Advanced Topics in Virtual Garment Simulation Part 1

B. Thomaszewski[1] , M.Wacker[1,2] , and W.Straßer[1]

[1]WSI/GRIS, University of Tübingen, Germany
[2]University of applied sciences Dresden, Germany

Abstract

For more than two decades, cloth simulation has been an active research area in computer graphics. In order to create efficient high-quality animations, techniques from many research fields have to be thoroughly combined. The ongoing interest in this field is also due to the multidisciplinary nature of cloth simulation which spurs development and progress in collision detection, numerical time integration, constrained dynamics, or motion control, to name just a few areas. Beyond the very basic approaches, the complexity of the material can be daunting if no guidance is given. It is therefore the goal of this tutorial to provide the reader with an introduction and a guideline to the relevant matter. In order to provide a concise review, we will focus on advanced topics in cloth simulation, shedding light on both theoretical and practical aspects. This will pave the ground for those willing to implement a contemporaneous cloth simulation system as well as researchers who consider to start working in this area.

1. Introduction

The physical simulation of deformable objects is a central research area in computer graphics. Problems emanating from this field are usually complex to model and pose significant demands on computational resources. This is particularly true for the more specific case of cloth simulation. Due to the thin and flexible nature of cloth, it produces detailed folds and wrinkles, which in turn can lead to complicated self-collisions. In order to tackle this challenge within the requirements imposed by computer graphics, specialised methods have to be designed for concrete applications. These can roughly be divided into two categories: applications for which quality is most important and those for which speed is the urgent demand. To clarify this distinction, the requirement for time-critical cloth simulation could read: *given a fixed timing (e.g. 25 frames per second) optimize the (visual) quality of the simulation*. An analogous formulation can be stated for the first case. Because every category stands in its own right, we will address both of these fields in this tutorial.

Since cloth simulation has been an active research field for quite a while now, there is a broad variety of different approaches. One objective of this course is therefore to give the reader an introduction to thoughtfully chosen matter and a guideline towards practical applications. Another is to introduce algorithms and tools necessary for creating high quality animations. Finally, a further goal is to supply the audience with techniques for accelerating computations and eventually obtain fast simulations. The latter includes models specifically designed for computational efficiency. These methods can again be divided into two categories: algorithms which are optimized for sequential real-time computations and those which exploit the parallel potential of current hardware.

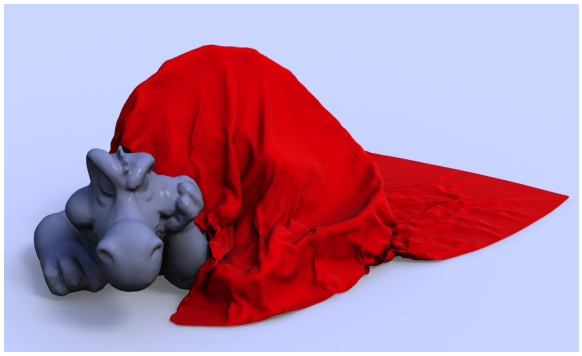

Figure 1: *The Eurographics 2007 Phlegmatic Dragon covered by a sheet of cloth.*

Because physically-based modelling has become the de

facto standard in cloth simulation, we will exclusively treat physically based methods in this tutorial and leave alternative approaches like geometry- or heuristics-based models aside.

1.1. Major Challenges in Cloth Simulation

Despite the comparably long history of cloth simulation this field can by no means be considered closed. For instance, the accurate modelling of real textile materials remains an open issue. Even if static properties can be measured quite accurately, textiles typically exhibit a high dynamical behaviour (e.g. hysteresis, damping, etc.) for which no accurate models are available yet. In the field of collision handling there is still room for improvement as well. Besides requirements like speed and robustness, error recovery is a point of particular practical relevance [VMT06a]. Furthermore, garment design and integration in CAD-like environments needs to be pushed forward if clothing simulation is to become a widely-used tool in industrial applications (see e.g. [CK05]). Likewise, electronic vending of clothing via the internet requires Virtual Try-on platforms specifically tailored to the individual needs of end customers. Finally, since high-quality animations can still be very time consuming, the need for designing faster techniques has not ceased. One way towards more efficient methods is to design, possibly from scratch, new algorithms that result in significant computational speed-up. Another is to develop or adapt algorithms which exploit parallelism on current hardware. In the course of this tutorial, we will address most of the aforementioned topics in great detail.

1.2. Overview

This tutorial assumes that the reader is already familiar with the basics of physically-based simulation. An overview of the state-of-the-art in this field is given in Sec. 2. The following sections of the first part are intended to supply the reader with the basics of contemporaneous cloth simulation. The third section gives an introduction to continuum mechanics and an example of numerical implementation. The way in which wrinkles and folds form depends mainly on the bending properties of the fabric. Because of its importance, a separate section (4) is devoted to this issue and practical ways of integrating bending into cloth simulation are discussed. In Sec. 5 we will describe how computations can be mapped onto parallel architectures. This includes generic modifications and extensions to existing algorithms in order to exploit potential parallelism. Both shared and distributed memory architectures will be considered, and we will address implementation related issues for both these settings.

The second part of these notes addresses the measurement of real world fabric parameters, including multi-layered textiles and seams, and their integration into cloth simulation. Furthermore, integrated Virtual Try-On applications are discussed, including models for parametrically deformable human bodies, body animation and motion retargeting as well as techniques for real-time cloth simulation.

2. State-of-the-Art in Virtual Clothing

In this section we will give an overview of the current state-of-the-art in virtual clothing. We will pay special attention to the topics covered in this tutorial. A more comprehensive summary of the evolution of this research can be found in [MTVW*05], Part 1.

2.1. Mechanical Models

For dynamically deformable surfaces, mass-spring systems [Pro95] and the more general particle systems [BHW94, VCMT95, EWS96] continue to be the most widely used simulation techniques in computer graphics. The popularity of mass spring systems is due to the ease of implementation and low computational costs. The accuracy offered by this method is, however, rather limited. As an example, simple homogeneous materials cannot be simulated consistently and the results highly depend on the specific mesh used in the simulation. If the reproduction of authentic material behaviour is desired (as, e.g., by the textile community), approaches based on continuum mechanics have to be used. Continuum-based approaches lead to a set of partial differential equations (PDEs), which have to be discretized in space and time. The spatial discretisation is usually carried out by means of finite differences (FDM) or finite element methods (FEM). Techniques based on finite elements and continuum mechanics (referred to as FE-approaches in the remainder) have not seen as much attention in cloth simulation as particle and mass spring systems. While we only mention the most relevant work here an extensive list can be found in [HB00]. Most of the existing FE-approaches are based on the geometrically exact thin shell formulation presented by Simo et al. [SFR89a]. Departing from the fully nonlinear theory, Eischen et al. [EDC96] proposed a cloth simulation method using quadrilateral, curvilinear elements. Because of the buckling behaviour of cloth, which can lead to divergence in the algorithm, an adaptive arc-length control is used. Etzmuss et al. [EKS03] presented a linear FE-approach based on the plane-stress assumption. Bending is treated separately from in-plane deformations and a co-rotational strain formulation is used to account for arbitrary rigid body transformations. The work presented in [TWS06] extends the concept of Subdivision Finite-Elements by Cirak et al. [COS00] to dynamic cloth simulation using a co-rotational strain formulation. As a middle course between simple mass spring systems and finite elements, Volino et al. [VMT05] proposed an accurate particle system which draws on notions from continuum mechanics but replaces the numerical discretisation by a more direct geometric formulation.

2.2. Numerical Time Integration

The mechanical model provides the means for computing internal forces due to fabric deformations. The dynamical evolution of the system (i.e., the trajectories of the nodes) is then determined by Newton's second law. In the discrete setting, the time dimension is decomposed into discrete time intervals and numerical integration methods are used to advance the system from a given state to its next state in time. Most generally, one distinguishes between two types of integration schemes: *explicit* methods compute the next state in time based on the current state derivatives, which are readily computed according to the mechanical model. Commonly used explicit integration methods are the second-order accurate Midpoint method used e.g. by Volino et al. [VCMT95] and the fourth-order accurate Runge-Kutta scheme used for instance by Eberhardt et al. [EWS96]. For computer graphics applications the numerical accuracy is usually less important than stability and robustness. As a well known fact, explicit schemes only provide conditional stability (see [HE01]). Since the differential equations resulting from cloth simulation are inherently stiff, explicit methods need small step sizes to guarantee a stable simulation. Consequently, computation times soon become excessive with increasing problem sizes. *Implicit* schemes do not suffer from this restriction since, in this case, stability is independent of the step size. Since the seminal work of Baraff et al. [BW98] implicit methods have therefore become predominant for physically-based simulations in computer graphics. Implicit methods include the unknown state of the system at the end of the time step in the update formula. Therefore, a system of (nonlinear) equations has to be solved in every time step. This process is one of the most time consuming parts in cloth simulation. Widely used representatives of this class are the first-order accurate backward Euler scheme [BW98] and the second-order BDF-2 scheme [HE01]. Recently, the Newmark family of integrators (including both explicit and implicit variants) found its way into computer graphics [BMF03, GH*03]. For a detailed overview and comparison of existing integration schemes and their efficiency for cloth simulation, the reader is referred to [VMT01] and [HE01].

2.3. Collision Handling

Besides the simulation of the mechanical properties of cloth the interaction with its environment has to be modeled as well. This involves the detection of any collisions and an adequate response to prevent the clothes from intersecting. The proper treatment of these two components (to which we refer as collision handling in the remainder) is a very complex task [THM*05]. While the physical simulation engine computes new states at distinct intervals only, collisions can occur at any instant in between such intervals. Algorithms based on continuous collision detection can handle these cases in a robust way, but are often very complex and time consuming. Therefore, the collision handling step is a major bottleneck in the simulation pipeline. Basically, detecting interference between two arbitrarily shaped objects breaks down to determining the interference between all primitives (i.e. faces, edges, and vertices) of one mesh with every primitive of the mesh representing the other object. With complex objects comprising thousands of faces, this naïve approach soon becomes too expensive due to its quadratic average run time with respect to the number of faces. A common way to accelerate the interference tests is to structure the objects under consideration hierarchically with bounding volumes. A bounding volume hierarchy (BVH) is then constructed for each object in the scene (including deformable as well as rigid objects) in a preprocessing step, using, for example, a top-down approach. In this case, a bounding volume enclosing the entire object is set as the root node of the tree representing the hierarchy. This node is then subdivided recursively until a leaf criterion is reached. Usually, the leaves contain one single primitive. Common choices for bounding volumes in cloth simulation are axis aligned bounding boxes (AABB) [van97, LAM01] or the more general discrete oriented polytopes (k-DOPs) [KHM*98, MKE03]. For treating self-collisions efficiently, it is necessary to adapt the general BVH algorithms to this special case. Measures related to the curvature of the surface can be used to quickly rule out flat, non-intersecting parts of the surface [VMT94, Pro97]. As another useful extension for self-collisions, continuous collision detection based on BVHs can be used, in which the exact contact points between two successive time steps are detected [BFA02].

Once the intersecting parts of a garment have been determined, an appropriate response has to computed in order to prevent the imminent intersections. A method well-suited for cloth simulation is the one presented by Bridson et al. [BFA02]. The essence of this method is to apply a stopping impulse to approaching triangles (i.e., to adjust their nodal velocities) whenever their distance falls below a certain threshold. However, even with such robust approaches there can be situations in which intersections cannot be prevented. An example for this are complicated multi-layer collisions or when cloth is pinched together due to character motion. In such cases special care is necessary in order to restore an intersection free state and to keep the simulation running [BWK03, VMT06a].

Despite the aforementioned acceleration techniques, collision handling remains a major bottleneck of cloth simulation. Recent developments aimed at further accelerating these algorithms by migrating computations to the graphics card [GKJ*05] or by exploiting parallel architectures [TB07, TPB07].

3. Mathematical and Physical Foundations

In this section, we will describe how a physically based model for deformable objects can be derived. As already pointed out in Sec. 2, mass-spring systems are still widely

used for cloth simulation in computer graphics. However, for authentic material mapping and hence realistic and reliable draping behaviour of cloth, as required e.g. by the textile community, one must necessarily resort to continuum mechanics. We will therefore restrict our considerations to methods based on continuum mechanics and begin with a brief review of the relevant foundations. As we will see, the central quantities in this theory are *strain* and *stress* which are related to each other via material or constitutive laws. We will only discuss some general material laws at this point and postpone more sophisticated models especially suited for textiles to the second part of these notes. Having established the governing equations, we will turn to an exemplary spatial discretisation.

3.1. Continuous Models

The structure of textiles or woven fabrics is clearly different from continuous media. However, modelling each single thread would certainly be an inefficient approach. We will instead approximate the garment geometrically with a polygonal mesh. If the regions of each polygon contains a sufficiently large number of threads (or weave periods) we can safely approximate the fabric as a continuous medium. Departing from a continuum model, we can derive a consistent spatial discretisation. Consistency here means that with increasing resolution the computed approximation converges to the actual solution of the continuous problem. This allows us to choose the spatial resolution in accordance to computational requirements without changing the properties of the cloth. In doing so, the dynamic motion of garments can be simulated efficiently and independently of discretisation throughout a broad range of resolutions. The first ingredient for such a continuum model is introduced in the next section.

3.2. Deformation Measures

In order to describe the equations that govern the (dynamic) behaviour of deformable objects, we consider the conditions that must hold in an equilibrium state. Generally speaking, a deformable object is in equilibrium if the internal forces due to deformation exactly cancel the external forces acting on its volume and boundary. The first step in this analysis is to compute deformation, which requires an appropriate measure.

A conceptional description of a deformable object embedded in three-dimensional is given by its *configuration mapping*

$$\varphi : \Omega \subset \mathbf{R}^3 \rightarrow \mathbf{R}^3 \,, \tag{1}$$

where Ω is its parameter domain. In dynamic scenes, where the object undergoes translation, rotation, and deformation this mapping also depends on time

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

It is common to base descriptions of state and behaviour of deformable objects on an initial and a current configuration. For simplicity, the initial mapping is assumed to be the identity

$$\bar{\varphi}(x_1,x_2,x_3) = \varphi(x_1,x_2,x_3,0) = \mathrm{id} \,.$$

The configuration mapping can be given the interpretation of transforming positions of material points in the initial, undeformed state to corresponding positions in the current, deformed configuration (see Fig. 2).

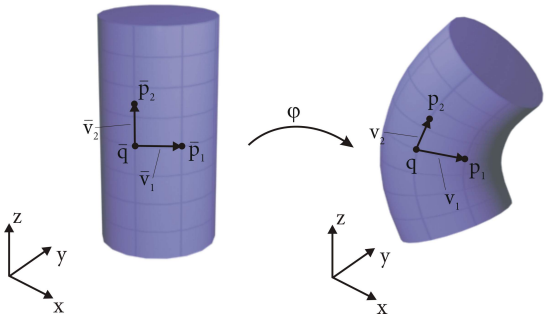

Figure 2: *Relative change of positions from undeformed to deformed configuration.*

For later derivations, it is convenient to express the current state in terms of a displacement field $\mathbf{u}$ from the initial configuration as

$$\varphi = \bar{\varphi} + u = id + u : \Omega \rightarrow \mathbf{R}^3 \,. \tag{2}$$

The displacement field itself is not an adequate measure of deformation, since invariance under rigid body motion such as translation is not given. A general deformation measure should capture the relative change between two elemental vectors in their initial and current configuration. Besides the obvious direct change in length, the angle formed by two vectors can change as well. One measure that takes both these characteristics into account is the scalar product. Consider the two vector pairs in the undeformed and deformed configuration in Fig. 2. The vectors are given by $\bar{\mathbf{v}}_i = \bar{\mathbf{p}}_i - \bar{\mathbf{q}}$, respectively $\mathbf{v}_i = \mathbf{p}_i - \mathbf{q}$. We can use a Taylor series expansion to express the vectors in the current configuration in an alternative form as

$$\begin{aligned} \mathbf{v}_i &= \varphi(\bar{\mathbf{q}} + \bar{\mathbf{v}}_i) - \varphi(\bar{\mathbf{q}}) \\ &= \varphi(\bar{\mathbf{q}}) + \nabla\varphi(\bar{\mathbf{q}}) \cdot \bar{\mathbf{v}}_i + O(\bar{\mathbf{v}}_i^2) - \varphi(\bar{\mathbf{q}}) \\ &\approx \nabla\varphi(\bar{\mathbf{q}})\bar{\mathbf{v}}_i = (\nabla u(\bar{\mathbf{q}}) - \mathrm{id})\bar{\mathbf{v}}_i \,. \end{aligned}$$

For later use, we define the deformation gradient F as

$$F = \nabla\varphi \,, \tag{3}$$

which can be given the interpretation of mapping vectors in the initial configuration to vectors in the current configuration. A general deformation measure can now be derived as

the difference of scalar products in the rest and current state:

$$\mathbf{v}_1 \cdot \mathbf{v}_2 - \bar{\mathbf{v}}_1 \cdot \bar{\mathbf{v}}_2 = \bar{\mathbf{v}}_1 \cdot (\nabla\varphi^T \nabla\varphi - id) \cdot \bar{\mathbf{v}}_2. \quad (4)$$

Using Eq. (2) we can identify from Eq. (4) the symmetric *Green* strain tensor as

$$\varepsilon_G = \frac{1}{2}(\nabla\varphi^T \nabla\varphi - id) = \frac{1}{2}(\nabla u^T + \nabla u + \nabla u^T \nabla u) \; . \quad (5)$$

An alternative expression can be obtained using the deformation gradient

$$\varepsilon_G = \frac{1}{2}(F^T F - id) \; . \quad (6)$$

For practical purposes, Eq. (5) can be written in indicial notation as

$$(\varepsilon_G)_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} + \sum_k \frac{\partial u_k}{\partial x_i}\frac{\partial u_k}{\partial x_j}\right) \; . \quad (7)$$

In this form, the entries of the strain tensor can be interpreted geometrically: diagonal components ε_{ii} measure the change in length in the direction of the x_i-axis and off-diagonal entries ε_{ij} measure the shearing between two axes. For small displacements, the nonlinear terms are negligible. This gives rise to the linear *Cauchy* strain tensor

$$\varepsilon_C = \frac{1}{2}(\nabla\mathbf{u} + \nabla\mathbf{u}^T) \; , \quad (8)$$

which is used in small strain analysis. Because of its linearity, the Cauchy tensor is very common in computer graphics. A closer look reveals that while being invariant under translations, rotational invariance is not given. In dynamic simulations, where the objects usually undergo large rotations, this is a severe restriction. A possibility to circumvent this problem is to extract the rotational part from the deformation field. The key observation is that, using polar decomposition, F can be written as

$$F = RU \; , \quad (9)$$

where R is a rotation and U is a pure deformation. The rotation can then be determined by finding the eigenvectors of $F^T F$. From this together with Eq. (6) it can also be seen that ε_G is invariant under rotations since

$$F^T F = U^T R^T R U = U^T U \quad (10)$$

due to the orthogonality of R. Once the rotation field R is known, the rotated linear strain tensor can be computed as

$$\varepsilon_{CR}(\varphi) = \varepsilon_C(R^T \varphi). \quad (11)$$

In the expressions derived so far we have implicitly assumed a Cartesian reference frame. We may think of garments or more generally deformable surface as 2-manifolds embedded in three-dimensional space. As such, a two-dimensional parametrisation of the surface is necessary to correctly establish differential measures like deformation. For instance, approaches based on thin shell formulations (see Sec. 4.4) rely on (parametrised) curvilinear systems. In such settings, all expressions will depend on partial derivatives of the parametrisation with respect to local coordinates. Although in a different way, common deformation measures are eventually recovered (see Sec. 4.4).

Strains in a deformable solid are accompanied by counteracting forces, which are the subject of the following section.

3.3. Internal Stress and Equilibrium

To describe the distribution of internal forces, we will first consider a differential surface element with area dA on a cross section of a deformable body as depicted in Fig. 3. Let $\mathbf{n}$ denote the normal of that element and let the resultant force acting on it be $d\mathbf{f}$.

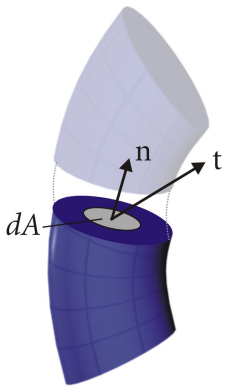

Figure 3: *Deformable solid in cross-sectional view. Traction vector* $\mathbf{t}$ *resulting from forces on area dA with normal* $\mathbf{n}$.

Then, the traction vector $\mathbf{t}$ is defined as

$$\mathbf{t} = \lim_{dA \to 0} \frac{d\mathbf{f}}{dA} \; .$$

Generalizing this expression for every normal direction leads to a mapping (or tensor) σ that maps every unit normal direction onto its traction vector,

$$\mathbf{t}(\mathbf{n}) = \sigma\mathbf{n} \; . \quad (12)$$

The tensor σ is called the *Cauchy* stress tensor. This symmetric tensor will be the subject of the next step on our way to the equilibrium equations.

Consider now a volume element V of a deformable body and let $\mathbf{f}$ be the body forces per unit volume acting on V. If we neglect forces due to inertia, translational equilibrium [†] implies that the sum of all forces acting on the volume V is

[†] as opposed to rotational equilibrium (see [BW97]p.103)

zero. Using equation (12), this can be expressed in integral form as

$$\int_{\partial V} \mathbf{t}\, da + \int_V \mathbf{f}\, dv = \int_{\partial V} \boldsymbol{\sigma}\mathbf{n}\, da + \int_V \mathbf{f}\, dv = 0\,, \qquad (13)$$

where $\mathbf{t}$ denotes the externally applied traction forces on the border ∂V. With the Gaussian divergence theorem, the surface integral of $\boldsymbol{\sigma}\mathbf{n}$ can be transformed into a volume integral,

$$\int_{\partial V} \boldsymbol{\sigma}\mathbf{n}\, da + \int_V \mathbf{f}\, dv = \int_V (\operatorname{div}\boldsymbol{\sigma} + \mathbf{f})\, dv = 0\,. \qquad (14)$$

Since this must hold for any enclosed volume, the (point wise) equilibrium equation of a deformable solid follows as

$$\operatorname{div}\boldsymbol{\sigma}(\mathbf{x}) + \mathbf{f}(\mathbf{x}) = 0\,. \qquad (15)$$

For dynamic simulation this equation has to be augmented by terms accounting for intertial forces. Furthermore, $\boldsymbol{\sigma}$ will also include viscous stress contributions due to material damping, leading to

$$\operatorname{div}\boldsymbol{\sigma}(\mathbf{x}, \dot{\mathbf{x}}) + \mathbf{f}(\mathbf{x}) = \rho\ddot{\mathbf{x}}\,. \qquad (16)$$

Here, ρ is the mass density and $\mathbf{x}$ denotes the vector field of positions. This continuous formulation is the starting point for numerical treatment. In Sec. 3.6 we will look at an exemplary spatial numerical discretisation. Before we continue with constitutive laws in Sec. 3.5 we will take a look at a hitherto neglected aspect of deformation.

3.4. Bending

As we have seen in the previous sections, the deformation modes in general (three-dimensional) continuum mechanics can be separated into stretching and shearing (cf. Eq. 7). These two deformation modes are *orthogonal* to each other: pure stretching does not lead to shear deformation and vice versa. In the special case of thin deformable surfaces, we intuitively identify *bending* as a further deformation mode. The question arises as to whether for bending deformation an analogous orthogonality relation holds. We will pursue this issue in the following.

The concept of bending cannot be derived from the three-dimensional, point wise view of continuum mechanics, since this standpoint is indifferent of *shape*. If we consider for example an elastic ball it is immediately clear that we cannot *bend* such an object. The same holds for other objects that do not exhibit a direction with significantly smaller size. If there is, however, such a direction like in the case of a thin plate or an elastic rod, bending deformations become possible.

As we can see from these examples, the ability to bend an object is closely related to its shape, or more precisely, to the proportion of its extents in the different dimensions. The ball is perfectly isotropic and has no intrinsic orientation. Therefore, the choice of a reference frame is arbitrary. While the geometry of the cube furnishes an intrinsic coordinate system, none of the dimensions is accentuated above

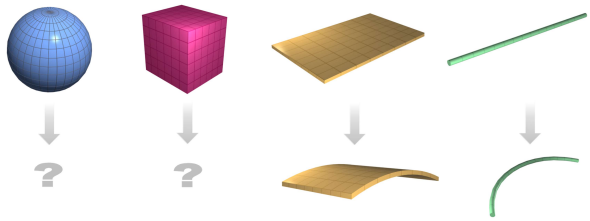

Figure 4: *Whether an object can be bent depends mainly on its shape. It is not clear how to bend a sphere or a cube. For a thin plate or a rod the notion of bending is intuitively clear.*

the others. In the example of the plate however, one direction can be distinguished, in which the lengths are clearly inferior to those in the orthogonal directions. The same holds for the case of a thin flexible rod. Note that the intuitive bending deformation of thin objects like the plate in the example is such that it causes as little in-plane deformation as possible. Hence, we may assume a bending mode which is orthogonal to the deformation modes of stretching and shearing. For the moment, we can most generally identify bending deformation of an embedded manifold as a change in curvature. We will return to this issue in a more specific discussion in the following paragraphs. In analogy to strain associated with stretching and shearing (to which we collectively refer as *membrane*-strains) we define the bending strain $\gamma_{\alpha\beta}$, $\alpha,\beta \in \{1,2\}$, where the components $\gamma_{\alpha\alpha}$ are normal curvatures in the directions of surface coordinates and $\gamma_{\alpha\beta}$, $\alpha \neq \beta$, is the torsional component.

3.5. Constitutive Relations in Linear Elasticity

The previous section led to the formulation of equilibrium equations involving the stress tensor $\boldsymbol{\sigma}$. In general, the relationship between stress and strain can be of high complexity. Here, we will focus on situations where stresses in a body depend only on its current state of deformation. Materials that fulfil this requirement are called *hyperelastic*. Generally, for two different types of elastic material, the same deformation will result in different stresses in terms of magnitude and possibly direction. This relationship between stress and strain is described by the elasticity tensor $\mathcal{C}$.

In the simplest case, the relationship between stress and strain is linear. Note that the term *linear* appears in two different contexts. First, there is a geometric relationship between displacement and strain. A linear strain-displacement assumption results in the Cauchy strain tensor, as described above. Second, the dependence between strain and stress itself can be either linear or nonlinear. We will start with a linear material law, for which the stress tensor can be written as

$$\boldsymbol{\sigma} = \mathcal{C} : \boldsymbol{\varepsilon}\,, \qquad (17)$$

where tensorial notation was used. This formulation accounts for the fact that a component ε_{ij} can potentially have influence on every entry σ_{kl} of the stress tensor. Using the summation convention, the stress tensor can conviniently be expressed as

$$\sigma_{ij} = \mathcal{C}_{ijkl}\varepsilon_{kl} \; . \tag{18}$$

Therefore, the entry $\mathcal{C}_{ijkl}$ can be interpreted as linking σ_{ij} to ε_{kl}.

As the simplest model, a linear-elastic isotropic material is governed by only two independent constants. In this case, the elasticity tensor[‡] $\mathcal{C}$ reads

$$\mathcal{C}_{ijkl} = \lambda\delta_{ij}\delta_{kl} + 2\mu\delta_{ik}\delta_{jl} \; , \tag{19}$$

where λ and μ are the *Lamé* constants (cf. [BW97]). These constants can be used to express the well-known Young modulus E and the Poisson ratio ν as

$$E = \frac{\mu(3\lambda+2\mu)}{\lambda+\mu}, \qquad \nu = \frac{\lambda}{2(\lambda+\mu)} \; . \tag{20}$$

Fig. 5 illustrates the meaning of these constants. In this example, a simple elastic beam is subjected to a longitudinal loading along the x-axis. This loading leads to a deformation $\varepsilon_{xx} = \frac{l-l_0}{l}$ counteracted by the stress σ_{xx} in the same direction. In linear small strain elasticity, these quantities are related by

$$\frac{\sigma_{xx}}{\varepsilon_{xx}} = E \; .$$

In addition to the extensional strain in the direction of the

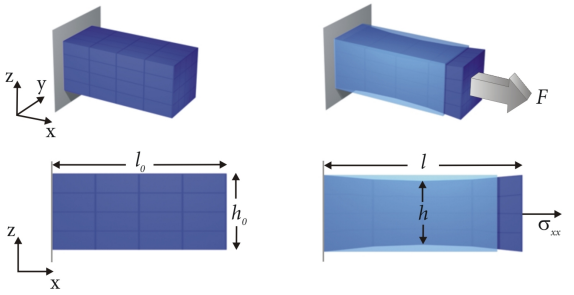

Figure 5: *An elastic beam is subjected to a horizontal loading in x direction. Longitudinal as well as transverse deformation can be observed.*

loading, we can usually also observe deformations $\varepsilon_{yy} = \varepsilon_{zz} = \frac{h-h_0}{h}$ in the orthogonal directions. Since this transverse contraction can be completely attributed to the axial loading, we can write

$$\varepsilon_{yy} = \varepsilon_{zz} = \frac{\nu}{E}\sigma_{xx} \; ,$$

(see also [Hau04]). For physical realism, ν has to be such that $0 \leq \nu \leq 0.5$. In the case of textiles, ν is usually smaller than 0.3.

The material model described above corresponds to Hooke's law in three dimensions. Because of its linearity and easy implementation, it is widely used in computer graphics. In order to model the basic in-plane properties of textiles, a more general material law is needed. In two dimensions, stress and strain are described by 2×2 tensors and the elasticity tensor consists of 16 components. Symmetry considerations[§] imply that

$$\mathcal{C}_{ijkl} = \mathcal{C}_{klij} \; ,$$

as well as

$$\mathcal{C}_{ijkl} = \mathcal{C}_{jikl}, \qquad \mathcal{C}_{ijkl} = \mathcal{C}_{ijlk} \; .$$

This results in an elasticity tensor of the form

$$\mathcal{C} = \begin{bmatrix} \mathcal{C}_{1111} & \mathcal{C}_{1112} & \mathcal{C}_{1122} \\ & \mathcal{C}_{1212} & \mathcal{C}_{1222} \\ \text{sym.} & & \\ & & \mathcal{C}_{2222} \end{bmatrix} \; ,$$

where components determined by symmetry have been omitted. This tensor is minimal in the sense that, assuming complete anisotropy, it cannot be further reduced. Hence, we can identify at most six independent constants describing an anisotropic, linear elastic material in two dimensions. For implementation purposes, vectorial notation is more convenient. As an example, the symmetric 2×2 stress tensor can be written as a 3-vector,

$$\sigma = \begin{bmatrix} \sigma_{xx} & \sigma_{yy} & \sigma_{xy} \end{bmatrix}^T \; .$$

An analogous elasticity matrix can be derived from the elasticity tensor (see e.g. [Bel00]) as

$$\mathbf{C} = \begin{bmatrix} \mathcal{C}_{1111} & \mathcal{C}_{1122} & \mathcal{C}_{1112} \\ & \mathcal{C}_{2222} & \mathcal{C}_{1222} \\ \text{sym.} & & \mathcal{C}_{1212} \end{bmatrix} . \tag{21}$$

As an example for textile simulation, one could use four of these constants: $\mathcal{C}_{1111}$ and $\mathcal{C}_{2222}$ are readily related to Young moduli E_u and E_v in the fabric's yarn directions *weft* and *warp*, which do not necessarily coincide with the coordinate directions x and y. Furthermore, $\mathcal{C}_{1212}$ is related to a shear modulus G and $\mathcal{C}_{1122}$ to transverse contraction coefficients ν_u and ν_v. With this material model, the stress-strain relationship can be written in matrix form as

$$\begin{bmatrix} \sigma_u \\ \sigma_v \\ \sigma_{uv} \end{bmatrix} = \frac{1}{1-\nu_u\nu_v} \begin{bmatrix} E_u & \nu_u E_v & 0 \\ \nu_v E_u & E_v & 0 \\ 0 & 0 & G(1-\nu_u\nu_v) \end{bmatrix} . \tag{22}$$

[‡] In this context, the formulation in terms of tensors is common but basically indicial, tensorial, and matrix notations are equivalent (see also [Bel00]).

[§] The elasticity tensor can mathematically be derived in terms of second order partial derivatives of a stored strain energy function (see [BW97]). The symmetry follows from the commutability of the mixed partial derivatives.

In a similar way to Eq. (22), the bending stress τ can be related to the bending strain γ as

$$\tau = \mathbf{C}_{bend}\gamma\,, \tag{23}$$

where $\mathbf{C}_{bend}$ models the bending properties of a specific material (see e.g. [VMT05]).

Because of its linearity and easy implementation, the material model described above is widely used for garment simulation. Beyond the small deformation range, the behaviour of most materials cannot reasonably be approximated by a linear model. In part 2 of the tutorial advanced constitutive laws are discussed, which capture fabric behaviour more accurately.

3.6. Spatial Discretisation with Linear Finite Elements

Having discussed constitutive models, we can now proceed with the discretisation of the governing equilibrium equations (14), respectively (15). We will use a formulation based on linear finite elements as an illustrative example [EKS03].

The finite element method is a procedure for the numerical solution of partial differential equations. The origins of this method can be found in structural mechanics, where the distribution of strains and stresses throughout a body in static equilibrium is sought. For the sake of brevity, we will not describe in detail how the final discrete equations used for implementation are derived. Instead, we will depart directly from the discrete setting. The reader interested in detailed derivations is referred to the standard text books [ZT00a] and [Bat96].

Assuming a decomposition of the domain into disjoint elements, the problem under consideration is formulated in terms of a displacement approach. This means, that for every node defined by the geometric decomposition, a displacement vector from its initial position is to be determined, such that the resulting final configuration is a solution to the elasticity problem. This process derives from a *variational principle*, the so called virtual work equation, which is mathematically rooted in variational calculus. In such a principle, an energy functional $\Pi = \Pi(\mathbf{u})$ is sought to be rendered stationary, in this case with respect to nodal displacements $\mathbf{u}$. In static elasticity problems, this functional is the sum of potential and strain energy[¶]. The functional is stationary if

$$\delta\Pi(\mathbf{u}) = 0\,, \qquad \text{for all variations } \delta\mathbf{u}\,. \tag{24}$$

From this, the (dynamic) virtual work equation can be derived as

$$\int_\Omega \delta\varepsilon : C(\varepsilon)d\Omega - \int_\Omega \delta u\; f\; d\Omega + \int_\Omega \delta u\; \rho\; \ddot{u}\; d\Omega = 0\,. \tag{25}$$

Here, the first term is due to the internal strain energy, the second one accounts for body forces per volume, and the last term represents inertia forces. This equation forms the basis for the subsequent finite element discretisation. An important observation on the way towards the discrete formulation is that, using further mathematical transformations, the virtual work equation per element can be expressed alternatively in terms of equivalent internal and external nodal forces. If we assume a linear relationship between displacement and strain, as well as between strain and stress, the problem can finally be expressed in the form of

$$\mathbf{K}\mathbf{u} = \mathbf{f}\,. \tag{26}$$

The matrix $\mathbf{K}$ is called the *stiffness matrix* and $\mathbf{f}$ is the vector of external forces per node. Visually, this equation states that in an equilibrium state the displacement of the nodes is such that the equivalent internal nodal forces (resulting from internal stress) exactly cancel the external forces. For dynamic problems, this equation has to be augmented by terms accounting for inertia and viscous forces. This results in a second order initial value problem which reads

$$\mathbf{M}\ddot{\mathbf{u}}(t) + \mathbf{D}\dot{\mathbf{u}}(t) + \mathbf{K}\mathbf{u}(t) + \mathbf{f} = 0\,, \tag{27}$$

where $\mathbf{M}$ is the mass matrix and $\mathbf{D}$ is the viscosity matrix. In this equation, the initial conditions $\mathbf{u}(0) = \mathbf{u}_0$ and $\dot{\mathbf{u}}(0) = \mathbf{v}_0$ are assumed, where $\mathbf{v}_0$ denotes the initial velocity. Note that the vector of nodal displacements $\mathbf{u} = \mathbf{u}(t)$ now depends on time. In the following, we will explain how to set up the stiffness matrix for a practical example, namely the plane stress analysis of two-dimensional elasticity. This particular case can be used as a basis for a continuum mechanics-based cloth simulator.

3.6.1. Plane Stress Analysis

Two of the most compelling advantages of the finite element method are its modularity and versatility. Once a general framework for the method has been laid out it can be applied to a broad range of problems. The specialisation on the actual problem follows through decisions on additional properties like material laws and, what is most important, the choice of an actual element type. As a concrete example, we will investigate the special case of plane stress analysis in this section. Being the simplest candidate, the linear triangular element with nodal displacement as only degrees of freedom will be used.

Plane stress analysis can be applied to elasticity problems that are inherently two-dimensional, i.e. where the in-plane deformation is predominant. This restricts the range of problems to settings which are essentially *flat* or which can at least be reasonably approximated with flat elements. Nevertheless, it is possible to use the plane stress assumption as a basis for cloth simulation (see [EKS03]). In the following, we will, for simplicity, assume that the object under consideration lies in the *xy*-plane. As the name already indicates, in the case of plane stress, the out-of-plane components of the stress tensor are zero, i.e. $\sigma_{iz} = \sigma_{zj} = 0$.

The geometry of the linear triangular element is given by

[¶] The extension to kinetic energy is omitted here for simplicity.

the coordinates of its three points $\mathbf{p}_1$, $\mathbf{p}_2$ and $\mathbf{p}_3$ (cf. Fig. 6). The three linear nodal shape functions are completely

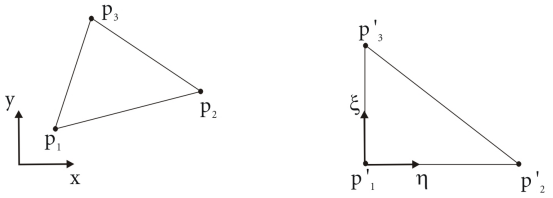

Figure 6: *Geometry of a triangular element (*left*) and its corresponding generic element (*right*).*

defined by the requirement

$$N_i(\mathbf{p}_j) = \delta_{ij} \; .$$

Further, the approximation of the displacement field over the element is uniquely defined by the nodal in-plane displacements $\tilde{\mathbf{u}}_x$ and $\tilde{\mathbf{u}}_y$ in the x and y-direction as

$$\mathbf{u} = \sum_i N_i \tilde{\mathbf{u}}_i .$$

The definition of the shape functions, which are depicted in Fig. 7, automatically ensures displacement continuity across elements. For this simple element, an explicit expression for

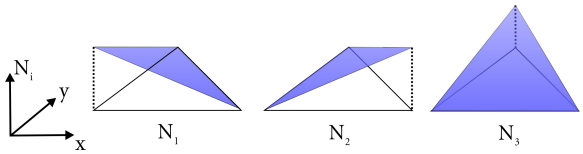

Figure 7: *The three linear shape functions of a triangular element.*

the shape functions in terms of the Cartesian coordinate can readily be derived. Nevertheless, it is instructive for the general case to take another approach. Notice that every triangular element can be transformed to a generic element as shown in Fig. 6. In this local space with coordinates ξ and η, the shape functions are trivially given by

$$N_1 = 1 - \xi - \eta, \quad N_2 = \xi, \quad N_3 = \eta \; .$$

Likewise, it can easily be verified that the shape function derivatives with respect to the local coordinates follow as

$$\frac{\partial N_1}{\partial \vartheta} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \quad \frac{\partial N_2}{\partial \vartheta} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \frac{\partial N_3}{\partial \vartheta} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \; ,$$

where $\vartheta = \begin{bmatrix} \xi & \eta \end{bmatrix}^T$ denotes the vector of local coordinates. For the subsequent formulations, the shape function derivatives with respect to the Cartesian coordinates are required. These are obtained by use of the chain rule as

$$\frac{\partial N_i}{\partial x_j} = \frac{\partial N_i}{\partial \xi}\frac{\partial \xi}{\partial x_j} + \frac{\partial N_i}{\partial \eta}\frac{\partial \eta}{\partial x_j} .$$

In practice, the necessary computation of

$$\left(\frac{\partial x}{\partial \vartheta}\right)^{-1}$$

can be accomplished without difficulty. As can be seen in the following, the shape function derivatives are indeed the only quantities actually needed in computation. Hence, there is no need for explicitly deriving the shape function expressions.

The next stage consists in formulating strain. In the case of moderately small deformations, a linear strain definition in terms of the displacement field is common. Over a single triangular element, the Cauchy strain is defined as

$$\varepsilon = \begin{bmatrix} \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \end{bmatrix} \mathbf{u} = \sum_{i=0}^{3} \begin{bmatrix} \frac{\partial N_i}{\partial x} & 0 \\ 0 & \frac{\partial N_i}{\partial y} \\ \frac{\partial N_i}{\partial y} & \frac{\partial N^i}{\partial x} \end{bmatrix} \tilde{\mathbf{u}}_i = \sum_{i=0}^{3} \mathbf{B}_i \tilde{\mathbf{u}}_i \qquad (28)$$

Once the strain is computed for an element, the stress follows by a simple linear relation (see Sec. 3.5).

Stiffness Matrix The point of departure for setting up the stiffness matrix is the elemental virtual work equation. Using some basic transformations, the integral term leading to the global stiffness matrix can be written in terms of elemental contributions as

$$\int_\Omega \mathbf{B}^T \mathbf{C}\mathbf{B}\mathbf{u} \; d\Omega + \mathbf{f} = 0 \sum_i \int_{\Omega_i} \mathbf{B}^T \mathbf{C}\mathbf{B}\mathbf{u}_i \; d\Omega_i \; ,$$

provided that $\Omega = \cup_i \Omega_i$ and $\Omega_i \cap \Omega_j = \emptyset$ for all $i \neq j$ holds. Hence, the stiffness matrix can be assembled in an element-wise manner

$$\mathbf{K}_{ij} = \sum_e \mathbf{K}^e_{ij}$$

where $\mathbf{K}^e_{ij}$ is the contribution of element e to the the entry of the global stiffness matrix linking nodes i and j. In the geometrically linear approach, the involved matrices $\mathbf{B}_i$ from equation (28) are constant over one element. Together with a linear elastic material law the above integral reduces to

$$\mathbf{K} = \sum_i \int_{\Omega_i} \mathbf{B}_i^T \mathbf{C}\mathbf{B}_i \; d\Omega = \sum_i \mathbf{B}_i^T \mathbf{C}\mathbf{B}_i t A_i$$

where t is the constant material thickness and A_i is the area of a triangular element.

Practical Considerations With the methods presented so far only static analysis is possible. However, the extension to dynamic simulation is not difficult as it consists mainly of the addition of inertia and viscous forces already mentioned in Eq. (16). The actual way in which this is accomplished depends on the actual numerical time integration scheme. It is worth noting that the stiffness matrix automatically supplies a means of evaluating internal forces in the current configuration. In this case a simple matrix-vector multiplication is sufficient. However, if elastic forces are not integrated implicitly (as in explicit schemes or certain variants

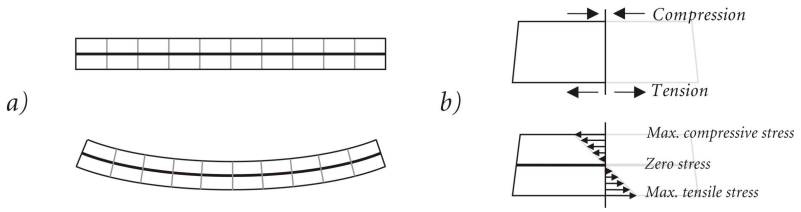

Figure 8: *Cross-sectional view of bending deformation with finite thickness.* a) *Bending leads to a combination of compressive (top layer) and tensile (bottom layer) deformations. The neutral axis (shown in bold) remains unstretched.* b) *Linearly varying stresses through the thickness imply the existence of a zero-stress axis, the* neutral axis*.*

of the Newmark algorithm [BMF03]), they can be computed directly without having to assemble the matrix itself.

4. The Importance of Bending in Cloth Simulation

Wrinkles and folds play an important role in the appearance of real textiles. The way in which they form depends mainly on the bending properties of the specific material type. While a remarkable amount of effort has been spent on precisely reproducing the in-plane forces, few existing models are concerned with an accurate and consistent way of modelling bending energy. Nevertheless, the characteristic folding and buckling behaviour of cloth highly depends on bending properties. The range of existing approaches to modelling bending is broad and we will investigate the most relevant methods in this section. From the field of engineering, the thin plate or thin shell equations, which will also be introduced in the following, are known to be an adequate approach to this problem. The associated minimisation problem includes fourth order derivatives with respect to the displacements, which in most of the frameworks are not readily computable. Therefore, a direct implementation of such energy-based methods is often avoided. However, many methods draw their inspiration from the theory of elastic beams or plates and we will therefore start with a review of the relevant matter.

4.1. Bending with Finite Thickness

Let us consider a thin plate and assume, for the moment, that the plate is cylindrically bent. In this case, we can – without loss of generality – restrict our investigations to a thin slice of the plate. Thus, we arrive at a geometry corresponding to the classical beam element (see e.g. [ZT00a]). In the cross-sectional view shown in Fig. 8 it can be seen that the bottom layer is stretched while the top layer has been compressed (cf. [Kee99]). We can reasonably assume that the maximum values of tension and compression occur on the boundary layers. If we further assume that the induced stresses vary monotonically between these maxima we arrive at an axis with zero stress, the so called *neutral axis* (see Fig. 8). is the primary parametrisation domain. In the following sections we will see that the theory of elastic beams and plates is essentially based on this dimension reduction.

4.2. Linear Elasticity of Beams and Plates

The simplest model corresponding to our interests is the one-dimensional, linear elastic beam. As we will see in section 4.3, many existing approaches to bending in cloth simulation rely on this model. In the corresponding elasticity problem, the central unknown is the lateral deflection w of the neutral axis. We begin the description by introducing the kinematic constraints which lead to the common model of the *Euler-Bernoulli* beam. In this model, the *Kirchhoff Assumptions* are used, i.e. lines that are initially normal to the neutral axis are supposed to remain straight (i.e. they do not bend), normal to the neutral axis, and unstretched. The deformed state of the beam can be described by the displacements u_0 and w_0 of the neutral axis and a rotation θ of the normal (see Fig. 9).

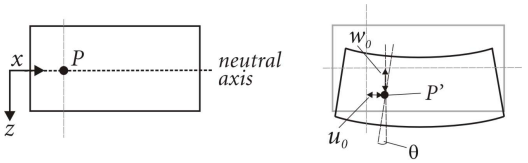

Figure 9: *Displacements u_0 and w_0 of the neutral axis and cross-sectional rotation θ for a deformed beam element.*

The horizontal and vertical displacements of any material point in the beam are given by

$$u(x,z) = u_0(x) - z\theta(x), \quad w(x,z) = w_0(x) \ ,$$

with the strains

$$\varepsilon_x = \frac{\partial u}{\partial x} = \frac{\partial u_0}{\partial x} - z\frac{\partial \theta}{\partial x} \ .$$

Because normal lines are assumed to remain unstretched, the strain ε_z in this direction can be neglected. Further, the requirement that normal lines remain perpendicular to the neutral axis implies

$$\theta = \frac{\partial w_0}{\partial x} \ . \tag{29}$$

Hence, we have for the transverse shear strain

$$\varepsilon_{xz} = \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} = -\theta + \frac{\partial w_0}{\partial x} = 0 \,. \tag{30}$$

With the strains defined, the stresses now follow with an appropriate constitutive law. If we assume a linear elastic material law we have

$$\sigma_x = \frac{E}{1-\nu^2}\varepsilon_x \,, \tag{31}$$

where E is Young's modulus and ν is Poisson's ratio. The bending moment around the horizontal axis is obtained as

$$M = D\frac{\partial \theta}{\partial x} = \frac{Eh^3}{12(1-\nu^2)}\frac{\partial^2 w_0}{\partial x^2} \,, \tag{32}$$

where the cubic thickness term $h^3/12$ is due to the second moment of area of the beam assuming a rectangular cross section. Note that for small deflections w_0, the term $\frac{\partial^2 w_0}{\partial x^2}$ is actually the curvature κ of the beam. This allows us to write the generalised stress-strain relationship of the Euler-Bernoulli beam in a clearer manner as $M = D\kappa$. The linear moment-curvature relationship is exploited by some approaches in cloth simulation to directly model bending forces (e.g. [VCMT95]).

In order to establish the governing equilibrium equations, we consider the forces acting on a differential beam element. In Fig. 10 loads and stress resultants on the beam are shown.

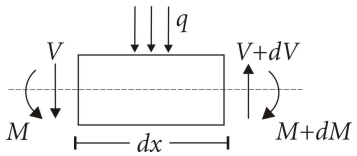

Figure 10: *Distributed lateral forces q, transverse shear force V, and bending moment M acting on a differential beam element of length dx.*

The beam is in equilibrium if the transverse internal force V (or shear resultant) and the external distributed load q are in balance. This leads to the equilibrium equation of the Euler-Bernoulli beam

$$\frac{Eh^3}{12(1-\nu^2)}\frac{\partial^4 w}{\partial x^4} = -q \,. \tag{33}$$

The above formulations directly carry over to cylindrically deformed plates. They can as well be translated to the general theory of (doubly curved) thin plates (see [ZT00b]). In engineering, thin plate elements are used to support lateral loads. Because curvature now occurs in both transverse directions, one speaks of the *neutral surface*, or simply *mid-surface*, in analogy to the neutral plane. Again, it is assumed that the stretch deformations of the mid-surface are negligible. Hence, the primary unknown is again the lateral deflection w. However, the deflection now varies in both x and y direction which renders the problem two-dimensional. For thin plates, the equilibrium equation is

$$\frac{Eh^3}{12(1-\nu^2)}\left(\frac{\partial^4 w}{\partial x^4} + 2\frac{\partial^4 w}{\partial x^2 \partial y^2} + \frac{\partial^4 w}{\partial y^4}\right) = -p \,. \tag{34}$$

This is a biharmonic equation involving fourth order partial derivatives. Investigating the corresponding strains it can be seen, that second order derivatives of the (lateral) displacement field are required. The thin plate equations have been used in computer graphics, too. For instance, they appear in a common minimisation problem from variational design (see e.g. [WW98]). Despite its demands on continuity, the thin plate approach can be used in physically based simulation. Since this theory does not take into account in-plane deformations it has to be augmented by an appropriate membrane model. This conjunction can be found in the class of *Kirchhoff-Love* thin shell theories (see Sec. 4.4).

4.3. Existing Approaches: From Springs to Shells

In this section we will report on bending models used in physically based simulation. In the seminal work of Terzopoulos et al. [TPBF87] a model for the animation of elastically deformable surfaces based on continuum mechanics is presented. The authors derive an elastic strain energy depending on the nonlinear metric and curvature tensors. The associated partial differential equations are discretised in space using finite differences on a regular quadrilateral grid. Although this approach is based on a physically sound theory, it was not widely adopted in the computer graphics community, due to its significant computational complexity. In the following years, approaches for the simulation of deformable surfaces like cloth mainly relied on particle and mass-spring systems. For modelling forces due to bending deformation, most of these methods use some kind of angular measure to approximate curvature. Breen et al. [BHW94] were among the first to use a coupled particle system in cloth simulation. The authors present an approach based on energy potentials for modelling the static drape of cloth. Departing from linear beam theory (see section 4.2), they first derive the bending energy between two successive edges in a rectangular discretisation. Curvature is approximated by fitting a circle through the three points involved in such a bending element. Using a biphasic curvature expression, Breen et al. model bending energy by approximating the nonlinear curves obtained from measurements with the *Kawabata Evaluation System* [Kaw80] numerically with quadratic fits. Once the energies are set up at the nodes, the gradients have to be computed to obtain nodal forces. The approach presented by Eberhardt et al. [EWS96] extends this work to the dynamic range. Computation times are greatly reduced through the use of sophisticated integration schemes. However, Eberhardt et al. do not approximate curvature but directly use the angle as a deformation measure.

Volino et al. [VCMT95] use a mass-spring system in-

spired by continuum mechanics. The basic bending element is formed by two adjacent triangles from the underlying unstructured grid (see Fig. 11). To determine curvature, a circle fitting inside the two triangles is found using the dihedral angle. The curvature over an element is then obtained as the inverse of the circle's radius. The authors point out that the curvature has to be limited to a certain maximum to prevent bending forces from reaching infinity. Using linear beam theory, the actual forces are deduced from the geometry of the involved triangles.

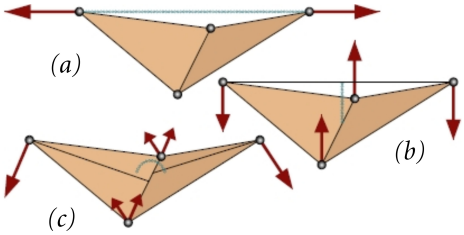

Figure 11: *Different ways of modelling bending forces on triangle meshes. Forces can be computed according to tensile crossover springs (a), based on the dihedral angle (b), or based on a weighted sum of vertex positions (c). (Fig. taken from [VMT06b])*

Baraff et al. [BW98] use the same basic bending element as in [VCMT95]. Following their proposed computational framework, a constraint expression for bending energy is derived. This essentially corresponds to an energy term which depends quadratically on the dihedral angle.

A different approach to cloth simulation was proposed by Eischen et al [EDC96, EB00]. Their method is based on the nonlinear shell theory derived by Simo et al. [SFR89a, SFR89b]. A four node bilinear element with nodal displacements and director rotations as the primary unknowns is used for discretisation (see [SFR89a]). In the context of shell theory, curvature is directly accessible through bending strains and does not have to be approximated otherwise. Like in [BHW94] Eischen et al. use measured data obtained from the Kawabata Evaluation System and use a 5th-order polynomial fit to approximate the curves. In sum, the approach leads to highly nonlinear equilibrium equations which have to be solved, for example, with the Newton-Raphson procedure. This solver is coupled with an adaptive arc length control to account for limit or bifurcation points in the solution due to buckling instabilities. The proposed method is limited to the static case and does not account for dynamic effects. For subsequent comparison, Eischen et al. [EB00] present a particle-based approach inspired from continuum mechanics. Like Breen et al. they use a regular quadrilateral discretisation but derive forces directly without explicit resort to energy potentials. However, their constructions are based on linear elasticity theory. Bending forces are derived using linear beam theory which again results in a linear moment-curvature relationship. The angle formed by two consecutive edges is taken as a direct measure for curvature. The authors state that the outcome of the two methods cannot be visually distinguished on the scale of the images they produced.

More recently, Choi et al. [CK02] proposed a bending model based on assumptions on the buckling behaviour of fabric. Starting from a standard quadrilateral mass-spring system, the basic bending element consists of an interleaved spring. The authors advocate that compressive in-plane forces on textiles lead to large out-of-plane deflections once a critical loading is reached. For the notoriously instable post-buckling state the buckled shape is predicted as a circular arc of constant length and curvature. With this assumption, the curvature can be computed analytically without any angle appearing. Hence, linear beam theory can be applied to derive the bending energy. Lastly, the authors derive expressions for force vectors and Jacobians at the nodes which allows the use of an implicit time integration scheme.

Bridson et al. [BMF03] proposed another derivation of bending forces for cloth simulation. Again, two adjacent triangles form the basic bending element. With the assumption that bending forces should neither cause in-plane deformation of the fabric nor lead to rigid body motions, they derive the directions and relative magnitudes for the four bending force vectors of an element. These vectors are then scaled with a bending stiffness constant and the sine of the dihedral angle (see Fig. 11, b). An additional scaling factor accounts for anisotropy of the mesh. For the numerical time integration, Bridson et al. suggest to use a mixed implicit-explicit integration scheme in which the (comparably small) bending forces can be handled in an explicit manner while viscous damping forces are treated implicitly. In this way computing the complicated derivatives of the bending forces is avoided.

In the bending model used by Etzmuß et al. [EKS03] forces are computed according to an approximation of the surface Laplacian. This idea is motivated by the fact that discrete mean curvature is closely related to the discrete surface Laplacian (see [MDSB03]). Using linear finite elements, the approximate Laplacian is computed for each element and projected onto the corresponding vertex normals. As usual, the element contributions are summed up to give the point wise Laplacian for every vertex.

While in most of the previous approaches curvature is approximated rather inaccurately, Grinspun et al. [GH*03] presented a method which is based on a sound curvature derivation. Their work extends existing cloth simulators to the range of objects for which the bending resistance is predominant. To this end, a discrete flexural energy potential is established using differential geometry. Again, the basic bending element consists of two adjacent triangles. The energy derives from an approximation to the squared difference of mean curvature in the current and initial configuration. The derivatives of the bending energy are intricate to com-

pute and hence the authors suggest the use of an automatic differentiation system.

Recently, Volino et al. [VMT06b] presented an approach which combines fairly good accuracy for representing quantitative bending stiffness with a simple and efficient computational procedure. In this method, a *bending vector* is computed that represents the bending of the surface through a simple linear combination of particle positions (see Fig. 11, c). This vector is then redistributed as particle forces according to the bending stiffness of the surface. It can be shown that this scheme preserves total translational and rotational momentum without the need of recomputing the distribution coefficients according to the current position of the particles. This leads to a very simple computation process which is entirely linear, and thus very well adapted to implicit numerical integration. In terms of computational simplicity and speed, this approach competes well with simpler spring models, since it requires only linear operations, which can conveniently be cast into matrix form. Additionally, the Jacobian of the bending forces is constant throughout the simulation, which is a considerable computational advantage when using implicit numerical integration methods. The authors demonstrated that the accuracy of their model is close to the more accurate normal-based method as used in, e.g, [BMF03]. Nevertheless, it does not require expensive operations like trigonometric functions or square root evaluations.

Assuming an inextensible surface and, thus, isometric deformations, Bergou et al. [BWH*06] derived a bending energy which depends quadratically on nodal positions. Similarly to [VMT06b], this leads to a computationally economic model with bending forces linear in positions and a constant associated Jacobian. An extensive analysis of this and other isometric bending models based on discrete curvature energies can be found in [WBH*07].

4.4. SubdivisionFE

As pointed out above, the thin shell equations are a good choice when physical accuracy is important. A corresponding finite element approach requires a C^1-continuous displacement field (to be exact the shape functions have to be in H^2). The main problem with this requirement is guaranteeing continuity across elements which usually necessitates the use of additional variables (e.g. nodal rotations and slopes). An elegant and convenient way to avoid this additional complexity is to use subdivision finite elements as a basis (see [COS00]). The following section gives a brief account of the approach presented in [TWS06], which utilizes this type of finite elements .

4.4.1. Thin Shell Mechanics

In the Kirchhoff-Love theory of thin shells the configuration mapping (2) is expressed in terms of the mid-surface parametrisation $\mathbf{x}(\theta^1,\theta^2)$ (see Fig. 12) as

$$\varphi(\theta^1,\theta^2,\theta^3) = \mathbf{x}(\theta^1,\theta^2) + \theta^3\mathbf{a}_3(\theta^1,\theta^2)\,, \qquad (35)$$

where θ^i denote curvilinear coordinates and $\mathbf{a}_3$ is the director field normal to the surface. In analogy to Eq. (2) we write

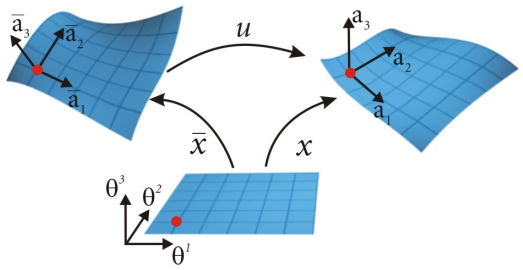

Figure 12: *A material point (red) on the shell's mid-surface with basis vector frame in the initial, reference, and current configuration (from left to right).*

$$\mathbf{x}(\theta^1,\theta^2) = \bar{\mathbf{x}}(\theta^1,\theta^2) + \mathbf{u}(\theta^1,\theta^2)\,. \qquad (36)$$

From this, tangential surface basis vectors can be defined as

$$\mathbf{a}_\alpha = \mathbf{x}_{,\alpha}\,, \qquad (37)$$

where the comma denotes partial differentiation. From hereon, we use Greek indices to denote variables, which can take on the values $\{1,2\}$ in order to distinguish from Latin indices, which can take on the values $\{1,2,3\}$. Moreover, the covariant tangent base vectors are given through differentiation of the configuration mapping as

$$\mathbf{g}_\alpha = \varphi_{,\alpha} = \mathbf{a}_\alpha + \theta^3\mathbf{a}_{3,\alpha} \qquad (38)$$

from which the surface *metric tensor* is derived as

$$g_{ij} = \mathbf{g}_i \cdot \mathbf{g}_j. \qquad (39)$$

Following Eq. (5) this leads to the definition of the Green strain

$$\varepsilon^G_{ij} = \frac{1}{2}(g_{ij} - \bar{g}_{ij}) = \alpha_{ij} + \theta^3\beta_{ij}\,, \qquad (40)$$

where α and β are membrane and bending strains, respectively. In the Kirchhoff-Love theory (cf. Sec. 4.2), the director $\mathbf{a}_3$ is assumed to stay normal to the surface, straight and unstretched,

$$\mathbf{a}_3 = \frac{\mathbf{a}_1 \times \mathbf{a}_2}{|\mathbf{a}_1 \times \mathbf{a}_2|}\,. \qquad (41)$$

Consequently, we have $\alpha_{3\beta} = \alpha_{\alpha 3} = 0$. The strains then simplify to

$$\alpha_{\alpha\beta} = \frac{1}{2}(\mathbf{a}_\alpha \cdot \mathbf{a}_\beta - \bar{\mathbf{a}}_\alpha \cdot \bar{\mathbf{a}}_\beta), \quad \beta_{\alpha\beta} = (\bar{\mathbf{a}}_{\alpha,\beta} \cdot \bar{\mathbf{a}}_3 - \mathbf{a}_{\alpha,\beta} \cdot \mathbf{a}_3)\,.$$

Departing from $\mathbf{a}_\alpha = \bar{\mathbf{x}}_{,\alpha} + \mathbf{u}_{,\alpha}$ and neglecting nonlinear terms, this can be recast to an expression which is linear in

displacements [COS00]. Resultant membrane and bending stresses follow as

$$n^{\alpha\beta} = \frac{\partial \Psi}{\partial \alpha_{\alpha\beta}}, \quad m^{\alpha\beta} = \frac{\partial \Psi}{\partial \beta_{\alpha\beta}}, \tag{42}$$

where Ψ is the strain energy density. The particular form of Ψ depends again on the specific material law used (see Sec. 3.5). As for the plane stress case, it is possible to use the corotational strain measure of Eq. (11), although the rotation field is determined in a slightly different way (see [TWS06]).

4.4.2. Subdivision-Based Finite Elements

Subdivision is a process for constructing smooth limit surfaces through successive refinement of an initial control mesh. As one of the most prominent examples Loop's subdivision scheme fulfils all prerequisites to serve as a basis for the thin shell discretisation. Besides the usual C^1-continuity inherent to subdivision surfaces, an important feature of this schemes is that the curvature of the resulting limit surface is L_2- or square integrable [RS01]. Due to this property, the subdivision basis functions can be used as shape functions for the FE-solution of the thin shell equations. In each step of this subdivision method, the positions of newly inserted nodes as well as those of old nodes are computed through a linear combination of vertices from the coarse mesh determined by the so called subdivision mask.

Only the immediate neighbours (i.e. the 1-ring) of a vertex have influence on this computation which gives rise to an efficient implementation. The process of subdivision itself can be considered as a linear operation and consequently be written in matrix form. It is therefore possible to directly derive properties like derivatives of the limit surface using an Eigenanalysis of the subdivision matrix. This yields simple expressions that can be computed efficiently. The resulting limit surface (or more generally, the limit field) can be evaluated at an arbitrary point in the interior of a patch, which is an important property for numerical integration. The key observation is that in regular settings (i.e. when all involved vertices have valence 6) Loop's scheme leads to generalised quartic box splines. In this case surface properties in one patch are completely defined by the 12 nodal values in the 1-neighbourhood (see Fig. 14) and the associated box spline basis functions N_i.
For instance, if we denote the local patch coordinates by θ^α, the limit surface can be expressed as

$$\mathbf{x}(\theta^1,\theta^2) = \sum_{i=1}^{12} N_i(\theta^1,\theta^2)\mathbf{x}_i, \tag{43}$$

where $\mathbf{x}_i$ are the nodal positions of the underlying mesh. In the same way, the displacement field interpolation is obtained from the nodal values. Additionally, differential quantities can be determined as

$$\mathbf{x}_{,\alpha}(\theta^1,\theta^2) = \sum_{i=1}^{12} N_{i,\alpha}(\theta^1,\theta^2)\mathbf{x}_i. \tag{44}$$

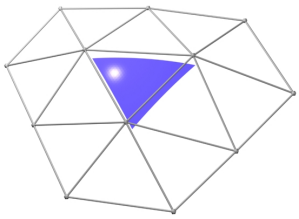

Figure 14: *1-ring neighbourhood of a regular patch consisting of 12 nodes.*

If the patch has an irregular vertex, the box spline assumption no longer holds and thus interior parameter points cannot be evaluated. For the following finite element discretisation, however, only quantities at the barycenter of the triangles are needed for integral evaluation. Hence, Cirak et al. required the initial mesh to have at most one irregular vertex per triangle. Then, after one subdivision step the barycenter lies again inside a regular patch (see Fig. 14). This process of subdivision and evaluation of the newly generated patch can again be expressed as a sequence of matrix multiplications.

Spatial Discretisation With the definition of the membrane and bending strains and assuming a linear elastic material (Eq. (17)) the internal energy from the virtual work equation (25) can be rewritten as

$$\int_\Omega \delta\varepsilon : C(\varepsilon) d\Omega = \int_\Omega \left(\delta\alpha^T \mathbf{H}_m \alpha + \delta\beta^T \mathbf{H}_b \beta\right) d\Omega, \tag{45}$$

where $\mathbf{H}_m$ and $\mathbf{H}_b$ are matrices corresponding to the membrane and bending part of the material law (see [COS00] for a complete derivation). Due to the linear strain interpolation, we have

$$\alpha(\theta^1,\theta^2) = \sum_i^N \mathbf{M}_i(\theta^1,\theta^2)\mathbf{u}_i, \quad \beta(\theta^1,\theta^2) = \sum_i^N \mathbf{B}_i(\theta^1,\theta^2)\mathbf{u}_i$$

for matrices $\mathbf{M}_i$ and $\mathbf{B}_i$ relating nodal displacements $\mathbf{u}_i$ to membrane and bending strain. This gives rise to a formulation of the complete system in the classical form of

$$\mathbf{K}\mathbf{u} = \mathbf{f} \tag{46}$$

with vectors of nodal displacement $\mathbf{u}$ and forces $\mathbf{f}$. The stiffness matrix $\mathbf{K}$ can be assembled in the usual element-wise fashion

$$\mathbf{K}_{ij} = \sum_e \int_{\Omega_e} \left(\mathbf{M}_i^T \mathbf{H}_m \mathbf{M}_j + \mathbf{B}_i^T \mathbf{H}_b \mathbf{B}_j\right) d\Omega = \sum_e \mathbf{K}_{ij}^e. \tag{47}$$

The integral in this equation can be evaluated using numerical quadrature. In this form, the above equations are only valid on regular patches. However, as mentioned above, in irregular settings one subdivision step is sufficient for evaluations at the barycenters. For a patch with an irregular vertex of valence N let $\mathbf{S}$ denote the the subdivision operator

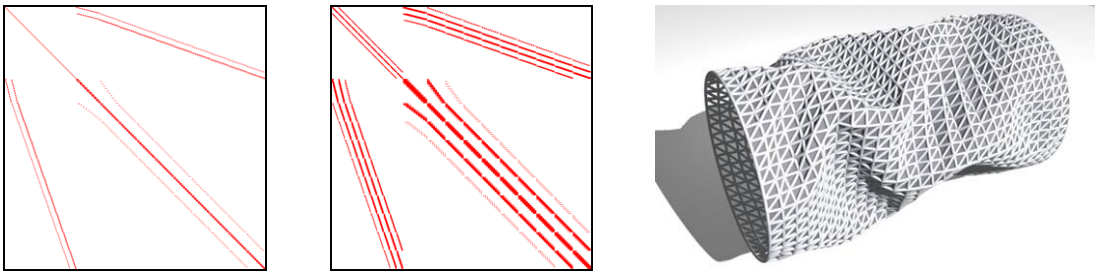

Figure 13: *A comparsion of the sparsity structures resulting from an ordinary finite element approach (*left*) and the subdivision FE-based method (*middle*) for a regularly tesselated mesh comprised of 1448 vertices (*right*).*

(see [COS00]). Further, let $\mathbf{P}$ be the projection operator extracting the 12 vertices corresponding to the central regular subpatch (Fig. 14, right). Then we can write

$$\mathbf{K}_{ij}^{e} = \int_{\Omega_e} \left[\mathbf{S}^T \mathbf{P}^T \left(\mathbf{M}_i^T \mathbf{H}_m \mathbf{M}_j + \mathbf{B}_i^T \mathbf{H}_b \mathbf{B}_j \right) \mathbf{P}\mathbf{S} \right] d\Omega \quad (48)$$

and thus simply include the conceptual subdivision step into the stiffness matrix.

To cover dynamic effects of moving and deforming objects inertia as well as viscous forces have to be included. This leads to a second order ODE in time analogous to Eq. (27). After transformation into a set of coupled first order ODEs, standard numerical time integration schemes can be applied (see [HE01]).

Discussion The setup process for the matrix $\mathbf{K}$ in Eq. (48) is very similar to a common FE-formulation. The evaluation cost for an element matrix in this approach, however, is slightly higher than for usual methods. This is mainly due to the increased connectivity of the elements: a regular patch contains a neighbourhood of 12 nodes where for standard approaches there are only three. This means that more entries have to be computed and written into the system matrix (see Fig. 13). As a consequence, the setup an resolution of the linear system is slower than for standard approaches like the one described in Sec. 3.6. To put this into the right context, processing times for collision detection and response can still be much larger. A striking advantage of this approach is that it alleviates the modelling of a broad range of different materials (see Fig. 15). For example, because the full curvature tensor is available (cf. Eq. (25)), bending properties can directly be controlled, allowing for complex anisotropic models. Since this approach is entirely based on the sound mathematical foundation of continuum mechanics, the simulation is largely independent of discretisation throughout a broad range of resolutions.

5. Parallel Techniques for Cloth Simulation

Up to this point we have mostly dealt with methods that were designed to exhibit as much visual realism as possible. Moreover, techniques for measuring and reproducing

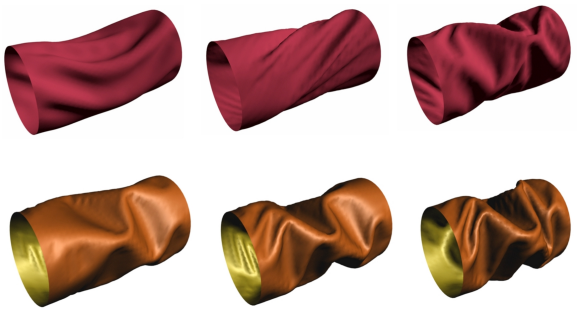

Figure 15: Top: *Different types of folds on a garment's sleeve generated using a typical textile material.* Bottom: *Sequence taken from an animation of axial compression of a cylinder with a metal-like material.*

real world materials in an accurate way were discussed. This realism and accuracy, however, comes at a price: the computational costs can be very high and run times for realistic scenarios are often excessive. Most of the computation time is spent on two stages, time integration and collision handling. In the following, we will therefore examine these two major bottlenecks, which are present in every physically-based cloth simulation system.

5.1. Implicit time integration

The ordinary differential equation resulting from the temporal discretisation of Eq. (27) are notoriously stiff. Due to stability reasons implicit schemes are widely accepted as the method of choice for numerical time integration (cf. [BW98]). Implicit schemes require the solution of a usually nonlinear system at each time step. As a result of the spatial discretisation, the matrix of this system is usually very sparse. There are essentially two alternatives for the solution of the system. One is to use an iterative method like the popular conjugate gradients (cg) algorithm [She94]. Another is to use direct solvers based on factorization. The cg-method is more popular in computer graphics as it offers much sim-

pler user interaction, alleviates the integration of arbitrary boundary conditions and allows balancing accuracy against speed. We will therefore focus on the cg-method in the following.

5.2. Collision Handling

For realistic scenes, the interaction of deformable objects with their virtual environment has to be modelled. This involves the detection of proximities (collision detection) and the reaction necessary to keep an intersection-free state (collision response). In the remainder, we refer to these two components collectively as *collision handling*. We usually distinguish between external collisions (with other objects in the scene) and self-collisions. For each of these types different variants of algorithms are usually used. Even with common acceleration structures, these algorithms are still computationally expensive. For complex scenarios with complicated self-collisions the collision handling can easily make up more than half of the overall computation time. It is therefore a second bottleneck for the physical simulation and hence deserves special attention.

Target Architecture The distinctive characteristic of parallel platforms is their memory architecture. In the following we will consider both distributed and shared address spaces. As exemplary representatives we choose clusters built from commodity components for the distributed memory setting and multi-core, multi-processor systems for the shared memory case. Although the basic ideas remain the same, the specific form and implementation of the actual algorithms depend on the target architecture. This is due to the fact that for different platforms, optimal performance is achieved in quite different ways as well. For example, one of the most important objectives on distributed memory architectures (DMA) is to minimize communication between the nodes of the cluster. While (inter-process) communication on shared address space architectures is not a costly aspect, care has to be taken for synchronization (including e.g. data access by multiple threads) as well. We will point out important differences at the appropriate places.

Programming Paradigms and Implementation Another distinguishing property of a given architecture is how it supports and favours different programming paradigms. On DMA machines a widely adopted strategy is that of single program multiple data (SPMD) where every node executes the same code but has different data to operate on. The communication in this case is usually message passing style, for example an implementation of the message passing interface (MPI). As a concrete example, the MPI-based numerical toolkit PETSc offers extensive support for SPMD style programming in scientific computation. Although it supports other approaches and can be run on many architectures, PETSC is especially well suited for large scale applications on DMAs. It can also be used in the shared memory setting although in this case, message passing is actually not necessary and synchronization can often be implemented more efficiently. Additionally, it is often desirable on these platforms to use a multi-threaded programming approach which is not supported by PETSc. However, most of the operating systems directly support this paradigm and there are numerous toolkits and frameworks offering convenient interfaces for thread-based programming. Another interesting alternative with growing acceptance is the OpenMP interface. This interface offers extensive support for exploiting loop-level parallelism. It is entirely based on compiler directives and is therefore highly portable. Moreover, the user does not have to care about thread accounting (i.e. creation, synchronization, termination) which makes this interface easy to use.

5.3. Parallel Solution of Sparse Linear Systems

As was stated above, one of the major computational burdens is due to the solution of a sparse linear equation system related which derives from implicit time integration. We assume that a sparse linear system of the form $\mathbf{Ax} = \mathbf{b}$ is to be solved up to some residual tolerance using the cg-method. The number of necessary iterations and therefore the speed of convergence depends on the condition number of the matrix $\mathbf{A}$. Usually, this condition number is improved using a preconditioning matrix $\mathbf{M}$ leading to a modified system

$$\mathbf{M}^{-1}\mathbf{Ax} = \mathbf{M}^{-1}\mathbf{b},$$

where $\mathbf{M}^{-1}\mathbf{A}$ is supposed to have a better condition number and $\mathbf{M}^{-1}$ is fairly easy to compute. The choice of an appropriate preconditioner is crucial because it can reduce the number of iterations substantially. The setup and solution of the linear system now breaks down to a sequence of operations in which (due to their computational complexity) the sparse matrix vector multiplication (SpMV) and the application of the preconditioner are most important. As a basis for the actual parallelization we will consider problem decomposition approaches subsequently.

5.4. Problem Decomposition

In the following explanations we use the compressed row storage (CRS) format for sparse matrices in which nonzero entries are stored in an array along with a row pointer and a column index array (see [Saa03]). The most intuitive (and abstract) way to decompose the SpMV operation into a number of smaller sub-problems is to simply partition the matrix into sets of contiguous rows. The multiplication can then be carried out in parallel among the sets. This simple approach has several disadvantages. First, the matrices we deal with are always symmetric (due to the underlying PDE). Hence, only the upper triangular part, including the diagonal, has to be stored. This leads to smaller memory requirements for the data as well as for the index structure. In its sequential version, the resulting numerical kernel is more efficient (cf.

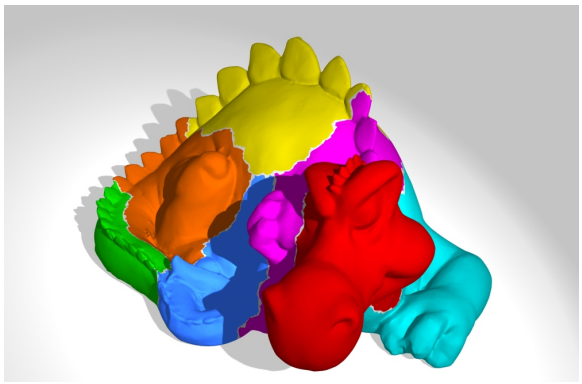

Figure 16: *The* Phlegmatic Dragon *model is decomposed into 8 disjoint regions (indicated by different colours). This partitioning serves as a basis for parallel processing.*

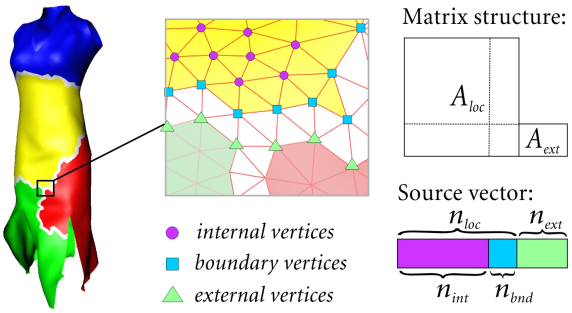

Figure 17: *Decomposition of a mesh into four disjoint partitions indicated by different colours. The associated vertex ordering leads to a special structure of the matrices and the source vector.*

[LVDY04]): it visits every matrix entry only once, performing both dot products and vector scalar products. However, the access pattern to the solution vector is not as local as for the non-symmetric case, i.e., entries from different sets need to be written by a single processor. The required synchronization would make a direct parallel implementation of the symmetric SpMV kernel inefficient. Another reason why the above decomposition is inadequate is that it does not take into account two other important components of linear system solution: matrix assembly and preconditioning. Methods based on domain decomposition are better suited for this case. They divide the input data geometrically into disjoint regions. Here, we will only consider non-overlapping vertex decompositions, which result in a partitioning P of the domain Ω into subdomains Ω_i such that $\Omega = \cup_i \Omega_i$ and $\Omega_i \cap \Omega_j = \emptyset$, for $i \neq j$. Decompositions can be obtained using graph partitioning methods such as Metis [KK96] in our case. An example of this can be seen in Fig. 16 and Fig. 17, which also shows a special vertex classification. This will be explained in the next section.

5.5. Parallel Sparse Matrix Vector Multiplication

Let $n_{i,loc}$ be the number of local vertices belonging to partition i and let V_i be the set of corresponding indices. These vertices can be decomposed into n_{int} internal vertices and n_{bnd} interface or boundary vertices, which are adjacent to n_{ext} vertices from other partitions (see Fig. 17). If we reorder the vertices globally such that vertices in one partition are enumerated sequentially we obtain again a partitioning of the matrix into a set of contiguous rows. The rows $a_{i,0}$ to $a_{i,n}$ of matrix A where $i \in V_i$ have the following special structure: the block $A_{i,loc}$ defined by $\{a_{lm} | l \in V_i, m \in V_i\}$ and lying on the diagonal of A is symmetric. The nonzero entries in this block describe the interaction between the local nodes of partition i. More specifically, this means that when nodes l and m are connected by an edge in the mesh, there is a nonzero entry a_{lm} in the corresponding submatrix of A. Besides this symmetric block on the diagonal there are further nonzero entries a_{le} where $l \in V_i$ is an interface node and $e \notin V_i$. These entries describe the coupling between the local interface nodes and neighbouring external nodes. The multiplication can be carried out efficiently in parallel if we adopt the following local vertex numbering scheme (cf. [Saa03]). The local vertices are reordered such that all internal nodes come before the interface nodes. For further performance enhancement, a numbering scheme that exploits locality (such as a self avoiding walk [OBHL02]) can be used to sort the local vertices. Then, external interface nodes from neighbouring partitions are locally renumbered as well. Let A_{ext} be the matrix which describes the coupling between internal and external interface nodes for a given partition. Notice that A_{ext} is a sparse rectangular matrix with n_{bnd} rows. With this setup the multiplication proceeds as follows

1. $y(0, n_{loc}) = A_{loc} \cdot x(0, n_{loc})$
2. $y(n_{int}, n_{loc}) += A_{ext} \cdot x_{ext}(0, n_{ext})$

The first operation is a symmetric SpMV, the second one is a non-symmetric SpMV followed by an addition. Both these operations can be carried out in parallel among all partitions. This decomposition is not only used for the SpMV kernel but also as a basis for the parallel matrix assembly, preconditioner setup and preconditioner application. Additionally, it can be implemented efficiently on both distributed and shared memory architectures.

5.6. Parallel Preconditioning

In order to make the cg-method fast, it is indispensable to use an efficient preconditioner. There is a broad variety of different preconditioners ranging from simple diagonal scaling (Jacobi preconditioning) to sophisticated multilevel variants. For the actual choice one has to weigh the time saved from the reduced iteration count against the cost for setup

and repeated application of the preconditioner. Additionally, one has to take into account how well a specific preconditioner can be parallelized. Unfortunately, designing efficient preconditioners is usually the most difficult part in the parallel cg-method [DHv93]. As an example, the Jacobi preconditioner is very simple to set up and apply even in parallel but the reduction of necessary iterations is rather limited. Preconditioners based on (usually incomplete) factorization of the matrix itself or an approximation of it are more promising. One example from this class is the SSOR preconditioner. It is fairly cheap to set up and leads to the solution of two triangular systems. For the sequential case, this preconditioner has proved to be a good choice in terms of efficiency [HE01]. However, parallelizing the solution of the triangular systems is very hard. Even if it is not possible to decouple the solution of the original triangular systems into independent problems we can devise an approximation with the desired properties. Let $\bar{A}$ be the block diagonal matrix with block entries $A_{ii} = A_{i,loc}$, i.e. the external matrices A_{ext} are dropped from A to give $\bar{A}$. Setting up the SSOR-preconditioner on this modified matrix leads again to the solution of two triangular systems. However, solving these systems breaks down to the solution of decoupled triangular systems corresponding to the $A_{i,loc}$ blocks on the diagonal. This means that they can be carried out in parallel for every partition. For reasons of data locality we use n smaller SSOR preconditioners constructed directly from the $A_{i,loc}$-blocks. Approximating A with $\bar{A}$ means a comparably small loss of information which in turn leads to a slightly increased iteration count. However, this increase is usually small compared to the speedup obtained through parallelization.

5.7. Optimizations

Besides the topics that were treated above a further aspect restricts the efficiency of a parallel implementation of the cg-method. Dense matrix multiplications usually scale very well since they have regular access patterns to memory and a high computational intensity. This is not true for the SpMV case. A typical SpMV algorithm using a matrix in the CRS format looks as follows:

Algorithm 1 Symmetric Spmv

```
1: for i = 1 to n_rows do
2:     start = ptr[i], end = ptr[i+1];
3:     for j = start to end do
4:         y[i] += dat[j] * x[ind[j]];
5:     end for
6: end for
```

It can be seen that the actual matrix data A_{dat} as well as the index structure *ind* is traversed linearly (with unit stride) while accesses to the vector's data x_{dat} occur totally arbitrary, i.e. the locality of these data accesses cannot be assumed. It can also be observed that the computational intensity per data element is rather modest. The performance of the SpMV algorithm is therefore mostly limited by memory bandwidth and cache performance. Because of this, it is important to improve data locality and thus cache performance. A good way to achieve this is to exploit the natural block layout of the matrix as determined by the underlying PDE: the coupling between two vertices is described by a 3x3 block – therefore nonzero entries in the matrix occur always in blocks. This blocked data layout already compensates for a lot of the inefficiency. An additional benefit can be achieved using single precision floating point data instead of double precision. This reduces the necessary matrix data (not including index structure) transferred from memory by a factor of two.

5.8. Parallel Collision Handling

From the parallelization point of view, the problem of collision handling differs substantially from the physical model. Collisions can be distributed very unevenly in the scene and their (typically changing) locations cannot be determined statically. This is why the naive approach of letting each processor care for the collisions of its own partition can lead to considerable processor idling, which seriously affects the overall parallel efficiency. Therefore, a dynamic problem decomposition is mandatory. Because bounding volume hierarchies (BVHs) are widely used for collision detection between deformable objects, it is reasonable to use this approach as a basis for a parallel implementation. As we have seen in a previous section, we can generally distinguish between two different types of collisions: external collisions and self collisions. To detect the first type we have to test our deformable object against every other (rigid or deformable) object in the scene. For the latter case, the deformable object has to be tested against itself. In this section, we will first lay down the basic algorithm for parallel collision handling. Subsequently, necessary adaption for both distributed and shared memory architectures will be addressed.

5.8.1. Basic Problem Decomposition

The recursive collision test of two BVHs can be considered as a depth-first tree traversal. For inducing parallelism, this procedure is implemented using a stack which holds individual tests of two BVs.

In the BVH testing procedure, expansion of a tree node results in two additional tree nodes each representing a refined BVH test. As in the sequential procedure, the first test is carried out by starting a new recursion level. However, before entering the recursion, the second test is pushed onto the stack. The traversal goes on downwards until a leaf is reached. Upward traversal begins by processing elements from the stack. In this way, all of the nodes in the tree are visited. Fig. 18 shows a snapshot of a BVH testing process along with the corresponding state of the stack. Note that, although we assume a binary tree in this example, an extension to general n-ary trees is possible (see [TPB07] for

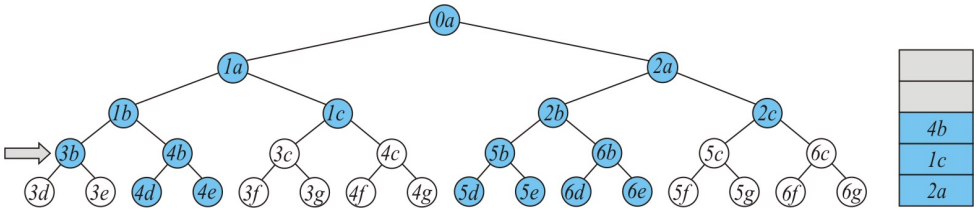

Figure 18: *Dynamic problem decomposition. The arrow indicates the current state of the BVH testing procedure. The stack on the right stores the root of untried branches.*

details). With a proper indexation of the hierarchies a single test can be represented using only a few indices and can thus be stored economically in appropriate data structures.

Tests which are recorded on the stack can be executed in one of the following ways:

- A test can be removed from the top of the stack and executed sequentially when the recursion gets back to the current level. Conceptually, this case is very similar to the procedure of the original algorithm.
- One or more tests can be removed from the bottom of the stack and executed by a newly generated (sub-)task. For each assigned test, this task executes a BVH testing procedure for which the considered test defines the root of a BVH testing tree.

In oder to prevent that tasks of too fine a granularity are generated, several tests can be removed at once from the stack and assigned to a single task. The actual decomposition and load balancing strategy depends on the specific platform and is discussed in subsequent paragraphs.

5.8.2. Distributed Memory

The first step for embedding the above algorithm into the SPMD framework is to construct a BVH. The physical modelling phase provides a decomposition of the input mesh into *local* meshes owned by nodes of the cluster. On each node, a BVH hierarchy is set up on this mesh using a standard top-down approach. Once this is done, the root nodes of the different processors can be combined to form a global hierarchy of the mesh. Since the structure of this hierarchy is kept constant throughout the simulation, this global hierarchy can safely be replicated once on every node. This replication late enables the location-independent execution of sub-tasks. While external collision can now be treated in the usual way, care has to be taken for self-collisions. Two different types of self-collisions have to be distinguished: collisions between sub-meshes of different processors and those that are real self-collisions on the processor-local mesh. For the latter case, existing techniques can be used since this corresponds to the usual self collision problem. For the case of inter-processor self-collisions, the corresponding BVHs are tested against each other, similar to the way standard collisions are treated.

All discussed BVH tests form a set of *top-level tasks* of the parallel collision handling method. For scenes where collisions are not uniformly distributed, the number of top-level tasks and also the amount of time to execute individual top-level tasks can differ considerably among the processors. Hence, a load balancing scheme which distributes the load among the processors evenly is mandatory.

Dynamic Problem Decomposition and Load Balancing As described above, every processor maintains a stack data structure where dynamically generating tasks are stored. For dynamic load balancing tasks must be transferred between processors at run-time. In distributed memory architectures, task transfers require explicit communication operations. Depending on the information associated with a task, task transfers can significantly contribute to the overall parallel overhead, especially when the computation to communication ratio of the tasks is poor. Finding a compact description of tasks is therefore crucial to minimize communication overhead. In our case, the cost for transferring a task is rather low.

Because the global BVH is replicated on every node, individual tests can be represented simply as an array of integers of the form $(obj1, dop1, obj2, dop2)$. The two BV hierarchies to be tested are identified by $obj1$ and $obj2$, and the root of the test is specified by $dop1$ and $dop2$.

The additional costs for building the copies of the BV hierarchies at the initialization phase are negligible since storage requirements are comparably low (say, only a few megabytes at max) even for complex scenes. The overhead during the course of the simulation is also insignificant: Updating hierarchies (at the beginning of every collision handling phase) requires one all-to-all broadcast operation to provide all processors the complete positions vector.

The dynamic load balancing process is responsible for triggering and coordinating task creation and task transfer operations in order to prevent that processors run idle. In order to achieve high scalability, a fully distributed scheme should be used. The distributed task pool model is a specifically well-suited basis. In this scheme every processor maintains a local task pool. Upon creation, a task is first placed in the local task pool. Subsequently, it can be instantiated

and executed locally, when the processor gets idle. It also might be transfered to a remote task pool for load balancing purposes. As discussed subsequently, task creation and task transfer operations are initiated autonomously by the processors.

To achieve a high degree of efficiency an (active) processor must be able to satisfy task requests from other processors immediately. Additionally, we must ensure that tasks with sufficient granularity are generated. In our case, this means that when a task is to be created, the stack must contain a minimum number of BVH tests which can be assigned to the task. In order to meet both requirements, we generate tasks in a proactive fashion, i.e. independently of incoming tasks requests. Tasks are generated (and placed in the local task pool) when the size of the stack exceeds a threshold value τ. Since task generation generally imposes an overhead (even when the new task is subsequently executed on the same processor) we increase τ linearly with the current size of the task pool σ: $\tau = a\sigma + b$. This simple heuristics enables us on the one hand to provide in a timely fashion tasks with a minimum granularity (determined by b) and ensures on the other hand that the overhead of additional task decomposition operations is compensated by an increased granularity of the resulting tasks. The parameter values a and b largely depend on the parallel architecture used and should be determined experimentally. Usually, choosing $a = 2$ and $b = 8$ delivers the best results.

A new task gets $\tau/2$ tests, where the tests are taken from the bottom of the stack. Tests are taken from the bottom of the stack for creating new tasks because such tests have a higher potential of representing a large testing tree since they originate closer to the root of the current testing tree.

For transferring tasks between task pools, a receiver initiated scheme is employed. When a processor runs idle and the local task pool is empty, it tries to steal tasks from remote pools. First, a victim node is chosen to which a request for tasks is sent. For selecting victims we apply a round robin scheme. If available, the victim transfers a task from its pool to the local pool. Otherwise the request is rejected by sending a corresponding message. In the latter case another victim node is chosen and a new request is issued.

For the actual implementation of the previously described methods, tools supporting distributed multithreading are needed. An efficient and convenient variant can be found in the parallel system platform DOTS [BKLwW99]. DOTS provides extensive support for the multithreading parallel programming model (not to be confused with the shared-memory model) which is particularly well suited for task-parallel applications that employ fully dynamic problem decomposition. In this way, the user does not have to care for task transfer and thread accounting explicitly, which is a great alleviation. Combining PETSc and DOTS, the collision handling algorithm can be implemented efficiently on DMAs. For a thorough performance analysis the interested reader is referred to [TB06, TB07].

5.8.3. Shared Memory

Compared to the approach for DMAs the basic parallelization strategy remains the same. Because communication is cheaper and thus dynamic collaboration between threads is less expensive, the shared memory setting enables us to set up heuristics exploiting temporal and spatial coherence. In this way, thread creation overhead can be controlled effectively.

Unlike in the distributed memory setting we do not have to care for load balancing explicitly. As long as there are enough threads ready for execution the scheduler will keep all cores busy. However, for problems with high irregularity, like parallel collision handling, it is generally impossible to precisely adjust the amount of logical parallelism to be exploited to the amount of available parallelism (i.e., idle processors). Especially on shared memory architectures, thread creation overhead can considerably contribute to the overall parallel overhead. Therefore, an over-saturation with threads has to be avoided as well.

In [TPB07] thread creation overhead is minimized on two levels. On the algorithmic level, a heuristics-based approach is used which prevents threads with too fine a granularity from being generated. On the implementation level, the process of thread creation and thread execution is decoupled. The next two paragraphs explain these optimizations in more detail.

Controlling Task Granularity For effectively controlling the granularity of a task, we need a good estimate of how much work corresponds to a certain task. The computational cost for carrying out a test in the collision tree is determined by the number of nodes in its subtree. Generally, this number is not known in advance. Because of the inherent temporal locality due to the dynamic simulation we can, however, exploit coherence between two successive time steps. After each collision detection pass we compute the number of tests in the respective subtree for every node in the collision tree using back propagation. This information is then used as a work estimate for tasks in the subsequent collision handling phase.

In this way, creating tasks with too small an amount of work can be avoided. Additionally, this information can be used to determine which tests should be carried out immediately. The error involved in the work estimation is usually very small. This can also be seen in Fig. 20 which shows a comparison of the estimated and the actual work load for 5.5 seconds of simulation for a representative test scene (see Fig. 19). The new task splitting scheme performs robustly and always keeps track with usual approaches. In most applications, this scheme even results in a significant performance

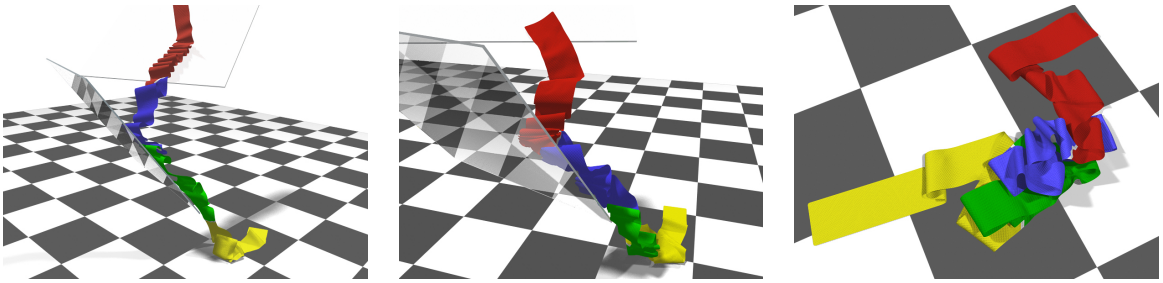

Figure 19: *Three shots from a representative test scene. A long (00.5m x 2.00m) ribbon consisting of 4141 vertices falls on two slightly inclined planes and slides onto the floor. Due to surface friction complex folds are formed as it slides over the planes. This again leads to complicated self-collisions which are reliably handled by the parallel collision handling algorithm.*

gain [TPB07]. This attests to the fact that temporal coherence in dynamic collision detection is a valuable source for performance improvements.

Figure 20: *Work estimate error for a typical test scene. The diagram shows the deviation from the actual amount of work over time in percent. Even in this very dynamic scene, the temporal coherence is high.*

Implementation For an implementation of the above algorithm, methods supporting the multithreaded programming are needed. Again, DOTS is an attractive candidate for this purpose. In order to ensure high performance on shared memory architectures, DOTS employs lightweight mechanisms for manipulating threads. Forking a thread results in the creation of a passive object, which can later be instantiated for execution. Thread objects can either be executed by a pre-forked OS native worker thread or can be executed as continuation of a thread which would otherwise be blocked, e.g., a thread reaching a synchronization primitive.

The above algorithm performs good and scales well even in challenging scenes (see Fig. 19). The actual speed-up that can be obtained depends, of course, on the specific scene and is in general the better the more work there is to be done in collision handling (see [TPB07]). A further aspect to note is the performance of the different task creation strategies. The naive stationary approach does not scale well when compared to the randomized version, which already shows quite a good performance. The work estimate approach performs very good and can, depending on the actual scenario, outperform the randomized version.

References

[Bat96] BATHE K.-J.: *Finite Element Procedures*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1996.

[Bel00] BELYTSCHKO T.: *Nonlinear Finite Elements for Continua and Structures*. John Wiley, 2000.

[BFA02] BRIDSON R., FEDKIW R. P., ANDERSON J.: Robust Treatment of Collisions, Contact, and Friction for Cloth Animation. In *Proceedings of ACM SIGGRAPH '02* (2002), ACM Press, pp. 594–603.

[BHW94] BREEN D., HOUSE D., WOZNY M.: Predicting the drape of woven cloth using interacting particles. In *Proceedings of ACM SIGGRAPH '94* (1994), ACM Press, pp. 365–372.

[BKLwW99] BLOCHINGER W., KÜCHLIN W., LUDWIG C., WEBER A.: An object-oriented platform for distributed high-performance Symbolic Computation. vol. 49, Elsevier, pp. 161–178.

[BMF03] BRIDSON R., MARINO S., FEDKIW R.: Simulation of clothing with folds and wrinkles. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA 2003)* (2003), ACM Press, pp. 28–36.

[BW97] BONET J., WOOD R. D.: *Nonlinear Continuum Mechanics for Finite Element Analysis*. Cambridge University Press, 1997.

[BW98] BARAFF D., WITKIN A.: Large steps in cloth simulation. In *Proceedings of ACM SIGGRAPH '98* (1998), ACM Press, pp. 43–54.

[BWH*06] BERGOU M., WARDETZKY M., HARMON

D., ZORIN D., GRINSPUN E.: A Quadratic Bending Model for Inextensible Surfaces. In *Fourth Eurographics Symposium on Geometry Processing* (Jun 2006), pp. 227–230.

[BWK03] BARAFF D., WITKIN A., KASS M.: Untangling cloth. In *Proceeding of ACM SIGGRAPH '03* (2003), ACM Press, pp. 862–870.

[CK02] CHOI K.-J., KO H.-S.: Stable but responsive cloth. In *Proceedings of ACM SIGGRAPH '03* (2002), vol. 21, pp. 604–611.

[CK05] CHOI K.-J., KO H.-S.: Research problems in clothing simulation. *Computer-Aided Design 37* (2005), 585–592.

[COS00] CIRAK F., ORTIZ M., SCHRÖDER P.: Subdivision surfaces: A new paradigm for thin-shell finite-element analysis. *Journal for Numerical Methods in Engineering 47* (2000), 2039–2072.

[DHv93] DEMMEL J., HEATH M., VAN DER VORST H.: Parallel numerical linear algebra. In *Acta Numerica 1993*. Cambridge University Press, 1993, pp. 111–198.

[EB00] EISCHEN J., BIGLIANI R.: Continuum versus particle representations. In *Cloth Modeling and Animation*, House D., Breen D., (Eds.). A. K. Peters, 2000, pp. 79–122.

[EDC96] EISCHEN J., DENG S., CLAPP T.: Finite-element modeling and control of flexible fabric parts. *IEEE Computer Graphics and Applications 16*, 5 (Sept. 1996), 71–80.

[EKS03] ETZMUSS O., KECKEISEN M., STRASSER W.: A Fast Finite Element Solution for Cloth Modelling. In *Proceedings of Pacific Graphics* (2003), pp. 244–251.

[EWS96] EBERHARDT B., WEBER A., STRASSER W.: A fast, flexible, particle-system model for cloth draping. *IEEE Computer Graphics and Applications 16*, 5 (Sept. 1996), 52–59.

[GH*03] GRINSPUN E., , HIRANI A., DESBRUN M., SCHRÖDER P.: Discrete shells. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA 2003)* (2003), ACM Press, pp. 62–67.

[GKJ*05] GOVINDARAJU N., KNOTT D., JAIN N., KABUL I., TAMSTORF R., GAYLE R., LIN M., MANOCHA D.: Interactive collision detection between deformable models using chromatic decomposition. In *Proceedings of ACM SIGGRAPH '05* (2005), ACM Press.

[Hau04] HAUTH M.: *Visual Simulation of Deformable Models*. Phd thesis, Wilhelm-Schickard-Institut für Informatik, University of Tübingen, Germany, July 2004.

[HB00] HOUSE D. H., BREEN D. E. (Eds.): *Cloth Modeling and Animation*. A K Peters, 2000.

[HE01] HAUTH M., ETZMUSS O.: A High Performance Solver for the Animation of Deformable Objects using Advanced Numerical Methods. In *Computer Graphics Forum* (2001), pp. 319–328.

[Kaw80] KAWABATA S.: The standardization and analysis of hand evaluation. *The Textile Machinery Society of Japan* (1980).

[Kee99] KEELER S.: The science of forming: A look at bending. *Metal Forming Magazine* (October 1999), 27–29.

[KHM*98] KLOSOWSKI J. T., HELD M., MITCHELL J. S. B., SOWIZRAL H., ZIKAN K.: Efficient collision detection using bounding volume hierarchies of *k*-DOPs. *IEEE Transactions on Visualization and Computer Graphics 4*, 1 (/1998), 21–36.

[KK96] KARYPIS G., KUMAR V.: *Parallel Multilevel k-way Partitioning Schemes for Irregular Graphs*. Tech. Rep. 036, Minneapolis, MN 55454, May 1996.

[LAM01] LARSSON T., AKENINE-MÖLLER T.: Collision detection for continuously deforming bodies. *Proceedings of Eurographics '01* (2001), 325–333.

[LVDY04] LEE B. C., VUDUC R. W., DEMMEL J. W., YELICK K. A.: Performance models for evaluation and automatic tuning of symmetric sparse matrix-vector multiply. In *ICPP '04: Proceedings of the 2004 International Conference on Parallel Processing (ICPP'04)* (2004), IEEE Computer Society, pp. 169–176.

[MDSB03] MEYER M., DESBRUN M., SCHRÖDER P., BARR A. H.: Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and Mathematics III*, Hege H.-C., Polthier K., (Eds.). Springer-Verlag, 2003, pp. 35–57.

[MKE03] MEZGER J., KIMMERLE S., ETZMUSS O.: Hierarchical Techniques in Collision Detection for Cloth Animation. *Journal of WSCG 11*, 2 (2003), 322–329.

[MTVW*05] MAGNENAT-THALMANN N., VOLINO P., WACKER M., THOMASZEWSKI B., KECKEISEN M.: Key Techniques for Interactive Virtual Garment Simulation. In *Proc. of Eurographics 2005, Tutorial 4* (2005).

[OBHL02] OLIKER L., BISWAS R., HUSBANDS P., LI X.: Effects of ordering strategies and programming paradigms on sparse matrix computations. *Siam Review 44:3* (2002).

[Pro95] PROVOT X.: Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *Proceedings of Graphics Interface (GI 1995)* (1995), Canadian Computer-Human Communications Society, pp. 147–154.

[Pro97] PROVOT X.: Collision and self-collision handling in cloth model dedicated to design garments. In *Proceedings of the Eurographics Workshop on Computer Animation and Simulation (CAS 1997)* (1997), Springer-Verlag, pp. 177–189.

[RS01] REIF U., SCHRÖDER P.: Curvature integrability of subdivision surfaces. *Advances in Computational Mathematics 14*, 2 (2001), 157–174.

[Saa03] SAAD Y.: *Iterative Methods for Sparse Linear Systems*, 2nd ed. SIAM, 2003.

[SFR89a] SIMO J. C., FOX D. D., RIFAI M. S.: On a stress resultant geometrically exact shell model. part i: Formulation and optimal parametrization. In *Computational Methods in Applied Mechanics and Engineering* (1989), vol. 72, pp. 267–302.

[SFR89b] SIMO J. C., FOX D. D., RIFAI M. S.: On a stress resultant geometrically exact shell model. part ii: The linear theory; computational aspects. In *Computational Methods in Applied Mechanics and Engineering* (1989), vol. 73, pp. 53–92.

[She94] SHEWCHUCK J. R.: An Introduction to the Conjugate Gradient Method Without the Agonizing Pain, 1994.

[TB06] THOMASZEWSKI B., BLOCHINGER W.: Parallel simulation of cloth on distributed memory architectures. In *Eurographics Symposium on Parallel Graphics and Visualization (EGPGV '06)* (2006).

[TB07] THOMASZEWSKI B., BLOCHINGER W.: Physically based simulation of cloth on distributed memory architectures. *Parallel Computing 33* (2007), 377–390.

[THM*05] TESCHNER M., HEIDELBERGER B., MANOCHA D., GOVINDARAJU N., ZACHMANN G., KIMMERLE S., MEZGER J., FUHRMANN A.: Collision Handling in Dynamic Simulation Environments. In *Eurographics Tutorials* (2005), pp. 79–185.

[TPB07] THOMASZEWSKI B., PABST S., BLOCHINGER W.: Exploiting parallelism in physically-based simulations on multi-core processor architectures. In *Eurographics Symposium on Parallel Graphics and Visualization (EGPGV '07)* (2007).

[TPBF87] TERZOPOULOS D., PLATT J., BARR A., FLEISCHER K.: Elastically deformable models. In *Proceedings of ACM SIGGRAPH '87* (July 1987), ACM Press, pp. 205–214.

[TWS06] THOMASZEWSKI B., WACKER M., STRASSER W.: A consistent bending model for cloth simulation with corotational subdivision finite elements. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA 2006)* (2006), Eurographics Association, pp. 107–116.

[van97] VAN DEN BERGEN G.: Efficient collision detection of complex deformable models using AABB trees. *Journal of Graphics Tools: JGT 2*, 4 (1997), 1–14.

[VCMT95] VOLINO P., COURCHESNE M., MAGNENAT-THALMANN N.: Versatile and efficient techniques for simulating cloth and other deformable objects. In *Proceedings of ACM SIGGRAPH '95* (1995), ACM Press, pp. 137–144.

[VMT94] VOLINO P., MAGNENAT-THALMANN N.: Efficient self-collision detection on smoothly discretized surface animations using geometrical shape regularity. In *Proceedings of Eurographics '94* (1994), Computer Graphics Forum, pp. 155–166.

[VMT01] VOLINO P., MAGNENAT-THALMANN N.: Comparing efficiency of integration methods for cloth simulation. In *Proceedings of Computer Graphics International 2001 (CGI 2001)* (2001), IEEE Computer Society, pp. 265–274.

[VMT05] VOLINO P., MAGNENAT-THALMANN N.: Accurate garment prototyping and simulation. *Computer-Aided Design & Applications 2*, 5 (2005), 645–654.

[VMT06a] VOLINO P., MAGNENAT-THALMANN N.: Resolving surface collisions through intersection contour minimization. In *Proceedings of ACM SIGGRAPH '06* (2006), vol. 25, ACM Press, pp. 1154–1159.

[VMT06b] VOLINO P., MAGNENAT-THALMANN N.: Simple linear bending stiffness in particle systems. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA 2006)* (2006), Eurographics Association, pp. 101–105.

[WBH*07] WARDETZKY M., BERGOU M., HARMON D., ZORIN D., GRINSPUN E.: Discrete Quadratic Curvature Energies. *Computer Aided Geometric Design (to* (2007).

[WW98] WEIMER H., WARREN J.: Subdivision schemes for thin plate splines. *Computer Graphics Forum 17*, 3 (1998), 303–314. ISSN 1067-7055.

[ZT00a] ZIENKIEWICZ O. C., TAYLOR R. L.: *The Finite Element Method. Volume 1: The Basis*, 5th ed. Butterworth Heinemann, 2000.

[ZT00b] ZIENKIEWICZ O. C., TAYLOR R. L.: *The Finite Element Method. Volume 2: Solid Mechanics*, 5th ed. Butterworth Heinemann, 2000.