A Numerical Simulation for Chaotic Dynamics of Pendulum

Jiaxuan Li[1*], Qingchuan Ma[1], Ziji Xu[1], Tianye Lv[1]

[1]Department of Astronomy, School of Physics
Peking University, Beijing, 100871, China

January 16, 2017

Part I
Introduction

We encountered the term chaos in our Calculus class this semester. When professor talked about the chaotic nature of our world, we were all shocked by this fact. From Newton's laws to Maxwell's Equations, we are impressed by the simplicity of physics laws. It may be a stereotype that physics is simple and beautiful. But when we know nonlinear appearance of physical laws and complexity of motion even for very simple physical model, then a new horizon rise.

Chaos is a new subject in physics and mathematics. Edward Lorenz, an American mathematician and meteorologist, is the pioneer of chaos theory. Strange attractor, and the popular term butterfly effect were introduced by him. The most outstanding scientist who studied dynamical systems and chaos is Henry Poincaré. Pendulum has been researched for over 400 years, but its mystery hasn't been discovered completely even for today. The first man who investigated the behavior of pendulum is Dutch scientist Huygens. In a damped pendulum with driven force, complexity of motion and chaotic behavior can be found, although this model is quite simple. Not only does chaos play an important role in mathematics and physics, it is also the basic concepts in economics and finance. Chaos is here and there in our daily life.

Several years ago, one team member saw a video of double pendulum, which is moving chaotically. At that time, the seed of chaos was planted into his heart. The seed of chaos now sprout. As freshmen, we are quite interested in chaos and dynamical systems. We hope this project can be a door to our physics and mathematics road.

Part II
Dynamics of Pendulum

1 Equations of Motion for Planar Double Pendulum

1.1 The mass is centered in the bobs

In this section, we are going to derive the equation of motion of a double pendulum system, which is swinging in a plane with the effect of gravity.

Consider two bobs m_1 and m_2 connected by two rigid light bars, the first of which is allowed to rotate about a fixed horizontal axis A_1, the second coupled to the first along the axis A_2, while A_2 is parallel to A_1. We are not going to consider friction throughout our research. The configuration of this system is determined by θ_1, θ_2, l_1 and l_2, as shown in Figure 1.

*Corresponding Author:jiaxuan_li@pku.edu.cn

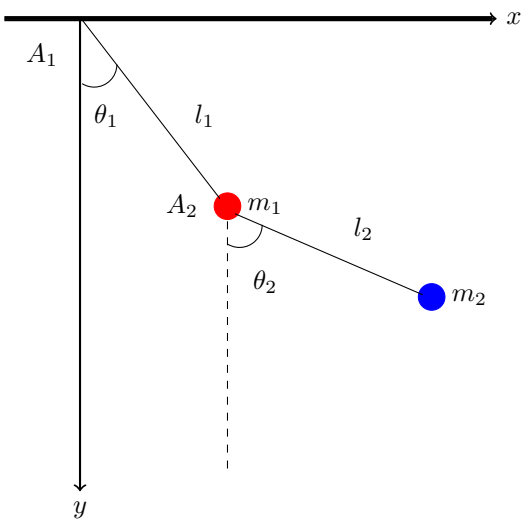

Figure 1: Double Pendulum (Mathematical)

In order to describe the position of each bob more easily, we constructed a rectangular coordinates xOy. Then bob m_1 and m_2 can be described by x_1, y_1, x_2, y_2, and their velocities which are the time derivative of their position, are denoted by $\dot{x}_1$, $\dot{y}_1$, $\dot{x}_2$, $\dot{y}_2$.

$$\begin{cases} x_1 = l_1 \sin\theta_1 \\ y_1 = l_1 \cos\theta_1 \\ \dot{x}_1 = l_1\dot{\theta}_1 \cos\theta_1 \\ \dot{y}_1 = -l_1\dot{\theta}_1 \sin\theta_1 \end{cases} \tag{1}$$

$$\begin{cases} x_2 = l_1 \sin\theta_1 + l_2 \sin\theta_2 \\ y_2 = l_1 \cos\theta_1 + l_2 \cos\theta_2 \\ \dot{x}_2 = l_1\dot{\theta}_1 \cos\theta_1 + l_2\dot{\theta}_2 \cos\theta_2 \\ \dot{y}_2 = -l_1\dot{\theta}_1 \sin\theta_1 - l_2\dot{\theta}_2 \sin\theta_2 \end{cases} \tag{2}$$

Kinetic energy of each bob:

$$T_1 = \frac{1}{2}m_1(\dot{x}_1^2 + \dot{y}_1^2) = \frac{1}{2}m_1 l_1^2\dot{\theta}_1^2$$

$$T_2 = \frac{1}{2}m_2(\dot{x}_2^2 + \dot{y}_2^2) = \frac{1}{2}m_2[l_1^2\dot{\theta}_1^2 + l_2^2\dot{\theta}_2^2 + 2l_1l_2\dot{\theta}_1\dot{\theta}_2\cos(\theta_1 - \theta_2)]$$

Potential energy of the system:

$$V = -m_1 g l_1 \cos\theta_1 - m_2 g(l_1\cos\theta_1 + l_2\cos\theta_2)$$

Lagrangian of the double pendulum system is:

$$\mathscr{L} = T - V = \frac{1}{2}m_1 l_1^2\dot{\theta}_1^2 + \frac{1}{2}m_2(l_1^2\dot{\theta}_1^2 + l_2^2\dot{\theta}_2^2 + 2l_1l_2\dot{\theta}_1\dot{\theta}_2\cos(\theta_1 - \theta_2)) + (m_1 + m_2)gl_1\cos\theta_1 + m_2 g l_2\cos\theta_2$$

From theoritical mechanics[1], we know the motion of a system can be determined using Lagrange Functions:

$$\frac{d}{dt}\left(\frac{\partial\mathscr{L}}{\partial\dot{\theta}_1}\right) - \frac{\partial\mathscr{L}}{\partial\theta_1} = 0$$

$$\frac{d}{dt}\left(\frac{\partial\mathscr{L}}{\partial\dot{\theta}_2}\right) - \frac{\partial\mathscr{L}}{\partial\theta_2} = 0$$

Then we calculate each partial derivative in these two Lagrange Functions:

$$\frac{\partial\mathscr{L}}{\partial\dot{\theta}_1} = (m_1 + m_2)\dot{\theta}_1 l_1^2 + m_2 l_1 l_2\dot{\theta}_2\cos(\theta_1 - \theta_2)$$

$$\frac{\partial \mathscr{L}}{\partial \theta_1} = -m_2 l_1 l_2 \dot{\theta}_1 \dot{\theta}_2 \sin(\theta_1 - \theta_2) - (m_1 + m_2) g l_1 \sin\theta_1$$

$$\frac{\partial \mathscr{L}}{\partial \dot{\theta}_2} = m_2 \dot{\theta}_2 l_2^2 + m_2 l_1 l_2 \dot{\theta}_1 \cos(\theta_1 - \theta_2)$$

$$\frac{\partial \mathscr{L}}{\partial \theta_2} = m_2 l_1 l_2 \dot{\theta}_1 \dot{\theta}_2 \sin(\theta_1 - \theta_2) - m_2 g l_2 \sin\theta_2$$

After substituting each derivative by concrete expressions, we can get two ordinary differential equations:

$$\begin{cases} (m_1 + m_2) l_1 \ddot{\theta}_1 + m_2 l_2 \cos(\theta_1 - \theta_2) \ddot{\theta}_2 + m_2 l_2 \sin(\theta_1 - \theta_2) \dot{\theta}_2^2 + (m_1 + m_2) g \sin\theta_1 = 0 \\ l_1 \cos(\theta_1 - \theta_2) \ddot{\theta}_1 + l_2 \ddot{\theta}_2 - l_1 \sin(\theta_1 - \theta_2) \dot{\theta}_1^2 + g \sin\theta_2 = 0 \end{cases} \tag{3}$$

After simplifying(actually solving linear equations of two arguments), we get:

$$\begin{cases} \ddot{\theta}_1 = \dfrac{m_2 g \sin\theta_2 \cos(\theta_1 - \theta_2) - m_2 \sin(\theta_1 - \theta_2)[l_2 \dot{\theta}_2^2 + l_1 \dot{\theta}_1^2 \cos(\theta_1 - \theta_2)]}{l_1[(m_1 + m_2) - m_2 \cos^2(\theta_1 - \theta_2)]} \\ \qquad - \dfrac{(m_1 + m_2) g \sin\theta_1}{l_1[(m_1 + m_2) - m_2 \cos^2(\theta_1 - \theta_2)]} \\ \ddot{\theta}_2 = \dfrac{(m_1 + m_2) l_1 \sin(\theta_1 - \theta_2) \dot{\theta}_1^2 + m_2 l_2 \dot{\theta}_2^2 \sin(\theta_1 - \theta_2) \cos(\theta_1 - \theta_2)}{} \\ \qquad - \dfrac{(m_1 + m_2) g [\sin\theta_2 - \sin\theta_1 \cos(\theta_1 - \theta_2)]}{[(m_1 + m_2) - m_2 \cos^2(\theta_1 - \theta_2)]} \end{cases} \tag{4}$$

Now we get the equation of motion for double pendulum.

1.2 The mass is distributed along the rod

Now we consider two rigid rods m_1 and m_2 with centers of mass C_1 and C_2, the first of which is allowed to rotate about a fixed horizontal axis A_1, the second coupled to the first along the axis A_2, while A_2 is parallel to A_1. For the sake of simplifying the problem, we assume that the mass is distributed homogenously along the rod, therefore the center of mass of each rod is at the midpoint. The configuration of this system is determined by θ_1, θ_2, l_1 and l_2, as shown in the figure.

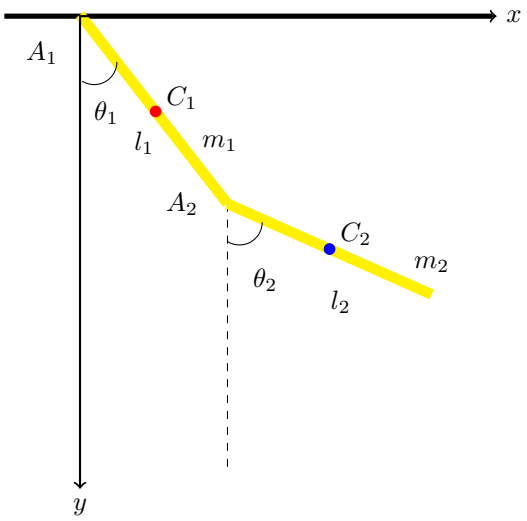

Figure 2: Double Pendulum (Physical)

We have: The kinetic energy of the stick equals to the kinetic energy of the center of mass plus the kinetic energy of the stick when you are in the center of mass reference frame, i.e.

$$T = T_{CM} + T_{CMframe}$$

For center of mass of m_1 :

$$\begin{cases} x_1 = \frac{1}{2} l_1 \sin\theta_1 \\ y_1 = \frac{1}{2} l_1 \cos\theta_1 \\ \dot{x}_1 = \frac{1}{2} l_1 \dot{\theta}_1 \cos\theta_1 \\ \dot{y}_1 = -\frac{1}{2} l_1 \dot{\theta}_1 \sin\theta_1 \end{cases} \tag{5}$$

For center of mass of m_2 :

$$\begin{cases} x_2 = l_1 \sin\theta_1 + \frac{1}{2} l_2 \sin\theta_2 \\ y_2 = l_1 \cos\theta_1 + \frac{1}{2} l_2 \cos\theta_2 \\ \dot{x}_2 = l_1 \dot{\theta}_1 \cos\theta_1 + \frac{1}{2} l_2 \dot{\theta}_2 \cos\theta_2 \\ \dot{y}_2 = -l_1 \dot{\theta}_1 \sin\theta_1 - \frac{1}{2} l_2 \dot{\theta}_2 \sin\theta_2 \end{cases} \tag{6}$$

The moment of inertia of the rod respect to the axis which is through its midpoint is:

$$I = \frac{1}{12} m R^2$$

Kinetic energy two rod:

$$T_1 = \frac{1}{6} m_1 l_1^2 \dot{\theta}_1^2$$

$$T_2 = \frac{1}{2} m_2 l_1^2 \dot{\theta}_1^2 + \frac{1}{6} m_2 l_2^2 \dot{\theta}_2^2 + \frac{1}{2} m_2 l_1 l_2 \dot{\theta}_1 \dot{\theta}_2 \cos(\theta_1 - \theta_2)$$

The potential energy of the system:

$$V = -\frac{1}{2} m_1 g l_1 \cos\theta_1 - m_2 g (l_1 \cos\theta_1 + \frac{1}{2} l_2 \cos\theta_2)$$

Lagrangian of the double pendulum system is:

$$\mathscr{L} = T - V = \frac{1}{6}(m_1 + 3m_2) l_1^2 \dot{\theta}_1^2 + \frac{1}{6} m_2 l_1^2 \dot{\theta}_2^2 + \frac{1}{2} m_2 l_1 l_2 \dot{\theta}_1 \dot{\theta}_2 \cos(\theta_1 - \theta_2) + \frac{1}{2} m_1 g l_1 \cos\theta_1 + m_2 g (l_1 \cos\theta_1 + \frac{1}{2} l_2 \cos\theta_2)$$

$$\begin{cases} \frac{\partial \mathscr{L}}{\partial \dot{\theta}_1} = \frac{1}{3}(m_1 + 3m_2) \dot{\theta}_1 l_1^2 + \frac{1}{2} m_2 l_1 l_2 \dot{\theta}_2 \cos(\theta_1 - \theta_2) \\ \frac{\partial \mathscr{L}}{\partial \theta_1} = -\frac{1}{2} m_2 l_1 l_2 \dot{\theta}_1 \dot{\theta}_2 \sin(\theta_1 - \theta_2) - \frac{1}{2}(m_1 + 2m_2) g l_1 \sin\theta_1 \\ \frac{\partial \mathscr{L}}{\partial \dot{\theta}_2} = \frac{1}{3} m_2 l_2^2 \dot{\theta}_2 + \frac{1}{2} m_2 l_1 l_2 \dot{\theta}_1 \cos(\theta_1 - \theta_2) \\ \frac{\partial \mathscr{L}}{\partial \theta_2} = \frac{1}{2} m_2 l_1 l_2 \dot{\theta}_1 \dot{\theta}_2 \sin(\theta_1 - \theta_2) - \frac{1}{2} m_2 g l_2 \sin\theta_2 \end{cases} \tag{7}$$

After substituting each derivative by concrete expressions, we can get two ordinary differential equations:

$$\begin{cases} \frac{1}{3}(m_1 + 3m_2) l_1 \ddot{\theta}_1 + \frac{1}{2} m_2 l_2 \cos(\theta_1 - \theta_2) \ddot{\theta}_2 + \frac{1}{2} m_2 l_2 \dot{\theta}_2^2 \sin(\theta_1 - \theta_2) + \frac{1}{2}(m_1 + 2m_2) g \sin\theta_1 = 0 \\ \frac{1}{2} l_1 \cos(\theta_1 - \theta_2) \ddot{\theta}_1 + \frac{1}{3} l_2 \ddot{\theta}_2 - \frac{1}{2} l_1 \sin(\theta_1 - \theta_2) \dot{\theta}_1^2 + \frac{1}{2} g \sin\theta_2 = 0 \end{cases} \tag{8}$$

i.e.

$$\begin{cases} 2(m_1 + 3m_2) l_1 \ddot{\theta}_1 + 3m_2 l_2 \cos(\theta_1 - \theta_2) \ddot{\theta}_2 + 3m_2 l_2 \dot{\theta}_2^2 \sin(\theta_1 - \theta_2) + 3(m_1 + 2m_2) g \sin\theta_1 = 0 \\ 3 l_1 \cos(\theta_1 - \theta_2) \ddot{\theta}_1 + 2 l_2 \ddot{\theta}_2 - 3 l_1 \sin(\theta_1 - \theta_2) \dot{\theta}_1^2 + 3 g \sin\theta_2 = 0 \end{cases} \tag{9}$$

After simplifying(actually solving linear equations of two arguments), we get:

$$\begin{cases} \ddot{\theta}_1 = \dfrac{\sin(\theta_1-\theta_2)(9m_2l_1l_2\cos(\theta_1-\theta_2)\dot{\theta}_1^2+6m_2l_2^2\dot{\theta}_2^2)}{l_1l_2(9m_2\cos^2(\theta_1-\theta_2)-4(m_1+3m_2))} \\ \qquad + \dfrac{6(m_1+2m_2)gl_2\sin\theta_1-9m_2gl_2\sin\theta_2\cos(\theta_1-\theta_2)}{l_1l_2(9m_2\cos^2(\theta_1-\theta_2)-4(m_1+3m_2))} \\ \ddot{\theta}_2 = \dfrac{\sin(\theta_1-\theta_2)(6(m_1+3m_2)l_1^2\dot{\theta}_1^2+9m_2l_1l_2\cos(\theta_1-\theta_2)\dot{\theta}_2^2)}{l_1l_2(4(m_1+3m_2)-9m_2\cos^2(\theta_1-\theta_2))} \\ \qquad + \dfrac{9(m_1+2m_2)gl_1\sin\theta_1\cos(\theta_1-\theta_2)-6(m_1+3m_2)gl_1\sin\theta_2}{l_1l_2(4(m_1+3m_2)-9m_2\cos^2(\theta_1-\theta_2))} \end{cases} \tag{10}$$

2 The forces acted on axis for double pendulum

Here we discuss the forces acted on axis for double pendulum when mass is distributed on the bobs. For the first bob m_1:

$$\begin{cases} x_1 = l_1\sin\theta_1 \\ y_1 = l_1\cos\theta_1 \\ \ddot{x}_1 = l_1\ddot{\theta}_1\cos\theta_1 - l_1\dot{\theta}_1^2\sin\theta_1 \\ \ddot{y}_1 = -l_1\ddot{\theta}_1\sin\theta_1 - l_1\dot{\theta}_1^2\cos\theta_1 \end{cases} \tag{11}$$

For the second bob m_2 :

$$\begin{cases} x_2 = l_1\sin\theta_1 + l_2\sin\theta_2 \\ y_2 = l_1\cos\theta_1 + l_2\cos\theta_2 \\ \ddot{x}_2 = l_1\cos\theta_1\ddot{\theta}_1 + l_2\cos\theta_2\ddot{\theta}_2 - l_1\sin\theta_1\dot{\theta}_1^2 - l_2\sin\theta_2\dot{\theta}_2^2 \\ \ddot{y}_2 = -l_1\sin\theta_1\ddot{\theta}_1 - l_2\sin\theta_2\ddot{\theta}_2 - l_1\cos\theta_1\dot{\theta}_1^2 - l_2\cos\theta_2\dot{\theta}_2^2 \end{cases} \tag{12}$$

For rod m_2, we have:

$$\boldsymbol{T}_2 + m_2\boldsymbol{g} = m_2\boldsymbol{a_2}$$

$$\boldsymbol{a}_2 = \ddot{x}_2\boldsymbol{i} + \ddot{y}_2\boldsymbol{j}$$

$$\boldsymbol{T_2} = m_2(\ddot{x}_2\boldsymbol{i} + (\ddot{y}_2 - g)\boldsymbol{j})$$

Here we denote the force component which is along the rod by T_{1r} and T_{2r}.

$$T_{2r} = m_2(\ddot{x}_2\sin\theta_2 + (\ddot{y}_2 - g)\cos\theta_2)$$

$$\ddot{x}_2 = l_1\cos\theta_1\ddot{\theta}_1 + l_2\cos\theta_2\ddot{\theta}_2 - l_1\sin\theta_1\dot{\theta}_1^2 - l_2\sin\theta_2\dot{\theta}_2^2$$
$$\ddot{y}_2 = -l_1\sin\theta_1\ddot{\theta}_1 - l_2\sin\theta_2\ddot{\theta}_2 - l_1\cos\theta_1\dot{\theta}_1^2 - l_2\cos\theta_2\dot{\theta}_2^2$$

From Newton's 3rd law, we have $\boldsymbol{T}_2' = -\boldsymbol{T}_2$.

$$\boldsymbol{T}_1 - \boldsymbol{T}_2 + m_1\boldsymbol{g} = m_1\ddot{x}_1\boldsymbol{i} + m_1\ddot{y}_1\boldsymbol{j}$$

$$\boldsymbol{T}_1 = (m_1\ddot{x}_1 + m_2\ddot{x}_2)\boldsymbol{i} + (m_1\ddot{y}_1 + m_2\ddot{y}_2 - (m_1+m_2)g)\boldsymbol{j}$$
$$T_{1r} = (m_1\ddot{x}_1 + m_2\ddot{x}_2)\sin\theta_1 + (m_1\ddot{y}_1 + m_2\ddot{y}_2 - (m_1+m_2)g)\cos\theta_1$$
$$\ddot{x}_1 = \frac{1}{2}l_1\ddot{\theta}_1\cos\theta_1 - \frac{1}{2}l_1\dot{\theta}_1^2\sin\theta_1$$
$$\ddot{y}_1 = \frac{1}{2}l_1\ddot{\theta}_1\sin\theta_1 - \frac{1}{2}l_1\dot{\theta}_1^2\cos\theta_1$$

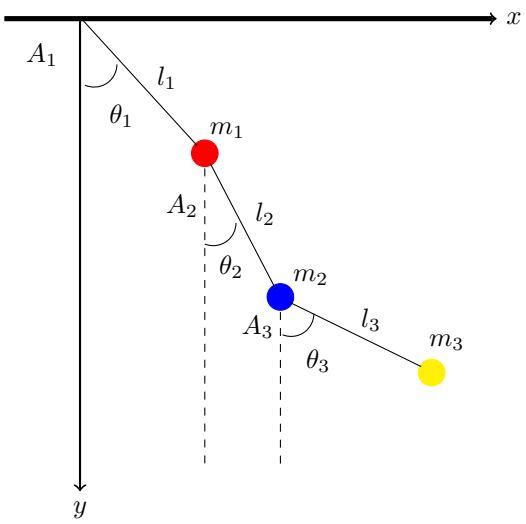

Figure 3: Triple Pendulum

3 Equations of Motion for Planar Triple Pendulum

Since the situation here is quite longish and boring, we only considered the configuration when the mass is centered in the bobs.

In order to describe the position of each bob more easily, we constructed a rectangular coordinates xOy. Then bob m_1, m_2 and m_3 can be described by x_1, y_1, x_2, y_2, x_3, y_3 and their velocities which are the time derivative of their position, are denoted by $\dot{x}_1$, $\dot{y}_1$, $\dot{x}_2$, $\dot{y}_2$, $\dot{x}_3$, $\dot{y}_3$.

It is easy to get:

$$\begin{cases} x_1 = l_1 \sin\theta_1 \\ y_1 = l_1 \cos\theta_1 \\ \dot{x}_1 = l_1\dot{\theta}_1 \cos\theta_1 \\ \dot{y}_1 = -l_1\dot{\theta}_1 \sin\theta_1 \end{cases}$$

$$\begin{cases} x_2 = l_1 \sin\theta_1 + l_2 \sin\theta_2 \\ y_2 = l_1 \cos\theta_1 + l_2 \cos\theta_2 \\ \dot{x}_2 = l_1\dot{\theta}_1 \cos\theta_1 + l_2\dot{\theta}_2 \cos\theta_2 \\ \dot{y}_2 = -l_1\dot{\theta}_1 \sin\theta_1 - l_2\dot{\theta}_2 \sin\theta_2 \end{cases}$$

$$\begin{cases} x_3 = l_1 \sin\theta_1 + l_2 \sin\theta_2 + l_3 \sin\theta_3 \\ y_3 = l_1 \cos\theta_1 + l_2 \cos\theta_2 + l_3 \cos\theta_3 \\ \dot{x}_3 = l_1\dot{\theta}_1 \cos\theta_1 + l_2\dot{\theta}_2 \cos\theta_2 + l_3\dot{\theta}_3 \cos\theta_3 \\ \dot{y}_3 = -l_1\dot{\theta}_1 \sin\theta_1 - l_2\dot{\theta}_2 \sin\theta_2 - l_3\dot{\theta}_3 \sin\theta_3 \end{cases}$$

Kinetic energy of each bob:

$$\begin{cases} T_1 = \dfrac{1}{2} m_1 l_1^2 \dot{\theta}_1^2 \\ T_2 = \dfrac{1}{2} m_2 [l_1^2\dot{\theta}_1^2 + l_2^2\dot{\theta}_2^2 + 2l_1 l_2 \dot{\theta}_1\dot{\theta}_2 \cos(\theta_1 - \theta_2)] \\ T_3 = \dfrac{1}{2} m_3 [l_1^2\dot{\theta}_1^2 + l_2^2\dot{\theta}_2^2 + l_3^2\dot{\theta}_2^2 + 2l_1 l_2 \dot{\theta}_1\dot{\theta}_2 \cos(\theta_1 - \theta_2) + 2l_2 l_3 \dot{\theta}_2\dot{\theta}_3 \cos(\theta_2 - \theta_3) + 2l_1 l_3 \dot{\theta}_1\dot{\theta}_3 \cos(\theta_1 - \theta_3)] \end{cases}$$

Potential energy of the system:

$$V = -m_1 g l_1 \cos\theta_1 - m_2 g(l_1 \cos\theta_1 + l_2 \cos\theta_2) - m_3 g(l_1 \cos\theta_1 + l_2 \cos\theta_2 + l_3 \cos\theta_3)$$

Lagrangian of the double pendulum system is:

$$\begin{aligned}\mathscr{L} = T - V = \frac{1}{2}(m_1+m_2+m_3)l_1^2\dot{\theta}_1^2 + \frac{1}{2}(m_2+m_3)l_2^2\dot{\theta}_2^2 + \frac{1}{2}m_3l_3^2\dot{\theta}_3^2 + (m_2+m_3)l_1l_2\dot{\theta}_1\dot{\theta}_2\cos(\theta_1-\theta_2) \\ + m_3l_1l_3\dot{\theta}_1\dot{\theta}_3\cos(\theta_1-\theta_3) + m_3l_2l_3\dot{\theta}_2\dot{\theta}_3\cos(\theta_2-\theta_3) \\ + (m_1+m_2+m_3)gl_1\cos\theta_1 + m_3gl_3\cos\theta_3\end{aligned} \tag{13}$$

From theoritical mechanics[1], we know the motion of a system can be determined using Lagrange Functions:

$$\begin{cases}\dfrac{d}{dt}\left(\dfrac{\partial\mathscr{L}}{\partial\dot{\theta}_1}\right) - \dfrac{\partial\mathscr{L}}{\partial\theta_1} = 0 \\ \dfrac{d}{dt}\left(\dfrac{\partial\mathscr{L}}{\partial\dot{\theta}_2}\right) - \dfrac{\partial\mathscr{L}}{\partial\theta_2} = 0 \\ \dfrac{d}{dt}\left(\dfrac{\partial\mathscr{L}}{\partial\dot{\theta}_3}\right) - \dfrac{\partial\mathscr{L}}{\partial\theta_3} = 0\end{cases}$$

Now we get equations of $\ddot{\theta}_1, \ddot{\theta}_2$ and $\ddot{\theta}_3$:

$$\begin{cases}(m_1+m_2+m_3)l_1\ddot{\theta}_1 + (m_2+m_3)l_2\cos(\theta_1-\theta_2)\ddot{\theta}_2 + m_3l_3\cos(\theta_1-\theta_3)\ddot{\theta}_3 \\ \qquad + (m_2+m_3)l_2\sin(\theta_1-\theta_2)\dot{\theta}_2^2 + m_3l_3\sin(\theta_1-\theta_3)\dot{\theta}_3^2 + (m_1+m_2+m_3)g\sin\theta_1 = 0 \\ \\ (m_2+m_3)l_1\cos(\theta_1-\theta_2)\ddot{\theta}_1 + (m_2+m_3)l_2\ddot{\theta}_2 + m_3l_3\cos(\theta_2-\theta_3)\ddot{\theta}_3 \\ \qquad - (m_2+m_3)l_1\sin(\theta_1-\theta_2)\dot{\theta}_1^2 + m_3l_3\sin(\theta_2-\theta_3)\dot{\theta}_3^2 + (+m_2+m_3)g\sin\theta_2 = 0 \\ \\ l_1\cos(\theta_1-\theta_3)\ddot{\theta}_1 + l_2\cos(\theta_2-\theta_3)\ddot{\theta}_2 + l_3\ddot{\theta}_3 \\ \qquad - l_1\sin(\theta_1-\theta_3)\dot{\theta}_1^2 - l_2\sin(\theta_2-\theta_3)\dot{\theta}_2^2 + g\sin\theta_3 = 0\end{cases} \tag{14}$$

Since solving these equations and get analytical expressions of $\ddot{\theta}_1, \ddot{\theta}_2$ and $\ddot{\theta}_3$ is too longish and meaningless, we use Gauss Elimination to get numerical value of $\ddot{\theta}_1, \ddot{\theta}_2$ and $\ddot{\theta}_3$ at each step.

Part III
Runge-Kutta Method in solving Ordinary Differential Equations(ODEs)

4 Runge-Kutta Method

A differential equation is an equation involving derivatives, which is in the form(from a physical view):

$$y'(t) = f(t, y(t))$$

Several singel differential equations can form a system of differential equations. The first-order ODE system has the form:

$$\begin{aligned}y_1' &= f_1(t, y_1, \cdots, y_n) \\ y_2' &= f_2(t, y_1, \cdots, y_n) \\ &\vdots \\ y_n' &= f_n(t, y_1, \cdots, y_n)\end{aligned}$$

We have an old, explicit, and fairly simple method to solve ODE, namely Euler's Method.

$$y_{i+1} = y_i + h * f(t_i, y_i)$$

But since Euler's Method has relatively huge error compared with other methods, we are not going to use Euler's Method to solve our ODEs.
Another method of solving ODEs is Runge-Kutta Method[2], which is actually a modified edition of Euler's Method.

$$
\begin{aligned}
s_1 &= f(t_i, y_i) \\
s_2 &= f(t_i + \frac{h}{2}, y_i + \frac{h}{2}s_1) \\
s_3 &= f(t_i + \frac{h}{2}, y_i + \frac{h}{2}s_2) \\
s_4 &= f(t_i + h, y_i + hs_3) \\
y_{i+1} &= y_i + \frac{h}{6}(s_1 + 2s_2 + 2s_3 + s_4)
\end{aligned}
$$

This method is called the forth-order Runge-Kutta (denoted by RK4), since the error of this method is limited to $O(h^5)$. But, typical Runge-Kutta Method is not that fast and accurate[2], we are going to use the modified Runge-Kutta Method.

5 Embedded Runge-Kutta Method

The general form of a fifth-order Runge-Kutta Method can be expressed as follows[3, 4]:

$$
\begin{aligned}
s_1 &= hf(t_i, y_i) \\
s_2 &= hf(t_i + a_2h, y_i + b_{21}s_1) \\
s_3 &= hf(t_i + a_3h, y_i + b_{31}s_1 + b_{32}s_2) \\
s_4 &= hf(t_i + a_4h, y_i + b_{41}s_1 + b_{42}s_2 + b_{43}s_3) \\
s_5 &= hf(t_i + a_5h, y_i + b_{51}s_1 + b_{52}s_2 + b_{53}s_3 + b_{54}s_4) \\
s_5 &= hf(t_i + a_6h, y_i + b_{61}s_1 + b_{62}s_2 + b_{63}s_3 + b_{64}s_4 + b_{65}s_5) \\
y_{i+1} &= y_i + h(c_1s_1 + c_2s_2 + c_3s_3 + c_4s_4 + c_5s_5 + c_6s_6) + O(h^6)
\end{aligned}
$$

The particular values of the various constants are given in Table 1.[4]. This method was proposed by Dormand and Prince. Also it is a member of Runge-Kutta methods. So we use both typical Runge-Kutta Method and Embedded Runge-Kutta Method to solve our differential equations of double pendulum.

i	a_i	b_{ij}							c_i
1									$\frac{35}{384}$
2	$\frac{1}{5}$	$\frac{1}{5}$							0
3	$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$						$\frac{500}{1113}$
4	$\frac{4}{5}$	$\frac{44}{45}$	$-\frac{56}{15}$	$\frac{32}{9}$					$\frac{125}{192}$
5	$\frac{8}{9}$	$\frac{19372}{6561}$	$-\frac{25360}{2187}$	$\frac{64448}{6561}$	$-\frac{212}{729}$				$-\frac{2187}{6784}$
6	1	$\frac{9017}{3168}$	$-\frac{355}{33}$	$\frac{575}{13824}$	$\frac{46732}{5247}$	$\frac{49}{176}$	$-\frac{5103}{18656}$		$\frac{11}{84}$
7	1	$\frac{35}{384}$	0	$\frac{575}{13824}$	$\frac{5000}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	0
$j =$		1	2	3	4	5	6	7	

Table 1: Dormand-Prince Parameters for Embedded Runge-Kutta Method

Part IV
Programming

6 Double Pendulum

In our code, we use the following symbols:

$$y_1 = \theta_1; \quad y_2 = \dot{\theta}_1;$$
$$y_3 = \theta_2; \quad y_4 = \dot{\theta}_2;$$

So we have:

$$\begin{cases} \dot{y}_1 = y_2 \\ \dot{y}_2 = \dfrac{m_2 g \sin y_3 \cos(y_1 - y_3) - m_2 \sin(y_1 - y_3)[l_2 y_4^2 + l_1 y_2^2 \cos(y_1 - y_3)] - (m_1 + m_2) g \sin y_1}{l_1[(m_1 + m_2) - m_2 \cos^2(y_1 - y_3)]} \\ \dot{y}_3 = y_4 \\ \dot{y}_4 = \dfrac{(m_1 + m_2) l_1 \sin(y_1 - y_3) y_2^2 + m_2 l_2 y_4^2 \sin(y_1 - y_3) \cos(y_1 - y_3) - (m_1 + m_2) g[\sin y_3 - \sin y_1 \cos(y_1 - y_3)]}{[(m_1 + m_2) - m_2 \cos^2(y_1 - y_3)]} \end{cases}$$

We built ***Double_Pendulum_Calculate*** class, containing the mass of each bob, the length of each rod, the angle θ_1, θ_2, angular velocity $\dot{\theta}_1$, $\dot{\theta}_2$, iteration function $\boldsymbol{f}$, and RungeKutta method ***RKCK4_solve***.

```
class Double_Pendulum_Culculate
{
public:
    Double_Pendulum_Culculate();
    void func(int);
    double ShowY_1();
    double ShowY_2();
    double ShowY_3();
    double ShowY_4();
    double get_time(){return t;}
    double get_y_1(){return y_1;}
    double get_y_2(){return y_2;}
    double get_y_3(){return y_3;}
    double get_y_4(){return y_4;}
    int get_n(){return n;}
    double get_m(){return m;}
    double get_l(){return l;}
    void set_begin_time(double input_begin_time=0);
    void set_last_time(double input_last_time);
    void set_time(double input_time){t=input_time;}
    void set_y_1_initial(double y_1_initial);
    void set_y_2_initial(double y_2_initial);
    void set_y_3_initial(double y_3_initial);
    void set_y_4_initial(double y_4_initial);
    void set_m(double mass){m=mass;}
    void set_l(double length){l=length;}
    void calculate_n();
    void f(Double_Pendulum_Culculate pendulum, double t, PVector y, PVector output);
    void RKCK4_solve(Double_Pendulum_Culculate pendulum, double y_1_initial, double
    y_2_initial, double y_3_initial, double y_4_initial,
                                PVector w, double* output_y_1, double* output_y_2, double*
    output_y_3, double* output_y_4, double* output_t);
private:
    double begin_time,last_time,y_1,y_2,y_3,y_4;
    double t=begin_time;
    int n;
    double m,l;
};
```

In solving ODEs, we constructed a solution vector ***Vector***, namely ****PVector***. The step length of iteration is:

```
#define h 0.001
```

```
typedef struct _Vector_4d_for_double
    {
      double x[4];
    }Vector, *PVector;
```

For function $\boldsymbol{f}$, we have:

```
void Double_Pendulum_Culculate::f(Double_Pendulum_Culculate pendulum, double t, PVector
    y, PVector output)
{
    output->x[0]=(y->x[1]);
    output->x[1]=(g*sin(y->x[2])*cos(y->x[0]-y->x[2])
            -sin(y->x[0]-y->x[2])*(pow(y->x[3],2)
            +l*pow(y->x[1],2)*cos(y->x[0]-y->x[2]))
            -(m+1)*g*sin(y->x[0]))
            /(m+1-pow(cos(y->x[0]-y->x[2]),2));
    output->x[2]=y->x[3];;
    output->x[3]=((m+1)*l*sin(y->x[0]-y->x[2])*pow(y->x[1],2)
            +pow(y->x[3],2)*sin(y->x[0]-y->x[2])*cos(y->x[0]-y->x[2])
            -(m+1)*g*(sin(y->x[2])-sin(y->x[0])*cos(y->x[0]-y->x[2])))
            /(m+1-pow(cos(y->x[0]-y->x[2]),2));
}
```

When it comes to our main function ***RKCK4_solve***, we used Embedded Runge-Kutta Method:

```
void Double_Pendulum_Culculate::RKCK4_solve(Double_Pendulum_Culculate pendulum, double
    y_1_initial, double y_2_initial, double y_3_initial, double y_4_initial,
                                            PVector w, double* output_y_1, double*
    output_y_2,double* output_y_3, double* output_y_4, double* output_t)
{

    pendulum.calculate_n();
    pendulum.set_time(pendulum.get_time());
    pendulum.set_y_1_initial(y_1_initial);
    pendulum.set_y_2_initial(y_2_initial);
    pendulum.set_y_3_initial(y_3_initial);
    pendulum.set_y_4_initial(y_4_initial);

    PVector w1 = new Vector;
    PVector w2 = new Vector;
    PVector w3 = new Vector;
    PVector w4 = new Vector;
    PVector w5 = new Vector;
    PVector w6 = new Vector;
    PVector w7 = new Vector;
    PVector s1 = new Vector;
    PVector s2 = new Vector;
    PVector s3 = new Vector;
    PVector s4 = new Vector;
    PVector s5 = new Vector;
    PVector s6 = new Vector;
    PVector s7 = new Vector;

    w->x[0]=pendulum.get_y_1();
    w->x[1]=pendulum.get_y_2();
    w->x[2]=pendulum.get_y_3();
    w->x[3]=pendulum.get_y_4();

    for(int i=1;i<=pendulum.get_n()+1;i++)
    {
        f(pendulum,pendulum.get_time(),w,s1);
        add1(w,s1,b_21*h,w1);
        f(pendulum, pendulum.get_time()+h*a_2,w1,s2);
```

```
            add2(w,s1,s2,b_31*h,b_32*h,w2);
            f(pendulum, pendulum.get_time()+h*a_3,w2,s3);
            add3(w,s1,s2,s3,b_41*h,b_42*h,b_43*h,w3);
            f(pendulum, pendulum.get_time()+h*a_4, w3,s4);
            add4(w,s1,s2,s3,s4,b_51*h,b_52*h,b_53*h,b_54*h,w4);
            f(pendulum, pendulum.get_time()+h*a_5, w4,s5);
            add5(w,s1,s2,s3,s4,s5,b_61*h,b_62*h,b_63*h,b_64*h,b_65*h,w5);
            f(pendulum, pendulum.get_time()+h*a_6, w5,s6);
            add6(w,s1,s2,s3,s4,s5,s6,b_71*h,b_72*h,b_73*h,b_74*h,b_75*h,b_76*h,w6);
            f(pendulum, pendulum.get_time()+h*a_7, w6,s7);
            sum(s1,s2,s3,s4,s5,s6,s7,w7);
            add1(w,w7,h,w);
            pendulum.set_time(pendulum.get_time()+h);
            output_y_1[i-1] = w->x[0];
            output_y_2[i-1] = w->x[1];
            output_y_3[i-1] = w->x[2];
            output_y_4[i-1] = w->x[3];
            output_t[i-1]= pendulum.get_time();
        }
```

Then we get the result of calculation in Vector output_y_1, output_y_2, output_y_3 and output_y_4.

7 Triple Pendulum

In our code, we use the following symbols:

$$y_1 = \theta_1; \quad y_2 = \dot{\theta}_1; \quad y_3 = \theta_2;$$
$$y_4 = \dot{\theta}_2; \quad y_5 = \theta_3; \quad y_6 = \dot{\theta}_3;$$

The method of solving triple pendulum is quite similar to double pendulum, while we used Gauss Elimination to reduce the complexity of formulae:

```
void Gauss_pivot(double A[N][N+1],double* &outputSolution_triple)
{
    int trytime;
    for(trytime=0;trytime<N;trytime++)
    {
    for(int i=trytime;i<N;i++)
    {
        for(int j=N;j>=trytime;j--)
        {A[i][j]=A[i][j]/A[i][trytime];}
    }
    for(int i=trytime+1;i<N;i++)
    {
        for(int j=0;j<N+1;j++)
        {A[i][j]=A[i][j]-A[trytime][j];}
    }
    outputSolution_triple[N-1]=-A[N-1][N];
    for(int i=N-2;i>=0;i--)
    {
        double temp=0;
        for(int j=i+1;j<=N-1;j++)
        {temp+=A[i][j]*outputSolution_triple[j];}
        outputSolution_triple[i]=-A[i][N]-temp;}
  }
}
```

8 *Dialog* class

Dialog is the interface class, and also the biggest class in our code. Almost all classes are instantiated in this class. For example, we instantiated the class *Double_Pendulum_Culculate* in *Dialog*, and passed the value from UI interface to this class. Then we calculated and solved double pendulum, got the

coordinates of bobs, passed them to other *Graph* classes to paint graphs.

In *Dialog* class, we used two ways to design the interfaces. The first one is UI design. This method is relatively easier, we can build the connections between widgets in *Dialog* and realize the switch among different pages only by drawing and draging widgets in Qt Designer, working with signal-slot in Qt and adding some ordinary code.

```
private:
    Double_Pendulum_Culculate *double_pendulum;
    Triple_Pendulum_Culculate *triple_pendulum;
    Graph1 *graph_one;
    Graph1 *graph_load_one;
    Graph1_1 *graph_one_one;
    Graph1_2 *graph_one_two;
    Graph1_3 *graph_one_three;
    Graph2 *graph_two;
    Graph3 *graph_three;
    Graph3 *graph_load_three;
    Graph3_1 *graph_three_one;
    Graph3_2 *graph_three_two;
    Graph3_3 *graph_three_three;
    Graph4 *graph_four;
    CalculateWidget *Calculating;
    CalculateWidget *Saving;
    CalculateWidget *Nothing;
    CalculateWidget *Loading;
    Fun *fun;
```

Switch among pages:

```
/**********************Switch among pages**************************/
void Dialog::on_StartButton_2_clicked()
void Dialog::on_pushButton_10_clicked()
void Dialog::on_ContinueButton_clicked()
{
    QFile file_1("theta1 of double pendulum.txt");
    QFile file_2("theta1 of triple pendulum.txt");
    if(!file_1.exists())
    {ui->Double_2->setCheckable(false); ui->Double_2->setChecked(false);}
    if(file_1.exists())
    {ui->Double_2->setCheckable(true); ui->Double_2->setChecked(true);}
    if(!file_2.exists())
    {ui->Triple_2->setCheckable(false); ui->Triple_2->setChecked(false);}
    if(file_2.exists())
    {ui->Triple_2->setCheckable(true);}
    bool double_2=ui->Double_2->isChecked();
    bool triple_2=ui->Triple_2->isChecked();
    if(!double_2&&!triple_2)
    {ui->StartButton_4->setEnabled(false);}
    ui->MainWidget->setCurrentIndex(9);
}
```

Data transforming in *Graph 1* class:

```
void Dialog::on_number_1_valueChanged(double arg1)
void Dialog::on_number_2_valueChanged(double arg1)
void Dialog::on_number_3_valueChanged(double arg1)
void Dialog::on_number_4_valueChanged(double arg1)
```

The second method is designing by code, which is more complicated than the former one. Property of widgets cannot be set directly like UI design, so we need to define property of widgets by ourselves. After this, we need to write slot function and signal, connect them together, realize the transformation of data at the same time. This process is quite longish and boring, and it also contribute to the length of *Dialog.cpp*, which has 1600 lines. Some examples are shown below:

```
    mylabel1 = new QLabel(tr("M1"));
    mylabel2 = new QLabel(tr("M2"));
    mylabel3 = new QLabel("w1");
    mylabel4 = new QLabel("w2");
    myslider1 = new QSlider(Qt::Horizontal);
    myslider2 = new QSlider(Qt::Horizontal);
    myslider3 = new QSlider(Qt::Horizontal);
    myslider4 = new QSlider(Qt::Horizontal);
/*************************First calculation*********************************/
void Dialog::transmit()
{
    double m1=double(myslider1->value());
    QString str1 = QString("%1").arg(m1);
    mylineEdit1->setText(str1);
    double m2=double(myslider2->value());
    QString str2 = QString("%1").arg(m2);
    mylineEdit2->setText(str2);
    double w1=(double(myslider3->value())*2)/100;
    QString str3 = QString("%1").arg(w1);
    mylineEdit3->setText(str3);
    double w2=(double(myslider4->value())*2)/100;
    QString str4 = QString("%1").arg(w2);
    mylineEdit4->setText(str4);
}
```

9 *Help/Connect/Fun* class

Help class provides help information for users. When click *Help* in the titlebar, user can read help and guide for using this software. *Connect* class provides the information of author, users can get the information by clicking *Graph* in the titlebar. *Fun* class provides users something interesting to try. All these windows are designed by UI design. When user click a button, the window is shown throught corresponding slot funtion.

10 *TitleBar* class

TitleBar class is written by code, rather than UI design. For some special reason, we cannot add new button to Dialog interface, so we made a new class *TitleBar*, while the dialog doesn't have frame. We used event filter to achieve functions like minimizing the window, closing the window, showing the name and icon of windows. Besides, we add a button to achieve saving data and seeking for help, etc.

```
/*************************TitleBar Class*****************************/
TitleBar::TitleBar(QWidget *parent)
    : QWidget(parent)
{
    setStyleSheet("TitleBar{border:2px groove gray;border-radius:10px;padding:2px 4px;}"
    );
    helper=new Help;
    contact=new Connect;
    setFixedHeight(43);
  ......................
```

11 *CalculateWidget* class

This function of this class is a reminder. Since calculating may be a little bit slow, this reminder box shows users the computer is calculating, not lagging. This kind of widget can also tell users if the data has been saved successfully. Some code is shown below:

```
CalculateWidget::CalculateWidget(QWidget *parent)
    : QWidget(parent)
```

```
{
    int width = parent->width();
    this->resize(width, 28);
    this->setWindowFlags(Qt::FramelessWindowHint);
    QPalette palette;
    QColor color(190, 230, 250);
    color.setAlphaF(0.6);
    palette.setBrush(this->backgroundRole(), color);
    this->setPalette(palette);
    this->setAutoFillBackground(true);
    setGeometry(0,520,width,28);
    close_button= new QToolButton(this);
    QPixmap close_pix = style()->standardPixmap(QStyle::SP_TitleBarCloseButton);
    close_button->setIcon(close_pix);
    close_button->setStyleSheet("QToolButton{background-color: transparent;}");
    int height = this->height();
    close_button->setGeometry(width-20, 0, 20, 20);
    msg_label = new QLabel(this);
    msg_label->setGeometry(QRect(5, 5, 20, 20));
    msg_label->setStyleSheet("background-color: transparent;");
    msg_label->setScaledContents(true);
    ask_label = new QLabel(this);
    ask_label->setStyleSheet("background-color: transparent; color: red;");
    ask_label->setGeometry(QRect(30, 0, width - 60, height));
    ask_label->setAlignment(Qt::AlignCenter);
    close_button->setCursor(Qt::PointingHandCursor);
    QObject::connect(close_button, SIGNAL(clicked()), this, SLOT(closeWidget()));
}
```

12 *Graph* class

To coorperate with OpenGL, we built these classes (4 classes). The base class of *Graph1* and *Graph3* is *QOpenGLWindow*, while the base class of *Graph2* and *Graph4* is *QGLWidget*. *Graph1* and *Graph3* are designed for keboard operating, *Graph2* and *Graph4* are designed for mouse operating. The mechanisms behind these two base classes have something in common. We used three virtual functions *void initializegl(), void resizegl(int w, int h), void paintgl()* to initialize and paint. *Qtimer* in Qt is used to time and slot function called *updateAnimation()* is used to initialize animate. *Set* function is used to deliver data from *Double_Pendulum_Calculate* to *Graph*.

```
protected:
    void keyPressEvent(QKeyEvent *event);
    void initializeGL();
    void resizeGL(int w, int h);
    void paintGL();
    void paintEvent(QPaintEvent *event);
    void resizeEvent(QResizeEvent *event);
private:
    QTimer *timer;
    QOpenGLContext* context;
    QOpenGLFunctions* openGLFunctions;
public slots:
    void updateAnimation();
```

Graph 1_1, *Graph 1_2*, *Graph 1_3*, *Graph 3_1*, *Graph 3_2* and *Graph 3_3* are used to paint phase graph for pendulums, which are also based on *QOpenGLWindow*. We painted phase diagram by idiotic OpenGL skill.

```
void Graph1_1::paintGL()
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    glTranslatef(moveX, moveY, moveZ);
    glBegin(GL_POINTS);
    for(int i=0;i<n_graph1_1;i++)
```

```
        {glVertex3f(2.5*y1[i],0.5*y2[i],-3);}
        glEnd();
        glLineWidth(2);
        glBegin(GL_LINES);
            glVertex3f(0.0,-100,-3);
            glVertex3f(0.0,100,-3);
        glEnd();
        glLineWidth(2);
        glBegin(GL_LINES);
            glVertex3f(-100,0.0,-3);
            glVertex3f(100,0.0,-3);
        glEnd();
        for(double i=-500;i<=500;i++)
        {glLineWidth(0.05);glBegin(GL_LINES);glVertex3f(i/5,-j,-3);glVertex3f(i/5,j,-3);
       glEnd();glBegin(GL_LINES);glVertex3f(j,i/5,-3);glVertex3f(-j,i/5,-3); glEnd();}
        glFlush();
}
```

13 Other functions of our software

Here we're going to introduce other functions of our software.

1. Volume of Audio

In Setting Area, user can adjust the volume of audio by a slider. It is realized by *QMediaPlayer* class, we only add a slot function to read the value of Audio_Slider and transfer it into the volume.

```
void Dialog::on_Audio_valueChanged(int value)
{
    player->setVolume(value);
}
```

2. Mouse and Keyboard

Users can choose operating ways in Setting Area, which contains by-keyboard mode and by-mouse mode. We can determine the mode and switch to other windows by checking the state of button. Users can use mouse to drag the bobs of pendulum, giving them initial configuration. Qt has monitoring function for mouse (detect whether the mouse is clicked or not), we need to rewrite this function. While moving the mouse, clicking different key of the mouse will have different effect. Data of mouse's position and motion is delivered to *Graph2* and *Graph4* through *Dialog* is the basic mechanism of mouse operating. As for keyboard operating, Qt has Keyboard event, we used and rewrite it.

```
void Dialog::mouseMoveEvent(QMouseEvent *e)
{
    if(e->buttons()&Qt::LeftButton)
    {
        if(e->x()<=520 && e->x()>=0 && e->y()<=520 && e->y()>=0)
        {
            graph_two->transmitposition1(e->x()-20,e->y()-50);
            graph_four->transmitposition1(e->x()-20,e->y()-50);
        }
    }
    else if(e->buttons()&Qt::RightButton)
    {
        if(e->x()<=520 && e->x()>=0 && e->y()<=520 && e->y()>=0)
        {
            graph_two->transmitposition2(e->x()-20,e->y()-50);
            graph_four->transmitposition3(e->x()-20,e->y()-50);
        }
    }
    else if(e->buttons()&Qt::MidButton)
    {
```

```
21          if(e->x()<=520 && e->x()>=0 && e->y()<=520 && e->y()>=0)
22          {
23              graph_four->transmitposition2(e->x()-20,e->y()-50);
24          }
25      }
26  }
27  void Dialog::keyPressEvent( QKeyEvent *e )
28  {
29
30      switch ( e->key() )
31  {
32      case Qt::Key_Escape:
33          close();
34          break;
35  }}
```

3. Save and Load

We used *QFile* and *Qtextstream* in Qt to read and write data via txt document. When user click *Save Double*, *TitleBar* will transmit signal to *Dialog*. After *Dialog* received the signal, slot function *void save1()* will be triggered. Data is saved through slot function in many txt documents in current dictionary. When user push the button *Load*, our software can judge whether saved files exist or not through Qfile. Then the software read data by *Qtextstrean*, return to the moment when user saved data.

```
1       QFile file("theta1 of double pendulum.txt");
2       QFile file2("omega1 of double pendulum.txt");
3       QFile file3("theta2 of double pendulum.txt");
4       QFile file4("omega2 of double pendulum.txt");
5       QFile file5("parameters of double pendulum.txt");
6       if (file2.open(QFile::ReadWrite | QIODevice::Truncate)|file.open(QFile::ReadWrite |
        QIODevice::Truncate)
7           |file3.open(QFile::ReadWrite | QIODevice::Truncate)|file4.open(QFile::ReadWrite
        | QIODevice::Truncate)
8               |file5.open(QFile::ReadWrite|QIODevice::Truncate))
9       {
10          QTextStream out(&file);
11          QTextStream out2(&file2);
12          QTextStream out3(&file3);
13          QTextStream out4(&file4);
14          QTextStream out5(&file5);
15          for(int i=0 ;i<n;i++)
16          {
17              out<<y1[i]<<"\r\n";
18              out2<<y2[i]<<"\r\n";
19              out3<<y3[i]<<"\r\n";
20              out4<<y4[i]<<"\r\n";
21          }
22          out5<<n<<"\r\n"<<graph_one->get_l1()<<"\r\n"<<graph_one->get_l2()<<"\r\n"<<
        graph_one->get_step_n();
23          file.close();
24          file2.close();
25          file3.close();
26          file4.close();
27          file5.close();
28          QApplication::restoreOverrideCursor();
29      }
30      }
```

4. Dialog Box

Some of boxes like *Help* and *Connect* are instantiated in *TitleBar* class, while *CalculateWidget* and *Fun* are instantiated in *Dialog* class. Because Menu is defined in *TitleBar*, we instantiate *About*, *Help*, and *Connect* in *Titlebar*.

```
TitleBar::TitleBar(QWidget *parent)
    : QWidget(parent)
{
    setStyleSheet("TitleBar{border:2px groove gray;border-radius:10px;padding:2px 4px;}");
    helper=new Help;
    contact=new Connect;
.................
}
void TitleBar::onMenu_Triggered(QAction *action)
{
    if(action == helpAction)
    {
        helper->show();
    }
    else if(action == save1Action)
    {
        emit save1();
    }
    else if(action == save2Action)
    {
        emit save2();
    }
    else if(action == aboutAction)
    {
        contact->show();
    }
}
```

14 *OpenGL Works*

Painting bobs and making the cartoon before simulation of pendulum used OpenGL. It is relatively easy to paint bobs. To make them look better, we used illumination. Setting illumination contains setting the position of light source, the background color of objects, the color of specular reflection lights and diffusing reflection lights. To make bobs look vivid, we must set the background color much darker than reflection color. The method of painting rods is connecting the centers of bobs directly. The trajectory of bobs is drawn by dots every 0.001 seconds in *Double_Pendulum_Calculate*. Our opening cartoon simulated the scene from some science fiction films. We constructed structure *Cylinder* to realize space roaming, using *Update* function realized the three dimensional motion of structure *Cylinder*.

Part V
Conclusion

We built a software which can simulate the planar motion of double pendulum and triple pendulum in gravitational field, while showing phase graphs of pendulums. We used Embedded Runge-Kutta Method to solve equations of motion of pendulums, and used Gauss Elimination Method to avoid longish simplifying of formulae. Using Qt Creator 5.7.1 and OpenGL, we made visualization and built the user interface. Users can explore chaotic motion of pendulums by himself, concluding the characteristics of chaotic motion directly by observing the motion of colorful bobs. Users can also learn the nature of chaotic motion by investigate phase graph of different initial value and different energy.

When the initial position of bobs is relatively small, the trajectory and phase diagram are all very beautiful and regular (initial value $\theta_1 = 1.000, \omega_1 = 0, \theta_2 = 1.000, \omega_2 = 0, M_1 = 1, M_2 = 1, L_1 = 1, L_2 = 1$):

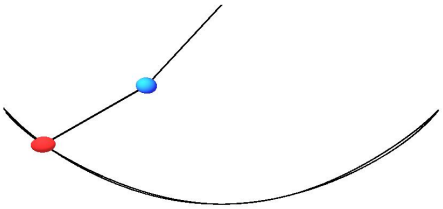

Figure 4: Trajectory of quasi-periodic double pendulum

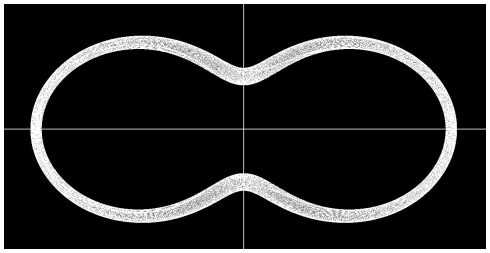

Figure 5: Phase graph of quasi-periodic double pendulum

Also we found that the pendulum is quasi-periodic when it is in very low energy (small amplitude situation). We also find a quasi-periodic initial value occasionally, i.e.

$$\begin{cases} \theta_1 = 1.000 \\ \omega_1 = 0 \\ \theta_2 = 2.500 \\ \omega_2 = 0 \end{cases}$$

$$\begin{cases} M_1 = 1.00 \\ M_2 = 3.00 \\ L_1 = 2.00 \\ L_2 = 1.00 \end{cases}$$

From these two figure we can clearly see that quasi-periodic motion's phase diagram is almost regular.

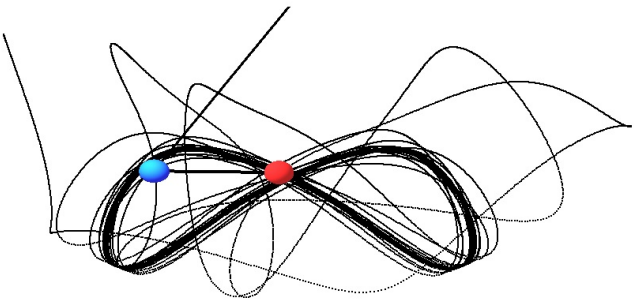

Figure 6: Trajectory of quasi-periodic double pendulum

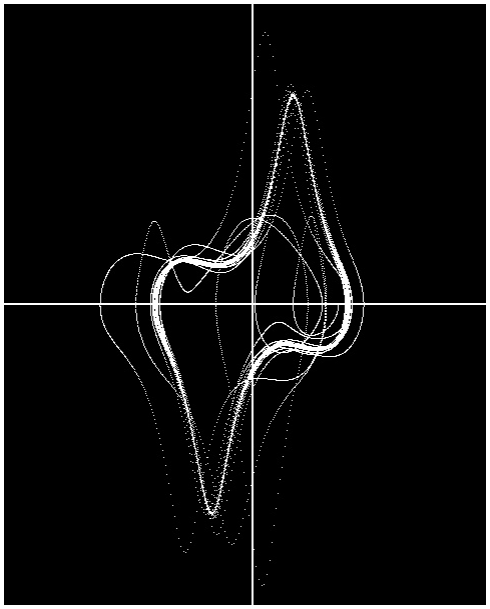

Figure 7: Phase graph of quasi-periodic double pendulum

But if the initial value is not so lucky, we will get chaotic motion(initial value $\theta_1 = 2.000, \omega_1 = 0, \theta_2 = 1.000, \omega_2 = 0, M_1 = 1, M_2 = 1, L_1 = 1, L_2 = 1$):

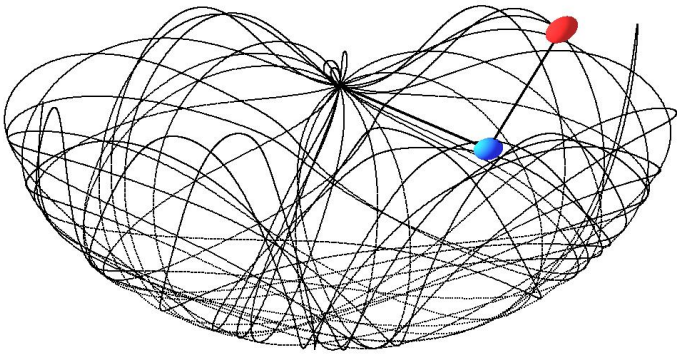

Figure 8: Trajectory of quasi-periodic double pendulum

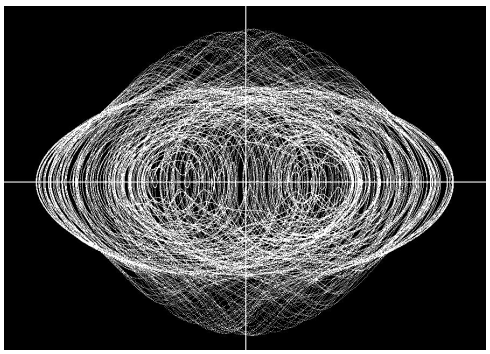

Figure 9: Phase graph of quasi-periodic double pendulum

Acknowledgements

Here we would like to show our appreciation to Yuyan Chen, Yaer from Shenzhen, Yiquersanli.

References

[1] L.D.Landau. Mechanics. 5th ed. Beijing: Higher Education Press, 2007.

[2] Sauer, Tim. Numerical Analysis. 2nd ed. Boston: Pearson Addison Wesley, 2006.

[3] Press, William H. Numerical Recipes in C : The Art of Scientific Computing. Cambridge, Angleterre: Cambridge UP, 2002.

[4] Dormand, J.R, and Prince, P.J. 1980, "A Family of Embedded Runge-Kutta Formulae," Journal of Computational and Applied Mathematics, vol. 6, pp. 19–26.