Interactive Simulation of Elastic Deformable Materials

Martin Servin*
Department of Physics,
Umeå University

Claude Lacoursière†
HPC2N/VRlab and
Computing Science Department,
Umeå University

Niklas Melin
Department of Physics,
Umeå University

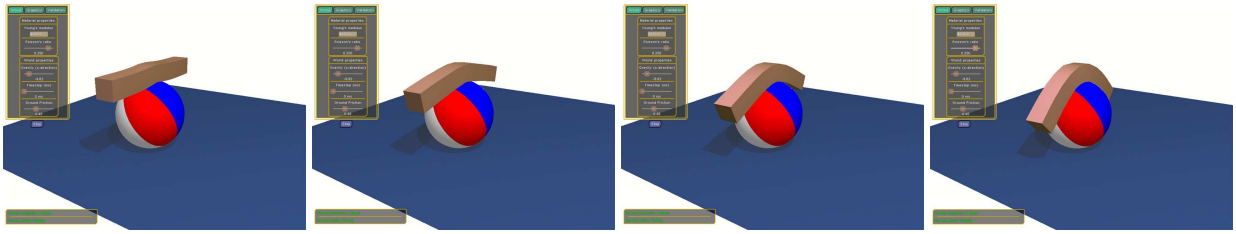

Figure 1: Snapshot from interactive simulation of a deformable beam and solid sphere

Abstract

A novel, fast, stable, physics-based numerical method for interactive simulation of elastically deformable objects is presented. Starting from elasticity theory, the deformation energy is modeled in terms of the positions of point masses using the linear shape functions of finite element analysis, providing for an exact correspondence between the known physical properties of deformable bodies such as Young's modulus, and the simulation parameter. By treating the infinitely stiff case as a kinematic constraint on a system of point particles and using a regularization technique, a stable first order stepping algorithm is constructed which allows the simulation of materials over the entire range of stiffness values, including incompressibility. The main cost of this method is the solution of a linear system of equations which is large but sparse. Commonly available sparse matrix packages can process this problem with linear complexity in the number of elements for many cases. This method is contrasted with other well-known point mass models of deformable solids which rely on penalty forces constructed from simple local geometric quantities, e.g., spring-and-damper models. For these, the mapping between the simulation parameters and the physical observables is not well defined and they are either strongly limited to the low stiffness case when using explicit integration methods, or produce grossly inaccurate results when using simple linearly implicit method. Validation and timing tests on the new method show that it produces very good physical behavior at a moderate computational cost, and it is usable in the context of real-time interactive simulations.

CR Categories: I.3.5 [Computer Graphics]: Physically based modeling— [I.3.7]: Computer Graphics—Virtual reality

Keywords: deformable simulation, elasticity, constrained dynamics, stability, numerical integration

1 Introduction

Simulation of elastically deformable objects is a necessity for creating certain types of virtual environments, such as those designed for surgical or heavy vehicle operator training applications, for instance. This problem has generated interest recently and a survey of well-known methods can be found in [Nealen et al. 2005]. We focus here on numerically stable methods for integrating elastic deformable objects at interactive rates over a wide range of stiffness, including incompressibility and the nearly rigid limit, and on establishing a clear correspondence between the known material parameters used in elasticity theory–such as Young's modulus–to the simulation parameters.

We will briefly describe the most commonly used techniques and explain how they fail for high stiffness, either losing stability, yielding grossly inaccurate results, or both, and how they lack a clear correspondence between simulation parameters and physical properties. Observe that both full incompressibility as well as full rigidity are cases of infinite stiffness. We argue that the key to handling the *infinite stiffness* regime is to reformulate the problem as a kinematically constrained dynamical system. The recovery of *finite stiffness* is done using a constraint regularization and stabilization technique which amounts to a numerically stable penalty technique with the strain energy as the penalty function. Formulation of constrained mechanical systems and techniques for solving them have been presented before in the graphics literature in [Witkin et al. 1990] [Baraff 1996] and [Erleben et al. 2005]. The idea of using standard energy terms for generating penalty force has also been used several times before [Terzopoulos et al. 1987][Teschner et al. 2004]. It is the mixing of these two ideas in a numerically stable way which is novel, and this is achieved by constructing a special integrator which is semi-implicit [Lacoursière 2006] in combination with appropriate formulation of energy and dissipation terms. The resulting method can handle stiffness from zero to infinity without developing instabilities though the effective stiffness and accu-

*e-mail: martin.servin@physics.umu.se
†e-mail: claude@hpc2n.umu.se

racy is limited by the size of the time step. Since stability hinges on reasonably accurate solutions of a large system of linear equations, performance is achieved by exploiting sparsity in a direct solver as was done in [Baraff 1996], for instance. Validation tests are conclusive in demonstrating good agreement between the simulated physical properties and the input parameters. Using a widely available sparse matrix package, UMFPACK [UMFPACK], we achieved linear complexity as a function of system size and sufficiently fast execution for interactive rates for moderately sized systems.

1.1 Contribution and organisation of this paper

Section 2 provides background of particle based simulation of deformable materials. Previous work is reviewed and the advantages and shortcomings of the different strategies are discussed. In section section 3 we consider a simple example that elucidates some of the ideas and pitfalls. We argue that the key to fast and stable simulation even in the stiff regime is to reformulate the problem as a kinematically constrained dynamical system. The new method for simulating elastic deformable materials is presented in section 4. This method combines the technique of regularization and stable stepping of constrained systems – described by Eqs. (20) – and elasticity modeling based on well established material models – specified by Eq. (15). The model is evaluated, through numerical experiments, and discussed in section 5. Summary and conclusions are found in section 6.

2 Particle system models for deformable objects

There are several ways to represent deformable objects but we restrict our attention to those represented as a set of interacting point masses, namely, lumped element models. Elastic properties of the bodies are constructed by defining various forces and constraints on the particles so that all constraints are satisfied and all internal forces cancel out when the body is in the reference configuration. A recent survey of techniques for simulating deformable objects is found in [Nealen et al. 2005]

We denote the particle positions by $x = (\mathbf{x}^{(1)T}, \mathbf{x}^{(2)T}, \ldots, \mathbf{x}^{(N)T})^T$, where $\mathbf{x}^{(i)} = (x_1^{(i)}, x_2^{(i)}, x_3^{(i)})^T$ is the 3D position vector of particle i, with the parentheses emphasizing that it is a particle index rather than a component of a vector, and N is the total number of particles. The particle velocities are $v = \dot{x}$ and the particle masses $m^{(i)}$ are collected in the diagonal mass matrix M of dimension $3N \times 3N$. The equations of motion are:

$$\dot{x} = v \tag{1}$$

$$M\dot{v} = F_{ext} + F_{int} + F_c \tag{2}$$

where the total force is divided into external force F_{ext}, internal force F_{int} and constraint force F_c. Sometimes we agglomerate the external and internal forces into $F = F_{ext} + F_{int}$.

2.1 Geometry, energy and force

General penalty forces derived from energy functions for simulating deformable elastic objects were introduced in the graphics literature by [Terzopoulos et al. 1987]. The idea is to define *geometric displacement* functions $\phi_i(x)$, which vanish in the rest configuration, and a potential energy $U(x) = \sum_i (k_i/2)\phi_i^2(x)$, where the k_i's are positive stiffness parameters. The functions $\phi_i(x)$ might be the local curvature for a string or that of the surface for a membrane instance. Other examples include deviation from rest distance between two particles, i, j, with $\phi_k(x) = (|\mathbf{x}_{(i)} - \mathbf{x}_{(j)}| - L_{ij})/L_{ij}$, where L_{ij} is the rest distance. The resulting energy represents that of a linear spring. The force generated from the potential $U(x)$ is well known to be: $F_{int} = -\partial U/\partial x$. Note that this is linear in $\phi_k(x)$ and therefore, all forces vanish at equilibrium, as needed. Provided the Jacobian matrix $J = \partial\phi/\partial x$ has full rank at the equilibrium point, the restoration forces are *linear* in the geometric displacements.

Therefore, the art of this formulation is to chose displacement functions $\phi_k(x)$ which do not have zero derivatives near the equilibrium, and which are linear independent of each other at or near the equilibrium point. In computing the bending energy for a cable made of point masses for instance, there is a zero derivative condition if one takes the bending displacement variable as: $\phi_k(x) = (|\mathbf{x}_{(i+1)} - \mathbf{x}_{(i-1)}| - 2L_0)/(2L_0)$. There are several such cases which are not so obvious and which arise accidentally when trying to brace a set of particles with springs so as to produce a unique rest configuration that is stable under arbitrary deformations. This sort of problem is usually solved using inverse trigonometric functions, as is done in the construction of shear restoration forces in cloth simulation [Baraff and Witkin 1998]. This can be quite expensive computationally and produce unexpected results.

This energy based penalty formulation was developed systematically for handling elastic solids in [Teschner et al. 2004] by defining three types of *geometric* displacements, namely, *distances*, *volumes*, and *surface areas*. These displacement functions involve the coordinates of two or more particles and generate forces on these via the potential energy function. To do this, the volume is first cut into a set of tetrahedra, each of which contains four particles, one at each vertex. Each particle is included in one or several of these. Then, distance displacement functions are defined between a particle and its nearest neighbors, using the rest configuration for labeling. Then, a volume displacement function is defined using the coordinates of the four particles in each tetrahedron. Finally, a surface area displacement function is defined by measuring the area of the free surfaces of the tetrahedron and subtracting the rest area. One can then choose independent parameters for the *compressibility*, the *stiffness*, and the *surface tension*. This approach is an improvement over standard spring-and-damper models, where extra springs are introduced to model some of the material properties.

The first problem with this model is that these three parameters are not completely independent so that mapping to known material properties is difficult as discussed in [Nealen et al. 2005]. Essentially, the true deformation energy, that is easily measured in the lab and tabulated in handbooks, might not have such a simple mapping to the simple geometric quantities of the system of particles. This is very clear in the case of a string where definition of bending energy is difficult and torsion impossible, when considering only point particles.

The second problem is that forces generated by this model do not converge in the limit of infinite stiffness, even though the trajectories of the particles are mathematically well behaved in this limit. Indeed, as demonstrated in [Rubin and Ungar 1957], the penalty forces typically oscillate wildly in the limit where the stiffness constants become large. This is not an intuitive result since the constraint forces are well behaved in the case where the kinematic constraints $\phi_i(x) = 0$ are rigorously enforced. Therefore, it is generally difficult to integrate systems with large penalty forces. When using explicit integration methods, the time step is limited to less than a fraction of the smallest natural period of oscillations. In the present case, if all particles have identical masses m, this period is: $T_{min} = 2\pi\sqrt{m/k_{max}}$. This stability requirement seriously limits the

maximum stiffness or the performance. Some special implicit numerical integration methods improve stability but great caution is needed. For instance, when using the linearly implicit integration strategy of [Baraff and Witkin 1998] however, the stiffness restrictions are not so severe but artificial damping is clearly noticeable, limiting the usefulness of the method.

Two remedies are provided in the next section. The first is a definition of the potential energy of deformation which is not based on the simple geometric quantities but which correctly maps the strain tensor of elasticity to particle positions. This is similar to what is done in finite element methods. The second is a stable regularization of constraints so as to remove the high oscillatory components of standard penalty force formulations.

2.2 Constraints and regularization

Kinematic constraints impose restrictions on the motion of the particles in the system. For instance, we can simulate a particle moving on the plane $z = 0$ by imposing that restriction directly on the coordinate, thus bypassing the computation contact forces and acceleration due to gravity. It is well known in physics that kinematic constraints of the form[1] $\phi_i(x) = 0$ are the physical limit of strong potential forces of the form $(k_i/2)\phi_i^2(x)$, for very large k_i and this has lead to two main strategies for solving constrained system.

The first is to formulate the equations of motion taking the restrictions $\phi_i(x) = 0$ into account as is described in [Erleben et al. 2005] for instance. This requires the computation of *constraint forces* which are the solution of a non-linear system of equations. The main problems here are that even when linearizing, this system of equations can be computationally expensive to solve, and that the trajectories x tend to drift away from $\phi_i(x) = 0$ because of discretization and approximation errors. The common strategy for linearizing the equations for the constraint forces [Baraff 1996] and stabilizing the constraints $\phi_i(x) = 0$ [Baumgarte 1972] is notoriously unstable and this has given the constraint method much bad press. Stable and efficient methods for solving constrained systems do exist though and we will provide one of these in the next section.

The second strategy is to include the potentials $(k_i/2)\phi_i^2(x)$ and the corresponding forces. This latter approach is known as a penalty force computation. At the mathematical level, there is a rigorous correspondence on the trajectories produced by these two methods in the limit of infinite k_i. Given the simplicity of this formulation, penalty forces are very attractive. But this is deceptive. Indeed, a little known fact is that penalty forces do not converge to the smooth constraint forces corresponding to $\phi_i(x) = 0$. As shown in [Rubin and Ungar 1957], the penalty forces oscillate with very high frequency in the limit of large k_i. In technical terms, the convergence of the penalty forces is only weak$*$. This means in particular that the average value of the penalty forces, over a short interval of time, Δt, say, does converge to the smooth constraint force. Numerically, this means that high penalty forces quickly generate instabilities which can only be resolved using special integration techniques designed for highly oscillatory systems. Some implicit integrators work well on highly oscillatory systems but some don't and a case of the latter is the first order implicit Euler method.

What we seek is a combination of the two strategies so that we can recover the stability of constraint computation but allow the modeling flexibility of penalty forces. To do this, we first state the equations of motion of the constrained system. We collect all the constraint terms in a vector function $\phi(x)$ with size n_c, the sum of the constraint dimensions. Of course, if $\phi(x) = 0$ at all times, we have $\dot{\phi}(x) = 0$ and by chain rule, this is $\dot{\phi}(x) = J\dot{x}$ where J is known as the Jacobian matrix of ϕ. The components of J are $[J]_{ij} = \partial\phi_i/\partial x_j$. Recall that for any surface defined with a scalar function, such as $\phi_i(x) = 0$, the gradient of $\phi_i(x)$ is normal to the surface, and that moving along the gradient is the fastest way to move away from the surface $\phi_i(x) = 0$. We want the constraint forces to directly oppose any force trying to move the system away from the surfaces $\phi_i(x) = 0$. It is therefore constructed as a linear combination of the constraint normals, $F_c = J^T\lambda$, where λ is a vector with n_c components, one for each constraint. What are the values of the components of λ? Well, just what is necessary to remove any part of resulting forces which point in the direction normal to any of the surfaces $\phi_i(x)$! Of course, since the point x moves in a direction *tangential* to any of the constraint surfaces $\phi_i(x) = 0$, the constraint forces we just defined are workless. When this is taken into consideration, and after writing $v = \dot{x}$ the *differential algebraic equations* of motion for the constrained system are:

$$\dot{x} = v \tag{3}$$

$$M\dot{v} = F_{ext} + F_{int} + F_c \tag{4}$$

$$\phi(x) = 0 \tag{5}$$

This is a nonlinear system of equation and a common strategy is to replace the constraint with a linear combination: $\ddot{\phi}(x) + a\dot{\phi}(x) + b\phi(x) = 0$, which is mathematically equivalent and stable if the coefficients a and b are positive. The resulting equation is linear in $\dot{v}$ as is easily verified, and the full system of equations (3)–(5) is then discretized as any other second order ordinary differential equation. This strategy is now getting known as the *acceleration based* method in contrast to so-called *velocity based* methods which we describe shortly. Note that this nomenclature is only used in the graphics literature. The problem here is that numerical stability is strongly dependent on the choice of the a and b parameters. However, there is no systematic strategy for choosing a and b which work in all cases. Choosing a and b too small leads to constraint drift so that $|\phi|$ increases with time, but choosing a and b too large makes the system explode. Conversely, a solution of the stabilized equations of motion with non-zero coefficients a, b, even when stable, is not necessarily a solution of the original system of equations. This method is a very bad idea indeed. Curiously, this is the most popular scheme in the graphics literature where it goes back to the early 1990s [Witkin et al. 1990].

The alternative is to discretize the velocity and acceleration *before* attempting to linearize the constraint $\phi(x) = 0$. This is covered in Sec. 2.4.

As was mentioned previously, constraints can be realized as the limit of strong potentials. If we keep the strength finite though large, this is a form of regularization if we find a way to write the equations of motion in terms of the inverse of the large stiffness. When discretized judiciously, this scheme can produce stable and efficient time stepping scheme of systems with either constraints or very strong forces. In fact, almost the entire range from 0 to infinity is allowed, although some elasticity may remain in the infinite case as a numerical error. Relaxing the constraints by keeping a finite but large penalty parameter (or small regularization parameter) has the added benefit of removing numerical problems which occur when constraints are degenerate or over defined, and to help stabilize the linear equation solving process by making the matrices strongly positive definite. In other word, we are trading the infinite stiff limit for speed and numerical stability.

Starting from a constraint ϕ we construct the potential energy:

$$U(x) = \tfrac{1}{2}\phi^T(x)\alpha^{-1}\phi(x) \tag{6}$$

[1] We consider only time independent equality constraints here. Inequality constraints may be used for non-penetration constraints, e.g., for collisions and contact.

for symmetric, positive definite matrix α of dimension $d_c \times d_c$. The correspondence to the penalty terms defined previously is the case where α^{-1} is a diagonal matrix and where the entries on the main diagonal are the stiffness parameters k_i. The limit $\alpha \to 0$ corresponds to infinite stiffness and $\alpha \to \infty$ to zero stiffness. In the case of distance constraints the corresponding potential is a spring potential, with spring stiffness α^{-1}. Since $U(x)$ is a potential energy term, it produces forces in the standard way, namely: $F_c = -\partial U/\partial x^T = -J^T \alpha^{-1}\phi$, where J is the Jacobian matrix of the functions $\phi(x)$ as before. Next, in order to replace the large parameters α^{-1} for the small α in the equations of motion, we introduce an artificial variable $\lambda = -\alpha^{-1}\phi$ such that $F_c = J^T\lambda$. For the regularized system the equations of motion are modified to:

$$\dot{x} = v \tag{7}$$

$$M\dot{v} = F_{ext} + F_{int} + F_c \tag{8}$$

$$\alpha\lambda(x,t) = -\phi(x,t). \tag{9}$$

Note that in the limit of infinite stiffness, $\alpha \to 0$, the Eqs. (7)–(9) are free from singularities and reproduces the system (3)–(5). The trick now is to discretize this avoiding the high frequency oscillations in the constraint forces. Note also that a first order dissipative term of the form $-\beta\dot{\phi}$, with $\beta > 0$, can be also added to the right hand side of (9) without affecting the limit $\alpha \to 0$ as long as $\beta \to 0$ simultaneously.

2.3 Elastic deformation energies

Now that we have seen how to transform quadratic potential energy terms into constrained systems and vice versa, we construct a potential energy term for elastically deformable materials and define the functions $\phi(x)$ used in (6). Instead of using the intuitive geometric displacement functions $\phi(x)$ however, we turn to elasticity theory in order to approximate the strain tensor–a measure of deformation–in terms of the coordinates of the constituent particles. The benefit here is that all parameters entering the simulation are directly related to the known, tabulated material properties. Elasticity theory is rigorously covered in Ref. [Fung and Tong 2001] and more accessible for the purpose of physics based animation in Ref. [Erleben et al. 2005].

We will consider only linear (Hookean) and isotropic elastic materials here. This means specifically that the relation between the deformation and the restoring force is linear and therefore, the potential energy is quadratic in the deformations. To discretize the deformation and thus express it in terms of the particle coordinates, we use a spatial discretization found in *finite element analysis*, restricting our choice to linear *shape functions* and tetrahedral meshes. The mass is lumped at the nodes which correspond to point particles.

To parametrize the deformation of a solid, we first consider that it occupies some domain B in 3D space, in its rest configuration. Considering infinitesimal displacements first, each point $\mathbf{r} = (r_1, r_2, r_3)^T$ is moved by a small amount, $\mathbf{u}(\mathbf{r})$, so that its new location is: $\mathbf{r}' = \mathbf{r} + \mathbf{u}(\mathbf{r})$. This defines the vector field $\mathbf{u}(\mathbf{r})$ over the domain B. The field is needed for constructing a measure of deformation which is estimated by measuring the change in distance between two nearby points.

We assume that B is divided into a set of tetrahedra in what follows and concentrate the analysis on a single tetrahedron composed of four nodal particles with current positions $\mathbf{x}^{(a)}(t), a = 1,2,3,4$.

The domain B is now the original, undisplaced, undistorted volume of our tetrahedron. To construct a mapping which relates the vector field $\mathbf{u}(\mathbf{r})$, to the *current* positions of the nodes, we need to compute the current displacement of each node as well as an interpolating shape function. The role of the shape function is to distribute the displacements at the nodes to displacements at the interior points in B so that if the node a_1 has a large displacement but node a_2 has none, the displacement field increases smoothly between these two nodes. We define $\mathbf{u}^{(a)}$ to be the full displacement vector of nodal particle (a) from an arbitrary initial position $\mathbf{x}_0^{(a)} = \mathbf{x}^{(a)}(0)$ where the tetrahedron is at rest, say, so we have: $\mathbf{u}^{(a)} = \mathbf{x}^{(a)}(t) - \mathbf{x}_0^{(a)}$. The simplest case is to use a linear interpolation so that if the vectors $\mathbf{u}^{(a)}$ are the nodal displacements at each node a, then, the displacement field reads:

$$\mathbf{u}(\mathbf{r}) = \sum_{(a)} N^{(a)}(\mathbf{r},x)\mathbf{u}^{(a)}, \tag{10}$$

where summation is over the four particles in each tetrahedron, and x is the current configuration vector. Details of the full derivation of the shape function are found in [Erleben et al. 2005] for instance. The linear shape function is a scalar function of the form

$$N^{(a)}(\mathbf{r},x) = V^{-1}(a^{(a)} + b^{(a)}r_1 + c^{(a)}r_2 + d^{(a)}r_3), \tag{11}$$

where $V(x)$ is the volume of the tetrahedra and the coefficients satisfy:

$$\begin{bmatrix} a^{(1)} & a^{(2)} & a^{(3)} & a^{(4)} \\ b^{(1)} & b^{(2)} & b^{(3)} & b^{(4)} \\ c^{(1)} & c^{(2)} & c^{(3)} & c^{(4)} \\ d^{(1)} & d^{(2)} & d^{(3)} & d^{(4)} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ x_1^{(1)} & x_1^{(2)} & x_1^{(3)} & x_1^{(4)} \\ x_2^{(1)} & x_2^{(2)} & x_2^{(3)} & x_2^{(4)} \\ x_3^{(1)} & x_3^{(2)} & x_3^{(3)} & x_3^{(4)} \end{bmatrix}^{-1} \tag{12}$$

A few important observations must be made here. First, the vector $\mathbf{r}$ used in $\mathbf{u}(\mathbf{r})$ is in the interior of the *current* tetrahedron formed by the current coordinates of the nodes $\mathbf{x}^{(a)}$. Second, the displacement field $\mathbf{u}(\mathbf{r})$ is not small. Indeed, a uniform translation of all the nodes by a vector $\mathbf{y}$ produces $\mathbf{u}(\mathbf{r}) = \mathbf{y}$, which is arbitrarily large. A uniform rotation of all the particles also causes large changes in $\mathbf{u}(\mathbf{r})$. But the displacement field itself is not the measure of deformation. Instead, the Green strain tensor, which measures local variations in distances between close points $\mathbf{r}$ and $\mathbf{r} + \delta\mathbf{r}$, is what is needed. This is defined in terms of the *derivatives* of the displacement field as:

$$\varepsilon_{ij} \equiv \frac{1}{2}\left[\frac{\partial u_i}{\partial r_j} + \frac{\partial u_j}{\partial r_i} + \sum_{k=1,2,3}\frac{\partial u_k}{\partial r_i}\frac{\partial u_k}{\partial r_j}\right]. \tag{13}$$

This tensor is symmetric and is therefore parametrized with a six dimensional vector: $\varepsilon = (\varepsilon_{11}, \varepsilon_{22}, \varepsilon_{33}, \varepsilon_{12}, \varepsilon_{13}, \varepsilon_{23})^T$. The quadratic term is often ignored but is necessary here if we want zero strain under rigid displacements.

The Green strain tensor is a good measure of small deformation but we use it for arbitrarily large ones. This can pose a problem if the four nodes collapse onto a plane, in which case, Eqn. (12) cannot be solved. Worse still, after going through a planar collapse, a tetrahedron can become *inverted* and eventually go to rest in an inside-out configuration. This can be remedied by adding an extra constraint to the system stating that the determinant of the matrix on the left side of Eq. (12) should be near unity but we do not pursue this further here.

Computing the derivatives of the displacement field is straightforward but tedious. The resulting expressions are found in [Erleben et al. 2005] for instance. The Ref. [Bro-Nielsen and Cotin 1996] is also useful for making implementations.

We now construct a potential energy in terms of the strain variables which are the natural measures of deformation. For an elastic material, the deformation energy per unit volume is given by $W(x) = \frac{1}{2}\varepsilon^T D\varepsilon$, where:

$$D = \begin{bmatrix} \lambda+2\mu & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda+2\mu & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & \lambda+2\mu & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu \end{bmatrix}, \qquad (14)$$

and λ and μ are the Lame constants which are directly related to other common material parameters such as Young's modulus, Y, bulk modulus B and Poisson ratio ν, via simple algebraic relations, namely: $Y = \mu(3\lambda+\mu)/(\lambda+\mu)$, $B = \lambda + (2/3)\mu$, and $\nu = (1/2)\lambda/(\lambda+\mu)$. The matrix D is symmetric and positive definite as is well known. For large values of λ the material approaches incompressibility and with increasing values of μ the resistance to shearing deformations increases. Typical solids have Young's modulus ranging from 10 *MPa* for rubber to 2 *GPa* for Nylon, and up to 1000 *GPa* for diamond. The Poisson ratio ranges between 0 (compressible) and $1/2$ (incompressible). This means that λ and μ ranges between the order of the Young's modulus and infinity, $\lambda \to \infty$ latter being the limit of incompressibility.

We are now ready to formulate the kinematic constraint producing rigid body motion for a tetrahedron and the potential energy for elastic deformations. As the deformation energy $W(x)$ is an energy density, the potential energy for elastic deformation of a tetrahedron is the volume integral of this expression:

$$U = VW = \tfrac{1}{2}(V^{\frac{1}{2}}\varepsilon)^T D(V^{\frac{1}{2}}\varepsilon). \qquad (15)$$

We identify the constraint for rigid body motion of a tetrahedron as $0 = \phi = V^{\frac{1}{2}}\varepsilon$ and the constant matrix α in the regularization description is identified as: $\alpha = D^{-1}$. We will employ this energy function for particle systems representing elastic deformable objects.

As noted previously, matrix D is constant, symmetric, and positive definite. It can therefore be factorized to the form $Q^T\alpha^{-1}Q$ where Q are constant orthogonal matrices and α^{-1} is diagonal with non-negative entries. The vector of constraints ϕ defined previously can therefore be associated with $\phi = Q\varepsilon$. Note here that the matrix Q essentially rotates the strain components, mixing them in a way that is dependent on the ratio between the Lamé constants. This mixing explains, in part, why simple geometric penalty function cannot be mapped easily to the physical parameters, even though the penalty terms are constructed from an identical tetrahedral mesh.

2.4 Time stepping

Details of numerical integration methods plays a critical role in simulation stability, accuracy–as well as the loosely defined notions of realism or plausibility–and even more so when using low order algorithms. The special objective we have in mind here is a stable integration of the highly oscillatory forces arising from the regularized constraint forces, combined with speed and a moderate level of accuracy. This contrasts with the numerical analysis literature where only *overall efficiency* is considered, namely, the accuracy achieved for a given unit of computational effort. In addition, accuracy is generally measured in terms of local or global error bounds on the solution itself. In simulating physical systems however, accuracy can also be measured in terms of preservation of known invariants of the trajectories such as momentum, energy, or symplecticity of the flow, see Ref. [Kharevych et al. 2006]. Since the equations of motion are differential formulations of the consequences of these global invariants on the trajectories, numerical preservation of these invariants has a strong impacts on realism. For low order methods, the distinction is particularly obvious as we illustrate below. In fact, some low order methods, such as the popular symplectic Euler [Kharevych et al. 2006]–also known as Störmer-Verlet, Leapfrog, or several other names as well–*globally* preserve some of these invariants within certain bounds during the entire simulation, provided some stability restrictions on the size of the time step are met. By contrast, a fourth order Runge Kutta method, which is locally more accurate, does not preserve any of the physical invariants globally and can accumulate significant errors over time. For instance, with the wrong choice of time step, the energy of the system might decay to zero unintentionally–and inexplicably for the user.

We will discuss benefits and shortcomings of two widely used methods in the context of simulation of elastic materials first before introducing the regularized stepper which meets our objectives. Further examples and discussions are provided in section 3.

Consider the family of first order time steppers for the equations of motion (1)–(2):

$$x_{n+1} = x_n + \Delta t v_{n+n_x} \qquad (16)$$

$$v_{n+1} = v_n + \Delta t M^{-1} F_{n+n_v}, \qquad (17)$$

where the index n denotes the time point $t = n\Delta t$, and the indexes $n_x = 0,1$ $n_v = 0,1$ denote the discrete time at which the various quantities are evaluated. The explicit Euler method corresponds to the case $n_x = 0$, $n_v = 0$. It fails unconditionally on the undamped harmonic oscillator, producing trajectories with $|x_n| = O(e^{n\Delta t^2\omega^2})$, where $\omega^2 = k/m$. This is a bad sign. In general, artificial damping terms must be added to avoid exponential growth but careful analysis reveals that when the damping is increased too much, the system is *again* unstable. In addition, the observed damping forces are generally much lower than the damping parameters would suggest due to the in combination with damping–in general much larger damping than ever occurs in reality. Tuning this for a complicated network of spring and dampers is a tedious and frustrating task hardly worth the effort.

The case with $n_x = 1$, $n_v = 0$ corresponds to the well known symplectic Euler method which has many remarkable properties. For one, it is computationally as cheap as the explicit Euler method. However, it is stable for integrating harmonic oscillator systems as long as the time step satisfies $\Delta t < 2/\omega = 2\sqrt{m/k}$, without any artificial damping. It does not exactly conserve the energy but it does exactly preserve another quadratic form which is close to the energy. This implies global stability at least for linear systems as the energy oscillates within bounds proportional to Δt^2. The reason for this conservation property is that it is the simplest example of a variational integrator [Kharevych et al. 2006], and like all such time stepping methods, it is symplectic, i.e., it preserves an integral invariant along the flow. But since this is not unconditionally stable, we expect to lose stability as soon as $\omega_m > 2/\Delta t$, where ω_m is the maximum frequency in the system. This can therefore not be applied to the high stiffness case.

To understand this limitation, consider an object of mass 1 *kg* and density roughly like water. Simulate the object with 1000 particles of equal masses connected with spring forces with stiffness constant k, and integrate this with fixed time step $\Delta t = 0.02$ *ms*. The stability requirement demands that $k \lesssim 2.5\ N/m$. The corresponding Young's modulus for this is roughly $Y \approx 250\ Pa$ which is extremely soft, as typical solids have Young's modulus ranging from 10 *MPa* up to 1000 *GPa*. At $Y = 250$ *Pa*, we are way off the scale. The

maximum stiffness become even less for a finer division, i.e., into more particles of lesser mass and shorter spring lengths. The way out is to integrate with smaller time steps or to use artificially small mass density, or to modify other physical parameters to achieve the desired visual result. But this only works if one is not interested in any sort of validation.

The case $n_x = 1$, $n_v = 1$ corresponds to the first order implicit Euler method. This is well known to be the most stable method in the book if the nonlinear system is solved accurately. It is also unconditionally stable for positive time step for the simple harmonic oscillator. When the nonlinear system of equations is solved approximately after a linearization, stability is no longer unconditional but still quite good. To do this, approximate the forces F_{n+1} with a Taylor expansion around the current state x_n, v_n to get the stepping formula:

$$x_{n+1} = x_n + \Delta t v_{n+1} \tag{18}$$

$$\left[1 - \Delta t^2 M^{-1} \frac{\partial F_n}{\partial x}\right] v_{n+1} = v_n + \Delta t M^{-1} F_n, \tag{19}$$

where the force derivatives are evaluated at time step n. The expression in the bracket on the left hand side of Eq. (19) we denote by $\mathbb{S}'_n$. This is a matrix of dimension $3N \times 3N$, that is typically sparse for the systems we are considering. For clarity we have discarded possible dependence on velocity in the force but this extension is simple to perform and well covered in the graphics literature. In particular the famous paper [Baraff and Witkin 1998] applies this to cloth simulations defined as networks of point masses attached with spring-damper. As observed in [Baraff and Witkin 1998], this is not unconditionally stable even for simple springs but stable enough to take much larger steps than would be possible with the symplectic Euler method, for instance. Solving the linear system of equations is not terribly expensive either and again, a good solution for speeding this up is provided in [Baraff and Witkin 1998].

We could of course consider this strategy to discretize the penalty forces or the regularization terms of (6). The problem lies elsewhere though. The implicit Euler method is stable because it artificially dissipates the energy of the system, faster with larger time step. On a simple linear harmonic oscillator, this artificial dissipation is so large that it significantly alters the observed oscillation frequency. Even worse, if one simulates a simple pendulum using a point mass attached to a stiff spring, integrating with the implicit Euler method alters the pendulum oscillation frequency and the observed acceleration of gravity, which should have nothing to do with the action of the stiff force! This artificial damping is immediately noticeable in cloth simulation, especially when comparison is made with energy preserving methods. This fact has not been reported in the graphics literature as an anomaly but is considered a welcome imitation of natural occurrences of friction although there is little control over it. We demonstrate how severe this damping can be in section 3.

When considering regularized equations of motion (7)–(9), we have additional variables to consider. Starting with the symplectic Euler parameters $n_x = 1$, $n_v = 0$ in (16–17), we still need to provide a correct interpretation of the regularized connection (9). We mentioned previously that penalty forces converge weakly to the constraint forces in the limit where $\alpha \to 0$. This means in particular that the time averages: $\tilde{\lambda} = (-1/\Delta t)\int_t^{t+\Delta t} dt\, \alpha^{-1}[\phi + \beta\dot{\phi}]$ should be well behaved. Formally integrating (8) over the time interval from t to $t + \Delta t$, where $t = n\Delta t$, we approximate $\int_t^{t+\Delta t} dt F_c \approx J_n^T \tilde{\lambda}$ and $\alpha\tilde{\lambda} = \Delta t^{-1}\int_t^{t+\Delta t} dt[\phi + \beta\dot{\phi}] \approx \Delta t^{-1}\phi_n + (1/2 + \Delta t\beta)J_n v_{n+1}$. Collecting the terms, we have:

$$x_{n+1} = x_n + \Delta t v_{n+1}$$

$$\begin{bmatrix} M & -J_n^T \\ J_n & \gamma\Delta t^{-2}\alpha \end{bmatrix} \begin{bmatrix} v_{n+1} \\ \Delta t\tilde{\lambda} \end{bmatrix} = \begin{bmatrix} Mv_n + \Delta t F_n \\ -\gamma\Delta t^{-1}\phi_n \end{bmatrix}, \tag{20}$$

where $\gamma = (1/2 + \Delta t\beta)^{-1}$. This linear equation may either be solved directly using a sparse matrix solver, or by first building the Schur complement, $\mathbb{S}_n \equiv J_n M^{-1} J_n^T + \Delta t^{-2}\alpha$, and solve for the Lagrange multiplier from

$$\mathbb{S}_n\tilde{\lambda} = -\Delta t^{-1} J_n v_n - J_n M^{-1} F_n - \gamma\Delta t^{-2}\phi_n \tag{21}$$

and then compute the velocity from the top row of equation Eq. (20) and finally the positions are updated. The stability of this method has not been analyzed theoretically, but as we will see in the numerical examples below, it appears to be strongly stable. Computationally, it is no more expensive than the linear implicit Euler stepper. In fact, since we are skipping all Jacobian derivative terms which are used in the linear implicit Euler formulation, it is somewhat faster, and far, far simpler to implement.

3 An elucidating example

Consider a cube composed of eight particles and five tetrahedra, producing a total of 18 edges as shown in Fig. 2 a). We use this example to illustrate both the strength of the new formulation and statements made about numerical integrators in Sec. 2.4. As a model of elastic deformable material we assign energy distance functions for $U(x) = \sum_l (1/2) k \phi_l^2$ with $\phi_l \equiv (||\mathbf{x}_{(i_l)} - \mathbf{x}_{(j_l)}|| - L_l)/L_l$, where l labels each of the 18 edges in the mesh with respective rest lengths L_l, and i_l, j_l are the labels of the two nodes and the end of edge k. The cube is dropped from rest and subjected to downward gravitational acceleration of magnitude $g = 10\ m/s^2$. One of the particles is fixed to the world so the cube swings back and forth at the same time as it undergoes elastic oscillations under its own weight. The mass of the particles are $m = 1\ kg$ and stiffness is set to $k = 10^5\ Nm$, yielding natural internal oscillation periods of $0.014\ s$. This is a nearly rigid case and the elastic deformation are expected to be small–of the order $\phi \sim 10^{-3}$–and hardly noticeable visually. We integrate the system with the three time steppers described in section 2.4 with time step $\Delta t = 0.05\ s$, and for a reference solution with $\Delta t = 0.005\ s$. The result is displayed in Fig. 2 and 3. The standard symplectic Euler stepper explodes almost instantly for $\Delta t = 0.05\ s$ so only the first state is shown in Fig. 2 a). Fig. 2 b) displays snapshots from a simulation with the linear implicit Euler stepper and our new regularized stepper in Fig. 2 c), both for time step $\Delta t = 0.05\ s$. The final position of the cube after a given number of steps is visibly different for the different steppers. In particular, the swinging motion is slower when using the linearly implicit Euler stepper than for the regularized one.

This is illustrated more precisely in Fig. 3 where the heights of a given particle is plotted as functions of time for all three methods using $\Delta t = 0.005\ s$ and for the two stable methods for $\Delta t = 0.05\ s$. A striking feature is that the *rate of fall* is slowed down by the linearly implicit Euler method when using the larger time step, taking nearly 35% more time to reach the minimum position, and this phenomena gets proportionally worse when either the time step or the stiffness is increased. The artificial damping of the integrator not only models linear drags in the direction of the spring forces but affects the physics of the *entire* system, even free fall under gravity! Worse yet, even when the time step is reduced by a factor of 10, down to less than one third of the natural frequency of the internal springs, damping is still of the order of 10% per cycle.

When using the smaller time step, the symplectic Euler stepper reproduces the same solution of as the regularized model. However, the regularized stepper nearly produces same trajectory for *both* time steps!

a) *b)* *c)*

Figure 2: An elastic cube anchored at one node, falling under gravity, is integrated with three different steppers: *a)* symplectic Euler stepper, that explodes almost instantly, *b)* linearly implicit Euler stepper and *c)* regularized stepper. The cubes are displayed at the time points $t = 0.0, 0.2, 0.4, 0.8$ *s* and integrated with large time step of $\Delta t = 0.05$ *s*; nearly 3 times larger than the natural oscillation period of the structural springs. For the linear implicit Euler stepper numerical damping makes the cube fall slower than with the regularized stepper.

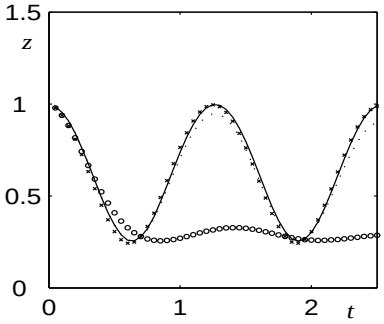

Figure 3: The height of one of the particles in the swinging cube as function of time. The severe damping of the linear implicit Euler stepper is clear. The motion is integrated with the linear implicit Euler stepper (circles for $\Delta t = 0.05$ *s*, dotted line for $\Delta t = 0.005$ *s*) and the regularized symplectic Euler stepper (crosses for $\Delta t = 0.05$ *s*, solid line for $\Delta t = 0.005$ *s*).

The observations are summarized as follows. For a given time step Δt, the symplectic Euler stepper is only stable as long as $k < 4m/\Delta t^2$. In other words, the time step must satisfy $\Delta t < T/\pi$ where T is the natural period of the oscillators, given by $T = 2\pi/\omega, \omega^2 = k/m$. For high stiffness, or short natural period, the solutions escapes to infinity. The linearly implicit Euler stepper is stable for much larger range of stiffness (which is difficult to determine exactly) but it adds artificial and spurious damping to the system. The result is that even the rigid body motion of very stiff materials are damped, producing noticeably inaccurate trajectories. The regularized symplectic Euler, on the other hand, produces an accurate trajectory even in the stiff limit and for large time steps. Some of this numerical behavior can be understood by comparing the matrices involved in the linear system of equations in Eq. (21) and (19), namely:

$$\mathbb{S}_n \equiv J_n M^{-1} J_n^T + \gamma \Delta t^{-2} \alpha \tag{22}$$

$$\begin{aligned}\mathbb{S}'_n &\equiv 1 - \Delta t^2 M^{-1} \frac{\partial F_n}{\partial x^T} \\ &= 1 + \Delta t^2 M^{-1} \left(J_n^T \alpha^{-1} J_n + \frac{\partial J_n}{\partial x^T} \alpha^{-1} \phi_n \right)\end{aligned} \tag{23}$$

In the limit where $\alpha \to 0$, the first of these matrices has the nice limit: $\mathbb{S}_n \to J_n M^{-1} J_n^T$. This is the same matrix equation that must be solved to integrate the purely constrained system. However, the matrix defined in Eq. (23) goes to infinity. Looking carefully at matrix $\mathbb{S}'_n$, it can fail to be positive definite, numerically at the very least, even for moderately large values in α^{-1}, at which point the stepping is unstable and can diverge. As the identity term in $\mathbb{S}'_n$ becomes negligible in comparison with the α^{-1} terms, we get the wrong physics due to numerical errors in solving the badly scaled system of equations.

4 Simulating elastic deformable materials

We are proposing and investigating the combination of physically based material energies and constraint regularization technique for stable time stepping in simulations where the elasticity may range from very soft to very stiff. Here we present the details in constructing an actual simulation based on these ideas. We have identified the symplectic Euler stepper in combination with constraint regularization as a suitable numerical integrator. As mentioned earlier, length and volume constraints (e.g. spring-and-damper models) is a possibility, but is associated with an uncontrolled mixing of material parameters. This makes it impractical to adjust the parameters for the object to behave as a specific material. Instead we propose using constraints (or rather regularized with energy functions) based on elasticity theory, that have been described in section 2.3. We divide the object into a mesh of N_T tetrahedra and N nodes, or particles, as we did for the box in Fig. 2. Each tetrahedron has four node particles $(a) = (1), (2), (3), (4)$. For each tetrahedron we compute local quantities, e.g., strain and energy. These are added to global quantities. The *global* constraint vector ϕ and *global* Jacobian J then has dimensions $6N_T$ and $6N_T \times 3N$. The local quantities are constructed by first defining the *local* position vector $\tilde{x}^T = (\mathbf{x}^{(1)^T}, \mathbf{x}^{(2)^T}, \mathbf{x}^{(3)^T}, \mathbf{x}^{(4)^T})^T$ which is of dimension 12. We then define the local strain $\tilde{\varepsilon}$, that is of dimension 6 and in turn the local constraint deviation $\tilde{\phi}$ (also this of dimension 6) and local Jacobian $\tilde{J} = \partial\tilde{\varepsilon}/\partial\tilde{x}^T$, having dimension 6×12. The algorithm for the proposed method is

initialize positions $x = x_0$ and velocities $v = v_0$
construct tetrahedral mesh
for each time step $n = 0, 1, 2, \ldots$ **do**
  accumulate external forces
  loop {all tetrahedra}
    read off particles $(a) = (1), (2), (3), (4)$
    get local position vector $\tilde{x}$
    compute $\tilde{\phi} \equiv \tilde{\varepsilon}$
    compute $\tilde{J} = \partial\tilde{\varepsilon}/\partial\tilde{x}^T$
    add local $\tilde{\varepsilon}$ and $\tilde{J}$ to global matrix equation (20)
  end loop
  solve linear equation (20) $\to v_{n+1}$
  update $x_n \to x_{n+1}$
end for

The global matrix equation here, can be either Eq. (20) or Eq. (21), depending on choice of linear equation solver. The most technical part is the computation of the local Jacobian and solving the linear system of equations. Some of the steps in computing the Jacobian are given in the Appendix.

5 Results

We now present results of validation and performance tests from an implementation of our new method.

5.1 Visual checks

Casual visual observations of the method in action reveal several nice properties, even with just a few tetrahedra. Still frames of simulation are shown in Fig. 4, where a beam made of 88 nodes and 105 tetrahedra is deformed by the action of a twisting force. The method is clearly capable of handling large deformations with a visually pleasing result, i.e., without kinks of collapsing regions. The same beam is illustrated in Fig. 5 in which the *nonlinear* terms in the Green strain, Eq. (13), were omitted. These terms are conventionally omitted when the displacements are only small. Such a linearization, whenever possible, largely improves the computational efficiency of the method. In that case the matrix in Eq. (20) is constant and its inverse may be precomputed to allow fast velocity updates. In the case of Fig. 5, the deformation is clearly too large for omitting the nonlinear terms. Otherwise, the result is an unrealistic volume deformation. We also show results for a supported beam bending under gravity in Fig. 6 (with mass 100 *ton*) and Fig. 7 (with mass 900 *ton*). In both cases we have Young's modulus $Y = 1.0$ *GPa* and the Poisson ratio is set to $\sigma = 0.25$. As expected, the material responds with stiffness increasing proportionally with the Young's modulus.

Figure 4: The result of a twisting force acting on the short ends of an elastic deformable beam.

Figure 5: The same setup as in Fig. 4, but with the nonlinear contribution to the strain omitted. The result is an unrealistic volume deformation.

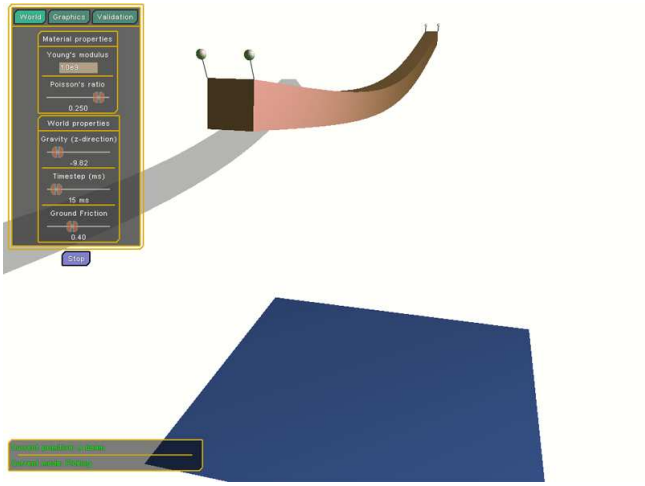

Figure 6: A supported beam of mass 100 *ton*, $Y = 1.0$ *GPa* and $\sigma = 0.25$.

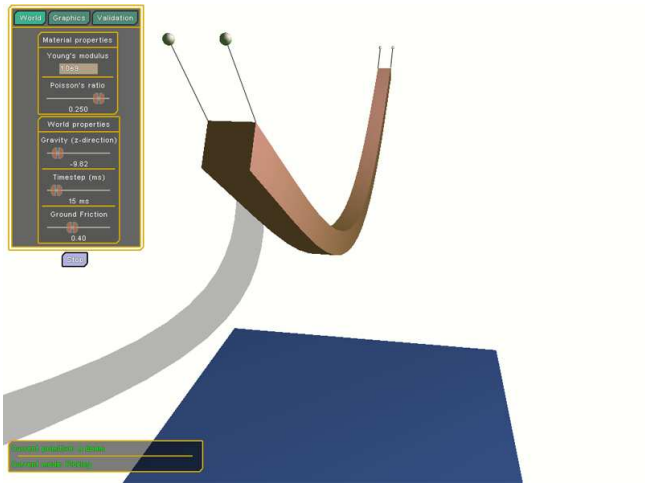

Figure 7: A supported beam of mass 900 *ton*, $Y = 1.0$ *GPa* and $\sigma = 0.25$.

5.2 Physical performance

Next we validate the physical behavior by conducting pull/push and twist tests and measuring the change in length and rotation for given applied forces. We validate against the theoretical relations between applied force and deformations from elasticity theory, i.e., Hooke's law for pull/push $\Delta L/L = F/YA$ and twist $\theta = 2\tau L(1+\sigma)/YK$, where L is the rest length, F is the applied force, A is the cross-section area, τ is the twisting moment, L is the length and K is a geometrical factor (torsion section constant). The result of these tests performed on a beam of 88 nodes and 105 tetrahedra are shown in Figs. 8 and 9. The results fit theory, but with a clear dependence on geometry/mesh. This is arguably, owed to the fact that we are

using 4-node tetrahedral mesh with *linear* shape function–simplest possible choice–which is known to be associated with particularly large mesh dependence. Using either finer discretizations or higher order shape functions should improve this. We also confirm that incompressibility is achieved for $\sigma = 0.5$.

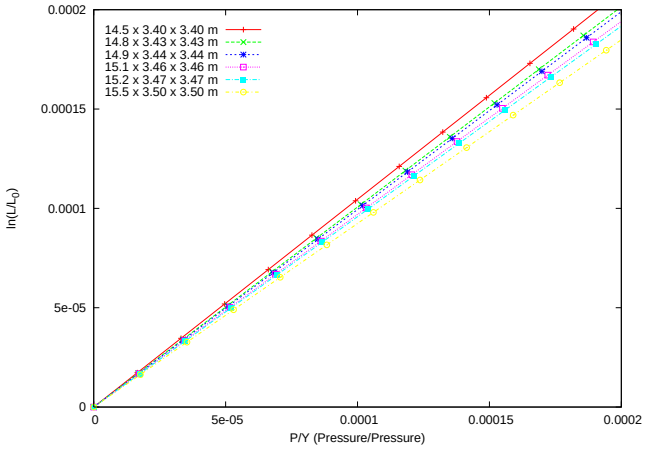

Figure 8: The pull/push test for five slightly varying geometries.

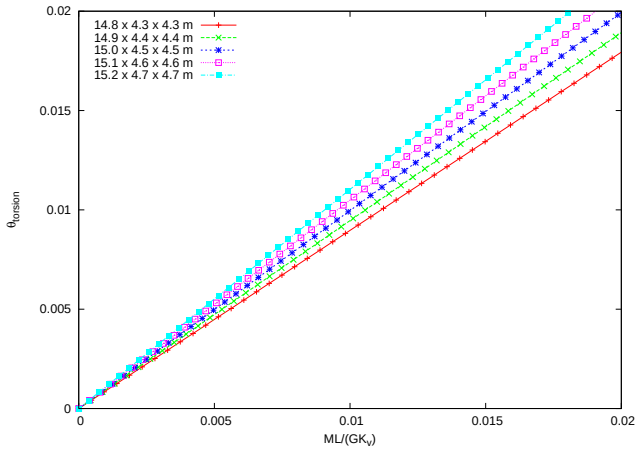

Figure 9: The twist test for five slightly varying geometries.

5.3 Computational properties

We consider two aspects of computational performance: stability and efficiency. The regularized symplectic Euler integrator is self-stabilizing. This means that, much like the linear implicit Euler method, the simulation dissipates energy and approaches a state of equilibrium. It should be emphasized that the dissipation is restricted to the *internal motion* and does not affect rigid motion, which the linear implicit Euler method does, c.f. the swinging cube example in section 3. In real-time applications with time step $\Delta t = 0.015$ *s* the simulations are stable for stiffnesses even well above that of diamond $Y = 1000$ *GPa*. We achieve stable simulation of very stiff materials under large tension and large deformations. It should be pointed out, however, that the time scale of the dissipation of internal vibrations becomes very short for stiff materials. In Fig. 10 we show results from a soft beam attached in one end and swinging freely under gravity. This figure confirms that the method is self-stabilizing. No energy dissipative terms have been added to the system, still, the system gradually relaxes to a state of equilibrium. The rate of numerical dissipation of the internal motion increases with decreasing size of the time step and with increasing stiffness. To resolve the issue of damped internal motion, the time stepper can be replaced with an energy preserving, though computationally more expensive, time stepper, e.g., a variational integrator as presented in ref. [Kharevych et al. 2006].

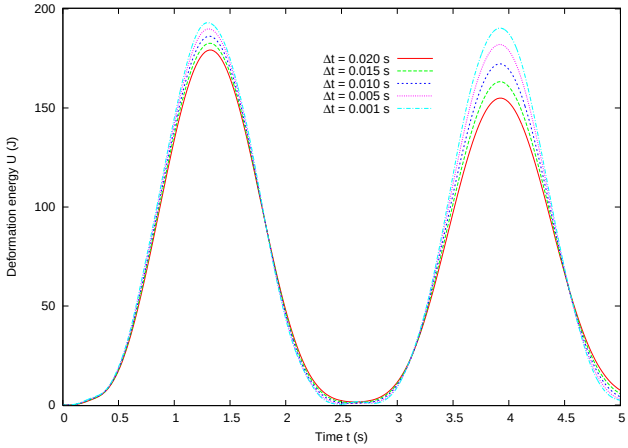

Figure 10: A beam is attached at one end and is swaying freely in a uniform gravitational field. This shows how the deformation energy vary over time at different sized time steps. Material parameters used are $Y = 8.0$ *kPa* and $\sigma = 0.25$ applied on the 10 kg beam.

Next we consider the efficiency of the method and how the computational time scales with system size. In this implementation we build and solve the Eq. (21) rather than Eq. (20) directly, making use of that the matrix is banded and achieve linear in time dependence on the number of particles in the system. The computational bottlenecks are recognized as the computation of the Jacobian and constraint, building and factorizing the matrix equation (21) and solving the equation. The contribution of these three processes in a simulation of a beam is displayed in Fig. 11. The simulation was run on an Intel Pentium 4 2.4 GHz CPU, with 1025 Mb DDR2 RAM internal memory. Most time, in our implementation, is consumed on factorizing the Schur complement matrix rather than solving it. The method we present clearly scales linearly with system size and real-time simulation (with time step of 15 ms) can be achieved for systems with size up to nearly 200 tetrahedra. There are several ways to improve the efficiency further, besides more efficient building and factorization of Eq. (21). In the computation of the new velocity, from Eq. (20), it is not necessary to build the Schur complement and first solve the Lagrange multiplier. Although Eq. (20), is larger in dimension than Eq. (21), it is a saddle-point matrix with a well ordered and sparse structure. There are efficient techniques, with linear complexity, for factorizing the matrix and solving this equation. The efficiency may be further increased using vector/parallel hardware, e.g., performing computations on the graphics processing unit. We estimate that with these optimizations real-time simulation of systems of the size of 1000 tetrahedra should be possible with currently available hardware.

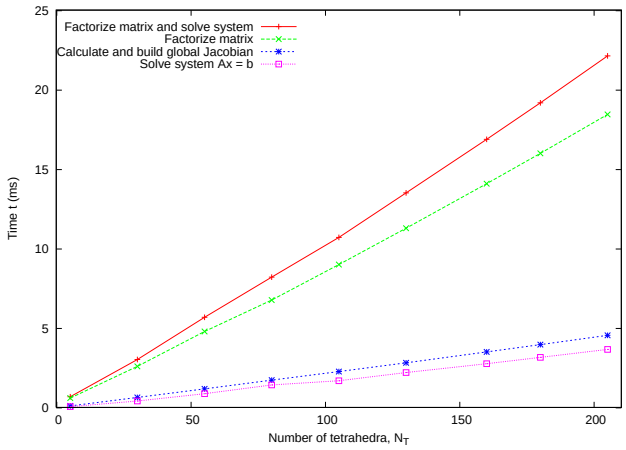

Figure 11: The computational time for the bottlenecks for a time step and the dependence on the number of tetrahedrons. The method we present has linear in time complexity and real time simulation is achieved with system size of about 200 tetrahedra.

6 Summary and Conclusions

We have constructed and investigated a method for stable simulations of elastically deformable objects at interactive rate with elasticity ranging from very elastic to highly stiff. With the novel combination of constraint regularization based on realistic material energies in combination with a symplectic Euler stepper we achieve stable time stepping for any value of stiffness of practical use, including the incompressible limit. This approach is compared to using linear implicit Euler, which damps all motion strongly in this limit. Any of the conventional integration methods for soft materials, including the standard symplectic Euler, has an upper limit on material stiffness for each size of time step - beyond this value the simulation becomes unstable.

Energy functions based on elasticity theory gives a direct relation between simulation parameters and real world measured material parameters. Simulated objects respond as expected to external forces and changes in the material parameters, e.g., Hook's law is obeyed and the material turns incompressible for the Poisson ratio $\sigma = 0.5$.

The method is self stabilizing. This guaranties stable simulation even in the regime of very stiff materials undergoing large deformations. On the downside, there is numerical, and thus artificial, damping in the system. Internal vibrations damp out very quickly for large material stiffness but rigid motion is unaffected. The numerical damping in our new method is thus less severe than for the linear implicit Euler method. The issue with numerical damping can be resolved by using time stepper that preserves this symmetry in the equations to a higher degree. These time steppers, known as variational integrators [Kharevych et al. 2006], are however more computationally expensive, they involve solving a non-linear system of equations rather than a linear system.

The method we present here scales linearly with system size. In the current implementation we achieve real-time performance of systems of the size of 200 tetrahedra. Performance can be improved by using other solver strategies, e.g., combining an iterative method and a good preconditioner, and utilizing vector/parallel hardware. We estimate that real-time simulation of systems of the size of 1000 tetrahedra should be possible with currently available hardware.

In comparison with other methods, it should be emphasized that the efficiency of the method we propose is *independent on the material stiffness* and not flawed by instabilities in the stiff regime.

In future work, we will improve the scaling and performance of the method, extend it to other type of constraints, e.g. contact constraints and improve the time stepping to reduce numerical dissipation without compromising stability and speed. Also, we will pursue issues of plasticity, mesh elements and adaptive level of detail techniques.

7 Acknowledgments

The research was supported in part by ProcessIT Innovations, "Objective 1 Norra Norrlands" EU grant awarded to HPC2N/VRlab at Umeå University, by Vinnova and the Foundation for Strategic Research (#V-247) and by the Swedish Foundation for Strategic Research (SSF-A3 02:128).

8 Appendix

Here we give some of the details in computing the Jacobian based on the energy function given in Eq. (15) from elasticity theory. Computing the Jacobian is part of building the equation 21. First, we order the strain vector ε in its *normal* components $\varepsilon_{normal} \equiv (\varepsilon_{11}, \varepsilon_{22}, \varepsilon_{33})^T$ and *shear* components $\varepsilon_{shear} \equiv (\varepsilon_{12}, \varepsilon_{13}, \varepsilon_{23})^T$ such that $\varepsilon = (\varepsilon_{normal}^T, \varepsilon_{shear}^T)^T$. We compute local quantities, e.g., the strain and contribution to the Jacobian for each tetrahedron. The local Jacobian is identified from the relation $\tilde{\phi}_i = \tilde{J}_{ij}\dot{\tilde{x}}_j$, where $j = 1, 2, \ldots, 12$. Using $\phi = \varepsilon$ we identify

$$\tilde{J}_{ij}^{normal} = \Lambda_{iji} + \left(\frac{\partial u_k}{\partial r_i}\right)\Lambda_{kji} + \frac{\tilde{\varepsilon}_i^{normal}}{2\sqrt{V}}\frac{\partial V}{\partial \tilde{x}_j} \quad (24)$$

$$\tilde{J}_{ij}^{shear} = \frac{1}{2}\Gamma_{imk}\Lambda_{mjk} + \frac{\tilde{\varepsilon}_i^{shear}}{2\sqrt{V}}\frac{\partial V}{\partial \tilde{x}_j} + \frac{1}{2}\chi_{imk}\left[\left(\frac{\partial u_n}{\partial r_k}\right)\Lambda_{njm} + \left(\frac{\partial u_n}{\partial r_m}\right)\Lambda_{njk}\right] \quad (25)$$

where

$$\Lambda_{ijk} \equiv \sqrt{V}\frac{\partial}{\partial r_k}\frac{\partial u_i}{\partial \tilde{x}_j}. \quad (26)$$

In all of these expressions the ranges of the indexes are $i = 1, 2, \ldots, 6$, $j = 1, 2, \ldots, 12$ and $k, m, n = 1, 2, 3$. For notational convenience we have introduced the constant matrices χ and Γ that are both of size $3 \times 3 \times 3$ and with the nonzero elements

$$\chi_{112} = \chi_{213} = \chi_{323} = 1 \quad (27)$$

$$\Gamma_{112} = \Gamma_{121} = \Gamma_{213} = \Gamma_{231} = \Gamma_{323} = \Gamma_{332} = 1 \quad (28)$$

References

ALLEN, M. P. 2004. Introduction to molecular dynamics simulation. *N. Attig, K. Binder, H. Grubmller, and K. Kremer, editors, Computational Soft Matter: From Synthetic Polymers to Proteins, John von Neumann Institue for Computing, Jlich, Germany,.*

BARAFF, D., AND WITKIN, A. 1998. Large steps in cloth simulation. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 43–54.

BARAFF, D. 1996. Linear-time dynamics using lagrange multipliers. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 137–146.

BAUMGARTE, J. 1972. Stabilization of constraints and integrals of motion in dynamical systems. *Computer Methods in Applied Mechanics and Engineering 1*, 1, 1–16.

BRO-NIELSEN, M., AND COTIN, S. 1996. Real-time volumetric deformable models for surgery simulation using finite elements and condensation. *Computer Graphics Forum 15*, 3, 57–66.

ERLEBEN, K., SPORRING, J., HENRIKSEN, K., AND DOHLMANN, H. 2005. *Physics-based Animation*. Charles River Media, Aug.

FUNG, Y. C., AND TONG, P. 2001. *Classical and Computational Solid Mechanics*. World Scientific, Singapore.

HAUTH, M., GROSS, J., AND STRASSER, W. 2003. Interactive physically based solid dynamics. *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*.

KHAREVYCH, L., WEIWEI, TONG, Y., KANSO, E., MARSDEN, J. E., SCHRDER, P., AND DESBRUN, M. 2006. Geometric, variational integrators for computer animation. *to appear in ACM/EG Symposium on Computer Animation 2006*.

LACOURSIÈRE, C. 2006. A regularized time stepper for multibody simulations. *Internal report, UMINF 06.04, issn 0348-0542*.

LANDAU, L., AND LIFSCHITZ, E. 1986. *Theory of Elasticity*. Pergamon Press, Oxford, 3rd ed.

MARSDEN, J. E., AND WEST, M. 2001. Discrete mechanics and variational integrators. *Acta Numerica 10*, 357–514.

MULLER, M., KEISER, R., NEALEN, A., PAULY, M., GROSS, M., AND ALEXA, M. 2004. Point based animation of elastic, plastic and melting objects. *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation 14*, 15.

MULLER, M., HEIDELRBERGER, B., TESCHNER, M., AND GROSS, M. 2005. Meshless deformations based on shape matching. *ACM Transactions on Computer Graphics (ACM SIGGRAPH 2005)*.

NEALEN, A., MULLER, M., KEISER, R., BOXERMAN, E., AND CARLSON, M. 2005. Physically based deformable models in computer graphics. *Eurographics State-of-the-Art Report*.

RUBIN, H., AND UNGAR, P. 1957. Motion under a strong constraining force. *Communications on Pure and Applied Mathematics X:65-67*.

TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISCHER, K. 1987. Elastically deformable models. *Communications on Pure and Applied Mathematics 21*, 205–214.

TESCHNER, M., HEIDELBERGER, B., MULLER, M., AND GROSS, M. 2004. A versatile and robust model for geometrically complex deformable solids. In *CGI '04: Proceedings of the Computer Graphics International (CGI'04)*, IEEE Computer Society, Washington, DC, USA, 312–319.

UMFPACK. `http://www.cise.ufl.edu/research/sparse/umfpack/`.

WITKIN, A., GLEICHER, M., AND WELCH, W. 1990. Interactive dynamics. *Computer Graphics 24*, 2, 11–21.