DOI: 10.1111/cgf.14625
ACM SIGGRAPH / Eurographics Symposium on Computer Animation 2022
D. L. Michels and S. Pirk
(Guest Editors)
Volume 41 (2022), Number 8

Surface-Only Dynamic Deformables using a Boundary Element Method

R. Sugimoto† C. Batty† T. Hachisuka†

University of Waterloo, Canada

Abstract

We propose a novel surface-only method for simulating dynamic deformables without the need for volumetric meshing or volumetric integral evaluations. While based upon a boundary element method (BEM) for linear elastodynamics, our method goes beyond simple adoption of BEM by addressing several of its key limitations. We alleviate large displacement artifacts due to linear elasticity by extending BEM with a moving reference frame and surface-only fictitious forces, so that it only needs to handle deformations. To reduce memory and computational costs, we present a simple and practical method to compress the series of dense matrices required to simulate propagation of elastic waves over time. Furthermore, we explore a constraint enforcement mechanism and demonstrate the applicability of our method to general computer animation problems, such as frictional contact.

CCS Concepts

• ***Computing methodologies*** → ***Physical simulation;***

1. Introduction

Physical simulation of dynamically deforming elastic objects (*elastodynamics*) is now widely deployed for computer animation. Most such simulation methods rely on discretizing an object into a *volumetric* mesh of tetrahedral or cubic elements. The simulation then processes both the exterior (surface) and the interior of the the object. Computer animation, however, is often concerned only with the visible motion of the object's surface. We propose a practical *surface-only dynamic deformables* simulation method (Fig. 1) suitable for computer animation. Being a surface-only method, our method does not need any volumetric discretization.

Our method builds upon advances in the boundary element method (BEM) [LMN*12]. BEM can solve partial differential equations (PDEs) while discretizing only the boundary (surface) of the domain. James et al. [JP99] applied BEM for the first time to computer animation based on linear elastostatics. Although BEM is becoming increasingly popular in computer animation [HW15; HW16; DHB*16], we are the first to solve elastodynamics simulation with BEM in computer animation.

BEM for elastodynamics is an active area of research even in computational mechanics [LMN*12]. The intuition behind elastodynamics BEM is that we can express deformation at a point on a surface by considering a superposition of elastic waves propagated from all surface points over time. Mathematically, we achieve this by solving a boundary integral equation (BIE) derived from the Navier-Cauchy equations of linear elasticity. Since this BIE is an integral over the object's surface, only the surface needs to be discretized. Unlike the BIE for elastostatics [JP99], this BIE for elastodynamics involves *convolution over time* to account for the influence of the past motion, leading to a recursive matrix equation after discretization.

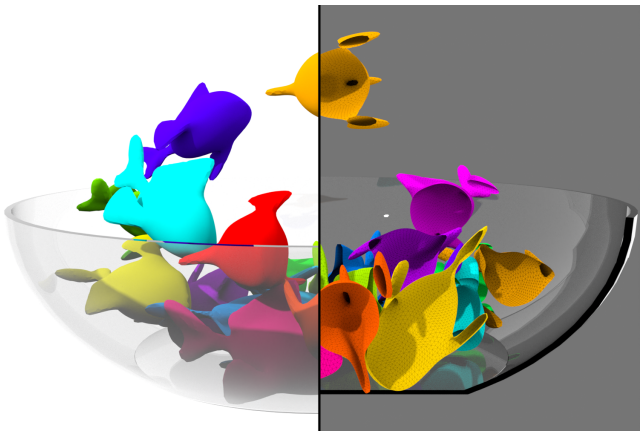


Figure 1: *Dynamic elastic fish falling into a bowl and colliding (left), simulated with our surface-only approach that requires no interior meshing (right).*

† email: {rsugimot|c2batty|thachisu}@uwaterloo.ca

We observed that simply employing BEM for elastodynamics would not result in a practical method. The first problem is that directly solving the BIE in a static frame results in artifacts due to the limitations of linear elasticity, such as volume gain under large rotations. We introduce a *moving* frame of reference and additional fictitious forces in BEM. This approach ensures that BEM only needs to handle local deformations, enabling large translation and rotation and leading to better numerical stability. Even compared to elastostatic methods that incorporate a moving rigid frame [HW16; JP02], adding dynamics requires we pay additional attention to ensure the method's stability. The second problem is that, unlike sparse matrices in volumetric simulations, the matrices in BEM are *dense*. In general, the computation and storage costs for dense matrices exhibit worse scaling compared to sparse matrices. We propose a compression method tailored to the dense matrices in BEM for elastodynamics. This compression method significantly reduces both the computational and storage costs by exploiting the smoothness of the integrands in the BIE. Lastly, we incorporate a constraint enforcement strategy that enables frictional contact and domain decomposition in order to support more complex scenarios. Our method thus goes beyond a simple application of BEM to elastodynamics animation. To summarize, our contributions are;

- an application and adaptation of elastodynamics BEM to computer animation problems,
- support for large global displacements for elastodynamics BEM using a moving body frame and fictitious forces,
- a simple matrix compression technique to reduce memory cost and computational cost.
- a constraint-based treatment of joints, frictional contact, and domain decomposition for our BEM scheme.

2. Related Work

Physics-based deformable body simulation. Physics-based simulations of deformable bodies typically solve partial differential equations with volumetric spatial discretizations. The pioneering work of Terzopoulos et al. [TPBF87] uses a finite difference method to discretize the space, whereas Teran et al. [TBNF03] applied the finite volume method. The finite element method (FEM) [SB12] later became increasingly popular due to its generality and strong theoretical foundations. Smoothed particle hydrodynamics [KBST19] and the material point method [JST*16; HZGJ19] have also been applied to deformable body simulation. All of the above methods require discretization of and computation over the interior of the volumetric domain; we avoid this interior discretization via BEM.

Geometry-based deformable body animation. Geometry-based (or position-based) methods are common alternatives to physics-based simulations. These methods often have attractive properties for computer animation, such as computational efficiency and unconditional stability. Early work in this direction includes shape matching [MHTG05] and position-based dynamics [MHHR07]. These methods have been extended in many ways, and we refer to the survey by Bender et al. [BMM17] for an overview. In their basic forms, such methods often lack the accuracy and realism of physics-based methods. Furthermore, these approaches also often rely on volumetric meshes. Our BEM method is physics-based and does not assume having any volumetric meshes.

BEM in graphics. In contrast to the methods above, BEM solves linear partial differential equations with only a *surface discretization*. The potential benefits of BEM in computer animation were first recognized by James et al. [JP99] for elastostatics simulation. Several applications in geometry processing and physics-based simulation have subsequently been explored. For example, James et al. [JBP06] and Umetani et al. [UPSW16] used BEM for acoustic transfer problems by solving the Helmholtz equation. Several authors have used elastostatics BEM to simulate brittle fractures [ZBG15; HW15; HW16]. Da et al. [DHB*16] and Huang et al. [HM20] explored BEM for surface-only liquid animation. They solve the Laplace equation with a mixed boundary condition to perform the pressure solve. A range of applications of BEM to geometry processing tasks have also been studied [SVB17; WSSK13; LW16]. A key distinction of our work compared to prior graphics work is that we consider, for the first time, solving a *dynamic* problem with BEM. For example, while the surface-only liquids method [DHB*16] is overall solving a dynamic problem, its BEM step is applied only to solve the static Laplace problem. Our method directly solves a hyperbolic PDE with BEM, whereas prior work uses BEM to solve only elliptic PDEs.

Elastodynamics BEM. The first time-domain elastodynamics BEM for 3D transient problems was proposed by Banerjee et al. [BAM86] and Manolis et al. [MB88]. They solve the boundary integral equation using analytical integration over time, assuming constant nodal displacements and tractions over each time step. This early work suffered from "intermittent instability" effects [PS97], meaning that the solution becomes unstable in a manner that depends on the time step size in an erratic and unpredictable way. Several researchers proposed approaches to alleviate this issue, for example by employing different fundamental solutions or using higher order interpolation functions. Among them, Schanz and Antes [SA97; Sch01] claim that their formulation called *convolution quadrature BEM* (CQBEM) is less sensitive to the choice of time step size compared to the classical counterpart. Our initial experiments supported this claim, and we base our method upon CQBEM. Li et al. [LZ13] give a summary on this topic and also conclude that CQBEM numerically outperforms other representative options in terms of stability and accuracy. There exist more recent methods [BMS12; ADFG12], but with increased implementation complexities and we leave exploring them as future work.

Fundamental solutions. For a time-dependent differential equation, a fundamental solution is an analytical solution to the problem in the case of a source (or load) that is concentrated in both space and time. A fundamental solution assumes an infinite domain without any specific boundary conditions. De Goes et al. [DJ18] use elastodynamics fundamental solutions directly on the infinite ambient volume containing an object in order to approximate simple secondary motions. This approach, however, cannot handle the boundary conditions necessary for general elastodynamics simulations. Our method instead relies on BEM, which uses fundamental solutions in its derivation, but supports finite domains and general boundary conditions needed for elastodynamics simulations.

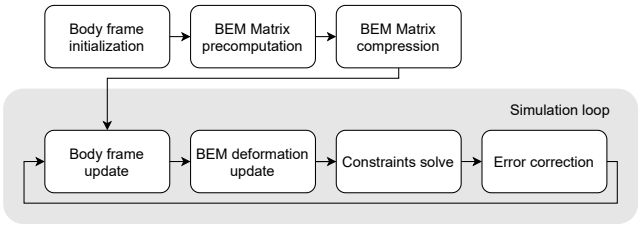


Figure 2: *Overview of our method.*

Linear elasticity and rotation effects. Currently, elastodynamics BEMs that avoid the need for computation over volumetric domain are based only on linear elasticity and thus cannot naturally handle large displacements. This fact has limited the type of dynamic motions one can simulate with BEM. James et al. [JP99] did not address this problem, nor has the computational mechanics community. In computer graphics, similar problems have been addressed (for non-BEM schemes) by coupling linear elasticity and rigid body dynamics [TW88] or a corotated linear strain measure for FEM [MDM*02; MG04]. Hahn et al. [HW16] augment a rigid body simulator with fracture effects due to collisions by solving instantaneous elastostatic BEM problems upon contact, assuming that deformations due to elasticity are almost negligible. The BEM traction field is computed from rigid body contact impulses. This method is not applicable to elastodynamics because it does not consider time-history effects. We address this issue by solving an elastodynamics BEM problem on a moving rigid frame.

3. Surface-Only Dynamic Deformables

Our surface-only dynamic deformables formulation begins by augmenting an elastodynamics BEM with a moving frame of reference undergoing rigid body dynamics. This moving reference frame can absorb effects due to large global rotational and translational displacements and the BEM needs to handle only the remaining local deformations. The motion of the frame imposes fictitious forces in the frame, which we can include as external forces in the BEM. In the following, we first introduce basic elastodynamics BEM along with our new fictitious forces (Section 3.1), and then describe how to apply the BEM within a moving frame (Section 3.2)

With this framework laid out, we introduce two further extensions to handle diverse animation scenarios. We discuss how to compress the dense matrices in our BEM method to reduce its computation and storage costs (Section 3.3). We then propose our constraint enforcement method to support joints, frictional contact effects, and domain decomposition for even larger deformations (Section 3.4). Fig. 2 shows the relation between the building blocks of our method. Our method consists of precomputation steps, which amortize costs for the simulation loop, and runtime steps, which carry out the simulation itself. We use a bold face font (i.e., $\mathbf{u}$) to denote vectors and matrices in the continuous setting and use a sans serif font (i.e., u) to denote vectors and matrices after discretization.

3.1. Elastodynamics BEM

Below we describe our chosen baseline time-domain elastodynamics BEM [SA97; Sch01], emphasizing differences with respect to elastostatics [JP99]. The most fundamental of these differences is the introduction of temporal convolutions. The original work by Schanz et al. [SA97] does not explain how to handle body forces such that the resulting BIE contains only surface integrals. If volume integrals remain, they might necessitate interior discretization which would defeat the purpose of BEM. We therefore explain how gravity and two types of fictitious forces from a moving frame can be incorporated into BEM as body forces using only surface integrals.

3.1.1. Boundary integral equation

Suppose that we have a homogeneous, isotropic, linearly elastic material with undeformed volumetric domain Ω having boundary $\Gamma = \partial\Omega$. The displacement field for the material, $\mathbf{u}(\mathbf{x},t)$, follows the Navier-Cauchy equations,

$$\mu\nabla^2\mathbf{u} + (\mu+\lambda)\nabla(\nabla\cdot\mathbf{u}) + \mathbf{b} = \rho\ddot{\mathbf{u}}, \tag{1}$$

where λ and μ are Lamé parameters, ρ is a constant mass density, and $\mathbf{b}(\mathbf{x},t)$ is the external forces applied to the body, e.g., gravity. We omit the dependency on variables $\mathbf{x}$ and t for brevity in the equation above. We can also derive the traction (i.e., force per unit area) applied on the surface, $\mathbf{p}(\mathbf{x},t)$, from $\mathbf{u}(\mathbf{x},t)$ by Cauchy's theorem [Sch01]. The presence of the second order time derivative of displacement (indicated by double overdots), absent for elastostatics, makes the equation a hyperbolic PDE.

The problem is to solve Eq. (1) for $\mathbf{x}\in\Omega, t\geq 0$ under time-dependent traction boundary conditions and zero initial displacements and velocities:

$$\begin{aligned} \mathbf{p}(\mathbf{x},t) &= \bar{\mathbf{p}}(\mathbf{x},t) && \text{for} \quad \mathbf{x}\in\Gamma, t\geq 0 \\ \mathbf{u}(\mathbf{x},0) &= \dot{\mathbf{u}}(\mathbf{x},0) = \mathbf{0} && \text{for} \quad \mathbf{x}\in\Omega, \end{aligned} \tag{2}$$

where the overline indicates the prescribed boundary function. Other boundary conditions, such as displacements specified at vertices, can be handled using a constraint solver (see Section 3.4) or penalty springs [MZS*11]. A fundamental solution for displacements $\mathbf{u}^*(\mathbf{x},\mathbf{y},t)$ and traction $\mathbf{p}^*(\mathbf{x},\mathbf{y},t)$ to Eq. (1) can be derived analytically as responses at $(\mathbf{x},t)$ to a unit impulsive load at $(\mathbf{y},\tau)$ in an infinite medium.

With zero initial displacements and velocities, these fundamental solutions allow us to transform Eq. (1) into a BIE in the form of

$$\begin{aligned} \mathbf{c}(\mathbf{x})\mathbf{u}(\mathbf{x},t) = &-\int_0^t\int_\Gamma \mathbf{p}^*(\mathbf{x},\mathbf{y},t-\tau)\mathbf{u}(\mathbf{y},\tau)\,d\Gamma_\mathbf{y}\,d\tau \\ &+\int_0^t\int_\Gamma \mathbf{u}^*(\mathbf{x},\mathbf{y},t-\tau)\mathbf{p}(\mathbf{y},\tau)\,d\Gamma_\mathbf{y}\,d\tau \\ &+\int_0^t\int_\Omega \mathbf{u}^*(\mathbf{x},\mathbf{y},t-\tau)\mathbf{b}(\mathbf{y},\tau)\,d\Omega_\mathbf{y}\,d\tau, \end{aligned} \tag{3}$$

where $\mathbf{c}(\mathbf{x})$ is called the integral free term and is associated with the smoothness of the boundary in the undeformed configuration.

The main difference from the elastostatic case [JP99] is the introduction of a time integral from the beginning of the simulation to the current time. Intuitively, this integral expresses that elastic waves that propagate through the body from any boundary point $\mathbf{y}$ are superposed in space and time and affect the displacement at $\mathbf{x}$ at the current time. The last term in Eq. (3) expresses body force

effects and it involves a volume integral. We need to avoid discretizing this volume integral to have a surface-only method. This term is typically ignored in the computational mechanics literature.

When we use this boundary integral equation within a moving, non-inertial frame of reference, such a frame introduces additional forces on objects, called *fictitious forces*. Let us denote the translational acceleration of the frame by $\mathbf{a}$, the rotational acceleration of the frame by α, the gravitational acceleration by $\mathbf{g}'$, expressed in the moving frame, respectively. We further denote a skew-symmetric cross-product matrix by $[\cdot]_\times$, and the center of mass of the body in the undeformed configuration by $\bar{\mathbf{x}}$. With this notation, we can introduce forces due to translational acceleration $-\rho\mathbf{a}(\tau)$ and rotational acceleration (the latter known as the Euler force) $-\rho[\mathbf{y}-\bar{\mathbf{x}}]_\times^T\alpha(\tau)$, omitting the centrifugal and Coriolis forces for simplicity. We explain more details regarding fictitious forces in a later subsection. We further include the gravitational force $\rho\mathbf{g}'(\tau)$. We incorporate these three forces into the BEM system as body force terms, i.e. $\mathbf{b}(\mathbf{y},\tau) = -\rho\mathbf{a}(\tau) - \rho[\mathbf{y}-\bar{\mathbf{x}}]_\times^T\alpha(\tau) + \rho\mathbf{g}'(\tau)$, computed based on the body's undeformed shape. Fortunately, we can analytically convert the volume integral in these body force terms into surface integrals as:

$$\begin{aligned}&\int_0^t \int_\Omega \mathbf{u}^*(\mathbf{x},\mathbf{y},t-\tau)\mathbf{b}(\mathbf{y},\tau)\, d\Omega_\mathbf{y}\, d\tau \\ &= \int_0^t \left(\int_\Gamma \mathbf{d}^*(\mathbf{x},\mathbf{y},t-\tau) d\Gamma_\mathbf{y}\right)(\mathbf{a}(\tau)-\mathbf{g}'(\tau))\, d\tau \\ &\quad + \int_0^t \left(\int_\Gamma \mathbf{q}^*(\mathbf{x},\mathbf{y},t-\tau)\, d\Gamma_\mathbf{y}\right)\alpha(\tau)\, d\tau.\end{aligned} \tag{4}$$

A general approach to convert volume integrals to boundary integrals in the context of BEM, called the multiple reciprocity method (MRM), is available for elastostatics [NB89] and applied in the work by James et al. [JP99]. To our knowledge, MRM has not been applied to elastodynamic problems, possibly due to its mathematical complexity. We instead applied a series of calculus identities to derive the expressions for $\mathbf{d}^*$ and $\mathbf{q}^*$. See the supplementary note for details.

3.1.2. Convolution quadrature BEM

We discretize Eq. (3) after substituting its last integral with Eq. (4). We use CQBEM which employs time-discretization based on the *convolution quadrature method* (CQM) [Lub88a; Lub88b]. CQM is a general numerical integration method that approximates a convolution integral of the form

$$y(t) = f(t) * g(t) = \int_0^t f(t-\tau)g(\tau)\, d\tau, \tag{5}$$

where the quadrature weights are determined using the Laplace transform and a linear multistep method. Specifically, given the closed-form expressions for the time-domain function $g(t)$ and the Laplace transformed function of $f(t)$, $\hat{f}(s)$, the function y at time $n\Delta t$ then can be approximated as

$$y(n\Delta t) \approx \sum_{k=0}^{n_{max}} \psi_k(\hat{f})\, g((n-k)\Delta t), \tag{6}$$

where n_{max} is the maximum number of time steps considered in the underlying linear multistep method, which is computed based on the input geometry and the input parameters, and $\psi_k(\hat{f})$ is an integration weight computed using $\hat{f}$. Section 4 discusses the details for n_{max} and this integration weight.

We assume that the input object is a triangle mesh with N vertices and M triangles, and discretize time with constant step size Δt. Using piecewise linear interpolation for the displacement and traction at each vertex, we have

$$\begin{aligned}\mathbf{u}(\mathbf{x},k\Delta t) &= \Phi(\mathbf{x})\mathsf{u}_k \\ \mathbf{p}(\mathbf{x},k\Delta t) &= \Phi(\mathbf{x})\mathsf{p}_k\end{aligned} \quad \text{for} \quad \mathbf{x}\in\Gamma, \tag{7}$$

where $\Phi(\mathbf{x})$ is a piecewise linear interpolation matrix and u_k and p_k are $3N$-vectors consisting of the nodal displacements and tractions at the k^{th} time step, respectively. We observed that piecewise linear interpolation offers better stability than piecewise constant interpolation.

We then apply CQM (Eq. (6)) for temporal discretization and integration, and piecewise linear interpolation for spatial discretization (Eq. (7)) to each term in Eq. (3) with the body force terms described in Eq. (4). At the i^{th} vertex, $\mathbf{x} = \mathbf{x}_\mathsf{i}$, at time $t = t_n$, we get

$$\begin{aligned}\mathbf{c}(\mathbf{x}_\mathsf{i})\mathsf{u}_{n,\mathsf{i}} = \sum_{k=0}^{n_{max}} \Bigg\{ &-\left(\int_\Gamma \psi_k(\hat{\mathbf{p}}^*)(\mathbf{x}_\mathsf{i},\mathbf{y})\, \Phi(\mathbf{y})\, d\Gamma_\mathbf{y}\right)\mathsf{u}_{n-k} \\ &+\left(\int_\Gamma \psi_k(\hat{\mathbf{u}}^*)(\mathbf{x}_\mathsf{i},\mathbf{y})\, \Phi(\mathbf{y})\, d\Gamma_\mathbf{y}\right)\mathsf{p}_{n-k} \\ &+\left(\int_\Gamma \psi_k(\hat{\mathbf{d}}^*)(\mathbf{x}_\mathsf{i},\mathbf{y})\, d\Gamma_\mathbf{y}\right)(\mathsf{a}_{n-k}-\mathsf{g}'_{n-k}) \\ &+\left(\int_\Gamma \psi_k(\hat{\mathbf{q}}^*)(\mathbf{x}_\mathsf{i},\mathbf{y})\, d\Gamma_\mathbf{y}\right)\alpha_{n-k}\Bigg\},\end{aligned} \tag{8}$$

where $\hat{(\cdot)}$ denotes a Laplace transformed function, $(\cdot)_{n-k} = (\cdot)((n-k)\Delta t)$, and $\mathsf{u}_{n,\mathsf{i}}$ is the nodal displacement 3-vector for the i^{th} vertex at time $t = n\Delta t$. The ordering of terms in Eq. (3) after substitution of Eq. (4) is retained in Eq. (8) to expose how the CQM and interpolation were applied. We provide expressions for $\hat{\mathbf{p}}^*$, $\hat{\mathbf{u}}^*$, $\hat{\mathbf{d}}^*$ and $\hat{\mathbf{q}}^*$ in Appendix A. .

We can precompute the boundary integrals in Eq. (8) by taking the sum of integrals over triangles. See Section 4 for implementation details. By assembling the equations for all vertices $\mathbf{x}_\mathsf{i}$ into a matrix equation and reordering terms, we get

$$\begin{aligned}&\mathsf{H}_0\mathsf{u}_n = \mathsf{G}_0\mathsf{p}_n + \mathsf{D}_0(\mathsf{a}_n - \mathsf{g}'_n) + \mathsf{Q}_0\alpha_n \\ &\underbrace{+\sum_{k=1}^{n_{max}}\left(-\mathsf{H}_k\mathsf{u}_{n-k} + \mathsf{G}_k\mathsf{p}_{n-k} + \mathsf{D}_k(\mathsf{a}_{n-k}-\mathsf{g}'_{n-k}) + \mathsf{Q}_k\alpha_{n-k}\right)}_{\text{time history effects}}.\end{aligned} \tag{9}$$

The matrices $\mathsf{H}_k, \mathsf{G}_k$ ($\in \mathbb{R}^{3N\times 3N}$) and $\mathsf{D}_k, \mathsf{Q}_k$ ($\in \mathbb{R}^{3N\times 3}$) correspond to the terms with $\hat{\mathbf{p}}^*, \hat{\mathbf{u}}^*, \hat{\mathbf{d}}^*$, and $\hat{\mathbf{q}}^*$ in Eq. (8), respectively. The integral free term $\mathbf{c}(\mathbf{x}_\mathsf{i})$ is absorbed into H_0. Unlike matrices in FEM, these matrices are *dense*.

This equation shows that, as a result of discretizing the convolution integrals, finding the unknown current displacement u_n requires data both from past steps and the current one. From the current time step, we require the traction and body acceleration vectors; from the past n_{max} time steps, we require the displacement, traction, and body acceleration vectors. Thus it is a recursive matrix

equation. This structure lets us store information in a surface-only manner, in contrast to volumetric elasticity formulations. Unlike elastostatics BEM [HW16], the H_0 matrix is full-rank in general (i.e., no null space) implying that we can straightforwardly solve problems with pure traction boundary conditions.

3.2. Local Deformation within a Moving Frame

One of our technical contributions is to use BEM within a moving frame of reference to alleviate the limitations due to linear elasticity, inspired by prior work outlined in Section 2. We approximate the motion of the reference frame with rigid body dynamics based on the elastic body's undeformed shape, based on the linear elasticity assumption of modest deformations. The traction applied on the surface of the deformable body drives the frame's motion and the recursive matrix equation Eq. (9) updates the displacement vector using the approximated frame's accelerations.

At the beginning of the simulation, we compute the mass m, center of mass $\bar{\mathbf{x}}$, and inertia tensor $\mathcal{I}$ of the body in its undeformed configuration using boundary integrals [Mir96]. At the k^{th} simulation step, we compute the translational and rotational accelerations due to surface traction with boundary integrals:

$$\begin{aligned} \mathsf{a}_k^{ext} &= \frac{1}{m}\left(\int_\Gamma \Phi(\mathbf{y})\, d\Gamma_\mathbf{y}\right)\mathsf{p}_k =: \mathsf{A}\mathsf{p}_k, \\ \alpha_k^{ext} &= \mathcal{I}^{-1}\left(\int_\Gamma (\mathbf{y}-\bar{\mathbf{x}})\times\Phi(\mathbf{y})\, d\Gamma_\mathbf{y}\right)\mathsf{p}_k =: \mathsf{C}\mathsf{p}_k. \end{aligned} \tag{10}$$

We move the frame using these accelerations:

$$\mathsf{g}'_k = \mathsf{R}_{k-1}^\top \mathbf{g} \tag{11a}$$
$$\mathsf{a}_k \approx \mathsf{g}'_k + \mathsf{a}_k^{ext}, \tag{11b}$$
$$\alpha_k \approx \alpha_k^{ext}, \tag{11c}$$
$$\dot{\mathsf{T}}_k = \dot{\mathsf{T}}_{k-1} + \Delta t \mathsf{R}_{k-1}\mathbf{a}_k, \tag{11d}$$
$$\omega_k = \omega_{k-1} + \Delta t \mathsf{R}_{k-1}\alpha_k, \tag{11e}$$
$$\mathsf{T}_k = \mathsf{T}_{k-1} + \Delta t \dot{\mathsf{T}}_k, \tag{11f}$$
$$\mathsf{R}_k = e^{\Delta t[\omega_k]_\times}\mathsf{R}_{k-1}, \tag{11g}$$

where ω is the angular velocity, R is the rotation of the frame, $\mathbf{g}$ is gravitational acceleration in the inertial frame. The subscript denotes the simulation step. We evaluate the matrix exponential in the rotation update equation using Rodrigues' rotation formula. We express the accelerations in the body frame coordinate system and we express the others in the inertial frame.

Additionally, we substitute Eq. (11b) and Eq. (11c) with Eq. (10) into Eq. (9) and multiply both sides by H_0^{-1} to get an approximation of the displacement vector,

$$\mathsf{u}_n \approx {\mathsf{H}_0}^{-1}\mathsf{G}'\mathsf{p}_n + {\mathsf{H}_0}^{-1}\mathsf{h}_n, \tag{12}$$

where h_n represents the time history terms in Eq. (9) and $\mathsf{G}' = \mathsf{G}_0 + \mathsf{D}_0\mathsf{A} + \mathsf{Q}_0\mathsf{C}$. Note the gravitational acceleration g'_n in Eq. (9) cancels out with the one in Eq. (11b). This proposed displacement is then corrected in the next step.

3.2.1. Error correction

These updates of the (moving) reference frame and displacement vector can remove the majority of the global translation and rotation from the displacement vector, yet there remains some global drift due to the approximations involved; left untreated, this drift can become a source of instability. Specifically, the motion of the frame does not account for the momentum due to deformation, and we have ignored some fictitious forces. In addition, the fact that the frame update process is based on the undeformed configuration may introduce additional error. To alleviate this problem, we apply a correction step at the end of each simulation step, in which we transfer the translational and rotational errors remaining in the displacement vector to the frame's motion. It suffices to design a strategy to extract the global translation and rotation remaining in the body frame in a consistent way over consecutive time steps, and we find the following simple geometry-based based approach greatly improves the stability of the method.

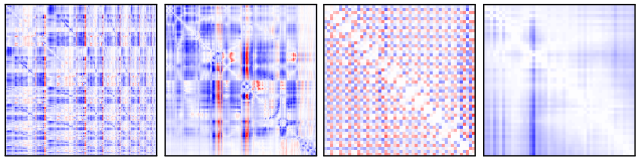
We find the translation remaining in the displacement vector by computing the difference between the current centroid and the initial centroid. We find the remaining rotation by applying shape matching [MHTG05] with the mass weighting $\mathsf{w} \in \mathbb{R}^N$ for each vertex defined as $\mathsf{w} = \int_\Gamma \mathbf{y}\cdot\mathbf{n}\Phi(\mathbf{y})\, d\Gamma_\mathbf{y}$, motivated by the surface integral to derive the volume of the body [Mir96].

We subtract the translational error from all elements of u_n and solve Eq. (9) for a_n in the least squares sense. We then remove the rotational error from u_n by $\mathsf{u}_n \leftarrow R^{-1}(\mathsf{x}+\mathsf{u}_n) - \mathsf{x}$, where x is a 3N-vector for undeformed vertex positions and R is a rotation matrix for the extracted rotational error. We solve Eq. (9) for α_n in the least squares sense with the updated a_n and u_n. As we only solved Eq. (9) in the least squares sense, the equality is not satisfied exactly in general. Therefore, we solve Eq. (9) for u_n with the updated a_n and α_n as a final step. The removed global translation and rotation are then reintroduced to the system as rigid body frame motions by updating the frame (Eq. 11) using the updated accelerations.

3.3. Compression of Matrices

Dense matrices in general require larger storage and computational costs than sparse matrices. In Eq. (9), all H and G matrices are dense, including the time history terms needed for dynamic effects. Each has $3N \times 3N$ entries, so the computational cost is quadratic in the vertex count and thus, unfortunately, scales worse than FEM, which scales linearly in the number of volumetric elements, in practice; we ignore the matrix inversion costs here since we perform inversion in the precomputation step. The BEM community has noticed this issue and proposed compression methods [LSSW12]. The existing methods [KS15; MS10] for elastodynamics simulations do not yet show test cases with large deformations and high compression ratios that we aim for. James et al. [JP03] used lifted wavelet transforms on the input mesh for elastostatics BEM assuming a multiresolution mesh as its input. We present a simple yet effective compression method based on wavelet transforms, which works on a general triangle surface mesh.

The time-domain fundamental solutions $\mathbf{u}^*$ and $\mathbf{p}^*$ are spatially smooth except at the wavefronts, and they decay smoothly as the source point $\mathbf{y}$ and target point $\mathbf{x}$ move apart. H and G matrices in Eq. (9) generally retain this property; the (i,j) entries of the matrices express how displacement or traction at the j^{th} vertex affect the i^{th} vertex's displacement. If the j^{th} and k^{th} vertices are spatially close



(a) *vertex reordering* **(b)** *coordinate subblock reordering*

Figure 3: *The reordering step yields matrices with smoother structure. (a) compares the x-x subblock of* H_1 *matrix for the fish in Fig. 1 without or with applying the vertex reordering, with the coordinate subblock reordering applied to both. (b) shows the top-left* 50×50 *submatrices of matrix* H_1 *without or with the coordinate subblock reordering, with the vertex reordering applied to both.*

to each other and have similar normals, the norm of the difference between columns j and k will be small, and the same holds for rows. We exploit this property in designing our compression scheme.

First, we reorder the vertices so their indices are more spatially smooth across the surface, using cache-oblivious mesh layouts [YLPM05]. We also order the matrix elements so that the x coordinates of all vertices come first, followed by all y and then all z coordinates, forming $3\times3=9$ subblocks. The reordered matrices exhibit several smooth dense blocks (Fig. 3). We then compress the matrices by applying the 2D (for row and column directions) Haar wavelet transform [BCR91] on the matrix elements. Because the transformation is linear, the transformed matrices support efficient evaluations of operations such as general matrix-vector multiplication and multiplication between a row of the matrix and a vector in the compressed form. We also tested other wavelet bases, but among those tested, the Haar basis offered the best balance between the compression ratio and the cost for multiplication operations.

We apply this compression method to the precomputed matrices H_0^{-1}, $\mathsf{H}_0^{-1}\mathsf{G}'$, H_k and G_k for $k=1,2,...,n_{max}$, and use them for all BEM solves, including constraint solves (Section 3.4). Note each D and Q matrix in Eq. (9) has only $3N\times3$ entries, making compression unnecessary.

3.4. Constraints, Friction, and Domain Decomposition.

To further extend the applicability of our method beyond prior BEM work, we implemented a Gauss-Seidel-based iterative position-level constraint solver. The previous methods solved contact problems by either simply specifying displacements of vertices [JP99] or using a rigid body contact method [HW16]. The computational mechanics literature [GS18] does not handle global motion of objects, to our knowledge. By contrast, we couple both the global rigid frame motion and deformation to solve the constraints and can handle fixed-position constraints, non-penetration constraints, friction constraints, and point-joint constraints.

We linearize the vertex position expressed in terms of the traction vector p_k at the current step as follows. The vertex position V_k at the k^{th} time step can be written as

$$\mathsf{V}_k^{\mathsf{i}} = \mathsf{R}_k(\mathsf{x}^{\mathsf{i}}+\mathsf{u}_k^{\mathsf{i}})+\mathsf{T}_k, \tag{13}$$

where $(\cdot)^{\mathsf{i}}$ denotes a 3-vector for the i^{th} vertex. Notice that R_k (Eq. (11g)) depends nonlinearly on p_k, while u_k (Eq. (12)) and T_k (Eq. (11f)) depend linearly on p_k with the approximations Eq. (11b) and Eq. (11c). We linearize this equation by considering the update of rotation by the rotational velocity from the last time step and the additional rotational velocity $\Delta\omega=\Delta t\,\mathsf{R}_{k-1}\mathsf{C}\mathsf{p}_k$ separately:

$$\mathsf{V}_k^{\mathsf{i}} \approx e^{\Delta t[\omega_{k-1}]_\times}\mathsf{R}_{k-1}(\mathsf{x}^{\mathsf{i}}+\mathsf{u}^{\mathsf{i}})+\Delta t\Delta\omega\times(\mathsf{x}^{\mathsf{i}}+\mathsf{u}_{k-1}^{\mathsf{i}})+\mathsf{T}_k. \tag{14}$$

With this approximation, V_k depends linearly on p_k.

To handle positional constraints, we adopt a force-based constraint solve. For each constrained point, we distribute the force to the triangle vertices with barycentric weights, and the force applied at each vertex is converted to traction by dividing the force by the Voronoi area of the vertex. Then, for non-penetration and friction constraints, we formulate complementarity problems with Eq. (14) and solve for p with the projected Gauss-Seidel method similarly to Duriez et al. [DAK04; DDKA06] with staggered projections [Löt84; KSJP08]. When a fixed-position constraint is imposed, we solve Eq. (14) for p with the standard Gauss-Seidel method. When a point-joint constraint is imposed, we couple two equations from Eq. (14) by specifying the positions of two points to be the same and the force to be applied in the opposite directions with the same magnitude, and solve them similarly with Gauss-Seidel iterations. Formally, convergence of Gauss-Seidel is only guaranteed if the matrix is symmetric positive definite, and the matrices for our constraint solver are not positive definite because the matrices H_0 and G_0 are asymmetric. However, we empirically observed that the solver converges to a solution with constraints satisfied for all the problems we considered.

Using the point-joint constraints, we can implement a domain decomposition approach [BZ11; KJ11; JP02], allowing simulation of larger local displacements in addition to large global displacements. Each subdomain is simulated independently with our method, and point joint constraints are applied between adjacent subdomains to allow simulation of one large object. Since each subdomain must be bounded by a closed surface, this approach requires the addition of triangulated interfaces between the subdomains, which may be undesirable. However, it makes each subdomain's matrix and cutoff time step size n_{max} smaller, making it an interesting alternative.

4. Implementation Details

In this section, we describe our choice of quadrature methods and other details for implementation.

CQM weights and time history size. The time-domain fundamental solutions, $\mathbf{u}^*$ and $\mathbf{p}^*$, vanish after time r_{max}/c_2, where r_{max} is the maximum length between any two points on the surface mesh and $c_2=\sqrt{\mu/\rho}$ is the shear wave speed. This is because the fundamental solutions vanish after the propagation of all elastic waves. Since the numerical integration weights of the CQM represent discretization of the corresponding time-domain functions, they also vanish after a certain number of steps, leaving only $n_{max}+1$ terms each in Eq. (8). In our implementation, we set the maximum number of time history steps to be $n_{max}=\lfloor r_{max}/(c_2\Delta t)+2\rfloor$. Then, we

let $L = n_{max}$ and $\mathcal{R} = \epsilon^{\frac{1}{2n_{max}}}$ where $\epsilon = 10^{-10}$, following the work of Schanz et al. [SA97], and define the integration weight as

$$\psi_k(\hat{f}) = \frac{\mathcal{R}^{-k}}{L} \sum_{l=0}^{L-1} \hat{f}\left(\frac{\gamma(\mathcal{R}e^{il\frac{2\pi}{L}})}{\Delta t}\right) e^{-ikl\frac{2\pi}{L}}, \tag{15}$$

where γ is a characteristic function for the linear multistep method. We tested two different underlying linear multistep methods for CQM: the backward differentiation formulas of first order (BDF1) and second order (BDF2). We observed BDF2 gives too much oscillation, so we use BDF1 for all examples presented in this paper. The characteristic function for BDF1 is $\gamma(s) = 1 - s$.

Spatial quadrature method on the surface. For each vertex $\mathbf{x}_i$, in order to evaluate all the surface integrals in Eq. (8), we evaluate the integrals over each triangle on the surface and take a sum of them, i.e., $\int_\Gamma (\cdot) d\Gamma_\mathbf{y} = \sum_{m=1}^{M} \int_{\Gamma_m} (\cdot) d\Gamma_\mathbf{y}$, where Γ_m denotes m^{th} triangle. The Laplace domain fundamental solutions $\hat{\mathbf{u}}^*(\mathbf{x}, \mathbf{y}, s)$ and $\hat{\mathbf{p}}^*(\mathbf{x}, \mathbf{y}, s)$ and the Laplace domain functions for body force terms $\hat{\mathbf{d}}^*(\mathbf{x}, \mathbf{y}, s)$ and $\hat{\mathbf{q}}^*(\mathbf{x}, \mathbf{y}, s)$ have singularities at $\mathbf{y} = \mathbf{x}$. Thus, for each Γ_m, if the vertex $\mathbf{x}_i$ corresponds to one of the vertices of the triangle, the integrand is singular there. We evaluate all non-singular integrals with a 9-point Gaussian quadrature formula that is exact up to fifth degree polynomials [Cow73], after dividing the triangle into four subtriangles. We observed that the use of lower degree formulae causes instabilities. We evaluate the singular integrals in different manners depending on the order of singularity. The terms $\hat{\mathbf{u}}^*$, $\hat{\mathbf{d}}^*$ and $\hat{\mathbf{q}}^*$ behave like $\hat{\mathbf{u}}^* \sim 1/r$ as $r \to 0$, and we can remove their singularities by performing integration in polar coordinate. The term $\hat{\mathbf{p}}^*$ behaves like $\hat{\mathbf{p}}^* \sim 1/r^2$ as $r \to 0$ and the integral needs to be evaluated in the Cauchy principal value sense; we employ the numerical integration method of Guiggiani et al. [GG90]. We evaluate non-singular integrals of $\hat{\mathbf{p}}^*$ and all the singular integrals with double precision, and use single precision for the rest.

Evaluation of the integral free term. The integral free term $\mathbf{c}(\mathbf{x}_i)$ in Eq. (8) is a 3×3-matrix associated with the smoothness of the surface. If the surface is smooth at $\mathbf{x}_i$, $\mathbf{c}(\mathbf{x}_i) = \frac{1}{2}\mathbf{I}$, where $\mathbf{I}$ is an identity matrix. Since $\mathbf{x}_i$ corresponds to the positions of vertices of input mesh, the surface is not necessarily smooth at $\mathbf{x}_i$ in general. Therefore, we employ a direct computation method proposed by Mantic [Man93] to evaluate the integral free terms.

Precomputation of matrices. We can precompute the matrices in Eq. (9) because they depend only on the undeformed input mesh and the input parameters. The integration weights of CQM (Eq. (15)) are the weighted sums of Laplace domain functions. Therefore, we can first perform numerical integration of the Laplace domain functions, multiplied by the interpolation matrix if necessary, in space with all possible complex parameters to get Laplace domain matrices and form the final time-domain matrices by taking weighted sums of them. Note we can compute about half of the Laplace domain matrices by taking the conjugate of the other half of matrices. Also, we exploit the symmetry of $\hat{\mathbf{u}}^*$ to reduce the precomputation time. The precomputation of all necessary matrices took between a few minutes and a little under an hour in total for the examples we tested using a GPU implementation; we have not included the exact numbers in this paper simply because our test machine had only 16GB of RAM and disk I/O to store some temporary data was the bottleneck for large meshes.

Traction discontinuity. We use a piecewise linear interpolation function over the surface, and therefore special care must be paid to handle traction discontinuities (e.g., at geometric corners where the applied tractions may differ on one side versus the other). We use double nodes [BTW84] to handle this problem: we duplicate the vertices when there are traction discontinuities and construct matrices based on the modified mesh with discontinuous parts. With this method, G_0 becomes singular but H_0 does not, which allows us to use the constraint solver without modifications. For the sake of compression, any duplicated vertex is given a vertex index next to that of the original vertex. We apply this technique to compute the matrices for the objects in Fig. 5 and Fig. 9.

5. Results

We implemented our method using C++ and performed benchmarks on a desktop computer with an AMD Ryzen 2700X processor and 16GB of RAM using a single thread, except the scene in Fig. 1, for which we exploit object-level parallelism and a simple batch processing for frictional contact solves. Further parallelization such as matrix-vector multiplications for time history terms using a low priority thread is possible. We use a simple spatial hashing-based collision detection method. Other collision detection methods or a penalty springs approach [MZS*11] could be easily combined with our method.

Table 1 shows the computation time, memory consumption for compressed matrix storage, and relevant parameters for the scenes in this paper, and videos for all examples are available in the supplementary video.

Elastostatics vs. elastodynamics. Elastodynamic simulation produces more realistic and lively animations compared to elastostatic simulation. Fig. 4 shows how we simulate the secondary motions using an elastodynamic formulation, making our method more visually plausible and suitable for general computer animation. The elastostatic simulation is based on the work by James et al. [JP99] and we replaced its interpolation function with linear interpolation for consistency.

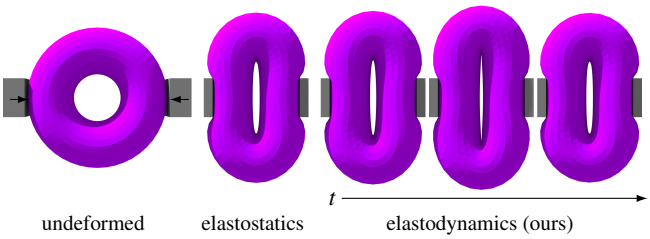


Figure 4: *Comparison between elastostatics and elastodynamics. A torus is squished from both sides by specifying displacements. Even after the boundary conditions become fixed, we obeserve the dynamic secondary motions with our method. For consistency, the body frame update is disabled for this scene because a similar method is not available for elastostatics.*

Table 1: *Performance and parameters. μ is the shear modulus (=Lamé's second parameter), and ν is Poisson's ratio. CR is the compression ratio applied to matrices computed with (#entries before compression)/(#entries kept after compression), and MR is the total memory size required to store the original or compressed matrices. When the compression ratio is not listed, the original dense matrices without compression are used. The times listed are the average time in seconds for one simulation step. UNCNST is the time to compute unconstrained motion, COLLD is the time taken for collision detection, CONST is the time required for the constraint solve, and ERRCR is the time taken for correcting errors and for processing data for the next iterations.*

scene	Fig.	N	Δt(s)	n_{max}	ρ(kg/m^3)	μ(Pa)	ν	CR	MR(GB)	UNCNST(s)	COLLD(s)	CONST(s)	ERRCR(s)
fish	1	2843×25	0.001	3	10^3	10^7	0.4	5	1.4	$4.78\cdot10^{-1}$	$2.42\cdot10^{-1}$	$1.82\cdot10^{2}$	$1.90\cdot10^{-1}$
torus	4	1152	0.001	32	$5\cdot10^2$	$5\cdot10^4$	0.48	–	6.3	$2.37\cdot10^{-1}$	–	$2.03\cdot10^{-2}$	$3.98\cdot10^{-3}$
beam (with frame update)	5	2818	0.005	8	10^3	$3\cdot10^5$	0.5	2	9.0	$5.28\cdot10^{-1}$	–	$3.05\cdot10^{0}$	$9.38\cdot10^{-2}$
beam (w/o frame update)	5	2818	0.005	8	10^3	$3\cdot10^5$	0.5	2	9.0	$5.30\cdot10^{-1}$	–	$2.51\cdot10^{0}$	$3.16\cdot10^{-2}$
bumpy cube (uncompressed)	6	3000	0.01	5	10^3	$5\cdot10^5$	0.5	–	7.8	$2.68\cdot10^{-1}$	–	$1.90\cdot10^{0}$	$7.41\cdot10^{-2}$
bumpy cube (8x compressed)	6	3000	0.01	5	10^3	$5\cdot10^5$	0.5	8	1.5	$8.55\cdot10^{-2}$	–	$2.72\cdot10^{0}$	$2.40\cdot10^{-2}$
bumpy cube (32x compressed)	6	3000	0.01	5	10^3	$5\cdot10^5$	0.5	32	0.4	$2.20\cdot10^{-2}$	–	$1.71\cdot10^{0}$	$6.55\cdot10^{-3}$
bumpy cube (128x compressed)	6	3000	0.01	5	10^3	$5\cdot10^5$	0.5	128	0.1	$6.75\cdot10^{-3}$	–	$1.15\cdot10^{0}$	$2.49\cdot10^{-3}$
bunny (friction=0.2)	8	3485	0.003	5	$3\cdot10^3$	10^6	0.3	10	1.6	$9.12\cdot10^{-2}$	$5.23\cdot10^{-2}$	$1.25\cdot10^{0}$	$2.52\cdot10^{-2}$
bunny (friction=0.4)	8	3485	0.003	5	$3\cdot10^3$	10^6	0.3	10	1.6	$9.06\cdot10^{-2}$	$5.55\cdot10^{-2}$	$6.96\cdot10^{0}$	$2.50\cdot10^{-2}$
bunny (friction=0.6)	8	3485	0.003	5	$3\cdot10^3$	10^6	0.3	10	1.6	$9.04\cdot10^{-2}$	$7.50\cdot10^{-2}$	$1.81\cdot10^{0}$	$2.50\cdot10^{-2}$
beam (with decomposition)	9	770×5	0.005	5	10^3	10^5	0.45	–	0.6	$1.14\cdot10^{-1}$	–	$6.64\cdot10^{0}$	$3.12\cdot10^{-2}$
beam (w/o decomposition)	9	2818	0.005	12	10^3	10^5	0.45	2	13.0	$7.82\cdot10^{-1}$	–	$3.05\cdot10^{0}$	$9.45\cdot10^{-2}$

Body frame update. Our body frame update effectively decouples the global translation and rotation from the local deformation. Fig. 5 shows the effectiveness of our method using the beam bending configuration. Our method alleviates the volume inflation problem significantly in this example.

Matrix compression. By applying our matrix compression technique, we can lessen the memory and computational costs. Fig. 6 gives a visual comparison between simulations with different compression ratios. The compression ratio is given by (#entries before compression)/(#entries kept after compression). Our matrix compression method successfully preserves the visual quality while significantly reducing the memory and computational costs (Table 1). For instance, applying the compression ratio of 32 reduces the memory cost by a factor of 21 and the computational cost by 22%. Fig. 7 shows that our method is most effective when both of the reordering methods for matrices are enabled, and is more efficient than naively pruning the smallest entries without wavelet transforms except when an unreasonably high compression ratio is applied.

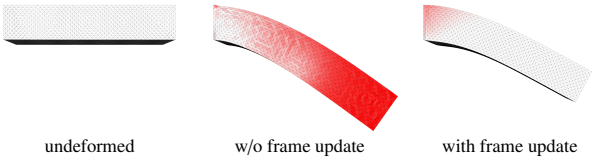


Figure 5: *Beam under gravity. Starting from an initial configuration (left), we apply gravity to the body. Without the frame update (middle), we observe a maximum of 30.4% of volume inflation due to linear elasticity's limitations. With our frame update method, the maximum volume inflation is only 1.91%. The faces are colored by the triangle areas' relative inflation rate. Red represents more inflation.*

Frictional contact. Unlike prior work [JP99; HW15; HW16], our surface-only dynamic deformables method can handle more general computer animation tasks. Fig. 8 shows how an object simulated with our method can interact with the floor with different friction coefficients. The friction from the floor causes additional dynamic motion of the object, and it comes to rest at different positions on the floor depending on the friction coefficient.

Domain decomposition. With domain decomposition, we can separately extract the global displacement for each subdomain, as illustrated in Fig. 9. Domain decomposition should be applied when one expects large local displacements, so as to circumvent the limitations of the linear model. The tradeoff is that the simulation with domain decomposition is slower due to the constraints solve iterations, and applying domain decomposition deviates from our surface-only philosophy due to the interior interfaces. Thus, all the other results in this paper are without domain decomposition.

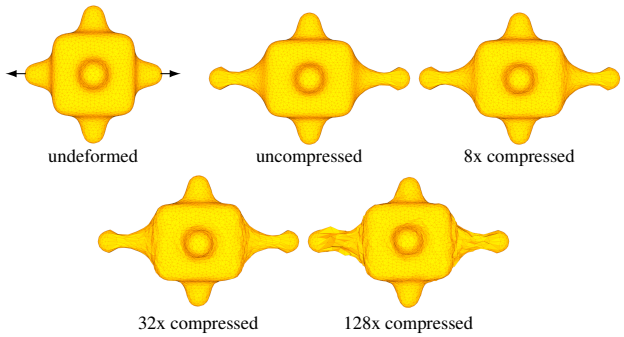


Figure 6: *Matrix compression test. We pull the two sides of bumpy cube sideways. The simulation results are shown using four different compression ratios. The compression parameters and reduction in memory and computational costs are listed in Table 1.*

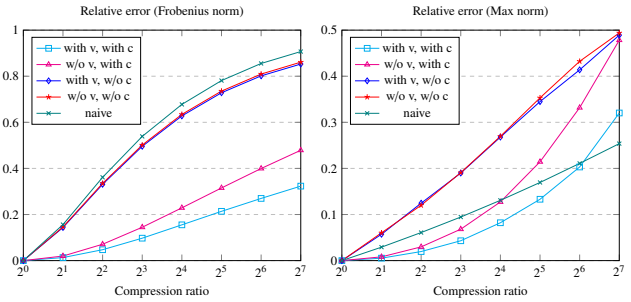


Figure 7: *Matrix reconstruction error with our compression method. The compression method is applied to the H_1 matrix of the bumpy cube scene in Fig. 6. We denote the vertex reordering by "v" and the coordinate reordering by "c" in the legends. The naive method prunes the smallest entries in the input matrix without wavelet transform. The compression errors are the smallest when we enable both reordering methods.*

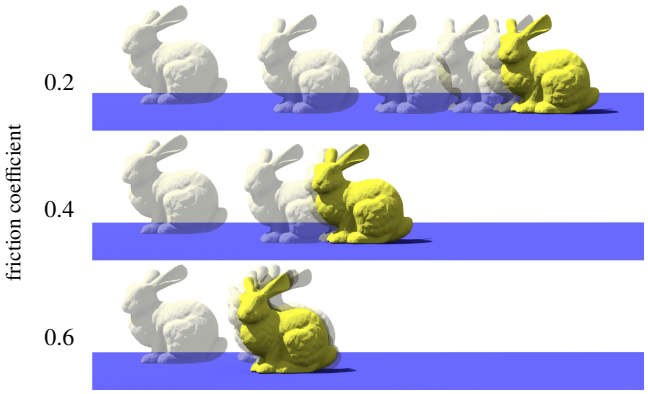


Figure 8: *Frictional contact test. The same initial velocity is applied to the object in its initial position. The elastic body behaves differently according to the friction coefficients. Transparent bunnies indicate the position at equally spaced times.*

Shared matrices. When a scene consists of objects with identical base geometry and material parameters, we only need to store one set of precomputed matrices for all the corresponding objects and can amortize the storage cost. We apply this technique in the decomposition example (Fig. 9) for its five subdomains and in the teaser scene (Fig. 1) for its many fish.

6. Conclusions and future work

We have presented a novel surface-only dynamic deformables simulation method using an elastodynamics BEM. The method requires no volumetric meshing or volumetric integral evaluations. The precomputation of surface integrals yields a series of matrices that encode the propagation of elastic waves through the body, and the translational and rotational momenta are transferred between the BEM in the moving frame and the rigid body frame to alleviate the linear elasticity artifacts. The computational and storage costs related to dense matrices are reduced using a wavelet-based compression scheme. We also demonstrated the generality of our method with several inter-element constraints. Our method still assumes modest deformations due to the underlying linear elasticity model and we omit effects due to centrifugal and Coriolis forces. Nevertheless, our work represents the first exploration of BEM-based dynamic deformable body simulation methods for animation, and we are excited to explore the many new possibilities that it raises, some of which we outline below.

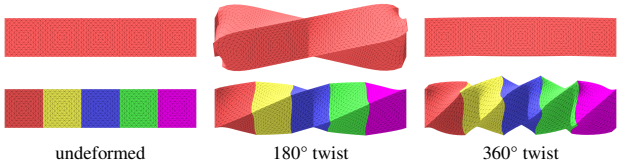


Figure 9: *Domain decomposition demonstration. We twist a beam from its undeformed configuration (left) by gradually rotating both ends. The simulation result without domain decomposition (top) fails to simulate the scene due to the limitations of linear elasticity model, and the object with domain decomposition consisting of five cubes (bottom) alleviates the problem. (More elaborate continuity constraints could help to smooth the transitions between subdomains.)*

Adaptive simulations. Our method precomputes a series of dense matrices and compresses them before use. The memory and computational costs for this precomputation are significant and grow quadratically. When a high compression ratio is used, the compression yields some errors in the simulation. In addition, the runtime costs are not yet competitive with well-studied, state-of-the-art real-time solutions. One promising direction would be to precompute the matrices only for a coarse model at the beginning of the simulation and dynamically update them according to a remeshing strategy. Surface remeshing is generally more resource-efficient than volumetric meshing; we believe this could be an efficient approach over adaptive volumetric simulation methods. The challenge would lie in keeping the time history vectors consistent when a vertex is inserted or removed.

Nonlinearity. Our method is based on a linear elasticity constitutive model. Global rigid body modes are captured by the moving frame, but it cannot simulate large local deformations. The limitation of the linear model has been recognized by the boundary element method community and is an active area of research [LMN*12]. We could incorporate additional nonlinear terms into Eq. (1) to support some other constitutive models such as a geometrically nonlinear model, but currently, when there exist nonlinear terms in the equation, they are treated similarly to body force terms and induce volumetric integrals [DRYG18]. An efficient meshless volumetric quadrature method would be the key to making the method practical.

Direct simulation on other surface representations. Due to our truly surface-only formulation (in contrast to the shape matching element method [TCL21], which uses interior quadrature points) with boundary integrals and the manner in which BEM simulates propagation of elastic waves from one point to another, it would be

easy to extend the method to objects with other surface representations besides triangle meshes, including those with vaguely defined boundaries. This includes surfaces defined with NURBS, triangle meshes with holes or overlaps, triangle soups, and point clouds. Evaluation of the boundary integrals given the available information and ensuring stability would be the hurdles.

References

[ADFG12] Aimi, Alessandra, Diligenti, Mauro, Frangi, Attilio, and Guardasoni, Chiara. "A stable 3D energetic Galerkin BEM approach for wave propagation interior problems". *Engineering Analysis with Boundary Elements* 36.12 (2012), 1756–1765. ISSN: 0955-7997. DOI: 10.1016/j.enganabound.2012.06.003 2.

[BAM86] Banerjee, Prasanta K., Ahmad, Shahid, and Manolis, George D. "Transient elastodynamic analysis of three-dimensional problems by boundary element method". *Earthquake Engineering & Structural Dynamics* 14.6 (1986), 933–949. DOI: 10.1002/eqe.4290140609 2.

[BCR91] Beylkin, Gregory, Coifman, Ronald, and Rokhlin, Vladimir. "Fast wavelet transforms and numerical algorithms I". *Communications on Pure and Applied Mathematics* 44.2 (1991), 141–183. DOI: 10.1002/cpa.3160440202 6.

[BMM17] Bender, Jan, Müller, Matthias, and Macklin, Miles. "A Survey on Position Based Dynamics, 2017". *Proceedings of the European Association for Computer Graphics: Tutorials*. EG '17. Lyon, France: Eurographics Association, 2017. DOI: 10.2312/egt.20171034 2.

[BMS12] Banjai, Lehel, Messner, Matthias, and Schanz, Martin. "Runge–Kutta convolution quadrature for the Boundary Element Method". *Computer Methods in Applied Mechanics and Engineering* 245-246 (2012), 90–101. ISSN: 0045-7825. DOI: 10.1016/j.cma.2012.07.007 2.

[BTW84] Brebbia, Carlos A., Telles, Jose C. F., and Wrobel, Luiz C. *Boundary Element Techniques: Theory and Applications in Engineering*. Berlin, Heidelberg: Springer, 1984. ISBN: 9783540124849. DOI: 10.1007/978-3-642-48860-3 7.

[BZ11] Barbič, Jernej and Zhao, Yili. "Real-Time Large-Deformation Substructuring". *ACM SIGGRAPH 2011 Papers*. SIGGRAPH '11. Vancouver, British Columbia, Canada: Association for Computing Machinery, 2011. ISBN: 9781450309431. DOI: 10.1145/1964921.1964986 6.

[Cow73] Cowper, G. R. "Gaussian quadrature formulas for triangles". *International Journal for Numerical Methods in Engineering* 7.3 (1973), 405–408. DOI: 10.1002/nme.1620070316 7.

[CR68] Cruse, Thommas A. and Rizzo, Frank J. "A direct formulation and numerical solution of the general transient elastodynamic problem. I". *Journal of Mathematical Analysis and Applications* 22.1 (1968), 244–259. ISSN: 0022-247X. DOI: 10.1016/0022-247X(68)90171-6 12.

[DAK04] Duriez, Christian, Andriot, Claude, and Kheddar, Abderrahmane. "Signorini's contact model for deformable objects in haptic simulations". *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. Vol. 4. Sendai, Japan: IEEE, 2004, 3232–3237 vol.4. DOI: 10.1109/IROS.2004.1389915 6.

[DDKA06] Duriez, Christian, Dubois, Frederic, Kheddar, Abderrahmane, and Andriot, Claude. "Realistic Haptic Rendering of Interacting Deformable Objects in Virtual Environments". *IEEE Transactions on Visualization and Computer Graphics* 12.1 (Jan. 2006), 36–47. ISSN: 1077-2626. DOI: 10.1109/TVCG.2006.13 6.

[DHB*16] Da, Fang, Hahn, David, Batty, Christopher, et al. "Surface-Only Liquids". *ACM Trans. Graph.* 35.4 (July 2016). ISSN: 0730-0301. DOI: 10.1145/2897824.2925899 1, 2.

[DJ18] De Goes, Fernando and James, Doug L. "Dynamic Kelvinlets: Secondary Motions Based on Fundamental Solutions of Elastodynamics". *ACM Trans. Graph.* 37.4 (July 2018). ISSN: 0730-0301. DOI: 10.1145/3197517.3201280 2.

[DRYG18] Deng, Yani, Rong, Junjie, Ye, Wenjing, and Gray, Lesley J. "An efficient grid-based direct-volume integration BEM for 3D geometrically nonlinear elasticity". *Computational Mechanics* 62.4 (2018), 603–616. DOI: 10.1007/s00466-017-1515-z 9.

[GG90] Guiggiani, Massimo and Gigante, A. "A General Algorithm for Multidimensional Cauchy Principal Value Integrals in the Boundary Element Method". *Journal of Applied Mechanics* 57.4 (Dec. 1990), 906–915. ISSN: 0021-8936. DOI: 10.1115/1.2897660 7.

[GS18] Gwinner, Joachim and Stephan, Ernst Peter. *Advanced Boundary Element Methods. Treatment of Boundary Value, Transmission and Contact Problems*. 1st ed. Springer Series in Computational Mathematics, 52. Springer International Publishing, 2018. ISBN: 3-319-92001-4. DOI: 10.1007/978-3-319-92001-6 6.

[HM20] Huang, Libo and Michels, Dominik L. "Surface-Only Ferrofluids". *ACM Trans. Graph.* 39.6 (Nov. 2020). ISSN: 0730-0301. DOI: 10.1145/3414685.3417799 2.

[HW15] Hahn, David and Wojtan, Chris. "High-Resolution Brittle Fracture Simulation with Boundary Elements". *ACM Trans. Graph.* 34.4 (July 2015). ISSN: 0730-0301. DOI: 10.1145/2766896 1, 2, 8.

[HW16] Hahn, David and Wojtan, Chris. "Fast Approximations for Boundary Element Based Brittle Fracture Simulation". *ACM Trans. Graph.* 35.4 (July 2016). ISSN: 0730-0301. DOI: 10.1145/2897824.2925902 1–3, 5, 6, 8.

[HZGJ19] Hu, Yuanming, Zhang, Xinxin, Gao, Ming, and Jiang, Chenfanfu. "On Hybrid Lagrangian-Eulerian Simulation Methods: Practical Notes and High-Performance Aspects". *ACM SIGGRAPH 2019 Courses*. SIGGRAPH '19. Los Angeles, California: Association for Computing Machinery, 2019. ISBN: 9781450363075. DOI: 10.1145/3305366.3328075 2.

[JBP06] James, Doug L., Barbič, Jernej, and Pai, Dinesh K. "Precomputed Acoustic Transfer: Output-Sensitive, Accurate Sound Generation for Geometrically Complex Vibration Sources". *ACM Trans. Graph.* 25.3 (July 2006), 987–995. ISSN: 0730-0301. DOI: 10.1145/1141911.1141983 2.

[JP02] James, Doug L. and Pai, Dinesh K. "Real time simulation of multizone elastokinematic models". *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*. Vol. 1. 2002, 927–932 vol.1. DOI: 10.1109/ROBOT.2002.1013475 2, 6.

[JP03] James, Doug L. and Pai, Dinesh K. "Multiresolution Green's Function Methods for Interactive Simulation of Large-Scale Elastostatic Objects". *ACM Trans. Graph.* 22.1 (Jan. 2003), 47–82. ISSN: 0730-0301. DOI: 10.1145/588272.588278 5.

[JP99] James, Doug L. and Pai, Dinesh K. "ArtDefo: Accurate Real Time Deformable Objects". *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '99. USA: ACM Press/Addison-Wesley Publishing Co., 1999, 65–72. ISBN: 0201485605. DOI: 10.1145/311535.311542 1–4, 6–8.

[JST*16] Jiang, Chenfanfu, Schroeder, Craig, Teran, Joseph, et al. "The Material Point Method for Simulating Continuum Materials". *ACM SIGGRAPH 2016 Courses*. SIGGRAPH '16. Anaheim, California: Association for Computing Machinery, 2016. ISBN: 9781450342896. DOI: 10.1145/2897826.2927348 2.

[KBST19] Koschier, Dan, Bender, Jan, Solenthaler, Barbara, and Teschner, Matthias. "Smoothed Particle Hydrodynamics Techniques for the Physics Based Simulation of Fluids and Solids". *Eurographics 2019 - Tutorials*. Ed. by Jakob, Wenzel and Puppo, Enrico. Genova, Italy: The Eurographics Association, 2019. DOI: 10.2312/egt.20191035 2.

[KJ11] Kim, Theodore and James, Doug L. "Physics-Based Character Skinning Using Multi-Domain Subspace Deformations". *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '11. Vancouver, British Columbia, Canada: Association for Computing Machinery, 2011, 63–72. ISBN: 9781450309233. DOI: 10.1145/2019406.2019415 6.

[KS15] Kager, Bernhard and Schanz, Martin. "Fast and data sparse time domain BEM for elastodynamics". *Engineering Analysis with Boundary Elements* 50 (2015), 212–223. ISSN: 0955-7997. DOI: 10.1016/j.enganabound.2014.08.001 5.

[KSJP08] Kaufman, Danny M., Sueda, Shinjiro, James, Doug L., and Pai, Dinesh K. "Staggered Projections for Frictional Contact in Multibody Systems". *ACM Trans. Graph.* 27.5 (Dec. 2008). issn: 0730-0301. doi: 10.1145/1409060.1409117 6.

[LMN*12] Liu, Yijun, Mukherjee, Subrata, Nishimura, Naoshi, et al. "Recent Advances and Emerging Applications of the Boundary Element Method". *Applied Mechanics Reviews* 64.3 (Mar. 2012). issn: 0003-6900. doi: 10.1115/1.4005491 1, 9.

[Löt84] Lötstedt, Per. "Numerical Simulation of Time-Dependent Contact and Friction Problems in Rigid Body Mechanics". *SIAM J. Sci. Stat. Comput.* 5.2 (June 1984), 370–393. issn: 0196-5204. doi: 10.1137/0905028 6.

[LSSW12] Langer, Ulrich, Schanz, Martin, Steinbach, Olaf, and Wendland, Wolfgang L., eds. *Fast Boundary Element Methods in Engineering and Industrial Applications*. 1st ed. Vol. 63. Lecture Notes in Applied and Computational Mechanics. Springer, 2012. isbn: 978-3-642-25669-1. doi: 10.1007/978-3-642-25670-7 5.

[Lub88a] Lubich, Christian. "Convolution quadrature and discretized operational calculus. I". *Numerische Mathematik* 52.2 (1988), 129–145. doi: 10.1007/BF01398686 4.

[Lub88b] Lubich, Christian. "Convolution quadrature and discretized operational calculus. II". *Numerische Mathematik* 52.4 (1988), 413–425. doi: 10.1007/BF01462237 4.

[LW16] Levi, Zohar and Weber, Ofir. "On the Convexity and Feasibility of the Bounded Distortion Harmonic Mapping Problem". *ACM Trans. Graph.* 35.4 (July 2016). issn: 0730-0301. doi: 10.1145/2897824.2925929 2.

[LZ13] Li, Yuan and Zhang, Jianming. "A comparative study of time domain BEM for 3D elastodynamic analysis". *WIT Transactions on Modelling and Simulation* 56 (2013), 503–514. doi: 10.2495/BEM360411 2.

[Man93] Mantic, Vladislav. "A new formula for the C-matrix in the Somigliana identity". *Journal of Elasticity* 33.3 (1993), 191–201. doi: 10.1007/BF00043247 7.

[MB88] Manolis, George D. and Beskos, Dimitrios E. *Boundary Element Methods in Elastodynamics*. London: Spon Press, 1988. isbn: 0046200193 2.

[MDM*02] Müller, Matthias, Dorsey, Julie, McMillan, Leonard, et al. "Stable Real-Time Deformations". *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '02. San Antonio, Texas: Association for Computing Machinery, 2002, 49–54. isbn: 1581135734. doi: 10.1145/545261.545269 3.

[MG04] Müller, Matthias and Gross, Markus. "Interactive Virtual Materials". *Proceedings of Graphics Interface 2004*. GI '04. London, Ontario, Canada: Canadian Human-Computer Communications Society, 2004, 239–246. isbn: 1568812272. url: https://dl.acm.org/doi/10.5555/1006058.1006087 3.

[MHHR07] Müller, Matthias, Heidelberger, Bruno, Hennix, Marcus, and Ratcliff, John. "Position Based Dynamics". *J. Vis. Comun. Image Represent.* 18.2 (Apr. 2007), 109–118. issn: 1047-3203. doi: 10.1016/j.jvcir.2007.01.005 2.

[MHTG05] Müller, Matthias, Heidelberger, Bruno, Teschner, Matthias, and Gross, Markus. "Meshless Deformations Based on Shape Matching". *ACM Trans. Graph.* 24.3 (July 2005), 471–478. issn: 0730-0301. doi: 10.1145/1073204.1073216 2, 5.

[Mir96] Mirtich, Brian. "Fast and Accurate Computation of Polyhedral Mass Properties". *J. Graph. Tools* 1.2 (Feb. 1996), 31–50. issn: 1086-7651. doi: 10.1080/10867651.1996.10487458 5.

[MS10] Messner, Matthias and Schanz, Martin. "An accelerated symmetric time-domain boundary element formulation for elasticity". *Engineering Analysis with Boundary Elements* 34.11 (2010), 944–955. issn: 0955-7997. doi: 10.1016/j.enganabound.2010.06.007 5.

[MZS*11] McAdams, Aleka, Zhu, Yongning, Selle, Andrew, et al. "Efficient Elasticity for Character Skinning with Contact and Collisions". *ACM Trans. Graph.* 30.4 (July 2011). issn: 0730-0301. doi: 10.1145/2010324.1964932 3, 7.

[NB89] Nowak, Andrzej J. and Brebbia, Carlos A. "The multiple-reciprocity method. A new approach for transforming BEM domain integrals to the boundary". *Engineering Analysis with Boundary Elements* 6.3 (1989), 164–167. issn: 0955-7997. doi: 10.1016/0955-7997(89)90032-5 4.

[PS97] Peirce, Anthony and Siebrits, Eduard. "Stability analysis and design of time-stepping schemes for general elastodynamic boundary element models". *International Journal for Numerical Methods in Engineering* 40.2 (1997), 319–342. doi: 10.1002/(SICI)1097-0207(19970130)40:2<319::AID-NME67>3.0.CO;2-I 2.

[SA97] Schanz, Martin and Antes, Heinz. "A new visco- and elastodynamic time domain Boundary Element formulation". *Computational Mechanics* 20 (Jan. 1997), 452–459. doi: 10.1007/s004660050265 2, 3, 7.

[SB12] Sifakis, Eftychios and Barbic, Jernej. "FEM Simulation of 3D Deformable Solids: A Practitioner's Guide to Theory, Discretization and Model Reduction". *ACM SIGGRAPH 2012 Courses*. SIGGRAPH '12. Los Angeles, California: Association for Computing Machinery, 2012. isbn: 9781450316781. doi: 10.1145/2343483.2343501 2.

[Sch01] Schanz, Martin. *Wave propagation in viscoelastic and poroelastic continua: a boundary element approach*. Vol. 2. Berlin, Heidelberg: Springer, 2001. doi: 10.1007/978-3-540-44575-3 2, 3.

[SVB17] Solomon, Justin, Vaxman, Amir, and Bommes, David. "Boundary Element Octahedral Fields in Volumes". *ACM Trans. Graph.* 36.4 (May 2017). issn: 0730-0301. doi: 10.1145/3072959.3065254 2.

[TBNF03] Teran, Joseph, Blemker, Silvia, Ng-Thow-Hing, Victor, and Fedkiw, Ronald. "Finite Volume Methods for the Simulation of Skeletal Muscle". *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '03. San Diego, California: Eurographics Association, 2003, 68–74. isbn: 1581136595. url: https://dl.acm.org/doi/10.5555/846276.846285 2.

[TCL21] Trusty, Ty, Chen, Honglin, and Levin, David I. W. "The Shape Matching Element Method: Direct Animation of Curved Surface Models". *ACM Trans. Graph.* 40.4 (July 2021). issn: 0730-0301. doi: 10.1145/3450626.3459772 9.

[TPBF87] Terzopoulos, Demetri, Platt, John, Barr, Alan, and Fleischer, Kurt. "Elastically Deformable Models". *SIGGRAPH Comput. Graph.* 21.4 (Aug. 1987), 205–214. issn: 0097-8930. doi: 10.1145/37402.37427 2.

[TW88] Terzopoulos, D. and Witkin, A. "Physically based models with rigid and deformable components". *IEEE Computer Graphics and Applications* 8.6 (1988), 41–51. doi: 10.1109/38.20317 3.

[UPSW16] Umetani, Nobuyuki, Panotopoulou, Athina, Schmidt, Ryan, and Whiting, Emily. "Printone: Interactive Resonance Simulation for Free-Form Print-Wind Instrument Design". *ACM Trans. Graph.* 35.6 (Nov. 2016). issn: 0730-0301. doi: 10.1145/2980179.2980250 2.

[WSSK13] Wang, He, Sidorov, Kirill A., Sandilands, Peter, and Komura, Taku. "Harmonic Parameterization by Electrostatics". *ACM Trans. Graph.* 32.5 (Oct. 2013). issn: 0730-0301. doi: 10.1145/2503177 2.

[YLPM05] Yoon, Sung-Eui, Lindstrom, Peter, Pascucci, Valerio, and Manocha, Dinesh. "Cache-Oblivious Mesh Layouts". *ACM SIGGRAPH 2005 Papers*. SIGGRAPH '05. Los Angeles, California: Association for Computing Machinery, 2005, 886–893. isbn: 9781450378253. doi: 10.1145/1186822.1073278 6.

[ZBG15] Zhu, Yufeng, Bridson, Robert, and Greif, Chen. "Simulating Rigid Body Fracture with Surface Meshes". *ACM Trans. Graph.* 34.4 (July 2015). issn: 0730-0301. doi: 10.1145/2766942 2.

Appendix A: Laplace domain fundamental solutions and surface-only body force terms

The Laplace domain fundamental solutions $\hat{\mathbf{u}}^*$ and $\hat{\mathbf{p}}^*$ appear in the work of Cruse et al. [CR68]. We present them here for the readers' convenience. We use a non-bold font with subscripts to denote components of a vector or matrix in this appendix. The Laplace domain displacement fundamental solution for elastodynamics is

$$\hat{u}^*_{ij}(\mathbf{x},\mathbf{y},s) = \frac{1}{4\pi\rho}\left\{\left(\frac{3r_ir_j}{r^5}-\frac{\delta_{ij}}{r^3}\right)\left(\frac{s\frac{r}{c_1}+1}{s^2}e^{-\frac{r}{c_1}s}-\frac{s\frac{r}{c_2}+1}{s^2}e^{-\frac{r}{c_2}s}\right) + \frac{r_ir_j}{r^3}\left(\frac{1}{c_1^2}e^{-\frac{r}{c_1}s}-\frac{1}{c_2^2}e^{-\frac{r}{c_2}s}\right)+\frac{\delta_{ij}}{rc_2^2}e^{-\frac{r}{c_2}s}\right\}, \tag{16}$$

and the Laplace domain traction fundamental solution for elastodynamics is:

$$\begin{aligned}
&\hat{p}^*_{ij}(\mathbf{x},\mathbf{y},s)\\
&=\frac{1}{4\pi}\left\{\frac{6c_2^2}{r^3}\left(r_in_j+r_jn_i+\left(\delta_{ij}-5\frac{r_ir_j}{r^2}\right)\mathbf{r}\cdot\mathbf{n}\right)\right.\\
&\quad\left[\frac{e^{-\frac{r}{c_1}s}}{c_1^2}\left(\frac{c_1}{rs}+\frac{c_1^2}{r^2s^2}\right)-\frac{e^{-\frac{r}{c_2}s}}{c_2^2}\left(\frac{c_2}{rs}+\frac{c_2^2}{r^2s^2}\right)\right]\\
&\quad+\frac{e^{-\frac{r}{c_1}s}}{r^3c_1^2}\left[2c_2^2\left(2r_in_j+r_jn_i-\left(\frac{6r_ir_j}{r^2}-\delta_{ij}\right)\mathbf{r}\cdot\mathbf{n}\right)-c_1^2r_in_j\right]\\
&\quad+\frac{e^{-\frac{r}{c_2}s}}{r^3}\left[\frac{12r_ir_j\mathbf{r}\cdot\mathbf{n}}{r^2}-2r_in_j-3r_jn_i-3\delta_{ij}\mathbf{r}\cdot\mathbf{n}\right]\\
&\quad-\frac{e^{-\frac{r}{c_1}s}s}{r^2c_1^3}\left[c_1^2r_in_j+2c_2^2\left(\frac{r_ir_j\mathbf{r}\cdot\mathbf{n}}{r^2}-r_in_j\right)\right]\\
&\quad\left.+\frac{e^{-\frac{r}{c_2}s}s}{r^2c_2}\left[\frac{2r_ir_j\mathbf{r}\cdot\mathbf{n}}{r^2}-\delta_{ij}\mathbf{r}\cdot\mathbf{n}-r_jn_i\right]\right\},
\end{aligned} \tag{17}$$

where $\mathbf{r}=\mathbf{y}-\mathbf{x}$, $r=\sqrt{r_1^2+r_2^2+r_3^2}$, $\mathbf{n}$ is the outward unit normal at $\mathbf{y}$, δ_{ij} is Kronecker's delta, and c_1 and c_2 are the longitudinal and shear wave speeds, respectively, computed from the material parameters;

$$c_1=\sqrt{\frac{\lambda+2\mu}{\rho}},\quad c_2=\sqrt{\frac{\mu}{\rho}}. \tag{18}$$

The newly derived surface-only body force terms are as follows, and we provide the detailed derivations in the supplementary note. The Laplace domain function for fictitious force term due to translational acceleration is

$$\hat{d}^*_{ij}(\mathbf{x},\mathbf{y},s) = -\frac{1}{4\pi}\left\{\frac{r_in_j}{r^3s^2}\left[\left(s\frac{r}{c_2}+1\right)e^{-\frac{r}{c_2}s}-\left(s\frac{r}{c_1}+1\right)e^{-\frac{r}{c_1}s}\right] + \frac{\delta_{ij}\,\mathbf{r}\cdot\mathbf{n}}{r^3s^2}\left[1-\left(s\frac{r}{c_2}+1\right)e^{-\frac{r}{c_2}s}\right]\right\}, \tag{19}$$

and the Laplace domain function for the Euler force term is

$$\hat{\mathbf{q}}^* = \hat{\mathbf{l}}^* + \hat{\mathbf{d}}^*[\mathbf{x}-\bar{\mathbf{x}}]^T_\times, \tag{20}$$

where

$$\begin{aligned}
&\hat{l}^*_{ij}(\mathbf{x},\mathbf{y},s)\\
&=-\frac{1}{4\pi}\left\{\frac{r_ir_{j+1}n_{j+2}-r_ir_{j+2}n_{j+1}}{r^3s^2}\left[\left(s\frac{r}{c_2}+1\right)e^{-\frac{r}{c_2}s}-\left(s\frac{r}{c_1}+1\right)e^{-\frac{r}{c_1}s}\right]\right.\\
&\qquad\left.+\frac{\delta_{i(j+1)}n_{j+2}-\delta_{i(j+2)}n_{j+1}}{c_2s}e^{-\frac{r}{c_2}s}\right\}.
\end{aligned} \tag{21}$$