Lyra: A Thin Shell Simulator with FEM and FDM

XingHao Wang[*]
ShanghaiTech University
wangxh3@shanghaitech.edu.cn

HongBo Chen[†]
ShanghaiTech University
chenhb@shanghaitech.edu.cn

WenJi Liu[‡]
ShanghaiTech University
liuwj@shanghaitech.edu.cn

1. Introduction

Physically-based animation has been a problem of interest to the graphics community. Thin shells, a kind of thin flexible structures with a high ratio of width to thickness. Specially, cloth, is greatly developed to a problem of interest to graphic and textile community in recent years. Thin shell simulation is all about calculating the effect of forces on one/several pieces of patches. Early work by Terzopoulos et al. [9] and Terzopoulos and Fleischer [8] on deformable models correctly characterized thin shell simulation as a problem in deformable surfaces, and applied techniques from the mechanical engineering and finite element communities to the problem. Since then, other research groups (notably Carignan et al. [4]and Volino et al. [10]; Breen et al. [3]; and Eberhardt et al. [5]) have taken up the challenge of thin shell.

Among these approaches, there is a deep commonality: physically-based thin shell simulation is formulated as a time-varying partial differential equation:

$$\frac{\partial}{\partial t}\left(\mu\frac{\partial \mathbf{x}}{\partial t}\right) + \gamma\frac{\partial \mathbf{x}}{\partial t} + \frac{\delta \mathcal{E}(\mathbf{x})}{\delta \mathbf{x}} = \mathbf{f}(\mathbf{x}, t) \tag{1}$$

Where $\mathbf{x}$ is a $3n$ stacked position vector of mesh vertices, n is the number of vertices in mesh, t is time. μ and γ represent the mass density and damping density of body at position $\mathbf{x}$, respectively. δ denotes variational derivative. $\mathcal{E}$ describes internal energy, such as stretch, shear and bending energy, etc. $\mathbf{f}$ describes external energy, such as gravity, wind force and constrained force etc.

Unfortunately, there is no analytical solution to this equation, and it is difficult to calculate its numerical solution. So, we combined the Finite Element Method (FEM) and Finite Difference Method (FDM) to solve this equation.

Lyra, a constellation name, we give our program which solve Partial Differential Equation (PDE) of thin shell object by Finite Element Method (FEM) in space and Implicit Finite Difference Method (FDM) in time. Fundamentally, lyra provide large time step thin shell simulation by backward Euler. For evaluating internal force, specially, stretch force and shear force in plane and bending force out of plane, FEM is necessary numerical tool. Moreover, lyra provide simple external force implementation such as gravity, wind and point group constraint for generating interesting results. Unfortunately, robust thin shell simulator with collision detection and response is a hard and burdensome task, we don't implement those important and impressive traits in the simulator.

[*]Graduate Student ID: 2018233111
[†]Graduate Student ID: 2018231063
[‡]Graduate Student ID: 2019233165

2. Solution

Taking notice of equation of thin shell motion is time development in 3D space. To solving the problem, it is necessary to discretize continuous thin shell material into small finite elements for mechanical analysis. Moreover, lyra uses implicit Euler method to discretize time for robust simulation. The two methods are described in detail below.

2.1. Finite Element Method (FEM)

Compared with traditional community which regards thin shell materials as plate with small thickness and discretize it into tetrahedrons, in graphic community, we take thin shell materials as 2D manifold embedding into 3D space. Therefore, we only discretize thin shell materials into 2D unstructed triangle mesh and tackle bending force separately.

Following with FEM depicted in [2] and using linear anisotropic model obtained by generalizing Hooke's law [7], we get stress-strain relationship:

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_{uu} \\ \sigma_{vv} \\ \sigma_{uv} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{13} & c_{23} & c_{33} \end{bmatrix} \begin{bmatrix} \varepsilon_{uu} \\ \varepsilon_{vv} \\ \varepsilon_{uv} \end{bmatrix} = \mathbf{C}\boldsymbol{\varepsilon} \tag{2}$$

in which $\boldsymbol{\sigma}$ and $\boldsymbol{\varepsilon}$ are the stress and strain tensor, and $\mathbf{C}$ is a 3×3 symmetric stiffness tensor matrix with six parameters. $\boldsymbol{\varepsilon}$ is Green strain tensor in finite strain theory, i.e. $\boldsymbol{\varepsilon} = \frac{1}{2}(\mathbf{F}^{\mathbf{T}}\mathbf{F} - \mathbf{I})$ where $\mathbf{F}$ is deformation gradient and we approximate $\mathbf{F}$ following [1]. Furthermore, according to [11], $\boldsymbol{\sigma}$ can be simplified into a matrix of four parameters($c_{12} = c_{21}$):

$$\mathbf{C} = \begin{bmatrix} c_{11} & c_{12} & 0 \\ c_{12} & c_{22} & 0 \\ 0 & 0 & c_{33} \end{bmatrix} \tag{3}$$

$c_{11}, c_{12}, c_{22}, c_{33}$ indicate material parameters, different thin shell has various value.

Meanwhile, we take $E_p = \frac{1}{2} a \cdot \boldsymbol{\varepsilon}^{\boldsymbol{T}} \cdot \mathbf{C} \cdot \boldsymbol{\varepsilon}$ and $E_b = \frac{(\theta_e - \bar{\theta}_e)^2}{\bar{h}_e} \|\bar{e}\|$ as energy function of plane force and bending force separately, in which a is element area and θ_e and $\bar{\theta}_e$ are corresponding complements of the dihedral angle of edge e measured in the deformed and undeformed configuration respectively, and $\bar{h}_e$ is a third of the average of the heights of the two triangles incident to the edge e [6]. Following with [2], we can derive the corresponding force $\mathbf{f_p}, \mathbf{f_b}$ and stiffness matrix K_p and K_b in every elements. Detailed derivation is complicated and tedious, you can see derivation in supplement manuscript appendix.

After processing with FEM, equation (1) becomes

$$\frac{\partial}{\partial t}\left(\mu \frac{\partial \mathbf{x}}{\partial t}\right) + \gamma \frac{\partial \mathbf{x}}{\partial t} + \mathbf{K_p x} + \mathbf{K_b x} = \mathbf{f}(\mathbf{x}, t) \tag{4}$$

where $\mathbf{K_p}, \mathbf{K_b}$ are stiffness matrix of total mesh collecting all of stiffness matrix K_p and K_b of every elements. Obviously, equation (4) is Ordinary Differentical Equation (ODE) with nonlinear item $\mathbf{f}(\mathbf{x}, t)$ with respect to time t. Assuming mass density μ is constant in thin shell, we can derive further from equation (4)

$$\underbrace{\mathbf{M}\frac{\partial^2 \mathbf{x}}{\partial t^2}}_{\text{inertia force}} + \underbrace{\gamma \frac{\partial \mathbf{x}}{\partial t}}_{\text{damping force}} + \underbrace{\mathbf{K_p x} + \mathbf{K_b x}}_{\text{internal force}} = \underbrace{\mathbf{f}(\mathbf{x}, t)}_{\text{external force}} \tag{5}$$

in which $\mathbf{M}$ is $3n \times 3n$ diagonal mass matrix whose entry is mass of vertex, m_i indicates mass of vertex v_i,

$$\begin{pmatrix} m_1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & m_1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & m_1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & m_n & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & m_n & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & m_n \end{pmatrix} \tag{6}$$

In equation (5), $\mathbf{M}\frac{\partial^2 \mathbf{x}}{\partial t^2}$ is inertia force, $\gamma \frac{\partial \mathbf{x}}{\partial t}$ is damping force, $\mathbf{K_p x} + \mathbf{K_b x}$ is internal force and $\mathbf{f}(\mathbf{x}, t)$ is external force. That is to say, equation (5) is Newton's second theorem which is time-varying ODE. For convenience, we write equation (5) as

$$\mathbf{M}\frac{\partial^2 \mathbf{x}}{\partial t^2} = \mathbf{f_I}(\mathbf{x}, \mathbf{t}) + \mathbf{f_E} \tag{7}$$

$\mathbf{f_I}(\mathbf{x}, \mathbf{t})$ is total internal force and corresponding damping force, $\mathbf{f_E}$ is total external force. The following section discretize time domain with FDM.

2.2. Finite Difference Method (FDM)

Given the known position $\mathbf{x}(t)$ and velocity $\mathbf{v}(t)$ of the system at time t, our goal is to determine a new position $\mathbf{x}(t + \Delta t)$ and velocity $\mathbf{v}(t + \Delta t)$ at time $t + \Delta t$. To compute the new position and velocity using an implicit technique, we must first transform equation (7) into a differential equation. Here, because $\mathbf{f_E}$ is independent of $\mathbf{x}$ and $\mathbf{v}$, for convenience, it can be omitted in the following derivation. Besides, both $\mathbf{K_p}$ and $\mathbf{K}_b$ are denoted as $\mathbf{K}$ (the simplification is reasonable and only incurs a sum $\mathbf{K_p}$ and $\mathbf{K}_b$ in final implementation) Define the system's velocity $\mathbf{v}$ as $\mathbf{v} = \frac{\partial \mathbf{x}}{\partial t}$ and then write equation (7) as system of difference equations:

$$\frac{\partial}{\partial t}\begin{pmatrix} \mathbf{x} \\ \frac{\partial \mathbf{x}}{\partial t} \end{pmatrix} = \frac{\partial}{\partial t}\begin{pmatrix} \mathbf{x} \\ \mathbf{v} \end{pmatrix} = \begin{pmatrix} \mathbf{v} \\ \mathbf{M^{-1}} \mathbf{f_I}(\mathbf{x}, \mathbf{v}) \end{pmatrix} \tag{8}$$

To simplify notation, we will define $\mathbf{x} = \mathbf{x}(t)$ and $\mathbf{v} = \mathbf{v}(t)$. We also define $\mathbf{\Delta x} = \mathbf{x}(t + \Delta t) - \mathbf{x}(t)$ and $\mathbf{\Delta v} = \mathbf{v}(t + \Delta t) - \mathbf{v}(t)$.

The explicit *forward* Euler method applied to equation (8) approximates $\mathbf{\Delta x}$ and $\mathbf{\Delta v}$ as

$$\begin{pmatrix} \mathbf{\Delta x} \\ \mathbf{\Delta v} \end{pmatrix} = \Delta t \begin{pmatrix} \mathbf{v} \\ \mathbf{M^{-1}} \mathbf{f_I} \end{pmatrix} \tag{9}$$

Unfortunately, the step size $\mathbf{\Delta t}$ must be quite small to ensure stability when using this method. The implicit *backward* Euler method appears similar at first: $\mathbf{\Delta x}$ and $\mathbf{\Delta v}$ are approximated by

$$\begin{pmatrix} \mathbf{\Delta x} \\ \mathbf{\Delta v} \end{pmatrix} = \Delta t \begin{pmatrix} \mathbf{v} + \mathbf{\Delta v} \\ \mathbf{M}^{-1} \mathbf{f_I}(\mathbf{x} + \mathbf{\Delta x}, \mathbf{v} + \mathbf{\Delta v}) \end{pmatrix} \tag{10}$$

Recalling $\mathbf{f_I}$ indicates total internal force and corresponding damping force, we rewrite the second row of equation (10) as follows to solve $\mathbf{\Delta v}$.

$$M \frac{\mathbf{\Delta v}}{\Delta t} = \mathbf{K}(\mathbf{x} + \mathbf{\Delta x}) + \gamma \cdot \mathbf{K}(\mathbf{v} + \mathbf{\Delta v}) \tag{11}$$

Recalling our sign appointment at the beginning of section, $\mathbf{K}$ indicates $\mathbf{K_p}$ or $\mathbf{K_b}$. And then

$$\begin{aligned} M\mathbf{\Delta v} &= \Delta t[\mathbf{K}(\mathbf{x} + \mathbf{\Delta x}) + \gamma \cdot \mathbf{K}(\mathbf{v} + \mathbf{\Delta v})] \\ &= \Delta t[\mathbf{Kx} + \mathbf{K\Delta x} + \gamma \cdot \mathbf{Kv} + \gamma \cdot \mathbf{K\Delta x}] \\ &= \mathbf{Kx}\Delta t + \mathbf{K}\Delta t\mathbf{\Delta x} + \gamma \cdot \mathbf{K}\Delta t\mathbf{v} + \\ &\quad \mathbf{K} \cdot \gamma \cdot \Delta t\mathbf{\Delta v} \end{aligned} \tag{12}$$

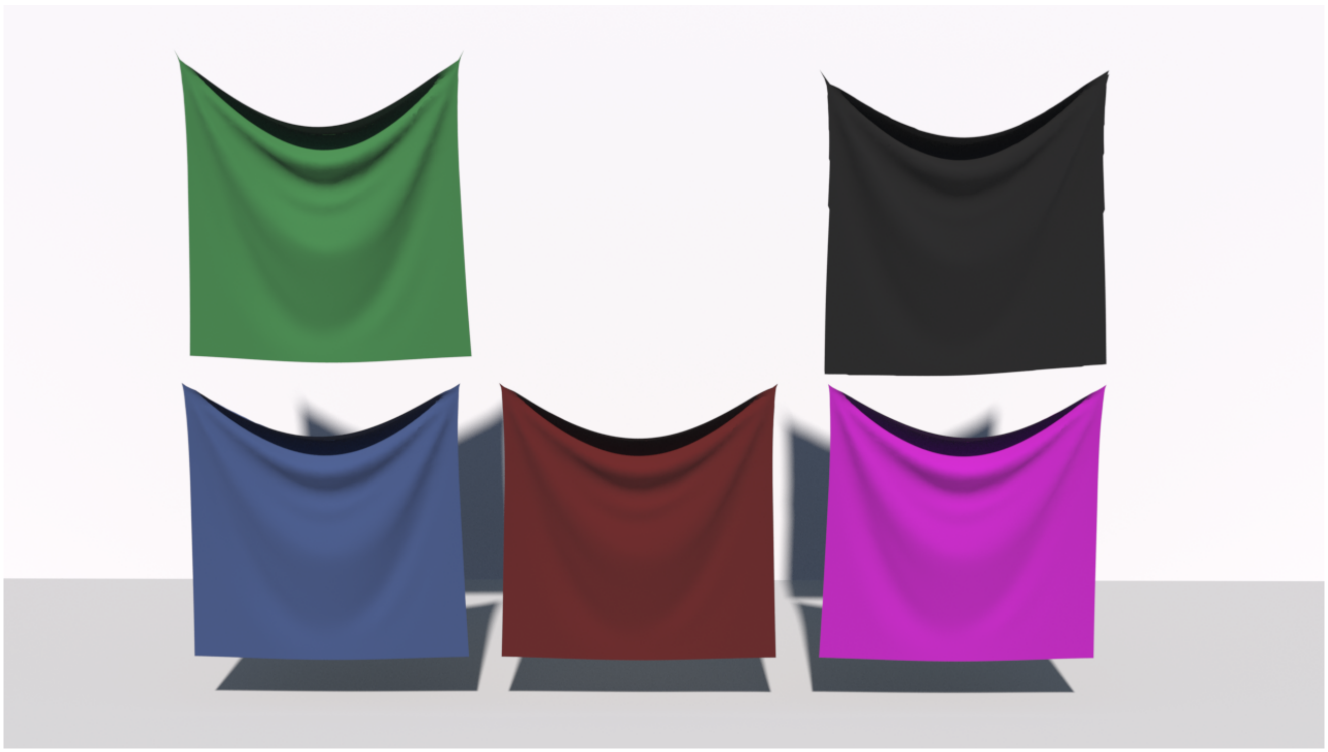

Figure 1. Simulation Visualization With Offline Render (Render using Houdini FX 17.0.352)

Note that $\mathbf{\Delta x} = \Delta t(\mathbf{v} + \mathbf{\Delta v})$, substitute it into equation (12), we get

$$\begin{aligned} M\mathbf{\Delta v} =& \mathbf{Kx}\Delta t + \mathbf{K}(\Delta t)^2\mathbf{v} + \mathbf{K}(\Delta t)^2\mathbf{\Delta v} + \\ & \gamma \cdot \mathbf{K}\Delta t\mathbf{v} + \mathbf{K} \cdot \gamma \cdot \Delta t\mathbf{\Delta v} \end{aligned} \tag{13}$$

Let $\mathbf{Kx} = \mathbf{f}$, $\mathbf{f}$ indicates $\mathbf{f_p}$ or $\mathbf{f_b}$, to rearrange equation (13):

$$\begin{aligned} &(M - (\Delta t)^2\mathbf{K} - \Delta t \cdot \gamma \cdot \mathbf{K})\mathbf{\Delta v} \\ &\qquad = \mathbf{Kx}\Delta t + \mathbf{Kv}(\Delta t)^2 + \Delta t \cdot \gamma \cdot \mathbf{Kv} \\ &\qquad = \Delta t\mathbf{f} + (\Delta t)^2\mathbf{Kv} + \Delta t \cdot \gamma \cdot \mathbf{Kv} \end{aligned} \tag{14}$$

$$\Longrightarrow$$

$$\begin{aligned} &[M - \Delta t(\Delta t(\Delta t + \gamma)\mathbf{K})]\mathbf{\Delta v} \\ &\qquad = \Delta t[\mathbf{f} + (\Delta t + \gamma)\mathbf{Kv}] \end{aligned} \tag{15}$$

Therefore, equation (15) can be regarded as a system of linear equations,

$$A\mathcal{X} = b \tag{16}$$

Where,

$$\begin{aligned} A &= M - \Delta t(\Delta t(\Delta t + \gamma)\mathbf{K}) \\ b &= \mathbf{f} + (\Delta t + \gamma)\mathbf{Kv} \\ \mathcal{X} &= \Delta v \end{aligned} \tag{17}$$

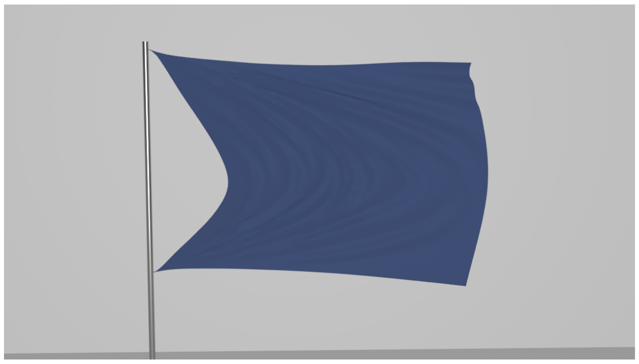

Figure 2. Wind blows from the side towards the flag (Render using Houdini FX 17.0.352)

After solving Δv, substitute it into the first row of equation (10) to get Δx. Finaly, the value of next step $\mathbf{x}$ can be calculated by

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \mathbf{\Delta x} \tag{18}$$

3. Results

To implement more realistic thin shell simulation, it is necessary to use practical material. Acquiring practical material parameters is not an easy task. Fortunately, [11]

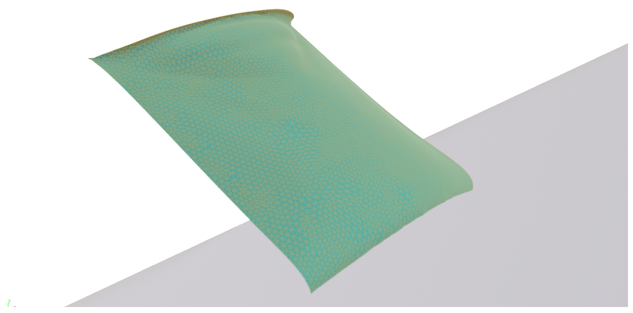

Figure 3. Wind blows from behind towards the mesh (Render using Houdini FX 17.0.352)

acquires some cloth materials by data-driven method, we use those materials as input of simulator. You can change material types by configuring the configuration file of lyra. Please refer to README.md for detailed usage of lyra.

For visualizing simulation results and checking our final project, lyra provides a simple realtime render based on OpenGL. But our simulation data can be rednered by any other offline render such as Houdini, C4D etc. Figure 1,2,3 are some examples which rendered by Houdini.

4. Acknowledgement

Although lyra is a simple thin shell simulator far too from practical application. It is not a easy task to implementing it from scratch using C++. we draw lessons from some excellent open source code such as opencloth, Argus etc. For solving linear system, we reuse packaging eigen solver and fundamental data structure(vector, matrix and transformation etc) from Argus. For thin shell motion parser, we reuse motion parser from opencloth.

References

[1] 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, page 43–54, New York, NY, USA, 1998. Association for Computing Machinery. 1

[2] Bgreen. 2d triangular elements. Technical report, 2010. 1, 2

[3] D. E. Breen, D. H. House, and M. J. Wozny. Predicting the drape of woven cloth using interacting particles. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, page 365–372, New York, NY, USA, 1994. Association for Computing Machinery. 1

[4] M. Carignan, Y. Yang, N. M. Thalmann, and D. Thalmann. Dressing animated synthetic actors with complex deformable clothes. *SIGGRAPH Comput. Graph.*, 26(2):99–104, July 1992. 1

[5] B. Eberhardt, A. Weber, and W. Strasser. A fast, flexible, particle-system model for cloth draping. *IEEE Computer Graphics and Applications*, 16(5):52–59, Sep. 1996. 1

[6] E. Grinspun, A. N. Hirani, M. Desbrun, and P. Schröder. Discrete shells. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '03, page 62–67, Goslar, DEU, 2003. Eurographics Association. 2

[7] W. Slaughter. *The Linearized Theory of Elasticity*. Springer, 2002. 1

[8] D. Terzopoulos and K. Fleischer. Deformable models. *The Visual Computer*, 4:306–331, 11 1988. 1

[9] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, page 205–214, New York, NY, USA, 1987. Association for Computing Machinery. 1

[10] P. Volino, M. Courchesne, and N. Magnenat Thalmann. Versatile and efficient techniques for simulating cloth and other deformable objects. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, page 137–144, New York, NY, USA, 1995. Association for Computing Machinery. 1

[11] H. Wang, J. F. O'Brien, and R. Ramamoorthi. Data-driven elastic models for cloth: Modeling and measurement. *ACM Trans. Graph.*, 30(4), July 2011. 2, 3