Eulerian-on-Lagrangian Simulation

Ye Fan, Joshua Litven, David I.W. Levin, and Dinesh K. Pai
Department of Computer Science, University of British Columbia[1]

We describe an Eulerian-on-Lagrangian solid simulator that reduces or eliminates many of the problems experienced by fully Eulerian methods but retains its advantages. Our method does not require the construction of an explicit object discretization and the fixed nature of the simulation mesh avoids tangling during large deformations. By introducing Lagrangian modes to the simulation we enable unbounded simulation domains and reduce the time-step restrictions which can plague Eulerian simulations. Our method features a new solver that can resolve contact between multiple objects while simultaneously distributing motion between the Lagrangian and Eulerian modes in a least-squares fashion. Our method successfully bridges the gap between Lagrangian and Eulerian simulation methodologies without having to abandon either one.

Categories and Subject Descriptors: I.6.8 [**Simulation and Modeling**]: Types of Simulation

Additional Key Words and Phrases: Deformation, Continuum Mechanics, Rigid Bodies, Eulerian, Contact

1. INTRODUCTION

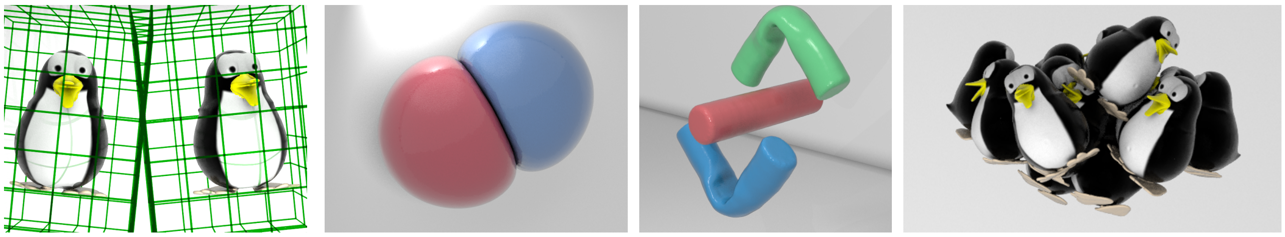
Eulerian simulations have proven to be indispensable tools in the computer graphics field. They are used to produce stunning animations of complicated fluids, and more recently, of solids and granular materials. Because these simulations rely on a fixed spatial discretization they avoid many of the perils encountered by Lagrangian simulation of highly deformable objects. However, these methods suffer from inherent drawbacks stemming from the fixed spatial discretization which is their defining characteristic. Chief amongst these is the necessary discretization of the entire simulation domain, difficulties in computing accurate and dissipation free advection, and potential time-step restrictions. Many algorithms attempt to avoid these issues by replacing parts, or all of the simulation with Lagrangian methods, but this is done at the peril of losing the advantages of the Eulerian approach. In this paper we present a clean, hybrid technique which allows us to leverage the benefits of Lagrangian and Eulerian simulators by treating rigid and low frequency modes of a deformable object as Lagrangian and the high frequency deformable modes as Eulerian. We will show that this approach alleviates or significantly reduces all of the problems listed above but still allows us to resolve large deformations conveniently in an Eulerian context. We also detail a contact-aware solver for these types of mixed Eulerian-Lagrangian simulations. See Figure 1 for a preview of the results.

2. RELATED WORK

Eulerian simulations have become commonplace in computer graphics, especially in fluids simulation. Eulerian simulations have been used to simulate melting solids [Carlson et al. 2002], viscoelastic fluids [Goktekin et al. 2004; Losasso et al. 2006] and elastic or elasto-plastic solids [Sulsky et al. 1994; Trangenstein 1994; Miller and Colella 2001; Tran and Udaykumar 2004; Banks et al. 2007; Barton and Drikakis 2010; Kamrin and Nave 2009; Levin et al. 2011; Kamrin et al. 2012]. However, Eulerian simulations suffer from several inherent drawbacks such as spatio-temporal time step limitations [Osher and Fedkiw 2002], dissipation caused by certain advection schemes [Stam 1999] and the necessity of defining a computational domain that extends throughout the simulation domain [Foster and Metaxas 1996]. For Eulerian fluid simulations these shortcomings have been addressed by using more complicated, conservative advection schemes [Lentine et al. 2011] and data structures [Wang et al. 2005]. An alternate approach involves replacing the Eulerian computational machinery with Lagrangian analogs [Premože et al. 2003]. However in theses cases we are faced with losing the advantages conferred by Eulerian simulation. Methods which maintain Eulerian and Lagrangian characteristics simultaneously can prove to be useful alternatives to these either-or approaches. Such methods have been used effectively for simulation of constrained, one-dimensional strands [Sueda et al. 2011], dissipation free advection [Zhu and Bridson 2005], robust pressure-projection in particle-based fluid simulations [Narain et al. 2010; Raveendran et al. 2011] and as a means of solid fluid coupling in finite-element simulations [Belytschko and Kennedy 1978]. Additionally, methods which translate [Shah et al. 2004] or rotate and scale [Poludnenko and Khokhlov 2007] the simulation domain have been used in fluid simulation. In this paper we will take a different approach from the methods listed above. Instead of assigning algorithmic steps or discrete points to be Lagrangian or Eulerian we will imbue generalized configuration variables with these characteristics via a sequence of mappings. This will allow us to assign Lagrangian and Eulerian behavior to the different modes of a motion (not just to the nodes of a domain). We will give two specific examples of Eulerian-on-Lagrangian methods but note that the algorithm described allows a completely general decomposition to be applied. As opposed to the Arbitrary-Lagrangian-Eulerian method [Belytschko et al. 2000] (ALE) and Lagrangian methods which use remeshing to avoid ill-conditioned elements [Bargteil et al. 2007; Wicke et al. 2010], our method describes a principled numerical "glue" that can be used to merge Eulerian and Lagrangian Simulations, both described using arbitrary generalized coordinates, and inherit the advantages of both schemes.

2.1 Contributions

Our main contribution is an Eulerian-on-Lagrangian simulator for deformable elastic and elastoplastic solids, which decomposes the motion of objects using both Lagrangian and Eulerian configuration variables. Our second contribution is a novel solver that automatically resolves the inherent redundancy of this decomposition in an optimal fashion by projecting momentum onto the low dimensional Lagrangian space while simultaneously resolving contact constraints acting on the system. Our third contribution is a method for simulating plasticity without smoothing of the plastic deformation in an Eulerian context but that avoids remeshing as is used in previous methods [Bargteil et al. 2007; Wicke et al. 2010]. The presented method also features a dynamic time stepping scheme which maximizes the time step of the Eulerian integrator while attempting to guarantee that the stability requirement of the Eulerian advection stage will be met in the presence of con-



(a) Eulerian-on-Lagrangian Grids (b) Large Deformation (c) Plasticity (d) Multiple Objects

Fig. 1: (a) The Eulerian-on-Lagrangian method embeds an Eulerian solid simulator in a Lagrangian grid. (b) Elastic spheres undergo a collision with large deformations, and return to precise rest shapes. (c) Two plastic cylinders after colliding with a rigid rod. (d) Complex contact between several Eulerian-on-Lagrangian objects.

tact. The end result is a simulator that can adaptively function as a Lagrangian simulator, a fully Eulerian simulator or a combination of the two.

3. METHODS

For the reader's convenience we will begin by outlining the notational conventions used in this paper (Table I). With a slight abuse of notation we denote both nodal variables for single elements as well as entire objects by the same bold face letters, and the meaning will be evident from the context. When necessary we denote the stacked vector of all configuration variables as $\boldsymbol{q}$.

Table I. : Notation used

Notation	Definition
f	scalar or 3-vector
$\boldsymbol{f}$	$n \times 1$ vector resulting from stacking all f
$\overset{\bullet}{f}$	$\frac{df}{dt}$ in spatial domain
$\dot{f}$	$\frac{df}{dt}$ in intermediate domain
$[f]$	cross product matrix: $\boldsymbol{f}\times$

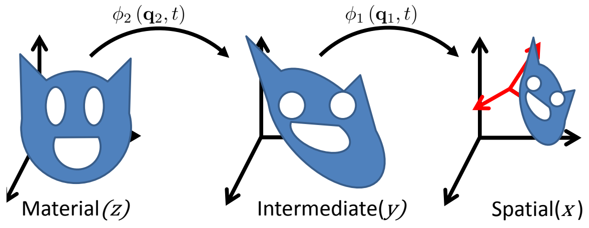

$\phi_2\left(\mathbf{q}_2, t\right)$
$\phi_1\left(\mathbf{q}_1, t\right)$
Material (z)
Intermediate(y)
Spatial(x)

Fig. 2: The continuous domains used for the Eulerian-on-Lagrangian simulator as well as the discrete grid structure stores in the Eulerian domain. The mapping between the domains is described using a set of generalized configuration variables.

Our exposition begins by describing three separate continuous domains: The material domain, the intermediate domain (also known as the reference domain) and the spatial domain (Figure 2). We define invertible mappings ϕ_i between these spaces and parameterize these mappings using generalized configuration variables $\boldsymbol{q_i}$. ϕ_2 maps reference objects into their deformed state due to Eulerian motion and ϕ_1 maps deformed objects into space via Lagrangian motion. The use of configuration variables is key to our approach since it allows us to build in any Lagrangian motion using a reduced set of coordinates. In practice we compute ϕ_2^{-1} by advecting material coordinates on a uniform grid located in the intermediate domain. In order to elucidate the mechanics involved, we begin with the pedagogical case of a rigid Lagragangian motion.

3.1 Rigid Motion Eulerian-on-Lagrangian Simulation

To begin we define mappings ϕ_1 and ϕ_2 as

$$\begin{aligned} \phi_1 : \quad x &= \boldsymbol{R}\left(y+p\right) \\ \phi_2 : \quad y &= z + u\left(z,t\right), \end{aligned} \tag{1}$$

where $\boldsymbol{R}$ and p are a rotation and translation (in the intermediate space) that map y to x, and u is the displacement of a material point at z to its deformed position y. Thus ϕ_1 represents the rigid motion (Lagrangian), while ϕ_2 represents the deformation (Eulerian). By taking the time derivative of the composition of these functions we arrive at the spatial velocity of any point in the material domain

$$\overset{\bullet}{x} = \boldsymbol{R}\left(\left[y\right]^T \omega + v + \nu\right), \tag{2}$$

where $[y]$ is the cross product matrix, ω is an angular velocity and ν is a linear velocity of the rigid motion and $v = \dot{u}$ which results from Equation 1. In the intermediate space u is a function of $y(t)$ and so $\dot{u}$ corresponds to the material derivative, $\frac{Du}{Dt}$. This gives rise to the Eulerian-Lagrangian velocity

$$\overset{\bullet}{x} = \boldsymbol{R}\left(\left[y\right]^T \omega + \frac{Du}{Dt} + \nu\right). \tag{3}$$

We take the Lagrangian configuration variables, $\boldsymbol{q}_1$, to be the axis-angle of rotation θ, i.e., $\boldsymbol{R} = \exp[\theta]$, and position p (as seen in y) of the rigid frame in which the Eulerian simulation is embedded.

The displacements are then discretized in the intermediate domain, y; this is in contrast with the material domain in the Lagrangian approach and the spatial domain in the Eulerian approach. We use a hexahedral finite element grid with trilinear shape functions; the nodal displacements become our configuration variables $\boldsymbol{q}_2$. For a single element this yields

$$\boldsymbol{q} = \begin{pmatrix} \theta \\ p \\ \mathbf{u} \end{pmatrix} \quad \dot{\boldsymbol{q}} = \begin{pmatrix} \omega \\ \nu \\ \boldsymbol{v} \end{pmatrix}. \tag{4}$$

Again we note that the total time derivative of the Eulerian velocity implies the application of the material derivative. In our method this is computed using an independant advection step (as is usually done in fluid mechanics, see Bridson [2008] for a good description). Other methods bake this material derivative into the equations of motion (see Sueda *et al.* [2011] for a depiction of such an approach).

For the i^{th} element we can define the Lagrangian function of our system as

$$L = \frac{1}{2}\dot{\boldsymbol{q}}^T \boldsymbol{M}^i \dot{\boldsymbol{q}} - V(\boldsymbol{q}) \tag{5}$$

where

$$\boldsymbol{M}^i = \int_{\Omega^i} \rho \bar{J} \begin{pmatrix} [y]\,[y]^T & [y] & [y]\,\boldsymbol{N}^i \\ [y]^T & \boldsymbol{I} & \boldsymbol{N}^i \\ \boldsymbol{N}^{iT}\,[y]^T & \boldsymbol{N}^{iT} & \boldsymbol{N}^{iT}\boldsymbol{N}^i \end{pmatrix} d\Omega^i, \tag{6}$$

ρ is the density, $\bar{J}$ is the deformation gradient determinant, $\boldsymbol{N}^i$ is the matrix of element shape functions and V is a potential energy accounting for all integrable forces. We evaluate these integrals numerically [Belytschko and Moran 2000]. It is convenient to think of this mass matrix in block form in which the diagonal blocks are the Lagrangian and Eulerian mass matrices while the off-diagonal blocks are coupling matrices (Figure 3). Note that, as in a reference coordinate formulation, the Eulerian displacements can be used to directly compute strain and avoid drift away from the object's rest configuration. We pause here to offer a second derivation which demonstrates the general nature of the method.

3.2 Linear Modal Eulerian-on-Lagrangian Simulation

In order to show how other Lagrangian discretizations can be utilized we will now repeat the process above for a more general case in which the Lagrangian motion is represented using linear modal models. We again begin by defining our mappings for an arbitrary mesh element, i:

$$\begin{aligned} \phi_1 :\quad & x = \boldsymbol{N}(y)\,\boldsymbol{U}\boldsymbol{q}_1 \\ \phi_2 :\quad & y = z + u(z,t), \end{aligned} \tag{7}$$

where $\boldsymbol{N}$ is a matrix of shape functions, $\boldsymbol{U}$ is the matrix of linear modes for the element (comprised of the appropriate rows from the global linear deformation basis), and $\boldsymbol{q}_1$ are the degrees of freedom for the modal model. The spatial position of a given reference point is then

$$x = \boldsymbol{N}(z + u(z,t)))\,\boldsymbol{U}\boldsymbol{q}_1 \tag{8}$$

and the velocity can be computed as

$$\overset{\bullet}{x} = \boldsymbol{F}_1 v + \boldsymbol{N}\boldsymbol{U}\dot{\boldsymbol{q}}_1 \tag{9}$$

where $\boldsymbol{F}_1$ is the Lagrangian deformation gradient, given by $\nabla_y\left(\boldsymbol{q}_1^T\boldsymbol{U}^T\boldsymbol{N}^T\right)$ and v is (as before) the intermediate space Eulerian velocity. In this case we choose our generalized coordinates to be $\boldsymbol{q}_1$ and the intermediate space, Eulerian displacements $\boldsymbol{u}$. We form the Lagrangian of the system and write the per-element mass matrix, which in this case becomes

$$\boldsymbol{M}^i = \int_{\Omega^i} \rho \bar{J} \begin{pmatrix} \boldsymbol{U}^T\boldsymbol{N}^T\boldsymbol{N}\boldsymbol{U} & \boldsymbol{U}^T\boldsymbol{N}^T\boldsymbol{F}_1\boldsymbol{N} \\ \boldsymbol{N}^T\boldsymbol{F}_1^T\boldsymbol{N}\boldsymbol{U} & \boldsymbol{N}^T\boldsymbol{F}_1^T\boldsymbol{F}_1^T\boldsymbol{N} \end{pmatrix} d\Omega^i. \tag{10}$$

3.3 The Momentum Equation

We note that both the rigid body and linear modal element mass matrices share a similar block structure. In fact this structure is common to all Eulerian-on-Lagrangian (EoL) derivations. The mass matrices appear as

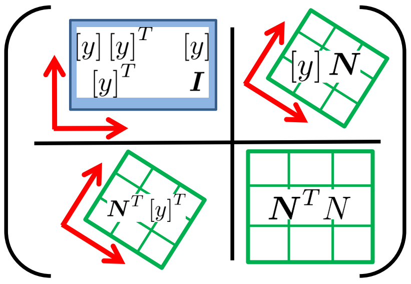


Fig. 3: The block structure of the Eulerian-on-Lagrangian mass matrix using rigid motion. The diagonal blocks are Lagrangian and Eulerian respectively while the off-diagonal blocks represent the coupling terms of the system.

$$\begin{pmatrix} \boldsymbol{M}_1 & \boldsymbol{M}_{12} \\ \boldsymbol{M}_{12}^T & \boldsymbol{M}_2 \end{pmatrix} \tag{11}$$

where the diagonal blocks are the mass matrices for our individual Lagrangian and Eulerian dynamical systems (respectively) and the off-diagonal blocks are the coupling matrices. Applying the Lagrange-d'Alembert principle [Lanczos 1986] gives rise to the equations of motion for our element. After assembling the per-element mass matrices and force vectors we arrive at

$$\boldsymbol{M}\ddot{\boldsymbol{q}} = \boldsymbol{f}_{qvv} + \boldsymbol{f}_e + \boldsymbol{f}_b + \boldsymbol{f}_c = \boldsymbol{f} \tag{12}$$

where $\boldsymbol{M}$ is the global mass matrix, $\boldsymbol{f}_{qvv}$ are the centrifugal and Coriolis forces acting on the Eulerian domain, $\boldsymbol{f}_e$ are the forces of elasticity, $\boldsymbol{f}_b$ are body forces, and $\boldsymbol{f}_c$ are forces due to contact. The force $\boldsymbol{f}_{qvv}$ results from the quadratic velocity terms in the Euler-Lagrange equations [Murray et al. 1994]. We observe that this global mass matrix retains the block structure of Equation 11 which is illustrated in Figure 3. From here we can proceed in a completely general fashion, assuming only that the mass matrix has the described structure.

The crucial part of this system is the acceleration term $\ddot{\boldsymbol{q}}$, which can be thought of as the solution to the set of differential equations

$$\begin{pmatrix} \ddot{\boldsymbol{q}}_1 \\ \frac{\partial v}{\partial t} + v \cdot \nabla v \end{pmatrix} = \boldsymbol{M}^{-1}\boldsymbol{f} \tag{13}$$

where we have expanded the material derivative for the sake of clarity. One oft-used method of solving such a differential equation is splitting, wherein one first solves

$$\begin{pmatrix} \ddot{\boldsymbol{q}}_1 \\ \frac{\partial \boldsymbol{v}}{\partial t} \end{pmatrix} = \boldsymbol{M}^{-1}\boldsymbol{f} \tag{14}$$

and then uses the result, v^*, to solve nodal equations of the form

$$\frac{\partial v^*}{\partial t} + v \cdot \nabla v^* = 0 \tag{15}$$

which is now an advection and can be solved using any relevant algorithm (e.g., [Lentine et al. 2011]). Advection occurs on an Eulerian grid defined in the intermediate space.

Thus our solution procedure is equivalent to a standard Lagrangian dynamics simulator with extra advection steps required at the velocity and position levels (Algorithm 1). Below we will describe our particular solver in greater detail.

Algorithm 1 A high level overview of the Eulerian-on-Lagrangian solution procedure. Here we explicitly denote the partial time derivatives of Eulerian qualities for clarity.

```
1: Solve $M \begin{pmatrix} \ddot{q}_1 \\ \frac{\partial v}{\partial t} \end{pmatrix} = f$ for $\dot{q}_1^{t+1}$ and $v^*$
2: $v^{t+1}$ = advect($v^*$)
3: Solve $\begin{pmatrix} q_1 \\ \frac{\partial u}{\partial t} \end{pmatrix} = \begin{pmatrix} \dot{q}_1^{\,t+1} \\ v^{t+1} \end{pmatrix}$ for $q_1$ and $u^*$
4: $u$ = advect($u^*$)
```

To solve step 1 of Algorithm 1, we discretize the acceleration $\ddot{q}$ and solve at the velocity-impulse level, which gives

$$M\dot{q}^{t+1} = \Delta t f + M\dot{q}^{t} \overset{\text{def}}{=} p^{t+1}. \tag{16}$$

Using the block structure of M and suppressing the time step $t+1$ for clarity,

$$M_1\dot{q}_1 + M_{12}v^* = p_1 \tag{17a}$$
$$M_{12}^T\dot{q}_1 + M_2v^* = p_2, \tag{17b}$$

where M_1 is the nonsingular Lagrangian mass matrix, M_2 is the nonsingular Eulerian mass matrix, M_{12} is the coupling matrix between the two modes, p_1 is the generalized Lagrangian momentum and p_2 is the generalized Eulerian momentum(Figure 3). Because we lump the mass to the nodes, M_2 is diagonal.

3.4 Exploiting Redundancies

Solving Equation 16 is difficult because the mass matrix, M, contains a non-trivial nullspace. Consider the case of a simple translation. The same motion can be described by advection through the Eulerian grid or a Lagrangian translation of the grid itself (see Figure 4). The total velocity of the system is unique but its decomposition into Eulerian and Lagrangian parts is not. We can illustrate this more formally using the fundamental property of EoL: *Any momentum acting on the system can be solved purely on the Eulerian velocities* v^*. Thus, by setting $\dot{q}_1 = 0$ we get

$$\begin{aligned} p_1 &= M_{12}v^* \\ p_2 &= M_2v^* \\ \Rightarrow p_1 &= M_{12}M_2^{-1}p_2. \end{aligned} \tag{18}$$

Now suppose we induce a momentum via a Lagrangian velocity v_L. Then

$$\begin{aligned} p_1 &= M_1v_L \\ p_2 &= M_{12}^Tv_L. \end{aligned} \tag{19}$$

Combining this with Equation 18 yields

$$\begin{aligned} M_1v_L &= M_{12}M_2^{-1}M_{12}^Tv_L \\ \Rightarrow M_1 &= M_{12}M_2^{-1}M_{12}^T \end{aligned} \tag{20}$$

since v_L was arbitrary. Finally, suppose we have a solution $(\dot{q}_1, v^*)$ to Equation 17b which we write as $v^* = M_2^{-1}(p_2 - M_{12}^T\dot{q}_1)$. Evaluating the left hand side of Equation 17a gives

$$\begin{aligned} &M_1\dot{q}_1 + M_{12}v^* \\ =&(M_1 - M_{12}M_2^{-1}M_{12}^T)\dot{q}_1 + M_{12}M_2^{-1}p_2 \\ =&M_{12}M_2^{-1}p_2 \\ =&p_1, \end{aligned} \tag{21}$$

which follows from Equation 18 and Equation 20. Thus a solution to Equation 17b always satisfies Equation 17a and it is therefore redundant.

Examination of Equation 17b reveals an underdetermined system with a nullspace of dimension equal to the number of Lagrangian configuration variables. To see this, observe that any solution to Equation 17b may be written as

$$\begin{pmatrix} \dot{q}_1 \\ v^* \end{pmatrix} = \begin{pmatrix} I \\ -M_2^{-1}M_{12}^T \end{pmatrix}\dot{q}_1 + \begin{pmatrix} 0 \\ M_2^{-1}p_2 \end{pmatrix}. \tag{22}$$

Due to the full row rank of M_{12}, the matrix $\begin{pmatrix} I \\ -M_2^{-1}M_{12}^T \end{pmatrix}$ is a basis for the nullspace. In other words, we are free to choose any $\dot{q}$, and the Eulerian velocities "compensate" via Equation 22.

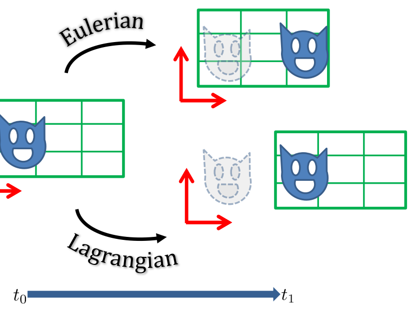


Fig. 4: Due to redundancy of the Lagrangian modes, the motion of an object can be specified by advection through the fixed Eulerian grid (top right), or fixing the object in the grid and moving the grid itself via the Lagrangian degrees of freedom (bottom right).

We specify a unique solution by maximizing the size of the time step taken which is equivalent to minimizing $||v^*||$. Since the time step is specified by the maximum grid velocity, the ∞-norm would be ideal; however, this requires the solution of a linear program on the order of the number of Eulerian variables. Instead we approximate with the 2-norm, giving

$$\dot{q}_1^{t+1} = \underset{\dot{q}_1}{\text{argmin}}||M_2^{-1}p_2 - M_2^{-1}M_{12}^T\dot{q}_1||_2, \tag{23}$$
$$v^* = M_2^{-1}p_2 - M_2^{-1}M_{12}^T\dot{q}_1^{t+1}. \tag{24}$$

This is a least squares problem (LS) in the small number of Lagrangian variables, which can be computed efficiently. As M_2 is diagonal its inverse is computed explicitly. In this discussion we have assumed an Eulerian momentum p_2 is given. In subsection 3.5 we detail a method for computing the global Eulerian velocity v in the spatial domain. We then simply compute $p_2 = M_2Ov$, where O selects the components of v for the given object and transforms its spatial velocity to its intermediate velocity, since this is the space in which advection occurs. Finally, it will be useful for us to rewrite Equation 24 as

$$v^* = \text{proj}_{M_C}Ov, \tag{25}$$

where $M_C = M_2^{-1}M_{12}^T$ and proj is the projection operator.

3.5 Solving the Dynamics in the Presence of Contact

We first solve the dynamics exclusively on the Eulerian grid (i.e. only using the Eulerian degrees of freedom), which can be done using any Eulerian simulation method suitable for resolving contact.

In our case, we invoke Gauss' principle of least constraint [Lanczos 1986], which gives the constrained optimization problem

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\boldsymbol{a}^T\tilde{\boldsymbol{M}}\boldsymbol{a} - \tilde{\boldsymbol{f}}^T\boldsymbol{a} \\ \text{subject to} \quad & \boldsymbol{a} \in \mathcal{A}, \end{aligned} \tag{26}$$

where $\tilde{\boldsymbol{M}}$ is a global mass matrix, $\boldsymbol{a}$ and $\tilde{\boldsymbol{f}}$ are the accelerations and forces defined in the spatial domain, and $\mathcal{A}$ is a constraint manifold imposed on the acceleration; these constraints ensure interpenetration between objects does not occur. Notice that $\tilde{\boldsymbol{M}}$ is assembled from the Eulerian mass matrices $\boldsymbol{M}_2$ in Equation 17. First order discretization of the acceleration gives rise to a quadratic program (QP) on the spatial velocities with constraints we now derive. Consider a colliding pair of objects A and B with surfaces Γ_A and Γ_B, respectively. Due to the discrete time integration there is some volume of intersection Ω. We impose a velocity level constraint specifying that the volume of intersection at the next time step must be smaller than Ω, giving the surface integration constraint

$$\int\limits_{\Gamma_A \cap \Omega} \boldsymbol{v}^A \cdot \boldsymbol{n} \mathrm{d}\Gamma \quad + \int\limits_{\Gamma_B \cap \Omega} \boldsymbol{v}^B \cdot \boldsymbol{n} \mathrm{d}\Gamma \leq \boldsymbol{0}, \tag{27}$$

where $\boldsymbol{n}$ are surface normals. Spatial velocities are represented on the Eulerian grid by trilinear shape functions. Thus expanding the velocity terms in Equation 27 yields the constraint

$$\begin{aligned} &\sum_{s=1}^{n} \left(\boldsymbol{v}_s^A \int\limits_{\Gamma_A \cap \Omega} \boldsymbol{\Phi}_s \cdot \boldsymbol{n} \mathrm{d}\Gamma \quad + \boldsymbol{v}_s^B \int\limits_{\Gamma_B \cap \Omega} \boldsymbol{\Phi}_s \cdot \boldsymbol{n} \mathrm{d}\Gamma \right) \\ &= \boldsymbol{j}^T \boldsymbol{v} \leq \boldsymbol{0} \end{aligned} \tag{28}$$

where s indexes the nodal velocities and their associated interpolating functions given by $\boldsymbol{\Phi}_s$.

The constraints are assembled into a global constraint matrix $\boldsymbol{J}$ and the QP solved at each time step is given by

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\boldsymbol{v}^T\tilde{\boldsymbol{M}}\boldsymbol{v} - \tilde{\boldsymbol{p}}^T\boldsymbol{v} \\ \text{subject to} \quad & \boldsymbol{J}\boldsymbol{v} \leq \boldsymbol{0}, \end{aligned} \tag{29}$$

where $\tilde{\boldsymbol{p}}$ is a globally assembled momentum vector. In the case of rigid bodies with externally prescribed motion the formulation changes slightly as the right hand side is no longer $\boldsymbol{0}$.

Collision detection and the computation of the surface integrals in Equation 28 require a representation of each object's surface in the spatial domain. Any suitable method could be used, such as the volumetric method of [Levin et al. 2011] or recent methods based on ray tracing [Wang et al. 2012]. In our implementation we use the reconstructed surface (see subsection 3.8), with a ray tracing collision detector to produce a collection of intersection rays, barycentric coordinates w.r.t. the surface mesh, and normals for every pair of colliding objects. Since spatial velocities are stored in the intermediate domain (on the Eulerian grid), we use the barycentric coordinates of the intersected rays to find the corresponding positions y and their associated degrees of freedom, from which the shape function values can be computed as well as the nonzero pattern of $\boldsymbol{J}$. Finally, we note that constraints between objects are decomposed via a collision grid defined on the spatial domain to enhance the resolution of constraints along deforming surfaces.

Equation 29 is solved via an efficient primal-dual active set method [Ito and Kunisch 2008]. Due to the diagonal structure of the mass matrix, the complexity of the QP solver is $O(m^3)$, where m is the number of constraints. Once the spatial velocity $\boldsymbol{v}$ is known, we can compute the Eulerian momentum $\boldsymbol{p}_2$ of each object. Thus step 1 of Algorithm 1 amounts to a QP solve to determine the global momentum followed by a LS solve to optimally compute Eulerian and Lagrangian velocities. See Table II for computational runtimes.

3.6 Dynamic Time Step

Crucial to the efficiency of our approach is determining an appropriate time step Δt. Here we follow the approach of [Levin et al. 2011] and extend it to the Eulerian-on-Lagrangian formulation. First-order advection has the stability requirement

$$\Delta t \max \left(\frac{||\boldsymbol{v}_x^*||}{\Delta x}, \frac{||\boldsymbol{v}_y^*||}{\Delta y}, \frac{||\boldsymbol{v}_z^*||}{\Delta z} \right) \leq \alpha \tag{30}$$

for a parameter $\alpha < 1$. We use a pseudo-implicit method to determine Δt which satisfies Equation 30. One can describe the solution to the QP Equation 29 as

$$\boldsymbol{v} = \boldsymbol{v}_0 + \Delta t \boldsymbol{v}_1, \tag{31}$$

where $\boldsymbol{v}_0$ and $\boldsymbol{v}_1$ are basis vectors which can be found by solving Equation 29 at a given time followed by a linear solve. Substituting Equation 31 into Equation 25 yields

$$\begin{aligned} \boldsymbol{v}^* &= \mathrm{proj}_{\boldsymbol{M}_C} \boldsymbol{O}(\boldsymbol{v}_0 + \Delta t \boldsymbol{v}_1) && \text{(32a)} \\ &= \mathrm{proj}_{\boldsymbol{M}_C} \boldsymbol{O}\boldsymbol{v}_0 + \Delta t \mathrm{proj}_{\boldsymbol{M}_C} \boldsymbol{O}\boldsymbol{v}_1 && \text{(32b)} \\ &= \boldsymbol{v}_0^* + \Delta t \boldsymbol{v}_1^* && \text{(32c)} \end{aligned}$$

where $\boldsymbol{v}_0^*$ and $\boldsymbol{v}_1^*$ can be computed from $\boldsymbol{v}_0$ and $\boldsymbol{v}_1$ by solving two least squares problems. Finally, we combine Equation 32c and Equation 30 to get a quadratic equation in Δt:

$$\gamma_1 \Delta t^2 + \gamma_0 \Delta t - \alpha = 0, \tag{33}$$

where $\gamma_i = \left(\frac{||\boldsymbol{v}_{ix}^*||}{\Delta x} + \frac{||\boldsymbol{v}_{iy}^*||}{\Delta y} + \frac{||\boldsymbol{v}_{iz}^*||}{\Delta z} \right)$. The resultant time step is optimal in the sense that it minimizes $||\boldsymbol{v}^*||$ while obeying the stability criterion.

3.7 Plasticity

One advantage of Eulerian simulations is that we can avoid material, intermediate and spatial domain remeshing when simulating plastic deformations. Our implementation of elastoplasticity is based on that of Bargteil *et al.* [2007] and Wicke *et al.* [2010]. The multiplicative plasticity model starts from the following decomposition of the total deformation gradient

$$\boldsymbol{F}_{total} = \boldsymbol{F}_e \boldsymbol{F}_p, \tag{34}$$

where $\boldsymbol{F}_e$ is the elastic deformation gradient and $\boldsymbol{F}_p$ is the plastic deformation gradient. The evolution of the plastic deformation gradient is given by

$$(\boldsymbol{F}_p^{-1})^{t+1} = (\boldsymbol{F}_p^{-1})^t \Delta \boldsymbol{F}_p^{-1}, \tag{35}$$

where $\Delta \boldsymbol{F}_p^{-1}$ is an update applied to $\boldsymbol{F}_p$. This update is computed using the singular value decomposition (SVD) of the current elastic deformation gradient:

$$\boldsymbol{F}_e = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T. \tag{36}$$

The plastic update is then computed as

$$\Delta \boldsymbol{F}_p^{-1} = \boldsymbol{V} \left(\frac{\boldsymbol{\Sigma}}{(\det \boldsymbol{\Sigma})^{1/3}} \right)^{-\gamma} \boldsymbol{V}^T, \tag{37}$$

where $\gamma = \nu \left(\frac{\|\sigma\| - \|\sigma_y\|}{\|\sigma\|} \right)$, $0 \leq \gamma \leq 1$ and $\|\sigma_y\|$ is the plastic yield.

Our approach is distinguished from previous methods [Bargteil et al. 2007; Wicke et al. 2010] by the manner in which the plastic deformation is stored. Though we allow Eulerian deformation, the availability of a mapping to z allows us to store the plastic deformation in the material domain. When computing (Equation 34) in the spatial domain, we perform a lookup for $\boldsymbol{F}_p$ in z. When evolving $\boldsymbol{F}_p$, we construct a mapping from the material domain to the spatial domain using local meshless interpolation methods. Using this interpolated displacement $\tilde{\boldsymbol{u}}$, $\boldsymbol{F}_{total}$ can be computed as $\boldsymbol{I} + \frac{\partial \tilde{\boldsymbol{u}}}{\partial \boldsymbol{z}}$. Subsequently, $\boldsymbol{F}_e$ can be computed, followed by the SVD (Equation 36). We incorporate the Lagrangian displacements into $\tilde{\boldsymbol{u}}$ in order to ensure proper evolution of $\boldsymbol{F}_p$. Our approach does not require remeshing, and by storing $\boldsymbol{F}_p$ in the material domain, the plastic deformation gradient will not smear out during advection.

3.8 Surface Reconstruction

We leverage our Eulerian displacements to perform surface reconstruction directly from the material coordinates z. This imparts a guarantee that the surface mesh will always return to its original shape when the simulation displacements are zero (something that cannot be said for advecting mesh vertices in an Eulerian flow).We store a surface mesh for our object in the material domain. Due to the use of Lagrangian modes and adaptive time stepping, the Eulerian displacements stay small over the course of a time step. This allows us to perform a search for the intermediate position, in y, for a given vertex of our mesh using the simple, iterative procedure given by Algorithm 2. Once this is done we can use the Lagrangian degrees of freedom to transform the mesh into the spatial domain for rendering. The availability of a fixed surface representation in z allows us to perform texture mapping with zero drift. In Algorithm 2 v is a vertex in our surface mesh, $\epsilon > 0$ is a user defined tolerance and $\boldsymbol{F}$ is the Eulerian deformation gradient computed at p. Note that at time t we do not know the intermediate space position of v so we guess that it has not moved since time $t - 1$. We compute a reference space error using the function $\phi_2^{-1}(y, t)$ which is simply our Eulerian mapping (and is thus always known) and finally an intermediate space correction using $\boldsymbol{F}$. Upon completion of the algorithm $\boldsymbol{p}$ holds the correct intermediate space position of $\boldsymbol{v}$ at t. In practice we execute Algorithm 2 in parallel for each mesh vertex.

Algorithm 2 Search procedure for surface reconstruction.

```
1: for $v \in$ Surface Mesh do
2:     $p = y^{t-1}(v)$
3:     repeat
4:         $e = \phi_2^{-1}(v, t) - \phi_2^{-1}(p, t)$
5:         $p = p + \boldsymbol{F}e$
6:     until $||e|| < \epsilon$
7: end for
```

4. RESULTS AND DISCUSSION

Figure 6 shows the benefits of the Eulerian-on-Lagrangian simulation for simple translation. Note that the Eulerian case (top row) requires the discretization of all of space. This has two obvious limitations: a drastic increase in memory use and the restriction of the object's motion to the domain. Note that the EoL simulation is free to roam wherever it pleases while wasting far less memory on extraneous grid points. The numerical advantages are made clear during rotation. When a constant angular velocity is applied to a purely Eulerian object it can be damped out by elastic forces on the grid (Batty *et al.* [2008] show this for viscoelastic fluids). When the rotational degree of freedom is Lagrangian damping is greatly reduced. Given an initial angular velocity the purely Eulerian case comes to rest while the EoL method continues turning almost indefinitely (Figure 7). Our solver automatically extracts these Lagrangian modes from arbitrary motion. In some sense our method serves as a generalization of floating frame-based methods for purely Lagrangian simulation [Galoppo et al. 2007; Barbič and Zhao 2011] wherein a deformable Lagrangian simulation is coupled with a rigid frame. Explicitly exploiting the singularities in the EoL mass matrix allows us to utilize any kinematic representation for the Lagrangian component such as linear modal models (Figure 5)

Figure 8 shows our dynamic time-stepping scheme in action, which ensures stability for the latter advection scheme. Once objects exit the collision they gradually become Lagrangian. At this point the time step returns to its maximum value which is determined by the (typically less restrictive) stability conditions of the Lagrangian time integrator. The dynamic time stepping and the continuous transition between Eulerian and Lagrangian states are handled automatically by the EoL solver. Figure 9 shows that the Eulerian-on-Lagrangian simulator enjoys a large advantage in terms of timestep size over its purely Eulerian counterpart. This manifests itself as an order-of-magnitude increase for most of the simulation. Note that the dissipative effect of Eulerian advection, in the purely Eulerian case, can also be seen; the collision begins later in time, indicating that the object has lost some velocity. Figure 1 shows results from several applications of the Eulerian-on-Lagrangian method. The method is well suited for simulating highly deformable objects in an unbounded domain. Furthermore, like its Eulerian relatives it does not require an object specific discretization thus avoiding the complexities of mesh generation.

All the examples in the video are simulated on an Intel i7 2.8GHz CPU with an Nvidia GeForce GTX 580 graphics card. Timings are reported in Table II. In our current implementation each object is processed in serial, so the runtime for the least squares solver is proportional to the number of objects. For complex scenes, more collisions yield greater number of constraints and the QP solver performs accordingly. Memory requirements are large due to the use of the ray-tracing based collision detection engine [Wang et al. 2012].

5. LIMITATIONS AND FUTURE WORK

The Eulerian-on-Lagrangian Simulation framework does away with many of the issues that plague previous approaches of Eulerian simulation. However, there are still many opportunities to explore. First, the EoL integrator treats elastic forces explicitly. Though dynamic time stepping provides stability, performance can suffer when simulating very stiff materials. Implicit integrators for these types of dynamics algorithms would be a useful avenue of work. Implicit integration may also aid adaptive time stepping as, in rare cases, instabilities may occur even when consecutive time steps satisfy the stability criteria [Wright 1998] (though we do not observe this problem in our work). An extension to incompressible materials using a pressure projection step in the intermediate space would also be interesting and such an algorithm would be of great utility for the simulation of fluids and of other incompressible solids such as many biological tissues. Incompressibility is nicely described in the intermediate space as a divergence free velocity making the Eulerian-on-Lagrangian approach an intriguing method for this type of simulation. Other fascinating areas to explore include the application of Eulerian soft-tissue to skeleton based characters or simulating large fluid flows using many simultaneous EoL domains.

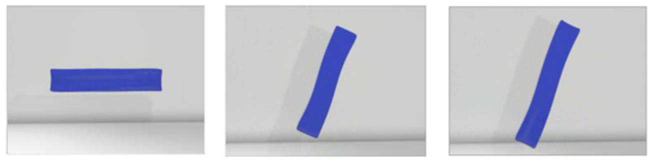


Fig. 5: Left: Purely Eulerian motion. Middle: Purely Lagrangian motion. Right: Overall motion

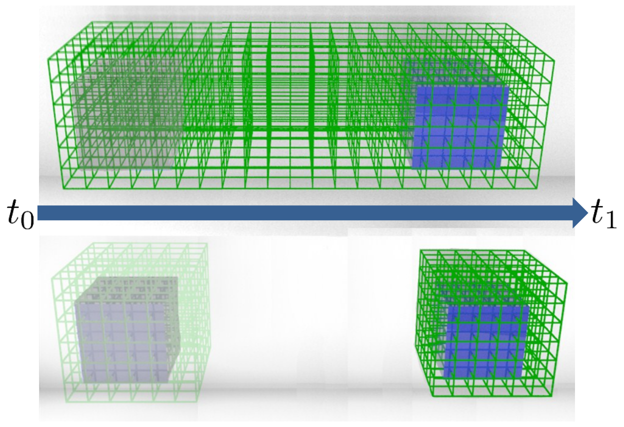


Fig. 6: Top Row: Purely Eulerian translation. Bottom Row: Eulerian-on-Lagrangian Translation

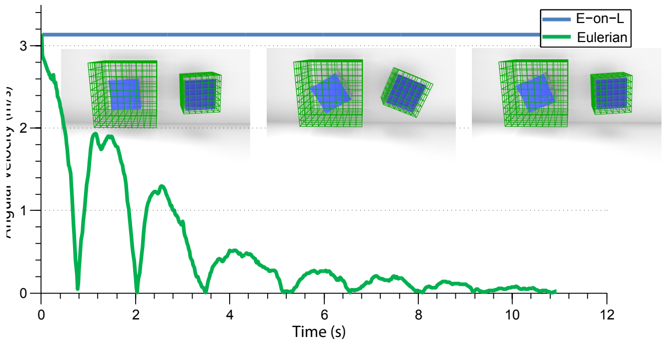


Fig. 7: The angular velocity of an Eulerian-on-Lagrangian object (E-on-L) and a purely Eulerian object undergoing rotation. Both were given an initial angular velocity of π radians/s.

6. CONCLUSION

We introduced an Eulerian-on-Lagrangian simulation method that leverages a mixed formulation to alleviate many of the shortcomings of purely Eulerian and purely Lagrangian simulation methods. The method allows Eulerian simulations over an unbounded domain of elastic and elastoplastic objects, reduces time step restrictions induced by Eulerian advection but retains the Eulerian robustness with regard to large deformations and allows the simulation of plastic materials. All this is accomplished while simultaneously resolving complicated contact scenarios involving multiple deforming bodies. It is a significant step in increasing the utility of Eulerian simulations for materials other than fluids.

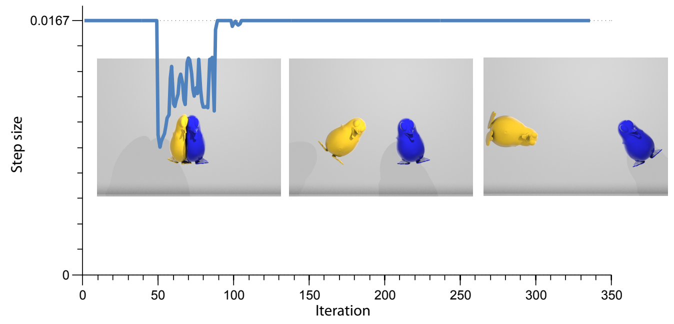


Fig. 8: The timestep, in seconds, of an Eulerian-on-Lagrangian simulation. Note that during contact the time step is automatically reduced. After contact the time step increases until it reaches the rendering frame rate.

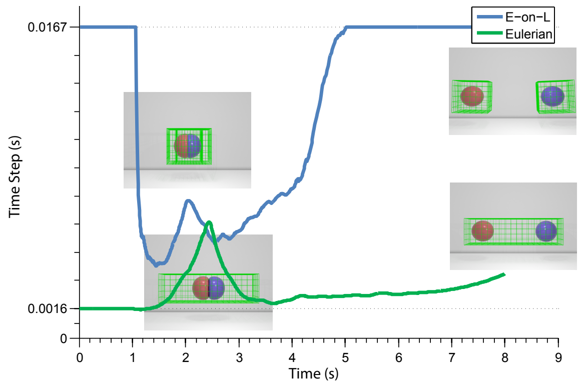


Fig. 9: The timestep size for both the purely Eulerian and Eulerian-on-Lagrangian simulations for the two ball collision. Note that in most cases the EoL timestep is an order of magnitude greater than that of the purely Eulerian simulation.

Acknowledgements

The authors would like to thank Bin Wang, Shinjiro Sueda and the anonymous reviewers for their help. This work was funded in part by grants from the Canada Research Chairs Program, NSERC, Peter Wall Institute for Advanced Studies, CIHR, and CFI.

REFERENCES

BANKS, J. W., SCHWENDEMAN, D. W., KAPILA, A. K., AND HENSHAW, W. D. 2007. A high-resolution Godunov method for compressible multi-material flow on overlapping grids. *J. Comput. Phys. 223,* 1, 262–297.

BARBIČ, J. AND ZHAO, Y. 2011. Real-time large-deformation substructuring. In *ACM Transactions on Graphics (TOG)*. Vol. 30. ACM, 91.

BARGTEIL, A. W., WOJTAN, C., HODGINS, J. K., AND TURK, G. 2007. A Finite Element Method for Animating Large Viscoplastic Flow. *ACM Trans. Graph. 26,* 3 (July), 16:1–16:8.

BARTON, P. T. AND DRIKAKIS, D. 2010. An Eulerian method for multi-component problems in non-linear elasticity with sliding interfaces. *J. Comput. Phys. 229,* 15, 5518–5540.

Example	Dim	Lag DOF	Mem(MB)	Steps	Forces	Collision	Assembly	LS	QP
2 Spheres	$2 \times 32 \times 32 \times 32$	6	799	2773	2.0	15.7	2.9	21.6	54.4
2 Penguins	$2 \times 32 \times 32 \times 32$	6	777	1531	1.7	169	1.9	15.0	22.3
Spinning Bar	$64 \times 32 \times 32$	16	737	2087	0.7	109.0	0.8	6.1	12.0
Plastic Cylinders	$2 \times 64 \times 32 \times 32$	12	833	1636	2.7	102.4	2.4	18.5	28.4
Poking	$128 \times 64 \times 64$	12	931	2284	3.3	103.4	64.1	45.0	27.4
16 Penguins	$16 \times 32 \times 32 \times 32$	6	1070	4382	25.4	894.9	168.5	150.1	93.4
Sphere Squeezer	$8 \times 48 \times 48 \times 48$	9	1018	1918	14.9	163.3	166.6	89.7	538.6
Toys Squeezer	$8 \times 48 \times 48 \times 48$	9	1112	2881	13.8	424.2	207.9	55.4	159.4

Table II. : For each example, the size of the spatial grid (Dim), maximum GPU memory used, total number of time steps taken, and average runtimes in **milliseconds** for each stage of the algorithm is shown.

BATTY, C. AND BRIDSON, R. 2008. Accurate viscous free surfaces for buckling, coiling, and rotating liquids. In *Proceedings of the 2008 ACM/Eurographics Symposium on Computer Animation*. 219–228.

BELYTSCHKO, T., KAM, L. W., AND MORAN, B. 2000. *Nonlinear Finite Elements for Continua and Structures*. John Wiley & Sons.

BELYTSCHKO, T. AND KENNEDY, J. 1978. Computer models for subassembly simulation. *Nuclear Engineering and Design 49,* 1-2, 17 – 38.

BELYTSCHKO, W. K. AND MORAN, B. New York, 2000. *Nonlinear Finite Elements for Continua and Structures*. Wiley.

BRIDSON, R. 2008. *Fluid Simulation*. A. K. Peters, Ltd., Natick, MA, USA.

CARLSON, M., MUCHA, P. J., VAN HORN, III, R. B., AND TURK, G. 2002. Melting and flowing. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*. SCA '02. 167–174.

FOSTER, N. AND METAXAS, D. 1996. Realistic animation of liquids. *Graph. Models Image Process. 58,* 5, 471–483.

GALOPPO, N., OTADUY, M., TEKIN, S., GROSS, M., AND LIN, M. 2007. Soft articulated characters with fast contact handling. In *Computer Graphics Forum*. Vol. 26. Wiley Online Library, 243–253.

GOKTEKIN, T. G., BARGTEIL, A. W., AND O'BRIEN, J. F. 2004. A method for animating viscoelastic fluids. *ACM Trans. Graph. 23*, 463–468.

ITO, K. AND KUNISCH, K. 2008. *Lagrange Multiplier Approach to Variational Problems and Applications*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

KAMRIN, K. AND NAVE, J.-C. 2009. An eulerian approach to the simulation of deformable solids: Application to finite-strain elasticity. Submitted (available: `http://arxiv.org/PS_cache/arxiv/pdf/0901/0901.3799v2.pdf`).

KAMRIN, K., RYCROFT, C., AND NAVE, J. 2012. Reference map technique for finite-strain elasticity and fluid–solid interaction. *Journal of the Mechanics and Physics of Solids*.

LANCZOS, C. 1986. *The Variational Principles of Mechanics*. Dover Publications.

LENTINE, M., AANJANEYA, M., AND FEDKIW, R. 2011. Mass and momentum conservation for fluid simulation. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '11. ACM, New York, NY, USA, 91–100.

LEVIN, D. I. W., LITVEN, J., JONES, G. L., SUEDA, S., AND PAI, D. K. 2011. Eulerian solid simulation with contact. *ACM Trans. Graph. 30*, 36:1–36:10.

LOSASSO, F., SHINAR, T., SELLE, A., AND FEDKIW, R. 2006. Multiple interacting liquids. *ACM Trans. Graph. 25,* 3 (July), 812–819.

MILLER, G. AND COLELLA, P. 2001. A High-Order Eulerian Godunov Method for Elastic-Plastic Flow in Solids* 1. *J. Comput. Phys. 167,* 1, 131–176.

MURRAY, R. M., SASTRY, S. S., AND ZEXIANG, L. 1994. *A Mathematical Introduction to Robotic Manipulation*, 1st ed. CRC Press, Inc., Boca Raton, FL, USA.

NARAIN, R., GOLAS, A., AND LIN, M. C. 2010. Free-flowing granular materials with two-way solid coupling. *ACM Trans. Graph. 29*, 173:1–173:10.

OSHER, S. AND FEDKIW, R. 2002. *Level set methods and dynamic implicit surfaces*. Springer-Verlag.

POLUDNENKO, A. Y. AND KHOKHLOV, A. M. 2007. Computation of fluid flows in non-inertial contracting, expanding, and rotating reference frames. *Journal of Computational Physics 220,* 2, 678 – 711.

PREMOŽE, S., TASDIZEN, T., BIGLER, J., LEFOHN, A., AND WHITAKER, R. T. 2003. Particle-based simulation of fluids. *Computer Graphics Forum 22,* 3, 401–410.

RAVEENDRAN, K., WOJTAN, C., AND TURK, G. 2011. Hybrid smoothed particle hydrodynamics. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '11. ACM, New York, NY, USA, 33–42.

SHAH, M., COHEN, J. M., PATEL, S., LEE, P., AND PIGHIN, F. 2004. Extended galilean invariance for adaptive fluid simulation. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*. SCA '04. Eurographics Association, 213–221.

STAM, J. 1999. Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. SIGGRAPH '99. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 121–128.

SUEDA, S., JONES, G. L., LEVIN, D. I. W., AND PAI, D. K. 2011. Large-scale dynamic simulation of highly constrained strands. *ACM Trans. Graph. 30*, 39:1–39:10.

SULSKY, D., CHEN, Z., AND SCHREYER, H. 1994. A particle method for history-dependent materials. *Computer Methods in Applied Mechanics and Engineering 118,* 1-2, 179–196.

TRAN, L. B. AND UDAYKUMAR, H. S. 2004. A particle-level set-based sharp interface cartesian grid method for impact, penetration, and void collapse. *J. Comput. Phys. 193,* 2, 469–510.

TRANGENSTEIN, J. 1994. A second-order Godunov algorithm for two-dimensional solid mechanics. *Computational Mechanics 13,* 5, 343–359.

WANG, B., FAURE, F., AND PAI, D. K. 2012. Adaptive image-based intersection volume. *ACM Trans. Graph. (Proc. SIGGRAPH) 31,* 4.

WANG, H., MUCHA, P. J., AND TURK, G. 2005. Water drops on surfaces. *ACM Trans. Graph. 24,* 3 (July), 921–929.

WICKE, M., RITCHIE, D., KLINGNER, B. M., BURKE, S., SHEWCHUK, J. R., AND O'BRIEN, J. F. 2010. Dynamic local remeshing for elastoplastic simulation. *ACM Trans. Graph. 29*, 49:1–49:11.

WRIGHT, J. P. 1998. Numerical instability due to varying time steps in explicit wave propagation and mechanics calculations. *Journal of Computational Physics 140,* 2, 421 – 431.

ZHU, Y. AND BRIDSON, R. 2005. Animating sand as a fluid. *ACM Trans. Graph. 24,* 3 (July), 965–972.