An efficient implicit constraint resolution scheme for interactive FE simulations

Ziqiu Zeng, Hadrien Courtecuisse

An efficient implicit constraint resolution scheme for interactive FE simulations

ZIQIU ZENG, University of Strasbourg, France
HADRIEN COURTECUISSE, University of Strasbourg, France

1 ABSTRACT

This paper presents a novel implicit scheme for the constraint resolution in real-time finite element simulations in the presence of contact and friction. Instead of using the standard motion correction scheme, we propose an iterative method where the constraint forces are corrected in Newton iterations. In this scheme, we are able to update the constraint directions recursively, providing more accurate contact and friction response. However, updating the constraint matrices leads to massive computation costs. To address the issue, we propose separating the constraint direction and geometrical mapping in the contact Jacobian matrix and reformulating the schur-complement of the system matrix. When combined with GPU-based parallelization, the reformulation provides a very efficient updating process for the constraint matrices in the recursive corrective motion scheme. Our method enables the possibility to handle the inconsistency of constraint directions at the beginning and the end of time steps. At the same time, the resolution process is kept as efficient as possible. We evaluate the performance of our fast-updating scheme in a contact simulation and compare it with the standard updating scheme.

CCS Concepts: • **Computing methodologies** → **Physical simulation**; **Parallel algorithms**.

Additional Key Words and Phrases: physics-based animation, finite element method, contact simulation

2 INTRODUCTION

Simulating multi-object systems always receives strong interest in the computer graphics community. Physics-based simulation is required to handle various materials containing rigid and soft solids. For soft materials, dealing with the elastic functions in solid deformation leads to challenging problems. The Finite Element Method (FEM) has been a gold-standard approach in this context [33]. On the other hand, coupling the simulation of different objects remains one of the other challenges in multi-object systems. The interactions in physics-based animations have been intensively studied in many works. Readers can find a general introduction of contact simulation in [2].

Simulating in real-time is essential for many tasks such as robot controlling [14], [4] and virtual surgeries [13]. In these tasks, the simulators are required to provide relevant information in real-time for users (or robots) to have efficient control and timely visual/haptic feedback to manipulators. While computing the contact and friction responses, the time-stepping scheme is suitable for real-time simulations. On this basis, constraint-based techniques have been prevalent for solving the complementarity problems as they guarantee non-interpenetration within a time step. Another essential issue in FE simulations is that accuracy often requires sufficiently detailed discretization of solids, leading to costly computations for contact responses. To address the issue, the methods using parallel architectures such as graphics processing units (GPUs) are developed, providing high computation speed while satisfying the requirement of accuracy.

In practice, the constraints are discretized and linearized in time steps. In many recent works, to simplify the solving process, such linearization is usually carried out once in each time step, making the constraint resolution in an explicit scheme for the time integration. In some contact scenarios, the constraints undergo large deviations from the initial evaluations at the beginning of time steps, leading to instability with large time steps. However, in FE simulations, updating the constraints through the constraint resolution is very difficult since it brings about intensive computation costs such as rebuilding the compliance matrix and updating the proximity information.

Authors' addresses: Ziqiu Zeng, University of Strasbourg, 4 Rue Blaise Pascal, Strasbourg, 67081, France, zengziqiu1995@gmail.com; Hadrien Courtecuisse, University of Strasbourg, 4 Rue Blaise Pascal, Strasbourg, 67081, France, hcourtecuisse@unistra.fr.

The present work is motivated by providing an implicit scheme for constraint resolution in interactive FE simulations. Such an implicit scheme can handle the inconsistency of constraint directions at the beginning and the end of time steps. We propose using a recursive corrective motion scheme to update the constraint directions in Newton iterations. The constraints could be re-linearized through the iterative method, involving the constraint resolution into an implicit scheme. To efficiently update the constraint matrices in the recursive scheme, we propose a reformulation of the contact Jacobian matrix, quickly rebuilding the compliance matrix and efficiently computing the boundary state after each correction. Besides the computational efficiency, our method is also straightforward since it only requires additional matrix multiplication operations, where we could find proven techniques on both the CPU and the GPU. Moreover, our method remains flexible to be compatible with different collision detection algorithms and resolution methods.

3 RELATED WORKS

3.1 Physics-based models

Compared to the fast and simple explicit schemes [10], implicit time integration scheme [6] are more suitable in interactive simulations as the external and internal forces are balanced at the end of the time steps. In multi-object systems, simulating soft solids usually leads to more computational issues than rigid solids due to high degrees of freedom (DOF) and nonlinear mechanics for deformation behavior. As a gold-standard method, finite element (FE) method is extended in a wide variety of works, such as the linear elastic models [9], the co-rotational formulation [17], and the hyperelastic or viscoelastic materials [26]. Although providing a good understanding of the deformation mechanisms, FE models remain complex and expensive.

On the other hand, discrete methods such as Position-Based Dynamics (PBD) [27] provide simple, fast, and robust simulations but suffer from handling the material properties [7]. The Projective Dynamics [8] gives a good trade-off between the performance of PBD and the accuracy of continuum mechanics, addressing the volume conservation problem in PBD. This work is extended in recent papers such as hyperelastic materials [22] and frictional contact [23]. Despite being computationally efficient, the Projective Dynamics remains to validate its capability of simulating realistic materials. In addition, meshless methods and Neural Networks are other strategies to model soft tissues in real-time [36].

3.2 Contact generation

The interactions in contact simulations may lead to discontinuity in the velocities in the mechanical motion. Generally, two schemes could be used to address this issue: The *event driven* scheme [5] gives a smooth and accurate generation for contacts while the time integration needs to be blocked and the number of instantaneous contacts is restricted. On the contrary, the *time stepping* scheme [34], [3] involves all potential contact during a fixed time step. The scheme has no limit on the generated contact and is compatible with both rigid and soft bodies [31] in implicit time integration.

Searching for the potential contact is commonly performed by the collision detection process. Fast and straightforward methods may use analytic shapes to approximate the surface of a solid. Using the bounding volume hierarchy (BVH), a detailed irregular shape may be represented as a combination of multiple simple shapes [35]. The intersections could be tested with efficient algorithms for common shapes such as sphere and box. On the other hand, the solid shape could also be represented as polygonal meshes on the surface. In a 3D problem, the surface typologies usually consist of triangle elements for rigid and soft solids modeled with FEM. Testing the intersections could be performed either by the typical way that searches the closest distances between the surface elements or by the advanced methods such as the image-based algorithm using layered depth image (LDI) [18]. [1] extends the work of LDI to GPU implementation and provides a simplification of contact constraints at arbitrary geometry-independent resolution.

The collision detection could be processed either by a discrete or continuous scheme. Discrete collision detection (DCD) algorithms search the potential contacts in each time step, usually at the beginning. DCD provides an approximate evaluation of constraint directions and often suffers from the resolution error, which leads to a slight separation or penetration at the end of time steps. In contrast, continuous collision detection (CCD) algorithms can accurately detect the first time of interaction between solids within a given time step. A survey of recent works on CCD combined with different detection approaches can be found in [28] Nevertheless, with respect to computational cost, CCD is significantly more expensive than DCD, especially in the scenarios with detail shapes for meshes.

3.3 Constraint resolution

The constraint-based methods using Lagrange multipliers ([21], [30]) solve the contact problem in a coupled way, providing accurate and robust solutions in contact mechanics for large time steps, where interpenetration is entirely eliminated at the end of time steps. The Lagrange multipliers methods are commonly formulated as a linear complementarity problem (LCP). Different numerical methods can be used to address the LCPs in physics-based animations [16]. Direct methods such as pivoting methods give exact resolution, but they are not computationally efficient. In contrast, iterative methods have been more widely applied in large-scale simulations, especially for those required to perform real-time computations. Being simple to implement, projected Gauss-Seidel (PGS) ([15], [11], [25]) can handle the friction response with the Coulomb's friction cone combined in the LCP formulation. [24] proposes using a Newton method to solve the non-smooth functions that are reformulated from complementarity problems.

To couple contact forces in the resolution and to formulate a system in the *constraint space*, one efficient solution is to formulate the schur-complement of the augmented system, resulting in a compliance matrix (also called *delasus operator*). Nevertheless, as discussed in [2], the computation of schur-complement tends to be costly when dealing with soft-body since it implies solving a linear system with multiple right-hand sides. Many methods have been proposed to efficiently process the schur-complement in interactive FE simulations, such as the compliance warping technique [31], the CPU-based parallelization [32] and the incomplete factorization [29] in *Pardiso solver project*, the asynchronous precondition-based method [12] and the GPU-based parallelization [13], and the updating Cholesky factor in consecutive time steps [19] [20].

In different methods for constraint resolution, a common strategy is to linearize the constraints after the collision detection. The linearization is carried out once in each time step, while the constraint directions may change from the beginning to the end. The current paper aims to provide a constraint resolution scheme where the constraints could be re-linearized in a recursive method, which is able to handle the change of constraint directions and to bridge the gap between the DCD and CCD. A fast-updating strategy is proposed to guarantee the efficiency of the recursive scheme, using a straightforward GPU-based implementation.

4 BACKGROUND

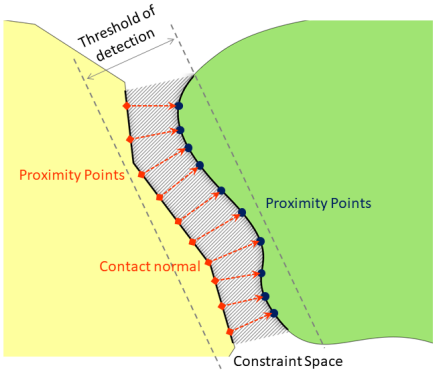
4.1 Implicit time integration

For each independent object in a multi-object system, a general description for the physical behavior can be expressed through the Newton's second law:

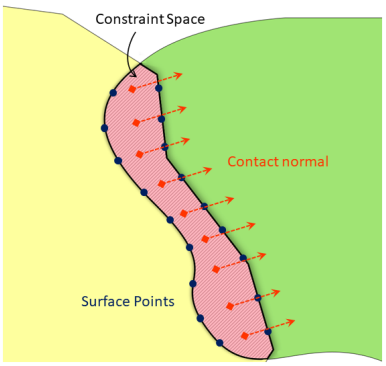
$$\mathbf{M}\ddot{\mathbf{q}} = \boldsymbol{p} - \mathcal{F}(\mathbf{q}, \dot{\mathbf{q}}) + \boldsymbol{c} \tag{1}$$

where the derivative of the velocity $\ddot{\mathbf{q}}$ is integrated with the mass matrix $\mathbf{M}$, the external forces $\boldsymbol{p}$, the internal forces $\mathcal{F}(\mathbf{q}, \dot{\mathbf{q}})$, and the constraint forces $\boldsymbol{c}$. To have a balance between different forces at the end of time steps, we choose to integrate the time step t with a backward Euler scheme:

$$\dot{\mathbf{q}}_{t+h} = \dot{\mathbf{q}}_t + h\ddot{\mathbf{q}}_{t+h} \quad \mathbf{q}_{t+h} = \mathbf{q}_t + h\dot{\mathbf{q}}_{t+h} \tag{2}$$



(a) For typical method of discrete collision detection, a threshold is usually used to test if the proximity points are close enough. The evaluation of potential contact normal depends directly on the surface elements (positions and normals) at the beginning of time steps.



(b) For the image-based method of collision detection, the evaluation of contacts is usually based on penetrated volume. The evaluation of constraint normal depends on the boundary of the interpenetration area, which is computed by the positions of surface elements.

Fig. 1. Constraint linearization with different types of collision detection: To simplify the solving process, collision detection is performed providing a set of discretized constraints between both objects (red arrows). Each contact constraint involves the proximity points and a direction of contact normal, which is used to apply the force to separate objects. According to different collision detection algorithms, the contact normal is dependent, directly or indirectly, upon the position of proximity points (red and black points on the surface of contacting bodies).

where h is the length of time interval $[t, t+h]$.

For rigid solids, as there are no internal forces, the dynamic equation can be simplified as:

$$\mathbf{M}\Delta\dot{\mathbf{q}}_{t+h} = h\boldsymbol{p} + h\boldsymbol{c} \tag{3}$$

with $\Delta\dot{\mathbf{q}} = h\ddot{\mathbf{q}}$ is the unknown vector to be solved.

For soft solids, the non-linear function of internal forces is linearized with a first-order Taylor expansion:

$$\mathcal{F}(\mathbf{q}_{t+h}, \dot{\mathbf{q}}_{t+h}) = \mathcal{F}(\mathbf{q}_t, \dot{\mathbf{q}}_t) + \frac{\partial\mathcal{F}(\mathbf{q}, \dot{\mathbf{q}})}{\partial\mathbf{q}} h\dot{\mathbf{q}}_{t+h} + \frac{\partial\mathcal{F}(\mathbf{q}, \dot{\mathbf{q}})}{\partial\dot{\mathbf{q}}} h\ddot{\mathbf{q}}_{t+h} \tag{4}$$

In the practice of finite-element simulations, the partial derivative terms are expressed as matrices: $\frac{\partial\mathcal{F}}{\partial\dot{\mathbf{q}}}$ at $(\mathbf{q}_t, \dot{\mathbf{q}}_t)$ as a damping matrix $\mathbf{B}$, and $\frac{\partial\mathcal{F}}{\partial\mathbf{q}}$ at $(\mathbf{q}_t, \dot{\mathbf{q}}_t)$ as a stiffness matrix $\mathbf{K}$. The dynamic equation for soft solids finally results in a second order differential equation:

$$\underbrace{\left[\mathbf{M} + h\mathbf{B} + h^2\mathbf{K}\right]}_{\mathbf{A}} \Delta\dot{\mathbf{q}}_{t+h} = \underbrace{\left(h\boldsymbol{p}_t - h\boldsymbol{f}_t\right) - h^2\mathbf{K}\dot{\mathbf{q}}_t + h\boldsymbol{c}}_{\mathbf{b}} \tag{5}$$

with $\boldsymbol{f}_t = \mathcal{F}(\mathbf{q}_t, \dot{\mathbf{q}}_t)$. For both rigid and soft solids, we have a common formulation of a linear system $\mathbf{A}\Delta\dot{\mathbf{q}} = \mathbf{b} + h\boldsymbol{c}$ to be solved.

4.2 Contact generation

As illustrated in Figure 1, a contact constraint involves a contact normal and a pair of proximity points that is referred to the objects in touch. The contact normal defines the direction of a non-interpenetration force to separate the objects. The pair of proximity points $\mathbf{p}$ represent the current state of the surfaces of the objects. In different methods, $\mathbf{p}$ can be directly the points on the object surface, or the representative points using a geometric mapping to transform between $\mathbf{q}$ and $\mathbf{p}$. Considering the former as an identity mapping, we have a uniformed relationship:

$$\mathbf{p} = \mathcal{G}(\mathbf{q}) \tag{6}$$

where the function $\mathcal{G}$ is the geometric mapping from the positions of *mechanical DOFs* $\mathbf{q}$ to the proximity positions $\mathbf{p}$.

4.3 Contact and friction model

The contacts between two objects are modeled as constraints, which are discretized and linearized through the collision detection process. As illustrated in Figure 1, to prevent interpretations, the distance between two solids 1 and 2 can be formulated as a gap function:

$$\boldsymbol{\delta}_n(\mathbf{q}) = \vec{\mathbf{n}}[\mathbf{p}_1 - \mathbf{p}_2] = \vec{\mathbf{n}}[\mathcal{G}_1(\mathbf{q}) - \mathcal{G}_2(\mathbf{q})] = \mathcal{H}_{n1}(\mathbf{q}) - \mathcal{H}_{n2}(\mathbf{q}) \tag{7}$$

where the interpretation $\delta_n(\mathbf{q})$ is the distance measurement between the proximity positions $\mathcal{G}(\mathbf{q})$ projected on the contact normal $\vec{\mathbf{n}}$. The contact normal is the direction of a force $\mathbf{f}_n$ that separates the interpenetrating solids. The geometrical mapping function $\mathcal{G}(\mathbf{q})$ describes the mapping from the *mechanical DOFs* space to the proximity space. Integrating $\vec{\mathbf{n}}$ into $\mathcal{G}(\mathbf{q})$ results in a mapping function $\mathcal{H}_n(\mathbf{q})$. Signorini's law presents the complementarity relationship along the constraint direction $\vec{\mathbf{n}}$ for each potential contact:

$$0 \leq \boldsymbol{\delta}_n(\mathbf{q}) \perp \boldsymbol{\lambda}_n \geq 0 \tag{8}$$

where the multiplier $\boldsymbol{\lambda}_n$ is the magnitude of the contact force $\mathbf{f}_n$ along $\vec{\mathbf{n}}$ as the constraint direction has been normalized.

Equation (8) only guarantees to separate the contacting objects. To model the friction response, the frictional constraints should be added along with the contact normal. In a 3D problem, a common frictional model complements each contact normal with two tangential directions $\vec{\mathbf{f}}$. When a contacting is validated ($\boldsymbol{\lambda}_n \geq 0$), following Coulomb's friction law, we have:

$$0 \leq \boldsymbol{\delta}_f(\mathbf{q}) \perp \mu\boldsymbol{\lambda}_n - \boldsymbol{\lambda}_f \geq 0 \tag{9}$$

where $\boldsymbol{\delta}_f(\mathbf{q})$ is the the interpenetration distance measurement projected on the tangential directions $\vec{\mathbf{f}}$, μ is the coefficient of friction, and $\boldsymbol{\lambda}_f$ is the value of frictional force along $\vec{\mathbf{f}}$. The friction model describes two states for the kinematic behavior: the contacting objects are stuck ($\boldsymbol{\delta}_f = 0$) while $\boldsymbol{\lambda}_f \leq \mu\boldsymbol{\lambda}_n$, and are slipping while $\boldsymbol{\lambda}_f$ achieves the maximum value $\mu\boldsymbol{\lambda}_n$. In addition, $\boldsymbol{\delta}_f(\mathbf{q})$ has a similar gap function to Equation (7):

$$\boldsymbol{\delta}_f(\mathbf{q}) = \vec{\mathbf{f}}[\mathbf{p}_1 - \mathbf{p}_2] = \vec{\mathbf{f}}[\mathcal{G}_1(\mathbf{q}) - \mathcal{G}_2(\mathbf{q})] = \mathcal{H}_{f1}(\mathbf{q}) - \mathcal{H}_{f2}(\mathbf{q}_2) \tag{10}$$

The governing equations including the complementarity relationships result in the following non-linear system:

$$\begin{cases} \mathbf{A}_1 \Delta\dot{\mathbf{q}}_1 = \mathbf{b}_1 + h\boldsymbol{c}_1 & \text{(11a)} \\ \mathbf{A}_2 \Delta\dot{\mathbf{q}}_2 = \mathbf{b}_2 + h\boldsymbol{c}_2 & \text{(11b)} \\ \boldsymbol{\delta}_n(\mathbf{q}) = \mathcal{H}_{n1}(\mathbf{q}) - \mathcal{H}_{n2}(\mathbf{q}) & \text{(11c)} \\ \boldsymbol{\delta}_f(\mathbf{q}) = \mathcal{H}_{f1}(\mathbf{q}) - \mathcal{H}_{f2}(\mathbf{q}) & \text{(11d)} \\ 0 \le \boldsymbol{\delta}_n(\mathbf{q}) \perp \boldsymbol{\lambda}_n \ge 0 & \text{(11e)} \\ 0 \le \boldsymbol{\delta}_f(\mathbf{q}) \perp \mu\boldsymbol{\lambda}_n - \boldsymbol{\lambda}_f \ge 0 & \text{(11f)} \end{cases}$$

Since the contact normal constraint along $\vec{\mathbf{n}}$ and the frictional constraint along $\vec{\mathbf{f}}$ have the same mapping function in the gap functions, in practice, the constraints are grouped as constraint sets, where each one of them involves a normal constraint $\vec{\mathbf{n}}$ and two tangential constraints $\vec{\mathbf{f}}$. Consequently, the functions and vectors can be grouped: $\mathcal{H}_n(\mathbf{q})$ and $\mathcal{H}_f(\mathbf{q})$ are grouped as $\mathcal{H}(\mathbf{q})$; $\boldsymbol{\delta}_n(\mathbf{q})$ and $\boldsymbol{\delta}_f(\mathbf{q})$ are grouped as $\boldsymbol{\delta}(\mathbf{q})$; $\boldsymbol{\lambda}_n$ and $\boldsymbol{\lambda}_f$ are grouped as $\boldsymbol{\lambda}$.

4.4 Constraint linearization

When dealing with the dynamic equation (5), a common strategy is to separate the resolution into two motion processes:

$$\Delta\dot{\mathbf{q}}_{t+h} = \underbrace{\mathbf{A}^{-1}\mathbf{b}}_{\Delta\dot{\mathbf{q}}_{\text{free}}} + \underbrace{h\mathbf{A}^{-1}\boldsymbol{c}}_{\Delta\dot{\mathbf{q}}_{\text{cor}}} \tag{12}$$

where the first integration called *free motion* computes the temporary motion $\Delta\dot{\mathbf{q}}_{\text{free}}$, which mathematically corresponds to physics dynamics without considering the constraints of contact and friction. The second part integrates a *corrective motion* to obtain the motion $\Delta\dot{\mathbf{q}}_{t+h}$ at the end of time steps.

With the implicit integration (Equation (2)), we have the position integration from the intermediate state of *free motion*:

$$\mathbf{q}_{t+h} = \mathbf{q}_{\text{free}} + h\Delta\dot{\mathbf{q}}_{\text{cor}} \tag{13}$$

where the *corrective motion* could be obtained after the unknown constraint forces $\boldsymbol{c}$ are solved: $\Delta\dot{\mathbf{q}}_{\text{cor}} = \mathbf{A}^{-1}\boldsymbol{c}$. The mapping functions $\mathcal{H}(\mathbf{q})$ are linearized following a first-order Taylor expansion:

$$\mathcal{H}(\mathbf{q}_{t+h}) \approx \mathcal{H}(\mathbf{q}_{\text{free}}) + h\frac{\partial\mathcal{H}(\mathbf{q})}{\partial\mathbf{q}}\Delta\dot{\mathbf{q}}_{\text{cor}} \tag{14}$$

where the Jacobian function $\frac{\partial\mathcal{H}(\mathbf{q})}{\partial\mathbf{q}}$ at t the beginning of time steps is usually linearized as the contact Jacobian matrix $\mathbf{J}$ that impose the constraints in the mechanical motion of the object.

On the one hand, the contact Jacobian maps *mechanical DOFs* velocities into the velocities for the gap function, allowing to rewrite the gap function in Equation (7) and (10) as:

$$\begin{aligned} \boldsymbol{\delta}^{t+h} &= \mathcal{H}_1(\mathbf{q}_1^{t+h}) - \mathcal{H}_2(\mathbf{q}_2^{t+h}) \\ &\approx \underbrace{\mathcal{H}_1(\mathbf{q}_1^{\text{free}}) - \mathcal{H}_2(\mathbf{q}_2^{\text{free}})}_{\boldsymbol{\delta}^{\text{free}}} + h(\mathbf{J}_1\Delta\dot{\mathbf{q}}_1^{\text{cor}} - \mathbf{J}_2\Delta\dot{\mathbf{q}}_2^{\text{cor}}) \end{aligned} \tag{15}$$

where $\boldsymbol{\delta}^{t+h}$ and $\boldsymbol{\delta}^{\text{free}}$ respectively measure the interpenetration at the end of time steps and the state of *free motion*

On the other hand, the contact Jacobian apply the constraint forces into the mechanical motion space:

$$\boldsymbol{c}_1 = \mathbf{J}_1{}^{\mathrm{T}}\boldsymbol{\lambda} \quad \boldsymbol{c}_2 = -\mathbf{J}_2{}^{\mathrm{T}}\boldsymbol{\lambda} \tag{16}$$

where the forces are applied for the two objects in opposite directions ($\vec{\mathbf{n}}_1 = -\vec{\mathbf{n}}_2$, $\vec{\mathbf{f}}_1 = -\vec{\mathbf{f}}_2$). Replace the unknown $\Delta\dot{\mathbf{q}}^{\text{cor}}$ in Equation (15) by Equation (12) and (16), and we obtain a linear system:

$$\boldsymbol{\delta}^{t+h} = \boldsymbol{\delta}^{\text{free}} + h^2 \underbrace{[\mathbf{J}_1\mathbf{A}_1^{-1}\mathbf{J}_1{}^{\mathrm{T}} + \mathbf{J}_2\mathbf{A}_2^{-1}\mathbf{J}_2{}^{\mathrm{T}}]}_{\mathbf{W}} \boldsymbol{\lambda} \tag{17}$$

where $\mathbf{W} \in \mathbb{R}^{c\times c}$ (also called *compliance matrix* or *delassus operator* in constrained dynamics) is the schur-complement of the system matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, coupling the constraint forces through the motion space n.

4.5 Constraint resolution

With Equations (11) and (17), we have The governing equations (11) with the schur-complement result in a complementarity system in a constraint space $\mathbb{R}^c$:

$$\begin{cases} \boldsymbol{\delta}^{t+h} = \boldsymbol{\delta}^{\text{free}} + h^2\mathbf{W}\boldsymbol{\lambda} & \text{(18a)} \\ 0 \le \boldsymbol{\delta}_n^{t+h} \perp \boldsymbol{\lambda}_n \ge 0 & \text{(18b)} \\ 0 \le \boldsymbol{\delta}_f^{t+h} \perp \mu\boldsymbol{\lambda}_n - \boldsymbol{\lambda}_f \ge 0 & \text{(18c)} \end{cases}$$

The non-linear system in Equation (18) is solved by a projected Gauss-Seidel algorithm [15] during the successive iterations (i):

$$\delta_\alpha - \mathbf{W}_{\alpha\alpha}\boldsymbol{\lambda}_\alpha^{(i)} = \sum_{\beta=1}^{\alpha-1} \mathbf{W}_{\alpha\beta}\boldsymbol{\lambda}_\beta^{(i)} + \sum_{\beta=\alpha+1}^{c} \mathbf{W}_{\alpha\beta}\boldsymbol{\lambda}_\beta^{(i-1)} + \delta_\alpha^{\text{free}} \tag{19}$$

where $\mathbf{W}_{\alpha\beta}$ is a local matrix of $\mathbf{W}$ that couples the contact α and β. The complementarity problem for each contact group α is solved in the local solution following Signorini's law for the unilateral contact response and Coulomb's law for the frictional response.

Once the $\boldsymbol{\lambda}$ is solved, a *corrective motion* is processed to integrate the final motion $\Delta\dot{\mathbf{q}}_{t+h}$:

$$\begin{aligned} \Delta\dot{\mathbf{q}}_1^{t+h} &= \Delta\dot{\mathbf{q}}_1^{\text{free}} + h\mathbf{A}_1^{-1}\mathbf{J}_1{}^{\mathrm{T}}\boldsymbol{\lambda} \\ \Delta\dot{\mathbf{q}}_2^{t+h} &= \Delta\dot{\mathbf{q}}_2^{\text{free}} - h\mathbf{A}_2^{-1}\mathbf{J}_2{}^{\mathrm{T}}\boldsymbol{\lambda} \end{aligned} \tag{20}$$

5 RECURSIVE CORRECTIVE MOTION

For both discrete and continuous collision detection algorithms, the constraints will be linearized only once each time step. Nevertheless, it cannot be guaranteed that the initial evaluation of the constraint directions is always consistent with the state at the end of the time step. Figure 2 gives an example: in a grasping scenario, while acting a moving/rotation, the constraint directions undergo large deviations from the initial guess.

To address this problem, we propose a recursive motion correction scheme: Instead of using a single corrective motion as in Equation (12), a iterative correction is applied:

$$\Delta\dot{\mathbf{q}}^{t+h} = \Delta\dot{\mathbf{q}}^{\text{free}} + (\Delta\dot{\mathbf{q}}_1^{\text{cor}} + \Delta\dot{\mathbf{q}}_2^{\text{cor}} + \cdots + \Delta\dot{\mathbf{q}}_n^{\text{cor}}) \tag{21}$$

where n interactions are performed, and the corrective motion of a given iteration k is computed as following:

$$\Delta\dot{\mathbf{q}}_k^{\text{cor}} = h\mathbf{A}_k^{-1}\mathbf{J}_k{}^{\mathrm{T}}\boldsymbol{\lambda}_k \tag{22}$$

where $\mathbf{A}_k$ and $\mathbf{J}_k$ are reevaluated in each iteration according to the current mechanical state ($\mathbf{q}_k$). For the integration between two successive iterations k and $k+1$ in Equation (21), we have:

$$\mathbf{q}_{k+1} - \mathbf{q}_k = (\mathbf{q}^{\text{free}} + h\Delta\dot{\mathbf{q}}_{k+1}) - (\mathbf{q}^{\text{free}} + h\Delta\dot{\mathbf{q}}_k) = h\Delta\dot{\mathbf{q}}_{k+1}^{\text{cor}} \tag{23}$$

ALGORITHM 1: Standard scheme of simulation loop

while *simulation* **do**
  collision_detection($\mathbf{q}^t$) ;
  constraint_linearization($\mathbf{p}^t$) ;
  foreach *object* **do**
    Assemble $\mathbf{A}$, $\mathbf{b}$, $\mathbf{J}$;
    $\Delta\dot{\mathbf{q}}^{\text{free}} = \mathbf{A}^{-1}\mathbf{b}$;
    $\mathbf{q}^{\text{free}} = \mathbf{q}^h + h(\dot{\mathbf{q}}^h + \Delta\dot{\mathbf{q}}^{\text{free}})$;
  end
  Compute $\boldsymbol{\delta}^{\text{free}}$ according to $\mathbf{q}^{\text{free}}$;
  $\mathbf{W} = \sum \mathbf{J}\mathbf{A}^{-1}\mathbf{J}^{\mathrm{T}}$;
  foreach $i \in$ *PGS_iterations* **do**
    foreach $j \in$ *constraint_groups* **do**
      $\boldsymbol{\delta}^c = \boldsymbol{\delta}^{\text{free}} + \mathbf{W}\boldsymbol{\lambda}^i$;
      $\boldsymbol{\lambda}^i_j = solve(\boldsymbol{\lambda}^i, \boldsymbol{\delta}^c, \mathbf{W})$;
    end
    $\epsilon = \frac{|\boldsymbol{\lambda}^i - \boldsymbol{\lambda}^{i-1}|}{|\boldsymbol{\lambda}^i|}$;
    if $\epsilon \leq$ *PGS_error* **then**
      break ;
    end
  end
  foreach *object* **do**
    $\Delta\dot{\mathbf{q}}^{t+h} = \Delta\dot{\mathbf{q}}^{\text{free}} + h\mathbf{A}^{-1}\mathbf{J}^{\mathrm{T}}\boldsymbol{\lambda}$;
    $\dot{\mathbf{q}}^{t+h} = \dot{\mathbf{q}}^h + \Delta\dot{\mathbf{q}}^{t+h}$;
    $\mathbf{q}^{t+h} = \mathbf{q}^h + h\dot{\mathbf{q}}^{t+h}$;
  end
end

Such a recursive scheme is a Newton-Raphson method that provides a more accurate correction. However, performing such a scheme requires repeating the compliance assembly, the constraint resolution, the corrective motion, and the time integration in recursive iterations. These additional processes multiply the computational cost, making the system resolution very inefficient.

ALGORITHM 2: Recursive correction scheme with standard approach to update the constraint matrices

foreach $k \in$ *Newton_iterations* **do**
1 Linearize contact constraints with the proximity positions $\mathbf{p}_k$, then update $\mathbf{J}_k$ with the constraint directions.
2 Update compliance matrix: $\mathbf{W}_k = \sum \mathbf{J}_k \mathbf{A}_k^{-1} {\mathbf{J}_k}^{\mathrm{T}}$
3 Compute constraint forces within local PGS: $\boldsymbol{\lambda} = \mathbf{W}^{-1}\boldsymbol{\delta}$
4 Compute corrective motion: $\Delta\dot{\mathbf{q}}_k^{\text{cor}} = h\mathbf{A}_k^{-1}{\mathbf{J}_k}^{\mathrm{T}}\boldsymbol{\lambda}_k$
5 Integrate corrective motion: $\dot{\mathbf{q}}_k = \dot{\mathbf{q}}_{k-1} + \Delta\dot{\mathbf{q}}_k^{\text{cor}}$, $\mathbf{q}_k = \mathbf{q}_0 + h\dot{\mathbf{q}}_k$
6 Update the proximity positions: $\mathbf{p}_k = \mathcal{G}(\mathbf{q}_k)$
end

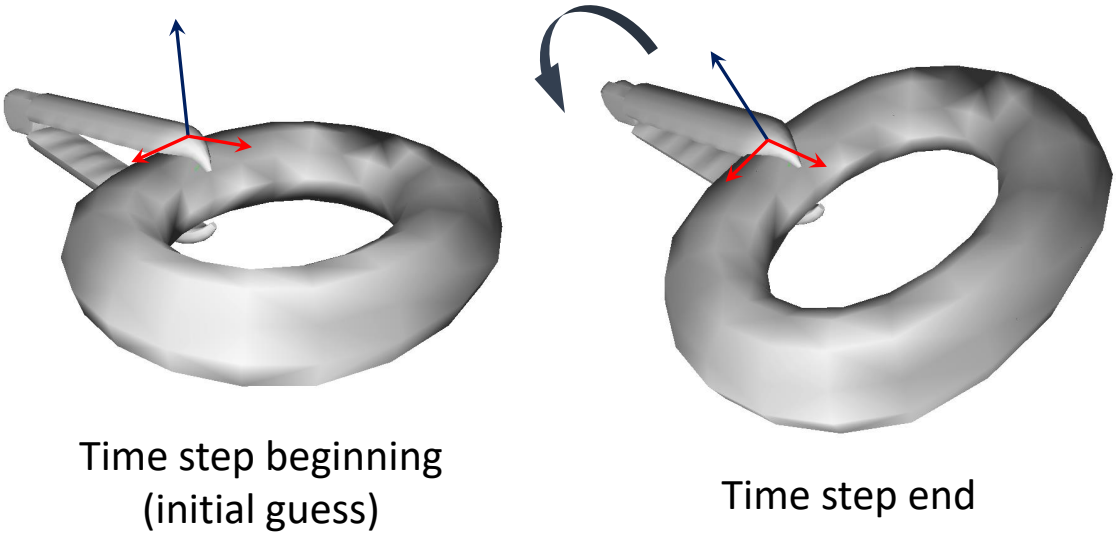


Fig. 2. Grasping a solid while applying a fast rotation will case a large deviation of constraint directions from the beginning to the end of a time step. Such a deviation may cause inaccurate contact responses, instability of contact forces, and eventually failed simulations.

6 UPDATING CONSTRAINT MATRICES

In this section, we propose a strategy that is able to efficiently process the iterations in Equation (21). In the recursive motion correction scheme (Equation (21)), the system matrix $\mathbf{A}$ is actually the mass matrix $\mathbf{M}$ for a rigid object. Therefore $\mathbf{A}_k = \mathbf{A}_0 = \mathbf{M}$, $k \in \mathbb{R}^n$. On the other hand, the corrective motion usually causes small deformation in motion corrections. Relying on this hypothesis, we have an approximation for $\mathbf{A}_k$ in the iterations: $\mathbf{A}_k = \mathbf{A}_0$, $k \in \mathbb{R}^n$. Following Equation (21) and (22), we have:

$$\Delta\dot{\mathbf{q}}^{t+h} = \Delta\dot{\mathbf{q}}^{\text{free}} + h\mathbf{A}^{-1}(\mathbf{J}_0{}^{\mathrm{T}}\boldsymbol{\lambda}_0 + \mathbf{J}_1{}^{\mathrm{T}}\boldsymbol{\lambda}_1 + \cdot\mathbf{J}_n{}^{\mathrm{T}}\boldsymbol{\lambda}_n) \tag{24}$$

We recall the definition of the contact Jacobian $\mathbf{J}$. For a given constraint with a direction $\vec{\mathbf{c}}$, the contribution in $\mathbf{J}$ is expressed as:

$$\mathbf{J}_{(\vec{\mathbf{c}})} = \frac{\partial\mathcal{H}(\mathbf{q})}{\partial\mathbf{q}} = \frac{\partial(\vec{\mathbf{c}}\mathcal{G}(\mathbf{q}))}{\partial\mathbf{q}} = \vec{\mathbf{c}}\frac{\partial\mathcal{G}(\mathbf{q})}{\partial\mathbf{q}} \tag{25}$$

as in practice, the constraint direction $\vec{\mathbf{c}}$ is independent of the mechanical state after a constraint linearization.

With Equation (25), we propose to assembly a Jacobian matrix $\mathbf{H}$ for the Jacobian of the geometric mapping $\frac{\partial\mathcal{G}(\mathbf{q})}{\partial\mathbf{q}}$, and a block-diagonal matrix $\mathbf{C}$ to store the constraint directions:

$$\mathbf{C} = \begin{bmatrix} \vec{\mathbf{c}}_1 & & & \\ & \vec{\mathbf{c}}_2 & & \\ & & \ddots & \\ & & & \vec{\mathbf{c}}_c \end{bmatrix} \tag{26}$$

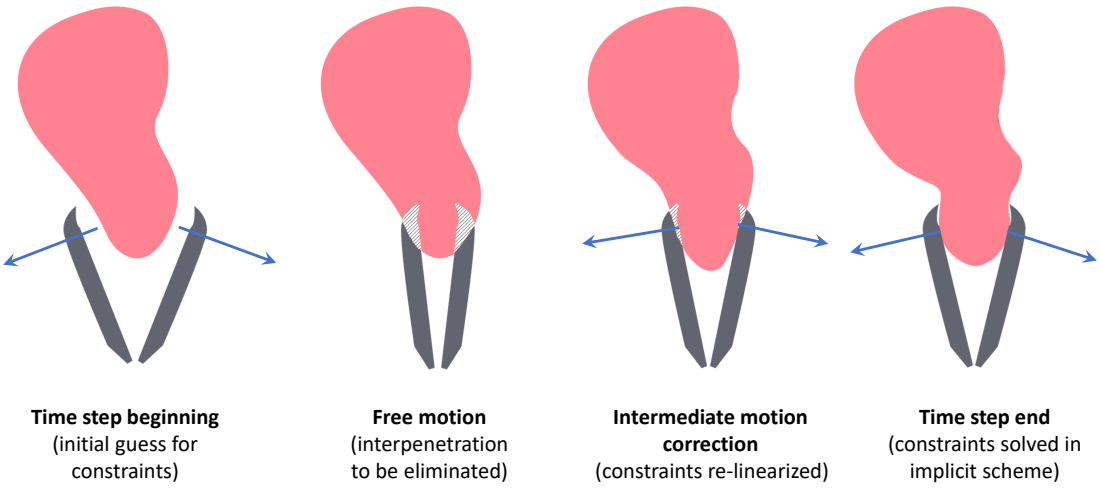


Fig. 3. Our method solves the constraints in an implicit scheme: The constraints are linearized at the beginning of each time step using discrete collision detection. Contact and friction responses are computed to eliminate the interpenetration between solids in the free motion. The constraint resolution is performed in a recursive corrective motion scheme, where contact forces are computed in each iteration by a local solver. Using the contact forces, the boundary conditions of colliding solids and the constraint matrices are updated in the next iteration to compute new motion corrections. In this way, the constraint directions are re-linearized recursively until the interpenetration is eliminated.

where $\vec{\mathbf{c}} = \begin{bmatrix}\vec{\mathbf{n}} & \vec{\mathbf{f}}_{(1)} & \vec{\mathbf{f}}_{(2)}\end{bmatrix}$ groups the normal and frictional constraints on a shared proximity point. The relation between the two matrices can be actually expressed by a matrix-matrix multiplication:

$$\mathbf{J} = \mathbf{C}\mathbf{H} \tag{27}$$

With Equation (27), a standard compliance assembly (Equation (17)) can be then reformulated as:

$$\mathbf{W} = \sum \mathbf{J}\mathbf{A}^{-1}\mathbf{J}^{\mathrm{T}} = \sum \mathbf{C}\underbrace{\mathbf{H}\mathbf{A}^{-1}\mathbf{H}^{\mathrm{T}}}_{\mathbf{W}_{\mathbf{H}}}\mathbf{C}^{\mathrm{T}} \tag{28}$$

where $\mathbf{W}$ can be built with $\mathbf{J}$ and $\mathbf{W}_{\mathbf{H}}$. The geometrical mapping $\mathcal{G}(\mathbf{q})$ usually undergoes a slight change in the recursive corrective motion scheme. Based on this hypothesis, $\mathbf{H}$ and $\mathbf{W}_{\mathbf{H}}$ are considered invariant during a time step.

Figure 1 illustrates that the linearization of contact constraints will based on the proximity positions $\mathbf{p}$. With a Taylor expansion, $\mathbf{p}$ in the recursive motion correction scheme can be as follows:

$$\mathbf{p}_{k+1} = \mathcal{G}(\mathbf{q}_{k+1}) \approx \mathcal{G}(\mathbf{q}_k) + \frac{\partial \mathcal{G}(\mathbf{q})}{\partial \mathbf{q}}(\mathbf{q}_{k+1} - \mathbf{q}_k) \tag{29}$$

where k and $k+1$ represents two successive iterations in the recursive motion correction scheme. With Equation (23) and $\frac{\partial \mathcal{G}(\mathbf{q})}{\partial \mathbf{q}}$ linearized as $\mathbf{H}$, we continue the development in Equation (29):

$$\mathbf{p}_{k+1} \approx \mathcal{G}(\mathbf{q}_k) + \frac{\partial \mathcal{G}(\mathbf{q})}{\partial \mathbf{q}}(\mathbf{q}_{k+1} - \mathbf{q}_k) = \mathbf{p}_k + h\mathbf{H}\Delta\dot{\mathbf{q}}^{\mathrm{cor}}_{k+1} \tag{30}$$

Combining Equation (22), (30), (27) and (28), we have:

$$\begin{aligned}\mathbf{p}_{k+1} &\approx \mathbf{p}_k + h\mathbf{H}\Delta\dot{\mathbf{q}}^{\text{cor}}_{k+1} \\ &= \mathbf{p}_k + h\mathbf{H}(h\mathbf{A}^{-1}{\mathbf{J}_k}^{\mathrm{T}}\boldsymbol{\lambda}_k) \\ &= \mathbf{p}_k + h^2\mathbf{H}\mathbf{A}^{-1}\mathbf{H}^{\mathrm{T}}{\mathbf{C}_k}^{\mathrm{T}}\boldsymbol{\lambda}_k \\ &= \mathbf{p}_k + h^2\mathbf{W}_{\mathbf{H}}{\mathbf{C}_k}^{\mathrm{T}}\boldsymbol{\lambda}_k\end{aligned} \tag{31}$$

Now with Equation (28) and (31), once $\mathbf{W}_{\mathbf{H}}$ is built, a recursive scheme can be performed in Algorithm 3.

ALGORITHM 3: Recursive correction scheme with fast update of the constraint matrices

foreach $k \in Newton\_iterations$ **do**

1. Linearize contact constraints with the proximity positions $\mathbf{p}_k$, then update $\mathbf{C}_k$ with the constraint directions.
2. Update compliance matrix: $\mathbf{W}_k = \sum \mathbf{C}_k\mathbf{W}_{\mathbf{H}}{\mathbf{C}_k}^{\mathrm{T}}$
3. Compute constraint forces within local PGS: $\boldsymbol{\lambda}_k = \mathbf{W}_k^{-1}\boldsymbol{\delta}$
4. Update the proximity positions: $\mathbf{p}_k = \mathbf{p}_{k-1} + h^2\mathbf{W}_{\mathbf{H}}{\mathbf{C}_k}^{\mathrm{T}}\boldsymbol{\lambda}_k$

end

Compared with the standard update strategy, the computation is greatly simplified in the new scheme. Instead of inverting the system matrix $\mathbf{A}$, the new scheme only needs matrix-matrix multiplications to update the compliance matrix $\mathbf{W}$, and a matrix-vector multiplication to update the boundary state (proximity positions $\mathbf{p}$). As the basic operations in linear algebraic, the matrix multiplications have highly efficient implementations, especially with parallelization on the GPU.

7 RESULTS AND CONCLUSION

In this section, we evaluate the computation cost of the method presented in Section 6 that provides a fast update of constraint matrices in the recursive correction scheme proposed in Section 5. The simulation tests are conducted in the open-source SOFA framework with a CPU AMD@ Ryzen 9 5950X 16-Core at 3.40GHz with 32GB RAM and a GPU GeForce RTX 3080 10GB.

In order to have an efficient resolution, we use the precondition-based technique in [13] to assemble the compliance matrix $\mathbf{W}$. As illustrated in Algorithm 3, a recursive corrective motion scheme helps to improve

DOFs	Cst.	Method	Build $\mathbf{W}_{\mathbf{H}}$	Rebuild $\mathbf{W}$	PGS	Corr.	Newton ite.	Total	Update
18003	308.19	standard		19.89 ms	3.76 ms	0.82 ms	24.47 ms	122.39 ms	84.60 %
		fast	19.26 ms	0.59 ms		0.08 ms	4.43 ms	41.45 ms	15.09 %
24066	365.97	standard		32.21 ms	4.84 ms	1.19 ms	38.24 ms	191.21 ms	87.34 %
		fast	31.27 ms	0.78 ms		0.07 ms	5.69 ms	59.72 ms	14.94 %
30033	410.58	standard		45.47 ms	5.87 ms	1.47 ms	52.81 ms	264.07 ms	88.88 %
		fast	45.16 ms	0.98 ms		0.06 ms	6.91 ms	79.73 ms	15.04 %
36069	482.25	standard		64.32 ms	7.91 ms	1.87 ms	74.10 ms	370.52 ms	89.32 %
		fast	63.73 ms	1.29 ms		0.07 ms	9.27 ms	110.10 ms	14.67 %

Table 1. Comparison of performance between the standard update scheme and the fast update scheme for different numbers of *mechanical DOFs* (*DOFs*) and constraints (Cst.): For the fast update scheme, building the mapping compliance matrix (Build $\mathbf{W}_{\mathbf{H}}$) is performed once each time step. The following processes are performed once in each Newton iteration: updating the compliance matrix (Rebuild $\mathbf{W}$), processing the local PGS (PGS), and computing the corrective motion (Corr.). Then the cost of each Newton iteration (Newton ite.) and the whole recursive corrective motion scheme (Total) are compared. Finally, the proportion of the update constraint matrix process in the Newton iteration (Update) is illustrated.

the constraint correction in an iterative way, which performs Newton-Rapshon iterations. Within each Newton iteration, a local **PGS** is performed to solve the complementarity problem in contact and friction responses, giving temporary contact forces. The matrix operations in Algorithm 3 are realized with typical GPU-based implementation in **cuBLAS** and **cuSPARSE** libraries.

Such contact forces are used to compute the position of the surface elements, which are further used to define the constraint directions in the next iteration. In Table 1, we compare the computational cost between the standard scheme and the fast update scheme in In a simple contact scenario, a recursive corrective motion scheme with 5 Newton iterations is performed. Each Newton iteration is accompanied by a local constraint resolution with 30 **PGS** iterations. In the standard scheme (see Algorithm 2), updating constraint matrices takes massive computation cost: Rebuilding the compliance matrix $\mathbf{W}$ and applying the constraint correction with traditional way (see Equation (20)) require to invert the system matrix $\mathbf{A}$. The extra cost by these updating processes is enormous, taking 84-90 % of each Newton iteration. On the other hand, in the fast updating scheme (see Algorithm 3), the computation of the mapping compliance matrix $\mathbf{W_H}$ is outside of the recursive scheme and is performed only once in each time step. Our method allows fast rebuilding of the compliance matrix (by matrix-matrix multiplications) and efficiently computing the correction on proximity positions (by matrix-vector multiplication). The matrix multiplication operations are very suitable to be parallelized on GPU. Moreover, the extra updating cost takes 14-15 % of each Newton iteration. Our method is 6.97 × faster than the standard scheme for each Newton iteration. For the cost of the whole Newton scheme, including the overhead of building $\mathbf{W_H}$, our method benefits a speedup of 3.20 × compared to the standard scheme.

Besides the computational performance, our method is straightforward since only matrix operations are required in the recursive scheme. Moreover, the implementation of the method remains flexible. In fact, Algorithm 2 could be applied in different ways. Since the update process is very efficient, it is possible to update the constraint directions in each PGS iteration, or even to carry out the update after the resolution of each constraint, which is similar to the resolution scheme in the Position-Based Dynamics [27].

Our future work is to find more applications for the recursive corrective motion scheme with the fast update method. We hope that our work could inspire readers to help them improve the constraint resolution performance in their works on interactive FE simulations.

Acknowledgement: This work was supported by French National Research Agency (ANR) within the project SPERRY ANR-18-CE33-0007.).

REFERENCES

[1] Jérémie Allard, François Faure, Hadrien Courtecuisse, Florent Falipou, Christian Duriez, and Paul G. Kry. 2010. Volume contact constraints at arbitrary resolution. *ACM SIGGRAPH 2010 Papers, SIGGRAPH 2010* C, 3 (jul 2010), 1–10. https://doi.org/10.1145/1778765.1778819

[2] Sheldon Andrews and Kenny Erleben. 2021. Contact and friction simulation for computer graphics. *ACM SIGGRAPH 2021 Courses, SIGGRAPH 2021* (8 2021). https://doi.org/10.1145/3450508.3464571

[3] Mihai Anitescu and Florian A. Potra. 2002. A time-stepping method for stiff multibody dynamics with contact and friction. *Internat. J. Numer. Methods Engrg.* 55 (11 2002), 753–784. Issue 7. https://doi.org/10.1002/NME.512

[4] Paul Baksic, Hadrien Courtecuisse, and Bernard Bayle. 2021. Shared control strategy for needle insertion into deformable tissue using inverse Finite Element simulation. In *IEEE International Conference on Robotics and Automation*. 12442–12448.

[5] David Baraff. 1994. Fast contact force computation for nonpenetrating rigid bodies. *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1994* (7 1994), 23–34. https://doi.org/10.1145/192161.192168

[6] David Baraff. 1996. Linear-time dynamics using Lagrange multipliers. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1996*. ACM, 137–146. https://doi.org/10.1145/237170.237226

[7] Jan Bender, Matthias Müller, Miguel A. Otaduy, Matthias Teschner, and Miles Macklin. 2014. A survey on position-based simulation methods in computer graphics. *Computer Graphics Forum* 33, 6 (2014), 228–251. https://doi.org/10.1111/cgf.12346

[8] Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. 2014. Projective dynamics. *ACM Transactions on Graphics (TOG)* 33 (7 2014). Issue 4. https://doi.org/10.1145/2601097.2601116

[9] Morten Bro-Nielsen and Stéphane Cotin. 1996. Real-time volumetric deformable models for surgery simulation using finite elements and condensation. *Computer Graphics Forum* 15 (1996), 57–66.

[10] Olivier Comas, Zeike A Taylor, Jérémie Allard, Sébastien Ourselin, Stéphane Cotin, and Josh Passenger. 2008. Efficient Nonlinear FEM for Soft Tissue Modelling and Its GPU Implementation within the Open Source Framework SOFA. In *Biomedical Simulation*, Vol. 5104 LNCS. 28–39. https://doi.org/10.1007/978-3-540-70521-5_4

[11] Hadrien Courtecuisse and Jeremie Allard. 2009. Parallel dense gauss-seidel algorithm on many-core processors. In *2009 11th IEEE International Conference on High Performance Computing and Communications, HPCC 2009*. IEEE, 139–147. https://doi.org/10.1109/HPCC.2009.51

[12] Hadrien Courtecuisse, Jérémie Allard, Christian Duriez, and Stéphane Cotin. 2010. Asynchronous preconditioners for efficient solving of non-linear deformations. In *VRIPHYS 2010 - 7th Workshop on Virtual Reality Interactions and Physical Simulations*. 59–68. https://doi.org/10.2312/PE/vriphys/vriphys10/059-068

[13] Hadrien Courtecuisse, Jérémie Allard, Pierre Kerfriden, Stéphane P.A. Bordas, Stéphane Cotin, and Christian Duriez. 2014. Real-time simulation of contact and cutting of heterogeneous soft-tissues. *Medical Image Analysis* 18, 2 (2014), 394–410. https://doi.org/10.1016/j.media.2013.11.001

[14] Christian Duriez. 2013. Control of elastic soft robots based on real-time finite element method. *Proceedings - IEEE International Conference on Robotics and Automation* (2013), 3982–3987. https://doi.org/10.1109/ICRA.2013.6631138

[15] C. Duriez, F. Dubois, A. Kheddar, and C. Andriot. 2006. Realistic haptic rendering of interacting deformable objects in virtual environments. *IEEE Transactions on Visualization and Computer Graphics* 12, 1 (apr 2006), 36–47. https://doi.org/10.1109/TVCG.2006.13 arXiv:0804.0561

[16] Kenny Erleben. 2013. Numerical methods for linear complementarity problems in physics-based animation. *ACM SIGGRAPH 2013 Courses, SIGGRAPH 2013* February (2013). https://doi.org/10.1145/2504435.2504443

[17] Ca Felippa. 2000. *A systematic approach to the element-independent corotational dynamics of finite elements*. Technical Report January. College Of Engineeringuniversity Of Colorado. http://www.colorado.edu/engineering/cas/Felippa.d/FelippaHome.d/Publications.d/Report.CU-CAS-00-03.pdf

[18] Bruno Heidelberger, Matthias Teschner, and Markus Gross. 2003. Real-Time Volumetric Intersections of Deforming Objects. *VMV'03: Proceedings of Vision, Modeling, Visualization* 2003, 461–468.

[19] Philipp Herholz and Marc Alexa. 2018. Factor once: Reusing Cholesky factorizations on sub-meshes. *SIGGRAPH Asia 2018 Technical Papers, SIGGRAPH Asia 2018* 37, 6 (jan 2018), 1–9. https://doi.org/10.1145/3272127.3275107

[20] Philipp Herholz and Olga Sorkine-Hornung. 2020. Sparse cholesky updates for interactive mesh parameterization. *ACM Transactions on Graphics* 39, 6 (2020). https://doi.org/10.1145/3414685.3417828

[21] M. Jean. 1999. The non-smooth contact dynamics method. *Computer Methods in Applied Mechanics and Engineering* 177, 3-4 (1999), 235–257. https://doi.org/10.1016/S0045-7825(98)00383-1

[22] Tiantian Liu, Sofien Bouaziz, and Ladislav Kavan. 2017. Quasi-Newton methods for real-time simulation of hyperelastic materials. *ACM Transactions on Graphics* 36, 3 (2017). https://doi.org/10.1145/2990496

[23] Mickaël Ly, Jean Jouve, Laurence Boissieux, and Florence Bertails-Descoubes. 2020. Projective dynamics with dry frictional contact. *ACM Transactions on Graphics (TOG)* 39 (7 2020). Issue 4. https://doi.org/10.1145/3386569.3392396

[24] Miles Macklin, Kenny Erleben, Matthias Müller, Nuttapong Chentanez, Stefan Jeschke, and Viktor Makoviychuk. 2019. Non-smooth Newton methods for deformable multi-body dynamics. *ACM Transactions on Graphics* 38, 5 (oct 2019). https://doi.org/10.1145/3338695 arXiv:1907.04587

[25] Miles Macklin, Matthias Müller, and Nuttapong Chentanez. 2016. XPBD: Position-based simulation of compliant constrained dynamics. *Proceedings - Motion in Games 2016: 9th International Conference on Motion in Games, MIG 2016* (10 2016), 49–54. https://doi.org/10.1145/2994258.2994272

[26] Stéphanie Marchesseau, Tobias Heimann, Simon Chatelin, Rémy Willinger, and Hervé Delingette. 2010. Multiplicative Jacobian Energy Decomposition Method for Fast Porous Visco-Hyperelastic Soft Tissue Model. In *Lecture notes in computer science*. Vol. 6361. Springer, 235–242. https://doi.org/10.1007/978-3-642-15705-9_29

[27] Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. 2007. Position based dynamics. *Journal of Visual Communication and Image Representation* 18 (4 2007), 109–118. Issue 2. https://doi.org/10.1016/J.JVCIR.2007.01.005

[28] Quan Nie, Yingfeng Zhao, Li Xu, and Bin Li. 2020. A Survey of Continuous Collision Detection. *undefined* (12 2020), 252–257. https://doi.org/10.1109/ITCA52113.2020.00061

[29] Cosmin G. Petra, Olaf Schenk, Miles Lubin, and Klaus Gäertner. 2014. An augmented incomplete factorization approach for computing the schur complement in stochastic optimization. *SIAM Journal on Scientific Computing* 36, 2 (2014), C139–C162. https://doi.org/10.1137/130908737

[30] Yves Renard. 2013. Generalized Newton's methods for the approximation and resolution of frictional contact problems in elasticity. *Computer Methods in Applied Mechanics and Engineering* 256 (2013), 38–55. https://doi.org/10.1016/j.cma.2012.12.008

[31] Guillaume Saupin, Christian Duriez, Stephane Cotin, and Laurent Grisoni. 2008. Efficient Contact Modeling using Compliance Warping. *Computer graphics international*. http://hal.inria.fr/hal-00844039

[32] Olaf Schenk and Klaus Gärtner. 2006. On fast factorization pivoting methods for sparse symmetric indefinite systems. *Electronic Transactions on Numerical Analysis* 23 (2006), 158–179.

[33] E Sifakis and Jernej Barbič. 2012. FEM simulation of 3D deformable solids: a practitioner's guide to theory, discretization and model reduction. *ACM SIGGRAPH 2012 Posters* 85, 7 (2012), 1–35. https://doi.org/10.3987/Contents-12-85-7

[34] D. E. Stewart and J. C. Trinkle. 1996. An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction. *Internat. J. Numer. Methods Engrg.* 39, 15 (1996), 2673–2691. https://doi.org/10.1002/(SICI)1097-0207(19960815)39:15<2673::AID-NME972>3.0.CO;2-I

[35] M. Teschner, S. Kimmerle, B. Heidelberger, G. Zachmann, L. Raghupathi, A. Fuhrmann, M. P. Cani, F. Faure, N. Magnenat-Thalmann, W. Strasser, and P. Volino. 2005. Collision Detection for Deformable Objects. *Computer Graphics Forum* 24 (3 2005), 61–81. Issue 1. https://doi.org/10.1111/J.1467-8659.2005.00829.X

[36] Jinao Zhang, Yongmin Zhong, and Chengfan Gu. 2018. Deformable Models for Surgical Simulation: A Survey. *IEEE Reviews in Biomedical Engineering* 11 (11 2018), 143–164. https://doi.org/10.1109/RBME.2017.2773521