Awesome work on Machine Learning and Finite Element Models

Anonymous Author(s)
Affiliation
Address
email

1 Finite Element Methods

1.1 Physics of Elastic Objects

The simulation of deformable elastic solids has become a rich field of research due to its usefulness in computer animation and graphics. The Finite Element Method (FEM) is the most popular tool to address this problem due to its versatility in representing elastic bodies with intricate geometric features and diverse material properties.

To understand FEM, the first step is understanding what it means for an object to be deformed due to some external forces and to its elastic properties. A thorough analysis of the physics laws of deformable objects is out of the scope of this work, the interested reader can refer to [REF] for more details. From a very high level perspective, each object occupies a precise volume Ω in a specific coordinate system. Due to external forces, the object can be subject to deformation, i.e. each material point $\overrightarrow{X} \in \Omega$ will be displaced to a *new* deformed location $\overrightarrow{x}$. Typically, this deformation is defined by a *deformation map* $\phi(\overrightarrow{X}) = \overrightarrow{x}$ (see Figure 1.1).

One of the consequences of elastic deformation is the accumulation of potential energy in the deformed body. This energy is often defined as $E[\phi]$, to suggest that it is fully determined by the deformation map of a given object. More precisely, different parts of a deforming object undergo shape changes in of different severity. As a consequence, the relation between deformation and energy is better defined on a local scale. We introduce the energy density function $\Psi[\phi; \overrightarrow{X}]$ which measures the energy per unit of deformed volume on an infinitesimal volume dV around the material point $\overrightarrow{X}$. We can define the total energy of the deforming body, by integrating over the entire object volume Ω:

$$E[\phi] = \int_{\Omega} \Psi[\phi; \overrightarrow{X}] d\overrightarrow{X} \tag{1}$$

Without going into the full details, it is possible to write $\Psi[\phi; \overrightarrow{X}] = \Psi(\mathbf{F}(\overrightarrow{X}))$, where $\mathbf{F}$ is the Jacobian of the deformation map, referred to as the deformation gradient, calculated at a specific material point $\overrightarrow{X}$. This definition of the energy density function is necessary to introduce properties related to the material models of a specific object, i.e., the precise mathematical expression of $\Psi(\mathbf{F})$ will be the defining property of the material model[1].

[1] We will not provide here concrete examples of material models and their corresponding energy definition. The reader can refer to [REF] for more details.

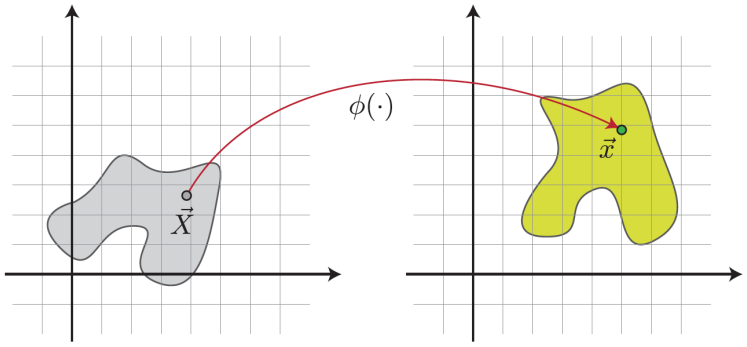

Figure 1: Example of the deformation map

1.2 Discretization

In order to enable numerical simulation, the physical laws introduced in the section above need to be discretized. Elastic energy and elastic forces (which we will describe below) need to be reformulated as functions of discrete state variables.

1.3 Elastic Forces

When modeling a deformable body on a computer, we only store the values of the deformation map $\phi(\vec{X})$ on a finite number of points $\vec{X}_1, \vec{X}_2, \vec{X}_3, \ldots, \vec{X}_N$ corresponding to the vertices of a discretization mesh. The respective deformed vertex locations $\vec{x}_i = \phi(\vec{X}_i),\ i = 1, 2, \ldots, N$ are defined as the degrees of freedom of the problem. The idea is then to specify a method to reconstruct a continuous deformation map $\hat{\phi}$, from the discrete samples $\vec{x}_i = \phi(\vec{X}_i)$, i.e., by defining a specific interpolation scheme. We denote the interpolated deformation map as $\hat{\phi}[\vec{X}; \mathbf{x}]$, which emphasizes the dependency on the discrete state $\mathbf{x} = (\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_N)$.

We can define the discrete Energy, as a function of the degrees of freedom $\mathbf{x}$, by simply plugging the interpolated deformation $\hat{\phi}$ into the definition of the standard energy:

$$E(\mathbf{x}) := E\left[\hat{\phi}(\vec{X}; \mathbf{x})\right] = \int_\Omega \Psi\left(\hat{\mathbf{F}}(\vec{X}; \mathbf{x})\right) \tag{2}$$

where $\hat{\mathbf{F}}(\vec{X}; \mathbf{x}) = \partial\hat{\phi}(\vec{X}; \mathbf{x})/\partial\vec{X}$ is the deformation gradient computed from the interpolated map $\hat{\phi}$.

Having defined the discrete energy $E(\mathbf{x})$ we can now compute the elastic forces associated with individual mesh nodes, by taking the negative gradient of the elastic energy with respect to the corresponding degree of freedom:

$$\vec{f}_i(\mathbf{x}) = \frac{\partial E(\mathbf{x})}{\partial \vec{x}_i} \quad \text{or, collectively} \quad \mathbf{f} := (\vec{f}_1, \vec{f}_2, \ldots, \vec{f}_N) = -\frac{\partial E(\mathbf{x})}{\partial \mathbf{x}}$$

In practice, before computing each force, the energy integral in Equation 2 is separated into the contributions of the individual elements Ω_e as follows:

$$E(\mathbf{x}) = \sum_e E^e(\mathbf{x}) = \sum_e \int_{\Omega_e} \Psi(\hat{\mathbf{F}}(\vec{X}; \mathbf{x}))d\vec{X} \tag{3}$$

In our work, without losing generality, we will focus on linear tetrahedra, which are among the most popular discrete volumetric representations and which offer on of the most straightforward

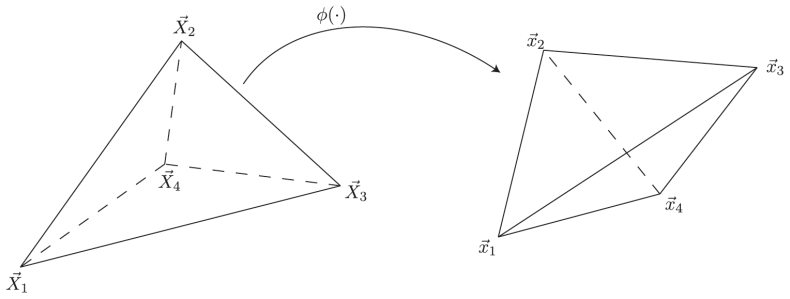

Figure 2: A tetrahedron before and after a deformation

option for constructing a discretization of the elasticity equations, i.e. the reconstructed deformation map $\hat{\phi}$ can be defined as a *piecewise linear function* over each tetrahedron. Specifically, for each tetrahedron $\mathcal{T}_i$ we have:

$$\hat{\phi}(\overrightarrow{X}) = \mathbf{A}_i \overrightarrow{X} + \overrightarrow{b}_i \quad \text{for all} \quad \overrightarrow{X} \in \mathcal{T}_i \tag{4}$$

Figure 1.3 shows an example of a tetrahedron. Differentiating Equation 4 with respect to $\overrightarrow{X}$ reveals that the deformation gradient $\mathbf{F} = \partial\hat{\phi}/\partial\overrightarrow{X} = \mathbf{A}_i$ is a constant on each element. Thus, for simplicity we can write $\hat{\phi}(\overrightarrow{X}) = \mathbf{F}\overrightarrow{X} + \overrightarrow{b}_i$. Then, since $\mathbf{F}$ is constant, the energy of a single tetrahedron can be rewritten (after some manipulation) as

$$E_i = \int_{\mathcal{T}_i} \Psi(\mathbf{F}) d\overrightarrow{X} = \Psi(\mathbf{F}_i) \int_{\mathcal{T}_i} d\overrightarrow{X} = W \cdot \Psi(\mathbf{F}_i) \quad \text{or} \quad E(\mathbf{x}) = W \cdot \Psi(\mathbf{F}(\mathbf{x})) \tag{5}$$

We can then use Equation 5 to derive the contribution of element $\mathcal{T}_i$ to the elastic forces on its four vertices as $f_k^i = -\partial E_i(\mathbf{x})/\partial\overrightarrow{x}_k$.

1.4 Forces Differentials

The most common integration scheme used to simulate elastic objects is the Backward Euler method. At any moment in time, this method simulates the action of a force over a vertex of an element of a deformable object. To do so, in addition to discrete nodal forces it also requires a process to compute *force differentials*, i.e., linearized nodal force increments around a configuration $\mathbf{x}_*$, relative to a small nodal force displacement $\delta\mathbf{x}$. We denote this by:

$$\delta\mathbf{f} = \delta\mathbf{f}(\mathbf{x}_*; \delta\mathbf{x}) := \left.\frac{\partial\mathbf{f}}{\partial\mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_*} \cdot \delta\mathbf{x} \tag{6}$$

The expression in Equation 6 makes use of the stiffness matrix $\partial\mathbf{f}/\partial\mathbf{x}$, which is typically hard to calculate. Instead, the aim is to compute the force differential vector $\delta\mathbf{f}$ on an element-by-element basis, accumulating the contribution of each element to the aggregate value of each of its nodes. This is done using only the information in the current state $\mathbf{x}_*$, the displacement $\delta\mathbf{x}$ and a small amount of additional meta-data (see [REF] for the specific derivation).

2 The Backward Euler Method

Given the background described in the previous section, we can now describe a complete implicit-time integration scheme for non-linear elastic bodies. In order to define the backward Euler integrations scheme (BE), we need to maintain both the position ($\mathbf{x}^n$) and the velocity ($\mathbf{v}^n$) of the deforming body at time t^n. The BE scheme calculate the position $\mathbf{x}^{n+1}$ and velocity $\mathbf{v}^{n+1}$ at time $t^{n+1}(:= t^{n+1} + \Delta t)$ as the solution of the following non-linear system of equations:

$$\mathbf{x}^{n+1} = \mathbf{x}^n + \Delta t \mathbf{v}^{n+1} \tag{7}$$

$$\begin{aligned}\mathbf{v}^{n+1} &= \mathbf{v}^n + \Delta t \mathbf{M}^{-1} \mathbf{f}(\mathbf{x}^{n+1}, \mathbf{v}^{n+1}) \\ &= \mathbf{v}^n + \Delta t \mathbf{M}^{-1} \left(\mathbf{f}_e(\mathbf{x}^{n+1}) + \mathbf{f}_d(\mathbf{x}^{n+1}, \mathbf{v}^{n+1}) \right)\end{aligned} \tag{8}$$

where we introduce the following notation:

- $\mathbf{f}_e(\mathbf{x}_*)$: Elastic forces at configuration $\mathbf{x}_*$, as defined in Section 1.3.
- $\mathbf{f}_d(\mathbf{x}^*, \mathbf{v}^*) = -\gamma \mathbf{K}(\mathbf{x}^*)\mathbf{v}^*$: damping forces at position $\mathbf{x}^*$ and velocity $\mathbf{v}^*$ according to the Rayleigh damping model. γ does not have a predefined range.
- $\mathbf{K}(\mathbf{x}^*) = \frac{\partial \mathbf{f}_e}{\partial \mathbf{x}}\big|_{\mathbf{x}^*}$ is the elasticity stiffness matrix evaluated at $\mathbf{x}^*$. This matrix is typically constructed using the force differential defined in Section 1.4.
- $\mathbf{f}(\mathbf{x}^*, \mathbf{v}^*) = \mathbf{f}_e(\mathbf{x}_*) + \mathbf{f}_d(\mathbf{x}^*, \mathbf{v}^*)$: aggregate forces including elastic and damping constraints.
- $\mathbf{M}$: the mass matrix.

Now, Equation 8 makes the Backward Euler system non-linear. To address this issue, an iterative process is defined to compute the unknowns $\mathbf{x}^{n+1}$ and $\mathbf{v}^{n+1}$, such that the approximations converge to the real value as the number of iterations goes to infinity: $\mathbf{x}^{n+1}_{(k)} \rightarrow_{k\rightarrow\infty} \mathbf{x}^{n+1}$ and $\mathbf{v}^{n+1}_{(k)} \rightarrow_{k\rightarrow\infty} \mathbf{v}^{n+1}$ (the initial guesses will be the position and velocity of the previous step). We define the position and velocity correction variables as:

$$\Delta \mathbf{x}_{(k)} := \mathbf{x}^{n+1}_{(k+1)} - \mathbf{x}^{n+1}_{(k)} \quad \text{and} \quad \Delta \mathbf{x}_{(k)} := \mathbf{v}^{n+1}_{(k+1)} - \mathbf{v}^{n+1}_{(k)}$$

The key idea of the correction scheme is the following: at each step of the iterative scheme, Equations 7 and 8 are linearized around the current iterate $(\mathbf{x}^{n+1}_{(k)}, \mathbf{v}^{n+1}_{(k)})$. The solutions of such linearization are then used to define the next iterate $(\mathbf{x}^{n+1}_{(k+1)}, \mathbf{v}^{n+1}_{(k+1)})$. Referring to [REF], for further details, using some mathematical manipulation, we linearize Equation 8 as:

$$\begin{aligned}&\mathbf{v}^{n+1}_{(k)} + \Delta \mathbf{v} = \\ &\qquad \mathbf{v}^n + \Delta t \mathbf{M}^{-1} \left(\mathbf{f}_e(\mathbf{x}^{n+1}_{(k)}) + \frac{\partial \mathbf{f}_e}{\mathbf{x}}\Big|_{\mathbf{x}^{n+1}_{(k)}} \cdot \Delta \mathbf{x} - \gamma \mathbf{K}(\mathbf{x}^{n+1}_{(k)})(\mathbf{v}^{n+1}_{(k)} + \Delta \mathbf{v}) \right)\end{aligned} \tag{9}$$

We obtain a new equation which can be solved using well known solvers such as the Conjugate Gradients method or the Jacobi method.

2.1 Research Challenges

Our initial idea is to augment the Backward-Euler method using data. We want to maintain the scheme defined in Equations 7, 9 and use data— in the forms of mappings $\langle (\mathbf{x}^n, \mathbf{v}^n); (\mathbf{x}^{n+1}, \mathbf{v}^{n+1}) \rangle$ generated by a base-model— to improve its accuracy and its speed.

In more, detail, solving Equations 7, 9 involve several design choices which are going to affect the performance of the algorithm. Adjusting these choices is often tedious, and engineers tend to use standard instantiations, without exploring systematically the space of all different alternatives to understand the best ones. In the context of physical simulation of elastic bodies, we could imagine introducing the following improvements:

- Force model optimization: given an elastic force model $\mathbf{f}_e$, we can learn its parameters using data, i.e., given a number of constants (e.g., λ in the St. Venant-Kirchoff model) which are used to instantiate a model, what are the best values to guarantee the highest animation quality?
- Model selection: learn which model to use based on data. We could imagine defining an ensemble, i.e., a weighted combination of all the existing energy models (e.g., Corotational Linear, Isotropic, St. Venant-Kirchoff and Neo-Hookean) and learn the best configuration of the weights to maximize accuracy.

- Solver selection: learn which algorithm (Jacobi Iteration, Conjugate Gradient, etc. etc) is faster and more accurate to calculate both the elastic forces and the force differentials. One key aspect here, is that since the implicit Backward-Euler scheme is iterative, a sequence of different approaches might be better that using the same approach at each step.
- Solver optimization: given a solver, learn its parameters using data, for example the learning rate of any gradient-based algorithm.

The motivation behind the idea of tuning the Backward-Euler method using data is that we maintain its properties (i.e., good performance), and build over them. Supposedly, this should improve the performance of what is already considered state-of-the-art, compared to a completely new method, which might not work as well.