第 18 章

单摆和双摆模拟

本章首先介绍单摆和双摆系统的公式推导，然后通过 odeint()对其进行数值求解并制作动画演示程序。

18.1 单 摆 模 拟

如图 18-1 所示，有一根不可伸长、质量不计的细棒，上端固定，下端系一质点，这样的装置叫做单摆。

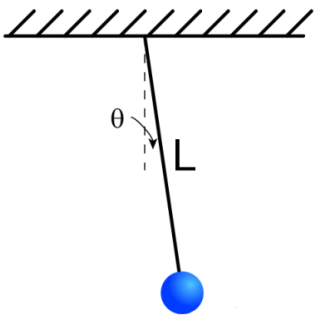

图 18-1 单摆装置示意图

根据牛顿力学定律，我们可以列出如下微分方程：

$$\frac{d^2\theta}{dt^2}+\frac{g}{l}\sin\theta=0$$

其中，θ 为单摆的摆角，l 为单摆的长度，g 为重力加速度。

此微分方程的符号解无法直接求出，因此只能调用 odeint()对其求数值解。

odeint()的调用参数如下：

```
odeint(func, y0, t, ...)
```

其中，func 是 Python 的一个函数对象，用来计算微分方程组中每个未知函数的导数；y0 为微分方程组中每个未知函数的初始值；t 为需要进行数值求解的时间点。它返回的是一个二维数组 result，其第 0 轴的长度为 t 的长度，第 1 轴的长度为变量的个数，因此 result[:,i]为第 i 个未知函数的解。

计算微分的 func 函数的调用参数为 func(y, t)，其中 y 是一个数组，为每个未知函数在 t 时刻的值，而 func 的返回值是每个未知函数在 t 时刻的导数。

odeint()要求每个微分方程只包含一阶导数，因此我们需要对前面的微分方程进行如下变形：

$$\frac{d\theta(t)}{dt}=v(t)$$

$$\frac{dv(t)}{dt} = -\frac{g}{l}\sin\theta(t)$$

下面是计算单摆轨迹的程序，摆角和时间的关系如图 18-2 所示。

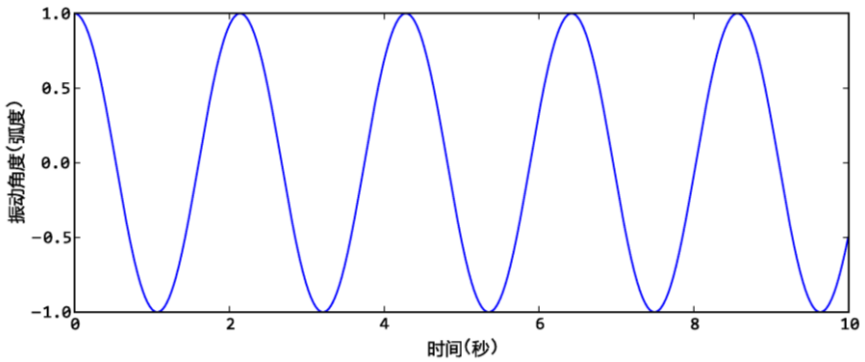

图 18-2　初始角度为 1 弧度的单摆摆动角度和时间的关系

simple_pendulum_odeint.py
用 odeint 计算单摆轨迹

```
from math import sin
import numpy as np
from scipy.integrate import odeint

g = 9.8

def pendulum_equations(w, t, l):
    th, v = w
    dth = v
    dv  = - g/l * sin(th)
    return dth, dv

if __name__ == "__main__":
    import pylab as pl
    t = np.arange(0, 10, 0.01)
    track = odeint(pendulum_equations, (1.0, 0), t, args=(1.0,))
    pl.plot(t, track[:, 0])
    pl.xlabel(u"时间(秒)")
    pl.ylabel(u"震度角度(弧度)")
    pl.show()
```

odeint()还有一个 args 参数，参数值为一个元组，这些值都会作为额外的参数传递给计算导数的函数。程序使用这种方式将单摆的长度传递给 pendulum_equations()。

18.1.1 小角度时的摆动周期

高中物理课上讲过，当最大摆动角度很小时，单摆的摆动周期可以使用如下公式计算：

$$T_0 = 2\pi\sqrt{\frac{l}{g}}$$

这是因为当 $\theta << 1$ 时，$\sin\theta \approx \theta$，这样微分方程就变成了：

$$\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2} + \frac{g}{l}\theta = 0$$

此微分方程的解是一个简谐振动方程，很容易计算其摆动周期。下面我们用 SymPy 对这个微分方程进行符号求解：

```
>>> from sympy import symbols, Function, dsolve
>>> t,g,l = symbols("t,g,l",real=True) # 分别表示时间、重力加速度和长度
>>> y = Function("y") # 摆角函数用 y(t)表示
>>> dsolve(y(t).diff(t,2) + g/l*y(t), y(t))
```

$$y(t) = C_1 \sin(\frac{t\sqrt{g}}{\sqrt{l}}) + C_2 \cos(\frac{t\sqrt{g}}{\sqrt{l}})$$

可以看到，简谐振动方程的解是由两个频率相同的三角函数构成的，其周期为 $2\pi\sqrt{\frac{l}{g}}$。

18.1.2 大角度时的摆动周期

但是当初始摆角增大时，上述近似处理会带来无法忽视的误差。下面让我们看看如何用数值计算的方法求出单摆在任意初始摆角时的摆动周期。

simple_pendulum_period.py
数值法求单摆摆动周期

要计算摆动周期，只需要计算从最大摆角到 0 摆角所需的时间，摆动周期是此时间的 4 倍。为了计算出这个时间值，首先需要定义一个函数 pendulum_th()，计算任意时刻的摆角：

```
def pendulum_th(t, l, th0):
    track = odeint(pendulum_equations, (th0, 0), [0, t], args=(l,))
    return track[-1, 0]
```

pendulum_th()计算长度为 l、初始角度为 th0 的单摆在时刻 t 的摆角。此函数仍然使用 odeint()进行微分方程组求解，只是我们只需要计算时刻 t 的摆角，因此传递给 odeint()的时间序列为

[0, t]。odeint()内部会对时间进行细分，以保证最终的解是正确的。

接下来只需找到第一个使 pendulum_th()的值为 0 的时间即可。这相当于对 pendulum_th()求解，可以使用 scipy.optimize.fsolve()对这种非线性方程进行求解：

```
def pendulum_period(l, th0):
    t0 = 2*np.pi*sqrt( l/g ) / 4
    t = fsolve( pendulum_th, t0, args = (l, th0) )
    return t*4
```

和 odeint()一样，我们通过 fsolve()的 args 参数将额外的参数传递给 pendulum_th()。fsolve()求解时需要一个初始值尽量接近真实值的解，用小角度单摆的周期的 1/4 作为这个初始值是一个很不错的选择。下面利用 pendulum_period()计算出初始摆动角度从 0° 到 90° 的摆动周期：

```
ths = np.arange(0, np.pi/2.0, 0.01)
periods = [pendulum_period(1, th) for th in ths]
```

为了验证结果的正确性，可以从维基百科中找到摆动周期的精确解：

$$T = 4\sqrt{\frac{l}{g}}K(\sin\frac{\theta_0}{2})$$

其中的函数 K 为第一类完全椭圆积分函数，其定义如下：

$$K(k) = \int_0^{\pi/2} \frac{d\theta}{\sqrt{1-k^2\sin^2\theta}}$$

可以用 scipy.special.ellipk()计算此函数的值：

```
periods2 = 4*sqrt(1.0/g)*ellipk(np.sin(ths/2)**2) # 计算单摆周期的精确值
```

图 18-3 比较了这两种计算方法，我们看到它们的结果是完全一致的(见文前彩插)。

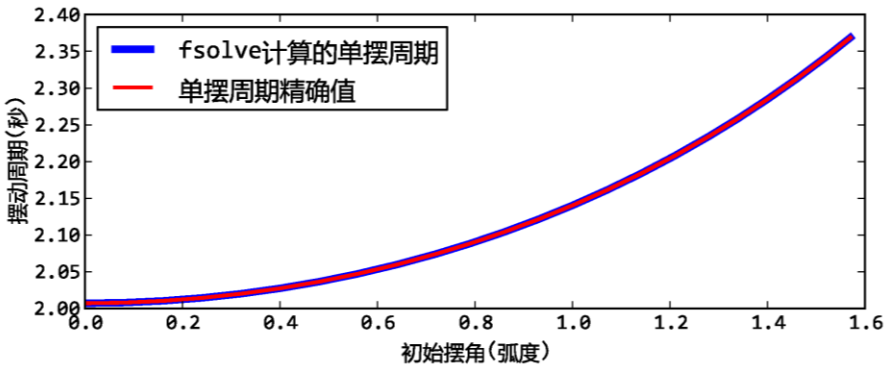

图 18-3　单摆的摆动周期和初始角度的关系

18.2　双 摆 模 拟

接下来让我们看看如何对双摆系统进行模拟。双摆系统的示意图如图 18-4 所示。两根长度为 L_1 和 L_2 的无质量细棒的顶端有质量分别为 m_1 和 m_2 的两个球，初始角度为 θ_1 和 θ_2，要求计算从此初始状态释放之后两个球的运动轨迹。

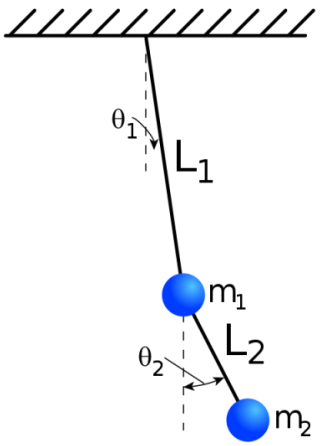

图 18-4　双摆装置示意图

18.2.1　公式推导

本节首先介绍如何利用拉格朗日力学获得双摆系统的微分方程组。

拉格朗日力学

拉格朗日力学是分析力学中的一种。它于 1788 年由拉格朗日创立，拉格朗日力学是对

经典力学的一种新的理论表述。

经典力学最初的表述形式由牛顿建立，它着重于分析位移、速度、加速度、力等矢量间的关系，又称为矢量力学。拉格朗日引入了广义坐标的概念，又运用达朗贝尔原理，求得与牛顿第二定律等价的拉格朗日方程。不仅如此，拉格朗日方程还具有更普遍的意义，适用范围更广泛。此外，选取恰当的广义坐标，可以大大地简化拉格朗日方程的求解过程。

假设细棒 L_1 连接的球体的坐标为 x_1 和 y_1，L_2 连接的球体的坐标为 x_2 和 y_2，那么 x_1、y_1、x_2、y_2 和两个角度 θ_1、θ_2 之间有如下关系：

$$x_1 = L_1 \sin(\theta_1), y_2 = -L_1 \cos(\theta_1)$$

$$x_2 = L_1 \sin(\theta_1) + L_2 \sin(\theta_2), y_2 = -L_1 \cos(\theta_1) - L_2 \cos(\theta_2)$$

根据拉格朗日力学公式：

$$\mathcal{L} = T - V$$

其中，T 为系统的动能，V 为系统的势能，可以得到如下公式：

$$\mathcal{L} = \frac{m_1}{2}(\dot{x}_1^{\,2} + \dot{y}_1^{\,2}) + \frac{m_2}{2}(\dot{x}_2^{\,2} + \dot{y}_2^{\,2}) - m_1 g y_1 - m_2 g y_2$$

其中正号的项为两个小球的动能，负号的项为两个小球的势能。

将前面的坐标和角度之间的关系公式代入并整理可得：

$$\mathcal{L} = \frac{m_1 + m_2}{2} L_1^{\,2} \dot{\theta}_1^{\,2} + \frac{m_2}{2} L_2^{\,2} \dot{\theta}_2^{\,2} + m_2 L_1 L_2 \dot{\theta}_1 \dot{\theta}_2 \cos(\theta_1 - \theta_2) + (m_1 + m_2) g L_1 \cos(\theta_1) + m_2 g L_2 \cos(\theta_2)$$

对于变量 θ_1 的拉格朗日方程：

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\theta}_2} - \frac{\partial \mathcal{L}}{\partial \theta_1} = 0$$

得到：

$$L_1[(m_1 + m_2)L_1\ddot{\theta}_1 + m_2 L_2 \cos(\theta_1 - \theta_2)\ddot{\theta}_2 + m_2 L_2 \sin(\theta_1 - \theta_2)\dot{\theta}_2^2 + (m_1 + m_2) g \sin(\theta_1)] = 0$$

对于变量 θ_2 的拉格朗日方程：

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\theta}_2} - \frac{\partial \mathcal{L}}{\partial \theta_2} = 0$$

得到：

$$m_2 L_2 [L_2\ddot{\theta}_2 + L_1 \cos(\theta_1 - \theta_2)\ddot{\theta}_1 - L_1 \sin(\theta_1 - \theta_2)\dot{\theta}_1^2 + g \sin(\theta_2)] = 0$$

这一计算过程可以用 SymPy 进行推导：

double_pendulum_solver.py
用 SymPy 推导双摆的常微分方程

```
from sympy import *
from sympy import Derivative as D

var("x1 x2 y1 y2 l1 l2 m1 m2 th1 th2 dth1 dth2 ddth1 ddth2 t g tmp")

sublist = [
(D(th1(t), t, t), ddth1),
(D(th1(t), t), dth1),
(D(th2(t), t, t), ddth2),
(D(th2(t),t), dth2),
(th1(t), th1),
(th2(t), th2)
]

x1 = l1*sin(th1(t))
y1 = -l1*cos(th1(t))
x2 = l1*sin(th1(t)) + l2*sin(th2(t))
y2 = -l1*cos(th1(t)) - l2*cos(th2(t))

vx1 = diff(x1, t)
vx2 = diff(x2, t)
vy1 = diff(y1, t)
vy2 = diff(y2, t)

# 拉格朗日力学公式
L = m1/2*(vx1**2 + vy1**2) + m2/2*(vx2**2 + vy2**2) - m1*g*y1 - m2*g*y2

# 拉格朗日方程
def lagrange_equation(L, v):
    dvt = D(v(t), t)
    a = L.subs(dvt, tmp).diff(tmp).subs(tmp, dvt) ❶
    b = L.subs(dvt, tmp).subs(v(t), v).diff(v).subs(v, v(t)).subs(tmp, dvt) ❷
    c = a.diff(t) - b
    c = c.subs(sublist)
    c = trigsimp(simplify(c))
    c = collect(c, [th1,th2,dth1,dth2,ddth1,ddth2])
    return c

eq1 = lagrange_equation(L, th1)
eq2 = lagrange_equation(L, th2)
```

```
print eq1
print eq2
```

执行此程序之后，eq1 对应于θ_1的拉格朗日方程，eq2 对应于θ_2的拉格朗日方程。

由于 SymPy 只能对符号变量求导数，即只能计算 D(L, t)，而不能计算 D(f, v(t))。❶因此在求偏导数之前，将偏导数变量置换为一个 tmp 变量，然后对 tmp 变量求导数，例如下面的程序对 D(v(t), t)求偏导数，即计算$\partial\mathcal{L}/\partial\dot{v}$：

```
L.subs(D(v(t), t), tmp).diff(tmp).subs(tmp, D(v(t), t))
```

❷而在计算$\partial\mathcal{L}/\partial v$时，需要将 v(t)替换为 v 之后再进行微分计算。由于将 v(t)替换为 v 的同时，会将 D(v(t), t)中的也进行替换，这不是我们希望的结果，因此先将 D(v(t), t)替换为 tmp，微分计算完毕之后再替换回去：

```
L.subs(D(v(t), t), tmp).subs(v(t), v).diff(v).subs(v, v(t)).subs(tmp, D(v(t), t))
```

最后得到 eq1 和 eq2 的值为：

```
>>> eq1
ddth1*(m1*l1**2 + m2*l1**2) +
ddth2*(l1*l2*m2*cos(th1)*cos(th2) + l1*l2*m2*sin(th1)*sin(th2)) +
dth2**2*(l1*l2*m2*cos(th2)*sin(th1) - l1*l2*m2*cos(th1)*sin(th2)) +
g*l1*m1*sin(th1) + g*l1*m2*sin(th1)
>>> eq2
ddth1*(l1*l2*m2*cos(th1)*cos(th2) + l1*l2*m2*sin(th1)*sin(th2)) +
dth1**2*(l1*l2*m2*cos(th1)*sin(th2) - l1*l2*m2*cos(th2)*sin(th1)) +
g*l2*m2*sin(th2) + ddth2*m2*l2**2
```

结果看上去挺复杂，其实只要运用如下的三角公式，就和前面的结果一致了：

$$\sin(x+y)=\sin x\cos y+\cos x\sin y$$
$$\cos(x+y)=\cos x\cos y-\sin x\sin y$$
$$\sin(x-y)=\sin x\cos y-\cos x\sin y$$
$$\cos(x-y)=\cos x\cos y+\sin x\sin y$$

18.2.2 微分方程的数值解

接下来要做的事情就是对如下的微分方程组求数值解：

$$(m_1+m_2)L_1\ \ddot{\theta}_1+m_2L_2\cos(\theta_1-\theta_2)\ddot{\theta}_2+m_2L_2\sin(\theta_1-\theta_2)\dot{\theta}_2^{\,2}+(m_1+m_2)g\sin(\theta_1)=0$$

$$L_2\ \ddot{\theta}_2+L_1\cos(\theta_1-\theta_2)\ddot{\theta}_1-L_1\sin(\theta_1-\theta_2)\dot{\theta}_1^{\,2}+g\sin(\theta_2)=0$$

由于方程中包含二阶导数，因此无法直接使用 odeint()进行数值求解，我们很容易将其改写为 4 个一阶微分方程组，4 个未知变量为θ_1、θ_2、v_1、v_2，其中，v_1、v_2为两个细棒转动的角速度。

$$\dot{\theta}_1 = v_1$$

$$\dot{\theta}_2 = v_2$$

$$(m_1+m_2)L_1\ \dot{v}_1 + m_2L_2\cos(\theta_1-\theta_2)\dot{v}_2 + m_2L_2\sin(\theta_1-\theta_2)\dot{\theta}_2^{\ 2} + (m_1+m_2)g\sin(\theta_1) = 0$$

$$L_2\ \dot{v}_2 + L_1\cos(\theta_1-\theta_2)\dot{v}_1 - L_1\sin(\theta_1-\theta_2)\dot{\theta}_1^{\ 2} + g\sin(\theta_2) = 0$$

下面的程序对此微分方程组进行数值求解。

double_pendulum_odeint.py
用 odeint 求解双摆系统

```
g = 9.8
class DoublePendulum(object):
    def __init__(self, m1, m2, l1, l2):
        self.m1, self.m2, self.l1, self.l2 = m1, m2, l1, l2
        self.init_status = np.array([0.0,0.0,0.0,0.0])

    def equations(self, w, t): ❶
        """
        微分方程公式
        """
        m1, m2, l1, l2 = self.m1, self.m2, self.l1, self.l2
        th1, th2, v1, v2 = w
        dth1 = v1
        dth2 = v2

        #eq of th1
        a = l1*(m1+m2)  # dv1 parameter
        b = m2*l2*cos(th1-th2) # dv2 paramter
        c = m2*l2*sin(th1-th2)*dth2*dth2 + (m1+m2)*g*sin(th1)

        #eq of th2
        d = l1*cos(th1-th2) # dv1 parameter
        e = l2 # dv2 parameter
        f = -l1*sin(th1-th2)*dth1*dth1 + g*sin(th2)

        dv1, dv2 = np.linalg.solve([[a,b],[d,e]], [-c,-f])  ❷

        return np.array([dth1, dth2, dv1, dv2])
```

❶DoublePendulum 类的 equations()用于计算各个未知函数的导数，输入参数 w 数组中的变

量依次为 th1、th2、v1 和 v2，它们分别表示上球角度、下球角度、上球角速度和下球角速度。

返回值为每个变量的导数 dth1、dth2、dv1 和 dv2，它们分别表示上球角速度、下球角速度、上球角加速度和下球角加速度。其中，dth1 和 dth2 很容易计算，它们直接等于传入的角速度变量。

为了计算 dv1 和 dv2，需要将微分方程组变形为如下格式：

$$\dot{v}_1 = ..., \dot{v}_2 = ...$$

由于两个微分方程对于$\dot{v}_1$和$\dot{v}_2$来说是两个如下形式的一次方程：

$$a\dot{v}_1 + b\dot{v}_2 + c = 0, d\dot{v}_1 + e\dot{v}_2 + f = 0$$

因此可以求出其中的系数 a、b、c、d、e、f，❷然后调用 linalg.solve()解出$\dot{v}_1$和$\dot{v}_2$。

```
def double_pendulum_odeint(pendulum, ts, te, tstep):
    """
    对双摆系统的微分方程组进行数值求解，返回两个小球的 X-Y 坐标
    """
    t = np.arange(ts, te, tstep)
    track = odeint(pendulum.equations, pendulum.init_status, t) ❸
    th1_array, th2_array = track[:,0], track[:, 1]
    l1, l2 = pendulum.l1, pendulum.l2
    x1 = l1*np.sin(th1_array)
    y1 = -l1*np.cos(th1_array)
    x2 = x1 + l2*np.sin(th2_array)
    y2 = y1 - l2*np.cos(th2_array)
    #将最后的状态赋给 pendulum
    pendulum.init_status = track[-1,:].copy() ❹
    return x1, y1, x2, y2 ❺
```

在 double_pendulum_odeint()中，❸调用 odeint()对双摆的方程组求数值解。❹将 odeint()得到的最终的小球状态保存到 pendulum.init_status 中，作为下一次调用 odeint()的初始值，因此多次调用 double_pendulum_odeint()可以生成连续的运动轨迹。❺函数返回 4 个数组，分别是两个小球的 X-Y 轴的坐标。

最后是主程序部分，我们使用小角度和大角度的初始值分别计算双摆的摆动轨迹。小初始角度时小球的运动轨迹如图 18-5 所示(见文前彩插)。大初始角度时小球的运动轨迹如图 18-6 所示(见封三彩插)。可以看出，当初始角度很大时，摆动出现混沌现象。

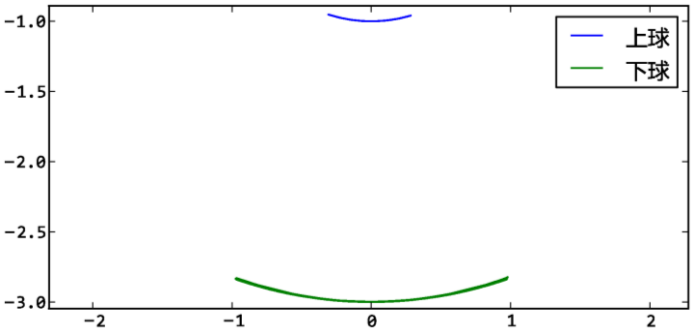

图 18-5　小初始角度时双摆的摆动轨迹

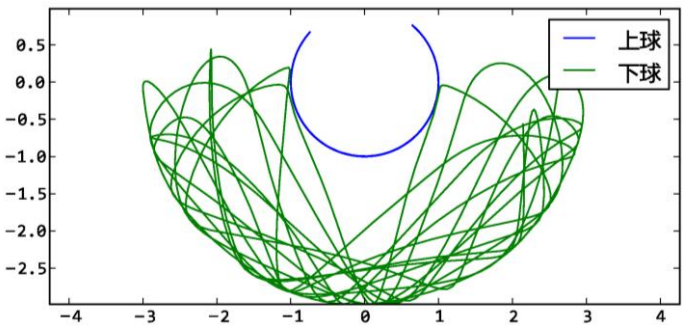

图 18-6　大初始角度时双摆的摆动轨迹呈现混沌现象

18.2.3　动画演示

有了计算双摆运动轨迹的程序之后，我们很容易使用前面介绍过的动画技术直观地演示双摆系统的运算结果。制作动画可以有多种选择：

- 使用 VPython 制作三维动画。
- 使用 Tkinter 或 wxPython 直接在界面上绘制动画。
- 使用 Enable、Chaco 或 matplotlib 制作动画。

本书提供使用 matplotlib 和 Enable 制作的双摆动画演示程序，它们调用上一节介绍的 double_pendulum_odeint()来计算双摆的运动轨迹。

double_pendulum_pylab_ani.py, double_pendulum_enable_ani.py
用 matplotlib 和 Enable 制作双摆的动画演示

使用 Enable 制作的动画演示程序的界面截图如图 18-7 所示。在此界面中，用户可以修改双摆的质量和长度，并且可以在动画框中按住并拖动鼠标，修改双摆的初始角度。

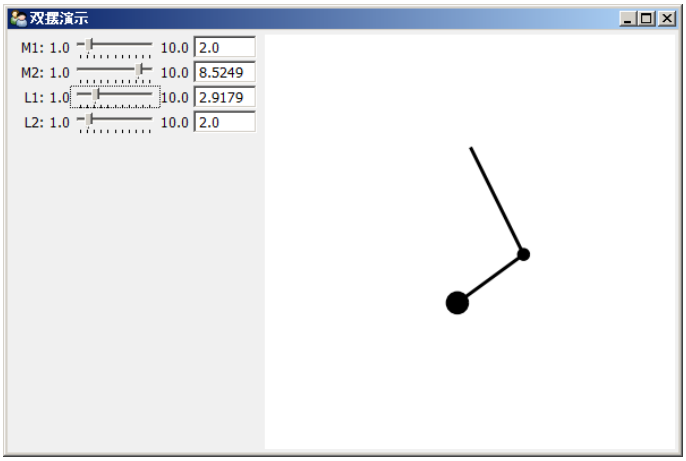

图 18-7　用 Enable 制作的双摆动画演示程序