SIAM J. Sci. Comput. © 2019 Society for Industrial and Applied Mathematics
Vol. 41, No. 5, pp. S433–S461

SUBDIVISION-BASED NONLINEAR MULTISCALE CLOTH SIMULATION[*]

ALENA KOPANIČÁKOVÁ[†] [‡], ROLF KRAUSE[†], AND RASMUS TAMSTORF[‡]

Abstract. Cloth simulation is an important topic for many applications in computer graphics, animation, and augmented virtual reality. The mechanical behavior of cloth objects can be modeled by the Kirchhoff–Love thin shell equations, which lead to large-scale, nonlinear, ill-conditioned algebraic equations. We propose to solve these nonlinear problems efficiently using the recursive multilevel trust region (RMTR) method. Our multilevel framework for cloth simulations is based on Catmull–Clark subdivision surfaces, which facilitates generation of the mesh hierarchy and also provides the basis for the finite element discretization. The prolongation and restriction operators are similarly constructed based on the subdivision rules. Finally, we leverage a reverse subdivision operator to transfer iterates from fine levels to coarser levels. The novel use of this fine-to-coarse operator provides a computationally efficient alternative to the least-square approach used elsewhere. Using the resulting RMTR variant, we present numerical examples showing a reduction in the number of iterations by several orders of magnitude when compared to a single-level trust region method.

Key words. thin shell cloth simulation, isogeometric analysis, subdivision surfaces, trust region methods, multilevel optimization

AMS subject classifications. 74K25, 74B20, 65M55, 65K10

DOI. 10.1137/18M1194870

1. Introduction.

Cloth simulation is a well-established topic in computer graphics. It is essential for creating compelling animated characters, and it is also increasingly important for virtual try-on applications in e-commerce. Convincing visual results often require high levels of detail to capture wrinkles and folding patterns, but these results can be computationally expensive to obtain, and faster simulations are often achieved at the expense of realism by using simplified physical models. The goal of this paper is to accelerate cloth simulations without degrading the accuracy of the simulation. In particular, we focus on cloth modeled as orthotropic Kirchhoff–Love thin shells. However, the results obtained also apply to other applications that depend on thin shell simulation with large strains and displacements.

The partial differential equations for the Kirchhoff–Love model are strongly nonlinear. A standard approach for dealing with the nonlinearities is Newton's method which is quadratically convergent close to the solution, but its convergence is erratic when a poor initial guess is supplied and no globalization strategy is employed. Numerical evidence indicates that Newton's method alone may not converge, unless extremely small continuation steps are taken [1]. Furthermore, each Newton step requires solving a large linear system. Given the fourth-order nature of the Kirchhoff–Love equations, the condition number of this linear system scales as $\mathcal{O}(h^{-4})$, where h represents the mesh size. Consequently, solving large systems arising from fine

[*]Received by the editors June 18, 2018; accepted for publication (in revised form) July 25, 2019; published electronically October 29, 2019.
https://doi.org/10.1137/18M1194870

Funding: This work was partly supported by Swiss National Science Foundation (SNF) and DFG (Deutsche Forschungsgemeinschaft) under the project SPP1748 "Reliable Simulation Techniques in Solid Mechanics. Development of Nonstandard Discretization Methods, Mechanical and Mathematical Analysis."

[†]Institute of Computational Science, Università della Svizzera italiana, Lugano 6900, Switzerland (alena.kopanicakova@usi.ch, rolf.krause@usi.ch).
[‡]Walt Disney Animation Studios, Burbank, CA 91521 (rasmus.tamstorf@disneyanimation.com).

discretizations, which allow for high levels of detail, can be prohibitive.

Globalization strategies, such as line-search [20], trust region methods [19], or adaptive cubic regularization strategies [12], provide a remedy for Newton's method. However, they do not address the ill-conditioning of the arising linear systems. In this work, we employ a trust region method inside of a multilevel framework to address both of the above problems. More specifically, we use the recursive multilevel trust region (RMTR) strategy [24, 28]. This method internally employs models on multiple levels and provides global convergence guarantees based on multiscale optimization. We note here that for multiscale methods, the use of a globalization strategy is particularly important since the solution obtained on a coarse level may not be within the basin of attraction for Newton's method on the next finer level.

To generate multiple levels of detail, we leverage the Catmull–Clark subdivision scheme and the corresponding subdivision finite elements. In feature films, cloth objects are typically modeled and rendered as subdivision surfaces, but often simulated using triangle meshes and (approximations of) linear finite elements. However, using different discretizations at different stages requires multiple stages of remeshing, which leads to a time-consuming set-up process and reduces the overall accuracy of the solutions. By using an "isogeometric analysis" approach based on subdivision surfaces, we avoid this problem as advocated in [32]. The subdivision scheme also provides a natural way to generate the multilevel hierarchy required for the RMTR method. In the end, we therefore incorporate the subdivision scheme in four ways: to describe the geometry, to describe the displacements/solution, to connect different levels of the multilevel hierarchy, and to produce the final rendered surface.

Our primary contribution is to show that the recursive multilevel trust region (RMTR) method can be used effectively in the context of cloth simulations. The existing literature offers several linear geometric and algebraic multilevel methods, but to our knowledge, the RMTR method has not previously been used with isogeometric analysis for thin shells or cloth. We do not consider collision handling in this paper, although it is important for cloth simulation, but it can be incorporated into the proposed method and we hope to address it in future work.

The paper is organized as follows: Section 2 provides an overview of related work. The model we use for cloth simulation is introduced in section 3, while section 4 describes the RMTR algorithm, and section 5 introduces subdivision surfaces. In particular, we describe how to construct a subdivision-based multilevel hierarchy, how to employ the subdivision-based multilevel discretization, and how to assemble the subdivision-based transfer operators. We comment on the implementation aspects of the described multilevel method in section 6. Section 7 reports on the performance of the RMTR algorithm based on a convergence study for four examples. We also provide a comparison of the RMTR method with a single-level trust region method. Some conclusions and perspectives are finally discussed in section 8.

2. Related work.

Much cloth simulation in graphics is based on the seminal paper by Baraff and Witkin [4] and later work by Bridson and colleagues [7, 8]. Subdivision finite elements were first introduced by Cirak, Ortiz, and Schröder [15] around the same time for linear elasticity and extended to nonlinear elasticity in [14], and later this was used for cloth simulation in [60]. More recently, Clyde, Teran, and Tamstorf [17] used a subdivision finite element discretization of cloth to estimate material parameters based on a secant method (BFGS).

Independently, linear multigrid methods have been developed for more than 50 years and are used extensively in many areas ranging from simple Poisson equations to quantum chromodynamics problems. An introduction to these methods can be found in, for example, [9, 29]. Linear multilevel methods have also been applied in different contexts related to shell/subdivision/cloth simulations. For example, Green, Turkiyyah, and Storti [26] show speedups in solution time for subdivision shell simulations by applying a geometric multigrid method, and Tamstorf, Jones, and McCormick [57] apply smoothed aggregation algebraic multigrid to achieve speedups for cloth simulations without leveraging subdivision methods.

Nonlinear generalizations of linear multigrid methods such as the full approximation scheme [6] and nonlinear multigrid [29, 30, 41] compute coarse-level corrections based on nonlinear representation of the original objective function. The idea of employing optimization strategies inside of a multilevel framework can be traced back to Nash [45], where the author introduces a line-search–based method called MG/OPT. In the context of shell problems, a full approximation scheme is applied by Gee and Tuminaro in [22], and Bandara and Cirak [3] employ a subdivision-based multilevel framework for a gradient-based shape optimization. Although all of these nonlinear multilevel algorithms yield remarkable performance, the algorithms do not provide global convergence properties [40].

The work by Gratton, Sartenaer, and Toint [24] suggests using a trust region strategy inside of a multilevel framework, which results in a globally convergent multiscale method for nonconvex minimization problems. Gross and Krause [28] prove that this method is second-order convergent. They also suggest replacing the commonly used restriction operator for transferring iterates from a fine to a coarse level by a projection operator in order to accelerate convergence.

Alternatives, in a more parallel spirit, consider nonlinear domain-decomposition methods, e.g., the additively preconditioned inexact Newton (ASPIN) method [10], nonlinear elimination strategies [11], or nonlinear finite element tearing and interconnect (FETI) method [39]. Combinations of multilevel and domain-decomposition strategies provide another interesting possibility and were successfully applied in [33, 27, 42].

As a different approach to accelerating cloth simulations, numerous papers have considered various parallelization strategies using both CPUs [36, 59] and GPUs [58]. We include some basic parallelization in our implementation, but more work can certainly be done to optimize our results.

3. Thin shell cloth simulation.

Cloth animation amounts to numerically simulating the motion of a piece of cloth in a three-dimensional (3D) environment. A cloth object is often subjected to external forces such as gravity or wind, and additionally a set of prescribed boundary conditions that have to be satisfied at each time step. Figure 1 illustrates a simple example, where a piece of cloth is subjected to the gravitational force and time-dependent boundary conditions.

In this work, we model cloth objects by employing shell kinematics, as the cloth thickness is significantly smaller than its other geometric dimensions. The piece of cloth is modeled in terms of a two-dimensional (2D) midsurface with constant thickness $\tau \in \mathbb{R}$. The deformed midsurface is represented by a map $\boldsymbol{x} : \omega \to \Omega$ from a 2D parameter space ω to worldspace $\Omega \subset \mathbb{R}^3$. A volumetric cloth object is described by a map $\boldsymbol{r} : \omega^\tau \to \Omega^\tau$, where ω^τ is the 3D parameter space and $\Omega^\tau \subset \mathbb{R}^3$ is the region occupied by the shell. Both maps $\boldsymbol{x}$ and $\boldsymbol{r}$ are time-dependent, i.e., $\boldsymbol{x} = \boldsymbol{x}(t)$ and $\boldsymbol{r} = \boldsymbol{r}(t)$, where t denotes time.

The kinematics of thin shells can be expressed by using Kirchhoff–Love theory [38], which assumes that straight lines initially normal to the midsurface remain straight

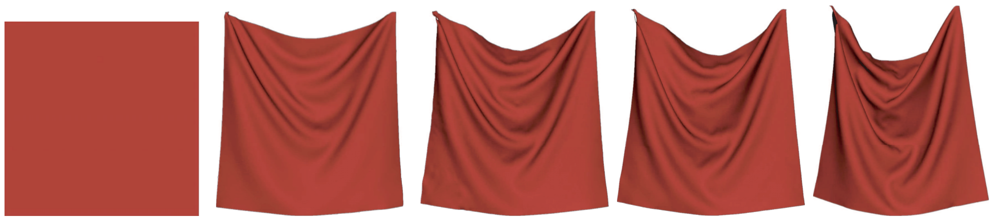


FIG. 1. *Cloth animation of* $1m \times 1m$ *piece of fabric. The opposite top corners are moved closer to each other, until their distance reaches* $0.6m$.

and normal during deformation. Equivalently, this means that the transverse shear strain must be zero. This assumption directly implies that

$$\boldsymbol{r}\,(\xi^1,\xi^2,\xi^3) = \boldsymbol{x}\,(\xi^1,\xi^2) + \xi^3\boldsymbol{a}_3(\xi^1,\xi^2), \quad -\frac{\tau}{2} \leq \xi^3 \leq \frac{\tau}{2}, \tag{3.1}$$

where $\boldsymbol{\xi} = (\xi^1,\xi^2,\xi^3)$ represent curvilinear coordinates and $\boldsymbol{a}_3(\xi^1,\xi^2)$ is the unit normal to the deformed midsurface.

The equilibrium configuration of the cloth can be found by minimizing the total mechanical energy

$$\psi^{\mathrm{Mech}}(\boldsymbol{x}) := \underbrace{\frac{1}{2}\int_{\omega^\tau} \hat{\rho}\ \|\dot{\boldsymbol{x}} + \xi^3\dot{\boldsymbol{a}}_3\|^2 \bar{J}\ d\boldsymbol{\xi}}_{:=\psi^{\mathrm{Kin}}} + \underbrace{\int_{\omega^\tau} \hat{\rho}\ \boldsymbol{g}^T\boldsymbol{r} + \psi^E\ \bar{J}\ d\boldsymbol{\xi}}_{:=\psi^{\mathrm{Pot}}}, \tag{3.2}$$

which is defined as the sum of kinetic energy ψ^{Kin} and potential energy ψ^{Pot}. Here, we use overbar notation for quantities related to the undeformed configuration and overdot notation to indicate time derivatives. The symbol $\bar{J}$ denotes the Jacobian determinant $|\frac{\partial\bar{\boldsymbol{r}}}{\partial\boldsymbol{\xi}}|$, and $\hat{\rho} = \bar{\rho}\circ\bar{\boldsymbol{r}}$ is the density pullback. The total potential energy, ψ^{Pot}, consists of two terms. The first term represents a gravitational force defined by the gravity field $\boldsymbol{g} \in \mathbb{R}^3$. The second term, ψ^E, denotes the hyperelastic energy density which describes an orthotropic elastic material model and is defined as follows:

$$\psi^E(\boldsymbol{E},\boldsymbol{O},\boldsymbol{k}) := \frac{k_1}{2}\varphi_1(\tilde{E}_{11}^2) + k_2\varphi_2(\tilde{E}_{11}\tilde{E}_{22}) + \frac{k_3}{2}\varphi_3(\tilde{E}_{22}^2) + k_4\varphi_4(\tilde{E}_{12}^2), \tag{3.3}$$

where the vector $\boldsymbol{k} = [k_1,k_2,k_3,k_4]$ contains the material parameters. The $\tilde{E}_{jj}$ represents an element of the reduced Green–Lagrange strain tensor $\tilde{\boldsymbol{E}} = \boldsymbol{O}^T\boldsymbol{E}\boldsymbol{O}$, where $\boldsymbol{O} = [\boldsymbol{o}_1,\boldsymbol{o}_2,\boldsymbol{o}_3]$ provides an orthotropic basis corresponding to the normalized material warp, weft, and normal directions. The functions φ_i describe nonlinear material response and are defined as

$$\varphi_i(u) = \sum_{i=1}^{d_j} \frac{\mu_{ji}}{\alpha_{ji}}((u+1)^{\alpha_{ji}} - 1), \tag{3.4}$$

where d_j, α_{ji}, and μ_{ji} are also material parameters. A detailed description of the described model (3.3) can be found in [17, 18]. In particular, it can be shown that we must have $\boldsymbol{x} \in H^2(\omega \to \mathbb{R}^3)$ in order to satisfy the zero transverse shear strain constraint [37].

3.1. Minimization problem. The unknown deformation is defined as a minimizer of the following minimization problem:

$$\begin{aligned} \boldsymbol{x} = & \underset{\boldsymbol{x}\,\in\, H^2(\omega\to\mathbb{R}^3)}{\operatorname{argmin}} \quad \psi^{\text{Mech}}(\boldsymbol{x}) \\ & \text{subject to} \qquad \boldsymbol{x}(\xi^1,\xi^2) = \boldsymbol{b}(\xi^1,\xi^2) \quad \text{on } \partial\omega. \end{aligned} \tag{3.5}$$

The equality constraint in (3.5) enforces the boundary conditions $\boldsymbol{b}(\xi^1,\xi^2)$ for all parameter space points (ξ^1,ξ^2) along the boundary segment $\partial\omega$.

In this work, we impose the boundary conditions (3.5) by using a penalty method [2]. In particular, we expand the objective function from (3.5) by adding a penalty term as

$$\tilde{\psi}^{\text{Mech}}(\boldsymbol{x}) := \psi^{\text{Mech}}(\boldsymbol{x}) + \frac{\beta}{2}\|\boldsymbol{x}-\boldsymbol{b}\|^2_{L^2(\partial\omega^\tau)}, \tag{3.6}$$

where $\beta > 0$ is the penalty parameter, and $\|\cdot\|_{L^2(\partial\omega^\tau)}$ denotes the L^2 norm on the boundary. We employ $\beta = h^{-\sigma}$, where h represents the mesh size and $\sigma > 0$ is a constant; see [2, 5].

The solution $\boldsymbol{x}$ of the constrained minimization problem (3.5) is then approximated by the minimizer $\tilde{\boldsymbol{x}}$ of the following unconstrained minimization problem:

$$\boldsymbol{x} \approx \tilde{\boldsymbol{x}} = \underset{\tilde{\boldsymbol{x}}\,\in\, H^2(\omega\to\mathbb{R}^3)}{\operatorname{argmin}} \quad \tilde{\psi}^{\text{Mech}}(\tilde{\boldsymbol{x}}). \tag{3.7}$$

The solution $\tilde{\boldsymbol{x}}$ of (3.7) approximates $\boldsymbol{x}$ well for sufficiently large values of β, and $\tilde{\boldsymbol{x}} \to \boldsymbol{x}$ as $\beta \to \infty$. While the boundary conditions are not strictly satisfied, their violation does not affect the space of admissible solutions since the additional term in (3.7) is clearly finite when $\boldsymbol{x} \in H^2 \subset L^2$. However, the violation of the boundary conditions does influence convergence rates of the finite element method under refinement. Inspired by [15] and [25], we satisfy the H^2 requirement by using subdivision basis functions for the finite element discretization; cf. section 5.2.

As an alternative to the penalty method, the boundary conditions could be imposed using the method of Lagrange multipliers [25, 17] or by using Nitsche's method [35]. The use of the Lagrange multipliers leads to constrained minimization problems with a saddle point structure, which complicates the solution process, and the proper derivation of Nitsche's method for shells is beyond the scope of this paper. Hence, our decision to use the penalty method.

3.1.1. First-order optimality conditions. The minimization of (3.7) is an optimization problem with the following optimality conditions:

Find $\boldsymbol{x} \in H^2(\omega \to \mathbb{R}^3)$ satisfying the first-order optimality condition

$$\nabla_{\boldsymbol{x}}\tilde{\psi}^{\text{Mech}}(\boldsymbol{x};\boldsymbol{v}) = 0 \quad \forall \boldsymbol{v} \in H^2(\omega \to \mathbb{R}^3). \tag{3.8}$$

The directional derivative of the energy with respect to $\boldsymbol{x}$ is defined as

$$\begin{aligned} \nabla_{\boldsymbol{x}}\tilde{\psi}^{\text{Mech}}(\boldsymbol{x};\boldsymbol{v}) = & \int_{\omega^\tau} \hat{\rho}\, \ddot{\boldsymbol{x}}^T \boldsymbol{v} \bar{J} d\boldsymbol{\xi} - \int_{\omega^\tau} \bigg(\frac{\partial\psi^E}{\partial\boldsymbol{x}_{,1}}\boldsymbol{v}_{,1} + \frac{\partial\psi^E}{\partial\boldsymbol{x}_{,2}}\boldsymbol{v}_{,2} + \frac{\partial\psi^E}{\partial\boldsymbol{x}_{,11}}\boldsymbol{v}_{,11} \\ & + \frac{\partial\psi^E}{\partial\boldsymbol{x}_{,12}}\boldsymbol{v}_{,12} + \frac{\partial\psi^E}{\partial\boldsymbol{x}_{,22}}\boldsymbol{v}_{,22} + \hat{\rho}\boldsymbol{g}^T\boldsymbol{v}\bigg)\bar{J}d\boldsymbol{\xi} + \int_{\partial\omega^\tau} \beta\Big(\boldsymbol{x}-\boldsymbol{b}\Big)^T \boldsymbol{v}\bar{J}d\boldsymbol{\xi}, \end{aligned} \tag{3.9}$$

where we use comma notation to denote partial derivatives, e.g., $\boldsymbol{x}_{,\alpha} = \frac{\partial\boldsymbol{x}}{\partial\xi^\alpha}$. The symbol $\ddot{\boldsymbol{x}}$ in (3.9) denotes the second derivative of $\boldsymbol{x}$ with respect to time t. For simplicity, we discard the term $\xi^3 \dot{\boldsymbol{a}}_3$ from (3.2) in (3.9), as suggested in [15, 37, 17]. A detailed derivation of the weak form (3.9) can be found in [18] and [16].

In order to solve the minimization problem (3.7) numerically, we discretize the Euler–Lagrange equations (3.9) in space by the finite element method (FEM) and in time by the implicit Euler method. Solving the minimization problem (3.7) is challenging, because the functional (3.2) is nonconvex and may therefore admit many local minimizers. In addition, as mentioned in the introduction, the condition number of the resulting linear systems is of order $\mathcal{O}(h^{-4})$, where h represents the mesh size; see [56, Theorem 5.1]. This condition number can in theory increase additionally due to the penalty term introduced in (3.6). We discuss the effect of the penalization term on the condition number based on a numerical example in section 7.1. This work addresses both nonconvexity and ill-conditioning by employing the RMTR algorithm [24, 28]; see section 4.

4. Recursive multilevel trust region.

Our goal is to minimize the energy functional in (3.5) by using the RMTR method [24, 23, 28]. This particular method provides several benefits. On one hand, trust region–based globalization ensures convergence to first-order critical points. On the other hand, the multilevel framework addresses problems with ill-conditioning and many degrees of freedom. Our implementation of the RMTR method follows [28]. Extended details about the method and its convergence properties can also be found in [24, 23]. The monograph [19] offers a comprehensive introduction of trust region methods.

4.1. Trust region method.

The trust region (TR) method minimizes an objective function $f : \mathbb{R}^n \to \mathbb{R}$ by producing a sequence $\{\boldsymbol{x}_i\}$ of iterates converging to a first-order critical point, i.e., $\boldsymbol{g}(\boldsymbol{x}) := \nabla f(\boldsymbol{x}) = 0$. In this work, the objective function f denotes the energy functional (3.6), and the first-order critical points satisfy the optimality conditions (3.8).

At each iterate i, the trust region method approximates f by a quadratic model m_i around the current iterate $\boldsymbol{x}_i \in \mathbb{R}^n$. The model m_i is assumed to be adequate only in a "trust region," specified around current iterate $\boldsymbol{x}_i$ with radius $\Delta_i > 0$, defined by the ℓ_2 norm. A correction $\boldsymbol{s}_i$ is then computed as an approximate solution to the following constrained quadratic minimization problem:

$$\begin{aligned} &\underset{\boldsymbol{s}_i \in \mathbb{R}^n}{\text{minimize}} \quad m_i(\boldsymbol{s}_i) = f_i + \boldsymbol{g}_i^T \boldsymbol{s}_i + \tfrac{1}{2} \boldsymbol{s}_i^T \mathbf{H}_i \boldsymbol{s}_i \\ &\text{subject to} \quad \|\boldsymbol{s}_i\| \le \Delta_i, \end{aligned} \tag{4.1}$$

where $\mathbf{H}_i := \nabla^2 f(\boldsymbol{x}_i)$. Trust region convergence theory does not require exact solution of (4.1). However, the approximate solution $\boldsymbol{s}_i$ of (4.1) has to fulfill the following sufficient decrease condition [19]:

$$m_i(\boldsymbol{x}_i) - m_i(\boldsymbol{x}_i + \boldsymbol{s}_i) \ge \kappa_r \|\boldsymbol{g}_i\| \min \left[\frac{\|\boldsymbol{g}_i\|}{1 + \|\mathbf{H}_i\|}, \Delta_i \right], \tag{SDC}$$

where $\kappa_r \in (0, 1)$.

In contrast to line-search algorithms, the iterate obtained in this way (also called the trial point), $\boldsymbol{x}_i + \boldsymbol{s}_i$, is not used immediately. The trial point is accepted only if the ratio of actual and predicted reduction

$$\rho_i = \frac{f(\boldsymbol{x}_i) - f(\boldsymbol{x}_i + \boldsymbol{s}_i)}{m_i(\mathbf{0}) - m_i(\boldsymbol{s}_i)} = \frac{\text{actual reduction}}{\text{predicted reduction}} \tag{4.2}$$

is larger than some constant $\eta_1 \in (0, 1)$. If the trial point is accepted, then the trust region radius is increased. Otherwise it is decreased or left unchanged. Algorithm 4.1 summarizes this process.

Algorithm 4.1. Local_TR.

```
Require: $f : \mathbb{R}^n \to \mathbb{R},\ \boldsymbol{x}_0 \in \mathbb{R}^n,\ \Delta_0,\ \epsilon_g \in \mathbb{R},\ i_{\max} \in \mathbb{N}$
Constants: $\eta_1 \in \mathbb{R}$, where $0 < \eta_1 < 1$
1: for $i = 0, \ldots, i_{\max}$ do
2:     $\min_{\|\boldsymbol{s}_i\| \le \Delta_i} m_i(\boldsymbol{s}_i) = f_i + \boldsymbol{g}_i^T \boldsymbol{s}_i + \frac{1}{2}\boldsymbol{s}_i^T \mathbf{H}_i \boldsymbol{s}_i$        ▷ Solve constrained QP problem
3:     $\rho_i = \frac{f(\boldsymbol{x}_i) - f(\boldsymbol{x}_i + \boldsymbol{s}_i)}{m_i(0) - m_i(\boldsymbol{s}_i)}$        ▷ Trust region ratio computation
4:     if $\rho_i > \eta_1$ then
5:         $\boldsymbol{x}_{i+1} := \boldsymbol{x}_i + \boldsymbol{s}_i$        ▷ Trial point acceptance
6:     else
7:         $\boldsymbol{x}_{i+1} := \boldsymbol{x}_i$        ▷ Trial point rejection
8:     end if
9:     $\Delta_{i+1} = \text{Radius\_update}(\rho_i, \Delta_i)$
10: end for
11: return $\boldsymbol{x}_{i+1}, \Delta_{i+1}$
```

4.2. Multilevel framework. Multilevel methods provide solvers of optimal complexity by using a hierarchy $\{\mathcal{X}^l\}_{l=0}^L$ of usually nested finite element spaces. Here, we use the subscript l to denote the level and assume $L \ge 1$ to be the finest level. The transfer of data between subsequent levels is done by means of three transfer operators. The prolongation operator $\mathbf{I}_l^{l+1} : \mathcal{X}^l \to \mathcal{X}^{l+1}$ is designed to transfer primal variables, such as corrections from a coarse level to a finer level. The restriction operator $\mathbf{R}_{l+1}^l : \mathcal{X}^{l+1} \to \mathcal{X}^l$ transfers dual variables, e.g., gradients, from a fine level to a coarser level. The operators $\mathbf{R}_{l+1}^l$ and $\mathbf{I}_l^{l+1}$ are related by the relation $\mathbf{R}_{l+1}^l = (\mathbf{I}_l^{l+1})^T$. The projection operator $\mathbf{P}_{l+1}^l : \mathcal{X}^{l+1} \to \mathcal{X}^l$ transfers primal variables from a finer level to a coarser level.

The computational cost of minimizing the original objective function can be reduced by employing simplified model problems. As is common for nonlinear multilevel schemes, e.g., FAS [6], NMG [29], NMM [41], and MG/OPT [45], we create level-dependent models $h^l : \mathbb{R}^{n^l} \to \mathbb{R}$, defined as

$$h_c^l(\boldsymbol{x}_c^l) := f_c^l(\boldsymbol{x}_c^l) + \langle \delta\boldsymbol{g}_c^l, \boldsymbol{x}_c^l - \boldsymbol{x}_{c,0}^l \rangle, \tag{4.3}$$

with

$$\delta\boldsymbol{g}_c^l := \begin{cases} \mathbf{R}_{l+1}^l \nabla h_c^{l+1}(\boldsymbol{x}_{c,\mu_1}^{l+1}) - \nabla f_c^l(\boldsymbol{x}_{c,0}^l) & \text{if} \quad l < L, \\ 0 & \text{if} \quad l = L, \end{cases} \tag{4.4}$$

where we use a triplet (cycle, level, iterate) to denote all computational quantities. For example the solution computed as the ith iterate during the cth cycle on level l is denoted by $\boldsymbol{x}_{c,i}^l$. The symbols μ_1 and n^l denote the number of pre-smoothing steps and number of unknowns on a given level, respectively.

On the finest level, we assume that the level-dependent objective function h_c^L coincides with the original objective function, i.e., $h_c^L := f_c^L$. On the coarser levels, the level-dependent functions $\{h_c^l\}_{l=0}^{L-1}$ consist of two terms. The first term, f_c^l, represents the original minimization problem discretized on level l. The second term,

$\langle \delta \boldsymbol{g}_c^l, \boldsymbol{x}^l - \boldsymbol{x}_{c,0}^l \rangle$, contains the coupling term $\delta \boldsymbol{g}_c^l$ that describes the difference between the restricted fine-level gradient and the initial coarse-level gradient. The presence of this term enforces the following relation:

$$\boldsymbol{g}_{c,0}^{l-1} = \nabla h_{c,0}^{l-1}(\boldsymbol{x}_{c,0}^{l-1}) = \mathbf{R}_l^{l-1} \boldsymbol{g}_{c,\mu_1}^l. \tag{4.5}$$

In other words, the initial coarse-level gradient is nothing but the restricted fine-level gradient. We refer to relation (4.5) as first-order consistency, since the first step of the coarse-level minimization process will be done in the direction of the negative restricted fine-level gradient.

4.3. RMTR algorithm. Having defined the coarse level minimization problems, we can proceed to the description of the RMTR method shown in Algorithm 4.2. We use a V-cycle for the RMTR method, and our description follows [28] closely. In addition, all theoretical properties of Algorithm 4.2 shown in [28] hold, when applied to the minimization problem (3.7).

As depicted in Figure 5, each V-cycle consists of nonlinear smoothing steps and a coarse-level solve. While smoothing reduces the high-frequency error associated with each level, the coarse-level solve eliminates the low-frequency error remaining on the coarsest level. Starting from the finest level, $l = L$, the RMTR algorithm performs $\mu_1 \in \mathbb{N}$ pre-smoothing steps. This is done by a trust region solver; cf. Algorithm 4.1.

Algorithm 4.2. RMTR.

```
Require: $l \in \mathbb{N}, \boldsymbol{x}_{c,0}^l, \boldsymbol{g}_{c,\mu_1}^{l+1} \in \mathbb{R}^{n^l}, \Delta_{c,0}^l, \epsilon_g^l \in \mathbb{R}$
Constants: $\mu_1, \mu_2 \in \mathbb{N}$, $\eta_1, \epsilon^\Delta \in \mathbb{R}$, where $0 < \eta_1 < 1$
 1: Compute $\delta \boldsymbol{g}_c^l$ and generate $h_c^l$ by means of (4.3).   ▷ Initialization of given level
 2: $[\boldsymbol{x}_{c,\mu_1}^l,\ \Delta_{c,\mu_1}^l] =$ Local_TR($h_c^l,\ \boldsymbol{x}_{c,0}^l,\ \Delta_{c,0}^l,\ \epsilon_g^l,\ \mu_1$)   ▷ Pre-smoothing
 3: if $l > 0$ then
 4:     $[\boldsymbol{x}_{c,\mu^{l-1}}^{l-1}]$=RMTR($l-1, \mathbf{P}_l^{l-1}\boldsymbol{x}_{c,\mu_1}^l, \mathbf{R}_l^{l-1}\boldsymbol{g}_{c,\mu_1}^l, \Delta_{c,\mu_1}^l, \epsilon_g^{l-1}$)
 5:     $\boldsymbol{s}_{c,\mu_1+1}^l = \mathbf{I}_{l-1}^l(\boldsymbol{x}_{c,\mu^{l-1}}^{l-1} - \boldsymbol{x}_{c,0}^{l-1})$   ▷ Prolongation of coarse-level correction
 6:     Compute $\rho_{c,\mu_1+1}^l$ by means of (4.11).   ▷ Computation of multilevel TR ratio
 7:     if $\rho_{c,\mu_1+1}^l > \eta_1$ then
 8:         $\boldsymbol{x}_{c,\mu_1+1}^l := \boldsymbol{x}_{c,\mu_1}^l + \boldsymbol{s}_{c,\mu_1+1}^l$   ▷ Acceptance of coarse-level correction
 9:     else
10:         $\boldsymbol{x}_{c,\mu_1+1}^l := \boldsymbol{x}_{c,\mu_1}^l$   ▷ Rejection of coarse-level correction
11:     end if
12:     $[\Delta_{c,\mu_1+1}^l] =$ Radius_update($l,\ \rho_{c,\mu_1+1}^l,\ \Delta_{c,\mu_1}^l,\ \Delta^{l+1}$)
13:     if $\|\boldsymbol{g}_{c,\mu_1+1}^l\| \le \epsilon_g^l$ then
14:         return $\boldsymbol{x}_{c,\mu_1+1}^l$   ▷ Termination
15:     end if
16: end if
17: $[\boldsymbol{x}_{c,\mu^l}^l,\ \Delta_{c,\mu^l}^l] =$ Local_TR($\boldsymbol{x}_{c,\mu_1+1}^l,\ \Delta_{c,\mu_1+1}^l,\ \epsilon_g^l,\ \mu_2$)   ▷ Post-smoothing
18: if $l = L$ then
19:     Set $\Delta_{c+1,0}^l = \Delta_{c,\mu^l}^l,\ \boldsymbol{x}_{c+1,0}^l = \boldsymbol{x}_{c,\mu^l}^l,\ c = c+1$   ▷ Initialization of next V-cycle
20:     Go to: 1   ▷ Go to next V-cycle
21: else
22:     return $\boldsymbol{x}_{c,\mu^l}^l$   ▷ Continue recursion and go to next finer level
23: end if
```

The result of pre-smoothing is an intermediate iterate $\boldsymbol{x}_{c,\mu_1}^l$. The projection of this iterate to the next coarser level yields the initial guess on level $l-1$, i.e.,

$$\boldsymbol{x}_{c,0}^{l-1} = \mathbf{P}_l^{l-1} \boldsymbol{x}_{c,\mu_1}^l. \tag{4.6}$$

Once we have reached the coarser level, we can perform the minimization of the level-dependent objective function h_c^{l-1}. The result of this minimization is a coarse-level correction, $\boldsymbol{s}_c^{l-1}$, which is prolongated back to the fine level. For this reason, we have to make sure that the prolongated coarse-level correction does not exceed the current fine-level trust region radius:

$$\|\mathbf{I}_{l-1}^l \boldsymbol{s}_c^{l-1}\| \leq \Delta_{c,\mu_1}^l. \tag{4.7}$$

This can be achieved by defining the trust region subproblems on the lower levels with respect to the current fine-level trust region radius Δ_{c,μ_1}^l as follows:

$$\begin{aligned} &\underset{\boldsymbol{s}_c^{l-1} \in \mathbb{R}^{n^{l-1}}}{\text{minimize}} && h_c^{l-1}(\boldsymbol{x}_{c,0}^{l-1} + \boldsymbol{s}_c^{l-1}) \\ &\text{subject to} && \|\boldsymbol{s}_c^{l-1}\|_{l-1} \leq \Delta_{c,\mu_1}^l, \end{aligned} \tag{4.8}$$

where $\|\cdot\|_l$ is the level-dependent norm defined by

$$\|\boldsymbol{x}_{c,i}^l\|_l := \begin{cases} \|\mathbf{I}_{L-1}^L \cdots \mathbf{I}_l^{l+1} \boldsymbol{x}_{c,i}^l\|_2 & \text{if} \quad l < L, \\ \|\boldsymbol{x}_{c,i}^l\|_2 & \text{if} \quad l = L. \end{cases} \tag{4.9}$$

The constraint in (4.8) guarantees that the prolongated correction stays inside the fine-level trust region radius.

4.3.1. Update of the trust region radius.

We can incorporate the constraint from (4.8) explicitly into the trust region radius update rules. As demonstrated by Algorithm 4.3, the new TR radius, $\Delta_{c,i}^l$, on a given level l and iterate i, is obtained in two steps. First, we compute the intermediate radius Δ_+^l, which follows standard TR update rules. In the second step, we determine a threshold $\delta_{c,i}^l$ on the forthcoming coarse-level correction $\boldsymbol{s}_{c,i+1}^{l-1}$. The actual value of the $\delta_{c,i}^l$ depends on the condition $\|\boldsymbol{x}_{c,i}^l - \boldsymbol{x}_{c,0}^l\|_l > (1-\epsilon^\Delta)\Delta_{c,\mu_1}^{l+1}$; see line 4 of Algorithm 4.3. If the condition is not satisfied, $\delta_{c,i}^l$ is set to zero, since it is impossible to obtain any further coarse-level correction without violating the fine-level trust region radius; see [24, Lemma 4.1]. If the condition holds, $\delta_{c,i}^l$ is computed such that the lower-level iterates remain in the fine-level trust region radius, i.e., such that (4.8) holds. Finally, the new TR radius, $\Delta_{c,i+1}^l$, is defined as $\min(\Delta_+^l, \delta_{c,i}^l)$. Figure 2 illustrates the process. Note that we assume $\Delta^{l+1} = \infty$ for $l = L$.

4.3.2. Acceptance of the coarse-level correction.

After the approximate minimization on level $l-1$ has been performed, we prolongate the coarse-level correction to the finer level, i.e.,

$$\boldsymbol{s}_{c,\mu_1+1}^l = \mathbf{I}_{l-1}^l (\boldsymbol{x}_{c,\mu^{l-1}}^{l-1} - \boldsymbol{x}_{c,0}^{l-1}). \tag{4.10}$$

The symbol μ^{l-1} in (4.10) denotes the number of all computed corrections on level $l-1$ during cycle c. The coarse-level correction is accepted only if it provides a decrease in the fine-level objective function, i.e., only if $h_{c,\mu_1+1}^l < h_{c,\mu_1}^l$. The quality of $\boldsymbol{s}_{c,\mu_1+1}^l$

is determined by means of the multilevel trust region ratio ρ^l_{c,μ_1}, which measures the agreement between the fine-level and the coarse-level reductions:

$$\rho^l_{c,\mu_1+1} = \frac{h^l_c(\boldsymbol{x}^l_{c,\mu_1}) - h^l_c(\boldsymbol{x}^l_{c,\mu_1} + \boldsymbol{s}^l_{c,\mu_1})}{h^{l-1}_c(\mathbf{P}^{l-1}_l \boldsymbol{x}^l_{c,\mu_1}) - h^{l-1}_c(\boldsymbol{x}^{l-1}_{c,\mu^{l-1}})} = \frac{\text{fine-level reduction}}{\text{coarse-level reduction}}. \tag{4.11}$$

The correction $\boldsymbol{s}^l_{c,\mu_1+1}$ is accepted only if $\rho^l_{c,\mu_1+1} > \eta_1$. This is in contrast to the other nonlinear multilevel schemes, such as the full approximation scheme [6], where the coarse-level correction $\boldsymbol{s}^l_{c,\mu_1+1}$ is accepted without verifying its quality.

Algorithm 4.3. Radius_update.

```
Require: $l \in \mathbb{N}, \rho_i,\ \Delta^l_{c,i},\ \Delta^{l+1}_{c,\mu_1} \in \mathbb{R}$
Constants: $\epsilon^\Delta, \eta_1, \eta_2, \gamma_1, \gamma_2 \in \mathbb{R}$, where
           $0 < \eta_1 \leq \eta_2 < 1,\ \epsilon^\Delta \in (0,1)$,
           $0 < \gamma_1 < 1 < \gamma_2$
1:
2: $\Delta^l_+ = \begin{cases} \gamma_1 \Delta^l_{c,i} & \rho_i < \eta_1 \\ \Delta_{l,i} & \rho_i \in [\eta_1, \eta_2] \\ \gamma_2 \Delta^l_{c,i} & \rho_i > \eta_2 \end{cases}$
3: $\delta^l_{c,i} = 0$
4: if $\|\boldsymbol{x}^l_{c,i} - \boldsymbol{x}^l_{c,0}\|_l > (1-\epsilon^\Delta)\Delta^{l+1}_{c,\mu_1}$ then
5:     $\delta^l_{c,i} = \Delta^{l+1}_{c,\mu_1} - \|\boldsymbol{x}^l_{c,i} - \boldsymbol{x}^l_{c,0}\|_l$
6: end if
7: return $\Delta^l_{c,i+1} = \min(\Delta^l_+,\ \delta^l_{c,i})$
```

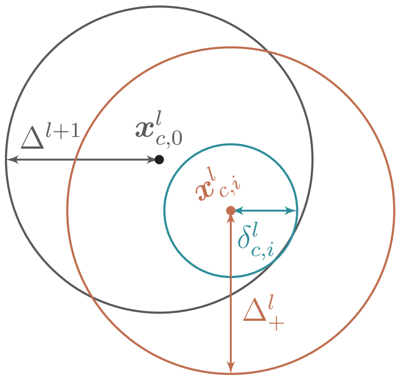


Fig. 2. *Preservation of the fine-level trust region radius.*

Once the correction $\boldsymbol{s}^l_{c,\mu_1+1}$ has been accepted or rejected, the V-cycle continues with $\mu_2 \in \mathbb{N}$ post-smoothing steps. The outlined process is performed recursively on each level, except on the coarsest one, where no recursion is called. On the finest level, the last iterate of cycle c is taken over as the initial iterate of the next cycle $c+1$.

5. Subdivision surfaces.

As noted before, the Kirchhoff–Love formulation requires test and trial functions to be H^2 continuous [37]. Such smoothness can be guaranteed, for example, by employing NURBS [32] or subdivision basis functions [15]. Subdivision schemes generally provide the advantage that they are defined on control meshes with arbitrary topology. For cloth simulations in the animation industry, a subdivision-based discretization provides an additional advantage, since cloth objects are often also modeled and rendered as subdivision surfaces.

Many subdivision schemes exist, but the most widely used are the Catmull–Clark [13] and the Loop [44] schemes. This work employs the Catmull–Clark subdivision scheme as implemented by Pixar's open-source package OpenSubdiv [47]. The Catmull–Clark scheme was developed for quadrilateral meshes as a generalization of bi-cubic B-spline surfaces [13]. The resulting surfaces are C^2 continuous everywhere except at extraordinary vertices where they are only C^1 continuous.

In the following, the subdivision surfaces are used to represent both the geometry and the finite element space $\mathcal{X}^l$ on every level $l = 0, \ldots, L$ in the multilevel hierarchy. Additionally, we employ subdivision-based transfer operators for transferring the data between adjacent levels.

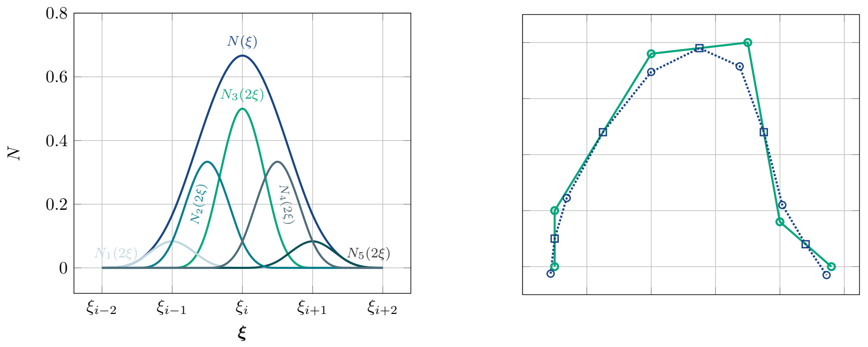


FIG. 3. *Left: Refinement of one-dimensional (1D) cubic B-spline ($N(\xi)$). Right: A subdivision applied to the 1D coarse-level control polygon (green) results in refined control polygon (blue). (Color available online only.)*

5.1. Subdivision-based multilevel hierarchy. The subdivision schemes are developed from the refinability property of the B-spline basis functions [63]. Consider the initial control mesh $\mathcal{T}^0$. We denote coarse-level control points by $\boldsymbol{q}_I^0 = (q_{I1}^0, q_{I2}^0, q_{I3}^0)^T \in \mathbb{R}^3$, where $I = 1, \ldots, n^0$ and n^0 is the number of control points on the coarsest level. Here, we slightly abuse a notation and use the symbol n^l to denote the number of control points on a given level l. For the time being, we assume that there exist B-spline basis functions $\{N_I^0\}$ such that the mid-surface $\boldsymbol{x}$ is represented as

$$\boldsymbol{x}(\boldsymbol{q}, \xi^1, \xi^2) = \sum_{I=1}^{n^0} \boldsymbol{q}_I^0 N_I^0(\xi^1, \xi^2). \tag{5.1}$$

The refinability property states that every B-spline basis function N_I^0 can be expressed as a translated and dilated copy of itself, i.e.,

$$N_I^0(\xi^1, \xi^2) = (\boldsymbol{s}_I^1)^T \boldsymbol{N}_I^0(2\xi^1, 2\xi^2), \tag{5.2}$$

where the vector $\boldsymbol{s}_I^1 \in \mathbb{R}^{n^s}$ defines weights for the linear combination of refined basis functions $\boldsymbol{N}_I^0(2\xi^1, 2\xi^2)$. The size n^s of $\boldsymbol{N}_I^0(2\xi^1, 2\xi^2)$ depends on the order of N_I^0. Figure 3 demonstrates the refinability property of a 1D cubic B-spline, which can be written as a sum of five refined cubic B-splines.

The relation (5.2) allows us to change the basis in (5.1), such that

$$\boldsymbol{x}(\boldsymbol{q}, \xi^1, \xi^2) = \sum_{I=1}^{n^0} \boldsymbol{q}_I^0 N_I^0 = \sum_{I=1}^{n^0} \boldsymbol{q}_I^0((\boldsymbol{s}_I^1)^T \boldsymbol{N}_I^1) = \cdots = \sum_{I=1}^{n^0} \boldsymbol{q}_I^0((\boldsymbol{s}_I^l)^T \boldsymbol{N}_I^l). \tag{5.3}$$

The mid-surface $\boldsymbol{x}$ in (5.3) is defined in terms of the refined basis $\boldsymbol{N}_I^l = \boldsymbol{N}_I^0(2^l\xi^1, 2^l\xi^2)$, which can be expressed by using matrix notation as

$$\boldsymbol{x}(\boldsymbol{q}, \xi^1, \xi^2) = \mathbf{N}^0 \boldsymbol{q}^0 = (\mathbf{S}^1)^T \mathbf{N}^1 \boldsymbol{q}^0 = \cdots = (\mathbf{S}^l)^T \mathbf{N}^l \boldsymbol{q}^0, \tag{5.4}$$

where $(\mathbf{S}^l)^T \in \mathbb{R}^{3n^l \times 3n^{l+1}}$ is the subdivision matrix containing entries from $\boldsymbol{s}_I^l$. Instead of applying subdivision weights $\mathbf{S}^l$ to the basis functions $\mathbf{N}^l$, we can perform an alternative operation in terms of control points

$$\boldsymbol{q}^l = \mathbf{S}^l \boldsymbol{q}^{l-1}. \tag{5.5}$$

Reformulating (5.1) with the help of (5.5) results in

$$\boldsymbol{x}(\boldsymbol{q}, \xi^1, \xi^2) = \mathbf{N}^0 \boldsymbol{q}^0 = \mathbf{N}^1 \boldsymbol{q}^1 = \cdots = \mathbf{N}^l \boldsymbol{q}^l. \tag{5.6}$$

The formulation in (5.6) suggests that the same mid-surface $\boldsymbol{x}$ can be represented by basis functions and control cages of the different refinement levels. This allows us to create the multilevel hierarchy required by the RMTR algorithm in a very natural way as

$$\mathcal{X}^L(\mathcal{T}^L) \supset \mathcal{X}^{L-1}(\mathcal{T}^{L-1}) \supset \cdots \supset \mathcal{X}^0(\mathcal{T}^0), \tag{5.7}$$

where the spaces associated with different subdivision levels are defined by

$$\mathcal{X}^l(\mathcal{T}^l) := \operatorname{span}(N_I^l : I = 1, \ldots, n^l). \tag{5.8}$$

5.2. Subdivision FEM. On each level l of the multilevel hierarchy (5.7), the RMTR method, Algorithm 4.2, requires the computation of the multilevel objective function, its gradient, and its Hessian. Following an isogeometric analysis approach, we approximate the domain geometry and the solution of (3.7) by the same basis functions, $\{N_I^l \boldsymbol{e}_j\}_{j=1,2,3; I=1,\ldots,n^l}$, where $\boldsymbol{e}_j \in \mathbb{R}^3$ denotes standard basis vector. The finite element formulation is then obtained by inserting

$$\boldsymbol{x}(\boldsymbol{q}^l, \xi^1, \xi^2) = \sum_{I=1}^{n^l} \boldsymbol{q}_I^l N_I^l(\xi^1, \xi^2) \quad \text{and} \quad \boldsymbol{v} = \sum_{I=1}^{n^l} N_I^l \boldsymbol{e}_j \tag{5.9}$$

into (3.9). Integration in (3.9) is performed by numerical quadrature, which requires evaluation of the basis functions $\{N_I^l\}_{I=1}^{n^l}$ and their first- and second-order derivatives at any parameter location $(\xi^1, \xi^2) \in \omega$. Since the parametrization of a subdivision surface depends on the neighborhood of a given face F, we split the faces of the control cage into two groups: ordinary and extraordinary. An ordinary face only contains ordinary vertices defined as vertices with valence $\vartheta = 4$ (i.e., four incident edges), while an extraordinary face contains at least one extraordinary vertex, i.e., a vertex with valence $\vartheta \neq 4$.

For ordinary patches as shown in Figure 4a, the Catmull–Clark scheme reduces by design to a uniform bi-cubic B-spline surface. These regions are easy to evaluate since the B-spline basis functions are polynomials. The surface patch corresponding to a given face F depends only on the one-ring neighborhood of the face. By one-ring neighborhood, we mean the set of all faces incident to the given face F. In the case of an ordinary patch, the one-ring is topologically a 4×4 rectangular grid and there are exactly 16 nonzero values of N_I^l.

The extraordinary surface patches, such as the one shown in Figure 4b, can be difficult and expensive to evaluate because the surface is no longer defined by a B-spline basis. However, for any point away from an extraordinary point, the evaluation can be performed by applying a finite number of subdivision steps, such that the given point is no longer in an extraordinary region. Stam [54] demonstrated that the eigenstructure of the local subdivision matrix may be exploited in order to evaluate the subdivision basis functions and their derivatives even at extraordinary vertices.

5.2.1. Multilevel treatment of boundary conditions. The use of subdivision basis functions requires careful treatment of the prescribed boundary conditions

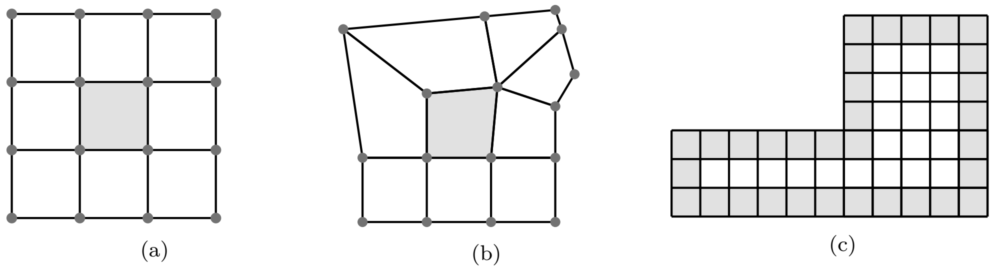


FIG. 4. (a) *Ordinary subdivision finite element (gray). A bi-cubic B-spline is defined by* 16 *control vertices (indicated by dots) in a one-ring neighborhood (white).* (b) *Extraordinary subdivision finite element (gray) and its one-ring neighborhood (white) defined by* 16 *control vertices (indicated by dots).* (c) *Open control mesh with one layer of ghost faces (gray).*

$\boldsymbol{b}(\xi^1, \xi^2)$ for the problem (3.7). This is due to the fact that the discretization of boundary conditions

$$\boldsymbol{b}(\boldsymbol{q}^l, \xi^1, \xi^2) = \sum_{I=1}^{n^l} \boldsymbol{q}_I^l N_I^l(\xi^1, \xi^2) \tag{5.10}$$

employs basis functions $\{N_I^l\}$, which have one-ring support. As a consequence, we have to attach a one-ring of artificial/ghost faces around the boundary; see Figure 4c. It is important to notice that the mid-surface is not parametrized over ghost faces, but the ghost control points affect the solution on the neighboring interior faces.

Our implementation incorporates ghost faces into the design of the multilevel hierarchy as follows. During the modeling process, we add one layer of ghost faces to the coarse-level control cage $\mathcal{T}^0$. In the next step, we subdivide $\mathcal{T}^0$ and, as a result, obtain the control cage $\mathcal{T}^1$, which now contains two layers of ghost faces. Since the outer layer of ghost faces does not influence the geometry along the boundary, it can be discarded. In fact, removing unused vertices/faces saves a significant amount of memory. Figure 5 provides an example, where the initial control cage with ghost faces is subdivided two times. The parametrized surface, shown in blue, retains its domain on each level. The red color represents the one-ring of ghost vertices. On levels $l = 1, 2$, additional layers of ghost vertices (yellow color) are removed.

5.3. Transfer operators via subdivision scheme.

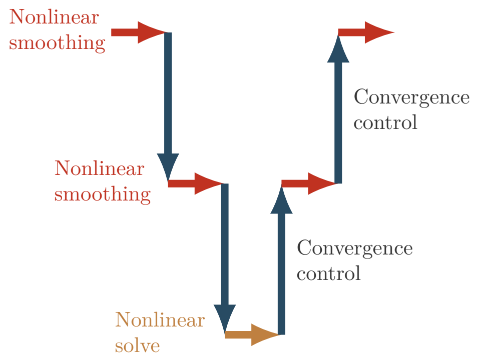
The exchange of data between adjacent levels of the multilevel hierarchy is ensured by transfer operators. As already introduced in section 4.2, the RMTR algorithm employs three different types of transfer operators: prolongation, restriction, and projection. In this work, all three transfer operators are based on subdivision surfaces.

5.3.1. Prolongation and restriction.

A solution of the discretized Euler–Lagrange equation (3.9) is a set of optimal fine-level control points $(\boldsymbol{q}^L)^* \in \mathbb{R}^{3n^L}$. Recalling section 5.1, the relation between the fine-level and the coarse-level control points can be expressed by means of the subdivision matrix $\mathbf{S}_l^{l+1} \in \mathbb{R}^{3n^{l+1} \times 3n^l}$ as

$$\boldsymbol{q}^{l+1} = \mathbf{S}_l^{l+1}\, \boldsymbol{q}^l. \tag{5.11}$$

As a consequence, the global subdivision matrix $\mathbf{S}_l^{l+1}$ is a natural choice for the prolongation of the primal variables (control points).



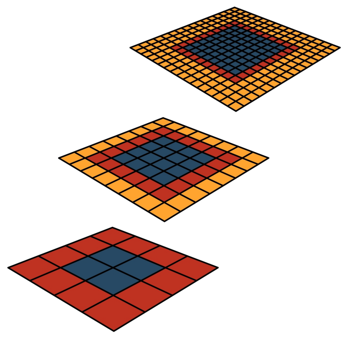


Fig. 5. *Left: Nonlinear V-cycle. Right: Multilevel hierarchy obtained by the Catmull–Clark subdivision scheme. Blue represents the parametrized surface. The ghost vertices associated with a given level are marked by red. The yellow part represents the redundant part of the domain, which is removed. (Color available online only.)*

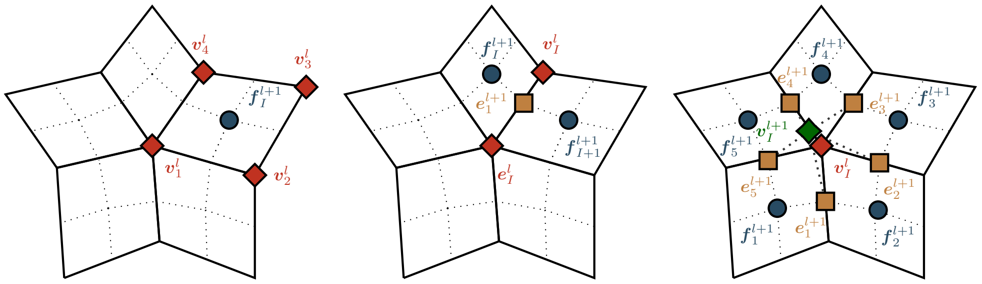


Fig. 6. *Catmull–Clark subdivision scheme for extraordinary control points. Left: Construction of a face point (blue circle). Middle: Computation of an edge point (brown rectangle). Right: Reposition of the coarse-level vertex (green diamond). Red diamond markers represent coarse-level control points. (Color available online only.)*

The assembly of $\mathbf{S}_l^{l+1}$ can be performed using the Catmull–Clark subdivision scheme, which consists of two steps: refinement and smoothing. Refinement introduces new control points, called face points $\{\boldsymbol{f}_I^{l+1}\}$ or edge points $\{\boldsymbol{e}_I^{l+1}\}$. Smoothing repositions the coarse-level control points. The fine-level control points created in this way are denoted as $\{\boldsymbol{v}_I^{l+1}\}$. The fine-level control cage $\mathcal{T}^{l+1}$ contains all three types of control points, collectively denoted by $\boldsymbol{q}^{l+1} \in \mathbb{R}^{3n^{l+1}}$.

In the following, we describe subdivision rules for the Catmull–Clark scheme in the generic form, such that they can be applied to both ordinary and extraordinary control points. Figure 6 illustrates the subdivision steps, (5.12)–(5.15) explained below, for extraordinary control points. The local subdivision rules for the Catmull–Clark scheme are the following:

1. A face point $\boldsymbol{f}_I^{l+1}$ is created by taking an average of all control points surrounding a given element/face F_I, i.e.,

$$\boldsymbol{f}_I^{l+1} = \frac{1}{N} \sum_{\boldsymbol{v}_I^l \in F_I} \boldsymbol{v}_I^l, \tag{5.12}$$

where $N = |F_I|$.

2. A new edge point $\boldsymbol{e}_I^{l+1}$ is computed by averaging the endpoints $(\boldsymbol{e}_I^l, \boldsymbol{v}_I^l)$ of a given edge together with the face points $(\boldsymbol{f}_I^{l+1}, \boldsymbol{f}_{I+1}^{l+1})$ of the two faces incident

to the edge as follows:

$$\boldsymbol{e}_I^{l+1} = \frac{\boldsymbol{e}_I^l + \boldsymbol{v}_I^l + \boldsymbol{f}_I^{l+1} + \boldsymbol{f}_{I+1}^{l+1}}{4}. \tag{5.13}$$

The face points $\boldsymbol{f}_I^{l+1}$ and $\boldsymbol{f}_{I+1}^{l+1}$ are obtained by (5.12).

3. The fine-level position $\boldsymbol{v}_I^{l+1}$ of the coarse-level control point $\boldsymbol{v}_I^l$ is obtained as

$$\boldsymbol{v}_I^{l+1} = \frac{\vartheta - 2}{\vartheta}\boldsymbol{v}_I^l + \frac{1}{\vartheta^2}\sum_{I=0}^{\vartheta-1}\boldsymbol{e}_I^l + \frac{1}{\vartheta^2}\sum_{I=0}^{\vartheta-1}\boldsymbol{f}_I^{l+1}, \tag{5.14}$$

where ϑ is the valence of $\boldsymbol{v}_I^l$. Employing (5.13), we can replace $\boldsymbol{e}_I^l$ in (5.14) with the newly created edge points $\boldsymbol{e}_I^{l+1}$ as

$$\boldsymbol{v}_I^{l+1} = \frac{\vartheta - 3}{\vartheta}\boldsymbol{v}_I^l + \frac{4}{\vartheta^2}\sum_{I=0}^{\vartheta-1}\boldsymbol{e}_I^{l+1} - \frac{1}{\vartheta^2}\sum_{I=0}^{\vartheta-1}\boldsymbol{f}_I^{l+1}. \tag{5.15}$$

The fine-level position $\boldsymbol{v}_I^{l+1}$ of $\boldsymbol{v}_I^l$ is then expressed as the weighted average of $\boldsymbol{v}_I^l$ and the newly created points in its one-ring neighborhood.

In addition to (5.12)–(5.15), subdivision on the boundaries requires a different set of rules; see Appendix A.

5.3.2. Projection by reverse subdivision. The RMTR method in Algorithm 4.2 employs a projection operator in order to transfer primal variables (control points) from a finer level to the next coarser level. This operator is used to generate the initial guess on a coarser level, given a fine-level iterate. Gross and Krause (see [28, 27]) show that the quality of the projection operator impacts the overall convergence behavior of the RMTR method. An ideal projection operator, $\mathbf{P}_{l+1}^l \in \mathbb{R}^{3n^l \times 3n^{l+1}}$, should satisfy the following relation:

$$\boldsymbol{q}^l = \mathbf{P}_{l+1}^l \underbrace{(\mathbf{S}_l^{l+1}\boldsymbol{q}^l)}_{\boldsymbol{q}^{l+1}}. \tag{5.16}$$

In other words, the transfer of the iterate between two levels should not alter the original iterate. The restriction operator, $(\mathbf{S}_l^{l+1})^T$, typically does not satisfy (5.16).

An operator, $\mathbf{P}_{l+1}^l$, that meets requirement (5.16) can be found by solving the following least-square minimization problem:

$$\boldsymbol{q}^l = \arg\min_{\boldsymbol{q}} \|\boldsymbol{q}^{l+1} - \mathbf{S}_l^{l+1}\boldsymbol{q}\|^2. \tag{5.17}$$

This gives

$$\boldsymbol{q}^l = ((\mathbf{S}_l^{l+1})^T\mathbf{S}_l^{l+1})^{-1}(\mathbf{S}_l^{l+1})^T\boldsymbol{q}^{l+1}. \tag{5.18}$$

The unique operator $\mathbf{P}_{l+1}^l$ is thus given by $\mathbf{P}_{l+1}^l = ((\mathbf{S}_l^{l+1})^T\mathbf{S}_l^{l+1})^{-1}(\mathbf{S}_l^{l+1})^T$, i.e., as the Moore–Penrose pseudo-inverse of $\mathbf{S}_l^{l+1}$. Unfortunately, the operator $\mathbf{P}_{l+1}^l$ obtained from (5.17) requires matrix inversion, which makes the algorithm computationally expensive. Moreover, the operator $\mathbf{P}_{l+1}^l$ may be a dense matrix, even though the matrix $\mathbf{S}_l^{l+1}$ is sparse.

Given the goal of designing a computationally efficient multilevel algorithm, we do not employ $\mathbf{P}^l_{l+1}$ but instead introduce a novel and computationally cheaper operator, $\mathbf{Q}^l_{l+1} \in \mathbb{R}^{3n^l \times 3n^{l+1}}$, which is based on reverse subdivision. Following the work presented in [43], the reverse subdivision operator $\mathbf{Q}^l_{l+1}$ is affine and satisfies (5.16). Reverse subdivision reconstructs coarse-level control points $\boldsymbol{v}^l_I$ by using knowledge about the fine-level control points in the one-ring neighborhood of $\boldsymbol{v}^{l+1}_I$. The coarse-level point $\boldsymbol{v}^l_I$ can be expressed in a generic way as

$$\boldsymbol{v}^l_I = c_1 \boldsymbol{v}^{l+1}_I + c_2 \sum_{I=0}^{\vartheta-1} \boldsymbol{e}^{l+1}_I + c_3 \sum_{I=0}^{\vartheta-1} \boldsymbol{f}^{l+1}_I, \tag{5.19}$$

where c_1, c_2, c_3 for the moment are coefficients to be determined, and ϑ denotes the valence of $\boldsymbol{v}^l_I$. We can substitute $\boldsymbol{v}^{l+1}_I$ and $\boldsymbol{e}^{l+1}_I$ in (5.19) by (5.14) and (5.13), respectively. Equation (5.19) can then be reformulated as follows:

$$\boldsymbol{v}^l_I = \left(\frac{\vartheta-2}{\vartheta}c_1 + \frac{\vartheta}{4}c_2\right)\boldsymbol{v}^l_I + \left(\frac{c_1}{\vartheta^2} + \frac{c_2}{4}\right)\sum_{I=0}^{\vartheta-1} \boldsymbol{e}^l_I + \left(\frac{c_1}{\vartheta^2} + \frac{c_2}{2} + c_3\right)\sum_{I=0}^{\vartheta-1} \boldsymbol{f}^{l+1}_I.$$

The coefficients c_1, c_2, c_3 can now be obtained by solving the simple 3×3 system of equations

$$\begin{pmatrix} (\vartheta-2)/\vartheta & \vartheta/4 & 0 \\ 1/\vartheta^2 & 1/4 & 0 \\ 1/\vartheta^2 & 1/2 & 1 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}. \tag{5.20}$$

The solution of (5.20) is

$$c_1 = \frac{\vartheta}{(\vartheta-3)}, \qquad c_2 = \frac{-4}{\vartheta(\vartheta-3)}, \qquad c_3 = \frac{1}{\vartheta(\vartheta-3)}. \tag{5.21}$$

The reverse subdivision formula is finally obtained by inserting (5.21) into (5.19). The formula (5.19) is straightforward to implement. The assembly of the reverse subdivision operator is based only on the information from the local neighborhood of the fine-level control point $\boldsymbol{v}^{l+1}_I$. This leads to a transfer operator with sparse structure, in contrast to the one obtained in (5.18).

The result in (5.19) cannot be applied for extraordinary vertices with valence $\vartheta = 3$ because the determinant of the system in (5.20) equals zero in that case. Luckily, an alternative reverse subdivision rule can be applied; see Appendix C. Additionally, the formula (5.19) cannot be used to reconstruct coarse-level ghost points. This is due to the fact that during the construction of the multilevel hierarchy (see section 5.2.1), we delete one layer of the ghost faces on each level. Therefore, we do not have complete information about the one-ring neighborhood of $\boldsymbol{v}^{l+1}_I$ required by (5.19). We can overcome this difficulty by introducing a new set of reverse subdivision rules; see Appendix B.

Even though the results in this paper are obtained using the Catmull–Clark scheme, the reverse subdivision operator also exists for the other subdivision schemes, e.g., the Loop scheme [52].

6. Implementation.

The nonlinear multilevel scheme requires discretization of the problem (3.9) on different levels of the multilevel hierarchy. This requires several

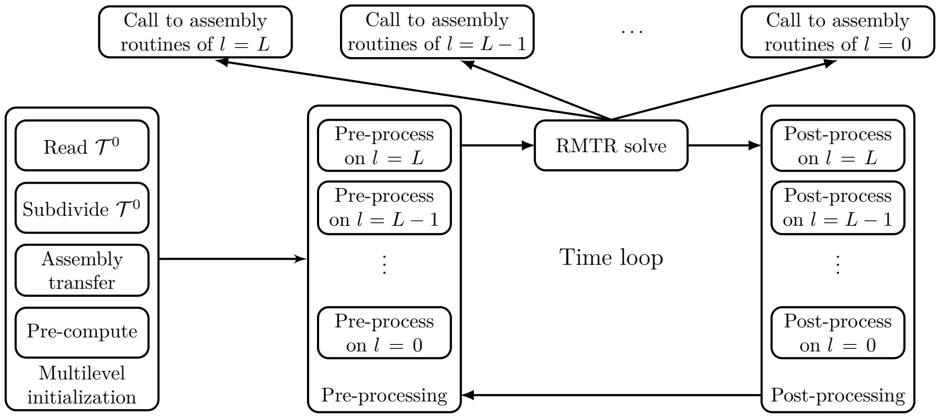

Call to assembly routines of $l = L$
Call to assembly routines of $l = L - 1$
...
Call to assembly routines of $l = 0$
Read $\mathcal{T}^0$
Subdivide $\mathcal{T}^0$
Assembly transfer
Pre-compute
Multilevel initialization
Pre-process on $l = L$
Pre-process on $l = L - 1$
Pre-process on $l = 0$
Pre-processing
RMTR solve
Time loop
Post-process on $l = L$
Post-process on $l = L - 1$
Post-process on $l = 0$
Post-processing

FIG. 7. *Nonlinear multilevel transient framework.*

adjustments to the traditional transient FEM framework. Our proposed design is illustrated in Figure 7.

The simulation starts with the multilevel initialization. During this process, the algorithm generates the hierarchy of control cages $\{\mathcal{T}^l\}_{l=0,\ldots,L}$ by subdividing the initial control cage $\mathcal{T}^0$ and assembling the transfer operators. For the assembly of the global subdivision matrix, we use the OpenSubdiv library [47], while the assembly of the reverse subdivision operator is implemented separately. Note that the transfer operators are only assembled once since the topology of the control cages does not change during the simulation. The initialization phase also contains a pre-computation step, where we compute basis functions and their derivatives at quadrature points. These values only depend on the local topology, so we only do this for one ordinary face and each type of extraordinary face. The resulting stencils of basis function values are stored in a hash-table, which is later used during the solution process for assembly of the gradient and the Hessian on every level. Our current implementation supports extraordinary vertices with valence up to 99.

Once the initialization has been completed, the simulation continues into a time-stepping loop, which consists of pre-processing, nonlinear solve, and post-processing. Pre-processing and post-processing have to be executed on each level of the multilevel hierarchy. This ensures synchronization of different levels in time. For example, at each time step, the animator prescribes a set of boundary conditions for the finest level. Those conditions need to be distributed to the coarser levels. Otherwise, the coarse-level models would not be suitable representations of the fine-level problem. The RMTR method has access to the assembly routines of the coarse-level models, which allows the level-dependent models described in (4.3) to be created.

Our implementation of this framework leverages Intel's Math Kernel Library library [34] heavily for linear algebra.

7. Numerical examples.

In this section, we study the performance of the RMTR method using four scenarios, each using a 1 m $\times$ 1 m square piece of cloth. The four scenarios are subject to different sets of boundary conditions:

- *Trampoline.* All four sides of the boundary are held fixed.
- *Drooping.* Two neighboring sides of the boundary are fixed, while the remaining two sides are free to fall.
- *Re-entrant corner.* A 0.3 m $\times$ 0.3 m square is removed from the corner of

Table 1
Material parameters for the denim material used in all experiments. The thickness τ is given in mm, the density ρ in kg/m^2, and k_i in MPa. All other parameters are dimensionless.

k_1	4.79	d_1	5	μ_{12}	-1482.1	μ_{33}	-185.7	α_{11}	-7.80	α_{32}	-12,802
k_2	4.52	d_2	1	μ_{13}	276.43	μ_{34}	-45.82	α_{12}	10.80	α_{33}	17.91
k_3	9.03	d_3	5	μ_{14}	2,407.67	μ_{35}	146.98	α_{13}	18.95	α_{34}	30.07
k_4	2.44	d_4	3	μ_{15}	3,416.79	μ_{42}	0.79	α_{14}	-12.89	α_{35}	24.68
τ	0.66	ρ	0.4	μ_{32}	0.658	μ_{43}	11.08	α_{15}	1.61	α_{41}	2.95
								α_{21}	1	α_{42}	-2,823.9
								α_{31}	12.71	α_{43}	4.29

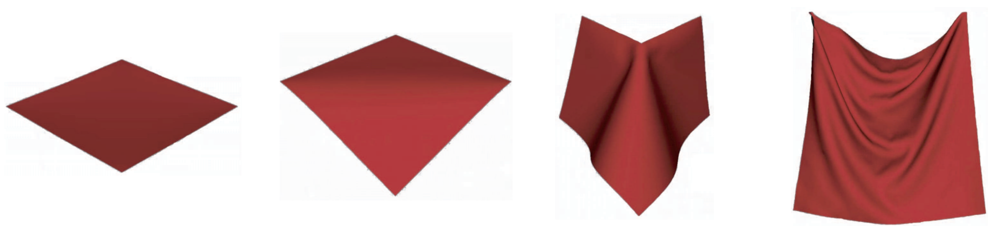


Fig. 8. *Test cases, from left to right: Trampoline, Drooping, Re-entrant corner, Drape.*

the original piece of the cloth, and the edges adjacent to the removed area are held fixed. This example introduces a structural singularity due to the re-entrant corner.

- *Drape.* Time-dependent boundary conditions are applied by moving two opposite corners of the cloth together. Each corner is translated 20 cm in the direction of the opposite corner. This example is of interest because it contains a lot of wrinkles and folding patterns, which cannot be represented on the coarsest level.

All prescribed boundary conditions are applied just on the displacement field, while rotational degrees of freedom are unconstrained. The presented numerical examples employ vertices with valences 2, 3, and 4. Throughout the following we use a denim material given by the parameters in Table 1. Figure 8 depicts the simulation results.

7.1. Condition number and machine precision. As noted in section 3.1.1, the condition number of the linear systems is of order $\mathcal{O}(h^{-4})$, where h represents the mesh size. Figure 9 (left) demonstrates how the condition number increases with respect to the number of degrees of freedom (dofs). As we can see, the condition number is always larger than 10^8.

Figure 9 (right) illustrates the effect of the penalty parameter β on the overall condition number. For this specific example, the overall condition number of the Hessian $\mathbf{H}$ is not affected by the penalty term β for $\beta < 10^5$. For $\beta > 10^5$, the overall condition number increases. This numerical behavior is in agreement with the theory discussed by Pospíšil in [48, Theorem 1.7.2]. In particular, let $\mathbf{M}$ denote the second derivative of the penalty term from (3.7) and let $\mathbf{H}_{\text{mech}}$ denote the second derivative of the mechanical energy (3.2). Pospíšil shows that if $\text{cond}(\mathbf{M}) \leq \text{cond}(\mathbf{H}_{\text{mech}})$, then $\text{cond}(\mathbf{H}) = \text{cond}(\mathbf{H}_{\text{mech}})$.

The presented numerical examples use $\beta = h^{-\sigma}$, where h is the mesh size and $\sigma = 5$. This particular choice provides a reasonable tradeoff between the accuracy with which the boundary conditions are imposed and the ill-conditioning of the arising linear systems.

Level	# dofs	$\approx$ cond($\mathbf{H}$)
1	147	10^{8}
2	363	10^{9}
3	$1,083$	10^{10}
4	$3,675$	10^{11}
5	$13,467$	10^{12}
6	$51,483$	10^{13}

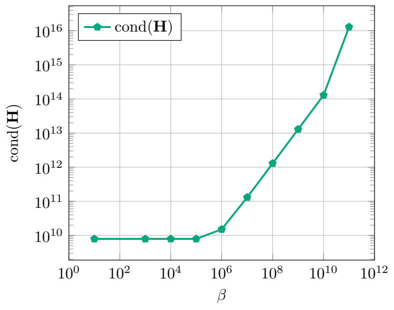


FIG. 9. *Left: Condition number as function of the number of dofs. Right: Condition number as function of penalty parameter β. The data are based on the trampoline example with* 1,083 *dofs and a time step of* $dt = 1$.

7.2. Convergence to critical points and stopping criterion. The RMTR method in Algorithm 4.2 applied to the minimization problem (3.7) converges to the first-order critical points since the energy functional $\widetilde{\psi}^{\mathrm{Mech}}$ is continuously differentiable, is bounded from below, and has uniformly bounded Hessians [24, Theorem 4.13], [28, Theorem 4.7]. In order to identify the convergence numerically, we use the following stopping criterion:

$$\|\boldsymbol{g}(\boldsymbol{x}_i)\| < 10^{-7} \text{ or } \big((|f(\boldsymbol{x}_{i-1}) - f(\boldsymbol{x}_i)| \text{ or } \|\boldsymbol{x}_{i-1} - \boldsymbol{x}_i\|) < 10^{-12} \text{ if } \Delta_i > 10^{15}\big), \tag{7.1}$$

where $\|\boldsymbol{g}(\boldsymbol{x}_i)\|$ is the norm of the gradient at the iterate $\boldsymbol{x}_i$, $|f(\boldsymbol{x}_{i-1}) - f(\boldsymbol{x}_i)|$ measures the difference in the objective function between two successive iterations, and $\|\boldsymbol{x}_{i-1} - \boldsymbol{x}_i\|$ denotes correction size. The second part of (7.1) is activated only if the current trust region radius Δ_i is bigger than 10^{15}. In this case, the current iterate $\boldsymbol{x}_i$ is close to the minimizer $\boldsymbol{x}^*$ and the trust region acts as Newton's method [28, Theorem 4.10]. Unfortunately, the convergence behavior of Newton's method in floating point arithmetic is restricted by the limiting accuracy and the limiting gradient [31, Chapter 25]. The limiting accuracy is proportional to the condition number of the Hessian at the solution $\boldsymbol{x}^*$ and the accuracy with which gradient is evaluated [61, section 2.2]. The limiting gradient provides a lower bound on the norm of gradient [61, section 2.3]. In particular, $\|\boldsymbol{g}(\boldsymbol{x}_i)\|$ is bounded below by the error made in computing the gradient plus the term eps$\|\mathbf{H}(\boldsymbol{x}_i)\|\|\boldsymbol{x}_i\|$, where eps denotes machine precision.

The cloth simulations considered in this work are severely ill conditioned; see section 7.1. In addition, the orthotropic elastic material model (3.3) describing the cloth behavior is defined as a sum of several polynomials. Hence, the evaluation of the gradient (3.9) is subjected to large rounding errors. For this reason, it is of major importance to prevent the algorithm from meandering aimlessly, which we achieve by including the second condition in (7.1). In practice, $\|\boldsymbol{g}(\boldsymbol{x})\| \leq 5 \times 10^{-5}$ is satisfied by all presented numerical examples.

7.3. Choice of constrained QP solver. Both the TR and the RMTR methods require the solution of a constrained quadratic (QP) subproblem (4.1) within each iteration. A suitable QP solver must produce a correction that satisfies the sufficient descent condition (SDC); see [19]. In the context of the cloth simulations considered here, we have tested the Dogleg method [49, 50] and the Steihaug–Toint truncated conjugate-gradient (ST-CG) method [55, 62]. The ST-CG method suffers more from the severe ill-conditioning of the Hessian than the Dogleg method. As a consequence,

TABLE 2
Choice of parameters used inside of TR/RMTR algorithms.

Parameter	η_1	η_2	γ_1	γ_2	$\Delta_{0,0}^L$	μ_1	μ_2	ϵ^Δ
Value	0.1	0.75	0.5	2.0	1	1	1	0.001

the TR method configured with ST-CG reports failure even for examples with $1,083$ dofs. The failure is caused by exceeding $500,000$ iterations without satisfying the stopping criterion (7.1). For this reason, the numerical results presented here for the single level TR are obtained by using the Dogleg method.

Our choice of the constrained QP solver on each level of the multilevel hierarchy follows concepts well known from multigrid. First, we use a fact that the low-frequency components of the error appear more oscillatory on a coarser grid. Some iterative solvers can eliminate those high-frequency components of the error quickly, while leaving low frequencies essentially unchanged [51]. We employ 10 iterations of the ST-CG method on all levels except on the coarsest, since it is known to reduce the components of the gradient associated with large eigenvalues first [21]. In addition, the ill-conditioning of the Hessian is not reflected in the conjugate-gradient iterations until the gradients associated with the large eigenvalues have been made small [21]. We also utilize the fact that the computational cost on the coarsest level is significantly lower than on the finer levels. Our RMTR setup therefore employs the Dogleg method with the sparse direct solver PARDISO [53] on the coarsest grid. In this way, we obtain the exact solution of (4.1), once the trust region radius Δ_i is sufficiently large. This assumption is usually fulfilled close to the minimizer; see [28, Theorem 4.10]. Hence, the nonlinear multilevel method mimics the behavior of linear multigrid close to the minimizer.

7.4. Numerical results. During all simulations, the RMTR solver was configured to use a V-cycle with one pre- and one post-smoothing step and the parameters shown in Table 2.

7.4.1. Robustness with respect to time-stepping. Our first benchmark investigates the robustness of the RMTR method with respect to large time steps. We consider examples with $1,083$ dofs and monitor the largest possible time step, dt, that can be used during simulation, such that the solver successfully converges to the desired tolerance. We do not test time steps larger than $dt = 1$ sec, since at least one result is usually required per frame in an animation (with 1 frame $= 1/24$ sec).

We compare the robustness of the RMTR algorithm to the following solvers: single-level trust region method (Algorithm 4.1), backtracking line-search (LS) Newton's method [46, Chapter 3], damped Newton's method, and the "linearized equation." The last two solvers are included because they are commonly used in the graphics community for cloth simulations. The damped Newton's method is Newton's method without any convergence control and with constant step size $\alpha = 0.3$. The "linearized equation" [4] denotes a solution strategy, which exploits the fact that if the time step dt is very small, then the nonlinearity in (3.9) becomes weaker (almost linear). Therefore, the solution of (3.9) can be found by simply solving a linear system of equations.

Figure 10 illustrates the results. We observe that the trust region–based algorithms are very robust with respect to the large time steps. For all test scenarios, the solvers report convergence independently of the size of dt. In contrast, backtracking

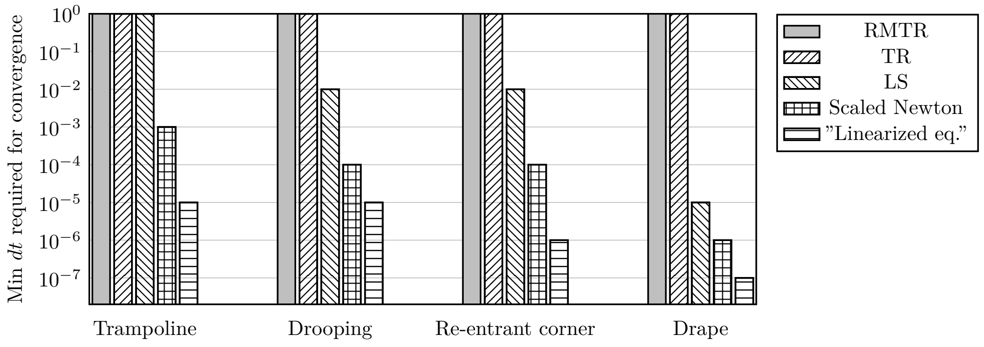


FIG. 10. *Robustness of solution strategies with respect to different time steps. The benchmark was run with* 1,083 *dofs for each example.*

TABLE 3
Number of nonlinear iterations for the single-level TR method as a function of the number of dofs. The time step is $dt = 1$.

# dofs	147	363	1,083	3,675	13,467	51,483
Trampoline	1,385	29,426	34,456	—	—	—
Drooping	1,976	43,475	15,444	12,792	—	—
Re-entrant corner	8,293	13,965	18,873	47,536	334,463	—
Drape	7,331	53,654	175,132	—	—	—

line-search, scaled Newton's, and "linearized equation" require smaller time steps to maintain convergence. We also notice that with increased complexity of the example, the performance of those solvers decreases. For instance, the difference in the number of required time steps between trust region–based methods and the "linearized equation" approach is about 10 million for the drape example.

7.4.2. Number of nonlinear iterations per time step. Next, we compare the convergence behavior of the RMTR method to its single-level counterpart, the TR method. Our analysis focuses on the ability of the methods to solve the cloth simulations as well as on their convergence behavior with respect to number of dofs. In addition, we compare the convergence speed and the computational complexity of both methods. During these tests, we set the time step to $dt = 1$.

Solvability. We analyze the performance of the trust region–based solvers by measuring the number of required nonlinear iterations/V-cycles. Table 3 summarizes the results obtained by using the single-level TR. As we can see, the number of iterations increases rapidly with the increased number of dofs. This is not surprising, as the condition number of the linear systems arising on each iteration grows rapidly. For example, the condition number for the test cases with approximately 50,000 dofs is on the order of 10^{13}. This is very close to one over the machine precision, and the trust region solver reports failure because the desired stopping criterion (7.1) is never met. We also note that solvability becomes an issue already for problems with fewer than 4,000 dofs. In contrast, the RMTR method behaves as a preconditioner, and it is able to solve all our numerical examples, as shown in Table 4.

Convergence speed of RMTR. Table 4 demonstrates that the number of V-cycles decreases with increased number of levels. Intuitively, this behavior can be explained by the fact that the setup of our RMTR method uses approximate constrained qua-

Table 4
Number of nonlinear V-cycles for the RMTR method as a function of the number of dofs. The time step is $dt = 1$.

Example/levels	2	3	4	5	6
Trampoline	633	150	81	88	52
Drooping	4,671	645	707	1,636	333
Re-entrant corner	3,188	1,508	282	124	30
Drape	9,732	7,653	5,718	792	423

dratic solvers on each level of the multilevel hierarchy. If more levels are used, the coarse-level models are better able also to capture the low-frequency parts of the error. This is caused by the fact that a trust region method is also employed on the coarsest level. Depending on the trust region radius, this might act more as a gradient method—or smoother—than as a Newton step—or full coarse grid correction. Consequently, adding more levels improves convergence speed, because it ensures that a bigger part of the spectrum is covered.

The fact that the number of V-cycles required decreases as the number of levels is increased is consistent with the theoretical results obtained in [24, Theorem 4.10]. The upper bound on the total number of successful iterations π produced by the RMTR method is defined as

$$\pi = \sum_{l=0}^{L} \pi^l \leq \frac{h^L(\boldsymbol{x}_{0,0}^L) - h^*}{\theta(\epsilon_g)}, \tag{7.2}$$

where the $h(\boldsymbol{x}_{0,0}^L)$ is the value of the objective function at the initial iterate. The symbol h^* denotes a constant, such that $h^* \leq h^L(x_{i,c}^L)$, for every $x_{i,c}^L \in \mathbb{R}^{3n^L}$. The constant $\theta(\epsilon_g)$ depends on the properties of the minimization problem, on the constants chosen inside of Algorithm 4.2, and on the desirable stopping tolerance ϵ_g. Interestingly, the definition of the constant $\theta(\epsilon_g)$ is dimension-independent, which makes the complexity bound (7.2) mesh-independent [24, Theorem 4.10]. While the upper bound in (7.2) is constant, the total number of successful iterations π is defined as a sum of all successful iterations over all levels. Consequently, the successful coarse-level iterates help to decrease the number of expensive, fine-level iterates. For instance, for a single-level method, i.e., $L = 0$, π represents the successful iterates produced only on the fine level. However, for a three-grid method, i.e., $L = 2$, π contains the sum of the successful coarse-level and fine-level iterates. Note that an iteration is successful in the multilevel context if it is also accepted on the fine level; see section 4.3.2.

Although we expect fewer V-cycles as the number of levels increases, we note that for the Drooping example, the number of V-cycles increases for four and five levels. This does not mean that the bound (7.2) is violated, since Table 4 reports the total number of V-cycles rather than π. The total number of V-cycles might increase, for instance, if the prolongated coarse-level corrections are not accepted by the fine level; see section 4.3.2. Alternatively, the trust region radius might be too small, which causes termination of the recursion before reaching the coarsest level; see section 4.3.1. Despite the fact that the use of the globalization strategy might seem to slow down convergence, its use is crucial as it provides convergence guarantees.

Computational cost: TR versus RMTR. The performance of the TR method deteriorates with increasing number of dofs, while the performance of the RMTR

TABLE 5
Number of nonlinear iterations required by TR versus the number of nonlinear V-cycles required by the RMTR method. The simulation time step is $dt = 1$.

Method	TR	RMTR	TR	RMTR	TR	RMTR
Example/ # dofs	1,083		3,675		51,483	
Trampoline	34,456	150	—	81	—	52
Drooping	15,444	645	12,792	707	—	333
Re-entrant corner	18,873	1,508	47,536	282	—	30
Drape	175,132	7,653	—	5,718	—	423

method improves. Table 5 provides a side-by-side comparison of the results. In order to compare the computational cost of TR and RMTR, we focus on the computationally most expensive parts of both algorithms: assembly of the derivatives (gradient and Hessian) and computation of the (approximate) solution of the QP subproblem (4.1).

First, we estimate the computational cost required by the assembly routines. Due to the sparsity of the Hessian matrices, we assume that the cost on each level, W^l, is proportional to the number of control points, i.e., $W^l = Cn^l$, where C is a constant. Let one *work unit*, W^L, be defined as the computational cost of matrix assembly on the finest level. Assuming a coarsening factor of 2 between each level, the corresponding cost on coarser levels is then given as $W^l = 2^{-2(L-l)}W^L$. From this, the total assembly cost W measured in work units can be computed as

$$W \approx \sum_{l=0}^{L} Q^l W^l = \sum_{l=0}^{L} 2^{-2(L-l)} Q^l \; W^L, \tag{7.3}$$

where Q^l denotes the number of assembly calls on the level l. Let one iteration in the RMTR solver correspond to one V-cycle. The average number of work units per iteration is then simply

$$W_{\text{avg}} = \frac{W}{\#\text{iterations}}. \tag{7.4}$$

Table 6 shows the comparison of W and W_{avg} for the TR and RMTR methods based on our examples with 1,083 dofs. This resolution has been chosen since it is the finest resolution for which the single-level TR method converges. We notice that the TR method requires less than one work unit per iteration, which is due to the fact that the derivatives are recomputed only after a successful iteration, e.g., when the trial point is accepted. Given our RMTR setup with one pre-/post-smoothing step, the assembly is performed every time we enter the given level. However, the number of assembly calls on each level of RMTR can vary, as the recursion may be terminated before reaching a given level; see section 4.3.1. In practice, we see that the RMTR method requires roughly 2.5 work units per V-cycle, and based on (7.3) we compute that the cost of one V-cycle of the RMTR method is approximately 2.8 times higher than the cost of one TR iteration. However, in terms of total computational cost, the RMTR method is significantly more efficient than the single-level TR (from 4.4 to 83.7 times). This is not surprising since the RMTR requires a considerably lower number of V-cycles than the single-level TR iterations to achieve convergence.

Next, we estimate the computational cost of the QP solvers. We follow the same reasoning as for the assembly cost and employ formulas (7.3) and (7.4). The quantities W^l and Q^l now represent the cost of performing one QP solve and the number of calls

Table 6

The total and average computational cost measured in work units required by the assembly routines for the TR and RMTR methods. These numbers represent W and W_{avg}, respectively. The experiment is performed for the problem with 1,083 *dofs. The RMTR method was set up with three levels.*

	Total cost			Average cost	
Example/method	TR	RMTR	TR/RMTR	TR	RMTR
Trampoline	30,981	370	83.73	0.89	2.47
Drooping	13,858	1,612	8.59	0.90	2.50
Re-entrant corner	16,632	3,788	4.39	0.88	2.51
Drape	120,089	19,163	6.27	0.69	2.50

Table 7

The computational cost measured in work units for the QP solvers for the TR and RMTR methods. These numbers represent W and W_{avg}, respectively. The experiment is performed for the problems with 1,083 *dofs. The RMTR method was set up with three levels.*

	Total cost			Average cost	
Example/method	TR	RMTR	TR/RMTR	TR	RMTR
Trampoline	34,456	370	93.12	1	2.47
Drooping	15,444	1,612	9.58	1	2.50
Re-entrant corner	18,873	3,788	4.98	1	2.51
Drape	175,132	19,163	9.14	1	2.50

into the QP solver on a given level l, respectively. We note that the assumption that the cost is proportional to the size of the problem is not valid for a generic QP solver. However, our setup of the RMTR method employs a constant number of 10 iterations of the Steihaug–Toint conjugate-gradient method (ST-CG) per smoothing step, each with cost $O(n^l)$. On the coarsest level, $l = 0$, we employ the Dogleg method with a sparse direct linear solver. Although the complexity of the sparse direct solver is in general higher than $O(n^0)$, the difference can be hidden in the constant inside of the complexity bound, as n^0 is very small. In particular, all presented examples employ a problem with 147 dofs on the coarsest grid. Additionally, the contribution from the coarsest grid is scaled by the coarsening factor 2^{-2L}, which makes the coarse grid cost almost negligible.

Table 7 compares the computational cost needed by the QP solvers for the RMTR and the TR method. We see that the TR method requires one work unit per iteration because the QP solver is used once per iteration. In contrast, the RMTR method requires around 2.5 work units per V-cycle. In spite of that, the RMTR method is significantly more efficient in terms of the total computational cost (from 5 to 93 times). Additionally, we note that this comparison strongly favors the single-level TR method since TR employs the Dogleg method, which is computationally more expensive than 10 steps of ST-CG. We could make the comparison fairer by replacing W^L in (7.3) with the complexity bound of the direct solver. However, this would only enhance the efficiency of the RMTR method compared to the single-level TR method.

7.4.3. Reverse subdivision operator versus least square fit. Finally, we evaluate the quality of the reverse subdivision operator (5.19). Our study compares the performance of the RMTR method with the reverse subdivision operator versus the RMTR method setup with the projection operator obtained by (5.18). Table 8 shows the results in terms of number of V-cycles. We see that the difference between

TABLE 8
Number of V-cycles required by the RMTR method to converge for examples with $3,675$ dofs. The RMTR method was configured with four levels.

Example/operator	Least square fit	Reverse subdivision
Trampoline	84	81
Drooping	659	707
Reentrant corner	277	282
Drape	5,691	5,718

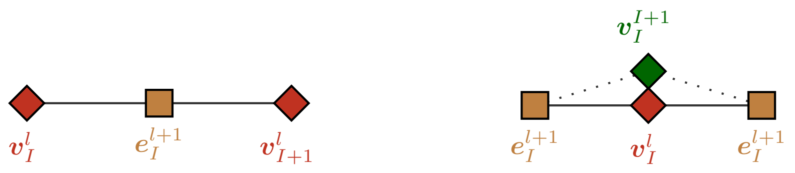


FIG. 11. *Catmull–Clark boundary subdivision. Left: Edge midpoint construction (brown square). Right: Reposition of coarse level point v_I^l into v_I^{l+1} (green diamond). (Color available online only.)*

the number of required nonlinear V-cycles is negligible. However, the reverse subdivision operator offers significant benefits in terms of computational complexity and memory requirements.

8. Conclusion.

In this paper, we have applied the globally convergent recursive multilevel trust region method [24, 28] to simulation of thin shells and cloth. The presented multilevel framework is based on subdivision surfaces, which are also used for the construction of prolongation and restriction operators. Additionally, we have incorporated a reverse subdivision operator for transferring the iterates from the fine levels to coarser levels. The novel use of this operator provides a computationally efficient alternative to the least-square-projection operator used elsewhere. Our numerical examples demonstrate the robustness of the RMTR method with respect to large time steps. A comparison with a single-level TR method has been made, and this has shown a reduction in the number of iterations by several orders of magnitude. In addition, we have shown that for problems with more than $\approx 15,000$ dofs, the single-level trust region method does not converge, while the RMTR method does.

As future work, we intend to investigate the influence of the choice of QP solver on the performance of the RMTR algorithm. We also hope to fully parallelize the code to leverage both vectorization and MPI to enable experimentation with large-scale examples. Finally, our goal is to include contact/collision handling in the overall algorithm, as this is essential for cloth simulations.

Appendix A. Catmull–Clark subdivision: Boundary rules.

We consider two types of boundary subdivision rules; see Figure 11. The first type computes the midpoint $\boldsymbol{e}_I^{l+1}$ of a given edge, i.e.,

$$\boldsymbol{e}_I^{l+1} = 0.5(\boldsymbol{v}_I^l + \boldsymbol{v}_{I+1}^l). \tag{A.1}$$

The second type repositions the coarse-level control point $\boldsymbol{v}_I^l$ into $\boldsymbol{v}_I^{l+1}$ as

$$\boldsymbol{v}_I^{l+1} = 0.125\boldsymbol{e}_I^{l+1} + 0.75\boldsymbol{v}_I^l + 0.125\boldsymbol{e}_{I+1}^{l+1}. \tag{A.2}$$

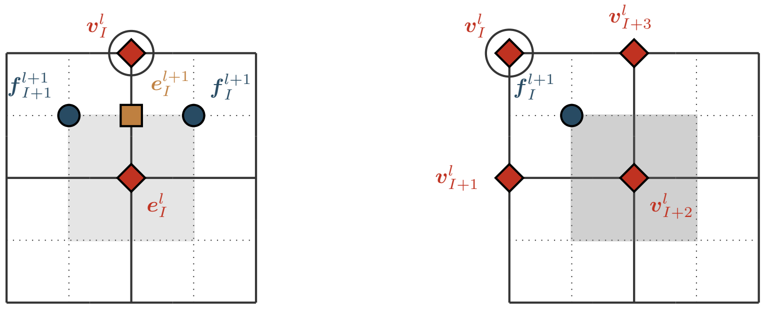


FIG. 12. *Reverse subdivision rules. Left: Reconstruction of regular ghost point (in the circle). Right: Reconstruction of corner ghost point (in the circle).*

Appendix B. Catmull–Clark reverse subdivision: Reconstruction of coarse-level ghost control points. We assume that the coarse-level inner regular and extraordinary control points have already been constructed. The reverse subdivision scheme for a regular ghost control point reformulates the edge point equation

$$\boldsymbol{e}_I^{l+1} = (\boldsymbol{e}_I^l + \boldsymbol{v}_I^l + \boldsymbol{f}_j^{l+1} + \boldsymbol{f}_{I+1}^{l+1})/4 \tag{B.1}$$

as

$$\boldsymbol{v}_I^l = 4\boldsymbol{e}_I^{l+1} - \boldsymbol{e}_I^l - \boldsymbol{f}_I^{l+1} - \boldsymbol{f}_{I+1}^{l+1}. \tag{B.2}$$

Equation (B.2) states that the $\boldsymbol{v}_I^l$ can be obtained as a weighted sum of the fine-level control points $\boldsymbol{f}_I^{l+1}$, $\boldsymbol{f}_{I+1}^{l+1}$, $\boldsymbol{e}_I^{l+1}$ together with the already reconstructed coarse-level inner control point $\boldsymbol{e}_I^l$.

After the regular ghost points have been reconstructed, the corner ghost control point $\boldsymbol{v}_I^l$ can be found by adjusting the face point equation

$$\boldsymbol{f}_I^{l+1} = \frac{1}{N}\left(\boldsymbol{v}_I^l + \sum_{i=1}^{N-1} \boldsymbol{v}_{I+i}^l\right) \tag{B.3}$$

for the given face F_I, where $N = |F_I|$. This leads to the following reverse subdivision rule:

$$\boldsymbol{v}_I^l = N\boldsymbol{f}_I^{l+1} - \sum_{i=1}^{N-1} \boldsymbol{v}_{I+i}^l. \tag{B.4}$$

Figure 12 demonstrates the described steps.

Appendix C. Catmull–Clark reverse subdivision: Extraordinary control points with valence 3. Assume that the control cage has just one extraordinary control point per face. If necessary, this can be achieved by performing one subdivision step in order to obtain the control cage $\mathcal{T}^0$. The reverse subdivision scheme for an extraordinary control point with $\vartheta = 3$ then follows from (B.2) in Appendix B.

REFERENCES

[1] J. C. Alexander and J. A. Yorke, *The homotopy continuation method: Numerically implementable topological procedures*, Trans. Amer. Math. Soc., 242 (1978), pp. 271–284, https://www.jstor.org/stable/1997737.

[2] I. Babuška, *The finite element method with penalty*, Math. Comp., 27 (1973), pp. 221–228, https://doi.org/10.1090/S0025-5718-1973-0351118-5.

[3] K. Bandara and F. Cirak, *Isogeometric shape optimisation of shell structures using multiresolution subdivision surfaces*, Comput.-Aided Des., 95 (2018), pp. 62–71, https://doi.org/10.1016/j.cad.2017.09.006.

[4] D. Baraff and A. Witkin, *Large steps in cloth simulation*, in Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '98, New York, 1998, ACM, pp. 43–54, https://doi.org/10.1145/280814.280821.

[5] J. W. Barrett and C. M. Elliott, *Finite element approximation of the Dirichlet problem using the boundary penalty method*, Numer. Math., 49 (1986), pp. 343–366, https://doi.org/10.1007/BF01389536.

[6] A. Brandt, *Multi-level adaptive solutions to boundary-value problems*, Math. Comp., 31 (1977), pp. 333–390, https://doi.org/10.2307/2006422.

[7] R. Bridson, R. Fedkiw, and J. Anderson, *Robust treatment of collisions, contact and friction for cloth animation*, ACM Trans. Graph., 21 (2002), pp. 594–603, https://doi.org/10.1145/566654.566623.

[8] R. Bridson, S. Marino, and R. Fedkiw, *Simulation of clothing with folds and wrinkles*, in SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, Eurographics Association, 2003, pp. 28–36.

[9] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*, 2nd ed., SIAM, Philadelphia, 2000, https://doi.org/10.1137/1.9780898719505.

[10] X.-C. Cai and D. E. Keyes, *Nonlinearly preconditioned inexact Newton algorithms*, SIAM J. Sci. Comput., 24 (2002), pp. 183–200, https://doi.org/10.1137/S106482750037620X.

[11] X.-C. Cai and X. Li, *Inexact Newton methods with restricted additive Schwarz based nonlinear elimination for problems with high local nonlinearity*, SIAM J. Sci. Comput., 33 (2011), pp. 746–762, https://doi.org/10.1137/080736272.

[12] C. Cartis, N. I. M. Gould, and P. L. Toint, *An adaptive cubic regularization algorithm for nonconvex optimization with convex constraints and its function-evaluation complexity*, IMA J. Numer. Anal., 32 (2012), pp. 1662–1695, https://doi.org/10.1093/imanum/drr035.

[13] E. Catmull and J. Clark, *Recursively generated B-spline surfaces on arbitrary topological meshes*, Comput.-Aided Des., 10 (1978), pp. 350–355, https://doi.org/10.1016/0010-4485(78)90110-0.

[14] F. Cirak and M. Ortiz, *Fully C^1-conforming subdivision elements for finite deformation thin-shell analysis*, Internat. J. Numer. Methods Engrg., 51 (2001), pp. 813–833, https://doi.org/10.1002/nme.182.

[15] F. Cirak, M. Ortiz, and P. Schröder, *Subdivision surfaces: A new paradigm for thin-shell finite-element analysis*, Internat. J. Numer. Methods Engrg., 47 (2000), pp. 2039–2072, https://doi.org/10.1002/(SICI)1097-0207(20000430)47:12<2039::AID-NME872>3.0.CO;2-1.

[16] D. Clyde, *Numerical Subdivision Surfaces for Simulation and Data Driven Modeling of Woven Cloth*, Ph.D. thesis, University of California, Los Angeles, 2017; available online at https://escholarship.org/uc/item/30g0h9r5.

[17] D. Clyde, J. Teran, and R. Tamstorf, *Modeling and data-driven parameter estimation for woven fabrics*, in Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '17, ACM, New York, 2017, 17, https://doi.org/10.1145/3099564.3099577.

[18] D. Clyde, J. Teran, and R. Tamstorf, *Simulation of nonlinear Kirchhoff-Love thin shells using subdivision finite elements*, in Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '17, ACM, New York, 2017, supplemental technical document, https://doi.org/10.1145/3099564.3099577.

[19] A. R. Conn, N. I. M. Gould, and P. L. Toint, *Trust Region Methods*, MOS-SIAM Ser. Optim., SIAM, Philadelphia, 2000, https://doi.org/10.1137/1.9780898719857.

[20] J. E. Dennis, Jr., and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, SIAM, Philadelphia, 1996, https://doi.org/10.1137/1.9781611971200.

[21] A. Forsgren, *On the Behavior of the Conjugate-Gradient Method on Ill-Conditioned Problems*, Technical Report TRITA-MAT-2006-OS1, Department of Mathematics, Royal Institute of Technology, Stockholm, Sweden, 2006.

[22] M. W. Gee and R. S. Tuminaro, *Nonlinear Algebraic Multigrid for Constrained Solid Mechanics Problems Using Trilinos*, Technical Report SAND2006-2256, Sandia National Laboratories, Livermore, CA, 2006.

[23] S. Gratton, M. Mouffe, A. Sartenaer, P. L. Toint, and D. Tomanos, *Numerical experience with a recursive trust-region method for multilevel nonlinear bound-constrained optimization*, Optim. Methods Softw., 25 (2010), pp. 359–386, https://doi.org/10.1080/10556780903239295.

[24] S. Gratton, A. Sartenaer, and P. L. Toint, *Recursive trust-region methods for multiscale nonlinear optimization*, SIAM J. Optim., 19 (2008), pp. 414–444, https://doi.org/10.1137/050623012.

[25] S. Green and G. Turkiyyah, *Second-order accurate constraint formulation for subdivision finite element simulation of thin shells*, Internat. J. Numer. Methods Engrg., 61 (2004), pp. 380–405, https://doi.org/10.1002/nme.1070.

[26] S. Green, G. Turkiyyah, and D. Storti, *Subdivision-based multilevel methods for large scale engineering simulation of thin shells*, in Proceedings of the Seventh ACM Symposium on Solid Modeling and Applications, SMA '02, K. Lee and N. M. Patrikalakis, eds., ACM, New York, 2002, pp. 265–272, https://doi.org/10.1145/566282.566321.

[27] C. Gross, *A Unifying Theory for Nonlinear Additively and Multiplicatively Preconditioned Globalization Strategies: Convergence Results and Examples from the Field of Nonlinear Elastostatics and Elastodynamics*, Ph.D. thesis, Universität Bonn, Bonn, Germany, 2009, http://hss.ulb.uni-bonn.de/2009/1868/1868.htm.

[28] C. Gross and R. Krause, *On the convergence of recursive trust-region methods for multiscale nonlinear optimization and applications to nonlinear mechanics*, SIAM J. Numer. Anal., 47 (2009), pp. 3044–3069, https://doi.org/10.1137/08071819X.

[29] W. Hackbusch, *Multi-Grid Methods and Applications*, Springer Ser. Comput. Math. 4, Springer-Verlag, Berlin, Heidelberg, 1985, https://doi.org/10.1007/978-3-662-02427-0.

[30] W. Hackbusch and A. Reusken, *On global multigrid convergence for nonlinear problems*, in Robust Multi-Grid Methods, Notes Numer. Fluid Mech. 23, Friedr. Vieweg, Braunschweig, 1989, pp. 105–113, https://doi.org/10.1007/978-3-322-86200-6_9.

[31] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed., SIAM, Philadelphia, 2002, https://doi.org/10.1137/1.9780898718027.

[32] T. Hughes, J. Cottrell, and Y. Bazilevs, *Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement*, Comput. Methods Appl. Mech. Engrg., 194 (2005), pp. 4135–4195, https://doi.org/10.1016/j.cma.2004.10.008.

[33] F.-N. Hwang and X.-C. Cai, *A class of parallel two-level nonlinear Schwarz preconditioned inexact Newton algorithms*, Comput. Methods Appl. Mech. Engrg., 196 (2007), pp. 1603–1611, https://doi.org/10.1016/j.cma.2006.03.019.

[34] Intel Developer Zone, *Intel Math Kernel Library*, 2007, https://software.intel.com/en-us/mkl.

[35] M. Juntunen and R. Stenberg, *Nitsche's method for general boundary conditions*, Math. Comp., 78 (2009), pp. 1353–1374, https://doi.org/10.1090/S0025-5718-08-02183-2.

[36] M. Keckeisen and W. Blochinger, *Parallel implicit integration for cloth animations on distributed memory architectures*, in Eurographics Workshop on Parallel Graphics and Visualization, D. Bartz, B. Raffin, and H.-W. Shen, eds., The Eurographics Association, 2004, https://doi.org/10.2312/EGPGV/EGPGV04/119-126.

[37] J. Kiendl, K.-U. Bletzinger, J. Linhard, and R. Wüchner, *Isogeometric shell analysis with Kirchhoff-Love elements*, Comput. Methods Appl. Mech. Engrg., 198 (2009), pp. 3902–3914, https://doi.org/10.1016/j.cma.2009.08.013.

[38] G. Kirchhoff, *Über das Gleichgewicht und die Bewegung einer elastischen Scheibe*, J. Reine Angew. Math., 40 (1850), pp. 51–88, https://doi.org/10.1515/crll.1850.40.51.

[39] A. Klawonn, O. Rheinbach, and O. B. Widlund, *An analysis of a FETI–DP algorithm on irregular subdomains in the plane*, SIAM J. Numer. Anal., 46 (2008), pp. 2484–2504, https://doi.org/10.1137/070688675.

[40] R. Kornhuber and R. Krause, *Adaptive multigrid methods for Signorini's problem in linear elasticity*, Comput. Vis. Sci., 4 (2001), pp. 9–20, https://doi.org/10.1007/s007910100052.

[41] R. Krause, *A nonsmooth multiscale method for solving frictional two-body contact problems in 2D and 3D with multigrid efficiency*, SIAM J. Sci. Comput., 31 (2009), pp. 1399–1423, https://doi.org/10.1137/070682514.

[42] R. Krause, A. Rigazzi, and J. Steiner, *A parallel multigrid method for constrained minimization problems and its application to friction, contact, and obstacle problems*, Comput. Vis. Sci., 18 (2016), pp. 1–15, https://doi.org/10.1007/s00791-016-0267-1.

[43] S. Lanquetin and M. Neveu, *Reverse Catmull-Clark subdivision*, in Conference Proceedings WSCG'2006, UNION Agency—Science Press, 2006.

[44] C. Loop, *Smooth Subdivision Surfaces Based on Triangles*, master's thesis, Department of Mathematics, University of Utah, Salt Lake City, UT, 1987.

[45] S. G. Nash, *A multigrid approach to discretized optimization problems*, Optim. Methods Softw., 14 (2000), pp. 99–116, https://doi.org/10.1080/10556780008805795.

[46] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed., Springer, New York, 2006, https://doi.org/10.1007/978-0-387-40065-5.

[47] Pixar Animation Studios, *OpenSubdiv*, 2017, http://graphics.pixar.com/opensubdiv.

[48] L. Pospíšil, *Development of Algorithms for Solving Minimizing Problems with Convex Quadratic Function on Special Convex Sets and Applications*, Ph.D. thesis, Vysoká škola báňská-Technická univerzita Ostrava, Czech Republic, 2015; available online at http://hdl.handle.net/10084/110918.

[49] M. J. D. Powell, *A new algorithm for unconstrained optimization*, in Nonlinear Programming, J. B. Rosen, O. L. Mangasarian, and K. Ritter, eds., Academic Press, 1970, pp. 31–65, https://doi.org/10.1016/B978-0-12-597050-1.50006-3.

[50] M. J. D. Powell, *A hybrid method for nonlinear equations*, in Numerical Methods for Nonlinear Algebraic Equations, P. Rabinowitz, ed., Gordon and Breach Science Publishers, London, 1970, pp. 87–114.

[51] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia, 2003, https://doi.org/10.1137/1.9780898718003.

[52] F. Samavati, H.-R. Pakdel, C. Smith, and P. Prusinkiewicz, *Reverse Loop Subdivision*, Technical Report 2003-730-33, University of Calgary, Calgary, AB, Canada, 2003, https://doi.org/10.11575/PRISM/30995.

[53] O. Schenk and K. Gärtner, *Solving unsymmetric sparse systems of linear equations with PARDISO*, Future Gener. Comput. Syst., 20 (2004), pp. 475–487, https://doi.org/10.1016/j.future.2003.07.011.

[54] J. Stam, *Exact evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values*, in Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '98, ACM, 1998, pp. 395–404, https://doi.org/10.1145/280814.280945.

[55] T. Steihaug, *The conjugate gradient method and trust regions in large scale optimization*, SIAM J. Numer. Anal., 20 (1983), pp. 626–637, https://doi.org/10.1137/0720042.

[56] G. Strang and G. Fix, *An Analysis of the Finite Element Method*, 2nd ed., Wellesley-Cambridge Press, Wellesley, MA, 2008.

[57] R. Tamstorf, T. Jones, and S. F. McCormick, *Smoothed aggregation multigrid for cloth simulation*, ACM Trans. Graph., 34 (2015), 245, https://doi.org/10.1145/2816795.2818081.

[58] M. Tang, H. Wang, L. Tang, R. Tong, and D. Manocha, *CAMA: Contact-aware matrix assembly with unified collision handling for GPU-based cloth simulation*, Comput. Graph. Forum, 35 (2016), pp. 511–521, https://doi.org/10.1111/cgf.12851.

[59] B. Thomaszewski, S. Pabst, and W. Blochinger, *Parallel techniques for physically based simulation on multi-core processor architectures*, Comput. Graph., 32 (2008), pp. 25–40, https://doi.org/10.1016/j.cag.2007.11.003.

[60] B. Thomaszewski, M. Wacker, and W. Strasser, *A consistent bending model for cloth simulation with corotational subdivision finite elements*, in ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '06, The Eurographics Association, 2006, pp. 107–116, https://doi.org/10.2312/SCA/SCA06/107-116.

[61] F. Tisseur, *Newton's method in floating point arithmetic and iterative refinement of generalized eigenvalue problems*, SIAM J. Matrix Anal. Appl., 22 (2001), pp. 1038–1057, https://doi.org/10.1137/S0895479899359837.

[62] P. L. Toint, *Towards an efficient sparsity exploiting Newton method for minimization*, in Sparse Matrices and Their Uses, I. S. Duff, ed., Academic Press, London, 1981, pp. 57–88.

[63] D. Zorin, *A method for analysis of C^1-continuity of subdivision surfaces*, SIAM J. Numer. Anal., 37 (2000), pp. 1677–1708, https://doi.org/10.1137/S003614299834263X.