# I won't just try defying gravity, I'll succeed:

A mathematical model that turns snowboarding into snowinteresting.

Georgia Witchel

Harvey Mudd College

Claremont CA.

Advisor: Prof Yong

## I. Introduction

I used a mathematical model to determine the optimal time a snowboarder should jump whilst traveling over a set of moguls in order to maximize their time airborne. The successful completion of any snowboarding trick is dependent on three factors: the strength and agility of the snowboarder, and the amount of time they spend airborne. Because the strength and agility of the snowboarder is not changed by their airtime, we can assume that airtime is a function of strength, not vice versa. By completing this model I will show that the optimal time for the snowboarder to jump is not only a function of the shape of the run but is also effected by the initial velocity in the x direction, their strength, and the frictional coefficient of the snow.

## II. Assumptions

- **Wind resistance is negligible:** Wind resistance would greatly complicate the model as it would require us to determine the snowboarders in-air position as well as their bodily position during their time in the air. Similarly, modeling wind resistance wouldn't make sense in the context of the model, as most ski runs are effected by a directional wind that varies depending on the weather.
- **The frictional constant (μ) between the bottom of the snowboard and the top of the snow remains constant throughout the jump:** In a real-world scenario the type of snow (icy and hard vs soft and fluffy) varies greatly over the course of the run as a result of consistent use or inconsistent weather patterns. In this model we assume that all areas of the jump have similar snow compositions. We also assume that the base of the snowboarder's board remains fully intact with the snow until the moment they take off.
- **The jump happens instantaneously:** In reality, the snowboarders jump would include a windup, a tilt of the board, and would involve different parts of the board leaving the ground at different times. To simplify the model we assume that the snowboarder's takeoff happens instantaneously and without a windup, causing them to gain an upward velocity of $v_{y0}$ without needing to act on the system.
- **On the downward slope the snowboarder pumps their legs:** we assume that while the snowboarder is on the downward sloping portion of the run then they are able to gain some velocity proportionate to their jump in the y direction by pumping their legs. We will denote this proportionality via a strength constant S. This gained velocity is not effected by μ.

- **When the snowboarder jumps, he only jumps in the positive Y direction:** the snowboarder cannot jump forward or backward or downward.

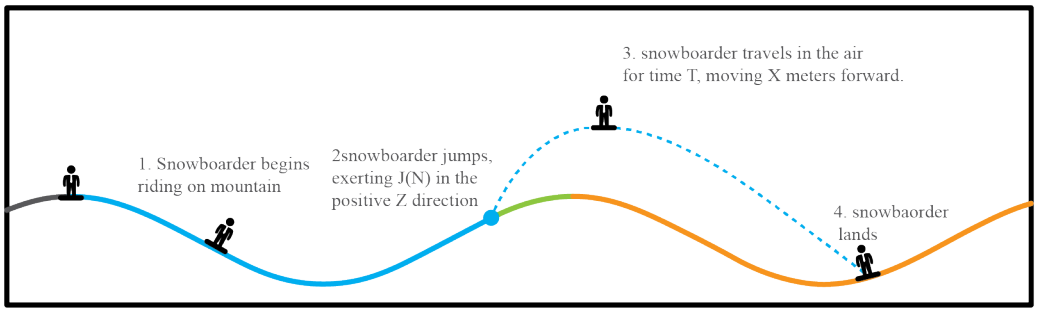
# III. Constructing the model:

In this model we assume the jump to be part of a flat mogul run that can be modeled as the cos wave:

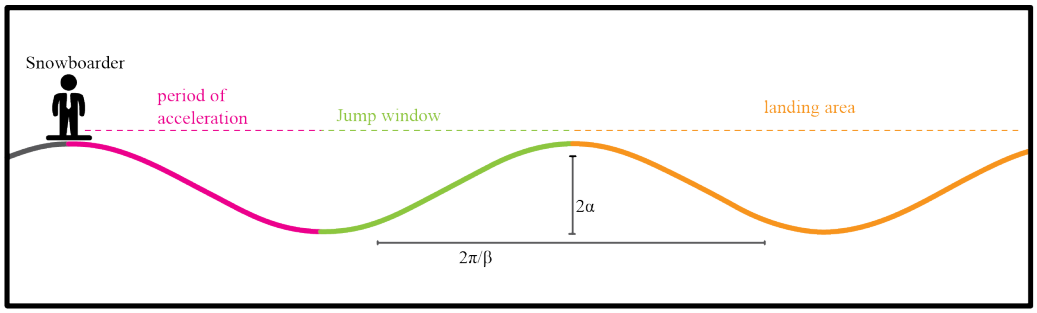
$$y = \alpha\cos(\beta x)$$

Where the snow has frictional coefficient $\mu$ with the bottom of the snowboard.

The snowboarder begins at the peak of a mogul with some horizontal speed $v_{x0}$, and then takes off somewhere before the peak of the next mogul, landing somewhere after that mogul's peak.




We will initially break this mathematical model into two sections. Section 1 models the snowboarder's acceleration to the bottom of the first mogul (shown in pink) and then some period of upward deacceleration as they travel up the second Mogul, sections 1 ends with the snowboarders jump (somewhere within the green section). Section 2 models the snowboarder's airtime as a function of their initial speed in the x and y directions as well as the horizontal distance between their jumping and landing points.




## A. Calculating leaving velocity

First, we begin by determining the speed of the snowboarder at point $\left(P_1, P_2\right)$, which represents the x and y coordinates of their takeoff. We can model the snowboarder as a block having mass $m$ and frictional coefficient $\mu$ with the run.

Say the snowboarder begins at position $(0, 2\alpha)$ and jumps at position $\left(P_1, P_2\right) = \left(P_1, \alpha\cos\left(\beta P_1\right) + \alpha\right)$ where $P_1 < 2\alpha, 0 < P_2 < \frac{2\pi}{\beta}$.

In order to model the snowboarder's final velocity at $\left(P_1, P_2\right)$ we can use conservation of energy, accounting for loss of momentum due to friction, and then add in the velocity gained by pumping their legs.

$$\frac{1}{2}mg\left(2\alpha - P_2\right) = \frac{1}{2}mv^2 - \int_0^{P_1} u \cdot mg\cos(\theta(x)) \cdot l(x)\,dx$$

$$l(x) = \sqrt{1 + \frac{dy}{dx}^2} = \sqrt{1 + \alpha^2\beta^2\sin^2(\beta x)}$$

$$\theta(x) = arc\tan\left(\frac{dy}{dx}\right) = arc\tan(-\beta\alpha\sin(\beta x))$$

This equation can be solved for the final velocity without pumping , which can be expressed as

$$v = \sqrt{g\left(\left(2\alpha - P_2\right) + 2\int_0^{P_1} u\cos(\theta(x)) \cdot l(x)\,dx\right)}$$

Finally we can add the gained velocity from pumping

$$v_f = \sqrt{g\left(\left(2\alpha - P_2\right) + 2\int_0^{P_1} u\cos(\theta(x)) \cdot l(x)\,dx\right)} + \int_0^{P1} v_{pump}\,dx$$

$$V_{pump}(\theta) = \begin{cases} -\frac{v_{y0}}{S}\theta & \theta < 0 \\ 0 & \theta \geq 0 \end{cases}$$

s = some strength constant

The final velocity when the snowboarder jumps will be broken into an x and a y component, which is dependent on $\left(P_1, P_2\right)$ and the initial upward velocity gained from the snowboarder jumping.

$$v_x = v_f\cos\left(\theta\left(P_1\right)\right) + v_{x0}, v_y = v_f\sin\left(\theta\left(P_1\right)\right) + v_{y0}$$

Where $v_{y0}$ is the upwards velocity gained from the snowboarders jump.

## B. Calculating the jump

Once we have determined the velocity, we can use it to plot the snowboarder's trajectory as a quadratic function by setting the derivative of the function at $\left(P_1, P_2\right)$ to $\frac{v_y}{v_x}$ and the derivative of the function at $\left(P1 + v_x \cdot time\ to\ max\ height\right)$ to 0.

$$y(x) = ax^2 + bx + c\ ,\ y' = 2ax + b$$

$$y(P1) = P2$$

$$y'(P_1) = \frac{v_y}{vx},\ y'\left(P1 + t_{up} \cdot vx\right) = 0\ \left(t_{up} = \frac{v_y}{-g}\right)$$

$$\frac{v_y}{v_x} = 2aP_1 + b,\ 2a\left(P1 - \frac{v_y}{g}\right) + b = 0$$

$$P2 + aP1^2 - bP1 = + c$$

$$a = -\frac{g}{2v_x},\ b = \frac{1}{v_x}\left(v_y - gP_1\right), c = P_2 + \frac{gP_1{}^2}{2v_x} - \frac{P_1}{\mathrm{v_x}}\left(\mathrm{v_y} - \mathrm{gP_1}\right)$$

$$y(x) = -\frac{g}{2v_x}x^2 + \frac{1}{v_x}\left(v_y + gP_1\right)x + P_2 + \frac{gP_1{}^2}{2v_x} - \frac{P_1}{\mathrm{v_x}}\left(\mathrm{v_y} - \mathrm{gP_1}\right)$$

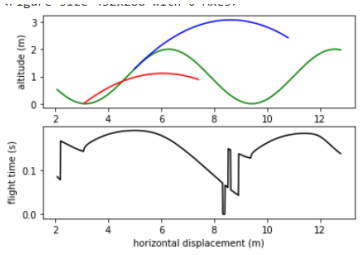
Finally, we can then determine the points of intersection $\left(P_1, P_2\right), \left(P_3, P_4\right)$ and use the equation $\left(P_1 + tv_x, \alpha\cos\left(\beta \cdot \left(P_1 + tv_x\right)\right)\right) = \left(P_3, P_4\right)$ to determine the total airtime as a function of $P_1$

$$-\frac{g}{2v_x}x^2 + \frac{1}{v_x}\left(v_y - gP_1\right)x + P_2 + \frac{gP_1{}^2}{2v_x} - \frac{1}{\mathrm{v_x}}\left(\mathrm{v_y} - \mathrm{gP_1}\right)P_1 = \alpha\cos(\beta x) + \alpha$$

# IV. Implementation:

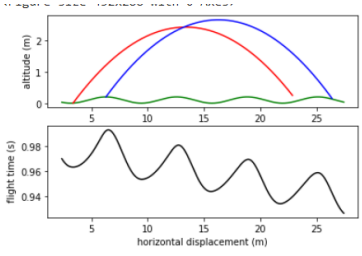
In my implementation I used a Jupytr notebook with four libraries: `math`, `scipy`, `matplotlib`, and `numpy`. I defined each function outlined above, then made a function called airtime, which takes in a point of jump and outputs the total airtime by applying a root finding method to the final airtime equation shown above. Airtime implements `fsolve` (scipys root finding function) to find the intersection of the jump arc and the hill. I then used the two intersection points to calculate and return the total airtime given P1. Following that I determined the minimum and maximum of `airtime(p1)` using Scipys `optimize.minimize_scalar(method="Bounded")`.

# V. Results:



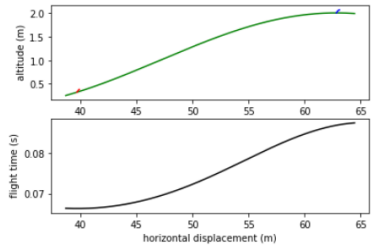

```
jump from point 3.07 to achedive a minimum Airtime of: 0.14 seconds
jump from point 5.0 to achedive a maximum Airtime of: 0.19 seconds

==============
b =  1
a =  1
Vx0 =  30
Vy0 =  30
Mu =  0.01
S =  10
==============
```



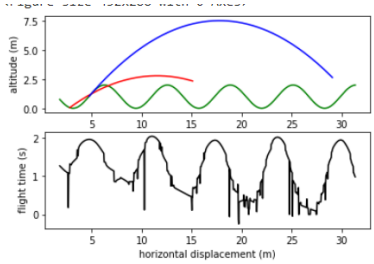

```
jump from point 3.36 to achedive a minimum Airtime of: 0.96 seconds
jump from point 6.28 to achedive a maximum Airtime of: 0.99 seconds

==============
b =  1
a =  0.1
Vx0 =  20
Vy0 =  10
Mu =  0.001
S =  10
==============
```



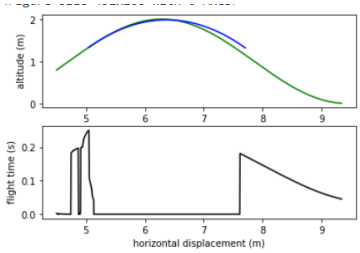

```
jump from point 39.71 to achedive a minimum Airtime of: 0.07 seconds
jump from point 62.83 to achedive a maximum Airtime of: 0.09 seconds

==============
b =  0.1
a =  1
Vx0 =  3
Vy0 =  3
Mu =  0.01
S =  10
==============
```



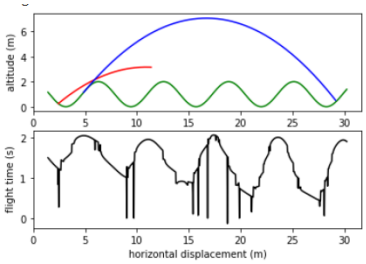

```
jump from point 2.81 to achedive a minimum Airtime of: 1.07 seconds
jump from point 4.75 to achedive a maximum Airtime of: 1.97 seconds

==============
g =  -1
b =  1
a =  1
Vx0 =  10
Vy0 =  10
Mu =  0.7
S =  10
==============
```




```
jump from point 5.5 to achedive a minimum Airtime of: -0.0 seconds
jump from point 5.05 to achedive a maximum Airtime of: 0.25 seconds

==============
b =  1
a =  1
Vx0 =  10
Vy0 =  10
Mu =  0.0
S =  10
==============
```




```
jump from point 2.42 to achedive a minimum Airtime of: 0.84 seconds
jump from point 4.87 to achedive a maximum Airtime of: 2.06 seconds

==============
g =  -1
b =  1
a =  1
Vx0 =  10
Vy0 =  10
Mu =  0.0
S =  10
==============
```

From the results, it is clear that some variables effect the system more than others. For example, Mu and Gravity effect the system greatly, whereas the initial x velocity has little effect on the jumping point. The ratio of the Snowboarders jumping velocity and the maximum height of the run also effect the model greatly, with larger jumping strengths correlating to earlier jumping points. The smaller B, the smoother the airtime plot. Ultimately, to maximize airtime, the jumping point appears to be the highest point, but the minimum time doesn't follow the same pattern. Unless otherwise specified, the jumping point is always zero, which arises from the fact that they begin at the highest point of the hill and have the most horizontal speed. This model is useful in demonstrating that maximizing the airtime of a jumper is a complex process that cannot be easily computed by hand. While I don't believe that anyone will be using it on the mountain, it is helpful to understanding when planning beforehand, and in showing that the best jumping point varies from situation to situation.

# VI. Limitations and future work:

One of the major problems encountered was the complex nature of the equations, which made the roots exceedingly difficult to find. This is illustrated in the images, where you can clearly see a gap between where the jump arc ends and where it is supposed to end (intersecting with the main curve). This is also due to the low number of divisions taken when plotting. For many of the situations outlined above, the constraints are unrealistic, such as the snowboarder jumping upwards at 30 meters per second. If some

of the initial conditions were more relaxed (such as the instantaneous nature of the snowboarders jump or wind resistance) then the model would be more accurate.

## VII. Conclusion:

This model sought to implement a root finding method to determine the optimal time for a snowboarder to jump in order to maximize their airtime. The model took into account the equation for the run, the snowboarder's initial velocity in the x direction and the upward velocity they gain from jumping. It also accounted for the frictional coefficient between the snowboard and the snow. One conclusion we can draw from experimenting with this model is that, unless the snowboarder is superhuman or on the moon, their airtime is mostly dependent on their leaving velocity rather than the velocity gained from the jump. This model is also valid for skiers.

One further implementation of the model might be modifying it so that it can allow any function to model the hill, not just a cosine function. This would be relatively simple as it would just involve plugging in different derivatives.