Modeling Pendulum Motion

Sergey Voronin

February 25, 2015

Abstract

We describe here the kinematics and numerical solution of a single pendulum system. (Double pendulum simulation to be added).

1 Introduction

The pendulum is perhaps one of the most well known examples studied in classical mechanics. However, a clear exposition of analytical and numerical techniques to model the single and double pendulum systems are often difficult to find. This article attempts to present the analytical and numerical analysis of a pendulum system in an easy to understand format. This article supplements the Matlab and C with OpenGL source code which models the systems.

2 Single Pendulum

Figure below illustrates a single pendulum system. The length of the pendulum is l. It makes an angle of θ from the equilibrium position. Note that the range of θ is assumed to be in the interval $-\pi \leq \theta < \pi$. The green bon is attached to the pendulum rod. The direction of the bobs motion is along the circular arc, while the direction of the instantaneous velocity vector is along the black line with the arrow, tangent to the arc. The pendulum rod and the Bob are supposed to be massless. Both the angle and velocity of the Bob are functions of time: $\theta(t)$ and $v(t) = \frac{d\theta}{dt}(t)$. By Newton's second law:

$$F = ma \implies -mg\sin(\theta) = ma \implies a = -g\sin(\theta) \tag{2.1}$$

It remains to express the acceleration a in terms of l and θ. Let us take s to be the arc length along the circular direction of motion. Then we have that:

$$\frac{s}{2\pi l} = \frac{\theta}{2\pi} \implies s = l\theta \implies v = \frac{ds}{dt} = l\frac{d\theta}{dt} \implies a = \frac{d^2s}{dt^2} = l\frac{d^2\theta}{dt^2} \tag{2.2}$$

Substituting into (2.1), we arrive at the second order nonlinear ODE:

$$l\frac{d^2\theta}{dt^2} = -g\sin(\theta) \implies \frac{d^2\theta}{dt^2} + \frac{g}{l}\sin(\theta) = 0 \tag{2.3}$$

The initial value problem for the single pendulum system without air resistance is usually given as:

$$\frac{d^2\theta}{dt^2} + \frac{g}{l}\sin(\theta) = 0 \quad ; \quad \theta(0) = \theta_0 \quad \text{and} \quad \frac{d\theta}{dt}(0) = v_0 \tag{2.4}$$

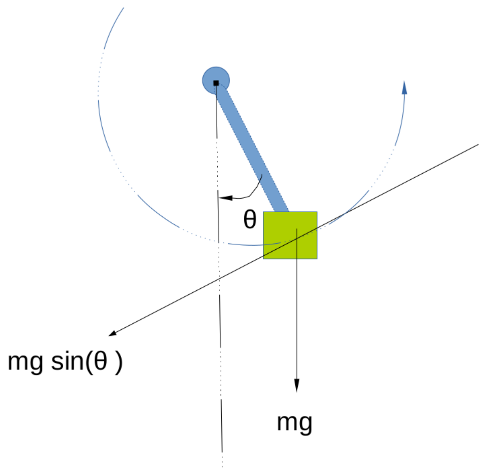


Figure 1: A single pendulum system. Pendulum moves along a circular arc.

If we like to model a damped system (modeling motion with some kind of simple air resistance scheme) we can include a term in Newton's second law equations proportional to the angular velocity:

$$F = ma \implies -\alpha v - mg\sin(\theta) = ma \implies a = -\frac{\alpha}{m}v - g\sin(\theta) \tag{2.5}$$

Plugging in for v and a in terms of θ, we obtain:

$$l\frac{d^2\theta}{dt^2} = -\frac{\alpha}{m}l\frac{d\theta}{dt} - g\sin(\theta) \implies \frac{d^2\theta}{dt^2} = -\frac{\alpha}{m}\frac{d\theta}{dt} - g\sin(\theta)$$

and the IVP:

$$\frac{d^2\theta}{dt^2} + \frac{\alpha}{m}\frac{d\theta}{dt} + \frac{g}{l}\sin(\theta) = 0 \quad ; \quad \theta(0) = \theta_0 \quad \text{and} \quad \frac{d\theta}{dt}(0) = v_0 \tag{2.6}$$

Where as before, the initial conditions give the initial displacement (angle) and velocity (directed speed) of the bob.

2.1 Numerical Modeling

We want to model (2.6). An analytical solution of (2.6) cannot be obtained in terms of elementary functions. However, it is easy to model numerically via numerical methods. We first rewrite the IVP as a first order system, by defining $p = \theta$ and $q = \theta'$:

$$\begin{aligned} \frac{dp}{dt} &= q \\ \frac{dq}{dt} &= -\frac{\alpha}{m}q - \frac{g}{l}\sin(p) \\ p(0) &= \theta_0 \quad \text{and} \quad q(0) = v_0 \end{aligned}$$

Next, we rewrite the system as a single first order ODE of a vector valued function. Letting:

$$u(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} = \begin{bmatrix} p(t) \\ q(t) \end{bmatrix} \quad \text{and} \quad f(t) = \begin{bmatrix} q(t) \\ -\frac{\alpha}{m}q(t) - \frac{g}{l}\sin(p(t)) \end{bmatrix} = \begin{bmatrix} u_2(t) \\ -\frac{\alpha}{m}u_2(t) - \frac{g}{l}\sin(u_1(t)) \end{bmatrix}$$

We get:

$$\frac{d}{dt}u(t) = f(t) \quad ; \quad u(0) = \begin{bmatrix} p(0) \\ q(0) \end{bmatrix} = \begin{bmatrix} \theta_0 \\ v_0 \end{bmatrix} \tag{2.7}$$

We can solve (2.7) using the Runge-Kutta 4th order scheme using the following sequence of steps. We first discretize a time interval, say $t_i = 0$ to $t_f = 10$ using N steps, resulting in time step $h = \frac{t_i - t_f}{N}$. Then, we perform numerical integration of the system by looping the formulas:

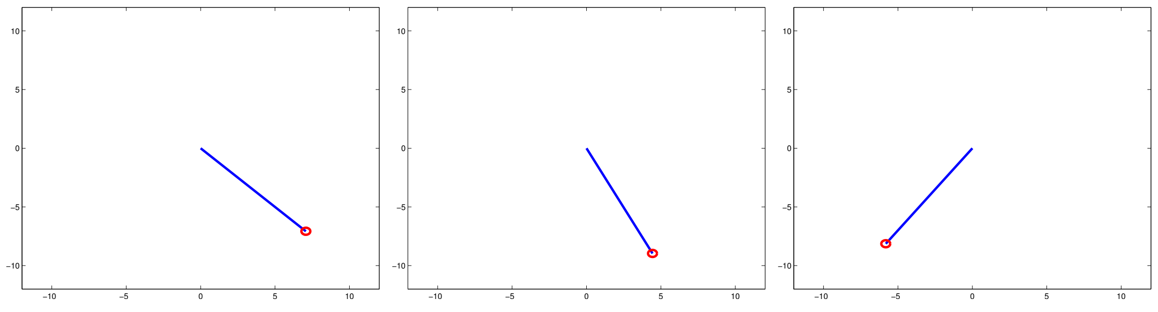
$$\begin{aligned} k_1 &= hf(t, u) \\ k_2 &= hf\left(t + \frac{h}{2}, u + \frac{k_1}{2}\right) \\ k_3 &= hf\left(t + \frac{h}{2}, u + \frac{k_2}{2}\right) \\ k_4 &= hf(t + h, u + k_3) \\ u &= u + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\ t &= t + h \end{aligned}$$

Then at each step, $u(1) \approx \theta(t)$ and $u(2) \approx v(t)$.

In order to plot the position of the pendulum as a function of time, we must convert the angles $\theta(t)$ into (x, y) coordinates on the xy-plane. We now briefly describe how the plotting is done. Note that the motion is entirely two dimensional. For simplicity we assume the pendulum rod is attached to an anchor at the origin $(0, 0)$, but can rotate freely around the anchor so that $-\pi \leq \theta < \pi$. There are four possibilities we must consider depending on which of the four quadrants the angle corresponds to. The following mapping formulas can be readily verified by making a small graph:

- If $0 \leq \theta < \frac{\pi}{2}$, then $x = l\cos(\frac{\pi}{2} - \theta)$ and $y = -l\sin(\frac{\pi}{2} - \theta)$.
- If $\frac{\pi}{2} \leq \theta < \pi$, then $x = l\cos(\theta - \frac{\pi}{2})$ and $y = l\sin(\theta - \frac{\pi}{2})$.
- If $-\frac{\pi}{2} \leq \theta < 0$, then $x = -l\cos(\frac{\pi}{2} - |\theta|)$ and $y = -l\sin(\frac{\pi}{2} - |\theta|)$.
- If $-\pi \leq \theta < -\frac{\pi}{2}$, then $x = -l\cos(|\theta| - \frac{\pi}{2})$ and $y = l\sin(|\theta| - \frac{\pi}{2})$.

The pendulum motion is very predictable. Starting with no initial velocity and from an angle of $\frac{\pi}{4}$ we get motion similar to below:



The OpenGL version of the simulation is written with the standard Free Glut library. The C code is implemented similar to the Matlab code, except it is easier to treat the two components of the function f separately in the RK4 loop code. Essentially, the following code can be used for the ODE integration:

```
    theta = theta0;
    v = v0;
    for(i = 0; i<N; i++){
        k11 = h*v;
        k12 = -h*(alpha/m)*v - h*(g/l)*sin(theta);

        k21 = h*(v + k11/2);
        k22 = -h*(alpha/m)*(v + k12/2) - h*(g/l)*sin(theta + k12/2);

        k31 = h*(v + k21/2);
        k32 = -h*(alpha/m)*(v + k22/2) -h*(g/l)*sin(theta + k22/2);

        k41 = h*(v + k31);
        k42 =  -h*(alpha/m)*(v + k32) - h*(g/l)*sin(theta + k32);

        theta = theta + (k11 + 2*k21 + 2*k31 + k41)/6;
        v = v + (k12 + 2*k22 + 2*k32 + k42)/6;
    }
```

Note that in the simple classical case of this problem f does not depend on t. The OpenGL graphics part is simple: we compute an array of angles and velocities and then convert angles into coordinates as before. We call the *display()* function each time we loop though the $\theta(t)$ array in order to plot a sequence of steps of the pendulum motion.