Flexible Simulation of Deformable Models Using Discontinuous Galerkin FEM

Peter Kaufmann [a,*], Sebastian Martin [a], Mario Botsch [b], Markus Gross [a]

[a] *Computer Graphics Laboratory, ETH Zurich*

[b] *Computer Graphics Group, Bielefeld University*

Abstract

We propose a simulation technique for elastically deformable objects based on the discontinuous Galerkin finite element method (DG FEM). In contrast to traditional FEM, it overcomes the restrictions of conforming basis functions by allowing for discontinuous elements with weakly enforced continuity constraints. This added flexibility enables the simulation of arbitrarily shaped, convex and non-convex polyhedral elements, while still using simple polynomial basis functions. For the accurate strain integration over these elements we propose an analytic technique based on the divergence theorem. Being able to handle arbitrary elements eventually allows us to derive simple and efficient techniques for volumetric mesh generation, adaptive mesh refinement, and robust cutting. Furthermore, we show DG FEM not to suffer from locking artifacts even for nearly incompressible materials, a problem that in standard FEM requires special handling.

Key words: Physically-Based Simulation, Finite Element Methods, Discontinuous Galerkin

1. Introduction

Finite element methods (FEMs) have become an indispensable tool in computer graphics, where they are mostly used for physically-based simulation of deformable objects or fluids. Their solid mathematical foundation helps to achieve realistic simulation results, for instance in computer animation or surgery simulation.

In particular in computer graphics, FEM simulations are mostly based on tetrahedral or hexahedral meshes. While this allows for simple and efficient implementations, topological changes of the simulation domain require complex and error-prone remeshing to maintain a consistent simulation mesh. Dynamically adjusting the mesh is, however, of crucial importance in several simulation scenarios, such as fracture, interactive cutting in medical applications, or adaptive refinement of complex domains.

The use of more general polyhedral elements in FEM was recently shown to considerably simplify cutting and fracture simulations [Wicke et al., 2007; Martin et al., 2008]. However, the strict conformity constraints of standard FEM require comparatively

* Corresponding author.

Email addresses: peterkau@inf.ethz.ch (Peter Kaufmann), smartin@inf.ethz.ch (Sebastian Martin), botsch@techfak.uni-bielefeld.de (Mario Botsch), grossm@inf.ethz.ch (Markus Gross).

complex shape functions for those elements. In a slightly different context, the discontinuous element meshes of the PriMo framework enable adaptive mesh refinement for interactive shape deformation [Botsch et al., 2006, 2007]. Due to the missing physical accuracy this method is not directly useful for physically-based simulations though.

In this paper we propose a flexible and efficient simulation technique for corotated linear elasticity based on the *discontinuous Galerkin* finite element method (DG FEM) [Cockburn, 2003]. Our approach conceptually generalizes the aforementioned techniques, and overcomes their limitations by combining their respective strengths: Like standard continuous Galerkin FEM (CG FEM), the DG formulation is *physically accurate*, in the sense that under element refinement the approximation converges toward the exact solution of the involved PDE. Similar to PriMo, our DG approach supports *arbitrary polyhedral elements* and *discontinuous meshes* with weakly enforced continuity, thereby allowing for easy and flexible mesh restructuring.

In comparison to CG FEM, the increased flexibility of DG FEM enables adaptive refinement of mesh elements (*h-refinement*) and of the shape functions' polynomial degree (*p-refinement*) in a simple and efficient manner. Furthermore, in order to support flexible simulations of deformable models for Computer Graphics applications, we extend DG FEM by the following components:

- We simulate arbitrary polyhedral elements using simple and efficient polynomial basis functions and a fast and accurate volumetric integration technique (Section 5).
- We generalize stiffness warping to discontinuous polyhedral elements, thereby allowing linear strain measures to be used even in the presence of large deformations (Section 6).
- For embedded simulations we reconstruct from the discontinuous mesh a smooth displacement field based on moving least squares (MLS) interpolation (Section 7) and present a suitable collision handling technique (Section 8).

This paper is an extended version of the conference paper [Kaufmann et al., 2008], which enables us to introduce the fundamentals of DG FEM in more detail (Section 3) as well as to demonstrate the versatility of our approach on more examples, including slicing-based mesh generation, adaptive stress-based element refinement, flexible and efficient cutting, and locking analysis (Section 9).

2. Related Work

Starting with Terzopoulos et al. [1987], physically-based methods have been successfully employed for the simulation of deformable solids, thin shells, cloth, and fluids. The focus of this paper, and of the discussions in this section, is on deformable solids, and on the finite element method (FEM) as the underlying simulation scheme. For a more detailed survey of this topic we refer the reader to [Nealen et al., 2006].

2.1. Cutting & Fracture

Fracturing can efficiently be performed by restricting cuts to existing element boundaries [Müller and Gross, 2004], but this approach typically is not accurate enough for more sophisticated simulations. Splitting individual elements allows for precise fracturing and cutting, but in turn requires element decompositions [Bielser and Gross, 2000; Bielser et al., 2003] and/or general remeshing [O'Brien and Hodgins, 1999; O'Brien et al., 2002; Steinemann et al., 2006a]. When accommodating the crack surface, special care has to be taken to avoid numerically unstable sliver elements. Similarly, Bargteil et al. [2007] performed remeshing to remove degenerate elements during large plastic deformations.

Meshless approaches intrinsically avoid remeshing by using particles instead of a simulation mesh [Müller et al., 2004a]. While this considerably simplifies the actual topological changes, the material distance, which controls the mutual influence of simulation nodes, has to be adjusted. This can be accomplished either by recomputing special shape functions [Pauly et al., 2005] or by updating a distance graph [Steinemann et al., 2006b]. Note, however, that these approaches still require resampling in order to guarantee a sufficiently dense discretization in the vicinity of cracks and cuts.

A mesh-based alternative to remeshing is the virtual node algorithm [Molino et al., 2004], which, instead of splitting elements, duplicates them and embeds the surface in both copies. While the origi-

nal approach was limited to cutting each element at most three times, its recent generalization [Sifakis et al., 2007a,b] overcomes this restriction. Wicke et al. [2007] and Martin et al. [2008] avoid remeshing of cut elements into consistent tetrahedra by directly supporting general polyhedra in FEM simulations. The drawback of their methods, however, is the comparatively complex computation and integration of the employed generalized barycentric shape functions.

In the context of cutting and fracturing our approach is most similar to [Wicke et al., 2007; Martin et al., 2008], but it is more flexible and more efficient due to the use of simple polynomial shape functions.

2.2. Adaptive Simulation

The steadily growing complexity of geometric objects as well as of physical models results in an increasing demand for adaptive simulations, allowing to concentrate computing resources to interesting regions of the simulation domain [Debunne et al., 2001; Grinspun et al., 2002; Capell et al., 2002; Otaduy et al., 2007]. When adaptively refining the mesh, special care has to be taken to avoid or to properly handle hanging nodes.

This problem can be circumvented by subdividing basis functions instead of elements [Grinspun et al., 2002; Capell et al., 2002]. However, in order to ensure linear independence of basis functions, Grinspun et al. [2002] restrict the refinement to one level difference between neighboring elements. In contrast, the hybrid simulation [Sifakis et al., 2007b] allows for multi-level hanging nodes by constraining them to edges using either hard or soft constraints.

Another approach for reducing computational complexity is to embed a high resolution surface mesh into a coarser simulation mesh [Faloutsos et al., 1997; Capell et al., 2002; Molino et al., 2004; Müller and Gross, 2004; Müller et al., 2004b; James et al., 2004; Sifakis et al., 2007b]. The nodal displacements of the coarse mesh are then interpolated onto the surface mesh. A similar space deformation approach was employed for interactive shape deformation in [Botsch et al., 2007], where furthermore a discontinuous mesh with "glue-like" continuity energies allowed for easy and flexible mesh refinement.

Our method is based on DG FEM, and hence also employs discontinuous element meshes, with continuity being weakly enforced through penalty forces. This, in combination with the support for arbitrary elements, makes adaptive refinement both easy and efficient. Moreover, our smooth, MLS-based embedding technique works on arbitrary elements and provides higher smoothness compared to the typically employed barycentric interpolation.

2.3. Discontinuous Galerkin FEM

The basic idea of DG FEM, i.e., employing discontinuous shape functions and weakly enforcing boundary constraints and inter-element continuity through penalty forces, is rather old (see, e.g., [Babuška and Zlámal, 1973; Douglas and Dupont, 1976]). In the last decade, however, DG FEM regained increasing attention in applied mathematics [Arnold et al., 2001; Cockburn, 2003].

The main strength of DG FEM is its support for irregular, non-conforming meshes, and for shape functions of different polynomial degree, which in combination allows for flexible hp-refinement. In applied mathematics and mechanics, DG FEM has successfully been employed for linear and nonlinear elasticity (see, e.g., [Lew et al., 2004; Ten Eyck and Lew, 2006; Wihler, 2006]), where it was shown to provide an accuracy similar to CG FEM at comparable computational cost. Another advantage of DG FEM is the absence of locking even for nearly incompressible deformable objects [Wihler, 2006], which in CG FEM typically requires special handling [Irving et al., 2007]. Since physical accuracy is not the primary goal in most graphics applications, we resort to the physically plausible, robust, and efficient *co-rotated linear elasticity* [Müller and Gross, 2004; Hauth and Strasser, 2004].

To our best knowledge DG FEM has not been used in graphics before, besides the shorter conference version of this paper [Kaufmann et al., 2008]. We therefore first introduce the main concepts of DG FEM based on a simple 2D Poisson problem (Section 3), before deriving equations and techniques for 3D linear elasticity (Section 4). We further extend DG FEM by directly simulating arbitrary polyhedra (Section 5), by generalizing stiffness warping to DG FEM (Section 6), and by using embedded simulation (Section 7) with suitable collision handling (Section 8). Equipped with those techniques, we demonstrate the versatility of our framework on a set of different applications in Section 9.

3. Introduction to DG FEM

In this section we introduce the concepts of DG FEM and point out the main differences to standard CG FEM. For a better understanding we discuss both CG and DG FEM based on a simple 2D Poisson problem with homogeneous Dirichlet boundary constraints:

$$-\Delta u = f \quad \text{in } \Omega \subset \mathbb{R}^2, \quad u = 0 \quad \text{on } \partial\Omega. \tag{1}$$

For more details on either CG FEM or DG FEM we refer the interested reader to the textbooks [Bathe, 1995; Hughes, 2000] or the survey articles [Arnold et al., 2001; Cockburn, 2003], respectively.

3.1. CG FEM

The standard FEM approach is to multiply the above so-called *strong form* (1) by a suitable scalar test function v and to formally integrate by parts over the domain Ω. This yields the *weak form*

$$a_{\mathrm{CG}}(u, v) := \int_\Omega \nabla u \cdot \nabla v \;=\; \int_\Omega f v, \tag{2}$$

which is defined in terms of the bilinear form $a(\cdot, \cdot)$. The goal is to find a function u, such that the weak form (2) holds for all suitable test functions v vanishing on the boundary $\partial\Omega$.

In order to discretize (2) the domain Ω is partitioned into finite elements $K \in \mathcal{T}$. On top of this tessellation a set of basis functions $\{N_1, \ldots, N_n\}$ is defined and used to approximate u as

$$u(\boldsymbol{x}) \;\approx\; \sum_{i=1}^{n} u_i \, N_i(\boldsymbol{x}) \,. \tag{3}$$

For a weak form containing m'th partial derivatives, standard FEM requires basis functions N_i from the Sobolev space $H^m(\Omega)$. This in particular restricts the basis functions to be *conforming*, i.e., C^m continuous within and C^{m-1} continuous across elements [Hughes, 2000]. For our Poisson example with weak form (2) the N_i therefore have to be C^0 *continuous* across elements.

Approximating both u and v by the shape functions N_i and exploiting the bilinearity of $a(\cdot, \cdot)$ finally leads to the linear system

$$\mathbf{K} \cdot \begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix} \;=\; \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}, \tag{4}$$

with $\mathbf{K}_{ij} = a_{\mathrm{CG}}(N_i, N_j)$ and $f_i = \int_\Omega f \, N_i$, which is solved for the unknown coefficients u_i.

3.2. DG Primal Formulation

In contrast to the above approach, DG FEM allows for *non-conforming* or *discontinuous* shape functions N_i, thereby resulting in discontinuous approximations of u. The weak form will therefore first be formulated for each element $K \in \mathcal{T}$ individually, and those are to be combined by taking the discontinuities across neighboring elements into account. Before doing so, the second order PDE of the strong form (1) is split into two first order PDEs by introducing the helper function $\boldsymbol{\sigma} : \Omega \to \mathbb{R}^2$:

$$\boldsymbol{\sigma} = \nabla u \,, \quad -\nabla \cdot \boldsymbol{\sigma} = f. \tag{5}$$

To derive the weak form of a single element K, these two equations are multiplied by scalar- and vector-valued test functions v and $\boldsymbol{\tau}$, respectively. Integrating the result by parts over K yields additional boundary integrals over ∂K, leading to the *local* weak form of element K

$$\int_K \boldsymbol{\sigma} \cdot \boldsymbol{\tau} \;=\; -\int_K u \nabla \cdot \boldsymbol{\tau} \;+\; \int_{\partial K} u \, \boldsymbol{\tau} \cdot \boldsymbol{n}_K, \tag{6}$$

$$\int_K \boldsymbol{\sigma} \cdot \nabla v \;=\; \int_K f v \;+\; \int_{\partial K} v \, \boldsymbol{\sigma} \cdot \boldsymbol{n}_K, \tag{7}$$

where $\boldsymbol{n}_K$ denotes the unit outward normal of K.

The *global* weak form, which integrates over the whole domain Ω, is built by summing up the individual elements' weak forms (6), (7). Note that in CG FEM the boundary integrals over interior edges would cancel out, eventually leading to (2). In the DG setting, however, u and $\boldsymbol{\sigma}$ are *discontinuous* across elements, hence requiring special attention to be paid to the integrals over ∂K.

To account for that, the DG formulation replaces the functions u and $\boldsymbol{\sigma}$ in those boundary integrals by their so-called *numerical fluxes* $\hat{u}$ and $\hat{\boldsymbol{\sigma}}$, respectively. The fluxes are responsible for "gluing together" the functions u and $\boldsymbol{\sigma}$ across element boundaries, which is achieved by some penalty term that *weakly* enforces continuity. We will later present concrete examples for the fluxes $\hat{u}$ and $\hat{\boldsymbol{\sigma}}$.

For now they can be imagined as the average of the function values from both sides of the edge. This yields the global weak form

$$\int_\Omega \boldsymbol{\sigma} \cdot \boldsymbol{\tau} \;=\; -\int_\Omega u \nabla \cdot \boldsymbol{\tau} \;+\; \sum_{K \in \mathcal{T}} \int_{\partial K} \hat{u}\, \boldsymbol{\tau} \cdot \boldsymbol{n}_K, \quad (8)$$

$$\int_\Omega \boldsymbol{\sigma} \cdot \nabla v \;=\; \int_\Omega f v \;+\; \sum_{K \in \mathcal{T}} \int_{\partial K} v\, \hat{\boldsymbol{\sigma}} \cdot \boldsymbol{n}_K. \quad (9)$$

After introducing the fluxes $\hat{u}$ and $\hat{\boldsymbol{\sigma}}$, we can remove the helper function $\boldsymbol{\sigma}$ by choosing $\boldsymbol{\tau} = \nabla v$ in (8) and inserting the result into (9). Applying integration by parts once more then leads to

$$\int_\Omega \nabla u \cdot \nabla v \;+\; \sum_{K \in \mathcal{T}} \int_{\partial K} \left((\hat{u} - u)\, \nabla v - v\, \hat{\boldsymbol{\sigma}} \right) \cdot \boldsymbol{n}_K \;=\; \int_\Omega f v. \quad (10)$$

In the above equations each interior edge e, shared by two elements K^- and K^+, is integrated over twice, since $e \subset \partial K^-$ and $e \subset \partial K^+$. In order to exploit this, let us for a function q on e denote by

$$q^\pm \;:=\; q|_{\partial K^\pm}$$

its function value taken from either ∂K^+ or ∂K^-, respectively. With this we define the *average* operator $\{\cdot\}$ and the *jump* operator $[\![\cdot]\!]$ for scalar-valued functions u and vector-valued functions $\boldsymbol{\sigma}$ as

$$\{u\} := \frac{1}{2}\left(u^- + u^+\right), \quad [\![u]\!] := u^- \boldsymbol{n}^- + u^+ \boldsymbol{n}^+,$$

$$\{\boldsymbol{\sigma}\} := \frac{1}{2}\left(\boldsymbol{\sigma}^- + \boldsymbol{\sigma}^+\right), \quad [\![\boldsymbol{\sigma}]\!] := \boldsymbol{\sigma}^- \cdot \boldsymbol{n}^- + \boldsymbol{\sigma}^+ \cdot \boldsymbol{n}^+,$$

with "$\cdot$" denoting the vector dot product. Note that the average operator maps scalars to scalars and vectors to vectors, whereas the jump operator swaps these representations. With those operators, and with

$$\Gamma \;:=\; \cup_K \partial K \quad \text{and} \quad \Gamma^\circ \;:=\; \Gamma \setminus \partial\Omega$$

denoting the set of all edges and all interior edges, respectively, we can simplify (10) to

$$\begin{aligned} a_{\mathrm{DG}}(u,v) \;:=& \int_\Omega \nabla u \cdot \nabla v \quad (11)\\ &+ \boxed{\int_\Gamma \left([\![\hat{u} - u]\!] \cdot \{\nabla v\} - [\![v]\!] \cdot \{\hat{\boldsymbol{\sigma}}\} \right)}\\ &+ \boxed{\int_{\Gamma^\circ} \left(\{\hat{u} - u\} \cdot [\![\nabla v]\!] - \{v\} \cdot [\![\hat{\boldsymbol{\sigma}}]\!] \right)}\\ =& \int_\Omega f v. \end{aligned}$$

This equation is called the *primal formulation*, and is the DG equivalent to the CG weak form (2). It differs in the framed edge integrals only, which — with suitable fluxes $\hat{u}$ and $\hat{\boldsymbol{\sigma}}$ — penalize the discontinuities across elements, as discussed in the following.

3.3. DG Weak Form

The actual choice of numerical fluxes is where the various DG FEM methods differ, and it is an important design decision, since the fluxes determine important properties like consistency, symmetry, and stability of the FE method. In the following we will only present our choice of fluxes and shortly discuss its consequences. For an in-depth discussion of different fluxes and their respective properties we refer the reader to [Arnold et al., 2001].

Since fluxes are responsible for weakly enforcing inter-element continuity, i.e., for "gluing" neighboring elements, a straightforward approach is to penalize the squared jump $(u^- - u^+)^2$. This corresponds to the method of Babuška and Zlámal [1973], denoted by BZ, which employs the fluxes

$$\hat{u} := u|_K\,, \quad \hat{\boldsymbol{\sigma}} := -\eta\, [\![u]\!]\,.$$

Inserting the fluxes into (11) and simplifying the resulting equations by exploiting the identities

$$\{\{\cdot\}\} = \{\cdot\}\,, \quad \{[\![\cdot]\!]\} = [\![\cdot]\!]\,, \quad [\![\{\cdot\}]\!] = [\![[\![\cdot]\!]]\!] = 0,$$

leads to the weak form of the BZ method

$$\begin{aligned} a_{\mathrm{BZ}}(u,v) \;:=& \int_\Omega \nabla u \cdot \nabla v + \boxed{\int_\Gamma \eta_e\, [\![u]\!] \cdot [\![v]\!]} \quad (12)\\ =& \int_\Omega f v, \end{aligned}$$

which differs from the CG weak form (2) in the framed penalty term, being weighted by a scalar function $\eta_e = \eta\, \|e\|^{-1}$ inversely proportional to the edge length $\|e\|$. Analyzing the internal energy

$$a_{\mathrm{BZ}}(u,u) \;=\; \int_\Omega \nabla u \cdot \nabla v + \int_\Gamma \eta_e\, [\![u]\!] \cdot [\![u]\!]$$

reveals that the BZ method in fact penalizes the squared jump $[\![u]\!] \cdot [\![u]\!] = (u^- - u^+)^2$.

Just as in CG FEM, approximating u and v by shape functions N_i as in (3) leads to a linear system equivalent to (4), with matrix entries determined by $\mathbf{K}_{ij} = a_{\mathrm{BZ}}(N_i, N_j)$. The important difference is the contribution of edges, i.e., the framed integral over Γ in (12). Note that for *continuous* functions u and

v, as in the case of CG FEM, this integral would vanish, since then $[\![u]\!] = [\![v]\!] = 0$ and $v = 0$ on $\partial\Omega$, thereby reproducing the CG weak form (2).

The BZ method is geometrically intuitive and easy to implement. Moreover, it is *stable* in the sense that the stiffness matrix $\mathbf{K}$ is positive definite for any $\eta > 0$. However, as detailed in [Arnold et al., 2001], the method is *not consistent*: A continuous solution u of the problem might not satisfy the BZ weak form (12). Consequently, the approximate solution u does in general not converge toward the exact solution under element refinement.

A more accurate alternative is the so-called *interior penalty* (IP) method [Douglas and Dupont, 1976], whose fluxes are defined as

$$\hat{u} := \{u\}\,, \quad \hat{\boldsymbol{\sigma}} := \{\nabla u\} - \eta_e\,[\![u]\!]\,. \tag{13}$$

Inserting them into (11) and simplifying terms yields the IP weak form

$$\begin{aligned} a_{\mathrm{IP}}(u,v) \;&:= \int_\Omega \nabla u \cdot \nabla v \\ &\boxed{- \int_\Gamma \left([\![v]\!]\cdot\{\nabla u\} \;+\; [\![u]\!]\cdot\{\nabla v\} \;-\; \eta_e\,[\![u]\!]\cdot[\![v]\!]\right)} \\ &= \int_\Omega f v. \end{aligned} \tag{14}$$

The IP method uses three individual penalty terms in the framed Γ-integral:

- The first term ensures *consistency*: Any continuous solution u of the problem (1) also satisfies the DG weak form (14).
- The second term achieves *symmetry* of the bilinear form $a_{\mathrm{IP}}(u,v)$, and thereby also of the stiffness matrix $\mathbf{K}$.
- The last term ensures *stability*: For a sufficiently large penalty η it guarantees $a_{\mathrm{IP}}(u,u) > 0$, i.e., $\mathbf{K}$ to be positive definite.

The IP fluxes are one of few choices to yield a stable as well as consistent DG method. Consistency guarantees that if a *continuous* solution of either (2) or (14) exists in the space of shape functions N_i, then the IP method will find this solution as a function u with $[\![u]\!] = 0$ (cf. Fig. 1). Furthermore, due to consistency and stability the IP method converges under refinement towards the exact solution of the PDE, with a convergence rate determined by the degree of N_i [Arnold et al., 2001].

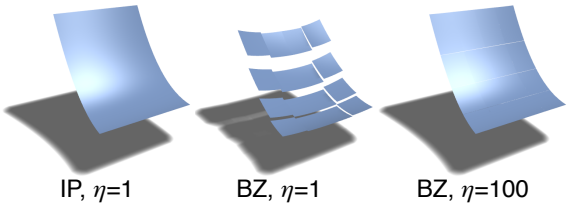

Fig. 1. Solution of $\Delta u = 2$, with Dirichlet boundary conditions corresponding to $u(x,y) = x^2$, using quadratic shape functions N_i on a (4×4) quad mesh. The *consistent* IP method finds the exact solution x^2 independently of the penalty η, since x^2 lies in the space of shape functions. This is not the case for the *inconsistent* BZ method, although increasing η improves the approximation by decreasing the jumps $[\![u]\!]$.

This leads to the main advantage of DG FEM: The missing conformity constraints allow simple polynomials $\{1, x, y, x^2, xy, \ldots, y^k\}$ of degree k to be used as basis functions for each element K. The convergence behavior of linear and quadratic DG basis functions is demonstrated in Fig. 2, which also shows bilinear CG FEM for comparison (see Section 9 for a discussion of these results).

While the use of polynomial basis functions already simplifies simulations on regular 2D grids, the true value of this added flexibility will be demonstrated in the following sections, where we discuss elasticity simulation on irregular 3D meshes of dynamically changing topology.

4. Linear Elasticity using DG FEM

After introducing the concepts of DG FEM on the 2D Poisson problem, we now generalize this approach to 3D linearly elastic deformations, starting with elastostatics.

In the following we consider a 3D object with material coordinates $\boldsymbol{x} = (x,y,z)^T \in \Omega$, which is to be deformed by a displacement vector field $\boldsymbol{u} : \Omega \to \mathbb{R}^3$. A detailed derivation of CG FEM for linear elasticity can be found in many textbooks (e.g., [Hughes, 2000]) and also in the recent survey [Nealen et al., 2006]. Hence, we refer the reader to the literature for details, and only give the equations required for the DG FEM derivation below.

We measure local deformations of the material using the linear Cauchy strain

$$\boldsymbol{\epsilon}(\boldsymbol{u}) = \frac{1}{2}\left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T\right),$$

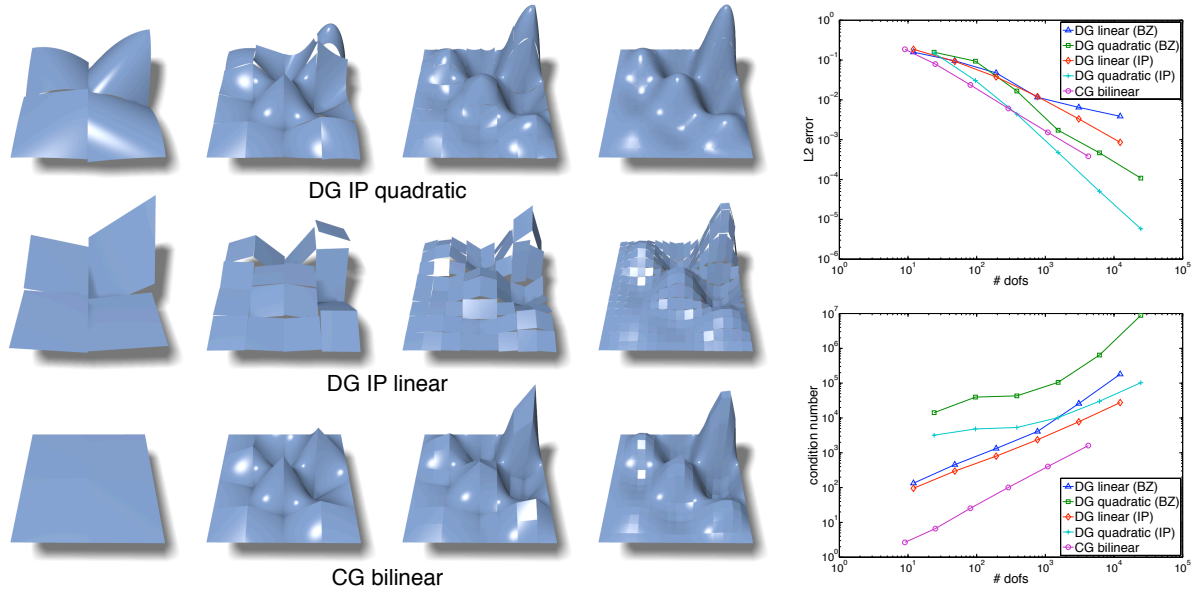

Fig. 2. Solution of the Poisson equation $-\Delta u = f$ on a regular quadrilateral grid of resolutions 2^2, 4^2, 8^2, and 16^2, using CG FEM and DG FEM. The plots compare the L^2 errors $\|\Delta u + f\|$ and the condition numbers of the stiffness matrix $\mathbf{K}$ for the BZ and IP method using linear and quadratic basis functions, and also include bilinear CG FEM as a reference.

which under the assumption of a Hookean material is linearly related to the stress

$$\boldsymbol{\sigma}(\boldsymbol{u}) \;=\; \mathbf{C} : \boldsymbol{\epsilon}(\boldsymbol{u}) \tag{15}$$

through a symmetric 4-tensor $\mathbf{C}$ containing material parameters. The colon operator ":" denotes the tensor product between two matrices $\mathbf{A}$ and $\mathbf{B}$ or between a matrix $\mathbf{A}$ and a 4-tensor $\mathbf{C}$ as

$$\begin{aligned}
\mathbf{A} : \mathbf{B} \;&:=\; \sum_{i,j} \mathbf{A}_{ij}\mathbf{B}_{ij}, \\
\mathbf{A} : \mathbf{C} \;&:=\; \sum_{i,j} \mathbf{A}_{ij}\mathbf{C}_{ijkl}, \\
\mathbf{C} : \mathbf{A} \;&:=\; \sum_{k,l} \mathbf{C}_{ijkl}\mathbf{A}_{kl}.
\end{aligned}$$

The dot operator "$\cdot$" denotes vector dot products $\boldsymbol{u} \cdot \boldsymbol{v}$ or matrix-vector products $\boldsymbol{A} \cdot \boldsymbol{v}$ and $\boldsymbol{v} \cdot \mathbf{A}$.

In static equilibrium the internal forces have to be in balance with the external forces $\boldsymbol{f}$, which is expressed by

$$-\nabla \cdot \boldsymbol{\sigma}(\boldsymbol{u}) \;=\; \boldsymbol{f}. \tag{16}$$

Equations (15) and (16), in combination with suitable boundary constraints on $\partial\Omega$, constitute the strong form of elastostatics. Multiplying them by test functions, integrating by parts over Ω, and combining the resulting equations yields the weak form of CG FEM:

$$a_{\mathrm{CG}}(\boldsymbol{u}, \boldsymbol{v}) := \int_\Omega \boldsymbol{\epsilon}(\boldsymbol{v}) : \mathbf{C} : \boldsymbol{\epsilon}(\boldsymbol{u}) \;=\; \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v}, \tag{17}$$

Analogous to (3) and (4), discretizing $\boldsymbol{u}$ as

$$\boldsymbol{u}(\boldsymbol{x}) \;\approx\; \sum_{i=1}^{n} \boldsymbol{u}_i N_i(\boldsymbol{x}) \quad \text{with} \quad \boldsymbol{u}_i \in \mathbb{R}^3 \tag{18}$$

leads to a $(3n \times 3n)$ linear system $\mathbf{K}\boldsymbol{U} = \boldsymbol{F}$ with

$$\begin{aligned}
\mathbf{K}_{ij} \;&=\; a_{\mathrm{CG}}(N_i, N_j) \cdot \mathbf{I}_3 \;\in \mathbb{R}^{3\times 3}, \\
\boldsymbol{U}_i \;&=\; \boldsymbol{u}_i \;\in \mathbb{R}^3, \\
\boldsymbol{F}_i \;&=\; \int_\Omega \boldsymbol{f}\, N_i \;\in \mathbb{R}^3,
\end{aligned}$$

where $\mathbf{I}_3$ denotes the (3×3) identity matrix.

4.1. DG Weak Form

The derivation of the DG weak form closely follows the procedure presented in Sections 3.2 and 3.3. Equations (15) and (16) are multiplied by test functions and integrated over each element K, yielding the individual elements' weak forms. Those are summed up, fluxes $\hat{\boldsymbol{u}}$ and $\hat{\boldsymbol{\sigma}}$ are introduced, and the two resulting equations are combined into one. The resulting equation corresponds to (10), and is to be simplified using the average and jump operators.

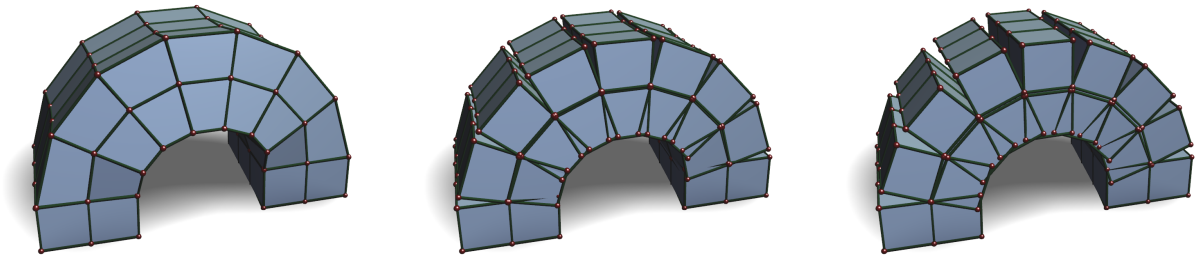

Fig. 3. Comparison of CG FEM (left), DG FEM (center), and the elastically coupled rigid cells of PriMo [Botsch et al., 2006, 2007] (right). The DG method conceptually spans the whole space from CG to PriMo, since for sufficiently large penalties η it approximates the CG results, and for an extremely stiff material and lower penalty η it reproduces the rigid cells of PriMo.

Those, however, have to be slightly redefined for vector-valued functions $\boldsymbol{u}$ and matrix-valued functions $\boldsymbol{\sigma}$ on a face f shared by two elements K^- and K^+, such that the jump operator maps vectors to matrices and vice versa. Using the outer product $\boldsymbol{u} \otimes \boldsymbol{n} := \boldsymbol{u}\,\boldsymbol{n}^T$ we define

$$\{\boldsymbol{u}\} := \frac{1}{2}\left(\boldsymbol{u}^- + \boldsymbol{u}^+\right), \quad [\![\boldsymbol{u}]\!] := \boldsymbol{u}^- \otimes \boldsymbol{n}^- + \boldsymbol{u}^+ \otimes \boldsymbol{n}^+,$$
$$\{\boldsymbol{\sigma}\} := \frac{1}{2}\left(\boldsymbol{\sigma}^- + \boldsymbol{\sigma}^+\right), \quad [\![\boldsymbol{\sigma}]\!] := \boldsymbol{\sigma}^- \cdot \boldsymbol{n}^- + \boldsymbol{\sigma}^+ \cdot \boldsymbol{n}^+.$$

Minimizing the jump $[\![\boldsymbol{u}]\!] : [\![\boldsymbol{u}]\!] = \|\boldsymbol{u}^- - \boldsymbol{u}^+\|^2$ by choosing the fluxes of [Babuška and Zlámal, 1973] leads to the weak form of the BZ method, which uses a_{BZ} instead of a_{CG} in (17):

$$a_{\mathrm{BZ}}(\boldsymbol{u}, \boldsymbol{v}) \;:=\; \int_\Omega \boldsymbol{\epsilon}(\boldsymbol{v}) : \mathbf{C} : \boldsymbol{\epsilon}(\boldsymbol{u}) \;+\; \int_\Gamma \eta_f \, [\![\boldsymbol{u}]\!] : [\![\boldsymbol{v}]\!]\,. \tag{19}$$

The penalty parameter η_f is defined per face f according to [Hansbo and Larson, 2002]:

$$\eta_f \;=\; \eta \cdot \mathrm{area}(f) \cdot \left(\frac{1}{\mathrm{vol}(K^-)} + \frac{1}{\mathrm{vol}(K^+)}\right) \tag{20}$$

using a global penalty parameter $\eta > 0$, which typically is in the order of 10^1–10^2 in our experiments.

The internal elastic energy of the deformed object can then be written as

$$a_{\mathrm{BZ}}(\boldsymbol{u}, \boldsymbol{u}) = \int_\Omega \boldsymbol{\sigma}(\boldsymbol{u}) : \boldsymbol{\epsilon}(\boldsymbol{u}) + \int_\Gamma \eta_f \left\|\boldsymbol{u}^- - \boldsymbol{u}^+\right\|^2,$$

which reveals an interesting connection to both CG FEM and the elastically coupled *rigid* cells of PriMo [Botsch et al., 2006]: CG computes elastic energies *within* elements only, using the Ω-integral, whereas PriMo employs only the "glue" energy *between* elements, represented by the Γ-integral.

Since BZ is based on *both* energy terms, with properly chosen penalty weight and material stiffness it can reproduce both methods, and can hence be considered as a generalization of them (cf. Fig. 3). As such, it combines the strengths of both approaches, since it inherits the physical accuracy of CG FEM, as well as the flexibility in element shapes and meshing of PriMo [Botsch et al., 2007], as we will demonstrate in Section 5.

The BZ penalty term is equivalent to both the glue energy of PriMo [Botsch et al., 2006] and the soft bindings employed by Sifakis et al. [2007b]. However, as discussed in Section 3.3 and shown in Figure 1, the BZ method is not *consistent* and therefore does not provide any convergence guarantees. Our experiments have nevertheless shown the BZ method to be very well suited for applications aiming at *physically plausible* deformations only.

However, if physical accuracy is important, other DG fluxes, such as those of the IP method, should be chosen instead. The weak form of the IP method [Douglas and Dupont, 1976] is defined by

$$\begin{aligned} a_{\mathrm{IP}}(\boldsymbol{u}, \boldsymbol{v}) := \; & \int_\Omega \boldsymbol{\epsilon}(\boldsymbol{v}) : \mathbf{C} : \boldsymbol{\epsilon}(\boldsymbol{u}) \\ & - \int_\Gamma \big([\![\boldsymbol{v}]\!] : \{\boldsymbol{\sigma}(\boldsymbol{u})\} \\ & \quad + [\![\boldsymbol{u}]\!] : \{\boldsymbol{\sigma}(\boldsymbol{v})\} - \eta_f \, [\![\boldsymbol{u}]\!] : [\![\boldsymbol{v}]\!] \big). \end{aligned} \tag{21}$$

Analogous to the Poisson problem (14), the three penalty terms ensure consistency, symmetry, and stability, and the method is guaranteed to converge under element refinement. Moreover, the IP method is still relatively easy to implement (see Section 4.2). While other (more complex) numerical fluxes exist (e.g., [Ten Eyck and Lew, 2006; Wihler, 2006]), for our applications the BZ and IP methods performed very well and have been fully sufficient.

4.2. Discretization & Matrix Assembly

To implement DG FEM for linear elasticity, we discretize $\boldsymbol{u}$ and $\boldsymbol{v}$ and set up the stiffness matrix $\mathbf{K}$. Since this is very similar to CG FEM, we refer the reader to [Hughes, 2000; Nealen et al., 2006] for more details on the following derivations.

The discretization (18) of $\boldsymbol{u}$ can be written in matrix notation as $\boldsymbol{u}(\boldsymbol{x}) = \mathbf{H}(\boldsymbol{x})\,\boldsymbol{U}$ using a $(3 \times 3n)$ interpolation matrix $\mathbf{H}(\boldsymbol{x})$ built from the basis functions $N_i(\boldsymbol{x})$, and a $3n$ vector $\boldsymbol{U}$ containing the unknown coefficients $\boldsymbol{u}_i \in \mathbb{R}^3$. Equivalently, the test function $\boldsymbol{v}$ can be represented as $\boldsymbol{v}(\boldsymbol{x}) = \mathbf{H}(\boldsymbol{x})\,\boldsymbol{V}$.

Moreover, we represent stress and strain by 6D vectors $\bar{\boldsymbol{\sigma}}$ and $\bar{\boldsymbol{\epsilon}}$ composed of the independent entries of the symmetric (3×3) matrices $\boldsymbol{\sigma}$ and $\boldsymbol{\epsilon}$, respectively. This leads to the matrix notation of the linear stress-strain relationship

$$\bar{\boldsymbol{\sigma}}(\boldsymbol{u}(\boldsymbol{x})) \;=\; \bar{\mathbf{C}}\,\bar{\boldsymbol{\epsilon}}(\boldsymbol{u}(\boldsymbol{x})) \;=\; \bar{\mathbf{C}}\,\mathbf{B}(\boldsymbol{x})\,\boldsymbol{U}, \qquad (22)$$

with a symmetric (6×6) matrix $\bar{\mathbf{C}}$ built from $\mathbf{C}$'s coefficients, and a $(6 \times 3n)$ matrix $\mathbf{B}(\boldsymbol{x})$ containing first derivatives of the N_i.

For the assembly of the stiffness matrix we use the above matrix notations to write the IP weak form (21) in terms of *element contributions* (Ω-integrals) and *face contributions* (Γ-integrals). Note that for the BZ method (19) only the last of the three face contributions in (21) is needed.

The element contributions are written in terms of element stiffness matrices $\mathbf{K}_K$ as in CG FEM:

$$\int_\Omega \boldsymbol{\epsilon}(\boldsymbol{v}) : \mathbf{C} : \boldsymbol{\epsilon}(\boldsymbol{u}) \;=\; \sum_{K \in \mathcal{T}} \boldsymbol{V}^T \mathbf{K}_K \boldsymbol{U} \qquad (23)$$

$$\text{with} \quad \mathbf{K}_K \;=\; \int_K \mathbf{B}^T(\boldsymbol{x})\,\bar{\mathbf{C}}\,\mathbf{B}(\boldsymbol{x})\,.$$

After expanding the operators $\{\cdot\}$ and $[\![\cdot]\!]$, and exploiting $\boldsymbol{n}^- = -\boldsymbol{n}^+$, the first two face contributions of $f = K^- \cap K^+$ have the form

$$[\![\boldsymbol{v}]\!] : \{\boldsymbol{\sigma}(\boldsymbol{u})\} \;=\; \left((\boldsymbol{v}^+ - \boldsymbol{v}^-) \otimes \boldsymbol{n}^+\right) : \frac{1}{2}\left(\boldsymbol{\sigma}^-(\boldsymbol{u}) + \boldsymbol{\sigma}^+(\boldsymbol{u})\right).$$

To write this in matrix notation, we need a matrix

$$\mathbf{N}_f := \begin{bmatrix} \boldsymbol{n}_x^+ & 0 & 0 & 0 & \boldsymbol{n}_z^+ & \boldsymbol{n}_y^+ \\ 0 & \boldsymbol{n}_y^+ & 0 & \boldsymbol{n}_z^+ & 0 & \boldsymbol{n}_x^+ \\ 0 & 0 & \boldsymbol{n}_z^+ & \boldsymbol{n}_y^+ & \boldsymbol{n}_x^+ & 0 \end{bmatrix}^T,$$

and differences and averages of matrices $\mathbf{B}$ and $\mathbf{H}$

$$\mathbf{H}_f^{[\![]\!]} := \left(\mathbf{H}_f^+ - \mathbf{H}_f^-\right), \quad \mathbf{B}_f^{\{\}} := \frac{1}{2}\left(\mathbf{B}_f^+ + \mathbf{B}_f^-\right),$$

which themselves are defined through restrictions

$$\mathbf{B}_f^\pm := \mathbf{B}|_{K^\pm}\,, \quad \mathbf{H}_f^\pm := \mathbf{H}|_{K^\pm}\,,$$

containing only the entries of $\mathbf{B}$ or $\mathbf{H}$ corresponding to basis functions of $K^\pm$.

With these matrices we can define three stiffness matrices $\mathbf{K}_{f1}$, $\mathbf{K}_{f2}$, and $\mathbf{K}_{f3}$ for each face f:

$$\begin{aligned} \mathbf{K}_{f1} &= \int_f -{\mathbf{H}_f^{[\![]\!]}}^T \mathbf{N}_f^T\,\bar{\mathbf{C}}\,\mathbf{B}_f^{\{\}}, \\ \mathbf{K}_{f2} &= \int_f -{\mathbf{B}_f^{\{\}}}^T \bar{\mathbf{C}}\,\mathbf{N}_f\,\mathbf{H}_f^{[\![]\!]}, \\ \mathbf{K}_{f3} &= \int_f \eta_f\,{\mathbf{H}_f^{[\![]\!]}}^T \mathbf{H}_f^{[\![]\!]}. \end{aligned}$$

The three face contributions in (21) can now be written in terms of these face stiffness matrices as

$$-\int_\Gamma [\![\boldsymbol{v}]\!] : \{\boldsymbol{\sigma}(\boldsymbol{u})\} \;=\; \sum_{f \in \mathcal{T}} \boldsymbol{V}^T \mathbf{K}_{f1}\,\boldsymbol{U}, \qquad (24)$$

$$-\int_\Gamma [\![\boldsymbol{u}]\!] : \{\boldsymbol{\sigma}(\boldsymbol{v})\} \;=\; \sum_{f \in \mathcal{T}} \boldsymbol{V}^T \mathbf{K}_{f2}\,\boldsymbol{U}, \qquad (25)$$

$$\int_\Gamma \eta_f\,[\![\boldsymbol{u}]\!] : [\![\boldsymbol{v}]\!] \;=\; \sum_{f \in \mathcal{T}} \boldsymbol{V}^T \mathbf{K}_{f3}\,\boldsymbol{U}. \qquad (26)$$

The global $(3n \times 3n)$ stiffness matrix $\mathbf{K}$ can therefore be assembled by doing one pass over all elements $K \in \mathcal{T}$ and accumulating their contributions $\mathbf{K}_K$, and a second pass over all faces $f \in \mathcal{T}$ that accumulates their contributions $\mathbf{K}_{fi}$. Equivalently to CG, the external force vector $\boldsymbol{F}$ is assembled from the elements' contributions $\int_K \mathbf{H}(\boldsymbol{x})^T\,\boldsymbol{f}$. Note that even for linear basis functions the integrands $\mathbf{H}(\boldsymbol{x})$ are not constant, requiring integration techniques as discussed in Section 5. The discretized weak form

$$\boldsymbol{V}^T \mathbf{K} \boldsymbol{U} \;=\; \boldsymbol{V}^T \boldsymbol{F}$$

has to hold for all test functions $\boldsymbol{v}$, i.e., all vectors $\boldsymbol{V}$, leading to the linear system $\mathbf{K}\boldsymbol{U} = \boldsymbol{F}$ to be solved for the *static* solution $\boldsymbol{U}$.

The assembly of element and face stiffness matrices into the global stiffness matrix $\mathbf{K}$ can be formulated most easily in terms of individual (3×3) matrices. Let $\mathbf{K}_{K[i,j]}$ denote the (3×3) submatrix of $\mathbf{K}_K$ corresponding to the global stiffness matrix entry $\mathbf{K}_{ij}$. The matrices $\mathbf{K}_{K[i,j]}$ can be precomputed

for all elements K using (23), and only those matrices that are non-zero need to be stored. Due to symmetry it holds that $\mathbf{K}_{K[i,j]} = \mathbf{K}_{K[j,i]}{}^T$, which further reduces the number of matrices to be precomputed. Similarly, $\mathbf{K}_{f1[ij]}$ and $\mathbf{K}_{f3[ij]}$ can be precomputed for all faces f. Note that $\mathbf{K}_{f3[ij]} = \mathbf{K}_{f3[ji]}{}^T$ and $\mathbf{K}_{f2[ij]} = \mathbf{K}_{f1[ji]}{}^T$, so $\mathbf{K}_{f2}$ does not need to be precomputed explicitly.

$\mathbf{K}_{ij}$ can now be defined in terms of element and face contributions as follows:

$$\mathbf{K}_{ij} = \sum_K \mathbf{K}_{K[ij]} + \sum_f \left(\mathbf{K}_{f1[ij]} + \mathbf{K}_{f1[ji]}{}^T + \mathbf{K}_{f3[ij]} \right).$$

Using a notation where the operator $\leftarrow$ denotes assembly into the global stiffness matrix for all K, f, this can equivalently be written as:

$$\begin{aligned} \mathbf{K}_{ij} &\leftarrow \mathbf{K}_{K[ij]} \\ \mathbf{K}_{ij} &\leftarrow \mathbf{K}_{f1[ij]} \\ \mathbf{K}_{ji} &\leftarrow \mathbf{K}_{f1[ij]}{}^T \\ \mathbf{K}_{ij} &\leftarrow \mathbf{K}_{f3[ij]} \end{aligned}$$

Dirichlet boundary constraints can be prescribed in DG FEM as weak or strong constraints. The latter simply removes some DOFs from the system, i.e., fixes the coefficients $\boldsymbol{u}_i$ for the corresponding N_i. Weak boundary conditions are imposed by appropriately defining averages and jumps at boundary elements. For a prescribed displacement $\boldsymbol{g}$ this means to define the function values on the "free" side of boundary faces $f \in \partial\Omega$ as

$$\begin{aligned} \boldsymbol{u}^- &:= \boldsymbol{g}, & \boldsymbol{\sigma}^-(\boldsymbol{u}) &:= \boldsymbol{\sigma}^+(\boldsymbol{u}), \\ \boldsymbol{v}^- &:= \mathbf{0}, & \boldsymbol{\sigma}^-(\boldsymbol{v}) &:= \boldsymbol{\sigma}^+(\boldsymbol{v}). \end{aligned}$$

Dynamic simulations of deformable objects with time-varying $\boldsymbol{U}(t)$ and $\boldsymbol{F}(t)$ require additional inertial and damping forces, resulting in the governing equations

$$\mathbf{M}\ddot{\boldsymbol{U}} + \mathbf{D}\dot{\boldsymbol{U}} + \mathbf{K}\boldsymbol{U} \;=\; \boldsymbol{F}, \tag{27}$$

with mass matrix $\mathbf{M}$ and damping matrix $\mathbf{D}$, equivalent to CG FEM [Nealen et al., 2006]. In order to guarantee stability we employ semi-implicit Euler time-integration, resulting in a sparse, symmetric, positive definite linear system to be solved for each time-step.

We compared two kinds of linear system solvers: preconditioned conjugate gradients [Saad and van der Vorst, 2000] and sparse Cholesky factorization [Toledo et al., 2003]. While both worked well in all our experiments, the Cholesky solver turned out to scale better to larger problems thanks to its quasi-linear asymptotic complexity, as also observed in [Botsch et al., 2005].

5. Arbitrary Polyhedral Elements

The main advantage of DG FEM is the possibility to use non-conforming, discontinuous shape functions N_i. This added flexibility allows us to employ simple degree-k polynomials $\{1, x, y, z, xy, \ldots, z^k\}$ as (non-nodal) basis functions within each element K. We used either 4 linear or 10 quadratic basis functions per element. Notice that k should be ≥ 1, since then the DG method can exactly reproduce rigid motions, yielding a linear, continuous displacement function $\boldsymbol{u}$ without jumps [Cockburn, 2003].

In contrast to nodal basis functions, these non-nodal basis functions no longer depend on the element shape, thereby enabling us to work with arbitrarily shaped elements. For practical reasons, however, we restrict ourselves to convex or non-convex polyhedra (i.e., planar faces and linear edges), which still is considerably more flexible than the convex polyhedra with triangulated faces of [Wicke et al., 2007]. Compared to the harmonic shape functions of [Martin et al., 2008], which also allow for non-convex elements, our polynomial basis functions are simpler and therefore more efficient to compute.

For a practical implementation we have to accurately and efficiently compute integrals of the form

$$\int_K N_a N_b, \quad \int_K \frac{\partial N_a}{\partial x_i}\frac{\partial N_b}{\partial x_j}, \quad \int_f N_a N_b, \quad \int_f \frac{\partial N_a}{\partial x_i} N_b,$$

over elements K and faces f, since they are the building blocks for the matrix assembly described in Section 4.2. While tetrahedra or hexahedra can be integrated analytically, general polyhedral elements typically require numerical integration, which trades accuracy for performance [Wicke et al., 2007].

In contrast, our polynomial basis functions can be integrated analytically over a polyhedron, which is exact up to numerical round-off errors. We use the divergence theorem for reducing the volume integral of a degree-k polynomial p_k over an element K to an area integral of a degree-$(k+1)$ polynomial p_{k+1} over its boundary ∂K, i.e., to a sum of integrals over its faces. Each face integral can in turn be reduced to line integrals over its edges $e \in \partial f$, which in the

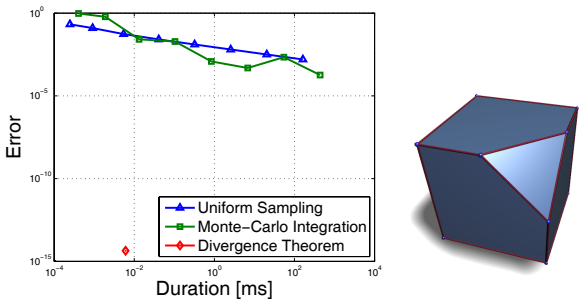

Fig. 4. Comparison of different numerical integration methods to our analytic method based on the divergence theorem. In this example the function $f(x, y, z) = x^2$ was integrated over the corner-cut cube model shown on the right.

end results in degree-$(k+3)$ polynomials in the edge endpoints.

The resulting expressions for polynomial basis functions can be (pre-)computed analytically. For linear and quadratic polynomials they are derived in detail by [Mirtich, 1996], who initially proposed this approach for accurately computing mass properties of polyhedra. Expressions for higher order polynomials can be derived accordingly. See Appendix A for a summary of how to find the polynomials in the edge endpoints for integrating an arbitrary polynomial over an element. These steps would typically be performed by a code generation tool.

Figure 4 compares our analytic integration to different numerical schemes in terms of accuracy and performance. While our method is exact up to round-off errors, it is also reasonably efficient: A straightforward numerical integration still shows an error of about 10^{-2} for the same computation time. Compared to CG FEM using the mean value polyhedral elements of [Wicke et al., 2007], our integration method is faster by an order of magnitude.

6. Stiffness Warping

Under large rotational deformations, linear FEM shows artifacts such as an unrealistic increase in volume. To avoid the cost of a full nonlinear simulation but still get physically plausible deformations in these cases, we employ a corotated formulation, which computes elastic forces in a rotated coordinate frame defined for each element [Müller and Gross, 2004; Hauth and Strasser, 2004].

In linear CG FEM, the forces acting on the nodes of an element K are computed from nodal displacements $\boldsymbol{U}$ and the element stiffness matrix $\mathbf{K}_K$ defined in (23) as follows:

$$\boldsymbol{F}_K \;=\; \mathbf{K}_K \boldsymbol{U} \;=\; \mathbf{K}_K \left(\boldsymbol{X} - \boldsymbol{X}^0 \right), \qquad (28)$$

with $\boldsymbol{X}$ and $\boldsymbol{X}^0$ denoting the deformed and undeformed nodal positions, respectively. In order to avoid the aforementioned rotational artifacts, the *corotational*, or *warped stiffness* approach [Müller and Gross, 2004; Hauth and Strasser, 2004] first reverts the element's rotation, computes displacements and forces in the un-rotated state, and re-rotates the resulting forces:

$$\boldsymbol{F}_K \;=\; \mathbf{R}_K \, \mathbf{K}_K \left(\mathbf{R}_K^T \boldsymbol{X} - \boldsymbol{X}^0 \right), \qquad (29)$$

where $\mathbf{R}_K$ is a block-diagonal matrix containing the (3×3) rotation matrix of element K on its diagonal.

This approach cannot be directly applied to DG for two reasons. First, the contributions resulting from integrals over interior faces are associated with two elements and require special treatment. Second, in case non-nodal basis functions are used, we will no longer be solving for nodal displacements, and $\boldsymbol{X}^0$ in (28) needs to be generalized to a set of degrees of freedom defining the undeformed state of the object in terms of the basis functions N_i.

6.1. Element and Face Contributions

Element contributions (23) can be treated just as in CG FEM using (29). We determine the rotations of general polyhedra by first fitting an affine transformation to the nodal displacements in the least squares sense, and then extracting its rotational component $\mathbf{R}_K$ using polar decomposition [Hauth and Strasser, 2004].

Note that for face contributions (24), (25), (26) we cannot simply apply (29) using the *face's rotation*, since that would lead to ghost forces and instabilities similar to the per-vertex stiffness warping of [Müller et al., 2002]. Moreover, the corotational method is only required to correct artifacts due to linear strain $\bar{\boldsymbol{\epsilon}} = \mathbf{B}\boldsymbol{U}$, and hence is not needed for (26).

For the face contributions (24) and (25) it is crucial that the strains $\mathbf{B}_f^+ \boldsymbol{U}$ and $\mathbf{B}_f^- \boldsymbol{U}$, which constitute $\mathbf{B}_f^{\{\}}$, are computed consistently with the strains of the element contributions (29) of K^+ and K^-. This requires to use the *elements' rotations* $\mathbf{R}_f^+$ and $\mathbf{R}_f^-$ for correcting $\mathbf{B}_f^+ \boldsymbol{U}$ and $\mathbf{B}_f^- \boldsymbol{U}$, respectively. We therefore have to split up the stiffness matrices $\mathbf{K}_{f1}$

and $\mathbf{K}_{f2}$ w.r.t. strain contributions from either K^+ or K^-, yielding the four stiffness matrices

$$\mathbf{K}^{\pm}_{f1} := -\frac{1}{2}\int_f \mathbf{H}^{[\![]\!]^T}_f \mathbf{N}^T_f \bar{\mathbf{C}}\, \mathbf{B}^{\pm}_f,$$
$$\mathbf{K}^{\pm}_{f2} := -\frac{1}{2}\int_f \mathbf{B}^{\pm T}_f \bar{\mathbf{C}}\, \mathbf{N}_f\, \mathbf{H}^{[\![]\!]}_f,$$

where $(\cdot)^{\pm}$ again denotes either $(\cdot)^+$ or $(\cdot)^-$. These stiffness matrices allow for a consistent warping of a face f's contributions, such that we get five corotated contributions:

$$\begin{aligned}
\boldsymbol{F}^{\pm}_{f1} &= \mathbf{R}^{\pm}_f \mathbf{K}^{\pm}_{f1}\left(\mathbf{R}^{\pm T}_f \boldsymbol{X} - \boldsymbol{X^0}\right),\\
\boldsymbol{F}^{\pm}_{f2} &= \mathbf{R}^{\pm}_f \mathbf{K}^{\pm}_{f2}\left(\mathbf{R}^{\pm T}_f \boldsymbol{X} - \boldsymbol{X^0}\right),\\
\boldsymbol{F}_{f3} &= \mathbf{K}_{f3}\left(\boldsymbol{X} - \boldsymbol{X^0}\right).
\end{aligned}$$

6.2. Non-Nodal Basis Functions

In order to use stiffness warping for non-nodal basis functions, we need to generalize the definition of the vector $\boldsymbol{X}^0$ representing the undeformed state. To this end, we have to find $\boldsymbol{X}^0 = (\boldsymbol{x}^0_1, \ldots, \boldsymbol{x}^0_n)$ satisfying the identity $\sum_i \boldsymbol{x}^0_i N_i(\boldsymbol{x}) \equiv \boldsymbol{x}$. For nodal basis functions, this vector would contain the nodal positions of the undeformed mesh. Since for each element K our non-nodal basis functions always contain the linear polynomials (cf. Section 5), finding $\boldsymbol{X}^0$ is trivial. For each element K, if its linear basis functions are

$$N_{i_K}(\boldsymbol{x}) = x, \quad N_{j_K}(\boldsymbol{x}) = y, \quad N_{k_K}(\boldsymbol{x}) = z,$$

we simply set the corresponding coefficients to

$$\boldsymbol{x}^0_{i_K} = (1,0,0)^T,\ \boldsymbol{x}^0_{j_K} = (0,1,0)^T,\ \boldsymbol{x}^0_{k_K} = (0,0,1)^T,$$

and use $\boldsymbol{x}^0_{l_K} = (0,0,0)^T$ for all its other basis functions. This results in a vector $\boldsymbol{X}^0$ representing the undeformed state, based on which stiffness warping can be performed just as for nodal basis functions.

Note that for quadratic or higher order basis functions, stiffness warping only removes the global element rotation, whereas local rotations due to bending might remain. While this was not a problem in all our experiments, such cases can easily be detected and the respective elements can be refined (see Section 9). We used stiffness warping for all 3D examples shown in this paper, and only provide a comparison to non-warped linear elasticity in the accompanying video.

6.3. Warped Assembly

To formulate the assembly of the stiffness matrix (cf. Section 4.2) in the presence of stiffness warping, we need to first split up (29) into a term proportional to $\boldsymbol{U}$ and a static force term as follows:

$$\mathbf{R}_K\, \mathbf{K}_K\, \mathbf{R}^T_K\, \boldsymbol{U} = \boldsymbol{F}_K + \mathbf{R}_K\, \mathbf{K}_K\, (\mathbf{I}_{3n} - \mathbf{R}^T_K)\boldsymbol{X}^0.$$

Every time the element rotations change, the warped element contributions are to be re-assembled as follows:

$$\begin{aligned}
\mathbf{K}_{ij} &\leftarrow \mathbf{R}_K\, \mathbf{K}_{K[ij]}\, \mathbf{R}^T_K\\
\boldsymbol{F}_i &\leftarrow \mathbf{R}_K\, \mathbf{K}_{K[ij]}\, (\mathbf{I}_3 - \mathbf{R}^T_K)\boldsymbol{X}^0_j
\end{aligned}$$

Note that contrary to the notation used previously, $\mathbf{R}_K$ and $\mathbf{R}^{\pm}_f$ denote (3×3) rotation matrices here. The face contributions are assembled as follows:

$$\begin{aligned}
\mathbf{K}_{ij} &\leftarrow \mathbf{R}^{\pm}_f\, \mathbf{K}^{\pm}_{f1[ij]}\, \mathbf{R}^{\pm T}_f\\
\boldsymbol{F}_i &\leftarrow \mathbf{R}^{\pm}_f\, \mathbf{K}^{\pm}_{f1[ij]}\, (\mathbf{I}_3 - \mathbf{R}^{\pm T}_f)\boldsymbol{X}^0_i\\
\mathbf{K}_{ji} &\leftarrow \mathbf{R}^{\pm}_f\, \mathbf{K}^{\pm}_{f1[ij]}{}^T\, \mathbf{R}^{\pm T}_f\\
\boldsymbol{F}_j &\leftarrow \mathbf{R}^{\pm}_f\, \mathbf{K}^{\pm}_{f1[ij]}{}^T\, (\mathbf{I}_3 - \mathbf{R}^{\pm T}_f)\boldsymbol{X}^0_j\\
\mathbf{K}_{ij} &\leftarrow \mathbf{K}_{f3[ij]}
\end{aligned}$$

7. MLS-Based Surface Embedding

When it comes to the simulation of complex models, a common approach for keeping computation costs low is to embed a high resolution surface mesh into a lower resolution simulation mesh. The latter can be simulated efficiently, and its displacement field $\boldsymbol{u}(\boldsymbol{x})$ is used to deform the surface mesh (see, e.g., [Faloutsos et al., 1997; Müller et al., 2004b; James et al., 2004; Sifakis et al., 2007b]). In DG FEM, the *discontinuous* displacement $\boldsymbol{u}$ cannot be applied directly to the high resolution surface, since it would lead to gaps and self-intersections.

To remove the discontinuities we first stitch the simulation mesh by averaging, for each node $\boldsymbol{x}_i \in \mathcal{T}$, the different displacements $\boldsymbol{u}|_K(\boldsymbol{x}_i)$ corresponding to its incident elements $K \in \mathcal{N}_i$, similar to [Botsch et al., 2006]:

$$\tilde{\boldsymbol{u}}_i = \frac{1}{|\mathcal{N}_i|}\sum_{K\in\mathcal{N}_i} \boldsymbol{u}|_K(\boldsymbol{x}_i)\,. \tag{30}$$

This results in a deformed, *continuous* simulation mesh, which is sufficient for visualizing the simulation mesh itself.

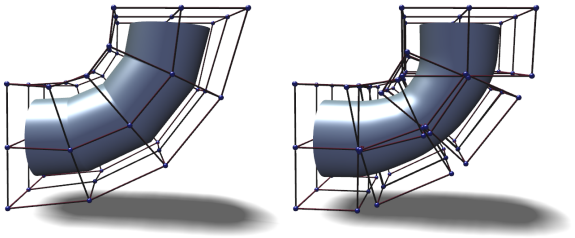

Fig. 5. Comparison of embedding techniques. Stitching the discontinuous simulation mesh, followed by barycentric interpolation, leads to C^0 artifacts (left). In contrast, our smooth MLS-based embedding yields a considerably higher surface quality (right).

The averaged nodal displacements have to be interpolated within elements in order to deform the embedded mesh. For tetrahedral or hexahedral elements this amounts to simple linear or trilinear interpolation, respectively. For more general convex or non-convex polyhedra, mean value coordinates [Floater et al., 2005; Ju et al., 2005] or harmonic coordinates [Joshi et al., 2007] can be employed. All these methods, however, correspond to a non-smooth, generalized barycentric C^0 interpolation, resulting in clearly visible shading artifacts for coarse simulation meshes (cf. Fig. 5, left).

Botsch et al. [2007] employ globally supported radial basis functions for high quality interpolation, but the involved dense linear systems are prohibitive for complex simulation meshes. To overcome these limitations, and inspired by meshless methods [Müller et al., 2004a; Pauly et al., 2005], we propose a smooth embedding based on moving-least-squares (MLS) interpolation.

If we denote by $\boldsymbol{x}_i$ the nodes of the undeformed simulation mesh, and by $\tilde{\boldsymbol{u}}_i$ their averaged displacements, then the displacement at a material point $\boldsymbol{x}$ is computed by fitting an affine transformation, which amounts to minimizing the weighted least square error

$$\sum_i \theta(\|\boldsymbol{x}-\boldsymbol{x}_i\|)\,\left\|\boldsymbol{a}(\boldsymbol{x})^T\,\boldsymbol{p}(\boldsymbol{x}_i)-\tilde{\boldsymbol{u}}_i\right\|^2, \qquad (31)$$

with $\boldsymbol{p}(x,y,z)=(1,x,y,z)^T$ and $\theta(x)$ a (truncated) Gaussian weight function. Solving a (4×4) linear system $\mathbf{A}(\boldsymbol{x})\,\boldsymbol{a}(\boldsymbol{x}) = \boldsymbol{b}(\boldsymbol{x})$ yields the coefficients $\boldsymbol{a}(\boldsymbol{x})$ for the interpolated displacement $\tilde{\boldsymbol{u}}(\boldsymbol{x}) = \boldsymbol{a}(\boldsymbol{x})^T\,\boldsymbol{p}(\boldsymbol{x})$ at the position $\boldsymbol{x}$. This MLS-based embedding has several interesting properties:

- The smoothness of the interpolation is determined by the weighting kernels $w_i(\boldsymbol{x}) := \theta(\|\boldsymbol{x}-\boldsymbol{x}_i\|)$, resulting in a high quality embedding for our choice of Gaussian kernels (cf. Fig. 5, right).
- The use of linear polynomials $\boldsymbol{p}(\boldsymbol{x})$, in combination with the partition of unity property of MLS shape functions, guarantees the exact reproduction of linear displacements $\boldsymbol{u}$, i.e., in particular of rigid motions [Fries and Matthies, 2004].
- Since the approach is entirely meshless it can be used to interpolate within arbitrarily shaped elements. Choosing the support radius of w_i proportional to the local sampling density at $\boldsymbol{x}_i^0$ (e.g., distances to one-ring neighbors), yields smooth interpolations even for irregular meshes.
- An accurate approximation of higher order polynomial displacements $\boldsymbol{u}$ only requires to add more samples $(\boldsymbol{x}_i^0, \tilde{\boldsymbol{u}}_i)$ to (31), such as edge, face, or element midpoints.
- The interpolated displacement $\tilde{\boldsymbol{u}}(\boldsymbol{x})$ of a vertex $\boldsymbol{x}$ of the embedded mesh linearly depends on $\boldsymbol{a}(\boldsymbol{x})$, which in turn linearly depends on the $\tilde{\boldsymbol{u}}_i$ used in (31), which finally linearly depend on $\boldsymbol{u}_i$ through (30) and (18). Hence, by combining these linear relationships, the weights $w_i(\boldsymbol{x})$ as well as the set $\mathcal{N}(\boldsymbol{x})$ of relevant basis functions N_i can be precomputed, such that during the simulation only

$$\tilde{\boldsymbol{u}}(\boldsymbol{x}) = \sum_{i\in\mathcal{N}(\boldsymbol{x})} w_i(\boldsymbol{x})\,N_i(\boldsymbol{x})\,\boldsymbol{u}_i =: \sum_{i\in\mathcal{N}(\boldsymbol{x})} W_i(\boldsymbol{x})\,\boldsymbol{u}_i \qquad (32)$$

has to be evaluated as a linear combination of $\boldsymbol{u}_i$.

8. Collisions

Since collision handling is not the focus of this work, we restrict ourselves to simple penalty-based collision response within the semi-implicit time integration. The basic approach is equivalent to CG FEM, therefore we only discuss the differences due to the discontinuous displacement $\boldsymbol{u}$.

Suppose that in the current time-step we detect a collision at a displaced material point $\boldsymbol{x}_c + \tilde{\boldsymbol{u}}(\boldsymbol{x}_c)$. Since we use the interpolated displacement $\tilde{\boldsymbol{u}}$ of (32), $\boldsymbol{x}_c$ can be an arbitrary embedded point, e.g., a vertex of the embedded surface mesh. Nodal collisions using the stitched displacement (30) is just a special case of this formulation.

For collision response a penalty force proportional to the penetration depth is added to the system. For

the semi-implicit solve this displacement-dependent force yields $\boldsymbol{f}(\boldsymbol{x}_c) = \mathbf{A} \cdot \tilde{\boldsymbol{u}}(\boldsymbol{x}_c) + \boldsymbol{b}$ with $\mathbf{A} \in \mathbb{R}^{3\times3}$ and $\boldsymbol{b} \in \mathbb{R}^3$. The corresponding penalty energy is

$$E_{\text{coll}}(\boldsymbol{x}_c) \;=\; \frac{1}{2}\,\tilde{\boldsymbol{u}}(\boldsymbol{x}_c)^T\,\mathbf{A}\,\tilde{\boldsymbol{u}}(\boldsymbol{x}_c) \;+\; \tilde{\boldsymbol{u}}(\boldsymbol{x}_c)^T\,\boldsymbol{b},$$

which after inserting the definition of $\tilde{\boldsymbol{u}}$ in (32) becomes

$$\frac{1}{2}\sum_{i,j}\boldsymbol{u}_i^T W_i(\boldsymbol{x}_c)\,\mathbf{A}\,W_j(\boldsymbol{x}_c)\,\boldsymbol{u}_j \;+\; \sum_i \boldsymbol{u}_i^T W_i(\boldsymbol{x}_c)\,\boldsymbol{b}.$$

Since this collision energy corresponds to an external force, it has to be either subtracted from the internal potential energy $\frac{1}{2}\boldsymbol{U}^T\mathbf{K}\boldsymbol{U}$ or to be added to the external energy $\boldsymbol{U}^T\boldsymbol{F}$. Hence, we can incorporate the collision energy E_{coll} into the system (27) by updating (3×3) blocks of the stiffness matrix $\mathbf{K}$ and 3-vectors of the external force $\boldsymbol{F}$ (see Section 4.2):

$$\begin{aligned}\mathbf{K}_{ij} \;-\!=\; & W_i(\boldsymbol{x}_c)\,\mathbf{A}\,W_j(\boldsymbol{x}_c)\,,\\ \mathbf{F}_i \;+\!=\; & \mathbf{b}\,W_i(\boldsymbol{x}_c)\,,\end{aligned}$$

for all $i,j \in \mathcal{N}(\boldsymbol{x}_c)$, i.e., the set of basis functions W_i, respectively N_i, influencing the collision point $\boldsymbol{x}_c$ (see (32)).

If the simulation mesh is also used for visualization, simple nodal collisions are sufficient in most cases, as for instance for the examples shown in Section 9. However, for embedded simulations collisions should be detected and handled on the vertices of the embedded surface (cf. Fig. 6).

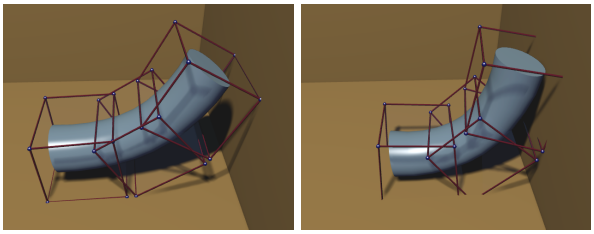

Fig. 6. Collision handling on the nodes of the simulation mesh (left) and the vertices of the embedded mesh (right).

9. Results

In this section we demonstrate how the possibility to use arbitrary polyhedral elements and simple polynomial shape functions can be exploited to derive a versatile and efficient simulation technique. Before presenting specific example applications, which are also shown in the accompanying video, we discuss some general advantages and disadvantages of DG FEM compared to CG FEM.

Method	Resolution	#DOFs	Spars.	Int.	Ass.	Solve
BZ lin.	$10\times10\times10$	12k	0.28%	532	22	656
IP lin.	$10\times10\times10$	12k	0.62%	1437	87	734
CG trilin.	$15\times15\times15$	12k	0.58%	3750	41	641
BZ quad.	$10\times10\times10$	30k	0.28%	3062	152	7797
IP quad.	$10\times10\times10$	30k	0.64%	8344	621	8484

Table 1
Comparison of BZ and IP with linear/quadratic basis functions to trilinear CG FEM for 3D elasticity. The mesh resolution is chosen to match the DOFs of DG and CG. The table lists matrix sparsity and timings (in ms) for volume integration, matrix assembly, and the solution of the linear system (taken on an Intel Core2 Duo 2.4 GHz).

9.1. DG FEM versus CG FEM

The accompanying video provides comparisons of CG and DG for 3D elasticity, on coarse and more detailed simulation meshes. However, a *qualitative* comparison between the two methods is generally hard. We therefore also *quantitatively* compare CG to DG, the latter using BZ/IP penalties and linear/quadratic basis functions, based on a 2D Poisson problem with analytically known solution (cf. Fig. 2). In addition, Table 1 gives some statistics and timings of the same five methods for 3D linear elasticity. Note that even for the same mesh and basis functions DG provides more degrees of freedom (DOFs) than CG, since nodes can "split" due to discontinuous displacements. The plots and timings are therefore with respect to DOFs.

As expected, the IP method converges regularly, at a rate similar to CG for linear shape functions, and at a faster rate for quadratic ones. By consequence, the jumps decrease under element refinement, eventually reconstructing the exact, continuous solution [Cockburn, 2003]. The only additional parameter compared to CG FEM is the penalty weight η in (20),(21), which has to be sufficiently high to guarantee stability. We simply start with a low value and double it until $\mathbf{K}$ is positive definite, which has never been a problem in our experiments and typically leads to η in the order of 10^1–10^2. Note that η should not be too high, since otherwise the method resembles CG and does not exploit its additional DOFs (Fig. 3).

The missing consistency terms of BZ (cf. (19), (21)) allow for sparser matrices and higher efficiency. Furthermore, the method is stable for any positive penalty η. Although lacking theoretical convergence

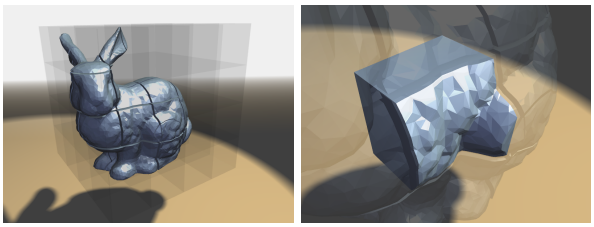

Fig. 7. Intersecting the bunny with a hex-grid generates 41 elements (left). Closeup view of a non-convex element (right).

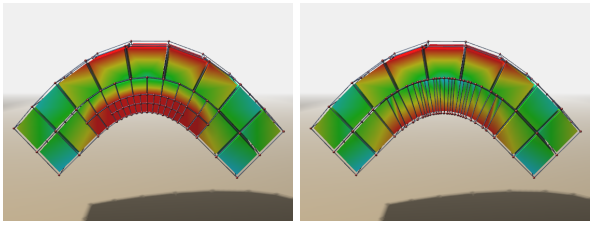

Fig. 8. A bar (36 hex-elements) is dynamically refined during bending. 1-to-8 subdivision results in 274 elements (left), whereas 1-to-2 refinement yields 77 elements (right).

guarantees, BZ shows a reasonable convergence behavior in practice and gives visually convincing results. We therefore consider it well suited for typical graphics applications requiring physically plausible deformations only. For more accurate simulations the IP method is the better choice. Highly accurate results can be achieved using more complex numerical fluxes in combinations with nonlinear strain measures [Ten Eyck and Lew, 2006].

Both DG methods lead to higher condition numbers of the linear systems, which, however, has not been a problem in all our examples, for both the conjugate gradients solver as well as the sparse Cholesky factorization.

For the same number of DOFs and basis functions of the same degree, CG FEM can be observed to be more accurate than DG FEM by a constant factor (Fig. 2) and to be slightly more efficient (Table 1). Since standard CG FEM is also easier to implement, it will stay the preferred method for many applications. However, as soon as topological changes of the simulation mesh are required or if complex element shapes have to be simulated, the higher flexibility of DG FEM pays off, as for instance in the following examples.

9.2. Mesh Generation by Hexahedral Slicing

A challenge in simulating deformable objects is the preservation of surface detail without introducing an excessive amount of simulation primitives. Commonly used approaches include voxelization of the object's volume or tetrahedrization. While voxelization is simple to implement and results in well-behaved elements, it cannot accurately represent surface details unless a high number of elements is used. On the other hand, tetrahedral meshes can accurately represent objects defined by surface meshes, but result in a higher number of elements.

Using arbitrary elements in DG FEM gives rise to an interesting mesh generation algorithm that decouples the number of elements (and thus the DOFs) from the resolution of the surface mesh. Combining the strengths of both voxelization and tetrahedrization, the simulation mesh is generated by intersecting the object with a hexahedral grid. Each intersected cell then corresponds to a finite element, resulting in hexahedral elements in the interior and arbitrary polyhedra at the object's surface (cf. Fig. 7). Note that the strain energy is integrated over the exact volume of the object, whereas a pure embedded simulation could in this case lead to an erroneous coupling of the bunny's ears.

9.3. Dynamic Adaptivity

In order to make optimal use of the available computational resources, it is often desirable to adaptively enhance the resolution of a dynamic simulation around a specific area of interest. Using arbitrary elements in a DG framework allows for easy and flexible refinement.

We chose a simple criterion based on stress concentration, refining an element when its largest absolute principal stress exceeds a given threshold. For the actual topological refinement, we can, e.g., perform a regular 1-to-8 subdivision of hexahedral elements, conceptually similar to [Grinspun et al., 2002]. An interesting alternative is the more flexible 1-to-2 split along the plane perpendicular to the principal stress direction, which generates fewer elements for the same refinement threshold (cf. Fig. 8).

Note that the refinement of an element is in no way restricted by the refinement level of its neighbors. When splitting an element, we simply copy the parent's coefficients for displacement $\boldsymbol{u}_i$ and velocity $\dot{\boldsymbol{u}}_i$ to its children. This heuristic causes the slight popping artifacts visible in the video, which could be avoided by a more sophisticated technique.

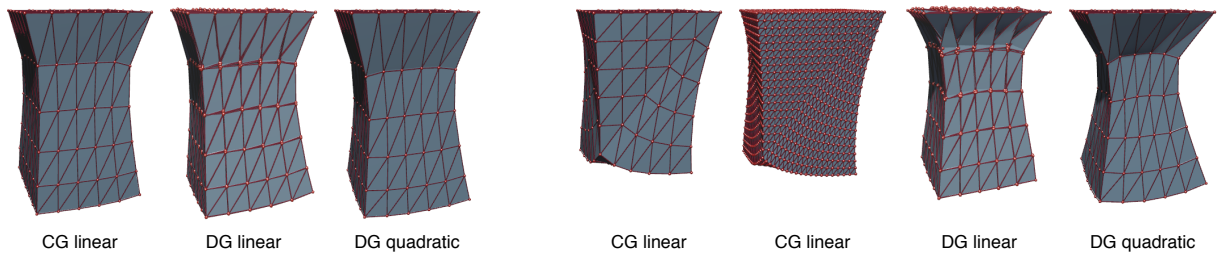

Fig. 9. A suspended cube consisting of 750 tetrahedra deforming under its own weight, simulated with linear CG FEM and linear/quadratic DG FEM (IP method), using a Poisson's ratio of $\nu = 0.3$ (left) and $\nu = 0.499$ (right). While DG gives the expected, symmetric solution, CG shows severe locking artifacts in the nearly incompressible case, even after mesh refinement.

9.4. Cutting

Using DG FEM for cutting simulations has a couple of advantages over existing methods. Being able to simulate arbitrary elements avoids complex remeshing of the simulation domain (cf. Fig. 10), similar in spirit to [Molino et al., 2004; Wicke et al., 2007; Sifakis et al., 2007a; Martin et al., 2008]. Furthermore, thanks to the analytic integration the contributions of newly created elements can be computed very efficiently and accurately, avoiding the need for expensive numerical integration during the simulation. By storing and reusing individual edge and face integrals, after splitting an element we only need to recompute integrals over edges and faces intersecting the cut plane.

Poorly shaped elements with negligible volume cause numerical problems, equivalently to CG FEM. However, those elements can effectively be avoided by simply merging them with neighboring elements, exploiting the fact that our method is not restricted to convex elements. Note that also for mesh generation and dynamic refinement we either prevent the generation of degenerate elements, or remove them by the mentioned sliver merging technique.

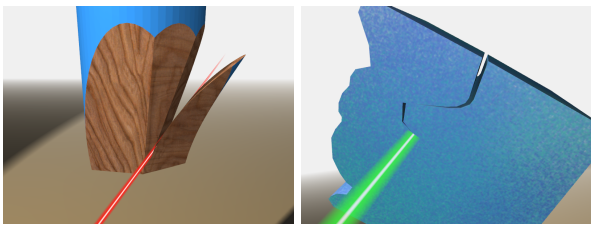

Fig. 10. Sharpening a pencil consisting of a single convex element (left). Cutting a bunny out of a cube (right).

9.5. Locking

In the case of nearly incompressible materials, as the Poisson's ratio ν approaches the limit value of 0.5, standard FEM is known to exhibit an overly stiff behavior termed *locking*.

An intuitive explanation for this phenomenon is provided by the counting argument [Irving et al., 2007]: Each element introduces an additional volume constraint in order to preserve its volume locally. However, a continuous FEM mesh with n nodes has only $3n$ degrees of freedom, while in the specific case of a tetrahedral mesh, the number of elements is at least $4n$, resulting in an overconstrained system.

On the other hand, the additional degrees of freedom present in discontinuous Galerkin FEM allow the method to effectively circumvent locking, as demonstrated in Fig. 9. In this example, increasing the number of elements will not prevent locking in the CG FEM case. Also note that the locking CG FEM solution is strongly influenced by the topology of the simulation mesh, resulting in an asymmetric solution, whereas the DG FEM solution is free of such artifacts.

9.6. Sliver Elements

In standard FEM with nodal basis functions, the computation of shape functions and their derivatives typically involves the inversion of a Jacobian matrix, causing numerical problems for ill-shaped elements. This affects the integration of basis functions over the element as well as other uses of basis functions such as the interpolation of nodal quantities.

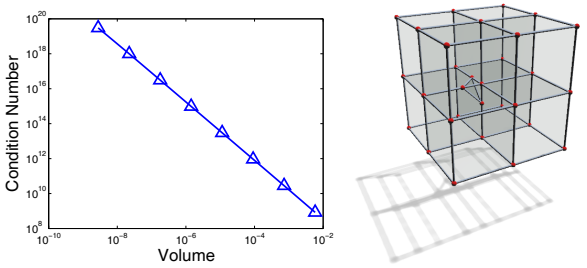

Fig. 11. Mesh of nine elements. As the volume of the innermost tetrahedral element decreases, the condition number of the global stiffness matrix increases.

This particular problem can be avoided in DG FEM, as basis functions are defined in global coordinates. However, elements of small volume still cause problems in DG FEM. As stated in [Shewchuck, 2002], the condition number of the stiffness matrix of a tetrahedral mesh is related to the ratio between the volume of the largest and the smallest element. We observe a similar behavior in DG FEM, as shown in Fig. 11.

On the other hand, we note that in DG FEM elements with *locally* small features do not cause problems, as long as the total volume of the element stays reasonably large (Fig. 12). Note that in order to compute the condition number of the stiffness matrices in those examples, appropriate boundary constraints were introduced in order to get a unique solution to the static problem.

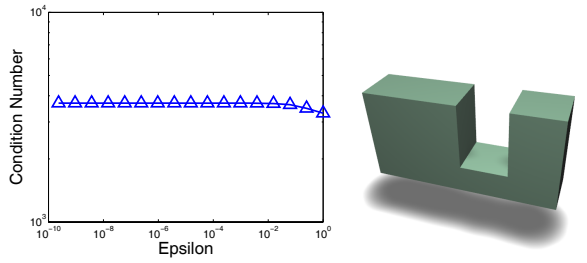

Fig. 12. Single non-convex element with a locally thin feature. Even as the height ε of the narrowed middle section approaches zero, the condition number of the stiffness matrix stays finite.

10. Conclusion

We presented a novel simulation technique for deformable models based on discontinuous Galerkin FEM. The main advantage of DG FEM is the flexibility to use discontinuous shape functions, which we exploited for the efficient simulation of arbitrary polyhedral elements. Our generalization of stiffness warping enables physically plausible large-scale deformations, and our MLS-based surface embedding allows to simulate complex models in the DG framework.

We demonstrated the versatility of our approach on conceptually simple, efficient, and robust techniques for mesh generation, adaptive refinement, and cutting. While there are successful methods for each individual problem, our approach provides an interesting alternative that handles all problems in a single, consistent DG FEM framework. Promising directions for future work include nonlinear elasticity simulations of both solids and shells, which would benefit even more from the flexibility offered by DG FEM.

Acknowledgments

The authors are grateful to Christoph Schwab for inspiring discussions, and to the anonymous reviewers of the conference paper [Kaufmann et al., 2008] for helpful comments and suggestions. Sebastian Martin was supported by the Swiss National Science Foundation through grant 200021-117756.

Appendix A. Volume Integration

The following steps show how to compute $\int_K p(\boldsymbol{x})\,d\boldsymbol{x}$ for an arbitrary polynomial $p(\boldsymbol{x})$ with $\boldsymbol{x} = (x_1, x_2, x_3)$.

(i) Integrate $p(\boldsymbol{x})$ formally to obtain the polynomial $q(\boldsymbol{x})$ in $\boldsymbol{x}$:

$$q(\boldsymbol{x}) \leftarrow \int p(\boldsymbol{x})\,dx_1$$

(ii) Perform the following three steps for i in $\{1, 2, 3\}$, with j and k defined as $j = (i \bmod 3) + 1$ and $k = ((i+1) \bmod 3) + 1$.

(a) Transform $q(\boldsymbol{x})$ into the polynomial $\hat{q}(\hat{\boldsymbol{x}}, \boldsymbol{n})$ in $\hat{\boldsymbol{x}} = (\hat{x}_1, \hat{x}_2, \hat{x}_3)$ with the symbolical constant $\boldsymbol{n} = (n_1, n_2, n_3)$ by performing the following substitutions:

$$x_i \rightarrow \frac{1}{n_i} - \hat{x}_j \frac{n_j}{n_i} - \hat{x}_k \frac{n_k}{n_i},$$
$$x_j \rightarrow \hat{x}_j, \quad x_k \rightarrow \hat{x}_k$$

(b) Integrate $\hat{q}(\hat{\boldsymbol{x}}, \boldsymbol{n})$ formally to obtain the polynomial $\hat{r}(\hat{\boldsymbol{x}}, \boldsymbol{n})$ in $\hat{\boldsymbol{x}}$:

$$\hat{r}(\hat{\boldsymbol{x}}, \boldsymbol{n}) \leftarrow \int \hat{q}(\hat{\boldsymbol{x}}, \boldsymbol{n})\,d\hat{x}_j$$

(c) Integrate formally over the edge connecting $\boldsymbol{a} = (a_1, a_2, a_3)$ and $\boldsymbol{b} = (b_1, b_2, b_3)$ to get a polynomial in $\boldsymbol{a}$ and $\boldsymbol{b}$:

$$P_i(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{n}) \leftarrow$$
$$(b_k - a_k) \int_0^1 \hat{r}(\boldsymbol{a}\,(1-t) + \boldsymbol{b}\,t, \boldsymbol{n})\,dt$$

(iii) The integral over the volume can now be computed as follows, where $\boldsymbol{n}^f$ defines the plane of face f as $\{\boldsymbol{x} \in \mathbb{R}^3 | \boldsymbol{x} \cdot \boldsymbol{n}^f = 1\}$. $d_f \in \{1, 2, 3\}$ is the direction of projection for face f which must be chosen such that $n^f_{d_f} \neq 0$. ${\boldsymbol{x_1}}^e$ and ${\boldsymbol{x_2}}^e$ are the nodes of edge e.

$$\int_K p(\boldsymbol{x})\,d\boldsymbol{x} = \sum_{f \in \partial K} \frac{n^f_1}{n^f_{d_f}} \sum_{e \in \partial f} P_{d_f}(\boldsymbol{x}^e_1, \boldsymbol{x}^e_2, \boldsymbol{n}^f)$$

References

Arnold, D. N., Brezzi, F., Cockburn, B., Marini, L. D., 2001. Unified analysis of discontinuous Galerkin methods for elliptic problems. SIAM J. Numer. Anal. 39 (5), 1749–1779.

Babuška, I., Zlámal, M., 1973. Nonconforming elements in the finite element method with penalty. SIAM J. Numer. Anal. 10, 863–875.

Bargteil, A. W., Wojtan, C., Hodgins, J. K., Turk, G., 2007. A finite element method for animating large viscoplastic flow. ACM Trans. on Graphics (Proc. SIGGRAPH) 26 (3), 16.1–16.8.

Bathe, K.-J., 1995. Finite Element Procedures. Prentice Hall.

Bielser, D., Glardon, P., Teschner, M., Gross, M., 2003. A state machine for real-time cutting of tetrahedral meshes. In: Proc. of Pacific Graphics. pp. 377–386.

Bielser, D., Gross, M., 2000. Interactive simulation of surgical cuts. In: Proc. of Pacific Graphics. pp. 116–125.

Botsch, M., Bommes, D., Kobbelt, L., 2005. Efficient linear system solvers for geometry processing. In: 11th IMA conference on the Mathematics of Surfaces.

Botsch, M., Pauly, M., Gross, M., Kobbelt, L., 2006. PriMo: Coupled prisms for intuitive surface modeling. In: Proc. of Symp. on Geometry Processing. pp. 11–20.

Botsch, M., Pauly, M., Wicke, M., Gross, M., 2007. Adaptive space deformations based on rigid cells. Computer Graphics Forum (Proc. Eurographics) 26 (3), 339–347.

Capell, S., Green, S., Curless, B., Duchamp, T., Popović, Z., 2002. A multiresolution framework for dynamic deformations. In: Proc. of Symp. on Computer Animation. pp. 41–47.

Cockburn, B., 2003. Discontinuous Galerkin methods. Z. Angew. Math. Mech. 80 (11), 731–754.

Debunne, G., Desbrun, M., Cani, M.-P., Barr, A. H., 2001. Dynamic real-time deformations using space and time adaptive sampling. In: Proc. of ACM SIGGRAPH. pp. 31–36.

Douglas, J., Dupont, T., 1976. Interior penalty procedures for elliptic and parabolic Galerkin methods. Computing Methods in Applied Science, Lecture Notes in Physics 58.

Faloutsos, P., van de Panne, M., Terzopoulos, D., 1997. Dynamic free-form deformations for animation synthesis. IEEE Transactions on Visualization and Computer Graphics 3 (3), 201–214.

Floater, M. S., Kos, G., Reimers, M., 2005. Mean value coordinates in 3D. Computer Aided Geometric Design 22 (7), 623–631.

Fries, T. P., Matthies, H. G., 2004. Classification and overview of meshfree methods. Informatikbericht 2003-03, revised 2004, Institute of Scientific Computing, Technical University Braunschweig.

Grinspun, E., Krysl, P., Schröder, P., 2002. CHARMS: A simple framework for adaptive simulation. ACM Trans. on Graphics (Proc. SIGGRAPH) 21 (3), 281–290.

Hansbo, P., Larson, M., 2002. Discontinuous Galerkin methods for incompressible and nearly incompressible elasticity by Nitsche's method. Comput. Methods Appl. Mech. Eng. 191 (17), 1895–1908.

Hauth, M., Strasser, W., 2004. Corotational simulation of deformable solids. In: Proc. of WSCG. pp. 137–145.

Hughes, T. J. R., 2000. The Finite Element Method. Linear Static and Dynamic Finite Element Analysis. Dover Publications.

Irving, G., Schroeder, C., Fedkiw, R., 2007. Volume conserving finite element simulations of deformable models. ACM Trans. on Graphics (Proc. SIGGRAPH) 26 (3).

James, D., Barbič, J., Twigg, C., 2004. Squashing cubes: Automating deformable model construction for graphics. In: Proc. of SIGGRAPH '04 Sketches and Applications.

Joshi, P., Meyer, M., DeRose, T., Green, B., Sanocki, T., 2007. Harmonic coordinates for character articulation. ACM Trans. on Graphics (Proc. SIGGRAPH) 26 (3).

Ju, T., Schaefer, S., Warren, J., 2005. Mean value coordinates for closed triangular meshes. ACM Trans. on Graphics (Proc. SIGGRAPH) 24 (3), 561–566.

Kaufmann, P., Martin, S., Botsch, M., Gross, M., 2008. Flexible simulation of deformable models using discontinuous Galerkin FEM. In: Proc. of

Symp. on Computer Animation. pp. 105–115.

Lew, A., Neff, P., Sulsky, D., Ortiz, M., 2004. Optimal BV estimates for a discontinuous Galerkin method in linear elasticity. Applied Mathematics Research Express (3), 73–106.

Martin, S., Kaufmann, P., Botsch, M., Wicke, M., Gross, M., 2008. Polyhedral finite elements using harmonic basis functions. Computer Graphics Forum (Proc. SGP) 27 (5), 1521–1529.

Mirtich, B., 1996. Fast and accurate computation of polyhedral mass properties. Journal of Graphics Tools 1 (2), 31–50.

Molino, N., Bao, Z., Fedkiw, R., 2004. A virtual node algorithm for changing mesh topology during simulation. ACM Trans. on Graphics (Proc. SIGGRAPH) 23 (3), 385–392.

Müller, M., Dorsey, J., McMillan, L., Jagnow, R., Cutler, B., 2002. Stable real-time deformations. In: Proc. of Symp. on Computer Animation. pp. 163–170.

Müller, M., Gross, M., 2004. Interactive virtual materials. In: Proc. of Graphics Interface. pp. 239–246.

Müller, M., Keiser, R., Nealen, A., Pauly, M., Gross, M., Alexa, M., 2004a. Point-based animation of elastic, plastic and melting objects. In: Proc. of Symp. on Computer Animation. pp. 141–151.

Müller, M., Teschner, M., Gross, M., 2004b. Physically based simulation of objects represented by surface meshes. In: Proc. of Computer Graphics International. pp. 26–33.

Nealen, A., Müller, M., Keiser, R., Boxerman, E., Carlson, M., 2006. Physically based deformable models in computer graphics. Computer Graphics Forum 25 (4), 809–836.

O'Brien, J. F., Bargteil, A. W., Hodgins, J. K., 2002. Graphical modeling and animation of ductile fracture. ACM Trans. on Graphics (Proc. SIGGRAPH) 21 (3), 291–294.

O'Brien, J. F., Hodgins, J. K., 1999. Graphical modeling and animation of brittle fracture. In: Proc. of ACM SIGGRAPH. pp. 137–146.

Otaduy, M. A., Germann, D., Redon, S., Gross, M., 2007. Adaptive deformations with fast tight bounds. In: Proc. of Symp. on Computer Animation. pp. 181–190.

Pauly, M., Keiser, R., Adams, B., Dutre, P., Gross, M., Guibas, L. J., 2005. Meshless animation of fracturing solids. ACM Trans. on Graphics (Proc. SIGGRAPH) 24 (3), 957–964.

Saad, Y., van der Vorst, H. A., 2000. Iterative solution of linear systems in the 20th century. J. Comput. Appl. Math. 123 (1-2), 1–33.

Shewchuck, J., 2002. What is a good linear finite element? interpolation, conditioning, and quality measures. In: Proc. of the 11th International Meshing Roundtable. pp. 115–126.

Sifakis, E., Der, K. G., Fedkiw, R., 2007a. Arbitrary cutting of deformable tetrahedralized objects. In: Proc. of Symp. on Computer Animation. pp. 73–80.

Sifakis, E., Shinar, T., Irving, G., Fedkiw, R., 2007b. Hybrid simulation of deformable solids. In: Proc. of Symp. on Computer Animation. pp. 81–90.

Steinemann, D., Harders, M., Gross, M., Szekely, G., 2006a. Hybrid cutting of deformable solids. In: Proc. of IEEE VR. pp. 35–42.

Steinemann, D., Otaduy, M. A., Gross, M., 2006b. Fast arbitrary splitting of deforming objects. In: Proc. of Symp. on Computer Animation. pp. 63–72.

Ten Eyck, A., Lew, A., 2006. Discontinuous Galerkin methods for non-linear elasticity. International Journal for Numerical Methods in Engineering 67 (9), 1204–1243.

Terzopoulos, D., Platt, J., Barr, A., Fleischer, K., 1987. Elastically deformable models. In: Proc. of ACM SIGGRAPH. pp. 205–214.

Toledo, S., Chen, D., Rotkin, V., 2003. Taucs: A library of sparse linear solvers. http://www.tau.ac.il/∼stoledo/taucs.

Wicke, M., Botsch, M., Gross, M., 2007. A finite element method on convex polyhedra. Computer Graphics Forum (Proc. Eurographics) 26 (3), 355–364.

Wihler, T. P., 2006. Locking-free adaptive discontinuous Galerkin FEM for linear elasticity problems. Mathematics of Computation 75 (255), 1087–1102.