

Fast Corotated FEM using Operator Splitting

T. Kugelstadt¹ and D. Koschier² and J. Bender¹

¹RWTH Aachen University, Germany

²University College London, UK

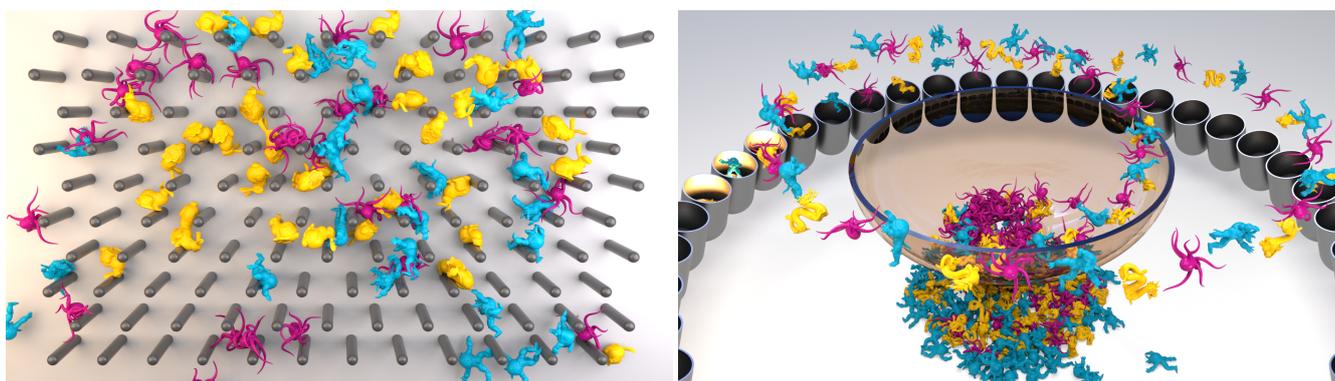


Figure 1: Left: Real-time simulation of 105 deformable solids consisting of 356k tetrahedral elements (without collision handling). Right: Large scene with 320 solids consisting of 1.42 million tetrahedral elements are colliding with each other as they are falling through a funnel after they have been shot through the air. The simulation of the deformation required only 140ms per step on a standard CPU.

Abstract

In this paper we present a novel operator splitting approach for corotated FEM simulations. The deformation energy of the corotated linear material model consists of two additive terms. The first term models stretching in the individual spatial directions and the second term describes resistance to volume changes. By formulating the backward Euler time integration scheme as an optimization problem, we show that the first term is invariant to rotations. This allows us to use an operator splitting approach and to solve both terms individually with different numerical methods. The stretching part is solved accurately with an optimization integrator, which can be done very efficiently because the system matrix is constant over time such that its Cholesky factorization can be precomputed. The volume term is solved approximately by using the compliant constraints method and Gauss-Seidel iterations. Further, we introduce the analytic polar decomposition which allows us to speed up the extraction of the rotational part of the deformation gradient and to recover inverted elements. Finally, this results in an extremely fast and robust simulation method with high visual quality that outperforms standard corotated FEMs by more than two orders of magnitude and even the fast but inaccurate PBD and shape matching methods by more than one order of magnitude without having their typical drawbacks. This enables a very efficient simulation of complex scenes containing more than a million elements.

CCS Concepts

•Computing methodologies → Physical simulation;

1. Introduction

The interactive physically-based simulation of deformable solids that undergo large deformations is an important research topic in computer graphics with many applications like games, VR, and medical training simulators. The main challenge is that realistic de-

formations are non-linear, volumetric effects. Many approaches use a continuum mechanical formulation in combination with a finite element method (FEM) to obtain realistic results. This enables the simulation of a wide range of effects like elastic and plastic deformation, lateral contraction, cutting, and fracture.

In recent years a lot of research focused on increasing the performance of FEM solvers so that they can be used interactively. However, an accurate spatial discretization of a deforming solid requires a large number of degrees of freedom which results in a large number of non-linear equations that have to be solved in each time step when using a stable, implicit time integration scheme. The corotated FEM considers the non-linearity by extracting per-element rotations. Therefore, only a linear system has to be solved in each step which increases the performance. However, so far only relatively small scenes with a few thousand tetrahedrons can be simulated at interactive frame rates on a standard CPU.

The main computational costs of corotated FEM arise from three parts of the algorithm, which have to be executed in every time step of the simulation. The first one is extracting the rotational part of the deformation gradient for each element by computing the polar decomposition. The second one is updating the stiffness matrix by applying the rotations to the matrix blocks that belong to each element which requires a large number of 3×3 matrix products. And the third one is solving the linear system for the implicit time integration. To drastically increase the computational performance of corotated FEM, it is necessary to make all three parts faster because their computation times lie within the same order of magnitude.

Contributions: In this paper we address these three performance bottlenecks of the corotated FEM. We formulate the implicit Euler method as an optimization problem and show that one of the two terms in the corotated deformation energy, which models stretching resistance, does not depend on the rotation. We use an operator splitting approach which allows us to perform the time integration of both terms in the deformation energy independently with different numerical methods. The equation for the stretching part is solved accurately with an optimization integrator. This can be done very efficiently because the Cholesky factorization of the system matrix can be precomputed once and does not change during the simulation. Afterwards, we approximate the contribution of the second term, which models resistance to volume change, by using a compliant constraint formulation. Overall, this results in large speedups because the Cholesky solves are extremely fast and updates of the stiffness matrix are omitted completely.

As a second contribution, we introduce the *analytic polar decomposition (APD)* to compute the rotational part of the deformation gradient. It can be easily computed by solving an optimization problem on $SO(3)$, exploits the temporal coherence of the simulation, is robust under single precision floating point arithmetic and does not require trigonometric or square root functions. Further, it robustly handles inverted elements without the need for special treatment and can be computed in a few lines of code without branching so that it can be easily vectorized and parallelized.

In summary, this results in stable simulations that nearly reach the visual quality of corotated FEM, but can be computed more than 100 times faster as demonstrated in several comparisons and benchmarks. Our method even outperforms geometrically motivated approaches like shape matching and position based dynamics (PBD) by more than one order of magnitude while avoiding their typical problem that the stiffness depends on the temporal and spatial discretization. Using our approach large scenes with 356k tetrahedrons can be simulated in real-time and it also efficiently

handles complex collision scenarios with 1.42 million tetrahedrons (see Figure 1).

2. Related Work

Many approaches for the simulation of deformable solids have been developed in the past decades. These methods can be loosely categorized into physical models, i.e. mass-spring models and continuum-based approaches, and geometrically motivated methods. Therefore, this section is organized as follows.

Physics-Based Models Many early approaches for the physically-based simulation of deformable objects employ discrete particle models that are coupled using linear springs or more general elastic potentials, e.g. [HH98, THMG04]. While these discrete systems are simple and robust, choosing the constitutive parameters to model a certain material is non-trivial. Moreover, these models can not guarantee convergence under spatial refinement (cf. [NMK*06]). Due to these issues, more and more attention has been paid to continuum-based models in recent years. Following the continuum approach, the dynamic behavior of a deformable object is mathematically described by a partial differential equation (PDE) that has to be discretized for numerical solving. Although many different techniques for spatial discretization exist, e.g. finite difference schemes [TPBF87], the boundary element method [JP99], or the finite volume method [TBHF03], the FEM has become increasingly popular due to its ability to solve PDEs on complex domains using irregular meshes. However, in order to simulate large deformations in deformable solids, non-linear material models have to be employed resulting in non-linear equation systems that have to be solved during runtime.

In order to alleviate the non-linearity, Müller et al. [MDM*02] introduced the concept of corotational elasticity to the computer graphics community using per vertex rotations. The main idea of the corotational approach is to rotate deformed finite elements back into their unrotated reference frame and to measure the deformation using a linear strain measure in this unrotated frame. However, since the per vertex rotation matrices are ambiguous due to the dependency on the vertex' adjacent vertices, the method introduces ghost forces. Therefore, Eitzmuß et al. [EKS03], Müller and Gross [MG04], and Hauth and Strasser [HS04] extended the approach to element-wise corotations for the simulation of cloth and three-dimensional deformables without the presence of ghost forces. Please note that all of the mentioned corotational approaches only compute an approximation to the exact stiffness matrix as pointed out by Barbič [Bar12]. Computing the exact stiffness matrix improves stability and robustness, but the additional terms are often dropped in interactive and real-time applications to improve the computational performance. Because we are aiming for fast simulations in this work, we also employ the approximate stiffness matrix.

The corotational approach was later adopted in many works. A method for the simulation of Kirchhoff-Love shells using corotational subdivision finite elements was proposed by Thomaszewski et al. [TWS06]. Kaufmann et al. [KMBG08] introduced a discontinuous Galerkin discretization based on a corotational material model in order to support non-conforming shape functions on

general polyhedral elements. A method for the art-directable animation of solid objects based on a corotational model was presented by Schumacher et al. [STC*12]. Instead of choosing a linear finite element basis, Bargeil and Cohen [BC14] presented a method that discretizes the corotational model using quadratic Bézier polynomials. In order to improve the computational performance, multigrid solvers were employed in several works, e.g. [GW08, DGW11, MZS*11]. Another way to improve the runtime performance of finite element simulations is to reduce the number of degrees of freedom using model order reduction techniques, e.g. [BJ05, AKJ08, KJ11, PBH15, XB16]. However, the reduction process is computationally intensive and inextricably linked with a loss of high-frequency features in the deformation. Another method to improve the simulation performance was proposed by Hecht et al. [HLSO12]. They exploit the fact that the stiffness matrix is constant as long as the rotation of the underlying finite elements remains constant. After computing an initial Cholesky factorization, they incrementally perform rank updates only for entries associated with elements that are subjected to substantial rotations. Similarly, we use a Cholesky factorization for solving. However, in our novel formulation, the system matrix remains constant as long as the mesh topology, the rest configuration, and the material parameters do not change, even in the case of large rotations. This means that the Cholesky factorization can be precomputed once and the linear solve in each time step can be computed very efficiently.

Most of the discussed corotational methods employ a linearized backward Euler time integrator which requires to solve a linear equation system during each time step. In order to speed up the simulation, faster and more robust solvers for time integration were developed. Gast et al. [GSS*15] reformulate the backward Euler time integration scheme as an optimization problem for simulating a variety of materials, including solids based on corotational formulations. The method is moreover able to handle hard constraints. Bouaziz et al. [BML*14] follow a similar strategy using an optimization-based integrator but suggest to use an efficient block coordinate descent solver to improve efficiency. It was later shown by Narain et al. [NOB16] that the approach of Bouaziz et al. is a special case of the alternating direction method of multipliers. Using this generalization they are able to handle a wider range of nonlinear constitutive models. An alternative approach to generalize and improve the method of Bouaziz et al. was proposed by Liu et al. [LBK17]. The generalization to arbitrary material models is achieved by reinterpreting the approach as a quasi-Newton method. They further show that the method can be accelerated using L-BFGS updates. We also use the optimization formulation of the backward Euler method. In the case of corotated finite elements, it leads to a linear least squares problem and it reveals the fact that one of the terms in the corotated deformation energy is invariant to rotation, which is the basis of our method.

Another important part of corotated FEM is the extraction of the rotational part of the deformation gradient. This can be done by using the polar decomposition (PD) as proposed by Müller and Gross [MG04]. A problem with this method is that the orthogonal factor of the PD contains a reflection in the case of inverted elements. As a result the element is forced towards the reflected rest pose, which can lead to inversions in adjacent elements and in the worst case destroy the entire mesh. Therefore, Irving et al. [ITF04]

proposed a heuristic approach, which robustly handles inverted elements by using the singular value decomposition (SVD) instead of the PD. A more detailed discussion of invertible FEM can be found in Section 4. This approach was further improved by Schmedding and Teschner [ST08], who used geometric properties of the inverted element to ensure that it recovers along the shortest possible path. Another inversion handling method was proposed by Civit-Flores and Susin [CFS14], who ensure a temporally consistent recovery from inverted states. The previously discussed methods have the disadvantage, that they contain several special cases which introduce branching and prevent efficient SIMD parallelism. To overcome this issue, Müller et al. [MBCM16] proposed to compute the polar decomposition as an optimization problem, which allows for robust inversion handling and can be solved without branching. Our analytic polar decomposition is closely related to their algorithm, but we use a more efficient optimization algorithm which converges much faster.

Geometrically Motivated Methods In comparison to physics-based models, geometrically motivated methods are often computationally more efficient, robust, and simple. Most of these methods share the concept of unconstrained particle advection followed by a constraint solver that aims to maintain the undeformed shape. A concept called *Shape Matching* was introduced by Müller et al. [MHTG05], where a particle cloud is partitioned into overlapping clusters. After the unconstrained advection, a goal position for each particle based on a cluster's deformation is computed in order to restore the initial shape. The computational efficiency of the method was later improved by fast summation methods for structured grids [RJ07] and unstructured meshes [DBB11]. Müller et al. [MHHR07] proposed a similar concept entitled – *Position Based Dynamics* (PBD) – for the simulation of cloth and cloth balloons where distance and bending constraints are enforced using Gauss-Seidel or Jacobi iterations. In the following years the PBD framework was broadly generalized to handle arbitrary position constraints and to model a wide range of physical phenomena in a visually plausible fashion. For a general review of position based approaches we would like to refer the reader to the work of Bender et al. [BMM14, BMM17]. Although these geometrically motivated approaches have proven to be very robust and efficient, they often appear less realistic than physics-based methods and are not convergent under neither temporal nor spatial refinement. Recently, it was shown by Macklin et al. [MMC16] and Tournier et al. [TNGF15] that PBD is closely related to optimization integrators. Based on this insight Macklin et al. modified PBD by using a compliant constraint formulation, such that the convergence under spatial and temporal refinement is restored while maintaining stability and robustness. We also employ the compliant constraints method to integrate the part of the corotated deformation energy that models resistance to volume changes. Due to the fact that we use the accurate solution of the optimization integrator as an initial guess, our method delivers visually plausible results with a small number of Gauss-Seidel iterations.

3. Method

Our method consists of three main components: the corotated optimization integrator, the volume conservation constraint, and the

analytic polar decomposition which will be discussed in detail in the following subsections.

3.1. Corotated Optimization Integrator

First, we will introduce the optimization formulation of the implicit Euler scheme and an operator splitting approach which allows us to solve the two terms of the corotated elastic energy separately. Then we show that the first term, that models stretching along the individual spatial directions, can be formulated as a linear least-squares problem. The optimum can be found by solving a single linear system. This optimization formulation has the advantage that the system matrix is constant for all time steps, as long as the rest-pose mesh and the material parameters do not change. This means that the Cholesky factorization can be precomputed in an initialization step and the linear solves during the simulation are extremely fast. Further, the linear system can be decomposed into three independent systems for the x -, y - and z -coordinates of the positions, which can be solved in parallel.

The optimization formulation of the backward Euler method has the time stepping scheme (cf. [MTGG11, GSS*15, NOB16]):

$$\mathbf{x}(t + \Delta t) = \operatorname{argmin}_{\mathbf{x}} \frac{1}{2\Delta t^2} \|\mathbf{M}^{\frac{1}{2}}(\mathbf{x} - \tilde{\mathbf{x}})\|^2 + E(\mathbf{x}) \quad (1)$$

$$\mathbf{v}(t + \Delta t) = \frac{\mathbf{x}(t + \Delta t) - \mathbf{x}(t)}{\Delta t}, \quad (2)$$

where $\mathbf{x} = (\mathbf{x}_1^T, \dots, \mathbf{x}_{n_p}^T)^T$ is the vector that contains all positions of the n_p nodes, \mathbf{v} contains all velocities in the same way and $E(\mathbf{x})$ is a potential energy which models internal forces. The vector $\tilde{\mathbf{x}} = \mathbf{x}(t) + \mathbf{v}(t)\Delta t + \mathbf{M}^{-1}\mathbf{f}_{ext}\Delta t^2$ contains the predicted state, which is computed using a forward Euler step without considering the internal forces/potentials. Here, \mathbf{M} is the (lumped) mass matrix, \mathbf{f}_{ext} is the vector that contains all external forces like gravity and Δt is the time step size.

We are especially interested in the corotated linear deformation energy density [SB12]

$$\Psi = \mu \|\mathbf{F} - \mathbf{R}\|_{\mathbb{F}}^2 + \frac{\lambda}{2} \operatorname{tr}(\mathbf{R}^T \mathbf{F} - \mathbb{1})^2, \quad (3)$$

where \mathbf{F} is the deformation gradient, \mathbf{R} is the rotational part of \mathbf{F} which can be computed via polar decomposition or SVD, $\mathbb{1}$ is the identity matrix and μ and λ are the Lamé parameters. The first term models stretching and compression resistance for the individual spatial directions and the second term describes resistance to volume change. As noted by Smith et al. [SDGK17] the volume term is a linearization of the non-linear volume term of the Neo-Hookean model $(\log(\det(\mathbf{F})))^2 \approx \operatorname{tr}(\mathbf{R}^T \mathbf{F} - \mathbb{1})^2$ at the rest pose. This means that volume change is poorly approximated for large deformations which we are interested in. The consequences are not only high numerical errors, but also severe visual artifacts like inversion of tetrahedral elements as noted by Stomakhin et al. [SHST12] and Smith et al. [SDGK17].

To overcome this issue Stomakhin et al. proposed a "fixed corotated" model in which they replaced the linearized volume term by the non-linear one $\lambda/2(\det(\mathbf{F}) - 1)^2$. This helps to avoid the artifacts but introduces non-linearity so that a non-linear system has to be solved in each time step instead of a linear one, which comes

along with high computational costs. Because we aim for interactive performance in large scenes with hundreds of deformable bodies, it is not an option to accurately solve the non-linear equations that arise from the fixed corotated or the Neo-Hookean energy. Instead, we will use an operator splitting approach to solve the linear equations, which result from the first term in Eq. (3), accurately with a direct solver. Afterwards, we approximate the effect of the fixed corotated volume term by using the compliant constraints formulation and applying Gauss-Seidel iterations. This provides a good trade-off between realistic volumetric compression resistance, computational costs, and numerical robustness. We will discuss volume preservation in more detail in Section 3.2.

Next, we will discuss how to discretize the stretching energy density and how to solve the optimization problem for backward Euler without considering the volume term. If we use tetrahedrons with linear shape functions, the deformation gradient is

$$\mathbf{F} = \mathbf{D}_s \mathbf{D}_m^{-1} = \mathbf{D}_s \mathbf{B} \quad (4)$$

$$\mathbf{D}_s = (\mathbf{x}_1 - \mathbf{x}_0 \quad \mathbf{x}_2 - \mathbf{x}_0 \quad \mathbf{x}_3 - \mathbf{x}_0), \quad (5)$$

$$\mathbf{D}_m = \mathbf{B}^{-1} = (\mathbf{X}_1 - \mathbf{X}_0 \quad \mathbf{X}_2 - \mathbf{X}_0 \quad \mathbf{X}_3 - \mathbf{X}_0), \quad (6)$$

where \mathbf{x}_i denote the vertex positions of the tetrahedron in the deformed pose and \mathbf{X}_i are the vertex positions in the rest pose. The discrete corotated deformation energy of tetrahedron t becomes

$$E_t(\mathbf{x}) = \mu V_t \|\mathbf{F}_t - \mathbf{R}_t\|_{\mathbb{F}}^2 = \mu V_t \|\operatorname{vec}(\mathbf{F}_t) - \operatorname{vec}(\mathbf{R}_t)\|^2 \quad (7)$$

$$= \mu V_t \|\mathbf{D}_t \mathbf{x} - \operatorname{vec}(\mathbf{R}_t)\|^2, \quad (8)$$

where V_t is the rest pose volume of the tetrahedron t , $\operatorname{vec}(\mathbf{F}_t)$ and $\operatorname{vec}(\mathbf{R}_t)$ are defined by stacking the columns into a single 9d vector, and \mathbf{D}_t is a $9 \times 3n_p$ matrix that relates the vertex positions to the vectorized deformation gradient $\mathbf{D}_t \mathbf{x} = \operatorname{vec}(\mathbf{F}_t)$. If we consider a single tetrahedron, this takes the form

$$\operatorname{vec}(\mathbf{F}_t) = \underbrace{\begin{pmatrix} -\sum_i \mathbf{B}_{i,0} \mathbb{1} & \mathbf{B}_{0,0} \mathbb{1} & \mathbf{B}_{1,0} \mathbb{1} & \mathbf{B}_{2,0} \mathbb{1} \\ -\sum_i \mathbf{B}_{i,1} \mathbb{1} & \mathbf{B}_{0,1} \mathbb{1} & \mathbf{B}_{1,1} \mathbb{1} & \mathbf{B}_{2,1} \mathbb{1} \\ -\sum_i \mathbf{B}_{i,2} \mathbb{1} & \mathbf{B}_{0,2} \mathbb{1} & \mathbf{B}_{1,2} \mathbb{1} & \mathbf{B}_{2,2} \mathbb{1} \end{pmatrix}}_{\mathbf{D}_t} \begin{pmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \end{pmatrix}. \quad (9)$$

The matrix \mathbf{D}_t only depends on the rest shape matrix \mathbf{D}_m of the tetrahedron and is constant as long as the rest pose does not change. To obtain the total energy we have to take the sum over all tetrahedrons which can be written in vector-matrix notation as

$$E(\mathbf{x}) = \sum_t E_t(\mathbf{x}) = \|\mathbf{K}^{1/2}(\mathbf{D}\mathbf{x} - \mathbf{r})\|^2, \quad (10)$$

where $\mathbf{K} = \operatorname{diag}(\mu V_1 \mathbb{1}_{9 \times 9}, \dots, \mu V_{n_t} \mathbb{1}_{9 \times 9})$ contains the material parameters and volumes, the $9n_t \times 3n_p$ matrix \mathbf{D} is a stack of the \mathbf{D}_t matrices of all n_t tetrahedrons and the $9n_t$ dimensional vector $\mathbf{r} = (\operatorname{vec}(\mathbf{R}_1)^T, \dots, \operatorname{vec}(\mathbf{R}_{n_t})^T)^T$ contains the vectorized rotations of all tetrahedrons. Details on the computation of these matrices can be found in Algorithm 1. Plugging the potential energy into (1) results in the weighted, linear least squares problem

$$\mathbf{x}(t + \Delta t) = \operatorname{argmin}_{\mathbf{x}} \frac{1}{2\Delta t^2} \|\mathbf{M}^{\frac{1}{2}}(\mathbf{x} - \tilde{\mathbf{x}})\|^2 + \|\mathbf{K}^{1/2}(\mathbf{D}\mathbf{x} - \mathbf{r})\|^2.$$

It is easy to see that the objective function is convex (we assume that \mathbf{r} is constant during one time step), meaning that we can find

Algorithm 1 Initialization

```

1: for all tetrahedrons  $t$  do
2:   compute  $\mathbf{D}_m$  using Eq. (6)
3:    $V_t \leftarrow \frac{1}{6} \det(\mathbf{D}_m)$ 
4:   for  $i \in \{0, \dots, 8\}$  do ▷ Init matrix  $\mathbf{K}$ 
5:      $\mathbf{K}(9t + i, 9t + i) \leftarrow \mu \mathbf{V}$ 
6:   for all Vertices  $v$  of tetrahedron  $t$  do ▷ Init mass  $\mathbf{M}$ 
7:      $\mathbf{M}(v, v) += \frac{1}{4} \rho V_t \mathbb{1}_{3 \times 3}$ 
8:   compute  $\mathbf{D}_t$  using Eq. (9) ▷ Init matrix  $\mathbf{D}$ 
9:   for  $i \in \{0, \dots, 3\}$  do
10:     $v \leftarrow$  index of the  $i$ -th vertex of tet  $t$ 
11:    for  $j \in \{0, \dots, 2\}$  do
12:       $\mathbf{D}(9t + 3j, 3v) \leftarrow \mathbf{D}_t(3j, 3i) \mathbb{1}_{3 \times 3}$ 
13: compute Cholesky factorization of  $\mathbf{M} + 2\Delta t^2 \mathbf{D}^T \mathbf{K} \mathbf{D}$ 
    
```

the global minimum by setting the gradient to $\mathbf{0}$ which results in the normal equations (see [NW06]):

$$(\mathbf{M} + 2\Delta t^2 \mathbf{D}^T \mathbf{K} \mathbf{D}) \mathbf{x}(t + \Delta t) = \mathbf{M} \bar{\mathbf{x}} + 2\Delta t^2 \mathbf{D}^T \mathbf{K} \mathbf{r}. \quad (11)$$

We will slightly rearrange these equations so that fixed vertices can be handled easily. We subtract $(\mathbf{M} + 2\Delta t^2 \mathbf{D}^T \mathbf{K} \mathbf{D}) \bar{\mathbf{x}}$ on both sides so that we solve for position changes $\Delta \mathbf{x} = \mathbf{x}(t + \Delta t) - \bar{\mathbf{x}}$ instead of the new positions, resulting in

$$(\mathbf{M} + 2\Delta t^2 \mathbf{D}^T \mathbf{K} \mathbf{D}) \Delta \mathbf{x} = 2\Delta t^2 \mathbf{D}^T \mathbf{K} (\mathbf{r} - \mathbf{D} \bar{\mathbf{x}}). \quad (12)$$

For fixed vertices there is no position change and we can simply remove the corresponding rows from Equation (12). From the implementation perspective it is useful to store all fixed vertices at the beginning or end of the position vector so that they can be removed efficiently with a simple block operation.

Note that the system matrix is constant as long as the mesh topology, the material parameters and the time step do not change. This means that the Cholesky factorization can be precomputed once and the linear solves can be computed extremely fast. Further, this system can be split into three independent systems for the x -, y - and z -components of the vertex positions which allows for parallelization of the forward and backward substitution steps. An overview of one time integration step can be found in Algorithm 2.

3.2. Volume Conservation

After performing time integration with the corotated optimization integrator without considering the volume term, we will take it into account now, by using the compliant constraints formulation. Tournier et al. [TNGF15] and Macklin et al. [MMC16] showed that it is equivalent to implicit integration of elastic potentials. Analogously to Macklin et al. we construct the compliant constraints $C_t(\mathbf{x})$ for each tetrahedron t by writing the fixed corotated volume energy from Stomakhin et al. [SHST12] as

$$E_{vol,t} = \frac{1}{2} C_t^T \alpha_t^{-1} C_t = \frac{\lambda}{2} V_t (\det(\mathbf{F}_t) - 1)^2. \quad (13)$$

This defines the volume constraint

$$C_t(\mathbf{x}) = \det(\mathbf{F}_t) - 1 = \frac{[(\mathbf{x}_1 - \mathbf{x}_0) \times (\mathbf{x}_2 - \mathbf{x}_0)] \cdot (\mathbf{x}_3 - \mathbf{x}_0)}{6V_t} - 1, \quad (14)$$

Algorithm 2 Time Step

```

1: for all vertices  $i$  do ▷ Optimization integrator
2:    $\bar{\mathbf{x}}_i = \mathbf{x}_i(t) + \mathbf{v}_i(t) \Delta t + \frac{1}{m_i} \mathbf{f}_{ext,i} \Delta t^2$ 
3:    $\mathbf{b} \leftarrow \mathbf{0} \in \mathbb{R}^{9n_t}$ 
4:   for all tetrahedrons  $t$  do
5:     compute  $\mathbf{F}_t$  using Eq. (4)
6:      $\mathbf{R}_t \leftarrow$  APD( $\mathbf{F}_t$ ) with Algorithm 3
7:      $\mathbf{b}[9t] \leftarrow \text{vec}(\mathbf{R}) - \text{vec}(\mathbf{F})$ 
8:      $\mathbf{b} \leftarrow 2\Delta t^2 \mathbf{D}^T \mathbf{K} \mathbf{b}$ 
9:     Solve  $(\mathbf{M} + 2\Delta t^2 \mathbf{D}^T \mathbf{K} \mathbf{D}) \Delta \mathbf{x} = \mathbf{b}$  with precomp. Cholesky dcmp
10:    for all tetrahedrons  $t$  do ▷ Solve volume constraints
11:       $\kappa_t \leftarrow 0$ 
12:    while  $i < solverIterations$  do
13:      for all tetrahedrons  $t$  do
14:        Update  $\mathbf{x}$  and  $\kappa_t$  using Eq. (15) and (16)
15:    for all vertices  $i$  do ▷ Update positions
16:       $\mathbf{v}_i \leftarrow \frac{1}{\Delta t} (\mathbf{x}_i(t + \Delta t) - \mathbf{x}_i(t))$ 
    
```

where \mathbf{x}_i are the four vertex positions of tetrahedron t and the compliance factor $\alpha_t^{-1} = \lambda V_t \Delta t^2$ which relates the material parameter, the rest volume V_t , and the time step Δt to the stiffness of the constraint. A similar deformation energy was used by Teschner et al. [THMG04] in combination with explicit time integration.

To solve the system of n_t volume constraints we use Gauss-Seidel iterations as proposed by Macklin et al. We will skip the derivation, which can be found in [MMC16], and just present the equations that are needed to update the Lagrange multipliers κ_t and the vertex positions \mathbf{x}_i when stepping from iteration n to $n + 1$:

$$\Delta \kappa_t = \frac{C_t - \alpha_t \kappa_t^n}{\sum_{i=0}^3 \frac{1}{m_i} \|\nabla_{\mathbf{x}_i} C_t\|^2 + \alpha_t} \quad (15)$$

$$\kappa_t^{n+1} = \kappa_t^n + \Delta \kappa_t, \quad \mathbf{x}_i^{n+1} = \mathbf{x}_i^n + \frac{1}{m_i} \nabla_{\mathbf{x}_i} C_t \Delta \kappa_t. \quad (16)$$

The required gradients of Eq. (14) can be found in the supplemental document. At the beginning of each time step we initialize the Lagrange multipliers with 0. Then we iterate over all tetrahedrons, update κ_t and the four vertex positions and move on to the next constraint. More details on how the volume constraints fit into our time stepping scheme can be found in Algorithm 2. In practice a small number of Gauss-Seidel iterations is sufficient to deliver visually plausible lateral contractions (see Figure 3) because the accurate solution of the stretching part provides a good initial guess.

3.3. Analytic Polar Decomposition (APD)

In the previous sections we showed how the linear solves can be done very efficiently with precomputed Cholesky factorizations. But another computational bottleneck of corotated FEM simulations is the extraction of the rotational part from the deformation gradient. In this section we will address this issue by introducing the analytic polar decomposition (APD). It can be easily implemented using only a few lines of code without branching, square roots or trigonometric functions. Element inversions are handled robustly without the requirement to treat any special cases. Further,

Algorithm 3 Analytic Polar Decomposition (APD) of matrix \mathbf{A}

```

1:  $\mathbf{q} \leftarrow \mathbf{q}(t - \Delta t)$   $\triangleright$  Start: quaternion from last time step
2: while  $\|\Delta\boldsymbol{\omega}\| > \tau$  do
3:    $\mathbf{R} \leftarrow \mathbf{q}.\text{rotationMatrix}()$ 
4:   compute gradient  $\mathbf{g}$  using Eq. (24)
5:   compute Hessian  $\mathbf{H}$  using Eq. (25)
6:    $\Delta\boldsymbol{\omega} \leftarrow \mathbf{H}^{-1}\mathbf{g}$ 
7:    $\text{clamp}(\|\Delta\boldsymbol{\omega}\|, -\pi, \pi)$ 
8:    $\mathbf{q} \leftarrow \mathbf{q} \circ \text{cay}(\Delta\boldsymbol{\omega})$  with Eq. (27)

```

the APD allows us to exploit the temporal coherence of the simulation, it is robust under single precision floating point arithmetic, and can be easily parallelized or vectorized. Overall this results in Algorithm 3 that outperforms the commonly used methods to compute the polar decompositions by a factor of up to 38.

Definition of the APD: First, we will give the definition of the APD and then we show how it can be efficiently computed. The standard polar decomposition of a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ is usually defined as $\mathbf{A} = \mathbf{R}\mathbf{S}$, where $\mathbf{R} \in \mathbb{R}^{m \times n}$ is an orthogonal matrix and $\mathbf{S} \in \mathbb{R}^{n \times n}$ is a symmetric, positive semidefinite matrix. Mehrmann and Rath [MR93] showed that the polar decomposition can be defined analogously for an analytic, matrix valued function $\mathbf{A}(t) : [a, b] \rightarrow \mathbb{R}^{m \times n}$ (e.g. the deformation gradient), where all components of the matrix \mathbf{A} are analytic functions of the variable t . The analytic polar decomposition (APD) is a path of factorizations $\mathbf{A}(t) = \mathbf{R}(t)\mathbf{S}(t)$, such that $\mathbf{R}(t)$ and $\mathbf{S}(t)$ are orthogonal and symmetric, respectively, and both are analytic functions in t . Further, Mehrmann and Rath proved that if the matrix $\mathbf{A}(t_0)$ at one specific point $t_0 \in [a, b]$ is non-singular, $\mathbf{S}(t_0)$ is positive definite and the APD is unique on $[a, b]$.

Note that the symmetric factor is not necessarily positive semidefinite for all $t \in [a, b]$. This is a useful property when we consider the APD of the deformation gradient because it ensures that in the case of inverted elements the reflection will be contained in the symmetric factor and the orthogonal factor will always be a pure rotation, given that $\det(\mathbf{R}(t_0)) = 1$. For instance, this is the case when t_0 is the starting point in the rest pose. Due to the fact that $\mathbf{R}(t)$ is a continuous function, its determinant is also continuous and there cannot be a jump from $+1$ to -1 so that the reflection has to be contained in the symmetric factor.

The APD as an optimization problem: Next, we will show that the APD of the 3×3 deformation gradient can be efficiently computed by solving an optimization problem on $SO(3)$. We adapt the pathfollowing approach of Janovsky et al. [JJT06], which was originally developed for computing the analytic singular value decomposition of arbitrary sized matrices, to determine the 3×3 APD. We define the function

$$\mathbf{G}(\mathbf{R}(t), \mathbf{S}(t), t) = \begin{pmatrix} \mathbf{A}(t) - \mathbf{R}(t)\mathbf{S}(t) \\ \mathbf{R}(t)^T \mathbf{R}(t) - \mathbb{1} \\ \mathbf{S}(t) - \mathbf{S}(t)^T \end{pmatrix}, \quad (17)$$

such that the curve, which is implicitly defined by $\mathbf{G} = \mathbf{0}$, defines the APD. The roots of \mathbf{G} can be computed by minimizing $\|\mathbf{G}\|^2$

over all rotations and all symmetric matrices

$$(\mathbf{R}(t), \mathbf{S}(t)) = \min_{\mathbf{R} \in SO(3), \mathbf{S}} \|\mathbf{A}(t) - \mathbf{R}\mathbf{S}\|^2 + \|\mathbf{S} - \mathbf{S}^T\|^2, \quad (18)$$

where the second component of \mathbf{G} can be dropped since we compute the minimum over all $\mathbf{R} \in SO(3)$. In a simulation with discrete time steps the APD can be computed by solving this optimization problem in each time step t_i for the deformation gradient $\mathbf{F}(t_i)$. By using the optimum of the last time step as starting point, we can exploit the temporal coherence of the simulation. In most cases the rotation during one time step is small so that we are already close to the optimum. Actually it is sufficient to start with $\mathbf{S} = \mathbb{1}$ and minimize over the rotation alone. It is well-known as Grioli's theorem [CMPG79] that minimizing $\|\mathbf{A} - \mathbf{R}\|$ over all rotations results in the orthogonal factor of the standard polar decomposition, so that we can find the minimizing symmetric matrix as $\mathbf{S} = \mathbf{R}^T \mathbf{A}$. This means that we have to minimize the objective function

$$E_{\text{APD}}(\mathbf{R}) = \frac{1}{2} \|\mathbf{R} - \mathbf{A}\|^2 = \frac{1}{2} \sum_{i=1}^3 (\mathbf{R}\mathbf{e}_i - \mathbf{A}\mathbf{e}_i)^2, \quad (19)$$

where \mathbf{e}_i denotes the i -th base vector.

Optimization on $SO(3)$: This can be done efficiently by using the manifold version of Newton's method which was presented by Taylor and Kriegman [TK94]. At every point \mathbf{R}_0 on the manifold $SO(3)$ we can define a local parametrization

$$\mathbf{R}(\boldsymbol{\omega}) = \mathbf{R}_0 \exp([\boldsymbol{\omega}]_{\times}) \quad \boldsymbol{\omega} \in \mathbb{R}^3, \|\boldsymbol{\omega}\| < \pi, \quad (20)$$

which is an invertible and differentiable mapping between the neighborhood of \mathbf{R}_0 and an open set in \mathbb{R}^3 . Here, the vector $\boldsymbol{\omega}$ is a rotation around the axis $\boldsymbol{\omega}$ with the angle $\|\boldsymbol{\omega}\|$ and $[\cdot]_{\times}$ is the operator that maps a given vector to the corresponding skew symmetric matrix. The matrix exponential $\exp(\cdot)$ is defined by the exponential series and can be efficiently computed with a closed form expression for the 3×3 case.

But first, we will discuss how to solve the optimization problem. We will represent the objective function in terms of $\boldsymbol{\omega}$ using Eq. (20) and compute a quadratic Taylor approximation

$$E_{\text{APD}}(\mathbf{R}(\boldsymbol{\omega})) = E_{\text{APD}}(\mathbf{R}_0) + \mathbf{g}^T \Delta\boldsymbol{\omega} + \frac{1}{2} \Delta\boldsymbol{\omega}^T \mathbf{H} \Delta\boldsymbol{\omega}, \quad (21)$$

where \mathbf{g} and \mathbf{H} are the gradient and Hessian of E_{APD} evaluated at $\boldsymbol{\omega} = \mathbf{0}$, which means at the rotation \mathbf{R}_0 . By setting the gradient of Eq. (21) to $\mathbf{0}$ we obtain the update rule

$$\Delta\boldsymbol{\omega} = -\mathbf{H}^{-1}\mathbf{g} \quad (22)$$

$$\mathbf{R}_0^{n+1} = \mathbf{R}_0^n \exp([\Delta\boldsymbol{\omega}]_{\times}), \quad (23)$$

which performs one Newton step on the local parametrization and then computes the new position on the manifold $SO(3)$. When we guarantee that $\|\Delta\boldsymbol{\omega}\| < \pi$ so that it stays in the range of the local parametrization then we can minimize the objective function by iterating these updates.

To compute the gradient and Hessian we have to determine the derivatives of $\mathbf{R}(\boldsymbol{\omega})$ which can be done component wise by using the exponential series. This can be found in detail in the supplemental document. Here we only present the final results for the gradient

\mathbf{g} and Hessian \mathbf{H} of the objective, which are

$$\mathbf{g} = \frac{\partial}{\partial \boldsymbol{\omega}} E_{\text{APD}}(\boldsymbol{\omega}) = -2\text{axl}\left(\mathbf{R}_0^T \mathbf{A}\right) \quad (24)$$

$$\mathbf{H} = \frac{\partial^2}{\partial \boldsymbol{\omega}^2} E_{\text{APD}}(\boldsymbol{\omega}) = \text{tr}\left(\mathbf{R}_0^T \mathbf{A}\right) \mathbb{1} - \text{sym}\left(\mathbf{R}_0^T \mathbf{A}\right), \quad (25)$$

where $\text{axl}(\mathbf{Y})_k = \frac{1}{2} \sum_i \sum_j \mathbf{Y}_{ij} \epsilon_{ijk}$ denotes the axial vector of the matrix \mathbf{Y} , ϵ_{ijk} is the Levi-Civita symbol, and $\text{sym}(\mathbf{Y}) = \frac{1}{2}(\mathbf{Y} + \mathbf{Y}^T)$ denotes the symmetric part of \mathbf{Y} .

Exponential and Cayley Maps: Finally, we discuss how to compute the exponential map so that we can compute the APD. There are several ways to this. E.g., one can use the Rodrigues formula. An even simpler way is to represent the rotation as a quaternion, for which the exponential map can be computed as (see [Sel10])

$$\exp(\boldsymbol{\omega}) = (\cos(\|\boldsymbol{\omega}/2\|), \sin(\|\boldsymbol{\omega}/2\|)\hat{\boldsymbol{\omega}}^T)^T, \quad (26)$$

where $\hat{\boldsymbol{\omega}} = \boldsymbol{\omega} / \|\boldsymbol{\omega}\|$. Then the update step in Equation (23) can be computed as a standard quaternion product $\mathbf{q}^{n+1} = \mathbf{q}^n \circ \exp(\Delta\boldsymbol{\omega})$. Note that it is not necessary to normalize the quaternion after the update step, because the exponential map always returns a unit quaternion.

Another alternative is the Cayley map which approximates the exponential map without using trigonometric functions and also returns a unit quaternion so that no square root for normalization is required. It can be computed as the quaternion (see [Sel10])

$$\text{cay}(\boldsymbol{\omega}) = \left(\frac{1 - \|\boldsymbol{\omega}/2\|^2}{1 + \|\boldsymbol{\omega}/2\|^2}, \frac{1}{1 + \|\boldsymbol{\omega}/2\|^2} \boldsymbol{\omega}^T \right)^T. \quad (27)$$

This quaternion also represents a rotation around the axis $\hat{\boldsymbol{\omega}}$, but the angle θ is given by $\|\boldsymbol{\omega}\| = \tan(\theta)$. This is a good approximation for small angles that are very common in our scenario because of the temporal coherence between two time steps of the simulation. For larger values of $\|\boldsymbol{\omega}\|$ the Cayley map results in too small rotations, which means that the step will reduce the objective function but convergence will be slower. So we trade slower convergence in the rare case of large rotations for faster computation times in the common scenario of small rotations. In all our experiments we used the Cayley map because it results in better performance.

4. Results

In this section we discuss our results and compare our method to other simulation methods in terms of computational performance and visual quality. All steps of our algorithm are explicitly vectorized using AVX instructions and the loop over all meshes is parallelized with OpenMP. The Cholesky factorizations are computed using the LL^T method from the Eigen library [GJ*10]. The timings in this section were measured on an Intel Core i7-7700k quad core processor.

Corotated / Non-Linear FEM Our method splits the energy density of the corotated material model in a stretching and a volume term. We solve the stretching part using a direct Cholesky solver and the volume part by compliant constraints. The significant speedup of our method in comparison to other approaches

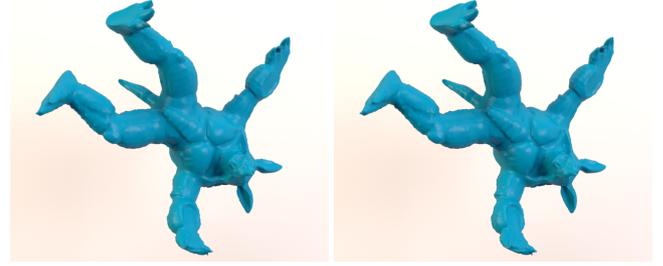


Figure 2: A swinging armadillo is simulated using our method (left) and the corotated FEM implementation in the library Vega FEM (right). The results are visually almost identical for small Poisson ratios.



Figure 3: Comparison of our method (top) and corotated FEM (bottom) in a simulation in which a beam that is fixed at both ends gets compressed, stretched and twisted (see video). The Poisson ratio was set to 0.49 such that the material is nearly incompressible. Although our method handles resistance to volume changes only approximately, it delivers visually plausible results.

is obtained by using only a weak coupling of both solutions. Our method is up to two orders of magnitude faster than the traditional corotated FEM but less accurate. In the following, we show that we still obtain a convincing visual quality.

We simulated a swinging armadillo model with our method and the corotated FEM implementation in the Vega FEM library [SSB13] using a small Poisson ratio to compare the stretching resistance (see Figure 2). In the accompanying video it can be seen that our results are almost visually identical to the full corotated FEM simulation. We performed another comparison with an almost incompressible beam model with a Poisson ratio of 0.49 which is fixed at both ends and gets compressed, stretched and twisted (see Figure 3). The results show that our method produces visually plausible lateral contractions. Note that in this experiment only one iteration of our volume constraint solver was enough to obtain a volume error of less than one percent.

It is clear that our weak coupling cannot reproduce the full visual quality of corotated or non-linear FEM simulations, especially

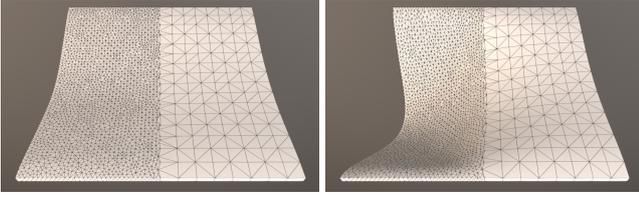


Figure 4: Comparison of our method (left) and position-based dynamics (right) in a simulation of an irregularly sampled thin sheet. PBD cannot use real material parameters and the material becomes softer in densely sampled regions. With our method we use the same value of Young’s modulus for all tetrahedrons and the stiffness is independent of the spatial sampling.

for meshes with fine sampling where many iterations are needed to reach convergence. However, in interactive applications the deformable solids are typically approximated by only a few hundred or a few thousand tetrahedral elements. In this case our method delivers visually comparable results at low computational costs.

PBD and Shape Matching Our method enables fast and high-quality simulations of deformable bodies and does not have the typical drawbacks of position based or shape matching simulations. That means our material stiffness can be specified in terms of Young’s modulus and Poisson ratio and is independent of the time step size and spatial resolution of the mesh, which is demonstrated in Figure 4. Further, there is no parameter like the shape matching cluster size which affects the material behavior and has to be tuned by the user. Recently, these issues have been addressed by Macklin et al. [MMC16] with their extended position based dynamics (XPBD) approach, which allows for using real material parameters. However, the computational costs for simulating elastic materials are higher than in standard PBD [BKCW14] because more complex, higher dimensional constraints are required.

Analytic Polar Decomposition Typically the rotational part of \mathbf{F} is found by computing the polar decomposition $\mathbf{F} = \mathbf{R}\mathbf{S}$, where \mathbf{R} and \mathbf{S} are an orthogonal and a symmetric, positive semidefinite matrix, respectively. However, in the case of inverted elements the orthogonal factor contains a reflection, which forces the element to return to the reflected rest configuration. In the worst case this can force adjacent elements into inversion which ultimately destroys the entire simulation mesh. Therefore, Irving et al. [ITF04] proposed the invertible finite elements (IFE) method that enables robust recovery from inverted states. They compute the SVD of the deformation gradient $\mathbf{F} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, where \mathbf{U} and \mathbf{V} are orthogonal and $\mathbf{\Sigma}$ is a diagonal matrix. Because the SVD is not unique, a standard convention is to choose the singular values to be positive and to arrange them in descending order. This means that the reflection of \mathbf{F} is contained in the orthogonal factors. Irving et al. made the heuristic assumption that the element is “as uninverted as possible” and proposed to obtain the rotational part of inverted elements by switching the sign of the smallest singular value and the corresponding singular vector, which does not change the SVD, and $\mathbf{R} = \mathbf{U}\mathbf{V}^T$ is a rotation matrix without a reflection.

Müller et al. [MBCM16] compute the polar decomposition of \mathbf{F}

as the minimum of $\|\mathbf{R} - \mathbf{F}\|^2$ over all rotation matrices \mathbf{R} . This is the same optimization problem as the one that we have to solve for the APD, meaning that it delivers the same results as the APD. But Müller et al. solve the optimization problem by interpreting the element as a rigid body and applying forces to drive it towards the minimum. Their approach can be interpreted as a kind of gradient descent method, which is expected to have linear convergence. In contrast, our Newton solver is expected to have quadratic convergence [TK94]. Therefore, it requires much less iterations to converge as discussed below.

In our work we use an APD to extract the rotational part of \mathbf{F} . A formal proof that the APD delivers the same results as the SVD method of Irving et al. [ITF04] can be found in the supplemental document. The advantage of the APD is that it is computationally faster than both methods above. Moreover, it is easy to implement and does not contain any branching, so that it can be easily parallelized on GPUs or on CPUs using SIMD instructions. To show that the APD faithfully handles inversions in practice, we started the simulation of an armadillo mesh in the reflected rest pose. It was able to recover its original shape in just a few time steps, which is shown in Figure 5.

Performance In this paragraph we will present several benchmark results, that are all computed on a single core. First, we compare the computational performance of our method to several corotated finite element solvers like the Vega FEM library [SSB13], the ADMM method of Narain et al. [NOB16] and the Quasi-Newton L-BFGS method of Liu et al. [LBK17]. Moreover, we compared our method to PBD, XPBD, and shape matching. We simulated the swinging armadillo scene (see Figure 2) with varying mesh resolutions and depict the computation times in Table 1. The parameters were: time step $\Delta t = 10ms$, Young’s modulus $E = 10^6 Pa$, and Poisson ratio $\nu = 0.33$. We performed 10 iterations of the ADMM and L-BFGS solver and one iteration of our volume constraint solver. The iteration count of PBD and the cluster size of shape matching were tuned such that we achieved a similar material behavior. The results show, that our method scales well with increasing mesh resolutions and that it outperforms standard corotated FEM simulations by two orders of magnitude. Our method also outperforms PBD, XPBD, and shape matching by one to two orders of magnitude, depending on the mesh resolution. In general PBD and shape matching require a higher iteration count or larger cluster sizes when the mesh resolution or the time step size increases.

We also compare the performance of the solvers for varying material stiffness and time step size, which is shown in Tables 2 and 3. In contrast to the commonly used PCG solver of the Vega FEM library, which needs more iterations with increasing stiffness or time step size, the computation times of our direct solver are not affected by these parameters. This means that we can efficiently simulate stiff materials with large time steps.

In order to show the computational performance of our APD, we made benchmarks of several polar and singular value decomposition methods. We randomly sampled 800k 3×3 matrices from a uniform distribution on $SO(3)$ with a maximal angle of 10° . Then they were used to transform the vertices of a tetrahedron that was slightly deformed by applying a random displacement of maximally 10% of the average edge length to all vertices. The maxi-



Figure 5: Left to right: Armadillo model recovers from an completely inverted state using the analytic polar decomposition.

Mesh Resolution		Simulation Times in Milliseconds / Speedup of our Method							Shape Matching
# of vert.	# of elem.	Our Method	Vega Fem	ADMM	L-BFGS	PBD	XPBD		
1849	6088	0.44	55.7 / 126x	81.6 / 184x	104 / 236x	7.12 / 16x	25.3 / 57x	6.31 / 14x	
3063	10349	0.85	120 / 140x	145 / 169x	178 / 208x	21.1 / 25x	42.9 / 50x	19.9 / 23x	
4716	16225	1.64	250 / 152x	225 / 138x	283 / 172x	50.5 / 31x	115 / 70x	57.9 / 35x	
9392	33076	4.13	704 / 170x	486 / 118x	563 / 136x	263 / 64x	494 / 120x	276 / 67x	
14791	55494	8.31	1509 / 182x	840 / 101x	997 / 120x	590 / 71x	1079 / 130x	908 / 109x	
43805	173196	28.2	6611 / 234x	1455 / 52x	3368 / 119x	6382 / 226x	9762 / 346x	-	
77954	330814	54.2	18489 / 341x	2666 / 49x	6423 / 118x	31377 / 579x	45110 / 832x	-	

Table 1: Comparison of simulation times in milliseconds with varying mesh resolution. We compare our method against the standard corotated linear FEM method from the Vega FEM library [SSB13], the ADMM optimization integrator [NOB16], the L-BFGS integrator [LBK17], the position based continuous materials method [BKCW14], the extended PBD (XPBD) method [MMC16], and the clustered shape matching method [MHTG05]. Please note, that we left out measurements for the very high-resolution meshes with shape matching due to the high memory demand associated with the growing cluster size.

E [Pa]	Simulation times in ms / Speedup of our method			
	Our method	Vega Fem	ADMM	L-BFGS
10^5	1.60	131 / 82x	222 / 139x	384 / 239x
$5 \cdot 10^5$	1.77	205 / 116x	226 / 128x	358 / 202x
10^6	1.63	236 / 145x	226 / 138x	282 / 173x
10^7	1.59	552 / 347x	226 / 142x	198 / 124x

Table 2: Comparison of simulation times per step in milliseconds with varying material stiffness. We simulated the Stanford armadillo with 16k elements and a time step of $\Delta t = 10\text{ms}$.

Δt [ms]	Simulation times in ms / Speedup of our method			
	Our method	Vega Fem	ADMM	L-BFGS
5	1.68	128 / 76x	226 / 134x	366 / 217x
10	1.63	240 / 147x	226 / 138x	282 / 173x
16	1.59	330 / 207x	226 / 142x	184 / 116x
33	1.64	576 / 352x	226 / 138x	174 / 107x

Table 3: Comparison of simulation times per step in milliseconds with varying time step sizes. We simulated the Stanford armadillo with 16k elements and Young's modulus of $E = 10^6\text{Pa}$.

mal angle and the maximal deformation mimics the temporal coherence of the FEM simulation, where the deformations and rotations between two time steps are commonly small. We enforced an average error below 0.05° (compared to the SVD solution using the angular metric $d(\mathbf{R}_1, \mathbf{R}_2) = \angle(\mathbf{R}_1^T \mathbf{R}_2)$). The timings for computing the rotational parts of the 800k matrices are summarized in Table 4. The best performance can be reached with our ADP implementation that uses AVX instructions to compute 8 APDs at once. It is more than two times faster than the SVD method of McAdams et al. [MST*11], which also uses AVX vectorization. In contrast to the approach of McAdams et al. the APD requires just a few lines of code and could be easily implemented on GPUs. Further, the APD is 21 times faster than the polar decomposition of Müller et al. [MBCM16], which required 5 iterations to con-

verge to the same error as the APD with 1 iteration. Compared to the standard polar decomposition (implementation from the PositionBasedDynamics library [Ben17]), the invertible FEM heuristic (code from [MBCM16]) and the implicitly shifted QR-SVD by Gast et al. [GFJT16], the APD is 14 – 15 times faster. The standard SVD from the Eigen library turned out to be the slowest method and needs 38 times more computation time than the APD. This shows that the APD is a fast, simple to implement and easy to use alternative to common polar decomposition algorithms.

Since the algorithm of Müller et al. [MBCM16] solves the same optimization problem as the one we have to solve for the APD, we compare the convergence of both methods. For this comparison we again use the random deformation gradient scenario (see above).

Polar Decomposition Times		
Method	Time [ms]	Speedup
Our Method (APD) SIMD	18.7	-
Our Method (APD)	73.5	3.9x
Müller et al. [MBCM16]	384	21x
SVD: McAdams et al. [MST*11]	40.5	2.2x
QR SVD: Gast et al. [GFJT16]	283	15x
Irving et al. [ITF04]	271	14x
Polar Decomposition	267	14x
Jacobi SVD Eigen [GJ*10]	704	38x

Table 4: Comparison of polar decomposition times in milliseconds and speedups of the SIMD APD for 800k deformation gradients of randomly rotated and deformed tetrahedrons. An average error below 0.05° was enforced. The starting rotation was limited to 10° , which corresponds to 167 rpm for $\Delta t = 10\text{ms}$, to mimic the effect of temporal coherence.

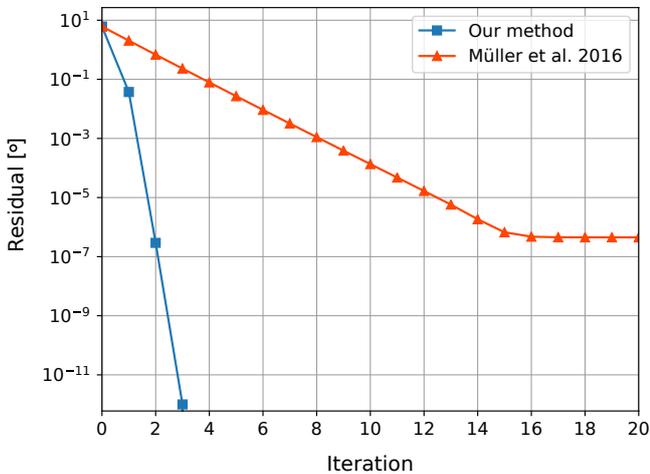


Figure 6: Comparison of the convergence of the APD, where we expect quadratic convergence, to the algorithm of Müller et al. where we expect a linear one. This result shows the expected behavior and our method needs much less iterations to converge. The computation times per iteration are nearly identical for both methods. The residual is computed as the angular distance $d(\mathbf{R}_1, \mathbf{R}_2) = \angle(\mathbf{R}_1^T \mathbf{R}_2)$ to the exact result which was computed with an SVD.

The results, shown in Figure 6, demonstrate the much faster convergence of the APD. The APD requires only 3% more time per iteration than the method of Müller et al. Therefore, our method is about 21 times faster (see Table 4).

Complex Scenarios To show that our method can simulate large scenes at interactive rates, we simulated 105 deformable solids consisting of 356k tetrahedrons falling through horizontal poles (see Figure 1, left). Collisions between the meshes were not handled in this scene. The step size was $\Delta t = 20\text{ms}$ and a complete simulation step required 17ms on average when all models are active. This means the simulation was faster than real-time. For the second

large scenario we added collision handling. We use the spatial hashing algorithm [THM*03] in combination with a Signed Distance Field (SDF) [KDBB17] in the reference configuration to perform the collision detection. Similar to McAdams et al. [MZS*11] the SDF is used to determine the closest point on the undeformed surface. The corresponding deformed position of this point is used to resolve the collision using the position-based approach [BMM17]. Figure 1 (right) shows 320 solids consisting of 1.42 million tetrahedrons that are colliding with each other as they are falling through a funnel after they have been shot through the air. The deformation was simulated in 140ms per time step on average.

5. Conclusion and Future Work

We presented a fast and stable method for interactive simulation of deformable bodies using corotated finite elements and optimization based implicit time integration. Our method builds upon the observation that one part of the corotated stiffness matrix, which accounts for stretching resistance in the individual spatial directions, is invariant to rotations. This means that we can use an operator splitting approach and solve the resulting linear system very efficiently, because the corresponding matrix is constant for all time steps and can be prefactored. The effect of volumetric compression resistance, which is modeled by the remaining term, is approximated using compliant constraints, which offers a good trade-off between accuracy and computation time. Further, we introduced the analytic polar decomposition as a fast and simple way to compute the rotational part of the deformation gradient with robust handling of inverted elements. Finally, this results in extremely fast simulations with high visual quality, that outperform standard corotated FEMs by more than two orders of magnitude and even outperform the fast but inaccurate PBD and shape matching methods by more than one order of magnitude without having their typical drawbacks. This enables a very efficient simulation of complex models consisting of more than a million finite elements.

A limitation of our method is that the volumetric compression resistance is roughly approximated using compliant constraints. This may not be a problem in typical interactive scenarios like games, where computational performance and numerical robustness are more important than physical accuracy, and where the commonly used techniques like PBD and shape matching deliver less accurate results at higher computational costs than our method. But there are many important applications of FEM simulations, where incompressibility is crucial, like character skinning for visual effects [MZS*11, SDGK17]. Therefore, we are planning to combine our method with more accurate volume conservation techniques like the one of Irving et al. [ISF07].

Another limitation is that our method suffers from numerical dissipation due to the implicit Euler integration with large time steps. Therefore, we are planning to combine our approach with the method of Dinev et al. [DLK18] to achieve energy conservation without sacrificing the stability of the implicit Euler scheme and the computational speed that comes along with being able to use large time steps.

Acknowledgment This work is supported by the German Research Foundation (DFG) under contract number BE 5132/3-1.

References

- [AKJ08] AN S. S., KIM T., JAMES D. L.: Optimizing cubature for efficient integration of subspace deformations. *ACM Transactions on Graphics* 27, 5 (2008), 1. 3
- [Bar12] BARBIĆ J.: *Exact Corotational Linear FEM Stiffness Matrix*. Tech. rep., University of Southern California, 2012. 2
- [BC14] BARGTEIL A. W., COHEN E.: Animation of deformable bodies with quadratic bézier finite elements. *ACM Transactions on Graphics* 33, 3 (June 2014), 27:1–27:10. 3
- [Ben17] BENDER J.: PositionBasedDynamics Library. <https://github.com/InteractiveComputerGraphics/PositionBasedDynamics>, 2017. 9
- [BJ05] BARBIĆ J., JAMES D. L.: Real-Time Subspace Integration for St. Venant-Kirchhoff Deformable Models. *ACM Transactions on Graphics* 24, 3 (July 2005), 982–990. 3
- [BKCW14] BENDER J., KOSCHIER D., CHARRIER P., WEBER D.: Position-Based Simulation of Continuous Materials. *Computers & Graphics* 44, 1 (2014), 1–10. 8, 9
- [BML*14] BOUAZIZ S., MARTIN S., LIU T., KAVAN L., PAULY M.: Projective Dynamics: Fusing Constraint Projections for Fast Simulation. *ACM Transactions on Graphics* 33, 4 (2014), 1–11. 3
- [BMM14] BENDER J., MÜLLER M., MACKLIN M.: A Survey on Position-Based Simulation Methods in Computer Graphics. *Computer Graphics Forum* 33, 6 (2014), 228–251. 3
- [BMM17] BENDER J., MÜLLER M., MACKLIN M.: A survey on position based dynamics, 2017. In *EUROGRAPHICS 2017 Tutorials* (2017), Eurographics Association. 3, 10
- [CFS14] CIVIT-FLORES Ó., SUSÍN A.: Robust treatment of degenerate elements in interactive corotational fem simulations. *Computer Graphics Forum* 33, 6 (2014), 298–309. 3
- [CMPG79] CARLOS MARTINS L., PODIO-GUIDUGLI P.: A variational approach to the polar decomposition theorem. *Rendi-conti/Accademia Nazionale dei Lincei, Roma, Classe di Scienze Fisiche, Matematiche e Naturali* 66 (1979). 6
- [DBB11] DIZIOL R., BENDER J., BAYER D.: Robust Real-Time Deformation of Incompressible Surface Meshes. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2011), ACM, pp. 237–246. 3
- [DGW11] DICK C., GEORGII J., WESTERMANN R.: A Hexahedral Multigrid Approach for Simulating Cuts in Deformable Objects. *IEEE Transactions on Visualization and Computer Graphics* 17, 11 (2011), 1663–1675. 3
- [DLK18] DINEV D., LIU T., KAVAN L.: Stabilizing integrators for real-time physics. *ACM Transactions on Graphics (TOG)* 37, 1 (2018), 9. 10
- [EKS03] ETZMUSS O., KECKEISEN M., STRASSER W.: A Fast Finite Element Solution for Cloth Modelling. In *Pacific Conference on Computer Graphics and Applications* (EKS 2003), IEEE Computer Society, pp. 244–251. 2
- [GFJT16] GAST T., FU C., JIANG C., TERAN J.: *Implicit-shifted symmetric qr singular value decomposition of 3×3 matrices*. Tech. rep., Technical report, University of California Los Angeles, 2016. 9, 10
- [GJ*10] GUENNEBAUD G., JACOB B., ET AL.: Eigen v3. <http://eigen.tuxfamily.org>, 2010. 7, 10
- [GSS*15] GAST T., SCHROEDER C., STOMAKHIN A., JIANG C., TERAN J.: Optimization Integrator for Large Time Steps. *IEEE Transactions on Visualization and Computer Graphics* 21, 10 (2015), 1103–1115. 3, 4
- [GW08] GEORGII J., WESTERMANN R.: Corotated Finite Elements Made Fast and Stable. In *Virtual Reality Interactions and Physical Simulations* (2008), Eurographics Association, pp. 11–19. 3
- [HH98] HOWLETT P., HEWITT W. T.: Mass-Spring Simulation using Adaptive Non-Active Points. *Computer Graphics Forum* 17, 3 (Aug. 1998), 345–353. 2
- [HLSO12] HECHT F., LEE Y. J., SHEWCHUK J. R., O'BRIEN J. F.: Updated Sparse Cholesky Factors for Corotational Elastodynamics. *ACM Transactions on Graphics* 31, 5 (2012), 1–13. 3
- [HS04] HAUTH M., STRASSER W.: Corotational Simulation of Deformable Solids. *Journal of WSCG* 12, 1 (2004), 137–145. 2
- [ISF07] IRVING G., SCHROEDER C., FEDKIW R.: Volume Conserving Finite Element Simulations of Deformable Models. *ACM Transactions on Graphics* 26, 3 (July 2007), 13. 10
- [ITF04] IRVING G., TERAN J., FEDKIW R.: Invertible Finite Elements For Robust Simulation of Large Deformation. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2004), Eurographics Association, pp. 131–140. 3, 8, 10
- [JJT06] JANOVSKÝ V., JANOVSKÁ D., TANABE K.: Computing the Analytic Singular Value Decomposition via a Pathfollowing. *Numerical Mathematics and Advanced Applications* 1, 1 (2006), 954–962. 6
- [JP99] JAMES D. L., PAI D. K.: ArtDefo: Accurate Real Time Deformable Objects. In *ACM Conference on Computer Graphics and Interactive Techniques* (1999), ACM, pp. 65–72. 2
- [KDBB17] KOSCHIER D., DEUL C., BRAND M., BENDER J.: An hp-adaptive discretization algorithm for signed distance field generation. *IEEE Transactions on Visualization and Computer Graphics* 23, 10 (2017), 2208–2221. 10
- [KJ11] KIM T., JAMES D. L.: Physics-based character skinning using multi-domain subspace deformations. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2011), ACM, pp. 63–72. 3
- [KMBG08] KAUFMANN P., MARTIN S., BOTSCH M., GROSS M.: Flexible Simulation of Deformable Models Using Discontinuous Galerkin FEM. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2008), pp. 105–115. 2
- [LBK17] LIU T., BOUAZIZ S., KAVAN L.: Quasi-Newton methods for real-time simulation of hyperelastic materials. *ACM Transactions on Graphics* 36, 3 (May 2017), 23:1–23:16. 3, 8, 9
- [MBCM16] MÜLLER M., BENDER J., CHENTANEZ N., MACKLIN M.: A robust method to extract the rotational part of deformations. In *ACM Motion in Games* (2016), ACM. 3, 8, 9, 10
- [MDM*02] MÜLLER M., DORSEY J., MCMILLAN L., JAGNOW R., CUTLER B.: Stable Real-Time Deformations. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2002), ACM, pp. 49–54. 2
- [MG04] MÜLLER M., GROSS M.: Interactive Virtual Materials. In *Graphics Interface* (2004), Canadian Human-Computer Communications Society, pp. 239–246. 2, 3
- [MHHR07] MÜLLER M., HEIDELBERGER B., HENNIX M., RATCLIFF J.: Position Based Dynamics. *The Visual Computer* 18, 2 (2007), 109–118. 3
- [MHTG05] MÜLLER M., HEIDELBERGER B., TESCHNER M., GROSS M.: Meshless Deformations Based on Shape Matching. *ACM Transactions on Graphics* 24, 3 (2005), 471–478. 3, 9
- [MMC16] MACKLIN M., MÜLLER M., CHENTANEZ N.: XPBD: Position-based Simulation of Compliant Constrained Dynamics. In *ACM Motion in Games* (2016), ACM, pp. 49–54. 3, 5, 8, 9
- [MR93] MEHRMANN V., RATH W.: Numerical methods for the computation of analytic singular value decompositions. *Electronic Transactions on Numerical Analysis* 1 (1993), 72–88. 6
- [MST*11] MCADAMS A., SELLE A., TAMSTORF R., TERAN J., SIFAKIS E.: *Computing the Singular Value Decomposition of 3×3 matrices with minimal branching and elementary floating point operations*. Tech. rep., University of California, Los Angeles, 2011. 9, 10

- [MTGG11] MARTIN S., THOMASZEWSKI B., GRINSPUN E., GROSS M.: Example-based elastic materials. *ACM Transactions on Graphics* 30, 4 (July 2011), 1. 4
- [MZS*11] MCADAMS A., ZHU Y., SELLE A., EMPEY M., TAMSTORF R., TERAN J., SIFAKIS E.: Efficient elasticity for character skinning with contact and collisions. *ACM Transactions on Graphics* 30, 4 (2011), 37:1–37:12. 3, 10
- [NMK*06] NEALEN A., MÜLLER M., KEISER R., BOXERMAN E., CARLSON M.: Physically Based Deformable Models in Computer Graphics. *Computer Graphics Forum* 25, 4 (2006), 809–836. 2
- [NOB16] NARAIN R., OVERBY M., BROWN G. E.: ADMM \supseteq Projective Dynamics: Fast Simulation of General Constitutive Models. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2016), pp. 1–8. 3, 4, 8, 9
- [NW06] NOCEDAL J., WRIGHT S. J.: *Numerical Optimization*, 2nd ed. Springer, New York, 2006. 5
- [PBH15] PAN Z., BAO H., HUANG J.: Subspace dynamic simulation using rotation-strain coordinates. *ACM Transactions on Graphics* 34, 6 (Oct. 2015), 242:1–242:12. 3
- [RJ07] RIVERS A. R., JAMES D. L.: FastLSM: Fast Lattice Shape Matching for Robust Real-Time Deformation. *ACM Transactions on Graphics* 26, 3 (July 2007), 82. 3
- [SB12] SIFAKIS E., BARBIČ J.: FEM Simulation of 3D Deformable Solids. In *ACM SIGGRAPH Courses* (2012), pp. 1–50. 4
- [SDGK17] SMITH B., DE GOES F., KIM T.: Stable neo-hookean flesh simulation. *ACM Transactions on Graphics* (2017). 4, 10
- [Sel10] SELIG J. M.: Exponential and cayley maps for dual quaternions. *Advances in applied Clifford algebras* 20, 3–4 (2010), 923–936. 7
- [SHST12] STOMAKHIN A., HOWES R., SCHROEDER C., TERAN J. M.: Energetically Consistent Invertible Elasticity. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2012). 4, 5
- [SSB13] SIN F. S., SCHROEDER D., BARBIČ J.: Vega: Non-Linear FEM Deformable Object Simulator. *Computer Graphics Forum* 32, 1 (2013), 36–48. 7, 8, 9
- [ST08] SCHMEDDING R., TESCHNER M.: Inversion handling for stable deformable modeling. *The Visual Computer* 24, 7–9 (2008), 625–633. 3
- [STC*12] SCHUMACHER C., THOMASZEWSKI B., COROS S., MARTIN S., SUMNER R., GROSS M.: Efficient simulation of example-based materials. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2012), Eurographics Association, pp. 1–8. 3
- [TBHF03] TERAN J., BLEMKER S., HING V. N. T., FEDKIW R.: Finite Volume Methods for the Simulation of Skeletal Muscle. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2003), Eurographics Association, pp. 68–74. 2
- [THM*03] TESCHNER M., HEIDELBERGER B., MÜLLER M., POMERANTES D., GROSS M. H.: Optimized spatial hashing for collision detection of deformable objects. In *Vision, Modeling and Visualization* (2003), vol. 3, pp. 47–54. 10
- [THMG04] TESCHNER M., HEIDELBERGER B., MÜLLER M., GROSS M.: A Versatile and Robust Model for Geometrically Complex Deformable Solids. In *Computer Graphics International* (2004), IEEE Computer Society, pp. 312–319. 2, 5
- [TK94] TAYLOR C. J., KRIEGMAN D. J.: Minimization on the lie group $so(3)$ and related manifolds. *Yale University* 16 (1994), 155. 6, 8
- [TNGF15] TOURNIER M., NESME M., GILLES B., FAURE F.: Stable Constrained Dynamics. *ACM Transactions on Graphics* (2015), 1–10. 3, 5
- [TPBF87] TERZOPOULOS D., PLATT J., BARR A., FLEISCHER K.: Elastically Deformable Models. In *ACM Conference on Computer Graphics and Interactive Techniques* (Aug. 1987), ACM, pp. 205–214. 2
- [TWS06] THOMASZEWSKI B., WACKER M., STRASSER W.: A Consistent Bending Model for Cloth Simulation with Corotational Subdivision Finite Elements. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2006). 2
- [XB16] XU H., BARBIČ J.: Pose-space subspace dynamics. *ACM Transactions on Graphics* 35, 4 (2016). 3