Integrators for Elastodynamic Simulation with Stiffness and Stiffening

by

Yu Ju Chen

BASc, University of British Columbia, 2014

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES
(Computer Science)

The University of British Columbia
(Vancouver)

October 2019

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, the thesis entitled:

Integrators for Elastodynamic Simulation with Stiffness and Stiffening

submitted by **Yu Ju Chen** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy** in **Computer Science**.

Examining Committee:

Uri Ascher, Computer Science
Supervisor

Dinesh Pai, Computer Science
Supervisor

Chen Greif, Computer Science
Supervisory Committee Member

Brian Wetton, Mathematics
University Examiner

Eldad Haber, Earth, Ocean and Atmospheric Sciences
University Examiner

Abstract

The main goal of this thesis is to develop effective numerical algorithms for stiff elastodynamic simulation, a key procedure in computer graphics applications. To enable such simulations, the governing differential system is discretized in 3D space using a finite element method (FEM) and then integrated forward in discrete time steps. To perform such simulations at a low cost, coarse spatial discretization and large time steps are desirable. However, using a coarse spatial mesh can introduce numerical stiffening that impede visual accuracy. Moreover, to enable large time steps while maintaining stability, the semi-implicit backward Euler method (SI) is often used; but this method causes uncontrolled damping and makes simulation appear less lively.

To improve the dynamic consistency and accuracy as the spatial mesh resolution is coarsened, we propose and demonstrate, for both linear and nonlinear force models, a new method called EigenFit. This method applies a partial spectral decomposition, solving a generalized eigenvalue problem in the leading mode subspace and then replacing the first several eigenvalues of the coarse mesh by those of the fine one at rest. We show its efficacy on a number of objects with both homogeneous and heterogeneous material distribution.

To develop efficient time integrators, we first demonstrate that an exponential Rosenbrock-Euler (ERE) integrator can avoid excessive numerical damping while being relatively inexpensive to apply for moderately stiff elastic material. This holds even in challenging circumstances involving non-convex elastic energies. Finally, we design a hybrid, semi-implicit exponential integrator, SIERE, that allows SI and ERE to each perform what they are good at. To achieve this we apply ERE in a small subspace constructed from the leading modes in the partial spectral

decomposition, and the remaining system is handled (i.e., effectively damped out) by SI. We show that the resulting method maintains stability and produces lively simulations at a low cost, regardless of the stiffness parameter used.

Lay Summary

We have all seen objects deform elastically under forces in animated films. To make such movements appear natural and physical, we need to model and simulate the underlying physics using computer algorithms. In many applications, it is desirable to have a simplified system to enable fast computation. However, existing algorithms can introduce unwanted artifacts if such simplification is applied without care. In this work, we investigate these fundamental challenges, and propose algorithms to improve simulation efficiency.

Preface

This thesis describes results in three research articles:

- Y. J. Chen, U. M. Ascher, D. K. Pai, (2018). Exponential rosenbrock-euler integrators for elastodynamic simulation. IEEE transactions on visualization and computer graphics, 24(10), 2702-2713.
- Y. J. Chen, D. I. W. Levin, D. M. Kaufman, U. M. Ascher, and D. K. Pai, (2019). EigenFit for consistent elastodynamic simulation across mesh resolution. Proceedings of the 18th annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation (p. 5). ACM.
- Y. J. Chen, U. M. Ascher, D. K. Pai, (2019). SIERE: a hybrid semi-implicit exponential integrator for efficiently simulating stiff deformable objects. Submitted and under review.

Paper 1 has been published in Proceedings of the 18th annual ACM SIGGRAPH/Eurographics SCA 2019 and is described in Chapter 3. Paper 2 has been published in IEEE TVCG 2018 and is presented in Chapter 4. Paper 3 has been submitted and is currently under review. In all three research projects, I was the sole junior researcher among my co-authors and implemented all the numerical algorithms derived. I wrote the majority part of the all papers and received advices for and help for editting from all co-authors. In all three projects I received guidance from my research supervisors Uri Ascher and Dinesh Pai. In Paper 2 I received coding advice from David Levin and Danny Kaufman during my internship at Adobe. This work includes numerical software written by me, which is built on top of the C++ simulation library GAUSS[49].

Table of Contents

List of Tables

List of Figures

Acknowledgments

First I would like to thank my supervisors, Uri Ascher and Dinesh Pai. Your guidance has supported me throughtout my degree, and learning from you is a pleasant and humbling experience. I would also like to thank Danny Kaufman and David Levin for providing guidance during my internship at Adobe. To my supervisory committee - thank you for your time and feedback. I am grateful for many friends who have supported me during this journey. I would like to say a special thank you to Darcy Harrison for teaching me math during my early years, and Ye Fan, Egor Larionov for teaching me various simulation related topics. Finally, I would like to thank my family, especially my wife, for their patience and loving support.

Chapter 1

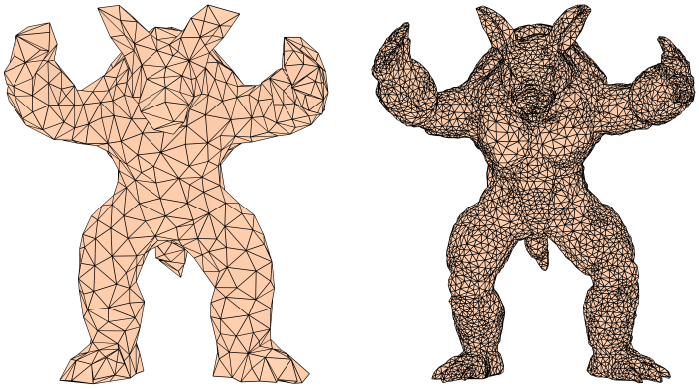
Introduction

Elastodynamic simulation, the process of simulating the motion of soft deforming objects is ubiquitous in computer graphics, robotics, biomechanics, and digital fabrication design. Examples include simulating the dynamics of cloth, skin, and other soft tissues of the human body. A fundamental computational bottleneck in all of these domains is the efficient forward simulation of elastodynamics. To simulate elastodynamics, researchers often first discretize the governing differential system in space employing a finite element method (FEM) [10, 78], and then in time using a time-stepping method.

For simulation-based applications there are many reasons to alter the spatial and temporal resolution of the computational mesh. Critically, coarsening in both space and time is often required in order to make applications practical. Most notably, runtime costs for soft-body simulations scale super-linearly in the number of nodes in the FE mesh. However, both spatial and temporal coarsening introduce undesirable and often unexpected numerical artifacts that reduce the controllability, consistency, accuracy and effectiveness of resulting simulations.

1.1 Dynamics simulation

Before diving into elastodynamic simulation, we first introduce general dynamic simulation in computer graphics. More detail will be described in Chapter 2. First, we write the positional degrees of freedom on a mesh (Figure 1.1), as single vector



(a) (b)

Figure 1.1: Coarse and fine meshes. Each mesh point has associated a position and velocity. Elastodynamics simulation produces elastic waves on these meshes with modeled forces.

$\mathbf{q}$, the corresponding velocity as $\mathbf{v}$, the inertia for $\mathbf{q}$ is written as a mass matrix $\mathbf{M}$, and the modeled force vector in the system is $\mathbf{f}(\mathbf{q})$. The equations of motion form a system of second-order ordinary differential equations (ODEs):

$$\mathbf{M}\ddot{\mathbf{q}} = \mathbf{f}(\mathbf{q}) \tag{1.1}$$

To simulate dynamics we first reduce Eq.(1.1) to first-order and solve the ODE system

$$\dot{\mathbf{u}}(t) = \begin{bmatrix} \dot{\mathbf{q}} \\ \dot{\mathbf{v}} \end{bmatrix} = \begin{bmatrix} \mathbf{v} \\ \mathbf{M}^{-1}\mathbf{f}(\mathbf{q}) \end{bmatrix} \stackrel{\text{def}}{=} \mathbf{b}(\mathbf{u}(\mathbf{t})) \tag{1.2}$$

The speed of a simulation refers to how much wall-clock time is required for computing the trajectory of states $\mathbf{q}(t)$, and velocities $\mathbf{v}(t)$ for a given time interval $t_{start} \leq t \leq t_{end}$. To achieve fast simulations, we can reduce the degrees of freedom by choosing a coarse mesh (Figure 1.1(a)) instead of a fine mesh (Figure 1.1(b)), step the system Eq. (1.2) with a large time step, or do both. However, both simplifications may introduce unwanted numerical errors to the propagation of elastic waves on the simulated mesh objects. To study these numerical errors, we have to

analyze the tangent stiffness matrix $\mathbf{K}(\mathbf{q})$ [1],

$$\mathbf{K}(\mathbf{q}) = -\frac{\partial}{\partial \mathbf{q}^T}\mathbf{f}(\mathbf{q}) \tag{1.3}$$

This matrix $\mathbf{K}$ holds key information about the elastic waves and relates to the major challenges for solving Eq. (1.2), as we should see in the next section.

1.2 Challenges

When choosing a time integrator and step size for Eq. (1.2), we need to consider the following issues:

- Stability
- Solving algebraic systems of equations that arise when using implicit integrators
- The effect of artificial damping that arises when using non-conservative integrators

Stability is an issue for fully explicit integrators where the step size must be short enough to resolve the elastic wave of the highest frequency, which relates to the largest eigenvalue of $\mathbf{K}$. While explicit methods are much cheaper per step in general, the stringent step size requirement can lead to a much more expensive overall method due to force evaluation and assembly through the FEM model. Due to this reason, the most popular integrator for elastodynamics in computer graphics is a semi-implicit (SI) method proposed for cloth simulation in [7]. This method (which is equivalent to backward Euler (BE) employing only one Newton iteration at each time step [3]) requires the solution of one linear system per step. In addition to being fast, the SI method also exhibits reasonable large step stability (see also [86]). However, this stability comes from the numerical damping inherent to the method, making it hard to capture the correct dynamic response. Such numerical

[1] In the computer graphics community, there has been inconsistency regarding the sign of the stiffness matrix. Many highly-cited papers used the opposite sign. Personally, I like to stick with this convention because it means that $\mathbf{K}(\mathbf{q})$ is symmetric positive definite (SPD) at rest state, which matches with the canonical Poisson problem subjected to Dirichlet boundary condition.

damping depends on the step size rather than on material properties, and it does not act uniformly on the entire spectrum of $\mathbf{K}$. It remains a popular choice nonetheless because this non-uniform damping which resembles the frequency damping from Rayleigh damping and provides visually appealing results.

In addition to this challenge caused by the largest eigenvalues of $\mathbf{K}$, there is a different challenge on the other end of the spectrum. The efficiency of any integrator used for Eq.(1.2) is contingent on the size of the underlying FEM mesh. While using a fine resolution mesh can capture dynamic deformation with higher accuracy, the resulting system size could be prohibitive to be stepped forward in time quickly, especially for any implicit integrator where iterative nonlinear solvers need to be used. For this reason, it is desirable to keep the mesh resolution as coarse as possible. However, using coarse meshes blindly can greatly reduce the simulation accuracy due to the inconsistency in the spectrum of $\mathbf{K}$ across different mesh resolutions, especially the lowest few eigenvalues. In particular, the term "numerical stiffening" has been used to describe the numerical artifacts in simulation of coarse meshes [17, 44, 66]. We will describe this phenomenon in more detail in Section 2.2.2.

1.3 Outline and contribution

In this thesis we investigate the challenges associated with both spatial and temporal discretization, and construct numerical solutions for elastodynamic simulations in the context of computer graphics, where being visually plausible is more important than being numerically accurate. This is important because perceptible elastic waves are the ones associated with low frequencies, whereas the high frequency waves are inconspicuous and die out quickly due to internal friction. In Chapter 3 and Chapter 6 we develop effective algorithms by finding and isolating these visually critical components first.

In Chapter 3 we propose and demonstrate, for both linear and nonlinear force models, a new method called EigenFit that improves the consistency and accuracy of the lower energy, primary deformation modes, as the spatial mesh resolution is coarsened. EigenFit applies a partial spectral decomposition, solving a generalized eigenvalue problem in the leading mode subspace and then rescaling the first sev-

eral eigenvalues of the coarse mesh by the stiffening ratio measured at rest. EigenFit's performance relies on a novel subspace model reduction technique which restricts the spectral decomposition to finding just a few of the leading eigenmodes. We demonstrate its efficacy on a number of objects with both homogeneous and heterogeneous material distributions.

In Chapter 4 we describe the effectiveness of an exponential Rosenbrock-Euler (ERE) method which avoids excessive discretization-dependent artificial damping. The method is relatively inexpensive and works well with the large time steps used in computer graphics. It retains correct qualitative behavior even in challenging circumstances involving non-convex elastic energies. ERE is designed to handle and perform well in the important cases where the symmetric stiffness matrix is not positive definite at all times. Thus we are able to address a wider range of practical situations than other related solvers. We also analyze the limitation of exponential integrators in the context of stiff elastodynamic simulation.

In Chapter 5 we introduce an additive method framework which may be used to compose new methods coupling two integrators together. Related technical details are described in Appendix A. Describing details of two instances, namely, stiff springs and discontinuous Galerkin spatial discretization, we demonstrate that in elastodynamics, systems can often be split into stiff and non-stiff parts and treated effectively using our additive method. However, in more demanding situations there are limitations to this approach as well, essentially because it splits only the right hand side $\mathbf{b}(\mathbf{u}(t))$ in Eq. (1.2) and does not transform or modify the unknown variables $\mathbf{u}$, which in turn limit the choice of integrators. In our computer graphics context, this is rectified by a method described in the following chapter.

In Chapter 6 we devise a hybrid, semi-implicit method, called SIERE, based on the techniques developed in Chapters 3-5. We show that SIERE produces simulations that are visually as good as those of the exponential method at a computational price that does not increase with stiffness, while displaying stability and damping with respect to the high frequency modes that often improve the performance of the veteran SI method.

In our concluding chapter, we summarize our methods and contributions, and shed light on future directions.

Chapter 2

Elastodynamics

2.1 Elastodynamic Simulation in Computer Graphics

In this chapter we describe the elasticity model with FEM semi-discretization, and analyze the numerical challenges that arise.

2.1.1 Kinematics

In the absence of applied force, an elastic object is in the *reference configuration* $\bar{\Omega} \subset \mathbb{R}^3$. When deformed, the object is in a *deformed configuration,* $\Omega \subset \mathbb{R}^3$. One can think that $\bar{\Omega}$ is resting in the material space and Ω is deforming in the physical space, where both $\bar{\Omega}$ and Ω are bounded open connect sets. The *deformation* can be described by an orientation-preserving mapping $\varphi : \bar{\Omega} \to \mathbb{R}^3$ that is injective on the set Ω. The *configuration space* is the infinite-dimensional space of all possible *deformed configurations* Ω from all admissible $\varphi(X)$, $X \in \bar{\Omega}$.

Deformation gradient

For a deformed object at any point $x = \varphi(X) \in \Omega$ with the corresponding reference point $x \in \bar{\Omega}$, the *deformation gradient* is defined as[1]

$$\nabla\varphi(X) = F(X) = \begin{pmatrix} \frac{\partial\varphi_1}{\partial X_1} & \frac{\partial\varphi_1}{\partial X_2} & \frac{\partial\varphi_1}{\partial X_3} \\ \frac{\partial\varphi_2}{\partial X_1} & \frac{\partial\varphi_2}{\partial X_2} & \frac{\partial\varphi_2}{\partial X_3} \\ \frac{\partial\varphi_3}{\partial X_1} & \frac{\partial\varphi_3}{\partial X_2} & \frac{\partial\varphi_3}{\partial X_3} \end{pmatrix}, \tag{2.1}$$

which turns out to be a useful quantity that relates to many geometrical changes (volume, area, and length) in the object [22]. The condition that φ must be orientation-preserving means

$$\det\nabla\varphi(X) > 0 \text{ for all } X \in \bar{\Omega},$$

which is a nonlinear constraint on admissible φ. Notice that the set of admissible φ is not convex (see [22] Sect. 4.7) because of this kinematic constraint.

Since the gradient operator ∇ is a linear map, the deformation gradient F is a *linearization* of the deformation φ.

Strain measure

A measurement of deformation describing the transformation of the length element, known as the *Green strain*, is defined as

$$E = \frac{1}{2}\left(F^{\mathrm{T}}F - I\right).$$

For *small strain approximation,* the Green strain can be *linearized* to get

$$\varepsilon = \frac{1}{2}\left(F^{\mathrm{T}} + F\right) - I.$$

Since F is linear in φ, we have that E and ε are quadratic and linear in φ, respectively.

[1] Here we define the deformation gradient as a function of X instead of x since x is the unknown that we want to solve for. If the mapping φ is bijective, we can also write the deformation gradient as

$$F(X) = F\left(\varphi^{-1}(x)\right) = \tilde{F}(x)$$

2.1.2 Hyperelastic materials and constitutive models

Intuitively, an elastic object is *hyperelastic* if there is a scalar potential field on the *configuration space* of the object, and the scalar potential of a configuration is the elastic energy of the object at the corresponding deformed state. Formally, if an object is hyperelastic, there exist an energy density function $W : \bar{\Omega} \times \mathbb{M}^3_+ \to \mathbb{R}$ such that $\left(-\nabla \cdot \frac{\partial W}{\partial F}(X,F)\right)$ is a conservative force for all $X \in \Omega$, $F \in \mathbb{M}^3_+$. Here we define $\mathbb{M}^3_+$ as the space for 3 by 3 matrices with nonnegative singular values. The partial derivative $\frac{\partial W}{\partial F}(X,F) = \hat{T}(X, F(X)) = T(X)$ is the first Piola-Kirchhoff stress tensor that maps $X \in \bar{\Omega}$, a point in the reference configuration, to its stress tensor T in the physical space.

If the material is homogeneous, the energy density is only a function of $F \in \mathbb{M}^3_+$. In general, the energy density $W(X,F)$ is dependent on both X (where the point is) and F (how much it deforms).

Intuitively, a constitutive model describes how the deformation of an infinitesimal volume relates to the change in energy density and the stress tensor.

2.1.3 Equations in elastodynamics

With energy density $W(X,F)$ defined by the constitutive model, we can get the total elastic energy $W_{total} = \int_{\bar{\Omega}} W(X,F)\,\mathrm{d}X$ of the object, and the elastic force in the physical space at each material point

$$f_{elastic}(X) = -\nabla \cdot \frac{\partial W}{\partial F}(X,F) = -\nabla \cdot T(X,F).$$

Combining the elastic force with Newton's second law, we arrive at the initial-boundary value problem for the deformation $\varphi(X)$ of a deforming object

$$\rho(X)\,\ddot{\varphi}(X,t) = f_{elastic}(X) + f_{external}(X,t), \qquad X \in \bar{\Omega}, \tag{2.2a}$$

$$\det F(X) > 0, \qquad X \in \bar{\Omega}, \tag{2.2b}$$

$$\varphi(X,t) = \varphi_0 \qquad \text{at } t = 0 \tag{2.2c}$$

$$\varphi(X,t) = \varphi_D \qquad \forall X \in \partial\bar{\Omega} \tag{2.2d}$$

Notice the density function $\rho(X)$ and the external force $f_{external}(X,t)$ are parametrized in the reference configuration since the deformed configuration is the unknown to be solved.

2.1.4 Finite element discretization

To solve Eq.(2.2) numerically, we first semi-discretize in space variables using a finite element method (FEM) and reduce it to a second-order ODE system in time t. We discretize the reference configuration $\bar{\Omega}$ as $\bar{\Omega}_l$[2] in the material space, such that each $X \in \bar{\Omega}_l$ is a material point on the discrete mesh. The discretized deformation is now a mapping $\varphi_l : \bar{\Omega}_l \to \mathbb{R}^3$ that stores the deformation at each material point. We can collect the deformation from these mesh points into a single vector $\mathbf{q}$, and the corresponding velocity $\mathbf{v}$.

$$\mathbf{M}\ddot{\mathbf{q}} = \mathbf{f_{tot}}(\mathbf{q},\mathbf{v}) = \mathbf{f_{els}}(\mathbf{q}) + \mathbf{f_{dmp}}(\mathbf{q},\mathbf{v}) + \mathbf{f_{ext}}(\mathbf{q}) \tag{2.3}$$

Eq.(2.3) is then further reduced to a first-order ODE system. By defining a new state vector $\mathbf{u} = (\mathbf{q},\mathbf{v})^T$, Eq.(2.3) can be written as Eq. (1.2). To complete the description for elastodynamics, we briefly describe popular force models used in computer graphics.

2.1.5 Elasticity model

The elastic force in Eq.(2.3) is derived from a predefined hyper-elastic potential energy model $W(\mathbf{q})$

$$\mathbf{f_{els}}(\mathbf{q}) = -\frac{\partial}{\partial \mathbf{q}} W(\mathbf{q}) \tag{2.4}$$

For example, $W(\mathbf{q})$ is quadratic in the linear elasticity model, and it is quartic in the StVK material model[3]. In other models such as co-rotated FEM with linear material, neo-Hookean material, nonlinear potentials are required to describe the desired physical behavior. From here, the tangent stiffness matrix Eq. (1.3), $\mathbf{K}(\mathbf{q})$,

[2]Here l is the discretization parameter, and for the discretization to be consistent, we require

$$\lim_{l\to 0} \bar{\Omega}_l = \bar{\Omega}.$$

In practice, this parameter can be set in many different ways in mesh generation softwares [37, 76].

[3]Notice that StVK material is still nonlinear, due to nonlinear strain measure.

is defined as

$$\begin{aligned}\mathbf{K}(\mathbf{q}) &= \frac{\partial^2}{\partial \mathbf{q}^T \partial \mathbf{q}} W(\mathbf{q}) \\ &= -\frac{\partial}{\partial \mathbf{q}^T} \mathbf{f}_{\mathbf{els}}(\mathbf{q}).\end{aligned} \tag{2.5}$$

With Eq. (2.5), we can further rewrite Eq. (1.2) as

$$\begin{aligned}\dot{\mathbf{u}}(t) &= \begin{bmatrix} \mathbf{v} \\ \mathbf{M}^{-1}\mathbf{f}_{\mathbf{tot}} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{v} \\ -\mathbf{M}^{-1}\mathbf{K}\mathbf{q} - \mathbf{M}^{-1}\mathbf{D}\mathbf{v} + \mathbf{g} \end{bmatrix} \\ &= \begin{bmatrix} 0 & \mathbf{I} \\ -\mathbf{M}^{-1}\mathbf{K} & -\mathbf{M}^{-1}\mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{q} \\ \mathbf{v} \end{bmatrix} + \begin{bmatrix} 0 \\ \mathbf{g} \end{bmatrix}, \\ \mathbf{g} &= \mathbf{g}(\mathbf{q}(t)) = \mathbf{M}^{-1}(\tilde{\mathbf{f}}_{\mathbf{els}} + \mathbf{f}_{\mathbf{ext}}), \\ \tilde{\mathbf{f}}_{\mathbf{els}}(\mathbf{q}) &= \mathbf{K}\mathbf{q} + \mathbf{f}_{\mathbf{els}}(\mathbf{q}).\end{aligned} \tag{2.6}$$

Solving the system of first order ODEs Eq. (2.6) will give us the motion and deformation trajectory of the simulated object. It is important to note that the dimension of this system can be large and that its assembly at any time point can be expensive.

2.1.6 Damping model

One of the most popular damping models in computer graphics used in Eq.(2.3) is the local Rayleigh damping model.

$$\mathbf{f}_{\mathbf{dmp}}(\mathbf{q}, \mathbf{v}) = -\mathbf{D}(\mathbf{q})\mathbf{v}, \tag{2.7a}$$

$$\mathbf{D}(\mathbf{q}) = \alpha\mathbf{M} + \beta\mathbf{K}(\mathbf{q}). \tag{2.7b}$$

The main feature of the Rayleigh damping model is the presence of frequency damping. It is popular for three reasons:

1. Due to the first and second laws of thermodynamics, internal friction naturally leads to frequency dependent damping [75], thus carefully tuned Rayleigh

damping can appear natural to the eye.

2. The presence of $\mathbf{K}(\mathbf{q})$ attenuates high frequencies exponentially and helps the stability of numerical integration.

3. If $\mathbf{K}$ is constant, then the entire force is linear and is cheap to evaluate.

Nevertheless, in recent years, there has been progress in using more accurate and controllable damping models [51, 92].

2.2 Related work

2.2.1 Integrators

As described in Chapter 1, SI has been the workhorse for integrating elastodynamic simulation in computer graphics. To derive SI, we first write BE for Eq. (2.6):

BE

$$\mathbf{u_{n+1}} = \begin{bmatrix} \mathbf{q_{n+1}} \\ \mathbf{v_{n+1}} \end{bmatrix} = \begin{bmatrix} \mathbf{q_n} + h\mathbf{v_{n+1}} \\ \mathbf{v_n} + h\mathbf{M}^{-1}\mathbf{f_{tot}}(\mathbf{q_{n+1}}, \mathbf{v_{n+1}}) \end{bmatrix}$$

which can be simplified to

$$\mathbf{v_{n+1}} = \mathbf{v_n} + h\mathbf{M}^{-1}\mathbf{f_{tot}}(\mathbf{q_n} + h\mathbf{v_{n+1}}, \mathbf{v_{n+1}}),$$

or

$$\mathbf{v_{n+1}} - \mathbf{v_n} - h\mathbf{M}^{-1}\mathbf{f_{tot}}(\mathbf{q_n} + h\mathbf{v_{n+1}}, \mathbf{v_{n+1}}) = 0. \tag{2.8}$$

The Jacobian of Eq. (2.8) is

$$\mathbf{I} + h\mathbf{M}^{-1}\mathbf{D} + h^2\mathbf{M}^{-1}\mathbf{K}. \tag{2.9}$$

Notice the sign in Eq. (2.9) came from Eqs. (2.5) and (2.7b). Here $\mathbf{K}$ is the stiffness at the current configuration. Applying Eq. (2.8) to one Newton iteration at the

beginning of each time step we have SI:

$$\mathbf{v_{n+1}} = \quad \mathbf{v_n} + h(\mathbf{I} + h\mathbf{M}^{-1}\mathbf{D} + h^2\mathbf{M}^{-1}\mathbf{K})^{-1}(\mathbf{M}^{-1}(\mathbf{f_{tot}} - h\mathbf{K}\mathbf{v_n})) \qquad (2.10)$$

We can solve the symmetric linear systems in Eq. (2.10) by either iterative methods, such as preconditioned conjugate gradient, or direct methods. In computer graphics, it is common to use direct Cholesky solvers [24, 47, 58, 88].

Other implicit time integrators have also been used for dynamics, including BDF2 [34], implicit/explicit methods [13, 29], and variational integrators [43]. Unlike the SI method and like BE, these methods require the solution of a system of nonlinear equations at every time step. In general applications these nonlinear equations are solved by some inexact Newton's method employing a iterative Krylov subspace method for the sub-iterations [45]. If the nonlinear equations are solved at each time step to sufficient accuracy, then these implicit Newton-Krylov methods are often more robust and accurate. Hence they are typically used when the cost of time stepping is not the major concern.

A branch of implicit integrators commonly used in the graphics community can be derived from the discretization for the Lagrangian [11, 27, 30, 53–55, 65]. While being implicit, all these algorithms avoid the expensive Newton-Krylov methods and accelerate the nonlinear solve by using superior optimization techniques. This series of methods also applies to more general and complicated material models. However, as material stiffness increases, the performance level of all the techniques mentioned above drops significantly. Specifically, as material stiffens, these methods require many iterations, and the simulated material will appear softer if the optimization process stops prematurely. To look at this challenge in more detail, let's look at the Jacobian for the nonlinear system from BE and implicit midpoint method (IM). For BE, we already have the nonlinear system Eq. (2.8), and its Jacobian Eq. (2.9)

For IM we have

$$\mathbf{u_{n+1}} = \begin{bmatrix} \mathbf{q_{n+1}} \\ \mathbf{v_{n+1}} \end{bmatrix} = \begin{bmatrix} \mathbf{q_n} + \frac{h}{2}(\mathbf{v_n} + \mathbf{v_{n+1}}) \\ \mathbf{v_n} + h\mathbf{M}^{-1}(\mathbf{f_{tot}}\left(\frac{\mathbf{q_n}+\mathbf{q_{n+1}}}{2}, \frac{\mathbf{v_n}+\mathbf{v_{n+1}}}{2}\right)) \end{bmatrix}$$

which can be simplified to

$$\mathbf{v_{n+1}} = \mathbf{v_n} + h\mathbf{M}^{-1}(\mathbf{f_{tot}}\left(\mathbf{q_n} + \tfrac{h}{4}(\mathbf{v_{n+1}} + \mathbf{v_n}), \tfrac{\mathbf{v_n}+\mathbf{v_{n+1}}}{\mathbf{2}}\right) + \mathbf{f_{tot}}\left(\mathbf{q_n}, \mathbf{v_n}\right))$$

or

$$\mathbf{v_{n+1}} - \mathbf{v_n} - h\mathbf{M}^{-1}\mathbf{f_{tot}}\left(\mathbf{q_n} + \frac{h}{4}\mathbf{v_{n+1}}, \frac{\mathbf{v_n} + \mathbf{v_{n+1}}}{\mathbf{2}}\right) = 0. \tag{2.11}$$

The Jacobian of Eq. (2.11) is

$$\mathbf{I} + \frac{1}{2}h^2\mathbf{M}^{-1}\mathbf{D} + \frac{1}{4}h^2\mathbf{M}^{-1}\mathbf{K} \tag{2.12}$$

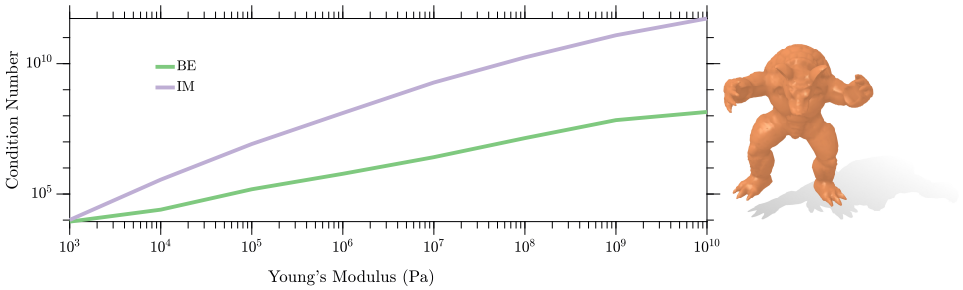


Figure 2.1: 1-norm condition estimate vs. material stiffness.

In both Eq. (2.9) and Eq. (2.12), **D** and **K** are evaluated at the current configuration. For one example, we simulate an armadillo with nonlinear neo-Hookean material model swinging under gravity without any Rayleigh damping. Figure 2.1 we plot the 1-norm condition estimate for the BE system matrix Eq. (2.9) and IM system matrix Eq. (2.12). Notice that the condition numbers increase rapidly as we crank up the stiffness (Young's Modulus, YM increases), which implies that we are solving a harder problem when we simulate stiff elastodynamics behavior. Here ill-conditioning is an issue not because we are losing multiple digits of accuracy: in computer animation high accuracy is rarely required. Nevertheless, ill-conditioning is still a challenge because it causes convergence difficulties to several iterative methods. As a demonstration, we record in Table 2.1 the number of

BFGS quasi-Newton iterations[4] and Newton iterations used for solving the implicit equations for BE observe a clear upward trend as we stiffen the material, especially for BFGS. In short, for stiff elastodynamic simulations the system became harder to solve using the optimization based solvers mentioned above.

Table 2.1: Number of BFGS quasi-Newton iteration and exact Newton iteration used for integrating a mass-spring system with 5 particles for 20 frames at $h = 0.1s$.

Spring constant (N/m)	$1e-1$	$1e0$	$1e1$	$1e2$	$1e3$	$1e4$
BFGS iterations	59	100	205	614	1199	1510
Newton	20	20	39	45	70	80

Due to this challenge, exponential integrators, a branch of integrators specialized in stiff problems, have been gaining popularity in graphics in recent years [19, 60–62, 70]. A review of such methods can be found in [35]. In [36, 56] exponential integrators are compared to Newton-Krylov methods for various stiff systems. However, traditional exponential integrators encounter difficulties when applied for very stiff elastodynamics problems in computer graphics. In Chapter 4 we will describe this challenge in detail.

2.2.2 Numerical Stiffening

The term "numerical stiffening" has been used to describe the numerical artifacts in simulation on coarser meshes [17, 18, 44, 66]. However, none of the works mentioned analyze the cause of numerical stiffening in the context of elastodynamics. In the next section, we explain numerical stiffening in a simpler context.

To better understand numerical stiffening, let us consider the discretization of the simplest classical wave equation given by the partial differential equation (PDE)

$$u_{tt} = c^2 \nabla^2 u, \tag{2.13}$$

[4]Here we use Matlab's function [38] with default tolerance.

where c is the speed of wave. Further, assume that the PDE is in one space variable and subject to the boundary conditions $u(t, x = 0) = u(t, x = 1) = 0$.

Looking for solutions of the form $u(t,x) = e^{t\sqrt{\lambda}t}U(x)$ leads to the eigenvalue problem

$$-c^2\nabla^2 U = \lambda U, \tag{2.14}$$

with eigenvalues

$$\lambda_j = (jc\pi)^2, \quad j \in \mathbb{N}. \tag{2.15}$$

Here the lowest eigenvalue is $\lambda_1 = \pi^2 c^2 \approx 9.86c^2$, with $\sqrt{\lambda_1}$ the frequency (in time) of the dominating dynamic mode.

Semi-Discretization It is common to semi-discretize the PDE in Eq. (2.13) in space, and then step the resulting ODE in time to simulate dynamics. This semi-discretization in space is, of course, closely related to the eigenvalue problem Eq. (2.14). However, the spatial discretization error will pollute the overall time trajectory (dynamic behavior) as it alters the analytic eigenvalues given in Eq. (2.15).

Finite Element Method Applying conformal FEM, we first convert Eq. (2.14) to weak form. The eigenvalue problem (2.14) is then equivalent to requiring that for all appropriate test functions V,

$$c^2 \int \nabla U \nabla V dx = \lambda \int VU dx. \tag{2.16}$$

We apply simplest Galerkin FEM by choosing piecewise linear "hat functions" satisfying the boundary conditions as the function space for U and V. The resulting element stiffness matrix from the left hand side and mass matrix from the right hand side of Eq. (2.16) are, respectively,

$$\mathbf{K}_{e,l} = \frac{c^2}{l}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, \quad \mathbf{M}_{e,l} = \frac{\lambda l}{6}\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}.$$

After assembly and elimination of the boundary variables, we have converted Eq. (2.16) to the system

$$\begin{aligned}
\mathbf{K}_l\mathbf{u} &= \lambda\mathbf{M}_l\mathbf{u}, \quad \text{where} \qquad (2.17)\\
\mathbf{K}_l &= -\frac{c^2}{l}\begin{bmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & -1 & 2 & -1 & \\ & & & -1 & 2 & \end{bmatrix},\\
\mathbf{M}_l &= \frac{l}{6}\begin{bmatrix} 4 & 2 & & & & \\ 2 & 4 & 2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & 2 & 4 & 2 & \\ & & & 2 & 4 & \end{bmatrix}.
\end{aligned}$$

We have discretized Eq. (2.14) and converted it into a generalized eigenvalue problem. The lowest eigenvalue at a range of mesh resolutions is plotted in Figure 2.2. Notice that approximated eigenvalues using the FEM approach the exact one in Eq. (2.15) *from above* as $l \to 0$. This artifact is numerical stiffening, and here the simulated object is stiffer than the physical model. Moreover, at each progressively coarser mesh a corresponding simulation will be stiffer when compared to a finer mesh simulation.

This numerical stiffening is a direct consequence of an old result [23] showing that the Galerkin approximation of the general eigenvalue problem will always lead to larger eigenvalues. In brief we are searching for the minimum of an approximated Ritz functional in an increasingly less inclusive function space as we coarsen the mesh in the weak form of a boundary value PDE problem. Note that this result will not hold in general, if we lump the mass matrix or use finite differences; although such method alterations will not improve the numerical stiffening: it is only the monotonic approach to the limit that would be lost. Based on this analysis and observation, we derive our algorithm, EigenFit, in the next chapter.

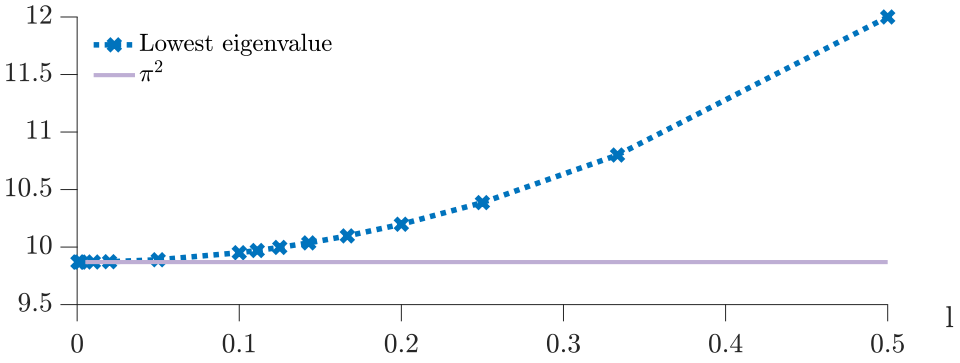


Figure 2.2: The lowest eigenvalue of Eq. (2.17) with $c = 1$ changes depending on the mesh resolution. This eigenvalue approaches the true eigenvalue π^2 of Eq. (2.15) monotonically from above as the mesh resolution improves.

Chapter 3

EigenFit for Consistent Elastodynamic Simulation Across Mesh Resolution

3.1 Introduction

The calibration, capture, animation and design of soft-body dynamics are critical for a wide range of domains spanning from robotics, automotive design and biomechanics to film, interactive animation and digital fabrication design. A fundamental computational bottleneck in all of these domains is the efficient forward simulation of elastodynamics. To simulate elastodynamics researchers often first discretize the governing differential system in space employing a finite element method (FEM) [10, 78], and then in time using a time-stepping method.

For simulation-based applications there are many reasons to alter the spatial and temporal resolution of the computational mesh. Critically, coarsening in both space and time is often required in order to make applications practical. Most notably, runtime costs for soft-body simulations scale superlinearly in the number of nodes in the FE mesh. However, both spatial and temporal coarsening introduce undesirable and often unexpected numerical artifacts that reduce the controllability, consistency, accuracy and effectiveness of resulting simulations. While un-

Figure 3.1: Numerical stiffening can cause animations on a coarser FEM mesh (sequence (c)) to deviate from corresponding ones on a finer mesh (sequence (a)). To fix this error, EigenFit adjusts the magnitudes of the leading modes on the coarse mesh using the fine mesh at rest state, resulting in sequence (b). The meshes are colored by normalized positional error with respect to the finer mesh. EigenFit often outperforms current leading alternatives.

derstanding and reducing artifacts due to *temporal coarsening* is a long-standing and active area of investigation, the complementary problem of treating artifacts due to *spatial coarsening* has remained much less addressed. Here, we propose a new method, EigenFit, to mitigate spatial coarsening artifacts in elastodynamic simulation.

While aggressive spatial coarsening can significantly improve runtime it has severe consequences for resulting soft-body simulations. First, as coarsening progresses the computational mesh loses geometric accuracy. Second, higher frequency modes that can only be expressed on finer meshes disappear as the mesh is progressively coarsened. Third, coarsening introduces *numerical stiffening*. The latter, in analogy to numerical dissipation, is the change in a simulated material's effective stiffness directly as consequence of spatial mesh coarsening.[1] This change in observed stiffness becomes more pronounced, despite material parame-

[1] Numerical stiffening is not to be confused with other notions of material stiffness that are independent of mesh discretization and often result from having very different scales in the underlying simulated differential system. Here it is the low eigenvalues that are at play, and numerical stiffness difficulties ebb away as the mesh resolution is increased.

ters remaining fixed, as we increase element sizes; see Figure 3.1.

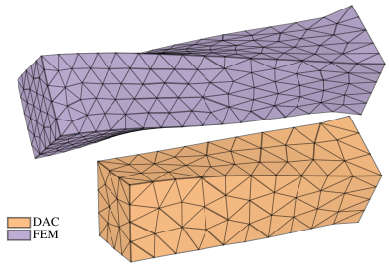
Numerical stiffening thus remains a fundamental block to consistent and accurate simulations across changing mesh resolutions. Mesh resolution changes all the time, but, in principle, simulated physics should not change with it. Towards this goal we propose and demonstrate a new algorithm, EigenFit, that improves the consistency and accuracy of both linear and nonlinear model elastodynamic simulations. Specifically, EigenFit corrects the lower energy, primary deformation modes of spatially coarsened meshes to gain consistency in elastodynamic simulations across a broader range of mesh resolutions and motions. In particular we will show that consistent dynamic behavior for the same shape and material is maintained as the spatial mesh resolution is varied in a range that preserves primary eigenmode shapes.

At its core EigenFit is conceptually direct. We apply partial spectral decomposition, solving a generalized eigenvalue problem to rescale leading eigenmodes of a coarsened model with initial-state ratios between fine- and coarse-resolution mesh eigenvalues. For nonlinear models EigenFit then updates this fit as time progresses with a local re-linearization and decomposition that calculates just leading modes at each time step. By restricting this per-time step correction to subspaces formed by just the leading eigenmodes, EigenFit avoids prohibitively expensive, repeated full decompositions and so efficiently performs subspace correction followed by a low-rank update to map the model back to the full deformation space for time stepping.

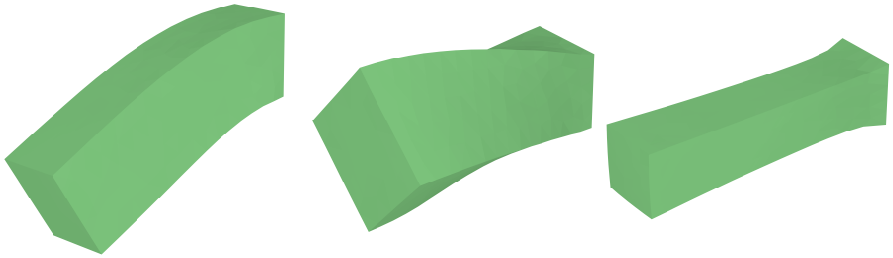
EigenFit performs exceptionally well for linear constitutive materials; see Figure 3.1 and Section 3.4.1. Furthermore, unlike some other methods EigenFit is effectively oblivious to material heterogeneity and performs without change for heterogeneous materials where the Young modulus is a distributed parameter function. It naturally handles relatively stiff objects where motion is generated through local softer joints, as demonstrated in Section 3.4 and the supplementary video.

3.2 Related work and numerical stiffening

In this work we focus on constructing coarse spatial-mesh elastodynamic simulations that improve accuracy and consistency of the primary, lowest energy eigen-



(a) DAC inconsistent



(b) 1st, 3rd, and 6th Modes of the bar mesh

Figure 3.2: DAC cannot fix some important deformation modes. In this figure both bars are simulated with Young's modulus $YM = 4kPa$ and Poisson's ratio $\nu = 0.3$. However, the stretching and twisting motions are inconsistent. DAC only uses the first mode to adjust the dynamic behavior of the object. But in this example, the first mode of the elastic bar is a bending mode (see (b)). Consequently, the twisting (3rd mode) and stretching (6th mode) motions are inconsistent. Subfigure (a) also demonstrates that numerical stiffening can still occur away from locking at low Poisson ratio.

modes with respect to a fine-resolution reference mesh. These leading modes generally exhibit the largest and most visible deformations in elastica and are thus critical for visual and functional applications in many domains including visual effects, e.g., film and animation, as well as functional design, e.g., robotics and biomechanics [17].

A key obstacle to this goal are spatial coarsening artifacts that arise in the form of numerical stiffening. In Section 2.2.2 we look further into the source of numerical stiffening. First, however, we briefly review existing remedies for it.

Numerical stiffening in conformal FEM is treated directly by adapting the spa-

tial mesh for accuracy until error is reduced and thus the artifact is mitigated [6]. These refinements traditionally increase mesh-resolution (h-refinement) and/or element order (p-refinement) [10]. Many refinement approaches monitor error on the deformed mesh and so adapt during simulation [9, 32, 64]. Most recently Schneider et al. [74] proposed p-refinement on unstructured meshes based on analysis of the undeformed mesh at rest. This latter approach can be highly effective when a small number of ill-shaped elements are ruining accuracy. However, in the spatial coarsening setting we have a global problem where all elements may be problematically large. Thus, if we were to adopt a refinement solution to alleviate numerical stiffening, the entire domain would require refinement, leading to impractical meshes with high degree of freedom that one wishes to avoid computing with in the first place [17].

Alternatively, numerical homogenization methods [16, 44, 66, 69, 85] have also been proposed to better model energy density at coarse resolution. These methods work well in the static setting but do not account for inertia and fail to extend in the dynamic setting [17]. Moreover, the primary focus in these methods is specifically homogenizing static solutions for heterogeneous materials. Here we focus on gaining accuracy for dynamics solutions without modifying the given constitutive material model.

Nonconforming FE methods, especially discontinuous Galerkin (DG), are another potential avenue for reducing numerical stiffening. In particular, DG methods can be applied to help reduce the related problem of numerical locking [41, 42]. By adding additional degrees of freedom to the system and weakly enforcing inter-element continuity with penalty terms, DG methods offer freedom to avoid degree-of-freedom locking. However, this additional flexibility unfortunately does not provide dynamical correction for numerical stiffening which occurs in addition to locking; see Figures 3.2a and 3.3.

Most recently, Chen et al. [18] combined homogenization and nonconforming FE for coarse static solutions to heterogeneous materials. They also interestingly demonstrated modest improvement on a simple dynamics example, although they note that no explicit correction in this method is made towards fixing phase error.

The closest work to ours is that of Chen et al. [17]. They observe that rather coarse meshes can accurately capture low-frequency mode geometry, and hence

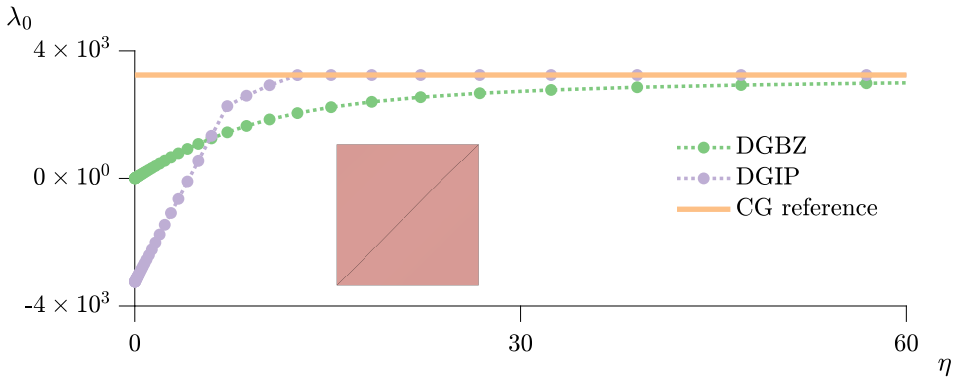


Figure 3.3: The lowest non-zero eigenvalue of a square mesh with two triangles under the Discontinuous Galerkin BZ formulation and IP formulation [18, 41, 42] with a reference continuous Galerkin (CG) solution. A penalty parameter η can be used to adjust the effective stiffness and alleviate numerical stiffening somewhat. However, the value of η is purely empirical.

can capture the primary mode shapes of a simulated object. To a priori correct this model towards handling numerical stiffening artifacts, they proposed a dynamics aware coarsening (DAC) method that precomputes and applies, per mesh, a one-time numerical rescaling of Young's modulus to match the lowest eigenmode frequency of a coarse mesh to an accurate sample. While this rescaling can be highly effective in many cases where a single primary mode dominates deformation dynamics, it is unable in many cases to fix a wide number of nonlocal and nonlinear errors and inconsistencies caused by numerical stiffening; see Figure 3.2. In particular, if deformations involve stretching, twisting and/or bending motions, then more than a single parameter adjustment will generally be needed. EigenFit iteratively re-fits a set of leading eigenmodes to achieve these corrections; see Figure 3.5 and Sections 3.3 and 3.4 for detailed discussion and comparisons. We refer to [78, 91] for a discussion of eigenmodes and model reduction. To compute the leading eigenmodes with symmetric stiffness matrix and mass matrix, we use the implicitly restarted Lanczos method [48, 79] from the Spectra C++ library [71]. Furthermore, the inverse problem solved per mesh as part of DAC could become difficult in some applications, a difficulty that EigenFit avoids by not solving any

inverse problems.

3.3 Method

We now describe in detail a new method that mitigates numerical stiffening by matching primary vibration modes. As in [17] our setting requires not to coarsen the mesh beyond a point where the low-energy mode shapes differ between fine and coarse by more than some small tolerance. The method described below applies for linear and nonlinear force models.

3.3.1 Mesh eigenmodes

To introduce our ideas gradually, suppose at first that we perform an FEM simulation of an elastic object motion under a linear elastic force at two distinct mesh resolutions. We then have respective time-independent mass matrices $\mathbf{M}_c, \mathbf{M}_f$ and stiffness matrices $\mathbf{K}_c, \mathbf{K}_f$ corresponding to the coarse and fine meshes. We can find the eigenmodes $\mathbf{u}_{c,i}$ and $\mathbf{u}_{f,i}$ (modes of deformation) of the coarse and fine meshes by carrying out the principal component analysis

$$\begin{aligned} \mathbf{K}_c\mathbf{u}_{c,i} &= \lambda_{c,i}\mathbf{M}_c\mathbf{u}_{c,i}, \quad i = 1,\ldots,N_c, \\ \mathbf{K}_f\mathbf{u}_{f,i} &= \lambda_{f,i}\mathbf{M}_f\mathbf{u}_{f,i}, \quad i = 1,\ldots,N_f, \end{aligned}$$

where $N_c < N_f$. While the eigenmodes $\mathbf{u}_{c,i}$ and $\mathbf{u}_{f,i}$ have different sizes, they correspond to the same ith deformation mode of the object for low indices i, $i = 1,2,\ldots,m$, where $m \leq N_c$. However, as discussed in Section 3.2, their corresponding eigenvalues $\lambda_{c,i}$ and $\lambda_{f,i}$ will be different due to numerical stiffening. Although their relative difference is only by $\mathscr{O}(1)$, this can change the object deformation properties significantly; see Figures 3.1 and 3.5. In particular, avoiding mass lumping we have $\lambda_{c,i} > \lambda_{f,i}$, and since these eigenvalues directly relate to the oscillation frequency of the corresponding mode, simulating on a coarse mesh has been observed to make the object look stiffer and oscillate faster.

To enable such an eigenvalue adjustment, the first idea that may spring to mind is to use the fact that, if the scalar λ and the vector u are an eigenpair of a matrix $\mathbf{A}$ satisfying $\mathbf{A}\mathbf{u} = \lambda\mathbf{u}$, then for any integer j, also $\mathbf{A}^j\mathbf{u} = \lambda^j\mathbf{u}$. Thus,

one could construct an interpolating polynomial $p(\mathbf{A})$ for $\mathbf{A} = \mathbf{M}_c^{-1}\mathbf{K}_c$ such that $p(\mathbf{A})\mathbf{u}_{c,i} = \lambda_{f,i}\mathbf{u}_{c,i}$ for the first few modes, $i = 1, 2, \ldots, m$, and replace $\lambda_{c,i}$ by $p(\lambda_{c,i})$. However, this idea does not work out, even for small fittings of $m = 3$. This is because the eigenvalue spread is typically wide and unequal. The resulting, *extrapolated* values of $p(\lambda_{c,i})$ for $i > m$ are often rather far from the corresponding $\lambda_{f,i}$, and the process is highly ill-conditioned even when using a Lagrange basis for the polynomial interpolation process. We thus proceed with a direct strategy for modifying the primary coarse-mesh eigenvalues.

3.3.2 Direct eigenvalue modification

Continuing our gradual development, in this section we assume that the stiffness matrix K is constant and symmetric positive definite. To reduce the numerical stiffening effect we modify the first m eigenvalues directly within the eigendecomposition, where the parameter m is not large (typically in our calculations, $m < 20$). First, using the shorthand $\mathbf{M} = \mathbf{M}_c,\ \mathbf{K} = \mathbf{K}_c$, we write the generalized eigenvalue problem from Section 3.3.1 above as

$$\mathbf{KU} = \mathbf{MUD}, \tag{3.1}$$

where $\mathbf{D}$ is a diagonal matrix having the eigenvalues on its main diagonal, and $\mathbf{U}$ is the matrix having the eigenvectors as columns:

$$\begin{aligned} \mathbf{D} &= \operatorname{diag}\left[\lambda_{c,1}, \lambda_{c,2}, \ldots, \lambda_{c,N_c}\right], \\ \mathbf{U} &= \begin{bmatrix} | & | & \cdots & | \\ \mathbf{u}_{c,1} & \mathbf{u}_{c,2} & \cdots & \mathbf{u}_{c,N_c} \\ | & | & \cdots & | \end{bmatrix}. \end{aligned} \tag{3.2}$$

Solving such problem also ensures mass-orthogonality, $\mathbf{U}^T\mathbf{MU} = \mathbf{I}$ and $\mathbf{MUU}^T = \mathbf{I}$.[2]

Next, correction of frequency and thus eigenvalues is direct in the linear setting.

[2]Note, however, that $\mathbf{U}$ is not an orthogonal matrix in general.

We simply replace $\mathbf{D}$ in Eq. (3.2) with

$$\mathbf{D}^{\dagger} = \text{diag}\,[\lambda_{f,1}, \lambda_{f,2}, \ldots, \lambda_{f,m}, \lambda_{c,m+1}, \ldots, \lambda_{c,N_c}]. \tag{3.3}$$

We select the parameter m using the Hausdorff distance between the eigenshapes of the coarse and fine meshes, following Chen et al. [17]. Now, using mass orthogonality [8], we can approximate the generalized eigenvalue equations (3.1) by $\mathbf{K}^{\dagger}\mathbf{U} = \mathbf{M}\mathbf{U}\mathbf{D}^{\dagger}$, and obtain our modified stiffness matrix $\mathbf{K}^{\dagger}$ directly from this expression:

$$\mathbf{K}^{\dagger} \quad = \quad \mathbf{M}\mathbf{U}\mathbf{D}^{\dagger}\mathbf{U}^{T}\mathbf{M}. \tag{3.4a}$$

3.3.3 Nonlinear Materials

The approach described in Section 3.3.2 is direct and simple. However, it is only applicable to linear force models, where the stiffness matrix and the modal analysis remain constant throughout the simulation.

In this section we extend our observations from the linear setting to nonlinear force models, e.g., neo-Hookean, StVK, and the spline model of Xu et al. [93], as well as to co-rotational FEM.

The obvious step to begin with is to consider a local linearization, i.e., reinitialization, at each time step. However, there are several obstructions to this approach. One is that the Jacobian matrix that arises at each given time step depends on the solution there which, in turn, depends in general on all modes. So there is some mixing of modes in the nonlinear case that does not happen in the linear one. Moreover, approximations for the stiffness matrix, for instance when using neo-Hookean force, might give rise to negative eigenvalues (corresponding possibly to material compression). Finally, the sheer cost of carrying out an eigendecomposition at each time step can quickly become prohibitive.

Rescaling a subset of eigenvalues

Rather than a one-time precomputed rescaling as performed in Chen et al. [17], we update our fitting iteratively with a local re-linearization at each time step. We first designate our set of *m target* eigenvalue ratios $r_i = \frac{\lambda_{f,i}}{\lambda_{c,i}}$ for $i = 1, 2 \ldots, m$, either precomputed from the rest shape of the meshes or else set by hand, e.g., for animation design.

At each time step we then

1. Build a local tangent stiffness matrix $\mathbf{K}$ and perform generalized eigenvalue decomposition on the coarse mesh as in Eqs. (3.1)–(3.2). Note that, unlike for linear forces, here both the eigenvectors and eigenvalues depend on the state $\mathbf{u}_{c,i} = \mathbf{u}_{c,i}(\mathbf{q}_t)$ and $\lambda_{c,i} = \lambda_{c,i}(\mathbf{q}_t)$.

2. Set the ratio matrix

$$\mathbf{R} = \text{diag}\,[r_1, \ldots, r_m, 1, \ldots, 1], \tag{3.5}$$

   and then $\mathbf{D}^{\dagger} = \mathbf{RD}$. This adjusts the leading eigenvalues to their target values.

3. In the locally linear setting the modified tangent stiffness matrix is now given by the expression in Eq. (3.4a). The force is correspondingly adjusted to match:

$$\mathbf{f}^{\dagger} = \mathbf{M}\left(\mathbf{URU}^{-1}\right)\mathbf{f} = \mathbf{MURU}^{T}\mathbf{f}. \tag{3.6}$$

In the general nonlinear material model setting the eigenvalues are not always positive [21]. This can mix the rescaling effect on different deformation modes. To mitigate this issue, during assembly of the tangent stiffness matrix, we project any indefinite element stiffness matrices to the PD cone, by replacing offending negative eigenvalues with a small positive value ε (we chose $\varepsilon = 1e-3$), in turn guaranteeing positive definiteness of the global stiffness matrix; see Teran et al. [84].

Although the generalized eigenvalue decomposition is performed on the coarse mesh at the beginning of each time step, the corresponding decomposition on the fine mesh is performed only once, at the beginning. Thus, in (3.5) we keep the

rescaling ratios fixed, to avoid calculating ratios of the eigenvalues of coarse and fine at different times. The updated coarse modes, however, are of course used in Eqs. (3.6) and (3.4a).

Subspace correction

As noted earlier, a fundamental drawback of our strategy so far is the computation of the full eigendecomposition of the system matrix at every time step. Observe, however, that our goal is just the re-fitting of the lowest m modes. Thus, we may focus much of our attention on a subspace of the eigendecomposition. This is especially important as computation of a low number of eigenpairs is significantly less expensive (see, e.g., [82]). Starting with this observation we can instead perform our rescaling in the reduced space spanned by the lowest s modes, where s is a small integer in the range $m \leq s \leq N_c$. Its value depends on the task complexity and the computational cost.

Concretely, at each time step we compute only the lowest s eigenpairs. Denote the corresponding reduced-space eigenvalue and eigenmode matrices $\mathbf{D}_s$ and $\mathbf{U}_s$. In contrast to standard linear and nonlinear modal analysis and dimension reduction methods, here we build a small, *local* subspace about each time step. In our setting the relevant eigendecomposition can be efficiently computed by performing a sequence of inverse power iterations to compute at each time step the lowest few eigenpairs corresponding to the dominant motions. Note that in the subspace of dimension $s < N_c$, $\mathbf{U}_s^T \mathbf{M} \mathbf{U}_s = \mathbf{I}_s$, but $\mathbf{M} \mathbf{U}_s \mathbf{U}_s^T \neq \mathbf{I}$.

Within each time step's local subspace, the corresponding mass matrix is the $s \times s$ identity matrix, while the subspace tangent stiffness matrix $\mathbf{K}_s$ is diagonal. The corresponding reduced stiffness matrix and force are then

$$\mathbf{K}_s = \mathbf{U}_s^T \mathbf{K} \mathbf{U}_s = \text{diag}[\lambda_{c,1}, \ldots, \lambda_{c,s}], \quad \mathbf{f}_s = \mathbf{U}_s^T \mathbf{f}. \tag{3.7}$$

Next, following Section 3.3.3, we build the diagonal subspace rescaling matrix of size s in the reduced space,

$$\mathbf{R}_s = \text{diag}\ [r_1, \ldots, r_s].$$

The corresponding modified subspace tangent stiffness matrix and forces, rescaled by r_i, are then

$$\mathbf{K}_s^\dagger = \mathbf{R}_s {\mathbf{U}_s}^T \mathbf{K} \mathbf{U}_s = \mathbf{R}_s \mathrm{diag}[\lambda_{c,1}, \ldots, \lambda_{c,s}], \tag{3.8a}$$

$$\mathbf{f}_s^\dagger = \mathbf{R}_s {\mathbf{U}_s}^T \mathbf{f}. \tag{3.8b}$$

The EigenFit method

Finally, our local subspace corrections to the primary dynamic modes must be added back to the full system for simulation of complete dynamics.

To move our subspace correction into the full system we start by observing that, for an arbitrary diagonal scaling matrix $\mathbf{A}_s$, the matrix $\mathbf{A}_s \mathbf{U}_s^T$ rescales and projects down to the s-mode basis, as in Eqs. (3.8) above. In turn, $\mathbf{M}\mathbf{U}_s^T \mathbf{A}_s$ lifts the projected quantity back to the full space of the FEM system. To jointly remove the current, uncorrected force contributions spanning the subspace, and to add their corrected counterparts back in, we then use $(\mathbf{R}_s - \mathbf{I}_s)$. Thus, the full-space tangent stiffness matrix, with our subspace correction, is

$$\mathbf{K}^\ddagger = \mathbf{K} + \mathbf{M}\mathbf{U}_s(\mathbf{R}_s - \mathbf{I}_s)\mathbf{U}_s^T \mathbf{K} \mathbf{U}_s \mathbf{U}_s^T \mathbf{M}, \tag{3.9a}$$

$$= \mathbf{K} + \mathbf{M}\mathbf{U}_s(\mathbf{R}_s - \mathbf{I}_s)\mathrm{diag}[\lambda_{c,1}, \ldots, \lambda_{c,s}]\mathbf{U}_s^T \mathbf{M}, \tag{3.9b}$$

and the corrected force is

$$\mathbf{f}^\ddagger = \mathbf{f} + \mathbf{M}\mathbf{U}_s(\mathbf{R}_s - \mathbf{I}_s)\mathbf{U}_s^T \mathbf{f}. \tag{3.9c}$$

Note that with the above model we have regained the target correction sought in Section 3.3.3, while only incurring the cost of the small s-mode eigendecomposition.

3.3.4 Implementing an EigenFit integration step

The matrix $\mathbf{K}^{\ddagger}$ in Eq. (3.9b) consists of the (relatively) large but sparse positive definite $N_c \times N_c$ matrix $\mathbf{K}$ plus an added full (dense) matrix of size $N_c \times N_c$ and rank s. This may make computations using any of our time integration methods expensive if one is not careful.

Let us define

$$\mathbf{Y} = \mathbf{M}\mathbf{U}_s(\mathbf{R}_s - \mathbf{I}_s), \quad \mathbf{Z}^T = \mathrm{diag}[\lambda_{c,1}, \ldots, \lambda_{c,s}]\mathbf{U}_s^T\mathbf{M}. \tag{3.10a}$$

Then $\mathbf{Y}$ and $\mathbf{Z}$ are both $N_c \times s$ (i.e., "long and skinny": $s \ll N_c$), and

$$\mathbf{K}^{\ddagger} = \mathbf{K} + \mathbf{Y}\mathbf{Z}^T. \tag{3.10b}$$

For the SI time discretization method we have to solve a system of the form

$$(\mathbf{M} + h^2\mathbf{K}^{\ddagger})\mathbf{v}_+ = \mathbf{z}$$

for the velocities $\mathbf{v}_+$ at the next time level (where h is the step size). Here $\mathbf{z}$ is a known right hand side. If we use a preconditioned conjugate gradient method for this, then an oracle for an efficient matrix-vector product can be readily constructed, taking into account the thinness of $\mathbf{Y}$ and $\mathbf{Z}$.

For some other cases, however, when N_c is not extremely large and the physical system is stiff, i.e., in the presence of large Young modulus values, we may well wish to solve the SI linear system (or any other such algebraic system arising from an implicit time difference method) by a direct method based somehow on Gaussian elimination. For this purpose we invoke the famous Sherman-Morrison-Woodbury (SMW) formula [68],

$$(\mathbf{A} + \mathbf{Y}\mathbf{Z}^T)^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{Y}(\mathbf{I} + \mathbf{Z}^T\mathbf{A}^{-1}\mathbf{Y})^{-1}\mathbf{Z}^T\mathbf{A}^{-1}. \tag{3.11}$$

In our case we have $\mathbf{Y}$ and $\mathbf{Z}$ defined as above in Eq. (3.10a) and $\mathbf{A} = h^{-2}\mathbf{M} + \mathbf{K}$. The formula is then applied to the right hand side $h^{-2}\mathbf{z}$, requiring one Cholesky decomposition of the sparse $\mathbf{A}$ followed by $s+1$ forward-backward solves. (Note

that the matrix $\mathbf{I} + \mathbf{Z}^T \mathbf{A}^{-1} \mathbf{Y}$ is only $s \times s$, and its inversion is thus assumed cheap.) In our context, the resulting direct inversion method is typically significantly more efficient, even when using SI, than iterative methods such as conjugate gradient.

For linear force models, we note that while Section 3.3.2 is there for didactic purposes, there is no reason to perform a full eigen-decomposition even in the linear case. In practice we thus apply EigenFit as is also for linear forces, noting that the low rank correction matrices $\mathbf{Y}$ and $\mathbf{Z}^T$ are calculated just once.

3.3.5 Practical considerations

In many of the experiments, video clips and figures of this paper we naturally compare our method's performance on a coarse mesh to similar results on a fine mesh. But in practice there will be no detailed fine mesh calculations, or else the purpose of using the coarse mesh in the first place would be nullified. We therefore must be able to predict, to a reasonable degree of assurance, if applying EigenFit on a particular coarse mesh would lead to reasonable approximations for a similar simulation on the finer mesh.

The essential reason why there is hope is the general observation, shared by both the mechanical engineering and the computer graphics communities, that the first s modes, $s \leq 20$, already essentially determine the visual result. This allows for the development of a model reduction technique such as we have just presented. It allows us to always obtain rather acceptable results for linear forces, as demonstrated throughout this paper and especially in the next section.

At the other end there is always an inherent restriction when applying at each time step a technique that is essentially a linearization of a nonlinear problem, especially when the time step is large and there is a lot of deformation action across it. This is what our work as well as all leading others are often doing. A key to our success is to keep the leading modes from tangling up: if a mode on the coarse mesh no longer corresponds to the one on the fine mesh, then of course the ratio of the corresponding eigenvalues becomes meaningless. See Figure 3.4.

Following extensive experimentation, we have arrived at the following criterion. Our algorithm verifies that at least half of the current eigenvectors match to the rest-state eigenvectors. The matching process simply uses the mass orthogonal-

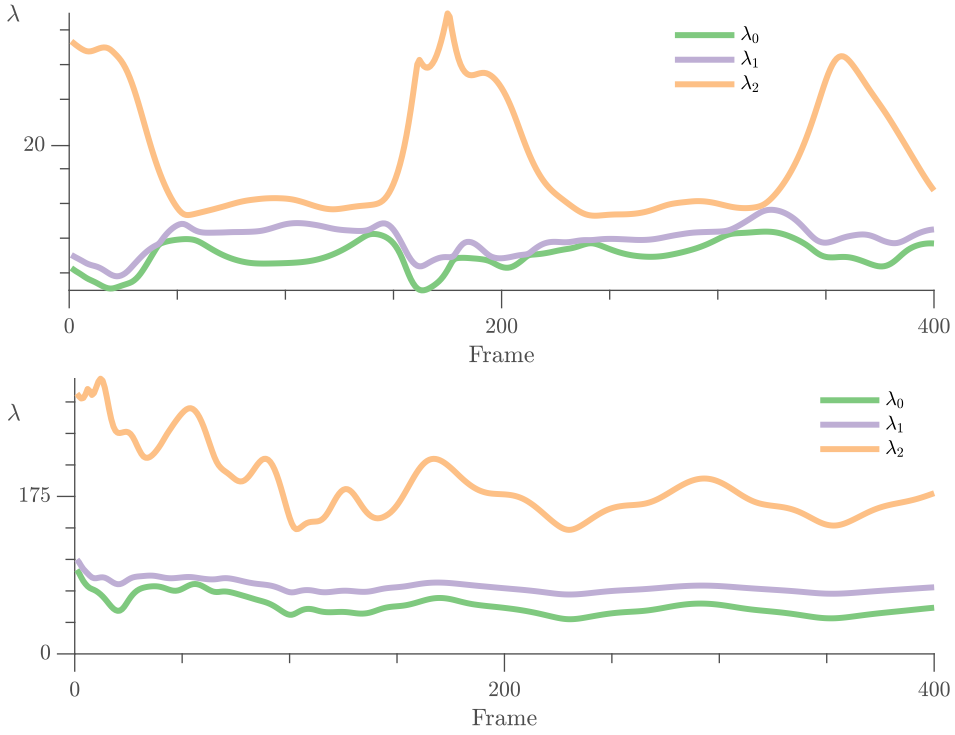


Figure 3.4: Eigenvalues corresponding to the first three leading modes of a bar with a neo-Hookean force. In the top subfigure (corresponding to a softer body $YM = 1.e4Pa$ there is a mode crossing. No such unfortunate behavior is observed for the stiffer object with $YM = 1.e5Pa$. in the bottom figure.

ity from Eqs. (3.1) and (3.2). This is a functional distance measurement between two meshes. Since true orthogonality is not achievable from this matching process, we use a loss $tol = 0.4$ throughout the simulation. Practically, we compute $\mathbf{U}_s^T \mathbf{M} \mathbf{U}_{s,r}$, where $\mathbf{U}_s, \mathbf{U}_{s,r}$ are the s current and rest state eigenvectors, and verify that at least half of the columns contain an entry larger than 0.6. At the first frame, a similar calculation is performed to verify that the resolution of the coarse mesh is capable of simulating the dominating motion of the fine mesh. That is, we calculate $\mathbf{U}_{s,r,c}^T \mathbf{B} \mathbf{M}_f \mathbf{U}_{s,r,f}$, where $\mathbf{U}_{s,r,c}, \mathbf{U}_{s,r,f}$ are the coarse and fine rest state eigenvectors, $\mathbf{M}_f$ is the mass matrix from the fine mesh, and $\mathbf{B}$ is a mapping matrix from the coarse mesh to the fine mesh through barycentric coordinates. Again, we verify

that at least half of the columns contain an entry larger than 0.6. The present simulation is deemed acceptable only if it passes both tests. In short, we use the criterion above to make sure the nonlinearity is not out of hand. Note that there are many types of nonlinearity that can appear in different applications, and different security measures can be used accordingly. For example the alignment process in [91] finds a set of transformation angles by solving an orthogonal Procrustes problem, and this set of angles can also be used as a measurement for nonlinearity.

3.4 Results

In this section we demonstrate the efficacy of EigenFit on homogeneous and heterogeneous material deformable objects of various shapes simulated under both linear and nonlinear forces. For simplicity, for all examples in this section, we use time step size $h = 1e-2$. In the comparisons below, we always use a fine spatial mesh simulated using plain FEM as the ground truth. We compare this fine mesh trajectory against various coarse mesh trajectories simulated by plain FEM, EigenFit or DAC. In this section we use a number of meshes for the reported simulations. Table 3.1 summarizes the mesh information used, while Table 3.2 summarizes our simulation times for each of the coarse mesh, fine mesh, and the EigenFit simulations. We found that for coarse mesh nonlinear material, 20% of the time increase comes from eigendecomposition, while the rest comes from the low rank update. Due to the nature of dynamic simulation, some results are easier to visualize and understand in animated form. For this we refer readers to our supplementary video. In all of the figures, the color code is calibrated using the maximum error from each simulation. We rendered an embedded fine mesh together with a wireframe from the simulated coarse mesh. Note further that many of the examples in this section and the supplementary video involve positional constraints.

In this project, we use nested cages [73] and TetWild [37] to generate meshes for our simulation inputs. We implemented our algorithm in C++ with GAUSS library, and use Spectra, which implemented Lanczos algorithm, for eigenvalue/eigenvector computation. We make our code available at https://github.com/edwinchenyj/GAUSS/tree/release-eigenfit, and the video can be found at https://www.youtube.com/watch?v=aIAKBsT96to.

Table 3.1: List of Meshes used for EigenFit experiments

Mesh ID	#DOF (3× #Vertices)	#Tetrahedrons
Arm fine	15,372	19,378
Arm coarse	7,800	8,948
Armadillo fine	40,539	54,233
Armadillo coarse	4,425	4,902
Bar fine	42,069	72,989
Bar coarse	3,846	5,020
Fert fine	36,324	47,813
Fert coarse	5,385	6,020
Rampant fine	59,382	78,686
Rampant coarse	2,820	3,070
Skater fine	32,169	42,161
Skater coarse	8,394	9,584

3.4.1 Linear hyperelastic constitutive models

While nonlinear models are commonplace in animation tasks nowadays, engineering and fabrication often rely on linear dynamical analysis. Furthermore, many animation techniques rely on underlying linear models which are modified via warping to yield visually acceptable results. Linear force models remain popular due to their simplicity and rapid execution, and have been used in some interesting and varied applications such as acoustic transfer [50]. In our specific context this section is a good place to demonstrate how EigenFit leverages the underlying principle behind numerical coarsening. We can also clearly see how EigenFit differs from DAC and improves upon it.

Under the setting of linear elasticity, the eigensystem is constant. We can perform the full eigendecomposition as described in Section 3.3.1. However, since the high energy modes rarely have observable amplitude, we only carry out a partial eigendecomposition to adjust the first few eigenvalues and eigenvectors, as described in Section 3.3.2. Notice that we only need to perform this decomposition and store $\mathbf{Y}, \mathbf{Z}$ in Eq. (3.10a) once for the entire simulation.

Bar

In this example, we simulate a twisted bar using linear material with constant Young's modulus $YM = 1e5Pa$, Poisson's ratio $\nu = 0.45$, and IM with $\alpha = 0, \beta = 0.01$ in Eq. (4.7). In this case, the fine mesh has $N_f = 14,023$ and the coarse mesh has $N_c = 1,282$ free vertices. In the EigenFit algorithm, we pick $m = s = 10$ modes to match from coarse mesh to the fine mesh. We track a corner point of the bar and plot the trajectories and errors in Figure 3.5. We also show the simulation results at $t = 2.9sec$ with different view angles to provide comparison with other methods.

Note that the Euclidean error of EigenFit at the corner point is consistently lower than those of the plain FEM and DAC. Interestingly, if we only look at the y trajectory of the corner point in Figure 3.5(c), DAC does a good job at matching the ground truth. By looking at the first few eigen-deformations in Figure 3.6, we can see why this is the case. The first eigen-deformation is a bending mode that affects the y coordinate only; see Figure 3.6a. Since DAC uses the first eigenvalue ratio to adjust the Young modulus, it fixes the stiffening effect in this motion. However, since we are simulating a twisted bar, there are other deformation modes that could not be captured by such eigen-deformation alone. In our EigenFit algorithm, the 3rd eigenvalue ratio is used to compensate the stiffening effect in the rotational motion; see Figure 3.6c. This suggests that EigenFit can be viewed as a high-order improvement to DAC.

Armadillo

Our algorithm can of course handle more complex geometries than a rectangular bar. In this example we simulate an armadillo using Armadillo meshes described in Table 3.1. Figure 3.1 demonstrates that EigenFit can match the corresponding modes to the correct frequency in the fine mesh. This results in a consistent simulation trajectory across different resolutions.

Summary for linear forces: We have carried out dozens of additional simulations for many different deformable objects under linear force. For all examples tried EigenFit performs very well indeed. For DAC, the observations above also hold consistently: it performs well (i.e., comparable to EigenFit) if there is one domi-

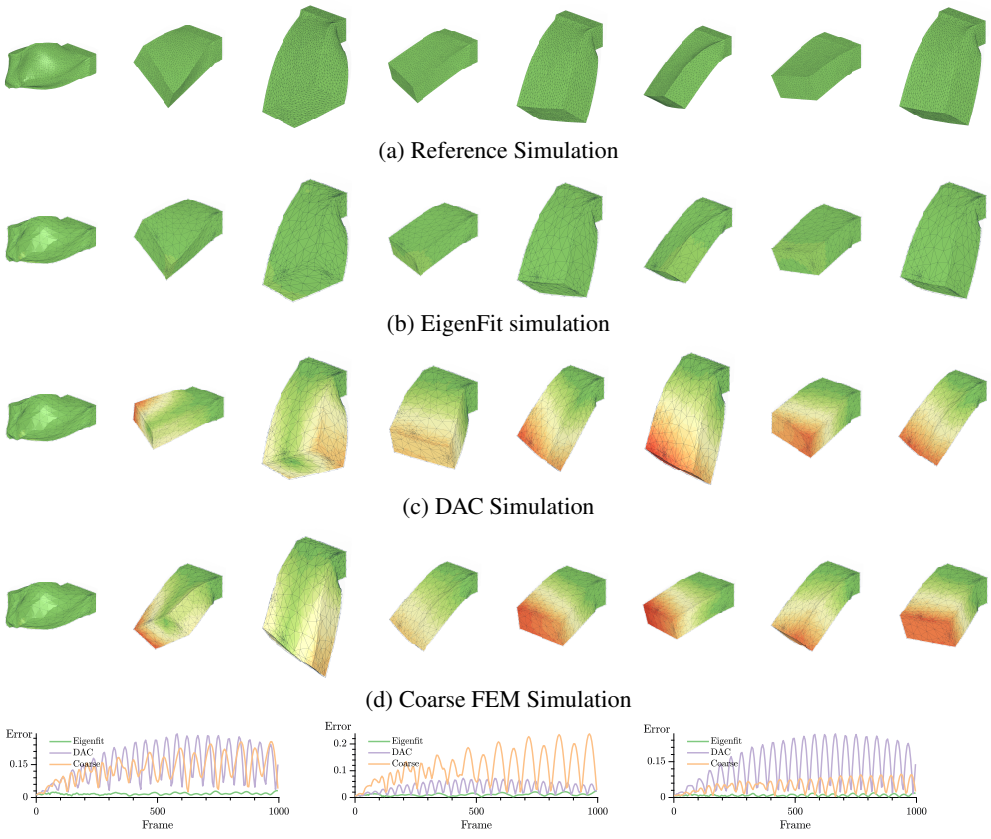
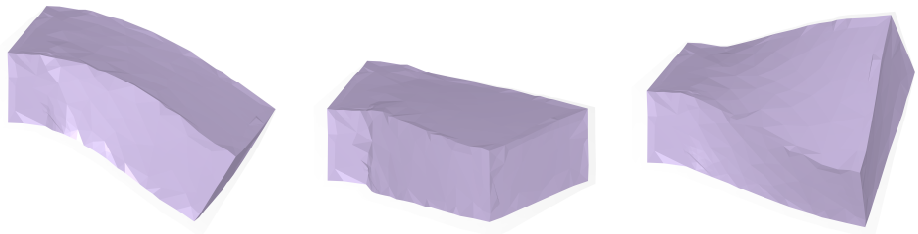


Figure 3.5: Animation shots for a twisted bar under a linear force, progressing in time from left to right. The fine mesh animation (top row) is faithfully reproduced by EigenFit on the coarse mesh (2nd row), displaying robustness to irregular meshes. On the other hand, DAC (3rd row) and the raw FEM coarse mesh simulations (bottom row) produce significant, visible disagreements with the fine mesh.



(a) Bar 1st mode **(b)** Bar 2nd mode **(c)** Bar 3rd mode

Figure 3.6: The first three modes of the bar example in Figure 3.5.

nating mode, but when several different dominating modes are present DAC falls behind.

3.4.2 Mildly nonlinear elastodynamics models

To further demonstrate the capacity of EigenFit, we have applied it for nonlinear material force models, including as rigid as possible (ARAP) [15] and neo-Hookean.

We have used ARAP energy, a popular nonlinear energy model in computer graphics, to simulate the Armadillo and Skater meshes with stiffness parameter 1e6Pa, simulated with the SI integrator. We fixed and shook the feet of both objects for 30 frames and observed the resulting oscillation afterwards. For EigenFit, we used 10 modes and applied the 10 calculated ratios at every frame, as described in Section 3.3.3. Although the result is not as impressively perfect as in the linear case, EigenFit still tracks the fine mesh motion much better than the regular coarse FEM simulation; see Figure 3.7 and Figure 3.9, respectively. In particular, notice that the position error and velocity are oscillatory and out of phase in the coarse FEM simulation (yellow line in the error plots), which means that its dominating motion has the wrong frequency. EigenFit (green line in the error plots) matches the frequency of this motion, and thus has a much lower error.

In the armadillo mesh, we also compared to DAC. Notice that EigenFit also performs better than DAC in this case, and it is not hard to see why by looking at the first 3 dominating modes. From our EigenFit calculation we found the domi-

nating three modes have eigenvalue ratios $r_1 = 0.536$, $r_2 = 0.711$, and $r_3 = 0.675$, and the corresponding motions are forward bending (Figure 3.7(c)), side shifting (Figure 3.7(d)), and slight twisting about the z-axis (Figure 3.7(e)). Effectively, DAC applied one single ratio, 0.536, to all of these motions, whereas EigenFit correctly matched their corresponding ratios.

Dynamic constraint

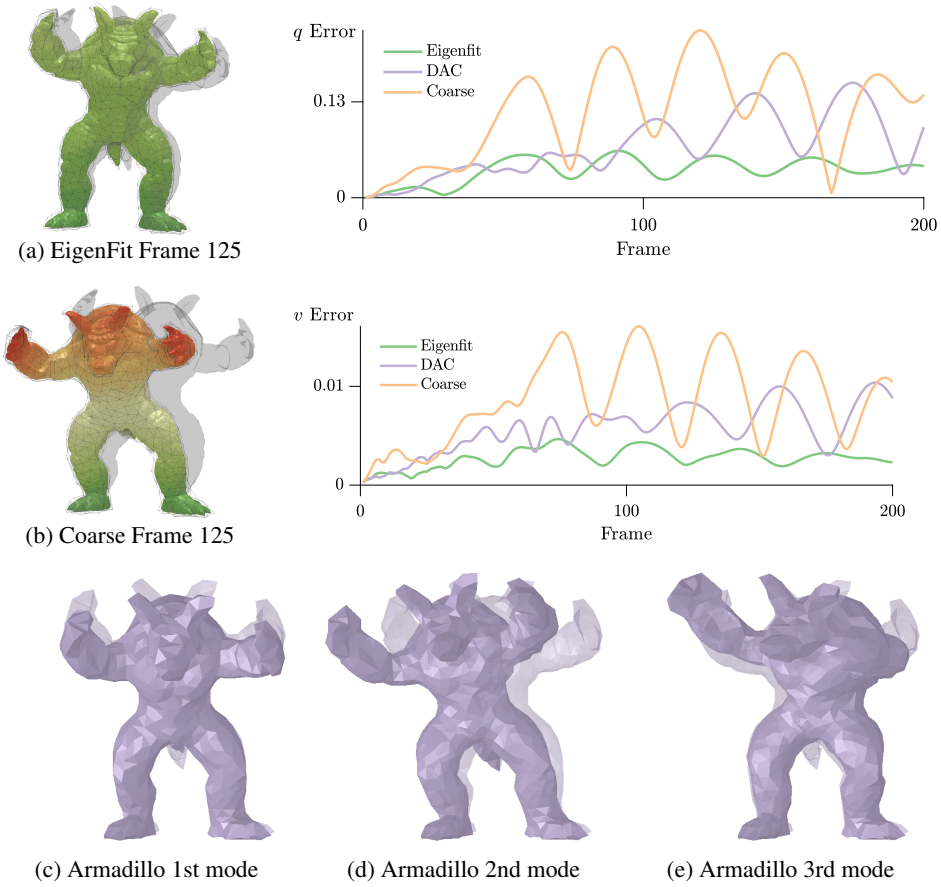
EigenFit also works when a constraint changes during the simulation. In Figure 3.8, we simulated a shaken armadillo, and released one leg during the simulation. The EigenFit ratios are recalculated when the boundary conditions are change. The plots show that using EigenFit reduced the error significantly.

Summary for mildly nonlinear scenarios: The results in Figures 3.7–3.9, as well as those in Figures 3.10–3.12, all indicate that when the nonlinear effect is sufficiently moderate so that a quasi-linearization at the beginning of each time step captures the essence of what happens throughout it, the conclusions drawn before for linear forces can be extended, albeit in an imperfect sense. We do observe mild mismatches, but both EigenFit and DAC are expected to perform better than regular FEM, and situations where DAC is worse than EigenFit arise here as well.

3.4.3 Large deformation for nonlinear numerical material

For more general nonlinear scenarios it is important to understand that there is not a panacea. The methods proposed in the literature all rely on some localization that may not always hold, and none of them performs satisfactorily in the large context. Figure 3.13 is a case in point, where both EigenFit and DAC give poorer results than the regular coarse mesh FEM.

In practice, of course the utility of any method of the sort considered here depends on not having to simulate the "ground truth" fine mesh trajectory. If the ultimate fine mesh is much finer than the coarse mesh (hence directly simulating on it could be prohibitively expensive), then the simplest cure may be to make the coarse mesh finer but still not as fine as the finest mesh. This is a common practice in scientific computing; see, e.g., [87]. The condition devised in Section 3.3.5 goes



(a) EigenFit Frame 125

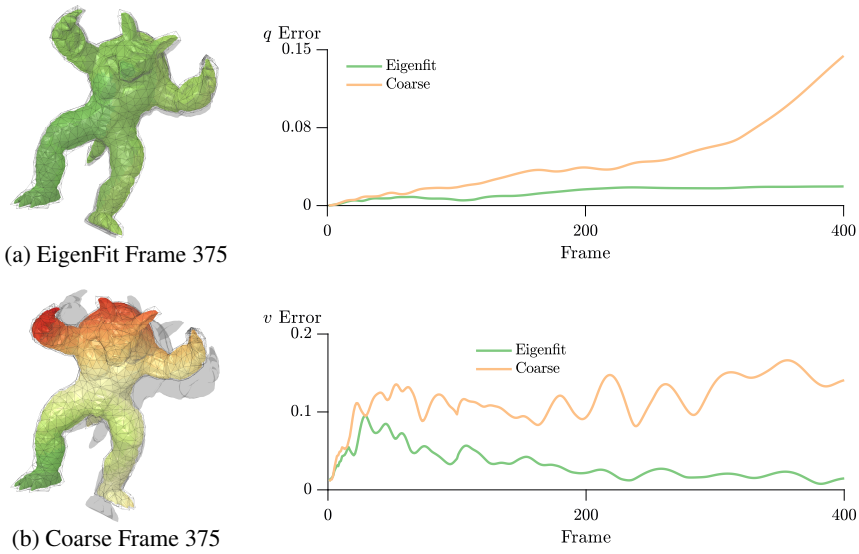
(b) Coarse Frame 125

(c) Armadillo 1st mode

(d) Armadillo 2nd mode

(e) Armadillo 3rd mode

Figure 3.7: ARAP armadillo with homogeneous material. The SI integrator was used to simulate 200 frames. Grey silhouette shows the reference simulation from a fine mesh. The two plots are maximum position error and velocity error.



(a) EigenFit Frame 375

(b) Coarse Frame 375

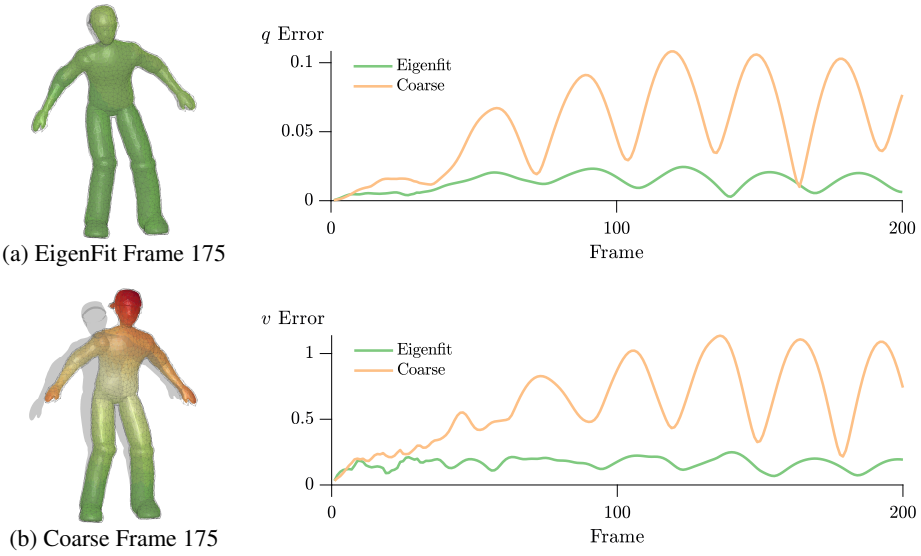
Figure 3.8: ARAP Armadillo mesh with changing constraints. Color bar was calibrated to fit the max error. The two plots are max position error and velocity error.

towards ensuring that EigenFit is not applied where its chances to perform well are deemed too low.

Remaining still with homogeneous objects for simplicity sake, we also observe that for the same forces applied to the same object except that the Young modulus varies, the error is smaller the stiffer the object gets; see Table 3.3. It's for rather soft bodies exhibiting large deformation where a finer coarse mesh is particularly desirable.

3.4.4 Heterogeneous deformable objects

Thus far in this narration we have restricted our attention to homogeneous objects in order to concentrate on the many other aspects of the numerical stiffness problem and to enable direct comparison to DAC. However, of course there are many deformable objects in practice, both natural and designed, where the Young modulus is a non-constant function over the object's domain; see Figure 3.14.



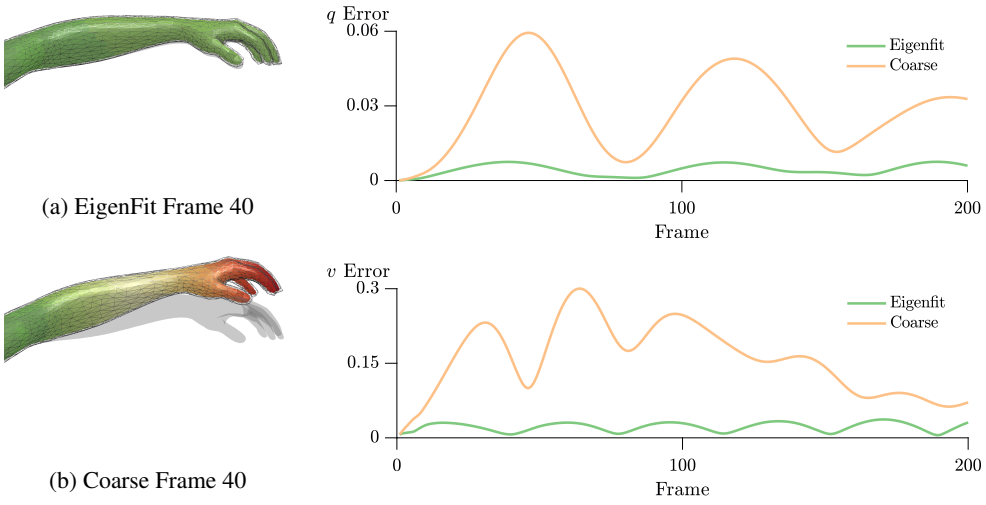
(a) EigenFit Frame 175

(b) Coarse Frame 175

Figure 3.9: ARAP skater with homogeneous material. The SI integrator was used to simulate 200 frames. Grey silhouette shows the reference simulation from a fine mesh. The two plots are maximum position error and velocity error.

Table 3.2: Relative Simulation Time. A set of coarse and fine meshes were simulated. We report the CPU time, with the relative simulation time given in brackets, using the fine mesh simulation time as the reference.

	Coarse	EigenFit	Fine
Arm (ARAP)	219 (0.26)	444 (0.53)	839
Armadillo (Linear)	224 (0.16)	300 (0.22)	1,387
Armadillo (ARAP)	198 (0.08)	546 (0.23)	2,399
Bar (Linear)	335 (0.20)	386(0.22)	1,683
Fert (Neo-Hookean)	633(0.15)	1,914(0.44)	4,331
Rampant (ARAP)	279(0.06)	634 (0.14)	4,496
Skater (ARAP)	298 (0.23)	865 (0.65)	1,311



(a) EigenFit Frame 40

(b) Coarse Frame 40

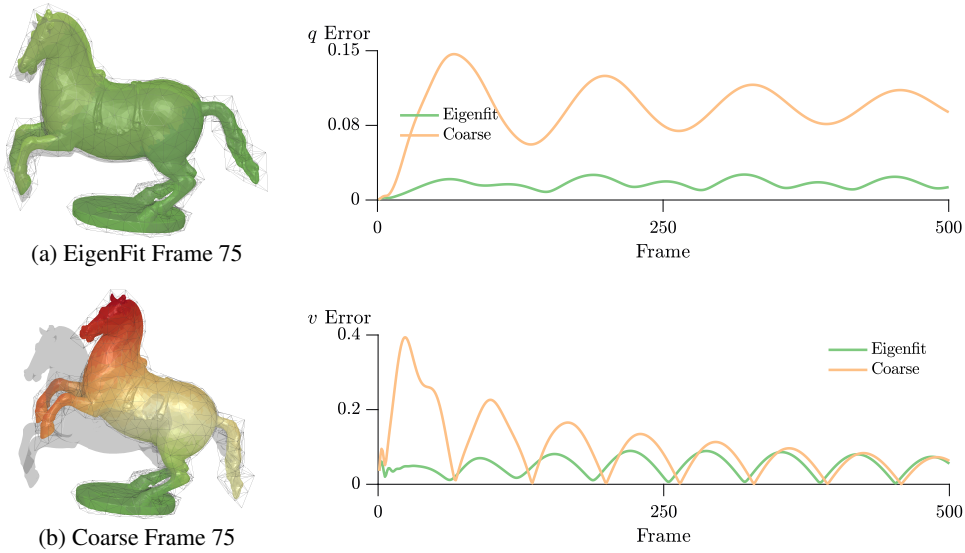
Figure 3.10: neo-Hookean arm with heterogeneous material. Color scheme is calibrated to fit the max error. EigenFit and regular coarse FEM are displayed. The error plots are for max position and max velocity error.

Table 3.3: Error and Stiffness

Stiffness Parameter (Pa)	1e4	1e5	5e5	1e6	1e7
Arm (ARAP)	0.91	1.39	0.615	0.611	0.438
Armadillo (ARAP)	0.838	0.755	0.622	0.307	0.192
Bar (Neo-Hookean)	1.431	0.7281	0.22	0.247	0.292
Hand (ARAP)	1.35	1.04	0.991	0.977	1.09
Skater (ARAP)	0.92	0.594	0.316	0.273	0.182

To find the distributed parameter function that is Young's modulus in such a case, especially for a natural object, can become significantly more difficult [90]. However, the application of EigenFit is independent of finding this function for different meshes. Thus, assuming the availability of Young's modulus and an adequate coarse spatial mesh, EigenFit applies to heterogeneous objects as readily as to homogeneous ones!

Furthermore, EigenFit naturally handles relatively stiff objects where motion



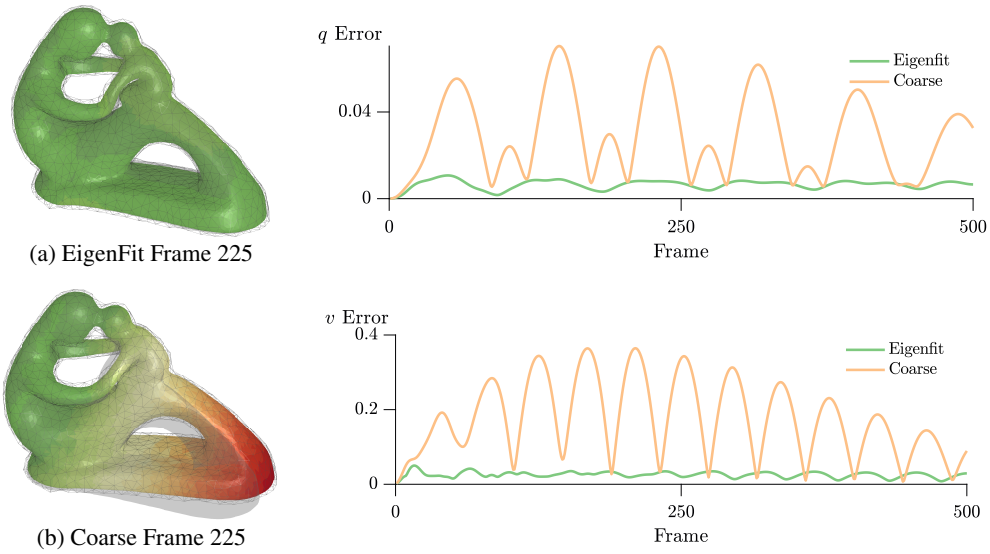
(a) EigenFit Frame 75

(b) Coarse Frame 75

Figure 3.11: ARAP Rampant mesh with heterogeneous material. Color scheme is calibrated to fit the max error, and EigenFit is compared to regular coarse FEM. The two plots are for max position error and max velocity error.

is generated through local softer joints, as demonstrated to various degrees in Figure 3.14. In such cases the leading modes corresponding to the softer joints can be captured well, and larger deformations in the stiffer part of the object can be successfully followed on the coarse mesh.

In Figures 3.10–3.12 we used heterogeneous material on the Arm, Rampant, and Fert meshes. Color coding in Figure 3.14 shows how the stiffness parameter varies spatially in the examples we used; darker color means stiffer, and lighter color is softer. For the Arm mesh, the peak stiffness is 1e8, and 2e4 at the minimum. For the Rampant mesh, the peak stiffness is 1e8, and 2e6 for the minimum. For the Fert mesh the peak stiffness is 1e9 and 1e5 at the minimum. Observe from these figures and the supplementary video that with nonlinear heterogeneous constitutive material, EigenFit improves the simulation results for non-mild deformations from the coarse FEM mesh.



(a) EigenFit Frame 225

(b) Coarse Frame 225

Figure 3.12: neo-Hookean Fert mesh with heterogeneous material. Color bar calibrated to fit the max error. The two plots are max position error and velocity error.

3.5 Conclusions, limitations and future

Mitigating numerical stiffening for coarsened meshes will open the door for elastodynamic simulation in many domains not previously possible. Automated fabrication design requires outer optimization loops that can call expensive FE simulations hundreds of times about each design sample. Simply decreasing the resolution needed for sufficient accuracy would open the door to much faster optimization. Furthermore, as geometric design parameters change so does the computational mesh. If changes in the meshing are allowed to change the effective material stiffness of the simulations, then the entire, exceedingly expensive design optimization is invalidated. Current optimization design tools, e.g., in the automotive industry, apply mesh-warping techniques in these settings that deform the rest mesh in attempt to maintain mesh consistency over changing design parameters. However, large changes in the design-space necessarily require large and often abrupt changes in mesh resolution, again forcing tools to conservatively ap-

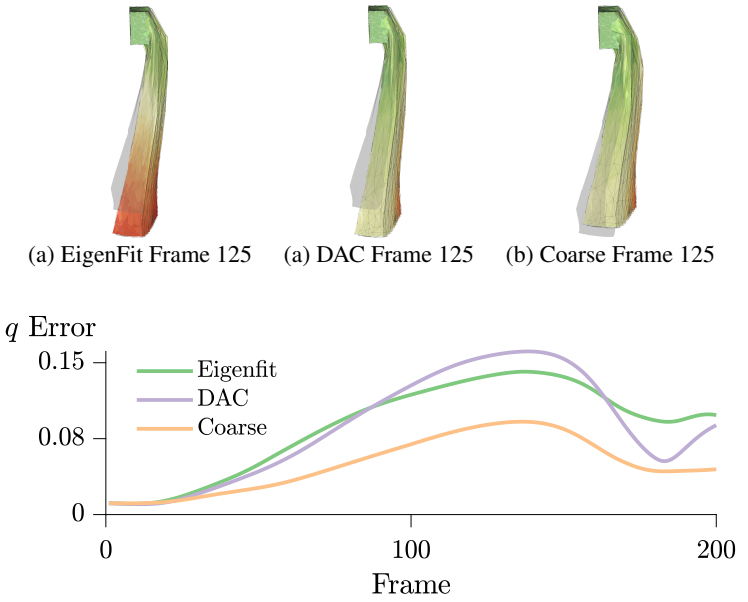


Figure 3.13: Under large deformation both EigenFit and DAC may perform poorly. Here both methods fail to improve on the original coarse FEM simulation. This is due to the fact that the underlying frequency matching (for DAC) and mode matching (for EigenFit) conditions fail to hold.

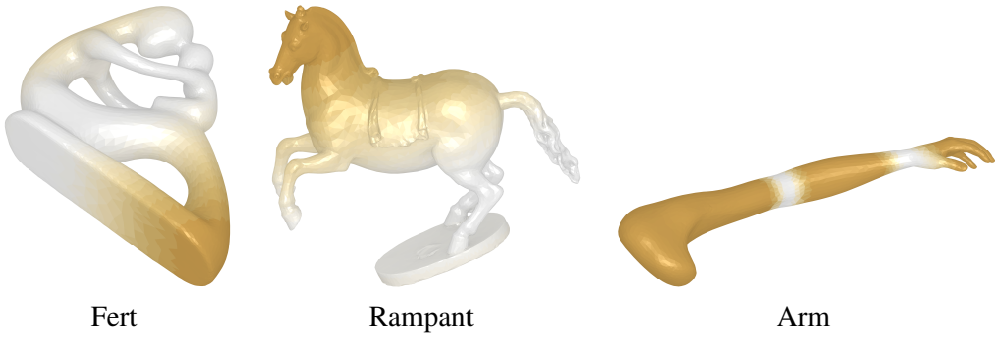


Figure 3.14: Examples of heterogeneous objects: lighter color corresponds to lower Young modulus.

ply high-resolution meshes.

In this work we have assumed that the material constitutive model is given and have not attempted to change it, e.g., through homogenization techniques. When changing the spatial mesh resolution the apparent physical behavior of the simulated motion of a flexible object may change. We have presented, analyzed and demonstrated a method that significantly reduces this unwanted mesh dependence, making the simulation consistent across mesh resolutions. The method involves matching the leading eigenvalues of a given coarse mesh with those of a fine reference mesh at rest. This necessitates a partial spectral decomposition, which for nonlinear constitutive laws must be carried out at each time step. For this we have proposed and implemented a model reduction method, EigenFit, where the essential work is carried out in a small subspace of the eigenmodes. We demonstrated EigenFit in action on a range of different meshes for both homogeneous and heterogeneous material objects. In addition to the figures in this paper we refer the reader to the supplementary video, where the gained motion consistency is clearly demonstrated.

We have applied EigenFit to a class of heterogeneous and nonlinear materials. To our knowledge, the dynamic aspect of numerical stiffening and numerical homogenization has not been discussed in detail in the literature, and our assumption that the coarse mesh is given and is "sufficiently fine" is not out of place in many heterogeneous simulation applications (e.g., for animating a teddy bear or a plant).

Notice that "large deformation" in the present context is the amount of deformation that would modify the ordering of the eigenmodes significantly. It is possible for certain mesh configurations to have visually small deformation that still leads to a significant degree of mode crossing. This is indeed the case with the Hand mesh listed in Table 3.3, where a cluster of eigenvalues/eigenmodes corresponding to the motion of each finger can easily exhibit mode crossing, making the EigenFit mode matching effort ineffective.

For general nonlinear forces with large deformations applied to heterogeneous material an effort is required, extending [17], to decide which coarse mesh is "sufficiently fine" to enable a reasonable treatment of the numerical stiffening phenomenon. To that end a practical direction is to start with a coarse mesh as in [17] and gradually refine it for a representative scenario until it is deemed fine enough

in the present context. Then use the obtained mesh as the new “coarse mesh” to which EigenFit can be applied.

Chapter 4

Exponential Rosenbrock-Euler Integrators for Elastodynamic sImulation

4.1 Exponential integrators

We begin by briefly describing exponential integrators as needed for our purposes. Much more is discussed in [35] and [63].

Consider the autonomous ODE

$$\dot{\mathbf{u}}(t) = \mathbf{b}(\mathbf{u}(t)),\ \mathbf{u}(0) = \mathbf{u}_0. \tag{4.1}$$

Exponential integrators are based on splitting $\mathbf{b}(\mathbf{u})$ as

$$\mathbf{b}(\mathbf{u}) = J\mathbf{u} + \mathbf{c}(\mathbf{u}),\ \text{where } J = \frac{\partial \mathbf{b}}{\partial \mathbf{u}}(\mathbf{u}_0),$$

and employing the variation-of-constants formula

$$\mathbf{u}(t) = \exp(tJ)\mathbf{u}(0) + \int_0^t \exp((t-s)J)\mathbf{c}(\mathbf{u}(\zeta))\mathrm{d}\zeta. \tag{4.2}$$

By approximating the integral in Eq. (4.2), we can obtain various exponential inte-

grators.

Exponential Rosenbrock-Euler (ERE) Method

Let us write Eq. (4.1) at $\mathbf{u}_n = \mathbf{u}(t_n)$ as

$$\begin{aligned} \dot{\mathbf{u}}(t) &= J_n \mathbf{u}(t) + \mathbf{c}_n(\mathbf{u}(t)), \\ J_n &= \frac{\partial \mathbf{b}}{\partial \mathbf{u}}(\mathbf{u}_n),\ \mathbf{c}_n(\mathbf{u}) = \mathbf{b}(\mathbf{u}_n) - J_n \mathbf{u}(t). \end{aligned} \tag{4.3}$$

We then use Eq. (4.2) to integrate Eq. (4.3) from t_n to $t_{n+1} = t_n + h$, and restart the linearization process at t_{n+1}. The simplest method of this form can be constructed by fixing J_n and $\mathbf{c}_n(\mathbf{u}(\zeta))$ at $\zeta = t_n$, enabling exact integration in Eq. (4.2) and thus leading to the exponential Rosenbrock-Euler (ERE) method

$$\begin{aligned} \mathbf{u}_{n+1} &= \exp(hJ_n)\mathbf{u}_n + h\phi_1(hJ_n)\mathbf{c}_n(\mathbf{u}_n) \\ &= \mathbf{u}_n + h\phi_1(hJ_n)\mathbf{b}(\mathbf{u}_n) \end{aligned} \tag{4.4}$$

with $\phi_1(z) = z^{-1}(\exp(z) - 1)$.

Henceforth we will use this method (and not higher order Rosenbrock[1]) because we are interested in a qualitatively correct, inexpensive integrator and are less focused on very small point-wise errors (unlike [62], for instance). On the other hand, we have found it necessary, for the applications considered here, to refresh J_n and re-evaluate its exponential at each time step. Furthermore, we have considered and discarded use of conservative average vector field (AVF) methods [89] in the present context, because they also lead to potentially difficult large-step nonlinear systems. In our context, where assembling all the forces to form $\mathbf{b}(\mathbf{u})$ is the major expense at a given time step, the use of ERE allows us to concentrate only on the evaluation of the exponential matrix function times a vector.

4.1.1 Implementation of ERE

The expression Eq. (4.4) seems computationally attractive as it only requires one matrix function evaluation. However, the evaluation of the function $\phi_1(z)$ suffers

[1] Higher order methods require additional matrix exponential evaluations in the internal stages at each time step, similar to RK methods. See, e.g., [35] for more detail.

from numerical instability due to cancellation error when $z \approx 0$ [40, 67]. In elastodynamics simulations, this could happen when the elastic waves are traveling on different time scales. To avoid computing ϕ_1, we follow [2, 72] and rewrite Eq. (4.4) as

$$\begin{aligned} \mathbf{u}_{n+1} &= \begin{bmatrix} I_N & 0_{N\times 1} \end{bmatrix} \exp(hA)\, \tilde{\mathbf{u}}_n, \quad \text{where} \\ A &= \begin{bmatrix} J_n & \mathbf{c}_n(\mathbf{u}_n) \\ 0_{1\times N} & 0 \end{bmatrix}, \quad \tilde{\mathbf{u}}_n = \begin{bmatrix} \mathbf{u}_n \\ 1 \end{bmatrix}, \end{aligned} \tag{4.5}$$

which only involves one product of a matrix exponential of the slightly larger matrix with a vector.

Computing the action of the matrix exponential

When computing the product $\exp(hA)\mathbf{u}$ for a large but sparse matrix A as in Eq. (4.5), it is important to avoid forming the full matrix exponential explicitly. Some of the most efficient algorithms implement sub-stepping (or scaling) of the form

$$\exp(hA)\mathbf{u} = \exp(\delta t_1 A)\exp(\delta t_2 A)\cdots\exp(\delta t_\tau A)\mathbf{u}, \ \sum_{i=1}^{\tau} \delta t_i = h \tag{4.6}$$

using one of the following methods:

- truncated Taylor series [2],
- Krylov subspace methods [77],
- Leja approximations [14].

The *number of stages* τ or the substeps δt_i are chosen to optimize the cost and accuracy based on desired tolerance and error estimation. See [63] for more detail and further justification. The costs of all three methods grow with the matrix norm $\|A\|$, which in our context is majorized by the system stiffness-density ratio $\|M^{-1}K\|$. For the present type of application, we found that the Krylov subspace based algorithm from [77] performed best, so we have subsequently used their Matlab toolbox for all the calculations involving matrix exponentials reported here. At each substep δt_i, this algorithm uses the Arnoldi process to project the

exponential operator onto a small Krylov subspace, where a small matrix exponential is calculated, and projects the result back. Further details of the algorithm are omitted here, as these are not contributions of the present paper and may be found in [77].

Let us denote the *material* stiffness-density ratio

$$k/m = \text{Young}'\text{s modulus}/\text{density},$$

measured in $(m/sec)^2$. This quantity relates to the natural frequency of the material at the rest shape. Figure 4.1 shows the relationship between k/m and the cost of computing the action of the matrix exponential with the Matlab package `expv` from `expokit` by [77]. (a) Number of substeps τ grows with the stiffness-density ratio k/m (lines with different color) and matrix size N (x-axis). (b) s grows (roughly) linearly with N, leading to super-linear growth in total runtime. (c) The running time for computing the action of the matrix exponential depends on k/m. Nevertheless, soft tissues and soft engineering materials of much current interest can be simulated effectively by ERE. We further make the following observations:

- Substepping of the form Eq. (4.6) is equivalent to time-stepping in a linear ODE with system matrix A. Hence, the growth in number of substeps τ and ERE cost is similar to what is obtained from the stringent time step requirement for classical explicit methods for ODE systems with fast traveling waves. However, the cost of substepping is cheap (e.g., involving an Arnoldi iteration and exponentiation of a small matrix when using a Krylov subspace), whereas small time steps with explicit methods are expensive (due to force calculations through FEM assembly from per element contribution). More comparisons with explicit methods are discussed in Section 4.2.2.

- ERE can become costly when $\|M^{-1}K\|$ is very large, so it is only suitable for materials that are not too stiff. On the other hand, when $\|M^{-1}K\|$ is large, regular implicit methods (e.g., backward Euler or implicit midpoint) encounter numerical difficulty from solving nonlinear systems at each time step. In particular, one can implement a more robust version of Newton's method by mixing it with gradient descent or damping the step [53, 58], but

such implementation might increase cost due to loss of quadratic convergence and involves picking ad hoc parameters.

- On a positive note, according to experimental measurements from [1, 46, 52], there is a wide range of soft tissues of much current interest that have the material properties in the range where ERE is efficient. In addition, soft engineering materials that exhibit highly dynamical behavior can also be simulated effectively by ERE.

Stability of exponential integrators

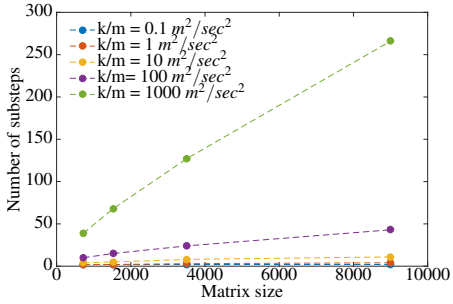
For ERE to be stable, a necessary condition is that all the eigenvalues of the Jacobian matrix J must have negative real part at every time step [35], which corresponds to the damping in the system. The elastodynamic system that is being simulated has this property since elastic objects are naturally damped due to internal friction. In our implementation we have introduced a minimal amount of damping using the Rayleigh damping model

$$f_{\text{dmp}}(\mathbf{q}, \mathbf{v}) = (\alpha M + \beta K_0)\mathbf{v}, \tag{4.7}$$

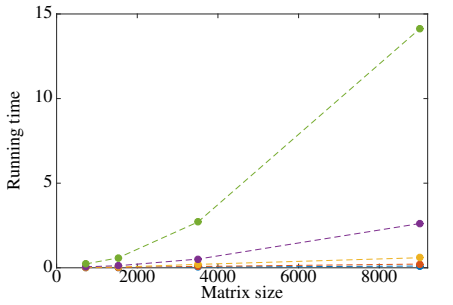
(i.e., the damping matrix is $D = \alpha M + \beta K_0$), where K_0 is the (SPD) stiffness matrix at the initial state and $\alpha, \beta > 0$. In our experience, although the simple Rayleigh damping model introduced in Eq. (4.7) does not always guarantee stability, it works for a wide range of examples. In particular, the stability of ERE suffers with large $\|M^{-1}K\|$, and under large deformation, when $\mathbf{g}(\mathbf{q}(t))$ in Eq. (2.6) grows larger. In the next section we show that ERE is stable for a wide range of soft materials that arise in computer graphics.

4.2 Results

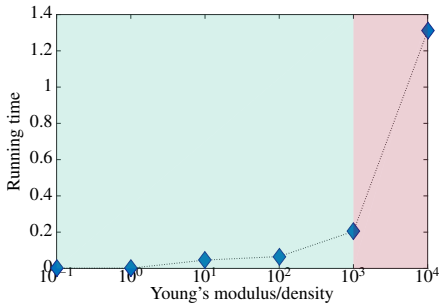
In this section we demonstrate the performance of ERE on (i) a linear mass-spring system with linear density 0.1kg/m; and (ii) a nonlinear neo-Hookean material model with density 1000kg/m^3 and Poisson ratio 0.48, discretized using linear displacement tetrahedral elements. In both cases we employ the Rayleigh damping



(a) Number of substeps τ vs. matrix size N with different stiffness-density ratio k/m for neo-Hookean material under small strain.



(b) Running time of `expv` vs. matrix size N with different k/m values for neo-Hookean material under small strain.



(c) Running time of `expv` vs. k/m for neo-Hookean material under small strain for a system of size 738.

Figure 4.1: Cost of calculating the matrix exponential

model (4.7) with $\alpha = 0.01$ and $\beta = 0.01$.[2]

For each simulation we choose a range of linear stiffness K_l or Young's modulus YM[3] and step size h to demonstrate performance. We emphasize that ERE is not limited by the above systems: it can be applied to the dynamic system of any constitutive model semi-discretized in the form of Eq. (2.3).

All reported times t and step sizes h are in terms of seconds (sec). For reading clarity we occasionally omit this unit in the sequel.

4.2.1 Cost of ERE under large deformation

As mentioned in Section 4.1.1, the number of substeps τ in evaluating the matrix exponential inside ERE depends on $\|M^{-1}K\|$, which depends on material properties. Under large deformation with a nonlinear material, $\|M^{-1}K\|$ can also increase due to stiffening, and thus plague the performance of ERE. In the present example, we simulated a neo-Hookean cylinder under both compression and stretching (Figure 4.2) and recorded the number of substeps τ used by `expv` within ERE in Table 4.1. We observe that τ increases under stiffening only by less than a factor of 3 at 50% compression and by less then a factor of 2 at 100% stretch. For materials such as soft tissue and organs, the region of elastic deformation is less then 20%, meaning the cost of ERE will not change much for practical use.

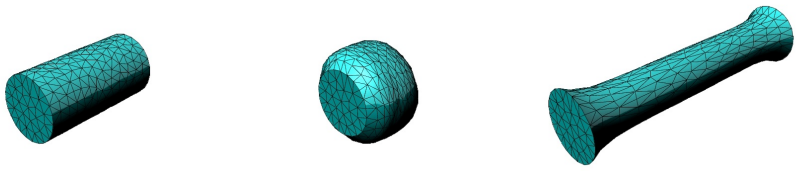


Figure 4.2: Compressing and stretching a neo-Hookean cylinder (2383 tetrahedrons)

[2]Note that our linear mass-spring system is only linear in stretch, and a nonlinear system arises because of rotation, in contrast to the mass-spring system in [60].

[3]Since the material is nonlinear, the stiffness parameters are chosen for the rest state only.

Table 4.1: Number of substeps τ used by `expv` in ERE for compressing and stretching a neo-Hookean cylinder (2383 tetrahedra) with different k/m at step size $h = 1/30$ at 2%, 50% strain along the cylinder axis. For stretching, 100% stretch is also tested.

	k/m	$s_{2\%}$	$s_{50\%}$	$s_{100\%}$
compression	$1e3$	2	3	
	$1e4$	5	7	
	$1e5$	12	23	
	$1e6$	48	123	
stretch	$1e3$	2	2	2
	$1e4$	4	4	5
	$1e5$	12	12	13
	$1e6$	47	51	58

4.2.2 Comparison with RK4

For soft material that is not extremely stiff, explicit methods such as the classical fourth order RK, denoted RK4, can be stable and efficient because the time step constraint is less stringent. To see that ERE compares favorably for soft materials nonetheless, we first test ERE against RK. In this comparison, we simulate one meter of elastic rope as 50 springs in sequence for 2 seconds and list the stable step size h_{rk} and the total running time t^c_{rk} for RK4 with different stiffness-density ratios. Results are collected in Table 4.2.[4] While ERE introduces with the listed step size some positional error ε_{ere}, the simulation results are qualitatively correct and the method remains stable for all stiffness parameter values at step size $h_{ere} = 1/60$ and has shorter total running time t^c_{ere}. The results demonstrate that RK4 is much more expensive (twice to 30 times). This is because calculating the force model (mass-spring and Rayleigh damping in this case) for small time steps is expensive, whereas the substepping for the matrix exponential in ERE is much cheaper.

[4]The solutions from RK4 are used as "ground truth" to calculate the error for ERE, since RK4 is a fourth order method that is necessarily evaluated using a much smaller time step.

Table 4.2: ERE and RK4 step sizes h and total running time t^c (measured in CPU seconds) for simulating an elastic rope with different linear stiffness $K_l(N/m)$ for 2 seconds. Positional error ε is measured in meters.

K_l	h_{ere}	t^c_{ere}	$\max \|\varepsilon_{ere}\|$	h_{rk}	t^c_{rk}
$1e0$	$1/60$	5.2278	0.0720	$1/180$	9.7558
$1e1$	$1/60$	4.9717	0.0210	$1/420$	19.0327
$1e2$	$1/60$	6.1674	0.0511	$1/1080$	41.7449
$1e3$	$1/60$	6.6823	0.1424	$1/4680$	179.1033

4.2.3 Comparison with semi-implicit method and implicit methods

Solving nonlinear equations

In the following examples, we used Newton's method for the nonlinear equations arising in backward Euler (BE) and implicit midpoint (IM) integrators. The step is successful if the iteration's residual is less then $\varepsilon_n = 10^{-6}$. If the residual does not decrease after 3 iterations, then we locally reduce the time step by half and repeat the procedure over two half-steps. This is a standard procedure in numerical ODE practice (see, e.g., [3]) that is a simple instance of a numerical continuation method, using smaller steps to approximate the path from t_n to $t_n + h$. Specifically, it leverages the physical structure of the problem because by reducing the time step the starting guess $\mathbf{u}_{n+1/2}$ is often closer to the desired $\mathbf{u}_{n+1}$. This procedure still guarantees quadratic convergence for small substep problems, whereas implementing linear search and Hessian fix [58] could reduce the convergence speed.

It is also possible to apply the semi-implicit approximation (i.e., a single Newton iteration) to IM rather than to BE. The cost per step is then the same as SI. However, we have observed that stability of the implicit midpoint method can be highly effected under this early termination. A similar effect was observed in [54]. Hence we do not consider this variant further.

1D Mass-spring systems

We first use the same rope simulation as described in Section 4.2.2 to compare the performance of ERE against the semi-implicit (SI) method and BE method (which both introduce artificial damping), as well as the IM method (which does not); see Figure 4.3. All the methods were run at step size 1/60 and remained stable for all stiffness parameters tested. Figure 4.3 shows that ERE is cheaper compared to the fully implicit methods (BE and IM) while having good accuracy (Figure 4.3b). Figure 4.3c shows energy profiles of different methods for the simulation with linear stiffness $K_l = 100N/m$. ERE is clearly better then BE and SI because less artificial damping is introduced.

2D Mass-spring systems

In this example we simulate a piece of cloth with 121 particles (11×11) and simple stretching and shearing springs. The cloth is dropped with four corners fixed, and then quickly shaken at $t = 1$; see Figure 4.5. In Figure 4.4 we compare ERE with SI, BE, and IM, all with step size $h = 1/60$, in terms of running time and maximum position error (RK4 with small time step serves as ground truth). Figure 4.5 depicts the frames at $t = 1.7$, after the shaking event at $t = 1$. The cloth looks more damped with SI and BE, whereas the ground truth is more responsive and dynamic. The ERE solution is only slightly damped, and it remains more faithful to the true solution. Once the SI step size is reduced down to $h_{si} = 1/120$, the simulation error and visual impressions are similar to ERE with step size $h_{ere} = 1/60$. This is in keeping with the fact that the damping introduced by SI reduces for a smaller step size. However, reducing the step size also increases the total running time (Figure 4.4a).

In the next example we make the same cloth mesh with linear stiffness $K_l = 100N/m$ collide with a sphere. The collision is resolved by projecting each particle onto the closest point on the sphere. Figure 4.6 depicts the state of the cloth after the collision. The cloth responds more dynamically in the simulation using ERE and IM. Both SI and BE introduce much more damping into the system, as shown in Figure 4.7. Table 4.3 displays the total simulation running time for each integrator with step size $h = 1/60$. ERE and SI are much cheaper than BE and IM, since no

additional Newton iterations are required.

In Figure 4.8 we changed half of the cloth to a slightly softer material $K_l = 20N/m$ and the other half to a stiffer material $K_l = 200N/m$. In this mixed material example, both SI and BE are seen to introduce non-uniform damping, relative to the stiffness, and fail to capture the state of the cloth even qualitatively, whereas ERE looks qualitatively similar to IM.

Character Cloth

In this example we simulate a cape (11×11) attached to a dancing character, with step size $h = 1/60$ (Figures 4.9 and 4.10). From these examples we conclude that ERE is an efficient explicit method for problems such as cloth simulation that does not introduce too much damping and responds more dynamically to user inputs and collision events.

Table 4.3: Running time of each method for simulating the cloth collision scene for 5 seconds.

Integrator	Total running time
ERE	8.4187
SI	6.8089
BE	18.9087
IM	29.8337

3D Volumetric Bar

Next we use a neo-Hookean material model to simulate elastic deformation of 3D volumetric bars. For the first volumetric simulation, we simulate a twisted soft elastic bar with Young's modulus values YM ranging from $10Pa$ to $10kPa$, see Figure 4.11.

In our second volumetric example we simulate elastic bars, with Young's modulus ranging from $10kPa$ to $50kPa$, dropping with one end fixed; see Figure 4.12. Similarly, we plot the running time of each method in Figure 4.12a, and the en-

ergy profile of the bar with Young's modulus $10kPa$ in Figure 4.12b. We observe that the running time for ERE increased as we stiffen the material, as predicted in Figure 4.1.

Both these examples clearly demonstrate that our previous conclusions extend to a volumetric simulation, namely, that the BE and SI methods lose too much energy while the fully implicit methods are too slow.

Stanford bunny

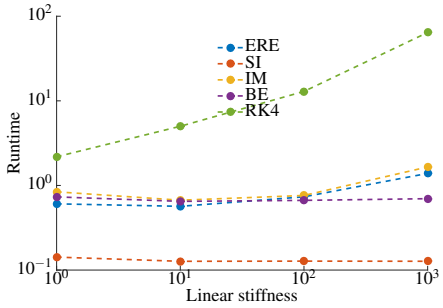
In our last volumetric simulation we simulate the Stanford bunny ($YM = 100kPa$, $h = 1/180$) bouncing due to gravity. Figure 4.13 shows the energy profile for each integrator in a similar fashion as before. Similar conclusions are drawn from this figure.

4.3 Conclusions

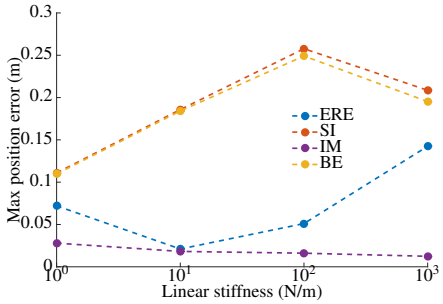
In this chapter we have analyzed and demonstrated a low cost algorithm for producing qualitatively correct simulations of elastodynamics motion involving soft materials. Our algorithm works well, using large time steps, even when the stiffness matrix is not positive definite everywhere, and it produces simulations that do not suffer heavy step size-related damping. Our system derives its relative efficacy from the fact that in typical soft body simulations in computer graphics the major cost per time step is in the assembly of the forces rather than in evaluating products with matrix exponentials. The need of solving nontrivial nonlinear systems of algebraic equations at each time step is avoided as well. The performance of our system has been demonstrated in Section 4.2; please see also our animation video clip.

Currently, the majority of researchers employ the semi-implicit (SI) method of [7] for similar simulation purposes. This method is attractive for several reasons: it is very simple, constraints are naturally incorporated at the end of each step, and stability is typically not an issue (though not always [3]). However, it is also well-known that a significant artificial damping is introduced by this method, thus making an artist using such a simulation tool having to deal with implementation-dependent artifacts and introducing distortion in the resulting animation. In simple

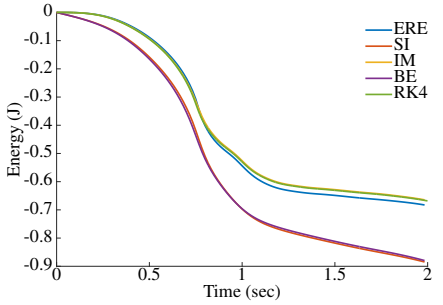
cases the obtained motion often looks realistic, but with a heavier damping than desired. In the experiments of Section 4.2, ERE shows a clear advantage over SI for cloth simulation: the step size dependent damping introduced by SI could significantly change the cloth response to external force, whereas ERE keeps the solution qualitatively similar to that of the physical model, even when using large steps in time. In addition, since ERE is explicit, popular constraints for cloth motion [13, 31] can be easily imposed.



(a) Running time of each integrator for the rope simulation with different stiffness.

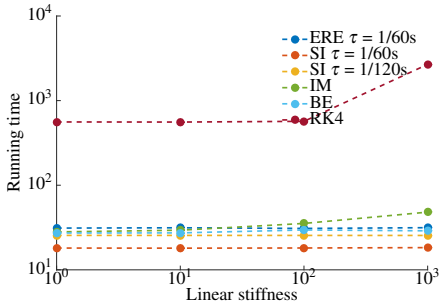


(b) ERE gives up some point-wise accuracy; however, it is close to that of IM and performs significantly better than BE and SI. The latter also introduce more artificial damping.

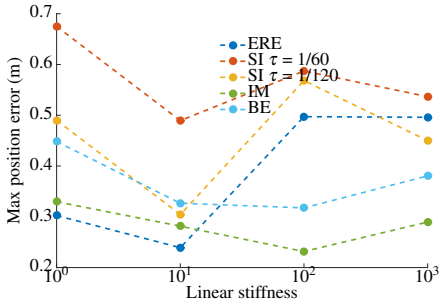


(c) Both SI and BE damp out the energy significantly, while the energy profile of ERE follows closely those of IM and the small-step RK4.

Figure 4.3: Comparison between ERE with SI, BE, IM, and RK4 over 2 seconds of the rope simulation. (a) running time, (b) position error, and (c) energy profile.



(a) Running time for simulating 5 seconds of cloth modeled by a 2D Mass-spring system.



(b) Max position errors for each integrator during the simulation.

Figure 4.4: These plots compare ERE to SI, BE, and IM over 5 seconds of cloth modeled by a mass-spring system. SI with $h = 1/120$ is added to demonstrate the h-dependent damping effect of this method.

(a) ERE $h = 1/60$ **(b)** IM $h = 1/60$ **(c)** RK4 $h = 1/1260$ **(d)** SI $h = 1/60$ **(e)** SI $h = 1/120$ **(f)** BE $h = 1/60$

Figure 4.5: Response of the cloth with ($K_l = 100N/m$) after the shaking event using different integrators.

(a) ERE **(b)** SI **(c)** IM **(d)** BE

Figure 4.6: Response of the cloth ($K_l = 100N/m$) after colliding with a sphere using different integrators.

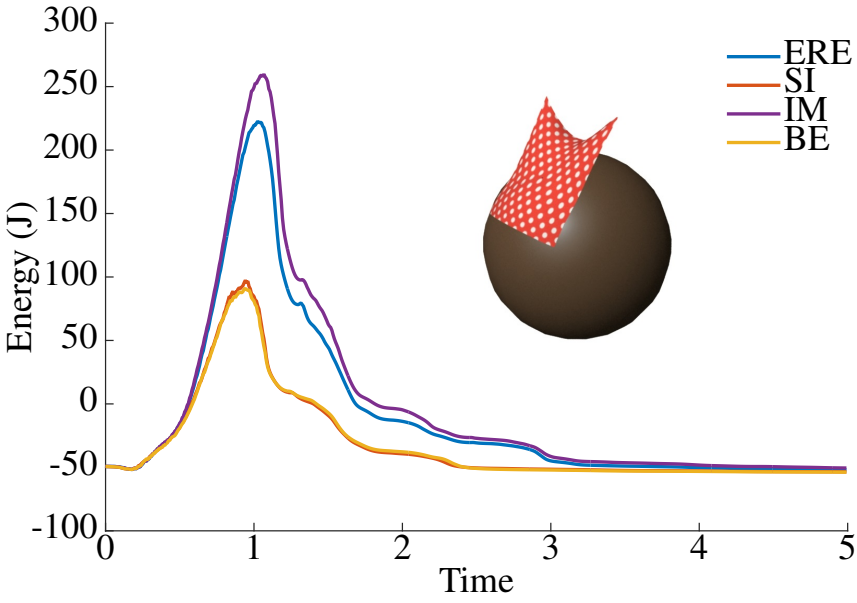
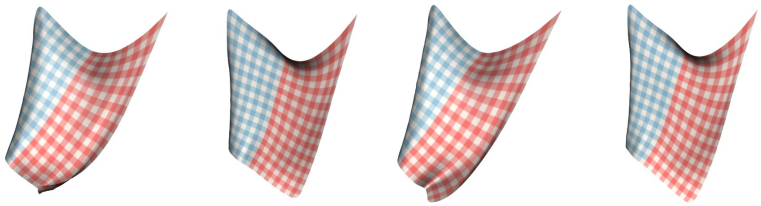


Figure 4.7: Energy profile of each method in the simulation with cloth collision.



(a) ERE **(b)** SI **(c)** IM **(d)** BE

Figure 4.8: Response of the cloth with mixed stiffness, $K_l = 20N/m$ (red) and $K_l = 200N/m$ (blue), after colliding with a sphere. The highly damping integrators miss the correct qualitative behavior.

(a) ERE **(b)** SI **(c)** IM **(d)** BE

Figure 4.9: Simulating the cape motion of a dancing character using different integrators ($K_l = 50N/m$).

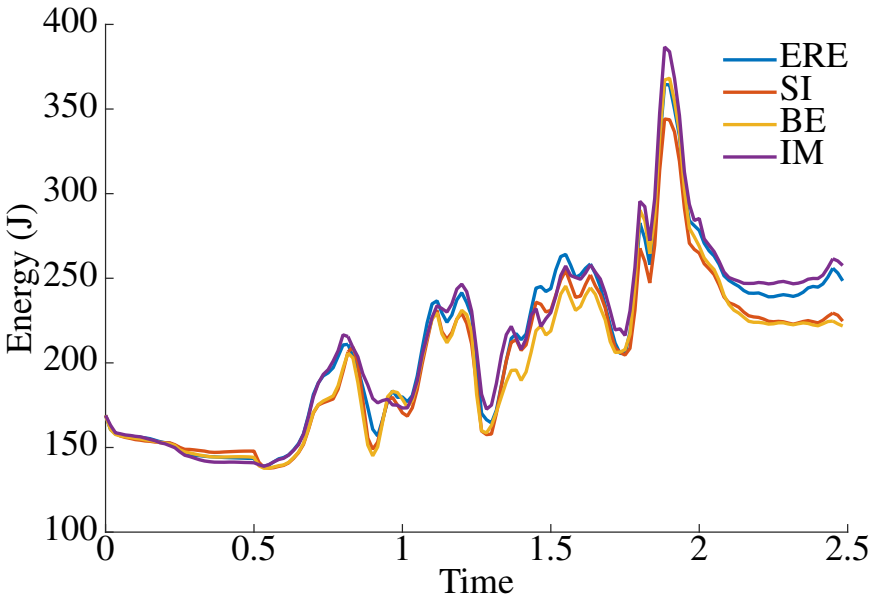
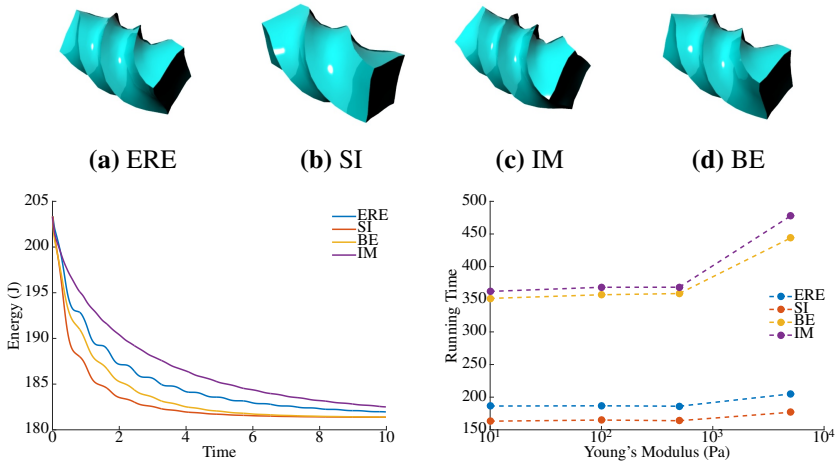


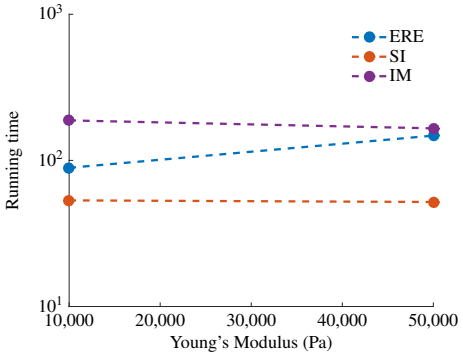
Figure 4.10: Energy profiles of the four integrators for the example of a cape attached to a dancer.



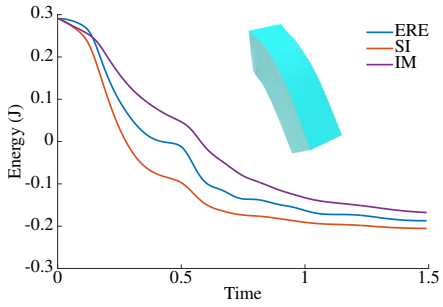
(a) ERE **(b)** SI **(c)** IM **(d)** BE

(e) Energy profile of different methods **(f)** Running time

Figure 4.11: These plots compare ERE to SI, BE, and IM in the twisted bar simulation.



(a) Running time of each method for simulating a elastic bars with Young's modulus from 10kPa to 50kPa.



(b) Energy profile of different methods for the elastic bar with Young's modulus $10kPa$.

Figure 4.12: These plots compare ERE to SI, BE, and IM in the simulation of a volumetric bar with one end fixed.

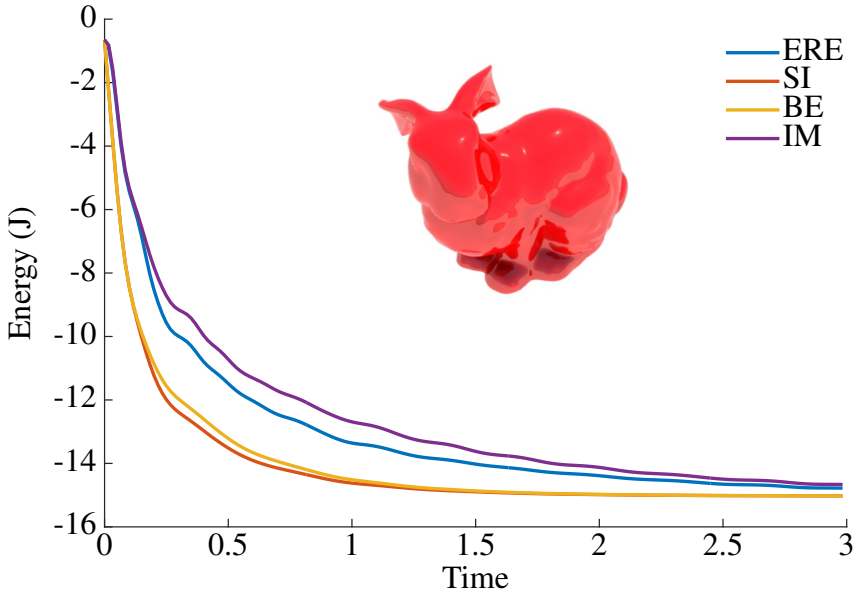


Figure 4.13: Energy profile of each method for the bunny with Young's modulus 100kPa.

Chapter 5

Additive method

In the previous chapter, we found that the ERE becomes costly as we increase the stiffness parameter and is not suitable for stiff elastodynamics. This is because the ODE systems that we solve here are so large that calculating the exponential matrix explicitly is out of the question, and hence, Krylov subspace methods are employed for approximately calculating its product with vectors. The latter method suffers from performance deterioration for very stiff problems. In this chapter we investigate ways to improve integrators for stiff elastodynamics by splitting the ODE system [3, 4, 33]. We'll return to ERE in the next chapter, where we use a splitting method to improve it. In particular, given an ODE system in the form

$$\dot{\mathbf{u}} = \mathbf{F}(\mathbf{u}) = \mathbf{G}(\mathbf{u}) + \mathbf{H}(\mathbf{u})$$

we can write

$$\mathbf{u}(t) = \mathbf{x}(t) + \mathbf{y}(t)$$

where

$$\dot{\mathbf{x}} = \mathbf{G}(\mathbf{x}+\mathbf{y}) \tag{5.1a}$$

$$\dot{\mathbf{y}} = \mathbf{H}(\mathbf{x}+\mathbf{y}) \tag{5.1b}$$

If Eq.(5.1a) is only moderately stiff while Eq.(5.1b) is potentially very stiff, the overall ODE Eq. (5) is still stiff. If we apply explicit forward Euler (FE) to Eq. (5),

we have to take unreasonably short steps to resolve the fast dynamics in Eq. (5.1b). On the other hand, applying implicit BE to Eq. (5) may be acceptable, but will introduce excessive damping to both Eqs. (5.1a) and (5.1b). In this case it makes sense to apply FE to Eq. (5.1a) and BE to Eq. (5.1b). This leads to an implicit-explicit (IMEX) method

$$\begin{aligned} \mathbf{x}_{n+1} &= \mathbf{x}_n + h\mathbf{G}(\mathbf{x}_n + \mathbf{y}_n) \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + h\mathbf{H}(\mathbf{x}_{n+1} + \mathbf{y}_{n+1}) \end{aligned}$$

or

$$\mathbf{u}_{n+1} = \mathbf{u}_n + h\mathbf{G}(\mathbf{u}_n) + h\mathbf{H}(\mathbf{u}_{n+1}).$$

Next, we can further replace the BE part by its semi-implicit version using the local linearization of $\mathbf{H}$

$$\mathbf{u}_{n+1} = \mathbf{u}_n + h\mathbf{G}(\mathbf{u}_n) + h\mathbf{H}(\mathbf{u}_n) + h\mathbf{J}_\mathbf{H}(\mathbf{u}_{n+1} - \mathbf{u}_n) \tag{5.2}$$

with

$$\mathbf{J}_\mathbf{H} = \frac{\partial}{\partial \mathbf{u}}\mathbf{H} \tag{5.3}$$

or

$$\begin{aligned} \mathbf{u}_{n+1} &= (\mathbf{I} - h\mathbf{J}_\mathbf{H})^{-1} (\mathbf{u}_n + h\mathbf{G}(\mathbf{u}_n) + h\mathbf{H}(\mathbf{u}_n) - h\mathbf{J}_\mathbf{H}\mathbf{u}_n) \\ &= (\mathbf{I} - h\mathbf{J}_\mathbf{H})^{-1} (\mathbf{u}_n + h\mathbf{F}(\mathbf{u}_n) - h\mathbf{J}_\mathbf{H}\mathbf{u}_n). \\ &= \mathbf{u}_n + (h\mathbf{I} - h\mathbf{J}_\mathbf{H})^{-1} \mathbf{F}(\mathbf{u}_n) \end{aligned} \tag{5.4}$$

Notice that Eq.(5.4) is a simple splitting integrator which does not require explicit knowledge about $\mathbf{x}$, $\mathbf{y}$, $\mathbf{G}$, and $\mathbf{H}$. The only additional evaluation required is $\mathbf{J}_\mathbf{H}$. This is important in some applications where the splitting in Jacobian $\mathbf{J} = \frac{\partial}{\partial \mathbf{u}}\mathbf{F}$ is easy to derive, but the splitting in $\mathbf{F}$ is not, as is the case in Section 5.1 below. If Eq.(5.1b) is the only stiff part, Eq.(5.4) should enable large time step without sacrificing stability. Next, to see this method in action, we apply and analyze Eq. (5.4) in mass-spring systems, where a natural splitting in the Jacobian exists.

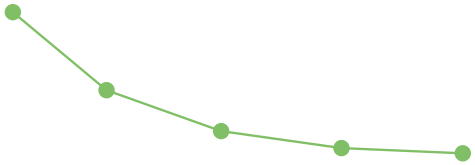


Figure 5.1: Simulation of a swinging rope using a mass-spring system.

5.1 Splitting for mass-spring system

In the following we introduce the variables for a single spring for simplicity, but all the vectors and matrices can be extended for a large spring network; see Figure 5.1. For a spring s, the position state

$$\mathbf{q} = (\mathbf{q}_0, \mathbf{q}_1)^{\mathrm{T}}$$

and velocity state

$$\mathbf{v} = \dot{\mathbf{q}} = (\dot{\mathbf{q}}_0, \dot{\mathbf{q}}_1)^{\mathrm{T}}$$

represent the total 12 degrees of freedom. Here $\mathbf{q}_0, \mathbf{q}_1$ are the end nodes of the springs. We define the strain

$$\begin{aligned} \varepsilon_s(\mathbf{q}) &= \|\mathbf{s}\| - l_0 \\ \mathbf{s} &= \mathbf{q}_0 - \mathbf{q}_1 \end{aligned}$$

where l_0 is the spring's rest length. In this spring case, the strain is a signed scalar measurement that shows how far away the spring is from its rest state. We further define the stress in the spring

$$\begin{aligned} \sigma(\mathbf{q}) &= k\varepsilon_s(\mathbf{q}) \\ &= k(\|\mathbf{q}_0 - \mathbf{q}_1\| - l_0) \\ &= k(\|\mathbf{s}\| - l_0) \end{aligned}$$

where k is the stiffness constant. Intuitively, stiffer springs generate larger

stress under the same strain, thus k is larger for stiffer springs. Notice that both strain and stress are scalar. To relate strain and stress to the force vector the spring exerts on its end nodes, we define the elastic energy of the spring

$$V = \frac{1}{2} k \varepsilon_s(\mathbf{q})^{\mathrm{T}} \varepsilon_s(\mathbf{q})$$

Spring network If we have a spring network with multiple springs, the total elastic energy can be written as

$$V_{tot} = \frac{1}{2} \boldsymbol{\varepsilon_s}(\mathbf{q})^{\mathrm{T}} \mathbf{D}_k \boldsymbol{\varepsilon_s}(\mathbf{q})$$

where $\boldsymbol{\varepsilon_s}(\mathbf{q})$ is a column vector containing the strain of each individual spring. $\mathbf{D}_k$ is a diagonal matrix with corresponding material constants along the diagonals.

Coming back to the single spring, the elastic force exerted on the nodes is written as

$$\begin{aligned} \mathbf{f}_{ela}(\mathbf{q}) &= -\frac{\partial V}{\partial \mathbf{q}}^{\mathrm{T}} \\ &= -\frac{\partial \varepsilon_s}{\partial \mathbf{q}}^{\mathrm{T}} \frac{\partial V}{\partial \varepsilon_s}^{\mathrm{T}} \\ &= -\mathbf{J}^{\mathrm{T}} \sigma(\mathbf{q}) \end{aligned} \tag{5.5}$$

with the 1×6 nonlinear Jacobian

$$\begin{aligned} \mathbf{J}(\mathbf{q}) &= \frac{\partial}{\partial \mathbf{q}} \varepsilon_s(\mathbf{q}) \\ &= \frac{1}{\|\mathbf{s}\|} \mathbf{s}^{\mathrm{T}} \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 \end{bmatrix} \end{aligned}$$

5.1.1 Equations of motion

Before we can apply Eq. (5.4), we also need to derive the stiffness matrix for the mass-spring system, which is the derivative of Eq. (5.5) with a negative sign:

$$\begin{aligned} \mathbf{K}(\mathbf{q}) &= -\frac{\partial}{\partial \mathbf{q}}\mathbf{f}_{ela}(\mathbf{q}) \\ &= \frac{\partial^2}{\partial \mathbf{q}^2}V(\mathbf{q}) \\ &= k\mathbf{J}^{\mathrm{T}}\mathbf{J} + k\frac{\partial \mathbf{J}^{\mathrm{T}}}{\partial \mathbf{q}}\varepsilon_s(\mathbf{q}) \end{aligned} \tag{5.6}$$

Material and geometric stiffness For a single spring the first term in (5.6) is

$$\mathbf{K}_s = k\mathbf{J}^{\mathrm{T}}\mathbf{J} = k\begin{bmatrix} \mathbf{s}\mathbf{s}^{\mathrm{T}} & -\mathbf{s}\mathbf{s}^{\mathrm{T}} \\ \mathbf{s}\mathbf{s}^{\mathrm{T}} & \mathbf{s}\mathbf{s}^{\mathrm{T}} \end{bmatrix} \tag{5.7}$$

called the material stiffness in [86], and the second term is

$$\tilde{\mathbf{K}} = k\frac{\partial \mathbf{J}^{\mathrm{T}}}{\partial \mathbf{q}}\varepsilon_s(\mathbf{q}) = k\frac{l_0 - \|\mathbf{s}\|}{\|\mathbf{s}\|}\begin{bmatrix} \mathbf{I}_3 - \mathbf{s}\mathbf{s}^{\mathrm{T}} & \mathbf{s}\mathbf{s}^{\mathrm{T}} - \mathbf{I}_3 \\ \mathbf{s}\mathbf{s}^{\mathrm{T}} - \mathbf{I}_3 & \mathbf{I}_3 - \mathbf{s}\mathbf{s}^{\mathrm{T}} \end{bmatrix}$$

called the geometric stiffness in [86]. Suppose the force at each spring is smooth throughout the simulation, and there is a maximum force f_{max}, then

$$\|k(\|\mathbf{s}\| - l_0)\| \leq f_{max}.$$

Now, in the case where the stiffness parameter is arbitrarily large, that is, $f_{max} \ll k$, we have material stiffness and geometric stiffness at two different scales, and we can apply Eq.(5.4). Since material stiffness can be very large, we use only the material stiffness matrix in the SI part, leading to a material stiffness semi-implicit (MSSI) method:

$$\mathbf{v}_{n+1} = \mathbf{v}_n + (\mathbf{I} - h^2\mathbf{M}^{-1}\mathbf{K}_s)^{-1}h\mathbf{M}^{-1}\mathbf{f}_{tot} \tag{5.8}$$

We have demonstrated how to use Eq. (5.4) in an elastodynamics system Eq. (1.2). In particular, we split the stiffness matrix for the mass-spring system Eq. (5.6), but not the force Eq. (5.5). Next, we look at another elastodynamics system, where the spatial discretization utilizes the FE discontinuous Galerkin method, and Eq. (5.4) can be applied as well.

5.2 Splitting for Discontinuous Galerkin method

Discontinuous Galerkin (DG) is a class of finite element methods that relax the strong constraints on using conforming elements, thus allowing elements to be piecewise continuous. More details on use of DG in computer graphics can be found in [41]. Different DG formulations define different numerical fluxes, which result in different gluing energy. For example, one of the most popular DG methods, DGBZ [5], defines the quadratic gluing energy

$$E_{BZ}(\mathbf{u}) = \eta_f \sum_i \|\mathbf{u}_{i,-} - \mathbf{u}_{i,+}\|^2, \tag{5.9}$$

where η_f is an empirical penalty constant, and the summation over i adds up all the energy at element interfaces. For our purpose, we skip the detailed explanation for discontinuous Galerkin. Using Eq. (5.9), we have the total energy

$$E_{DGBZ}(\mathbf{u}) = \boldsymbol{\sigma}(\mathbf{u}) : \boldsymbol{\varepsilon}(\mathbf{u}) + \eta_f \sum_i \|\mathbf{u}_{i,-} - \mathbf{u}_{i,+}\|^2 \tag{5.10}$$

The first term and second term in Eq. (5.10) correspond to the element and the interface energy, respectively[1]. In this formulation, we have a natural splitting for the force and the stiffness matrix

$$\begin{aligned} \mathbf{f}_{tot} = &\quad \mathbf{f}_{Element} + \quad \mathbf{f}_{Interface} \\ \mathbf{K}_{tot} = &\quad \mathbf{K}_{Element} + \quad \mathbf{K}_{Interface} \end{aligned}$$

We derive $\mathbf{f}_{Interface}$ and $\mathbf{K}_{Interface}$ for 2D linear triangle elements in Appendix . To model stiff material, we may have $\|\mathbf{K}_{Element}\| \gg \|\mathbf{K}_{Interface}\|$. This is an ideal

[1]Notice that in our notation for Eq. (5.10), the stress $\boldsymbol{\sigma}(\mathbf{u})$ and strain $\boldsymbol{\varepsilon}(\mathbf{u})$ tensors can both be nonlinear in $\mathbf{u}$.

case to apply Eq.(5.4), leading to a discontinuous Galerkin semi-implicit (DGSI) integrator[2]

$$\mathbf{v}_{n+1} = \quad \mathbf{v}_n + (\mathbf{I} - h^2\mathbf{M}^{-1}\mathbf{K}_{Element})^{-1} h\mathbf{M}^{-1}\mathbf{f}_{tot} \tag{5.11}$$

By now we have introduced two splittable elastodynamics systems where using Eq. (5.4) can be useful. In both systems, we integrate the stiff part with SI, and the non-stiff part with FE.

- In mass-spring systems, we integrate material stiffness with SI, and geometric stiffness with FE, as per Eq. (5.8).
- Under DG formulation, we integrate elastic forces with SI, and penalty forces with FE, Eq. (5.11).

The resulting integrators should have better dynamic behavior compared to applying full SI to the system. We will look at some comparative results in the next section.

5.3 Results

5.3.1 MSSI for a stiff pendulum

We simulate a stiff pendulum swinging under gravity with $m = 1kg$, $k = 1e4m/s^2$ using a step size $h = 0.1sec$, and plot the kinetic energy from MSSI and SI in Figure 5.2. Notice that by Eq. (5.8) MSSI and SI have the same cost, but at this step size SI significantly damps out the dynamic. The oscillatory energy plot of MSSI shows that it preserves the swing motion well.

5.3.2 DGSI for a 2D triangle mesh

We simulate a 2D triangle mesh bending and releasing with nonlinear neo-Hookean energy with $m = 1kg$, $YM = 1e5Pa$, $\eta_f = 1e2$ using a step size $h = 0.01sec$. We compare regular SI for CG simulation against DGSI Eq. (5.11) and plot the total

[2]Since $\mathbf{K}_{Element}$ is block-diagonal, if $\mathbf{M}$ is diagonally lumped Eq. (5.11) can be efficiently solved through parallelization.

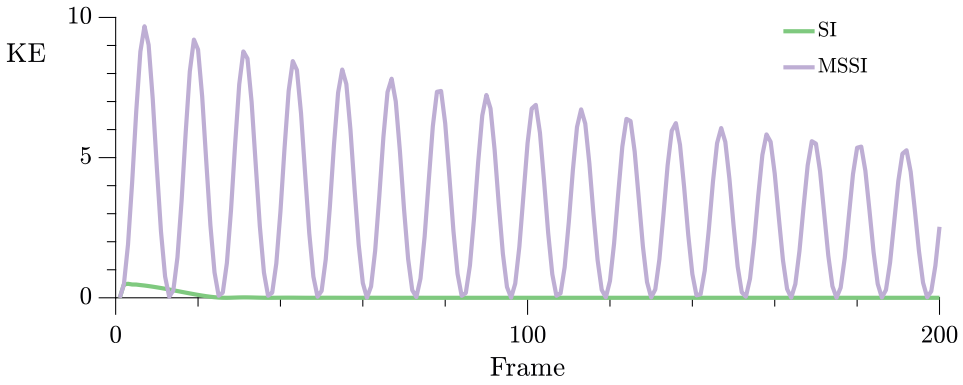


Figure 5.2: Kinetic energy for a pendulum simulated with MSSI and SI. The SI energy is strongly damped.

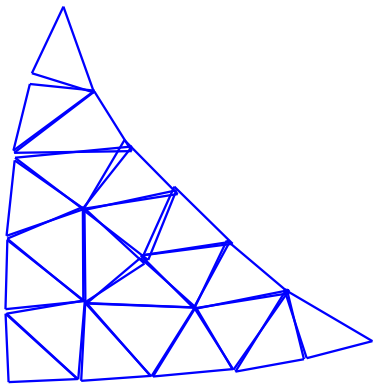


Figure 5.3: Simulation of a 2D triangle meshing using DG.

energy from frame 50 to frame 1000 in Figure 5.4. Again, we observe that DGSI has much better energy behavior since SI is not applied to the entire system.

5.4 Discussion

The integrators in this chapter, Eq. (5.8) and Eq. (5.11), were designed to allow large time step without introducing significant amount of uncontrolled damping. The energy plots from Section 5.3 demonstrate that using the additive methods Eq. (5.2) and Eq. (5.4) can efficiently achieve this goal, when there exists a splitting between the stiff and non-stiff part in the system. However, we also observe

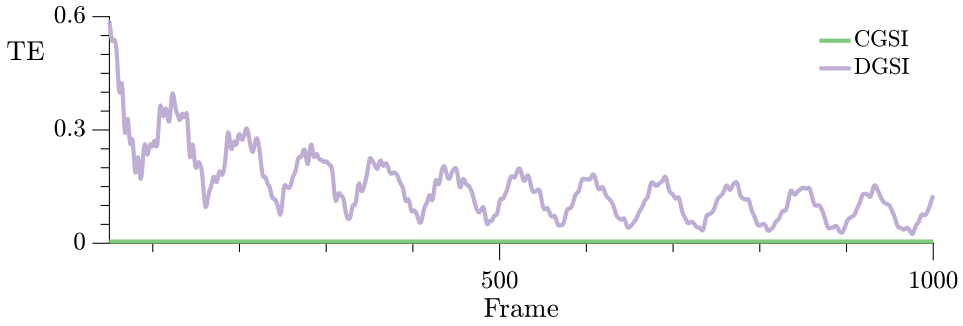


Figure 5.4: Total energy for a 2D triangle mesh simulated with SI using regular CG formulation (CGSI) and DGSI from Eq. (5.11) using DG formulation.

that, as the system size and stiffness were increased, both MSSI and DGSI became unstable. This is a limitation that arises from using a crude forward Euler integrator in Eq. (5.2) and Eq. (5.4). In addition, the splitting discussed in this chapter is achieved only in specific elastodynamics systems. For harder problems a transformation of the unknowns $\mathbf{u}$ may be required as well. In the next chapter, we overcome these limitations the limitations by using a subspace splitting similar to the one described in Chapter 3 with the exponential integrator from Chapter 4.

Chapter 6

SIERE: a hybrid semi-implicit exponential integrator for efficiently simulating stiff deformable objects

6.1 Introduction

Physics-based simulations of deformable objects are ubiquitous in computer graphics today. They arise in various applications including animation, robotics, control and fabrication. For almost two decades the semi-implicit backward-Euler (SI) method of Baraff and Witkin [7], has been widely employed [86]. This method allows for stable simulations even when large time steps are used for efficiency reasons, and it is very stable when incorporating contacts and collisions due to its heavy damping and error localization properties. Moreover, numerical damping is in agreement with the observation that our visual system does not detect high frequency vibrations, even when objects with large Young's modulus are simulated. However, recent years have also seen growing concerns that heavy numerical damping may be unsuitable for many purposes, because it can only be controlled via the time step size, so it does not distinguish phenomena related to material het-

erogeneity and more complex damping forces [19, 20]. Use of the SI method in applications such as control and fabrication has also been considered debatable [17]. A further concern is that SI has long been known to occasionally diverge wildly where the fully implicit backward Euler (BE) still yields acceptable results. This phenomenon, demonstrated in Sections 6.4.1 and 6.4.2, is due to divergence in low frequency modes.

Exponential time integration methods have been introduced relatively recently. These methods are attractive because, although non-conservative, they produce only little damping and approximate the entire modal spectrum acceptably well. These methods, like SI and unlike the full BE and various conservative methods, avoid the need of solving nonlinear algebraic equations at each time step [19, 62]. However, their performance cost becomes prohibitive when the simulated system of differential equations in time is stiff, which happens for large Young's modulus values or upon using a fine spatial discretization. See Figure 6.3 and Table 6.2.

In this chapter we combine the exponential Rosenbrock Euler (ERE) integrator described in [19] with the SI method in a manner that allows each to concentrate on what it is good at and improve upon the other method's deficiencies. The result is an integrator that outperforms both its predecessors, especially for stiff problems: it produces animations that are similar to the exponential integrator's at a computational cost that does not increase as the simulated object gets stiffer.

The key idea is to consider, at each time step, the appropriately transformed assembly of forces in the equations of motion as a weighted sum of high frequency modes that are dealt with well by the heavily damping SI method, and low-to-medium frequency modes (which are the ones that contribute to what our visual system senses) that are dealt with efficiently by ERE. No nonlinear algebraic equations arise, even in the presence of nonlinear elastic forces. To do this we employ at each time step a partial spectral decomposition which picks the lower, leading modes, and apply ERE in the corresponding subspace. The rest is handled (i.e., damped out) by SI. We call the resulting method *S*IERE. To re-cap, the advantages of the new method are:

- It produces lively animations of a similar quality to that of the exponential method ERE; the results are better than SI both in staying closer to the sim-

ulated energy manifold and in avoiding over-smoothing of artifacts such as material heterogeneity and secondary motion.

- It handles contacts and collisions robustly, inheriting this property from SI.
- The computational cost of SIERE, unlike that of ERE, remains a small multiple of SI cost as the stiffness increases.
- As the exponential part of the method is performed only in a small spectrally-decomposed subspace, the pain of evaluating the impact of a large exponential matrix is avoided altogether.
- Potential wild divergence of SI (which arises mostly when relatively soft material stiffens under large deformation) is avoided by SIERE.

We demonstrate the performance and utility of SIERE, and will make our code available with the published paper.

6.2 Context and related work

Eq. (1.2) must be discretized before simulation, but as stated above, the popular SI method and even fully implicit integrators such as BE and higher order backward differentiation formulae (BDF) introduce significant, step-size dependent, artificial damping (see also [3]). The potentially heavy damping of high frequency modes introduced by BDF has often been observed in practice.

The solution $q(t)$ of the scalar, constant coefficient ODE

$$\ddot{q} + \omega^2 q = 0$$

oscillates undamped with frequency ω. But applying BDF with step size h gives a numerical solution that more closely approximates the modified ODE

$$\ddot{q} + d^{\text{method}}\dot{q} + \omega^2 q = 0,$$

where the damping coefficient $d^{\text{method}} > 0$ depends on the step size h and on ω. Straightforward linear algebra gives a value such that d^{method}/ω depends only on the product $h\omega$ [19] .

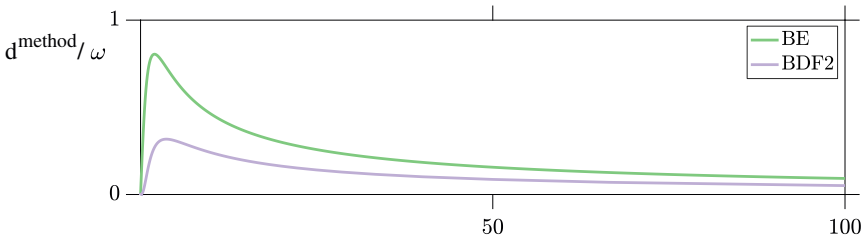


Figure 6.1: Using BDF methods such as BE and BDF2 introduces significant artificial damping that depends on the time step.

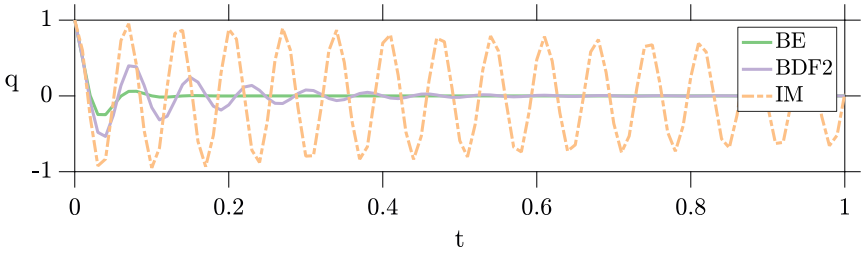


Figure 6.2: Solutions of the scalar test equation $\ddot{q} + d\dot{q} + \omega^2 q = 0$ for $\omega = 100, d = 1$, using BE, BDF2 and implicit midpoint (IM) with step size h=0.01.

Figure 6.1 shows the curves obtained for BE and BDF2, which are both popular methods for animating deformable objects. Figure 6.2 further depicts solution curves for particular, typical values of ω and h. As mentioned before, such artificial damping can lead to significant undesirable artifacts in the simulation when $h\omega$ is not small. On the other hand, the conservative implicit midpoint (IM) method introduces no artificial damping.

Note further that SI corresponds to applying one Newton iteration to solve the nonlinear algebraic equations arising from BE at each time step. Hence SI is the same as BE for linear problems. Further, we note in passing that an exponential method such as ERE reproduces the exact solution of this simple test problem. Although this ERE accuracy does not extend to general elastodynamics, the energy plots presented in the figures throughout Section 6.4 strongly indicate that the simple analysis in this appendix is indicative of the more general behaviour of BE and BDF2.

For linear problems, the methods SI and BE coincide. Further, a semi-implicit method such as SI cannot have an order of accuracy higher than 1. Replacing a nonlinear solver by such a linearization in BDF2 therefore reduces the resulting method's order of accuracy from 2 to 1.

As discussed in Chapter 4, several exponential integration schemes have been proposed in the computer graphics literature [19, 60–62].

However, as shown in Figure 6.3 and Table 6.2, these methods become expensive for stiff problems, thus limiting the utility of exponential integrators in the context of deformable object simulations [19, 62]. The root of this problem is in the fact that the required number of Krylov vectors needed in order to fully resolve the error for a stiff system as in Eq. (1.2) with a wide spectrum matrix and large $\mathbf{b}(\mathbf{u})$ can be very large. One might consider choosing a smaller time step to assist the matrix function evaluation for stiff objects; however, in physics-based simulations to stay competitive we expect to be able to keep employing large time steps independent of the material parameters. The method described in Section 6.3 alleviates this difficulty by applying the exponential integrator only in a suitable subspace, and the exponential matrix evaluation at each time step is simplified due to diagonalization.

A popular class of additive methods is the IMEX class, where an implicit discretization scheme is applied to Eq. (5.1b) and an explicit one to Eq. (5.1a) [4, 12, 29]. As described in Chapter 5, if the second term in Eq. (5) is stiff, while the first is not, it makes sense to apply one integration scheme to Eq. (5.1a) and another one to Eq. (5.1b), add them up and obtain a combined integration scheme for Eq. (1.2) [3, 57, 81]. Our additive method is similar in spirit to IMEX, although strictly speaking neither of its components is a fully implicit solver and both rely on solving a system of linear equations at each time step. The key is in determining a suitable splitting $\mathbf{G}$ and $\mathbf{H}$, described in the next section. Our spectral splitting allows us to achieve results in the graphics context that in general cannot be realized by the other additive methods cited above.

If a fully implicit method is employed, as is required for large-step conservative methods (see, e.g., [43]) and for higher order BDF methods such as BDF2, then a non-trivial nonlinear system of algebraic equations must be solved at each time step; see [27, 28, 83] and further references therein. Some of these efforts give rise

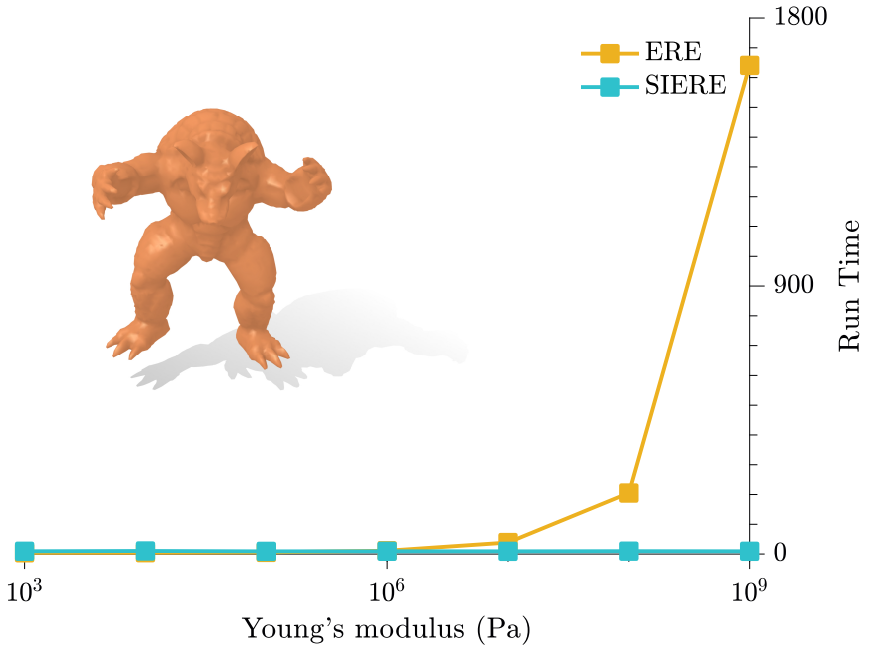


Figure 6.3: Exponential integrators (e.g., ERE) can better preserve the oscillations, but at a prohibitive cost as stiffness parameter and/or system size increases. By contrast, the cost of our method (SIERE) does not grow significantly with stiffness.

to rather elaborate schemes and packages. Note that in general energy projection methods cannot be symplectic or even reversible, and as such they also introduce some artificial dissipation. None of these methods are claimed to be faster or simpler than SI per step. SIERE elegantly avoids this significant and time consuming issue by using an inexpensive semi-implicit method throughout, resulting in a practical, fast and powerful method that is derived from solid, clear principles and can be easily incorporated to fit one's specific needs and code.

6.3 Method

Recall that we can write the equations of motion in the form Eq. (5), we consider the additive method

$$\mathbf{u}_{n+1} = \mathbf{u_n} + h\mathbf{H}(\mathbf{u}_{n+1}) + h\phi_1(h\mathbf{J_G})\mathbf{G}(\mathbf{u_n}),$$

where $\mathbf{J_G}$ is the Jacobian of the yet unspecified $\mathbf{G}(\mathbf{u_n})$. This combines the ERE method with the backward Euler (BE) method. One advantage of this is that calculating the ERE term first provides a "warm start" for the ensuing solution of the BE part.

Furthermore, if we want to avoid solving nonlinear equations then we can perform a single Newton iteration for $\mathbf{H}$ at each time step, replacing BE by SI. Thus, approximating $\mathbf{H}(\mathbf{u_{n+1}}) \approx \mathbf{H}(\mathbf{u_n}) + \mathbf{J_H}(\mathbf{u_{n+1}} - \mathbf{u_n})$, we obtain our proposed SIERE method as

$$\mathbf{u_{n+1}} = \mathbf{u_n} + (\mathbf{I} - \mathbf{h}\mathbf{J_H})^{-1}(\mathbf{h}\mathbf{H}(\mathbf{u_n}) + \mathbf{h}\phi_1(\mathbf{h}\mathbf{J_G})\mathbf{G}(\mathbf{u_n})). \tag{6.1}$$

Let us note in passing that, since this stable additive method consists of a combination of two first order methods, it converges like $O(h)$ as the step size h shrinks to 0. Our practical focus, however, remains on large time steps.

6.3.1 Model reduction and subspace splitting

Next, we define the splitting $\mathbf{G}$ and $\mathbf{H}$, crucial to the success of our method. The idea is to apply ERE in the subspace of the first s modes ($s \ll n$: typically, $5 \leq s \leq 20$) and project it back to the original full space. In the bridge example of Figure 3.1, $n \approx 100,000$, so this is a rather large reduction.

Then we use SI on the remaining unevaluated part, as per Eq. (6.1). We use mass-PCA, as introduced in Chapter 3, to find our reduced space. That is, considering at the beginning of each time step a solution mode of the form $\mathbf{q}(t) = \mathbf{w}\exp(\imath\sqrt{\lambda}t)$ for the ODE $\mathbf{M}\ddot{\mathbf{q}} + \mathbf{K}\mathbf{q} = 0$, we solve the generalized eigenvalue problem

$$\mathbf{K}\mathbf{w} = \lambda\mathbf{M}\mathbf{w} \tag{6.2}$$

for the s smallest eigenvalues λ and their corresponding eigenvectors w (dominant modes).

Next, we write Eq. (1.2) in the split form Eq. (5), with the splitting $\mathbf{H}$ and $\mathbf{G}$

defined based on the partial spectral decomposition. We define at each time step

$$\begin{aligned}
&\mathbf{G}(\mathbf{u_n}) = \begin{bmatrix} \mathbf{v_G} \\ \mathbf{M^{-1}f_G} \end{bmatrix}, \quad \mathbf{H}(\mathbf{u_n}) = \begin{bmatrix} \mathbf{v_H} \\ \mathbf{M^{-1}f_H} \end{bmatrix}, \\
&\mathbf{v_G} = \mathbf{U_s U_s^T M v}, \quad \mathbf{v_H} = \mathbf{v} - \mathbf{v_G}, \\
&\mathbf{f_G} = \mathbf{M U_s U_s^T f}, \quad \mathbf{f_H} = \mathbf{f} - \mathbf{f_G}.
\end{aligned} \tag{6.3}$$

We also need the Jacobian matrices

$$\mathbf{J_G} = \begin{bmatrix} 0 & \mathbf{U}_s\mathbf{U}_s^T\mathbf{M} \\ -\mathbf{U}_s\mathbf{U}_s^T\mathbf{K}\mathbf{U}_s\mathbf{U}_s^T M & 0 \end{bmatrix}, \quad \text{and} \tag{6.4a}$$

$$\mathbf{J_H} = \begin{bmatrix} 0 & \mathbf{I} \\ -\mathbf{M^{-1}K} & 0 \end{bmatrix} - \mathbf{J_G}. \tag{6.4b}$$

Notice that the ERE expression, $h\phi_1(h\mathbf{J_G})\mathbf{G}(\mathbf{u_n})$, can be evaluated in the subspace first, and then projected back to the original space.

The additive method defined by inserting Eq. (6.3) and (6.4) into Eq. (6.1) has three advantages:

1. At each time step, the majority of the update comes from ERE in the dominating modes. Thus it is less affected by artificial damping from SI.

2. The computation load of ERE is greatly reduced, because the stiff part is handled by SI (or BE for that matter). Furthermore, the evaluation of the exponential function in the subspace has only marginal cost since the crucial matrix involved has been diagonalized.

3. The "warm start" for SI makes its result closer to that of BE.

ERE update in the subspace: To evaluate the update in the subspace of dimension s we rewrite Eq. (6.1) as

$$\begin{aligned}
\mathbf{u}_{n+1} &= \mathbf{u_n} + (\mathbf{I} - \mathbf{h}\mathbf{J_H})^{-1}(\mathbf{h}\mathbf{H}(\mathbf{u_n}) \\
&+ h \begin{bmatrix} \mathbf{U}_s & 0 \\ 0 & \mathbf{U}_s \end{bmatrix} \phi_1(h\mathbf{J}_\mathbf{G}^r)\mathbf{G}^r(\mathbf{u_n})),
\end{aligned} \tag{6.5}$$

where

$$\mathbf{J}_{\mathbf{G}}^{r} = \begin{bmatrix} 0 & \mathbf{I} \\ -\mathbf{U}_s^T \mathbf{K} \mathbf{U}_s & 0 \end{bmatrix} = \begin{bmatrix} 0 & \mathbf{I} \\ -\mathbf{\Lambda}_s & 0 \end{bmatrix},$$
$$\mathbf{G}^{r}(\mathbf{u_n}) = \begin{bmatrix} \mathbf{U}_s^T \mathbf{M} \mathbf{v} \\ \mathbf{U}_s^T \mathbf{f} \end{bmatrix}. \tag{6.6}$$

The evaluation of the action of the matrix function ϕ_1 involves only matrices of size $2s \times 2s$. Furthermore, the matrix function ϕ_1 can be evaluated directly through the eigenpairs of $\mathbf{J}_{\mathbf{G}}^{r}$

$$\left\{ i\sqrt{\lambda}_l, \quad \begin{bmatrix} \mathbf{e}_l \\ \imath\sqrt{\lambda}_l \mathbf{e}_l \end{bmatrix} \right\}, \quad \left\{ -\imath\sqrt{\lambda}_l, \quad \begin{bmatrix} -\mathbf{e}_l \\ \imath\sqrt{\lambda}_l \mathbf{e}_l \end{bmatrix} \right\}, \quad l = 1, ..., s, \tag{6.7}$$

with $\mathbf{e}_l$ the l^{th} column of the identity matrix.

The large $n \times n$ linear system solved in Eq. (6.5) is not sparse due to the fill-in introduced by the small subspace projection. Specifically, the off-diagonal blocks of the Jacobian matrix $\mathbf{J_G}$ defined in Eq. (6.4a) are not sparse. If not treated carefully, solving the linear system in Eq. (6.1) and Eq. (6.5) can be prohibitively costly. Fortunately, this modification matrix has the low rank s. We can write

$$\mathbf{J_G} = \mathbf{Y_1 Z_1^T} + \mathbf{Y_2 Z_2^T},$$

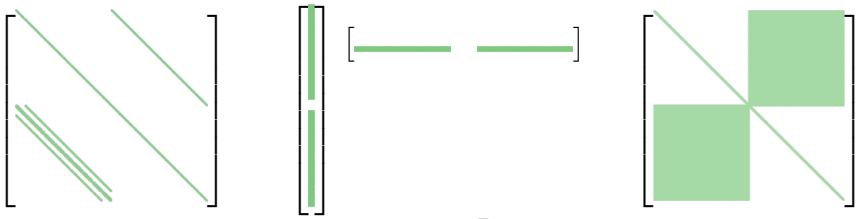
where

$$\mathbf{Y}_1 = \begin{bmatrix} \mathbf{U}_s \\ 0 \end{bmatrix}, \quad \mathbf{Z}_1 = \begin{bmatrix} 0 \\ \mathbf{M}\mathbf{U}_s \end{bmatrix},$$
$$\mathbf{Y}_2 = \begin{bmatrix} 0 \\ -\mathbf{U}_s \mathbf{U}_s^T \mathbf{K} \mathbf{U}_s \end{bmatrix}, \quad \mathbf{Z}_2 = \begin{bmatrix} \mathbf{M}\mathbf{U}_s \\ 0 \end{bmatrix}.$$

The linear system in Eq. (6.5) becomes

$$\mathbf{I} - \mathbf{h}\mathbf{J_H} = (\mathbf{I} - \mathbf{h}\mathbf{J}) + \mathbf{h Y_1 Z1^T} + \mathbf{h Y_2 Z_2^T}, \tag{6.8}$$

where the four matrices $\mathbf{Y}_i$ and $\mathbf{Z}_i$ are all "long and skinny" like $\mathbf{U}_s$, while the ma-



(a) (a)$I - hJ$ **(b)** (b)Y_i and Z_i^T **(c)** (c)$I - hJ_H$

Figure 6.4: The matrix $I - hJ_H$ in the linear system Eq. (6.5) is not sparse (c). Fortunately, by Eq. (6.8) the fill-in to the original sparse matrix $I - hJ$ (a) has low rank (b) allowing us to use the SMW formula Eq. (3.11).

trix $\mathbf{J}$ is square and large, but very sparse. Figure 6.4 illustrates this situation. For the linear system to be solved in Eq. (6.5) we may employ an iterative method such as conjugate gradient, whereby the matrix-vector products involving $\mathbf{J}$ or $\mathbf{Y}_i\mathbf{Z}_i^T$ are all straightforward to carry out efficiently. However, we have often found out that a direct solution method is more appropriate for these linear equations in our context. In our implementation we use `pardiso` [24, 47, 88]. For this we can employ the SMW formula, defined in Eq. 3.11, to solve the linear system in Eq. (6.5). In our specific notation we set at each time step $\mathbf{A} = \mathbf{I} - h\mathbf{J_H}$ in Eq. (3.11), and apply the formula twice: once for $\mathbf{Y} = \mathbf{Y}_1,\ \mathbf{Z} = \mathbf{Z}_1$, and once for $\mathbf{Y} = \mathbf{Y}_2,\ \mathbf{Z} = \mathbf{Z}_2$. Note that the matrices $\mathbf{I} + \mathbf{Z}^T\mathbf{A}^{-1}\mathbf{Y}$ in Eq. (3.11) are only $2s \times 2s$, and this results in an efficient implementation.

In Algorithm 1 we collect the pieces just derived.

6.4 Results

Here we compare the performance of the new method SIERE against SI and ERE on several examples. Our purpose is to show that SIERE produces simulation results that are similar to those of ERE (and thus do not suffer from the excessive damping of SI), with an implementation cost that does not increase when the stiffness increases (unlike that of ERE). Furthermore, we show examples where SIERE overcomes occasional divergence and instability phenomena arising in SI and ERE, as well as the fully implicit BE. Below in Sections 6.4.1 and 6.4.2 we begin with

ALGORITHM 1: SIERE step from t to $t+h$

Input: $\mathbf{u_n} = \begin{bmatrix}\mathbf{q_n}\\ \mathbf{v_n}\end{bmatrix}$, $\mathbf{f}$, $\mathbf{K}$, $\mathbf{M}$, h, s

; ▷ $\mathbf{K},\mathbf{M}$ `symmetric,` $\mathbf{M}$ `positive definite`

begin

- Solve (6.2) for $\mathbf{U}_s, \mathbf{\Lambda}_s$
- Construct $\mathbf{H}$; ▷ (6.3)
- Construct $\mathbf{J_G}^r, \mathbf{G}^r$; ▷ (6.6)
- Evaluate matrix function $\phi_1(h\mathbf{J_G}^r)$; ▷ `use` (6.7)
- $\delta \leftarrow h\mathbf{H} + h\begin{bmatrix}\mathbf{U}_s & 0\\ 0 & \mathbf{U}_s\end{bmatrix}\phi_1(h\mathbf{J_G}^r)\mathbf{G}^r$
- Construct $\mathbf{Y}_1, \mathbf{Z}_1, \mathbf{Y}_2, \mathbf{Z}_2$; ▷ (6.8)
- $\mathbf{x}_0 \leftarrow (\mathbf{I} - h\mathbf{J})^{-1}\delta$
- $\mathbf{x}_1 \leftarrow \mathbf{x}_0$ updated with $\mathbf{Y}_1, \mathbf{Z}_1$; ▷ `first` (3.11)
- $\mathbf{x}_2 \leftarrow \mathbf{x}_1$ updated with $\mathbf{Y}_2, \mathbf{Z}_2$; ▷ `second` (3.11)
- $\mathbf{u}_{n+1} \leftarrow \mathbf{u_n} + \mathbf{x}_2$; ▷ (6.5)

end

Output: $\begin{bmatrix}\mathbf{q}_{n+1}\\ \mathbf{v}_{n+1}\end{bmatrix} = \mathbf{u}_{n+1}$

the latter, as the examples used also allow us to investigate the effect of selecting the dimension s of the ERE update subspace. Later on we fix s and concentrate on the other aspects, comparing SIERE and SI simulation quality on larger and more geometrically complex objects where ERE performance is inadequate.

A list of meshes used and their sizes are summarized in Table 6.1. In Table 6.2 we list the CPU time for using the integrator for 100 frames. In our simulations we used a wide range of Young's moduli and a uniform Poisson's ratio $\nu = 0.45$. We also profiled the cost of the linear algebra calculations for SIERE, and found that under 20% of the runtime was due to partial eigendecomposition in Eq. (6.2), while the rest came from solving the linear system with low-rank update in Eq. (6.5). Notice that after the eigendecomposition, the exponential matrix evaluation is automatically completed from Eq. (6.7) at no extra cost.

For the backward Euler (BE) nonlinear solver we simply employed Newton's method at each time step, declaring convergence when the residual's norm was below 10^{-6}. Failure was declared if there was no convergence after 20 iterations. In our code we also have the option of cutting the step size by 2 and repeating, which is a form of numerical continuation [25], but we opted not to use this or any

Table 6.1: List of meshes used in SIERE representative experiments

Mesh ID	#Vertices)	#Tetrahedrons
Ball	1,018	4,637
Cuboid	1,245	4,624
Honeycomb	3,642	9,850
Moebius ball	5,829	16,857
Rope	12,791	47,279
Eiffel Tower	16,027	69,271
Tree	24,533	79,217
Bridge	30,133	99,455

Table 6.2: CPU Times (in seconds) for different methods, run on a core i7 machine with our C++ implementation. The reported stiffness is the peak stiffness of the object in Pa. The symbol $\otimes$ indicates unstable simulation, the relevant method's timing becoming irrelevant. The symbol $\times$ indicates that the simulation took far more than 20 times that of SIERE and thus we stopped the program before the simulation ended. All the SIERE results are for $s = 5$, except the rope example where $s = 1$.

Mesh	Stiffness	SI	SIERE	ERE
Ball	$1e8$	44	97	36
	$1e10$	42	96	112
Cuboid	$1e5$	$\otimes$	110	$\otimes$
Honeycomb	$1e8$	85	154	537
	$1e10$	84	156	5,840
Moebius ball	$1e8$	162	223	5,780
	$1e10$	154	260	$\times$
Rope	$1e6$	$\otimes$	651	$\times$
Eiffel Tower	$3e7$	631	2,315	$\times$
Tree	5e7	$\otimes$	2,690	$\times$
Bridge	$1e10$	1,375	4,585	$\times$

other more sophisticated techniques, as such is not our focus here.

6.4.1 Uncoiling rope

We start with an example to demonstrate the superior stability of SIERE arising from the "warm start" provided by ERE. In this example we simulate the process of uncoiling a soft homogeneous neo-Hookean rope (12,791 vertices and 47,279 tetrahedrons) with Young's modulus $YM = 1e6Pa$ and using $h = 0.01s$. The magnitude of deformation increases as the rope uncoils, and the stiffness increases.

When the rope is fully uncoiled, the deformation is localized at the very tip of the spring, leading to a challenging numerical system. SI quickly became unstable since one Newton iteration for backward Euler could not adequately reduce the residual for the system. On the other hand, the semi-implicit system arising from SIERE is stable. To emphasize the point, we chose the smallest subspace dimension $s = 1$ for this example: the exponential integrator still provides enough input to stabilize the simulation; see Figure 6.5.

In addition, we also observed that SIERE is able to uncoil each loop independently. This means that SIERE can simulate local deformation very well, unlike traditional linear subspace methods where global deformation artifacts appears for small subspace dimension [39]. SIERE does not have this issue because higher energy modes are not discarded.

Our BE implementation required 1257s for 400 frames, roughly double SIERE's runtime. The resulting energy curves are plotted in Figure 6.6. Notice that SIERE is not only faster, but also more dynamic. It can regain the elastic potential energy after each oscillation.

6.4.2 Untwisting cuboid

In this example we simulate the untwisting of a relatively soft elastic cuboid for a homogeneous neo-Hookean material with $YM = 1e5Pa$. See Figure 6.7. We have used the time step size $h = 0.01s$.

Note that the starting configuration here is challenging, since we begin the simulation where the stiffness matrix K not positive definite and with the cuboid under large deformation.

Both SI (see Figure 6.7(a) and ERE became unstable after a few steps, if for different reasons, while SIERE stayed stable.

Importantly, this example further demonstrates that it is unnecessary to employ a large subspace dimension s for SIERE to capture a complex deformation. We changed the subspace dimension s for SIERE and plotted the elastic potential energy for $s = 5, 6$ and 10; see Figures 6.7(b-d). Figure 6.7(e) shows that choosing a larger dimension s can improve the energy behavior. However, a small $s = 6$ suffices to capture this rather complex twisting simulation, and the difference in potential energy between $s = 6$ and $s = 20$ is small. Notice also that $s = 5$ already makes for a lively simulation, although $s = 6$ visibly improves upon it. Please watch our supplementary video for a better grasp of this demonstration as well as the following one.

ARAP energy

To demonstrate the universality of our method, we applied SIERE to the same cuboid example with ARAP energy [15, 80] instead of neo-Hookean. Using the same setup and simulating for 150 frames, SIERE required 87 CPU seconds and BE cost 129 seconds. SI failed here. Figure 6.8 shows the corresponding energy curves comparing SIERE and BE. Again, we see that SIERE is superior to BE with lower computational cost and better energy behaviour.

6.4.3 Ball movement

For the present example, as well as all the following ones in Sections 6.4.3–6.4.6, we have fixed the subspace dimension at $s = 5$, having verified for each example that the results using $s = 10$ and $s = 20$ are not significantly different. This allows us to concentrate on other issues and show cost comparisons in Table 6.2 as the problem size gets larger.

Hanging ball

In the present simulation ERE still performs well, simulating a stiff elastic ball under a uniform force field, with a fixed top. We use the neo-Hookean material and time step size $h = 0.005s$. The ball has radius 10cm with a hard shell of

thickness 2cm with peak Young's modulus $YM = 1e8Pa$ and inner core with $YM = 1e6Pa$. At such high stiffness, the ball would not deform under gravity, so we add a uniform downward pulling force ($20 \times g$) to simulate greater deformation. See Figure 6.9 for energy plots of simulations with the three methods. We also experimented with the damping behavior of SI, SIERE, and ERE. The runtime are recorded in Table 6.2, while the mesh size is in Table 6.1. Notice that SIERE has as good an energy behavior as ERE, whereas the uncontrolled damping of SI makes its resulting simulation look dull. In particular, the total energy plot in Figure 6.9 indicates that SI damps out gravitation potential very fast. For a small system like this one, ERE is fast as expected. We then increased the stiffness of the system by a factor of $1e2$ and re-simulated the scene, recording both timing data in Table 6.2. Notice that the cost for both SI and SIERE remained constant, but that for ERE increased by factor of 2. We also tried to reduce the damping error from SI by reducing the step size by factor 5; however, at such stiffness level, the uncontrolled damping is still large and no clear damping reduction was observed upon significantly reducing the step size.

Ball bouncing

In this example we drop the same ball as described above and simulate collision with the floor using simple penalty methods. The results are described visually in Figure 6.10 using both screenshots and energy plots.

We make two essential observations, which again are easier to appreciate by watching the supplementary video. The first is that, comparing to the SIERE animation (blue), in the SI animation (purple) the ball loses height faster than it should and is less dynamic. The second important observation, is that SIERE captures secondary motion that SI does not. This can also be seen in the potential energy plots in Figure 6.10(c).

6.4.4 Honeycomb structure

In this example we simulate a honeycomb sheet, a popular engineering structure that is strong and stiff in one direction but soft in the other two. We hang up the sheet and observe the ensuing oscillation. Again, we use neo-Hookean material.

We experiment with soft core at $YM = 1e6Pa$ and stiff edge $YM = 1e8Pa$, and integrate at $h = 0.01s$.

Plots of energy behavior are recorded in Figure 6.11 and runtime results are in Table 6.2. Notice that, even though the stiffness setting is similar to that in Section 6.4.3, ERE is already much more expensive than both SI and SIERE at this system size. When we further increase the stiffness by factor $1e2$ a similar trend can be observed more emphatically: ERE becomes prohibitively expensive, whereas SI and SIERE have a nearly constant cost at different stiffness levels. Finally, observe that SIERE has very good energy behavior for the dominating motions: damping out only the high frequency regime it achieves excellent stability similar to SI.

6.4.5 Moebius ball

Next we apply SIERE to simulate an exotic object, a Moebius ball, to demonstrate that wild shapes of the deforming object won't be the limiting factor of SIERE; see Figure 6.12. In this example we used nonlinear StVK material. We set a stiff $YM = 1e10Pa$ material for the core and $YM = 1e8Pa$ for the thin sheet spiral, and stepped forward with $h = 0.02s$. ERE is prohibitively expensive to use in this situation and we terminated the program before its simulation ended.

6.4.6 Large scale structures

To further demonstrate the capacity of SIERE, we applied it to some large scale examples. In the first of these, we simulated a toy Eiffel tower mesh swinging under strong wind; see Figure 6.13. We used homogeneous StVK material with $YM = 3e7Pa$. In the second larger scale example we simulated a multiscale system of a bridge using ARAP energy [15, 80]; see Figure 6.14. We used stiffness parameter $YM = 1e10Pa$ at the stiffest part, and 1e6 at the softest part. In both cases we simulated using a large step size $h = 0.1s$.

In both meshes, we observe similar features of SIERE and reach the same conclusion: SIERE can bypass the efficiency barrier of exponential integrators and remain cheap regardless of system size and material stiffness. Furthermore, SIERE achieves excellent damping behavior and stability by selectively damping out the high frequency oscillations.

Tree under strong wind

In our last example, we simulated a tree swaying in strong and varying wind using homogeneous neo-Hookean material with $YM = 5e7Pa$. The time step size was $h = 0.01s$. The tree branches have a complex structure with nearly 25,000 vertices; see Figure 6.15. SIERE can simulate a detailed motion for the branches and create a lively physical simulation with a small subspace dimension ($s = 5$). On the other hand, SI diverged for this simulation due to the large external force.

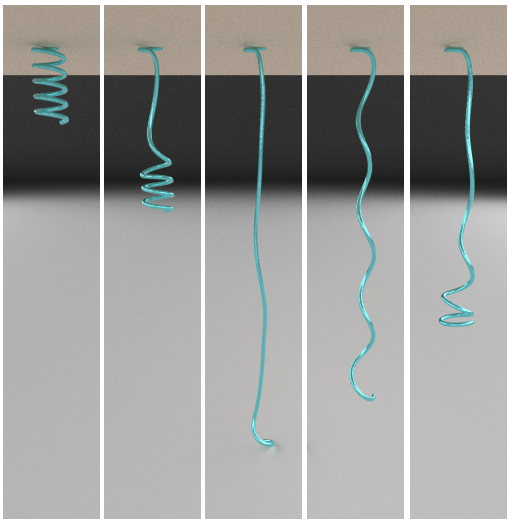
6.5 Conclusion

The quest of simulating, at a fair price, deformable structures that involve a high degree of stiffness in a manner that does not introduce over-damping is ubiquitous in computer graphics applications. Here we achieve this using an astonishingly simple approach. The key idea in this paper is to separate at each time step high frequency modes from the others, then employ for each regime a suitable method, and finally combine the results in an additive method that involves a low rank correction. Along the way we obtain a method that requires no solution of nonlinear algebraic equations and that is less prone to divergence and instabilities than both its components. We have demonstrated the efficacy of our method, and we urge the reader to watch our supplementary video since animations tell the story far better than stills.

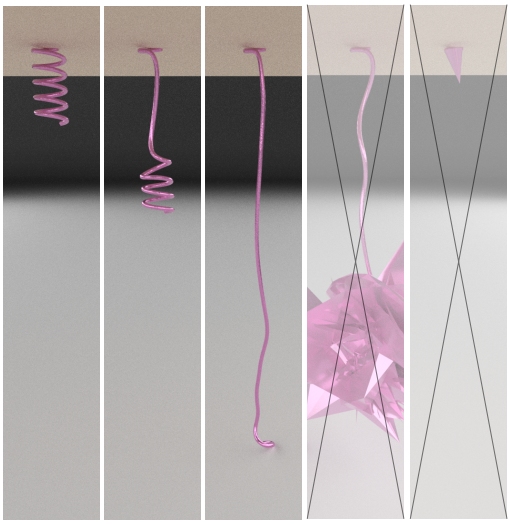
Limitations Our paper does not render the SI method obsolete: there are many situations where SI performs adequately, and in those cases it is the cheapest alternative.

At the other end, we note that our method is not structure-preserving as are suitable methods that allow no artificial damping at all [17]. We do claim that there is a significant range of applications in between these two extremes, however, where SIERE outperforms both.

For SIERE to succeed, we must assume that a moderate value of the subspace dimension s suffices. While this is the case with many visual simulations as we have shown, there are applications (such as sound) where other methods are more suitable.



(a) SIERE, $s = 1$



(b) SI

Figure 6.5: An uncoiling rope modeled with soft neo-Hookean material $YM = 1e6Pa$. (a) SIERE ($s = 1$) stays stable throughout the simulation, whereas (b) SI becomes unstable (diverges) when the rope stiffens at the end of the uncoiling process. This demonstrates the superior stability of SIERE.

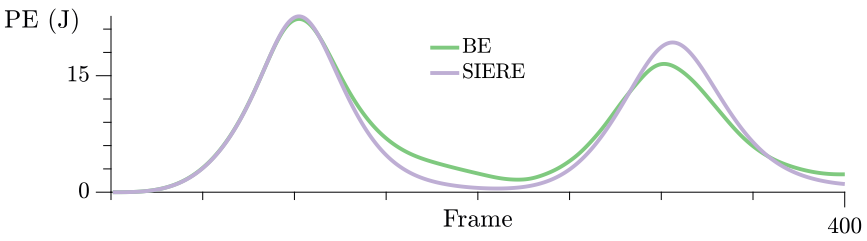
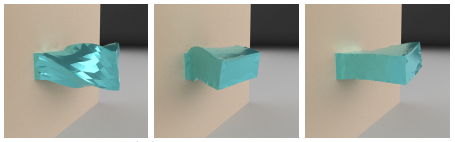
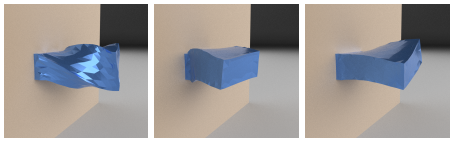


Figure 6.6: Energy curves comparing SIERE and BE over 400 frames for the uncoiling rope.

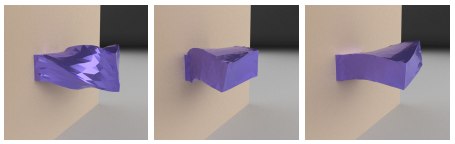
(a)SI



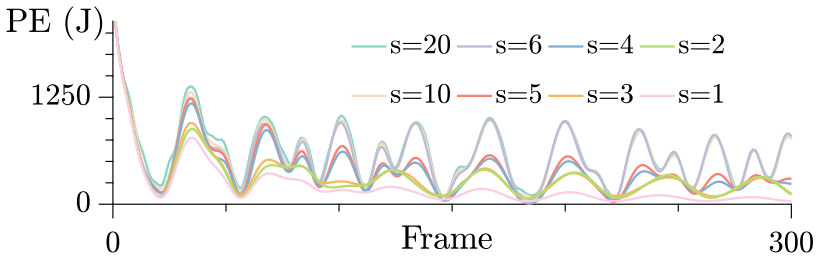
(a) (b) SIERE $s = 5$



(c) SIERE $s = 6$



(d) SIERE $s = 10$



(e) Potential Energy

Figure 6.7: Simulation of a cuboid untwisting with SI and different subspace dimensions s for SIERE: (a) SI failed in this challenging simulation; (b-d) SIERE results with different s values; (e) elastic potential Energy for these SIERE simulations and more. In this example, $s = 6$ can already capture the complex twisting deformation. While smaller s could not capture the energy trajectory fully, at $s = 5$ the simulation already looks energetic and lively.

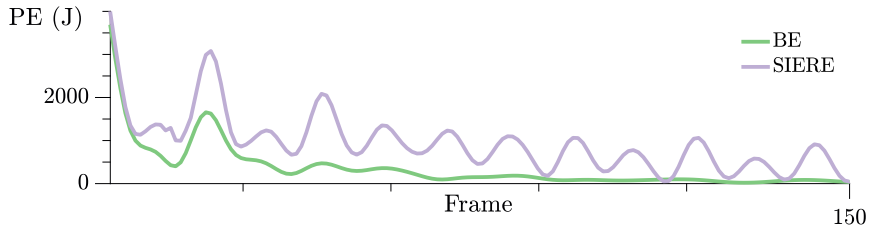


Figure 6.8: Energy curves comparing SIERE and BE over 150 frames for the cuboid simulation.

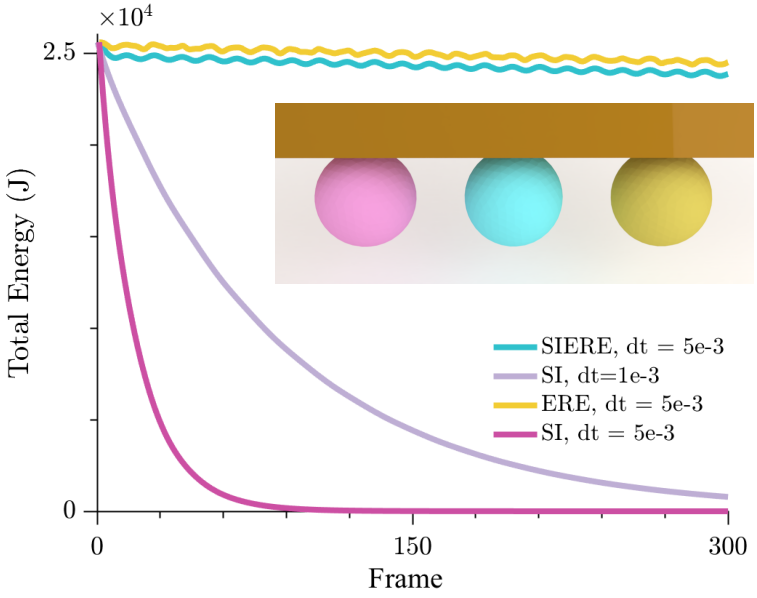


Figure 6.9: Energy plot from the simulation of a stiff elastic ball. While SI displays a typical decline in energy, both ERE and SIERE are close to conserving it.

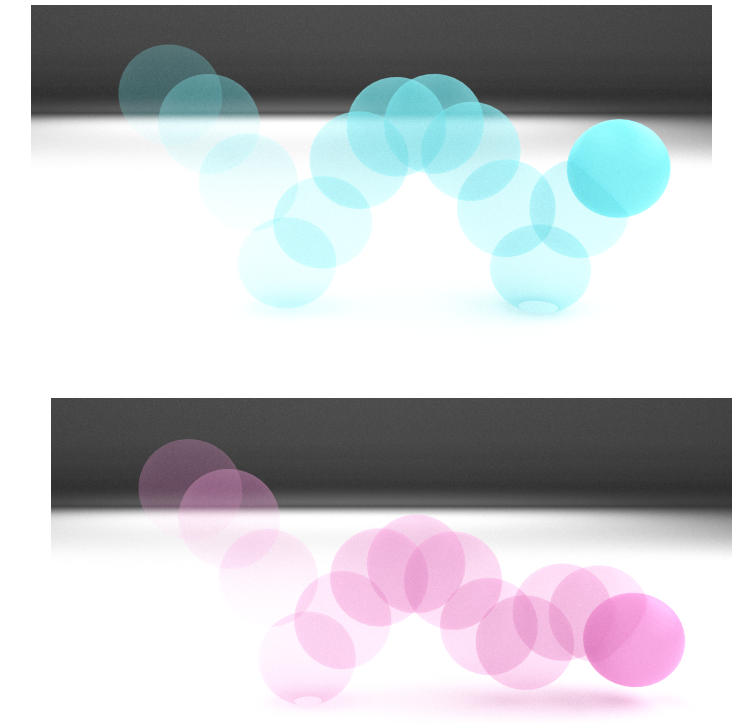


Figure 6.10: A soft ball bounces after colliding with the floor: (a) SIERE ball movement with ($s = 5$) can simulate the secondary motion with more dynamics; (b) SI is more lethargic, as it does not recover the ball height after collision well and it loses the secondary motion; (c) corresponding energy plots reflect both observations. Notice that SIERE preserves the energy level after two impacts.

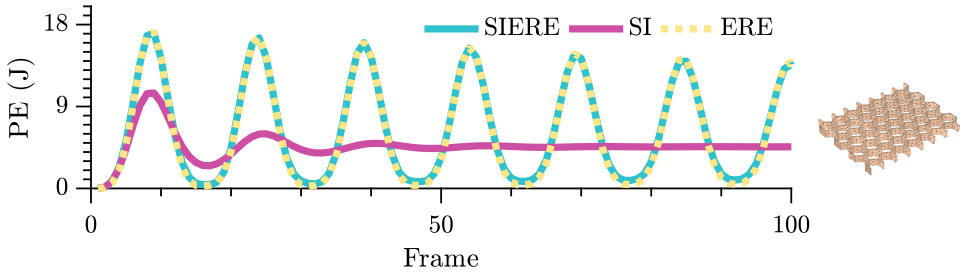


Figure 6.11: Simulation of a honeycomb sheet: energy plot.

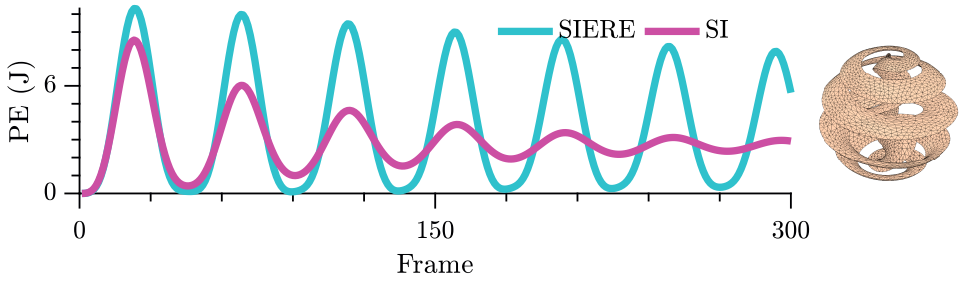


Figure 6.12: Simulation of a Moebius ball: energy plot.

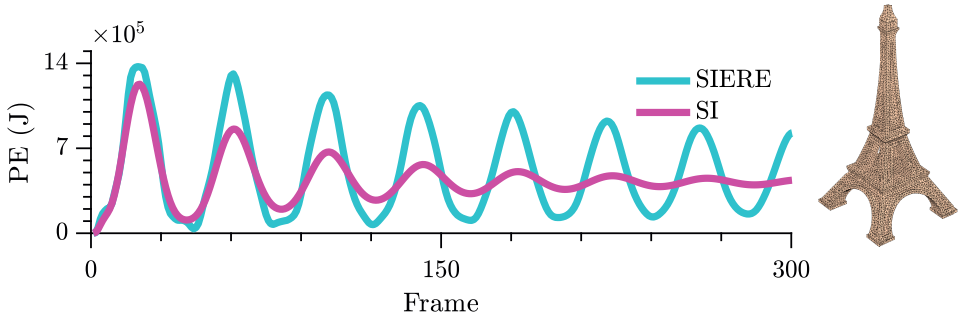
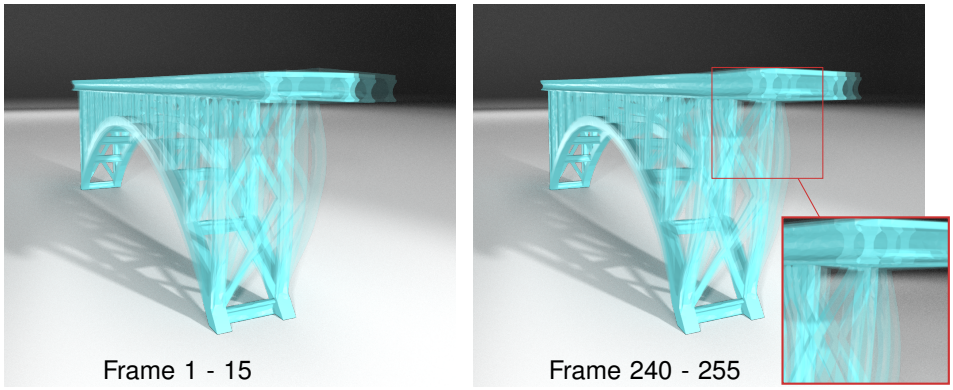
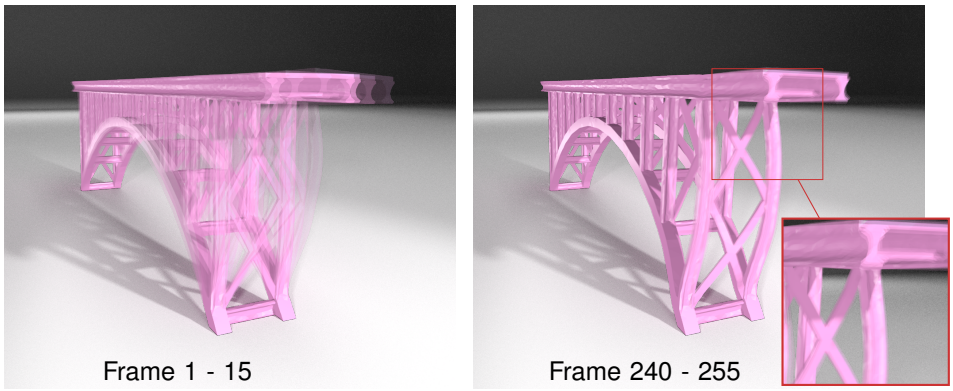


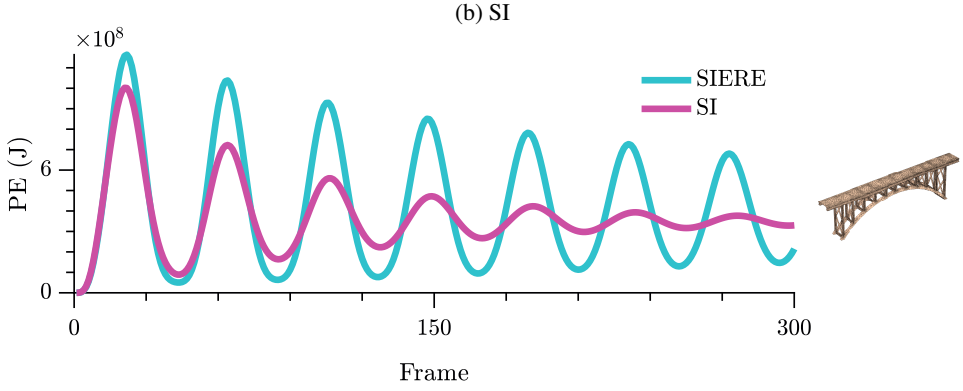
Figure 6.13: Simulation of an Eiffel Tower mesh: energy plot.



(a) SIERE



(b) SI



(c) Potential Energy

Figure 6.14: Simulation of a bridge deforming under strong wind: (a) SIERE remains dynamic with visible oscillations after 200 frames; whereas (b) oscillations under SI are heavily damped. A potential energy plot (c) underscores this observation.

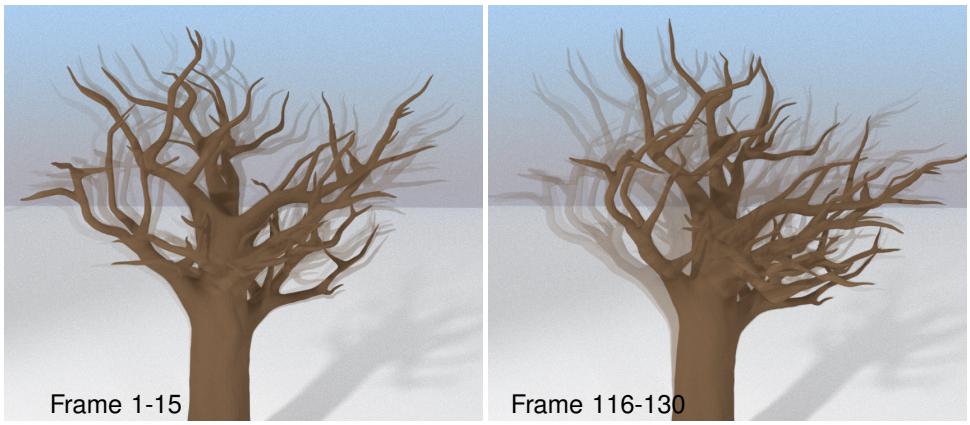


Figure 6.15: Simulation of a tree swaying in strong wind.

Chapter 7

Conclusion and future work

7.1 Conclusions

In this thesis we have examined the challenges in elastodynamic simulation in both spatial and temporal discretization and presented novel numerical methods to improve existing algorithms.

For the spatial discretization, we analyzed and presented numerical stiffening, a phenomenon that occurs during mesh coarsening, through a fundamental finite element method limitation. Galerkin approximation of the general eigenvalue problem will always lead to larger eigenvalues due to an increasingly less inclusive function space as we coarsen the mesh. Since the eigenvalues represent the oscillation frequencies in elastodynamics systems, numerical stiffening leads to inaccurate dynamic behavior as we coarsen the mesh. To address this issue, we developed EigenFit, an algorithm that improves the consistency in dynamic simulation across mesh resolutions. Using a partial mass-PCA, EigenFit matches the leading eigenvalues of a given coarse mesh with those of a fine reference mesh at rest. For nonlinear constitutive laws mass-PCA must be carried out at each time step. We have demonstrated the effectiveness of EigenFit on a range of different meshes for both homogeneous and heterogeneous material objects.

Next we demonstrated the difficulty in time discretization, or time integration, for stiff material. In Chapter 4 we extended the use of exponential integrators, a well-known class of integrators for stiff problems, to general elastodynamics sys-

tems. In particular, we described how to use the ERE integrator for challenging nonlinear simulations when the tangent stiffness matrix is potentially stiff and not always SPD. We demonstrated that ERE can be very effective because it does not introduce excessive damping even when using large time steps, and does not require solution to nonlinear algebraic equations. However, the cost of ERE increases with the stiffness of the elastic object simulated.

In Chapter 5 we introduced an additive method framework which may be used to compose new methods coupling two integrators together. We demonstrated that in elastodynamics, systems can often be split into stiff and non-stiff parts and treated effectively using our additive method.

Built on top of the previous chapters, we introduced another integrator, SIERE, in Chapter 6. In a manner similar to what we did in Chapter 3, we use a partial mass-PCA to split the low frequency modes from the rest in any given elastodynamics system. The non-stiff, low frequency part is stepped forward by ERE, while the rest of the system is handled by SI. The computational load of ERE is further reduced due to diagonalization in the partial mass-PCA. In addition, the resulting method requires no solution of nonlinear algebraic equations and is less prone to divergence and instabilities than both ERE and SI.

Both EigenFit and SIERE use partial mass-PCA as one of the key components of the algorithm. The main observation behind this is that a large component of observable elastic motions is represented by the low frequency modes, and the high frequency modes quickly die out due to internal friction. However, EigenFit has large deformation for nonlinear models as a major limitation, while SIERE has been shown effective even under large deformation. This observed difference is due to the fact that EigenFit centers on modification to the low eigenvalues, but for nonlinear models eigenvalues may cross, and the relative stiffening factors for the changing modes are hard to track due to nonlinear terms that may be large in comparison to these low eigenvalues. On the other hand, in SIERE we merely construct a subspace of the low frequency modes, subsequently applying a different integration method there, and keep the eigenvalues and eigenvectors untouched. SIERE also achieves superior stability by using ERE to warm-start SI. In addition, SIERE elegantly combines the subspace deformation with the global deformation by a low-rank update, and avoids global deformation artifacts from traditional lin-

ear subspace methods [39].

7.2 Future work

Several potential avenues for future work arise from this thesis.

1. Extending EigenFit to obtain better accuracy for nonlinear models with large deformation. This may be achieved by investigating the kinematic eigenvalues ([26, 59]) of the coarse and fine systems. Other, data based methods may be used as well.

2. Extending SIERE, a first order method, to higher order methods for stiff elastodynamics problem can be an interesting direction. Solving nonlinear algebraic equations may become an issue here.

3. SIERE is built based on the additive method described and demonstrated in Chapter 5. It is possible to apply SIERE for the mass-spring system and DG formulation described in Chapter 5, which can expand its applicability.

4. The methods presented in this thesis focus on the discretization of elastodynamics models. We have demonstrated how the methods respond to collision and changing boundary conditions. Further experiments with real-time user interaction can potentially expand the range of applications for our methods.

5. Some applications may require high accuracy for energy or momentum conservation properties. We can potentially combine ERE with other integrators, such as implicit midpoint, to achieve higher precision for such applications.

Bibliography

[1] R. Akhtar, M. J. Sherratt, J. K. Cruickshank, and B. Derby. Characterizing the elastic properties of tissues. *Materials Today*, 14(3):96–105, 2011. → page 52

[2] A. H. Al-Mohy and N. J. Higham. Computing the action of the matrix exponential, with an application to exponential integrators. *SIAM journal on scientific computing*, 33(2):488–511, 2011. → page 50

[3] U. Ascher. *Numerical Methods for Evolutionary Differential Equations*, volume 5. SIAM, 2008. → pages 3, 56, 59, 68, 79, 81

[4] U. Ascher, S. Ruuth, and B. Wetton. Implicit-explicit methods for time-dependent partial differential equations. *SIAM journal on numerical analysis*, 32(3):797–823, 1995. → pages 68, 81

[5] I. Babuška and M. Zlámal. Nonconforming elements in the finite element method with penalty. *SIAM Journal on Numerical Analysis*, 10(5):863–875, 1973. → page 73

[6] I. Babuska, J. E. Flaherty, W. D. Henshaw, J. E. Hopcroft, J. E. Oliger, and T. Tezduyar. *Modeling, mesh generation, and adaptive numerical methods for partial differential equations*, volume 75. Springer, 2012. → page 22

[7] D. Baraff and A. Witkin. Large steps in cloth simulation. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, pages 43–54, New York, NY, USA, 1998. ACM. ISBN 0-89791-999-8. doi:10.1145/280814.280821. → pages 3, 59, 77

[8] J. Barbic and D. James. Real-time subspace integration for st. venant-kirchhoff deformable models. *ACM Trans. Graphics*, 24(4):982–990, 2005. → page 26

[9] A. W. Bargteil and E. Cohen. Animation of deformable bodies with quadratic bézier finite elements. *ACM Trans. Graph. (TOG)*, 33(3), June 2014. → page 22

[10] T. Belytschko, W. K. Liu, B. Moran, and K. Elkhodary. *Nonlinear Finite Elements for Continua and Structures*. John Wiley & Sons, Nov. 2013. → pages 1, 18, 22

[11] S. Bouaziz, S. Martin, T. Liu, L. Kavan, and M. Pauly. Projective dynamics: fusing constraint projections for fast simulation. *ACM Transactions on Graphics (TOG)*, 33(4):154, 2014. → page 12

[12] E. Boxerman and U. Ascher. Decomposing cloth. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 153–161. Eurographics Association, 2004. → page 81

[13] R. Bridson, S. Marino, and R. Fedkiw. Simulation of clothing with folds and wrinkles. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 28–36. Eurographics Association, 2003. → pages 12, 60

[14] M. Caliari and A. Ostermann. Implementation of exponential Rosenbrock-type integrators. *Applied Numerical Mathematics*, 59(3): 568–581, 2009. → page 50

[15] I. Chao, U. Pinkall, P. Sanan, and P. Schröder. A simple geometric model for elastic deformations. *ACM Trans. Graph. (TOG)*, 29(4):38, 2010. → pages 37, 90, 92

[16] D. Chen, D. I. W. Levin, S. Sueda, and W. Matusik. Data-driven finite elements for geometry and material design. *ACM Trans. Graph.*, 34(4): 74:1–74:10, July 2015. ISSN 0730-0301. → page 22

[17] D. Chen, D. I. W. Levin, W. Matusik, and D. M. Kaufman. Dynamics-aware numerical coarsening for fabrication design. *ACM Trans. Graph.*, 36(4): 84:1–84:15, July 2017. ISSN 0730-0301. doi:10.1145/3072959.3073669. URL http://doi.acm.org/10.1145/3072959.3073669. → pages 4, 14, 21, 22, 24, 26, 27, 46, 78, 93

[18] J. Chen, H. Bao, W. Tianyu, M. Desbrun, and J. Huang. Numerical coarsening using discontinuous shape functions. *ACM Trans. Graphics (TOG)*, 37(4), August 2018. → pages 14, 22, 23

[19] Y. J. Chen, U. Ascher, and D. Pai. Exponential rosenbrock-euler integrators for elastodynamic simulation. *IEEE Transactions on Visualization and Computer Graphics*, 2017. → pages 14, 78, 79, 81

[20] J. Chung and G. Hulbert. A time integration algorithm for structural dynamics with improved numerical dissipation: the generalized-α method. *J. Applied Mech.*, 60:371–375, 1993. → page 78

[21] P. G. Ciarlet. *Three-dimensional elasticity*, volume 20. Elsevier, 1988. → page 27

[22] P. G. Ciarlet. *Mathematical Elasticity: Three-Dimensional Elasticity*, volume 1. Elsevier, 1993. → page 7

[23] P. G. Ciarlet, M. H. Schultz, and R. S. Varga. Numerical methods of high-order accuracy for nonlinear boundary value problems iv. periodic boundary conditions. *Numer. Math.*, 12(4):266–279, Nov. 1968. ISSN 0029-599X. doi:10.1007/BF02162508. URL http://dx.doi.org/10.1007/BF02162508. → page 16

[24] A. De Coninck, B. De Baets, D. Kourounis, F. Verbosio, O. Schenk, S. Maenhout, and J. Fostier. Needles: Toward large-scale genomic prediction with marker-by-environment interaction. 203(1):543–555, 2016. ISSN 0016-6731. doi:10.1534/genetics.115.179887. URL http://dx.doi.org/10.1534/genetics.115.179887. → pages 12, 86

[25] P. Deuflhard. *Newton Methods for Nonlinear Problems*. Springer, 2011. → page 87

[26] L. Dieci, R. D. Russell, and E. S. Van Vleck. On the compuation of lyapunov exponents for continuous dynamical systems. *SIAM journal on numerical analysis*, 34(1):402–423, 1997. → page 104

[27] D. Dinev, T. Liu, and L. Kavan. Stabilizing integrators for real-time physics. *ACM Transactions on Graphics*, 37(1):1–19, Jan 2018. ISSN 0730-0301. doi:10.1145/3153420. URL http://dx.doi.org/10.1145/3153420. → pages 12, 81

[28] D. Dinev, T. Liu, J. Li, B. Tomaszevski, and L. Kavan. Fepr: Fast energy projection for real-time simulation of deformable objects. *ACM Trans. Graph.*, 37(4), 2018. → page 81

[29] B. Eberhardt, O. Etzmuß, and M. Hauth. *Implicit-explicit schemes for fast animation with particle systems*. Springer, 2000. → pages 12, 81

[30] T. F. Gast, C. Schroeder, A. Stomakhin, C. Jiang, and J. M. Teran. Optimization integrator for large time steps. *IEEE Trans. Visualization and Computer Graphics*, 21(10):1103–1115, 2015. → page 12

[31] R. Goldenthal, D. Harmon, R. Fattal, M. Bercovier, and E. Grinspun. Efficient simulation of inextensible cloth. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007)*, 26(3):to appear, 2007. → page 60

[32] E. Grinspun, P. Krysl, and P. Schröder. Charms: A simple framework for adaptive simulation. *ACM Trans. Graph. (SIGGRAPH 2002)*, 21(3), July 2002. → page 22

[33] E. Hairer and G. Wanner. Stiff differential equations solved by radau methods. *Journal of Computational and Applied Mathematics*, 111(1-2): 93–111, 1999. → page 68

[34] M. Hauth and O. Etzmuss. A high performance solver for the animation of deformable objects using advanced numerical methods. In *Computer Graphics Forum*, volume 20, pages 319–328. Wiley Online Library, 2001. → page 12

[35] M. Hochbruck and A. Ostermann. Exponential integrators. *Acta Numerica*, 19:209–286, 2010. → pages 14, 48, 49, 52

[36] M. Hochbruck, A. Ostermann, and J. Schweitzer. Exponential Rosenbrock-type methods. *SIAM Journal on Numerical Analysis*, 47(1): 786–803, 2009. → page 14

[37] Y. Hu, Q. Zhou, X. Gao, A. Jacobson, D. Zorin, and D. Panozzo. Tetrahedral meshing in the wild. *ACM Trans. Graph.*, 37(4), 2018. → pages 9, 33

[38] T. M. Inc. Matlab optimization toolbox, R2018a. → page 14

[39] D. L. James and D. K. Pai. Dyrt: dynamic response textures for real time deformation simulation with graphics hardware. In *ACM Transactions on Graphics (TOG)*, volume 21, pages 582–585. ACM, 2002. → pages 89, 104

[40] A.-K. Kassam and L. N. Trefethen. Fourth-order time-stepping for stiff PDEs. *SIAM Journal on Scientific Computing*, 26(4):1214–1233, 2005. → page 50

[41] P. Kaufmann. *Discontinuous Galerkin FEM in Computer Graphics*. PhD thesis, ETH Zurich, 2012. → pages 22, 23, 73

[42] P. Kaufmann, S. Martin, M. Botsch, and M. Gross. Flexible simulation of deformable models using discontinuous galerkin fem. *Graphical Models*, 71 (4):153–167, 2009. → pages 22, 23

[43] L. Kharevych, W. Yang, Y. Tong, E. Kanso, J. E. Marsden, P. Schröder, and M. Desbrun. Geometric, variational integrators for computer animation. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 43–51. Eurographics Association, 2006. → pages 12, 81

[44] L. Kharevych, P. Mullen, H. Owhadi, and M. Desbrun. Numerical coarsening of inhomogeneous elastic materials. *ACM Trans. on Graphics*, 28(3):51:1–51:8, 2009. → pages 4, 14, 22

[45] D. A. Knoll and D. E. Keyes. Jacobian-free Newton–Krylov methods: a survey of approaches and applications. *Journal of Computational Physics*, 193(2):357–397, 2004. → page 12

[46] B. C. W. Kot, Z. J. Zhang, A. W. C. Lee, V. Y. F. Leung, and S. N. Fu. Elastic modulus of muscle and tendon with shear wave ultrasound elastography: variations with different technical settings. *PloS one*, 7(8): e44348, 2012. → page 52

[47] D. Kourounis, A. Fuchs, and O. Schenk. Towards the next generation of multiperiod optimal power flow solvers. *IEEE Transactions on Power Systems*, PP(99):1–10, 2018. ISSN 0885-8950. doi:10.1109/TPWRS.2017.2789187. URL https://doi.org/10.1109/TPWRS.2017.2789187. → pages 12, 86

[48] R. Lehoucq, D. Sorensen, and C. Yang. Arpack users' guide: Solution of large scale eigenvalue problems with implicitly restarted arnoldi methods. *Software Environ. Tools*, 6, 1997. → page 23

[49] D. Levin. Gaggle of algorithms and utilities for simulating stuff. Github Repository, 2017. URL https://github.com/dilevin/GAUSS. → page vi

[50] D. Li, Y. Fei, and C. Zheng. Interactive acoustic transfer approximation for modal sound. *ACM Trans. Graph.*, 35(1), 2015. doi:10.1145/2820612. → page 34

[51] J. Li, T. Liu, and L. Kavan. Laplacian damping for projective dynamics. In *VRIPHYS2018: 14th Workshop on Virtual Reality Interaction and Physical Simulation*, 2018. → page 11

[52] X. Liang and S. A. Boppart. Biomechanical properties of in vivo human skin from dynamic optical coherence elastography. *Biomedical Engineering, IEEE Transactions on*, 57(4):953–959, 2010. → page 52

[53] T. Liu, A. W. Bargteil, J. F. O'Brien, and L. Kavan. Fast simulation of mass-spring systems. *ACM Transactions on Graphics (TOG)*, 32(6):214, 2013. → pages 12, 51

[54] T. Liu, S. Bouaziz, and L. Kavan. Quasi-newton methods for real-time simulation of hyperelastic materials. *ACM Transactions on Graphics (TOG)*, 36(3):23, 2017. → page 56

[55] T. Liu, S. Bouaziz, and L. Kavan. Towards real-time simulation of hyperelastic materials. *ACM Transactions on Graphics (TOG)*, 36(3):23, 2017. → page 12

[56] J. Loffeld and M. Tokman. Comparative performance of exponential, implicit, and explicit integrators for stiff systems of ODEs. *Journal of Computational and Applied Mathematics*, 241:45–67, 2013. → page 14

[57] V. T. Luan, M. Tokman, and G. Rainwater. Preconditioned implicit-exponential integrators (imexp) for stiff pdes. *J. Computational Physics*, 335(15):846–864, 2017. → page 81

[58] S. Martin, B. Thomaszewski, E. Grinspun, and M. Gross. Example-based elastic materials. *ACM Transactions on Graphics (TOG)*, 30(4):72, 2011. → pages 12, 51, 56

[59] K. Mease, S. Bharadwaj, and S. Iravanchy. Timescale analysis for nonlinear dynamical systems. *Journal of guidance, control, and dynamics*, 26(2): 318–330, 2003. → page 104

[60] D. L. Michels, G. Sobottka, and A. Weber. Exponential integrators for stiff elastodynamic problems. *ACM Trans. Graph.*, 33(1), Jan. 2014. doi:10.1145/2508462. → pages 14, 54, 81

[61] D. L. Michels, J. P. T. Mueller, and G. A. Sobottka. A physically based approach to the accurate simulation of stiff fibers and stiff fiber meshes. *Computers & Graphics*, 53B:136–146, Dec. 2015.

[62] D. L. Michels, V. T. Luan, and M. Tokman. A stiffly accurate integrator for elastodynamic problems. *ACM Trans. Graph.*, 36(4), August 2017. → pages 14, 49, 78, 81

[63] C. Moler and C. Van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Rev.*, 45(1):3–49, 2003. → pages 48, 50

[64] J. Mosler and M. Ortiz. Variational h-adaption in finite deformation elasticity and plasticity. *International Journal for Numerical Methods in Engineering*, 72(5):505–523, 2007. → page 22

[65] R. Narain, M. Overby, and G. E. Brown. ADMM ⊇ projective dynamics: Fast simulation of general constitutive models. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '16, pages 21–28, Aire-la-Ville, Switzerland, Switzerland, 2016. Eurographics Association. ISBN 978-3-905674-61-3. URL http://dl.acm.org/citation.cfm?id=2982818.2982822. → page 12

[66] M. Nesme, P. G. Kry, L. Jeřábková, and F. Faure. Preserving topology and elasticity for embedded deformable models. *ACM Trans. Graph.*, 28(3): 52:1–52:9, July 2009. ISSN 0730-0301. → pages 4, 14, 22

[67] J. Niesen and W. M. Wright. Algorithm 919: A Krylov subspace algorithm for evaluating the ϕ-functions appearing in exponential integrators. *ACM Trans. Math. Software (TOMS)*, 38(3):article 22, 2012. → page 50

[68] J. Nocedal and S. Wright. *Numerical Optimization*. New York: Springer, 2006. 2nd Ed. → page 30

[69] J. Panetta, Q. Zhou, L. Malomo, N. Pietroni, P. Cignoni, and D. Zorin. Elastic textures for additive fabrication. *ACM Trans. on Graphics*, 34(4): 135:1–135:12, 2015. → page 22

[70] M. Piovarči, D. I. Levin, D. Kaufman, and P. Didyk. Perception-aware modeling and fabrication of digital drawing tools. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 37(4), 2018. → page 14

[71] Y. Qiu. Spectra: C++ library for large scale eigenvalue problems, 2015-2019. URL https://spectralib.org/. → page 23

[72] Y. Saad. Analysis of some Krylov subspace approximations to the matrix exponential operator. *SIAM Journal on Numerical Analysis*, 29(1):209–228, 1992. → page 50

[73] L. Sacht, E. Vouga, and A. Jacobson. Nested cages. *ACM Transactions on Graphics (TOG)*, 34(6), 2015. → page 33

[74] T. Schneider, Y. Hu, J. Dumas, X. Gao, D. Panozzo, and D. Zorin. Decoupling simulation accuracy from mesh quality. In *SIGGRAPH Asia 2018 Technical Papers*, SIGGRAPH Asia '18, pages 280:1–280:14, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-6008-1. doi:10.1145/3272127.3275067. URL http://doi.acm.org/10.1145/3272127.3275067. → page 22

[75] D. Schroeder. *An Introduction to Thermal Physics*. Addison Wesley, 1999. ISBN 9780201380279. URL https://books.google.ca/books?id=1gosQgAACAAJ. → page 10

[76] H. Si. Tetgen, a delaunay-based quality tetrahedral mesh generator. *ACM Transactions on Mathematical Software*, 41(2):1–36, Feb 2015. ISSN 0098-3500. doi:10.1145/2629697. URL http://dx.doi.org/10.1145/2629697. → page 9

[77] R. B. Sidje. Expokit: a software package for computing matrix exponentials. *ACM Transactions on Mathematical Software (TOMS)*, 24(1):130–156, 1998. → pages 50, 51

[78] E. Sifakis and J. Barbic. FEM simulation of 3D deformable solids: a practitioner's guide to theory, discretization and model reduction. In *ACM SIGGRAPH 2012 Courses*, page 20. ACM, 2012. → pages 1, 18, 23

[79] D. C. Sorensen. Implicitly restarted arnoldi/lanczos methods for large scale eigenvalue calculations. In *Parallel Numerical Algorithms*, pages 119–165. Springer, 1997. → page 23

[80] O. Sorkine and M. Alexa. As-rigid-as-possible surface modeling. *Eurographics Symposium on Geometry Processing*, 4:109–116, 2007. → pages 90, 92

[81] A. Stern and E. Grinspun. Implicit-explicit variational integration of highly oscillatory problems. *Multiscale Modeling & Simulation*, 7(4):1779–1794, 2009. → page 81

[82] G. W. Stewart. A krylov–schur algorithm for large eigenproblems. *SIAM Journal on Matrix Analysis and Applications*, 23(3):601–614, 2002. → page 28

[83] J. Su, R. Sheth, and R. Fedkiw. Energy conservation for the simulation of deformable bodies. *IEEE Visualization and Computer Graphics*, 19(2), 2013. → page 81

[84] J. Teran, E. Sifakis, G. Irving, and R. Fedkiw. Robust quasistatic finite elements and flesh simulation. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 181–190. ACM, 2005. → page 27

[85] R. Torres, A. Rodríguez, J. M. Espadero, and M. A. Otaduy. High-resolution interaction with corotational coarsening models. *ACM Trans. Graph.*, 35(6): 211:1–211:11, Nov. 2016. ISSN 0730-0301. → page 22

[86] M. Tournier, M. Nesme, B. Gilles, and F. Faure. Stable constrained dynamics. *ACM Trans. Graphics*, 34(4), 2015. → pages 3, 72, 77

[87] U. Trottenberg, C. W. Oosterlee, and A. Schuller. *Multigrid*. Academic press, 2000. → page 38

[88] F. Verbosio, A. D. Coninck, D. Kourounis, and O. Schenk. Enhancing the scalability of selected inversion factorization algorithms in genomic prediction. *Journal of Computational Science*, 22(Supplement C):99 – 108, 2017. ISSN 1877-7503. URL https://doi.org/10.1016/j.jocs.2017.08.013. → pages 12, 86

[89] B. Wang and X. Wu. A new high presicion energy-preserving integrator for system of oscillatory second-order differential equations. *Physics Letters A*, 376, 2012. doi:10.1145/2508462. → page 49

[90] B. Wang, L. Wu, K. Yin, U. M. Ascher, L. Liu, and H. Huang. Deformation capture and modeling of soft objects. *ACM Trans. Graph.*, 34(4):94–1, 2015. → page 42

[91] H. Xu and J. Barbič. Pose-space subspace dynamics. *ACM Transactions on Graphics (TOG)*, 35(4):35, 2016. → pages 23, 33

[92] H. Xu and J. Barbič. Example-based damping design. *ACM Transactions on Graphics (TOG)*, 36(4):53, 2017. → page 11

[93] H. Xu, F. Sin, Y. Zhu, and J. Barbič. Nonlinear material design using principal stretches. *ACM Transactions on Graphics (TOG)*, 34(4):75, 2015. → page 26

Appendix

DG energy, force, stiffness matrix

This appendix provides further details associated with Section 5.2.

A.1 BZ flux

Following the BZ method we have the energy

$$a_{BZ}(u,u) = \int_{\Omega} \sigma(u) : \varepsilon(u) + \int_{\Gamma} \eta_f \|u_- - u_+\|^2$$

Here we derive the force and the stiffness matrix associated with the second term.

A.1.1 Interface energy

Let's ignore the parameter η_f and look at one interface f (an edge in this case) and the two neighboring elements. Now the energy associate with f is

$$E_f(U_-, U_+) = \int_f \|u_- - u_+\|^2 d\boldsymbol{X}$$

where $U_\pm$ are the six vertices associated with the two neighboring triangles, 3 in U_- and 3 in U_+. For linear elements the deformations $u_\pm$ are

$$u_\pm(\boldsymbol{X}) = F_\pm X + b_\pm$$

with

$$\begin{aligned} F_{\pm} &= D_{\pm,s} D_{\pm,m}^{-1} \\ D_s &= \begin{bmatrix} x_1 - x_3 & x_2 - x_3 \\ y_1 - y_3 & y_2 - y_3 \end{bmatrix} \\ D_m &= \begin{bmatrix} X_1 - X_3 & X_2 - X_3 \\ Y_1 - Y_3 & Y_2 - Y_3 \end{bmatrix} \\ &= U^T G \end{aligned} \tag{1}$$

Thus the energy can be rewritten as

$$\begin{aligned} E_f(U_-, U_+) &= \int_f \|u_- - u_+\|^2 d\boldsymbol{X} \\ &= \int_f X^T (F_- - F_+)^T (F_- - F_+) X \\ &\quad + (b_- - b_+)^T (F_- - F_+) X \\ &\quad + X^T (F_- - F_+)^T (b_- - b_+) \\ &\quad + (b_- - b_+)^T (b_- - b_+) d\boldsymbol{X} \end{aligned}$$

A.1.2 Interface Integration

Since we choose linear basis functions, the integral above involves a line integral of a quadratic and a linear function. The quadratic integral is of the form

$$\begin{aligned} \int_{P_0}^{P_1} X^T A X dP &= \int_{P_0}^{P_1} \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} d\boldsymbol{X} \\ &= \int_{P_0}^{P_1} a_{11} x^2 + (a_{12} + a_{21}) xy + a_{22} y^2 d\boldsymbol{X} \end{aligned}$$

Remember all the a_{ij}'s are the components of the deformation graidents, which are linear functions of U_- and U_+, and thus remain constant on the line from P_0 to

P_1. Now we can parametrize the line integral by

$$\begin{aligned} x &= x_0 + t(x_1 - x_0), \\ y &= y_0 + t(y_1 - y_0). \\ P_0 &= \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}, \\ P_1 &= \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}. \end{aligned}$$

The integral is now

$$\begin{aligned} &\int_{P_0}^{P_1} X^T A X d\boldsymbol{X} = \|P_1 - P_0\| \int_0^1 a_{11}(x_0 + t(x_1 - x_0))^2 \\ &+ (a_{12} + a_{21})(x_0 + t(x_1 - x_0))(y_0 + t(y_1 - y_0)) + a_{22}(y_0 + t(y_1 - y_0))^2 dt \qquad (2) \\ =&\|P_1 - P_0\|[\frac{a_{11}}{3}(x_1^2 + x_1 x_0 + x_0^2) + \frac{a_{22}}{3}(y_1^2 + y_1 y_0 + y_0^2) \\ &+ (a_{12} + a_{21})(\frac{1}{6}x_0 y_1 + \frac{1}{6}y_0 x_1 + \frac{1}{3}x_1 y_1 + \frac{1}{3}x_0 y_0)]. \end{aligned}$$

Following the same procedure we can integrate the the linear function

$$\int_{P_0}^{P_1} \begin{pmatrix} B_{11} & B_{12} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} dP$$

$$= \|P_1 - P_0\|[B_{11}x_0 + B_{12}y_0 + \tfrac{1}{2}B_{11}(x_1 - x_0) + \tfrac{1}{2}B_{12}(y_1 - y_0)] \qquad (3)$$

And of course

$$\int_{P_0}^{P_1} c dP = c\|P_1 - P_0\|$$

A.1.3 Interface Force

To assemble the force, we have the following derivatives

$$\begin{aligned}
\frac{\partial}{\partial x_{-,i}^{(j)}} E_f &= \int_f X^T \left(\frac{\partial}{\partial x_{-,i}^{(j)}} F_-^T F_- + F_-^T \frac{\partial}{\partial x_{-,i}^{(j)}} F_- - \frac{\partial}{\partial x_{-,i}^{(j)}} F_-^T F_+ - F_+^T \frac{\partial}{\partial x_{-,i}^{(j)}} F_- \right) X \\
&+ \left((b_- - b_+)^T \left(\frac{\partial}{\partial x_{-,i}^{(j)}} F_- \right) + \left(\frac{\partial}{\partial x_{-,i}^{(j)}} b_- \right)^T (F_- - F_+) \right) X \\
&+ X^T \left(\left(\frac{\partial}{\partial x_{-,i}^{(j)}} F_- \right)^T (b_- - b_+) + (F_- - F_+)^T \left(\frac{\partial}{\partial x_{-,i}^{(j)}} b_- \right) \right) \\
&+ \left(\left(\frac{\partial}{\partial x_{-,i}^{(j)}} b_- \right)^T (b_- - b_+) + (b_- - b_+)^T \left(\frac{\partial}{\partial x_{-,i}^{(j)}} b_- \right) \right) d\boldsymbol{X}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial}{\partial x_{+,i}^{(j)}} E_f &= \int_f X^T \left(\frac{\partial}{\partial x_{+,i}^{(j)}} F_+^T F_+ + F_+^T \frac{\partial}{\partial x_{+,i}^{(j)}} F_+ - \frac{\partial}{\partial x_{+,i}^{(j)}} F_+^T F_- - F_-^T \frac{\partial}{\partial x_{+,i}^{(j)}} F_+ \right) X \\
&+ \left((b_- - b_+)^T \left(-\frac{\partial}{\partial x_{+,i}^{(j)}} F_+ \right) + \left(-\frac{\partial}{\partial x_{+,i}^{(j)}} b_+ \right)^T (F_- - F_+) \right) X \\
&X^T \left(\left(-\frac{\partial}{\partial x_{+,i}^{(j)}} F_+ \right)^T (b_- - b_+) + (F_- - F_+)^T \left(-\frac{\partial}{\partial x_{+,i}^{(j)}} b_+ \right) \right) \\
&+ \left(\left(-\frac{\partial}{\partial x_{+,i}^{(j)}} b_+ \right)^T (b_- - b_+) + (b_- - b_+)^T \left(-\frac{\partial}{\partial x_{+,i}^{(j)}} b_+ \right) \right) d\boldsymbol{X}
\end{aligned}$$

where $i \in [1,2,3]$ are the indices to the three vertices per element, and $j \in [1,2]$ are the x and y components, respectively. From Eq. (1) we have

$$\frac{\partial}{\partial x_{\pm,i}^{(j)}} F_\pm = \boldsymbol{e}_j \boldsymbol{e}_i^T D_{\pm,m}^{-1}, \quad i,j \in [1,2]$$

$$\frac{\partial}{\partial x^{(j)}_{\pm,3}} F_{\pm} \quad = \boldsymbol{e}_j \begin{pmatrix} -1 & -1 \end{pmatrix} D^{-1}_{\pm,m}, \quad j \in [1,2]$$

$$\frac{\partial}{\partial x^{(j)}_{\pm,i}} F^T_{\pm} \quad = D^{-T}_{\pm,m} \boldsymbol{e}_i \boldsymbol{e}^T_j, \quad i,j \in [1,2]$$

$$\frac{\partial}{\partial x^{(j)}_{\pm,3}} F^T_{\pm} \quad = D^{-T}_{\pm,m} \boldsymbol{e}_i \begin{pmatrix} -1 & -1 \end{pmatrix}, \quad j \in [1,2]$$

where $\boldsymbol{e}_i$ is a unit 2-vecotr. There are many ways to calculate $\frac{\partial}{\partial x^{(j)}_{\pm,i}} b_{\pm}$, here we use the barycentric coordinate at $(1\ 0\ 0)^T$

$$\begin{aligned}
\frac{\partial}{\partial x^{(j)}_1} b &= \boldsymbol{e}_j - \boldsymbol{e}_j \boldsymbol{e}^T_1 D^{-1}_m \begin{pmatrix} X_1 \\ Y_1 \end{pmatrix}, \ j \in [1,2] \\
\frac{\partial}{\partial x^{(j)}_2} b &= -\boldsymbol{e}_j \boldsymbol{e}^T_2 D^{-1}_m \begin{pmatrix} X_1 \\ Y_1 \end{pmatrix}, \ j \in [1,2] \\
\frac{\partial}{\partial x^{(j)}_3} b &= -\boldsymbol{e}_j \begin{pmatrix} -1 & -1 \end{pmatrix} D^{-1}_m \begin{pmatrix} X_1 \\ Y_1 \end{pmatrix}, \ j \in [1,2]
\end{aligned}$$

A.1.4 Interface Stiffness matrix

The interface stiffness matrix has four components

$$\begin{aligned}
\frac{\partial}{\partial x^{(l)}_{-,k}}\frac{\partial}{\partial x^{(j)}_{-,i}}E_f &= \int_f X^T\left(\frac{\partial}{\partial x^{(j)}_{-,i}}F_-^T\frac{\partial}{\partial x^{(l)}_{-,k}}F_- + \frac{\partial}{\partial x^{(l)}_{-,k}}F_-^T\frac{\partial}{\partial x^{(j)}_{-,i}}F_-\right)Xd\boldsymbol{X} \\
&+ \int_f\left(\frac{\partial}{\partial x^{(l)}_{-,k}}b_-\right)^T\left(\frac{\partial}{\partial x^{(j)}_{-,i}}F_-\right) + \left(\frac{\partial}{\partial x^{(j)}_{-,i}}b_-\right)^T\left(\frac{\partial}{\partial x^{(l)}_{-,k}}F_-\right)Xd\boldsymbol{X} \\
&+ \int_f X^T\left(\left(\frac{\partial}{\partial x^{(j)}_{-,i}}F_-\right)^T\left(\frac{\partial}{\partial x^{(l)}_{-,k}}b_-\right) + \left(\frac{\partial}{\partial x^{(l)}_{-,k}}F_-\right)^T\left(\frac{\partial}{\partial x^{(j)}_{-,i}}b_-\right)\right)d\boldsymbol{X} \\
&+ \int_f\left(\frac{\partial}{\partial x^{(j)}_{-,i}}b_-\right)^T\left(\frac{\partial}{\partial x^{(l)}_{-,k}}b_-\right) + \left(\frac{\partial}{\partial x^{(l)}_{-,k}}b_-\right)^T\left(\frac{\partial}{\partial x^{(j)}_{-,i}}b_-\right)d\boldsymbol{X}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial}{\partial x^{(l)}_{+,k}}\frac{\partial}{\partial x^{(j)}_{-,i}}E_f &= \int_f X^T\left(-\frac{\partial}{\partial x^{(j)}_{-,i}}F_-^T\frac{\partial}{\partial x^{(l)}_{+,k}}F_+ - \frac{\partial}{\partial x^{(l)}_{+,k}}F^{+,T}\frac{\partial}{\partial x^{(j)}_{-,i}}F_-\right)Xd\boldsymbol{X} \\
&+ \int_f\left(-\frac{\partial}{\partial x^{(l)}_{+,k}}b_+\right)^T\left(\frac{\partial}{\partial x^{(j)}_{-,i}}F_-\right) + \left(\frac{\partial}{\partial x^{(j)}_{-,i}}b_-\right)^T\left(-\frac{\partial}{\partial x^{(l)}_{+,k}}F_+\right)Xd\boldsymbol{X} \\
&+ \int_f X^T\left(\frac{\partial}{\partial x^{(j)}_{-,i}}F_-\right)^T\left(-\frac{\partial}{\partial x^{(l)}_{+,k}}b_+\right) + \left(-\frac{\partial}{\partial x^{(l)}_{+,k}}F_+\right)^T\left(\frac{\partial}{\partial x^{(j)}_{-,i}}b_-\right)d\boldsymbol{X} \\
&+ \int_f\left(\frac{\partial}{\partial x^{(j)}_{-,i}}b_-\right)^T\left(-\frac{\partial}{\partial x^{(l)}_{+,k}}b_+\right) + \left(-\frac{\partial}{\partial x^{(l)}_{+,k}}b_+\right)^T\left(\frac{\partial}{\partial x^{(j)}_{-,i}}b_-\right)d\boldsymbol{X}
\end{aligned}$$

$$
\begin{aligned}
\frac{\partial}{\partial x^{(l)}_{-,k}}\frac{\partial}{\partial x^{(j)}_{+,i}}E_f &= \int_f X^T\left(-\frac{\partial}{\partial x^{(j)}_{+,i}}F_+^T\frac{\partial}{\partial x^{(l)}_{-,k}}F_- - \frac{\partial}{\partial x^{(l)}_{-,k}}F_-^T\frac{\partial}{\partial x^{(j)}_{+,i}}F_+\right)X d\boldsymbol{X} \\
&+ \int_f \left(\frac{\partial}{\partial x^{(l)}_{-,k}}b_-\right)^T\left(-\frac{\partial}{\partial x^{(j)}_{+,i}}F_+\right) + \left(-\frac{\partial}{\partial x^{(j)}_{+,i}}b_+\right)^T\left(\frac{\partial}{\partial x^{(l)}_{-,k}}F_-\right)X d\boldsymbol{X} \\
&+ \int_f X^T\left(-\frac{\partial}{\partial x^{(j)}_{+,i}}F_+\right)^T\left(\frac{\partial}{\partial x^{(l)}_{-,k}}b_-\right) + \left(\frac{\partial}{\partial x^{(l)}_{-,k}}F_-\right)^T\left(-\frac{\partial}{\partial x^{(j)}_{+,i}}b_+\right)d\boldsymbol{X} \\
&+ \int_f \left(-\frac{\partial}{\partial x^{(j)}_{+,i}}b_+\right)^T\left(\frac{\partial}{\partial x^{(l)}_{-,k}}b_-\right) + \left(\frac{\partial}{\partial x^{(l)}_{-,k}}b_-\right)^T\left(-\frac{\partial}{\partial x^{(j)}_{+,i}}b_+\right)d\boldsymbol{X}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial}{\partial x^{(l)}_{+,k}}\frac{\partial}{\partial x^{(j)}_{+,i}}E_f &= \int_f X^T\left(\frac{\partial}{\partial x^{(j)}_{+,i}}F_+^T\frac{\partial}{\partial x^{(l)}_{+,k}}F_+ + \frac{\partial}{\partial x^{(l)}_{+,k}}F_+^T\frac{\partial}{\partial x^{(j)}_{+,i}}F_+\right)X d\boldsymbol{X} \\
&+ \int_f \left(-\frac{\partial}{\partial x^{(l)}_{+,k}}b_+\right)^T\left(-\frac{\partial}{\partial x^{(j)}_{+,i}}F_+\right) + \left(-\frac{\partial}{\partial x^{(j)}_{+,i}}b_+\right)^T\left(-\frac{\partial}{\partial x^{(l)}_{+,k}}F_+\right)X d\boldsymbol{X} \\
&+ \int_f X^T\left(-\frac{\partial}{\partial x^{(j)}_{+,i}}F_+\right)^T\left(-\frac{\partial}{\partial x^{(l)}_{+,k}}b_+\right) + \left(-\frac{\partial}{\partial x^{(l)}_{+,k}}F_+\right)^T\left(-\frac{\partial}{\partial x^{(j)}_{+,i}}b_+\right)d\boldsymbol{X} \\
&+ \int_f \left(-\frac{\partial}{\partial x^{(j)}_{+,i}}b_+\right)^T\left(-\frac{\partial}{\partial x^{(l)}_{+,k}}b_+\right) + \left(-\frac{\partial}{\partial x^{(l)}_{+,k}}b_+\right)^T\left(-\frac{\partial}{\partial x^{(j)}_{+,i}}b_+\right)d\boldsymbol{X}
\end{aligned}
$$

with $i,k \in [1,2,3]$ and $j,l \in [1,2]$.

A.2 IP flux

The IP method for linear elasticity uses the following bilinear form

$$\begin{aligned} a_{IP}(\boldsymbol{u},\boldsymbol{v}): \quad = \quad & \int_{\Omega} \varepsilon(\boldsymbol{v}) : \boldsymbol{C} : \varepsilon(\boldsymbol{u}) \\ & - \int_{\Gamma} [\![\boldsymbol{v}]\!] : \{\sigma(\boldsymbol{u})\} \\ & - \int_{\Gamma} [\![\boldsymbol{u}]\!] : \{\sigma(\boldsymbol{v})\} \\ & + \int_{\Gamma} \eta_f [\![\boldsymbol{u}]\!] : [\![\boldsymbol{v}]\!] \end{aligned} \tag{4}$$

and the norm is simply

$$\begin{aligned} a_{IP}(\boldsymbol{u},\boldsymbol{u}): \quad = \quad & \int_{\Omega} \varepsilon(\boldsymbol{u}) : \boldsymbol{C} : \varepsilon(\boldsymbol{u}) \\ & - \int_{\Gamma} [\![\boldsymbol{u}]\!] : \{\sigma(\boldsymbol{u})\} \\ & - \int_{\Gamma} [\![\boldsymbol{u}]\!] : \{\sigma(\boldsymbol{u})\} \\ & + \int_{\Gamma} \eta_f [\![\boldsymbol{u}]\!] : [\![\boldsymbol{u}]\!] . \end{aligned} \tag{5}$$

The first term in Eq. (5) is the same elastic energy used in the usual FEM used elsewhere in the thesis, where $\Phi = \varepsilon : \boldsymbol{C} : \varepsilon$ is the energy density and

$$\boldsymbol{C} \quad = \quad \frac{\partial}{\partial \varepsilon} \frac{\partial}{\partial \varepsilon^T} \Phi$$

is the elasticity tensor. The linear strain measure is

$$\varepsilon \quad = \tfrac{1}{2}(\nabla \boldsymbol{u} + \nabla^T \boldsymbol{u}) - I \quad = \frac{1}{2}(F + F^T) - I$$

, and the stress tensor is

$$\sigma(\boldsymbol{u}) = \frac{\partial}{\partial \varepsilon} \Phi(\boldsymbol{u})$$

To generalize Eq. (5) to non-linear elasticty model, we rewrite it as

$$\begin{aligned} a_{IP}(\boldsymbol{u},\boldsymbol{u}): &= \int_\Omega \Phi(\boldsymbol{u}) \\ &\quad - \int_\Gamma [\![\boldsymbol{u}]\!] : \{P(\boldsymbol{u})\} \\ &\quad - \int_\Gamma [\![\boldsymbol{u}]\!] : \{P(\boldsymbol{u})\} \\ &\quad + \int_\Gamma \eta_f [\![\boldsymbol{u}]\!] : [\![\boldsymbol{u}]\!] \end{aligned} \tag{6}$$

where

$$P(\boldsymbol{u}) = \frac{\partial}{\partial \varepsilon}\Phi(\boldsymbol{u})$$

the 1st Piola-Kirchhoff stress tensor. In this case, the elasticity tensor becomes

$$C_P = \frac{\partial}{\partial F}\frac{\partial}{\partial F^T}\Phi$$

and we can rewrite Eq. (6) as

$$\begin{aligned} a_{IP}(\boldsymbol{u},\boldsymbol{u}): &= \int_\Omega F : C_P : F^T \\ &\quad - \int_\Gamma [\![\boldsymbol{u}]\!] : \{P(\boldsymbol{u})\} \\ &\quad + \int_\Gamma \eta_f [\![\boldsymbol{u}]\!] : [\![\boldsymbol{u}]\!]. \end{aligned} \tag{7}$$

A.2.1 IP force

The force corresponding to the first and the last terms remain the same, and the only difference comes from the terms in the middle. To do the integration, we

follow the same procedure as Eqs. (2) and (3). The integral form is

$$\begin{aligned}\int FXn^T : P & \\ &= \int vec(FXn^T)^T vec(P) \\ &= \int X^T (n \otimes F)^T vec(P)\end{aligned}$$

which reduces back to the form in Eq. (3).

Now we can write the second term inside the integral in Eq. (7) explicitly:

$$\begin{aligned}E_{IP} = \quad -\int [\![\boldsymbol{u}]\!] : \{P(\boldsymbol{u})\} = \quad & -\int \frac{1}{2}\left((F_- - F_+)X + (b_- - b_+)\right) n_-^T : (P_- + P_+) \\ = \qquad & -\frac{1}{2} X^T (n_- \otimes F_- - n_- \otimes F_+)^T vec\,(P_- + P_+) \\ & -\frac{1}{2}(b_- - b_+)^T (n_- \otimes I)^T vec\,(P_- + P_+)\end{aligned}$$

and we can proceed to take the derivatives for the corresponding force:

$$\begin{aligned}\frac{\partial}{\partial x_{-,i}^{(j)}}(E_{IP}) \quad = \quad & -\int \frac{1}{2} X^T (n_- \otimes \frac{\partial}{\partial x_{-,i}^{(j)}} F_-)^T vec\,(P_- + P_+) \\ & + \frac{1}{2} X^T (n_- \otimes F_- - n_- \otimes F_+)^T vec\left(\frac{\partial}{\partial x_{-,i}^{(j)}} P_-\right) \\ & + \frac{1}{2}(\frac{\partial}{\partial x_{-,i}^{(j)}} b_-)^T (n_- \otimes I)^T vec\,(P_- + P_+) \\ & + \frac{1}{2}(b_- - b_+)^T (n_- \otimes I)^T vec\left(\frac{\partial}{\partial x_{-,i}^{(j)}} P_-\right)\end{aligned}$$

$$\begin{aligned}\frac{\partial}{\partial x_{+,i}^{(j)}}(E_{IP}) &= -\int -\frac{1}{2}X^T(n_- \otimes \frac{\partial}{\partial x_{+,i}^{(j)}}F_+)^T vec\,(P_- + P_+) \\ &+\frac{1}{2}X^T(n_- \otimes F_- - n_- \otimes F_+)^T vec\left(\frac{\partial}{\partial x_{+,i}^{(j)}}P_+\right) \\ &-\frac{1}{2}(\frac{\partial}{\partial x_{+,i}^{(j)}}b_+)^T(n_- \otimes I)^T vec\,(P_- + P_+) \\ &+\frac{1}{2}(b_- - b_+)^T(n_- \otimes I)^T vec\left(\frac{\partial}{\partial x_{+,i}^{(j)}}P_+\right)\end{aligned}$$

and the stiffness matrix

$$\begin{aligned}\frac{\partial}{\partial x_{-,k}^{(l)}}\frac{\partial}{\partial x_{-,i}^{(j)}}(E_{IP}) &= -\int \frac{1}{2}X^T(n_- \otimes \frac{\partial}{\partial x_{-,i}^{(j)}}F_-)^T vec\left(\frac{\partial}{\partial x_{-,k}^{(l)}}P_-\right) \\ &+\frac{1}{2}X^T(n_- \otimes \frac{\partial}{\partial x_{-,k}^{(l)}}F_-)^T vec\left(\frac{\partial}{\partial x_{-,i}^{(j)}}P_-\right) \\ &+\frac{1}{2}(\frac{\partial}{\partial x_{-,i}^{(j)}}b_-)^T(n_- \otimes I)^T vec\left(\frac{\partial}{\partial x_{-,k}^{(l)}}P_-\right) \\ &+\frac{1}{2}(\frac{\partial}{\partial x_{-,k}^{(l)}}b_-)^T(n_- \otimes I)^T vec\left(\frac{\partial}{\partial x_{-,i}^{(j)}}P_-\right)\end{aligned}$$

$$\begin{aligned}\frac{\partial}{\partial x_{+,k}^{(l)}}\frac{\partial}{\partial x_{-,i}^{(j)}}(E_{IP}) &= -\int \frac{1}{2}X^T(n_- \otimes \frac{\partial}{\partial x_{-,i}^{(j)}}F_-)^T vec\left(\frac{\partial}{\partial x_{+,k}^{(l)}}P_+\right) \\ &-\frac{1}{2}X^T(n_- \otimes \frac{\partial}{\partial x_{+,k}^{(l)}}F_+)^T vec\left(\frac{\partial}{\partial x_{-,i}^{(j)}}P_-\right) \\ &+\frac{1}{2}(\frac{\partial}{\partial x_{-,i}^{(j)}}b_-)^T(n_- \otimes I)^T vec\left(\frac{\partial}{\partial x_{+,k}^{(l)}}P_+\right) \\ &-\frac{1}{2}(\frac{\partial}{\partial x_{+,k}^{(l)}}b_+)^T(n_- \otimes I)^T vec\left(\frac{\partial}{\partial x_{-,i}^{(j)}}P_-\right)\end{aligned}$$

$$
\begin{aligned}
\frac{\partial}{\partial x_{-,k}^{(l)}}\frac{\partial}{\partial x_{+,i}^{(j)}}(E_{IP}) &= -\int -\frac{1}{2}X^T(n_- \otimes \frac{\partial}{\partial x_{+,i}^{(j)}}F_+)^T vec\left(\frac{\partial}{\partial x_{-,k}^{(l)}}P_-\right) \\
&\quad +\frac{1}{2}X^T(n_- \otimes \frac{\partial}{\partial x_{-,k}^{(l)}}F_-)^T vec\left(\frac{\partial}{\partial x_{+,i}^{(j)}}P_+\right) \\
&\quad -\frac{1}{2}(\frac{\partial}{\partial x_{+,i}^{(j)}}b_+)^T(n_- \otimes I)^T vec\left(\frac{\partial}{\partial x_{-,k}^{(l)}}P_-\right) \\
&\quad +\frac{1}{2}(\frac{\partial}{\partial x_{-,k}^{(l)}}b_-)^T(n_- \otimes I)^T vec\left(\frac{\partial}{\partial x_{+,i}^{(j)}}P_+\right)
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial}{\partial x_{+,k}^{(l)}}\frac{\partial}{\partial x_{+,i}^{(j)}}(E_{IP}) &= -\int -\frac{1}{2}X^T(n_- \otimes \frac{\partial}{\partial x_{+,i}^{(j)}}F_+)^T vec\left(\frac{\partial}{\partial x_{+,k}^{(l)}}P_+\right) \\
&\quad -\frac{1}{2}X^T(n_- \otimes \frac{\partial}{\partial x_{+,k}^{(l)}}F_+)^T vec\left(\frac{\partial}{\partial x_{+,i}^{(j)}}P_+\right) \\
&\quad -\frac{1}{2}(\frac{\partial}{\partial x_{+,i}^{(j)}}b_+)^T(n_- \otimes I)^T vec\left(\frac{\partial}{\partial x_{+,k}^{(l)}}P_+\right) \\
&\quad -\frac{1}{2}(\frac{\partial}{\partial x_{+,k}^{(l)}}b_+)^T(n_- \otimes I)^T vec\left(\frac{\partial}{\partial x_{+,i}^{(j)}}P_+\right)
\end{aligned}
$$