THE SPHERICAL PENDULUM

DERIVING THE EQUATIONS OF MOTION

The spherical pendulum is similar to the simple pendulum, but moves in 3-dimensional space. This means we need to introduce a new variable φ in order to describe the rotation of the pendulum around the z-axis. We can then describe the position of the pendulum in reference to the variables θ and φ, and so the system has 2 degrees of freedom.

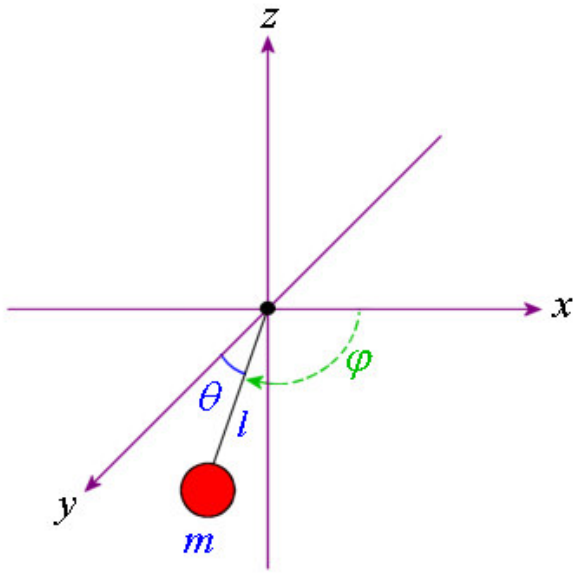

Figure 1: The Spherical Pendulum

In order to describe this system with the new variable φ, we use spherical polar coordinates:

$$x = l\sin(\theta)\cos(\varphi)$$
$$y = l\sin(\theta)\sin(\varphi)$$
$$z = l\cos(\theta)$$

Now, as with the double pendulum, we need to find the Lagrangian of the system.

Remember that:

$$L = T - U$$

Where (in this case):

$$T = \frac{m}{2}\left(\dot{x}^2 + \dot{y}^2 + \dot{z}^2\right)$$

$$U = -mgz$$

Thus in order to find T, we start by calculating:

$$\dot{x} = l\dot{\theta}\cos(\theta)\cos(\varphi) - l\dot{\varphi}\sin(\theta)\sin(\varphi)$$
$$\dot{y} = l\dot{\theta}\cos(\theta)\sin(\varphi) + l\dot{\varphi}\sin(\theta)\cos(\varphi)$$
$$\dot{z} = -l\dot{\theta}\sin(\theta)$$

So:

$$\dot{x}^2 = l^2\dot{\theta}^2\cos^2(\theta)\cos^2(\varphi) - 2l^2\dot{\theta}\dot{\varphi}\sin(\theta)\cos(\theta)\sin(\varphi)\cos(\varphi) + l^2\dot{\varphi}^2\sin^2(\theta)\sin^2(\varphi)$$
$$\dot{y}^2 = l^2\dot{\theta}^2\cos^2(\theta)\sin^2(\varphi) + 2l^2\dot{\theta}\dot{\varphi}\sin(\theta)\cos(\theta)\sin(\varphi)\cos(\varphi) + l^2\dot{\varphi}^2\sin^2(\theta)\cos^2(\varphi)$$
$$\dot{z}^2 = l^2\dot{\theta}^2\sin^2(\theta)$$

And therefore:

$$T = \frac{m}{2}\begin{pmatrix} l^2\dot{\theta}^2\cos^2(\theta)\cos^2(\varphi) - 2l^2\dot{\theta}\dot{\varphi}\sin(\theta)\cos(\theta)\sin(\varphi)\cos(\varphi) + l^2\dot{\varphi}^2\sin^2(\theta)\sin^2(\varphi) + \\ l^2\dot{\theta}^2\cos^2(\theta)\sin^2(\varphi) + 2l^2\dot{\theta}\dot{\varphi}\sin(\theta)\cos(\theta)\sin(\varphi)\cos(\varphi) + l^2\dot{\varphi}^2\sin^2(\theta)\cos^2(\varphi) + \\ l^2\dot{\theta}^2\sin^2(\theta) \end{pmatrix}$$

Simplifying we get*:*

$$T = \frac{m}{2}\left(l^2\dot{\theta}^2 + l^2\dot{\varphi}^2\sin^2(\theta)\right)$$

And we know:

$$U = -mgz = -mgl\cos(\theta)$$

So the Lagrangian is:

$$L = \frac{m}{2}\left(l^2\dot{\theta}^2 + l^2\dot{\varphi}^2\sin^2(\theta)\right) + mgl\cos(\theta) \qquad \textbf{(1)}$$

Using the property **(1)** from the documentation for the double pendulum, we can solve for $\ddot{\theta}$ by using:

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

Where:

$$\frac{\partial L}{\partial \theta} = \dot{\varphi}^2 ml^2 \sin(\theta)\cos(\theta) - mgl\sin(\theta)$$

$$\frac{\partial L}{\partial \dot{\theta}} = ml^2 \dot{\theta}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) = ml^2 \ddot{\theta}$$

We get:

$$ml^2 \ddot{\theta} - \dot{\varphi}^2 ml^2 \sin(\theta)\cos(\theta) + mgl\sin(\theta) = 0$$

And rearranging for $\ddot{\theta}$ gives:

$$\ddot{\theta} = \frac{l\dot{\varphi}^2 \sin(\theta)\cos(\theta) - g\sin(\theta)}{l} \qquad \textbf{(2)}$$

Similarly, in order to find the equation we need for $\ddot{\varphi}$ we use:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\varphi}}\right) - \frac{\partial L}{\partial \varphi} = 0$$

Where:

$$\frac{\partial L}{\partial \varphi} = 0$$

$$\frac{\partial L}{\partial \dot{\varphi}} = \dot{\varphi}\, ml^2 \sin^2(\theta)$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\varphi}}\right) = \ddot{\varphi}\, ml^2 \sin^2(\theta) + 2\dot{\varphi}\, ml^2 \dot{\theta}\sin(\theta)\cos(\theta)$$

So:

$$\ddot{\varphi}\, ml^2 \sin^2(\theta) + 2\dot{\varphi}\, ml^2 \dot{\theta}\sin(\theta)\cos(\theta) = 0$$

And rearranging for $\ddot{\varphi}$ gives:

$$\ddot{\varphi} = \frac{-2\dot{\varphi}\dot{\theta}\cos(\theta)}{\sin(\theta)} \qquad \textbf{(3)}$$

PROGRAMMING THE JAVA APPLET

I realised at this point that there could well be a problem with using these equations in the applet code even after I had integrated them numerically – the '$\sin(\theta)$' in the denominator of the equation for $\ddot{\varphi}$ would mean that $\ddot{\varphi}$ would tend to infinity whenever $\theta \approx n\pi$.

I tried implementing the geometric method to integrate equations **(2)** and **(3)** and programmed it into a basic applet to see exactly how it would react; and as predicted, whenever the angle θ neared $n\pi$, $\dot{\varphi}$ tended to infinity, and obviously the motion of the pendulum was ruined.

To double check that this was not just a problem with the geometric integration method I wrote a small program to calculate the fourth-order Runge-Kutta method within the applet. However, as could be expected, this made little difference to the pendulum's behaviour.

The next possible solution I tried was using 'if' statements within the applet code to ignore values of θ that were close to $n\pi$. I found that although it was possible to stop the pendulum from swinging wildly by doing this, the motion of the pendulum was erratic due to the alterations of the way the motion was calculated.

Another solution I tried involved using the conservation of angular momentum together with a new half-step numerical integration method. The necessary inputs for this new method are found by using the following properties (see José and Saletan, 2002):

$$P_i = \frac{\partial L}{\partial \dot{q}_i} \qquad \textbf{(4)}$$

$$\dot{P}_i = \frac{\partial L}{\partial q_i} \qquad \textbf{(5)}$$

Where P_i is the generalised momentum with respect to q_i.

Thus, the angular momentum P_φ is found by using the Lagrangian **(1)** together with the equation **(4)**:

$$P_\varphi = \frac{\partial L}{\partial \dot{\varphi}} = ml^2\dot{\varphi}\sin^2(\theta) \qquad \textbf{(6)}$$

Also note that using equation **(5)** it can be confirmed that P_φ is constant:

$$\dot{P}_\varphi = 0$$

Next we find an equation for P_θ using **(4)**:

$$P_\theta = \frac{\partial L}{\partial \dot{\theta}} = ml^2\dot{\theta}$$

So therefore:

$$\dot{\theta} = \frac{P_\theta}{ml^2} \tag{7}$$

Finally we find $\dot{P}_\theta$ using **(5)**:

$$\dot{P}_\theta = \frac{\partial L}{\partial \theta} = \dot{\varphi}^2 ml^2 \sin(\theta)\cos(\theta) - mgl\sin(\theta) \tag{8}$$

The next step is to rearrange equation **(6)** to find $\dot{\varphi}$:

$$\dot{\varphi} = \frac{P_\varphi}{ml^2 \sin^2(\theta)} \tag{9}$$

Then we substitute equation **(9)** into equation **(8)**:

$$\dot{P}_\theta = \frac{{P_\varphi}^2 ml^2 \sin(\theta)\cos(\theta)}{m^2 l^4 \sin^4(\theta)} - mgl\sin(\theta)$$

Simplifying:

$$\dot{P}_\theta = \frac{{P_\varphi}^2 \cos(\theta)}{ml^2 \sin^3(\theta)} - mgl\sin(\theta) \tag{10}$$

Using the equations **(7)**, **(9)**, and **(10)**, I created an applet that used the geometric integration method, but using half-steps for P_θ (see Leimkuhler and Reich (2000), unpublished), and solving a set of first order ordinary differential equations as opposed to the second order equations I had been integrating up until this point. This method is shown over the page.

$$P_\theta^{\,n+\frac{1}{2}} = P_\theta^{\,n} + \frac{\Delta t}{2}\left(\frac{P_\varphi^{\,2}\cos(\theta^n)}{ml^2\sin^3(\theta^n)} - mgl\sin(\theta^n)\right)$$

$$\theta^{n+1} = \theta^n + \Delta t\left(\frac{P_\theta^{\,n+\frac{1}{2}}}{ml^2}\right)$$

$$P_\theta^{\,n+1} = P_\theta^{\,n+\frac{1}{2}} + \frac{\Delta t}{2}\left(\frac{P_\varphi^{\,2}\cos(\theta^{n+1})}{ml^2\sin^3(\theta^{n+1})} - mgl\sin(\theta^{n+1})\right)$$

$$\varphi^{n+1} = \varphi^n + \Delta t\left(\frac{P_\varphi}{ml^2\sin^2(\theta^{n+1})}\right)$$

Since P_φ is a constant, it was possible to calculate it from the initial conditions (θ at time $t = 0$, and $\dot{\varphi}$ at time $t = 0$) before the numerical integration was employed in the code.

Whilst attempting to solve the problem with the numerical integration, I was also working to improve the 3-dimensional graphical aspects of the applet.

I began by using the same program structure as the 2 previous applets – solving for θ and φ first; then calculating the coordinates of the bob; and finally plotting them.

The obvious fault with using this method as before, was that I had originally only taken 2 dimensions into account, so there was no sense of perspective.

I decided to tackle the problem of perspective by thinking about how a 3-dimensional pendulum would change as you looked at it – both the length of the rod and the size of the bob would alter dynamically depending on the angle at which you are looking at it, and how far away the bob is from your eye.

Using the spherical polar coordinate system to calculate the position of the bob solved the problem of the dynamically changing rod length:

$$\begin{aligned} x &= l'\sin(\theta)\cos(\varphi) \\ y &= l'\sin(\theta)\sin(\varphi) \\ z &= l'\cos(\theta) \end{aligned} \qquad \textbf{(11)}$$

Note that l' denotes the maximum visual length of the rod, and is measured in pixels.

By using this system, it was possible to plot the pendulum in the $x-z$ plane (i.e. plot x against z, and imagine the y-axis coming out of the computer screen) and the visual length of the pendulum rod would vary according to the coordinate system.

Since the pendulum was plotted in the $x-z$ plane, only the y coordinate of the bob had any effect on how far away from the user the bob should appear. Thus it made sense to create a linear function of y to determine the visual size of the bob – the larger the y-value, the closer to the user the bob is, and so the larger the bob is, and vice versa.

Unfortunately, due to the way a circle (used to draw the bob) is plotted in Java, it was impossible to create such a generalised function. The circle size is determined by an integer pixel radius, and since the normal radius of the bob was only 5 pixels to begin with, it did not leave much scope for altering the size of the bob.

After some brief investigation, it became apparent that in order to keep the size of the bob sensible, 4 different bob radii could be used. Thus, I created 4 areas of y, with the idea that the bob would change size each time it moved into a different area.

The next step was to determine which area the bob was in. Obviously use of the y-coordinate of the bob was needed to do this, and since the maximum value of y occurs when $\sin(\theta)=\sin(\varphi)=1$ (see equation **(11)**), then the maximum value of y is $y_{MAX}=l'$. Similarly, the minimum value of y is $y_{MIN}=-l'$. Therefore, the range of the y-coordinate of the bob is $l'\geq y\geq -l'$.

Since there are 4 possible sizes for the bob, it made sense to split this range into 4 to give the aforementioned 'areas' for each bob size:

Range	$l'\leq y\leq \frac{l'}{2}$	$\frac{l'}{2}\leq y\leq 0$	$0<y\leq -\frac{l'}{2}$	$-\frac{l'}{2}<y\leq -l'$
Area	**1**	**2**	**3**	**4**

Figure 2: Ranges of y and their Corresponding Areas

It is now possible to see that when the y-coordinate of the bob is in **Area 1**, the bob is at its closest to the user and so the radius of the bob should be at its maximum value. Similarly, when the bob is in **Area 4**, the bob is at its farthest from the user and so the radius of the bob should be at its minimum. Using this logic, it was simple to create a function to use the y-coordinate of the bob (obtained by using equation **(11)**) to detect which area the bob was in, and set the radius of the bob to the appropriate value.

This function gave the applet the illusion of perspective, but I found it was still difficult to visualise exactly what the pendulum was doing – it was very easy to become disorientated.

To solve this problem I created an 'if' statement to determine whether the y-coordinate was positive or negative; and then plotted a set of axes onto the applet. The axes were plotted behind the pendulum if the y-coordinate was positive, or in front of the pendulum if it was negative. This gave the impression that the pendulum was moving in front of or behind the axes dependent on the y-coordinate.

With this done, it was easy to visualise the motion of the pendulum.