Rigid Body Modeling in *FronTier++* and Its Application to Parachute System Simulations

A Dissertation Presented

by

Xiaolei Chen

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Applied Mathematics and Statistics

Stony Brook University

August 2017

Stony Brook University

The Graduate School

Xiaolei Chen

We, the dissertation committee for the above candidate for the
Doctor of Philosophy degree, hereby recommend
acceptance of this dissertation.

Xiaolin Li - Dissertation Advisor
Professor, Department of Applied Mathematics and Statistics

name - Chairperson of Defense
Professor, Department of Applied Mathematics and Statistics

name - Member
Professor, Department of Applied Mathematics and Statistics

name - Outside Member
Professor, Department of Mathematics

This dissertation is accepted by the Graduate School.

Charles Taber
Dean of the Graduate School

Abstract of the Dissertation

Rigid Body Modeling in *FronTier++* and Its Application to Parachute System Simulations

by

Xiaolei Chen

Doctor of Philosophy

in

Applied Mathematics and Statistics

Stony Brook University

2017

Abstract Here

Key Words: Rigid Body

To my Parents and all loving ones

Table of Contents

List of Figures

List of Tables

Acknowledgements

Chapter 1

Introduction

[8, 9]

Chapter 2

Mathematical Models

In this chapter, we first have a review of the rigid body dynamics which is governed by the Euler's equations, and the fluid dynamics that is described by three conservation laws. Then, we give a detailed description of the new features for our parachute simulation platform, such as enhanced spring-mass model, parachutist inclusion and collision handling. As mentioned in Chapter 1, our model is built on the platform *FronTier++* which handles moving boundaries by front tracking method [15, 11].

2.1 Rigid Body Dynamics

A rigid body is defined as a system of mass points subject to the holonomic constraints that the distances between all pairs of points remain constant throughout the motion. Of essential importance is the rotational motion of a rigid body. These considerations lead directly to the relation between the time rate of change of a vector in an inertial frame (space axes) and the time rate of change of the same vector in a rotating frame (body axes).

In physics, the angular momentum $\mathbf{L}$ is a measure of the amount of rotation an object has, taking into account its mass, shape and speed. The angular momentum of a particle about a given origin is defined as

$$\mathbf{L} = \mathbf{r} \times \mathbf{p}, \tag{2.1}$$

where $\mathbf{r}$ is the position vector of the particle relative to the origin, and $\mathbf{p}$ is the linear momentum of the particle. This is equivalent to

$$\mathbf{L} = \mathbf{r} \times (m\mathbf{v}), \tag{2.2}$$

where, m is the mass of the particle and $\mathbf{v}$ is the velocity of the particle. With the relationship between the angular velocity and linear velocity, $\mathbf{v} = \mathbf{r} \times \mathbf{w}$, the angular momentum can be expressed as

$$\mathbf{L} = m\mathbf{r} \times (\mathbf{w} \times \mathbf{r}) = m(\mathbf{w}(\mathbf{r} \cdot \mathbf{r}) - \mathbf{r}(\mathbf{r} \cdot \mathbf{w})), \tag{2.3}$$

where $\mathbf{w}$ is the angular velocity of the particle. The equation Eq. (2.3) can be expanded into each component as

$$\begin{cases} L_1 = m(r^2 - x^2)w_1 - mxyw_2 - mxzw_3 \\ L_2 = -mxyw_1 + m(r^2 - y^2)w_2 - myzw_3 \\ L_3 = -mxzw_1 - myzw_2 + m(r^2 - z^2)w_3 \end{cases}, \tag{2.4}$$

where r is the length of the position vector $\mathbf{r}$. Then the angular momentum is a vector quantity that represents the product of a particle's rotation inertia

and angular velocity about a particular axis.

$$\mathbf{L} = \mathbf{I}\mathbf{w},\ \mathbf{I} = \begin{pmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{pmatrix}, \tag{2.5}$$

where $\mathbf{I}$ is called the inertia tensor, and $I_{ab} = m(r^2\delta_{ab} - ab)$.

Moment of inertia is the mass property of a rigid body that determines the torque needed for a desired angular acceleration about an axis of rotation. It can be defined by the inertia tensor $\mathbf{I}$, and in a continuous case, we have

$$I_{ab} = \oint_V \rho(V)(r^2\delta_{ab} - ab)dV. \tag{2.6}$$

The inertia tensor $\mathbf{I}$ is constant with respect to the body axes (the axes that moves along with the rigid body). Apply the eigen-decomposition to the inertia tensor, and we can obtain

$$I = Q\Lambda Q^T,\ \Lambda = diag\{I_1, I_2, I_3\}. \tag{2.7}$$

We call I_i principle moments of inertia and the columns of $Q = (Q_1, Q_2, Q_3)$ are the corresponding principle axes. The rigid body rotates in the angular velocity $\mathbf{w} = (w_1, w_2, w_3)$ with respect to the principle axes Q_1, Q_2, Q_3. Specially, in the case where the inertia tensor $\mathbf{I}$ is diagonal, such as a ball rotating about its center of mass, a cone rotates about its apex, the principles axes are exactly the x, y, z axis.

Euler's Equations

In the rigid body motion process, there are two systems of axes, space axes (also called world axes) and the body axes. For any three dimensional vector $\mathbf{G}$, its time rate of change in two systems of axes has the following relationship

$$(\frac{d\mathbf{G}}{dt})_s = (\frac{d\mathbf{G}}{dt})_b + \mathbf{w} \times \mathbf{G}, \tag{2.8}$$

where, the subscript s means the space axes and the subscript b means the body axes. Apply the angular velocity L to Eq. (2.8).

$$(\frac{d\mathbf{L}}{dt})_b + \mathbf{w} \times \mathbf{L} = (\frac{d\mathbf{L}}{dt})_s \tag{2.9}$$

$$\frac{d(\mathbf{Iw})}{dt} + \mathbf{w} \times (\mathbf{Iw}) = \tau, \tag{2.10}$$

where τ is the outside torque, the time rate of change of the angular moment in the space axes. The right hand side of Eq. (2.10) is derived as

$$\begin{aligned}(\frac{d\mathbf{L}}{dt})_s &= (\frac{d(\mathbf{r} \times \mathbf{p})}{dt})_s \\ &= (\frac{d\mathbf{r}}{dt} \times \mathbf{p})_s + (\mathbf{r} \times \frac{d\mathbf{p}}{dt})_s \\ &= (\mathbf{v} \times (m\mathbf{v}))_s + (\mathbf{r} \times \mathbf{F})_s \\ &= \tau\end{aligned} \tag{2.11}$$

Expanding Eq. (2.10) in each component leads to the Euler's equations for rigid body dynamics.

$$\begin{cases} I_1\dot{w}_1 - w_2w_3(I_2 - I_3) = \tau_1 \\ I_2\dot{w}_2 - w_3w_1(I_3 - I_1) = \tau_2 \\ I_3\dot{w}_3 - w_1w_2(I_1 - I_2) = \tau_3 \end{cases} \tag{2.12}$$

Eq. (2.12) plays a key role to simulate the rigid body motion.

Euler Angles

The Euler Angles are three angles introduced to describe the orientation of a rigid body, and they are typically denoted as ϕ, θ, ψ (refer to figure Fig. 2.1). They represent the spatial orientation as a composition of three elemental rotations starting from a known standard orientation.

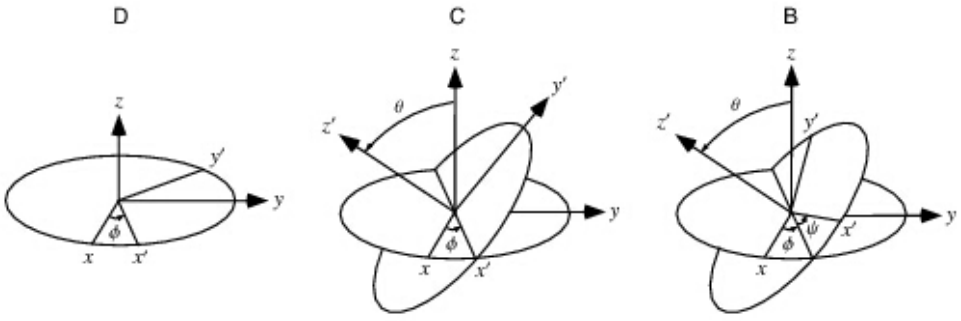


Figure 2.1: Demonstration of the Euler Angles. The first angle ϕ is with reapect to the z axis; the second angle θ is with respect to the x axis; the third angle ψ is with repect to the y axis.

Each angle provides a rotation matrix.

$$D = \begin{pmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix}, C = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{pmatrix}, B = \begin{pmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.13}$$

One thing that we need to keep in mind is that the order of the three rotations matters. Different order results in different final status. The composition of them $A = BCD$ is the rotation matrix

$$A = \begin{pmatrix} \cos\psi\cos\phi - \cos\theta\sin\phi\sin\psi & \cos\psi\sin\phi + \cos\theta\cos\phi\cos\psi & \sin\psi\sin\theta \\ -\sin\psi\cos\phi - \cos\theta\sin\phi\cos\psi & -\sin\psi\sin\phi + \cos\theta\cos\phi\cos\psi & \cos\psi\sin\theta \\ \sin\theta\sin\phi & -\sin\theta\cos\phi & \cos\theta \end{pmatrix}. \tag{2.14}$$

Angles are commonly defined according to the right hand rule. Namely, they have positive values when they represent a rotation that appears clockwise when looking in the positive direction of the axis, and negative values when the rotation appears counter-clockwise. About the ranges, ϕ and ψ have a range of 2π, and θ has a range of π.

2.2 Fluid Dynamics

Two most important set of governing equations that describe the motion of the fluid are Navier-Stokes equations and Euler equations. The basic method to derive both formulas is through principles of conservation of mass, momentum and energy [20, 30].

- conservation of mass:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0, \tag{2.15}$$

where $\rho(\mathbf{x}, t)$ is the spatial mass density field and $\mathbf{u}(\mathbf{x}, t)$ is the associated spatial velocity field.

- conservation of linear momentum:

$$\rho[\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u}] = \nabla \cdot \mathbf{S} + \rho \mathbf{b}, \tag{2.16}$$

where $\mathbf{S}(\mathbf{x}, t)$ is the Cauchy stress field and $\mathbf{b}(\mathbf{x}, t)$ is the spatial body force per unit mass. In addition, the conservation of angular momentum law leads to the symmetry constraint

$$\mathbf{S}^T = \mathbf{S}. \tag{2.17}$$

- conservation of energy:

$$\rho[\frac{\partial e}{\partial t} + (\mathbf{u} \cdot \nabla)e] = \nabla(\mathbf{S} \cdot \mathbf{u}) - \nabla \cdot \mathbf{q} + \rho r \tag{2.18}$$

where $e(\mathbf{x}, t)$ is the internal energy field per unit mass, $\mathbf{q}(\mathbf{x}, t)$ is the Fourier-Stokes heat flux vector field, $r(\mathbf{x}, t)$ is the heat supply field per unit mass.

Different choices of stress fields $\mathbf{S}(\mathbf{x}, t)$ leads to equation for different type of fluid.

Euler Equations

In the world of ideal fluid which is inviscid, the Cauchy stress field is given as

$$\mathbf{S}(\mathbf{x}, t) = -p(\mathbf{x}, t)\mathbf{I}. \tag{2.19}$$

Then, for the ideal fluid, in the case of absent extra body force, i.e. $\mathbf{b}(\mathbf{x}, t) = 0$, Euler equation can be written as

$$\begin{cases} \dfrac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \\ \dfrac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\dfrac{1}{\rho}\nabla p \\ \dfrac{\partial E}{\partial t} + \nabla \cdot (\mathbf{u}(E + p)) = 0, \end{cases} \tag{2.20}$$

where ρ is the fluid density, $\mathbf{u}$ is the velocity of the fluid field, p is the pressure, $E = \rho e + \frac{1}{2}\rho \mathbf{u}^T \mathbf{u}$ is the total energy density with e being the specific internal energy per unit mass.

In order to make the system of equation closed, an equation of state is essential. Under classical ideal gas law, it can be expressed as

$$p = \rho(\gamma - 1)e, \tag{2.21}$$

where γ is the adiabatic index.

With some algebric calculation, the momentum conservation can be rewritten in the conservative form

$$\frac{\partial(\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u}\mathbf{u}^T + p\mathbf{I}) = 0, \tag{2.22}$$

where $\mathbf{I}$ here is the 3×3 identity matrix. Then, the Euler equations become a hyperbolic system when no external force is taken into account,

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}(\mathbf{U})}{\partial x} + \frac{\partial \mathbf{G}(\mathbf{U})}{\partial y} + \frac{\partial \mathbf{H}(\mathbf{U})}{\partial z} = 0, \tag{2.23}$$

where

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ E \end{pmatrix}, \mathbf{F} = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ (E+p)u \end{pmatrix}, \mathbf{G} = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ (E+p)v \end{pmatrix}, \mathbf{H} = \begin{pmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ (E+p)w \end{pmatrix} \tag{2.24}$$

Due to the model assumption, Euler equation is most appropriate for simulating compressible inviscid fluid which has large Reynolds number, such as bullet ejection, spacecraft entrance and so on.

Navier-Stokes Equations

In the incompressible Newtonian fluid case, mass density field is uniform which means it is constant ρ in the field. Thus, Eq. (2.15) implies the so-called divergence-free condition

$$\nabla \cdot \mathbf{u} = 0. \tag{2.25}$$

The Newtonian Cauchy stress field is given as

$$\mathbf{S}(\mathbf{x}, t) = -p(\mathbf{x}, t)\mathbf{I} + \mu(\nabla \mathbf{u}(\mathbf{x}, t) + \nabla \mathbf{u}(\mathbf{x}, t)^T), \tag{2.26}$$

where μ is dynamic viscosity.

Considering only the gravitational force, i.e. $\mathbf{b}(\mathbf{x}, t) = \mathbf{g}$, Navier-Stokes equations for incompressible Newtonian fluid can be rewritten as

$$\begin{cases} \nabla \cdot \mathbf{u} = 0 \\ \dfrac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\dfrac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{g} \end{cases} \tag{2.27}$$

where $\nu = \mu/\rho$ is the kinematic viscosity and ρ_0 is the fluid density, $\mathbf{u}$ is the velocity of the fluid field, p is the pressure.

Due to the incompressible and viscous assumption, Navier-Stokes equations have wide applications in physics, geoscience, aerospace and even medical simulation when coupled with certain boundary conditions and other appropriate constraints.

2.3 Spring-Mass Model

Using spring system to model the dynamic motion of fabric surface has been studied by computer scientists and applied mathematicians in the past several decades, such as [7, 22, 23, 1]. Its main advantage is its simplicity and easy implementation. We built our spring-mass model with similar idea to [7] and improved it with Delingette's modification [10] to simulate the parachute canopy.

Triangulized mesh is adopted to represent the surface where the edges of those triangles are springs and the vertices are mass points, as shown in Fig. 2.2. For each spring vertex in the spring-mass model, its motion is gov-

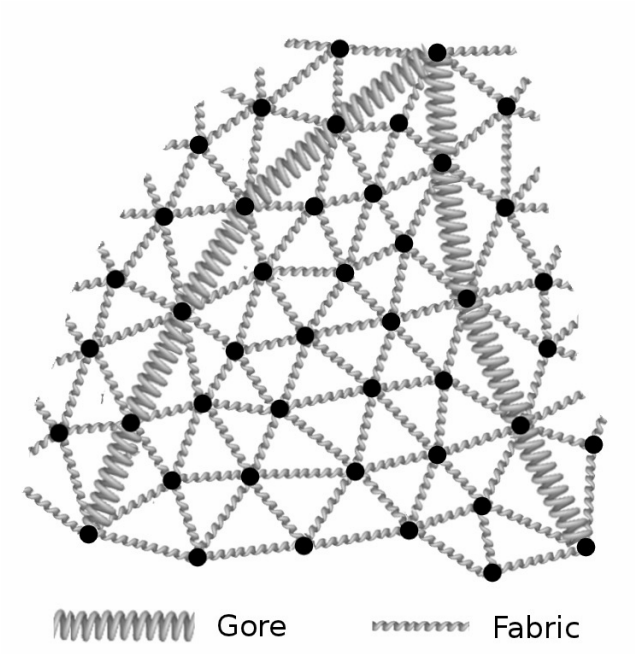


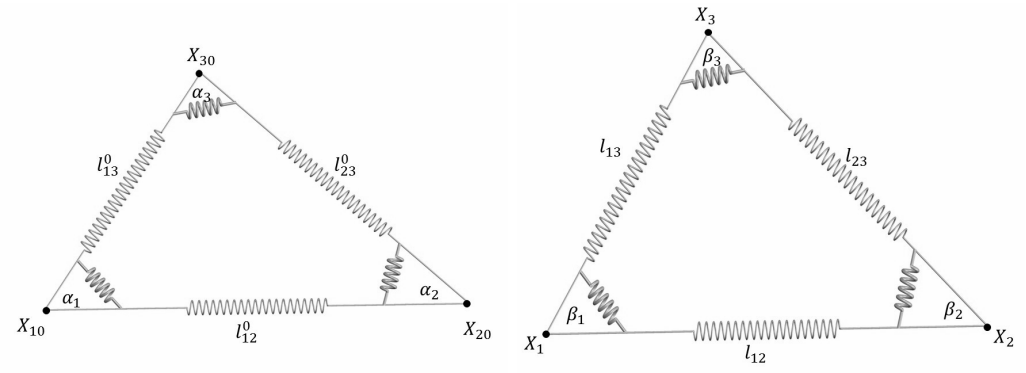
Figure 2.2: The spring model on a triangulated mesh. Each vertex point in the mesh represents a mass point with point mass m. Each edge of a triangle has a tensile stiffness. With the equilibrium lengths set during the initialization, the changing length of each side exerts a tensile spring force on the two neighboring vertex in opposite directions. Gore boundaries used in the parachute system can be added and modeled by curves with a different tensile stiffness.

erned by Newton's second law

$$m_i \frac{d\dot{\mathbf{X}}_i}{dt} = \mathbf{F}_i, \ i = 1, 2, \cdots, N, \tag{2.28}$$

where m_i is the mass, $\mathbf{X}_i$ is the position, and $\mathbf{F}_i$ is the total force inserted on the spring vertex.

In a elastic structure dynamics simulation, $\mathbf{F}_i$ consists of a force $\mathbf{F}_i^s$ due



(a) Triangle in the rest status (b) Triangle in the deformed status

Figure 2.3: Left: rest triangle T_{X_0} with vertex X_{i0}. Right: deformed triangle T_X with vertex X_i. Each edge is represented as a spring and the tensile stiffness prevents the deformation of the edge. At each corner, an artifical spring prevents the change of angles.

to the deformation of triangles and a damping force which slows the motion of the spring system due to friction $\mathbf{F}_i^d$. The external force $\mathbf{F}_i^e$, representing the effects of air drag and gravity is considered by adding an additional term on the right-hand side.

Our spring-mass model [29, 33] assumes the force to stretch the spring is proportional to its displacement from the equiblibrium distance between the two adjacent spring vertices. In addition, the force that prevents the change of angles is described by the angular stiffness in Delingette's modification [10] which is favored by its relation to the elastic spring model in continuum mechanics.

With $dl_{ij} = l_{ij} - l^0_{ij}$ representing the length change of the spring between

$\mathbf{X}_i$ and $\mathbf{X}_j$, the potential energy is given in [10] as

$$W(T) = \sum_{\substack{i=1 \\ j=(i+1) \mod 3}}^{3} \frac{1}{2} k_{ij}^T (dl_{ij})^2 + \sum_{\substack{i=1 \\ j=(i+1) \mod 3 \\ k=(i+2) \mod 3}}^{3} \gamma_i^T dl_{ij} dl_{ik}, \tag{2.29}$$

where the first term is the stretching energy, and the second term is the angular energy from Delingette's modification.

In Eq. (2.29), the tensile stiffness and the angular stiffness are calculated as

$$k_{ij}^T = \frac{(l_{ij}^0)^2 (2\cot^2 \alpha_k (\lambda + \mu) + \mu)}{8A}, \tag{2.30}$$

$$\gamma_i^T = \frac{l_{ik}^0 l_{ij}^0 (2\cot \alpha_j \cot \alpha_k (\lambda + \mu) - \mu)}{8A}, \tag{2.31}$$

where A denotes the area of the triangle in equilibrium, $i = 1, 2, 3$, $j = (i+1) \mod 3$, and $k = (i+2) \mod 3$. λ and μ are the Lamé coefficients of the material, and they are related to Young's modulus E and the Poisson ratio ν [14]

$$\lambda = \frac{E\nu}{1-\nu^2}, \ \mu = \frac{E(1-\nu)}{1-\nu^2}. \tag{2.32}$$

Apply Rayleigh-Ritz analysis and it tells that the fabric surface, represented by the triangular mesh, should evolve by minimizing its membrane energy. Therefore, along the opposite derivative of that energy with respect

to the nodes of the system, that is, the deformed positions X_i:

$$\begin{aligned}\mathbf{F}_i^s(T_{\mathbf{X}_0}) &= -\frac{\partial W(T_{\mathbf{X}_0})}{\partial \mathbf{X}_i} \\ &= \sum_{\substack{j=1 \\ j\neq i}}^{3} k_{ij}^{T_{\mathbf{X}_0}} dl_{ij} \frac{\mathbf{X}_j - \mathbf{X}_i}{l_{ij}} + \sum_{\substack{j=1, j\neq i \\ k\neq i, k\neq j}}^{3} (\gamma_k^{T_{\mathbf{X}_0}} dl_{kj} + \gamma_i^{T_{\mathbf{X}_0}} dl_{ij}) \frac{\mathbf{X}_k - \mathbf{X}_i}{l_{ik}}\end{aligned} \tag{2.33}$$

This is the force on vertex $\mathbf{X}_i$ due to the deformation of the triangular $T_{\mathbf{X}_0}$ and it can be split into two components

$$\mathbf{F}_i^s(T_{\mathbf{X}_0}) = \mathbf{F}_{ij}^s(T_{\mathbf{X}_0}) + \mathbf{F}_{ik}^s(T_{\mathbf{X}_0}), \tag{2.34}$$

where $j = (i+1)\ mod\ 3$ and $k = (i+2)\ mod\ 3$, and

$$\mathbf{F}_{ij}^s(T_{\mathbf{X}_0}) = k_{ij}^{T_{\mathbf{X}_0}} dl_{ij} \mathbf{e}_{ij} + (\gamma_i^{T_{\mathbf{X}_0}} dl_{ik} + \gamma_j^{T_{\mathbf{X}_0}} dl_{jk}) \mathbf{e}_{ij}, \tag{2.35}$$

$$\mathbf{F}_{ik}^s(T_{\mathbf{X}_0}) = k_{ik}^{T_{\mathbf{X}_0}} dl_{ik} \mathbf{e}_{ik} + (\gamma_i^{T_{\mathbf{X}_0}} dl_{ij} + \gamma_k^{T_{\mathbf{X}_0}} dl_{kj}) \mathbf{e}_{ik}. \tag{2.36}$$

Since each edge is shared by two triangles, the spring force between two vertex $\mathbf{X}_i$ and $\mathbf{X}_j$, as shown in Fig. 2.4, is the summation of the effects from both triangles and has the following formula:

$$\begin{aligned}\mathbf{F}_{ij}^s &= \mathbf{F}_{ij}^s(T_1) + \mathbf{F}_{ij}^s(T_2) \\ &= [(k_{ij}^{T_1} + k_{ij}^{T_2}) dl_{ij} + (\gamma_i^{T_1} dl_{im} + \gamma_j^{T_1} dl_{jm} + \gamma_i^{T_2} dl_{in} + \gamma_j^{T_2} dl_{jn})] \mathbf{e}_{ij} \\ &= \tilde{k}_{ij} dl_{ij} \mathbf{e}_{ij} + \tilde{\gamma}_{ij} dl_{ij} \mathbf{e}_{ij},\end{aligned} \tag{2.37}$$

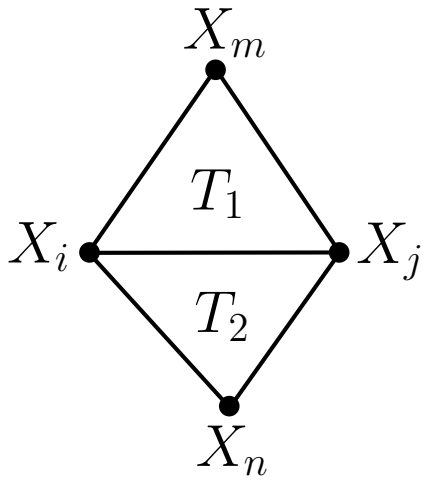


Figure 2.4: Two triangles T_1 and T_2 share $\mathbf{X}_i$ and $\mathbf{X}_j$; the other vertex of triangles T_1 and T_2 are $\mathbf{X}_m$ and $\mathbf{X}_n$ respectively.

where $\tilde{k}_{ij} = k_{ij}^{T_1} + k_{ij}^{T_2}$, $\tilde{\gamma}_{ij} = (\gamma_i^{T_1} dl_{im} + \gamma_j^{T_1} dl_{jm} + \gamma_i^{T_2} dl_{in} + \gamma_j^{T_2} dl_{jn})/dl_{ij}$.

As mentioned earlier, a damping force slows the spring system and we take it into account with the formula

$$\mathbf{F}_i^d = c\mathbf{V}_i^{(int)}, \tag{2.38}$$

where c is the viscous damping coefficient and $\mathbf{V}_i^{(int)}$ is the internal component of the vertex velocity that is due to the spring force only. On the other hand, the external component $\mathbf{V}_i^{(ext)}$ involves effects from gravity, fluid, and so on. With all forces considered, the equation of motion (2.28) can be re-written as an ODE system

$$\begin{cases} \dot{\mathbf{X}}_i = \mathbf{V}_i \\ m_i \dot{\mathbf{V}}_i = \mathbf{F}_i^s + \mathbf{F}_i^d + \mathbf{F}_i^e \end{cases} \tag{2.39}$$

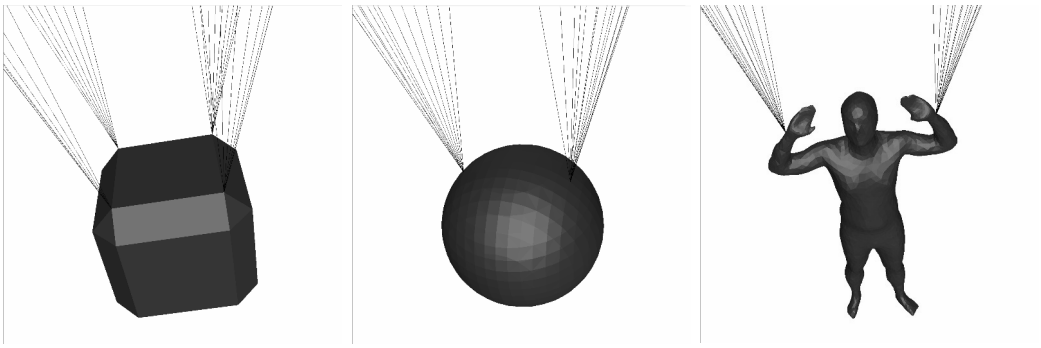
where $\mathbf{V}_i$ is the total velocity, i.e. the summation of $\mathbf{V}_i^{(int)}$ and $\mathbf{V}_i^{(ext)}$.

One thing that we need to keep in mind is that the canopy-like structures are lack of bending stiffness, therefore, the bending force can be negligible in the simulation of the parachute system. However, for simulations of other kind of fabric structures, bending force can be taken into account by adding an artificial spring between the two vertex, $\mathbf{X}_m$ and $\mathbf{X}_n$, that are not shared by two adjacent triangles. This additional resistance prevents local clustering of triangle meshes.

2.4 Parachutist/Cargo Inclusion

The dimensions of parachutist or cargo are usually smaller than the parachute canopy surface. However, when the fluid flow across the parachutist or cargo, the dynamical behavior of the flow will have different patterns based on different Reynolds numbers. These turbulent flow will reach the parachute canopy surface later and contribute to the motion of the canopy surface. Therefore the inclusion of parachutists will make numerical simulation of parachute systems much closer to reality. Fig. 2.5 show three examples of connecting the suspension lines to the parachutist.

In our model, the parachutist can be either a mass point with no instance or a rigid body with finite volume. In the former case, no interaction between the parachutist and the fluid is considered. Therefore, no turbulent flow is observed between the space of the canopy surface and the parachutist. However, in the latter case, the fluid-structure interaction is non-negligible, and there are two important aspects need to be taken into account carefully.



(a) A cuboid cargo (b) A spherical cargo (c) Human parachutist

Figure 2.5: Three examples of the connection amplification between the suspension lines and different types of cargos. Besides these three, *FronTier++* is capable of generating different kinds of rigid bodies.

- Calculation of force and torque on the cargo.

  The force consists of three parts: force of gravity, force exerted by the fluid flow and force from the suspension lines. The force and corresponding torque are then applied to the parachutist for translation and rotation.

  - The force exerted by the fluid flow can be broken into two parts: force due to pressure difference and force due to friction (viscosity). In the parachute simulation, we consider the first part as the dominant component.

    $$F_p = \int_s (p - p_0)\hat{n} dA, \tag{2.40}$$

    where $\hat{n}$ is the normal direction to the surface with area dA, p_0 is

the pressure far away from the surface dA and p is the pressure at surface dA.

- Propagation of the rigid body.

  The propagation must maintain the geometric shape of the parachutist. This has perfect match with the Lagrangian propagation of the front tracking method in *FronTier++* [15, 11].

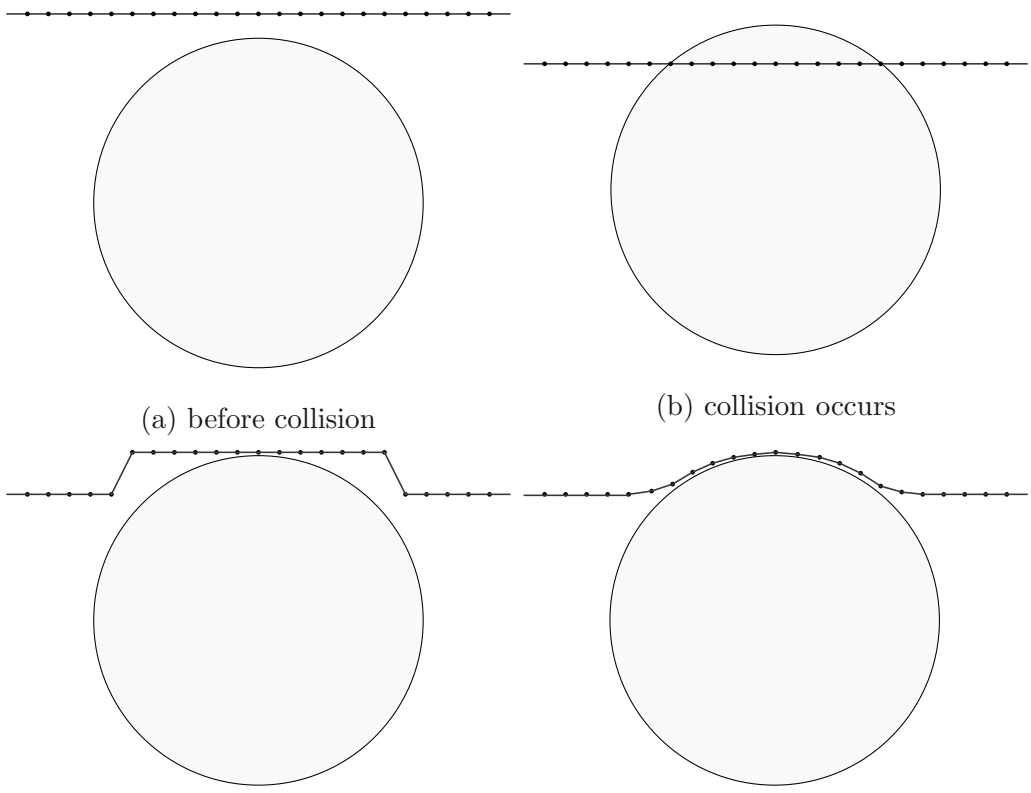
  - The translational motion is trivial. Simply apply the acceleration caused by the total force to the center of mass, and correspondingly propagate each point of the parachutist.
  - The rotational motion with respect to the center of mass is governed by Euler's equations Eq. (2.12) of rigid body dynamics [21].

2.5 Collision Handling

In cloth simulations, it is almost impossible to totally avoid collision. Thus, robust collision detection and handling is very important to make the numerical simulation visually realistic. The collision detection and handling problem has attracted scientists in many fields. Early work by Terzopoulosi *et al.*, [36, 35, 34], is governed by the mechanical laws of continuous bodies. Since then, many well-performed and well-known techniques are proposed by different groups, like Carignan *et al.* [6], Volino *et al.* [39, 37, 38], Baraff *et al.* [2, 3], Bridson *et al.* [4] and so on. Our collision algorithm is mainly based on the idea of combining a fail-safe geometric collision method and the repulsion

force method that is presented in [4]. In addition, we extended this algorithm to make it work for not only cloth, but also strings and rigid bodies.

In Fig. 2.6, the process of the collision algorithm is demonstrated for collision between fabric structure and a static object. As for self-collision of the fabric structure and collision between fabric structure and a movable object, the handling is similar. The only difference lies in how they move back to the time right before collision and how the impulse is applied.



(a) before collision

(b) collision occurs

(c) move back

(d) apply impulse

Figure 2.6: A demonstration of process how to resolve the collision. The red line represents the fabric structure; the light-blue disk represents an object; the points on the red line are the mass points in the spring model.

The following is a summary of the collision algorithm. The detection is done by axis-aligned bounding box (AABB) tree, and the details of impulse and impact zone technique are discussed in the following chapter.

- Pre-processing:
  - Starting at t^n with a time step Δt, record current positions $\mathbf{x}^n$ and velocities $\mathbf{v}^n$.
  - Propagate all structures to the candidate positions $\mathbf{x}^{can}$ with candidate velocities $\mathbf{v}^{can}$ at time t^{n+1}, and compute the average velocity during this time step, $\mathbf{v}^{ave} = (\mathbf{x}^{can} - \mathbf{x}^n)/\Delta t$.
  - Connect points on each rigid body (static or movable) as one impact zone.
- Detection and Handling:
  - Perform the proximity check for $\mathbf{x}^n$ and apply impulse and friction to update $\mathbf{v}^{ave}$.
  - Perform the collision check for propagation from $\mathbf{x}^n$ with velocity $\mathbf{v}^{ave}$, and resolve collisions by applying impulse and rigid impact zones technique to update $\mathbf{v}^{ave}$.
  - Update the candidate position of all the points by $\mathbf{x}^{can} = \mathbf{x}^n + \Delta t \mathbf{v}^{ave}$.
  - If there is no more collision with respect to $\mathbf{x}^{can}$, continue to the post-processing; otherwise, repeat the detection and handling process.

- Post-processing:
  - Advance the $\mathbf{v}^{ave}$ to $\mathbf{v}^{can}$ for points involved in collision, and, meanwhile, update the center of mass and center of mass velocity of the movable rigid bodies.
  - Continue to the next time step with $\mathbf{x}^{n+1} = \mathbf{x}^{can}$ and $\mathbf{v}^{n+1} = \mathbf{v}^{can}$ for all points.

Chapter 3

Numerical Implementations

As mentioned earlier, the simulation of parachute system involves three major parts: computational structure dynamics, computational fluid dynamics and fluid-structure interaction. In this chapter, we discuss the detailed numerical methods for each part. The data structures and many functionalities are based on *FronTier++* library developed for front tracking method [15, 11].

3.1 ODE Solvers

3.1.1 Rigid Body Rotation

As discussed in the Sec. 2.1, the governed equations for rigid body rotation are Eq. (2.12) which is an ODE system. We use the fourth-order Runge

Kutta method, Eq. (3.1), to solve for the angular velocity $\mathbf{w}$.

$$\begin{aligned} u^{n+1} &= u^n + h(k_1 + 2k_2 + 2k_3 + k_4)/6, \\ k_1 &= \mathscr{L}(t_n, u^n), \\ k_2 &= \mathscr{L}(t_n + h/2, u^n + hk_1/2), \\ k_3 &= \mathscr{L}(t_n + h/2, u^n + hk_2/2), \\ k_4 &= \mathscr{L}(t_n + h, u^n + hk_3), \end{aligned} \tag{3.1}$$

where h is the step size and $\mathscr{L}$ is a discretization spatial operator.

The Euler angles (ϕ, θ, ψ) are visually straightforward. However, they are difficult to use in numerical computation because of the involve of trigonometric functions in the rotation matrix Eq. (2.14), which could lead to singularity. For numerical purpose, we use the four-parameter representation, Euler parameters $\mathbf{e} = (e_0, e_1, e_2, e_3)$, for propagation of the rigid bodies, with the constraint Eq. (3.2).

$$e_0^2 + e_1^2 + e_2^2 + e_3^2 = 1 \tag{3.2}$$

Euler's rotation theorem states that, in three-dimensional space, any displacement of a rigid body such taht a point on the rigid body remain fixed, is equivalend to a single rotation about some axis that runs through the fixed point. This fixed point can be considered as the rotation center. The Euler parameters are an interpretation of Euler's rotation theorem, which is related

to an angle of rotation α and a unit vector along the axis of rotation $\mathbf{u}$.

$$\begin{aligned} e_0 &= \cos(\alpha/2) \\ \begin{pmatrix} e_1 \\ e_2 \\ e_3 \end{pmatrix} &= \mathbf{u}\sin(\alpha/2) \end{aligned} \tag{3.3}$$

The initial status means no rotation happens. Therefore, initially, the Euler parameters are $\mathbf{e} = (1, 0, 0, 0)$. The relationship between the Euler parameters $\mathbf{e}$ and the angular velocity $\mathbf{w}$ is given by Eq. (3.4).

$$\begin{pmatrix} \dot{e}_0 \\ \dot{e}_1 \\ \dot{e}_2 \\ \dot{e}_3 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} -e_1 & -e_2 & -e_3 \\ e_0 & e_3 & -e_2 \\ -e_3 & e_0 & e_1 \\ e_2 & -e_1 & e_0 \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} \tag{3.4}$$

As well, fourth-order Runge Kutta method Eq. (3.1) is applied.

Under this circumstance, the rotation matrix A can be written in terms of the Euler parameters $\mathbf{e}$ as

$$A = \begin{pmatrix} e_0^2 + e_1^2 - e_2^2 - e_3^2 & 2(e_1e_2 + e_0e_3) & 2(e_1e_3 - e_0e_2) \\ 2(e_1e_2 - e_0e_3) & e_0^2 - e_1^2 + e_2^2 - e_3^2 & 2(e_2e_3 + e_0e_1) \\ 2(e_1e_3 + e_0e_2) & 2(e_2e_3 - e_0e_1) & e_0^2 - e_1^2 - e_2^2 + e_3^2 \end{pmatrix} \tag{3.5}$$

The transpose A^T is also the inverse of the rotation matrix A. $x' = Ax$ represents the rotation process from the world/space coordinates system to

the body-fixed coordinates system, and $x = A^T x'$ does the opposite process. Because the order of rotation in 3 dimensional cases matters, when propagating the rigid body from time t_n to t_{n+1}, we have the following equation.

$$x_{n+1} = A_{n+1}^T A_n x_n \tag{3.6}$$

The subscripts n and $n+1$ are the time step index. This means we always first convert to the body-fixed coordinates system before apply the updated rotation matrix.

3.1.2 Spring Solver

For the spring-mass model, explicit integration with fourth order Runge-Kutta method Eq. (3.1) is adopted to solve the ODE system Eq. (2.39).

The explicit scheme requires us to pay attention to the numerical stability of the spring solver. Based on the previous analysis and numerical verification in [29, 27], the spring-mass system is a conservative system without external forces, and there exists an upper bound for the eigenfrequency of the oscillation

$$w_{max} \leq \sqrt{\frac{2Mk}{m}}, \tag{3.7}$$

where M is the maximum number of neighbors that a spring vertex has, k is the tensile stiffness and m is the point mass. The Delingette's modification adds variation to the spring model, but the upper bound is still valid except that the maximum tensile stiffness is used in Eq. (3.7) instead. Therefore, for

stability and accuracy purpose, we choose $w_{max}h < 0.1$ for simulations.

3.2 PDE Solvers

There have been many techniques to solve the governed equations that describes the motion of the fluid. For compressible fluid, Eq. (2.23), since it is a hyperbolic system, we apply the 5th Weighted Essentially Non-Oscillatory (WENO) scheme. For incompressible fluid, a pressure correction projection method is used, which can achieve second order accuracy.

3.2.1 WENO Scheme

High-order numerical methods have been widely used to effectively resolve complex flow features such as turbulent or vertical flows [12]. High-order shock-capturing schemes such as the Essentially Non-Oscillatory (ENO) and Weighted ENO (WENO) [31, 24] schemes not only make the computational fluid dynamics (CFD) solvers get rid of extremely fine mesh for complex flows, but also perfectly eliminate the oscillations near discontinuities. WENO scheme is used to reconstruct the spatial derivative term and then the hyperbolic system is converted to an ODE system and propagated by 3rd order Total Variation Diminishing (TVD) Runge-Kutta method.

Take one-dimensional Euler equation Eq. (3.8) for example

$$\mathbf{U}_t + \mathbf{F}\left(\mathbf{U}\right)_x = 0 \tag{3.8}$$

where

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho u \\ E \end{pmatrix}, \ \mathbf{F}(\mathbf{U}) = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ (E+p)u \end{pmatrix} \tag{3.9}$$

$$E = \rho(e + \frac{u^2}{2}), \ p = \rho e(\gamma - 1) \tag{3.10}$$

ρ, u, P, e and γ denote the density, velocity, pressure, internal energy per unit mass and ratio of specific heats, respectively. The Jacobian matrix of the Euler equations is defined as

$$\mathbf{A} = \frac{\partial \mathbf{F}(\mathbf{U})}{\partial \mathbf{U}} \tag{3.11}$$

The eigenvalues of the Jacobian matrix A are

$$\lambda_1 = u - c, \ \lambda_2 = u, \ \lambda_3 = u + c \tag{3.12}$$

where $c = \sqrt{\frac{\gamma p}{\rho}}$ is the sound speed, and the corresponding right eigenvectors are

$$\mathbf{r}_1 = \begin{pmatrix} 1 \\ u - a \\ H - ua \end{pmatrix}, \ \mathbf{r}_2 = \begin{pmatrix} 1 \\ u \\ \frac{u^2}{2} \end{pmatrix}, \ \mathbf{r}_3 = \begin{pmatrix} 1 \\ u + a \\ H + ua \end{pmatrix} \tag{3.13}$$

where H is the total specific enthalpy, which is related to the specific enthalpy h and other variables, namely

$$H = \frac{E+p}{\rho} = \frac{1}{2}u^2 + h, \ h = e + \frac{P}{\rho} \tag{3.14}$$

We denote the matrix whose columns are eigenvectors in Eq. (3.13) by

$$\mathbf{R} = (\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3) \tag{3.15}$$

and denote $\mathbf{L} = \mathbf{R}^{-1}$. The eigenvalue decomposition of $\mathbf{A}$ is

$$\mathbf{A} = \mathbf{R}\mathbf{\Lambda}\mathbf{L} \tag{3.16}$$

where $\mathbf{\Lambda} = diag(u - c, u, u + c)$. Then, Eq. (3.8) can be transformed as

$$(\mathbf{L}\mathbf{U})_t + \Lambda(\mathbf{L}\mathbf{U})_x = 0 \tag{3.17}$$

which consists of three independent one-dimensional hyperbolic equations. The advection term in each equation can be reconstructed using WENO scheme then solved by advection solver.

3.2.2 Projection Method

Projection method is an effective way to numerically solve the time-dependent incompressible fluid flow problems. We apply the pressure-Poisson version of projection method [8, 26, 5] is adopted to solve the Navier-Stokes Eq. (2.27).

- step 1: Solve for the intermediate velocity u^*.

$$\begin{aligned} \frac{u^* - u^n}{\Delta t} + \frac{1}{\rho}\nabla q &= -[(u \cdot \nabla)u]^{n+\frac{1}{2}} + \frac{\mu}{2\rho}\nabla^2(u^* + u^n), \\ B(u^*) &= 0, \end{aligned} \tag{3.18}$$

where q represents an approximation of $p^{n+\frac{1}{2}}$ and $B(u^*)$ is the boundary condition for u^*. The term $[(u \cdot \nabla)u]^{n+\frac{1}{2}}$ represents an approximation of the advection term, which is done by WENO scheme. Crank-Nicolson scheme is used to solve this advection-diffusion equation.

- step 2: Perform the projection

$$\begin{aligned} u^* &= u^{n+1} + \Delta t \nabla \phi^{n+1} \\ \nabla \cdot u^{n+1} &= 0 \end{aligned} \tag{3.19}$$

using the boundary conditions consistent with $B(u^*) = 0$ and $u^{n+1}|_{\partial\Omega} = u_b^{n+1}$. After elimimating u^{n+1}, the above two equation becomes a Poisson equation of ϕ which is solved by iterative methods.

- step 3: Update the pressure.

$$p^{n+\frac{1}{2}} = q + \phi^{n+1} - \frac{\mu \Delta t}{2\rho} \nabla^2 \phi^{n+1}. \tag{3.20}$$

The last term in the above equation gives the second-order accuracy.

3.3 Fluid-Structure Interaction

The interaction between fluid and fabric surface structure is a crucial part of the parachute system simulation. As mentioned above, the canopy experiences three forces, the gravitational force due to the weight of the material, the fluid-generated force due to the pressure difference on two sides of the canopy, and the internal spring force (both restoring force and friction force).

3.3.1 Force Calculation

We handle this by the impulse method [27] on the *FronTier++* platform. For each mass point in the spring system, we divide the impulse into three components, the gravitational impulse, the fluid impulse and the internal impulse due to the spring system, that is,

$$\mathbf{I}_i^C = \mathbf{I}_{gi}^C + \mathbf{I}_{pi}^C + \mathbf{I}_{si}^C. \tag{3.21}$$

Since no fluid interaction with the string chord is considered in our model, the impulse splitting for these mass points is

$$\mathbf{I}_i^S = \mathbf{I}_{gi}^S + \mathbf{I}_{si}^S. \tag{3.22}$$

The gravitational impulse is straightforward and has the same formula for the spring vertex on both the canopy and string chords.

$$\mathbf{I}_{gi} = \int_0^t m_i \mathbf{g} ds. \tag{3.23}$$

The fluid impulse is

$$\mathbf{I}_{pi} = \int_0^t \sigma(p^- - p^+)\mathbf{n} ds, \tag{3.24}$$

where σ is the mass density of canopy per unity area, p^- and p^+ are the pressure on the lower side and upper side of the canopy, and n is the unit normal vector pointing from lower to upper side of the canopy. This follows [25] for its simplicity. A more accurate calculation should also include the

velocity shear near the canopy surface and stress to the surface.

For the calculation of the reacting impulse between the canopy and the fluid, we apply the immersed boundary method which treats the reaction of the canopy as the normal force approximated by the smoothed delta function, that is Peskin's delta function,

$$\mathbf{f}(\mathbf{x},t) = \int \mathbf{F}(\mathbf{x},t)\delta(x - \mathbf{X}(s,t))ds. \tag{3.25}$$

The difference is that we use the impulse of the mass point as a result of the superposition of three forces from the spring system,

$$\mathbf{F}(\mathbf{x}_i,t) = d(\mathbf{I}_g + \mathbf{I}_p + \mathbf{I}_s)/dt. \tag{3.26}$$

Numerically, the surface force is the product of the normal component of the acceleration and mass density at the canopy surface

$$\mathbf{F}(s,t) = \rho_c(\mathbf{a}\cdot\mathbf{n})\mathbf{n}, \tag{3.27}$$

where $\mathbf{n}$ is the unit normal vector on the canopy surface, and ρ_c is the mass density of canopy per unit area.

3.3.2 Local Index Coating

Our simulation of the parachute system is performed on the *FronTier++* platform, which uses the front tracking method. This method is widely used for tracing the fluid interface in multi-phase flow problems, such as the Rayleigh-Taylor

instability [18, 16], RichtmyerMeshkov instability [13], and the jet problems [19, 17]. In problems like these, the fluid interface is topologically a manifold, the two sides of each surface are the boundaries of two separate domains. However, it is not the case in the simulation of parachute system where the canopy surface is thinness and forms an open boundary. You cannot tell which side of the canopy surface it belongs when a point is far away from it. But, for points that are sufficiently close to the canopy surface, we can still distinguish which side it belongs.

The reason that we pay attention to this is that the local side information of space points that are near the canopy surface contributes to the calculation of the pressure difference, and thus the fluid force on the canopy. The pressure on two sides of the canopy surface is not continuous and it should be interpolated using the pressure information on its own side, which is realized by locally mesh coating algorithm. Assign the domain in which the canopy is immersed with index l, and the local index coating algorithm follows three steps:

- For any grid point P with a distance $d \leq h$ away from the canopy, where h is the grid spacing, find the nearest point on the canopy surface (*FronTier++* is well equipped with these geometry functions) P_s. Using the sign of the scalar product $\overline{PP_s} \cdot \mathbf{n}_s$, we can determine the side of the point. Reassign domain index to $l-1$ if the point P is on the negative side, otherwise reassign the index to $l+1$.

- Reassign any grid point adjacent to a point indexed $l-1$ to the same; reassign any grid point adjacent to a point indexed $l+1$ to the same.

Repeat this assignment for three sweeps. The resulting landscape of the grid index will look like Fig. 3.1.

- The interpolation of side-sensitive variables will use grid points of the same locally coated index.

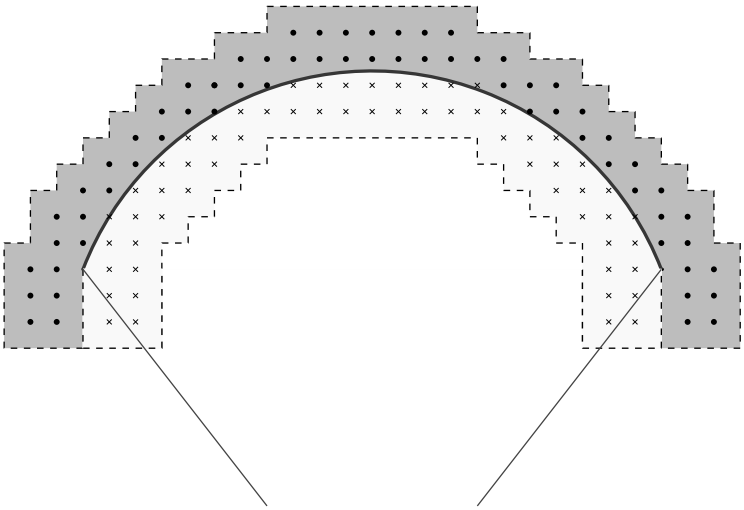


Figure 3.1: The parachute canopy is an open surface and cannot separate the space into sub-domains. But we can still coat different index for mesh cells close to the surface using the local geometrical information. The light and dark shaded polygons represent the sets of mesh cells on the positive and negative sides of the canopy, respectively. An interpolation is carried out on vertex of the same color.

3.4 Collision Features

Collision module for structures is a new feature added to the computational framework. With triangulized meshes, the proximity and collision

checks can be broken down into two major cases: a point against a triangle, an edge against another edge. For example, when there is collision detected between a pair of triangles, 15 tests need to be done in total.

- each point of one triangle against the other triangle
- each edge of one triangle against each edge of the other triangle

When strings are taken into account, the comparison pairs can be not only triangle-triangle, but also edge-edge and edge-triangle, because a string is represented by a group of connected edges. Similarly, there is 1 check for a edge-edge pair, and 5 checks for a edge-triangle pair.

When there are movable rigid bodies involved in collision, we make use of the impact zone technique which is first proposed in [32] and further developed in [4]. This technique was initially proposed to deal with multiple collisions and maintain collision consistency. The idea of the impact zone technique is to collect the points in multiple collisions into separated zones, and treat them as rigid bodies. Therefore, we can construct each movable rigid body as one independent impact zone before resolving collisions. Similar handling as cloth is applied to movable rigid bodies, but we adopted some differences.

- The impulse added to each mass point due to collision consists of two components: inelastic impulse and elastic impulse.
  - If the two structures that collid are the same type, i.e. two elastic structures or two rigid structures, the inelastic impulse is dis-

tributed according to the mass

$$I_{in,i} = m_1 m_2 v_n / (m_1 + m_2); \tag{3.28}$$

otherwise, it is distributed evenly

$$I_{in,i} = 0.5 m_i v_n. \tag{3.29}$$

In above, m_1 and m_2 are masses and v_n is the magnitude of relative velocity in the normal direction.

– When there is elastic structure(s) involved in the collision, the elastic impulse performs like a repulsion due to the elasticity

$$I_{el,i} = -min(\Delta t k d, m_i(0.1d/\Delta t - v_n)); \tag{3.30}$$

otherwise, it is the case of two rigid structures

$$I_{el,i} = cr I_{in,i}. \tag{3.31}$$

Here, d is the overlap length, k is the string stiffness and cr is the restitution coefficient.

- Whenever an impulse acts on some point on a movable rigid body, it is spread, via impact zone connection, to all the points on that rigid body. This is the constraint to maintain the geometric shape of the rigid bodies.

- The impulse due to cloth-involved collision and that due to rigid collision are recorded and averaged separately. Because there will be much more collision detected with cloth, while fewer between rigid bodies.

Chapter 4

Numerical Results

4.1 Rigid Body in Fluid Flow

Compressible Fluid

Incompressible Fluid

4.2 Parachute System Simulation

With our spring-mass model and the numerical methods mentioned above, we can simulate different types of parachute on *FronTier++* platform. Fig. 4.1 shows three types of parachute canopy after fully inflated due to the interaction from the inflow air.

4.2.1 Parachue Inflation

Parachute inflation process is one of the most crucial stages for the deceleration system. In Fig. 4.2, the cross-sectional velocity profile is demonstrated

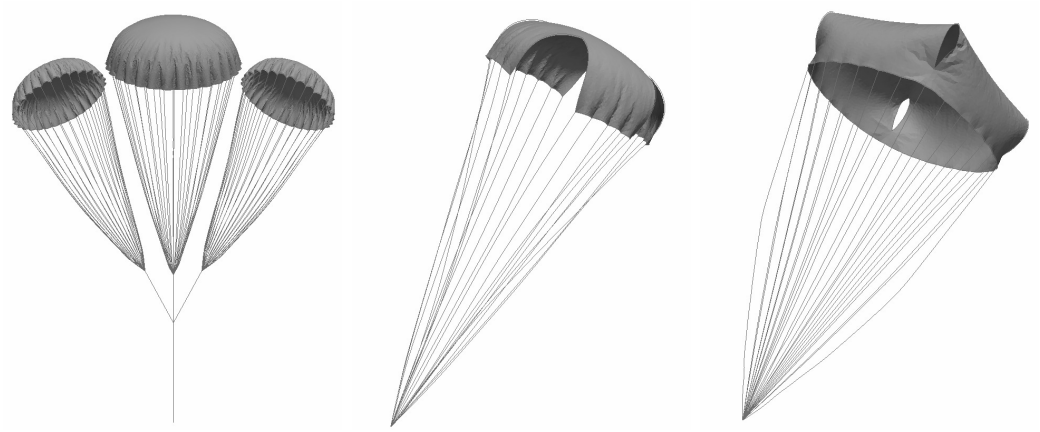


Figure 4.1: Three different types of parachute canopies simulated in *FronTier++* with our spring-mass model. Given parameters of a parachute, such as shape, size, number of chords, *FronTier++* has the capability to generate it.

at different stages. In this simulation, a C-9 personnel parachute with its nominal diameter $8.53m$ is taken as an example, and the computational domain is $16m \times 16m \times 36m$ with constant upward inflow velocity, outflow boundary condition at upper outlet with pressure $p = 0$ and periodic boundaries at the rest faces.

4.2.2 Wake of Parachutist

Coupled with spring model explained in the previous sub-sections, we demonstrate the effect caused by the weak of the parachutist taking the C-9 personnel parachute (without vent) with its nominal diameter $8.53m$ as an example, and set the computational domain to be $16m \times 16m \times 36m$ with constant upward inflow velocity, outflow boundary condition at upper outlet

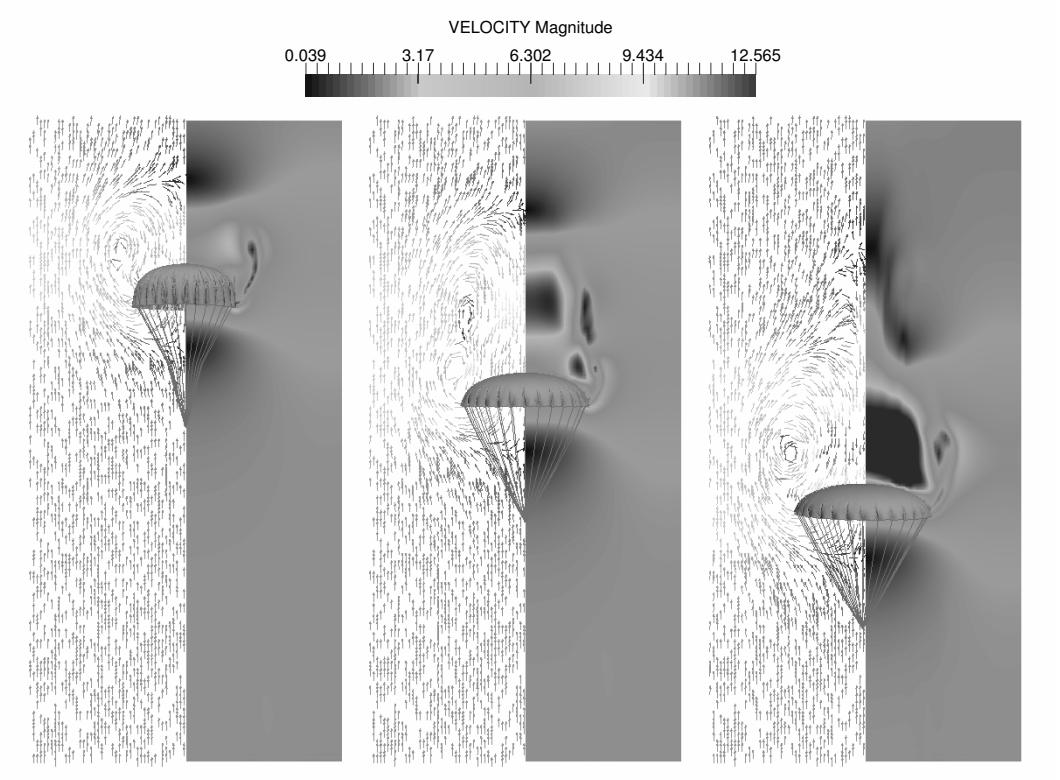


Figure 4.2: The cross-sectional velocity profiles around C-9 personnel parachute during the falling process, at time $t = 1.2s$, $t = 2.0s$ and $t = 3.2s$, respectively. In each plot, the left half shows the velocity direction by arrow and the right half shows the velocity magnitude by color, where red is large and blue is small.

with pressure $p = 0$ and periodic boundaries at the rest faces.

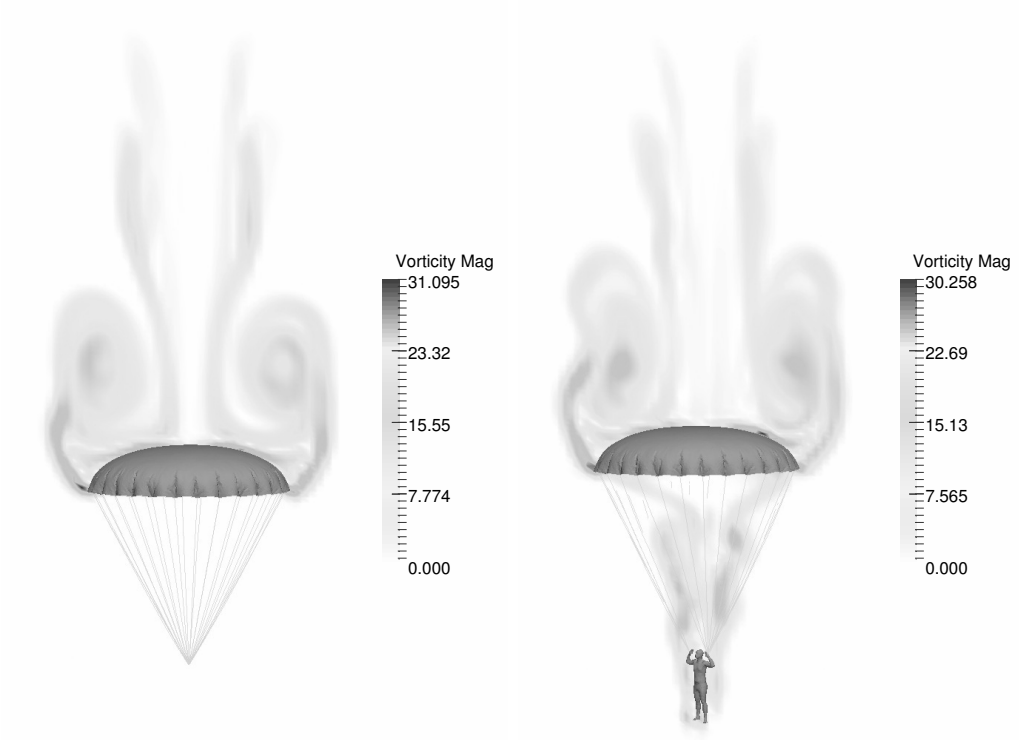


Figure 4.3: Comparison of the cross-sectional vorticity magnitude field. In the left figure, the parachutist is a mass point, while in the right figure, it is a rigid body with finite volume.

Observed from Fig. 4.3, there is turbulent flow as expected when the fluid flow passes the parachutist. Besides, the descent speed of the parachutist is slightly smaller when it is considered as a rigid body with finite volume. This is due to the interaction between the parachutist and the fluid flow, which generate an upward external force on it. Meanwhile, when the turbulent flow generated from fluid passing the parachutist reaches the canopy surface, it also has an upward force on the parachute system, and thus, reduces the descent

speed.

4.2.3 Uneven/Breaking Strings

When designing a parachute, it is undesirable to have uneven suspension lines or have some suspension lines broken at some time. Here, we demonstrate these two cases in a wind tunnle test on a C-9 personnel parachute with the computational domain size $12m \times 12m \times 24m$. The boundary conditions are set to be the same as above. In the left plot of Fig. 4.4, 10 of the suspension lines

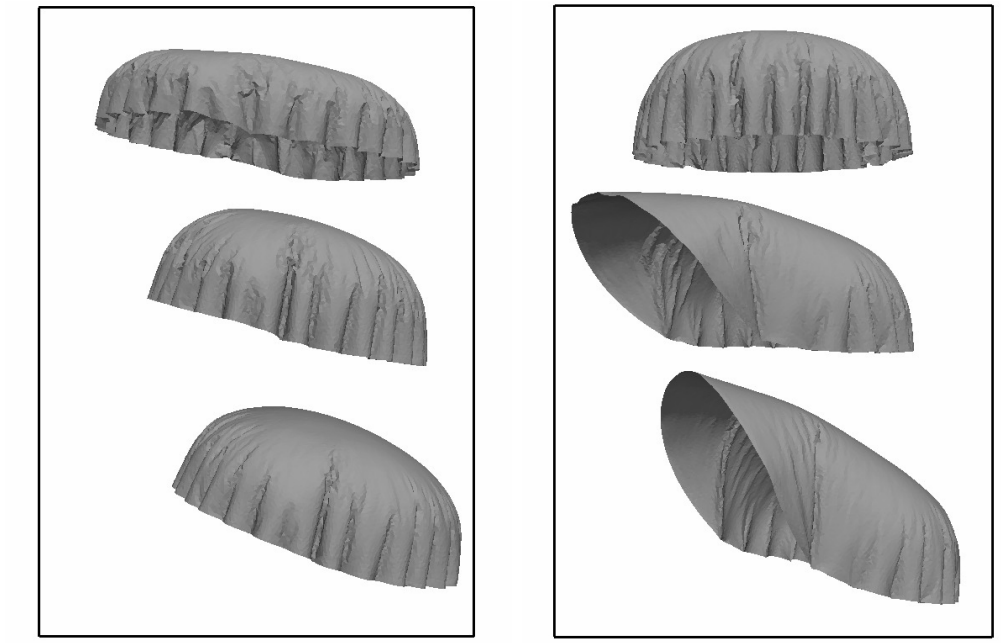


Figure 4.4: The canopy surface of the C-9 personnel parachute at different time. Left: 10 suspension lines are selected to have an equilibrium length than others. Right: 10 suspension lines are selected to break at time $t = 5.0s$.

have an equibluim length $0.3m$ shorter than normal which is $7m$; while in the

right plot of Fig. 4.4, 10 selected suspension lines break at $t = 5.0s$. Easy to observe that both cases cause the parachute canopy surface to be unstable and lead to an angled deployment. If the angle is too large, it could be dangerous.

4.3 Collision Simulations

As mentioned in the previous chapters, the algorithm is extended to work for rigid bodies as well. Since it is difficult to evaluate the correctness of cloth collision, we pay more attention to the robustness of the algorithm.

4.3.1 Benchmark: Rigid Bodies

Because one-dimensional collisions between rigid bodies can be analytically solved from the macro-aspect and have closed formula for velocities, we can use this test case to demonstrate the correctness of the algorithm. The closed formula is given by

$$\begin{aligned} \widetilde{v}_a &= \frac{C_R m_b(v_b - v_a) + m_a v_a + m_b v_b}{m_a + m_b} \\ \widetilde{v}_b &= \frac{C_R m_a(v_a - v_b) + m_a v_a + m_b v_b}{m_a + m_b} \end{aligned} \tag{4.1}$$

where C_R is the coefficient of restitution. It is 1 for a complete elastic collision, and 0 for complete inelastic collision.

Consider two cuboids initially moving toward each with velocity $0.5m/s$ and -$0.5m/s$ in x direction, and the masses are $1kg$ and $2kg$, respectively.

Using formula Eq. (4.1), the speed after collision is

$$\begin{aligned} \widetilde{v}_a &= -\frac{1}{6}m/s,\ \widetilde{v}_b = -\frac{1}{6}m/s,\ \text{with } C_R = 0; \\ \widetilde{v}_a &= -\frac{5}{6}m/s,\ \widetilde{v}_b = \frac{1}{6}m/s,\ \ \text{with } C_R = 1. \end{aligned} \tag{4.2}$$

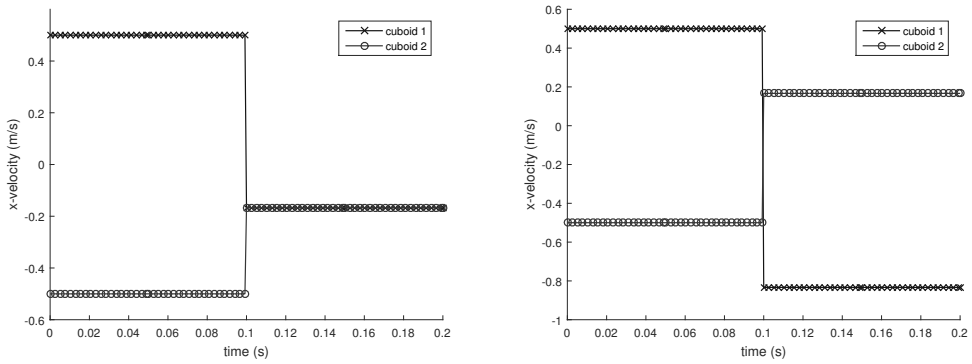


Figure 4.5: The velocity profile of the two cuboids. The left figure is the complete inelastic case ($C_R = 0$), and the right figure is the complete elastic case ($C_R = 1$).

Fig. 4.5 demonstrates the numerical profiles of velocity in two extreme case: $C_R = 0$ and $C_R = 1$. Clearly, it is consistent with the analytical values. Taking complete inelastic case for example, three states of the process are shown in Fig. 4.6.

4.3.2 Case 1: Cloth and Strings

The first example involves the collision between elastic cloth and elastic strings. Initially, a flat circular cloth is right above a bunch of strings which are straight and parallel, with their ends fixed. Due to the gravity, cloth

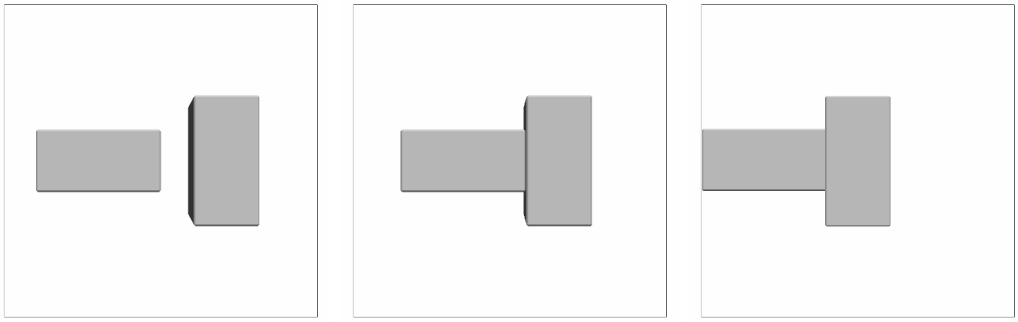


Figure 4.6: Complete inelastic collision. The left figure is the initial state ($t = 0s$) and two cuboids are moving toward each other; the middle figure is when they collide ($t = 0.1s$), after which both move to the left with the same velocity; the right figure is the final state ($t = 1.0s$).

and strings fall and collide with each. In this example, cloth-string collision happens first. Then, when the cloth is folded, the strings are clustered and string-string collision occurs. Finally, two half parts of the cloth meet below the strings, self-collision of cloth appears. Fig. 4.7 demonstrates this process, and we can observe, all the three kinds of collision are handled properly.

4.3.3 Case 2: Rigid Body and Cloth

Here, we exhibit two examples demonstrating the collision between rigid bodies and cloth. In Fig. 4.8, a circular fabric falls from some height above the static cuboid object. The bending force is taken into account in the way mentioned above. It highly reduces the number of collision pairs and forms the billows at four corners.

Fig. 4.9 is the falling process of two balls from some height onto cloth

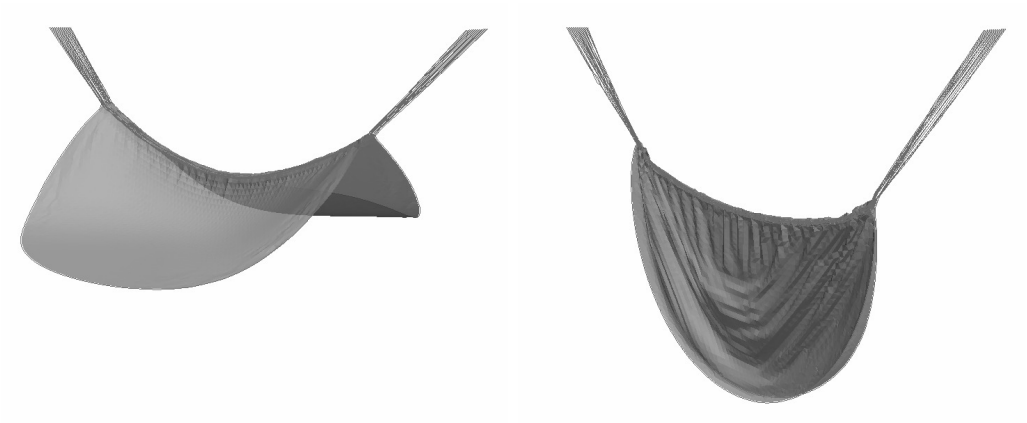


Figure 4.7: Two stages in the process of a circular cloth falling onto a bunch of elastic strings. In the left figure, there are collisions between cloth and strings. While in the right figure, the are also collisions between the strings and self-collision of cloth.

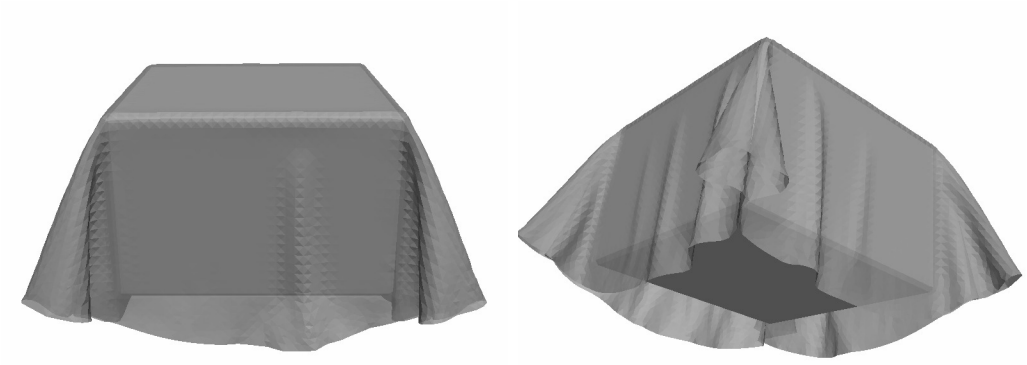


Figure 4.8: The front view and the side view of the cloth falling onto static objects. Benefitted from the bending effect mentioned at the end of the spring-mass model, billows appear at the courners.

with its left and right sides fixed. In the early stage, two balls collide with the cloth, respectively, and only rigid-elastic collision occurs. Due to the bowl

shape of the cloth, two balls are pushed towards each other and collide at some time. Since the collision between two balls is set to be completely inelastic, they keep contacted and collide with the cloth together in the later stage.

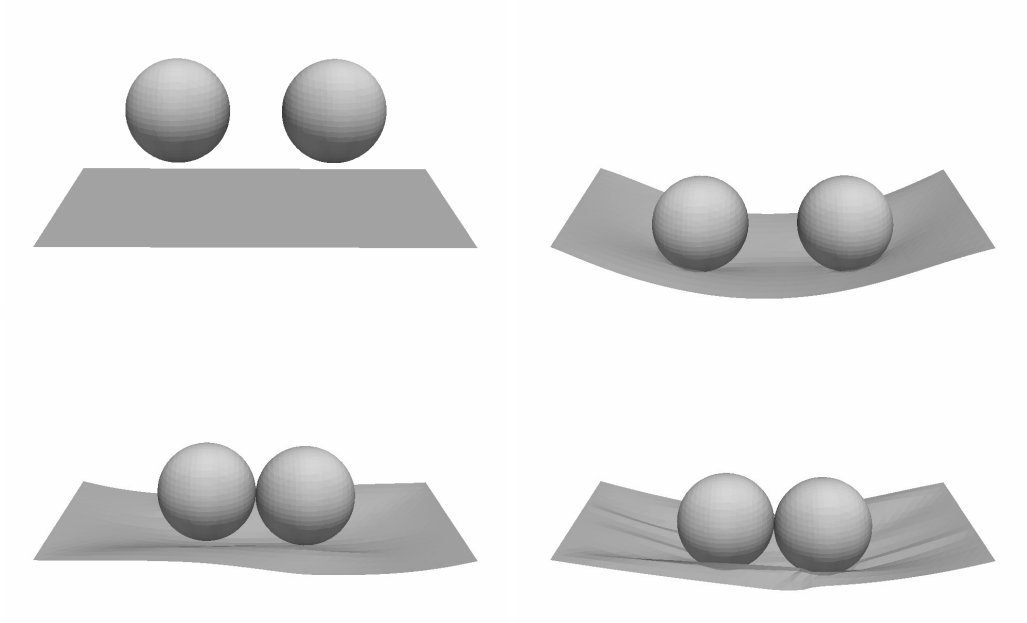


Figure 4.9: Two balls falling on to cloth whose two sides are fixed.

Chapter 5

Parallelization

ccuracy and robustness are the fundamental goals in computational simulation of physics problems, which can be achieved by adopting high order numerical schemes and advanced algorithms. Besides, the computational efficiency becomes more and more important with the development of the advancing computational technology. This requires intelligent and careful implementation of the numerical schemes and algorithms.

In our *FronTier++* platform, we make use of the hardware and implement CPU parallelization code by MPI and GPU parallelization code with CUDA. The computational platform consists of one head node, 21 computing nodes (20 CPU (Central Processing Unit) nodes and 1 GPU (Graphic Processing Unit) node), connected with $56Gb/s$ InfiniBand and $1000MB/s$ Ethernet. Each node was populated with dual Eight-Core Intel $E5-2630v3$ "Haswell" $2.4GHz$ processors with different size of RAM (Random-access Memory) and Storage. The head node and compute node have $32GB$ of RAM for each, and the GPU node has $128GB$ of RAM. A $32TB$ of network file system using

Node Type	CPU	RAM	Storage for data	Storage for OS	GPU
Head node	2x8-core Intel E5-2630v3	32GB	32TB (RAID 6)	2TB (SSD)	none
CPU node	2x8-core Intel E5-2630v3	32GB	none	1TB (SATA)	none
GPU node	2x8-core Intel E5-2630v3	128GB	none	240GB (SSD)	7 Tesla K40

Table 5.1: Summary of the hardware

RAID6 (Redundant Array of Independent Disks) is installed in the head node, and is shared with other nodes. Each node also has a clone of the operation system on its local disk: $2TB$ SSD on head node, $1TB$ SATA drive on compute nodes, and $240GB$ SSD on GPU node. The parallel computing can be further accelerated by including the GPU node, which contains seven NVIDIA Tesla K40 GPUs with $12GB$ of RAM for each device. A detailed description of the hardware is summarized in Table 5.1.

Fig. 5.1 is the complete flow chart of the algorithm. Upon testing, we identified that the fluid solver (the panel marked as red) is the most time consuming part. Beside, the spring model solver (the panel marked as blue) also takes significant percent of the computational time, especially when the number of mass points is large.

Graphics Processing Unit (GPU) compution [28] is to use the GPUs together with CPUs to accelerate a general-pourpose scientific and engineering application. For the blue panel, we make use of the GPU technique to accelerate the numerical calculation, i.e. the part inside the green dash-dotted line in Fig. 5.1. Solving the spring model by 4-th order Runge-Kutta method consists the following steps:

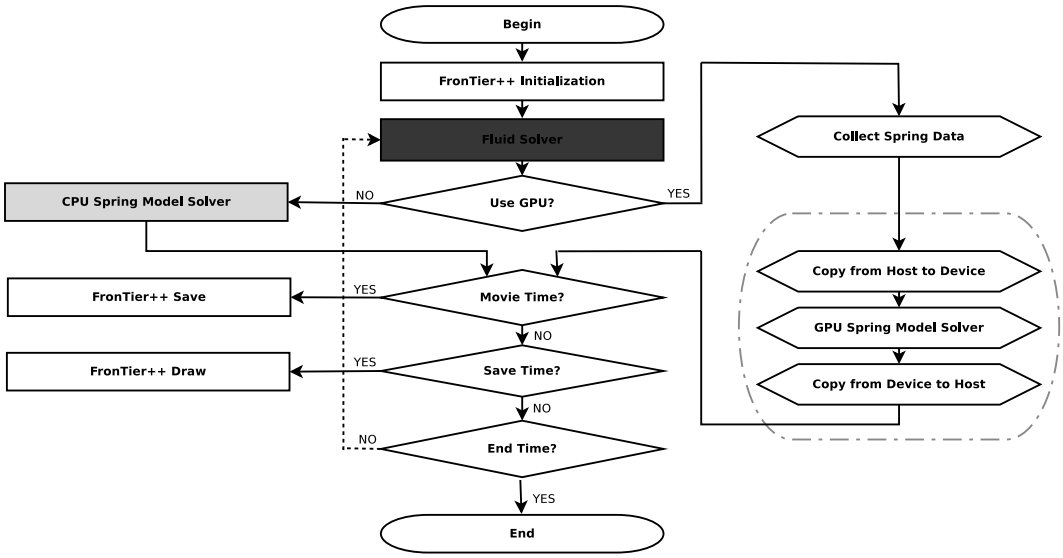


Figure 5.1: Flow chart of the complete simulation algorithm that involves both fluid dynamics and elastic structure dynamics. The part inside the green dash-dotted line is the work involves done by GPU.

1. Find the positions of all the neighbors of each spring vertex.

2. Calculate the total force on each spring vertex using Delingette method.

3. Calculate the acceleration of each vertex and update the location of them.

4. Go to 2 if this is not the end of the Runge-Kutta; finish this time step, otherwise.

To accelerate the calculation of this part, we then shift it to the GPU cores for massively parallel processing. When we apply the GPU code to the complete parachute system, including both the parachute canopy and all the suspension

Parachute type	CPU/GPU	Time(s)	Avg time per step(s)	Speedup
C-9	CPU	2805.85	3.39	1.00
	GPU	131.90	0.16	21.2
G-11	CPU	5101.47	5.41	1.00
	GPU	243.18	0.26	20.81
Intruder	CPU	1252.65	2.00	1.00
	GPU	69.67	0.11	18.18
T-10	CPU	5540.02	5.99	1.00
	GPU	282.74	0.36	16.64
T-11	CPU	6791.9	5.12	1.00
	GPU	352.07	0.29	17.66

Table 5.2: A comparison of computational time between different parachute types using CPU and GPU code. The speedup is calculated based on the computing time by CPU.

lines, we can achieve 16-21 $\times$ faster then pure CPU code for different types of parachutes, as shown in Table 5.2.

The solution for the red panel is to use multiple CPU cores; each CPU is in charge of the calculation of one part of the computational domain. The fluid computational domain is partitioned according to the computational mesh grid, however, it is not the case for the canopy surface. Intuitively, cutting the canopy surface through its edges is not reasonable in the spring-mass model. Therefore, a modified partitioning algorithm is applied which cut the canopy surface according to the particles on it, as demonstrated in Fig. 5.2 with 2×2

partitioning.

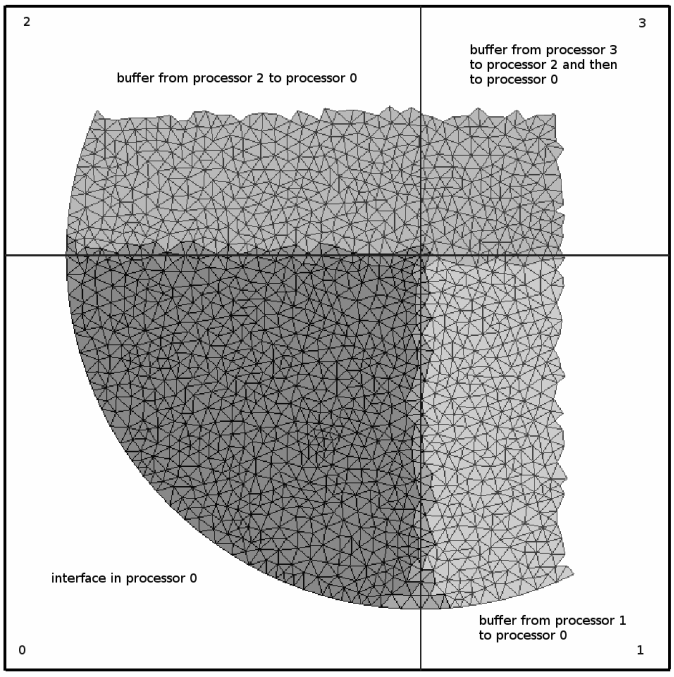


Figure 5.2: A demonstration of the clipped interface in processors 0 and its buffer interfaces from neighbor processors.

The lower left part is the interface in processor 0, it received buffer interface from two neighbor processors, 1 (lower right) and 2 (upper left). The buffer interface provides neighbors to spring vertices that are close to the sub-domain boundary. Since the communication of buffer interface is done dimension by dimension, the buffer interface from processor 2 contains part of the buffer interface from processor 3 to processor 2. With this partitioning algorithm, the spring-mass model work well with multiple processors.

Furthermore, two techniques can be coupled together in the simulation of parachute system.

1. Collect mass point information of the fabric structure from each processor to the main processor.

2. Call the GPU-based spring model solver in the main processor.

3. Distrubute the updated information of each mass point to its corresponding processor.

Chapter 6

Conclusions

Bibliography

[1] R. M. Aileni, D. Farima, and M. Ciocoiu. Simulating cloth in realistic way using vertices model based. *7th International Conference-TEXSCI*, 2010.

[2] David Baraff and Andrew Witkin. Large steps in cloth simulation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 43–54. ACM, 1998.

[3] David Baraff, Andrew Witkin, and Michael Kass. Untangling cloth. In *ACM Transactions on Graphics (TOG)*, volume 22, pages 862–870. ACM, 2003.

[4] Robert Bridson, Ronald Fedkiw, and John Anderson. Robust treatment of collisions, contact and friction for cloth animation. *ACM Trans. Graph.*, 21(3):594–603, July 2002.

[5] David L Brown, Ricardo Cortez, and Michael L Minion. Accurate projection methods for the incompressible navier-stokes equations. *Journal of Computational Physics*, 168(2):464–499, 2001.

[6] Michel Carignan, Ying Yang, Nadia Magnenat Thalmann, and Daniel Thalmann. Dressing animated synthetic actors with complex deformable clothes. In *ACM Siggraph Computer Graphics*, volume 26, pages 99–104. ACM, 1992.

[7] K.-J. Choi and H.-S. Ko. Stable but responsive cloth. *ACM Transactions on Graphics*, 21:604–611, 2002.

[8] A. J. Chorin. Numerical solution of the Navier Stokes equations. *Math. Comp*, 22:745–762, 1968.

[9] A. J. Chorin. On the convergence of discrete approximations to the Navier-Stokes equations. *Math. Comp*, 23:341, 1969.

[10] Herve Delingette. Triangular springs for modeling nonlinear membranes. *IEEE Transactions on Visualization and Computer Graphics Volume 14 Issue 2*, pages 723–731, March 2008.

[11] Jian Du, Brian Fix, James Glimm, Xicheng Jia, Xiaolin Li, Yunhua Li, and Lingling Wu. A simple package for front tracking. *J. Comput. Phys.*, 213:613–628, 2006.

[12] John A. Ekaterinaris. High-order accurate, low numerical diffusion methods for aerodynamics. *Progress in Aerospace Sciences*, 41(34):192 – 300, 2005.

[13] C. L. Gardner, J. Glimm, J. W. Grove, O. McBryan, R. Menikoff, D. H. Sharp, and Q. Zhang. A study of chaos and mixing in Rayleigh-Taylor and Richtmyer-Meshkov unstable interfaces. In M. Duong-van and B. Nichols, editors, *Proceedings of the International Conference on 'The Physics of Chaos and Systems Far from Equilibrium' (CHAOS' 87), Monterey, CA, USA, Jan. 11–14, 1987*. 1988. Special Issue of Nuclear Physics B (proceedings supplements section).

[14] James M. Gere. *Mechanics of materials, sixth edition.* Bill Stenquist, 2004.

[15] J. Glimm, J. W. Grove, X.-L. Li, and D. C. Tan. Robust computational algorithms for dynamic interface tracking in three dimensions. *SIAM J. Sci. Comp.*, 21:2240–2256, 2000.

[16] J. Glimm, X.-L. Li, R. Menikoff, D. H. Sharp, and Q. Zhang. A numerical study of bubble interactions in Rayleigh-Taylor instability for compressible fluids. *Phys. Fluids A*, 2(11):2046–2054, 1990.

[17] J. Glimm, X. L. Li, W. Oh, A. Marchese, M.-N. Kim, R. Samulyak, and C. Tzanos. Jet breakup and spray formation in a diesel engine. In *Proceedings of the Second MIT Conference on Computational Fluid and Solid Mechanics*. Cambridge, MA, 2003. SUNY Stony Brook preprint No. susb-ams-02-20.

[18] J. Glimm, O. McBryan, R. Menikoff, and D. Sharp. Front tracking applied to Rayleigh-Taylor instability. *SIAM J. Sci. Stat. Comput.*, 7:230–251, 1986.

[19] James Glimm, M.-N. Kim, X.-L. Li, R. Samulyak, and Z.-L. Xu. Jet simulation in a diesel engine. In *Computational Fluid and Solid Mechanics 2005*. Elsevier Science, 2005.

[20] Oscar Gonzalez and Andrew M. Stuart. *A First Course in Continuum Mechanics*. Cambridge University Press, 2008.

[21] John Safko Herbert Goldstein, Charles Poole. *Classical Mechanics, Third Edition*. Addison-Wesley, 2001.

[22] S.-W. Hsiao and R.-Q. Chen. A method of drawing cloth patterns with fabric behavior. *5th WSEAS International Conference on Applied Computer Science*, pages 635–640, 2006.

[23] F. Ji, R. Li, and Y. Qiu. Simulate the dynamic draping behavior of woven and knitted fabrics. *J. Industrial Textiles*, 35:201–214, 2006.

[24] Guang-Shan Jiang and Chi-Wang Shu. Efficient implementation of weighted ENO schemes. *Journal of Computational Physics*, 126(1):202 – 228, 1996.

[25] Vinay Kalro and Tayfun E. Tezduyar. A parallel 3D computational method for fluid-structure interactions in parachute systems. *Computer Methods in Applied Mechanics and Engineering*, 190:321–332, 2000.

[26] J. Kim and P. Moin. Application of a fractional-step method to incompressible Navier-Stokes equations. *J. Comput. Phys.*, 59:308, 1985.

[27] J.-D. Kim, Y. Li, and X.-L. Li. Simulation of parachute FSI using the front tracking method. *Journal of Fluids and Structures*, 37:101–119, 2013.

[28] David B Kirk and W Hwu Wen-mei. *Programming massively parallel processors: a hands-on approach*. Morgan Kaufmann, 2010.

[29] Y. Li, I-Liang Chern, J.-D. Kim, and X.-L. Li. Numerical method of fabric dynamics using front tracking and spring model. *Communications in Computational Physics*, 14(5):1228–1251, 2013.

[30] C. C. Lin and L. A. Segel. *Mathematics Applied to Deterministic Problems in the Natural Sciences*. Society for Industrial and Applied Mathematics, 1988.

[31] Xu-Dong Liu, Stanley Osher, and Tony Chan. Weighted essentially non-oscillatory schemes. *Journal of Computational Physics*, 115(1):200 – 212, 1994.

[32] Xavier Provot. *Collision and self-collision handling in cloth model dedicated to design garments*, pages 177–189. Springer Vienna, Vienna, 1997.

[33] Qiangqiang Shi, Daniel Reasor, Zheng Gao, Xiaolin Li, and Richard D Charles. On the verification and validation of a spring fabric for modeling parachute inflation. *Journal of Fluids and Structures*, 58:20–39, oct 2015.

[34] Demetri Terzopoulos. On deformable models. In *Geometric Analysis and Computer Graphics*, pages 181–192. Springer, 1991.

[35] Demetri Terzopoulos and Kurt Fleischer. Modeling inelastic deformation: viscolelasticity, plasticity, fracture. In *ACM Siggraph Computer Graphics*, volume 22, pages 269–278. ACM, 1988.

[36] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. In *ACM Siggraph Computer Graphics*, volume 21, pages 205–214. ACM, 1987.

[37] Pascal Volino, Martin Courchesne, and Nadia 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*, pages 137–144. ACM, 1995.

[38] Pascal Volino, N Magnenat Thalmann, Shen Jianhua, and Daniel Thalmann. An evolving system for simulating clothes on virtual actors. *IEEE Computer Graphics and Applications*, 16(5):42–51, 1996.

[39] Pascal Volino and Nadia Magnenat Thalmann. Collision and self-collision detection: Efficient and robust solutions for highly deformable surfaces. In *Computer Animation and Simulation95*, pages 55–65. Springer, 1995.