Geometric Interpretation of Physical Systems for Improved Elasticity Simulations

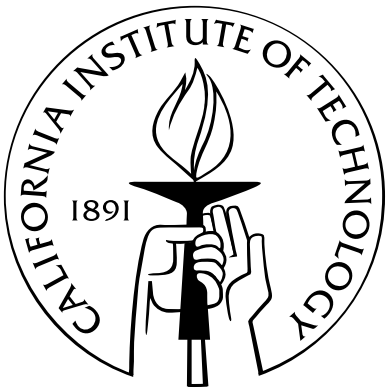
Thesis by

Liliya Kharevych

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy



California Institute of Technology

Pasadena, California

2010

(Defended 11 September 2009)

Мамі і Оксані.

Acknowledgements

First and foremost, I would like to thank my advisors, Peter Schröder and Mathieu Desbrun, for their guidance, support, encouragement, and enthusiasm. Next, I would like to thank the other members of my committee, Alan Barr, Houman Owhadi, Jerrold E. Marsden, for their suggestions, ideas, and advice. I would like to thank Jerrold E. Marsden and his colleagues for their work on variational time integrators, and Houman Owhadi and his colleagues for their work on homogenization that both became the basis for the work presented in this thesis.

I am very grateful to numerous people with whom I have had a pleasure to work over these years: Boris Springborn, Yiying Tong, Eva Kanso, Weiwei Yang, Patrick Mullen, Keenan Crane, Nathan Litke, and many others. They have been both friends and mentors providing me with great help and inspiration.

I would like to thank my family, especially my mom, my sister, and Andriy. They have always encouraged me to study and aim high, which led me on this path. Finally, I am thankful to my friends, in particular the ones that I met during these years at Caltech: the times that we spent hanging out, going out, working out, or just talking, has made the grad school years very enjoyable.

Abstract

The physics of most mechanical systems can be described from a geometric viewpoint; *i.e.*, by defining variational principles that the system obeys and the properties that are being preserved (often referred to as *invariants*). The methods that arise from properly discretizing such principles preserve corresponding *discrete invariants* of the mechanical system, even at very coarse resolutions, yielding robust and efficient algorithms. In this thesis geometric interpretations of physical systems are used to develop algorithms for discretization of both space (including proper material discretization) and time. The effectiveness of these algorithms is demonstrated by their application to the simulation of elastic bodies.

Time discretization is performed using variational time integrators that, unlike many of the standard integrators (*e.g.*, Explicit Euler, Implicit Euler, Runge-Kutta), do not introduce artificial numerical energy decrease (damping) or increase. A new physical damping model that does not depend on timestep size is proposed for finite viscoelasticity simulation. When used in conjunction with variational time integrators, this model yields simulations that *physically* damp the energy of the system, even when timesteps of different sizes are used. The usual root-finding procedure for time update is replaced with an energy minimization procedure, allowing for more precise step size control inside a non-linear solver. Additionally, a study of variational and time-reversible methods for adapting timestep size during the simulation is presented.

Spatial discretization is performed using a finite element approach for finite (non-linear) or linear elasticity. A new method for the coarsening of elastic properties of heterogeneous linear materials is proposed. The coarsening is accomplished through a precomputational procedure that converts the heterogeneous elastic coefficients of the very fine mesh into anisotropic elastic coefficients of the coarse mesh. This method does not depend on the material structure of objects, allowing for

complex and non-uniform material structures. Simulation on the coarse mesh, equipped with the resulting elastic coefficients, can then be performed at interactive rates using existing linear elasticity solvers and, if desired, co-rotational methods. A time-reversible integrator is used to improve time integration of co-rotated linear elasticity.

Contents

List of Figures

Chapter 1

Introduction

Describing the physical world using mathematic modeling has been a major scientific pursuit over hundreds of years. Our capacity to simulate and visualize physics took a great leap forward with the advent of the computer. Due to the discrete nature of computers, new algorithms for discretizing mathematical models needed to be developed. With continuously increasing computational power at our disposal, the physical systems that we are able to simulate are becoming increasingly more complex, calling for both more elaborate mathematical models of these systems and more robust discrete algorithms that realize them digitally.

Traditionally, discretization for physical simulations has been done by considering the final mathematical equations that model the physical system and deriving discrete approximations of the equations. These types of methods are often convergent under refinement; i.e., when the size of discrete element goes to zero they accurately model the continuous physical system; however, they often lead to poor solutions when used with coarse resolutions. In contrast, geometric discretization approaches consider the physical system from the unified geometry viewpoint and discretize the system using appropriate discrete geometry tools. The connections between physics and geometry have been extensively studied in the field of geometric mechanics, with fundamental principles first emerging from the work of Fermat, Newton, Euler, Lagrange, Hamilton, and many others. In geometrical approaches, the physical system is described through a variational principle that the system obeys and the main properties that are being preserved (often referred to as *invariants*). The methods that arise from properly discretizing such principles preserve corresponding *discrete invariants* of the mechanical system, even at very coarse resolutions, yielding robust and efficient algorithms.

The use of geometric discretization approaches for physically-based simulations in computer graphics is not purely an intellectual exercise; it leads to methods that have two important properties:

- they still converge to the correct physical solution in the limit of refinement,
- they are also robust to space and time discretization, preserving some of the most important invariants at very coarse resolutions.

These properties can be motivated further by providing some very basic examples:

Example: elasticity in one dimension. This example demonstrates the importance of considering the physics of the problem rather than just the final mathematical equations when discretizing a continuous problem. In graphics, a common way to represent an elastic medium is to approximate it with a series of masses and springs that obey Hooke's Law: $F = -k\Delta x$. Alternatively, from the elasticity theory point of view, such springs can be thought of as finite elements discretizing a continuous elasticity problem. The corresponding discrete equation is $F = -\frac{\lambda}{L_0}\Delta x$ for each element, where L_0 is the initial length of the element and λ is a constant that only depends on the intrinsic material properties. While both equations are equivalent for a particular resolution (given $k = \frac{\lambda}{L_0}$), the former will not converge to the correct solution as the mesh is refined without accounting for the dependance of k on the initial length of the spring. Even though this problem is easily solved by appropriate weighting during the refinement, it shows that physics equations should not be taken out of the context of the physical system being discretized when one seeks to achieve *resolution independent* models.

Example: time integration of a single spring. In this example, we show that even when different methods have the same accuracy and converge to the correct solution at the limit of refinement, methods that preserve discrete invariants of the system at coarse resolutions have great advantages over those that do not. Consider the time integration of the dynamics of a single spring. Its motion can be described using Newton's Second Law of Motion $F = ma$, or equivalently $\dot{x} = v$ and $\dot{v} = \frac{F}{m}$, where x is the position of the spring, and v is the velocity. Using the explicit Euler method to

discretize these ODE's gives update rules:

$$x_{k+1} = x_k + \Delta t v_k,$$
$$v_{k+1} = v_k + \Delta t \frac{F(x_k)}{m}.$$

Alternatively, one can observe that Newton's Second Law comes from the fundamental principles of mechanics and consider Hamilton's principle, which states that the dynamics of a physical system can be written as a variational, geometric problem in space and time. Discretizing Hamilton's principle using explicit quadrature and taking discrete variations leads to a different set of update rules:

$$x_{k+1} = x_k + \Delta t v_{k+1},$$
$$v_{k+1} = v_k + \Delta t \frac{F(x_k)}{m}.$$

Both sets of update rules are equivalent in terms of computational efficiency. The latter one, however, exhibits good long term behavior, keeping the energy of the system bounded (Section 3.2.12 will demonstrate that this property is linked to a symplectic nature of variational integrators). The former update procedure, instead, increases the energy of the system, causing simulations to blow up (the larger the timesteps that are used, the sooner the blow-up occurs).

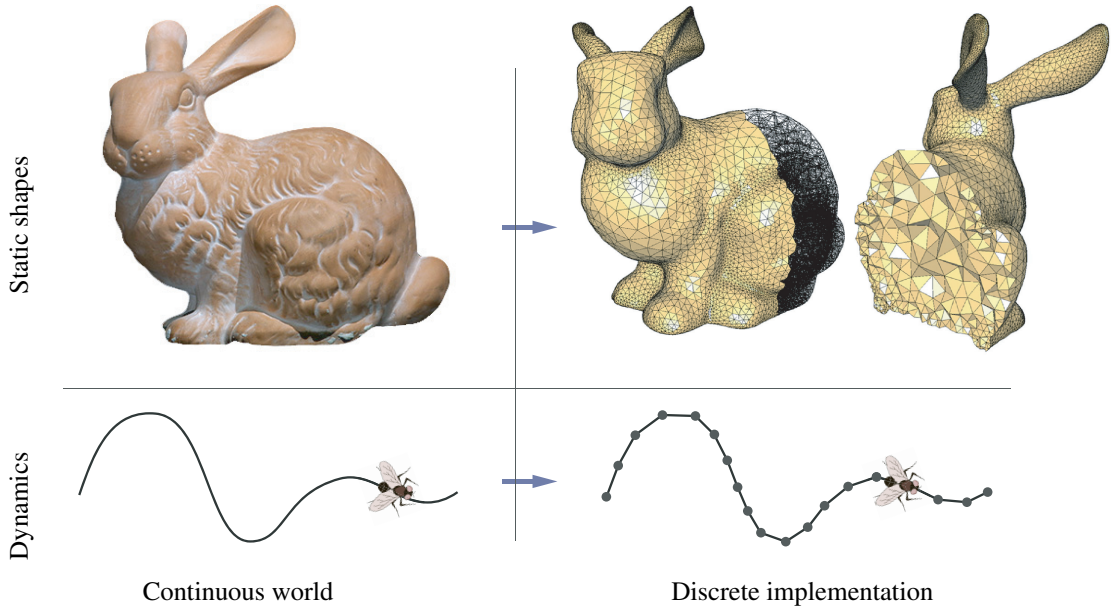


Figure 1.1: Discretization of space and time.

In this thesis, methods are developed for the discretization of both time and space, focusing on applications to elasticity simulations. The algorithms to be presented are efficient and can be used with very fine meshes and/or small timesteps, simulating the intricacies of continuous elastodynamics. However, they can also be applied to very coarse meshes and large timesteps for fast, real-time simulations that still effectively capture critical characteristics of the original system behavior. Such algorithms allow for a multi-resolution design process in computer animation, by which artists can tune simulations at coarse resolutions but do final runs at fine resolutions, adding more physical details without acquiring changes in the global behavior of animations. In addition to computer graphics, these algorithms can be used for applications in many other fields where efficient, resolution-independent, physical simulations are required; *e.g.*, for simulation of earthquake effects or virtual surgery.

1.1 Contributions

The time discretization problem is approached using structured time integrators developed during recent years in the field of geometric mechanics [27, 47, 70]. In particular, variational time integrators [70] are useful in many graphics applications, as they guarantee conservation of momentum and good energy behavior over long durations, which allows for stable and efficient simulations. Unlike the commonly used Implicit Euler integrator, these integrators do not introduce numerical damping. Since numerical damping is usually proportional to the timestep of the simulation, the same simulation run with different timesteps would have very different solutions. Instead, our new physically-based damping model does not depend on the timestep when used in conjunction with variational integration. This guarantees that using different timesteps will lead to similar solutions, an important property for preview purposes and real time simulations. We have also shown that the time update that arises when using variational integrators is a variational problem itself, allowing for more efficient numerical solvers. In this thesis, a time-adaptive integrator is also presented; *i.e.*, an integrator that automatically adjusts the size of the timestep as the simulation progresses. Variational and time-reversible approaches to this problem are introduced and compared.

During the spatial discretization process, our goal is to use the fewest degrees of freedom (nodes) possible, but still accurately model the physics of a continuous elastic body. This problem is most challenging when simulating heterogeneous elastic bodies; i.e., bodies that have spatially varying material properties, such as composite structures, porous materials, or biological tissues. Since the

material resolution of these bodies is often very fine, we cannot use a mesh that resolves all the material details if we want to achieve an efficient simulation. However, we often cannot ignore those details either. For example, a thin hard wire embedded in a large soft tissue will drastically change the elastic behavior of the tissue – thus, we must model the heterogeneous properties of such a composite material without incurring the excessive computational expense of directly resolving the wire. A new approach is proposed, based on a recent development in homogenization theory [56], to approximate a deformable object composed of arbitrary fine structures of various linear elastic materials with a dynamically-similar coarse model. Our method translates the heterogeneous elastic properties of a very fine mesh into anisotropic elastic properties of a coarse mesh that effectively capture the same physical behavior. Simulation of the coarse mesh, equipped with the resulting elastic coefficients, can then be performed at interactive rates using existing linear elasticity solvers. In computer animation, co-rotational methods are often used in conjunction with linear elasticity to allow for large rotational deformations without significant visual artifacts. Using co-rotational methods with variational integrators leads to a non-linear time update problem. To maintain the efficiency of the linear elasticity method, we introduce a time-reversible approach to updating elements' rotations. This approach keeps the energy and momenta of the system bounded, yielding stable, predictable, and efficient simulations.

1.2 Overview of the Content

Chapter 2 provides a summary of linear and non-linear elasticity models, gives details on their discretization using finite elements, and describes co-rotational methods to reduce numerical error associated with the linearization of elasticity. Section 2.3 presents our new physical structure-preserving damping model for finite viscoelasticity. Chapter 3 focuses on structure-preserving time integration, in particular variational and time-reversible integrators. Sections 3.3 and 3.4 describe structure-preserving integrators that automatically adjust the size of the timestep as during the simulation. In Section 3.4 we also present a time-reversible approach to time integration of systems that use co-rotational methods. In Chapter 4 we describe a numerical coarsening approach for the simulation of heterogeneous elastic bodies. Finally, conclusions, implications of this research, and future directions are presented in Chapter 5.

Chapter 2

Elasticity and Its Spatial Discretization

2.1 Introduction

To facilitate the understanding of the methods described in Chapters 3 and 4, this chapter gives a brief overview of elasticity theory and provides the information needed for implementation of described elasticity models. Section 2.2 summarizes discretization of non-linear (see also [10, 46, 55]) and linear (see also [18, 46, 43]) elasticity approaches. In Section 2.2.3 we describe and compare co-rotational methods introduced in [50, 31, 53, 19, 25]. Our novel method for the discretization of damping forces is described in Section 2.3. First, we give an overview of continuous elasticity theory and notations that are used throughout this chapter.

2.1.1 Continuous Formulation of Elasticity

An elastic body $\mathcal{B}$ undergoes reversible deformations (changes in shape) due to applied forces (see for example Figure 2.1). These may be *body forces* per unit volume or *surface traction* per unit area. Deformation typically depends on the material, size, and geometry of the body as well as the applied forces. A motion is a one-parameter (time) family of deformations and can be described by $x(X,t)$, where X denotes the position of a material particle of $\mathcal{B}$ in the reference configuration and t is time. That is, x is the particle position in the deformed or *current* configuration. Notice that usually capital letters (X, V, *etc*) are used to describe quantities in the reference (or undeformed) configuration, and small letters (x, v, *etc*) are used for the quantities in the current (or deformed) configuration. The kinetic energy of the body is given by:

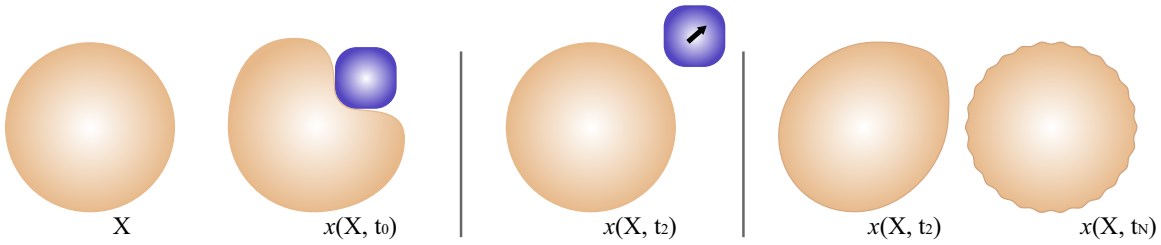


Figure 2.1: Behavior of an elastic body: (a) an elastic ball (undeformed configuration X) undergoes deformation x due to the external load at time t_0; (b) if the elastic material is damped, the ball will return to its undeformed configuration when the load is released; (c) if the elastic material has no or very little damping, the ball will keep deforming (preserving total kinetic and potential energy), and due to energy cascading, at some time t_N the deformation will be dominated by high frequency vibrations of the ball.

$$K = \frac{1}{2} \int_{\mathcal{B}} \rho \, v \cdot v \, \mathrm{d}V, \tag{2.1}$$

where ρ is the mass density, v is the velocity (function of material particle X and time t), and $\mathrm{d}V$ is a volume element. Further, in the pure mechanical theory of elasticity, there exists a *strain (or stored) energy density function* w per unit volume that represents the change in the internal energy due to mechanical deformations, which means that the potential energy (excluding gravity) is written as

$$W = \int_{\mathcal{B}} w \, \mathrm{dV}. \tag{2.2}$$

The rest state of the elastic body (static or quasi-static solutions) can be determined by minimizing potential energy (2.2) under prescribed traction or boundary conditions. Chapter 3 describes variational approach that can be employed to solve for the dynamic path using the Lagrangian ($\mathcal{L} = K - W$) of the elastic body.

In order to compute the potential elastic energy, the notion of the strain is usually required. For $2D$ or $3D$ the strain is computed and stored as a tensor that measures deformation in any direction. The strain tensor is computed using the deformation gradient F of the mapping $x(X, t)$:

$$F_{iJ} = \frac{\partial x_i}{\partial X_J}.$$

For example, in the case of non-linear elasticity *Cauchy strain* tensor $C = F^T F$ is most often used to measure the rotation-invariant strain $E = \frac{1}{2}(C - I)$. In the case of linear elasticity, the energy

density function is quadratic in the displacement and the strain is approximated using linearized strain tensor $\boldsymbol{\epsilon} = \frac{1}{2}(F + F^T) - I$. Then the energy density w is expressed as a function of a strain tensor, capturing a particular material law.

Another common formulation of elasticity uses the *Cauchy stress tensor* $\boldsymbol{\sigma}$ to describe the forces acting inside the body. Each component σ_{ij} of this tensor represents, pointwise, the i-component of the force on a surface with unit area whose normal is in the j-direction. Consequently, body forces are expressed by the divergence of the stress tensor, resulting in the static equation for elasticity

$$\operatorname{div} \boldsymbol{\sigma} = 0$$

under prescribed traction or boundary conditions, or equations of motion

$$\rho\, \ddot{\mathbf{u}} = \operatorname{div} \boldsymbol{\sigma} + \mathbf{f},$$

where $\mathbf{f}$ represents the external forces applied to the material.

2.1.2 Tensor Notation

Our exposition will make heavy use of tensors of various ranks as required in elasticity theory. To facilitate the direct implementation of our approach, we will often employ an index-based notation, where a rank-1 tensor X (*i.e.*, vector, or 1D array for coding purposes) has its components denoted as X_i (where i takes on the values $1, 2$, or 3), a rank-2 tensor Y (*i.e.*, a matrix, or 2D array) will have its components referred to as Y_{ij}, and so forth. The "Einstein" summation convention (where summation is implied by repeated indices [6]) will be assumed. We will also employ the concise notion of tensor *contraction*. A single contraction, where a summation over a single index is used, will be denoted as "$\cdot$"; $X \cdot Y$ will therefore refer to either a matrix-vector product $[X \cdot Y]_i = X_{ia} Y_a$ or a matrix-matrix product $[X \cdot Y]_{ij} = X_{ia} Y_{aj}$. A double contraction, where now a summation over two indices is performed, will be denoted as ":". It will be used for products of a rank-4 tensor by a rank-2 tensor, or products of two rank-4 tensors—$[F : G]_{ij} = F_{ijab} G_{ab}$, and $[F : G]_{ijkl} = F_{ijab} G_{abkl}$ respectively. Finally, a comma will indicate differentiation with respect to one or several of the coordinates as customary, while ∇ will refer to the gradient operator; *e.g.*, $(\nabla Y)_{ij} = Y_{i,j} = \partial Y_i / \partial x_j$ for a rank-1 tensor, $(\nabla Y)_{ijk} = Y_{ij,k} = \partial Y_{ij} / \partial x_k$ for a rank-2 tensor, etc. We will also use the

Kronecker delta δ function in our expressions, defined as:

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

Tensor-Matrix Conversion Tensors $\mathbf{T} = T_{abcd}$ of rank-4 in $3D$ (*i.e.*, $1 \leq a, b, c, d \leq 3$) contain 81 components, and are thus traditionally converted into a 9x9 matrix M through, assuming zero-based indexing:

$$T_{abcd} = M[3(a-1)+b,\ 3(c-1)+d]. \tag{2.3}$$

When $\mathbf{T}$ has minor symmetries ($T_{abcd} = T_{bacd}$ and $T_{abcd} = T_{abdc}$, like most of the tensors used in elasticity), one can further reduce the size of the representation by introducing a 6x6 matrix N such that:

$$N = R\,MR^T \quad \text{with } R = \begin{pmatrix} 1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1 \\ 0&1&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0 \end{pmatrix},$$

where R is called the *reduction matrix*. Converting back from this reduced matrix space to a full tensor is easily achieved as well: from a 6x6 matrix N, the equivalent 9x9 matrix M is found through:

$$M = E^T\,NE \quad \text{with } E = \begin{pmatrix} 1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1 \\ 0&1&0&1&0&0&0&0&0 \\ 0&0&1&0&0&0&1&0&0 \\ 0&0&0&0&0&1&0&1&0 \end{pmatrix},$$

where E is called the *expansion matrix*, and the final conversion to a tensor uses Eq. (2.3).

2.2 Discretization of Elastic Bodies

In this section we discuss the differences between non-linear, linear, and co-rotational elasticity models. We provide details on spacial discretization of corresponding elastic energies, forces, and stiffness matrices on simplicial meshes using finite element method. In addition, a novel structure preserving internal damping model is described. Discrete damping forces that arise from this model do not depend on the size of the time step of the simulation and preserve linear and angular momenta.

2.2.1 Finite Elasticity

Non-linear Material Laws There are two main advantages for using non-linear (finite) elasticity comparing to linear elasticity. First, the finite Lagrangian strain tensor

$$E = \frac{1}{2}(C - I) = \frac{1}{2}\left(\left(\frac{\partial x}{\partial X}\right)^T\left(\frac{\partial x}{\partial X}\right) - I\right)$$

measures a strain independent of the rotation of the rigid body, *i.e.*, when the elastic body undergoes rigid body rotation, this strain tensor does not change (unlike the linearized strain tensor). Second, the expression for potential elastic energy function can be non-linear, covering a wider range of material types. For example, many materials become stiffer when they are compressed, and such behavior can only be captured with a non-linear material law. For these reasons, finite elasticity is preferred for many applications in graphics when large deformations occur.

It is common to write non-linear elastic energy function w in terms of the right Cauchy-Green strain tensor C:

$$W = \int_{\mathcal{B}} w(C)\,\mathrm{dV}.$$

More specifically for isotropic materials, w can only depend on the three invariants I_1, I_2 and I_3 of the Cauchy-Green strain tensor:

$$I_1 = \mathrm{tr}(C), \quad I_2 = \mathrm{tr}(C^2) - \mathrm{tr}^2(C), \quad I_3 = \det(C).$$

The Jacobian of the deformation $J = \det(F) = \sqrt{\det(C)}$ can be often seen in the expressions for the w, representing the change of the volume due to the deformation. The function $w(I_1, I_2, I_3)$ varies depending on the material type, for instance, for Mooney-Rivlin materials, $w = a_1(I_1-3)+b_1(I_2-3)$, and for neo-Hookean materials $w = a_1(I_1 - 3) + b_1(\sqrt{I_3} - 1)^2$. For all the non-linear elasticity examples shown in this thesis we will use a modified neo-Hookean model [10]:

$$\mu\left(\frac{tr(C)}{\sqrt[d]{J^2}} - d\right) + \frac{1}{2}\kappa(J-1)^2, \tag{2.4}$$

where d is a number of dimensions (2 for 2D and 3 for 3D). The first part of this energy can be seen as isochoric, penalizing shears, and the second part is volumetric, penalizing changes in volume.

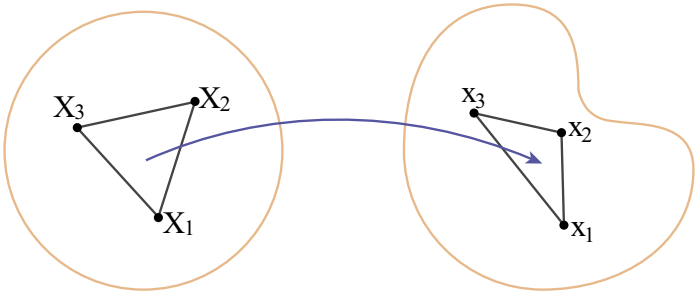


Figure 2.2: Discretization of an elastic body: the deformation of the discrete nodes is computed on the vertices of the mesh and interpolated using basis function inside the elements. For simplicial elements (triangle and tetrahedrals), linear basis functions are most often used in graphics.

For implementation convenience we will provide the expressions for the gradient and the Hessian of this energy in the Section 2.2.1. Notice that the gradient of this energy also corresponds to the negated body forces in the current configuration.

Discretization and Differentiation The purpose of this section is to provide the reader with more thorough details on the discretization of the finite elasticity energy, forces, and Jacobian of the forces. In particular we will concentrate on 3D elastic bodies discretized using tetrahedral finite elements. Such elements allow us to discretize shapes with complicated boundaries, approximating them much better and with lower element count comparing to regular grid discretization. We use variational tetrahedral meshing technique, see for instance [68], to obtain good quality meshes suitable for FEM simulations. To achieve the fastest simulations, we interpolate deformation inside the elements using linear basis functions. Higher order basis functions (as in [26, 48]) can also be used, however for simplicity and efficiency reasons we will concentrate on the linear basis function in this thesis. Given a tetrahedral mesh $\mathbf{D}$ (that approximates an elastic body) with M vertices (nodes) and E tets (elements), the original positions of the vertices X_i^n define the reference configuration of the body. Note that we will use superscripts to refer to the indices of the vertices in the mesh, ranging from 1 to M or the indices of the vertices in the tet, ranging from 1 to 4; subscripts will refer to the coordinate directions x, y, z, ranging from 1 to 3 in 3D. As the mesh evolves in space, new positions x_i^n define the current configuration of the body. The deformation can be interpolated inside the elements using nodal basis functions $\mathbb{N}$. For linear basis functions this amounts to a simple barycentric interpolation of positions inside each tet: $x = \sum_{p=1}^{4} \mathbb{N}(X^p)x^p$, with p ranging over the 4 vertices of the tet. Such linear interpolation leads to the deformation gradient F that is constant per tet and is

computed as

$$F_{ij} = \sum_{p=0}^{4} \nabla \mathbb{N}_j^p x_i^p,$$

where $\nabla \mathbb{N}^p$ is the gradient of the basis function associated with the node p computed in the reference configuration. Geometrically the gradient of the basis function of the node p can be seen as a vector perpendicular to the face opposite to p pointing outwards the tet, and can be computed as a cross product of two edges of this face divided by the volume of the tet.

The potential energy density function can now be computed for each tet and discretely integrated over the body by multiplying by the undeformed volume of the tet $\mathrm{Vol}(e)$ and summing over all the tets:

$$W = \int_{\mathcal{B}} w(C)\, \mathrm{dV} = \sum_{t=1}^{E} w(F(e)^T F(e))\ \mathrm{Vol}(e). \tag{2.5}$$

Internal elastic forces are the negative of the gradient of the energy with respect to the deformed positions, which can be computed using the chain rule:

$$\nabla W_i^n = \sum_{e \in \mathcal{N}(n)} \frac{\partial w^e}{\partial F_{qp}} \frac{\partial F_{qp}}{\partial x_i^n}\ \mathrm{Vol}(e), \tag{2.6}$$

where $\mathcal{N}(n)$ is the set of 1-ring neighboring elements of the vertex n, the dependance of w on F is assumed, and F is short for F^e. Solving quasi-static or implicit dynamic problems using Newton's method often requires the computation of the Hessian of the energy (sometimes referred to as the Jacobian of the forces), which is obtained by differentiating the forces again with respect to the deformed positions:

$$\nabla^2 W_{ij}^{nm} = \sum_{\substack{e \in \mathcal{N}(n) \\ e \in \mathcal{N}(m)}} \frac{\partial^2 w^e}{\partial F_{qp} \partial F_{lk}} \frac{\partial F_{qp}}{\partial x_i^n} \frac{\partial F_{lk}}{\partial x_j^m}\ \mathrm{Vol}(e). \tag{2.7}$$

In these expressions Einstein summation notation is used, *i.e.*, there are summations over the repeated indices q, p, l, k from 1 to 3. Notice that deformation gradient F is constant per triangle (to be more precise we should have used F^e to refer to a deformation gradient of an element e). Moreover, notice that $\frac{\partial F_{qp}}{\partial x_i^n} = \frac{\partial \nabla \mathbb{N}_p^n x_q^n}{\partial x_i^n} = \delta_{qi} \nabla \mathbb{N}_p^n$ (where $\nabla \mathbb{N}_p^n$ is a gradient of a shape function of a node n inside an element e), thus the equations (2.6) and (2.7) can be rewritten as:

$$\nabla W_i^n = \sum_{e \in \mathcal{N}(n)} \frac{\partial w^e}{\partial F_{ip}} \nabla \mathbb{N}_p^n\ \mathrm{Vol}(e), \tag{2.8}$$

and

$$\nabla^2 W_{ij}^{nm} = \sum_{\substack{e\in\mathcal{N}(n)\\ e\in\mathcal{N}(m)}} \frac{\partial^2 w^e}{\partial F_{ip}\partial F_{jk}} \nabla\mathbb{N}_p^n \nabla\mathbb{N}_k^m \ \mathrm{Vol}(e). \tag{2.9}$$

For efficiency reasons, the computations of $\frac{\partial w^e}{\partial F}$ and $\frac{\partial^2 w^e}{\partial F \partial F}$ can be done per tet, and $\nabla\mathbb{N}^n$ can be precomputed for each vertex n inside each tet e since they only depend on the undeformed positions.

Finally, the gradient and the Hessian of the energy defined in the Equation (2.4) are computed as follows:

Gradient:

$$\frac{\partial w^e}{\partial F_{ij}} = \frac{\mu}{\sqrt[3]{J^2}}\left[2F_{ij} - \frac{2}{3}F_{ji}^{-1}\mathrm{tr}(C)\right] + \kappa J F_{ji}^{-1}(J-1). \tag{2.10}$$

Hessian:

$$\begin{aligned}\frac{\partial^2 w^e}{\partial F_{kl}\partial F_{ij}} =& \frac{2\mu}{\sqrt[3]{J^2}}\delta_{ik}\delta_{jl} + \\ & \frac{2\mu}{3\sqrt[3]{J^2}}\left[\mathrm{tr}(C)(\frac{2}{3}F_{ji}^{-1}F_{lk}^{-1} + F_{li}^{-1}F_{jk}^{-1}) - 2(F_{kl}F_{ji}^{-1} + F_{ij}F_{lk}^{-1})\right] + \\ & \kappa J\left[(2J-1)F_{ji}^{-1}F_{lk}^{-1} - (J-1)F_{li}^{-1}F_{jk}^{-1}\right].\end{aligned} \tag{2.11}$$

Other expressions of energy can be similarly differentiated using tensor differentiation rules (see [59]).

2.2.2 Linear Elasticity

While linear elasticity cannot capture all the intricacies of the behavior of elastic materials, it is widely used in practice for its computational simplicity and efficiency. In order to obtain linear equations of elasticity, the strain tensor needs to be linearized, and the potential elastic energy is assumed to be quadratic. The linearized strain tensor is not invariant under rigid body rotations, leading to large artificial changes in volume due to a rotation of the object. Such behavior is not suitable for most of the graphics applications, and a co-rotational fix (see Section 2.2.3) is usually added to reduce these artifacts. In addition to computational efficiency, an attractive feature of linear elasticity for us is the linearity of the material law that results in constant elastic coefficients. We will use this property for our coarsening approach that allows to precompute elastic coefficients of the coarse mesh to closely mimic the behavior of the fine heterogeneous mesh (for a detailed

discussion see Chapter 4).

Summary of Linear Elasticity Linear elasticity has received extensive attention, and many detailed explanations of its foundations can be found throughout the literature. We recap the main notions nonetheless, by way of introduction to our notation and to the delicate issues of tensor symmetries that will arise in our coarsening approach.

Strain Tensor. Given a displacement field $\mathbf{u} = x - X$ defined over the undeformed configuration of an object, the symmetric part of the deformation gradient $\boldsymbol{\epsilon}$, computed as

$$\boldsymbol{\epsilon} = \frac{1}{2}(\nabla\mathbf{u} + \nabla\mathbf{u}^T) \quad (i.e., \;\; \epsilon_{ij} = \frac{1}{2}(u_{i,j} + u_{j,i}) \;), \tag{2.12}$$

represents the strain undergone by the object. Indeed, the remaining part is antisymmetric, and thus only represents a pure (infinitesimal) rotation–which does not induce a deformation. Note that by definition, we have $\epsilon_{ij} = \epsilon_{ji}$. This strain tensor is a linearized version of the more general Cauchy strain tensor considered in non-linear elasticity. Using notation from Section 2.2.1, we can also write linear strain as $\boldsymbol{\epsilon} = \frac{1}{2}(F + F^T) - I$.

Potential Energy. Hookean materials are assumed to have, just like a simple spring, a potential function $W(\mathbf{u})$ that is quadratic in the strain tensor:

$$W(\mathbf{u}) = \frac{1}{2}\,\boldsymbol{\epsilon} : \mathbf{C} : \boldsymbol{\epsilon} = \frac{1}{2}\,\epsilon_{ij}\,\mathrm{C}_{ijkl}\,\epsilon_{kl}, \tag{2.13}$$

where $\mathbf{C} = \{\mathrm{C}_{ijkl}\}$ is a rank-4 tensor called the *elasticity tensor* (sometimes referred to as the compliance tensor, or tensor of elastic compliances).

Symmetries of Elasticity Tensor. While being a rank-4 tensor (thus with 81 components in 3D), the elastic tensor $\mathbf{C}$ possesses several symmetries. The symmetry of the stress tensor implies that $\mathbf{C}$ is symmetric in its first pair of indices ($\mathrm{C}_{ijkl} = \mathrm{C}_{jikl}$), while the symmetry of the strain tensor results in $\mathbf{C}$ being symmetric in its second pair of indices ($\mathrm{C}_{ijkl} = \mathrm{C}_{ijlk}$), these properties often called minor symmetries. Finally, since the strain energy is a quadratic form, we also have symmetry under an interchange of the first and second pairs of slots ($\mathrm{C}_{ijkl} = \mathrm{C}_{klij}$), which often referred to as major symmetry [24]. This leaves only 21 independent components in 3D.

However, if the material is further assumed to be *isotropic*, as is pervasive in graphics, then $\mathbf{C}$ only possesses 2 independent components, usually expressed as Lamé coefficients λ and μ, or as Young modulus E and Poisson ratio ν. Using matrix notation described in Section 2.1.2, such isotropic $\mathbf{C}$ can be written as:

$$\mathbf{C} = \begin{pmatrix} \lambda+2\mu & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda+2\mu & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & \lambda+2\mu & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu \end{pmatrix}.$$

Note that this isotropic assumption amounts to assuming that deformations within the medium have no preferred direction.

Implementation Details Linear elastic forces can be computed as a matrix-vector multiply $\mathbf{Ku}$, where $\mathbf{K}$ is a stiffness matrix and $\mathbf{u}$ is a displacement vector. A stiffness matrix $\mathbf{K}$ for linear finite element discretization is computed as:

$$\mathbf{K}_{ij}^{nm} = \sum_{\substack{e \in \mathcal{N}(n) \\ e \in \mathcal{N}(m)}} C_{ipjk}^{e} \nabla \mathbb{N}_{p}^{n} \nabla \mathbb{N}_{k}^{m} \ \mathrm{Vol}(e). \tag{2.14}$$

For the purpose of efficiency, local stiffness matrices $\mathbf{K}^e$ are traditionally computed for each element e and then assembled into the global stiffness matrix.

The global stiffness matrix $\mathbf{K}$ is positive semidefinite, having a null space that corresponds to zero energy modes of the system (usually translation and infinitesimal rotation modes). For static problems

$$\mathrm{div}(\mathbf{C} : \boldsymbol{\epsilon}) = K\mathbf{u} = \mathbf{b}, \tag{2.15}$$

when no constraints are specified (such as in the problem described in Section 4.3.5), the force vector $\mathbf{b}$ need to be modified to remove translation and torque. Translation free force vector $\hat{\mathbf{b}}$ is computed as $\hat{\mathbf{b}}_k = \mathbf{b}_k - \frac{\sum_i^M \mathbf{b}_i}{M}$ for the vertex k, and the torque free vector $\tilde{\mathbf{b}}$ as $\tilde{\mathbf{b}}_k = \hat{\mathbf{b}}_k - m_k \mathbf{r}_k \times (\mathbb{I}^{-1} \sum_i^M \mathbf{r}_i \times \hat{\mathbf{b}}_i)$, where $\mathbf{r}_k$ is the vector from the center of the body to the vertex k, and $\mathbb{I}$ is the inertia tensor of the body:

$$\mathbb{I} = \sum_i^M m_i \begin{bmatrix} y_i^2 + z_i^2 & -x_i y_i & -x_i z_i \\ -x_i y_i & x_i^2 + z_i^2 & -x_i z_i \\ -x_i z_i & -x_i z_i & x_i^2 + y_i^2 \end{bmatrix}. \tag{2.16}$$

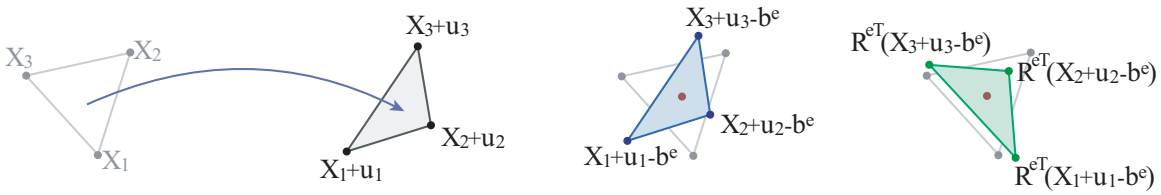


Figure 2.3: Visualization of the basic idea of corotational approaches: a deformed element (dark grey) is translated (blue) and rotated (green) to "best" align with an undeformed configuration (light green).

2.2.3 Co-Rotational Methods

Corotational methods all share the same intent: they try to render the linear strain tensor (Eq. (2.12)) more accurate for large deformation by removing as much of the *current local rotation* as possible. For each tetrahedral element a corotated reference frame is chosen, "with respect to which the relative displacements [...] due to the current deformations are minimum in some global sense" [19]. Instead of expressing the position of the deformed body always in the same coordinate frame (by adding the current displacements $\mathbf{u}$ to the rest state $\mathbf{X}$), a corotational approach defines a matrix $\mathbf{R}^e$ per element e (expressing a rotation around the current barycenter $\mathbf{b}^e$ of the element) such that the current deformed position $\mathbf{x}^e$ of the element's nodes satisfies:

$$\mathbf{R}^e\mathbf{x}^e = \mathbf{X}^e + \mathbf{u}^e - \mathbf{b}^e,$$

where $\mathbf{R}^e$ removes the purely rotational part of the current deformation (see Figure 2.3). This amounts to defining a "corotated displacement" $\widehat{\mathbf{u}}^e$ per element as:

$$\widehat{\mathbf{u}}^e = \mathbf{R}^{eT}(\mathbf{X}^e + \mathbf{u}^e - \mathbf{b}^e) - (\mathbf{X}^e - \mathbf{b}_0^e),$$

where $\mathbf{b}_0$ denotes the barycenter of the element in the rest configuration. The potential $W = \frac{1}{2}\mathbf{u}^T\mathbf{K}\mathbf{u}$ is then replaced by $\frac{1}{2}\widehat{\mathbf{u}}^T\mathbf{K}\widehat{\mathbf{u}}$, thus turning into a sum over all elements now function of both the displacement field $\mathbf{u}$ and the rotation field $\mathbf{R} = \{\mathbf{R}^1, \ldots, \mathbf{R}^{|T|}\}$. The force field in the elastic body becomes:

$$\nabla W = \frac{1}{2}\sum_e \Big[\mathbf{R}^e\mathbf{K}^e\mathbf{R}^{eT}(\mathbf{X}^e + \mathbf{u}^e - \mathbf{b}^e) - 2\mathbf{R}^e\mathbf{K}^e(\mathbf{X}^e - \mathbf{b}_0^e)\Big].$$

Notice that the material velocity $\dot{\mathbf{x}}$ is still equal to $\dot{\mathbf{u}}$: the corotational treatment presented above *only* affects the potential energy computations (and thus, the internal forces).

While corotational methods define various procedures to derive the rotation matrix $\mathbf{R}^e$ of each element [50, 31, 53], we follow the treatment of [19]. That is, we pick the corotational frame of each element by minimizing $|\widehat{\mathbf{u}}^e|^2$: our tests show improved behavior compared to QR or polar decomposition, as reported in [25]. We followed the implementation of this latter reference to solve the small non-linear system for each tetrahedral element.

Treatment of point constraints. We found that in practice more stable results are achieved when the rotation of the elements that have multiple nodes constrained is treated in a special manner. In particular, for a tetrahedron that has four or three nodes constrained, we assume the rotation matrix to be identity. If a tet has two vertices constrained, we only consider rotations around the vector defined by the two constrained vertices. If only a single vertex of a tet is constrained, the rotation is found in the traditional way.

The details of time integration for co-rotational approaches are discussed in Section 3.4.2, where an efficient and stable time integrator that does not suffer from numerical dissipation is introduced.

2.3 Damping Forces

The energy and momenta of physical objects are not preserved because of internal damping, drag, friction, and other external forces. People often use this claim to justify the use of implicit methods that numerically damp energy. However, artificial numerical energy loss does not correctly represent energy decrease due to the real physical external forces and the internal damping (see Figure 3.5). Moreover, numerical damping strongly depends on the timestep of the simulation and the resolution of the mesh. By using structure preserving time integrators we can first duplicate the physics of the conservation laws, and then have the option of adding damping, friction, air drag, and other external forces. Thus, there is a need for *resolution independent models* that capture physical behavior of these effects as well as possible. For example, internal damping forces that decrease the energy of the system without changing linear or angular momenta can be added. The idea is to use the strain energy function to "measure" and damp the amount of deformation happening in one step, tantamount to a generalized Rayleigh damping. Such damping model removes high frequency vibrations from the elastic body. Strongly damped elastic objects will quickly reach equilibrium positions or, if no constraints or other external forces are present, will move though space as a rigid body with constant kinetic energy (Figure 2.4).

Figure 2.4: An elastic damped bar is flowing though space rotating and bending. As a simulation progresses the bar behaves as a rigid body but still maintains its angular momenta.

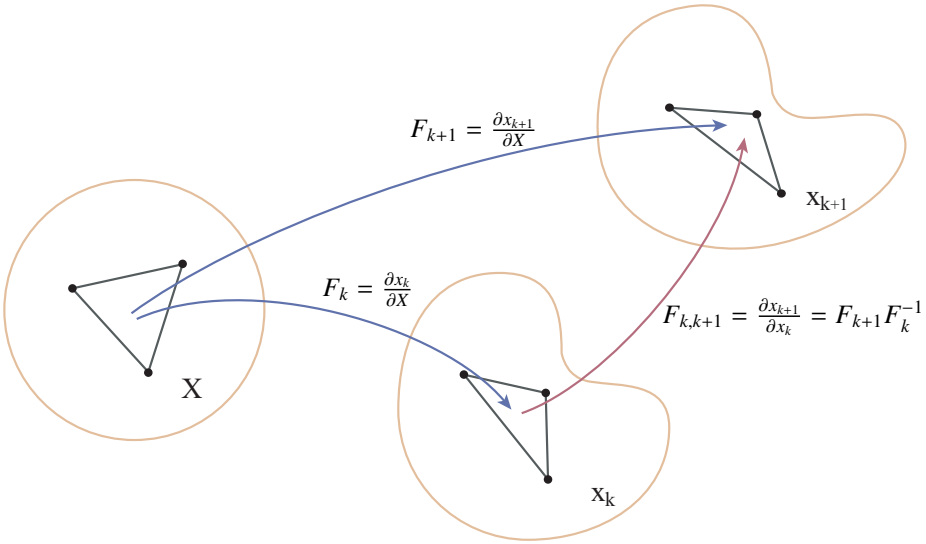


Figure 2.5: Schematic visualization of deformation gradients F_k, F_{k+1}, and $F_{k,k+1}$.

In order to compute damping, the same strain energy function W (as discussed in Section 2.2.1) of the Cauchy tensor $C = F^T F$ can be used. Deformation gradient F is itself a function of the initial configuration X and the deformed configuration x: $W = W(F(X, x))$. Consider the deformed configurations of the body at two consequent times t_k and t_{k+1}, *internal elastic forces* are then computed as $-\nabla W(F(X, x_k)) = \frac{\partial W(F(X,x_k))}{\partial x_k}$, $-\nabla W(F(X, x_{k+1})$, or $-\nabla W(F(X, \frac{x_k+x_{k+1}}{2}))$ if the forces are discretized at the midpoint. The discrete *damping forces* between times t_k and t_{k+1} are computed as $-\nabla W(F(x_k, x_{k+1}) = \frac{\partial W(F(x_k,x_{k+1}))}{\partial x_{k+1}}$. This amounts to computing damping forces as elastic forces based on x_k being undeformed configuration and x_{k+1} being deformed configuration. Using notation from the Figure 2.5 one can also write this expression of damping forces as $-\nabla W(F_{k+1}F_k^{-1})$.

Example. This example shows how to compute introduced damping forces for 1D linear spring. Consider a spring connecting nodes a and b with the original length $l_0 = |X^a - X^b|$ and the deformed

length $l = |x^a - x^b|$. The potential elastic energy of the spring is simply $W = \frac{1}{2}\lambda(F-1)^2 l_0$, with the deformation gradient $F = \frac{l}{l_0}$, and the gradient of the energy $\nabla W = \lambda(\frac{l}{l_0} - 1)\frac{\partial l}{\partial x}$. Thus the elastic force acting on the node a at the time t_k is $-\lambda(\frac{l_k}{l_0} - 1)\frac{x_k^a - x_k^b}{|x_k^a - x_k^b|}$ and the damping force acting on the same node is $-\lambda k_d(\frac{l_k}{l_{k-1}} - 1)\frac{x_k^a - x_k^b}{|x_k^a - x_k^b|}$, where l_k is the length of the spring at the time t_k, l_{k-1} is the length of the spring at the time t_{k-1}, and k_d is a damping coefficient.

Explicit and Implicit Damping Forces. Let x_{k-1}, x_k, and x_{k+1} denote deformed configurations of an elastic body at corresponding times t_{k-1}, t_{k-1}, t_{k+1}. Damping forces at time t_k can be computed to minimize deformation between x_{k-1} and x_k configurations:

$$F_{\text{dampExp}}(x_{k-1}, x_k) = -k_D \nabla W(F(x_{k-1}, x_k))$$

resulting in an explicit forces. Minimizing deformation between x_k and x_{k+1} configurations will provide an implicit damping force:

$$F_{\text{dampImp}}(x_k, x_{k+1}) = k_D \nabla W(F(x_{k+1}, x_k)).$$

Because in the both cases forces are computed based on deformed configuration x_k they do not create torque: $F^d_{\text{damp}} \times x_k = 0$. Numerical experiments demonstrating the quality of this damping model when used with variational time integrators (in particular, the fact that it does not reduce either linear or angular momentum) are described in Figure 3.4. For more information on how to include these damping forces into variational time integrator see Section 3.2.9.

Limitations. Since the damping model presented above is non-linear we use it for the simulations that use non-linear elasticity models. In the cases when linear or co-rotational elasticity models are used we revert to traditional damping forces $-\mathbf{K}\dot{\mathbf{x}}$ (where $\mathbf{K}$ is a stiffness matrix and $\dot{\mathbf{x}}$ is the velocity). A more extensive study of linearized damping models is desirable and is left for the future work.

Chapter 3

Structure Preserving Time Integrators

3.1 Introduction

Mathematical models of the evolution in time of dynamical systems (whether in biology, economics, or computer animation) generally involve systems of differential equations. *Solving* a physical system means figuring out how to move the system forward in time from a set of initial conditions, allowing the computation of, for instance, the trajectory of a ball (*i.e.*, its position as a function of time) thrown up in the air. Although this example can easily be solved analytically, direct solutions of the differential equations governing a system are generally hard or impossible—we need to resort to numerical techniques to find a discrete temporal description of a motion. Consequently, there has been a significant amount of research in applied mathematics on how to deal with some of the most useful systems of equations, leading to a plethora of numerical schemes with various properties, orders of accuracy, and levels of complexity of implementation [60]. In computer animation, these integrators are crucial computational tools at the core of most physics-based animation techniques, and classical methods (such as fourth-order Runge-Kutta, implicit Euler, and more recently the Newmark scheme) have been methods of choice in practice [57]. In this chapter, we present a general-purpose numerical scheme for time integration of Lagrangian dynamical systems - an important computational tool at the core of most physics-based animation techniques. Several features make this particular time integrator highly desirable for computer animation: it numerically preserves important invariants, such as linear and angular momenta; the symplectic nature of the integrator also guarantees a correct energy behavior, even when dissipation and external forces are added; holonomic constraints can also be enforced quite simply; finally, our simple methodology

allows for the design of high-order accurate schemes if needed. Two key properties set the method apart from earlier approaches. First, the nonlinear equations that must be solved during an update step are replaced by a minimization of a novel functional. Second, the formulation introduces additional variables that provide key flexibility in the implementation of the method. These properties are achieved using a discrete form of a general variational principle called the Pontryagin-Hamilton principle, expressing time integration in a discrete geometric manner analog in spirit to geometric modeling techniques to design smooth curves or surfaces.

3.2 Fully Variational Integrators

3.2.1 Background

Dynamics as a Variational Problem Considering mechanics from a variational point of view goes back to Euler, Lagrange and Hamilton. The form of the variational principle most important for continuous mechanics is due to Hamilton, and is often called *Hamilton's principle* or the *least action principle* (as we will see later, this is a bit of a misnomer: "stationary action principle" would be more correct): it states that a dynamical system always finds an optimal course from one position to another (a more formal definition will be presented in Section 3.2.2). One consequence is that we can recast the traditional way of thinking about an object accelerating in response to applied forces, into a geometric viewpoint. There the path followed by the object has *optimal geometric properties*—analogous to the notion of geodesics on curved surfaces. This point of view is equivalent to Newton's laws in the context of classical mechanics, but is broad enough to encompass areas ranging to E&M and quantum mechanics.

Geometric Integrators Geometric integrators are a class of numerical time-stepping methods that exploit the geometric structure of mechanical systems [27]. Of particular interest within this class, *variational integrators* [47] discretize the variational formulation of mechanics we mentioned above, providing a solution for most ordinary and partial differential equations that arise in mechanics. While the idea of discretizing variational formulations of mechanics is standard for elliptic problems using Galerkin Finite Element methods for instance, only recently did it get used to derive variational time-stepping algorithms for mechanical systems. This approach allows the construction of integrators with any order of accuracy [70, 44], that can handle constraints [40] as well as external forcing. The integrators that are derived from variational principles are symplectic, *i.e.*, they preserve area in phase space, where the phase space consists of discrete position and momenta vari-

ables. This property discretely mimics the symplectic nature of continuous Hamiltonian systems. Variational integrators have been shown remarkably powerful for simulations of physical phenomena when compared to traditional numerical time stepping methods [36]. This discrete geometric framework is thus versatile, powerful, and general. For example, the well-known symplectic variant of the Newmark scheme (velocity Verlet) can best be elucidated by writing it as a variational integrator [70]. Of particular interest in computer animation, the simplest variational integrator can be implemented by taking two consecutive positions $q_0 = q(t_0)$ and $q_1 = q(t_0 + dt)$ of the system to compute the next position q_2. Repeating this process calculates an entire discrete (in time) trajectory.

Accurate vs. Qualitative Integrators While it is unavoidable to make approximations in numerical algorithms (*i.e.*, to differ from the continuous equivalent), the matter becomes whether the numerics can provide satisfactory results. *Qualitative* reproduction of phenomena is often favored in computer animation over absolute *accuracy*. We argue in the following that one does not have to ask for *either* plausibility *or* accuracy. In fact, we seek a simple method robust enough to provide good, qualitative simulations that can also be easily rendered arbitrarily accurate. The symplectic character of variational integrators provides good foundations for the design of robust algorithms: this property guarantees *good statistical predictability* through accurate preservation of the geometric properties of the exact flow of the differential equations. As a consequence, subjectivity offers long-time energy preservation—a crucial property since large energy increase is often synonymous with numerical divergence while a large decrease dampens the motion, decreasing visual plausibility. A well-known example where this property is crucial is the simple pendulum (particularly relevant in robotic applications for articulated figures), for which other (even high-order) integrators can fail in keeping the amplitude of the oscillations (see Figure 3.2). With this in mind, we will pursue numerical schemes which offer qualitatively-correct *as well as* arbitrarily accurate solutions.

3.2.2 Overview of Continuous Lagrangian Dynamics

Consider a finite-dimensional dynamical system parameterized by the state variable q (*i.e.*, the vector containing all degrees of freedom). The Lagrangian function of the system is given as a function of q and $\dot{q}$. In classical mechanics, this Lagrangian function L is defined as the kinetic energy K

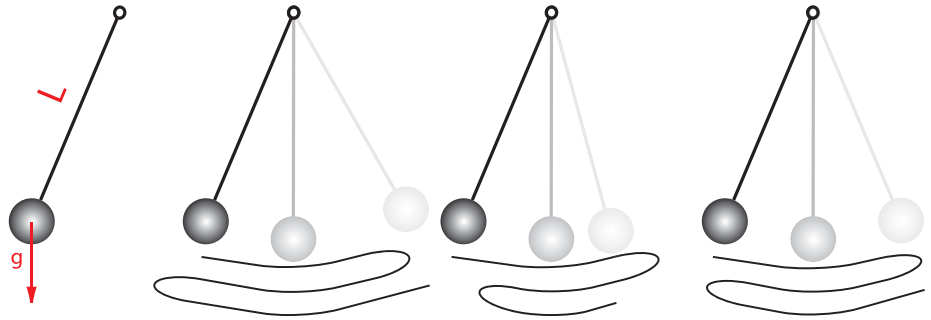


Figure 3.1: Advantages of symplecticity: for the equation of motion of a pendulum of length L in a gravitation field g (left), the usual explicit Euler integrator amplifies oscillations, the implicit one dampens the motion, while a *symplectic* integrator perfectly captures the periodic nature of the pendulum (see [66] for details).

minus the potential energy W of the system:

$$L(q, \dot{q}) = K(\dot{q}) - W(q).$$

The *action functional* is the integral of L along a path $q(t)$, over time $t \in [0, T]$. *Hamilton's principle* now states that *the correct path of motion of a dynamical system is such that its action has a stationary value*, *i.e.*, the integral along the correct path has the same value to within first-order infinitesimal perturbations. As an "integral principle" this description encompasses the entire motion of a system between two fixed times.

Computing variations of the action induced by variations δq of the path $q(t)$ results in:

$$\begin{aligned}
\delta S(q) &= \delta \int_0^T L(q(t), \dot{q}(t))\, dt \\
&= \int_0^T \left[\frac{\partial L}{\partial q} \cdot \delta q + \frac{\partial L}{\partial \dot{q}} \cdot \delta \dot{q} \right] dt \\
&= \int_0^T \left[\frac{\partial L}{\partial q} - \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{q}} \right) \right] \delta q \, dt + \left[\frac{\partial L}{\partial \dot{q}} \cdot \delta q \right]_0^T,
\end{aligned}$$

where integration by parts is used in the last equality. When the endpoints of $q(t)$ are held fixed with respect to all variations $\delta q(t)$ (*i.e.*, $\delta q(0) = \delta q(T) = 0$), the rightmost term in the above equation vanishes. Therefore, the condition of stationary action for arbitrary variations δq with fixed endpoints stated in Hamilton's principle directly indicates that the remaining integrand in the previous

equation must be zero for all time t, yielding the well-known Euler-Lagrange equations:

$$\frac{\partial L}{\partial q} - \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) = 0. \tag{3.1}$$

Standard Example. Let $K = \frac{1}{2}\dot{q}^T M\dot{q}$, where M is the mass matrix. Then (3.1) simply states Newton's law: $M\ddot{q} = -\nabla W(q)$, *i.e.*, mass times acceleration equals force. Here, the force is conservative (no damping occurs) since it is derived from a potential function.

Forced Systems. To account for non-conservative forces F (typically, dissipation), the least action principle is modified to become:

$$\delta \int_0^T L(q(t), \dot{q}(t))\, dt + \int_0^T F(q(t), \dot{q}(t)) \cdot \delta q\, dt = 0,$$

which is known as the *Lagrange-d'Alembert principle*.

Lagrangian vs. Hamiltonian Mechanics Lagrangian mechanics is not the only existing formalism available. In fact, Hamiltonian mechanics provides an alternative, closely related formulation. For later use we point out that Hamiltonian mechanics is described in *phase space*, *i.e.*, the current state of a dynamical system is given as a pair (q, p), where q is the state variable, while p is the momentum, defined as $p = \partial L/\partial \dot{q}$.

3.2.3 Discrete Lagrangian Mechanics

The least action principle stated above can be used as a guiding principle to derive discrete integrators. In fact, West [70] proposed a direct discretization of the integral of the Lagrangian to construct a proper and simple *discrete* action function. The main idea is to discretize the least action principle directly rather than discretizing Equation (3.1).

Time Discretization A motion $q(t)$ of the mesh, for $t \in [0, T]$, is replaced by a discrete sequence of poses q_k, with $k = 0, \ldots, N \in \mathbb{N}$, at discrete times: $\{t_0 = 0, \ldots, t_{k-1}, t_k, t_{k+1}, \ldots, t_N = T\}$; h_{k+1} is the timestep between time t_k and t_{k+1}. For the cases when constant size timesteps are used h_k, h_{k+1}, h_{k+2}, etc. are often abbreviated with h.

Discrete Euler-Lagrange Equations The integral $\int_{t_k}^{t_{k+1}} L\, dt$ is approximated on each time interval $[t_k, t_{k+1}]$ by a *discrete Lagrangian*[1] $L_d(q_k, q_{k+1}, h_{k+1})$, and can be evaluated using an appropriate quadrature rule. The discrete version of the Lagrangian formulation requires one to find paths $\{q_k\}_0^N$ such that for all discrete variations $\{\delta q_k\}_0^N$, one has:

$$\delta S_d = \delta \sum_{k=0}^{N} L_d(q_k, q_{k+1}, h_{k+1}) = 0.$$

This yields the discrete Euler-Lagrange (DEL) equations :

$$D_1 L_d(q_{k+1}, q_{k+2}, h_{k+2}) + D_2 L_d(q_k, q_{k+1}, h_{k+1}) = 0, \tag{3.2}$$

where D_1 and D_2 denote differentiation with respect to the first and second arguments. Notice that this condition only involves three consecutive positions. Therefore, for two given successive positions q_k and q_{k+1}, Eq. (3.2) *defines* q_{k+2}.

Quadrature-Based Discrete Lagrangian The accuracy of the resulting integrator depends on the quadrature rule used to approximate the discrete Lagrangian. Second-order explicit time integration can be derived by using the trapezoidal rule to evaluate the integral of potential energy and with $\dot{q}$ replaced by $(q_{k+1} - q_k)/h_{k+1}$:

$$L_d(q_k, q_{k+1}, h_{k+1}) = \frac{h_{k+1}}{2} \frac{q_{k+1} - q_k}{h_{k+1}}^T M \frac{q_{k+1} - q_k}{h_{k+1}} - h_{k+1} \frac{W(q_k) + W(q_{k+1})}{2} \tag{3.3}$$

For an implicit integrator, a simple one-point quadrature can be used:

$$L_d(q_k, q_{k+1}, h_{k+1}) = \frac{h_{k+1}}{2} \frac{q_{k+1} - q_k}{h_{k+1}}^T M \frac{q_{k+1} - q_k}{h_{k+1}} - h_{k+1} W((1-\alpha)q_k + \alpha q_{k+1}) \tag{3.4}$$

where $\alpha \in [0, 1]$. For α equal to $1/2$ the quadrature is second order accurate. Note that the equations for $\alpha = 0$ and 1 are the same as for the trapezoidal rule if constant size timesteps are used. Any other value of α leads to linear accuracy. The higher-order quadratures of the Lagrangian lead naturally to higher-order integration schemes. For more details on the DEL equations, we refer the reader to an introductory text on discrete mechanics [66].

[1] This term could also be called an action, as it is a time integral of a Lagrangian; however, just like the term "discrete curvature" in CG refers to a small local integral of a continuous curvature, we prefer this naming convention.

3.2.4 Continuous Hamilton-Pontryagin Principle

The *Hamilton-Pontryagin principle* (deeply rooted in the control of dynamical systems) states that the equations of mechanics are given by the critical points of the *Hamilton-Pontryagin action*:

$$\delta \int_0^T \Big[p(\dot{q} - v) + L(q, v) \Big] dt = 0,$$

where the configuration variable q, the velocity v and the momentum p are all viewed as *independent* variables. (See [72] for an exposition and history.) That is, $q(t)$, $v(t)$, $p(t)$ are varied independently (with end-point conditions on $q(t)$). Notice the similarity with Hamilton's principle: p can be interpreted as a *Lagrange multiplier* to enforce the equality between $\dot{q}$ and v. The Hamilton-Pontryagin principle yields equations equivalent to the Euler-Lagrange equations (3.1), since, for the respective variations $\delta p(t)$, $\delta q(t)$ and $\delta v(t)$ over the three independent variables, we get:

$$v = \dot{q}, \qquad \frac{dp}{dt} = \frac{\partial L(q, v)}{\partial q}, \qquad p = \frac{\partial L(q, v)}{\partial v}. \tag{3.5}$$

We stress the important feature this different variational approach brings and that points to the generality of this principle: with the addition of the new variables, these equations can be understood from a Lagrangian *and* Hamiltonian point of view since the formulation involves both phase-space variables q and p within the action. A more thorough discussion on this connection to Hamiltonian mechanics can be found in [42].

3.2.5 Discrete Hamilton-Pontryagin Principle

Discrete Variables Same as in the usual formulation, a motion $q(t)$, for $t \in [0, T]$, is replaced by a discrete sequence q_k, with $k = 0, \ldots, N \in \mathbb{N}$, at discrete times: $\{t_0 = 0, \ldots, t_{k-1}, t_k, t_{k+1}, \ldots, t_N = T\}$. We similarly discretize $v(t)$ and $p(t)$ by the sets $\{v_k\}_{k=0}^N$ and $\{p_k\}_{k=0}^N$. Velocities v_{k+1} and momenta p_{k+1} are viewed as approximations *within* the interval $[t_k, t_{k+1}]$, *i.e.*, staggered with respect to the positions q_k.

Quadrature-based Discrete Action We will remain agnostic as to the Lagrangian used in this section: the cases of non-linear and linear elasticity are addressed using potential elastic energy given in Chapter 2, but our explanations are valid for any continuous Lagrangian $L(q, \dot{q})$. The discrete Lagrangian $L^d(q_k, v_{k+1}, h_{k+1})$ is defined in terms of the new variables and is equal to traditional

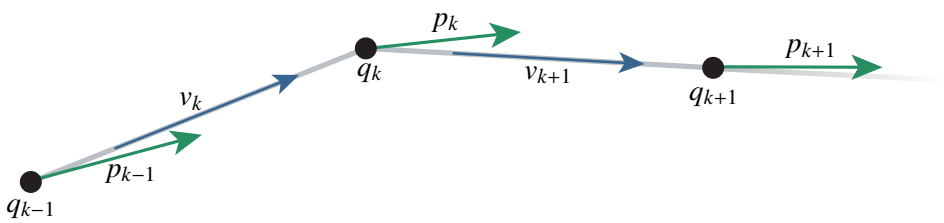


Figure 3.2: Discrete position (in black), momentum (in green), and velocity (in blue) variables.

$L_d(q_k, q_{k+1}, h_{k+1})$ defined in Section 3.2.3, assuming that $v_{k+1} = \frac{q_{k+1}-q_k}{h_{k+1}}$. For example, for one point quadrature:

$$L^d(q_k, v_{k+1}, h_{k+1}) = \frac{h_{k+1}}{2} v_{k+1}^T M v_{k+1} - h_{k+1} W(q_k + \alpha h_{k+1} v_{k+1}) = L_d(q_k, q_{k+1}, h_{k+1}) \tag{3.6}$$

This quadrature has quadratic accuracy for $\alpha = 1/2$ and linear accuracy for all other $\alpha \in [0, 1]$ (notice that for $\alpha = 0$ or 1 the update rules are explicit). Explicit second order accurate scheme can be achieved with trapezoidal discretization:

$$L^d(q_k, v_{k+1}) = \frac{h_{k+1}}{2} v_{k+1}^T M v_{k+1} - \frac{h_{k+1}}{2}(W(q_k) + W(q_k + h_{k+1} v_{k+1})) \tag{3.7}$$

More accurate quadrature rules (be they of Newton-Cotes or Gaussian type [60], for example) can be employed to increase the approximation order if necessary.

Discrete Variational Equations Once a discrete Lagrangian is given, a ***discrete Hamilton-Pontryagin principle*** can be expressed through:

$$\delta \sum_{k=0}^{N} \left[p_{k+1}\left(\frac{q_{k+1} - q_k}{h_{k+1}} - v_{k+1}\right) h_{k+1} + L^d(q_k, v_{k+1}) \right] = 0. \tag{3.8}$$

The discrete Hamilton-Pontryagin principle yields, upon taking discrete variations with respect to each state variable with fixed endpoints:

$$\delta p: \qquad q_{k+1} - q_k = h_{k+1} v_{k+1} \tag{3.9}$$

$$\delta q: \qquad p_{k+1} - p_k = D_1 L^d(q_k, v_{k+1}) \tag{3.10}$$

$$\delta v: \qquad h_{k+1}\ p_{k+1} = D_2 L^d(q_k, v_{k+1}) \tag{3.11}$$

where D_1 and D_2 denote the differentiation with respect to the first (q_k) and second (v_{k+1}) arguments of L^d and $h_{k+1} = t_{k+1} - t_k$.

Natural Update Procedure Given a point in the discrete Pontryagin-state space (q_k, v_k, p_k), the above equations are to be solved for $(q_{k+1}, v_{k+1}, p_{k+1})$ in the following way:

- *Plug* (3.10) *into* (3.11) so that p_{k+1} is replaced by a function of p_k and $D_1 L^d(q_k, v_{k+1})$.
- The resulting equation:

$$D_2 L^d(q_k, v_{k+1}) - h_{k+1} p_k - h_{k+1} D_1 L^d(q_k, v_{k+1}) = 0 \tag{3.12}$$

  can now be solved for v_{k+1} with any non-linear solver, or used to compute v_{k+1} directly in the explicit case.

- q_{k+1} and p_{k+1} are found with (3.9) and (3.11) respectively.

Example. For a system with a potential energy W, a mass matrix M, and a Lagrangian discretized as $L^d(q_k, v_{k+1}) = \frac{1}{2} v_{k+1}^T M v_{k+1} - W(q_k + \frac{1}{2} h v_{k+1})$ the update rules are as following:

$$\begin{aligned} &M v_{k+1} + \frac{h}{2} \nabla W(q_k + \frac{1}{2} h v_{k+1}) - p_k = 0 \\ &q_{k+1} = q_k + h v_{k+1} \\ &p_{k+1} = M v_{k+1} - \frac{h}{2} \nabla W(q_k + \frac{1}{2} h v_{k+1}). \end{aligned}$$

Equivalence with DEL Equations One can readily verify (using the chain rule) that the integration procedure (3.9-3.11) obtained from the discrete Hamilton-Pontryagin principle is mathematically *equivalent* to the variational integrator described in [70]. Thus, both schemes share the same numerical benefits such as the conservation of discrete momenta and energy, as we will discuss further in Section 3.2.12—but with now more variables to control the motion if needed.

3.2.6 Discrete Pontryagin-d'Alembert Principle

Pontryagin-d'Alembert principle is a generalization of Hamilton-Pontryagin principle for non-conservative mechanical systems, *i.e.*, for systems with exterior forces. The continuous Pontryagin-d'Alembert

principle is given by:

$$\delta \int_0^T \left[L(q,v) + p(\dot{q} - v) \right] \, dt + \int_0^T F_v(q,v) \cdot \delta q \, dt = 0$$

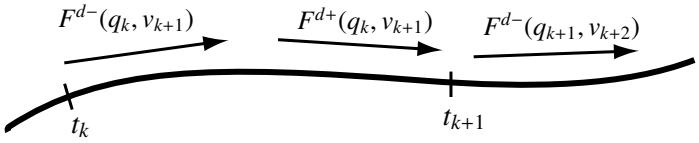
where $F(q,v)$ is an arbitrary external non-conservative force.

The ***discrete Pontryagin-d'Alembert principle*** can thus be defined as:

$$\begin{aligned} &\delta\Big(\sum_{k=0}^{N} p_{k+1}(q_{k+1} - q_k - h_k v_{k+1}) + L^d(q_k, v_{k+1}) \Big) + \\ &\sum_{k=0}^{N} \big(F^{d-}(q_k, v_{k+1}) \cdot \delta q_k + F^{d+}(q_k, v_{k+1}) \cdot \delta q_{k+1} \big) = 0, \end{aligned}$$

where F^{d-} and F^{d+} approximate the total forcing over a time step (see schematic figure below) through:

$$\int_{t_k}^{t_{k+1}} F(q,\dot{q}) \delta q \, dt \simeq F^{d-}(q_k, v_{k+1}) \delta q_k + F^{d+}(q_k, v_{k+1}) \delta q_{k+1}.$$



This yields, upon taking discrete variations, the following *forced discrete variational equations*:

$$\begin{aligned} &q_{k+1} - q_k = h_k v_{k+1} \\ &p_{k+1} - p_k = D_1 L^d(q_k, v_{k+1}) + F^{d-}(q_k, v_{k+1}) + F^{d+}(q_{k-1}, v_k) \\ &h_k \; p_{k+1} = D_2 L^d(q_k, v_{k+1}). \end{aligned} \tag{3.13}$$

3.2.7 Integration With Constraints

Imposing constraints on a mechanical system is another way to represent external forces, in cases when these forces do not need to be resolved directly, and only their effect on the motion of the system needs to be captured. Our integration scheme can accommodate holonomic constraints, *i.e.*,

constraints that are acting on the configuration space of the system and can be written as

$$g(q,t) = 0.$$

Some examples of such constraints include incompressibility of an elastic body, non-penetration, or fixing some of the degrees of freedom. A convenient way to resolve such constraints is to write the Hamilton-Pontryagin principle in terms of the variables q, p, and v, while using Lagrange multipliers λ to impose $g(q) = 0$:

$$\delta \int_0^T \left[L(q,v) + p(\dot{q} - v) \right] dt + \lambda\, g(q) = 0.$$

The discrete counterpart is then given by:

$$\delta \sum_{k=0}^{N} p_{k+1}(q_{k+1} - q_k - h_k v_{k+1}) + L^d(q_k, v_{k+1}) + h_k \lambda_{k+1}\, g(q_{k+1}) = 0,$$

which yields the following *constrained discrete Hamilton-Pontryagin equations*:

$$\begin{aligned} q_{k+1} - q_k &= h_k v_{k+1} \\ p_{k+1} - p_k &= D_1 L^d(q_k, v_{k+1}) + h_k\, \lambda_k \nabla g(q_k) \\ h_k\ p_{k+1} &= D_2 L^d(q_k, v_{k+1}) \\ g(q_{k+1}) &= 0. \end{aligned} \tag{3.14}$$

These equations can be used by a non-linear solver to derive new positions in time satisfying the holonomic constraints. For the treatment of non-holonomic constraints, *i.e.*, constraints that involve conditions on the velocities (that cannot be written as $\nabla B(q) \cdot v = 0$) see [11, 40].

3.2.8 Variational Update

The time integrator that is based on (3.9-3.11) can be replaced by a variational update procedure done via *minimization of an energy-like function* given that the dynamical system satisfies certain integrability conditions as discussed below. This technique extends an idea of Radovitzky and Ortiz [61], where Verlet's integrator was shown to satisfy a *minimum principle*—a surprising fact given the extremum nature of Hamilton's principle. Our construction extends this property to a whole family of arbitrarily high order schemes that we call *fully-variational integrators* as a variational principle is not only used for their derivation, but *also* for numerical computations.

Variational Integrability Assumption We consider the class of dynamical systems whose discrete Lagrangian L^d has the property:

$$D_1 L^d(q_k, v_{k+1}) = D_2 P(q_k, v_{k+1}) \tag{3.15}$$

for some function $P(q_k, v_{k+1})$. The property (3.15) will be referred to as the *variational integrability property*. One can view this property as a design criterion that some (exceptionally nice) variational integrators might have, and in fact this condition is strictly equivalent to another formulation given in Section 2.8 of [44]. However, this particular property is not as restrictive as indicated in this reference: in fact, most current models used in Computer Animation satisfy it. Indeed, this property is valid for any *quadrature-based discretization* of a Lagrangian describing an *arbitrary* elastic model. Thus, our assumption is general, and can directly be used to design higher-order accurate schemes (through higher order quadrature rules which map continuous integrals to discrete sums [47]) still satisfying this integrability criterion.

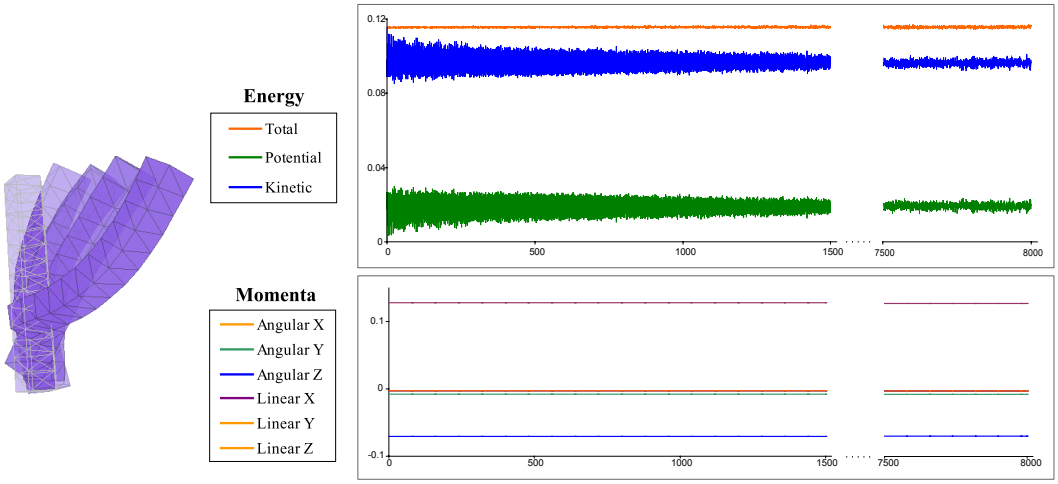


Figure 3.3: Momenta and energy behavior for explicit integration over 2 million time steps: non-linear elasticity with explicit integration is used to simulate an elastic rod (160 tets), given a non-zero initial position-momentum. No damping or external forces are used. Notice that the energy remains stable and the momenta are exactly preserved, even after 8000 seconds of simulation with a time step of $0.004s$.

Fully-Variational Update Now, start again with the variational equations (3.9-3.11). Clearly, (3.11) can be rewritten as:

$$\frac{\partial}{\partial v_{k+1}} \left[-h_k p_{k+1} v_{k+1} + L^d(q_k, v_{k+1}) \right] = -h_k p_{k+1} + D_2 L^d(q_k, v_{k+1}) = 0$$

We can substitute (3.10) in the above equation to get:

$$-h_k p_k - h_k D_1 L^d(q_k, v_{k+1}) + D_2 L^d(q_k, v_{k+1}) = 0$$

Thanks to the variational integrability property, this last equation can be rewritten as

$$\frac{\partial}{\partial v_{k+1}}[-h_k p_k v_{k+1} - h_k P(q_k, v_{k+1}) + L^d(q_k, v_{k+1})] = 0. \tag{3.16}$$

The quantity inside the bracket is an energy-like function of q_k, p_k and v_{k+1} and will be referred to hereafter as the ***Lilyan*** function $\mathcal{E}$:

$$\mathcal{E}(v_{k+1}) = -h_k p_k v_{k+1} - h_k P(q_k, v_{k+1}) + L^d(q_k, v_{k+1}). \tag{3.17}$$

The value of v_{k+1} can then be found as a *critical point* of the Lilyan. We can now state the following result:

Suppose that the variational integrability property (3.15) *holds. Given the triplet* (q_k, p_k, v_k)*, we can find* v_{k+1} *by minimizing the Lilyan defined by* (3.17)*, while* q_{k+1} *and* p_{k+1} *are then explicitly computed using* (3.9) *and* (3.10)*. The resulting triplet* $(q_{k+1}, p_{k+1}, v_{k+1})$ *satisfies* (3.9)*,* (3.10)*, and* (3.11)*, giving us a fully variational integration scheme. In particular, this procedure defines a (symplectic) update map* $(q_k, p_k) \mapsto (q_{k+1}, p_{k+1})$*.*

Proof. Of course (3.9) and (3.10) are satisfied by construction. We need to check that (3.11) holds when minimizing (3.17) with respect to v_{k+1}. However, this is a simple calculation:

$$\begin{aligned}
h_k p_{k+1} &= h_k p_k + h_k D_1 L^d(q_k, v_{k+1}) \quad \text{[definition of } p_{k+1}\text{]} \\
&= h_k p_k + h_k D_2 P(q_k, v_{k+1}) \quad \text{[using eq. (3.15)]} \\
&= \frac{\partial}{\partial v_{k+1}} \left[h_k p_k v_{k+1} + h_k P(q_k, v_{k+1})\right] \\
&= \frac{\partial}{\partial v_{k+1}} L^d(q_k, v_{k+1}) \quad \text{[assumed eq. (3.16)]} \\
&= D_2 L^d(q_k, v_{k+1}),
\end{aligned}$$

which is the desired equation (3.11). The last statement of our claim holds because this update map

is equivalent to the position momentum form of the DEL equation mentioned in [70]. □

Example. For the system mentioned in the example in Section 3.2.5 the update rules

$$
\begin{aligned}
& Mv_{k+1} + \frac{h}{2}\nabla W(q_k + \frac{1}{2}hv_{k+1}) - p_k = 0 \\
& q_{k+1} = q_k + hv_{k+1} \\
& p_{k+1} = Mv_{k+1} - \frac{h}{2}\nabla W(q_k + \frac{1}{2}hv_{k+1})
\end{aligned}
$$

can be written as the variational update procedure:

$$
\begin{aligned}
& \underset{v_{k+1}\in R^{3n}}{\operatorname{argmin}} \ \frac{1}{2}v_{k+1}^T Mv_{k+1} + W(q_k + \frac{1}{2}hv_{k+1}) - p_k v_{k+1} \\
& q_{k+1} = q_k + hv_{k+1} \\
& p_{k+1} = Mv_{k+1} - \frac{h}{2}\nabla W(q_k + \frac{1}{2}hv_{k+1}).
\end{aligned}
$$

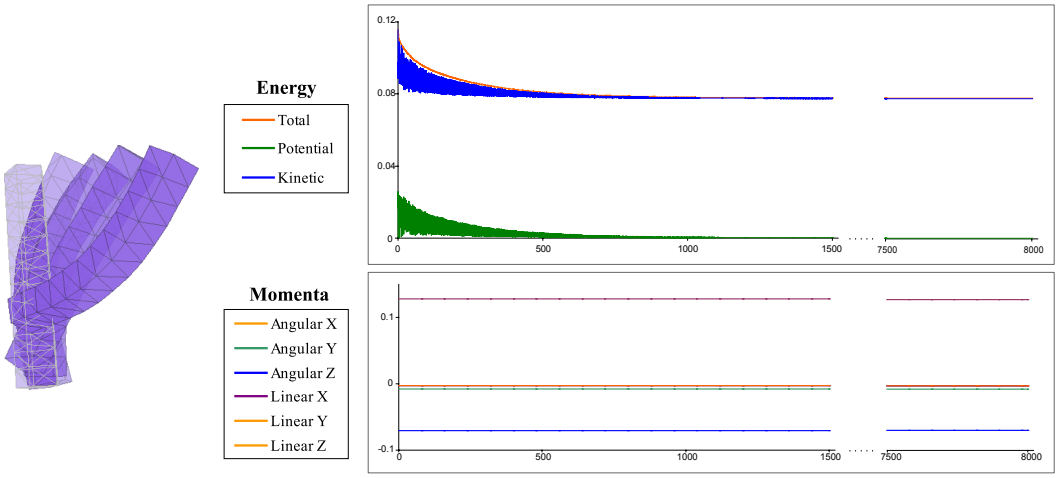


Figure 3.4: Damping is added to the same setup as in Fig. 3.3. The energy plot shows a smooth decrease over time, while momenta are *still* exactly preserved, even after 2 million time steps (explicit integration was used, with a constant time step of 0.004s).

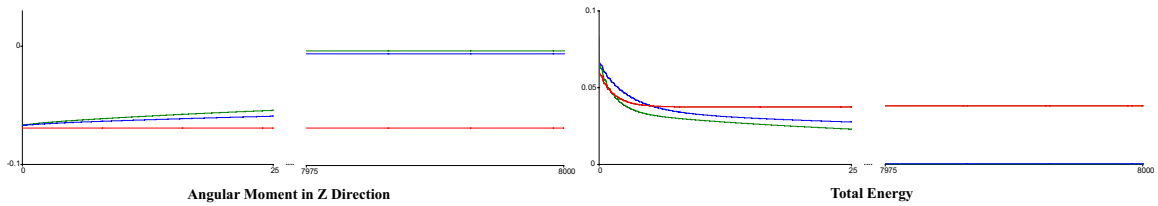


Figure 3.5: Comparison of our damping model (same setup as in Fig. 3.4) to numerical damping introduced by damped version of Newmark integrator [36]. Green and blue lines are angular momenta and energy of the bar with Newmark integrator with the time step 0.004 and 0.002 respectively. Red lines are angular momenta and energy of the bar when variational integrator is used with the time step 0.004 and 0.002 (notice that energy loss and momenta preservation is independent of the time step used).

3.2.9 Including External Forces into Variational Update

Most of the update rules that include external forces (such as Eqs. (3.13)) or constraints (such as Eqs. (3.14)) can also be transformed into the variational update procedure. To account for explicit forces or constraints the terms $F^{d+}(q_{k-1}, v_k) \cdot v_{k+1}$ and $\lambda_k \nabla g(q_k) \cdot v_{k+1}$ respectively are simply added to the Lilyan. To incorporate implicit forces $F^{d-}(q_k, v_{k+1})$, they need to be integrable, *i.e.*, there is a function $B(v_{k+1})$ such that

$$\nabla B(v_{k+1}) = F^{d-}(q_k, v_{k+1}).$$

Notice that such function exists when the Jacobian of the forces $\frac{\partial F^{d-}(q_k, v_{k+1})}{\partial v_{k+1}}$ is symmetric. If $B(v_{k+1})$ exists, it is added to the Lilyan, otherwise one will need to use this force explicitly, modify it to be integrable, or revert to root finding.

Example: Damping forces. The damping forces described in Section 2.3 can be incorporated into variational time integrators by either using the *explicit* damping force

$$F^{d+} = F_{\text{dampExp}}(q_{k-1}, q_k) = -k_D \nabla W(F(q_{k-1}, q_k)),$$

or the *implicit* damping force

$$F^{d-} = F_{\text{dampImp}}(q_k, v_{k+1}) = k_D \nabla W(F(q_{k+1}, q_k)).$$

However $F_{\text{dampImp}}(q_k, v_{k+1})$ is not integrable, so in practice we use a slight approximation:

$$F^{d}_{\text{dampImpApprox}}(q_k, v_{k+1}) = -k_D \nabla W(F(q_k, q_{k+1})).$$

Such force is integrable and is much more efficient to compute; although it does not conserve angular momenta exactly, it keeps the oscillations of the momenta bounded so the visual plausibility is not lost.

3.2.10 Resolving Collisions Using Penalty Potentials

Collisions and contact between deformable objects are an essential part of creating lively and realistic animations, so the problem of resolving collisions has received extensive attention from the computer graphics community. Numerous methods to handle collision response for rigid or deformable bodies have been developed, which can be loosely classified into constraint-based [3, 37], impulse-based [49], projection-based [38], or penalty-based [67] methods or often a combination of them [13, 30]. The geometric mechanics community has addressed variational approaches to collision response [22, 23], however, since these methods compute the exact time of each particle collision, they lack the efficiency needed for simulation of complex deformable objects.

In the examples presented in this thesis, a penalty-based, variational approach to collision response was employed. The penalty function is traditionally computed as a penetration distance, area, or volume. This function should have a minimum when the objects are touching and is set to zero when the objects are separated. Adding penalty function directly to the Lagrangian as an extra potential leads to a fully elastic collision response (the energy after the collision is reserved), eliminating artificial numerical friction and allowing for precise control of frictional forces. Similar to other penalty-based methods, the collisions are not exactly resolved at each given time, and we recommend using extra projection *only* for the visualization purposes in order to resolve these artifacts without corrupting the dynamics. A more robust approach to penalty-based, variational collision response was developed in [29] in the context of asynchronous variational integrators.

3.2.11 Discussion on Numerics

Current variational integrators resort to non-linear (root finding) solvers to find the next position so that it satisfies the DEL equations (typically using an algorithm such as Newton's method [60]). Our novel integration scheme is, so far, no different: Eq. 3.12 needs to be solved similarly. Although seemingly related to a *minimization*, solving a set of non-linear equations can be far more delicate. The reason is quite simple: while the notion of "downhill" for a scalar field is easy and well defined, it does not translate directly to the case of multidimensional fields where there are conflicting down-

hill directions in each dimension. To circumvent this issue, solvers traditionally use the notion of "merit function" (the squared norm of the residual) to monitor the progress made towards reaching the zero [54]. Significant computational gain could thus be achieved by having a scalar function to minimize instead, with lower order and complexity than the merit function. In fact, this idea is very much responsible for the success of the well-known Conjugate Gradient method to solve a linear system like $Ax = b$. Its foundations come from a minimization technique applied to the function $f(x) = \frac{1}{2}x^tAx - bx$. If one were to use the residual $\|Ax - b\|^2$ instead, the "merit function" has a term in $x^t(A^tA)x$, resulting in a worse condition number. When non-linear equations are to be solved, the gain can be even greater. A closer look shows that if h_{k+1} is small, the Lilyan $\mathcal{E}$ is quadratic in v_{k+1}: the terms depending on the potential energy are of order h_{k+1}^2, leaving only p_kv_{k+1} and the kinetic energy as terms of order h_{k+1}—and those form a quadratic function of v_{k+1}. Thus, for small enough time steps, one can *always* find v_{k+1} as the value that globally *minimizes* the Lilyan for the current values of q_k and p_k.

Improved Numerics for Implicit Integrators Although implicit symplectic integrators with fixed time-step size theoretically preserve energy extremely well, choosing a fairly large tolerance for the non-linear solver (as commonly used in graphics for efficiency reasons) can have spurious consequences in the long run: the inaccuracies in the solves can conspire to result in energy drift. A simple and inexpensive fix is to reinject whatever residual the solver gets after solving the DEL equations

$$\mathbf{r}_{k+1} = DEL(q_{k-1}, q_k, q_{k+1})$$

into the *next* DEL equations

$$DEL(q_k, q_{k+1}, q_{k+2}) + \mathbf{r}_{k+1} = 0$$

as if coming from an external forcing: in this fashion, numerical inaccuracies get accounted for at the next time step. Our tests show that this procedure allows for more slack on the tolerance without noticeable consequences (up to a certain point).

Example. Consider solving a system previously described in the example in Section 3.2.8:

$$\underset{v_{k+1}\in R^{3n}}{\operatorname{argmin}} \ \frac{1}{2}v_{k+1}^T M v_{k+1} + W(q_k + \frac{1}{2}hv_{k+1}) - p_kv_{k+1}.$$

The residual is computed as the value of the gradient of the above energy at the solution (ideally it

should be zero):

$$\mathbf{r}_{k+1} = Mv_{k+1} + \frac{h}{2}\nabla W(q_k + \frac{1}{2}hv_{k+1}) - p_k,$$

and reinjected into the system using momenta update:

$$p_{k+1} = Mv_{k+1} - \frac{h}{2}\nabla W(q_k + \frac{1}{2}hv_{k+1}) - \mathbf{r}_{k+1}$$

Notice that the above equation is equivalent to Equation (3.10) without accounting for the residual.

3.2.12 Pontryagin Version of the Discrete Noether's Theorem

A nice feature of discrete variational framework is that the relationship between symmetry and conserved quantities matches the continuous theory of mechanics. More precisely, the invariance of the (continuous) Lagrangian under a given set of transformations of its variables defines its symmetries. Clearly, these leave the action integral invariant as well. Thus symmetries give rise to *conserved quantities*, as stated in Noether's theorem. For example, the invariance of $L(q(t), \dot{q}(t))$ under translations and rotations results in the *conservation of linear and angular momenta*, respectively. This section shows how the invariants of the Lagrangian are used to derive conservation properties of the corresponding update rule.

Combining Equations (3.10) and (3.11), we get the Pontryagin version of DEL:

$$hD_1L^d(q_k, v_{k+1}) + D_2L^d(q_{k-1}, v_k) = D_2L^d(q_k, v_{k+1}). \tag{3.18}$$

In the following, suppose we organize q_k into groups of 3 DoFs that correspond to (x, y, z).

Linear and Angular Momenta To show the conservation of linear momenta, we assume invariance under a translation βu:

$$L^d(q_k + \beta u, v_{k+1}) = L^d(q_k, v_{k+1}),$$

where the same translation is applied to each group of D0Fs. Taking variations of this Lagrangian with respect to β at $\beta = 0$ gives the following equation:

$$D_1L^d(q_k, v_{k+1}) \cdot u = 0. \tag{3.19}$$

Plugging (3.19) into Equation (3.18)$\cdot u$ and using the expression for p_k and p_{k+1} from Equation (3.10), gives the linear momenta preservation formula expressed in terms of the discrete Lagrangian:

$$\frac{D_2L^d(q_k, v_{k+1})}{h} \cdot u = \frac{D_2L^d(q_{k-1}, v_k)}{h} \cdot u,$$

which implies

$$p_{k+1} \cdot u = p_k \cdot u.$$

A similar procedure can be done to show angular momenta preservation. Add an infinitesimal rotation $R_\omega(\beta)$ to the Lagrangian: $L^d(R_\omega(\beta)q_k, R_\omega(\beta)v_{k+1})$, where $R_\omega(\beta)$ is a block diagonal matrix with each block the same 3×3 rotation matrix representing a rotation around axis ω. Taking variations of this expression with respect to β gives:

$$D_1L^d(q_k, v_{k+1}) \cdot (\omega \times q_k) + D_2L^d(q_k, v_{k+1}) \cdot (\omega \times v_{k+1}) = 0, \tag{3.20}$$

where the cross product is applied to each group of 3 DoFs in q. Subtracting the above from Equation (3.18)$\cdot(\omega \times q_k)$ (and switching the order of cross and dot product using the triple product rule) gives the angular momenta preservation rule (around axis ω) as:

$$(\frac{D_2L^d(q_k, v_{k+1})}{h} \times q_{k+1}) \cdot (\omega \dots \omega)^t = (\frac{D_2L^d(q_{k-1}, v_k)}{h} \times q_k) \cdot (\omega \dots \omega)^t,$$

or, equivalently,

$$(p_{k+1} \times q_{k+1}) \cdot (\omega \dots \omega)^t = (p_k \times q_k) \cdot (\omega \dots \omega)^t,$$

where, again, cross product is done between groups of 3 DoFs.

Conservation of linear momenta and conservation of angular momenta can be seen as two special cases of the general Noether's theorem, where the invariance of the Lagrangian under translation and rotation is assumed. Similarly, if the Lagrangian is invariant under the action of another one parameter symmetry Lie group $G(\alpha)$, the preservation rule would be:

$$\frac{D_2L^d(q_k, v_{k+1})}{h} \cdot (G'(0)q_{k+1}) = \frac{D_2L^d(q_{k-1}, v_k)}{h} \cdot (G'(0)q_k).$$

Discrete Energy Behavior The symplectic nature of our scheme also guarantees good energy behavior. For conservative systems, the integration shows a nice energy preservation as demonstrated in Figure 3.3. An expression for the bounded difference of discrete energies $E(q_{k-1}, v_k) - E(q_k, v_{k+1})$ can be obtained by taking variations of the discrete action with respect to t_k. This expression is more conveniently derived from discrete Hamilton's principle, but can also be derived from Hamilton-Pontryagin principle. Below is an example derivation of the difference of discrete energies for the midpoint quadrature of the Lagrangian. Taking variation of (3.8) w.r.t. t_k (remember that $h_k = t_k - t_{k-1}$ and $h_{k+1} = t_{k+1} - t_k$) leads to:

$$\begin{aligned} &-\frac{Mv_{k+1}^2}{2} + W\left(q_k + \frac{h_{k+1}}{2}v_{k+1}\right) + p_{k+1}v_{k+1} + \frac{h_{k+1}v_{k+1}}{2}\nabla W\left(q_k + \frac{h_{k+1}}{2}v_{k+1}\right) \\ &-\left(-\frac{Mv_k^2}{2} + W\left(q_{k-1} + \frac{h_k}{2}v_k\right) + p_k v_k + \frac{h_k v_k}{2}\nabla W\left(q_{k-1} + \frac{h_k}{2}v_k\right)\right) = 0, \end{aligned} \tag{3.21}$$

using the identity obtained from taking variations of (3.8) w.r.t. v_{k+1}

$$p_{k+1} + \frac{h_{k+1}}{2}\nabla W\left(q_k + \frac{h_{k+1}}{2}v_{k+1}\right) = Mv_{k+1},$$

and plugging it into (3.21), the usual expression for the discrete energy is obtained: $E(q_k, v_{k+1}) = \frac{Mv_{k+1}^2}{2} + W\left(q_k + \frac{h_{k+1}}{2}v_{k+1}\right)$.

The proper treatment of forced systems handles energy dissipation gracefully as well (see Figure 3.4). Note that the energy dissipation in more traditional integrators is often a mix of user-prescribed damping *and* uncontrollable numerical viscosity (depending on the time step size). In sharp contrast, our algorithm allows a precise control of the amount of damping introduced in the simulation independent of the time step used for simulation—a particularly desirable property when the same simulation needs to be run with different time step, for example for preview purposes.

Special Case of Quadratic Potentials It is worth mentioning that, in special cases, when the Lagrangian is at most quadratic function of positions (*e.g.*, linear elasticity Lagrangian) and the potential energy is discretized using midpoint rule, the update equations obtained using variational integrator coincide with the discrete energy-momentum preserving updates from [64]. Thus, such update rule is both symplectic and energy preserving, where the energy exactly conserved is expressed now as $\tilde{E}_k = \frac{1}{2}p_k^T M^{-1} p_k + W(q_k)$.

3.3 Variational Approach to Time Adaption

While variational integrators offer robust and accurate time integration when a fixed time step size is used, the situation is quite different when the time step size is changed throughout simulation. In fact, there has been repeated evidence that naively changing the time step size can be quite harmful to the qualitative behavior of a simulation (see [65] and Fig. 3.6): symplecticity is no longer enforced, resulting in growing errors in energy and positions similar to non-symplectic integrators of equivalent accuracy order, *even* if the time step size is adapted in a way intended to improve accuracy [27]. This is a serious limitation, since a fixed time step is inefficient for most applications.

In recent years, time-adaptive integrators that maintain their symplectic nature have been proposed in the context of Hamiltonian systems [27, 14], often based on a Sundman/Poincaré transformation [9]. However, no equivalent has been proposed for variational integrators derived in the Lagrangian setting. We present a way to handle time adaption rigorously in the Lagrangian setting to preserve the typical numerical properties expected from a variational integrator.

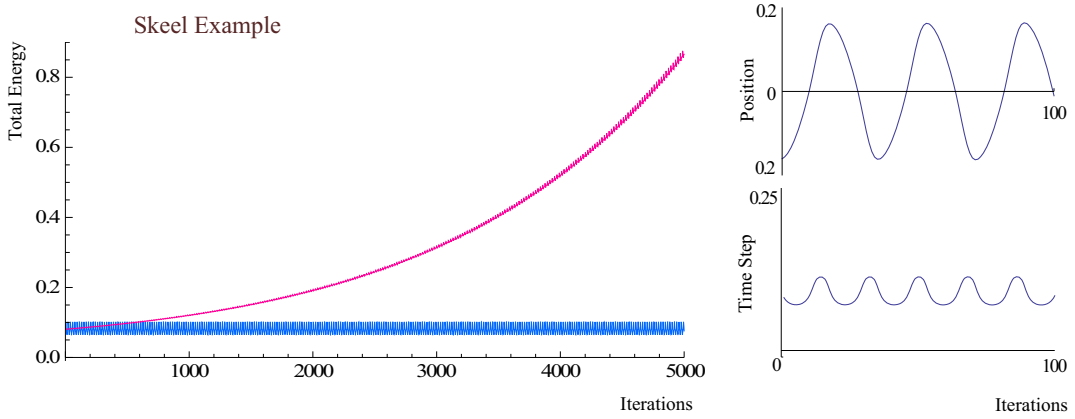


Figure 3.6: This 1D linear spring example, taken from Skeel [65], shows that certain time adaption strategies (in particular when time step is changed every quarter period) can lead to non-linear grows in energy (pink plot of the the total energy). Using our method for the same example leads to long term good energy behavior (blue energy plot).

3.3.1 Time Step Control

Both implicit and explicit integration methods require the choice of a time step size h for integration. If h is too small, the efficiency of an integration scheme can be dramatically low; if h is too large, numerical blow-ups are likely to occur. Thus, as dynamical systems usually evolve very unevenly through time (sometimes quickly, sometimes slowly), an obvious idea to optimize performance is

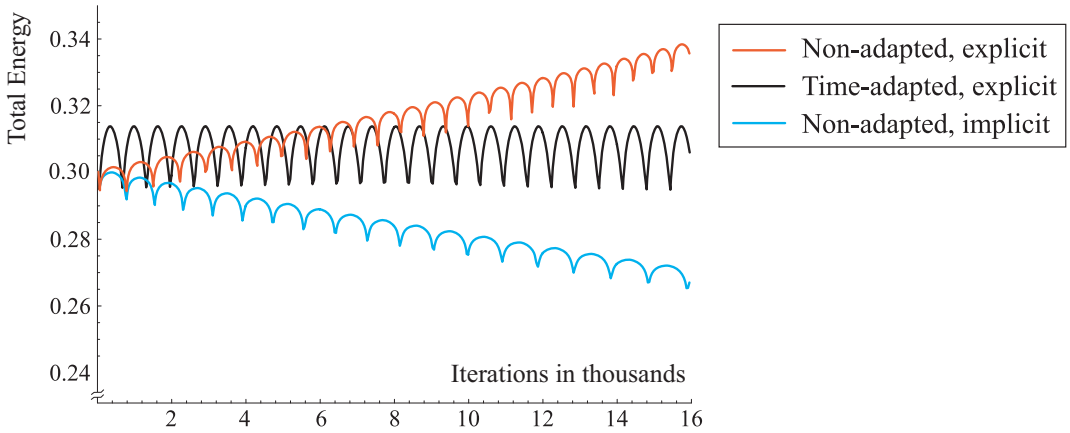


Figure 3.7: When a time step is naively changed during pendulum simulation as a function of current position (non-adapted explicit) or next position (non-adapted implicit), energy decay or grows can happen. Using out time adaption strategy fixes this drift in the energy.

to introduce variable time stepping. For generality, let's suppose that we have a heuristic for time adaption of the form:

$$t_{k+1} = t_k + h\sigma(q_k, q_{k+1}). \tag{3.22}$$

We will provide various such functions $\sigma(q_k, q_{k+1})$, or σ-*rules*, in Section 3.3.6, but remain agnostic about their exact expressions for now, as the design of a σ-rule is likely to be highly application-dependent. The only assumptions on σ we will make is that it is *differentiable*, and *bounded from below*, i.e., there exists a positive value $\sigma_{\min} > 0$ such that:

$$\forall(q, r),\ \ \sigma_{\min} \leq \sigma(q, r). \tag{3.23}$$

This value will guarantee that the simulation always goes strictly forward in time: for any given strictly positive h, time step sizes will be positive and larger than $h\sigma_{\min}$. Notice that for the trivial choice $\sigma = 1$, we get fixed time steps of size h.

3.3.2 Naive Enforcement of Time Adaption

In the case where σ depends *only* on q_k (i.e., for explicit time adaption heuristics), one could try to control the time step in a manner similar to previous adaption approaches: from the current position q_k, the next time step t_{k+1} is computed as $t_{k+1} = t_k + h\sigma(q_k)$; in turn, the next position q_{k+1} can be solved for using one's favorite time integrator. Although the order of accuracy can be maintained

even with such non-constant time step sizes, there is no guarantee that other numerical artifacts will not appear. In fact, Fig. 3.6 shows that such a time adaption strategy can fail even for a system as simple as a single spring.

One could instead be tempted to directly incorporate this time step update rule (explicit or implicit this time) into one's favorite *implicit integrator* to increase stability: it is likely more stable to find q_{k+1} *and* t_{k+1} *simultaneously* by performing a (non-linear) solve on the usual implicit update rules and the time step control in Eq. (3.22). While the stability of the resulting integrator may be improved, this approach is no longer symplectic, and can thus lead to energy drift, accompanied by numerical issues after a number of time steps (see Fig. 3.7 and [27]). A principled approach to time adaption is thus direly needed.

3.3.3 Hamilton Principle with Added Time Constraints

Rather than trying to add the time adaption into an existing time integration update rule such as Eq. (3.2) or Eqs. (3.9-3.11), we go back to Hamilton's key principle behind variational integrators and include a constraint to ensure time step control directly at the level of the discrete action. This will allow us to formulate the integrator in such a way that, by leveraging the symplecticity of variational integrators for fixed time steps, both stability *and* correct qualitative behavior can be achieved. For clarity we first assume no external, non-conservative forces, before presenting the full treatment. We still want to satisfy Hamilton's principle numerically, but this time under the constraint of the time adaptation rule $t_{k+1} - t_k = h\sigma(q_k, q_{k+1})$. Notice that it reduces to the regular case of variational integrators with constant time step if $\sigma(q_k, q_{k+1}) = 1$. To solve such a constrained extremization problem, a common technique is to enforce the constraint using a Lagrange multiplier λ as described in Section 3.2.7. In our case, we can rewrite the discrete (but now *constrained*) action $\widehat{S}_0^K$ as a function of the q_k's, t_k's, and λ_k's:

$$\widehat{S}_0^K = \sum_{k=0}^{K-1} \left[\mathcal{L}(q_k, q_{k+1}, t_{k+1} - t_k) + \lambda_k (t_{k+1} - t_k - h\sigma(q_k, q_{k+1})) \right] \tag{3.24}$$

Hamilton's principle states that the solution path $\{q_k, t_k\}$ is a critical point of the discrete time-adapted action $\widehat{S}$ for fixed end points (q_0, t_0) and (q_K, t_K). Thus, variations of the action with respect to q_k, t_k, and λ_k must vanish (the variation w.r.t t_k is often called *horizontal variation* in Lagrangian

mechanics literature), yielding respectively:

$$D_1\mathcal{L}_{k,k+1} + D_2\mathcal{L}_{k-1,k} - h\lambda_{k-1}\frac{\partial\sigma(q_{k-1},q_k)}{\partial q_k} - h\lambda_k\frac{\partial\sigma(q_k,q_{k+1})}{\partial q_k} = 0 \tag{3.25}$$

$$\lambda_k = \lambda_{k-1} + (E_{k+1} - E_k) \tag{3.26}$$

$$t_{k+1} = t_k + h\,\sigma(q_k, q_{k+1}) \tag{3.27}$$

where E_{k+1} is the discrete energy expressed as

$$E_{k+1} = -D_3\mathcal{L}(q_k, q_{k+1}, t_{k+1} - t_k),$$

E_0 is thus the initial energy of the system. These equations form an update rule, which solves for t_{k+1}, q_{k+1}, and λ_k from $t_{k-1}, t_k, q_{k-1}, q_k$, and λ_{k-1}. The system is bootstrapped with t_0, t_1, q_0, q_1, and $\lambda_0 = 0$. One can verify that the case $\sigma = 1$ gives back the ordinary DEL equation for fixed time step sizes, as Eq. (3.25) reduces to Eq. (3.2). Notice that Eq. (3.27), resulting from the variation with respect to λ_k, is the desired time update rule as expected; note also that Eq. (3.26) can be rewritten as $\lambda_k = E_{k+1} - E_0$.

3.3.4 Convergence Analysis

A closer look at Eqs. (3.25), (3.27), and (3.26) reveals that if λ_k stays near zero, the new time-adapted DEL equation is only a slight modification to the fixed-time-step DEL equation (Eq. (3.2)), and the fixed-time-step discrete energy E_k stays near the initial energy E_0. In other words, as long as λ_k remains small enough, our integrator provides a discrete path in space-time whose local flow remains nearby the one defined by the original system without losing its long term energy preservation properties.

To convince ourselves that any time adaption rule σ will have this property (and thus, result in a valid integrator), let us define *fictitious time steps* τ_k that are equispaced such that $h = \tau_{k+1} - \tau_k$. We may now view the above integrator as a variational integrator with fixed (fictitious) time steps h of a *modified* mechanical system with an action integral:

$$\sum_{k=0}^{K-1} [\mathcal{L}(q_k, q_{k+1}) + \lambda_k(t_{k+1} - t_k - (\tau_{k+1} - \tau_k)\sigma(q_k, q_{k+1}))]$$

Now taking variations with respect to τ_k gives the difference of discrete energies as:

$$\widehat{E}_{k+1} - \widehat{E}_k = \lambda_k \sigma(q_k, q_{k+1}) - \lambda_{k-1}\sigma(q_{k-1}, q_k)$$

Thus the discrete energy of this time-adapted system being essentially preserved by this integrator is:

$$\widehat{E}_{k+1} = \lambda_k\, \sigma(q_k, q_{k+1}) = (E_{k+1} - E_0)\, \sigma(q_k, q_{k+1}).$$

Since this is now a variational integrator with *fixed time steps*, albeit of a different system, we inherit the long-time energy conservation of symplectic schemes. In particular, it means that there is essentially no accumulation of errors in time of the energy $\widehat{E}$, and since $\widehat{E}_0 = 0$ by definition, we get $\widehat{E}_k = O(h^2)$ over exponentially long time intervals for integrators with quadratic accuracy. Since we required that σ be bounded from below by $\sigma_{\min}$, we have $|E_k - E_0| = O(h^2/\sigma_{\min})$, guaranteeing that the original energy E_k remains near E_0 if h is small enough: no noticeable energetic gain or loss is added to the system through time adaption, and each iteration corresponds to a slightly modified flow of the system we wish to simulate. (Note that since $|E_k - E_0| = O(h^2/\sigma_k)$, the original discrete energy may get further away from E_0 than for constant time steps due to the approximate nature of this energy estimate; but no error accumulation will be produced.) Consequently, our treatment of time adaption preserve the same numerical properties as variational integrators with fixed time steps.

3.3.5 External and Dissipative Forces

In continuum Lagrangian mechanics, external forces that cannot be expressed as deriving from a potential (and therefore result in loss or gain of energy) are treated through the Lagrange-d'Alembert principle, an extension of Hamilton's principle to account for the added dissipative forces:

$$\delta \int_0^T L dt + \int_0^T F(\delta q - \dot{q}\delta t) dt = 0,$$

where F is the external force. An important remark is in order here: notice that former approaches to dissipative forces for symplectic integrators do *not* include the term $\dot{q}\delta t$ as they are not considering variations of time [47]. However, the effective virtual spatial displacement for a variation in space *and* time does require this term: a change in the time sequence $\{t_k\}_0^N$ with a fixed space sequence $\{q_k\}_0^N$ results in a change in space $q(t)$ for a given instant t. Implementing this variational principle

is achieved quite simply through a two-point quadrature rule of the external forces integrated over a time interval $[t_k, t_{k+1}]$, leading to the expression of the approximate forcing term as:

$$[F^-(q_k, q_{k+1})(\delta q_k - v_{k,k+1}\delta t_k) + F^+(q_k, q_{k+1})(\delta q_{k+1} - v_{k,k+1}\delta t_{k+1})](t_{k+1} - t_k),$$

where $F^-(q_k, q_{k+1})$ and $F^+(q_k, q_{k+1})$ are the forces evaluated at the first and second quadrature points of the time interval respectively, and $v_{k,k+1} = \frac{q_{k+1}-q_k}{t_{k+1}-t_k}$ is the velocity between times t_k and t_{k+1}. Incorporating these forcing terms into the time-adapted Euler-Lagrange equations, Eq. (3.25) becomes:

$$D_1\mathcal{L}(q_k, q_{k+1}) + D_2\mathcal{L}(q_{k-1}, q_k) - h\lambda_{k-1}\frac{\partial\sigma(q_{k-1}, q_k)}{\partial q_k} - h\lambda_k\frac{\partial\sigma(q_k, q_{k+1})}{\partial q_k} + F^+(q_{k-1}, q_k)(t_k - t_{k-1}) + F^-(q_k, q_{k+1})(t_{k+1} - t_k) = 0,$$

and Eq. (3.26) becomes:

$$\lambda_k = \lambda_{k-1} + (E_{k+1} - E_k) - F^+(q_{k-1}, q_k)(q_k - q_{k-1}) - F^-(q_k, q_{k+1})(q_{k+1} - q_k).$$

These changes correspond to accounting for the external forces in the momentum update, and compensating for the work done by these forces in the change of energy, respectively. Note that the latter change will keep λ close to 0 as the energy level changes over time based on the external forces. In particular, damping forces often used in computer animation are accurately treated by this formulation despite uneven time step sizes. Similarly, nonconservative collision or friction forces can also be handled this way.

3.3.6 Examples of Particular Integrators

We now discuss the design of symplectic time adapted integrators by providing general principles as well as specific examples of σ-rules and different discretizations of the Lagrangian useful in the context of computer animation. We tested a series of relevant adaption strategies suggested in [27], including:

- Equispaced poses:
  $\sigma(q_k, q_{k+1}) = 1/\sqrt{E_0 - W(\frac{q_k+q_{k+1}}{2}) + \epsilon}$.
- Equispaced phase space points:

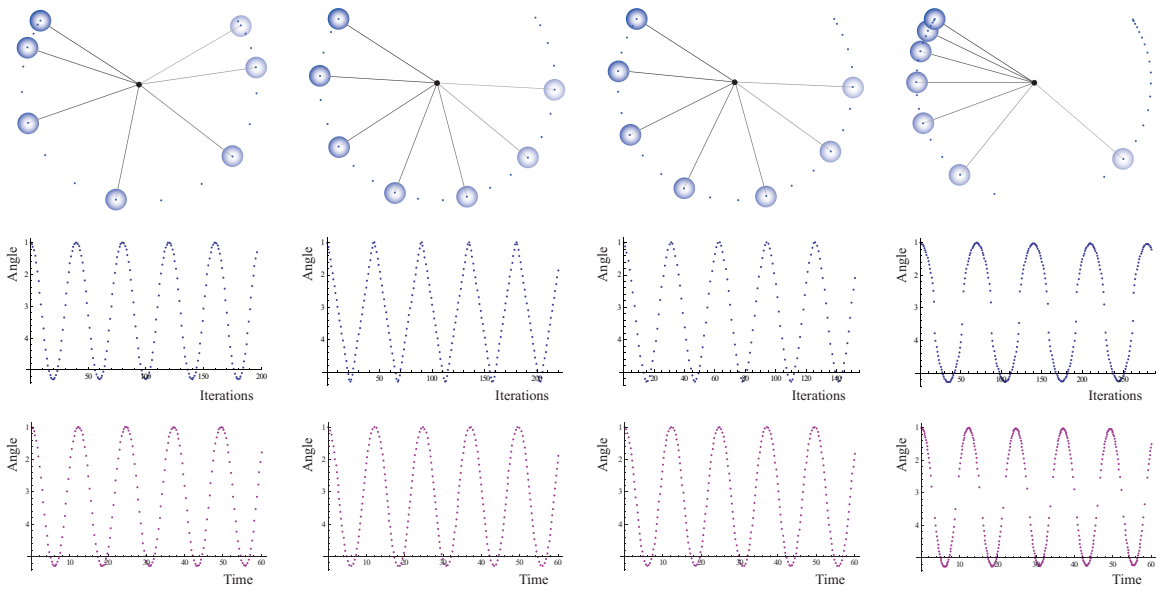


Figure 3.8: Using various time adaption strategies for a simple pendulum. The top row shows the motion of the pendulum for one period; the middle and the bottom rows plot the angle of the pendulum with respect to iterations and time respectively. (a) integration of the period with the fixed timestep, (b) equispaced positions, (c) equispaced phase space points, (d) time step adapted to acceleration.

$$\sigma(q_k,q_{k+1}) = 1/\sqrt{E_0 - W(\tfrac{q_k+q_{k+1}}{2}) + \|\nabla W(\tfrac{q_k+q_{k+1}}{2})\|^2 + \epsilon}.$$

- Time step adapted to acceleration:

  $\sigma(q_k, q_{k+1}) = \|\nabla W(q_k) + \nabla W(q_{k+1}) + \epsilon\|^{-1}$.

The results of using these particular σ-rules are shown in Figure 3.8. Notice that the coefficient ϵ serves two purposes, preventing division by zero while also providing a guaranteed lower bound for σ as required for symplecticity. More application-specific rules, such as decreasing time steps when the object is in a given region of space (for example for resolving collisions), can easily be implemented as well. Particularly for explicit integrators, it is often useful to base σ on a "smoothed" energy function, so that time steps can be decreased shortly before reaching a numerically stiff state. In fact, even totally aphysical rules can lead to a successful integrator, although this often places a heavy burden on the numerical solver.

After one picks a relevant time adaption strategy: the quadratures for integrating the Lagrangian, and the σ-function, the equations of motions are derived using Eqs. (3.25-3.27). For example,

update rules based on a discrete Lagrangian $L = \frac{(q_{k+1}-q_k)^T M(q_{k+1}-q_k)}{2(t_{k+1}-t_k)} - W(q_k)(t_{k+1}-t_k)$ and an explicit adaption function σ are:

$$
\begin{aligned}
q_{k+1} &= q_k + M^{-1}\left(\frac{M(q_k - q_{k-1})}{(t_k - t_{k-1})} - (t_{k+1} - t_k)\nabla W(q_k) - \lambda_k \nabla\sigma(q_k)\right)(t_{k+1} - t_k) \\
\lambda_k &= \lambda_{k-1} + \frac{(q_{k+1} - q_k)^T M(q_{k+1} - q_k)}{2(t_{k+1} - t_k)^2} + W(q_k) - \frac{(q_k - q_{k-1})^T M(q_k - q_{k-1})}{2(t_k - t_{k-1})^2} - W(q_{k-1}) \\
t_{k+1} &= t_k + h\sigma(q_k)
\end{aligned}
$$

Notice that even though the discretizations of both Lagrangian and σ are explicit, the resulting update equations are implicit because the update of λ_k is quadratic in q_{k+1}, and the update of q_{k+1} depends on λ_k. Here is another example of the update equations, now in the case when midpoint quadrature is used for both Lagrangian and σ:

$$
\begin{aligned}
&\frac{M(q_k - q_{k-1})}{(t_k - t_{k-1})} - \frac{M(q_{k+1} - q_k)}{(t_{k+1} - t_k)} - \frac{(t_{k+1} - t_k)}{2}\nabla W\left(\frac{q_k + q_{k+1}}{2}\right) - \frac{(t_k - t_{k-1})}{2}\nabla W\left(\frac{q_{k-1} + q_k}{2}\right) \\
&\qquad - \frac{\lambda_{k-1}}{2}\nabla\sigma\left(\frac{q_{k-1} + q_k}{2}\right) - \frac{\lambda_k}{2}\nabla\sigma\left(\frac{q_k + q_{k+1}}{2}\right) = 0, \\
&\lambda_k = \lambda_{k-1} + \frac{(q_{k+1} - q_k)^T M(q_{k+1} - q_k)}{2(t_{k+1} - t_k)^2} + W\left(\frac{q_k + q_{k+1}}{2}\right) - \frac{(q_k - q_{k-1})^T M(q_k - q_{k-1})}{2(t_k - t_{k-1})^2} - W\left(\frac{q_{k-1} + q_k}{2}\right), \\
&t_{k+1} = t_k + h\sigma\left(\frac{q_k + q_{k+1}}{2}\right).
\end{aligned}
$$

In this case, all the above equations (in total $3M + 2$ equations in $3D$ for a mesh with M vertices) need to be solved simultaneously using an appropriate non-linear solver.

3.3.7 Limitations

Our symplectic approach to time-adapted integrators requires similar type of non-linear solves as traditional implicit DEL equations for the fixed time step. The only noticeable difference is in the computation of the Jacobian of DEL: now that time *and* space are both solved for, the Jacobian is no longer symmetric; moreover, its terms due the $\nabla\sigma$ term in the DEL equations (Eq. (3.25)) can be quite involved, and make the matrix potentially dense. To palliate this last numerical issue, we simply disregard these terms, thus considerably simplifying the assembling of the Jacobian matrix—as is commonly done for complex energies (see, e.g., [4]). We did, however, notice that this simplification can result in more iterations needed for the convergence of the nonlinear solver.

The most important limitation that we have found while testing our method, is that the fictitious time

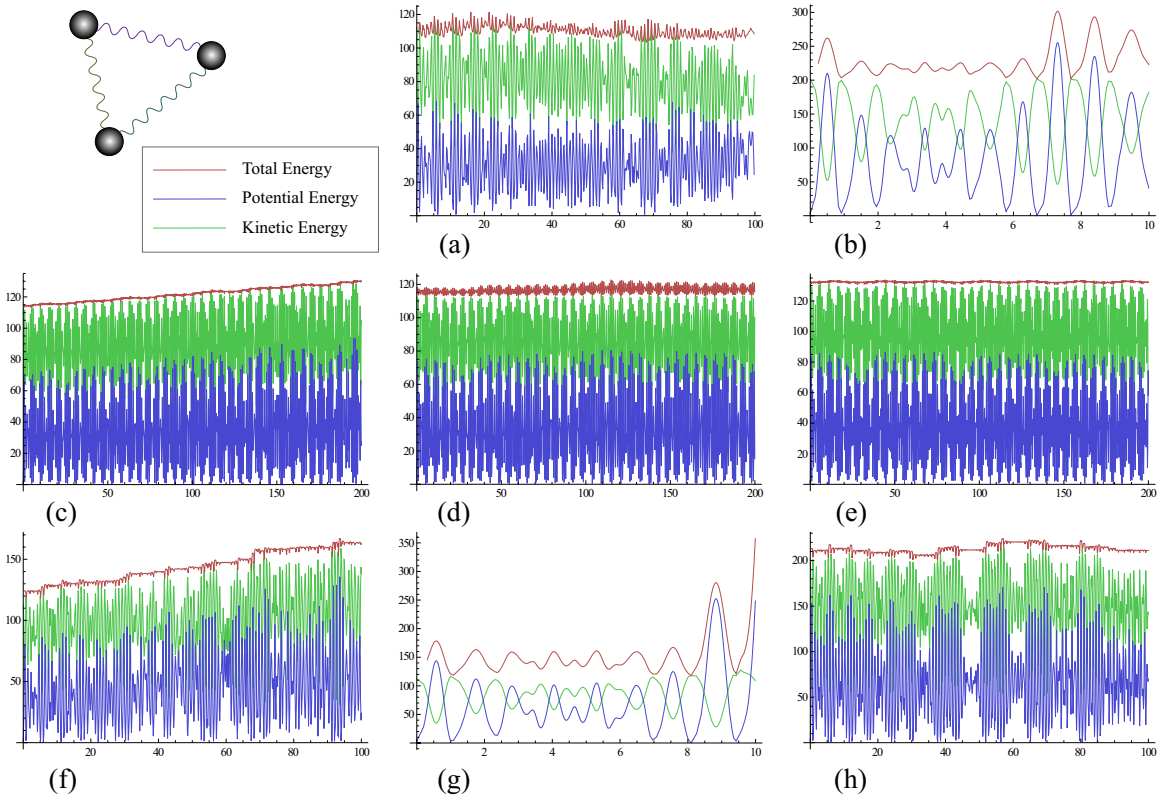


Figure 3.9: Energy plots of time adaption tests for a system of 3 point masses connected by 3 linear springs, the Lagrangian is discretized using midpoint quadrature. The time adaption strategy is picked so that $\sigma = \frac{1}{W/10+1}$ for (c, d, e) and $\sigma = \frac{2}{W/10+1}$ for (f, g, h, and b). In (c) and (f) naive time adaption is performed (new time step is computed using explicit σ-rule and is than used in the standard DEL equation); (d) and (g) use symplectic time adaption and explicit σ, and in (e) and (h) midpoint discretization is used, but symplecticity is not enforced, finally (b) uses midpoint discretization of σ with symplecticity enforced. Note that for larger timesteps symplectic time adaptive integrators "blow up" while time reversible ones (e and h) remains stable. Moreover, (a) shows that the symplectic integrator with the large constant time step (where the size of the timestep is the same as the largest step in all the other simulations) still behaves reasonably, while being significantly more efficient.

step $h = \tau_{k+1} - \tau_k$ might need to be decreased to guarantee the stability of the modified system. As Figure 3.9 shows, in such cases the stability restriction on real time steps $t_{k+1} - t_k$ could be stronger than if one was to use traditional variational integrators with the constant time step. Thus, if the goal of adapting time step size is achieving efficiency, symplectic time adaption may not always be the right choice. We now discuss an alternative, time reversible approach to time adaption that according to our tests allows for larger time steps.

3.4 Time Reversible Integrators

Time reversible continuous mechanical system have the property that if the initial velocity is negated and the initial position is kept the same, the direction of the motion of the system is inverted, but the shape of the trajectory in space-time remains the same. In other words, the system will follow the same motion backwards when run with negative timesteps. It turns out that keeping time reversibility property when creating discrete integrators leads to numerical solutions that have long-time behavior similar to that of symplectic integrators [27]. Discrete update rule $\Psi(q_k, q_{k+1}, q_{k+2}, h_{k+1}, h_{k+2}) : (q_k, q_{k+1}, h_{k+1}, h_{k+2}) \to q_{k+2}$ is said to be *time-reversible* or *symmetric* if $\Psi(q_k, q_{k+1}, q_{k+2}, h_{k+1}, h_{k+2}) = -\Psi(q_{k+2}, q_{k+1}, q_k, -h_{k+2}, -h_{k+1})$. For example, discrete Euler-Lagrange equation (3.2) is both symplectic and time-reversible when:

$$D_1 L_d(q_{k+1}, q_{k+2}, h_{k+2}) + D_2 L_d(q_k, q_{k+1}, h_{k+1}) = -D_1 L_d(q_{k+1}, q_k, -h_{k+1}) - D_2 L_d(q_{k+2}, q_{k+1}, -h_{k+2}). \tag{3.28}$$

In general, if the discrete Lagrangian has the property:

$$L_d(q_0, q_1, h) = -L_d(q_1, q_0, -h) \tag{3.29}$$

then the variational integrator obtained from such Lagrangian is time-reversible.

Proof. Differenting both side of the Eq. (3.29) by q_0 gives

$$D_1 L_d(q_0, q_1, h) = -D_2 L_d(q_1, q_0, -h).$$

This implies that $D_1 L_d(q_{k+1}, q_{k+2}, h_{k+2}) = -D_2 L_d(q_{k+2}, q_{k+1}, -h_{k+2})$ and $D_2 L_d(q_k, q_{k+1}, h_{k+1}) = -D_1 L_d(q_{k+1}, q_k, -h_{k+1})$, making the right and the left hand sides of Eq. (3.28) equal. □

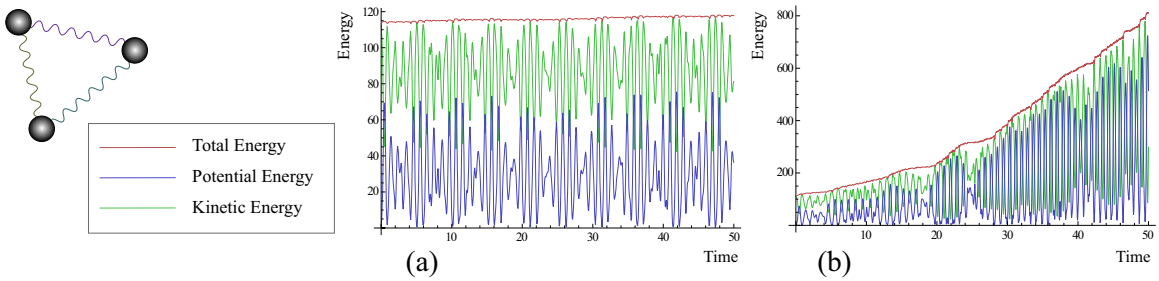


Figure 3.10: The same set-up as on the Figure 3.9: a system of 3 point masses connected by 3 linear springs, $\sigma = \frac{1}{W/10+1}$. (a) shows energy plot of the simulation where naive time adaption is performed (new time step is computed using explicit σ-rule and is than used in the standard DEL equation obtained from the Lagrangian which is discretized using midpoint quadrature: $\frac{(q_{k+1}-q_k)^T M(q_{k+1}-q_k)}{2h} - W(\frac{q_k+q_{k+1}}{2})h$); (b) shows energy plot of the simulation with the same σ discretization, but the Lagrangian is now not symmetric: $\frac{(q_{k+1}-q_k)^T M(q_{k+1}-q_k)}{2h} - W(\frac{3q_k}{4} + \frac{q_{k+1}}{4})h$. Notice that a symmetric update rule yields a slight energy drift when the time step is not constant, while a non-symmetric update yields the fast "blow up" of the simulation.

Examples of time reversible discretizations of the Lagrangian are midpoint and trapezoidal rules. For the systems that are time reversible, the use of time reversible variational integrators is preferred. Such integrators do not only capture the symmetry of the system on the discrete level, but also lead to update rules that are at least quadratic accurate. This property is especially important when the time step size keeps on changing, because linear accuracy in time can than cause very large errors, see Figure. 3.10.

3.4.1 Time Reversible Approach to Time Adaption

It turns out that enforcing symplecticity for timestep adaptation is not always required: one can obtain good long-term behavior using time reversible approach to time adaption (see Figure 3.9). A time adaptive integrator is time reversible (or symmetric) when both its update rule and its σ rule are symmetric. One type of integrators that satisfy this property can be written as:

$$\hat{h}_{k+1} = h\sigma(q_k, q_{k+1}) = h\sigma(q_{k+1}, q_k) \tag{3.30}$$

$$\hat{h}_{k+2} = h\sigma(q_{k+1}, q_{k+2}) = h\sigma(q_{k+2}, q_{k+1}) \tag{3.31}$$

$$D_1 L_d(q_{k+1}, q_{k+2}, \hat{h}_{k+2}) + D_2 L_d(q_k, q_{k+1}, \hat{h}_{k+1}) = 0, \tag{3.32}$$

where Equation (3.32) is a symmetric update rule as defined in Equation (3.28). Most of the useful σ rules will be nonlinear functions of the positions, in particular q_{k+2}, thus making update 3.32 implicit. To learn about how to design explicit, time reversible, integrators see [28].

Notice that when Equations (3.25-3.27) are discretized using midpoint quadrature or trapezoidal rule they also give another type of time reversible update rules which are in addition symplectic. However, for mechanical systems that we tested such symplectic/symmetric update rules generally require smaller timesteps and yield larger energy oscillations, thus they are less efficient and less stable when larger time steps are taken.

3.4.2 Time Reversible Co-Rotational Methods

Co-rotational methods described in Section 2.2.3 are used to reduce the problems that arise due to the fact that linear elasticity is not invariant under rigid body rotation. The main idea is to modify the linear system to be solved to account for the rotation of each element. Traditionally, time discretization is done using Implicit Euler or Runge-Kutta integrators and the stiffness matrix is modified using rotations computed at the current configuration (see [50, 31, 53, 19, 25]). Such discretization suffers from the usual numerical dissipation and, moreover, the time at which the rotations are computed is different from the time at which the elastic forces are computed, so the angular momenta of the system is not conserved. We will address these issues in this section.

First, we will consider time-reversible, variational, energy-preserving integrator for linear elasticity problem, obtained using midpoint discretization of quadratic elastic energy. Following the update procedure from Section 3.2.5 we write the update equations for the case of linear elasticity as:

$$\begin{aligned} Mv_{k+1} + \frac{h}{2}Ku_{k+\frac{1}{2}} - p_k &= 0 \\ q_{k+1} &= q_k + hv_{k+1} \\ p_{k+1} &= Mv_{k+1} - \frac{h}{2}Ku_{k+\frac{1}{2}}, \end{aligned}$$

where $K = \nabla^2 W(q_k + \frac{1}{2}hv_{k+1})$ is the constant stiffness matrix as described in Section 2.2.2, and $u_{k+\frac{1}{2}} = q_k + \frac{1}{2}hv_{k+1} - q_0$ is the total midpoint displacement. So the linear system that needs to be solved is:

$$v_{k+1} = \left[M + \frac{h^2}{4}K\right]^{-1} \left(-\frac{h}{2}(Kq_k - Kq_0) + p_k\right).$$

Since $Ku_{k+\frac{1}{2}}$ corresponds to the elastic forces computed at time $t_{k+\frac{1}{2}}$, in order to achieve proper symplectic integrator we would need to use the rotations computed at the time $t_{k+\frac{1}{2}}$ to modify the stiffness matrix K. Unfortunately, since the rotations are not linear, such integrator will require

solving a system of non-linear equations each update of v_{k+1}. Instead we will still use the rotations computed using known positions at time t_k, but we will use rotations computed at time t_{k+1} to update momenta p_{k+1}:

$$\begin{aligned} v_{k+1} &= \left[M + \frac{h^2}{4} \sum_e R_k^e K^e R_k^{eT} \right]^{-1} \left(-\frac{h}{2} (\sum_e R_k^e K^e R_k^{eT} q_k - \sum_e R_k^e K^e q_0) + p_k \right) \\ p_{k+1} &= M v_{k+1} - \frac{h}{2} \sum_e R_{k+1}^e K^e R_{k+1}^{eT} (q_k + \frac{1}{2} h v_{k+1}) - \sum_e R_{k+1}^e K^e q_0 \end{aligned} \tag{3.33}$$

This update rule is time reversible in R^e and q, so it is not a surprise that our empirical tests show good long term behavior of the simulated system. While this integrator does not exactly conserve energy or angular momenta, our test show that both quantities stay bounded during the simulations. Notice that this update procedure does not assume a specific rule of how the rotations R^e are computed, so any of the approaches described in the literature can be used. A development of (implicit) symplectic integrators for the co-rotational methods is left as a future work. Depending on the applications, one can use explicit symplectic integrators, since they will only require computation of the rotations in the known configurations. However, the implicit integrator described in Eqs. (3.33) allows for significantly larger timesteps compared to explicit ones, and thus, is much more efficient, especially for stiff elastic systems.

Chapter 4

Material Upscaling

4.1 Introduction

Simulating ever larger and more complex dynamical systems requires ever more elaborate computational methods. While improved CPU speed and faster numerical solvers (see for example Chapter 2 and Chapter 3 of this thesis) have allowed exquisitely detailed animation of complex deformable models, one outstanding limitation in computer animation (and in fact, in computational physics) is that simulation costs scale with structural complexity: capturing the proper dynamical behavior of a heterogeneous object requires a mesh fine enough to resolve the fine-scale heterogeneities. This sampling requirement can lead to prohibitive simulation times if the fine scales are geometrically complex, such as veinal structures in an organ. However, simply ignoring these fine scales can dramatically affect the overall dynamics of the object, rendering the object more or less rigid, or even failing to capture basic coarse deformation.

We present an approach to numerical coarsening of linear elastic objects to allow for interactive, realistic physical animation of structurally complex deformable objects. From a pair of meshes representing respectively a fine and a coarse geometric description of the elastic body, we devise a numerical procedure to turn the heterogeneous elastic properties of the fine mesh into possibly anisotropic elastic properties on the coarse mesh that effectively capture (in the H^1 sense) the same physical behavior. Simulation of the coarse mesh equipped with these resulting coarse elastic properties is thus faithful to the dynamics of the original fine-detailed object, but at a fraction of the cost.

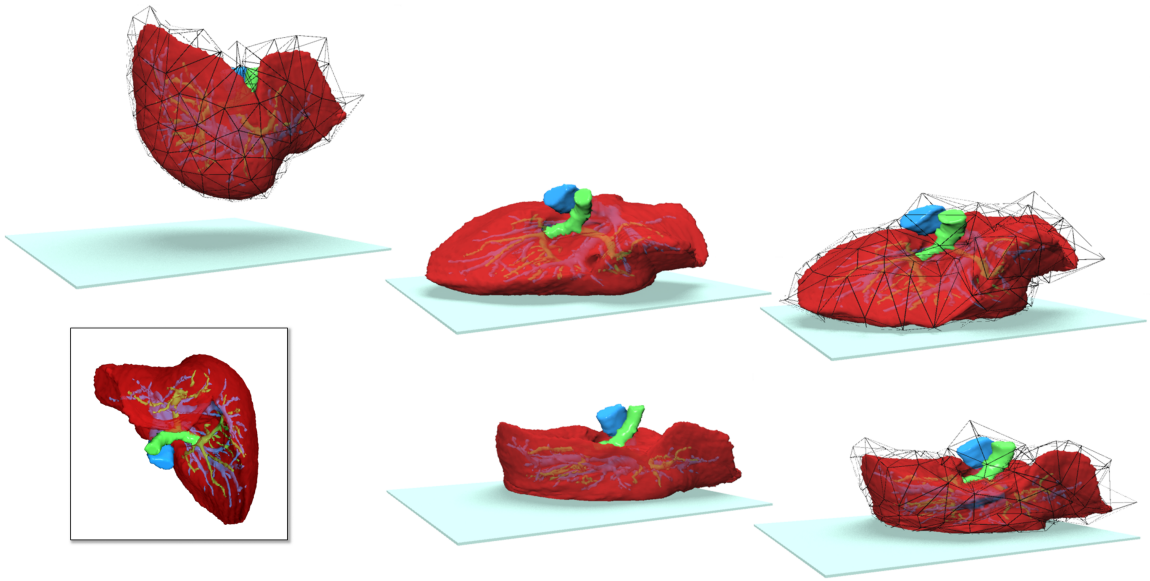


Figure 4.1: Numerical coarsening turns a *fine* mesh with heterogeneous elastic properties (here, a 200K-tet liver with veins, extracted from MRI data; courtesy of Dobrina Boltcheva, LSIIT, France) into a *coarse* mesh (640 tets) with anisotropic elastic properties that effectively capture the same physical behavior. The coarse mesh (top) can thus be used as a proxy to animate the object (falling on the ground) about a hundred times faster than it would take to compute the elastic behavior on the fine mesh (bottom). Collision detection is done using the interpolated fine mesh boundary.

4.2 Previous Work

Simplifying a model while still capturing (numerically or visually) its coarse physical behavior has been investigated in many areas of science, of which we discuss the two most relevant to our context.

4.2.1 Fast Deformable Models

As the complexity of objects required in computer animation grew, several strategies were devised to address the problem of efficiency and scalability. Solvers were made to scale nearly linearly in the number of nodes, through specific integration schemes providing fast, yet stable updates [4] and/or multiresolution strategies for which adaptive refinement of the simulation is often based on the local amount of deformation [20, 26, 16, 53]. Physical models were also drastically simplified to reduce computational requirements [62], leading to a resurgence of (sometimes quasi-static) linear elasticity [33] where the well-known limitations for large deformation are nicely compensated for by corotational methods as discussed in Section 2.2.3.

Another string of contributions focused on *model reduction* to achieve further efficiency. These

approaches reduced the state space dimension by limiting the space of possible deformations, typically though eigen-analysis of either the stiffness matrix [58] or of a set of observations [41]. Model reduction, however, often leads to a significantly reduced set of admissible interactions [32], to non-local reduced (Ritz) basis functions and a mismatch between geometric and deformation DOFs [58, 34, 17, 69], and/or to non-linear runtime complexity [45, 5, 1].

A simple, yet efficient way to simulate complex objects at low cost is through *domain embedding*, where a complex geometry is embedded in a coarse mesh (sometimes referred to as a cage). Animating the coarse volumetric mesh induces deformation of the high resolution embedded geometry, providing a cheap and easy dynamic approximation of a complex object's behavior [58, 20, 15, 51, 71]. The main advantage of this approach is that the geometric and physical DOFs coincide, eliminating the non-sparsity issue of reduced spaces; one can reuse the exact same material simulator, but now on a coarse embedding mesh rather than on a fine, detailed geometry. However, methods of this last class focus almost exclusively on homogeneous materials. When a heterogeneous object is at play, the coarse mesh must now be assigned "averaged" material properties to best match the behavior of the original object. This issue was addressed in computer animation in [53, 52], where a spatial average of the elasticity tensor was proposed on cubical grids. Yet, such a simple average does not accurately coarsen an elastic material: this procedure in 1D amounts to averaging a set of springs in series by their mean stiffness, while the correct equivalent stiffness is the inverse of the sum of the reciprocal of each fine stiffness (this relation is easily derived using the well-known electro-mechanical analogy; unfortunately, no such result in 2D or 3D is known). The difference between these two coarsening approaches can be quite significant: consider a 1D system made out of a very small and very soft section sandwiched between two large, extremely rigid sections; the resulting system is obviously very soft (see Figure 4.2(left)), whereas a spatial average would have made it much stiffer. A proper numerical coarsening is thus crucial when approximating the dynamics of heterogenous objects on coarse grids.

4.2.2 Scalar Homogenization

Homogenization theory [7, 35] has been developed for this exact purpose of extracting information from fine scales to computational scales in order to perform efficient computations over composite, inhomogeneous materials (*i.e.*, with spatially varying physical properties, such as laminates, rebar-reinforced concrete, etc). In essence, this theory replaces the microscopic structure of a composite

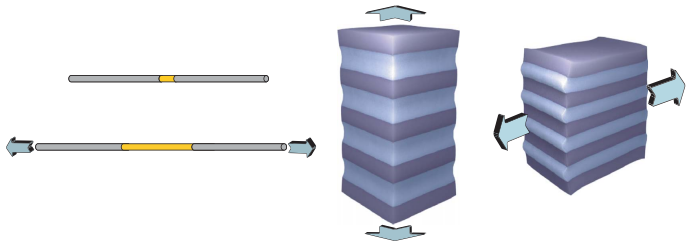


Figure 4.2: Inhomogeneous materials leads to Anisotropic Behavior: In 1D (left), even a tiny amount of soft material between two rigid rods renders the resulting bar highly deformable when pulled; a cube of composite material in 3D (right) made out of two materials (the blue one being softer than the mauve one) exhibits significant anisotropy due to its composition: in this case, it stretches much more vertically than horizontally.

by an idealized, locally-homogenous material with equivalent macroscopic physical properties—a procedure referred to as coarsening, homogenization, or upscaling. A variety of coarsening methods have been proposed, starting from well-known arithmetic and harmonic averages, to more involved ones like the renormalization method or the representative elementary volume; see [21] for a review. Unfortunately, the numerical homogenization techniques available so far can offer accurate results *only* if periodicity, ergodicity, or scale-separation assumptions on the material properties are satisfied, making them quite poorly adapted to our needs in computer animation.

Recently a homogenization technique of elliptic equations in divergence form requiring *no assumptions* on the material at hand was proposed in [56], providing a way to approximate a fine solution u of the static problem:

$$\begin{cases} \operatorname{div}(C(x)\nabla u) = f & \text{inside a domain } \mathbf{D} \\ u = u_0 & \text{on } \partial\mathbf{D} \end{cases} \tag{4.1}$$

with spatially-varying conductivity $C(x)$, by a coarse function u_h that provably satisfies:

$$\|u - u_h\|_{H^1} \leq Ch\|f\|_{L^2}.$$

This coarse function u_h is thus a good match (in the H^1 sense, thus even better in the L^2 sense) for the real (fine) solution, even if it requires a much cheaper numerical solve. While seemingly appropriate for elastostatics given the similarities between Eq. (4.1) and the balance equation in elasticity, this technique is limited to the scalar case (as well as other related recent methods, see [2, 63]). Additionally, general boundary conditions or non-conforming coarse meshes are neither discussed

nor tackled.

4.3 Coarsening Methodology of Linear Elasticity

We now present how we derive, from an elastic, inhomogeneous object, a coarse approximation that possesses a similar physical behavior. We assume that the procedure is performed in dimension d, where $d = 1, 2$, or 3 for generality. Additionally, for consistency and clarity, we use ROMAN characters to refer to quantities living on the fine mesh $\mathbf{D}$, and $\mathbb{BLACKBOARD}$ characters to refer to quantities on the coarse mesh $\mathbb{D}$.

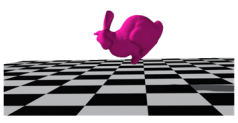
4.3.1 Problem Statement

We present a practical solution to the following numerical coarsening problem:

> *Given (a) a fine tetrahedral mesh* $\mathbf{D}$*, in which each tet* T_p *(*$p = 1 \ldots |\mathbf{D}|$*) has a different elasticity tensor* $\mathbf{C}_{T_p}$*, and (b) a coarse tetrahedral mesh* $\mathbb{D}$ *(of much smaller element count,* i.e.*,* $|\mathbb{D}| \ll |\mathbf{D}|$*) approximating the same geometry as* $\mathbf{D}$*, find an "effective" elasticity tensor* $\mathbb{C}_{\mathbb{T}_q}$ *per coarse tet* $\mathbb{T}_q$ *such that the overall dynamics obtained by an off-the-shelf elasticity simulator applied to either of them matches well.*

Our coarsening procedure is achieved by first computing on $\mathbf{D}$ a set of global harmonic displacements to analyze the heterogeneous fine-scale properties, then by deducing the effective coarse-scale property for each coarse mesh element. We show how our approach can be seen as a mollification of the displacement field to allow for proper averaging of the physical properties, and how it matches the simple, known case of elasticity in 1D. Note that even if the object described by the fine mesh is made out of different isotropic materials (*i.e.*, with the same stiffness independent of the directional orientation of the applied force), the resulting elasticity tensors at the coarse level are often *anisotropic* as they reflect the object's fine, inhomogeneous composition (see Figure 4.2).

Our approach contrasts with previous work in several ways. Unlike methods based on Krylov spaces (using various definitions of "eigen" deformations), our coarse model is not limited to a linear space of deformations, and does not involve reduced coordinates not matching the geometric description of the object. Instead, our coarse model is simulated with a traditional finite-element solver but on a coarser grid approximating the object's global geometry. The resulting dynamical system can thus be deformed arbitrarily with a computational complexity proportional to the size of the coarse mesh. Additionally, the accuracy of our approach decays gracefully with the maximum edge length



of the coarse mesh.

4.3.2 Coarsening Procedure Setup and Overview

We start from a fine mesh $\mathbf{D}$, in which each tet T_p ($p = 1 \ldots |\mathbf{D}|$) has a different elasticity tensor $\mathbf{C}_{\mathrm{T}_p}$ (*i.e.*, a different set of Lamé coefficients if we assume isotropy of each fine elements). We wish to approximate its dynamics on a given coarser mesh $\mathbb{D}$ (with $|\mathbb{D}| \ll |\mathbf{D}|$) that describes the same geometry. That is, we need to find, as a precomputation, an effective elasticity tensor $\mathbb{C}_{\mathbb{T}_q}$ on each coarse mesh element $\mathbb{T}_q$ so that the dynamics of the resulting coarse system closely matches the original fine object.

$$(\mathbf{D} = \{\mathrm{T}_p\}, \mathbf{C} = \{\mathrm{C}_{\mathrm{T}_p}\}) \xrightarrow{\text{coarsening}} (\mathbb{D} = \{\mathbb{T}_q\}, \mathbb{C} = \{\mathbb{C}_{\mathbb{T}_q}\})$$

Our approach first "probes" the fine material by computing $d(d+1)/2$ harmonic displacements to capture how the fine mesh behaves when forces are applied to the boundary of the material. This set of displacements will in turn be used to derive a coarsening procedure to enforce that the potential energy of the coarse mesh $(\mathbb{D}, \mathbb{C})$ exactly matches the integral of the potential of the fine mesh $(\mathbf{D}, \mathrm{C})$ within each coarse tet $\mathbb{T}_q$. This coarsening procedure can be seen as an extension of the upscaling procedure with discontinuous elements introduced in Section 1.3 of [56] for *scalar* equations.

4.3.3 Downsampling Fields

Downsampling a field from the fine mesh $\mathbf{D}$ to the coarse mesh $\mathbb{D}$ is easily achieved. Each coarse-mesh vertex position $\mathbb{X}_i$ is expressed as a linear combination (through barycentric coordinates) of the fine-mesh vertices $\mathbf{x}_i$ defining the fine tetrahedron in which $\mathbb{X}_i$ lies at rest. In other words, the vertex positions $\mathbf{x}_i$ of the fine mesh are first interpolated by linear finite elements on the fine mesh $\mathbf{D}$, then the coarse nodes are defined as samples of this linear reconstruction. Boundary nodes that lie outside of the fine domain $\mathbf{D}$ require special treatment: for these nodes that do not have a bounding fine tet, we find one (or more) fine element(s) closest to it and use barycentric extrapolation instead (*i.e.*, negative barycentric coordinates). More details of this procedure and special handling of boundaries will be discussed in Section 4.4.

Notice however that such a downsampling is accurate *only* if the field we downsample is sufficiently *smooth*. While displacements of the objects throughout an animation can be assumed to be fairly

smooth, this is far from true for a field like the elasticity tensor $\mathbf{C}$, as we assume the material to be inhomogeneous on fine scales.

4.3.4 Numerical Coarsening Rationale

While displacements are easily downsampled, the elasticity tensor and mass matrix require more care to enforce that the coarse dynamics closely approximates the fine dynamics.

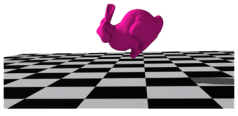
Potential Energy We first need to derive a tensor $\mathbb{C}_{\mathbb{T}_q}$ per coarse tet $\mathbb{T}_q$. In order to correctly reproduce the force field within the object, we should enforce that the potential energy of each coarse tet matches the integral of the potential energy over the fine tets contained within the coarse tet; *i.e.*, given our setup, we should target the following equality:

$$\int_{\mathbb{T}_q} \boldsymbol{\epsilon}(\mathbf{u}) : \mathbf{C} : \boldsymbol{\epsilon}(\mathbf{u})\, dV = \int_{\mathbb{T}_q} \boldsymbol{\epsilon}(\mathbb{U}) : \mathbb{C} : \boldsymbol{\epsilon}(\mathbb{U})\, dV$$

on each coarse tet $\mathbb{T}_q$ for *all* possible deformation fields $\mathbf{u}$. This is a tall order, as even if each (potentially anisotropic) coarse elasticity tensor has 21 degrees of freedom, the space of possible fine deformations is significant. Therefore the best we can hope to achieve is to perfectly capture this equality on a few displacements: if these displacements are characteristic of the typical deformations that the fine mesh can endure, we will have achieved our goals. As we will show below, we will introduce characteristic displacements $\mathbf{h}_{\alpha\beta}$ (with $1 \leq \alpha \leq \beta \leq d$, for a total of $d(d+1)/2$ displacement fields) and derive coarse elasticity tensors that enforce:

$$\int_{\mathbb{T}_q} \boldsymbol{\epsilon}(\mathbf{h}_{\alpha\beta}) : \mathbf{C} : \boldsymbol{\epsilon}(\mathbf{h}_{\delta\gamma})\, dV = \int_{\mathbb{T}_q} \boldsymbol{\epsilon}(\mathbb{H}_{\alpha\beta}) : \mathbb{C} : \boldsymbol{\epsilon}(\mathbb{H}_{\delta\gamma})\, dV \tag{4.2}$$

on each coarse tet $\mathbb{T}_q$, and for all $\alpha \leq \beta$ and $\delta \leq \gamma$ (note that $\mathbb{H}_{\alpha\beta}$ is the upscaled displacement based on $\mathbf{h}_{\alpha\beta}$ as defined in Section 4.3.3). This results in 21 independent equations in 3D after accounting for the major symmetry of $\mathbf{C}$, and their enforcement is equivalent to enforcing potential energy equality for all linear combinations of the test displacements. We will also demonstrate that our particular choice of characteristic displacements leads to a variational interpretation, giving another justification of our approach. Note that for clarity we will continue to use Greek letter indices for indexing among characteristic displacements, while Roman letter indices will still be used for coordinates and gradients.



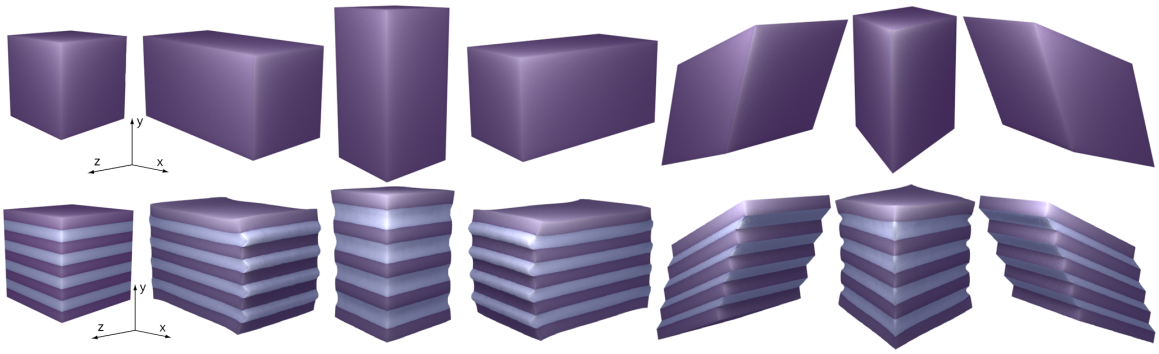


Figure 4.3: The six harmonic displacements obtained from a homogenous material (top), and a heterogeneous material made of layers of 2 different elastic materials (bottom). The deformations correspond to respectively: $\mathbf{h}_{11}, \mathbf{h}_{22}, \mathbf{h}_{33}, \mathbf{h}_{12}, \mathbf{h}_{23}$, and $\mathbf{h}_{13}$.

Kinetic Energy As one of our goals is to be able to reuse a conventional finite-element solver on the coarse model, we do not allow the coarse mass matrix to be anisotropic since most implementations that we are aware of in graphics assume a lumped, diagonal mass matrix. Therefore, we define the mass matrix $\mathbb{M}$ to be a diagonal matrix, for which the diagonal elements represent the usual lumped mass around each node of the coarse mesh. Note that to obtain a better coarsening of the mass matrix such that the kinetic energies match well:

$$\frac{1}{2}\int_{\mathbb{D}} \dot{\mathbb{U}}^T \cdot \mathbb{M} \cdot \dot{\mathbb{U}} \approx \frac{1}{2}\int_{\mathbf{D}} \dot{\mathbf{u}}^T \cdot \mathbf{M} \cdot \dot{\mathbf{u}},$$

our treatment of the potential energy could also be used–where now we need to compute "characteristic frequencies" as it involves time derivatives. We omit this treatment here because unless the mass density contrast in the object is significant, phase errors on the final behavior of the coarsened system are unlikely to be visually crucial to be worth the extra computational time required by non-lumped (non-diagonal) mass matrices.

4.3.5 Global Harmonic Displacements

We first compute a few defining displacements $\mathbf{h}_{\alpha\beta}$ to study how the fine mesh behaves under a set of chosen conditions. For our purposes of simulating elastic objects, we compute these displacements by solving the following set of static boundary value problems for $\{\mathbf{h}_{\alpha\beta}\}_{1\leq\alpha\leq\beta\leq d}$:

$$\begin{cases} \operatorname{div}\left(\mathbf{C} : \boldsymbol{\epsilon}(\mathbf{h}_{\alpha\beta})\right) = 0 & \text{inside } \Omega \\ \left(\mathbf{C} : \boldsymbol{\epsilon}(\mathbf{h}_{\alpha\beta})\right) \cdot \mathbf{n} = \boldsymbol{\epsilon}(x_\alpha \, \mathbf{e}_\beta) \cdot \mathbf{n} & \text{for } \mathbf{x} \in \partial\Omega, \end{cases} \tag{4.3}$$

where x_α denotes the α-th coordinate of space and $\mathbf{e}_\beta$ is the unit vector in the β-th coordinate direction. Note that we can thus rewrite $\boldsymbol{\epsilon}(x_\alpha\, \mathbf{e}_\beta)$ as $\frac{1}{2}(\mathbf{e}_\alpha \otimes \mathbf{e}_\beta + \mathbf{e}_\beta \otimes \mathbf{e}_\alpha)$. The harmonic displacements are computed by solving each system using linear elasticity setup from Section 2.2.2 and fixing the last six degrees of freedom (translation and rotation) of Eq. (4.3) by fixing the zero-th and first moments (also described in Secion 2.2.2), resulting in a unique solution. The reader may recognize the typical requirement of $\mathbf{C}-$harmonicity, along with Neumann boundary conditions prescribing surface tractions equal to $\boldsymbol{\epsilon}(x_\alpha\, \mathbf{e}_\beta) \cdot \mathbf{n}$. We will thus refer to this family of $d(d+1)/2$ static solutions as "global harmonic displacements" (see Figure 4.3 for basic examples on both homogeneous and inhomogeneous materials). These static solutions represent characteristic displacements resulting from a global "probing" of the object by a set of linear traction fields on the boundary. For notational simplicity, we will denote by $\mathbf{H}$ the rank-3 tensor whose components are the coordinates of every harmonic displacement $\mathbf{h}_{\alpha\beta}$, *i.e.*, $\mathbf{H}_{k\alpha\beta} = (\mathbf{h}_{\alpha\beta})_k$.We finally symmetrize $\mathbf{H}$ through $\mathbf{H}_{k\alpha\beta} = \mathbf{H}_{k\beta\alpha}$ for simplicity, as it avoids having the restriction $\alpha \leq \beta$ in further equations.

4.3.6 Harmonic Mollifier

The *symmetric part* of the gradient of the tensor $\mathbf{H}_{k\alpha\beta}$ will play a crucial role in coarsening. This rank-4 tensor $\mathbf{G}$ is defined as

$$\mathrm{G}_{kl\alpha\beta} = \frac{1}{2}\left(H_{k\alpha\beta,l} + H_{l\alpha\beta,k}\right).$$

Note that this last expression is a generalization of the symmetrized gradient operator $\boldsymbol{\epsilon}$ for rank-2 tensors, and therefore the resulting $\mathbf{G}$ has the minor symmetry $\mathrm{G}_{kl\alpha\beta} = \mathrm{G}_{lk\alpha\beta}$ as well as $\mathrm{G}_{kl\alpha\beta} = \mathrm{G}_{kl\beta\alpha}$ thanks to the symmetry of $\mathbf{H}$. Although of higher-order in our case, this tensor can be shown to help mollify solutions of the elastic equation just as [56] demonstrated in the scalar case of anisotropic Poisson equations: we also observe that for any displacement $\mathbf{u}$ of our fine object, the field $\mathbf{G}^{-1} : \boldsymbol{\epsilon}(\mathbf{u})$ becomes Hölder continuous, *i.e.*, quite smooth, though not necessarily Lipschitz. This will be particularly useful: this "mollified" field *can be approximated (for any reasonably smooth displacement field) on the coarse level without significant loss of information, by subsampling each term*:

$$\mathbf{G}^{-1} : \boldsymbol{\epsilon}(\mathbf{u}) \approx \mathbb{G}^{-1} : \boldsymbol{\epsilon}(\mathbb{U}) \tag{4.4}$$

where $\mathbb{G}$ is the coarse mesh analogue of $\mathbf{G}$. This property is crucial in getting accurate coarsening.

4.3.7 Homogenization of Fine Scales

We finally "downsample" the elasticity tensor $\mathbb{C}$ as follows, so as to preserve the symmetries of the coarse elasticity tensor mentioned in Section 2.2.2:

$$\mathbb{C}_{\mathbb{T}_q} := \mathbb{G}_{\mathbb{T}_q}^{-T} : \langle \mathbf{G}^T : \mathbf{C} : \mathbf{G} \rangle_{\mathbb{T}_q} : \mathbb{G}_{\mathbb{T}_q}^{-1}, \tag{4.5}$$

or rewritten using tensor notation,

$$\left[\mathbb{C}_{\mathbb{T}_q}\right]_{ijkl} := \left[\mathbb{G}_{\mathbb{T}_q}^{-T}\right]_{ij\alpha\beta} \left[\langle \mathbf{G}^T : \mathbf{C} : \mathbf{G} \rangle_{\mathbb{T}_q}\right]_{\alpha\beta\gamma\delta} \left[\mathbb{G}_{\mathbb{T}_q}^{-1}\right]_{\gamma\delta kl}.$$

This coarsening is achieved by first averaging quantities on the fine mesh $\mathbf{D}$ through:

$$\left[\langle \mathbf{G}^T : \mathbf{C} : \mathbf{G} \rangle_{\mathbb{T}_q}\right]_{\alpha\beta\gamma\delta} := \sum_{\substack{\mathrm{T}_p \in \mathbf{D} \\ \mathrm{T}_p \cap \mathbb{T}_q \neq \emptyset}} \frac{|\mathrm{T}_p \cap \mathbb{T}_q|}{|\mathbb{T}_q|} [\mathbf{G}_{\mathrm{T}_p}^T]_{\alpha\beta ij} [\mathrm{C}_{\mathrm{T}_p}]_{ijkl} [\mathbf{G}_{\mathrm{T}_p}]_{kl\gamma\delta},$$

then by computing the inverse of the tensor $\mathbb{G}$ on the coarse mesh. We stress that this inverse needs to be done with care: this is an inverse in the (reduced) space of tensors acting on *symmetric* tensors. However, as we will represent this tensor in the reduced space, this will be a standard 6x6 matrix inverse, the conversion of tensors to matrix form is described in Section 2.1.2.

This procedure for deriving an effective tensor not only respects the symmetries that any elastic tensor should have, but also satisfies Eq. (4.2).

4.3.8 Variational (Finite Element) Interpretation

Since the traditional finite-element variational treatment of elasticity considers the weaker form of the divergence term by pairing it with another arbitrary "test" deformation $\mathbf{z}$, we can now write

(discarding boundary terms for clarity):

$$\begin{aligned}\int_{\mathbb{D}} \operatorname{div}(\mathbf{C} : \boldsymbol{\epsilon}(\mathbf{u}))\, \mathbf{z} &= \int_{\mathbb{D}} \boldsymbol{\epsilon}(\mathbf{z}) : \mathbf{C} : \boldsymbol{\epsilon}(\mathbf{u}) \\ &= \int_{\mathbb{D}} \boldsymbol{\epsilon}(\mathbf{z}) : \mathbf{G}^{-T} : \mathbf{G}^{T} : \mathbf{C} : \mathbf{G} : \mathbf{G}^{-1} : \boldsymbol{\epsilon}(\mathbf{u}) \\ &\overset{*}{\approx} \int_{\mathbb{D}} \boldsymbol{\epsilon}(\mathbb{Z}) : \mathbb{G}^{-T} : \mathbf{G}^{T} : \mathbf{C} : \mathbf{G} : \mathbb{G}^{-1} : \boldsymbol{\epsilon}(\mathbb{U}) \\ &= \int_{\mathbb{D}} \boldsymbol{\epsilon}(\mathbb{Z}) : \mathbb{C} : \boldsymbol{\epsilon}(\mathbb{U}) = \int_{\mathbb{D}} \operatorname{div}(\mathbb{C} : \boldsymbol{\epsilon}(\mathbb{U}))\, \mathbb{Z}\end{aligned}$$

where the step marked by the asterisk is a consequence of the mollification property in Eq. (4.4) used on both $\mathbf{u}$ and $\mathbf{z}$. Therefore, our definition of the upscaled elasticity tensors can be seen as a particular choice of a test function $\mathbf{z}$ for which the upscaled test function $\mathbb{Z}$ is a linear basis function of the coarse mesh $\mathbb{D}$, so that *a typical linear finite-element treatment of the coarse mesh closely corresponds to a finite-element treatment of the fine mesh*.

4.3.9 Discussion

Our specific procedure to accurately downsample the elasticity tensor field of a fine elastic object can be understood either from the variational point of view (through mollification of the displacement, Section 4.3.8), or from the exact matching of the potential energy for a set of characteristic displacements. Other variants can also be derived, potentially at the cost of losing one of these two properties. For instance, characteristic functions satisfying an alternate set of boundary value problems could be chosen. In particular, the boundary conditions of the harmonic equation should be changed to mixed Dirichlet/Neumann conditions *if* some vertices are known to be always fixed during simulation: this will capture harmonic displacements that are more appropriate to this particular use. Other extensions could relax the exact enforcement of Eq. (4.2), and consider a least square solution for a larger family of carefully-tuned characteristic displacements instead if prior knowledge on the use of the coarse simulation is available. Also, computing *local* harmonic characteristic fields would become attractive if the mesh topology is allowed to change over time: we will leave this local approach for future work, as a careful study of the consequences of these multiple local solves versus a global solve for coarsening is delicate to perform, by lack of a proper metric to use for fair comparison.

With our proposed approach, we can piggyback on the analysis provided in [56] (with further details

in [8]) to conclude that our coarse simulation using $\mathbb{U}$ will satisfy:

$$\|\mathbf{u} - \mathbb{U}\|_{\mathcal{H}^1} \leq \alpha h \|\mathbf{f}\|_{\mathcal{L}^2}$$

(where h is the maximum size of a coarse element), thus by duality,

$$\|\mathbf{u} - \mathbb{U}\|_{\mathcal{L}^2} \leq C\alpha h^2 \|\mathbf{f}\|_{\mathcal{L}^2}.$$

In practice, this implies that the error made by the coarsening procedure is of the order of the size of the coarse mesh.

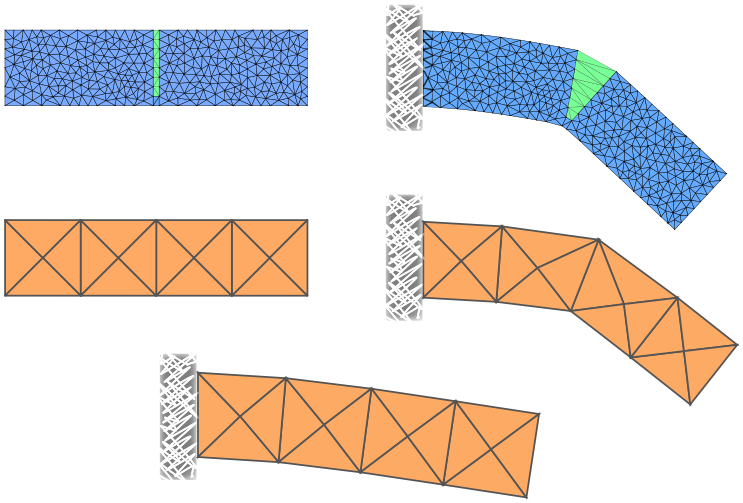


Figure 4.4: Coarsening of Cracks: (left) In this 2D example, coarsening is used to turn a bar-like object (blue) containing a thin slice of soft material (green) into a very coarse mesh (peach-colored mesh); (right) when deformed under gravity, both models present similar deformations; (bottom) a simple spatial averaging of the material elasticity coefficients or stiffness tensors does not capture this bending behavior, not accounting properly for the weak material in the middle.

4.4 Implementation Details

Although 3D numerical coarsening is mostly achieved by solving six harmonic displacement fields and a few linear algebra operations as we explained in the previous section, several components deserve more details.

4.4.1 Symmetric Tensor Representation

While our presentation has consistently used tensor notation, implementation can be done using 6D vectors to represent symmetric rank-2 (3x3) tensors, and 6x6 matrices to encode rank-4 tensors:

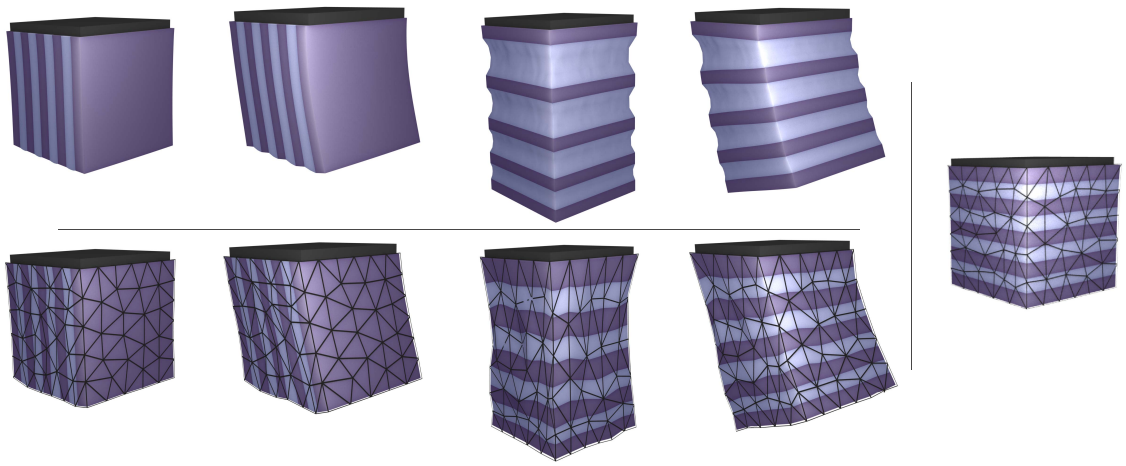


Figure 4.5: On the inhomogeneous layered cube used in Fig 4.3, a fine simulation (top, left) is well captured by our coarsening approach (bottom, left), despite the anisotropy of the object; if, however, the material coefficients (right, showing the most extreme extended position reached during the motion) or the stiffness matrices of the original object are simply averaged, coarse simulations do not match the fine behavior.

this memory-efficient representation often used in computer graphics exploits the symmetries of the tensor we have to deal with, as explained in Section 2.1.2. In this representation, double contractions can be performed by 6x6 matrix products, and the entity $\mathbb{G}^{-1}$ is exactly the inverse of the 6x6 matrix $\mathbb{G}$ used to encode the harmonic mollifier. Alternatively, one can implement coarsening by copying literally the formulae provided in our explanations using arrays and their indices—although $\mathbb{G}^{-1}$ will then require special care.

4.4.2 Boundary Treatment

As briefly mentioned earlier, we treat coarse nodes that are outside the fine domain through barycentric extrapolation, *i.e.*, when a field needs to be evaluated on this coarse node, we rely on the values of a few closest fine boundary nodes (between one and three in our implementation, depending on the local curvature of the object) to extrapolate the field based on the positions of the respective positions at rest. Additionally, we found it beneficial to alter the local definition of the normal $\mathbf{n}$ used in the Neumann condition of Eq. (4.3) to become the normal $\mathbf{N}$ of the *coarse bounding tet* instead. This change is more in line with the variational interpretation described in Section 4.3.8, as the test function near the boundary should be reverted to the coarse element basis function, hence improving boundary coarsening. Note that *only* the external boundary of the object should be subjected to traction: holes inside the domain must be left without traction to be treated as such when

computing harmonic displacements.

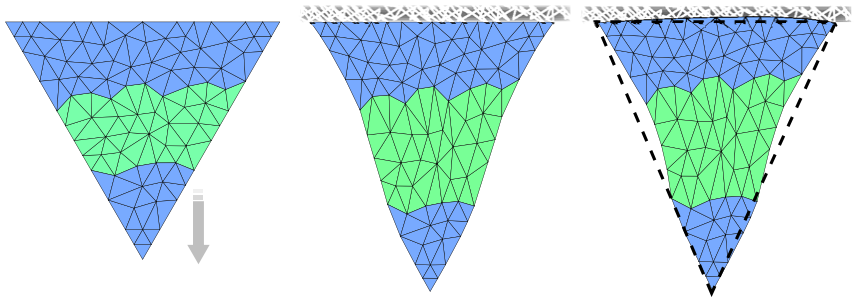


Figure 4.6: A 2-material composite object (left) is subjected to gravity with its top vertices fixed, resulting in a elongated deformation (middle). From a coarse mesh deformation made of a *single* triangle (right, dashed), we can reconstruct a quasi-static fine solution (right) using precomputed harmonic displacements: this cheap linear map from coarse to fine deformation enhances visual impact at low cost.

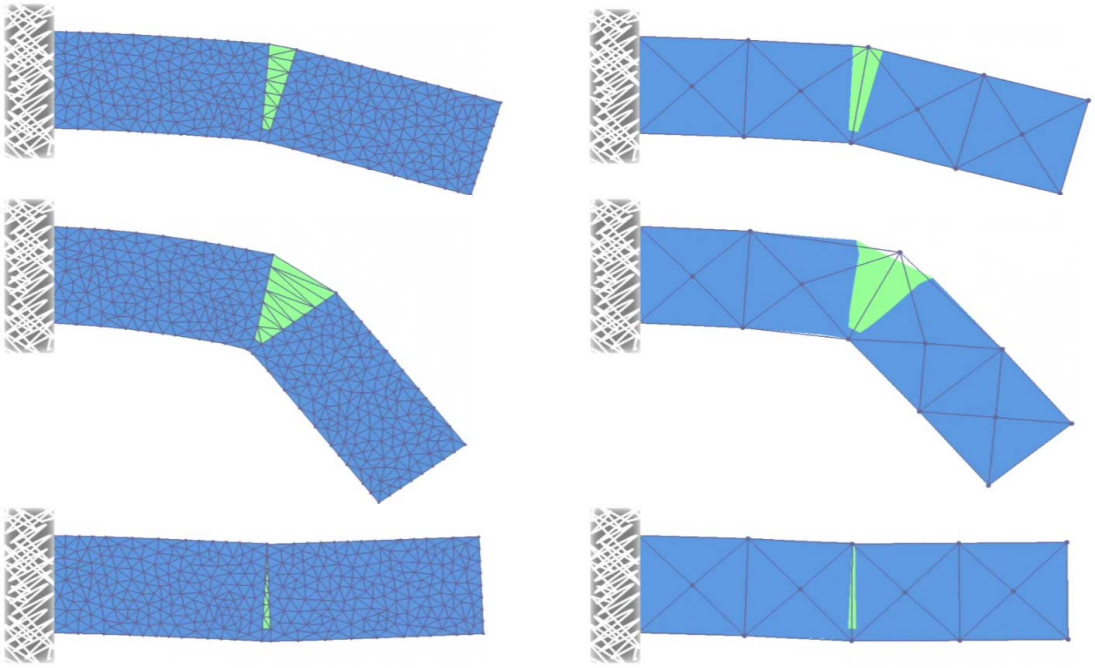


Figure 4.7: The same bar with a crack as in the Figure 4.4, now the fine mesh is interpolated using harmonic displacements over the coarse simulation (right), compare to the corresponding fine simulation on the left.

4.4.3 Coarse-to-Fine Mapping for Display

The tensors used for numerical coarsening can also be reused to deduce fine-mesh vertex positions deduced from coarse deformations. This coarse-to-fine interpolation is a quasi-static approximation, as it assumes equilibrium inside each coarse tetrahedron, so it is devoid of higher temporal frequencies of the fine mesh. However, it can be an effective way to reuse some of the information

gathered about the anisotropy of the object being simulated.

For each coarse tet $\mathbb{T}_q = \{\mathbf{v}^1, \mathbf{v}^2, \mathbf{v}^3, \mathbf{v}^4\}$ with undeformed vertex positions $\{\mathbf{x}_0^0, \mathbf{x}_0^1, \mathbf{x}_0^2, \mathbf{x}_0^3\}$ and current deformed positions $\{\mathbf{x}^0, \mathbf{x}^1, \mathbf{x}^2, \mathbf{x}^3\}$ we solve for a rotation matrix $\mathbf{R}$ and a symmetric matrix $\mathbf{S}$ such that

$$\begin{pmatrix} \mathbf{R}(\mathbf{x}^1 - \mathbf{x}^0) \\ \mathbf{R}(\mathbf{x}^2 - \mathbf{x}^0) \\ \mathbf{R}(\mathbf{x}^3 - \mathbf{x}^0) \end{pmatrix} = \begin{pmatrix} \mathbf{x}_0^1 - \mathbf{x}_0^0 \\ \mathbf{x}_0^2 - \mathbf{x}_0^0 \\ \mathbf{x}_0^3 - \mathbf{x}_0^0 \end{pmatrix} + \begin{pmatrix} (\mathbf{H}^1 - \mathbf{H}^0) : \mathbf{S} \\ (\mathbf{H}^2 - \mathbf{H}^0) : \mathbf{S} \\ (\mathbf{H}^3 - \mathbf{H}^0) : \mathbf{S} \end{pmatrix} \tag{4.6}$$

where $\mathbf{H}^i$ is the component of $\mathbf{H}$ (defined in Section 4.3.5) corresponding to the vertex $\mathbf{v}^i$. The interpolated current deformed position for any fine vertex $\mathbf{x}$ inside this tet can then be computed as

$$\mathbf{x} = \mathbf{x}^0 + \mathbf{R}^T(\mathbf{x}_0 - \mathbf{x}_0^0 + (\mathbf{H}^{\mathbf{x}} - \mathbf{H}^0) : \mathbf{S}) \tag{4.7}$$

where $\mathbf{H}^{\mathbf{x}}$ is the component of $\mathbf{H}$ for $\mathbf{x}$. This amounts to finding a rotation and linear combination of the harmonic displacements which matches the coarse tet vertices exactly, and then using this same rotation and linear combination to place the fine vertices. Blending of the displacements across adjacent coarse tets can also be added to avoid derivative discontinuities. As this interpolation relies on linear elasticity, it may not not be appropriate for large coarse deformations, and a robust solution based on projection onto the shape space spanned by harmonic displacements is left as future work. Figures 4.6 and 4.7 show how this coarse-to-fine map behaves on 2D examples.

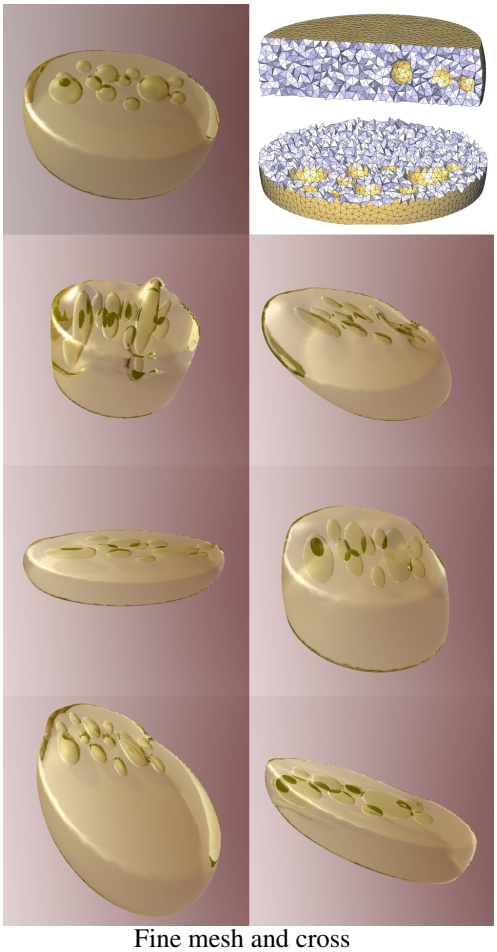
4.5 Results

In order to demonstrate the efficacy of numerical coarsening, we tested our approach on models of varying size, shape, and material composition. A first sanity check was to test that a homogenous object is coarsened into the same material—for coarse tets entirely inside the model. We then tried a layered object (Figures 4.2(right), 4.3(bottom), and 4.5) made out of two distinct materials. As expected, we witness an “accordion" effect when the object is deformed perpendicular to its layers, while lateral deformation are much less pronounced. This example is simple yet anisotropic enough to convincingly prove that other forms of coarsening (average of stiffness matrices, or of material coefficients) are just not enough to capture the proper dynamics on the coarse mesh.

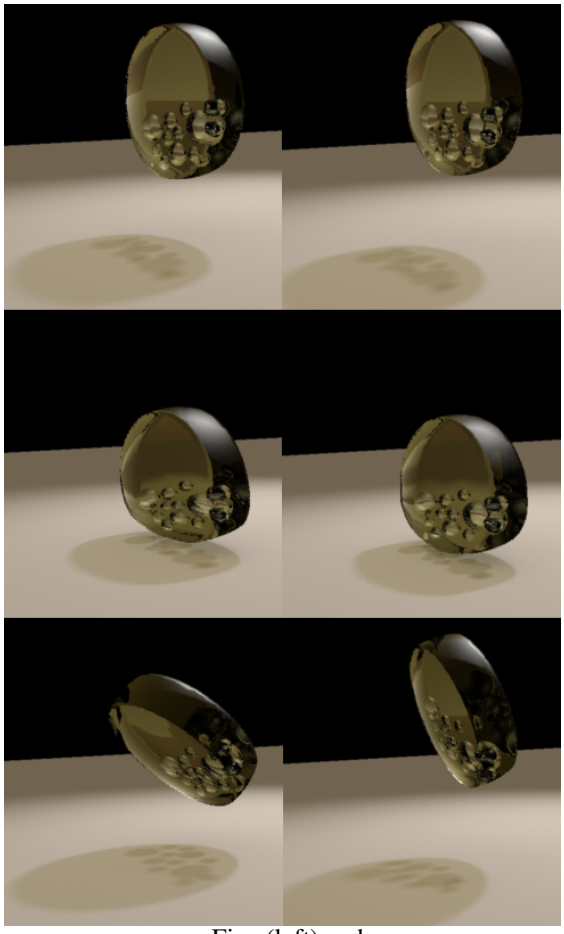
We also tested more subtle geometric details that can significantly affect the dynamics. In particular, Figures 4.4 and 4.7 show that a fine, but deep crack is properly taken into account, resulting in a

coarsened motion exhibiting much larger deformation due to the local “weakening" of the material. In Figure 4.8, we demonstrate the coarsening of a cheese wheel model, with half of the wheel containing gruyere-like holes. As the harmonic deformations clearly exhibit, the coarsened material properties are significantly affected by the inhomogeneity of the model. We also compare the motion of the coarsened model (200 tets) to the much finer original tetrahedral mesh (35K tets needed in order to represent the holes) of the wheel, indicating good visual agreement while reducing the computational complexity: the coarse animation runs 150 times faster than the fine one. Finally, we applied our numerical coarsening technique to a medical model, consisting of a liver and its two interior veins (portal vein, and inferior vena cava; see Figure 4.1). The veins act as reinforcement, rendering the liver stiffer. We started from a MRI dataset made out of 200K tets tagged as either belonging to the liver, or one of the two veins. After assigning material properties to these three components, we numerically coarsened this model to obtain an anisotropic coarse material made out of 210 vertices behaving dynamically similar to the original model.

Limitations. It should be reemphasized that our coarsening is currently limited to linear elasticity. The use of corotational methods injects geometric nonlinearity to coarse simulations, thus limiting the visual drawbacks of linear elasticity. However, the same use of corotated elements for fine meshes will add non-linear details that low tet-count meshes will be unable to match, even after proper coarsening. An extension to non-linear coarsening is thus desirable.

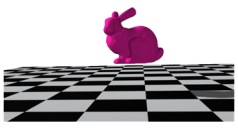


Fine mesh and cross sections (35K tets, top), along with its harmonic deformations.



Fine (left) and Coarse (right) animations. The coarse mesh has only 200 tets.

Figure 4.8: Elastic properties of a wheel of cheese with holes of various sizes in half of the wheel are turned into anisotropic elastic properties on a *coarse* mesh (200 tets). Animating the coarse object (right) takes only a fraction (~1/150) of the cost it would take to compute the elastic behavior on the fine mesh. Notice that the side containing the holes behaves softer, even though the coarse mesh does not spatially capture these cavities.



Chapter 5

Conclusions

Structure-preserving approaches to time integration of Lagrangian systems and a method to upscale heterogeneous elastic material properties have been presented in this thesis.

The design of time integrators has received little attention in the graphics community despite their widespread use. Given the importance of qualitatively correct behavior in computer animation, the geometric view is particularly pertinent as it ensures conservation of important quantities, even for lower accuracy/higher speed simulations. An approach to derive general purpose, fully variational time integrators for a wide class of mechanical systems using a discrete Hamilton- Pontryagin principle has been developed. One of the innovative aspects of this work is the introduction of the variational integrability condition that allows to solve the non-linear problem at each time step (when using implicit integration) through a minimization procedure instead of computation-intensive multidimensional root-finding. Together with the use of velocity, momentum, and position variables it promises to play an important role in motion control.

Two different approaches to time adaption, *i.e.*, automatic adjustment of timestep size during the simulation, have been presented. The variational approach enforces the particular timestep size using Lagrange multipliers inside the action integral, yielding a symplectic integrator with variable timesteps, including the cases when external forces are present. The symmetric approach utilizes the time-reversibility property of continuous dynamical systems, yielding a time adaptive time-reversible integrator. Our test showed that the latter integrator leads to more efficient time updates, as the system solved is less stiff. However, both methods appeared to be less efficient than integration with a constant timestep, thus, the problem of developing efficient and robust time adaptive

approaches remains open.

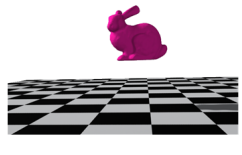
A formulation of elasticity modeling on coarse resolutions was proposed, where the influence of known fine scales is modeled through the derivation of effective anisotropic elasticity tensors. The effective tensors are obtained by solving inhomogeneous Laplace equations with Neumann boundary conditions, leading to non-trivial, symmetry-respecting averaging of the material properties. Directions of future work are manifold. An extension for which coarsening is performed through a series of local computations over each coarse tetrahedron would be highly desirable, as this would allow for fast updates of fine structures, during tearing for example. Geometric approaches to reconstruct fine deformations from the coarse ones would be valuable complements in order to further improve visual impact at low cost. Using non-diagonal mass matrices on the coarse mesh would also most likely bring benefits, albeit at a non-negligible computational cost.

5.1 Future Work

While this work has lead to a number of contributions, there are still many simulation problems that could benefit from more rigorous discretizations of space and time. In this section, potential avenues of future research will be discussed, which may directly or indirectly follow the work heretofore completed.

Model Reduction. Model reduction is an essential tool for achieving real-time simulations of complex models or systems. It consists of finding a coarse model (or coarse variables) that can reproduce the most critical aspects of the behavior of a complex dynamical system. Improving upon our material coarsening method (see Chapter 4 and [39]), an analogous method would be desired to allow for the coarsening of non-linear heterogeneous elastic materials to better handle large deformations. In the same vein, another interesting direction for research is coarsening of shells. This task involves finding a few coarse variables that can efficiently capture the overall motion of a very complicated shell model. Such a problem is very non-linear, since coarse variables may not only depend on the material parameters of the fine model, but also on the shape (and curvature) of the original fine shell.

Variational Approach to Fracture. In computer animation, elastic or rigid bodies involved in a simulation often undergo many complex changes, such as plastic deformations, fracture, or both of them combined (ductile fracture). Geometric models of such effects can be developed and combined



with the viscoelastic dynamics model described in this thesis. For example, crack propagation through inhomogeneous media can be formulated as a variational problem [12], thus geometric discretization techniques can be applied in order to preserve the underlying geometric structures in the discrete model.

Control of Simulation. For many applications in computer graphics and robotics, direct simulation of a physical system with given initial conditions is not sufficient. One may instead want to control the system to reach predetermined states (often at given times) such that the least amount of external force or internal power (fuel, steering, or wheel turning) is applied. Since this problem often involves solving large systems of equations for space-time degrees of freedom, a good reduced model of the system and a careful time discretization are both necessary for creating efficient control algorithms. The work presented in this thesis can be particularly useful for such applications.

Bibliography

[1] AN, S., KIM, T., AND JAMES, D. L. Optimizing cubature for efficient integration of subspace deformations. *ACM Transactions on Graphics (SIGGRAPH Asia) 27*, 4 (Dec. 2008).

[2] BABUŠKA, I., AND SAUTER, S. A. Efficient solution of anisotropic lattice equations by the recovery method. *SIAM J. Sci. Comput. 30*, 5 (2008), 2386–2404.

[3] BARAFF, D. Analytical methods for dynamic simulation of non-penetrating rigid bodies. In *ACM Trans. on Graphics, SIGGRAPH '89* (1989), ACM, pp. 223–232.

[4] BARAFF, D., AND WITKIN, A. P. Large steps in cloth simulation. In *ACM SIGGRAPH Proceedings* (July 1998), pp. 43–54.

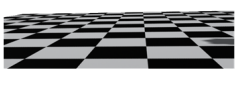
[5] BARBIČ, J., AND JAMES, D. Real-time subspace integration for St. Venant-Kirchhoff deformable models. *ACM Trans. on Graphics 24*, 3 (Aug. 2005), 982–990.

[6] BARR, A. H. The Einstein summation notation: introduction and extensions. ACM SIGGRAPH Course Notes #30 *"Topics in Physically-based Modeling"*, 1989, pp. J1–J12.

[7] BENSOUSSAN, A., LIONS, J. L., AND PAPANICOLAOU, G. *Asymptotic analysis for periodic structure*. North Holland, Amsterdam, 1978.

[8] BERLYAND, L., AND OWHADI, H. Finite dimensional approximation of solutions of divergence form systems of equations with rough and high contrast coefficients. *To appear* (2009).

[9] BLANES, S., AND BUDD, C. Explicit adaptive symplectic (EASY) integrators: a scaling invariant generalisation of the Levi-Civita and KS regularizations. *Celestial Mechanics and Dynamical Astronomy 89*, 4 (2004), 383–405.



[10] BONET, J., AND BURTON, A. A simple average nodal pressure tetrahedral element for incompressible and nearly incompressible dynamic explicit applications. *Comm. in Num. Meth. in Eng. 14*, 5 (1998), 437–449.

[11] BOU-RABEE, N., AND MARSDEN, J. E. HamiltonŰpontryagin integrators on lie groups part i: Introduction and structure-preserving properties. *Found. Comput. Math. 9*, 2 (2009), 197–219.

[12] BOURDIN, B., FRANCFORT, G. A., AND MARIGO, J.-J. The variational approach to fracture. *Journal of Elasticity 91*, 1-3 (Mar. 2008), 5–148.

[13] BRIDSON, R., FEDKIW, R., AND ANDERSON, J. Robust treatment of collisions, contact and friction for cloth animation. In *ACM Trans. on Graphics, SIGGRAPH '02* (2002), ACM, pp. 594–603.

[14] CALVO, M. P., LOPEZ-MARCOS, M. A., AND SANZ-SERNA, J. M. Variable step implementation of geometric integrators. *Applied Numerical Mathematicsg 28*, 1 (1998), 1–16.

[15] CAPELL, S., GREEN, S., CURLESS, B., DUCHAMP, T., AND POPOVIĆ, Z. Interactive skeleton-driven dynamic deformations. *ACM SIGGRAPH 21*, 3 (July 2002), 586–593.

[16] CAPELL, S., GREEN, S., CURLESS, B., DUCHAMP, T., AND POPOVIĆ, Z. A multiresolution framework for dynamic deformations. In *Symposium on Computer Animation* (July 2002), pp. 41–48.

[17] CHOI, M.-G., AND KO, H.-S. Modal warping: real-time simulation of large rotational deformation. *IEEE Trans. on Visualization and Computer Graphics 11*, 1 (2005), 91–101.

[18] CIARLET, P. *Mathematical elasticity, Volume I: three-dimensional elasticity*. North-Holland, 1988.

[19] DE VEUBEKE, B. F. The dynamics of flexible bodies. *International Journal of Engineering Science 14* (1976), 895–913.

[20] DEBUNNE, G., DESBRUN, M., CANI, M.-P., AND BARR, A. H. Dynamic real-time deformations using space & time adaptive sampling. In *Proceedings of ACM SIGGRAPH* (Aug. 2001), pp. 31–36.

[21] FARMER, C. L. Upscaling: A review. In *International Journal for Numerical Methods in Fluids* (2002), vol. 40, pp. 63–78.

[22] FETECAU, R. C. *Variational methods for nonsmooth mechanics*. PhD thesis, Caltech, April 2003.

[23] FETECAU, R. C., MARSDEN, J. E., ORTIZ, M., AND WEST, M. Nonsmooth lagrangian mechanics and variational collision integrators. *SIAM Journal on Applied Dynamical Systems 2* (2003), 381–416.

[24] FEYNMAN, R., LEIGHTON, R., AND SANDS, M. *The Feynman lectures on physics*. Addison-Wesley, 2006.

[25] GEORGII, J., AND WESTERMANN, R. Corotated finite elements made fast and stable. In *Proceedings of the 5th Workshop on Virtual Reality Interaction and Physical Simulation* (2008).

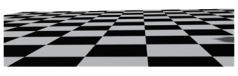
[26] GRINSPUN, E., KRYSL, P., AND SCHRÖDER, P. Charms: A simple framework for adaptive simulation. *ACM Transactions on Graphics 21*, 3 (July 2002), 281–290.

[27] HAIRER, E., LUBICH, C., AND WANNER, G. *Geometric numerical integration: structure-preserving algorithms for ODEs*. Springer, 2002.

[28] HAIRER, E., AND SÖDERLIND, G. Explicit, time reversible, adaptive step size control. *SIAM J. Sci. Comput. 26*, 6 (2005), 1838–1851.

[29] HARMON, D., VOUGA, E., SMITH, B., TAMSTORF, R., AND GRINSPUN, E. Asynchronous contact mechanics. *ACM Trans. Graph. 28*, 3 (2009).

[30] HARMON, D., VOUGA, E., TAMSTORF, R., AND GRINSPUN, E. Robust treatment of simultaneous collisions. In *ACM Trans. on Graphics, SIGGRAPH '08* (2008), ACM, pp. 1–4.



[31] HAUTH, M., AND STRASSER, W. Corotational simulation of deformable solids. In *Winter School on Computer Graphics* (Feb. 2004), pp. 137–144.

[32] JAMES, D. L., AND FATAHALIAN, K. Precomputing interactive dynamic deformable scenes. *ACM Transactions on Graphics 22*, 3 (July 2003), 879–887.

[33] JAMES, D. L., AND PAI, D. K. ARTDEFO: Accurate real time deformable objects. In *ACM SIGGRAPH Proceedings* (Aug. 1999), pp. 65–72.

[34] JAMES, D. L., AND PAI, D. K. DyRT: Dynamic response textures for real time deformation simulation with graphics hardware. *ACM Trans. on Graphics 21*, 3 (July 2002), 582–585.

[35] JIKOV, V. V., KOZLOV, S. M., AND OLEINIK, O. A. *Homogenization of differential operators and integral functionals*. Springer-Verlag, 1991.

[36] KANE, C., MARSDEN, J. E., ORTIZ, M., AND WEST, M. Variational integrators and the Newmark algorithm for conservative and dissipative mechanical systems. *I.J.N.M.E. 49* (2000), 1295–1325.

[37] KAUFMAN, D. M., EDMUNDS, T., AND PAI, D. K. Fast frictional dynamics for rigid bodies. In *ACM Trans. on Graphics, SIGGRAPH '05* (2005), ACM, pp. 946–956.

[38] KAUFMAN, D. M., SUEDA, S., JAMES, D. L., AND PAI, D. K. Staggered projections for frictional contact in multibody systems. In *ACM SIGGRAPH Asia 2008 papers* (2008), ACM, pp. 1–11.

[39] KHAREVYCH, L., MULLEN, P., OWHADI, H., AND DESBRUN, M. Numerical coarsening of inhomogeneous elastic materials. *ACM Trans. on Graphics 28*, 3 (2009).

[40] KOBILAROV, M., CRANE, K., AND DESBRUN, M. Lie group integrators for animation and control of vehicles. *ACM Trans. Graph. 28*, 2 (2009), 1–14.

[41] KRYSL, P., LALL, S., AND MARSDEN, J. Dimensional model reduction in non-linear finite element dynamics of solids and structures. *I.J.N.M.E. 51* (2000), 479–504.

[42] LALL, S., AND WEST, M. Discrete variational Hamiltonian mechanics. *J. Phys. A: Math. Gen. 39* (2006), 5509–5519.

[43] LANDAU, L., AND LIFSHITZ, E. *Theory of elasticity*. Oxford, England: Butterworth Heinemann, 1986.

[44] LEW, A. *Variational time integrators in computational solid mechanics*. PhD thesis, Caltech, May 2003.

[45] LI, R.-C., AND BAI, Z. Structure preserving model reduction using a Krylov subspace projection formulation. *Comm. Math. Sci. 3*, 2 (2005), 179–199.

[46] MARSDEN, J. E., AND HUGHES, T. J. R. *Mathematical foundations of elasticity*. Prentice-Hall, 1983.

[47] MARSDEN, J. E., AND WEST, M. Discrete mechanics and variational integrators. *Acta Numerica* (2001), 357–515.

[48] MARTIN, S., KAUFMANN, P., BOTSCH, M., WICKE, M., AND GROSS, M. Polyhedral finite elements using harmonic basis functions. *Computer Graphics Forum 27*, 5 (2008), 1521–1529.

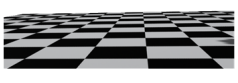
[49] MIRTICH, B., AND CANNY, J. F. Impulse-based dynamic simulation. Tech. rep., 1994.

[50] MÜLLER, M., DORSEY, J., MCMILLAN, L., JAGNOW, R., AND CUTLER, B. Stable real-time deformations. In *Proceedings of the Symposium on Computer Animation* (2002), pp. 49–54.

[51] MÜLLER, M., AND GROSS, M. Interactive virtual materials. In *Proceedings of Graphics Interface* (2004), pp. 239–246.

[52] NESME, M., KRY, P. G., JEŘÁBKOVÁ, L., AND FAURE, F. Preserving topology and elasticity for embedded deformable models. *ACM Trans. on Graphics 28*, 3 (Aug. 2009).

[53] NESME, M., PAYAN, Y., AND FAURE, F. Animating shapes at arbitrary resolution with non-uniform stiffness. In *Eurographics Workshop in Virtual Reality Interaction and Physical Simulation (VRIPHYS)* (Nov 2006).



[54] NOCEDAL, J., AND WRIGHT, S. J. *Numerical optimization*. Series in Operations Research. Springer, 1999.

[55] OGDEN, R. W. *Non-linear elastic deformations*. Dover Publications, 1997.

[56] OWHADI, H., AND ZHANG, L. Metric-based upscaling. *Communications on Pure and Applied Math. 60* (2007), 675–723.

[57] PARENT, R. *Computer animation: algorithms and techniques*. Morgan Kaufmann, 2001.

[58] PENTLAND, A., AND WILLIAMS, J. Good vibrations: modal dynamics for graphics and animation. In *ACM SIGGRAPH Proceedings* (1989), pp. 215–222.

[59] PETERSEN, K. B., AND PEDERSEN, M. S. The matrix cookbook, oct 2008. Version 20081110.

[60] PRESS, W. H., FLANNERY, B. P., TEUKOLSKY, S. A., AND VETTERLING, W. T. *Numerical recipes in C: the art of scientific computing*, 2nd ed. Cambridge University Press, 1992.

[61] RADOVITZKY, R., AND ORTIZ, M. Error estimation and adaptive meshing in strongly nonlinear dynamic problems. *Comput. Method. Appl. M 172*, 1-4 (1999), 203–240.

[62] RIVERS, A. R., AND JAMES, D. L. Fastlsm: Fast lattice shape matching for robust real-time deformation. *ACM Trans. on Graphics 26*, 3 (July 2007), 82:1–82:6.

[63] SHU, S., BABUŠKA, I., XIAO, Y., XU, J., AND ZIKATANOV, L. Multilevel preconditioning methods for discrete models of lattice block materials. *SIAM Journal on Scientific Computing 31*, 1 (2008), 687–707.

[64] SIMO, J. C., AND TARNOW, N. The discrete energy-momentum method: conserving algorithms for nonlinear elastodynamics. *Z. Angew. Math. Phys. 43*, 5 (1992), 757–792.

[65] SKEEL, R. D. Variable step size destabilizes the Störmer/leapfrog/Verlet method. *BIT Numerical Mathematics 33*, 1 (1993), 172–175.

[66] STEIN, A., AND DESBRUN, M. Discrete geometric mechanics for variational time integrators. In *Discrete Differential Geometry*. ACM SIGGRAPH Course Notes, 2006.

[67] TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISCHER, K. Elastically deformable models. In *ACM Trans. on Graphics* (1987), vol. 21, ACM Press, pp. 205–214.

[68] TOURNOIS, J., WORMSER, C., ALLIEZ, P., AND DESBRUN, M. Interleaving delaunay refinement and optimization for practical isotropic tetrahedron mesh generation. *ACM Trans. Graph. 28*, 3 (2009).

[69] TREUILLE, A., LEWIS, A., AND POPOVIĆ, Z. Model reduction for real-time fluids. *ACM Trans. on Graphics 25*, 3 (July 2006), 826–834.

[70] WEST, M. *Variational integrators*. PhD thesis, Caltech, June 2003.

[71] WOJTAN, C., AND TURK, G. Fast viscoelastic behavior with thin features. *ACM Trans. on Graphics 27(3)*, 47 (Aug. 2008).

[72] YOSHIMURA, H., AND MARSDEN, J. E. Dirac structures and Lagrangian mechanics. *J. Geom. and Physics* (2006).