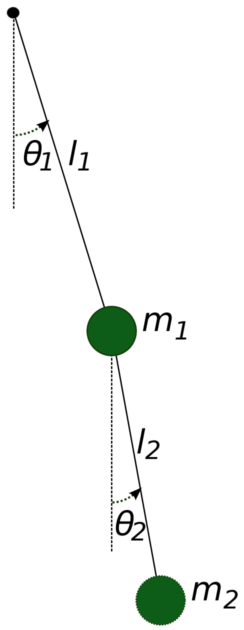
Simulating the double pendulum

R. A. Blythe

January 6, 2010

1 Derivation of the equations of motion



The double pendulum has a mass m_1 attached to the origin by a string of fixed length ℓ_1; attached to this is a second mass m_2 by a string of length ℓ_2. We restrict all initial positions and velocities such that they lie in a single plane with the origin, see figure. There are various ways to obtain equations of motion for the double pendulum. We will use a Lagrangian approach.

Our aim is to write down differential equations involving the angles θ_1 and θ_2 that the two pendulum bobs make with the vertical. First, we write down the positions of the masses in Cartesian coordinates:

$$\mathbf{r}_1 = (\ell_1 \sin\theta_1, -\ell_1 \cos\theta_1) \tag{1}$$

$$\mathbf{r}_2 = (\ell_1 \sin\theta_1 + \ell_2 \sin\theta_2, -\ell_1 \cos\theta - \ell_2 \cos\theta_2) \ . \tag{2}$$

Now we differentiate to get velocities:

$$\mathbf{v}_1 = (\ell_1 \dot\theta_1 \cos\theta_1, \ell_1 \dot\theta_1 \sin\theta_1) \tag{3}$$

$$\mathbf{v}_2 = (\ell_1 \dot\theta_1 \cos\theta_1 + \ell_2 \dot\theta_2 \cos\theta_2, \ell_1 \dot\theta_1 \sin\theta_1 + \ell_2 \dot\theta_2 \sin\theta_2) \ . \tag{4}$$

Here dots denote derivatives with respect to time.

The kinetic energy of the system can now be written as

$$\begin{aligned} T &= \frac{1}{2} m_1 \mathbf{v}_1 \cdot \mathbf{v}_1 + \frac{1}{2} m_2 \mathbf{v}_2 \cdot \mathbf{v}_2 & (5)\\ &= \frac{1}{2} m_1 \ell_1^2 \dot\theta_1^2 + \frac{1}{2} m_2 \left[\ell_1^2 \dot\theta_1^2 + 2\ell_1 \ell_2 \dot\theta_1 \dot\theta_2 \left(\cos\theta_1 \cos\theta_2 + \sin\theta_1 \sin\theta_2 \right) + \ell_2^2 \dot\theta_2^2 \right] & (6)\\ &= \frac{1}{2} m_1 \ell_1^2 \dot\theta_1^2 + \frac{1}{2} m_2 \left[\ell_1^2 \dot\theta_1^2 + 2\ell_1 \ell_2 \dot\theta_1 \dot\theta_2 \cos(\theta_1 - \theta_2) + \ell_2^2 \dot\theta_2^2 \right] \ , & (7) \end{aligned}$$

where we have used the trigonometric identity $\cos(A - B) = \cos A \cos B + \sin A \sin B$. It is this interaction term $\cos(\theta_1 - \theta_2)$ that makes the double pendulum analytically intractable, and why we turn to a computer to simulate it.

The potential energy is more straightforward:

$$\begin{aligned} V &= m_1 g \mathbf{r}_1 \cdot \hat{\mathbf{y}} + m_2 g \mathbf{r}_2 \cdot \hat{\mathbf{y}} & (8)\\ &= -m_1 g \ell_1 \cos\theta_1 - m_2 g \left(\ell_1 \cos\theta_1 + \ell_2 \cos\theta_2 \right) & (9) \end{aligned}$$

in which $\hat{\mathbf{y}}$ is the unit vector in the y-direction (vertically upwards) and g is the acceleration due to gravity.

We can now write down the Lagrangian $\mathcal{L} = T - V$, which is a function of the angles θ_i and angular velocities $\dot{\theta}_i$ (where $i = 1, 2$). The equations of motion are then obtained by applying the Euler-Lagrange equation

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \mathcal{L}}{\partial \dot{\theta}_i}\right) = \frac{\partial \mathcal{L}}{\partial \theta_i} \tag{10}$$

for $i = 1, 2$. The key thing to remember here is that θ_i and $\dot{\theta}_i$ are considered to be independent variables, but that both depend on time (so we will need to use the chain rule when doing the time derivative).

For the case $i = 1$, we find

$$\frac{\mathrm{d}}{\mathrm{d}t}\left[(m_1 + m_2)\ell_1^2\dot{\theta}_1 + m_2\ell_1\ell_2\dot{\theta}_2\cos(\theta_1 - \theta_2)\right] = -m_2\ell_1\ell_2\dot{\theta}_1\dot{\theta}_2\sin(\theta_1 - \theta_2) - (m_1 + m_2)g\ell_1\sin\theta_1 \tag{11}$$

which, after performing the differentiation and rearranging, becomes

$$(m_1 + m_2)\ell_1^2\ddot{\theta}_1 + m_2\ell_1\ell_2\ddot{\theta}_2\cos(\theta_1 - \theta_2) = -m_2\ell_1\ell_2\dot{\theta}_2^2\sin(\theta_1 - \theta_2) - (m_1 + m_2)g\ell_1\sin\theta_1\ . \tag{12}$$

Likewise, for $i = 2$, we find first

$$\frac{\mathrm{d}}{\mathrm{d}t}\left[m_2\ell_1\ell_2\dot{\theta}_1\cos(\theta_1 - \theta_2) + m_2\ell_2^2\dot{\theta}_2\right] = m_2\ell_1\ell_2\dot{\theta}_1\dot{\theta}_2\sin(\theta_1 - \theta_2) - m_2 g\ell_2\sin\theta_2 \tag{13}$$

and then

$$m_2\ell_1\ell_2\ddot{\theta}_1\cos(\theta_1 - \theta_2) + m_2\ell_2^2\ddot{\theta}_2 = m_2\ell_1\ell_2\dot{\theta}_1^2\sin(\theta_1 - \theta_2) - m_2 g\ell_2\sin\theta_2\ . \tag{14}$$

Equations (12) and (14) constitute the equations of motion for the system.

2 Expression as a system of first-order ordinary differential equations

Equation (12) and (14) are a set of *coupled*, *second-order* differential equations. Computational algorithms usually require the system to be expressed as a set of *first-order* equations of the form

$$\frac{\mathrm{d}}{\mathrm{d}t}y_i = f_i(y_1, y_2, \ldots, y_n, t) \tag{15}$$

for each variable y_i, $i = 1, 2, \ldots, n$ depending on time t. That is, each time derivative is expressed as some (possibly nonlinear) combination of the variables y_i that we want to solve for.

To turn the second-order into first-order equations we use a standard trick, which is to replace $\dot{\theta}_i$ with ω_i, and hence $\ddot{\theta}_1 = \dot{\omega}_i$. Then, all the second derivatives in (12) and (14) disappear—but we know have four unknowns to solve for:

$$y_1 = \theta_1\ , \quad y_2 = \theta_2\ , \quad y_3 = \omega_1 \quad \text{and} \quad y_4 = \omega_2\ . \tag{16}$$

The first two f_i functions are easy to identify:

$$\frac{\mathrm{d}}{\mathrm{d}t}y_1 = \dot{\theta}_1 = \omega_1 = y_3 \quad \Longrightarrow \quad f_1(y_1, y_2, y_3, y_4, t) = y_3 \tag{17}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}y_2 = \dot{\theta}_2 = \omega_2 = y_4 \quad \Longrightarrow \quad f_2(y_1, y_2, y_3, y_4, t) = y_4\ . \tag{18}$$

The remaining two are a little harder. We obtain them by noting that the structure of the equations (12) and (14) is

$$A\dot{\omega}_1 + B\dot{\omega}_2 = E \tag{19}$$

$$C\dot{\omega}_1 + D\dot{\omega}_2 = F \tag{20}$$

where A, B, C, D, E and F are various combinations of θ_i and ω_i (i.e., the y_i). Solving this linear system yields the expressions

$$\frac{\mathrm{d}}{\mathrm{d}t} y_3 = \dot{\omega}_1 = \frac{DE - BF}{AD - BC} \tag{21}$$

$$\frac{\mathrm{d}}{\mathrm{d}t} y_4 = \dot{\omega}_2 = \frac{AF - CE}{AD - BC} \; . \tag{22}$$

The quantities $A, B, \ldots, F$ can now be obtained by inspecting Eqs. (12) and (14). Before we do, however, it is useful to introduce the three parameters

$$\alpha = \frac{m_2}{m_1} \; , \quad \beta = \frac{\ell_2}{\ell_1} \quad \text{and} \quad \gamma = \frac{g}{\ell_1} \; . \tag{23}$$

Then, (12) and (14) can be written as

$$[1+\alpha]\,\dot{\omega}_1 + [\alpha\beta\cos(\theta_1 - \theta_2)]\,\dot{\omega}_2 = \left[-(1+\alpha)\gamma\sin\theta_1 - \alpha\beta\omega_2^2\sin(\theta_1-\theta_2)\right] \tag{24}$$

$$[\cos(\theta_1-\theta_2)]\,\dot{\omega}_1 + [\beta]\,\omega_2 = \left[-\gamma\sin\theta_2 + \omega_1^2\sin(\theta_1-\theta_2)\right] \tag{25}$$

The square brackets enclose the expressions that need to be substituted into (21) and (22). We ultimately find

$$f_3(y_1, y_2, y_3, y_4, t) = - \frac{(1+\alpha)\gamma\sin y_1 + \alpha\beta y_4^2 \sin(y_1-y_2) + \alpha\cos(y_1-y_2)\left[y_3^2\sin(y_1-y_2) - \gamma\sin y_2\right]}{1+\alpha\sin^2(y_1-y_2)} \tag{26}$$

$$f_4(y_1, y_2, y_3, y_4, t) = \frac{(1+\alpha)\left[y_3^2\sin(y_1-y_2) - \gamma\sin y_2\right] + \cos(y_1-y_2)\left[(1+\alpha)\gamma\sin y_1 + \alpha\beta y_4^2\sin(y_1-y_2)\right]}{\beta[1+\alpha\sin^2(y_1-y_2)]} \tag{27}$$

3 Runge-Kutta algorithm for solving a set of first-order ordinary differential equations

The Runge-Kutta algorithm is a method for solving a set of first-order ordinary differential equations, and works by estimating the value of some function $y(t+h)$ given its value at $y(t)$, and evaluating derivatives at the initial time point t *and* at intermediate points. This way, the error in the estimate of $y(t+h)$ is of order h^5 rather h^2 as it is with simpler schemes.

The algorithm can be expressed mathematically as follows. We compute four n-dimensional vectors (where n is the number of variables to solve for) $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ and $\mathbf{d}$ whose elements are the derivatives of y_i, that is the functions $f_1, \ldots, f_n$ discussed above, evaluated at various points:

$$a_i = h f_i(y_1, \ldots, y_n, t) \tag{28}$$

$$b_i = h f_i(y_1 + \frac{1}{2}a_1, \ldots, y_n + \frac{1}{2}a_n, t + \frac{1}{2}h) \tag{29}$$

$$c_i = h f_i(y_1 + \frac{1}{2}b_1, \ldots, y_n + \frac{1}{2}b_n, t + \frac{1}{2}h) \tag{30}$$

$$d_i = h f_i(y_1 + c_1, \ldots, y_n + c_n, t + h) \; . \tag{31}$$

The estimate of $y_i(t+h)$ is then given as

$$y_i(t+h) = y_i(t) + \frac{a_i}{6} + \frac{b_i}{3} + \frac{c_i}{3} + \frac{d_i}{6} + \mathrm{O}(h^5) \; . \tag{32}$$

It is worth spending a few moments working out how to code this efficiently in a computer program, bearing in mind that computers don't work the same way that humans (or mathematicians) do. We need a sequence of steps that performs the sum (32), ideally using as few intermediate arrays as possible (since creation of arrays is a costly enterprise).

Suppose the current state of the system (the y_i values at time t) is stored in an array called `state`. At the end of one iteration, we want `state` to be updated so it contains the estimate of the y_i values at time $t + h$, as given by (32). We can do this with the help of three further arrays, `midpoint`, to hold the midpoint positions that we will evaluate the derivatives at, `derivatives` to hold the derivatives themselves and `previous`, to keep a copy of the state of the system at time t as we are updating it. Then, the algorithm can be implemented as

- Copy the array `state` into `previous` and `midpoint`.
- Evaluate the derivatives at `midpoint` and time t, and store the results in `derivatives`.
- Modify each element of `state` by adding to it $\frac{h}{6}$ multiplied by the corresponding element of `derivatives`.
- Set up the next `midpoint` by setting each of its elements to the corresponding element of `previous` plus $\frac{h}{2}$ multiplied by the corresponding element of `derivatives`.
- Evaluate the derivatives at `midpoint` and time $t + \frac{h}{2}$, and store the results in `derivatives`.
- Modify each element of `state` by adding to it $\frac{h}{3}$ multiplied by the corresponding element of `derivatives`.
- Set up the next `midpoint` by setting each of its elements to the corresponding element of `previous` plus $\frac{h}{2}$ multiplied by the corresponding element of `derivatives`.
- Evaluate the derivatives at `midpoint` and time $t + \frac{h}{2}$, and store the results in `derivatives`.
- Modify each element of `state` by adding to it $\frac{h}{3}$ multiplied by the corresponding element of `derivatives`.
- Set up the final `midpoint` by setting each of its elements to the corresponding element of `previous` plus h multiplied by the corresponding element of `derivatives`.
- Evaluate the derivatives at `midpoint` and time $t + h$, and store the results in `derivatives`.
- Modify each element of `state` by adding to it $\frac{h}{6}$ multiplied by the corresponding element of `derivatives`.

You should check that after performing this sequence of steps, the array `state` contains the values $y_i(t + h)$ as given by Equation (32).