Using Physics Engine to Iterate the Maximum Height of Multiple Kids on Trampoline

Team 138, Problem B

November 5, 2023

Abstract

In this paper, we find the maximum height that each of the kids can reach under the condition that they jump simultaneously on the trampoline. We set up the basic model of the trampoline based on the real-life trampolines on the market with the similar parameters like diameter and shape as the trampoline used in this problem. We investigate the time length during which the kids are in contact with the trampoline, helping to calculate the leaving speed of the kids. We discussed the influence of the air resistance on the surface of the trampoline, tying the model to the realistic conditions. We created a physics engine that can calculate the height of a kid with respect to the jumping time delay of the other two kids, using the python and Matlab coding. We analyze the motion from a rough trend to an accurate numerical level. The results show that the kid of 25kg can reach a maximum height of 1.6890m; the kid of 40kg can reach a maximum height of 1.2810m; for the 50kg kid, the height is 1.3668m. The maximum height of each kid are found when two kid reaches the lowest point with the target kid exactly falling on the trampoline. This study offers the producers and designers an approach to test the feasibility of a trampoline, ensuring the safety of the users.

Key Words: Trampoline, Physics Engine, Simulation, Real Life Application

Contents

1 Introduction and Problem Restatement

As a fantastic way of amusement and exercises, trampolines are becoming increasingly popular among people, especially the children. Sales have rocketed up in the past decades. [1] Safety problems do occur after the trampolines are popularized. In the research of Sylvia et. al., twelve injuries are reported and tracked, among which 7 are fracture injuries. [2] As fracture injuries are closely related to the speed at which the kids land on the trampoline, the maximum height that the kids can reach comes into our sight. However, not much knowledge is gained about the maximum possible height that one can reach on the trampoline. Therefore, in this essay, we will discuss the factors affecting the maximum height and simulate a condition with kids of assumed weight. A model of the motion of the kids and the surface of the trampoline that takes many influential factors into account is generated. Real case tests based on the masses of the kids are then conducted to give results on the jumping strategy to achieve maximum height of each kid.

2 Assumptions and Reasoning

Throughout this entire essay, the following assumptions are acknowledged.

1. The kids are viewed as mass points. In real life trampoline jumping, the feet of each person differs in area and shape and even the angle at which the feet get in touch with the mat affects the maximum height. However, such complex factors are impossible to consider in a simplified model. This assumption aims at simplifying the model.
2. When the kid gets over the the equilibrium location, the velocity is considered to be 80% of its initial velocity. From trampoline videos on Youtube, if players exerts little force on the trampoline, i.e. contacting the trampoline with hip, the rebounding height, using the equilibrium position of the trampoline as reference, is approximately 60%-70%. Thus, a relatively realistic 20% reduction in velocity is set.

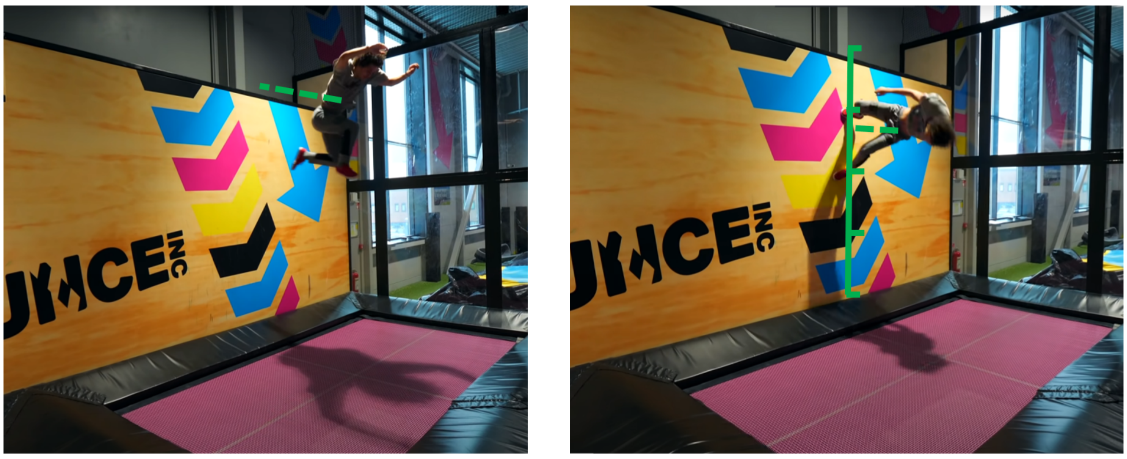


Figure 1: The screenshot of the jumping motion from video. The height reached after leaving the trampoline is 60% of the initial height.[3][4]

3. The three kids are considered to land on the same point at the center of the trampoline surface. As a landing aside the center generates a horizontal component of force, making the vertical component of the velocity gets smaller, the maximum height becomes impossible. The landing all on the center generating a vertical returning velocity is what we expect

4. The air resistance exerted on the kids are negligible.

5. All the discussion of one single kid is based on the condition that the kid has already reached the steady state where after each jump, the kid get to his maximum height.

6. The forces exerted on the springs of the trampoline do not exceed elastic region. In other word, the springs strictly follow the Hooke's Law.

7. The boundary effect of the trampoline surface is neglected. Instead of exerting discrete forces on the boundary, the forces of the springs are considered to distribute uniformly on the mat.

8. The collision between the kids and the mat is completely inelastic.

9. The trampoline surface is considered to be of a V-shape throughout the descending and lifting process. The reason is to be discussed in the discussion part.

3 Notations

Symbols	Description	Value
R_m	Radius of the mat of the trampoline	$2.50m$
g	Gravity Acceleration	$9.81\ m/s^2$
m_k	Effective mass of the mat	
k_s	The sum of the spring constants of all the springs	
m_0	Mass of the kid	
c	Damping coefficient caused by mat material	
x	The upward displacement of the mat	
x'	The upward velocity of the mat	
x''	The upward acceleration velocity of the mat	
c_a	The air resistance coefficient of the mat	
N	Stretching force caused by man	

Table 1: Notation Table

The main notations in the modelling process are defined in Table 1. Some of the values are already given, while the other values will be discussed and given later.

4 Model and Analysis

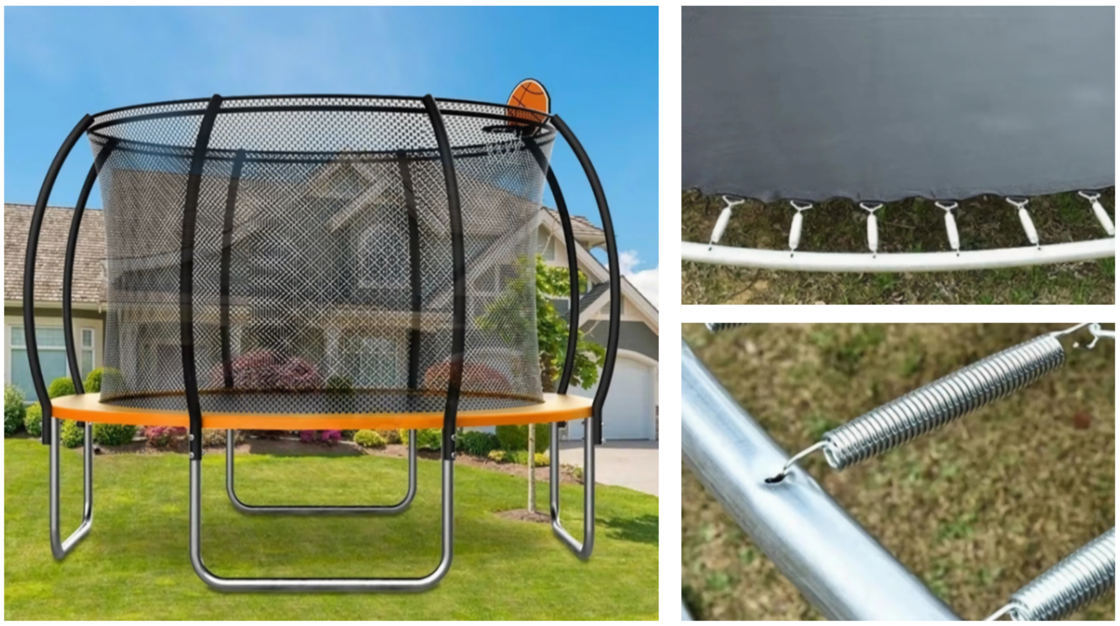


Figure 2: This is the trampoline used as the model for this problem.[5]

A trampoline in real-life is selected as the model for this problem. In this model, the trampoline includes an inelastic mat with a radius of 2.5m. On the side of the mat arrays the spring, whose direction is the same as the edge of the trampoline.

4.1 Effective mass of the mat

From resources on the online shops of trampolines, the elastic mat has a surface mass density of around 300 g/m^2. In this problem, the mat has a diameter of 5m. Calculating the real mass of the mat, we get

$$M_{real} = 300 \times 2.5^2\pi = 5.89kg$$

Assuming that the mat forms a V shape after the kids fall on the mat and the descending velocity of the mat increases linearly from the edge to center, we then can calculate the relative effective mass

$$m_k = \frac{\int_0^{R_m} \frac{v(R_m - r)}{R_m} dr M_{real}}{\pi r^2 v} \approx 2kg$$

where v is the descending speed of the center of the mat, R_0 is the radius of the mat.

4.2 Elastic constant of the spring

The spring constant can be calculated with the following formula.

$$k = \frac{Gd^4}{8nD^3} \qquad [6]$$

To simulate the trampoline used in this problem, a similar trampoline on sale in an online shop is researched into. The diameter of the trampoline is 4.88m. The springs has a $D = 3cm$ diameter, made of No.70 steel wires with a $d = 5mm$ diameter. Each spring has $n = 40$ turns.[7] The sheer modulus of No.70 steel is $G = 73.0Gpa$.[8] With these data, we calculate that the spring constant of the spring is around

$$k_{sp} = 5281N/m$$

Data show that a normal trampoline for competition has a perimeter of around 12.84m, and has 110 springs in total.[9] With the same proportion between perimeter and number of springs, the trampoline with diameter 5m should have 135 springs in total.

$$F_s = k_s(\sqrt{R_m^2 + x^2} - R_m)\frac{x}{\sqrt{R_m^2 + x^2}}[10]$$

where R_m is the radius of the mat, k_s is the sum of the spring constants of all the springs, and x is the displacement of the effective mass of mat and the kid above the surface of an empty trampoline.

4.3 Analysis of the motion in two periods

From the previous parts, we discovered the spring constants and the effective mass. In this section, we start analyzing the forces exerted on the effective mass (i.e. the simplified mat and the kid).

4.3.1 From falling to after collision

After one kid is "released" from the corresponding maximum height, the motion before the kid falls on the trampoline can be considered as a free fall motion as the air resistance on the kid is so small that it can be neglected.

After falling a distance H, the kid then collides with the mat with a totally inelastic collision. Thus the velocity of the kid, together with the effective mass of the mat, can be calculated as:

$$v_0 = -\frac{m_0}{m_0 + m_k}\sqrt{2gh}$$

4.3.2 Before reaching the lowest point

After the kid falls on the trampoline, we assume that the kid does not exert an extra force to the surface of the trampoline in the descending process of the motion. The factors considered in this process include the air resistance on the mat (f_a), damping force (R), supporting force exerted by the spring (F_s), and the total gravity (G).

$$f_a = -\frac{\pi c_a R_m^3}{6\sqrt{R_m^2 + x^2}}\left|x'\right|x'$$
$$R = -cx'$$
$$F_s = k_s(\sqrt{R_m^2 + x^2} - R_m)\frac{-x}{\sqrt{R_m^2 + x^2}}$$
$$G = -(m_0 + m_k)g$$

With $a = \frac{F}{m}$, we get that

$$x'' = \frac{1}{m_0 + m_k}[-\frac{\pi c_a R_m^3\left|x'\right|x'}{6\sqrt{R_m^2 + x^2}} - cx' + \frac{-k_s(\sqrt{R_m^2 + x^2} - R_m)x}{\sqrt{R_m^2 + x^2}} - (m_0 + m_k)g]$$

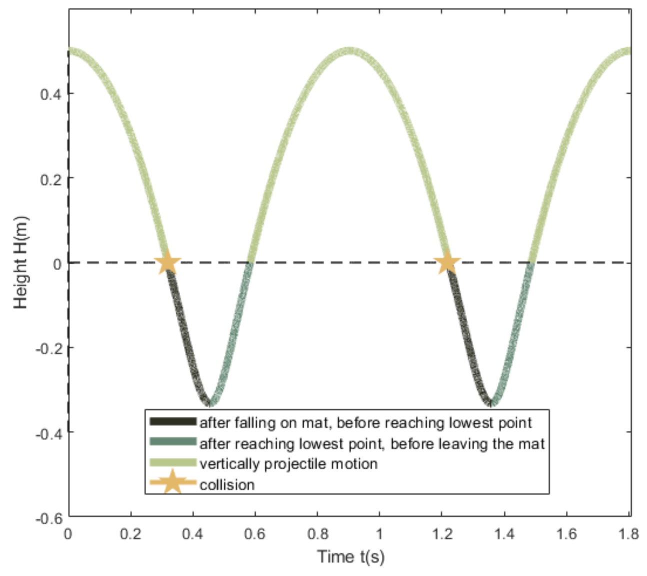


Figure 3: This relates the time with the height of the kid. The line sections above H=0 is the free fall motion before the kid falls on the trampoline and after he leaves the trampoline. The dark-color line section stands for the motion between the collision and the reach of lowest point. The green line below H=0 stands for the motion after passing the lowest point to the instant when the kid leaves the trampoline. Two yellow stars stand for the collision instant.

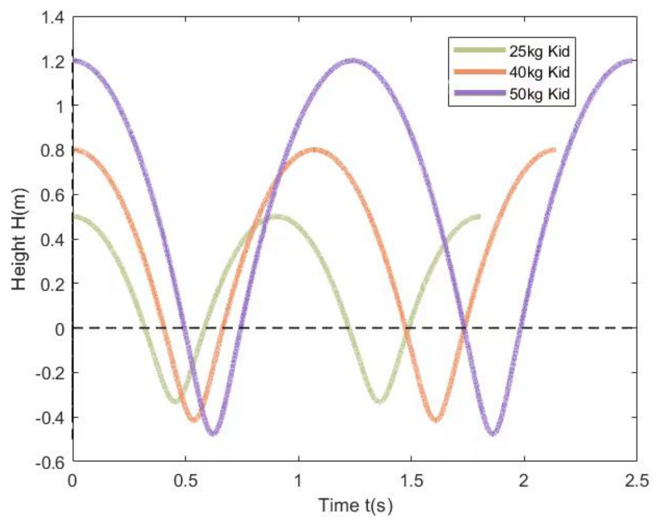


Figure 4: The motion of three kids in two periods. The green line stands for Kid A; the yellow line stands for Kid B; the purple line stands for Kid C. Each line follows the display method described in Figure 3.

And the initial condition is $x(0) = 0$, $x'(0) = v_0$.

If the kid does not exert any extra force on the mat, then the whole process obeys the equation above. And from Assumption A, the velocity when passing through the equilibrium position is $-0.8v_0$. Then by adjusting its value and calculate the average velocity ratio between initial velocity and velocity passing through the equilibrium position, the damping coefficient is:

$$c = 9.40\ N \cdot s/m$$

Correspondingly, the time cost, the center of mass of the kid and the effective mass of mat, after touching the elastic mat but before reaching the lowest point, are shown in Table 2.

Index	**Mass/kg**	**Height/m**	**time b. l./s**	**Max. displacement/m**	**time a. l./s**	**s. force/N**
	m_0/kg	h/m	t/s	x_m/m	t'/s	N/N
Kid A	25	0.5	0.1360	-0.3327	0.1285	236
Kid B	40	0.8	0.1335	-0.4147	0.1281	343
Kid C	50	1.2	0.1270	-0.4765	0.1225	473

Table 2: The time cost, and the center of mass of the kid and the efficiency mass of mat, after contacting the elastic mat before reaching the lowest point. 'b.l.' and 'a.l.' separately means 'before lowest point' and 'after lowest point', 's. force' means 'stretching force'.

4.3.3 After passing the lowest point

We now assume that after reaching the lowest point, the kid start exerting extra force on the trampoline surface to gain a larger leaving velocity in the upward motion. The supporting force is generated by straightening the legs. This entire system leads to a slower contraction of the springs on the side of the mat than the contraction without the force, which brings about an additional force contributing to the acceleration of the kid. The body of the kid thus experiences a larger upward acceleration.To describe this difference, we denote the force as N for later calculations and assume that the force acting upon the kid remains constant throughout the whole process of the kid's upward motion.

After taking in the force, the differential equation is revised into

$$x'' = \frac{1}{m_0 + m_k}[-\frac{\pi c_a R_m^3 \left| x' \right| x'}{6\sqrt{R_m^2 + x^2}} - cx' + \frac{-k_s(\sqrt{R_m^2 + x^2} - R_m)x}{\sqrt{R_m^2 + x^2}} - (m_0 + m_k)g + N]$$

with boundary condition $v(0) = 0$, $x(0) = x_m$. When it passes the equilibrium position,the kid separates from the mat, with the velocity $v = \sqrt{2gh}$. The time spent on this process t' and the stretching force N can thus be worked out and listed in Table 2.

4.4 Rough Measurement of Maximum Height

The primary objective of our approach is to employ iterative methods to track the kinematic characteristics of both the children and the mat. To achieve this, we have developed a self-made physics engine tailored to the specific challenges of this problem (The codes are listed in Appendix A).

Our approach begins with the establishment of the system's physics properties and simulation parameters. Subsequently, we implement physical formulas that control the kinematic behavior of the entities involved. These formulas allow us to compute the physical attributes of the system at each frame, with a frame duration of 1ms. Following this, our system incorporates a collision detection function. When a frame indicates that a child is situated beneath the mat, we apply the principles of the momentum theorem to recalculate their velocity and adjust their displacement accordingly. Ultimately, within a specified time frame of 5 seconds, we determine the maximum height reached by all the children.

After the system is set up, we assume that the kids can reach their maximum heights the same as the height when they jump alone with an arbitrary time interval. As a result, we can simplify the calculation by controlling the motion of one kid, who is also the target kid for whom we want to calculate the maximum height. We set the instant when the target kid start falling from the maximum height as the reference time, and let the other two kids fall from their maximum height at the instants during a 12s period after the target kid falls down.12s time length is selected because after over 10 periods the mutual velocity influence of two children has been fully displayed. The least common multiple of the individual vibration periods of any two out of three children is also less than 12s, meaning that mutual influence of the three kids are all considered in an way.

Finally, given that physics engines have inaccuracy in complex scenarios like successive collisions, noise points may influence the results to some extent. To mitigate this disturbance, we apply a Gaussian blur to the maximum height image, ensuring the stability and reliability of the results.

Based on the model and the simulation methods mentioned above, the results of the simulation are generated and listed below. Figure 5, Figure 6, and Figure 7 show the relationship between maximum height of Kid A,B,C with respect to the starting time delay of the other two kids.

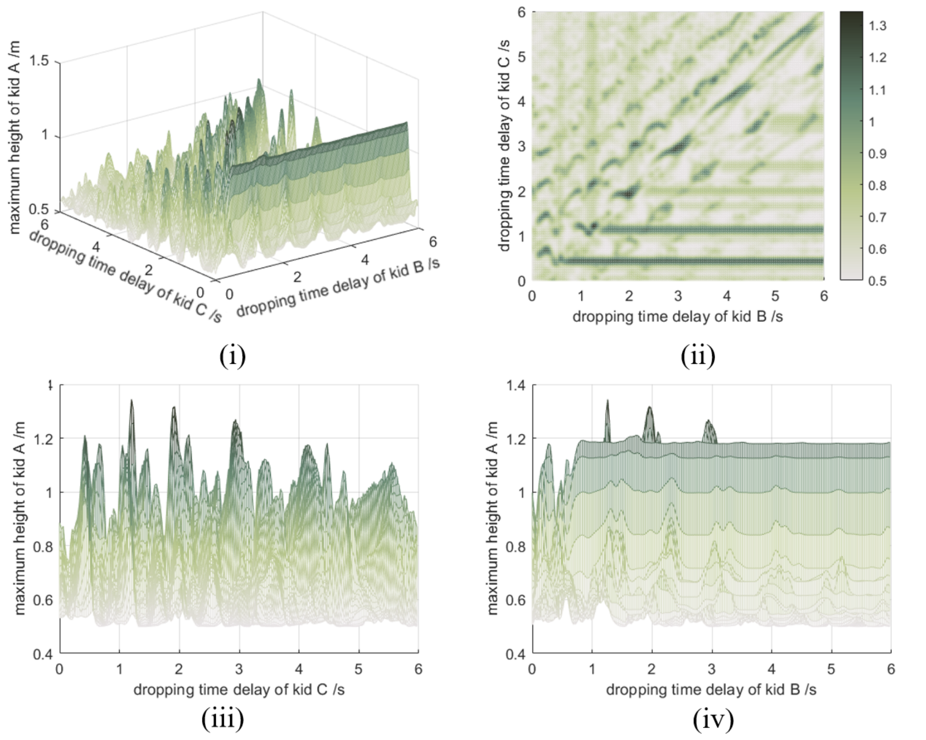


Figure 5: (i) The maximum height of Kid A when the starting time delay of Kid B and Kid C are different. (ii) The top view of (i). (iii) Kid A's maximum height-Time delay of Kid C plot. (iv) Kid A's maximum height-Time delay of Kid B plot.

From the figures we can identify some characteristics as follows:

1. For a higher dropping time delay, the maximum height is generally lower.

2. When the dropping time delay of one kid remains constant, the maximum height varies periodically with the dropping time delay of another kid, that is, within each period, there is a moment when the maximum ascent height is extremely high. Between two kids, the kid with a greater weight and stretching force generates more pronounced changes in height.

3. The maximum height appears when the two periodical maxima take place closely.

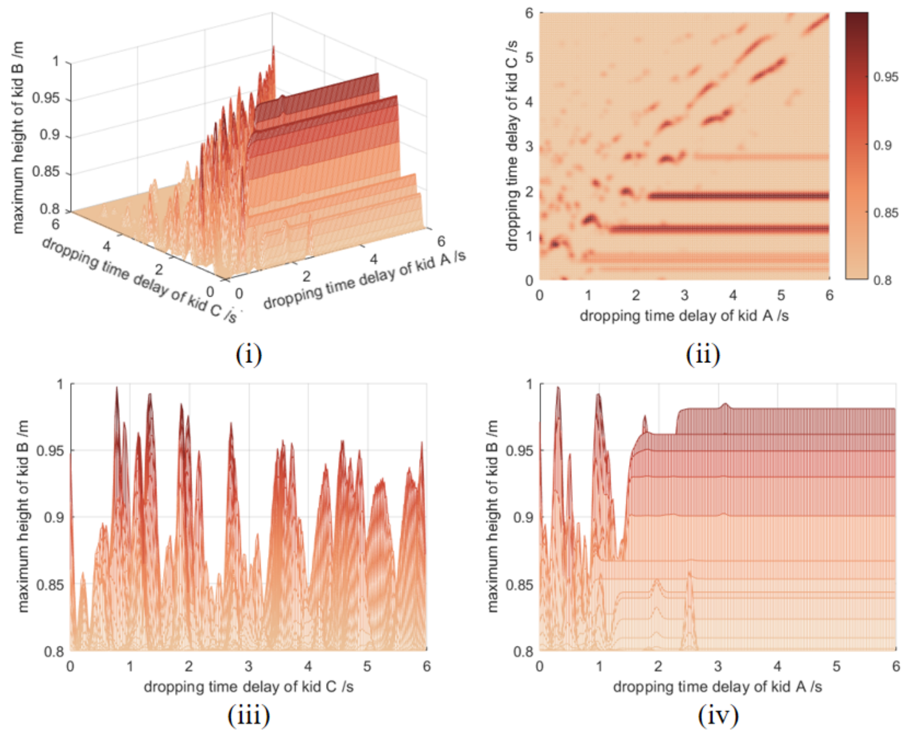


Figure 6: (i) The maximum height of Kid B when the starting time delay of Kid A and Kid C are different. (ii) The top view of (i). (iii) Kid B's maximum height-Time delay of Kid C plot. (iv) Kid B's maximum height-Time delay of Kid A plot.

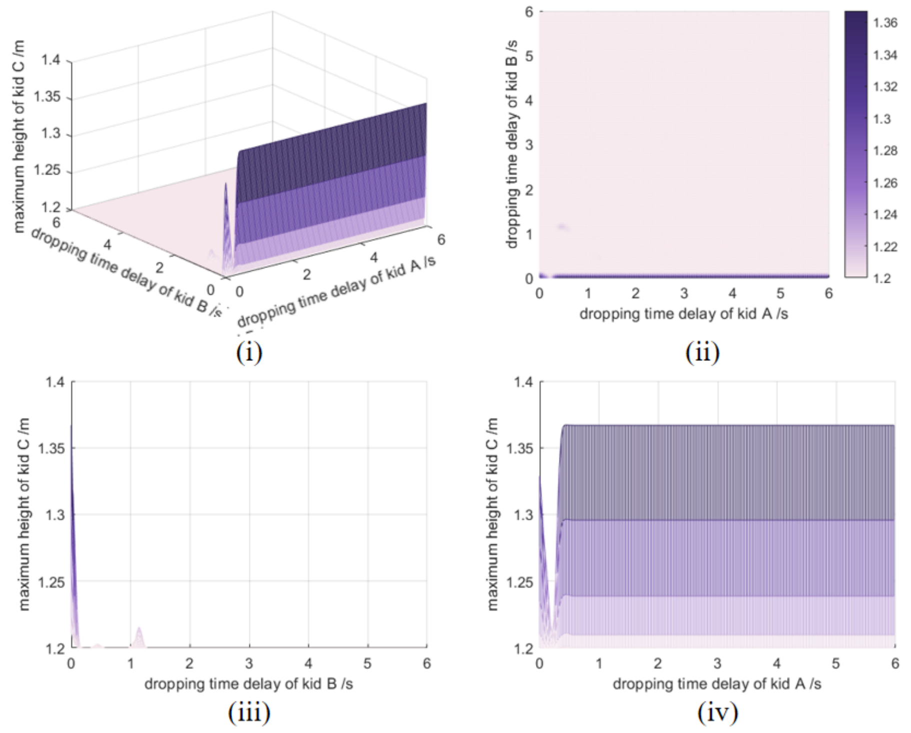


Figure 7: (i) The maximum height of Kid C when the starting time delay of Kid A and Kid B are different. (ii) The top view of (i). (iii) Kid C's maximum height-Time delay of Kid B plot. (iv) Kid C's maximum height-Time delay of Kid A plot.

4.5 Accurate Measurement of the Maximum Height

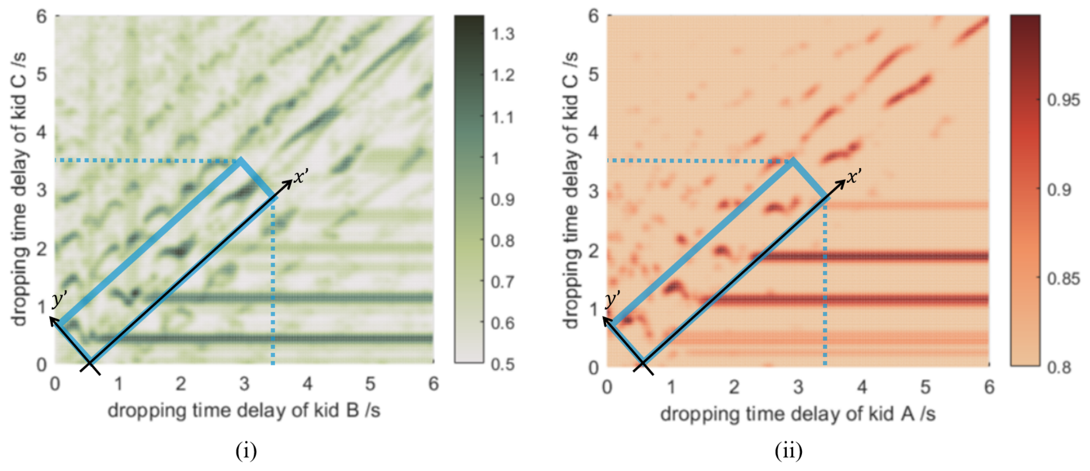


Figure 8: (i) The height of Kid A with respect to dropping time delay of the other kids (shown from the color). (ii) The height of Kid B with respect to dropping time delay of the other kids (shown from the color). The rectangle shows the region that contains the potential global maximum height, which is simulated more accurately in the later section.The coordinate system after transformation is also shown in this figure.

Based on the results gained in rough measurement in section 4.4, we locate the time delay region of two kids where the maximum height of the target kid gets to a highest interval. We then focus more on this time delay region and carry out more detailed simulation of the motion of the target Kid with respect to the time delay difference of the other two kids as the global maximum height must appear in this region.

In this section, the motion of Kid C, who has a largest weight, is neglected as the results from the Figure 7 indicates that no matter how the dropping time delays of kid A and B are arranged, Kid C can never reach a larger height if A and B do not fall during a period of several seconds after C falls.

Then the algorithm and the code is modified, with the range of data set to a more precise rectangle to enhance the accuracy of the results. Using coordinate transformation to form a new coordinate with center at (0.5,0) and with the original axes rotating $45°$ counterclockwise, the environment of further simulation is set up. The distribution of the maximum height is then simulated in the transformed coordinate system. The visualized results are displayed in Figure 9 and 10. The time delay in this section is identical to that in the last section.

Also it is worth noting that in $t_b{+}t_c$ and $t_a{+}t_c$ dimension, the maximum displacement has a stable period, indicating its accuracy.

Index	Mass/kg	TD Kid A/s	TD Kid B/s	TD Kid C/s	Max. Height/m
	m_0/kg	t_A/s	t_B/s	t_C/s	H/m
Kid A	25	0	1.26	1.20	1.6890
Kid B	40	0.3	0	0.78	1.2810
Kid C	50	>0.39	0	0	1.3668

Table 3: The mass, time delay, and the resulted maximum height of the three kids. The Gaussian blur has a standard deviation of 1.5.

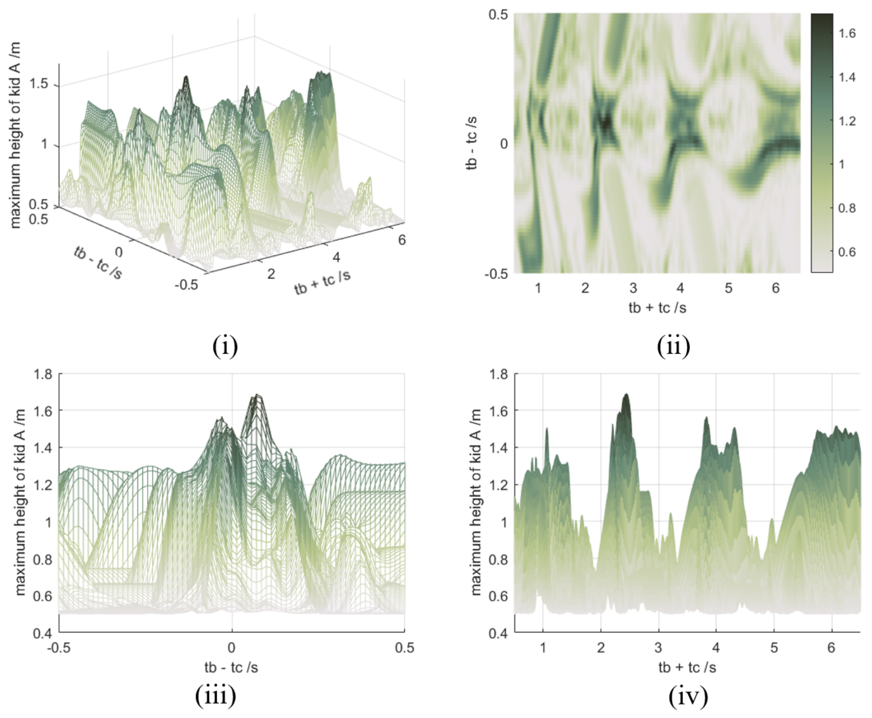


Figure 9: (i) The 3D display of the maximum height of Kid A in the target region with the potential global maximum height. (ii) The top view of (i). (iii) The perspective of (i) from the side of tb-tc axis. (iv) The perspective of (i) from the side of tb+tc axis.

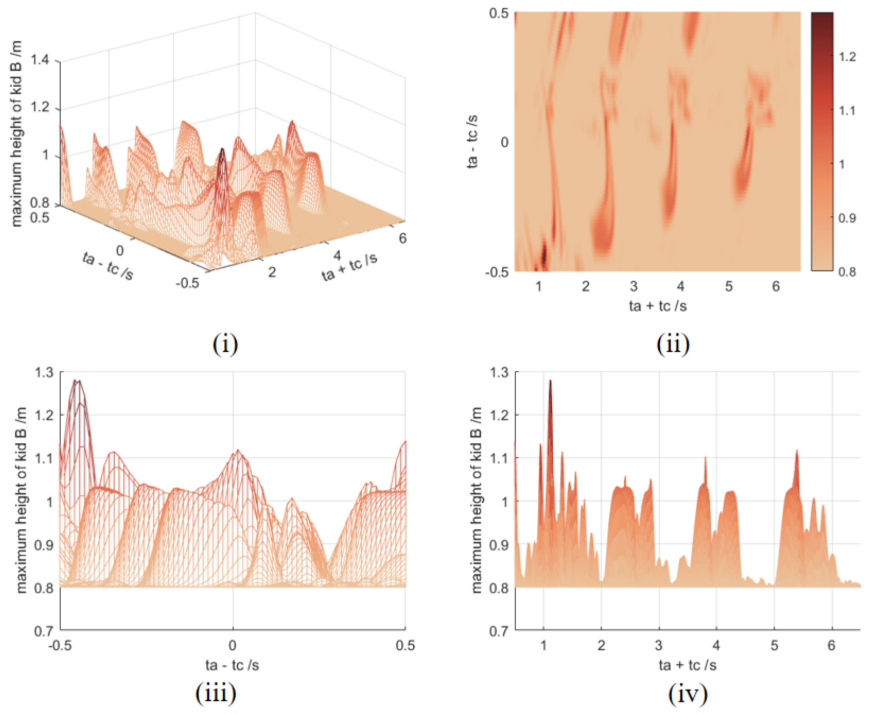
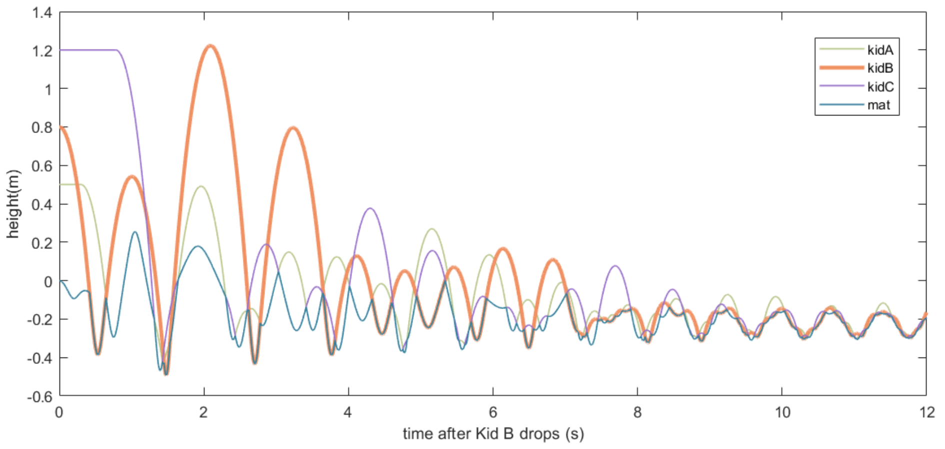


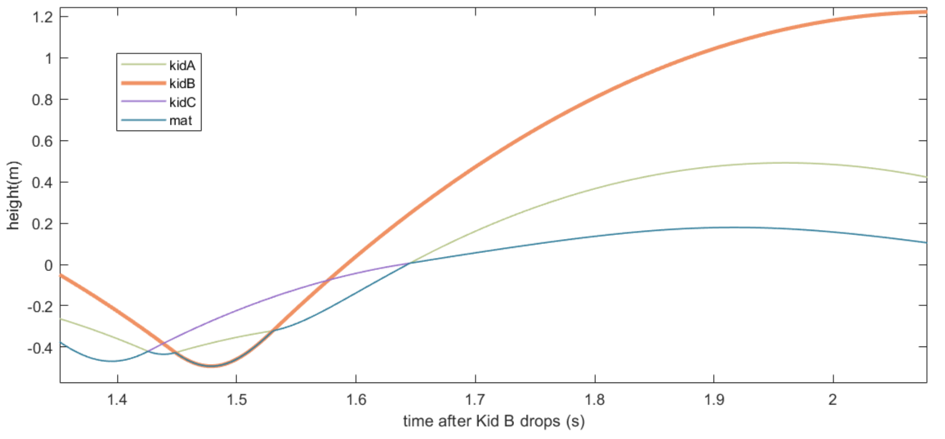
Figure 10: (i) The 3D display of the maximum height of Kid B in the target region with the potential global maximum height. (ii) The top view of (i). (iii) The perspective of (i) from the side of tb-tc axis. (iv) The perspective of (i) from the side of tb+tc axis.

4.6 Secret of Bouncing High

To elaborate on how the kid can be bounced so high, we take the maximum height condition of kid B, and plot the height-time function of all entities, as shown in Figure 11.



(a)



(b)

Figure 11: The relationship between the time after Kid B drops and the height of all entities in the system. (a) The displacement in the whole process of simulation. (b) The process before the Kid B is bounced up to 1.2m. It is clear that before Kid B is bounced by the mat, the mat is pressed by Kid C and kid A respectively for around 1.4s.

After several cycles, all entities began exhibiting periodic oscillations with diminishing amplitude. The maximum height is achieved only when the heaviest child, Kid 3, jumps down and applies pressure to the mat. Upon closer examination of the sequence prior to Kid B reaching his maximum height, as depicted in Figure 10 (b), it becomes evident that the sustained compression from Kid A and C before the descent of Kid B, coupled with the high-speed impact of Kid B, significantly contributes to this peak height. It's worth noting that the energy stored in the mat is directly proportional to the fourth power of the displacement, with a mere 0.4 meters of displacement yielding a substantial amount of energy.

5 Results and Discussion

5.1 Setting the V-shape model of the mat

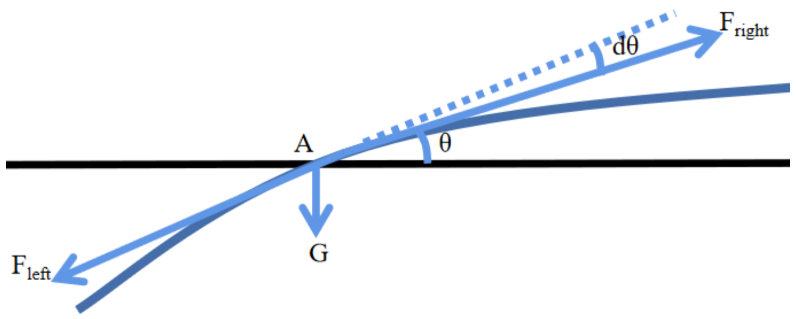


Figure 12: The free-body diagram of a single point on the elastic mat.

According to the free-body diagram of a single point on the elastic mat, we denote the horizontal force with F_H and the surface density as σ.

As the forces act on the center of the mat, the horizontal force on the edge is considered to be uniform, and the horizontal force over the whole mat is uniform. Thus, the vertical force equilibrium is

$$2\pi r\sigma\frac{\partial^2 x}{\partial t^2} = F_H(tan\theta - tan(\theta + d\theta)) - 2\pi r\sigma g$$

It can be simplified into

$$\frac{\partial^2 x}{\partial t^2} = -\frac{F_H}{2\pi r\sigma}\frac{\partial^2 x}{\partial r^2} - g$$

As a second order partial differential equation, it has gone beyond our capacity to solve it. So we assume that the weight of the elastic mat is negligible, i.e. $\sigma = 0$. Thus the differential equation can be simplified into

$$\frac{\partial^2 x}{\partial r^2} = 0, \; \frac{\partial^2 x}{\partial t^2} = -g$$

It is manifested that only the gravitational force, together with other possible external forces act on the mat as on independent points. Also, the whole mat is pulled straight with second derivative equal to zero.

5.2 The time and supporting force discussion

The contacting times between kid and the mat gained in previous parts are respectively 0.2645s, 0.2629s, 0.2512s. After checking videos of trampoline park frame by frame, it's found that for a video of 24 frames, the bouncing process includes 6-7 frames, which is a 0.250s-0.292s time period. Therefore, all the time periods are considered reasonable. Besides, the stretching forces generated by the kids are respectively 236N, 343N, and 473N, which are between 80% and 100% of the kids' weight respectively, which are also realistic. The frames from the video used to estimated the contacting time are shown in Figure 13 below.

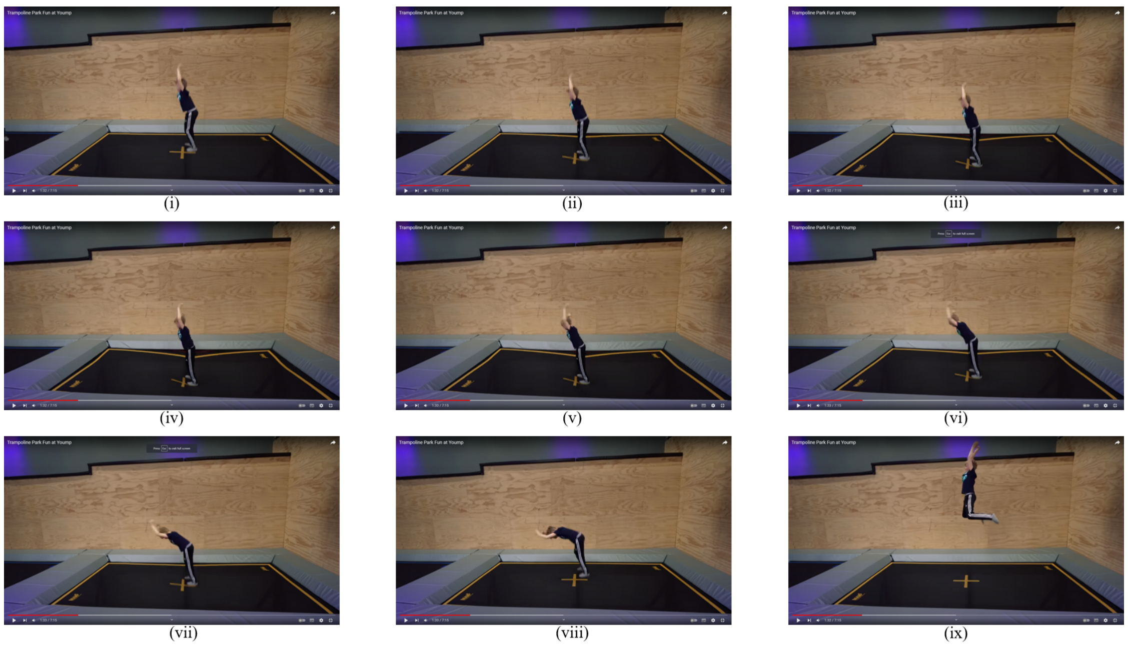


Figure 13: One example for the frames used to evaluate the bouncing time of an ordinary kid on the Trampoline. Picture (i) to (viii) shows the whole process of a kid contacting the trampoline by time. In (i) the kid just contacts with the trampoline, in (vii) the trampoline is about to return to equilibrium position and in (viii) they have separated. We can estimate the contacting time as 6.5 frames, equivalent to 0.271s. The frame (ix) shows the maximum height, which is approximately 1m-1.5m, which corresponds with Kid C. The frames are shot from ref[8].

5.3 The choice of standard deviation of Gaussian filter

In our study, we observed that when computing the maximum height, potential artifacts in the simulation model could lead to abrupt changes in the obtained height, even when the time step was reduced to 0.01 seconds, which is significantly smaller than the motion period in the single-person scenario. This phenomenon is illustrated in the first part of Figure 14. To address this issue, we applied Gaussian filtering to the computed array of maximum values, aiming to mitigate noise and reduce sudden variations in the resulting heights, thereby visualizing the trend of the maximum height and enhancing the validity of the maximum height.

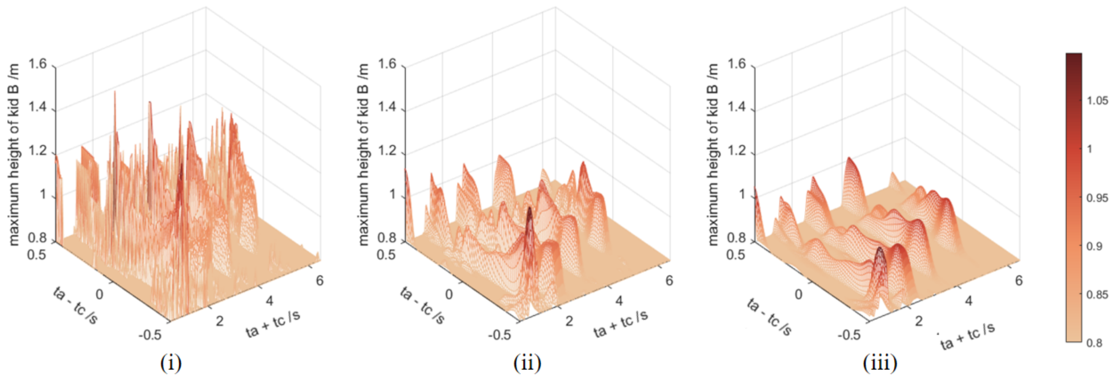


Figure 14: The 3D display of the maximum height of Kid A in the target region with the potential global maximum heightiwithout Gaussian filter; (ii) with Gaussian filter with a standard deviation of 1.5; (iii) with Gaussian filter with a standard deviation of 3.0

We found that when using a standard deviation of 3 for Gaussian filtering, as depicted in the third part of Figure 14, a substantial loss of local features occurred, with significant effects on the values at the maxima. Therefore, we opted for a Gaussian filter with a standard deviation of 1.5, achieving noise reduction while preserving features to the greatest extent possible. This approach resulted in a more representative estimation of the maximum height.

6 Model Evaluation

6.1 Strengths

1. Various factors (e.g. air resistance, damping force of the mat, and etc.) are taken into consideration when building up the model.
2. Data from the real life are applied in the calculation and simulation process, leading to a result closer to the real life.
3. Simulating the motion of the kids with a program, generating a more well-rounded result.
4. This study applied special algorithms like the Gaussian blurring to enhance the validity of the results.

6.2 Weaknesses

1. Since the situation of many people moving on the trampoline may resemble a chaotic phenomenon, the small deviation in the initial state setting, together with flaw in the physics engine, may bring errors.
2. Using a single stretching force to denote the stretching is not reliable enough. In section 4.6, the amplitude has decreased much, which contradicts the real situation, indicating that such a stretching force is not sustainable in some specific conditions.
3. Due to the limitations of the physics engine, the situation that delay time <0 is not discussed in this study, causing potential maximum height to be left out of solution.

6.3 Comparison with product in use

In our analysis, we observed that the maximum height from the results of all test cases reaches approximately 1.69 meters. This value falls below the established guard height of 1.83 meters, which is commonly found in mature products of similar dimensions [11]. It's worth noting that products available in the market have undergone rigorous safety evaluations, thereby indirectly ensuring the credibility of our predictive model. However, when considering the problem from a different perspective, we must take into account the practical application of safety nets, which may exhibit elasticity and may not consistently maintain a fixed height of 1.83 meters. Since our calculations are centered on the body's gravitational point, the proximity of the highest jumping height to the safety net's designated height poses a notable safety risk. Therefore, we recommend that the vendor explore the possibility of raising the safety net height to enhance safety.

7 Conclusion

As the trampoline jumping becomes more and more popular in today's world, it gets extremely important to pay attention to the safety issues. This study researches into the maximum height that a kid can reach under the condition that three kids of different masses are jumping simultaneously on the trampoline. Through our analysis and discussion, a physical model is made based on the parameters of the real-life trampolines. A physics engine is built up with codes based on the given data and the basic physical formulas. The time when the kids start to fall strongly affects the resulting maximum height that a target kid can reach. With the masses of the three kids being 25kg, 40kg, and 50kg, the maximum height of them are calculated to be 1.6890m, 1.2810m, and 1.3668m, respectively. Each maximum height is gained under the condition where the target kid falls on the mat with a maximum speed and that the other two kids are exactly at the lowest point. The results of our study offers an effective method to the designers of trampolines to control the maximum height that the user may reach during the exercise, leading to a safer exercising environment.

References

[1] Garden Trampolining. *ROSPA,* August, 2015.

[2] Sylvia J. Gautreau et. al. Home versus indoor trampoline park injuries: A four-year review of hospital admissions, associated costs, and impact on patients. *Trauma Case Reports*, Vol 31, 2021. `https://www.sciencedirect.com/science/article/pii/S2352644020301096`

[3] Trampoline Videos `https://www.youtube.com/watch?v=O0ZbDeN5AK4&t=37s&ab_channel=FamilyPlaylab\`

[4] Trampoline Videos `https://www.youtube.com/watch?v=zdFS8K0fJJI&t=41s&ab_channel=FamilyPlaylab\`

[5] Safety net in practical use `https://item.m.jd.com/product/10077284531727.html\`

[6] Determinant of spring coefficient `https://www.tokaibane.com/`

[7] `https://item.jd.com/69989606193.html#crumb-wrap`

[8] `https://matweb.com`

[9] FÉDÉRATION INTERNATIONALE DE GYMNASTIQUE. David Eager, Shilei Zhou, Karlos Ishac, Imam Hossain, Adam Richards and Lisa N. Sharwood. Apparatus Norms. October 1, 2019.

[10] Investigation into the Trampoline Dynamic Characteristics and Analysis of Double Bounce Vibrations. *Sensors*, April 11, 2022.

[11] Trampoline used as the model `https://item.m.jd.com/product/10084696293385.html\`

A iterA.py

This program serves as a physical engine and simulates the kinetic performance of kids and mat in the whole process within 12s. In the simulation some kids, excluding the one to be discussed, falls down from their individual maximum height after a certain time. Finally, the maximum height for every time interval is output.

```
import math
import numpy as np
import pandas as pd
import time
#Define the basic amounts
MASS=np.array([25,40,50])
Fa=236
Fb=343
Fc=473
X=[0,0,0,0]
V=[0,0,0,0]
MAXHeight=[[],[],[]]
t=0
g=9.81 # acceleration of gravity
m0 = 2 # equivalent mass of the mat
rm = 2.5 # radius of the mat
ks = 712940 # spring constant of all springs
Ca = 2.3#
C_d=9.40
MembraneWeight=2#
TIMEINTERVAL=0.001
TOTALLENGTH=12
Combination=[False,False,False]

def air_res(X,V):
    return -(math.pi*Ca*rm**3*V[3]*abs(V[3])/math.sqrt(rm**2+X[3]**2)
    /6)

def damping(V):
    return -C_d*V[3]

def returnForce(X):
    #print (X[3])
    return -ks*(math.sqrt(rm**2+X[3]**2)-rm)*X[3]/math.sqrt(rm**2+X
    [3]**2)

def Gravity(M):
    return -g*M

def collisionCheck(X):
    if X[3]-X[2]>10^-4:
        if X[3]-X[1]>10^-4:
            if X[3]-X[0]>10^-4:
                return True
    return False

def MembraneWeight(Combination):
    Weight=2
    for index in [0,1,2]:
        if Combination[index]==True:
            Weight+=MASS[index]
    return Weight
```

```python

def collisionJudgement(X0,XAfter,V0,VAfter,Combination,timeInterval):
    if collisionCheck(XAfter)==False:
        return VAfter,XAfter,Combination
    else:
        X_modified=XAfter
        V_modified=VAfter
        for index in [0,1,2]:
            if X_modified[index]<X_modified[3]-10**(-4):
                Combination=[False,False,False]
                Combination[index]=True
                #collisionTime=(X0[index]-X0[3])/(V0[index]-V0[3]) TO
    BE IMPROVED
                V_modified[index]=V_modified[3]=(VAfter[3]*m0+VAfter[
    index]*MASS[index])/(m0+MASS[index])
                X_modified[index]=X_modified[3]=(XAfter[3]*m0+XAfter[
    index]*MASS[index])/(m0+MASS[index])
    return V_modified,X_modified,Combination


def stretching(Combination):
    Weight=0
    if Combination[0]==True:
        Weight+=Fa
    if Combination[1]==True:
        Weight+=Fb
    if Combination[2]==True:
        Weight+=Fc
    return Weight

def Acceleration(X,V,combination):
    return (stretching(combination)+returnForce(X)+Gravity(
    MembraneWeight(combination))+damping(V)+air_res(X,V))/
    MembraneWeight(combination)

def update(Combination,X,V,TIMEINTERVAL,bF,cF):
    X0=X
    V0=V
    for index in [0,1,2,3]:
        X[index]+=0.5*TIMEINTERVAL*V[index]
    matAcce=Acceleration(X,V,Combination)
    for index in [0,1,2]:
        if Combination[index]==False and not ((index==1 and bF==False)
     or (index==2 and cF==False)) :
            V[index]-=g*TIMEINTERVAL
        elif Combination[index]==True:
            V[index]+=matAcce*TIMEINTERVAL
    V[3]+=matAcce*TIMEINTERVAL
    for index in [0,1,2,3]:
        X[index]+=0.5*TIMEINTERVAL*V[index]
    V,X,Combination=collisionJudgement(X0,X,V0,V,Combination,
    TIMEINTERVAL)
    return V,X,Combination

def Attempt(bIndex,cIndex,ExperimentLength):
    V=[0,0,0,0]
    X=[0.5,0.8,1.2,0]
    MAXH=[-1,-1,-1]
    bF=cF=False
    Combination=[False,False,False]
    for exIndex in range(math.ceil(ExperimentLength/TIMEINTERVAL)):
        if TIMEINTERVAL*exIndex>0.03*bIndex:
            bF=True
        if TIMEINTERVAL*exIndex>0.03*cIndex:
            cF=True
        V,X,Combination=update(Combination,X,V,TIMEINTERVAL,bF,cF)
        for index in range(3):
            if MAXH[index]<X[index]:
                MAXH[index]=X[index]
    return MAXH

T1=time.time()
for BHEIGHT in range(200):
    for index in range(3):
        MAXHeight[index].append([])
    for CHEIGHT in range(200):
        MAXH=Attempt(BHEIGHT,CHEIGHT,TOTALLENGTH)
        for index in range(3):
            MAXHeight[index][BHEIGHT].append(MAXH[index])
    print(BHEIGHT+1,"finished")
    print("Estimated time",(time.time()-T1)/(BHEIGHT+1)*(199-BHEIGHT)/60,"minutes")


AH=np.array(MAXHeight[0])
BH=np.array(MAXHeight[1])
CH=np.array(MAXHeight[2])
data1 = pd.DataFrame(AH)
data1.to_csv('AH_C.csv')
data1 = pd.DataFrame(BH)
data1.to_csv('BH_C.csv')
data1 = pd.DataFrame(CH)
data1.to_csv('CH_C.csv')

```

B iterASpecified.py

This program is similar, but it depicts the rectangle region after coordinate transformation.

```
...............(same as iterA.py before its line 98)...............
def Attempt(bIndex,cIndex,ExperimentLength):
    V=[0,0,0,0]
    X=[0.5,0.8,1.2,0]
    MAXH=[-1,-1,-1]
    bF=cF=False
    Combination=[False,False,False]
    for exIndex in range(math.ceil(ExperimentLength/TIMEINTERVAL)):
        if TIMEINTERVAL*exIndex>0.01*bIndex:
            bF=True
        if TIMEINTERVAL*exIndex>0.01*cIndex:
            cF=True
        V,X,Combination=update(Combination,X,V,TIMEINTERVAL,bF,cF)
        for index in range(3):
            if MAXH[index]<X[index]:
                MAXH[index]=X[index]
    return MAXH

T1=time.time()
for BHEIGHT in range(424):
    for index in range(3):
        MAXHeight[index].append([])
    for CHEIGHT in range(71):
        #print(0.707*(BHEIGHT-CHEIGHT+70),0.707*(BHEIGHT+CHEIGHT))
        MAXH=Attempt(0.707*(BHEIGHT+CHEIGHT),0.707*(BHEIGHT-CHEIGHT
    +70),TOTALLENGTH)
        for index in range(3):
            MAXHeight[index][BHEIGHT].append(MAXH[index])
    print(BHEIGHT+1,"finished")
    print("Estimated time",(time.time()-T1)/(BHEIGHT+1)*(423-BHEIGHT)
    /60,"minutes")

AH=np.array(MAXHeight[0])
BH=np.array(MAXHeight[1])
CH=np.array(MAXHeight[2])
data1 = pd.DataFrame(AH)
data1.to_csv('AH_C.csv')
data1 = pd.DataFrame(BH)
data1.to_csv('BH_C.csv')
data1 = pd.DataFrame(CH)
data1.to_csv('CH_C.csv')

```

C iterBMax.py

This program can figure out the position of all kids and mat in the whole process of function.

```
  Rep=[[],[],[],[]]
..........(similar with iterA.py before its line 98).........
def Attempt(bIndex,cIndex,ExperimentLength):
    V=[0,0,0,0]
    X=[0.5,0.8,1.2,0]
    MAXH=[-1,-1,-1]
    bF=cF=False
    Combination=[False,False,False]
    for exIndex in range(math.ceil(ExperimentLength/TIMEINTERVAL)):
        if TIMEINTERVAL*exIndex>0.03*bIndex:
            bF=True
        if TIMEINTERVAL*exIndex>0.03*cIndex:
            cF=True
        V,X,Combination=update(Combination,X,V,TIMEINTERVAL,bF,cF)
        for index in [0,1,2,3]:
            Rep[index].append(X[index])
        for index in range(3):
            if MAXH[index]<X[index]:
                MAXH[index]=X[index]
                if MAXH[index]>1.2:
                    print(exIndex)
    return MAXH

MAXh=Attempt(10,26,12)
print(MAXh)
```

```
AH=np.array(Rep)
data1 = pd.DataFrame(AH)
data1.to_csv('REP.csv')

```

D CalculateCN.m

By solving ODE, the algorithm gives out the damping coefficient c and the stretching force N of all kids.

```
    clear,clc

%% find the value of C
result1 = Generate_down(9.4, 50, -4.6655923);
min1 = Take_min(result1);
index = find(result1(:,2) == min1);
m3down = result1(1:index,:);
plot(m3down(:,1),m3down(:,2))
%%
n = zeros(1,21);
c = 8:0.1:10;
for i = 1:21
    result1 = Generate_down(c(i), 25, -2.90);
    v1 = Take_dev(result1);
    n1 = v1 / 2.90;

    result2 = Generate_down(c(i), 40, -3.77316);
    v2 = Take_dev(result2);
    n2 = v2 / 3.77316;

    result3 = Generate_down(c(i), 50, -4.6655923);
    v3 = Take_dev(result3);
    n3 = v3 / 4.6655923;
    n(i) = (n1 + n2 + n3) / 3;
end
[~,index] = min(abs(n-0.8));
disp(c(index))

%% find the value of N
clear,clc
c = 9.4;
%for m1
m2 = 40;
judge1 = zeros(1,101);
v10 = -sqrt(2*9.81*0.8);
v11 = v10*(m1)/(m1+2);
res1 = Generate_down(c, m1, v11);
min1 = Take_min(res1);
index1 = find(res1(:,2) == min1);
for N1 = 200:300
    res11 = Generate_down_amend(N1, m1, min1);
    v1 = Take_dev(res11);
    judge1(N1-199) = abs(abs(v1/v10)-1);
end
n1 = find(judge1 == min(judge1))+199;
res11 = Generate_down_amend(n1, m1, min1);
disp(n1)
```

```

%% find the value of N
clear,clc
c = 9.4;
%for m2
m2 = 40;
judge2 = zeros(1,101);
v20 = -sqrt(2*9.81*0.8);
v21 = v20*(m2)/(m2+2);
res2 = Generate_down(c, m2, v21);
min2 = Take_min(res2);
index2 = find(res2(:,2) == min2);
for N2 = 300:600
    res21 = Generate_down_amend(N2, m2, min2);
    v2 = Take_dev(res21);
    judge2(N2-299) = abs(abs(v2/v20)-1);
end
n2 = find(judge2 == min(judge2))+299;
res21 = Generate_down_amend(n2, m2, min2);
disp(n2)

%% find the value of N
clear,clc
c = 9.4;
%for m3
m3 = 50;
judge3 = zeros(1,101);
v30 = -sqrt(2*9.81*1.2);
v31 = v30*(m3)/(m3+2);
res3 = Generate_down(c, m3, v31);
min3 = Take_min(res3);
index3 = find(res3(:,2) == min3);
t3 = res3(index3,1);
for N3 = 300:600
    res31 = Generate_down_amend(N3, m3, min3);
    v3 = Take_dev(res31);
    judge3(N3-299) = abs(abs(v3/v30)-1);
end
n3 = find(judge3 == min(judge3))+299;
res31 = Generate_down_amend(n3, m3, min3);
disp(n3)

%% relevent functions
function v1 = Generate_down(c, mk, v0)
% mass, velocity, output(t,res)

m0 = 2; % equivalent mass of the mat
rm = 2.5; % radius of the mat
m = m0 + mk; % total equivalent mass
ks = 7.1294 * 10^5; % spring constant of all springs
Ca = 2.3;
g = 9.81;   % acceleration of gravity

% Define the time range and initial conditions
tspan = [0 5];  % Time range from 0 to 10
x0 = [0; v0];  % Initial displacement and initial velocity

% Define an anonymous function to represent the differential equation
f = @(t, x) [x(2); 1/m * ((-pi * Ca * rm^3) / (6 * sqrt(rm^2 + x(1)^2)
    ) * x(2) * abs(x(2)) - c * x(2) + ks * (sqrt(rm^2 + x(1)^2) - rm)
    * (-x(1)) / (sqrt(rm^2 + x(1)^2)) - m * g)];

options = odeset('MaxStep', 0.0001);  % Set the maximum time step to
    0.01 (adjust as needed)
[t, x] = ode45(f, tspan, x0, options);  % Pass the options to the
    ode45 function
v1 = [t, x(:, 1)];
end

function v1 = Generate_down_amend(N, mk, h0)
% mass, velocity, output(t,res)

m0 = 2; % equivalent mass of the mat
rm = 2.5; % radius of the mat
m = m0 + mk; % total equivalent mass
ks = 7.1294 * 10^5; % spring constant of all springs
Ca = 2.3;
c = 9.4;
g = 9.81;   % acceleration of gravity

% Define the time range and initial conditions
tspan = [0 5];  % Time range from 0 to 10
x0 = [h0; 0];  % Initial displacement and initial velocity

% Define an anonymous function to represent the differential equation
f = @(t, x) [x(2); 1/m * ((-pi * Ca * rm^3) / (6 * sqrt(rm^2 + x(1)^2)
    ) * x(2) * abs(x(2)) - c * x(2) + ks * (sqrt(rm^2 + x(1)^2) - rm)
    * (-x(1)) / (sqrt(rm^2 + x(1)^2)) - m * g+N)];
options = odeset('MaxStep', 0.0001);  % Set the maximum time step to
    0.01 (adjust as needed)
[t, x] = ode45(f, tspan, x0, options);  % Pass the options to the
    ode45 function
v1 = [t, x(:, 1)];
end

function dev = Take_dev(res)
i = 1;
while true
    if res(i+1, 2) > res(i, 2) && res(i, 2) >= -10^(-4)
        dev = (res(i+1, 2) - res(i, 2)) / (res(i+1, 1) - res(i, 1));
        break;
    end
    i = i + 1;
end
end

function min = Take_min(res)
i = 1;
while true
    if res(i+1, 2) > res(i, 2)
        min = res(i,2);
        break;
    end
    i = i + 1;
end
end

```