Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2008)
M. Gross and D. James (Editors)

Flexible Simulation of Deformable Models Using Discontinuous Galerkin FEM

Peter Kaufmann[1], Sebastian Martin[1], Mario Botsch[1,2], Markus Gross[1]

[1]Computer Graphics Laboratory, ETH Zurich,
[2]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.

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 convex elements in FEM was recently shown to considerably simplify cutting and fracture simulations [WBG07]. However, the strict conformity constraints of standard FEM require comparatively 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 [BPWG07]. 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) [Coc03]. 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, this added flexibility 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 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 6).
- We generalize stiffness warping to discontinuous polyhedral elements, thereby allowing linear strain measures to be used for large deformations (Section 7).
- For embedded simulations we reconstruct from the discontinuous mesh a smooth displacement field based on moving least squares (MLS) interpolation (Section 8).

In Section 10 we demonstrate the versatility of our approach on slicing-based mesh generation, adaptive stress-based element refinement, and flexible and efficient cutting.

2. Related Work

Starting with Terzopoulos et al. [TPBF87], 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 [NMK*06].

Cutting & Fracture. Fracturing can efficiently be performed by restricting cuts to existing element boundaries [MG04], 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 [BG00, BGTG03] and/or general remeshing [OH99, OBH02, SHGS06]. When accommodating the crack surface, special care has to be taken to avoid numerically unstable sliver elements. Similarly, Bargteil et al. [BWHT07] performed remeshing to remove degenerate elements during large plastic deformations.

Meshless approaches intrinsically avoid remeshing by using particles instead of a simulation mesh [MKN*04]. 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 [PKA*05] or by updating a distance graph [SOG06]. 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 [MBF04], which, instead of splitting elements, duplicates them and embeds the surface in both copies. While the original approach was limited to cutting each element at most three times, its recent generalization [SDF07, SSIF07] overcomes this restriction. Wicke et al. [WBG07] avoid remeshing of cut elements into consistent tetrahedra by directly supporting convex polyhedra in FEM simulations. The drawback of their method is the comparatively complex computation and integration of their mean value shape functions.

In the context of cutting and fracturing our approach is most similar to [WBG07], but its ability to handle arbitrary convex and non-convex polyhedra provides a higher flexibility, and it is more efficient due to the use of simple polynomial shape functions.

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 [DDCB01, GKS02, CGC*02, OGRG07]. 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 [GKS02, CGC*02]. However, in order to ensure linear independence of basis functions, Grinspun et al. [GKS02] restrict the refinement to one level difference between neighboring elements. In contrast, the hybrid simulation [SSIF07] allows for multi-level hanging nodes, also 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 [FvdPT97, CGC*02, MBF04, MG04, MTG04, JBT04, SSIF07]. 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 [BPWG07], 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.

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., [BZ73, DD76]). In the last decade, however, DG FEM regained increasing attention in applied mathematics [ABCM01, Coc03].

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., [LNSO04, TEL06, Wih06]), where it was shown to be locking-free even for nearly incompressible materials and to provide an accuracy similar to CG FEM at comparable computational cost.

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*. After reviewing CG FEM for linear elasticity (Section 3), we introduce the main concepts and differences of DG FEM (Section 4), before discussing the simulation of arbitrary polyhedra (Section 6), the generalization of stiffness warping to DG FEM (Section 7), embedded simulation (Section 8), and collision handling (Section 9). Equipped with those techniques, we demonstrate the versatility of our framework on a set of different applications in Section 10.

3. Linear Elasticity using CG FEM

In this section we briefly review the main equations of 3D linear elasticity and their respective discretization using CG FEM, in order to contrast them with DG FEM in the following sections. A more detailed derivation of the CG formulation can be found in many textbooks, e.g. [Bat95, Hug00].

In the following we consider a 3D object with material coordinates $\mathrm{x} = (x,y,z)^T \in \Omega$, which is to be deformed by a displacement vector field $\mathrm{u} : \Omega \to \mathbb{R}^3$. We measure local deformations using the linear Cauchy strain $\varepsilon(\mathrm{u}) = \frac{1}{2}\left(\nabla \mathrm{u} + \nabla \mathrm{u}^T\right)$, which under the assumption of a Hookean material is linearly related to the stress

$$\sigma(\mathrm{u}) \;=\; \mathrm{C} : \varepsilon(\mathrm{u}) \tag{1}$$

through a symmetric 4-tensor C containing material parameters[†]. In static equilibrium the internal forces have to be in balance with the external forces f, which is expressed by

$$-\nabla \cdot \sigma(\mathrm{u}) \;=\; \mathrm{f}\,. \tag{2}$$

Equations (1) and (2), in combination with suitable boundary constraints on $\partial\Omega$, constitute the strong form of the problem. The standard approach is to multiply (1) and (2) by suitable test functions, to formally integrate by parts over the domain Ω, and combine the resulting equations. This yields the weak form

$$a_{\mathrm{CG}}(\mathrm{u},\mathrm{v}) := \int_\Omega \varepsilon(\mathrm{v}) : \mathrm{C} : \varepsilon(\mathrm{u}) \;=\; \int_\Omega \mathrm{f}\cdot\mathrm{v}\,, \tag{3}$$

which is defined in terms of the bilinear form $a_{\mathrm{CG}}(\cdot,\cdot)$. The goal is to find a displacement function u, such that the weak form (3) holds for all suitable test functions v.

In order to discretize (3) 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

$$\mathrm{u}(\mathrm{x}) \;\approx\; \sum_{i=1}^{n} \mathrm{u}_i N_i(\mathrm{x})\,. \tag{4}$$

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 restrict the basis functions to be *conforming*, i.e., C^m continuous within and C^{m-1} continuous across elements [Hug00]. For the linear elasticity problem with weak form (3), the N_i therefore have to be C^0 *continuous* across elements.

[†] The colon operator denotes the tensor product between matrices A and B as $\mathrm{A}:\mathrm{B} = \sum_{i,j} \mathrm{A}_{ij}\mathrm{B}_{ij}$, and between matrix A and 4-tensor C as $\mathrm{A}:\mathrm{C} = \sum_{i,j} \mathrm{A}_{ij}\mathrm{C}_{ijkl}$ or $\mathrm{C}:\mathrm{A} = \sum_{k,l} \mathrm{C}_{ijkl}\mathrm{A}_{kl}$. The dot denotes vector dot products $\mathrm{u}\cdot\mathrm{v}$ or matrix-vector products $\mathrm{A}\cdot\mathrm{v}$ and $\mathrm{v}\cdot\mathrm{A}$.

Approximating both u and v by the shape functions N_i and exploiting the bilinearity of $a_{\mathrm{CG}}(\cdot,\cdot)$ yields the linear system

$$\mathrm{K}\cdot\begin{bmatrix}\vdots\\ \mathrm{u}_i\\ \vdots\end{bmatrix} = \begin{bmatrix}\vdots\\ \mathrm{F}_i\\ \vdots\end{bmatrix}\,,\ \text{with}\ \left\{\begin{array}{rcl}\mathrm{K}_{ij} &=& a_{\mathrm{CG}}\left(N_i,N_j\right)\cdot \mathrm{I}_3\\ \mathrm{F}_i &=& \int_\Omega \mathrm{f}\,N_i\end{array}\right.\,, \tag{5}$$

where I_3 denotes the 3×3 identity matrix. This system is finally solved for the unknown coefficients $\mathrm{u}_i \in \mathbb{R}^3$.

4. Linear Elasticity using DG FEM

After reviewing CG FEM, we will now introduce the main concepts of DG FEM and point out the differences to standard CG FEM. Due to space constraints we only provide the most important equations, and refer the interested reader to the survey articles [ABCM01, Coc03] for more details. In contrast to CG FEM, DG FEM allows for *non-conforming* or *discontinuous* shape functions N_i, thereby resulting in discontinuous approximations of u. Those discontinuities have to be taken into account when deriving the weak form of the problem, which will lead to penalty terms that *weakly* enforce continuity across elements.

Analogous to CG FEM, equations (1) and (2) are multiplied by test functions and integrated over the domain $\Omega = \cup_{K\in\mathcal{T}} K$, formulated as a sum of integrals over elements $K \in \mathcal{T}$. Integration by parts over these K leads to additional integrals over all element boundaries $\Gamma = \cup_K \partial K$, which due to the discontinuities of u and σ do *not* cancel out as in CG FEM. In order to "glue" the discontinuities, the functions u and σ are replaced by their so-called *numerical fluxes* on the element boundaries Γ. The various DG methods differ in exactly these fluxes, a detailed overview and classification of which can be found in [ABCM01]. In the resulting DG weak form they show up as penalty terms punishing discontinuities, thereby weakly enforcing continuity.

To formalize this we introduce the *average* operator $\{\cdot\}$ and the *jump* operator $[\![\cdot]\!]$ for both vector-valued functions u and matrix-valued functions σ on an element boundary, i.e., on a face $f = K^- \cap K^+$ shared by two elements K^- and K^+. If we denote by $\mathrm{u}^\pm$ and $\sigma^\pm$ functions evaluated on $f \subset \partial K^\pm$, by $\mathrm{n}^\pm$ the outward normal of $K^\pm$ on f, and by $\mathrm{u}\otimes\mathrm{n} = \mathrm{u}\,\mathrm{n}^T$ the outer product, then these two operators are defined as

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

A straightforward approach is to minimize the squared jump $[\![\mathrm{u}]\!] : [\![\mathrm{u}]\!] = \|\mathrm{u}^- - \mathrm{u}^+\|^2$. This corresponds to the method of Babuška and Zlámal [BZ73], denoted by BZ, whose weak form uses a_{BZ} instead of a_{CG} in (3):

$$a_{\mathrm{BZ}}(\mathrm{u},\mathrm{v}) \;:=\; \int_\Omega \varepsilon(\mathrm{v}):\mathrm{C}:\varepsilon(\mathrm{u}) \;+\; \int_\Gamma \eta_f\, [\![\mathrm{u}]\!] : [\![\mathrm{v}]\!]\,. \tag{6}$$

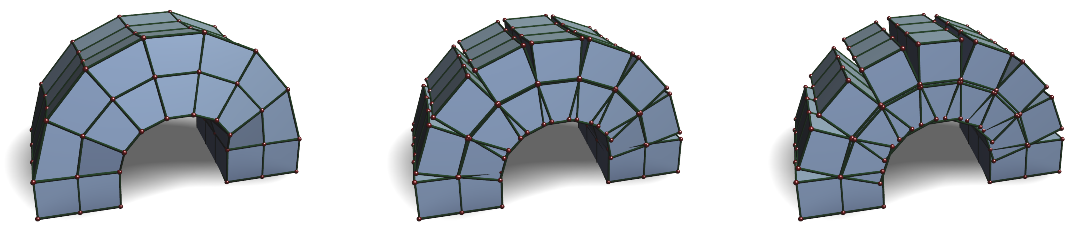

Figure 1: *Comparison of CG FEM (left), DG FEM (center), and the elastically coupled rigid cells of PriMo [BPGK06] (right). The DG method conceptually spans the whole space from CG to PriMo, since for sufficiently large penalties η it approximates the CG results, and for an extremely stiff material and lower penalty η it reproduces the rigid cells of PriMo.*

Note that the BZ weak form (6) differs from the CG weak form (3) in the Γ-integral only, which punishes the jump $[\![\mathrm{u}]\!]$ weighted by a penalty parameter per face f [HL02]

$$\eta_f \;=\; \eta \cdot \mathrm{area}(f) \cdot \left(\frac{1}{\mathrm{vol}(K^-)} + \frac{1}{\mathrm{vol}(K^+)} \right), \qquad (7)$$

using a global penalty parameter $\eta > 0$ typically being in the order of 10^1–10^2 in all our experiments. The internal elastic energy of the deformed object can then be written as

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

which reveals an interesting connection to both CG FEM and the elastically coupled *rigid* cells of PriMo [BPGK06]: 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. 1). 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 [BPWG07], as we will show in Section 6.

The BZ method is geometrically intuitive and easy to implement. Its penalty term is equivalent to both the glue energy of PriMo [BPGK06] and the soft bindings of [SSIF07]. Moreover, it is *stable* in the sense that the stiffness matrix K is positive definite for any $\eta > 0$. However, as detailed in [ABCM01], the method is *not consistent*: A continuous solution u of the problem might not satisfy the BZ weak form (6). Consequently, the approximate solution u does in general not converge toward the exact solution under element refinement. Our experiments have shown that the BZ method is very well suited for applications aiming at *physically plausible* deformations only. However, if physical accuracy is important, other DG methods should be chosen.

A more accurate alternative is the *interior penalty* (IP) method [DD76], whose weak form is defined by

$$\begin{aligned} a_{\mathrm{IP}}(\mathrm{u},\mathrm{v}) := & \int_\Omega \varepsilon(\mathrm{v}) : \mathrm{C} : \varepsilon(\mathrm{u}) \\ & - \int_\Gamma \left([\![\mathrm{v}]\!] : \{\sigma(\mathrm{u})\} + [\![\mathrm{u}]\!] : \{\sigma(\mathrm{v})\} - \eta_f \, [\![\mathrm{u}]\!] : [\![\mathrm{v}]\!] \right) . \end{aligned} \qquad (8)$$

This method consists of three penalty terms in the Γ-integral:

- The first term ensures *consistency*: Any continuous solution u of the problem (1), (2) also satisfies (8).
- The second term achieves *symmetry* of the bilinear form $a(\mathrm{u},\mathrm{v})$, and thus of the stiffness matrix K.
- The last term ensures *stability*: For a sufficiently large penalty η, $a(\mathrm{u},\mathrm{u}) > 0$, i.e., K is positive definite.

It follows from consistency and stability that the IP method converges under refinement towards the exact solution, with a convergence rate determined by the polynomial degree of N_i [ABCM01]. Another advantage is that the IP method is still relatively easy to implement (see Section 5). While other (more complex) numerical fluxes exist (see, e.g., [TEL06, Wih06]), for our applications the BZ and IP methods performed very well and have been fully sufficient.

5. Discretization & Matrix Assembly

In order to implement DG FEM for linear elasticity, we have to discretize both u and v, and set up the stiffness matrix K of the problem. Since this is very similar to CG FEM discussed in Section 3), we refer the reader to [Hug00, NMK*06] for more details on the following derivations.

The discretization (4) of u can be written in matrix notation as $\mathrm{u}(\mathrm{x}) = \mathrm{H}(\mathrm{x})\,\mathrm{U}$ using a $3 \times 3n$ interpolation matrix $\mathrm{H}(\mathrm{x})$ built from the basis functions $N_i(\mathrm{x})$, and a $3n$ vector U containing the unknown coefficients $\mathrm{u}_i \in \mathbb{R}^3$. Equivalently, the test function v can be represented as $\mathrm{v}(\mathrm{x}) = \mathrm{H}(\mathrm{x})\,\mathrm{V}$.

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

$$\bar{\sigma}(\mathrm{u}(\mathrm{x})) \;=\; \bar{\mathrm{C}}\,\bar{\varepsilon}(\mathrm{u}(\mathrm{x})) \;=\; \bar{\mathrm{C}}\,\mathrm{B}(\mathrm{x})\,\mathrm{U}, \qquad (9)$$

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

For the assembly of the stiffness matrix we use the above matrix notations to write the IP weak form (8) in terms of *element contributions* (Ω-integrals) and *face contributions* (Γ-integrals). Note that for the BZ method (6) only the last of the three face contributions in (8) is needed.

The element contributions are written in terms of element stiffness matrices K_K as in CG FEM:

$$\int_\Omega \varepsilon(\mathrm{v}):\mathrm{C}:\varepsilon(\mathrm{u}) \;=\; \sum_{K\in\mathcal{T}} \mathrm{V}^T \underbrace{\int_K \mathrm{B}^T(\mathrm{x})\,\bar{\mathrm{C}}\,\mathrm{B}(\mathrm{x})}_{\mathrm{K}_K}\,\mathrm{U}\,. \tag{10}$$

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

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

To write this in matrix notation, we need a "normal matrix"

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

and difference and average versions of matrices B and H

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

which themselves are defined in terms of the restrictions $\mathrm{B}_f^\pm := \mathrm{B}|_{K^\pm}$ and $\mathrm{H}_f^\pm := \mathrm{H}|_{K^\pm}$ containing only the entries of B or H corresponding to basis functions of $K^\pm$. With these matrices the three face contributions in (8) can be written in terms of stiffness matrices K_{f1}, K_{f2}, K_{f3} for each face f:

$$-\int_\Gamma [\![\mathrm{v}]\!]:\{\sigma(\mathrm{u})\} \;=\; \sum_{f\in\mathcal{T}} \mathrm{V}^T \underbrace{\int_f -\mathrm{H}_f^{[\![]\!]^T}\mathrm{N}_f^T\bar{\mathrm{C}}\mathrm{B}_f^{\{\}}}_{\mathrm{K}_{f1}}\,\mathrm{U}, \tag{11}$$

$$-\int_\Gamma [\![\mathrm{u}]\!]:\{\sigma(\mathrm{v})\} \;=\; \sum_{f\in\mathcal{T}} \mathrm{V}^T \underbrace{\int_f -\mathrm{B}_f^{\{\}^T}\bar{\mathrm{C}}\mathrm{N}_f\mathrm{H}_f^{[\![]\!]}}_{\mathrm{K}_{f2}}\,\mathrm{U}, \tag{12}$$

$$\int_\Gamma \eta_f\,[\![\mathrm{u}]\!]:[\![\mathrm{v}]\!] \;=\; \sum_{f\in\mathcal{T}} \mathrm{V}^T \underbrace{\int_f \eta_f\,\mathrm{H}_f^{[\![]\!]^T}\mathrm{H}_f^{[\![]\!]}}_{\mathrm{K}_{f3}}\,\mathrm{U}. \tag{13}$$

The $3n\times 3n$ stiffness matrix K can therefore be assembled by doing one pass over all elements $K\in\mathcal{T}$ and accumulating their contributions K_K, and a second pass over all faces $f\in\mathcal{T}$ that accumulates their contributions K_{fi}. Equivalently to CG, the external force vector F is assembled from the elements' contributions $\int_K \mathrm{H}(\mathrm{x})^T\mathrm{f}$. Note that even for linear basis functions the integrands $\mathrm{H}(\mathrm{x})$ are not constant, requiring integration techniques as discussed in Section 6. The weak form $a(\mathrm{u},\mathrm{v}) = \mathrm{V}^T\mathrm{K}\mathrm{U} = \mathrm{V}^T\mathrm{F}$ has to hold for all test functions v, i.e., all vectors V, leading to the linear system $\mathrm{K}\mathrm{U} = \mathrm{F}$ to be solved for the *static* solution U.

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 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 g this means to define the function values on the "free" side of boundary faces $f\in\partial\Omega$ as

$$\mathrm{u}^- := \mathrm{g},\ \mathrm{v}^- := 0,\ \sigma^-(\mathrm{v}) := \sigma^+(\mathrm{v}),\ \sigma^-(\mathrm{u}) := \sigma^+(\mathrm{u}).$$

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

$$\mathrm{M}\ddot{\mathrm{U}} + \mathrm{D}\dot{\mathrm{U}} + \mathrm{K}\mathrm{U} \;=\; \mathrm{F}, \tag{14}$$

with mass matrix M and damping matrix D, equivalently as for CG FEM [NMK*06]. 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 [SvdV00] and sparse Cholesky factorization [TCR]. 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 [BBK05].

6. 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 in that case a linear, continuous displacement function u without jumps [Coc03].

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 [WBG07].

For a practical implementation we have to accurately and efficiently compute the 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 5. While tetrahedra or hexahedra can be integrated analytically, general polyhedral elements typically require numerical integration, which trades accuracy for performance [WBG07].

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 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 [Mir96], who initially proposed this approach for accurately computing mass properties of polyhedra. Expressions for higher order polynomials can be derived accordingly.

The resulting analytic integration is exact up to round-off-errors, and 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 [WBG07], our integration method is faster by an order of magnitude.

7. 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 [MG04, HS04].

In linear CG FEM, the forces acting on the nodes of an element K are computed from nodal displacements U and the element stiffness matrix K_K defined in (10) as follows:

$$\mathrm{F}_K \;=\; \mathrm{K}_K \mathrm{U} \;=\; \mathrm{K}_K \left(\mathrm{X} - \mathrm{X}^0\right) , \qquad (15)$$

with X and X^0 denoting the deformed and undeformed nodal positions, respectively. In order to avoid the aforementioned rotational artifacts, the *corotational*, or *warped stiffness* approach [MG04, HS04] first reverts the element's rotation, computes displacements and forces in the un-rotated state, and re-rotates the resulting forces:

$$\mathrm{F}_K \;=\; \mathrm{R}_K \mathrm{K}_K \left(\mathrm{R}_K^T \mathrm{X} - \mathrm{X}^0\right) , \qquad (16)$$

where 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 X^0 in (15) 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.

Element and Face Contributions. Element contributions (10) can be treated just as in CG FEM using (16). 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 R_K using polar decomposition [HS04].

Note that for face contributions (11), (12), (13) we cannot simply apply (16) using the *face's rotation*, since that would lead to ghost forces and instabilities similar to the per-vertex stiffness warping of [MDM*02]. Moreover, the corotational method is only required to correct artifacts due to the linear strain $\bar{\varepsilon} = \mathrm{BU}$, and therefore it is not needed for (13).

For the face contributions (11) and (12) it is crucial that the strains $\mathrm{B}_f^+\mathrm{U}$ and $\mathrm{B}_f^-\mathrm{U}$, which constitute $\mathrm{B}_f^{\{\}}$, are computed consistently with the strains of the element contributions (16) of K^+ and K^-. This requires to use the *elements' rotations* R_f^+ and R_f^- for correcting $\mathrm{B}_f^+\mathrm{U}$ and $\mathrm{B}_f^-\mathrm{U}$, respectively. We therefore have to split up the stiffness matrices K_{f1} and K_{f2} w.r.t. strain contributions from either K^+ or K^-, yielding the four stiffness matrices

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

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:

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

Non-Nodal Basis Functions. In order to use stiffness warping for non-nodal basis functions, we need to generalize the definition of the vector X^0 representing the undeformed state. To this end, we have to find $\mathrm{X}^0 = (\mathrm{x}_1^0, \ldots, \mathrm{x}_n^0)$ satisfying the identity $\sum_i \mathrm{x}_i^0 N_i(\mathrm{x}) \equiv \mathrm{x}$. 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 6), finding X^0 is trivial. For each element K, if its linear basis functions are

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

we simply set the corresponding coefficients to

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

and use $\mathrm{x}_{l_K}^0 = (0,0,0)^T$ for all its other basis functions. This results in a vector 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 10). 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.

8. 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 $\mathrm{u}(\mathrm{x})$ is used to deform the surface mesh (see, e.g., [FvdPT97, MTG04, JBT04, SSIF07]). In DG FEM, the *discontinuous* displacement 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 $\mathrm{x}_i \in \mathcal{T}$, the different displacements $\mathrm{u}|_K(\mathrm{x}_i)$ corresponding to its incident elements $K \in \mathcal{N}_i$, similar to [BPGK06]:

$$\tilde{\mathrm{u}}_i = \frac{1}{|\mathcal{N}_i|} \sum_{K \in \mathcal{N}_i} \mathrm{u}|_K(\mathrm{x}_i) \,. \tag{17}$$

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

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 [FKR05, JSW05] or harmonic coordinates [JMD*07] 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. 2, left).

Botsch et al. [BPWG07] 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 [MKN*04, PKA*05], we propose a smooth embedding based on moving-least-squares (MLS) interpolation.

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

$$\sum_i \theta(\|\mathrm{x}-\mathrm{x}_i\|) \left\| \mathrm{a}(\mathrm{x})^T \mathrm{p}(\mathrm{x}_i) - \tilde{\mathrm{u}}_i \right\|^2 , \tag{18}$$

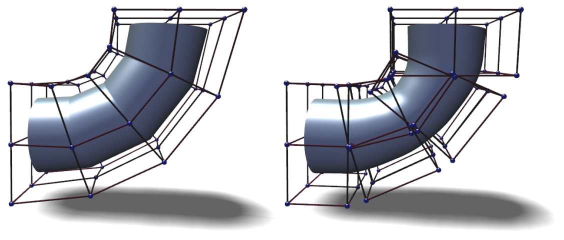

Figure 2: *Comparison of embedding techniques. Stitching the discontinuous simulation mesh, followed by barycentric interpolation, leads to C^0 artifacts (left). In contrast, our smooth MLS-based embedding yields a considerably higher surface quality (right).*

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

- The smoothness of the interpolation is determined by the weighting kernels w_i, resulting in a high quality embedding for our choice of Gaussian kernels (cf. Fig. 2, right).
- The use of linear polynomials $\mathrm{p}(\mathrm{x})$, in combination with the partition of unity property of MLS shape functions, guarantees the exact reproduction of linear displacements u, i.e., in particular of rigid motions [FM04].
- 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 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 u only requires to add more samples $(\mathrm{x}_i^0, \tilde{\mathrm{u}}_i)$ to (18), such as edge, face, or element midpoints.
- The interpolated displacement $\tilde{\mathrm{u}}(\mathrm{x})$ of a vertex x of the embedded mesh linearly depends on $\mathrm{a}(\mathrm{x})$, which in turn linearly depends on the $\tilde{\mathrm{u}}_i$ used in (18), which finally linearly depend on u_i through (17) and (4). Hence, by combining these linear relationships, the weights $w_i(\mathrm{x})$ as well as the set $\mathcal{N}(\mathrm{x})$ of relevant basis functions N_i can be precomputed, such that during the simulation only

$$\tilde{\mathrm{u}}(\mathrm{x}) = \sum_{i \in \mathcal{N}(\mathrm{x})} w_i(\mathrm{x}) N_i(\mathrm{x})\, \mathrm{u}_i =: \sum_{i \in \mathcal{N}(\mathrm{x})} W_i(\mathrm{x})\, \mathrm{u}_i \tag{19}$$

has to be evaluated as a linear combination of u_i.

9. 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 u.

Suppose that in the current time-step we detect a collision at a displaced material point $x_c + \tilde{u}(x_c)$. Since we use the interpolated displacement $\tilde{u}$ of (19), x_c can be an arbitrary embedded point, e.g., a vertex of the embedded surface mesh. Nodal collisions using the stitched displacement (17) 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 $f(x_c) = A \cdot \tilde{u}(x_c) + b$ with $A \in \mathbb{R}^{3\times3}$ and $b \in \mathbb{R}^3$. The corresponding penalty energy is

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

which after inserting the definition of $\tilde{u}$ in (19) becomes

$$\frac{1}{2}\sum_{i,j} u_i^T W_i(x_c)\,A\,W_j(x_c)\,u_j \;+\; \sum_i u_i^T W_i(x_c)\,b\,.$$

Since this collision energy corresponds to an external force, it has to be either subtracted from the internal potential energy $\frac{1}{2}U^T KU$ or to be added to the external energy $U^T F$. Hence, we can incorporate the collision energy E_{coll} into the system (14) by updating 3×3 blocks of the stiffness matrix K and 3-vectors of the external force F (see Section 5):

$$\begin{aligned} K_{ij} &\;-=\; W_i(x_c)\,A\,W_j(x_c)\,,\\ F_i &\;+=\; b\,W_i(x_c)\,,\end{aligned}$$

for all $i, j \in \mathcal{N}(x_c)$, i.e., the set of basis functions W_i, respectively N_i, influencing the collision point x_c (see (19)).

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 10. However, for embedded simulations collisions should be detected and handled on the vertices of the embedded surface (cf. Fig. 3).

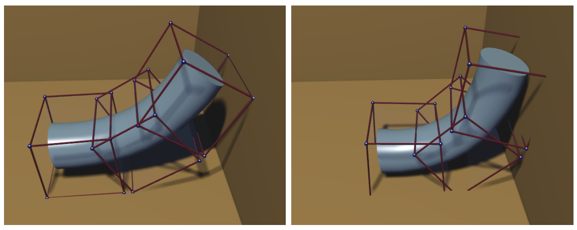
Figure 3: *Collisions handling on the nodes of the simulation mesh (left) and the vertices of the embedded mesh (right).*

10. 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
DG BZ lin.	10×10×10	12k	0.28%	532	22	656
DG IP lin.	10×10×10	12k	0.62%	1437	87	734
CG trilin.	15×15×15	12k	0.58%	3750	41	641
DG BZ quad.	10×10×10	30k	0.28%	3062	152	7797
DG IP quad.	10×10×10	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).*

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. 4). 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 [Coc03]. The only additional parameter compared to CG FEM is the penalty weight η in (7), which has to be sufficiently high to guarantee stability. We simply start with a low value and double it until 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. 1).

The missing consistency terms of BZ (cf. (6), (8)) allow for sparser matrices and higher efficiency. Furthermore, the method is stable for any positive penalty η. Although lacking theoretical convergence 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 [TEL06].

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.

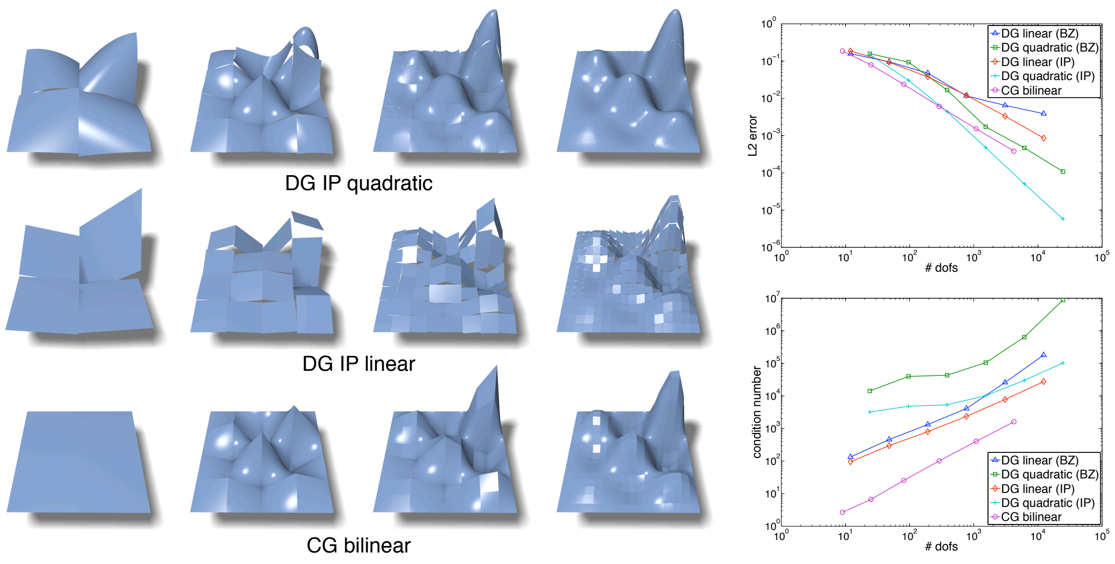

Figure 4: *Solution of the Poisson equation $-\Delta u = f$ on a regular quadrilateral grid of resolutions 2^2, 4^2, 8^2, and 16^2, using CG FEM and DG FEM. The plots compare the L^2 errors $\|\Delta u + f\|$ and the condition numbers of the stiffness matrix K for the BZ and IP method using linear and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

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. 4) 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.

Mesh Generation by Hexagonal 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. 5). 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.

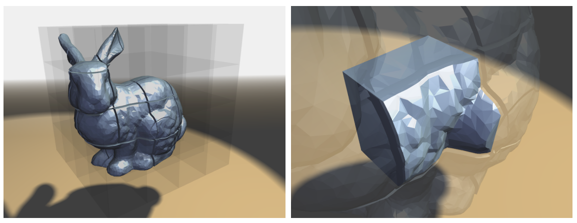

Figure 5: *Intersecting the bunny with a hexahedral grid generates 41 elements (left). Closeup view of a non-convex element (right).*

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 [GKS02]. 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. 6).

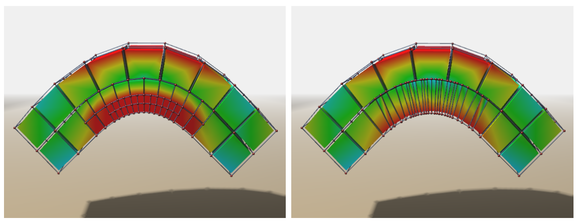

Figure 6: *A bar (36 hex-elements) is dynamically refined during bending. 1-to-8 subdivision results in 274 elements (left), whereas 1-to-2 refinement yields 77 elements (right).*

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 u_i and velocity $\dot{u}_i$ to its children. This straightforward heuristic causes the slight popping artifacts visible in the video, which could probably be avoided by a more sophisticated technique.

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. 7), similar in spirit to [MBF04, WBG07, SDF07]. 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.

11. 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

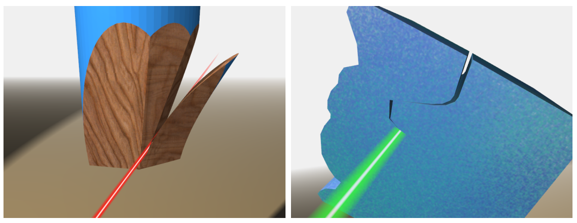

Figure 7: *Sharpening a pencil consisting of a single convex element (left). Cutting a bunny out of a cube (right).*

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. This research was supported by the Swiss National Science Foundation through grant 200021-117756.

References

[ABCM01] Arnold D. N., Brezzi F., Cockburn B., Marini L. D.: Unified analysis of discontinuous Galerkin methods for elliptic problems. *SIAM J. Numer. Anal. 39*, 5 (2001), 1749–1779.

[Bat95] Bathe K.-J.: *Finite Element Procedures*. Prentice Hall, 1995.

[BBK05] Botsch M., Bommes D., Kobbelt L.: Efficient linear system solvers for geometry processing. In *11th IMA conference on the Mathematics of Surfaces* (2005).

[BG00] Bielser D., Gross M.: Interactive simulation of surgical cuts. In *Proc. of Pacific Graphics* (2000), pp. 116–125.

[BGTG03] Bielser D., Glardon P., Teschner M., Gross M.: A state machine for real-time cutting of tetrahedral meshes. In *Proc. of Pacific Graphics* (2003), pp. 377–386.

[BPGK06] Botsch M., Pauly M., Gross M., Kobbelt L.: PriMo: Coupled prisms for intuitive surface modeling. In *Proc. of Symp. on Geometry Processing* (2006), pp. 11–20.

[BPWG07] Botsch M., Pauly M., Wicke M., Gross M.: Adaptive space deformations based on rigid cells. *Computer Graphics Forum (Proc. Eurographics) 26*, 3 (2007), 339–347.

[BWHT07] Bargteil A. W., Wojtan C., Hodgins J. K., Turk G.: A finite element method for animating large viscoplastic flow. *ACM Trans. on Graphics (Proc. SIGGRAPH) 26*, 3 (2007), 16.1–16.8.

[BZ73] Babuška I., Zlámal M.: Nonconforming elements in the finite element method with penalty. *SIAM J. Numer. Anal. 10* (1973), 863–875.

[CGC*02] Capell S., Green S., Curless B., Duchamp T., Popović Z.: A multiresolution framework for dynamic deformations. In *Proc. of Symp. on Computer Animation* (2002), pp. 41–47.

[Coc03] COCKBURN B.: Discontinuous Galerkin methods. *Z. Angew. Math. Mech. 80*, 11 (2003), 731–754.

[DD76] DOUGLAS J., DUPONT T.: Interior penalty procedures for elliptic and parabolic Galerkin methods. *Computing Methods in Applied Science, Lecture Notes in Physics 58* (1976).

[DDCB01] DEBUNNE G., DESBRUN M., CANI M.-P., BARR A. H.: Dynamic real-time deformations using space and time adaptive sampling. In *Proc. of ACM SIGGRAPH* (2001), pp. 31–36.

[FKR05] FLOATER M. S., KOS G., REIMERS M.: Mean value coordinates in 3D. *Computer Aided Geometric Design 22*, 7 (2005), 623–631.

[FM04] FRIES T. P., MATTHIES H. G.: *Classification and overview of meshfree methods*. Informatikbericht 2003-03, revised 2004, Institute of Scientific Computing, Technical University Braunschweig, 2004.

[FvdPT97] FALOUTSOS P., VAN DE PANNE M., TERZOPOULOS D.: Dynamic free-form deformations for animation synthesis. *IEEE Transactions on Visualization and Computer Graphics 3*, 3 (1997), 201–214.

[GKS02] GRINSPUN E., KRYSL P., SCHRÖDER P.: CHARMS: A simple framework for adaptive simulation. *ACM Trans. on Graphics (Proc. SIGGRAPH) 21*, 3 (2002), 281–290.

[HL02] HANSBO P., LARSON M.: Discontinuous Galerkin methods for incompressible and nearly incompressible elasticity by Nitsche's method. *Comput. Methods Appl. Mech. Eng. 191*, 17 (2002), 1895–1908.

[HS04] HAUTH M., STRASSER W.: Corotational simulation of deformable solids. In *Proc. of WSCG* (2004), pp. 137–145.

[Hug00] HUGHES T. J. R.: *The Finite Element Method. Linear Static and Dynamic Finite Element Analysis*. Dover Publications, 2000.

[JBT04] JAMES D., BARBIČ J., TWIGG C.: Squashing cubes: Automating deformable model construction for graphics. In *Proc. of SIGGRAPH '04 Sketches and Applications* (2004).

[JMD*07] JOSHI P., MEYER M., DEROSE T., GREEN B., SANOCKI T.: Harmonic coordinates for character articulation. *ACM Trans. on Graphics (Proc. SIGGRAPH) 26*, 3 (2007).

[JSW05] JU T., SCHAEFER S., WARREN J.: Mean value coordinates for closed triangular meshes. *ACM Trans. on Graphics (Proc. SIGGRAPH) 24*, 3 (2005), 561–566.

[LNSO04] LEW A., NEFF P., SULSKY D., ORTIZ M.: Optimal BV estimates for a discontinuous Galerkin method in linear elasticity. *Applied Mathematics Research Express*, 3 (2004), 73–106.

[MBF04] MOLINO N., BAO Z., FEDKIW R.: A virtual node algorithm for changing mesh topology during simulation. *ACM Trans. on Graphics (Proc. SIGGRAPH) 23*, 3 (2004), 385–392.

[MDM*02] MÜLLER M., DORSEY J., MCMILLAN L., JAGNOW R., CUTLER B.: Stable real-time deformations. In *Proc. of Symp. on Computer Animation* (2002), pp. 163–170.

[MG04] MÜLLER M., GROSS M.: Interactive virtual materials. In *Proc. of Graphics Interface* (2004), pp. 239–246.

[Mir96] MIRTICH B.: Fast and accurate computation of polyhedral mass properties. *Journal of Graphics Tools 1*, 2 (1996), 31–50.

[MKN*04] MÜLLER M., KEISER R., NEALEN A., PAULY M., GROSS M., ALEXA M.: Point-based animation of elastic, plastic and melting objects. In *Proc. of Symp. on Computer Animation* (2004), pp. 141–151.

[MTG04] MÜLLER M., TESCHNER M., GROSS M.: Physically based simulation of objects represented by surface meshes. In *Proc. of Computer Graphics International* (2004), pp. 26–33.

[NMK*06] NEALEN A., MÜLLER M., KEISER R., BOXERMAN E., CARLSON M.: Physically based deformable models in computer graphics. *Computer Graphics Forum 25*, 4 (2006), 809–836.

[OBH02] O'BRIEN J. F., BARGTEIL A. W., HODGINS J. K.: Graphical modeling and animation of ductile fracture. *ACM Trans. on Graphics (Proc. SIGGRAPH) 21*, 3 (2002), 291–294.

[OGRG07] OTADUY M. A., GERMANN D., REDON S., GROSS M.: Adaptive deformations with fast tight bounds. In *Proc. of Symp. on Computer Animation* (2007), pp. 181–190.

[OH99] O'BRIEN J. F., HODGINS J. K.: Graphical modeling and animation of brittle fracture. In *Proc. of ACM SIGGRAPH* (1999), pp. 137–146.

[PKA*05] PAULY M., KEISER R., ADAMS B., DUTRE P., GROSS M., GUIBAS L. J.: Meshless animation of fracturing solids. *ACM Trans. on Graphics (Proc. SIGGRAPH) 24*, 3 (2005), 957–964.

[SDF07] SIFAKIS E., DER K. G., FEDKIW R.: Arbitrary cutting of deformable tetrahedralized objects. In *Proc. of Symp. on Computer Animation* (2007), pp. 73–80.

[SHGS06] STEINEMANN D., HARDERS M., GROSS M., SZEKELY G.: Hybrid cutting of deformable solids. In *Proc. of IEEE VR* (2006), pp. 35–42.

[SOG06] STEINEMANN D., OTADUY M. A., GROSS M.: Fast arbitrary splitting of deforming objects. In *Proc. of Symp. on Computer Animation* (2006), pp. 63–72.

[SSIF07] SIFAKIS E., SHINAR T., IRVING G., FEDKIW R.: Hybrid simulation of deformable solids. In *Proc. of Symp. on Computer Animation* (2007), pp. 81–90.

[SvdV00] SAAD Y., VAN DER VORST H. A.: Iterative solution of linear systems in the 20th century. *J. Comput. Appl. Math. 123*, 1-2 (2000), 1–33.

[TCR] TOLEDO S., CHEN D., ROTKIN V.: Taucs: A library of sparse linear solvers. http://www.tau.ac.il/∼stoledo/taucs.

[TEL06] TEN EYCK A., LEW A.: Discontinuous Galerkin methods for non-linear elasticity. *International Journal for Numerical Methods in Engineering 67*, 9 (2006), 1204–1243.

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

[WBG07] WICKE M., BOTSCH M., GROSS M.: A finite element method on convex polyhedra. *Computer Graphics Forum (Proc. Eurographics) 26*, 3 (2007), 355–364.

[Wih06] WIHLER T. P.: Locking-free adaptive discontinuous Galerkin FEM for linear elasticity problems. *Mathematics of Computation 75*, 255 (2006), 1087–1102.