specific regions of the mesh. The same restriction concept is used to pin points or faces during a simulation. The Visualizer family of classes, attached to the mesh, is used to display the resulting constraints and forces during runtime. An example is shown in figure 5.5, where we used four NormalRestrictors to select the four opposite faces shown in pink, LiveRstTangent-Load forces are used to produce the shear pattern. A BoundingBoxRestrictor in combination with a SurfacePointImmobilizer fixes the lower edge, as shown by the red cubes displayed by the ConstraintVisualizer. Another example of the output of stress and strain visualizers is presented in figure 2.1, where we used ellipsoid glyphs to show stress and strain.

The most important communication procedures of VirTis are protected by mutexes, to allow the use in a multi-threaded application. The computationally most intensive routines are easily parallelised: the finite-element routines are expensive loops over all tetrahedra or edges, for the sparse solvers already parallel implementations exist, and the collision detection also offers possibilities for a parallel execution. The communication latency though must be short in order to achieve interactivity or real-time: if for example 10ms time steps are taken, all latencies should be considerably lower. Hence, the parallelisation on a closely coupled shared memory multi-processor would be an interesting option.

VirTisInteractive: The VirTisInteractive application is multi-threaded, although on a coarse level. It uses several parallel threads to carry out a simulation. The sequence diagram of a typical execution is shown in figure 5.6. First, the main thread forks off a GUI, that is itself multi-threaded through the use of QT, and a heartbeat thread, that ensures a smooth animated rendering

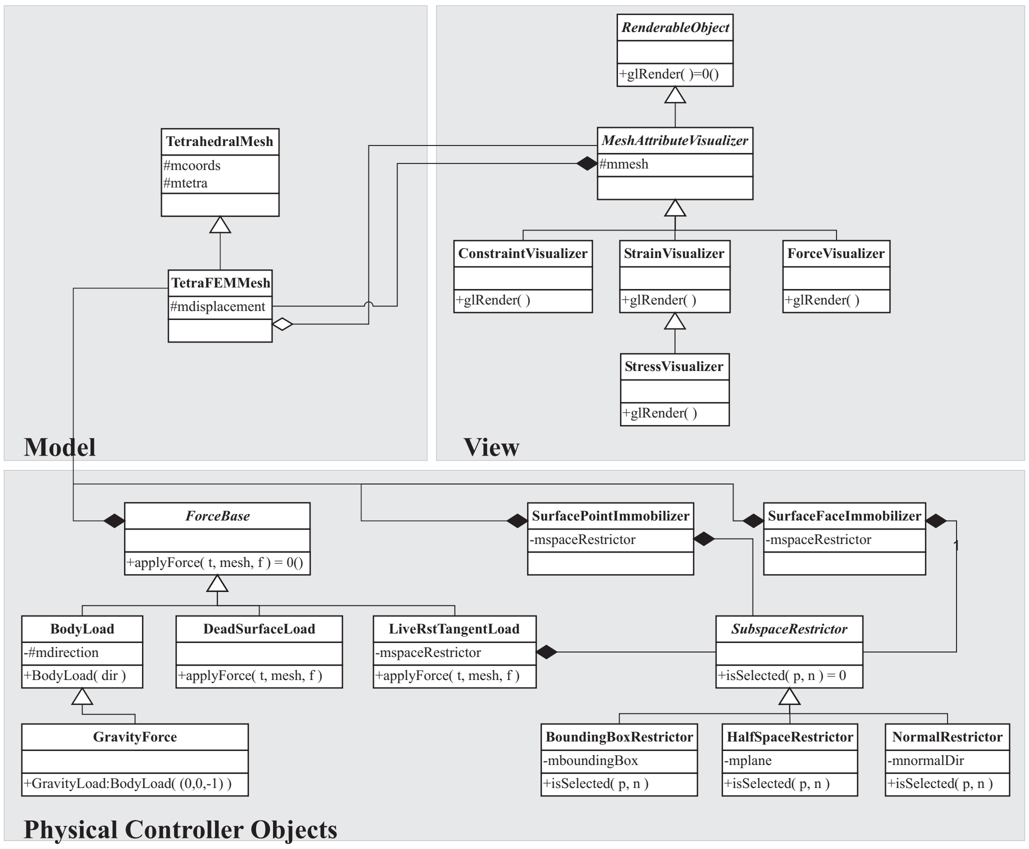


Figure 5.4: Subset of the model-view-(physical) controller architecture of the VirTis library.

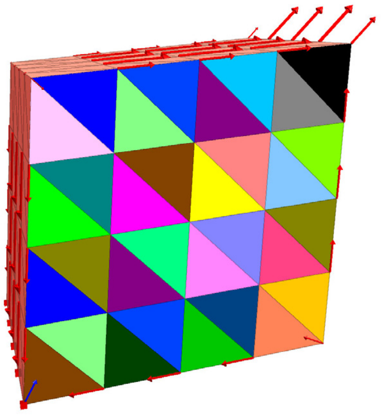


Figure 5.5: The sample scene was produced by attaching a shear force pattern to a TetraFEMMesh and fixing an edge in space. A ConstraintVisualizer and a ForceVisualizer class is used to produce the picture. Note that it is correct, that the forces have different magnitude: they are produced by integrating the product of a force-density and the shape functions over the surface. The forces shown in red are unconstrained; the blue one is nullified by the constraint.

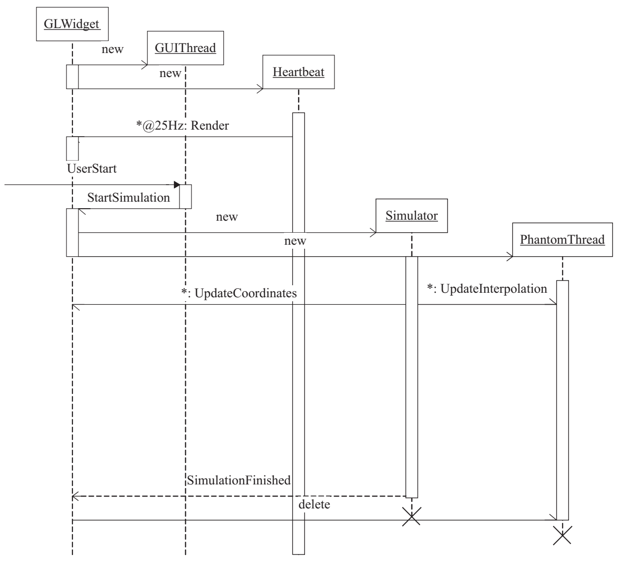


Figure 5.6: UML Sequence diagram for the multi-threaded VirTisInteractive application.

by generating refreshes at an interactive rate. As soon as the user has configured the scene and started the simulation, a thread for simulation and force feedback is created. The simulation thread communicates updated coordinates to the main thread, which does the rendering, and to the Phantom thread for extrapolation. Using a separate thread for the haptic rendering ensures the tight 1 kHz refresh and is recommended and supported by the Sensable GHOST toolkit [Sen03]. The separate GUI and rendering threads guarantee responsiveness. Even if a simulation step would take several hundred milliseconds, the model could for example be rotated smoothly by the user. All screenshots and live-captured animations are taken from VirTisInteractive.

VirTisScript: The VirTisScript application serves a different purpose. It is intended to perform unattended experiments, with an easy variation of parameters, e.g. to generate the work-precision diagrams shown below. Using VirTisScript, about a thousand test-runs were performed with only several dozen lines of configuration. VirTisScript is single threaded and used from the command line. The scene and the components are configured by an XML-configuration file; XML-processing nodes are used for scripting commands. The results are again written to XML-files. The underlying paradigm is that of a state machine. XML nodes configure the state, and as soon as a processing command is encountered, it is executed with the current state. Then some variables may be changed, and the command may be repeated with the updated state. We use expat [eXp03] as an XML parser, that implements the publicly developed SAX (Simple API for XML), and is also used in the Mozilla project. In this case, SAX is superior to the W3C DOM specification [W3C], because the XML document is parsed in a serial, event-based manner, which nicely fits the scripting and state-machine model. An example configuration file is shown in figure 5.7. It first loads a scene, configures the 'gold standard' Dopri853 solver to compute a reference solution, loads the solution, and then performs some experiments with different solvers.

```
<VirTisScript>

<!-- Mesh -->

<FEMesh>
 <MeshFile type="string">        ../models/diss/bar30x_144.volmesh
  </MeshFile>
 <lambda type="double">           300000                    </lambda>
 <mu type="double">               3000                      </mu>
 <dlambda type="double">          3000                      </dlambda>
 <dmu type="double">              30                        </dmu>
</FEMesh>

<!-- Compute reference solution -->

<Simulator>
 <writeEndMesh type="string">    refsol/solution           </writeEndMesh>
 <statusOut type="string">       refsolStatus.xml          </statusOut>
</Simulator>
<Solver>
 <integrator type="string">       Dopri853                  </integrator>
 <timestep type="vdouble">        1e-14                     </timestep>
 <starttime type="double">        0                         </starttime>
 <endtime type="double">          1                         </endtime>
</Solver>
<?simulate ?>

<!-- Do some testing with different integration methods -->

<Simulator>
 <writeEndMesh type="string">     testResults               </writeEndMesh>
<!-- This decorates the results by _solvername_simtime_timestep -->
 <perEndFileTags type="string"> nth                         </perEndFileTags>
<!-- Writes status information like time
     and error to a protocol file -->
 <statusOut type="string">       Protocol.xml              </statusOut>
</Simulator>
<Solver>
 <integrator type="string">       IEuler                    </integrator>
 <timestep type="vdouble">        1e-1 1e-2 1e-3 1e-4       </timestep>
 <linearSolver type="string">    SparseLU                  </linearSolver>
</Solver>
<?loadReference refsol/solution.volmesh?>
<?simulate ?>
<Solver>
<integrator type="string">       BDF3                      </integrator>
</Solver>
<?simulate ?>
<Solver>
<integrator type="string">        RK4                       </integrator>
</Solver>
<?simulate ?>

</VirTisScript>
```

Figure 5.7: A sample script for processing by VirTisScript. It loads a mesh, configures DOPRI853 to compute a reference solution, loads the solution, and then evaluates the performance of some time integrators.

5.2 Time Integration

This section will investigate the performance of the time integration schemes provided by ode-solver, and compare the linear system solvers. As already laid out, theory allows a decision upon a suitable set of methods, but in the end, the most efficient method can only be found by experiment. We therefore constructed two benchmarks for evaluation and testing. Especially the second one, carefully constructed around a potential source of instability, will be a hard test.

5.2.1 Example 1: A Bar under Gravity

Setup

The first example employs a bar of 0.3x0.1x0.1m^3 edge length, with homogeneous Dirichlet boundary conditions on the left end. The bar has a density of $\rho = 1000$kg/m^3 and is subjected to a gravity load. We take a St.Venant-Kirchhoff material with parameters $\lambda = 0.3$MPa and $\mu = 3$kPa. For damping we employ the visco-elastic correspondence principle, viz. we use the same elastic operator now applied to the velocity (see for example [OH99]), with parameters d$\lambda = 3\mathrm{kPa} \cdot \mathrm{s}$ and d$\mu = 0.03\mathrm{kPa} \cdot \mathrm{s}$. The bar is discretised into a mesh with 144 tetrahedra, and 63 nodes, leading to 378 coupled equations. Starting from a horizontal position, the bar drops about 3.5cm in about 0.5s, and then swings back about 0.5cm. Some small vibrations die out between $t =$1.0s and $t = 2.0$s.

5.1

The diagrams are generated by using all methods with different time steps and measuring the execution time. In addition, for BDF methods, the backward Euler, and the implicit midpoint rule, we vary the exit conditions from the embedded Newton method and exchange the linear system solver. To establish a measure for the error, a reference solution y^* was computed using Dopri853 with a very small error tolerance $\mathrm{atol} = \mathrm{rtol} = 10^{-14}$ for the integrated stepsize controller [HNW93]. We take the reference solution at $t = 0.5$s, which coincides roughly with the point of maximum displacement. Taking a point later than 1-2s would distort the picture, because the equation would be approaching its stable limit, as the flow of the differential equation is contractive. Any stable integration will reach this limit eventually, no matter how accurate the dynamics have been computed, which results in a completely different work-precision diagram (see [HES03] for a discussion of the toy example of section 3.2.2). The error is then measured in the RMS-norm

$$e = \|y^* - y\| = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i^* - y_i)^2}.$$

time step [s]	run time [s]	#Newton	#Newton/step
0.100	0.156	84	14.00
0.050	0.218	125	10.42
0.020	0.218	175	6.73
0.010	0.234	226	4.43
0.005	0.218	198	1.96
0.002	0.468	396	1.58
0.001	0.765	648	1.29

Table 5.2: Details for example 1, 0.5s simulation time, using BDF(3).

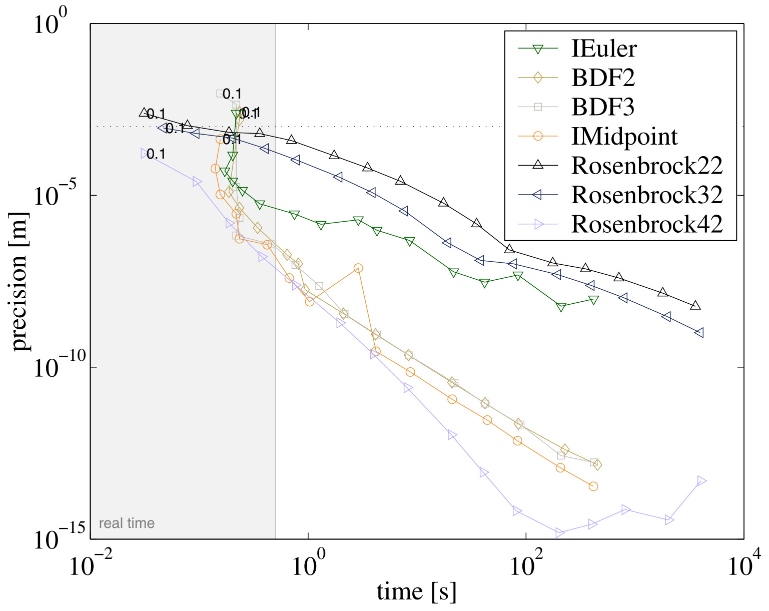


Figure 5.8: Work-precision diagram for the implicit methods with loose Newton tolerances applied to example 1.

Results and Discussion

The first two figures 5.8 and 5.9 show the various methods applied to the example. For the first 4 methods (implicit Euler and midpoint, BDF), Newton's method is terminated, whenever the correction step length is smaller than 10^{-5}m/s, or the residual force is less than 10^{-5}N. This means the rooted sum of squares of *all* velocity corrections is less than 10^{-5}m/s, which means for an individual point less than 1μm/s in the average, and the same for the forces. The time step varies between 0.1s and 10^{-6}s. In addition to the time-error trade-off, the figures depict the largest tested time step, for which the method was stable, given by the numerical values at the specific data point. The gray rectangle denotes the region, where the method is able to give real-time results. The dotted line finally marks a RMS-error of 1mm, which we consider as tolerable.

All the implicit methods (figure 5.8) allow a stable real-time simulation of the bar. For the largest time step, the Rosenbrock-type methods perform best, the fourth order method in the lead. The methods of order two and three although are at the accuracy-limit we wish to achieve. For large time steps, the schemes using an embedded Newton method show a peculiar behaviour. Namely, for decreasing time steps they need the same or even less work while simultaneously the error decreases. This can be explained by a decreasing number of Newton iterations per time step, and is typical for these codes. In table 5.2 we give the relevant data for this phenomenon using the BDF(3) method, for the other methods the picture is the same. The decreasing number of iterations is due to the smaller time steps: the state vector does change less from one step to the other and the iteration starts with a better initial guess, and therefore converges faster.

Looking at the behaviour for small time steps, all methods smoothly converge towards machine precision. As soon as the error drops below the femto-meter scale, the reference solution start to loose its accuracy, which can be rated at about 10^{-14}. At this level, also rounding errors start to dominate the process, such that more time steps mean less accuracy. Furthermore, the

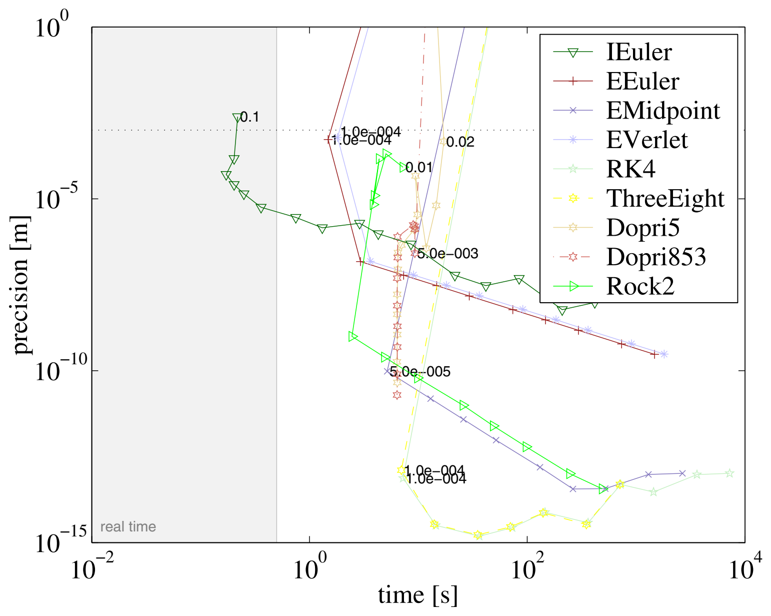


Figure 5.9: Work-precision diagram for the explicit methods applied to example 1. The implicit Euler method from above is shown for comparison.

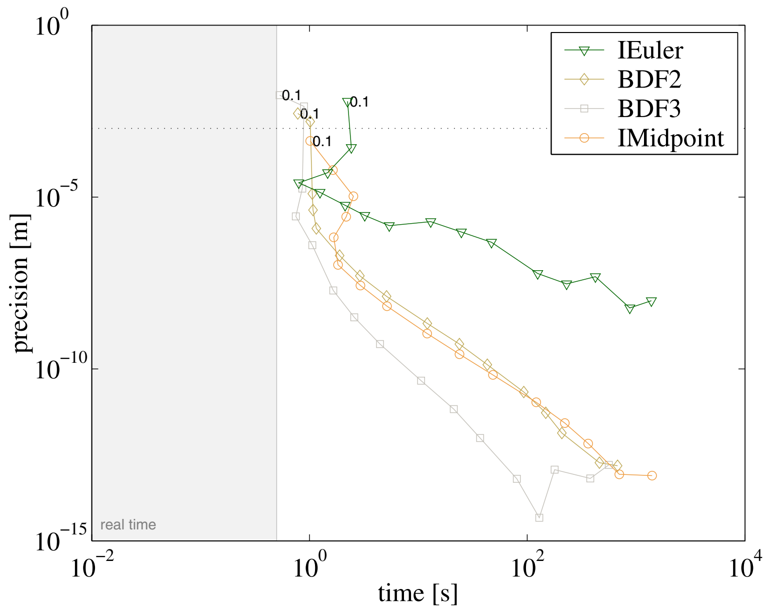


Figure 5.10: Work-precision diagram for some of the implicit methods with strict Newton tolerances applied to example 1.

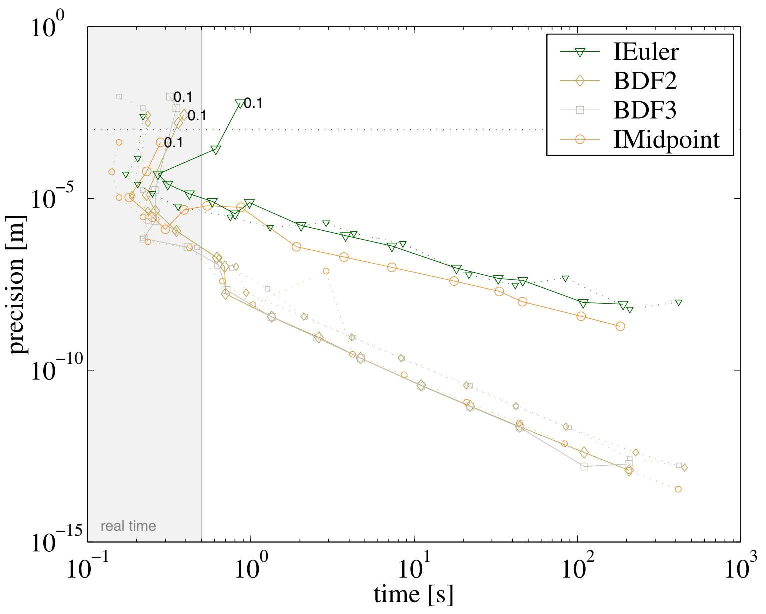


Figure 5.11: Work-precision diagram for some of the implicit methods. The dotted lines are for a sparse direct solver, the solid ones for a SSOR-preconditioned conjugate gradient method.

picture reveals the order of the method as the slope of the curve in the double logarithmic plot. This can be observed nicely for the Rosenbrock42, implicit midpoint and BDF(2) method. Note that BDF(3) has not its full order, which is due to the lax Newton stopping criteria, as we will show below.

None of the explicit methods (fig. 5.9) is suitable for real time for this example. They need small time steps, usually below 1ms to be stable. The only exception is the stabilised ROCK integrator. It is stable up to 10ms, although the convergence is irregular for large steps. A soon, as the time step drops enough, it competes with the best methods. As there are no iterative processes for the explicit methods, the convergence behaviour is generally smoother than for the implicit methods. Note that the Dopri methods possess an embedded step size controller, the numbers specified are error tolerances not step lengths for this integrator. The curves for Dopri are nearly vertical: after reaching the stability limit, only small changes in step size at the most critical points in time allows a very fast reduction of the error.

Figure 5.10 shows the convergence of implicit Euler, midpoint, and BDF, when the Newton iteration is run until convergence. Now the BDF(3) method reaches its full order three, BDF(2) and the implicit midpoint rule run nicely in parallel as before.

The last figure 5.11 compares the performance of the integrators, if the sparse direct solver is replaced by a SSOR-preconditioned conjugate gradient method in Eisenstats implementation. The dotted lines show the Gauss-type methods and are the same as in figure 5.8, the solid lines denote the cg method. For this dimension, the direct solver is about two to three times faster than the iterative solver. The experiment also shows that the use of an inner-outer iteration like Newton/cg inside the numerical time integration does not disturb the convergence of the methods.

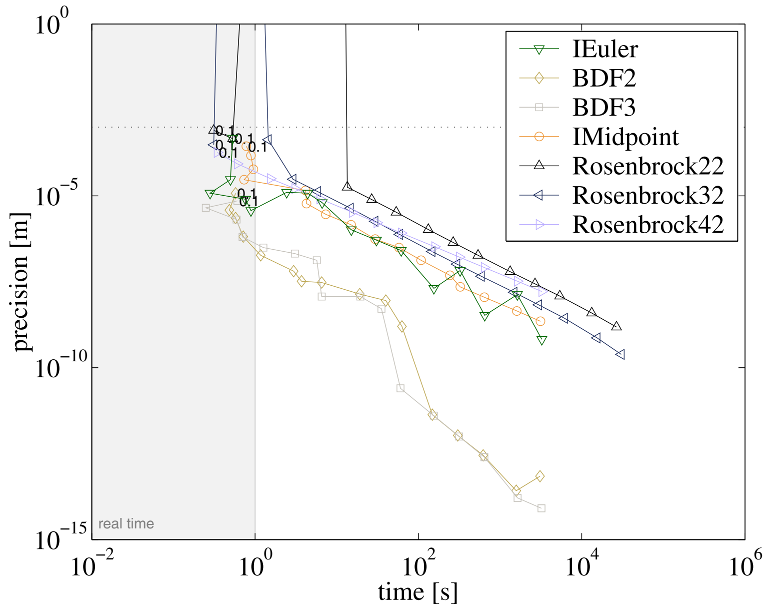


Figure 5.12: Work-precision diagram for the implicit methods with loose Newton tolerances applied to example 2.

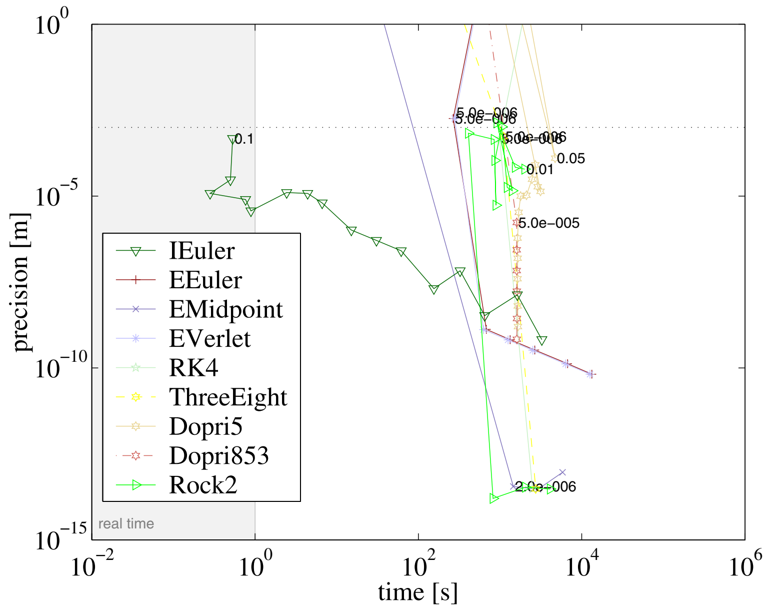


Figure 5.13: Work-precision diagram for the explicit methods applied to example 2. The implicit Euler method from above is shown for comparison.

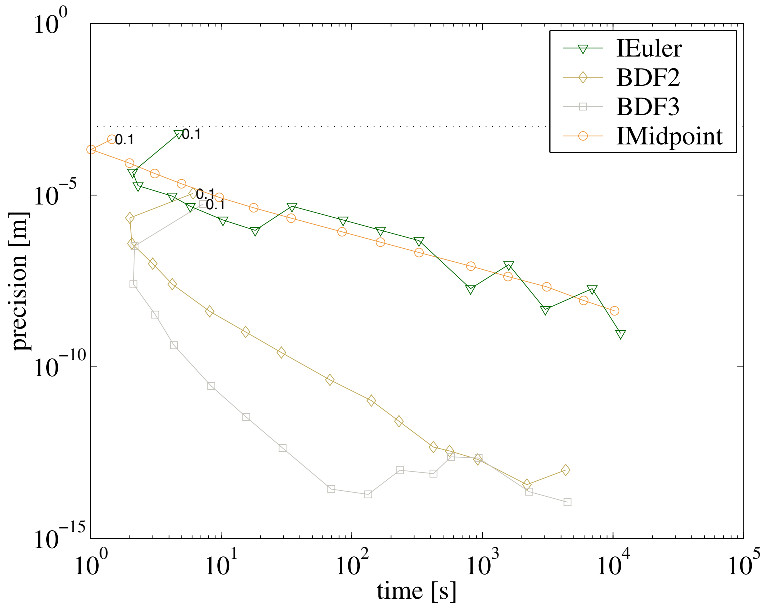


Figure 5.14: Work-precision diagram for some of the implicit methods with strict Newton tolerances applied to example 2.

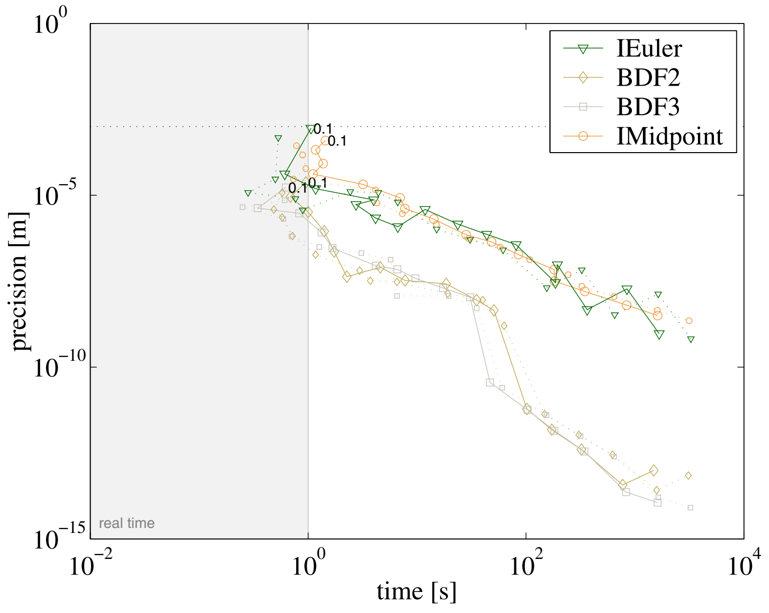


Figure 5.15: Work-precision diagram for some of the implicit methods, applied to example 2. The dotted lines are for a sparse direct solver, the solid ones for a SSOR-preconditioned conjugate gradient method.

5.2.2 Example 2: Pure Shear

Setup

The second example used to evaluate the time convergence is a so-called pure shear experiment, illustrated in figure 5.5. We apply a set of tangential shear forces to two pairs of opposite faces of a cubic object, such that the forces on one side cancel the ones on the other side. Note that the forces are live, as their direction depends on the surface normal. The object is fixed at the lower left edge. Ideally, the forces cancel during the whole shear movement, and the object only deforms without any global rotation. This behaviour however critically depends on the face normals of the mesh, which are computed from the current vertex positions. As the forces are live, the normals directly influence the force computations. Due to the finite machine precision, after a few time steps, these normals on a single face coincide only up to an angle of say 10^{-12} degrees. If the time integration does not update the vertex positions accurately enough, this deviation increases very fast and the object drifts into a skewed rotation. Therefore, this benchmark is very sensitive to the accurate and uniform update of the vertex positions. Small-scale numerical noise quickly builds up to a macroscopic error.

As material parameters we chose $\rho = 10^3\text{kg/m}^3$, $\lambda = 0.1\text{MPa}$, $\mu = 1\text{kPa}$, $\text{d}\lambda = 0.02\text{MPa} \cdot \text{s}$, $\mu = 0.1\text{kPa} \cdot \text{s}$. 5.2 The cube of edge length 0.1m is discretised into 384 tetrahedra with 125 nodes. We employ a ramp force to the four sides, rising from zero to a force density of 200N/m^2 in one second. This gives a final force of 2N to each of the four sides. 5.3 The viscous parameters are chosen in such a way, that the elastic response is nearly on the instant and the cube reaches its final elongation with a displacement of about 1cm (10%) at about 1.2s, with rapidly dying out oscillations. 5.4 A good method manages to keep the cube at this position for about 20s, a bad one doesn't even reach one second. We again perform the same set of experiments with the same evaluation as in the previous section, this time measuring the error at $t = 1.0$s.

Results and Discussion

Figure 5.12 shows the results of the implicit methods in this experiment. All methods allow a real time simulation of the benchmark. The low order Rosenbrock methods, the most inaccurate in the last example, have difficulties for the larger step sizes to meet the accuracy goal for a stable simulation. The fourth order Rosenbrock method, the best performing method in the last experiment, now is only average. The other methods pass the test, though convergence is slightly more irregular, and again not all methods reach their full order for loose Newton tolerances. Note that for the largest time step we take only 10 steps of a time varying, slightly ill conditioned, geometrically non-linear problem. This again requires a high number of Newton iterations per step, leading to the same phenomenon as discussed in the last section.

For the explicit methods shown in figure 5.13 this setting is even less forgiving than the last. Whereas before they missed the real time capability by a factor of five to ten, this time it is closer to a hundred and more. An interesting observation can be made with the two Euler methods: As soon as the forward Euler method becomes stable, both methods show a very similar performance. The major difference is that the curve of the implicit method continues to the left to a larger error with lower cost. After the explicit methods reach their stability limit, they are immediately very accurate due to the small time step.

In figure 5.14 we again run the Newton iteration until convergence. As before, the convergence behaviour becomes more regular. The first two data points of each curve show, that the largest time step is in fact too large: with a smaller one, all methods need considerably less work.

We also exchanged the solver for Eisenstat's method; the results are shown in figure 5.15. Due to the larger dimension, the gap between the direct (shown by the dotted lines) and the

#tetrahedra	absolute error [mm]	relative error in %
6	0.588	4.26
48	0.780	5.65
384	0.407	2.95
3072	0.092	0.66
24576	0.021	0.15
1168	0.015	0.11

Table 5.3: Comparison of the error in the z-component of the left front corner of the cube in figure 5.16.

iterative method becomes smaller, as already seen in the experiments of section 3.4.4. It also can be seen, that the convergence is very similar and does not depend on the linear solver.

Summarising the results of this section, we state that most of the higher order implicit integration methods allow the simulation of soft objects with time steps of 25Hz and above, yielding a precision that is sufficient for visual simulations.

5.3 Materials and Spatial Discretisation

The next set of experiments investigates the behaviour of the spatial discretisation techniques we presented. The first one will show the convergence of finite elements. The second one emphasises, that although finite elements are a valuable method, convergence can sometimes be difficult to achieve, especially when important physical features can only be resolved by a fine mesh. The experiments will be repeated using mass-spring-damper meshes and reveal that this approach suffers from some difficulties, e.g. topology-dependant material attributes and a lack of volume conservation. It will be shown, that the hierarchical basis is equivalent to the nodal one, and we will demonstrate its benefits.

Some experiments with corotated strain will certify this linear, rotational-invariant strain measure as useful for reducing computational costs, at the expense of some accuracy. Finally some experiments with visco-elasticity will be given. Visco-elastic laws are needed on the one hand to reproduce biological tissue properties realistically, and on the other hand can provide effects for animation purposes that can otherwise only be produced manually.

5.3.1 Example 3: A Suspended Cube

The next example considers a cube, which is fixed at its upper face and subjected to gravity. The material parameters are set to $\lambda = \mu = 1\text{kPa}$, allowing some volume compression, and dλ =d$\mu = 0.1\text{kPa} \cdot \text{s}$, quickly damping out most oscillations, as we are heading for the rest state. For time integration we use BDF(3) with a small step size of $h = 1\text{ms}$. Figure 5.16 shows the relaxation state of the several different tetrahedralisations of the cube after 10s of simulation. The rest length visually appears the same. To quantify this, we compare the z-displacements of the left front corner. The error is shown in table 5.3. For the first level of subdivision, the error gets larger. This shows, that the coarsest mesh cannot really represent the deformation state, but just has been lucky in this example. With an increasing number of tetrahedra, the error decreases, as the underlying theory predicts. It also predicts that the achievable accuracy critically depends on the quality of the mesh, which is indicated by the last line. Whereas the other meshes have been generated by subdivison of the initial coarse mesh, this mesh with 1168 nodes has been directly generated from a cube by an external triangulator, which takes quality measures, like

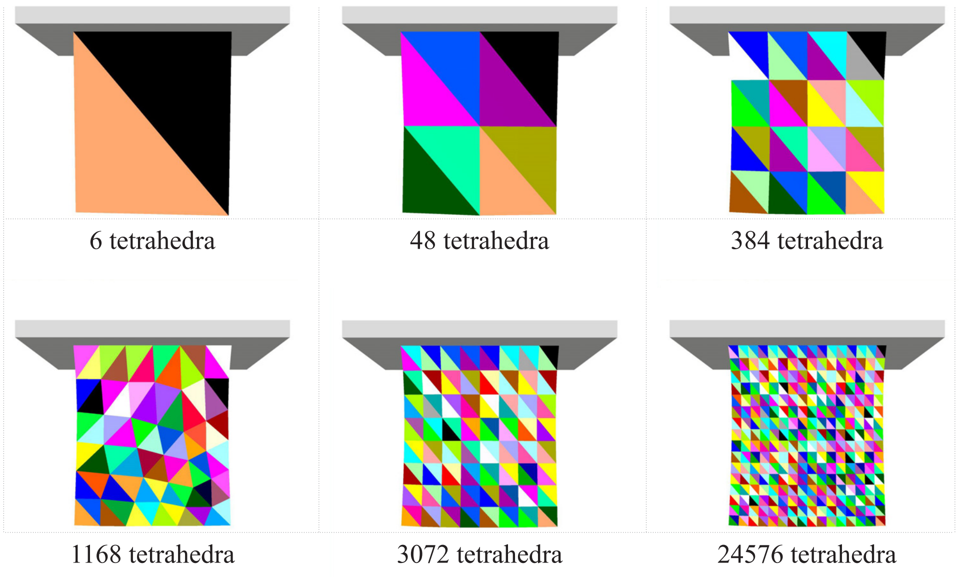


Figure 5.16: Rest state of a cube under the influence of gravity. The tetrahedralisation is varied from coarse to fine, the meshes with 6, 48, 384, 3072 and 24576 tetrahedra are generated by octasection. The remaining two meshes are unrelated. Besides the coarsest mesh, the relaxed length of the meshes visually looks the same.

edge length grading and minimal angles into account. As a result, it shows the smallest error, despite possessing only a relatively small number of elements.

5.3.2 Example 4: A Bar under Gravity Revisited

The following experiment again employs the cantilever from the first benchmark for time convergence (sec. 5.2.1). Now, the mesh will be varied. This example has been inspired by a similar setup used by Müller et. al. [MMD$^+$02]. In their presentation however, Müller et al. did not investigate the effect of changing the discretisation, and thus the authors did not discuss the peculiarity that will be described next.

We use different related and unrelated tetrahedralisations. For time integration, we make use of Dopri853 with a strict tolerance to ensure stability and exclude artifacts from the temporal discretisation. The results are given by the blue bars in figure 5.17. This time the results on refining the meshes are quite different, although a general trend is visible. The bar globally drops farther with increasing resolution. To investigate the phenomenon, we did the same simulations with comparable tetrahedralisations using ABAQUS, one of the leading commercial products for finite element elasticity. The red bars show the results.

The coarse red bar, computed by ABAQUS using the same elastic model with linear tetrahedral elements, and the coarsest blue bar look similar, indeed the difference in displacement is about 2mm in z-direction, taking the left upper front corner as a reference point, which is one of the points of maximum elongation. Moreover, the fine, rightmost red bar looks similar to the fine blue one; in this case, the difference is 5mm or about 3.7%. The displacement computed by

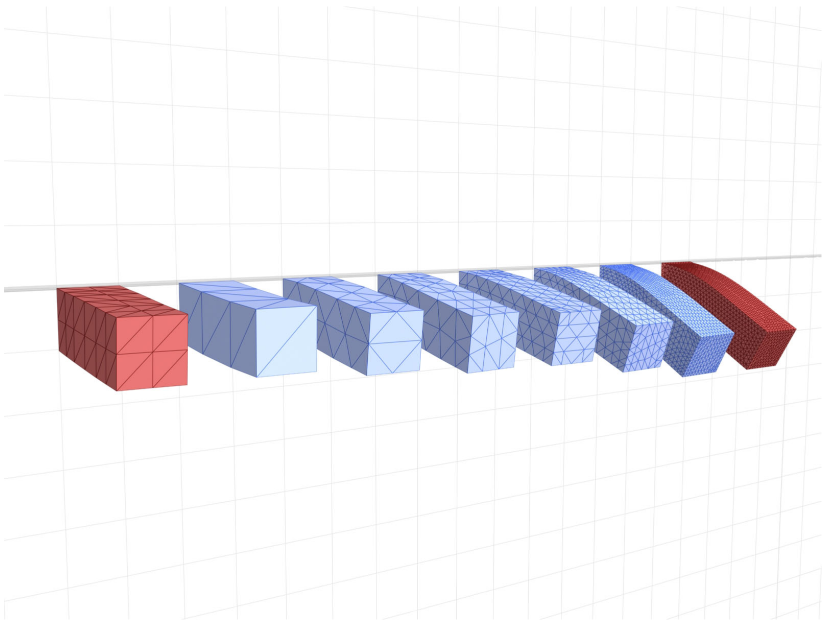


Figure 5.17: A cantilever attached to the wall, computed with different meshes (144, 16, 85, 260, 680, 2080, 16640, 21784 tetrahedra, from left to right). The red bars are computed by ABAQUS, the blue ones with VirTis. From coarse to fine, the global shape changes considerably, indicating slow convergence. A third ABAQUS simulation (not shown) indicates that the fine red mesh is a good approximation to the true state.

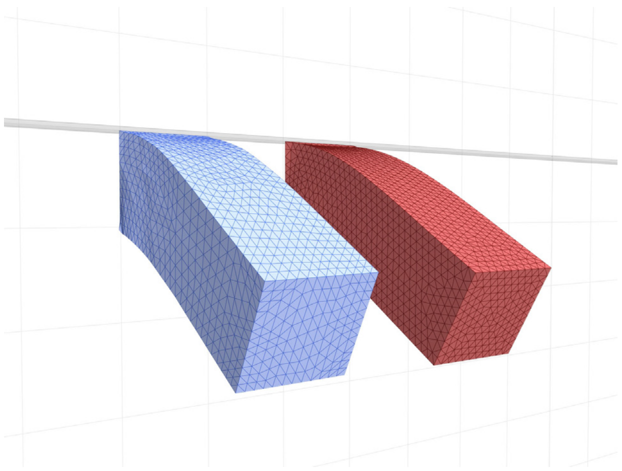


Figure 5.18: Detail comparison of the last two beams from figure 5.17. Visually, the results are similar, indeed the difference in z-displacement is at most about 4%.

ABAQUS is 133.86mm, VirTis yields 128.8mm with the coarser mesh[3]. The fine mesh computed by ABAQUS possesses 21784 tetrahedra and has been generated using quadratic, mixed elements with decoupled pressure to counteract any locking. Due to the quadratic formulation, the number of degrees of freedom corresponds to a linearly simulated mesh with 174272 tetrahedra, although this comparison neglects the higher approximation order of the quadratic displacement field.

Hence, the VirTis framework is able to reproduce the results of a commercial program within a tolerance of a few percent, but in this special benchmark, the convergence is slow and needs more than 10000 elements to yield an acceptable accuracy. However, this behaviour is also present in ABAQUS; it is a problem of the method, not of the implementation. The reason behind this can be found in the nature of the experiment and is illustrated in figures 5.19 and 5.20, which show the red meshes from above visualised in ABAQUS. The colour code denotes the van-Mises stress, which is defined as the rooted sum of the squared differences of the three eigen-stresses. This is a measure for the shape of the local stress tensor and is considered a good indicator for the focus points of stress, where the material may fail[4]. The finely resolved bar shows a very homogeneous stress on the right-hand side, which is nearly straight. In the left-hand part, there is a sharp increase of van-Mises stress of about three orders of magnitude towards two small geometric features next to the upper and lower fixed edge. This local features, with an extent of less than 1cm carry the main load, their configuration ultimately determines the global shape and the height displacement of the other end. A coarse mesh, as shown in figure 5.19 is simply not fine enough to represent these features, and consequently the global error is large. With the coarse resolution, the stress contour is a smooth slope from left to right of only one order of magnitude.

For comparison, we display the van-Mises stress contours of the previous example in figure 5.21. Although there is also a steep rise towards the fixed corners, the overall contour is more regular, which can be represented on a coarser mesh with a reasonable accuracy. This explains the smoother and quicker convergence in the previous example.

Discussion

As a conclusion, we therefore state, that finite elements give good results, as long as the features, at least those who determine the global shape of the object, are reasonably resolved by the mesh. In engineering, such cases are handled by a non-uniform meshing (here, concentrating elements on the left), which is usually generated manually by the user, who has insight in the physical nature of the problem. This is difficult for interactive visual simulations, as the loading conditions cannot always be anticipated, which is why we chose to present this example.

5.3.3 Mass-Spring-Damper Meshes

After the investigation of the capabilities and limitations of the finite-element approach, we model the same examples with a mass-spring-damper mesh. Mass-spring-damper meshes are very common in graphics, hence their use is considered for comparison. The implementation of a tetrahedral mass-spring-damper mesh, especially the fitting of spring and damper constants, was designed carefully. The evaluation procedure for springs and damper forces was organised in a way to provide a sensible amount of efficiency, such that a fair comparison is possible. Details about our implementation can be found in Appendix B.

[3]A second ABAQUS simulation with a finer mesh (75931 quadratic tetrahedra) yields a displacement of 133.68mm, a deviation of 0.2mm, which confirms that finally for the finer resolutions the simulation is convergent up to the millimetre range and thus can be considered as a reference for VirTis. It took 17 Newton iterations to reach equilibrium, requiring a total of 7h computation time (wall clock time).

[4]At least for the common engineering materials like metal and concrete.

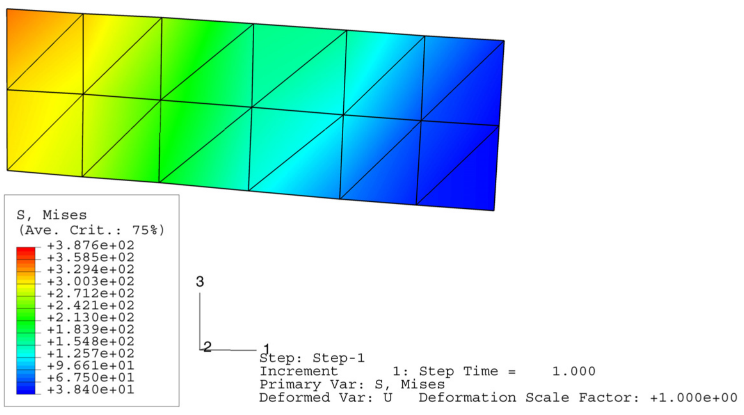


Figure 5.19: Van-Mises stress contours of example 4 in a coarse simulation, computed by ABAQUS.

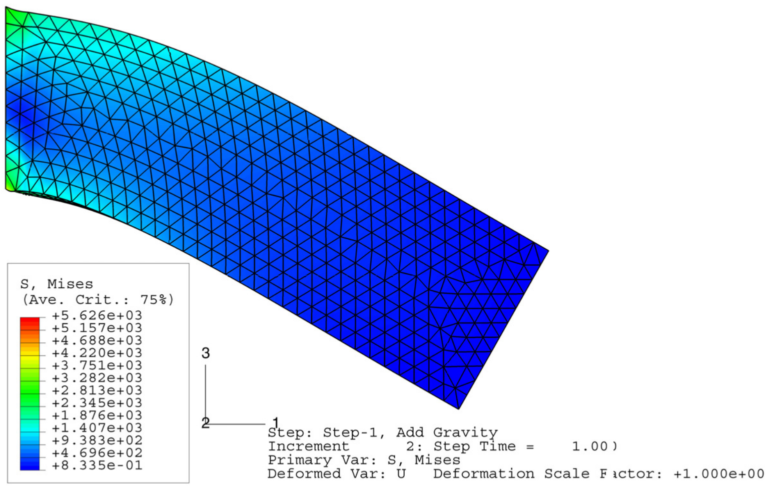


Figure 5.20: Van-Mises stress contours of example 4 in a fine simulation, computed by ABAQUS.

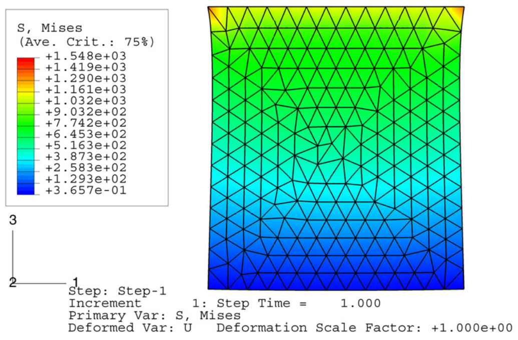


Figure 5.21: Van-Mises stress contours of example 3, computed by ABAQUS.

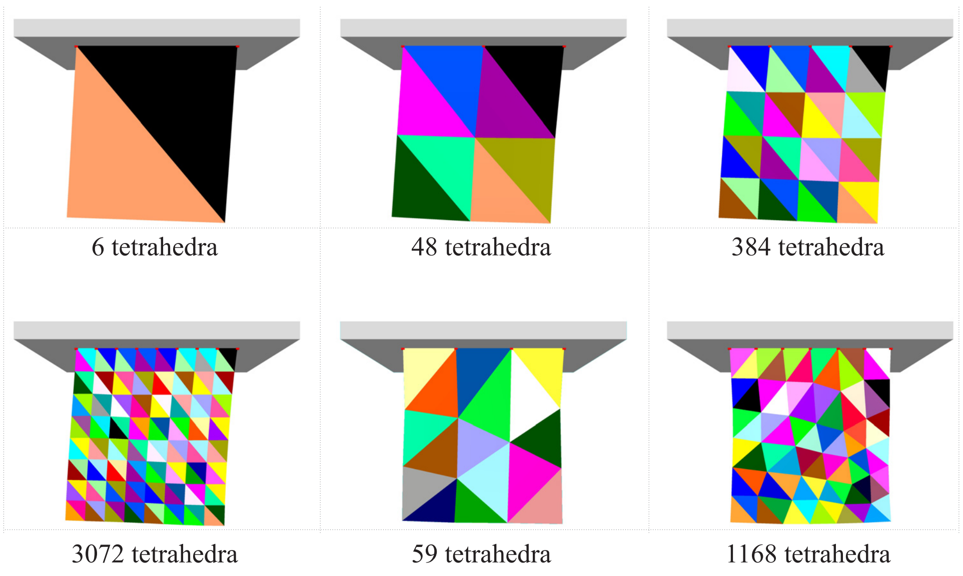


Figure 5.22: Example 3 with mass-spring meshes of different topologies. Each mesh yields some more or less severe distortions.

Example 3 using Mass-Spring-Damper Meshes

We begin with the suspended cube. The spring constants were computed using the formula (B.1.1) of van Gelder [VG98]. Figure 5.22 shows, that all meshes derived from the 6-tetrahedra cube suffer from an incorrect distortion to the left, which apparently is an artefact of the mesh topology. The last two subfigures employ directly generated tetrahedralisations, which give better results. Nevertheless, the cube with 59 tetrahedra is also biased to the left and its left edge is straight and not symmetric to the right. The cube with 1168 tetrahedra is slightly biased to the right. Here the right edge does not possess the correct concave shape, disturbing the appearance as it again is not symmetric.

5.5 5.6 5.7 5.8

Dynamics: It becomes worse, when we look at the dynamics of the meshes. For the following experiments we used BDF(3) with a time step of 1ms. We investigated high and medium settings for the viscous damping and compare them to the St.Venant-Kirchhoff material with $\lambda = 1\text{kPa}$, and $\mu = 1\text{kPa}$, $\mathrm{d}\lambda_1 = 0.1\text{kPa}\cdot\text{s}$, $\mathrm{d}\mu_1 = 0.1\text{kPa}\cdot\text{s}$, and $\mathrm{d}\lambda_2 = 5\text{Pa}\cdot\text{s}$, $\mathrm{d}\mu_2 = 5\text{Pa}\cdot\text{s}$. The elastic parameters imply a Young modulus $E = 2.5\text{kPa}$, which is plugged into van-Gelders formula. As it can be seen from the rest experiments in figure 5.22 this results in approximately the same relaxed length. A Poisson ratio cannot be set. The damping constants were tuned manually, such that in the first case, there are a few oscillations and the cube comes to rest in the first 5 seconds. In the second case, we headed for visible oscillations up to 10 seconds.

We used three different strategies to distribute the damping parameters onto the mesh. The first uses the same damping parameter for all spring-damper pairs, taken to be proportional to Young's modulus. The selection of the proportionality factor then allows us to choose the amount of damping. For high damping we chose $d_1 = E/100$, for the low $d_2 = E/1500$. The second strategy sets the damping proportional to the spring stiffness. We used $d_1 = k/3$ and $d_2 = k/15$. As said, the constants have been adopted manually, taking the ratio from the first strategy did not give acceptable results. Finally, the third strategy computes the damping parameters such that the spring damper pair has a given mechanical quality factor Q. The rationality behind this approach

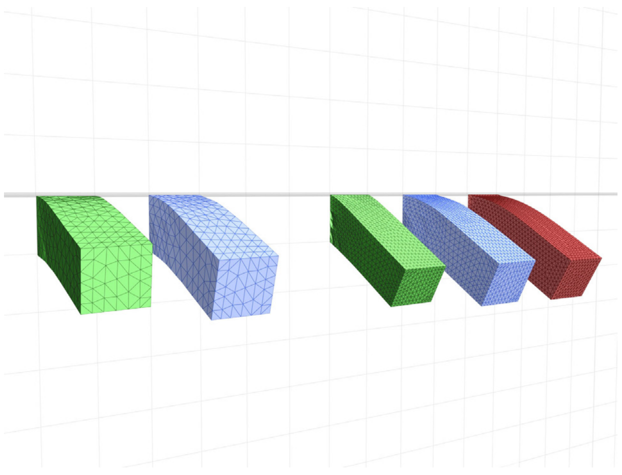


Figure 5.23: Relaxed state of the beam of example 4 using mass-spring systems (green) and finite elements (blue). The red bar again shows the ABAQUSsolution.

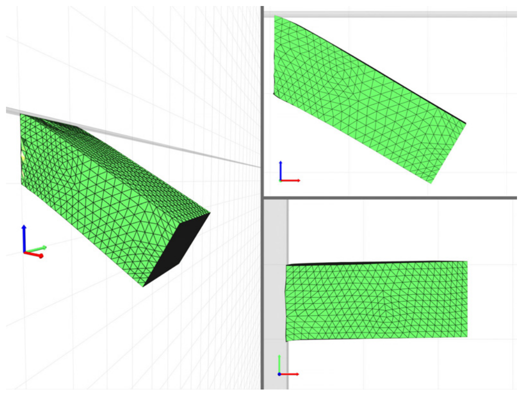
is, that for a given frequency all springs perform the same number of oscillations. The resulting formula is discussed in Appendix B. We selected $Q_1 = 0.1$ and $Q_2 = 0.25$. Note that a higher quality results in less damping.

Results and discussion: Neglecting the wrong rest state, all movies reveal wrong oscillations in the x-y-plane, independent of the strategy used for selecting the parameters. The St.Venant-Kirchhoff material does not show these oscillations. Their presence does not depend on the tesselation used, as an additional experiment shows. They are an artefact of the mass-spring-damper approach. Hence mass-spring-damper systems with a three dimensional topology should only be used with critical damping to disguise the wrong dynamics.

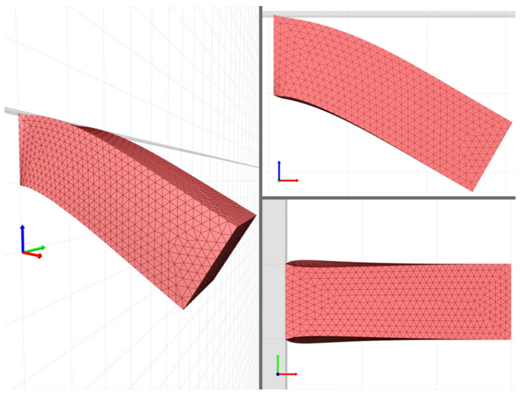
Besides the overall performance tuning, we did not optimise any of the animations for speed. In addition, the movies are created by grabbing the OpenGL frame-buffer and live encoding it into an avi-stream on the same machine, which consumes additional resources. Hence, the films do not allow an absolute judgement of the speed of the application. Nevertheless, comparing the relative speed shows that the mass-spring-damper meshes are faster by a factor of two to three than the finite-element simulation. Hence, the speed penalty of properly implemented finite elements is not as large as commonly believed.

Example 4 using Mass-Spring-Damper Meshes

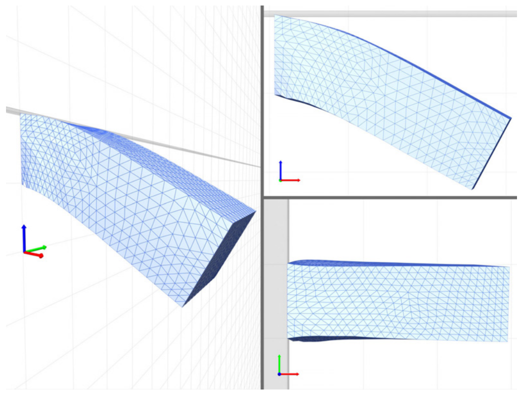
This last test with mass-spring networks repeats the cantilever benchmark that proved difficult for the finite-element discretisation, using again different mesh resolutions. As above we set the spring stiffness according to van-Gelder, with a Young modulus of $E = 8.997$kPa, computed from the parameters of the St.Venant Material. The damping constants, irrelevant for the rest state, are set for critical damping, such that no tetrahedral cell ‘inverts’ due to high inertia forces. The results, compared to the ABAQUS and VirTis finite-element simulation, are displayed in figure 5.23 and 5.24. For a coarse mesh, also the mass-spring mesh suffers from a reduced displacement. In addition, in neither resolution, it does develop the geometric features at the suspension caused by the concentrated stress and the high Poisson ratio. For the higher resolution, the springs at the suspension are unable to handle the load during the simulation and the first layer of nodes is squeezed into the suspended face. As a consequence, there are unappealing errors at the surface and the global form looses its curved shape to be replaced by an approximately



(a) Mass-Spring-Damper



(b) ABAQUS



(c) VirTis

Figure 5.24: Side-by-side comparison of a mass-spring and two finite element simulations, shown from different views.

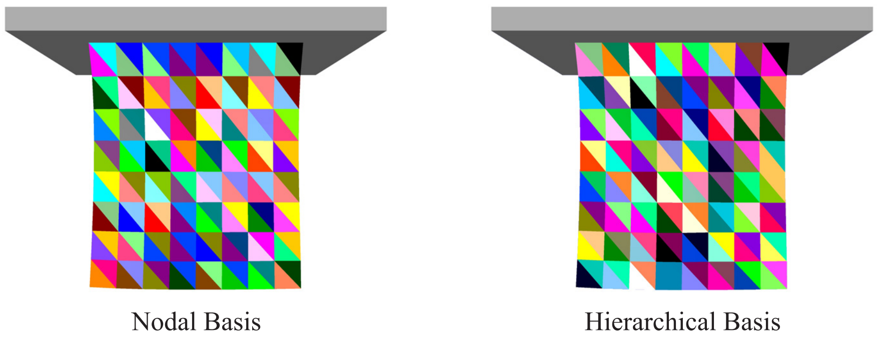


Figure 5.25: Side by side comparison of the same example using a nodal and a hierarchical basis. Visually hardly any difference is found.

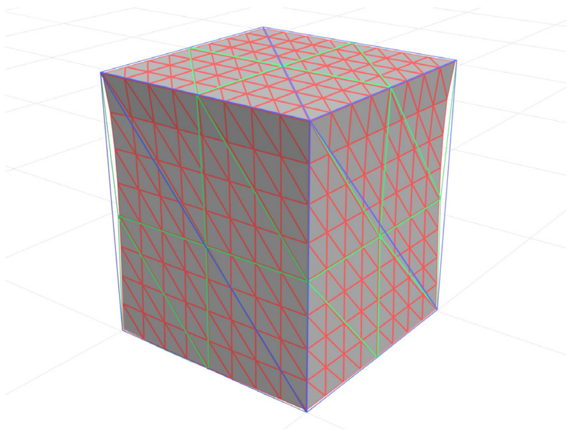


Figure 5.26: Convergence of the hierarchical basis against the solution. The grey solid cube gives the finest level; the overlaid wire frame shows the nodal basis result for comparison. The two coarse wire frames display the first and second level of the hierarchy.

straight line.

Discussion

The examples of this section showed, that mass-spring-damper networks are a useful method for unspecific soft objects, as long as no mapping of physical parameters is required. Their limitations are a strong influence of topology and the inability to model volume conservation. Adding additional non-spring forces helps for the last point, but then the border to finite element methods or the related mass-tensor models has already been crossed, and a pure approach often makes more sense.

5.3.4 Hierarchical Bases

We will experiment with the implementation of the hierarchical basis next. As a first experiment, we again consider the cube under a gravity load. The simulation is on the one hand performed using a hierarchical basis and on the other hand with a standard nodal basis defined on the finest mesh. We use the 6-tetrahedra cube as a base and add 3 levels of subdivision, yielding 3072 tetrahedra on the finest level. The relaxed state of both meshes is shown in figure 5.25. Visually, the states are barely distinguishable. As both bases span the same space, the simulations should give exactly the same result, which then indicates the correctness of the implementation. In practise, this is only partly true, because of the finite machine precision, and more important because of the very different range of the numerical values. Whereas for the nodal basis, all displacements are of the same magnitude, for the hierarchical basis, the scaling of the degrees of freedom is quite different by construction, which results in a different roundoff behaviour. Using once more the z-displacement of the lower left corner, the standard basis yields a displacement of -13.894mm, whereas the hierarchical basis gives -13.958mm, a difference of 0.064mm or about 0.5%. The geometrical convergence of the hierarchy is depicted in figure 5.26 showing different levels at once. The grey solid cube is the finest level; the red mesh gives the nodal simulation for comparison. The coarser wireframe meshes show the first and second level of the hierarchy.

Example 5: An Embedded Sphere

More interesting of course is not a full hierarchical simulation, but to use only a part of the hierarchy to add local detail at a region of interest, for example at the point of user-interaction. Detail coefficients can be computed where needed and left away elsewhere. For refinement and unrefinement they can be initialised to zero, representing the interpolation, and dropped when small. On a deformed solid, they can be frozen and the deformation stays locally the same, although at a coarser level the solid may continue to deform. To maintain integrity, it only has to be assured that all parent vertices of active vertices are present in the basis. A simple set of rules to assure consistent meshes has been presented by Krysl and Grinspun [KGS01, GKS02]. An advantage is that it suffices to use regular, red refinements [Bey97]. The resulting T-vertices or hanging nodes pose no problem for the simulation. At the hanging nodes, the deformation can be interpolated from the non-subdivided tetrahedra. On the surface, we always display the finest level, avoiding discontinuities. The drawback of not closing the mesh using green refinements at the T-vertices is, that the local deformations do not communicate a re-actio to the adjacent coarser level nodes.

A tetrahedron will be allowed to refine, if all its neighbouring tetrahedra have at least the same refinement level, i.e. none of its vertices is hanging. If any vertex is hanging, which prevents a refinement, we refine all its adjacent tetrahedra first thus adding detail to this region. Therefore during the next refinement step it could be refined if still necessary. A tetrahedron can be unrefined, if none of its neighbouring tetrahedra possesses a higher refinement level.

To decide, where to refine, we use a criterion similar to Debunne et al. [DDCB01] and decide upon the magnitude of the internal shear forces. A hysteresis, i.e. a higher refinement than a coarsening threshold counteracts unwished refine/unrefine cycles.

The local refinement is demonstrated in the experiment illustrated in figure 5.27, which shows a cube of 25x25x5cm^3 with an embedded region with different material properties. The 5.9 material inside the dark blue sphere has a nearly incompressible constant Q material, which will be discussed in section 5.3.6. The outside has a lower μ of 0.4kPa and a Poisson ratio of 0.3. In the wireframe animation, the tetrahedra of the inner sphere are blue. In the first still of the animation you can see the resulting slight bulge under gravity. We use a two-level hierarchical basis. The tetrahedra of the higher level are darker; the lower levels are activated when the pressure of the tool gives rise to high forces. In the second part of the animation, it is shown, how the material reacts differently. This is how a surgeon could feel an embedded tumour. The video is live captured from the application, in the lower left corner the ratio between real-time and simulation time is displayed, which is between 0.4 and 0.7.

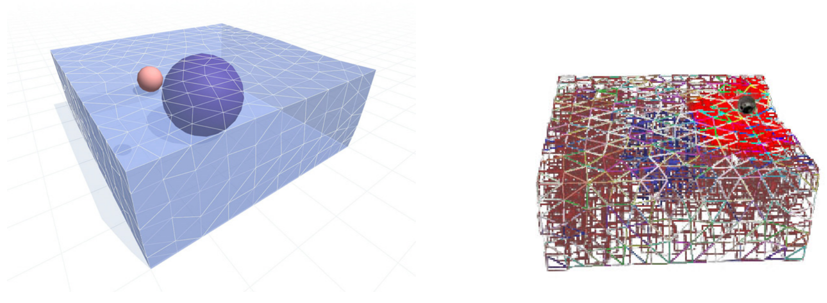


Figure 5.27: A parallelepiped with an inhomogeneous material. The left picture shows the setup, the right a simulation using a two-level basis. The bright elements are the second refinement level, activated near the point of interaction, where the shear stress is large.

Discussion

The use of a hierarchical basis proved to be especially useful for adding local detail to a simulation. The limits we encountered, and the reason for the shallow hierarchies we use, are not due to the method itself, but are induced by the subdivison scheme, which generates a very fast growing number of elements. The hierarchical bases themselves can be combined with other subdivision methods, a possibility we did not find the time to evaluate exhaustively. In addition, the method should be combined with multi-grid solvers or multi-level preconditioning, which we also did not consider in this work. As soon as other subdivision methods are evaluated, the point of attention will be shifted towards an efficient implementation of the level-of-detail data-structure, which we consider as an interesting point of future work, especially in combination with cutting into these meshes.

5.3.5 Corotated Strain

In this section, we will take a closer look at the corotated strain formulation proposed in sections 2.2 and 3.1.3.

Example 6: Forces on a Rotated and Sheared Cube

To compare the quality of the warping heuristics proposed by Müller et al. [MMD+02] (applied
5.10 per element) and the polar decomposition we perform the following experiment: Using Green's tensor for simulation we rotated a cube of 0.1m side length ($\mu = 1\text{kPa}, \nu = 0.45$) with $\omega = 0.63s^{-1}$ performing a single rotation in 10s. Then the cube is sheared with a constant force on the top surface. After coming to rest, it is again rotated. The resulting local coordinate systems are shown in figure 5.28(a) and animation 5.10. They pass the visual test of rotating globally with the rigid rotations and locally with the shear deformation.

For the next video, the internal stress forces from Green's (black), Cauchy's (red), corotation/polar (green) and warped tensors (orange/purple) were computed. The gold standard is
5.11 given by the quadratic Green strain, ideally all should match this model. During the first rotation with no deformation the phantom forces from Cauchy's tensor can be seen, which will be switch off for the rest of the video. All other forces are zero. During the shear movement, the corotated and warped forces split from the Green forces. This is due to the fact, that Green's tensor is a

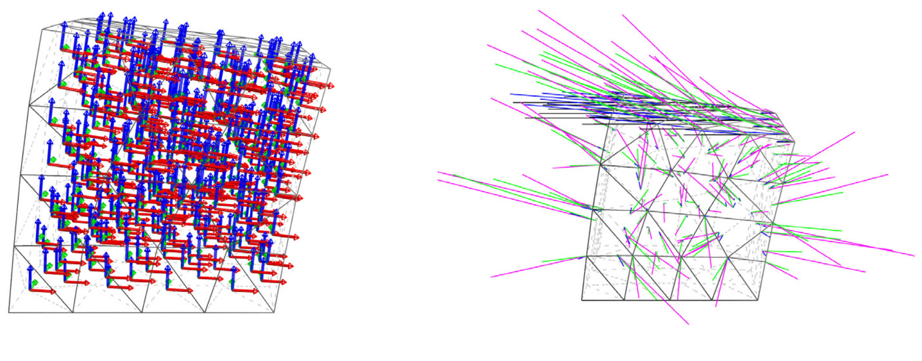
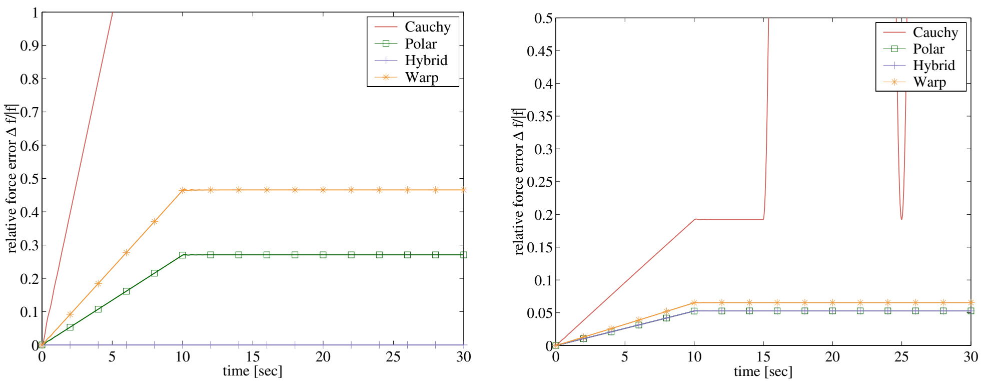

(a) (b)

Figure 5.28: Local coordinate systems generated by the polar decomposition (a), and the resulting force (b) using different strain measures: Green's tensor (black), corotation/polar (green) and warped tensors (purple).



(a) Pressure Forces. (b) Shear Forces.

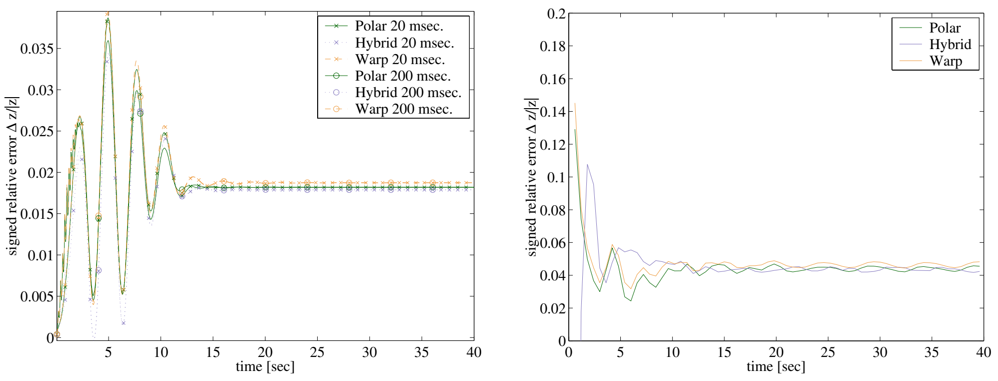
Figure 5.29: Example 6: Relative error with respect to the forces generated by using Green's tensor.

second order model of strain, whereas the other ones are linear. The polar decomposition roughly halves the error compared to warping. Of course, all are rotational invariant.

In figure 5.29 we plotted the relative error $\|f-f^G\|/\|f^G\|$ of the different force contributions, leaving away the first 10s. The picture confirms the above observations, and shows that the error is especially large in the pressure component. Therefore, we introduced a hybrid tensor shown in blue, combining Green's tensor for pressure and the corotated tensor for shear forces. This is very cheap in terms of force computations, but of course makes the system nonlinear again. For this tensor, the Jacobian is updated correctly including the nonlinear terms, whenever the rotation frame is updated.

Example 7: Shearing of a Long Bar

This example employs a bar similar to figure 2.5, of size 0.1x0.1x0.6m^3, with $\lambda = 100\text{kPa}, \mu = 10\text{kPa}$, $\mathrm{d}\lambda = 10\text{Pa}\cdot\text{s}, \mathrm{d}\mu = 1\text{Pa}\cdot\text{s}$ and a time step of 20ms with a Gauss solver. The bar is



(a) Shearing.

(b) Shearing while rotating.

Figure 5.30: Displacement error for Example 7.

	CPU time [s]	Newton	evals	LU dec.(Jacobi)
time step 20ms, update rate 20ms				
Green	16.5	6490	21276	105
Warped	27.4	-	2000	2000
Polar	31.4	-	2000	2000
Hybrid	33.8	3092	3092	2000
time step 20ms, update rate 200ms				
Green	16.5	6490	21276	105
Warped	5.5	-	2000	201
Polar	5.9	-	2000	201
Hybrid	22.5	9981	10538	422
time step 60ms, update rate 300ms, rotating				
Green	13.6	4489	16688	202
Warped	2.7	-	667	135
Polar	3.0	-	667	135
Hybrid	22.4	7142	21576	411

Table 5.4: Runtime Details for Example 7.

discretised in 337 tetrahedra. A shear force rising linearly from 0 to $200N/m^2$, is applied to the top surface. As an error measure, we chose the relative error in the z-component of the upper right front corner (compared to Green). The results (fig. 5.30(a)) are similar to those from above, identifying the hybrid formulation as best, warping as worst. This does not change, when the rotated frame is updated less than once per time step, e.g. only every 200ms. This causes Gauss- or preconditioner updates to be less frequent and makes the cost for computing the reference frame negligible (table 5.4).

To bring the method to its limit, we again performed this test, but imposing a rotation, choosing a time step of 60ms and updating the frame only every 300ms (about every 10 degrees of rotation, fig. 5.30(b)). This brings the methods very close together, showing again a favour for the tensors based on the polar decomposition. Now after reaching the rest position the bar still vibrates, resulting in an oscillating error. The hybrid tensor suffers from the bad frame, combined with an inaccurate Jacobian, due to the nonlinear pressure forces. Delaying updates more than 0.5s (or about 20 degree) makes the linearised methods unstable, whereas using Green's tensor the Newton iterations increase but the system remains stable. Without the rotation, the corota-

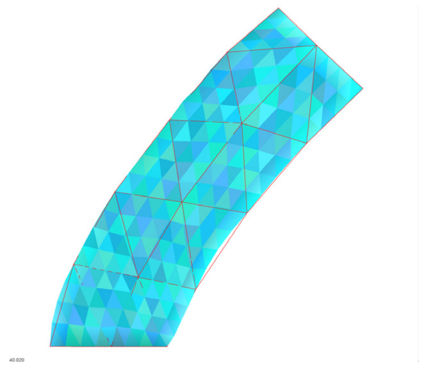


Figure 5.31: Example 8. Bar with 70/4480 Tetrahedra subjected to a shear deformation.

tional formulations stay stable for larger update rates and time steps, and with Green's tensor one can even use steps of 1s and above.

Example 8: Hierarchical Simulation

The next example addresses the use of corotational tensors in a hierarchical setting. For the coarsest level, Green's tensor is used, allowing a very stable simulation. From the configuration of the coarse level, the rotation state is computed by the polar decomposition of the deformation gradient, then propagated to the finer levels. If necessary, this could be iterated, recomputing rotations at several levels. Here we just do it for the top level. Thus at the coarse level, where linear algebra is cheap, we perform a full nonlinear simulation, at the finer levels the system is linearised and now solved by cg, adding detail for reduced computational costs. Nevertheless the computational costs of the finer levels dominate.

The bar (0.1x0.1x0.3m^3,$\mu = 1$kPa, $\nu = 0.3$) shown in figure 5.31 is discretised by 70 tetrahedra, two octasection steps give 4480, i.e. $8^2 \times 70$, fine elements. As before we add a rising top shear force of $80N/m^2$. We rotate the bar as shown in the video during the 10s simulation, with a time step of 30ms and an update rate of 150ms for the coordinate systems. The specular highlights in the video and the silhouette in fig. 5.31 show the smooth deformation gained by the fine discretisation. The coarse surface is given by the red wireframe. The computation time of 11.2s is almost real-time. It takes 6693 preconditioned cg steps, i.e. about 20 iterations per time steps to smooth out the error at the finer levels. For a comparison a complete nonlinear simulation was computed in 71.7s, the relative deformation error at the corner was 16%. This is larger than in the previous examples, because we are using the same frame for 64 tetrahedra. 5.12

Example 9: Deforming a Liver

The last experiment shows an application in virtual medicine. It employs a liver model (about 0.2x0.3x0.2cm^3, $\mu = 1$kPa, $\nu = 0.495$, roughly as measured in [GHEB01]), shown in figure 5.32 at the left. The opaque parts possess homogeneous Dirichlet boundary condition, i.e. are fixed. The initial tetrahedralization has 327 elements; we add a single level with 2616 tetrahedra. The liver is pushed in the region marked by the green arrows. Again, thanks to the fine level the surface stays very smooth, compared to the coarse initial disrcetization, which can be seen at the fixed parts and the silhouette. 5.13

For a simulation of 40s the application spent 25.8s, the time step was 30ms using BDF(3). The video discloses one small drawback of using a large number of coarse tetrahedra. About

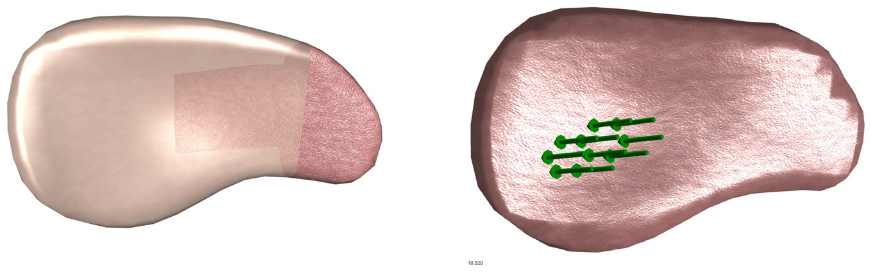


Figure 5.32: Example 9: Liver with 327/2616 tetrahedra on two hierarchy levels. We use a predefined ramp force, pushing in the region marked by the green arrows.

every second, the update of the rotation frames, which recomputes the linearised corotated Jacobian and the update of the coarse nonlinear Jacobian, including its LU-decomposition, coincide with each other, giving an unpleasant speed shift. This could be cured by just delaying one of the tasks, or by using a coarser top-level mesh, such that the decomposition is faster.

Discussion

The examples show, that the corotational approximation induces an error of about 5-20%, but can give a speedup of about a factor of seven or more, when applied in the hierarchical case. It is the most central tool for trading accuracy for speed. The use of the polar decomposition instead of the warping heuristics helps to decrease the introduced error. Using a hybrid tensor can further reduce the error, but computation times rise due to the reintroduced nonlinearity. Hence, we did not carry this concept further. The stability is enhanced when comparing to the admissible time steps published by Müller et al. [MMD+02], which were all around 10ms or below. We use step sizes of 20-60ms in the above examples and are stable for 100ms and beyond. Especially the hierarchical approach benefits from its unconditionally stable top level.

5.3.6 Viscoelastic Materials

This last section will perform a set of experiments to compare true visco-elasticity to a simple elastic model with damping added. By the first, we mean general (linear) visco-elasticity, modelled as a Prony series (section 2.6), here used for a constant Q material. By the second, further simply denoted as Hooke model, we mean a St.Venant-Kirchhoff material, with damping added by the use of the visco-elastic correspondence principle. Usually not even this is found in most systems, where the damping matrices are often lumped, sometimes called Rayleigh damping[5]. If the damping matrices are lumped, also rigid body modes will be damped, as the damping now simply is proportional to the nodal velocity, clearly an unwanted effect.

Example 10: Comparing CQ and Hooke

Our first example shows the advantages of using the constant Q material. It applies a shear force with a force density of 10 N/m^2 to the top surface of a cube with 0.1m edge length. The bottom of the cube is fixed to the ground. We apply the force for 4s and let the cube swing into its rest position afterwards (fig. 5.34). The red cube has a constant Q material, fitted with 3 memory

[5]Some authors also refer to the un-lumped model as Rayleigh damping

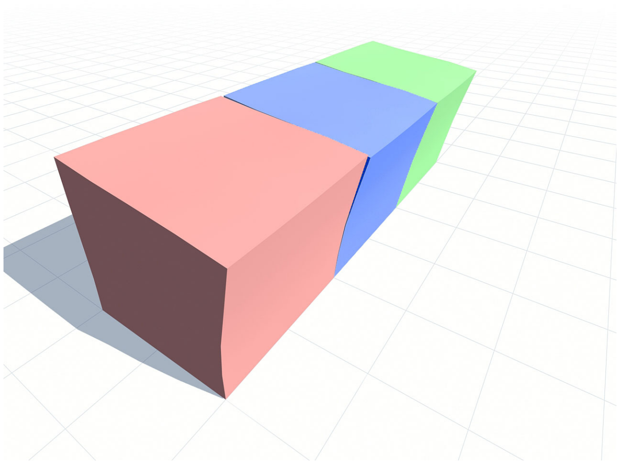


Figure 5.33: Experiment 10: 439 Tetrahedra with different material laws.

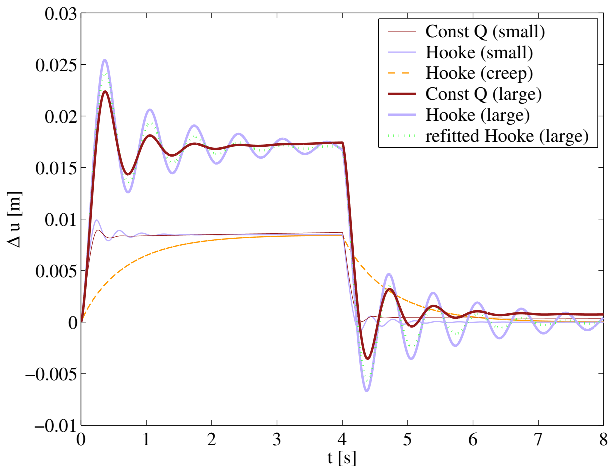


Figure 5.34: Example 10: Displacement of the upper right corner.

parameters, the blue one has a Hooke material. The constant Q material has the parameters measured for liver tissue $Q = 4.8$, and $\mu = 2.5kPa$ at 2Hz. As the liver is filled with blood, we choose a Poisson ratio of 0.495. In the nearly incompressible case, it makes no sense to use the constant Q model for the longitudinal waves, so we apply the constant Q material law only to the shear ϵ^{dev} part of Green's strain tensor ϵ. As a viscous damping constant for the compression part, we generally use $\lambda/100$.

The constants λ and μ of the first Hooke material (blue cube) are fitted, such that it possesses a Q of 4.8 at the resonance frequency of the cube, i.e. the intersection of the light green curve in figure 2.13 is located at the resonance frequency. This gives a similar initial slope. The arising displacements of the upper right front corner are plotted in figure 5.34. Note that it oscillates longer, and does not show the correct creep behaviour. 5.14

The orange cube was an attempt to show creep between $t = 2$ and $t = 4$. This results in an overdamped system. So we again note that it is not possible to fit a Hooke material law to

	ROCK		BDF(3)		
Material	time [s]	av. step	time[s]	#Newton	Jacobi/LU
small cube, 60s simulation time					
Hooke (matched)	57.9	$\approx$ 1.5ms	5.95	1833	4
constant Q	75.6	$\approx$ 1.5ms	7.27	1896	5
large cube, 60s simulation time					
Hooke (same)	42.7	$\approx$ 4ms	6.83	2197	4
Hooke (newly matched)	38.1	$\approx$ 5ms	-	-	-
constant Q	16.0	$\approx$ 7ms	4.14	1506	4

(a) 60s simulation with different solvers and material laws.

solver	time	av. step	stages
ROCK2	16.0s	$\approx$ 7ms	$\approx$4
DOPRI853	100.5s	$\approx$ 9ms	$\approx$8
DOPRI5	215.3	$\approx$ 3ms	5
Midpoint	56.9	1.5ms	2

(b) 60s simulation with different explicit solvers and a constantQ material.

Table 5.5: Runtimes for example 10, using different material laws and solvers.

the model. This results either in a crude creep approximation or in a strongly oscillating system, showing no creep at all.

In the second part, we rescaled the red and blue cubes with a factor of two, keeping all
5.15 material constants. Now the blue cube gives a different behaviour and oscillates much longer. This is due to its new resonance frequency. The new green cube is again a Hooke material with a model fitted to the new resonance frequency. Thus using Hooke's law, one has to tune the parameters to the given geometry.

For the videos, we choose a stress impulse of 30s to show creep more clearly. The experiment was originally designed for the publication at SCA [HGS03a], which promoted the use of ROCK. At that time, the implementation of the implicit methods had not been finished. To show the efficiency of ROCK, we compared this simulation using other explicit solvers. All run times are given in table 5.5. As predicted, the performance impact of the Prony model is about 15-20% of the total costs, i.e. about 5% per memory parameter.

After completing the implicit solver module, we re-ran the example using BDF(3) with a time step of 40ms, also shown in table 5.5. For the small cube, we have approximately the same increase of runtime by about 20% as for the explicit method when switching from Hooke to constant Q. These are the expenses for updating the memory parameters and computing the more complex stress-strain relationship. Note that the number of Newton iterations also grows by about 3%, but this is too small to account for this increase. Conversely, for the large cube the constant Q material permits a faster simulation, as already observed with the explicit methods. This is due to the fact, that the constant Q model does not give more oscillations for the larger model, as opposed to Hooke. Therefore the explicit integrators are able to take larger steps, the implicit ones need less nonlinear solves, as they are easily able to follow the slow creep phase. The overall simulation time for BDF(3) is so low due to this creep phase: during the 60s it needs only very few Jacobian updates and decompositions, which gives a clear edge over the explicit methods.

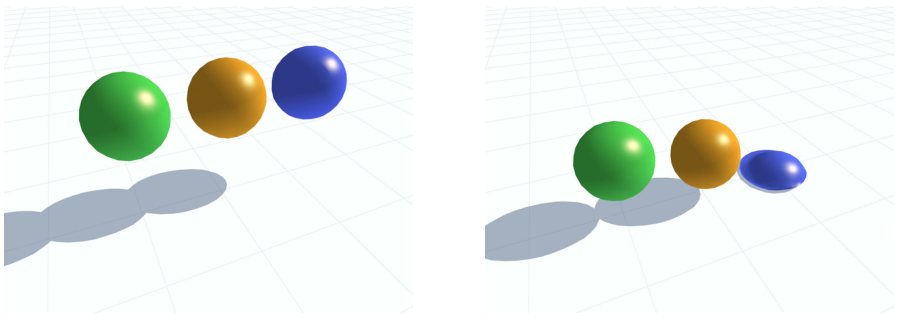


Figure 5.35: Three spheres with different materials. The green one is hard, the blue one soft. The orange one has a complex visco-elastic material, being hard for high and soft for low frequencies.

Example 11: Three Spheres

The effects of the constant Q model are subtle. To demonstrate the benefit of being able to shape a frequency dependent visco-elastic law at will, we set up another experiment, using three spheres with different materials as show in figure 5.35 and let them bounce on a virtual ground. We choose $\nu = 0.25$ to allow some material compression and employed the blue sphere with a soft Hooke material with $\mu = 2\text{kPa}$, the green one with a hard Hooke material using $\mu = 300\text{kPa}$. The orange one is equipped with a linear visco-elastic material law with the relaxation function

$$\hat{G}(t) = 2 \cdot 10^3\text{Pa} + 5 \cdot 10^4\, \mathrm{e}^{-\mathrm{t}/1.0\mathrm{s}}\, \text{Pa} + 3 \cdot 10^5\, \mathrm{e}^{-\mathrm{t}/6.0\mathrm{s}}\, \text{Pa}.$$

As a result, it behaves like the first material for high frequencies, and as the second one in the transient limit.

When we let the spheres drop, the blue one gets smashed due to its softness, while the other two rebound, as the impact consists of high frequency components. At soon as the green sphere comes to rest on the ground after some bounces, the orange sphere stops behaving alike and starts to break down. This is due to the now absent high frequency components, hence the material is soft. Waiting long enough, it will take the rest shape of the blue one. Note that due to the imperfect, discrete model and the collision response, this takes a very long time: each time a node hits the ground and is suddenly stopped, a new shock wave travels through the mesh and makes the material stiff again. 5.16

Example 12: Deforming a Liver

The last experiment again employs the liver model (about 20x30x20 cm), shown in figure 5.32 from example 9, this time equipped with Green's strain. As before the opaque parts possess homogeneous Dirichlet boundary condition, i.e. are fixed. 5.17

The first material is again the constant Q material from above. The second is the Hooke material using the constants of the large, green cube. The video shows the deformation by a predefined force and subsequent relaxation allowing a direct comparison. The counter in this video shows the simulation time. The Hooke material shows very different dynamics. Because not tool is present, the framerate was faster than real-time. 5.18

The second video shows the same setting, now with a user controlled tool. The counter now shows again the ratio between simulation and real time. We again employed ROCK for time integration. Due to collision detection, the framerates drop to 60-75% real-time. Later we will show an example of a liver with the same material model, then with about twice the number of tetrahedra simulated in real-time using BDF(3).

Discussion

This section presented one of the most important pieces for a soft tissue core. It is the first implementation of a real dynamic material in this field, incorporating a defined frequency behaviour that is capable of modelling organic materials more accurately. The additional costs for the higher fidelity are about 15%-25% (compared to the usual model), depending on the number of memory parameters necessary. The fidelity of the simulation builds on the reproduction of measured data by the fitting process described in section 2.7. For animation purposes, we do not deny, that a Hooke model, *manually* tuned by an artist to a special geometry, varying the stiffness and damping constants over time may also be able to mimic some of the effects. However, this does not hold in general, requires tedious manual tuning of a possibly large set of parameters for each simulation, and lacks physical reasoning. The advantage of the viscoelastic fitting is, that it can be adopted to a wide class of physical material models as described in section 2.6, not only to constant Q. In fact, linear visco-elastic materials are the broadest class offered by the common off-line finite element packages. For biological soft tissue, we prefer the restriction to constant Q, because measurements for two important classes of materials have shown, that the usage of a one-parameter viscoelastic description already gives good results.

5.4 Interaction

A simulator cannot be called interactive, until there is a way of manipulating the scene during simulation. Although we already showed examples including interactively controlled tools, up to now we did not describe the techniques used. Interaction is usually achieved by manipulating the boundary conditions of the deformable object, in order to prevent and to resolve interpenetration of the object with the static environment or user controlled tools. We therefore describe first how we handle static Dirichlet boundary conditions. Then we describe the basics of general collision handling. A central component of our collision handling toolbox is a constraint mechanism. Constraints need to be enforced by the numerical solver, usually within the linear solver. For this, a modification of the method of conjugate gradients has been proposed by Baraff, which we extend to Eisenstat's implementation. In addition, we propose a new formulation, which allows the combination of constraints and direct solvers for sparse systems.

The last part of this section describes the integration of the PHANToM, for coordinate input and haptic output. The standard method to accomplish this is to employ boundary conditions of traction, in this case referred to as virtual coupling, which we finally describe for the sake of completeness.

5.4.1 Static Displacement Constraints

One of the geometric restrictions that can be posed to a deformable object is, that a part of its boundary remains fixed in space, viz. homogeneous Dirichlet boundary conditions to the displacement field. The proper way to achieve this is to restrict the space from which the solution is taken, i.e. to admit only basis functions, which are zero on the restricted part of the boundary. This means pruning the basis for each new combination of boundary conditions, which deems inflexible for visual simulations. A common way to achieve the same effect is to take care, that the coefficient of the basis function is zero, which for finite elements means, that the displacement of the associated node is zero. For elasto-dynamics, this is fulfilled, if the initial displacement is zero, and all forces acting on the node vanish during the simulation. This concept immediately generalises to constraining only some coordinates of a node, e.g. for dragging something along rails, or posing a constraint along a general polygon in space. This boils down to restricting forces to a subspace. This task is usually solved by an orthogonal projection P_s onto the unconstrained space.

Hence the basic ODE (3.2.1) given by Newton's law of motion is changed to

$$M\mu'' = P_s f,$$

with a static projection matrix P_s. Using an implicit time integration scheme, possibly in combination with Newton's method and a reduction, now leads to a linear system of the form (see equation (3.4.2))

$$[M - P_s A]x = b, \tag{5.4.1}$$

where we abbreviated the linear combination of the Jacobian blocks by A. Unfortunately, this system ceases to be symmetric, a fact that can easily be seen when considering a P_s that fixes a single node, which cancels a row of A (the actio on the node) but not the associated column (the re-actio). The system still is invertible due to the mass matrix, but formally, the conjugate-gradient method cannot be applied legally and will at least have difficulties to converge. At this point, one could switch to the expensive GMRES, QMR or any other Lanczos method [Gre97, Saa96]. We choose to transform the system back to symmetry, multiplying by a decomposition of the identity matrix I from the right

$$[M - P_s A(P_s + I - P_s)]x = [M - P_s A P_s]x + P_s A(I - P_s)x = b.$$

Note that $I - P_s$ is the orthogonal projection to P_s, i.e. extracts just the components that remain fixed. For these components we know x, such that we simply move $P_s A(I - P_s)x$ to the right hand side and solve the now again symmetric system

$$[M - P_s A P_s]x = b - P_s A(I - P_s)x.$$

Although simple and elegant, we have not been able to find this published anywhere. Fixing a node, this correction matrix is just the column of A associated with that node (minus the diagonal block, cancelled by the left-multiplication with P_s), such that we are able to build $P_s A(I - P_s)$ and $P_s A P_s$ by deciding during the assembly of A if a node is fixed and writing the block to the respective matrix. This technique is applicable in combination with any linear solver, even multi-grid, is completely transparent, and only needs a way to multiply with $P_s A(I - P_s)$. It therefore does not disturb convergence. Its drawback is, that A is dissected into parts, which requires some memory shuffling, and - more severe for direct solvers - renders a factorisation with a different set of constraints useless. Thus, we primarily promote this trick for static boundary conditions.

5.4.2 Collisions

The handling of collisions is perhaps the most important point that distinguishes visual simulations from classical engineering approaches. Contact modelling is among the hardest problems in structural mechanics. Determining the exact point in time, when interpenetration occurs during a dynamic simulation, can consume considerably more time than computing a normal integration step. Therefore, simulation in graphics uses crude approximations, and detects collisions when they occur and tries to handle them heuristically, applying some corrections during the next time step.

Collision detection, possessing a quadratic run-time in the worst case, usually is sped up by using bounding volume hierarchies. For the proximity detection between general triangle meshes, we employ the k-dop hierarchy as described by Mezger [Mez01], for simpler objects like planes, we use directly available distance measures, and for point-mesh collisions, e.g. when using the Phantom, we omit the bounding volumes altogether and only compute point-triangle distances. As an output, pairs of close primitives are handed to a collision response algorithm.

Much has been written about collision handling, we have nothing to add to the basic techniques of this subject per se. However, as we propose to use several new numerical techniques, some effort is needed in order to combine them with common collision handling schemes. These fundamental schemes, which will briefly be described, have already been discussed by Platt and Barr [PB88], since then combinations and improvements of these key techniques allow to resolve even difficult problems in an acceptable way, recently demonstrated by Bridson [BFA02], to whom we refer for an overview over the different contributions.

Essentially, there are three sets of state variables that can be manipulated to resolve a collision: the displacement or configuration state, the velocity field and finally the force or energy functional. The most drastic correction is done when directly manipulating the displacement field, as this all of a sudden may distort some of the elements, artificially adding strain energy to the system. The same, with a lesser extent, holds for changing the velocities, usually made by applying impulse corrections. The most graceful way is to manipulate the force field. However, this can either not resolve very fast collisions or requires excessively large forces, that again threaten the physical stability of the system. So usually, a hybrid strategy is employed, that is as smooth as possible but proceeds by all means necessary to resolve the collisions. The construction of a method normally is governed by the goal to conserve as many physical quantities as possible, like energy, impulse, and moment, in order to guarantee the *physical* stability of the system, and to do this in a way that does not jeopardise the *numerical* stability of the actual implementation. When a node encounters multiple simultaneous collisions, usually the heuristics are at their limits, and one either proceeds with fingers crossed, or monitors the conservation quantities and employs corrections to assure the physical stability.

Once a collision has occurred, the objects must be stopped to interpenetrate each other further, that is the simulation poses constraints on the movements of the nodes. When the collision is detected exactly at the moment it happens, constraints suffice to handle the collision, as no interpenetration takes place. Of course this requires some event location mechanism and rollback in time, but is the most correct way to handle the problem (cf. [HNW93, pp.195-200]). The sheer amount of collisions prohibits this approach for visual simulation. Therefore, the collision is corrected in one of the ways described above, then constraints are applied[6]. This can be done using the penalty method, (augmented) Lagrangian multipliers, or reaction constraints [PB88].

The penalty method proceeds by adding corrective force or energy terms to the equation, that pull the nodes towards the constraint manyfold like a strong rubber band. Its drawback is, that it does not enforce constraints exactly, besides the selection of the penalty function or constant is a free parameter. A function that grows too strong adds unnecessary stiffness and may overshoot, a function that grows too weak will not enforce the constraint accurately enough.

Lagrangian multipliers, the dominant technique used in numerical analysis and engineering [ZT00], couple the constraint equations through a set of additional variables, the multipliers, and add additional equations for this variables resulting from the constraint conditions. Besides the additional computational burden caused by these equations, this turns the minimisation problem to a saddle point problem, resulting in the system matrix loosing positive definiteness, which again is a major obstacle for using a conjugate gradient method as a linear solver. The advantage of Lagrangian multipliers is that the constraints will be enforced exactly, and the initial system

[6]Actually a constraint restricts the degrees of freedom of the system, which means that the number of coordinates is reduced. As above for the Dirichlet boundary conditions, an obvious method would be to reduce the number of coordinates assigned to a constrained node dynamically, but this would complicate the system dramatically.

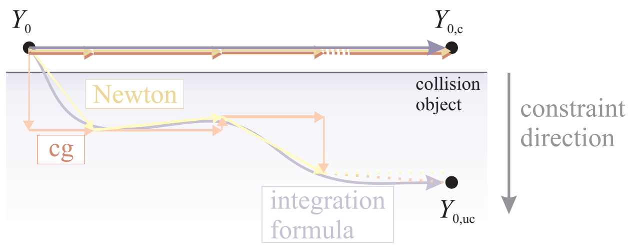


Figure 5.36: Constraint directions are enforced inside the linear solver. As a result, the constraint is also fulfilled by the Newton method and the integration formula.

state does not need to fulfil the constraints, viz. the technique is also able to resolve interpenetrations.

The most common technique in graphics to enforce constraints are reaction constraints, or some similar variant (e.g. by Baraff [BW98]) of the projection method. The idea is simple, and more or less the same as described above for the static boundary conditions. If forces and velocities are restricted to a space orthogonal to the constraints, the node will not move into the forbidden directions during a time step. Of course, this requires some procedure for an initial resolution of an interpenetration in the direction orthogonal to the constraint. Reaction constraints in their pure form use critically damped spring forces, but also point relocation or Baraff's position alteration (working with velocities) is feasible. The advantage of this projection technique is, that it solves for unconstrained dynamics as before, whereas the constraint parts are under user control. Note that the difference to the static constraints is, that every constraint may change at every time step, such that the above-described method has an unacceptable high overhead. The system to be solved is the same

$$[M - P_d A]x = b, \tag{5.4.2}$$

only P_d now changes frequently.

Modified cg

A central point of the contribution of Baraff and Witkin [BW98] was the construction of a modified conjugate gradient method, recently analysed by Ascher and Boxerman [AB03], that allows an efficient treatment of dynamically changing orthogonal projection matrices P_d.

First we note that the constrained problem (5.4.2) is equivalent to the equations

$$\begin{aligned} P_d[M - A]x &= P_d b \\ (I - P_d)x &= (I - P_d)b, \end{aligned}$$

where the first equation is the one that needs to be solved, unfortunately now with a singular matrix $P_d[M - A]$. The elements of x in the null-space of this matrix are defined by the second equation. Using that the cg-method is an iterative method, adding scalar multiples of a search direction to an initial guess, the modified cg method simply filters the search direction by P_d. As a result, no corrections violating the constraints are added, hence if the initial guess fulfils the constraint, the final solution also does. In addition, the residual update is also projected, such that the filtering does not disturb convergence. Indeed, Ascher proved, that the iterates generated by this approach coincide with those of the system projected to the unconstrained subspace. The overhead compared to the normal cg method are two applications of the filter per iteration, which typically for each constraint consists of a multiplication of a three-by-three matrix with the corresponding vector blocks and therefore is considerably cheap.

Embedding this method into an implicit numerical time integration method results in an overall behaviour as illustrated in figure 5.36. As the result of the linear solver satisfying the constraint, the Newton update direction also does. Hence, the step computed by the integration formula also meets the constraint. Baraff only used a semi-implicit or linearised Euler method, but adding an outer Newton iteration does not change the effect.

Modifying Eisenstat's implementation

Eisenstat's implementation [Eis81] of the cg method is based on restating the system (5.4.2) as

$$\big\{(D+L)^{-1}P_d[M-A](D+L)^{-T}\big\}\big\{(D+L)^T x\big\} = (D+L)^{-1}b,$$

where $M-A = L+D+L^T$, and solving it with a diagonal preconditioner D, which is equivalent to solving the original system with an SSOR-preconditioner. The iterates are correlated by

$$\hat{x} = (D+L)^T x \quad \text{and} \quad \hat{r} = (D+L)^{-1} r,$$

and the same relation holds for the search and update directions.

In order to construct a modified Eisenstat iteration, we exploit, that the projection matrices P_d are assembled of individual constraint directions n_i

$$P_d = I - \sum n_i n_i^T.$$

Thus, if we filter the update p in the modified cg method by P_d, this corresponds to

$$\hat{p} = (D+L)^T P_d p = (D+L)^T P_d (D+L)^{-T}\hat{p} = \Big[I - \sum \big\{(D+L)^T n_i\big\}\big\{(D+L)^{-1} n_i\big\}^T\Big]\hat{p},$$

and the new filter directions are given by the expressions in the curly brackets. A similar argument holds for the update of the residuals.

Of course, the projection matrix is never assembled explicitly, but the multiplication is performed using dot products and vector additions. Hence, whenever a constraint is posed in the standard basis, we transform it to the coordinates of the Eisenstat iteration and are able to proceed as before. A small handicap is, that n_i usually has only three entries, that correspond to the contrained direction in 3D, whereas $(D+L)^T n_i$ may be less sparse due to the transformation. As a result, the dot product is slightly more expensive to compute than before. Compared to a matrix-vector multiplication, which dominates the workload of a cg-iteration and one of whom is saved by Eisenstat's implementation, it is still considerably cheap.

Additive Decomposition and Constraints for Direct Solvers

As already mentioned the workload of direct solvers is divided in a very unsymmetric way between factorisation and solve (e.g. consult table 3.5). Therefore, we want to avoid a new factorisation, whenever a constraint changes, which usually happens every time step. Ideally, we looked for some sort of pre- and postprocessing using the projection and one of the two equivalent formulations of the problem, involving both the vectors b and x, and the factors L and U. After some fruitless experiments however, we rejected this idea and looked for a new formulation.

The key for an efficient algorithm to tackle this problem is to switch from a *multiplicative* formulation to an *additive* representation for the perturbation. Consequently, we write the full order system as

$$[M - P_d A] = [M-A] + A - P_d A = \big\{[M-A]\big\} + \big\{(I-P_d)A\big\}.$$

The first system is just the unconstrained one, the second one is composed of projections of the rows corresponding to constrained nodes, thus it has very few entries. Inserting the definition of the projections, it holds

$$(I - P_d)A = \sum n_i n_i^T A = \sum u_i v_i^T,$$

if we define

$$u_i := n_i \quad \text{and} \quad v_i := A^T n_i.$$

To permit a more compact notation, we define the matrices U and V to be assembled from the columns u_i and v_i. Note that v_i is efficiently computable in constant time, as it is a linear combination of three columns of A^T, where the number of non-zero elements per column is proportional to the number of neighbours of the constrained node. Hence, for m constraints, we have an additive perturbation of rank m of $M - A$.

This is exactly the problem solved by the multi-dimensional Sherman-Morrison-Woodbury formula[7] [Hag89], which states

$$\begin{aligned}(B + UV^T)^{-1} &= B^{-1} - B^{-1}UD_m^{-1}V^T B^{-1}, \quad \text{with} \\ D_m &:= I_m + V^T B^{-1} U,\end{aligned}$$

where we added the index m to accentuate matrices in $\mathbb{R}^{m\times m}$. So in order to solve a constrained system, we proceed in two phases: after the constraints are defined, we compute the auxiliary vectors u_i, v_i, $\hat{u}_i$ with

$$\hat{u}_i = B^{-1}u_i.$$

From these the capacitance matrix D_m can be calculated by m^2 dot products. An inversion of D_m is cheap due to the low dimension. The computation of $\hat{u}_i$ needs to perform m solves with $B = M - A$, which heavily relies on the fact, that these are inexpensive due to an existing factorisation. Note that $\hat{u}_i$ is also sparse due to reordering, although possibly less sparse than u_i and v_i. Whenever a system needs to be solved, the second phase calculates

$$\begin{aligned}\hat{x} &= B^{-1}b \\ x &= \hat{x} - \hat{U}\big(D_m^{-1}(V^T\hat{x})\big). \qquad (5.4.3)\end{aligned}$$

If the second line of (5.4.3) is evaluated as hinted by the brackets, it consists of m sparse dot-products, a matrix-vector multiplication in $\mathbb{R}^{m\times m}$ and m sparse vector additions. Hence the overhead over an unconstrained solve in the second phase is at most $O(mN)$, and typically due to sparsity $O(m)$ on average, where m is the number of constraints. Thus, we precisely achieved what we initially set out to find: an inexpensive post-processing of an unconstrained solution vector.

Examples

To demonstrate the feasibility of the constraint enforcement mechanisms, we implemented a prototype of the above-sketched algorithms. Due to time restrictions, it lacks several points. First, it does not exploit sparse vector operations, which would be important for speeding up the direct solver. More severe, no time coherence is exploited, that is, in every step we perform a collision detection and pose new constraints, even if they coincide with the old ones, which means for the direct solver the expensive phase one is processed every step for every constraint.

[7] More correctly the Woodbury formula, although it appeared in several papers before being published by Woodbury for this task (cf. [Hag89]).

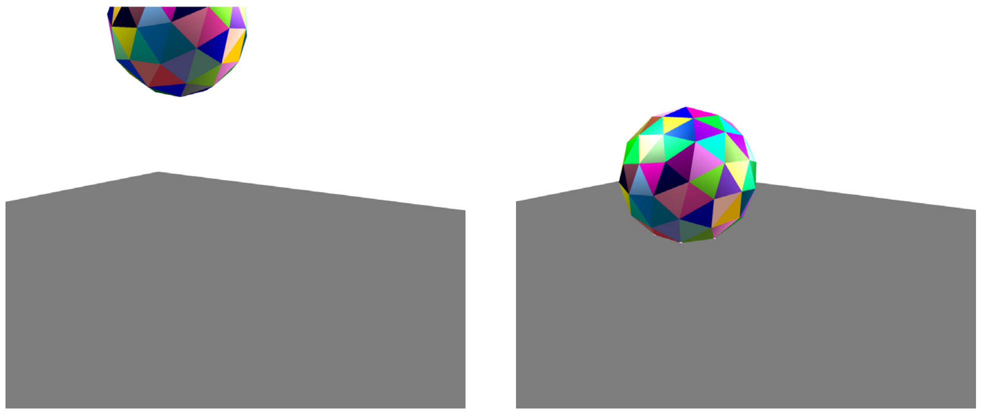


Figure 5.37: Sphere hitting a table.

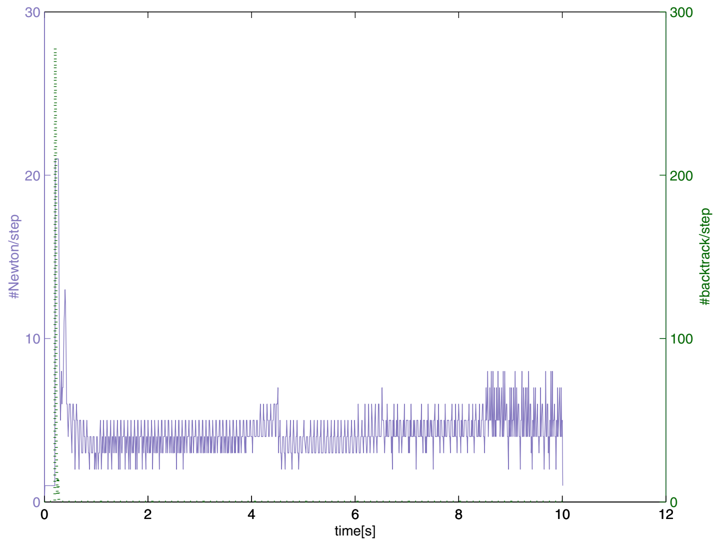


Figure 5.38: Double axed plot of Newton and backtracking iterations per time step. At the moment of impact, both increase, ensuring stability.

Also the collision response has its shortcomings in several ways, and is by far not as elaborate as for example the one of Bridson. Because it's simple to implement, we use direct point relocation for the resolution of interpenetrations. This has the disadvantage that it threatens physical stability, as it continually injects energy into the system that needs to be consumed by the viscous component. It also is a hard stress test for the stability of the numerical solver, as it has to compensate and relax suddenly seriously distorted element. On the other hand, this demonstrates the robustness of the solver.

5.19 The first example is a sphere of 0.2m diameter shown in figure 5.37, that is dropped from a height of 0.2m and bounces for 10s. It hits the ground after a falling time of 0.2s with a speed of 2m/s, using a gravity constant of $g = 10\text{m/s}^2$. With a time step of 0.01s, this means for the first collision response step to move the nodes by 2cm. The solver is able to cope with this without loosing stability. This is made possible by the global convergent Newton method, which detects

sphere, small mesh (172)						
	#newton	av. time/newton[ms]	#Jacobian	#backsubst	#cgiter	#backtrack
mod. cgeis	1788	3.4	33	-	62346	842
direct+smw	2409	3.0	35	11235	-	781

sphere, larger mesh (275)						
	#newton	av. time/newton[ms]	#Jacobian	#backsubst	#cgiter	#backtrack
mod. cgeis	5852	4.5	57	-	127031	819
direct+smw	6563	7.7	135	37512	-	896

Table 5.6: Details for the collision of a sphere and the ground.

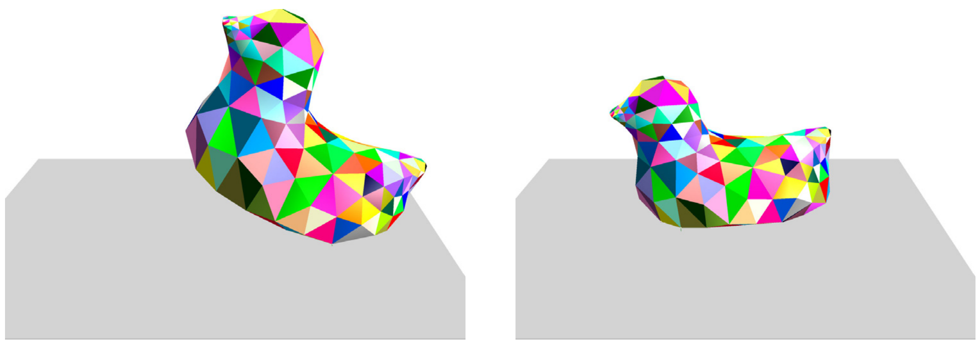


Figure 5.39: Toy duck hitting a table.

the difficult situation and uses backtracking to maintain convergence, as shown in figure 5.38. The price is an increase in Newton iterations per step at the moment of impact, which vanish as soon as the discontinuity has passed.

In table 5.6, we give some detail about the performance of the algorithms, when using two different tetrahedralisations with 172 and 275 tetrahedra. As expected, for the smaller mesh, the direct solver takes the lead. The implementation of the direct solver without time coherence requires a high number of backsubstitution solves. The average Newton time for the direct solver includes this cost of phase one in each integration step, averaged over the number of Newton iterations. Still, it is faster than the modified Eisenstat method. The remaining characteristics are alike, which shows again, that the choice of the solver does not influence stability or the characteristics of the outer time integration.

As the dimension of the system increases, the iterative solver gains advantages, similar to the unconstrained case, although the break-even point comes earlier in this case. We conjecture that this is mainly due to the current implementation, which is clearly biased. In this case, all constraints have the same direction. Moreover they are applied to a set of about a dozen points. Exploiting time coherence will supposedly bring down the number of backsubstitutions below a hundred, and thus restore the lead of the direct solver up to a higher dimension.

The second example is a proof of concept for the more complex geometry of a duck, shown
in figure 5.39. We also let it drop from a height of 0.2m. It is discretised into 511 tetrahedra 5.20
of various sizes, which makes the numerical simulation more difficult due to a wide spread
spectrum of eigen-frequencies. Because we actually do not know much about the material of a
toy duck, we once used a moderately compressible St.Venant-Kirchhoff material with $\nu = 0.25$ 5.21
and $\mu = 1$kPa. As there are some sorts of hollow rubber ducks, we also employed a mass-
spring mesh with practically no force preventing a change of volume, using a Young modulus
of $E = 15$kPa. This is possible, because the collision handling concept and its implementation 5.22

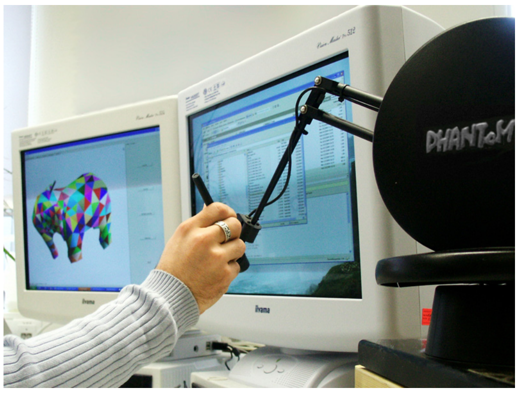


Figure 5.40: Using the phantom to manipulate a deformable model.

is completely orthogonal to the physical modelling. The two movies again demonstrate that the resulting materials lead to a very different outcome of the simulation. Due to no resistance to volume changes and the added rotational moment, the front of the mass-spring duck is squeezed in at the impact and the head nods forward, resulting in an unbalanced, translational rigid body mode. As we model no friction or damping of rigid modes, this lets the duck slowly slide of the scene.

5.4.3 Haptics

The central application, a soft tissue module to be included in a virtual surgery simulation, requires some way of direct user interaction, which we realised by the use of a Sensable PHANToM 1.5 Premium (fig. 5.40). This version of the PHANToM allows a six degree of freedom user input, i.e. position and orientation, but only three degrees of output. From an input point of view, it therefore represents a rigid body; the appropriate paradigm for the output though is merely a point.

A peculiarity of current haptic hardware is the fact, that it requires a very high force input rate beyond 1kHz from the application. It is a common misperception that this frequency-limit is due to human haptic perception being especially sensitive to high frequencies. It is true that haptic perception reaches several hundred Hertz, but the high servo rates are motivated by their relation to the range of impedance or ‘z-width’, the device is able to display [CB94]. Higher servo rates, even beyond 1kHz, are needed in the case of stiffer objects, while lower rates suffice for softer objects. The common devices implement their main servo loop on the CPU, thus depend on the high input rate. On the other hand, the time integrator, that generates the soft object’s response, is more effective for a low rate of less than a hundred Hertz. We solve this rate problem by computing the haptic response in a separate thread. Because the haptic response needs to correspond to the current state of the object, which is only known *after* the time integrator has completed a step, some sort of extrapolation is needed. Picinbono et al. [PLDA02] solve this problem by a *spatial* extrapolation of the finite element forces along the line that connects the PHANToM positions of the two last time steps. We found it better to use a local representation of the deformable object that can be computed at the high servo rate and is updated from the slower simulation. This concept is generally called a local buffer model [Bal99]. At the same time, the local buffer concept defines the way the user interacts with the simulation by boundary conditions of traction, which solves another issue described next.

In principle, the techniques used for collision detection from above could be used for em-

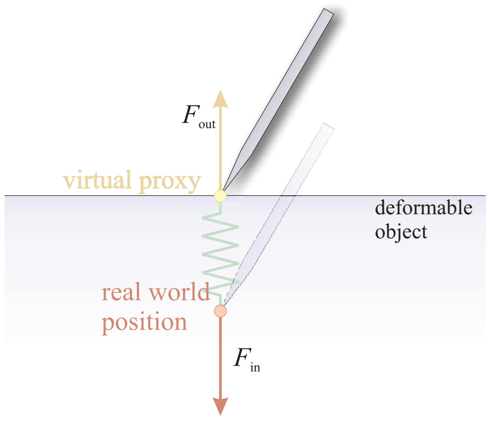


Figure 5.41: Using a virtual proxy at the surface, force input and output can be modelled using a stiff spring.

bedding an interactively controlled rigid object into the simulation and displaying the reaction forces from the collision to the user. This was the first plan of attack, which we used to deal with the problem. There is a severe problem with this approach. Caused by the stiffness of the differential equation, the reaction forces at the very first instant can be high, if we simply force the deformable tissue to the outside of the tool by geometrical corrections. This is an artefact of the numerical solution, and the fact, that we sometimes are unable to relax the system totally. These forces should not be displayed to the user, so we experimented with several strategies, including low pass filters and intermediate force buffers.

Again, another approach proved to be more satisfactory. It embeds the user-controlled instrument using boundary conditions of traction, rather than using displacement boundary conditions. From a modelling point of view, this is more natural, as the user exerts a force, rather than moving the surface of a deformable object to a specific place. Only as a result, the nodes then take specific positions in space. Unfortunately, this is exactly the opposite way common haptic devices are constructed, which do not include a force sensor. Hence, a method is needed to translate positions from the haptic device to forces.

For this purpose, we employ a virtual proxy or god object as the local buffer model, which is also used for generating force feedback when interacting with rigid objects, and credited to Ruspini et al. [RKK97] and Zilles and Salisbury [ZS95]. The concept lets the virtual counterpart of the haptic device penetrate the virtual object (figure 5.41). It introduces the closest surface point of the deformable objects surface, called a proxy. The force rendered to the user is then given as a spring force, i.e. proportional to the penetration depth. In the above-cited early work the proxy is computed with respect to the rest position of the object, as a result the force output to the user does not reflect any material properties and is determined by the spring constant. We update the position of the proxy at every step of the time integrator, performing a proximity search. In addition we take the same spring to compute a force that is applied to the mesh, more precisely to the surface triangle that contains the proxy. As a consequence of the actio-reactio principle, if the mesh is in equilibrium, the force that is sent to the haptic device has the same length but the opposite direction of the one, that is applied to the simulation, and thus is the same as the finite element response. The error that is caused by this indirect formulation is rather located in the position than in the force, as the haptic real world position must be inside the virtual object. For visualisation, the virtual representation of the PHANToM is positioned at the proxy point, so no penetration is visible. Hence, as soon as a deformable object is touched, the

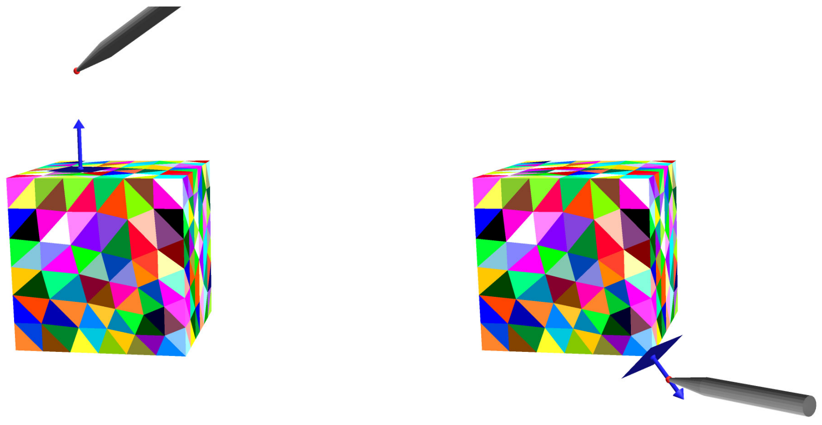


Figure 5.42: A collision proxy plane separates the haptic point from the deformable object. For smooth regions (left) plane normal and surface normal coincide, for sharp features, the closest distance vector used as plane normal shows its benefit.

real world and the position of the graphical feedback are different, but the user is usually not sensitive to this error. In fact, the error is proportional to the stiffness of the spring, for infinite stiffness there is no error. Commonly the stiffness is fitted to the limited z-width of the haptic device, which cannot display infinite stiffness. It may not be set to arbitrary high values, because this may introduce an instability in the haptic loop (cf. [SCB04]). Hence, in our application we use 50-500N/m.

To represent the force in the finite element setting, we use a linear hat density-function over the triangle, the peak is located at the proxy, the height is chosen, such that the density integrated over the triangle gives the input-spring force. The force is distributed to the nodes by taking the scalar product of this force density with the corresponding basis function (cf. equation (3.1.2)). This calculation can be performed symbolically; as a result, the input force is distributed weighted by the barycentric coordinates of the proxy.

For slow movements and equilibrium states, this approach gives a smooth force-output. However, for fast movements establishing or loosing contact, we were not content with the quality, as the contact feels sticky or squashy. This is due to the fact, that collision detection and updates of the proxy are only performed with the rate of the simulation, that is, a contact may be established with a latency of possibly 10ms or more. To cure this flaw, we enhanced the local buffer model by a substitute for the local geometry. For this purpose, we use a plane, given by the virtual proxy and the direction from the proxy to the haptic position (figure 5.42). Other authors use the surface normal of the closest triangle for orienting the plane, but this gives strange results at sharp features as edges and corners. There the normal vector may jump between two triangles at either side, which gives oscillating inside/outside answers and ghost forces. For smooth sections, both choices coincide. In each servo loop, we perform a collision detection with this plane. If the haptic position is outside, no force is sent to the device, if inside, the force is rendered as described above. To improve the approximation further, we move the plane by the speed of the triangle, which contains the proxy. As a result, we have a crisp contact and release force. An example force protocol is shown in figure 5.43.

The algorithm works up to a simulation time step of 10-20ms. An increase to 40ms, which is the visually acceptable limit, would be desirable, but for this large step, the haptics show perceptible vibrations. A practical solution would be to smooth the haptic output with a low pass filter. This induces a time lag of several milliseconds, which is the reason we gave up this

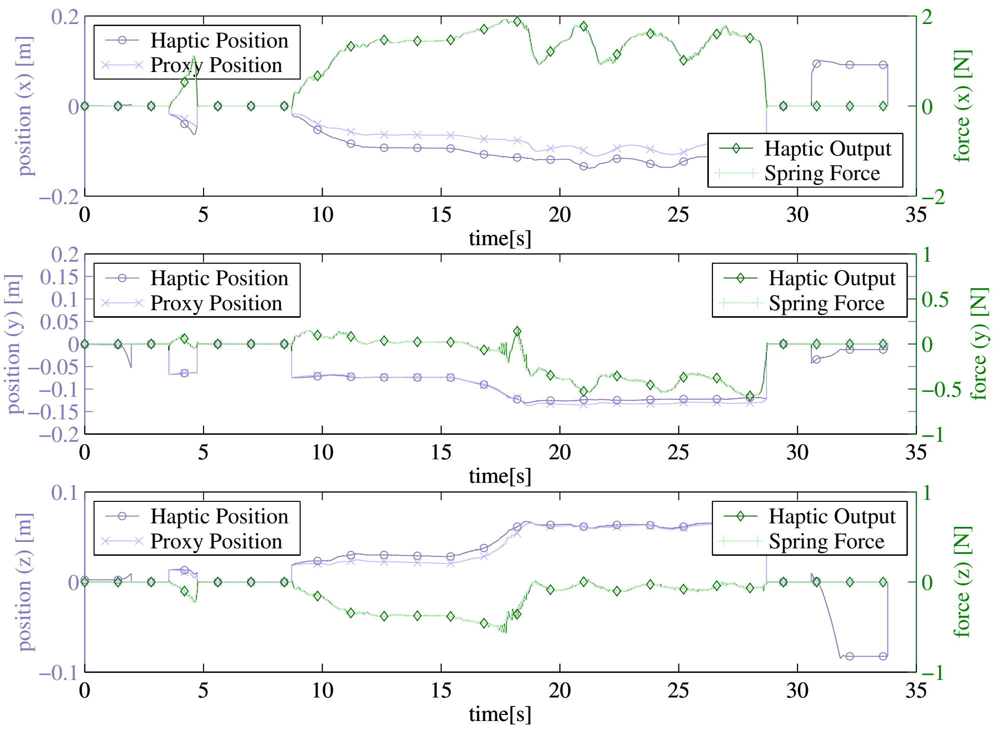


Figure 5.43: Double axed plot of positions and forces when manipulating a liver model. Forces and positions are only recorded when the stylus is in vicinity of the object. The liver is first poked twice slightly, then heavily pushed and held, where it shows some creep. Afterwards its pushed cyclically, suddenly released, and finally the stylus is moved around the object without touching it directly. Forces are generated as soon as proxy and haptic position cease to coincide.

enhancement.

5.4.4 Interactive Examples

This last example section demonstrates the interactive features of the simulator. The different animations show the interactive manipulation of the duck model and the objects depicted in figure 5.44, using several of the discretisation techniques and material laws we proposed.

The smallest example from figure 5.44 again uses the liver model, this time discretised with 676 tetrahedra. For all the examples in this section BDF(3) is used for time integration, for this mesh with a time step of 15ms. This is the example used for recording the forces displayed in figure 5.43. The corresponding video shows, that with or without active haptics, with or without a Prony series material, the simulation achieves real-time frame rates with nonlinear strain. As soon as the haptic output is activated, the frame rate drops considerably. In either case, the haptic servoing is active at 1kHz in a separate thread, polling the position of the PHANToM, even the virtual proxy forces are calculated in both cases inside the servo thread. The only difference is, that with output activated, the force output routine of the GHOST SDK is called. As the system time increases from below 10% to 50% and beyond, we conjecture, that the PHANToMdevice driver routine is responsible for the high load. SensAble Technologies recommends the use of multi-processor workstations.

5.23
5.24
5.25
5.26
5.27

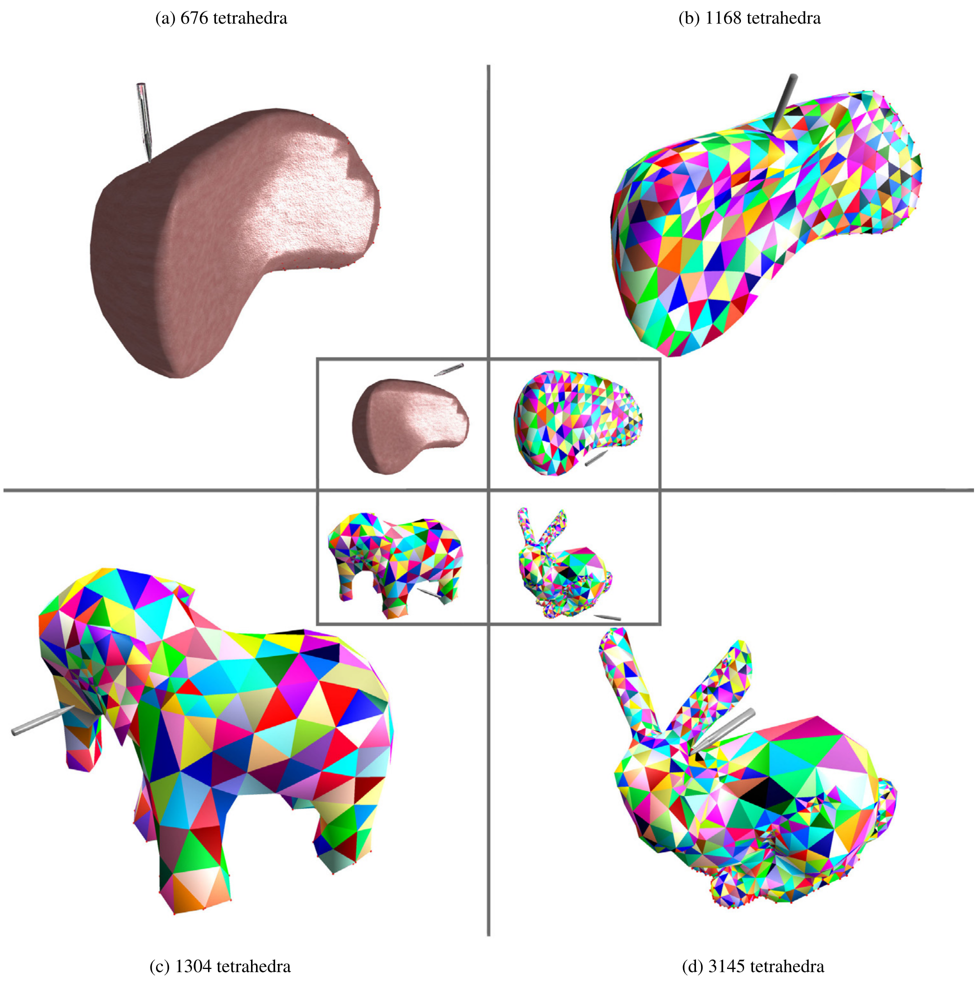


Figure 5.44: Screenshots from the interactive manipulation of several objects. The insets display the rest states of the models.

The next model, the liver with 1168 tetrahedra, is located at the limit that we are able to simulate with a nonlinear strain model at 25Hz, i.e. a time step of 40ms. We therefore once use a co-rotated strain with a 20ms time step.

The elephant model has 1304 tetrahedra and still can comfortably simulated with a time step of 20ms using a corotated strain with an update of the local coordinate systems every 200ms. In contrast to the elephant, which permitted a nice tetrahedral approximation, the original Stanford bunny model, after eliminating some intersecting and non manyfold triangles, is not easily approximated with tetrahedra. It passes the mark of 3000 tetrahedra, and the resulting discretisation is very bad in terms of grading and other quality measures. Nevertheless, a stable interactive simulation is possible, now using a time step of 40ms. We again used a Cauchy strain model for comparison, which permits a faster simulation with a time step of 20ms, but is not suited for the large strain we produce. Hence the artificial volume inflation effect, also reported by Müller [MG04] can be seen, especially at the ears of the model. Using the co-rotated strain, this phenomenon vanishes.

Discussion

The final application section finally assembled the numerical core components to an interactive visual simulator. We extended the direct and iterative linear solvers to be compatible with common collision techniques and showed the feasibility of the approach. Haptic in- and output was described, built around the paradigm of a virtual proxy.

The resulting program is capable of simulating several hundred up to a few thousand tetrahedra at real time rates. The lower limit is given for non-linear simulations, using haptic output. Haptics pose a limit to the maximum step size of about 20ms and in addition consume about half of the CPU cycles. If only a corotated strain is used with a time step of 40ms that corresponds to a visual simulation rate of 25Hz, the number of tetrahedra can be increased to over 3000 elements. Using Cauchy's strain would permit to push the number of elements even higher, but leads to unpleasing and visible volume inflation.

5.5 Conclusion and Directions for Further Work

This thesis presented a framework for the simulation of deformable objects. We build upon a tetrahedral discretisation of the solid object and use finite elements with linear shape functions to obtain an ordinary differential equation. For the reproduction of material parameters, the finite element method showed superior to mass-spring meshes. In addition, finite elements allow an efficient representation of visco-elastic materials, which permitted the first interactive simulation of visco-elastic solids. An important part of the framework is the model for strain, which should be rotationally invariant to be in line with the large displacements that occur during the manipulation. Besides the non-linear Green strain, we proposed to use a corotated Cauchy strain based on the polar decomposition, which leads to a 'quasi-linear' ordinary differential equation.

The framework permits to employ different numerical methods for the solution of this equation. Some benchmarks showed that the implicit methods work best, already anticipated as the equation can be assumed to be stiff. As implicit time integration relies on an efficient solution of linear systems, we employed several methods: the (modified) cg method of Baraff, the newly introduced (modified) Eisenstat method, that already includes preconditioning, and sparse direct solvers, which are superior for smaller dimensions. Both methods have been extended to be compatible with constraints. The interactive application has been augmented by haptic output.

When this work was started, nonlinear finite elements have been considered too slow for this kind of applications. We achieve real-time frame rates for up to about a thousand elements on a single processor workstation. Using corotated strain, the application permits the interactive simulation of several thousand elements at real time frame rates, even though it is only optimised on an algorithmical level and uses double precision floating point numbers. A specialisation to the SIMD floating point architecture of current processors could further improve these results. Today nearly every research group in this field is switching or already has switched to finite elements or related methods. Properly specialised for interactivity they are only two to three times more expensive than mass-spring meshes.

A straightforward extension of our approach is the use of quadratic shape functions on the tetrahedral mesh. As already mentioned in section 3.1.4 this results in a piecewise linear approximation of the strain tensor, which means that more complex computations need to be performed per element. On the other hand, the deformed shape is approximated piecewise quadratically, which would justify the use of a surface mesh with a much higher resolution for the visualisation. This would also pave the way for a locking free formulation; perhaps even with a reduced integration that decreases the computational expenses.

We did not discuss cutting into these meshes, which is a subject of active research. The approach of Delingette [DCA00] to remove the tetrahedra, which are touched by a scalpel, is commonly considered to be not satisfactory. Hence, as long as a model with a topological structure is used, the area of the cut needs to be remeshed. Recently an interesting approach has been presented by Bielser et al. [BGTG03], extending his previous work on using a state machine for remeshing. An efficient implementation and support data structure for changing mesh topology is also required by an adaptive simulation. We investigated adaptivity using a hierarchical basis, which provides a sound numerical foundation for local refinements. The octasection method however generates a too fast growing number of elements to allow a satisfying adaptive simulation. We regard other subdivision methods and their efficient implementation for interactive applications, especially in combination with topological modifications, as an interesting area of future work. Another approach, eliminating topology related problems completely, could consist in using meshless or element-free finite element methods [ZT00], which are currently an area of active research in numerical analysis.

Appendix A

Additional Remarks to Chapter 2 and 3

A.1 Some Propositions from the Theory of Elasticity

In this addendum, we provide some more precise formulated theorems and lemmata to chapter 2. For completeness, we begin with Cauchy's fundamental theorem on the existence of the stress tensor field, which provides the basis of the mathematical treatment of continuum mechanics.

Theorem A.1 (Cauchy's theorem) *Assume, that the body force density f^φ and surface traction g^φ applied to a body D^φ are continuous, and that the Cauchy stress* vector *field $\sigma^\varphi(x^\varphi, n^\varphi)$ is continuously differentiable with respect to x^φ and continuous in n^φ. Then the axiom of force balance implies, that there exists a continuous differentiable* tensor *field σ^φ on D^φ, such that*

$$\sigma^\varphi(x^\varphi, n^\varphi) = \sigma^\varphi(x^\varphi)n^\varphi$$

and

$$\begin{aligned} -\operatorname{div}^\varphi \sigma^\varphi(x^\varphi) &= f^\varphi(x^\varphi) && \text{for all } x^\varphi \in D^\varphi \\ \sigma^\varphi(x^\varphi)n^\varphi &= g^\varphi(x^\varphi) && \text{for all } x^\varphi \in \partial D^\varphi . \end{aligned}$$

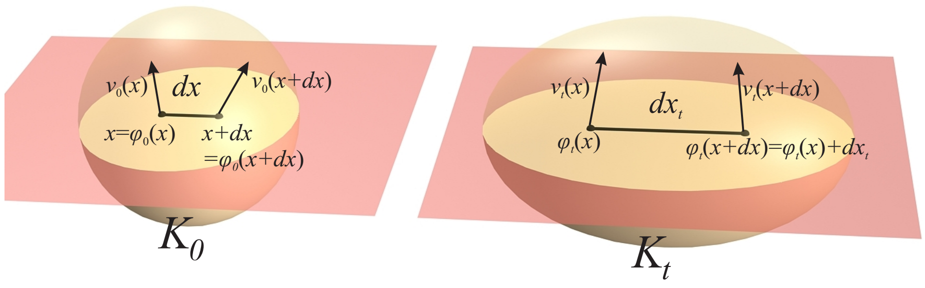


Figure A.1: Computing the spatial rate of deformation.

If additionally the axiom of moment balance is fulfilled, as a consequence

$$\sigma^{\varphi} = \sigma^{\varphi T}$$

holds.

Proposition A.2 *For the spatial rate of deformation $D = \frac{1}{2}(\frac{\partial v_t}{\partial x_t} + \frac{\partial v_t}{\partial x_t}^T)$ and the rate of Green's tensor $\dot{\epsilon}^G$ it holds*

$$\dot{\epsilon}^G = \dot{\nabla\varphi}^T \nabla\varphi + \nabla\varphi^T \dot{\nabla\varphi} = \nabla\varphi^T D \nabla\varphi.$$

Proof: With $F := \nabla\varphi$, the first equality follows from the definition $\epsilon^G = F^T F - \text{id}$ and the chain rule. For the second equality we use that $\nabla v F = \dot{F}$, because (fig. A.1)

$$dv_t = \frac{\partial v_t}{\partial x_t} dx_t = \frac{\partial v_t}{\partial x_t} F dx$$

and

$$dv_t = \frac{\partial}{\partial t}(dx_t) = \frac{\partial}{\partial t}(F dx) = \dot{F} dx.$$

The proposition follows. ■.

Proposition A.3 (Double contraction) *For a symmetric matrix A and an arbitrary matrix B, it holds*

$$A : B = A : \{\frac{1}{2}(B + B^T)\}.$$

Proof: For an antisymmetric $C = \frac{1}{2}(C - C^T)$ the equation

$$A : C = \frac{1}{2}(A : C - A : C^T) = (A : C - C : A) = (A : C - A : C) = 0$$

holds, because A is symmetric. Therefore the claim

$$A : B = A : \{\frac{1}{2}(B + B^T) + \frac{1}{2}(B - B^T)\} = A : \{\frac{1}{2}(B + B^T)\}$$

follows. ■.

A.2 Definition of the Variation

The finite element method heavily relies on the calculus of variation. In chapter 3 we used the first variation of a function without formally defining it. This section will at least give a definition and explain the symbols and expressions used. For further reading we refer to textbooks on functional analysis, to Blanchard and Brüning [BB82] or Klingbeil [Kli88].

The calculus of variation concerns the minimisation of a function $\mathcal{F}$, defined on a not necessary finite dimensional Banach space X. For example, this can be a mechanical energy defined on the set of admissible configurations. Similar to the finite dimensional case, a necessary condition is a vanishing first variation, which replaces the usual first derivative.

Definition A.4 (First variation) *Let $x, \delta x \in X$. Then*

$$\delta\mathcal{F}(\delta x) := \frac{\partial \mathcal{F}}{\partial \tau}(x + \tau\delta x)|_{\tau=0} \delta x$$

is called the first variation *of $\mathcal{F}$ in the direction δx.*

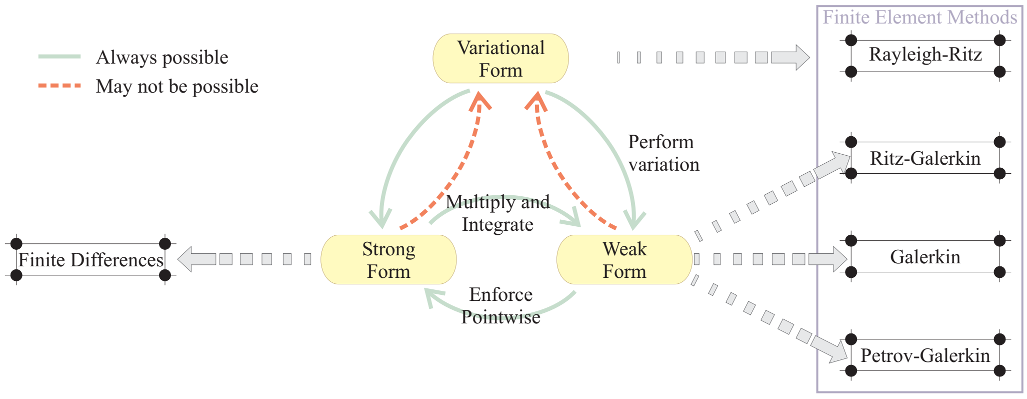


Figure A.2: Modelling and discretisation methods in computational mechanics.

If the domain of $\mathcal{F}$ is a genuine subset of X, additional restrictions may need to be posed to δx. For the reader familiar with functional analysis, the definition is linked to the Fréchet and Gâteaux derivative of $\mathcal{F}$. The variational derivative is by definition the weakest concept of the three. In the sense it is applied in most physical oriented textbooks, it is used in place of the Fréchet derivative, i.e. assumed to exist for all variations δx and to be linear and continuous in δx.

To conclude this short tour-de-horizon we present the variations used in chapter 3.

Proposition A.5 (Variation of deformation measures) *The first variations of $F = \nabla\varphi, D, C$ and ϵ expand to*

$$\begin{aligned}\delta F(\delta x) &= \nabla \delta x\\ \delta D(\delta x) &= \frac{1}{2}(\nabla \delta x + \nabla \delta x^T)\\ \delta\epsilon(\delta x) &= \frac{1}{2}\delta C(\delta x)\\ \delta C(\delta x) &= \nabla\varphi^T \nabla \delta x + \nabla \delta x^T \nabla\varphi.\end{aligned}$$

Proof: The first two equations follow trivially with the linearity of the derivative. For the last we differentiate

$$\frac{\partial C}{\partial \tau}(\varphi + \tau\delta x) = \frac{\partial}{\partial \tau}\big((\nabla\varphi + \tau\nabla\delta x)^T(\nabla\varphi + \tau\nabla\delta x)\big) = \nabla\varphi^T\nabla\delta x + \delta x^T\nabla\varphi + 2\tau\nabla\delta x^T\nabla\delta x,$$

which for $\tau = 0$ proofs the claim. ■.

A.3 Rayleigh-Ritz and Galerkin Methods

The methods described in chapter 2 and 3 are subtly linked. This section is intended to clarify this associations and the vocabulary used to distinguish between them. As already seen, the physical principles of mechanics can be stated in a strong form, i.e. a pointwise defined, partial differential equation with boundary conditions, or in a weak integral form, that combines both in a single integral statement. These forms can always be transformed into each other. The weak form is sometimes also called a variational principle or equation. Contrastingly, by variational form we mean, that the problem is presented as a functional to be minimised, and the stationary points of whom give the weak and strong forms. Usual this is the connected mechanical energy, which does not necessarily exist, as for example with path dependent materials. The dashed transformations in figure A.2 are therefore not always possible, at least not in standard variational calculus.

The numerical discretisation methods are distinguished accordingly. The Finite Difference method works with the strong form and is limited in terms of the regions and boundary conditions it can handle. Finite element methods work with the variational or weak form. Galerkin methods, being a subset of the finite element toolbox, work with the weak form. If the spaces for the solution and the test function are not the same, for example used for problems with singularities, the method is called a Petrov-Galerkin method. In this case, of course a somewhat different technique to proof error estimates and convergence is needed. The Rayleigh-Ritz methods work with the variational form either by directly discretising the energy and minimising the discrete function (as we described our approach in Hauth et al. [HGS03a]) or by using variational calculus to provide a basis free and discretisation independent weak equation, which is then discretised, as presented here. In these terms, it satisfies the definition of a Galerkin method. Of course the stronger assumptions on the existence of a variational form remain, therefore this is sometimes emphasised by the term Ritz-Galerkin method. The interchange of minimisation and discretisation does indeed give the same equations. Ritz-Galerkin and Ralyleigh-Ritz methods were the first finite element methods considered, as the variational form permits a more powerful mathematical treatment. For example, the more general Galerkin method does not automatically lead to symmetric or positive definite systems, as the transformation from strong to weak form does not require either. Thus, the concrete implementation of Galerkin and Rayleigh-Ritz methods is very similar to identical, but the mathematical starting point is different.

Appendix B

A Mass-Spring-Damper System

As mass-spring-damper systems are very popular in graphics, we implemented a system for comparison. This chapter provides the details.

A solid is again described by a tetrahedral mesh, but now the internal forces are described by a spring-damper pair for each edge. The advantages of this approach are a very efficient force evaluation and rotational invariant forces by design. There are several publication concerning the selection of spring constant for a given behaviour [DKT95, LPC95, MBT03]. Unfortunately it can be shown, that mass-spring-damper systems fail to describe a continuum mechanical solid [VG98], though this does not lessen their popularity.

The spring-force between two nodes on node i (figure B.1) with distance d and rest distance l_0 is given as

$$f_{ij} = -k\Delta l \frac{d}{\|d\|} = -k(1 - \frac{l_0}{\|d\|})d,$$

the reaction force on node j is $f_{ji} = -f_{ij}$. For an implicit integrator we need the derivative

$$J := \frac{\partial f_{ij}}{\partial x_i} = -\{k(1 - \frac{l_0}{\|d\|})\mathrm{id} + \frac{k-l_0}{\|d\|^2}[d_k d_l]_{k,l}.\}$$

The Jacobian of the local force pair $[f_{ij}, f_{ji}]^T$ evaluates to

$$\begin{bmatrix} f_{ij} \\ f_{ji} \end{bmatrix}' = \begin{bmatrix} J & -J \\ -J & J \end{bmatrix}$$

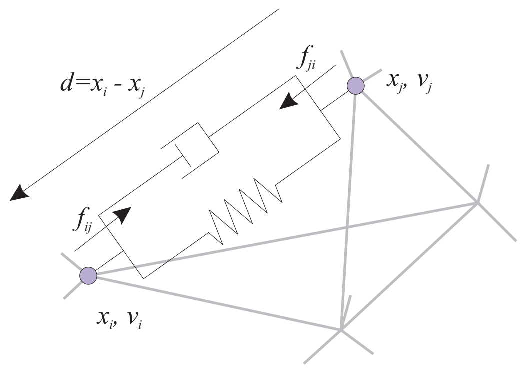


Figure B.1: A spring-damper connection along an edge.

The obvious damping force $f_d \sim -\Delta v$ has the disadvantage that rigid body rotations are damped as well. Therefore, the correct damping force has to be proportional to the projected velocity difference onto the edge

$$f_{ij}^R = -R\Delta v_d = -R\langle \Delta v, \tfrac{d}{\|d\|}\rangle \tfrac{d}{\|d\|} = -R\tfrac{\langle \Delta v, d\rangle}{\langle d,d\rangle} d.$$

with $\Delta v = v_i - v_j$, $R > 0$. The partial derivatives evaluate this time to

$$J_v^R := \frac{\partial f_{ij}^d}{\partial v_i} = -R\tfrac{1}{\langle d,d\rangle}[d_l d_k]_{k,l},$$

and, as f^R depends on d

$$J_x^R := \frac{\partial f_{ij}^R}{\partial x_i} = -R\{\tfrac{1}{\langle d,d\rangle} d\Delta v^T - \tfrac{2\langle \Delta v, d\rangle}{\langle d,d\rangle}[d_k d_l]_{k,l}\} - R\beta\mathrm{id},$$

with $\beta = \frac{\langle \Delta v, d\rangle}{\langle d,d\rangle}$.Note that due to the term $d\Delta v^T$ the matrix J_x^R is not symmetric. Finally the Jacobian is

$$\begin{bmatrix} f_{ij}^R \\ f_{ji}^R \end{bmatrix}' = \left[\begin{array}{cc|cc} J_x^R & -J_x^R & J_v^R & -J_v^R \\ -J_x^R & J_x^R & -J_v^R & J_v^R \end{array}\right].$$

B.1 Fitting of Constants

Van Gelder showed [VG98] that triangulated spring meshes fail to reproduce the results from a triangular finite element membrane, even qualitatively. Additionally he showed, that at least in the constant strain case with a Poisson ratio of $\nu = 0$, choosing the stiffness of an edge as the summation of the adjacent triangle areas over the rest length squared

$$k = \frac{E\sum_{T_a} \mathrm{vol}(T_a)}{\|l_0\|} \tag{B.1.1}$$

leads to an undistorted deformation state. He also describes the generalisation of formula (B.1.1) to tetrahedral meshes, replacing the area by the volume.

The damping constant is fitted afterwards, such that all the spring-damper pairs possess a user-defined mechanical quality. The mechanical quality of a spring-damper system is given as $Q = \frac{k}{\omega R}$, which we fit at the eigenfrequency of the system ω_0 We therefore solve the algebraic system

$$Q = \frac{k}{\omega_0 R}$$
$$\omega_0 = \frac{1}{2}\sqrt{4\frac{k}{m} + \left(\frac{R}{m}\right)^2}$$

for R. The resulting fourth order polynomial has exactly one real positive root,

$$R = \frac{1}{Q}\sqrt{2\,kmQ\sqrt{Q^2+1} - 2\,kmQ^2},$$

which we take as the desired damping constant.

B.2 Efficient Implementation

```
for all nodes $i$
  fetch $x_i, v_i$
  f=0
  for all neighs $j$ of $i$
    $f$+=$f_{ij}(x_j, v_j, l_{ij,0})$
  $f_i$=$f$
```

Algorithm B.1: Per Node

```
set (f ,0)
// loop arbitrarily over
// edges
for all edges $(i, j)$
  fetch $x_i, v_i, x_j, v_j, l_{ij,0}$
  $-f_{ji} = f_{ij}(x_j, v_j, l_{ij,0})$
  add $f_{ij}$ to $f_i$
  add $-f_{ij}$ to $f_j$
```

Algorithm B.2: Per Edge

```
set (f ,0)
// loop over sorted edges
for all edges $(i, j)$
  if ( i != lasti )
    write $flasti$=$f$
    fetch xi , vi
  fetch $x_j, v_j, l_{ij,0}$
  $f += f_{ij}(x_j, v_j, l_{ij,0})$
  add $-f_{ij}$ to $f_j$
```

Algorithm B.3: Hybrid

The assembly of the local spring forces can basically be performed in two fashions: per edge or per node. Whereas the per node variant is more efficient in terms of memory traffic, it computes f_{ij} and $f_{ji} = -f_{ij}$, therefore it the number of function evaluations equals two times the number of edges. Looping over all edges fixes this flaw, but now for each function evaluation all operands need to be fetched separately. Because function evaluations are very cheap, memory traffic cannot be neglected in this case. On the other hand, this variant possesses the advantage that no neighbourhood topology is needed in explicitly, only a way to iterate over the edges. The last variant is based on a sorted edge list. It combines the advantages of both variants, being as memory efficient as the first one, and as computational effective as the second one. Its disadvantage is, that it needs a minimum of topology preprocessing, as it requires a sorted edge list. On the other hand, by this all coordinate- and displacement-vectors are accessed more regularly, desirable for the streaming architecture of modern processors.

Appendix C

Computational details for Tetrahedral Elements

C.1 Shape Coefficients for a Linear Tetrahedral Element

The shape coefficients (figure C.1) can be computed by the following determinants [ZT00]

$$\alpha_{i0} = \tfrac{1}{6V} \det \begin{vmatrix} x_{j,1} & x_{j,2} & x_{j,3} \\ x_{k,1} & x_{k,2} & x_{k,3} \\ x_{l,1} & x_{l,2} & x_{l,3} \end{vmatrix} \qquad \alpha_{i1} = -\tfrac{1}{6V} \det \begin{vmatrix} 1 & x_{j,2} & x_{j,3} \\ 1 & x_{k,2} & x_{k,3} \\ 1 & x_{l,2} & x_{l,3} \end{vmatrix}$$

$$\alpha_{i2} = -\tfrac{1}{6V} \det \begin{vmatrix} x_{j,1} & 1 & x_{j,3} \\ x_{k,1} & 1 & x_{k,3} \\ x_{l,1} & 1 & x_{l,3} \end{vmatrix} \qquad \alpha_{i3} = -\tfrac{1}{6V} \det \begin{vmatrix} x_{j,1} & x_{j,2} & 1 \\ x_{k,1} & x_{k,2} & 1 \\ x_{l,1} & x_{l,2} & 1 \end{vmatrix}$$

with

$$[i, j, k, l] \in \{[0, 1, 2, 3], [1, 2, 3, 0], [2, 3, 0, 1], [3, 0, 1, 1]\}$$

and V denoting the volume of the tetrahedron.

C.2 Green's strain and St.Venant Kirchhoff material

Briefly, this section will describe an efficient computation of Green's strain and the forces resulting from a St.Venant Kirchhoff material. The generalisation to other material laws and a memory parameter model is straightforward.

As in the previous section and in section C.1, a tetrahedron is described by its shape coefficients $\alpha_{n,i}$ and the displacement $\mu_{n,j}$, where the first index denotes the node, the second one the coordinate. Following a Rayleigh-Ritz argumentation, this force is given by

$$\begin{aligned} f_{r,s} &= \int_V \sum_{i,j=1}^{3} \sigma_{ij} \frac{\partial \epsilon_{ij}}{\partial \mu_{r,s}} \\ &= V\Big[\sum_{i,j=1}^{3} 2\mu\epsilon_{ij} \frac{\partial \epsilon_{ij}}{\partial \mu_{r,s}} + \lambda \sum_{i=1}^{3} \epsilon_{ii} \sum_{j=1}^{3} \frac{\partial \epsilon_{jj}}{\partial \mu_{r,s}} \Big], \end{aligned} \tag{C.2.1}$$

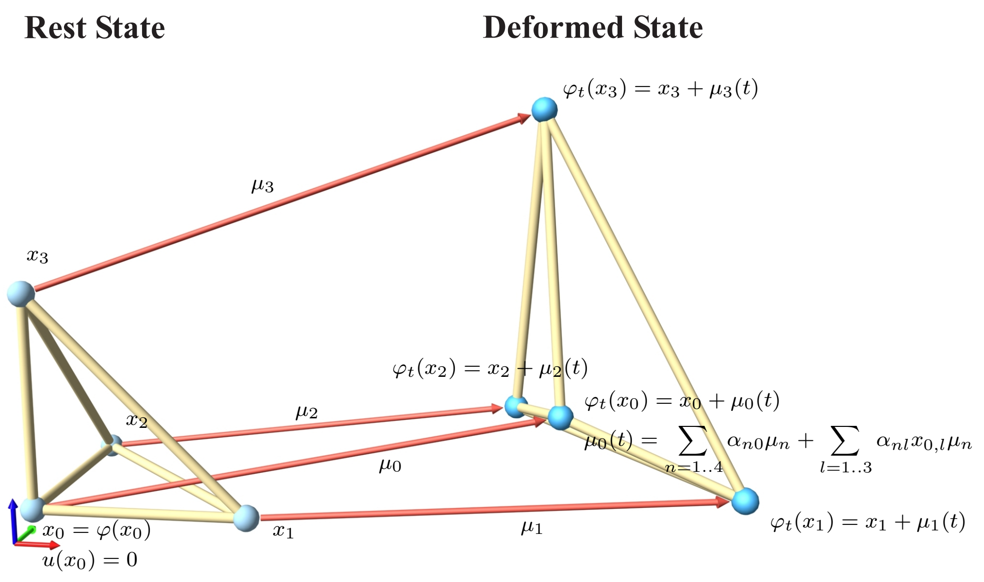

Rest State
Deformed State
$\varphi_t(x_3) = x_3 + \mu_3(t)$
μ_3
x_3
$\varphi_t(x_2) = x_2 + \mu_2(t)$
$\varphi_t(x_0) = x_0 + \mu_0(t)$
$\mu_0(t) = \sum_{n=1..4} \alpha_{n0}\mu_n + \sum_{l=1..3} \alpha_{nl} x_{0,l}\mu_n$
x_2
μ_2
μ_0
$x_0 = \varphi(x_0)$
$u(x_0) = 0$
x_1
μ_1
$\varphi_t(x_1) = x_1 + \mu_1(t)$

Figure C.1: Shape and state coefficients of a tetrahedral element.

inserting the material law and exploiting that the strain is constant over a tetrahedron. We abbreviate the shear part of σ by

$$g_{ij} := 2\mu\epsilon_{ij}$$

and the pressure part by

$$t := \lambda \sum_{i=1}^{3} \epsilon_{ii}.$$

In addition, we define the displacement gradients

$$e_{ij} := \sum_{n=1}^{4} \alpha_{n,i}\mu_{n,j}.$$

Computing the partial derivatives of the shape functions, the force then transforms to

$$f_{r,s} = V\Big\{t\Big[\alpha_{r,s} + \sum_{i=1}^{3} \alpha_{r,i}e_{is}\Big] + \sum_{j=1}^{3} \alpha_{r,j}\Big(g_{sj} + \sum_{i=1}^{3} g_{ij}e_{is}\Big)\Big\}. \tag{C.2.2}$$

Now the algorithm starts by computing e_{ij}, which takes $9 \cdot 7$ flops. From this Green's strain can be obtained as

$$\epsilon_{ij} = \frac{1}{2}\big(e_{ij} + e_{ji} + \sum_{k=1}^{3} e_{ik}e_{jk}\big).$$

As Green's strain is symmetric, this takes $6 \cdot 8$ operations. Another 12+3 operations then give the strain components t and g_{ij}. Identifying another common subexpression in the round brackets of (C.2.2), we define

$$h_{js} = \sum_{i=1}^{3} g_{ij}e_{is},$$

which uses $9 \cdot 5$ operations. Finally, from this the twelve force components can be computed as

$$f_{r,s} = V\Big\{t\Big[\alpha_{r,s} + \sum_{i=1}^{3} \alpha_{r,i} e_{is}\Big] + \sum_{j=1}^{3} \alpha_{r,j}\big(g_{sj} + h_{js}\big)\Big\}.$$

with $12 \cdot 10 + 3 \cdot 3$ operations, as the expression in the round brackets does not depend on r. This gives a total of $63 + 48 + 15 + 45 + 129 = 300$ operations per tetrahedron.

When memory parameters are used, we add the viscous stress computed according to equation (2.6.2) to the stress components g and t, which needs a few additional floating point operations per memory variable. Using non-linear stress measures, σ is computed from ϵ or F, and then plugged into equation C.2.1.

Appendix D

Frequently used Symbols

Ω	Parameter domain, domain of material coordinates.
$\varphi : \Omega \to \mathbb{R}^3$	Configuration of a deformable body.
u	Displacement vector field of a deformable body.
v	Velocity vector field.
t_n^φ	Traction or Cauchy stress vector in direction n.
σ^φ	Cauchy or true stress tensor.
σ^1	First Piola-Kirchhoff stress tensor.
σ	Second Piola-Kirchhoff stress tensor.
ϵ	Strain tensor, if not further specialised, Green's strain.
$\epsilon^C, \epsilon^{CR}$	Cauchy's strain tensor, the latter co-rotated.
F	Deformation gradient.
C	Cauchy-Green deformation tensor.
$\iota_1, \iota_2, \iota_{2a}, \iota_3$	Invariants of a symmetric three-by-three matrix.
W	Stored energy function.
$\hat{G}, G$	Relaxation function and its Prony series expansion.
Q	Mechanical quality factor.
ϕ	Nodal finite element basis.
ψ	Hierarchical finite element basis.
α	Finite element shape coefficients.
μ, ν	Finite-Element coefficient vector of u and v.
μ, λ, ν	In constitutive laws: Lamé's constants, and Poisson's ratio.
M	Mass matrix.
$y(t)$	State vector of the ordinary differential equation, usually $y = [\mu, \nu]^T$
f	Right-hand side of the ODE.
J	The Jacobian $\partial f / \partial y$.
Y_i	The numerical approximation to y at step i.
$\otimes$ (Kronecker product)	For $A = [a_{i,j}] \in \mathbb{R}^{n \times m}, B = [b_{k,l}] \in \mathbb{R}^{r \times s}$ the matrix $C = A \otimes B$ lies in $\mathbb{R}^{nr \times ms}$ and is given by $c_{(i-1)r+k,(j-1)s+l} = a_{i,j} b_{k,l}$, i.e. can be viewed as a block matrix composed of $a_{ij}B$-blocks.

Bibliography

[AB03] U. Ascher and E. Boxerman. On the Modified Conjugate Gradient Method in Cloth Simulation. *The Visual Computer*, 19:526–531, 2003.

[Abd01] A. Abdulle. *Chebychev Methods Based on Orthogonal Polynomials*. PhD Thesis, Université de Genève, 2001.

[ADD96] P. R. Amestoy, T. A. Davis, and I. S. Duff. An Approximate Minimum Degree Ordering Algorithm. *SIAM J. Matrix Anal. Appl.*, 17(4):886–905, 1996.

[AG00] T. M. Atanackovic and A. Guran. *Theory of Elasticity for Scientists and Engineers*. Birkhäuser, Boston, 2000.

[AM01] A. Abdulle and A. A. Medovikov. Second Order Chebyshev Methods based on Orthogonal Polynomials. *Numer. Math.*, 90(1):1–18, 2001.

[Bal99] R. Balaniuk. Using Fast Local Modeling to Buffer Haptic Data. In *Proc. of The Fourth PHANTOM Users Group Workshop (PUG99)*, Boston, USA, October 1999.

[Ban96] R. E. Bank. Hierarchical Bases and the Finite Element Method. *Acta Numerica*, pages 1–43, 1996.

[Bar84] A. H. Barr. Global and Local Deformations of Solid Primitives. In H. Christiansen, editor, *Computer Graphics (SIGGRAPH '84 Proceedings)*, volume 18, pages 21–30, July 1984.

[Bat82] K. L. Bathe. *Finite Element Methods*. Prentice Hall, Englewood Cliffs, 1982.

[BB82] P. Blanchard and E. Brüning. *Direkte Methoden der Variationrechnung*. Springer, 1982.

[BD96] S. Brandt and H. D. Dahmen. *Mechanik*. Springer, Berlin, Heidelberg, 3rd edition, 1996.

[Bey97] J. Bey. *Finite-Volumen- und Mehrgitterverfahren für elliptische Randwertverfahren*. PhD thesis, Eberhard-Karls-Universität Tübingen, 1997.

[BFA02] R. Bridson, R. Fedkiw, and J. Anderson. Robust Treatment of Collisions, Contact and Friction for Cloth Animation. *ACM Transactions on Graphics (Proc. SIGGRAPH 2002)*, 21(3):594–603, July 2002.

[BGTG03] D. Bielser, P. Glardon, M. Teschner, and M. Gross. A State Machine for Real-Time Cutting of Tetrahedral Meshes. In *Proc. Pacific Graphics 2003*, pages 377–386, Canmore, Canada, October 2003.

[BHM03a] D. Bartz, M. Hauth, and K. Mueller. Advanced Virtual Medicine: Techniques and Applications for Virtual Endoscopy. In *IEEE Visualization Tutorial T7*, 2003.

[BHM03b] D. Bartz, M. Hauth, and K. Mueller. Advanced Virtual Medicine: Techniques and Applications for Virtual Endoscopy. In *MICCAI Tutorial T8*, 2003.

[BHW94] D. E. Breen, D. H. House, and M. J. Wozny. Predicting the Drape of Woven Cloth Using Interacting Particles. In A. Glassner, editor, *Proceedings of SIGGRAPH '94*, pages 365–372. ACM SIGGRAPH, ACM Press, July 1994.

[BN98] M. Bro-Nielsen. Finite Element Modeling in Surgery Simulation. *Journal of the IEEE*, 86(3):490–503, 1998.

[BO02] C. Bruyns and M. Ottensmeyer. Measurements of Soft-Tissue Mechanical Properties to Support Development of a Physically Based Virtual Animal Model. In *Proc. MICCAI 2002*, volume 2488 of *LNCS*, pages 282 – 289. Springer, 2002.

[Bra97] D. Braess. *Finite Elemente*. Springer, 1997.

[BRM$^+$02] J. Brown, J. Rosen, M. Moreyra, M. Sinanan, and B. Hannaford. Computer-Controlled Motorized Endoscopic Grasper for In Vivo Measurement of Soft Tissue Biomechanical Characteristics. In *Medicine Meets Virtual Reality*, Studies in Health Technology and Informatics(85), pages 71–73, Newport Beach, CA, Jan. 23-26, 2002.

[BW98] D. Baraff and A. Witkin. Large Steps in Cloth Simulation. In M. Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, pages 43–54, July 1998.

[BW00] J. Bonet and R. D. Wood. *Nonlinear Continuum Mechanics for Finite Element Analysis*. Cambridge University Press, 2000.

[CB94] J. E. Colgate and J. M. Brown. Factors Affecting the Z-Width of a Haptic Display. In E. Straub and R. S. Sipple, editors, *Proceedings of the International Conference on Robotics and Automation. Volume 4*, pages 3205–3210, Los Alamitos, CA, USA, May 1994. IEEE Computer Society Press.

[CDA99] S. Cotin, H. Delingette, and N. Ayache. Real-Time Elastic Deformations of Soft Tissues for Surgery Simulation. *IEEE Transactions on Visualization and Computer Graphics*, 5(1):62–73, January/March 1999.

[CFL28] R. Courant, K. Friedrichs, and H. Lewy. Über die partiellen Differenzengleichungen der mathematischen Physik. *Math. Ann.*, 100:32–74, 1928.

[CGC$^+$02] S. Capell, S. Green, B. Curless, T. Duchamp, and Z. Popovic. Interactive Skeleton-Driven Dynamic Deformations. In *SIGGRAPH 2002*, pages 586–593. ACM SIGGRAPH, 2002.

[Cia92] P. G. Ciarlet. *Mathematical Elasticity. Vol. I.* North-Holland Publishing Co., Amsterdam, 1992. Three-Dimensional Elasticity.

[CK02] K.-J. Choi and H.-S. Ko. Stable but Responsive Cloth. In *SIGGRAPH 2002 Conference Proceedings*, pages 604–611. ACM Press/ACM SIGGRAPH, 2002.

[Coq90] S. Coquillart. Extended Free-Form Deformation: A Sculpturing Tool for 3D Geometric Modeling. In F. Baskett, editor, *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24, pages 187–196, August 1990.

[CSMT03] F. Cordier, H. Seo, and N. Magnenat-Thalmann. Made-to-Measure Technologies for an Online Clothing Store. *IEEE Computer Graphics and Applications*, 23(1):38–48, January/February 2003.

[CVT95] M. Courshesnes, P. Volino, and N. M. Thalmann. Versatile and Efficient Techniques for Simulating Cloth and Other Deformable Objects. In R. Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 137–144. ACM SIGGRAPH, Addison Wesley, August 1995. held in Los Angeles, California, 06-11 August 1995.

[CYMTT92] M. Carignan, Y. Yang, N. Magnenat-Thalmann, and D. Thalmann. Dressing Animated Synthetic Actors with Complex Deformable Clothes. *Computer Graphics (SIGGRAPH '92 Proceedings)*, 26(2):99–104, July 1992.

[CZ92] D. T. Chen and D. Zeltzer. Pump It Up: Computer Animation of a Biomechanically Based Model of Muscle Using the Finite Element Method. In E. E. Catmull, editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 89–98, July 1992.

[DCA00] H. Delingette, S. Cotin, and N. Ayache. A Hybrid Elastic Model allowing Real-Time Cutting, Deformations and Force-Feedback for Surgery Training and Simulation. In *Computer Animation Conference*, 2000.

[DD99] T. A. Davis and J. S. Duff. A Combined Unifrontal/Multifrontal Method for Unsymmetric Sparse Matrices. *ACM Trans. Math. Softw.*, 25(1):1–20, 1999.

[DDBC99] G. Debunne, M. Desbrun, A. Barr, and M.-P. Cani. Interactive Multiresolution Animation of Deformable Models. In N. Magnenat-Thalmann and D. Thalmann, editors, *Computer Animation and Simulation '99*, pages 133–144. Springer-Verlag, 1999.

[DDCB01] G. Debunne, M. Desbrun, M.-P. Cani, and A. H. Barr. Dynamic Real-Time Deformations using Space and Time Adaptive Sampling. In *SIGGRAPH 2001*, pages 31–36, 2001.

[DEG$^+$99] J. W. Demmel, S. C. Eisenstat, J. R. Gilbert, X. S. Li, and J. W. Liu. A Supernodal Approach to Sparse Partial Pivoting. *SIAM J. Matrix Anal. Appl.*, 20(3):720–755, 1999.

[Del98] H. Delingette. Towards Realistic Soft Tissue Modeling in Medical Simulation. *Proceedings of the IEEE : Special Issue on Surgery Simulation*, pages 512–523, April 1998.

[DER90] I. S. Duff, A. M. Erisman, and J. K. Reid. *Direct Methods for Sparse Matrices*. Monographs on numerical analysis. Clarendon, 1990.

[DG96] M. Desbrun and M.-P. Gascuel. Smoothed Particles : A new Paradigm for Animating Highly Deformable Bodies. In R. Boulic and G. Hegron, editors, *Computer Animation and Simulation '96*, pages 61–76. Springer, 1996. Poitiers, France, August 31–September 18, 1996.

[DGL99] J. W. Demmel, J. R. Gilbert, and X. S. Li. An Asynchronous Parallel Supernodal Algorithm for Sparse Gaussian Elimination. *SIAM J. Matrix Anal. Appl.*, 20(4):915–952, 1999.

[DKT95] O. Deussen, L. Kobbelt, and P. Tücke. Using Simulated Annealing to Obtain Good Nodal Approximations of Deformable Bodies. In D. Terzolpoulos and D. Thalmann, editors, *Computer Animation and Simulation '95*, pages 30–43. Springer-Verlag Wien New York, 1995. Proceedings of the Eurographics Workshop in Maastricht, Netherlands, September 2–3, 1995.

[DS96] J. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Classics in Applied Mathematics. 16. Philadelphia, PA: SIAM, Society for Industrial and Applied Mathematics., 1996.

[DSB99] M. Desbrun, P. Schröder, and A. Barr. Interactive Animation of Structured Deformable Objects. In *Graphics Interface*, pages 1–8, June 1999.

[dV76] B. F. de Veubeke. The Dynamics of Flexible Bodies. *Int. J. Engrg. Sci.*, pages 895–913, 1976.

[EDC96] J. W. Eischen, S. Deng, and T. G. Clapp. Finite-Element Modeling and Control of Flexible Fabric Parts. *IEEE Computer Graphics and Applications*, 16(5):71–80, September 1996. ISSN 0272-1716.

[EEH00] B. Eberhardt, O. Etzmuß, and M. Hauth. Implicit-Explicit Schemes for Fast Animation with Particle Systems. In *Eurgraphics Computer Animation and Simulation Workshop 2000*, pages 137–151, 2000.

[EGS03] O. Etzmuss, J. Gross, and W. Straßer. Deriving a Particle System from Continuum Mechanics. *Transactions on Visualization and Computer Graphics*, 2003. Accepted for publication.

[Eis81] S. C. Eisenstat. Efficient Implementation of a Class of Preconditioned Conjugate Gradient Methods. *SIAM J. Sci. Stat. Comput.*, 2:1–4, 1981.

[EKK$^+$01] O. Etzmuß, M. Keckeisen, S. Kimmerle, J. Mezger, M. Hauth, and M. Wacker. A Cloth Modelling System for Animated Characters. In *Proceedings Graphiktag*, 2001.

[EKS03] O. Etzmuss, M. Keckeisen, and W. Straßer. A Fast Finite Element Solution for Cloth Modelling. *Proceedings of Pacific Graphics 2003*, 2003.

[EOV90] S. Eisenstat, J. Ortega, and C. Vaughan. Efficient Polynomial Preconditioning for the Conjugate Gradient Method. *SIAM J. Sci. Stat. Comput.*, 11(5):859–872, 1990.

[Etz02] O. Etzmuss. *Animation of Surfaces with Applications to Cloth Modelling*. PhD thesis, University of Tuebingen, February 2002.

[EW99] B. Eberhardt and A. Weber. A Particle System Approach to Knitted Textiles. *Computers & Graphics*, 23(4):599–606, 1999.

[EWS96] B. Eberhardt, A. Weber, and W. Strasser. A Fast, Flexible, Particle-System Model for Cloth Draping. *IEEE Computer Graphics and Applications*, 16(5):52–60, September 1996.

[eXp03] eXpat. http://expat.sourceforge.net/, 2003.

[Fab09] G. Faber. Über stetige Funktionen. *Mathematische Annalen*, 66:81–94, 1909.

[FG99] S. F. Frisken-Gibson. Using Linked Volumes to Model Object Collisions, Deformation, Cutting, Carving, and Joining. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):333–348, October 1999.

[Fun93] Y. C. Fung. *Biomechanics: Mechanical Properties of Living Tissues*. Springer, New York, 1993.

[FvT97] P. Faloutsos, M. van de Panne, and D. Terzopoulos. Dynamic Free-Form Deformations for Animation Synthesis. *IEEE Transactions on Visualization and Computer Graphics*, 3(3):201–214, July–September 1997.

[GHE$^+$00] J. Gross, M. Hauth, B. Eberhardt, O. Etzmuss, L. Schnieder, and G. F. Buess. Physically Based Tissue Models for Surgery Simulation. *Min. Invas. Ther. & Allied Technol.*, 9(3/4):320, 2000.

[GHEB01] J. Gross, M. Hauth, O. Etzmuß, and G. F. Buess. Modelling Viscoelasticity in Soft Tissues. In *Int. Workshop on Deformable Modelling and Soft Tissue Simulation*. Elsevier, November 2001.

[GKS02] E. Grinspun, P. Krysl, and P. Schröder. CHARMS: A Simple Framework for Adaptive Simulation. In *SIGGRAPH 2002 Conference Proceedings*, pages 281–290. ACM Press/ACM SIGGRAPH, 2002.

[GL61] A. Guillou and B. Lago. Domaine de stabilité associe aux formules d'intégration numérique d'équations différentielles, à pas separes et à pas lies recherche de formules à grand rayon de stabilité. In *1er Congr. Assoc. Francaise Calcul AFCAL, Grenoble, 14-15-16 Sept. 1960, 43-56* . 1961.

[Gla03] E. Gladilin. *Biomechanical Modeling of Soft Tissue and Facial Expressions for Craniofacial Surgery Planning*. Phd thesis, FU Berlin, Germany, 2003.

[GM97] S. Gibson and B. Mirtich. A Survey of Deformable Modeling in Computer Graphics. MERL Technical Report TR97-19, MERL, 1997.

[GMS92] J. R. Gilbert, C. Moler, and R. Schreiber. Sparse Matrices in MATLAB: Design and Implementation. *SIAM J. Matrix Anal. Appl.*, 13(1):333–356, 1992.

[Gre97] A. Greenbaum. *Iterative Methods for Solving Linear Systems*, volume 17 of *Frontiers in Applied Mathematics*. SIAM, 1997.

[GTT89] J.-P. Gourret, N. M. Thalmann, and D. Thalmann. Simulation of Object and Human Skin Deformations in a Grasping Task. In J. Lane, editor, *Computer Graphics (SIGGRAPH '89 Proceedings)*, volume 23, pages 21–30, July 1989.

[Gur81] M. E. Gurtin. *An Introduction to Continuum Mechanics.* Mathematics in Science and Engineering, Vol. 158. New York - London: Academic Press (Harcourt Brace Jovanovich, Publishers). , 1981.

[GV02] A. Gerisch and J. Verwer. Operator Splitting and Approximate Factorization for taxis-diffusion-reaction Models. *Appl. Numer. Math.*, 42(1-3):159–176, 2002.

[GZDH02] E. Gladilin, S. Zachow, P. Deuflhard, and H.-C. Hege. A Nonlinear Elastic Soft Tissue Model for Craniofacial Surgery Simulations. *ESAIM, Proc.*, 12:61–66, 2002.

[Hac93] W. Hackbusch. *Iterative Lösung großer schwachbesetzter Gleichungssysteme. (Iterative solution of large sparse systems of equations).2., überarb. u. erw. Aufl.* Stuttgart: Teubner., 1993.

[Hag89] W. W. Hager. Updating the Inverse of a Matrix. *SIAM Rev.*, 31(2):221–239, 1989.

[Hau99a] M. Hauth. Iterative Verfahren in der Lösung großer Systeme von Differentialgleichungen. Diplomarbeit, University of Tübingen, December 1999.

[Hau99b] M. Hauth. Numerische Verfahren zur Simulation von Textilien. Diplomarbeit, University of Tübingen, March 1999.

[HB00] D. H. House and D. E. Breen, editors. *Cloth Modeling and Animation*. A. K. Peters, 2000.

[HE01] M. Hauth and O. Etzmuß. A High Performance Solver for the Animation of Deformable Objects using Advanced Numerical Methods. In *Proc. Eurographics*, pages 137–151, Manchester, UK, 2001.

[HEE^{+}02] M. Hauth, O. Etzmuss, B. Eberhardt, R. Klein, R. Sarlette, M. Sattler, K. Daubert, and J. Kautz. Cloth Animation and Rendering. In *Proc. of Eurographics, Tutorial 3*, 2002.

[HES03] M. Hauth, O. Etzmuss, and W. Strasser. Analysis of Numerical Methods for the Simulation of Deformable Models. *The Visual Computer*, 19:581–600, 2003.

[HGS03a] M. Hauth, J. Gross, and W. Straßer. Interactive Physically Based Solid Dynamics. In *Proc. SIGGRAPH Symposium on Computer Animation 2003*. ACM Press, 2003.

[HGS03b] M. Hauth, J. Gross, and W. Straßer. Soft Tissue Simulation based on Measured Data. In *Proc. MICCAI 2003*, 2003.

[HNW93] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations. I: Nonstiff problems. 2. rev. ed.* Springer-Verlag. , Berlin, 1993.

[HS90] N. J. Higham and R. S. Schreiber. Fast Polar Decomposition of an Arbitrary Matrix. *SIAM J. Sci. Stat. Comput.*, 11(4):648–655, 1990.

[HS03] M. Hauth and W. Strasser. Corotational Simulation of Deformable Solids. WSI Report WSI-2003-6, Wilhelm Schickard Institute for Computer Science, Graphical-Interactive Systems (WSI/GRIS), University of Tübingen, July 2003.

[HS04] M. Hauth and W. Strasser. Corotational Simulation of Deformable Solids. In *Proc. WSCG 2004*, pages 137–145, 2004.

[Hut99] R. Hutter. *Total hourgalss control- eine robuste FE-Methode zur Simulation von weichen Geweben*. PhD thesis, ETH Zürich, 1999.

[HW96] E. Hairer and G. Wanner. Solving Ordinary Differential Equations II. Springer-Verlag, Berlin, 1996.

[HWS04] M. Hauth, M. Wacker, and W. Strasser. Interactive Simulation of Soft Tissue Visco-Elasticity based on Measured Data. In *Proc. 38. Jahrestagung, Deutsche Gesellschaft für Biomedizinische Technik im VDE*, 2004.

[JF03] D. L. James and K. Fatahalian. Precomputing Interactive Dynamic Deformable Scenes. *ACM Transactions on Graphics (Proc. SIGGRAPH 2003)*, 22(3):879–887, July 2003.

[JP99] D. L. James and D. K. Pai. ArtDefo - Accurate Real Time Deformable Objects. In *SIGGRAPH 1999*, pages 65–72. ACM SIGGRAPH, 1999.

[JP02] D. L. James and D. K. Pai. DyRT: Dynamic Response Textures for Real-Time Deformation Simulation with Graphics Hardware. *ACM Transactions on Graphics*, 21(3):582–585, July 2002.

[Kaw80] S. Kawabata. *The Standardization and Analysis of Hand Evaluation*. The Textile Machinery Society of Japan, Osaka, 1980.

[KBF$^+$03] H.-S. Ko, D. Breen, R. Fedkiw, M. Hauth, and R. House. Clothing Simulation and Animation. In *ACM SIGGRAPH Course 29*, 2003.

[KCC$^+$00] Y.-M. Kang, J.-H. Choi, H.-G. Cho, D.-H. Lee, and C.-J. Park. Real-time Animation Technique for Flexible and Thin Objects. In *WSCG*, pages 322–329, February 2000.

[KCO$^+$03] A. E. Kerdok, S. M. Cotin, M. P. Ottensmeyer, A. M. Galea, R. D. Howe, and S. L. Dawson. Truth Cube: Establishing Physical Standards for Real Time Soft Tissue Simulation. *Medical Image Analysis*, 7(283–291), 2003.

[KGB98] R. M. Koch, M. H. Gross, and A. Bosshard. Emotion Editing using Finite Elements. *Computer Graphics Forum*, 17(3):295–302, 1998.

[KGC$^+$96] R. M. Koch, M. H. Gross, F. R. Carls, D. F. von Büren, G. Fankhauser, and Y. I. H. Parish. Simulating Facial Surgery Using Finite Element Models. *Computer Graphics*, 30(Annual Conference Series):421–428, 1996.

[KGKG98] E. Keeve, S. Girod, R. Kikinis, and B. Girod. Deformable Modeling of Facial Tissue for Craniofacial Surgery Simulation. *Computer Aided Surgery*, 3(5):228–238, 1998.

[KGS01] P. Krysl, E. Grinspun, and P. Schröder. Natural Hierarchical Refinement for Finite Element Methods. *Int. J. for Num. Meth. Eng.*, to appear, 2001.

[KHM04] D. Kowalski, M. Hauth, and U. Matern. Volumetrische und Scher - Messungen von Gewebeeigenschaften - vorläufige Resultate. In *Proc. 38. Jahrestagung, Deutsche Gesellschaft für Biomedizinische Technik im VDE*, 2004.

[Kil00] M. J. Kilgard. A Practical and Robust Bump-mapping Technique for Todays GPUs. In *Proc. Game Developement Conference 2000*, July 2000.

[Kja79] E. Kjartansson. Constant-Q, Wave Propagation and Attenuation. *Journal of Geophysical Research*, 85:4737–4748, 1979.

[KKK96] C. Kuhn, U. Kühnapfel, and H.-G. Krumm. A ’Virtual Reality’ based Training System for Minimally Invasive Surgery. In *Proc. Computer Assisted Radiology (CAR ’96)*, Paris, June 1996.

[Kli88] E. Klingbeil. *Variationsrechnung*. BI-Wiss.-Verl., second edition, 1988.

[KOGD03] D. Kalanovic, M. P. Ottensmeyer, J. Gross, and S. L. Dawson. Independent Testing of Soft Tissue Visco-Elasticity using Indentation and Rotary Shear Deformations. In *Proc. MMVR 11*. IOS Press, 2003.

[KSFS03] M. Keckeisen, S. L. Stoev, M. Feurer, and W. Straßer. Interactive Cloth Simulation in Virtual Environments. In *Proceedings of IEEE Virtual Reality 2003*, 2003.

[KVD$^+$01] M. Kauer, V. Vuskovic, J. Dual, G. Szekely, and M. Bajka. Inverse Finite Element Characterization of Soft Tissues. In *Proc. Medical Image Computing and Computer-Assisted Intervention - MICCAI 2001: 4th International Conference*, pages 128–136, 2001.

[Lov27] A. E. H. Love. *A Treatise on the Mathematical Theory of Elasticity. 4. ed.* Cambridge, University press. , 1927.

[LPC95] J. Louchet, X. Provot, and D. Crochemore. Evolutionary Identification of Cloth Animation Models. In D. Terzopoulos and D. Thalmann, editors, *Computer Animation and Simulation '95*, pages 44–54. Eurographics, Springer-Verlag, September 1995.

[LTW95] Y. Lee, D. Terzopoulos, and K. Waters. Realistic Face Modeling for Animation. In R. Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 55–62. ACM SIGGRAPH, Addison Wesley, August 1995. held in Los Angeles, California, 06-11 August 1995.

[MBB$^+$02] K. Montgomery, C. Bruyns, J. Brown, G. Thonier, A. Tellier, and J. Latombe. Spring: A General Framework for Collaborative, Real-Time Surgical Simulation. In *Proc. Medicine Meets Virtual Reality (MMVR02)*, Newport Beach, CA, January 23-26 2002.

[MBT03] A. Maciel, R. Boulic, and D. Thalmann. Deformable Tissue Parametrized by Properties of Real Biological Tissue. In *Proc. International Symposium on Surgery Simulation and Soft Tissue Modeling*, pages 74 – 87, June 2003.

[MDSB02] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr. Discrete Differential-Geometry Operators for Triangulated 2-Manifolds. In *Proc. VisMath 2002*, Berlin, Germany, 2002.

[Mez01] J. Mezger. Effiziente Kollisionsdetektion in der Simulation von Textilien. Diplomarbeit, WSI/GRIS, University of Tuebingen, 2001.

[MG04] M. Mueller and M. Gross. Interactive Virtual Materials. to appear in Proc. Graphics Interface 2004, 2004.

[MJ96] R. MacCracken and K. I. Joy. Free-Form Deformations with Lattices of Arbitrary Topology. *Computer Graphics*, 30(Annual Conference Series):181–188, 1996.

[MK99] H. Maaß and U. Kühnapfel. Noninvasive Measurement of Elastic Properties of Living Tissue. In *EMBEC'99*, volume 37, pages 1460–1461, 1999.

[MKE03] J. Mezger, S. Kimmerle, and O. Etzmuß. Hierarchical Techniques in Collision Detection for Cloth Animation. *Journal of WSCG*, 11(2):322–329, 2003.

[MMD$^+$02] M. Müller, L. McMillan, J. Dorsey, R. Jagnow, and B. Cutler. Stable Real-Time Deformations. In *Proceedings of the ACM SIGGRAPH Symposium on Computer Animation*, pages 49–54. ACM Press, 2002.

[NHS02] F. Neyret, R. Heiss, and F. Senegas. Realistic Rendering of an Organ Surface in Real-Time for Laparoscopic Surgery Simulation. *the Visual Computer*, 18(3):135–149, may 2002.

[NMK$^+$03] A. Nava, E. Mazza, F. Kleinermann, N. Avis, and J. McClure. Determination of the Mechanical Properties of Soft Human Tissues through Aspiration Experiments. In *Proc. Medical Image Computing and Computer-Assisted Intervention - MICCAI 2003: 6th International Conference*, 2003.

[ODGK03] C. O'Sullivan, J. Dingliana, T. Giang, and M. K. Kaiser. Evaluating the Visual Fidelity of Physically Based Animations. *ACM Transactions on Graphics*, 22(3):527–536, July 2003.

[OH99] J. F. O'Brien and J. K. Hodgins. Graphical Modeling and Animation of Brittle and Fracture. In *SIGGRAPH 1999*, pages 137–146, 1999.

[OKG02] M. P. Ottensmeyer, D. Kalanovic, and J. Gross. Comparison of Indentation and Rotary Shear as Modes for Interrogating Soft Tissue Visco-Elasticity. In *Proc. SMIT2002*, Oslo, Norway, September 2002.

[Ott02] M. P. Ottensmeyer. TeMPeST 1-D: An Instrument for Measuring Solid Organ Soft Tissue Properties. *Experimental Techniques*, 26(3):28–50, May/June 2002.

[PB81] S. M. Platt and N. Badler. Animating Facial Expressions. In *Computer Graphics (SIGGRAPH '81 Proceedings)*, volume 15, pages 245–252, August 1981.

[PB88] J. C. Platt and A. H. Barr. Constraint Methods for Flexible Models. In J. Dill, editor, *SIGGRAPH '88 Conference Proceedings*, Annual Conference Series, pages 279–288. ACM SIGGRAPH, Addison Wesley, August 1988.

[PDA00] G. Picinbono, H. Delingette, and N. Ayache. Real-Time Large Displacement Elasticity for Surgery Simulation: Non-Linear Tensor-Mass Model. In *Third International Conference on Medical Robotics, Imaging And Computer Assisted Surgery: MICCAI 2000*, pages 643–652, October 2000.

[PDA01] G. Picinbono, H. Delingette, and N. Ayache. Non-linear and anisotropic elastic soft tissue models for medical simulation. In *ICRA2001*, May 2001.

[PLDA02] G. Picinbono, J.-C. Lombardo, H. Delingette, and N. Ayache. Improving Realism of a Surgery Simulator: Linear Anisotropic Elasticity, Complex Interactions and Force Extrapolation. *Journal of Visualisation and Computer Animation*, 13(3):147–167, July 2002.

[Pol02] K. Polthier. Computational Aspects of Discrete Minimal Surfaces. In J. Hass, D. Hoffman, A. Jaffe, H. Rosenberg, R. Schoen, and M. Wolf, editors, *Proc. of the Clay Summer School on Global Theory of Minimal Surfaces*, 2002.

[Pro95] X. Provot. Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behavior. In W. A. Davis and P. Prusinkiewicz, editors, *Graphics Interface '95*, pages 147–154, May 1995.

[PTVF88] W. H. Press, S. A. Teukolski, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, first edition, 1988.

[PvdDJ$^+$01] D. K. Pai, K. van den Doel, D. L. James, J. Lang, J. E. Lloyd, J. L. Richmond, and S. H. Yau. Scanning Physical Interaction Behavior of 3D Objects. In *SIGGRAPH 2001*, pages 87–96. ACM Press, 2001.

[PW89] A. Pentland and J. Williams. Good Vibrations: Modal Dynamics for Graphics and Animation. In *SIGGRAPH '89)*, pages 215–222. ACM Press, July 1989.

[Ree83] W. T. Reeves. Particle Systems — A Technique for Modeling a Class of Fuzzy Objects. *Computer Graphics*, 17(3):359–376, July 1983.

[Rhe98] W. C. Rheinboldt. *Methods for Solving Systems of Nonlinear Equations*, volume 70 of *CBMS-NSF regional conference series in applied mathematics*. SIAM, second edition, 1998.

[RKK97] D. C. Ruspini, K. Kolarov, and O. Khatib. The Haptic Display of Complex Graphical Environments. *Computer Graphics*, 31(Annual Conference Series):345–352, August 1997.

[Saa96] Y. Saad. *Iterative Methods for Sparse Linear Systems*. PWS Publishing, Boston, 1996.

[Sai01] H. Sailer. Messung viskoelastischer Gewebeeigenschaften in vivo. Master thesis, Fachhochschule Furtwangen, 2001.

[SBD$^+$00] G. Székely, C. Brechbühler, J. Dual, R. Enzler, J. Hug, R. Hutter, N. Ironmonger, M. Kauer, V. Meier, P. Niederer, A. Rhomberg, P. Schmid, G. Schweitzer, M. Thaler, V. Vuskovic, G. Tröster, U. Haller, and M. Bajka. Virtual Reality-Based Simulation of Endoscopic Surgery. *Presence: Teleoperators and Virtual Environments*, 9(3):310–333, June 2000.

[SBHR98] G. Szekely, C. Brechbühler, R. Hutter, and A. Rhomberg. Modelling of Soft Tissue Deformation for Laparoscopic Surgery Simulation, 1998.

[SCB04] K. Salisbury, F. Conti, and F. Barbagli. Haptics Rendering: Introductory Concepts. *Computer Graphics and Applications*, March 2004.

[SD92] K. Shoemake and T. Duff. Matrix Animation and Polar Decomposition. In *Proceedings of Graphics Interface '92*, pages 258–264, May 1992.

[Sen03] Sensable Technologies. Ghost 4 SDK. available from Sensable Technologies, 2003.

[SGBM98] M. A. Schill, S. F. F. Gibson, H.-J. Bender, and R. Männer. Biomechanical Simulation of the Vitreous Humor in the Eye Using an Enhanced ChainMail Algorithm. *Lecture Notes in Computer Science*, 1496:679ff, 1998.

[SHT$^+$98] N. Suzuki, A. Hattori, A. Takatsu, T. Kumano, A. Ikemoto, Y. Adachi, and A. Uchiyama. Virtual Surgery System Using Deformable Organ Models and Force Feedback System with Three Fingers. In *Proc. MICCAI'98*, volume 1496 of *Lecture Notes in Computer Science*, pages 397–403, 1998.

[SL99] J. G. Siek and A. Lumsdaine. The Matrix Template Library: Generic Components for High-Performance Scientific Computing. *Computing in Science and Engeneering*, pages 70–78, November 1999. Avaiable via ftp from `http://www.lsc.nd.edu/ research/mtl`.

[SLS$^+$00] R. Sinkus, J. Lorentzen, D. Schrader, M. Lorenzen, M. Dargatz, and D. Holz. High-resolution Tensor MR Elastography for Breast Tumour Detection. *Physics in Medicine and Biology*, 45:1649–64, 2000.

[SP86] T. W. Sederberg and S. R. Parry. Free-Form Deformation of Solid Geometric Models. *SIGGRAPH '86 Proceedings)*, 20(4):151–160, August 1986.

[Sta97] J. Stam. Stochastic Dynamics: Simulating the Effects of Turbulence on Flexible Structures. *Computer Graphics Forum*, 16(3):159–164, August 1997. Proceedings of Eurographics '97. ISSN 1067-7055.

[SW79] T. Steihaug and A. Wolfbrandt. An Attempt to Avoid Rxact Jacobian and Nonlinear Equations in the Numerical Solution of Stiff Differential Equations. *Math. Comput.*, 33:521–534, 1979.

[TBHF03] J. Teran, S. Blemker, V. N. T. Hing, and R. Fedkiw. Finite Volume Methods for the Simulation of Skeletal Muscle. In D. Breen and M. Lin, editors, *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SGA-03)*, pages 68–74, Aire-la-Ville, July 26–27 2003. Eurographics Association.

[TF88a] D. Terzopoulos and K. Fleischer. Deformable Models. *The Visual Computer*, 4:306–331, 1988.

[TF88b] D. Terzopoulos and K. Fleischer. Modeling Inelastic Deformation: Viscoelasticity, Plasticity, Fracture. In J. Dill, editor, *SIGGRAPH 1988*, volume 22, 269–278 1988.

[TPBF87] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically Deformable Models. In M. C. Stone, editor, *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21, pages 205–214, July 1987.

[TW88] D. Terzopoulos and A. Witkin. Physically Based Models with Rigid and Deformable Components. *IEEE Comput. Graph.*, 8:41–51, 1988.

[TW90] D. Terzopoulos and K. Waters. Physically-Based Facial Modelling, Analysis, and Animation. *The Journal of Visualization and Computer Animation*, 1(2):73–80, 1990.

[VG98] A. Van Gelder. Approximate Simulation of Elastic Membranes by Triangle Meshes. *Journal of graphics tools*, 3:21–42, 1998.

[VMT00a] P. Volino and N. Magnenat-Thalmann. Implementing fast Cloth Simulation with Collision Response. In *Computer Graphics International Proceedings*, pages 257–268, 2000.

[VMT00b] P. Volino and N. Magnenat-Thalmann. *Virtual Clothing*. Springer, 2000.

[VMT01] P. Volino and N. Magnenat-Thalmann. Comparing Efficiency of Integration Methods for Cloth Animation. In *Computer Graphics International Proceedings*, pages 265–274, 2001.

[VSBH99] J. Verwer, E. Spee, J. Blom, and W. Hundsdorfer. A second-order Rosenbrock Method Applied to Photochemical Dispersion Problems. *SIAM J. Sci. Comput.*, 20(4):1456–1480, 1999.

[VSC01] T. Vassilev, B. Spanlang, and Y. Chrysanthou. Fast Cloth Animation on Walking Avatars. In A. Chalmers and T.-M. Rhyne, editors, *EG 2001 Proceedings*, volume 20(3) of *Computer Graphics Forum*, pages 260–267. Blackwell Publishing, 2001.

[W3C] W3C. W3C Document Object Model Specification. http://www.w3.org/DOM/.

[WDGT01] X. Wu, M. S. Downes, T. Goktekin, and F. Tendick. Adaptive Nonlinear Finite Elements for Deformable Body Simulation Using Dynamic Progressive Meshe. In *Proc. EG 2001*, volume 20(3) of *Comp. Graphics Forum*, pages 349–358, 2001.

[Wei86] J. Weil. The Synthesis of Cloth Objects. *Proc. SIGGRAPH*, 20:49–54, 1986.

[WG] M. Wacker and J. Gross. Cloth Measurements in the Virtual TryOn Project. Private communications.

[WKK$^+$04] M. Wacker, M. Keckeisen, S. Kimmerle, W. Strasser, V. Luckas, C. Groß, A. Fuhrmann, M. Sattler, R. Sarlette, and R. Klein. Virtual Try-On: Virtuelle Textilien in der Graphischen Datenverarbeitung. *Informatik Spektrum (Special Issue on Computer Graphics)*, 2004.

[Yan81] M. Yannakakis. Computing the Minimum Fill-In is NP-Complete. *SIAM J. Alg. Disc. Meth.*, 2:77–79, 1981.

[Yse92] H. Yserentant. Hierarchical Bases. In *ICIAM91*, pages 281–290. SIAM, 1992.

[ZGHD00] S. Zachow, E. Gladiline, H.-C. Hege, and P. Deuflhard. Finite-Element Simulation of Soft Tissue Deformation. In H. U. L. et al., editor, *Computer Assisted Radiology and Surgery (CARS)*, pages 23–28. Elsevier Science B.V., 2000.

[ZS95] C. B. Zilles and J. K. Salisbury. A Constraint-Based God-Object Method for Haptic Display. In *Proc. of the International Conference on Intelligent Robots and Systems*, volume 3, page 3146, 1995.

[ZT00] O. Zienkiewicz and R. Taylor. *The Finite Element Method. Vol. 1-3.* Oxford: Butterworth-Heinemann., 2000.

[ZWK68] O. C. Zienkiewicz, M. Watson, and I. P. King. A Numerical Method of Visco-Elastic Stress Analysis. *Int. J. Mech. Sci.*, 10:807–27, 1968.