MXB103 Project: Bungee!

Due: 11:59pm Friday of Week 13 (25 October)

Contents

1 Background

As part of Brisbane's "New World City" transformation, the Brisbane City Council is proposing to allow bungee jumping off the Story Bridge. A commercial bungee jump company has expressed interest, and provided Council with facts and figures concerning the proposal. Your group has been hired as consultants to verify certain aspects of this information, and to answer several key questions that have been raised about the proposal.

To answer these questions you will develop a mathematical model of the bungee jumping process, solve it numerically using MATLAB, and prepare a report with your results to present to Council. As the proposal is still only in preliminary stages, the scope of this report is limited to investigating a few key aspects of the proposal (as will be outlined below).

2 Model

The proposal calls for a platform to be installed at the very top of the bridge, from which the bungee jumps will take place. Suppose this platform is at height H from the water level. Now, let y represent the distance the jumper has *fallen*. Hence, $y = 0$ corresponds to the platform, and y increases as the jumper falls towards the river. Later, we will also be interested in the height of the deck of the bridge, which will be distance D from the water level.

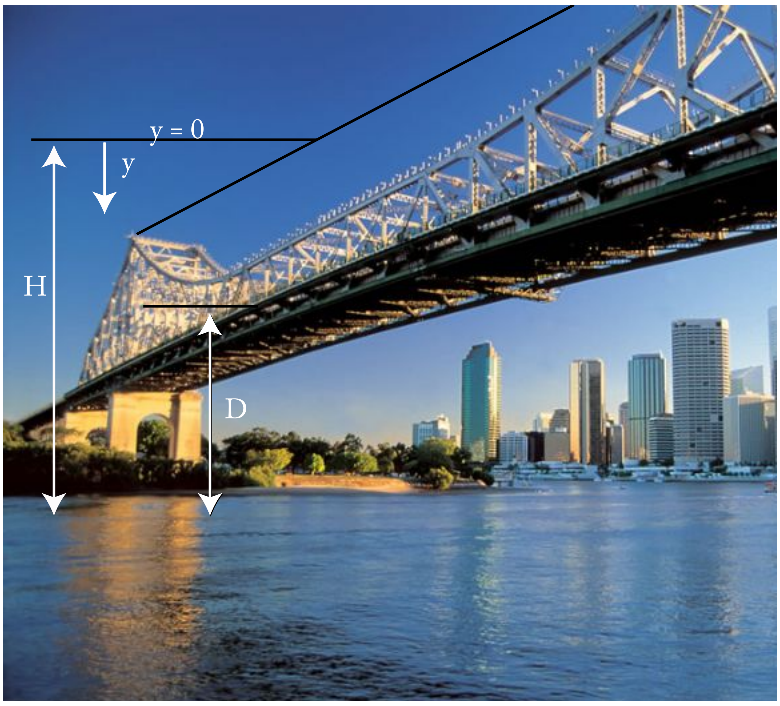


Figure 1: Illustration of platform height H, deck height D and jumper's position y, where the positive direction of y is downwards.

The mathematical model of bungee jumping can be derived by considering the forces acting on the jumper at all times. There are three forces we must consider:

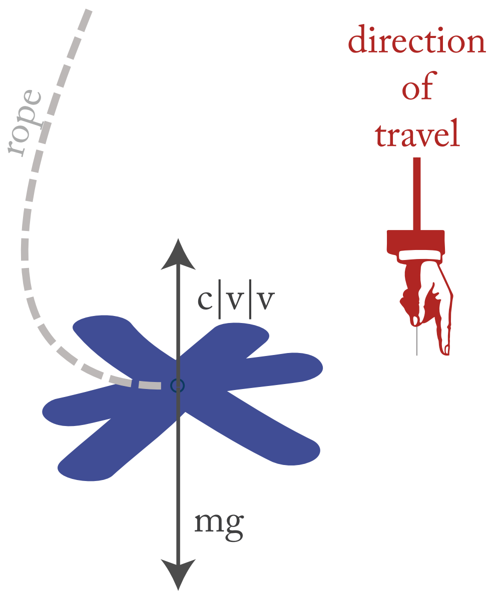
1. Gravity
2. Drag (also called air resistance): the frictional force due to the air
3. Tension: the pull of the bungee rope when it is taut

As Figure 2 (next page) illustrates, there are four different parts of the jump: falling with a slack bungee rope, falling with a taut bungee rope, upward bounce with a taut bungee rope, and upward bounce with a slack bungee rope (after which the cycle repeats). Depending on which part of the jump we consider, some of the above three forces may change direction, or be absent completely.

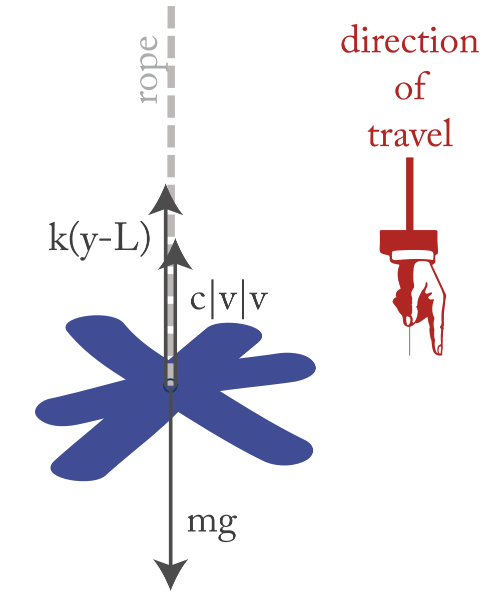
The simplest force to model is gravity: it always acts downwards, and its value is given by mg, where m is the mass of the jumper, and g is the gravitational acceleration.

The next force we consider is drag. This force always acts in the opposite direction to motion, so that it is always slowing the jumper down. Its value is given by $-c|v|v$, where c is the drag coefficient, and v is the velocity of the jumper. Notice that the strength of the drag force is proportional to the square of the velocity. The absolute value sign on the first factor of v ensures it always acts in the opposite direction to motion.

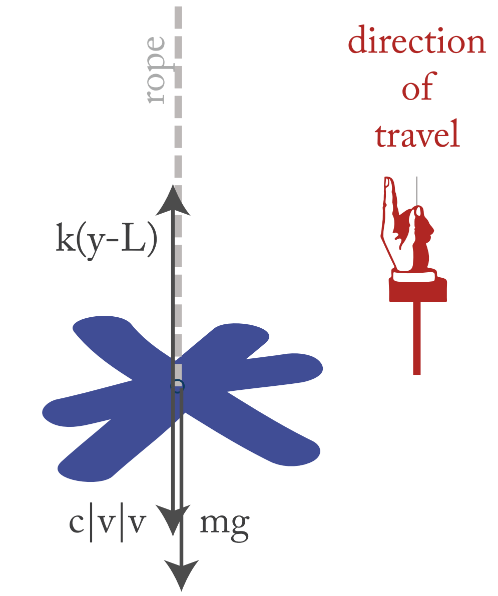
The final force we must consider is tension. When the bungee rope is taut, it exerts a force on the jumper proportional to how much it has been stretched (this is called *Hooke's law*). This force always acts upwards, pulling the jumper back up. If we let the length of the *unstretched*



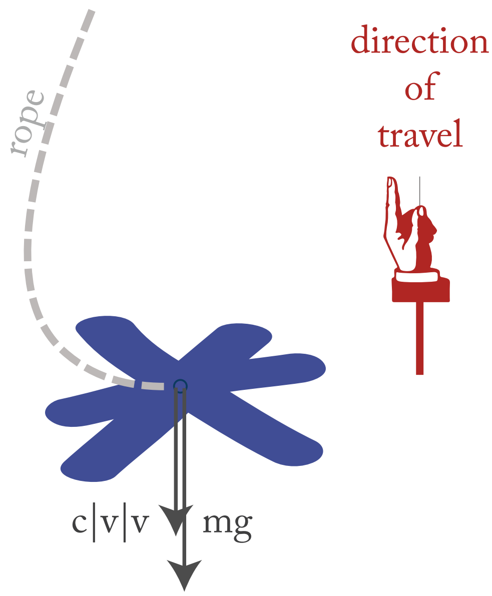
(a) Downward fall, bungee rope slack



(b) Downward fall, bungee rope taut



(c) Upward bounce, bungee rope taut



(d) Upward bounce, bungee rope slack

Figure 2: The four stages of a bungee jump and the forces acting on the jumper.

rope be L, then the tension when the rope is taut is given by $k(y - L)$, where k is the "spring constant", which measures the elasticity of the bungee rope. When the rope is slack (like at the beginning of the jump), there is no tension. Altogether, we may write the tension force as $-\max(0, k(y - L))$. The *max*imum function ensures that the tension only "switches on" when $y > L$ (i.e. when the rope is taut) and the minus sign in front ensures that it acts upwards.

To summarise, the forces acting on the jumper are

gravity	mg
drag	$-c\|v\|v$
tension	$-\max(0, k(y - L))$

Now, Newton's second law of motion says that the sum of these forces must equal the product of the jumper's mass and acceleration. Hence, the equation governing bungee jumping is the following ordinary differential equation (ODE):

$$m\frac{dv}{dt} = mg - c|v|v - \max(0, k(y - L)),$$

where $\frac{dv}{dt}$ is the jumper's acceleration. We can simplify this slightly by dividing through by m, to obtain

$$\frac{dv}{dt} = g - C|v|v - \max(0, K(y - L)), \tag{1}$$

where $C = c/m$, and $K = k/m$.

3 Numerical method

Equation (1) is too complicated to solve analytically, so you will need to use numerical methods to find solutions. You have developed numerical methods for solving ODEs in your lectures and practicals. Solving the bungee jumping equation will require just a slight modification of these methods. Since equation (1) actually involves *two* unknowns, v and y, we need a second equation that relates the two. This relationship is simply that the jumper's velocity v is the derivative of the jumper's position y. Hence for numerical purposes, we can think of this problem as *two* ODEs:

$$\frac{dy}{dt} = v \tag{2}$$

$$\frac{dv}{dt} = g - C|v|v - \max(0, K(y - L))\,. \tag{3}$$

We can extend any of our numerical methods for solving ODEs to two coupled ODEs:

$$\frac{dy}{dt} = f_1(t, y, v)$$
$$\frac{dv}{dt} = f_2(t, y, v).$$

For example, for Euler's method, expanding both $y(t)$ and $v(t)$ in Taylor polynomials of degree one centered at $t = t_i$ and evaluating the polynomials at $t = t_i$ (see Chapter 2 lectures) yields:

$$y_{i+1} = y_i + hy'(t_i) = y_i + hf_1(t_i, y_i, v_i)$$
$$v_{i+1} = v_i + hv'(t_i) = v_i + hf_2(t_i, y_i, v_i),$$

where h is the subinterval width and y_i and v_i are our numerical solutions for $y(t)$ and $v(t)$ at $t = t_i$ (we haven't bothered introducing a new variable w_i to represent the numerical solution, since we have enough different letters already!). Hence, for the bungee jump model where $f_1(t, y, v)$ and $f_2(t, y, v)$ are identified from the right-hand side of Equations (2)–(3), Euler's method is:

$$y_{i+1} = y_i + hv_i \tag{4}$$

$$v_{i+1} = v_i + h\big(g - C|v_i|v_i - \max(0, K(y_i - L))\big). \tag{5}$$

A complete implementation of Euler's method for the bungee jumping model is given below:

```
function [t, y, v, h] = euler_bungee(T, n, g, C, K, L)
%euler_bungee  Euler's method for the bungee jumping model
% [t, y, v, h] = euler_bungee(T, n, g, C, K, L) performs Euler's method on
% the bungee jumping model, taking n steps from t = 0 to t = T.
% The initial conditions are y(0) = 0 and v(0) = 0.
% The inputs g, C, K and L are parameters from the model (see project description).
% The outputs are the time array t, the solution arrays y and v, and the
% subinterval width h.

% Calculate subinterval width h
h = T / n;

% Create time array t
t = 0:h:T;

% Initialise solution arrays y and v
y = zeros(1,n+1);
v = zeros(1,n+1);

% Perform iterations
for j = 1:n
    y(j+1) = y(j) + h*v(j);
    v(j+1) = v(j) + h*(g - C*abs(v(j))*v(j) - max(0, K*(y(j) - L)));
end
```

4 Model parameters

You have been provided with the following parameters for the model. Some of these come from facts about the bridge itself, while others have been supplied by the bungee jump company assuming an 80kg jumper.

Height of jump point	$H = 74\,\mathrm{m}$
Deck height	$D = 31\,\mathrm{m}$
Drag coefficient	$c = 0.9\,\mathrm{kg/m}$
Mass of jumper	$m = 80\,\mathrm{kg}$
Length of bungee rope	$L = 25\,\mathrm{m}$
Spring constant of bungee rope	$k = 90\,\mathrm{N/m}$
Gravitational acceleration	$g = 9.8\,\mathrm{m/s^2}$

Table 1: Model parameters.

5 Your tasks

Numerical solution

i. Although you have been provided with an Euler method code for this model, the consultancy contract with the Council requires you to use a second order or higher numerical method. Hence you must implement the Second Order Taylor Method *or* the Modified Euler Method *or* the Classical Fourth Order Runge-Kutta method for this model. You only need to implement one of these: you may choose which.

ii. Use your code to obtain the numerical solution for the model and plot the jumper's position against time. You have been provided with the set of model parameters in Table 1. The scope of this report focuses on this particular set of parameters only, which are for an 80kg jumper.

Analysis

1. The bungee jump company suggests that the standard jump will consist of 10 "bounces" which should take approximately 60 seconds. (Although the jumper will still be in motion at this point, the jump will be considered to be over, and the the jumper will be gently raised all the way back onto the platform above.) Do your model results agree with this timing: 10 bounces in around 60 seconds?

2. The "thrill factor" of bungee jumping is partly determined by the maximum speed experienced by the jumper. What is this maximum speed and when does it occur in relation to the overall jump? Answer this question graphically by plotting the jumper's velocity against time.

3. Another factor for thrill-seekers is the maximum acceleration experienced by the jumper. More acceleration equals bigger thrills, but too much acceleration can be dangerous. The bungee jump company boasts that the jumper will experience acceleration "up to $2g$". Use numerical differentiation to find the acceleration predicted by your model, and plot the jumper's acceleration against time. What is this maximum acceleration and when does it occur in relation to the overall jump? Is the claim of "up to $2g$" acceleration supported by the model?

4. For the writing of promotional material it is of interest to know how far the jumper actually

travels in the 60 second jump. One way to answer this question is to compute the integral

$$\int_0^{60} |v| \; dt \, .$$

Use numerical integration to compute this integral and hence determine how far the jumper travels.

5. Part of the proposal is to have a camera installed on the bridge deck, at height D from the water. As the jumper first passes this point, the camera would take a photo which could then be offered for purchase afterwards. It is hoped that the model can provide sufficiently accurate results that the camera could be set to trigger at a predetermined time, for a given set of model parameters.

   The distance the jumper falls from the platform to the deck is $H - D$. Hence you need to compute an accurate value for t such that $y(t) = H - D$. Since you only know $y(t)$ as a discrete set of points, not as a function, you will need to fit an interpolating polynomial.

   (a) To begin with, determine the nearest four values y_i of your numerical solution that lie either side of $H - D$. That is, find values $y_i, y_{i+1}, y_{i+2}, y_{i+3}$ such that $y_i, y_{i+1} < H - D$ and $y_{i+2}, y_{i+3} > H - D$.

   (b) Fit the interpolating polynomial $p(t)$ through the four points (t_i, y_i), (t_{i+1}, y_{i+1}), (t_{i+2}, y_{i+2}), (t_{i+3}, y_{i+3}).

   (c) Use a rootfinding method of your choice to find the value of t such that $p(t) = H - D$.

   (d) Hence, for the model parameters provided, at what time should the camera trigger in order to capture the image of the jumper?

6. The bungee jump company has suggested a "water touch" option could be considered, whereby the jumper just touches the water at the bottom of the first bounce. For the given parameters, how close does the jumper come to touching the water? Investigate how the bungee rope could be altered (its length, its spring constant, or both) to produce a true water touch experience for an 80kg jumper, while keeping as close as possible to 10 bounces in 60 seconds. Note: any combination of parameters that produces acceleration of greater than $2g$ must be rejected as too dangerous.

6 Report and code

You must submit your findings in a report. This report will be written in MATLAB itself, using a script file with sections, which is then built into a PDF report using the Publish feature. A video tutorial on publishing with MATLAB is available here: `http://www.mathworks.com.au/videos/publishing-matlab-code-from-the-editor-69016.html`

A partially complete MATLAB script file for this report has been provided: you must complete the file by adding in the missing text and code.

Your final submission will consist of the report **in PDF format** as well as **all code**, including the completed MATLAB script file and all other MATLAB functions that you write, which must include at a minimum, *your own functions* to perform

- second order (or higher) numerical solution of model ODEs

- numerical differentiation
- numerical integration
- interpolation
- rootfinding

which will be called from the script file in the appropriate places. The script itself should run to completion without errors, and generate precisely the report that you submit.

7 Marking criteria

This assignment will be marked using the criteria accompanying this document. The group's final grade on a 1-7 scale will be a **weighted** average of the grades for the criteria:

$$\text{Grade} = (5 \times \text{Tasks} + \text{Clarity} + \text{Correctness} + \text{Language} + \text{Report} + \text{Script})/10\,.$$

This grade will then be mapped to a final score out of 30.

8 Statement of contribution

Each group member must complete a statement outlining, in that person's opinion, the relative contributions made by every member of the group. A template has been provided for this purpose. Each group member should contribute equally to the assignment. Failure to do so may result in lower marks being awarded to students who do not contribute sufficiently.

9 What to submit

One member of the group submits, as a single zip file:

- the MATLAB script file and all MATLAB function files written for the assignment;
- the report in PDF format.

Each member of the group submits (not in a zip file):

- a statement outlining the relative contributions made by each group member (use the template provided).