1D Strand Simulation with Cauchy-Green Invariants based ARAP Energy

Xijia Tao
Department of Computer Science, HKU

October 3, 2022

1 Introduction

The report is a summary of what I have done during my research *internship*[1] at HKU in the summer of 2022. Note that the correctness of some of my statements about the concerned paper is left unverified. Hence, my understanding and implementation of the corresponding parts should NOT be considered a reliable source of reference.

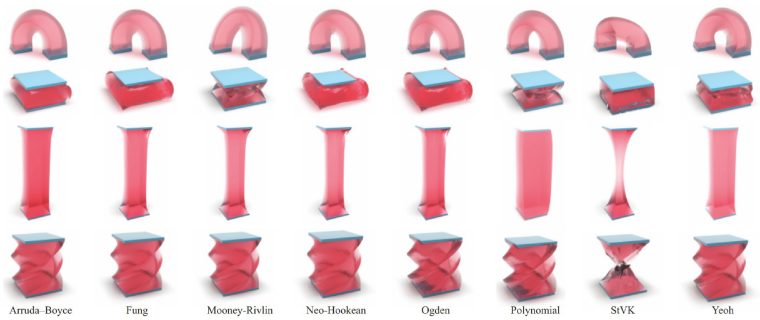
My research project targets the lack of a 1-dimensional formulation of a material model, namely ARAP energy, using Cauchy-Green (CG) invariants. In this report, I will first outline the background of material models, then give a description of the relevant paper and finally propose the 1D formulation.

2 Preliminaries

2.1 Background

In computer graphics, simulating the deformation of materials under stress has been a popular research topic.

Material models were introduced to best describe certain materials.



[1]Many thanks to Dr. Chitalu and my supervisor Prof Komura for their help.

2.2 Definition

To understand the research problem, it is necessary to rigorously formulate the dependencies between physical quantities.

In a n (usually 3) dimensional space, when a force is subjected to an object, very likely deformation occurs. We denote such a mapping by a deformation function

$$\vec{\phi} : \mathbb{R}^n \to \mathbb{R}^n, \quad \vec{x} = \vec{\phi}(\vec{X}),$$

where $\vec{X}$ is the rest position of a point on the object and $\vec{x}$ is the position of that point after deformation.

With $\vec{\phi}$, we can obtain its Jacobian matrix $\boldsymbol{F}$, known as the deformation gradient,

$$\boldsymbol{F} = \frac{\delta \vec{\phi}(\vec{X})}{\delta \vec{X}} \in \mathbb{R}^{n \times n}.$$

Some simple examples are

- translation: $\vec{\phi}(\vec{X}) = \vec{X} + \vec{t} \implies \boldsymbol{F} = \boldsymbol{I}$
- scaling: $\vec{\phi}(\vec{X}) = \gamma \vec{X} \implies \boldsymbol{F} = \gamma \boldsymbol{I}$
- rotation: $\vec{\phi}(\vec{X}) = \boldsymbol{R}_\theta \vec{X} \implies \boldsymbol{F} = \boldsymbol{R}_\theta$

Deformation increases the potential energy of an object, i.e.

$$\text{strain energy} := E(\phi).$$

Then it's natural to define the strain energy density function $\Psi[\phi; \vec{X}]$, which measures the energy intensity around a point on the object

$$E[\phi] = \int \Psi[\phi; \vec{X}] d\vec{X}.$$

From mathematical derivation, it is shown that $\Psi[\phi; \vec{X}]$ can be written purely in terms of the deformation gradient, i.e.

$$\Psi[\phi; \vec{X}] = \Psi(\boldsymbol{F}(\vec{X})).$$

Ψ can take on many forms, mostly depending on the material which it is describing. Some simple examples are

- $\Psi(F) = \frac{k}{2} \|F\|^2$ with $k > 0 \implies$ tendency to collapse to a single point;
- $\Psi(F) = \frac{k}{2} \|F - I\|^2 \implies$ does not treat a rotation as a rest configuration.

3 Paper Overview

The research is based on the preprint paper *A Cauchy-Green Invariant based Isotropic ARAP energy*[2].

In this section, I will describe the contributions of the paper and also its limitation.

[2]This is ongoing research at HKU. Reference is to be added once the paper is officially released.

3.1 Rationale

Isotropic As-Rigid-As-Possible (ARAP) energy has been a popular material model for almost two decades. However, a formulation using Cauchy-Green (CG) invariants has always been unclear, due to a rotation-polluted trace term that cannot be directly expressed using these invariants.

The most common isotropic hyperelastic energies for solid mechanics simulation in computer graphics can be expressed with CG invariants.

Unlike the Dirichlet, Neo-Hookean and St. Venant-Kirchhoff energies, ARAP remains the only model within this group whose formulation has thus far seemed inexpressible in terms of CG-invariants, which prohibits a generic (i.e. complete rogues' gallery) formulation of gradients and Hessians that is shared by these energies for Newton-type implicit integration.

3.2 Mathematics

The three CG invariants are defined to be

$$I_C = \|\boldsymbol{F}\|^2 \quad II_C = \|\boldsymbol{C}\|^2 \quad III_C = \det(\boldsymbol{C}),$$

where $\boldsymbol{C} = \boldsymbol{F}^T\boldsymbol{F}$ can be computed inexpensively.

The ARAP energy is originally formulated with

$$\Psi = \frac{\mu}{2}\|\boldsymbol{F} - \boldsymbol{R}\|^2 = \frac{\mu}{2}(\|\boldsymbol{S}\|^2 - 2\,\mathrm{tr}(\boldsymbol{S}) + n),$$

where the rotation matrix $\boldsymbol{R}$ and the scale matrix $\boldsymbol{S}$ can be obtained from $\boldsymbol{F}$ by polar decomposition $\boldsymbol{F} = \boldsymbol{R}\boldsymbol{S}$.

Since the Frobenius norm of $\boldsymbol{R}$ is simply 1, then we get $\|\boldsymbol{S}\|^2 = \|\boldsymbol{F}\|^2 = I_C$. By substituting this to the equation above, we have $\Psi = I_C - 2\,\mathrm{tr}(\boldsymbol{S}) + n$ (ignoring the constant coefficient for simplicity).

Now let's take a closer look at the potential relationship between CG invariants and $\mathrm{tr}(\boldsymbol{S})$. With singular value decomposition of $\boldsymbol{F} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}$, we obtain

$$I_C = \mathrm{tr}(\boldsymbol{\Sigma}^2) = \sum_{i=1}^{3}\sigma_i^2 \quad II_C = \mathrm{tr}(\boldsymbol{\Sigma}^4) = \sum_{i=1}^{3}\sigma_i^4 \quad III_C = \det(\boldsymbol{\Sigma}^2) = \prod_{i=1}^{3}\sigma_i^2$$

$$\mathrm{tr}(\boldsymbol{S}) = \mathrm{tr}(\boldsymbol{\Sigma}) = \sum_{i=1}^{3}\sigma_i,$$

where σ_i is a singular value of $\boldsymbol{F}$ lying on the i-th diagonal entry of $\boldsymbol{\Sigma}$. Note that the invariants and the trace can both be written in terms of the singular values.

3.3 Problem Solved

The problem of writing the ARAP energy in terms of CG invariants lies in representing the trace term. Although due to the non-linearity of CG invariants, $\mathrm{tr}(\boldsymbol{S})$ cannot be written directly in them. We could implicitly form a polynomial with CG invariants as the coefficients, such that its root equals to the trace term. Specifically, the polynomial is unique and is defined as follows

$$\mathcal{P}(t) = t^4 - 2I_C t^2 - 8\sqrt{III_C}t + I_C^2 - 4II_C^*,$$

where $II_C^* = \frac{1}{2}(I_C^2 - II_C)$. There is an analytical solution for the root that corresponds to the trace term.

3.4 Model Formulation

We arrive at our implicit ARAP energy

$$\Psi_{iARAP} = I_C - 2(\mathcal{P}(t) = 0) + n.$$

From the formulation, we can then calculate the gradient and Hessian of the energy w.r.t the node positions. The paper proposed an innovative way of ensuring the semi-positive definiteness of the Hessian so that a solution is guaranteed for implicit time-stepping.

3.5 Algorithm

The algorithm for updating the positions of vertices is shown in the figure below.

Algorithm 1 Projected Newton Solver

procedure MINIMISEINCREMENTALPOTENTIAL($\mathbf{x}^t$)
  $\mathbf{x} \leftarrow \mathbf{x}^t$
  $\mathbf{x}_{\text{prev}} \leftarrow \mathbf{x}$
  $E_{\text{prev}} \leftarrow E(\mathbf{x})$
  repeat
    $\mathbf{g} \leftarrow \frac{\partial E(\mathbf{x})}{\partial \mathbf{x}}$
    $\mathbf{H} \leftarrow \text{projectedHessian}\left(\frac{\partial^2 E(\mathbf{x})}{\partial \mathbf{x}^2}\right)$ ▷ Section § 5
    $\mathbf{p} \leftarrow -\mathbf{H}^{-1}\mathbf{g}$ ▷ Solve Eq. (40)
    $\alpha \leftarrow 1$
    do ▷ Line search
      $\mathbf{x} \leftarrow \mathbf{x}_{\text{prev}} + \alpha\mathbf{p}$
      $\alpha \leftarrow \frac{\alpha}{2}$
    while $E(\mathbf{x}) > E_{\text{prev}}$
    $\mathbf{x}_{\text{prev}} \leftarrow \mathbf{x}$
    $E_{\text{prev}} \leftarrow E(\mathbf{x})$
  until $\|\mathbf{g}\|_\infty < \epsilon$
  return $\mathbf{x}$ ▷ New positions $\mathbf{x}^{t+1}$
end procedure

3.6 Contribution & Limitation

A summary of the paper's contributions is as follows:

- An isotropic ARAP energy using Cauchy-Green invariants.
- A complete eigenanalysis of the energy.
- An analysis showing that the energy is equivalent to existing formulations of isotropic ARAP energy.
- A fast, analytic semi-positive definiteness projection method for the Hessian.

However, the authors did not consider the potential application of their method to the simulation of 1-dimensional strands. Hence, I base my research on this direction.

4 Moving to 1 Dimension

4.1 1D Energy

On the one hand, based on the formulation of CG invariants using the singular values, we get the following equalities applicable in 1 dimension.

$$I_C = \sigma_1^2 \quad II_C = \sigma_1^4 = I_C^2 \quad III_C = \sigma_1^2 = I_C$$

On the other hand, the trace term becomes

$$\mathrm{tr}(\boldsymbol{S}) = \mathrm{tr}(\boldsymbol{\Sigma}) = \sigma_1$$

Hence, we can take $tr(\boldsymbol{S}) = \sqrt{I_C}$ and the rewritten 1D energy becomes[3]

$$\Psi_{1D} = I_C - 2\sqrt{I_C} + 1.$$

4.2 Deformation Gradient

Now let's take a step back and (re-)define the deformation gradient $\boldsymbol{F}$. In 3D, $\boldsymbol{F}$ is 3×3 matrix encoding the deformation within a tetrahedral element. In 1D, a natural analogy is to define $\boldsymbol{F}$ using the relative displacement between two vertices of the same edge. Specifically, for vertex positions $\boldsymbol{x}_1, \boldsymbol{x}_2$ and rest length of their edge l, define

$$\boldsymbol{F} := (\boldsymbol{x}_1 - \boldsymbol{x}_2)/l.$$

With $\boldsymbol{F}$, we can then calculate I_C by definition and hence the ARAP energy.

4.3 PK1 and Hessian

The first Piola–Kirchhoff stress tensor (PK1) can be obtained by chain rule

$$\begin{aligned} \boldsymbol{P}(\boldsymbol{F}) = \frac{\delta \Psi_{1D}}{\delta \boldsymbol{F}} &= \frac{\delta \Psi_{1D}}{\delta \boldsymbol{I_C}} \frac{\delta I_C}{\delta \boldsymbol{F}} && (1) \\ &= (1 - 1/\sqrt{I_C}) \times 2\boldsymbol{F} && (2) \\ &= 2(1 - 1/\sqrt{I_C})\boldsymbol{F}. && (3) \end{aligned}$$

The Hessian is given by

$$\frac{\delta \boldsymbol{P}(\boldsymbol{F})}{\delta \boldsymbol{F}} = \frac{\delta^2 \Psi}{\delta I_C^2} \boldsymbol{g}_{I_C} \boldsymbol{g}_{I_C}^T + \frac{\delta \Psi}{\delta I_C} \boldsymbol{H}_{I_C}, \tag{4}$$

where

$$\begin{aligned} \boldsymbol{g}_{I_C} &= 2\boldsymbol{F} && (5) \\ \boldsymbol{H}_{I_C} &= 2\boldsymbol{I}_{3\times 3}. && (6) \end{aligned}$$

[3] In fact, this step might not be legitimate because $tr(\boldsymbol{S})^2 = I_C$ only implies $tr(\boldsymbol{S}) = \pm\sqrt{I_C}$. But let's assume there's some magic constraint which guarantees $tr(\boldsymbol{S})$ to be positive.

However, the Hessian obtained this way is not necessarily semi-positive definite, which is a desirable property under many scenarios. [TODO]

To ensure this property, we conduct eigenanalysis on the Hessian to obtain its eigenvalues λ_i and corresponding eigenvectors $\boldsymbol{q}_i$. Then we can set the new (and semi-positive definite) Hessian to be

$$\boldsymbol{H} = \sum_{i=1}^{N} \langle \lambda_i \rangle \boldsymbol{q}_i \boldsymbol{q}_i^T,$$

where $\langle x \rangle = \max\{x, 0\}$.

The eigenvalues and their corresponding eigenvectors are provided as follows

$$\lambda_0 = 1, \qquad Q_0 = \boldsymbol{U}\boldsymbol{T}_x\boldsymbol{T}^T, \tag{7}$$

$$\lambda_1 = 1 - \frac{1}{\sigma_1}, \qquad Q_1 = \boldsymbol{U}\boldsymbol{T}_y\boldsymbol{V}^T, \tag{8}$$

$$\lambda_2 = 1 - \frac{1}{\sigma_1}, \qquad Q_2 = \boldsymbol{U}\boldsymbol{T}_z\boldsymbol{V}^T, \tag{9}$$

with

$$\boldsymbol{T}_x = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \boldsymbol{T}_y = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad \boldsymbol{T}_z = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{10}$$

Rotation factors $\boldsymbol{U}$ and $\boldsymbol{V}$ of the non-square deformation gradient $\boldsymbol{F} \in \mathbb{R}^{3\times 1}$ are determined by first observing that $\boldsymbol{V}$ is an orthonormal order-zero tensor (i.e. scalar). That is $\boldsymbol{V} = 1$. Following the conclusion in the reference paper, the rotation part of $\boldsymbol{F}$ can be constructed as

$$\boldsymbol{R} = \boldsymbol{U} \begin{bmatrix} V^T \\ 0 \\ 0 \end{bmatrix} = \boldsymbol{U} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix},$$

which implies that the first column of $\boldsymbol{U} \in \mathbb{R}^{3\times 3}$ is $\boldsymbol{R}^{3\times 1}$. Now, we apply Eq. (35) in the original paper to compute $\boldsymbol{R}$.

$$\frac{\delta t}{\delta \boldsymbol{F}} = \frac{\delta \operatorname{tr}(\boldsymbol{S})}{\delta \boldsymbol{F}} = \frac{\delta \operatorname{tr}(\boldsymbol{F}^T\boldsymbol{R})}{\delta \boldsymbol{F}} \triangleq \boldsymbol{R} \tag{11}$$

$$= \frac{\delta t}{\delta I_C}\frac{\delta I_C}{\delta \boldsymbol{F}} + \frac{\delta t}{\delta II_C}\frac{\delta II_C}{\delta \boldsymbol{F}} + \frac{\delta t}{\delta J}\frac{\delta J}{\delta \boldsymbol{F}} \tag{12}$$

$$= \frac{\delta t}{\delta I_C}\frac{\delta I_C}{\delta \boldsymbol{F}} = \frac{1}{2} I_C^{-\frac{1}{2}} \times 2\boldsymbol{F} = \boldsymbol{F}/t \tag{13}$$

4.4 Completing the Algorithm

By comparing with the algorithm outlined in section 3.5, we can observe some differences in notation between it and the 1D formulation above. Now, I will attempt to draw some connections based on my own understanding, which is not guaranteed to be correct.

- $\boldsymbol{x}$ can either be a vector in $\mathbb{R}^3$ or $\mathbb{R}^{3n}$, where n is the number of vertices involved. I have implemented a local version using vectors in the former Euclidean space, for simplicity. The following discussion would also be based on these vectors.

- $E(\boldsymbol{x})$ calculates the energy accumulating at a given vertex. However, I am not sure what this implies. It seems that the energy of a system makes more sense than the energy of a single vertex. But my supervisor suggested that it is fine. So I did not dive deep into this doubt of mine.

- Note that there are two different *gradients* and *Hessians* involved in our discussion. In the algorithm, $\boldsymbol{g}_1 = \frac{\delta E(\boldsymbol{x})}{\delta \boldsymbol{x}}$ and $\boldsymbol{H}_1 = \frac{\delta^2 E(\boldsymbol{x})}{\delta \boldsymbol{x}^2}$ are different from $\boldsymbol{g}_2 = \boldsymbol{P}(\boldsymbol{F}) = \frac{\delta \Psi}{\delta \boldsymbol{F}}$ (the PK1 tensor, which can also be viewed as a gradient) and $\boldsymbol{H}_2 = \frac{\delta \boldsymbol{P}(\boldsymbol{F})}{\delta \boldsymbol{F}} = \frac{\delta^2 \Psi}{\delta \boldsymbol{F}^2}$. In fact, they are connected by the following equations.[4]

  - $\boldsymbol{g}_1 = \frac{\delta E(\boldsymbol{x})}{\delta \boldsymbol{x}} = \frac{\delta \Psi}{\delta \boldsymbol{x}} = \frac{\delta \Psi}{\delta \boldsymbol{F}} \frac{\delta \boldsymbol{F}}{\delta \boldsymbol{x}} = \boldsymbol{g}_2 \frac{\delta \boldsymbol{F}}{\delta \boldsymbol{x}}$
  - $\boldsymbol{H}_1 = \frac{\delta^2 E(\boldsymbol{x})}{\delta \boldsymbol{x}^2} = \frac{\delta^2 \Psi)}{\delta \boldsymbol{x}^2} = (\frac{\delta \boldsymbol{F}}{\delta \boldsymbol{x}})^T \frac{\delta^2 \Psi}{\delta \boldsymbol{F}^2} \frac{\delta \boldsymbol{F}}{\delta \boldsymbol{x}} = (\frac{\delta \boldsymbol{F}}{\delta \boldsymbol{x}})^T \boldsymbol{H}_2 \frac{\delta \boldsymbol{F}}{\delta \boldsymbol{x}}$

  Here, we ignore any external forces (e.g. gravity) to reach the equality $\Psi = E(\boldsymbol{x})$ and make our life easier.

By now, we have prepared all ingredients to implement the algorithm. In the next section, I will give a simple demonstration of the 1D ARAP model in practice.

5 Demonstration

Please view my code and results in `.gif` format *here*.

Remarks From the animations, you can see that the model shows some desirable properties such as a tendency to return to an equilibrium state (if there's any, i.e. where all the edges are in their rest lengths). However, you can also observe that under some settings, the displacement vector $\boldsymbol{p}$ as calculated in the algorithm becomes very unstable after a few seemingly reasonable updates. That is to say, $\boldsymbol{p}$ either explodes or vanishes.

As far as I'm concerned, this can be related to how I compute $\boldsymbol{p}$. As per the instruction given in the original paper, I apply the preconditioned conjugate gradient method to find $\boldsymbol{p}$ by solving the matrix equation $\boldsymbol{H}\boldsymbol{p} = \boldsymbol{g}$. However, the result can depend on the precondition chosen and might be subject to dramatic change when $\boldsymbol{H}$ or $\boldsymbol{g}$ changes a little.

Another possible reason can be volume preservation. In the original 3D implementation, volume preservation is directly accounted for by an additional term in the final energy formula. Without volume preservation, the change in lengths can be easier, which is not to be expected.

[4] This equation may look plausible at first sight. However, after noticing that $\frac{\delta \boldsymbol{F}}{\delta \boldsymbol{x}}$ is a length-3 vector, it's clear that the RHS expression of the second equation will be evaluated to a scalar. But we are expecting a 3×3 matrix for the Hessian. Hence, I just took $\boldsymbol{g}_2$ as $\boldsymbol{g}_1$ and $\boldsymbol{H}_2$ as $\boldsymbol{H}_1$ in my implementation.