DESIGN A ROLLER COASTER

SELECTED PROBLEM: B

Team 405

ABSTRACT

We present an analysis and design of a roller coaster that delivers a maximal amount of joy to the passengers without compromising their safety. We model track as two-dimensional parametric curves to achieve maximal fidelity to the mathematical model and precisely predict conditions during the ride. To ensure safety we measured the forces impacting on passengers in every point of the construction. We surveyed to determine which elements of the track are especially exciting. Based on this we designed a roller coaster made of simple shapes such as clothoid, parabola and Gaussian curve that delivers exciting and safe adventure.

1 Introduction

Roller coasters are a very popular attraction in amusement parks. Most of them consist of a combination of typical elements, such as vertical loops, hills or steep falls. On those elements, visitors are subjected to unusual g-forces, accelerations, heights, and orientations, which makes the experience unique and fun.

However, the more extreme the ride, the more important it is to make sure it is also safe. This situation is very unnatural for the human body and might cause some health concerns. Due to this, the first factor we are discussing is safety. We make sure to eliminate any situations that might be potentially dangerous from our model.

We then try to determine which elements and in what combinations are the most exciting for potential visitors. We define a simple theoretical model of motion for the roller coaster car and use it to simulate motion across the chosen elements. Finally, we develop and present the model – consisting of a combination of those elements – that is a successful combination of safety and maximal excitement.

2 Safety

Modern roller coaster carts and tracks are made out of hardened steel, aluminum, and other durable materials. Nowadays the main limitations of roller coasters are the capabilities of the human body. The biggest safety concern during roller coaster construction is the gravitational force equivalent (G-Force) that passengers will experience while accelerating.

However, rapid acceleration is part of human life. An uninhibited sneeze after sniffing ground pepper can cause head acceleration around 2.9 G. (M et al. [5])

Although the impact of acceleration on the human body varies from person to person, it is especially dangerous for kids, pregnant women and people with heart-related illnesses. In extreme cases, this may even lead to passengers death. Example of such roller coaster is "Euthanasia Coaster" [9].

In our analysis, we assume that all passengers are healthy adults. To make the roller coaster ride not only bearable but also pleasant, we restrict the G-Force up to 4G.

The expected effects of this acceleration on the human body are different for each direction.[4]

2.1 Vertical axis G_Z

- -2 to -3 G_Z: Headache, edema of eyelids, petechial hemorrhages in face and neck
- -1 G_Z: Sens of pressure and fullness in the head, congestion of eyes
- +1 G_Z: Erect terrestrial posture
- +3 to 4 G_Z: Problems with vision after 3-4 sec; arterial oxygen saturation below 93%

2.2 Horizontal axis G_X

- $\pm$1G$_X$: Slight increase in abdominal pressure; respiratory rate increases
- $\pm$2 to $\pm$ 3G$_X$ Difficulty in spatial orientation; Mechanical compression of the chest wall.

2.3 Lateral axis G_Y

Humans can tolerate high levels of lateral G's if properly restrained; otherwise, injury to the body can occur. D. [3] We assume that passengers are safely secured in their seats. Safety bars protect fragile parts of the body, such as the neck, from moving in an uncontrolled way.

Figure 1: Example of security bars protecting from neck injuries. Source: https://nypost.com/2015/05/21/my-2-minutes-of-terror-riding-six-flags-dizzying-new-roller-coaster/

3 Excitement of the ride

We have made a public questionnaire about importance of various aspects of a roller coaster ride and shared it on social media. The question was 'What elements are most important for you to find a rollercoaster exciting?'. We have received 30 anonymous answers.

Presented results show features indicated by respondents as the most important.

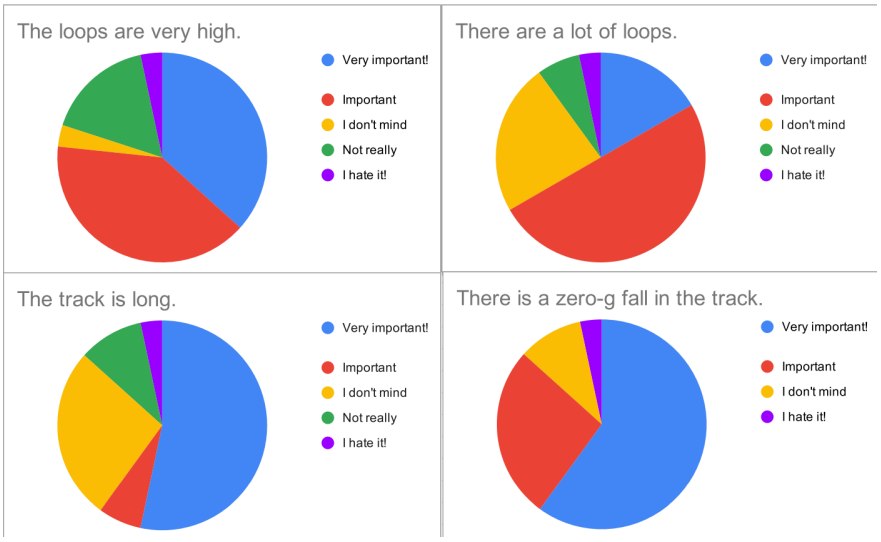

Figure 2: Results of our questionnaire.

We can clearly see that respondents show strong preference to both high and numerous loops. It is also very important for them that track includes zero-g fall. Results show that length of the track is also important factor.

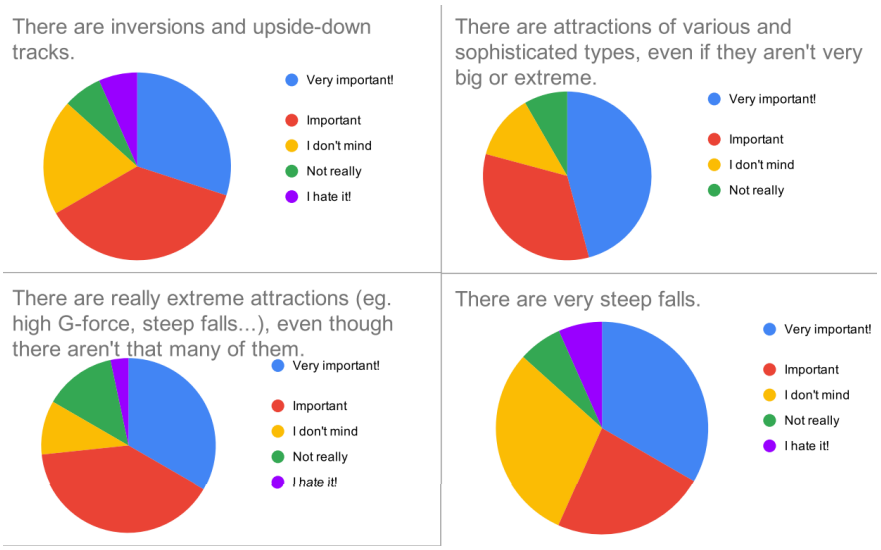

Figure 3: Rest of results of our questionnaire.

The vast majority of respondents also indicated that variety of attractions during ride is important to them, even if they are not connected with extreme experiences. On the other hand results show preference toward those extreme experiences.

Based on those results, we can see a demand for a long roller coaster that is made of a number of loops, zero-g fall and other smaller attractions that make ride surprising and trilling for the passengers.

4 Theory

4.1 A 2D model of car motion

Here we define a model of motion of the car along a 2D curve, given either as an implicit function $f(x, y) = 0$ or as parametric equations $x(\theta)$, $y(\theta)$. We will model this curve as a set of points (x_i, y_i). The closer the distance $\Delta r := \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2}$ between neighbouring points is to 0, the more accurate our model is.

The model also enforces that passengers are at all times tangential to the track. In this the way main axis where passengers experience G-force is the vertical axis (G_Z).

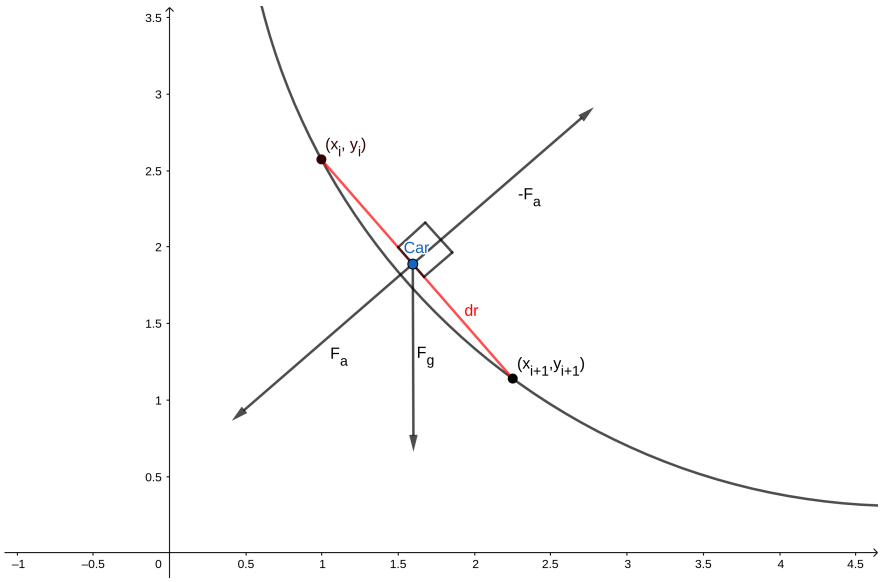

Figure 4: model visualisation

4.1.1 Energy losses

Let $\vec{F}_{diss}$ be a dissipative force in the direction opposite to the motion of the car. Then, we will calculate energy losses due to dissipative forces as:

$$\Delta E = F_{diss} \cdot \Delta r \tag{1}$$

Energy loss due to air resistance is equal to:

$$\Delta E_{air} = -F_{air}\Delta r = -\rho\frac{v^2 AS}{2}\Delta r \tag{2}$$

As we are dealing with rather huge velocities, we found that air resistance is usually large and cannot be neglected.

Energy loss due to rolling friction is equal to:

$$\Delta E_{rf} = -\mu F_{normal}\Delta r \tag{3}$$

where $\vec{s}$ is a vector normal to the surface, $s = 1$.

Since F_a is the centrifugal force and it must be normal to the surface, F_{air} is always tangent to the surface and might be neglected, and $\angle(\vec{F_g}, \vec{s})$ is defined as θ:

$$\Delta E_{rf} = -\mu F_{normal}\Delta r = -\mu\Delta r((\vec{F_a} + \vec{F_g}) \cdot \vec{s})\Delta r = -\mu\Delta r(F_a + F_g \cos\theta) \tag{4}$$

Energy loss due to air resistance is equal to:

$$\Delta E_{air} = -F_{air}\Delta r = -\rho\frac{v^2 AS}{2}\Delta r \tag{5}$$

Therefore, all energy losses can be described as:

$$\Delta E = -\mu mg\Delta x - \rho\frac{v^2 AS}{2}\Delta r \tag{6}$$

4.1.2 Energy balance and simulation steps

The potential energy of the car is equal to

$$E_p = Qy = Qx \tan\theta \tag{7}$$

where $Q = mg$. The kinetic energy is equal to

$$E_k = \frac{1}{2}mv^2 \tag{8}$$

We now consider a very small step along the curve. The car enters the step with a starting velocity v_0, tangent to the curve, and a starting available mechanical energy E_0. In general, energy is conserved, so after each step, starting with E_0, we have:

$$E_k + E_p = E_0 + \Delta E \tag{9}$$

where a negative amount of energy ΔE was dissipated and can no longer be used as the mechanical energy of the car.

5 Elements of the track

5.1 Included curves for attractions

We chose to simulate the motion on some of the most popular and exciting attractions, found in roller coasters around the world, in the way described above.

5.1.1 Clothoid loop

We use the clothoid loop as a model for a perfect vertical loop. Clothoid loops are used in roller coasters rather than simple circular or elliptical shapes, because of (reasons) (references).

Figure 5: A vertical loop on a roller coaster; source: https://en.wikipedia.org/wiki/Roller_coaster_elements#/media/File:SFMM-_Full_Throttle_4.JPG

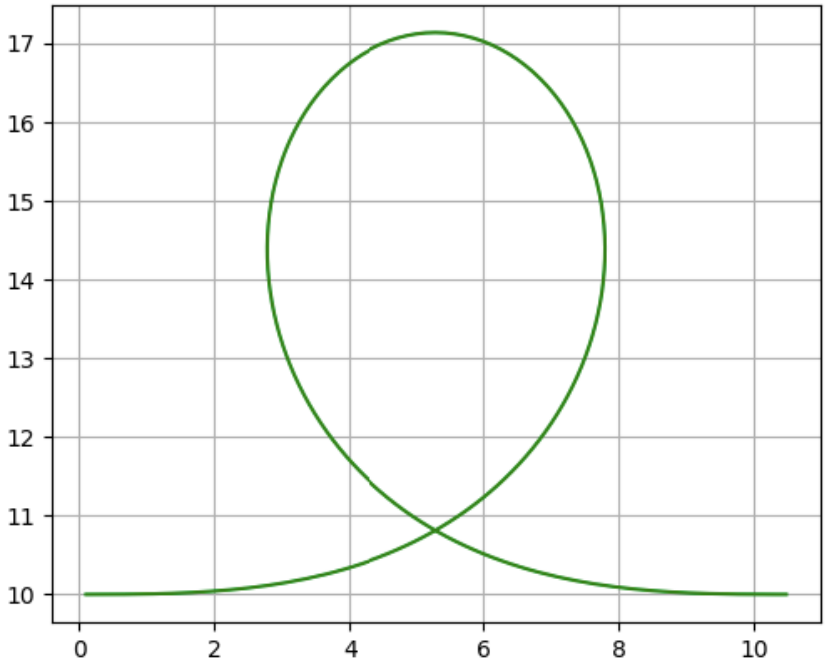

Figure 6: Clothoid loop generated for our simulation using Python

The clothoid loop is defined as a set of parametric equations in the following way:

$$x(\theta) = R * S(\theta) \tag{10}$$

$$y(\theta) = R * C(\theta) \tag{11}$$

where R is a scaling parameter and $S(\theta), C(\theta)$ are Fresnel integrals of the parameter θ (Müller [6]).

$$\begin{aligned} S(\theta) &= \int_0^\theta \sin(x^2)dx \\ C(\theta) &= \int_0^\theta \cos(x^2)dx \end{aligned} \tag{12}$$

which can be used in the following formula to compute curvature of a parametric curve. The curvature of the clothoid is equal to:

$$\frac{1}{R(x)} = \left|\frac{y"x' - y'x"}{((x')^2 + (y')^2)^{3/2}}\right| = \left|\frac{-\cos(\theta^2)\sin(\theta^2)2\theta - \sin(\theta^2)\cos(\theta^2)2\theta}{1}\right| \frac{1}{R} = 2|\theta|\frac{1}{R}. \tag{13}$$

The most important conclusion which we can obtain from equation (13) and [7], [2] is that this shape is more optimal for roller coasters, allows to build loops that are larger and more comfortable for passengers.

5.1.2 Gaussian curve

We use the Gaussian curve as a model for hills that well resembles elements found in real roller coasters.

Figure 7: A Gaussian-like hill on a roller coaster; source: http://www.coastergallery.com/japan/N11.html

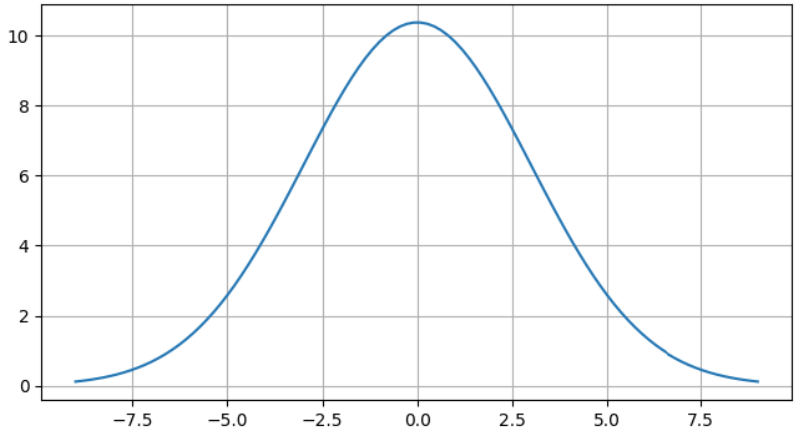

Figure 8: Gaussian curve generated for our simulation using Python

The Gaussian curve is defined as:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{14}$$

we calculate the value of its curvature given by equation Wikipedia [8]:

$$\frac{1}{R(x)} = \left|\frac{f''(x)}{(1+(f'(x))^2)^{\frac{3}{2}}}\right| = \left|\frac{-\sigma^2+x^2}{\sigma^4}\frac{f(x)}{(1+(-\frac{x}{\sigma^2}f(x)^2))^{\frac{3}{2}}}\right| \tag{15}$$

5.1.3 Parabolic curve

We use fragments of a parabolic curve as a model for a zero-g fall [10], the element that the highest number of respondents chose as crucial for the excitement of the ride.

Figure 9: A zero-g fall on a roller coaster; source: https://www.digitaltrends.com/cool-tech/5-places-to-reach-zero-gravity-within-earths-pull/

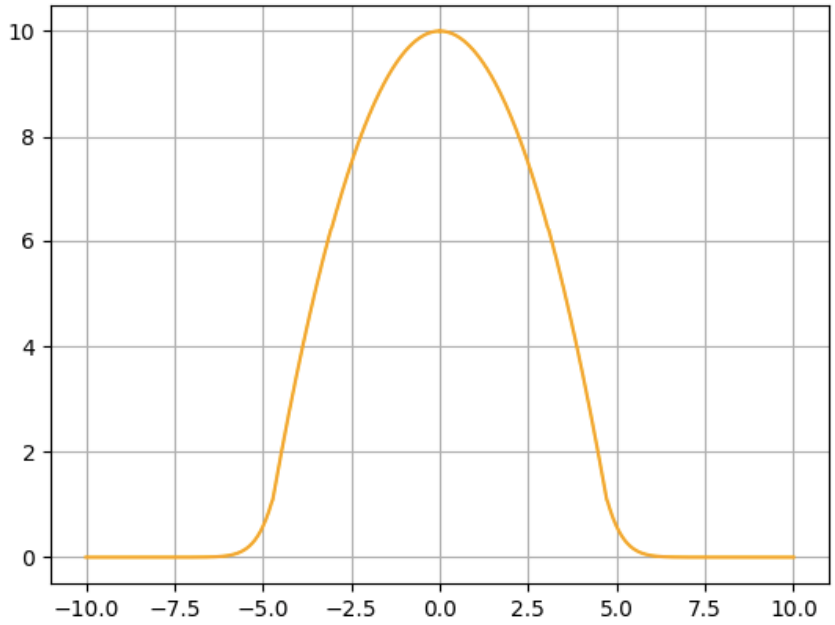

Figure 10: Parabola generated for our simulation using Python. Transition from flat to steep track is obtained by composing a parabola with a Gaussian curve.

Our parabolic function is defined as:

$$f(x) = ax^2 + bx + c \tag{16}$$

and we calculate the value of its curvature given by equation:

$$\frac{1}{R(x)} = \left| \frac{2a}{(4a^2x^2 + 4abx + 1 + b^2)^{3/2}} \right| \tag{17}$$

5.2 Rotation move

To move the roller coaster in the ground plane, we considered the rotation function that models the movement of the car moving in a circle. To calculate a normal force N in this simple case we have the following equations:

$$\begin{aligned} N &= mg\cos\phi + \frac{mv^2}{R}\sin\phi \\ \frac{\mathrm{d}N}{\mathrm{d}\phi} &= -mg\sin\phi + \frac{mv^2}{R}\cos\phi = 0 \\ \frac{\mathrm{d}N}{\mathrm{d}\phi}(\phi_0)g\sin\phi_0 &= \frac{v^2}{R}\cos\phi_0 \rightarrow \arctan\frac{gR}{v^2} = \phi_0 \end{aligned} \tag{18}$$

and to check which extreme point we have found:

$$\frac{\mathrm{d}^2N}{\mathrm{d}\phi^2}(\phi_0) = -mg\cos\phi_0 - \frac{mv^2}{R}\sin\phi_0 = -m\sin\phi_0(g\frac{gR}{v^2} + \frac{v^2}{R}) < 0 \tag{19}$$

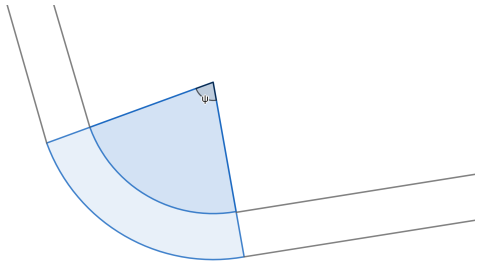

Figure 11: part of the circle representing the rotation of the car

Where: $\phi_0 \in (0, \frac{\pi}{2})$. From this we can easily obtain:

$$\Delta W = \left(mg\cos(\frac{Rg}{v^2}) + \frac{mv^2}{R}\sin(\frac{Rg}{v^2})\right)\mu\frac{R\psi}{n} + \rho\frac{ASv^2}{2}\frac{R\psi}{n} \tag{20}$$

where n is the number of steps in which we divided part of circle, and ψ is an angle indicated on the figure (11):

6 Simulations

6.1 Parameters and constants

1. $\mu = 0.0018$ from [1]
2. $g = 9.81\frac{m}{s^2}$ from Wikipedia
3. air resistance coefficient $A = 1.4$ from our simulation
4. car profile $S = 2m^2$ from our assumption
5. air density $\rho = 1\frac{kg}{m^3}$

6.2 Aerodynamics

To calculate the value of the drag coefficient we used Autodesk Flow Design. This computational fluid dynamics software allowed us to simulate flow around a car model with a velocity equal to 20 m/s. We obtained drag coefficient $A = 1.4$. We neglected effects of people sitting inside of car, as their shapes strongly vary so the influence on the drag coefficient is hard to predict.

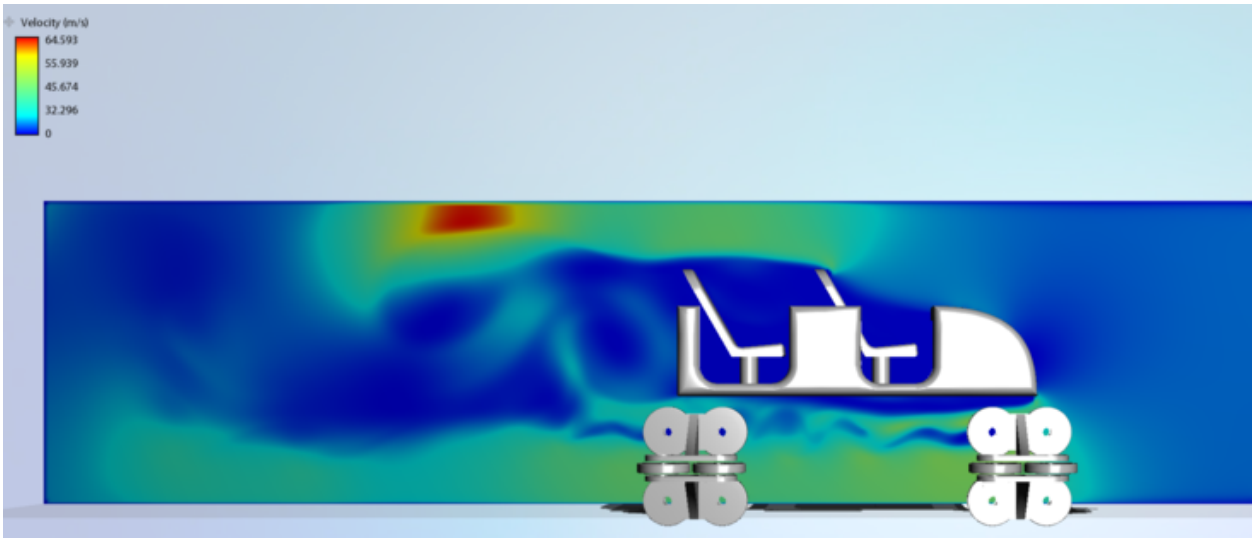

Figure 12: Results of flow simulation in Autodesk Flow Design with visible velocity. Car model source: https://www.cgtrader.com/free-3d-models/sports/toy/roller-coastor-car

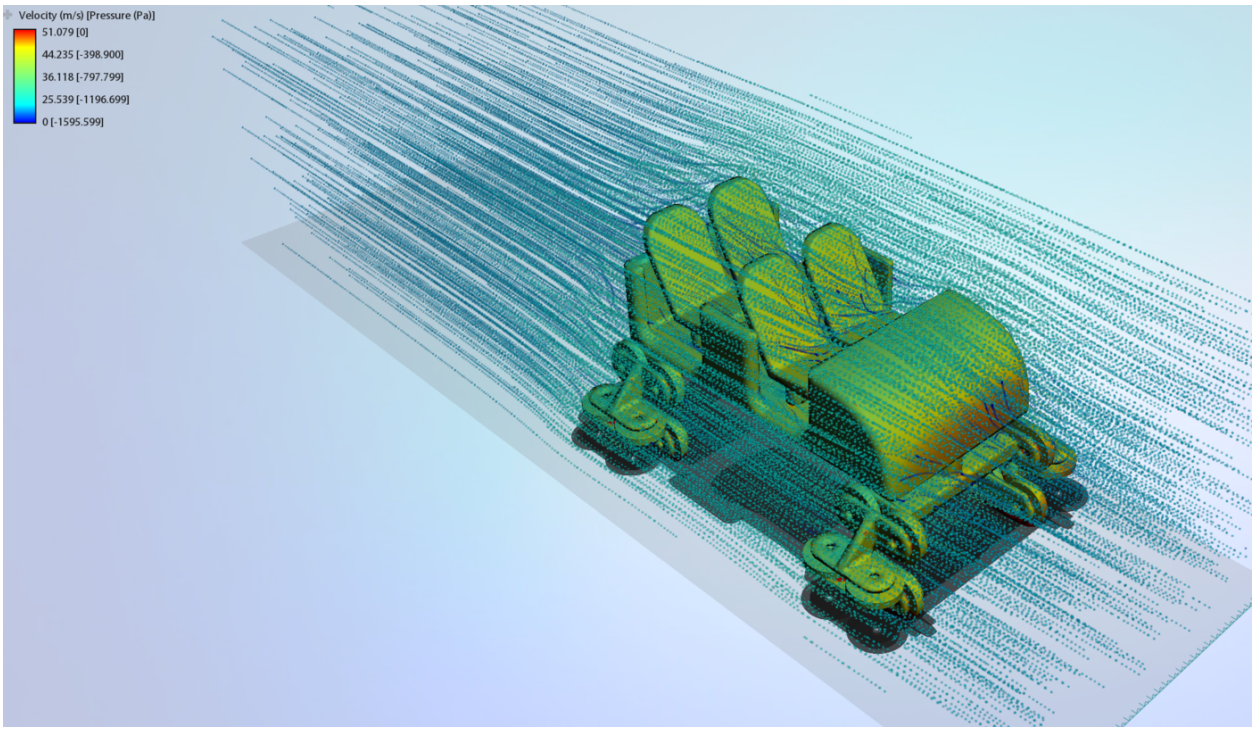

Figure 13: Results of flow simulation in Autodesk Flow Design. Visible both velocity and pressure distribution. Car model source: https://www.cgtrader.com/free-3d-models/sports/toy/roller-coastor-car

6.3 The track

The consists of three clothoid loops with height of respectively 24, 17.5 and 13 meters, two zero-g falls and several Gaussian hills. Attractions are connected with each other with flat track in order not too overload the passengers. Both 180 degrees turns are circular with radius equal to $5m$. Track ends with flat to break and turn to the starting point. The table describes ordered attractions that we have prepared with accurate track data obtained from simulations. A the beginning, we assume there is a very small (10 cm/s) initial velocity to start the ride.

No.	Type	v_0[m/s]	E_0 [kJ]	h_0 [m]	$\lvert G_{max}\rvert$ [$9.81m/s^2$]	Horizontal distance [m]
1.	Gaussian	0.1	161.9	30	1.07	9
2.	Parabolic	4.34	161.7	29	3.6	10
3.	Flat	23.08	151.9	1	1	2
4.	Clothoid	23.00	151.0	1	2.4	35.6
5.	Flat	20.9	125.5	1	1	2
6.	Clothoid	20.8	124.7	1	2.6	26.2
7.	Flat	19.3	107.5	0.1	1	2
8.	Gaussian	19.2	107.2	1	2.98	27
9.	Rotation	3.24	101.4	18.3	0.22	0
10.	Gaussian	3.08	101.1	18.3	2.27	-30
11.	Clothoid	17	96.1	2.95	2.51	-18.9
12.	Flat	16.2	87.7	2.95	1	-2
13.	Gaussian	16.1	88.6	2.95	0.65	-30
14.	Parabolic	2.1	83.4	15.3	1	-10
15.	Gaussian	11.3	82.2	8.7	1.59	-21

6.4 Overload control

In every point which we consider we can measure an overload from gravitational force and centripetal force in normal component from forward formula:

$$G = ||\vec{F_g}\cos\theta + \vec{F_a}|| = \pm\frac{Mv^2}{R} + Mg\cos\theta \tag{21}$$

We can neglect air resistance force, friction force and the transversal component of vectors due to 2 section. For our curves we have:

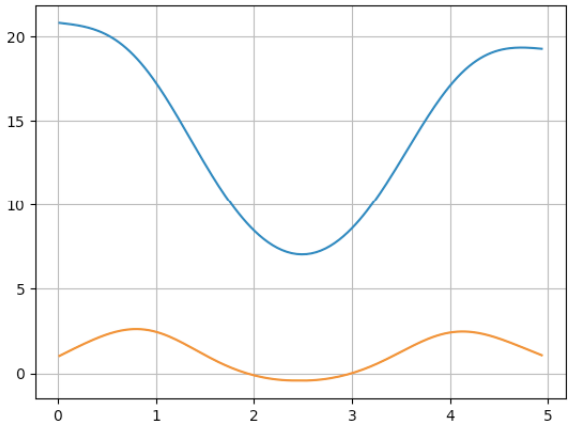

Figure 14: clothoid no. 2, velocity [m/(s*s) - blue line, overload [g] Rest of curves are analogous

In our model of rotation the most important thing is to protect the neck, so transversal component of centrifugal force gives the biggest impact, so we assume, that this is the only one force to consider.

$$G = ||\vec{F_a}||\frac{1}{mg} \tag{22}$$

where symbols are compatible with notation in 5.2.

6.5 Motion on the track

Using a model described in (4.1), we will now simulate the motion of the car on the curves from (5).

Energy balance is given as equation (9). After each step, we calculate $E_{new} = E_0 + \Delta E$ and $v_{new} = \sqrt{2 \cdot E_k/m}$. Then, we move by Δr along the curve and assign $v_0 := v_{new}$ and $E_0 := E_{new}$. In this way, we can define the entire motion along the curve.

Above we show an example plot the resulting G-force (yellow) and velocity (blue) change along the curve (clothoid loop 2).

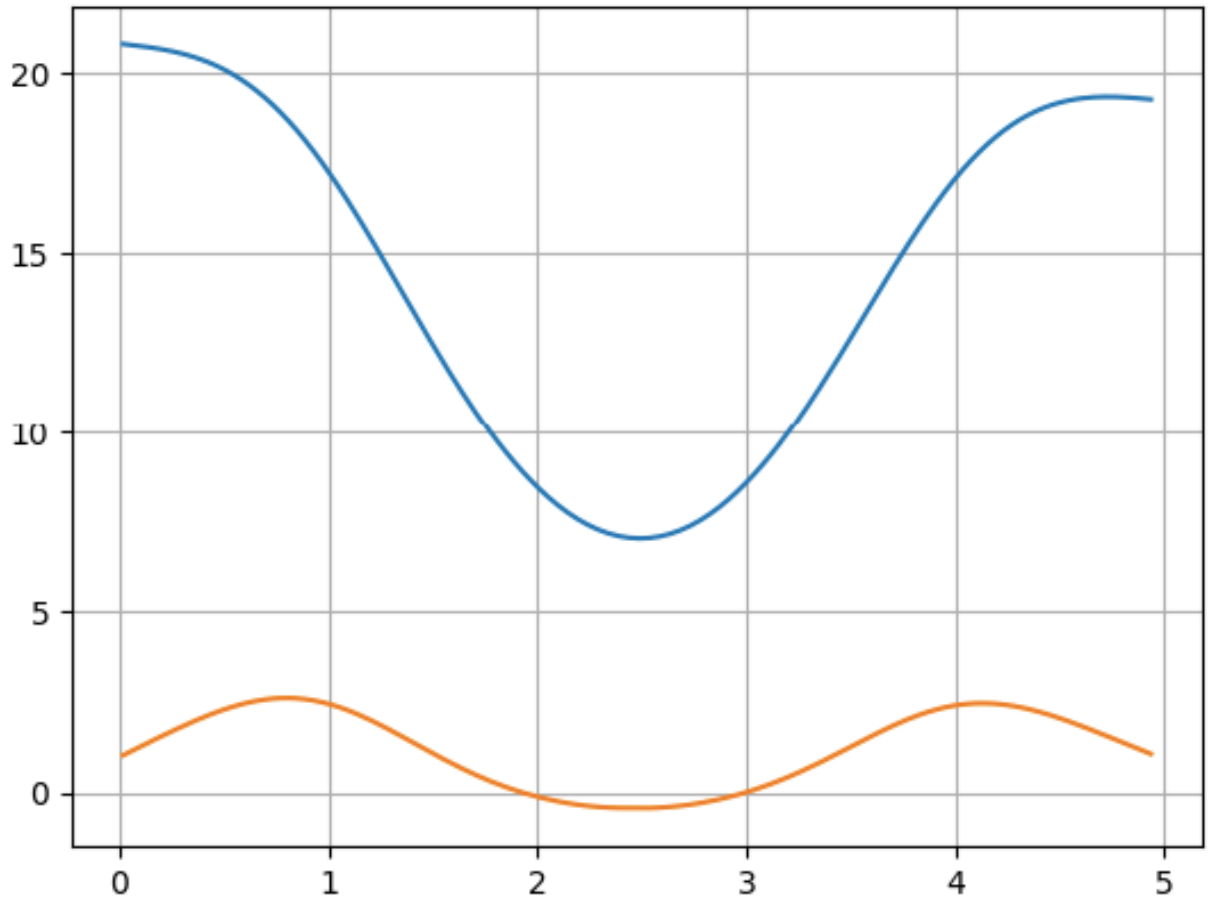

Figure 15

7 Strengths and weaknesses

7.1 strengths

The biggest advantage of our project is the variety of attractions and the opportunity to determine the exact parameters of various curves, which gives us unlimited possibility to attach new elements to the roller coaster.

7.2 weaknesses

The biggest weakness of our simulation is inability to see the curves in three dimensional space. We could not create crockscrew trick which might be composited by clothoids and rotation moves. We did not do this primarily because of the complexity of the problem.

In our work, the analysis of overloads acting on passengers is a very simple model that does not take into account the direction in which individual forces operate. To add more accurate descriptions, it would be necessary to determine for each movement of force along the transversal and normal component, which would reduce the transparency of the description.

8 Further discussion

Part of kinematics that we neglect is rotation around track, for example during entering corners. Additional loss of energy happens during angular acceleration connected with higher friction and change of effective cross-section that is taken into account while calculating drag force. Such rotational movement can also be attraction in itself, so it should forces impacting passengers should also be carefully investigated to ensure safety.

Another part that could be taken into account is that trains has length. That would strongly influence it's aerodynamics. During loops and corners mass center would not be in constant distance from tracks. This generates additional torque, which influences motion resisting forces.

9 Conclusions

Based on our research in both safety we determined the safe conditions based not only on magnitude of G-Force but also on it's direction. Thanks to this, we could ensure safety of the passengers during the ride.
To determine what makes roller coaster exciting we made an online survey. It's allowed us to have better information about what roller coaster users like (we've added larger G-zero loops). We recognize this as an important part of roller coaster design. The conditions of the task give us a lot of freedom in attractions selection, so we could place the elements better.

During our simulation, we didn't use all the energy at start, so with more time resources we could add more interesting loops as attractions. Their simulations would be analogous to those we did.

References

[1] URL https://www.coaster101.com/2011/10/24/coasters-101-wheel-design/.

[2] *Roller Coaster Physics: Clothoid Loop*. 2002. URL http://ffden-2.phys.uaf.edu/211_fall2002.web.dir/shawna_sastamoinen/Clothoid_Loop.htm.

[3] Paul M. D. *G Tolerance in 4 vectors*. 1964. URL https://coasterforce.com/physics/.

[4] Norfleet W. T Kumar K. V. *Issues on Human Acceleration Tolerance After Long-Duration Space Flights*. 1992. URL https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19930020462.pdf.

[5] Allen M, Weir-Jones I, Motiuk DR, Flewin KR, Goring RD, Kobetitch R, and Broadhurst A. *Acceleration perturbations of daily living. A comparison to 'whiplash'*. 1994. URL https://www.ncbi.nlm.nih.gov/pubmed/8073323?fbclid=IwAR1zZ-bkFdr0OTLURk7yLPUIqBnS6_xph4QMLgbDvnlXcZCdYN8mleQUtU4.

[6] Rainer Müller. *Roller coasters without differential equations — a Newtonian approach to constrained motion*. 2010. URL https://iopscience.iop.org/article/10.1088/0143-0807/31/4/013/pdf?fbclid=IwAR0jEx6defI6YmaUZdGwyM7B5X0PueFQ_NY3628VuSYCVo1pCDV5JWSEAsE.

[7] Ann-Marie Pendrill. *Rollercoaster loop shapes*. 2005. URL http://physics.gu.se/LISEBERG/eng/pe5601.pdf.

[8] Wikipedia. . URL https://en.wikipedia.org/wiki/Curvature.

[9] Wikipedia. . URL https://en.wikipedia.org/wiki/Euthanasia_Coaster.

[10] Wikipedia. . URL https://en.wikipedia.org/wiki/Reduced-gravity_aircraft.

10 Appendix

10.1 Half gaussian algorithm

Code in Python

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.special as sp
```

```
from scipy.stats import norm
import getopt
import sys
import os
import matplotlib.mlab as mlab

# # # datastreams
def remove(filePath):
    if os.path.exists(filePath):
        os.remove(filePath)

#clear datastreams
remove("halfgaussian_output.txt")

#define datastreams
output = open("halfgaussian_output.txt","w+")
sys.stdout = output

def K(Xscale, x, sigma, mu):
    Au = (x**2 - sigma**2)/sigma**4
    Gx = 7.8*Xscale*norm.pdf(x,mu,sigma)
    curva = Au*(Gx)/(1+(-x/sigma**2*Gx)**2)**1.5
    return curva

# # # Gaussian curve definition:
def halfGaussian(Xscale,mu,sigma, density):

    X = np.linspace(mu, mu + 3*sigma, density)
    Y = Xscale*7.8*norm.pdf(X,mu,sigma)
    curva = []
    for x in X:
        curva.append(K(Xscale, x, sigma, mu))

    return(X,Y, curva)
#
# # # Gaussian curve curvature definition:


# # # constants definition:

# for air resistance
rho = 1
A = 1.4
S = 1.2
B = rho*A*S

# for rolling friction
murf = 0.0018

# other constants
m = 550
g = 9.81

# # # generate X, Y, thetas

try:
        opts, args = getopt.getopt(sys.argv[1:], "", ["Gaussian",
        "halfGaussian", "clothoid", "parabolic"])
except getopt.GetoptError as err:
```

```
        print(help)
        sys.exit(2)


v0=np.float(args[0])
h0=np.float(args[1])
Xscale=np.float(args[2])
density=np.float(args[3])
mu=np.float(args[4])
sigma=np.float(args[5])


X, Y, curva = halfGaussian(Xscale, mu, sigma, density) # point density, radius

#adding initial height
Y = np.array(Y)
curva = np.array(curva)
Y = Y + h0

plt.plot(X,Y)
plt.plot(X,1/curva)
plt.show()


### steps:

def step(v0, x0, xk, y0, yk, E0, curva):
    # horizontal distance:
    delx=xk-x0
    dely=yk-y0
    # total distance:
    delr=np.sqrt((xk-x0)**2+(yk-y0)**2)
    #theta:
    th=np.arccos(delx/delr)
    ### add R(x0), f
    #radius of curvature R:
    Rad=1/curva
    #centripetal force C:
    C=m*v0**2/Rad
    #gravitational force component in the direction of C:
    Qc=m*g*delx/delr
    #normal force
    N = Qc + C
    #energy loss due to rolling friction:
    roll=-murf*N*delr
    #energy loss due to air resistance:
    drag=-delr*(B*v0**2)/2
    #energy available:
    E=E0+roll+drag
    #potential energy at this point:
    Ep=m*g*y0
    #kinetic energy at this point:
    Ek=E-Ep
    #final velocity:
    vk=np.sqrt(2*Ek/m)

    #write output
    print("Step no. ", no)
    print("Theta: ", th)
```

```
    print("delr:", delr)
    print("Radius of curvature:", Rad, "m")
    print("Centripetal force:",C,"N")
    print("Gravitational force — normal component:",Qc,"N")
    print("Normal force:", N, "N")
    print ("Energy loss due to air resistance: ", drag, "J")
    print ("Energy loss due to rolling friction: ", roll, "J")
    print ("Velocity: ", v0, "[m/s]")
    print ("Velocity: ", v0*3.6, "[km/h]")
    print ("Total energy available: ", E0, "J")
    print ("Height: ", y0, "m")
    print("Potential energy: ", Ep, "J")
    print("Kinetic energy: ", Ek, "J")
    print("\n")

    return(vk, E)
#


# define starting values:
h0=Y[0]
E0=m*v0**2/2+m*g*h0


for no in range(0, X.shape[0]-1):
    no2=no+1
    v0, E0 = step(v0, X[no], X[no2], Y[no], Y[no2], E0, curva[no])


output.close()
```

10.2 Clothoid loop algorithm

Code in Python

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.special as sp
from scipy.stats import norm
import getopt
import sys
import os


# # # datastreams
def remove(filePath):
    if os.path.exists(filePath):
        os.remove(filePath)

#clear datastreams
remove("clothoid_output.txt")

#define datastreams
output = open("clothoid_output.txt","w+")
sys.stdout = output


# # # clothoid definition:
def clothoid(density, Xscale):
```

```
    thetas=[]
    X = []
    Y = []

    R = Xscale # radius

    z=np.linspace(0.01, np.sqrt(2), density)
    for th in z:
        fres=sp.fresnel(th)
        x=fres[1]
        y=fres[0]
        thetas.append(th)
        X.append(R*x)
        Y.append(R*y)

    xhalf=x

    z=np.linspace(np.sqrt(2), 0.01, density)
    for th in z:
        fres=sp.fresnel(th)
        x=2*xhalf-fres[1]
        y=fres[0]
        thetas.append(th)
        X.append(R*x)
        Y.append(R*y)

    #

    #plt.plot(X,Y)
    #plt.show()
    return(X, Y, thetas)

### constants definition:

# for air resistance
rho = 1
A = 1.4
S = 1.2
B = rho*A*S

# for rolling friction
mu = 0.0018

# other constants
m = 550
g = 9.81

### generate X, Y, thetas

try:
        opts, args = getopt.getopt(sys.argv[1:], "", ["Gaussian",
        "halfGaussian", "clothoid", "parabolic"])
except getopt.GetoptError as err:
        print(help)
        sys.exit(2)


v0=np.float(args[0])
h0=np.float(args[1])
```

```
Xscale=np.float(args[2])
density=np.float(args[3])


X, Y, thetas = clothoid(density, Xscale) # point density, radius

#remove double indices due to symmetry
thetas.remove(thetas[int(density-1)])
X.remove(X[int(density-1)])
Y.remove(Y[int(density-1)])

#adding initial height
thetas = np.array(thetas)
Y = np.array(Y)
Y = Y + h0

plt.plot(X,Y)
plt.show()


### steps:

def step(th, v0, x0, xk, y0, yk, E0):
    # horizontal distance:
    delx=xk-x0
    dely=yk-y0
    # total distance:
    delr=np.sqrt((xk-x0)**2+(yk-y0)**2)
    ### add R(x0), f
    #radius of curvature R:
    Rad=Xscale/(2*th)
    #centripetal force C:
    C=m*v0**2*2*th/Xscale
    #gravitational force component in the direction of C:
    Qc=m*g*delx/delr
    #normal force
    N = Qc + C
    #energy loss due to rolling friction:
    roll=-mu*N*delr
    #energy loss due to air resistance:
    drag=-delr*(B*v0**2)/2
    #energy available:
    E=E0+roll+drag
    #potential energy at this point:
    Ep=m*g*y0
    #kinetic energy at this point:
    Ek=E-Ep
    #final velocity:
    vk=np.sqrt(2*Ek/m)

    #write output
    print("Step no. ", no)
    print("Radius of curvature:", Rad, "m")
    print("Centripetal force:",C,"N")
    print("Gravitational force -- normal component:",Qc,"N")
    print("Normal force:", N, "N")
    print ("Energy loss due to air resistance: ", drag, "J")
    print ("Energy loss due to rolling friction: ", roll, "J")
    print ("Velocity: ", v0, "[m/s]")
```

```
    print ("Velocity: ", v0*3.6, "[km/h]")
    print ("Total energy available: ", E0, "J")
    print ("Height: ", y0, "m")
    print("Potential energy: ", Ep, "J")
    print("Kinetic energy: ", Ek, "J")
    print("\n")

    return(vk, E)
#


# define starting values:
h0=Y[0]
E0=m*v0**2/2+m*g*h0


for no in range(0, thetas.shape[0]-1):
    no2=no+1
    v0, E0 = step(thetas[no], v0, X[no], X[no2], Y[no], Y[no2], E0)


output.close()
```

10.3 Rotating move algorithm

Code in C++, main function:

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <cmath>
#include <iomanip>
#include <fstream>
#include "functions.h"

int main()
{
  double ksi; //angle of rotating [rad]
  double R; //radius [m]
  double E_0; //Energy at the begining [J]
  double v_0; //velocity at the begining [m/s]
  std::cout<<"v_0: "; std::cin>>v_0; std::cout<<std::endl;
  std::cout<<"E_0: "; std::cin>>E_0; std::cout<<std::endl;
  std::cout<<"ksi: "; std::cin>>ksi; std::cout<<std::endl;
  std::cout<<"R: "; std::cin>>R; std::cout<<std::endl;

  double h = (E_0/550 - v_0*v_0/2)/9.81;

  show(ksi, R, E_0, h); //function to save the datas to text file

  return 0;
}
```

functions.cpp

```
#include <iostream>
#include <vector>
#include <cmath>
```

```
#include <cstdlib>
#include <ctime>
#include <fstream>
#include <iomanip>
#include <algorithm>

double dW( //change of energy
  const double ksi,
  const double R,
  const double h,
  const double v_0
)
{
  return (550*9.81*cos(atan(9.81*R/(v_0*v_0)))+550*(v_0*v_0)/R *
  sin(atan(9.81*R/(v_0*v_0))))*R*ksi*0.0018/1000.0
  + 0.5*1.4*2.0*v_0*v_0*ksi*R/1000.0;
}
double G( //function of overload
  const double ksi,
  const double R,
  const double h,
  const double v_0
)
{
  return v_0*v_0/(R*9.81);
}
double velocity(
  const double h,
  const double E_0
)
{
  if(E_0/550 - 9.81*h >= 0)
  return sqrt(2*(E_0/550 - 9.81*h));
  else
  {
    std::cout<<"v does not exist with h: "<<h<<", E: "<<E_0<<std::endl;
  }
}

void show( //the exact function of energy and velocity change
  const double ksi,
  const double R,
  const double E_0,
  const double h
)
{
  double v_0;
  double E=E_0;
  std::fstream test_file;

  test_file.open("test_gnuplot.txt", std::ios::out);
  for(int i=1; i<1000; i++)
  {
    v_0 = velocity(h, E);
    E= E-dW(ksi, R, h, v_0);
    test_file<<v_0<<" "<<E<<" "<<G(ksi, R, h, v_0)<<std::endl;
  }

  test_file.close();
```

```
}
```

functions.h

```
double velocity(
  const double h,
  const double E_0
);

double dW(
  const double ksi,
  const double R,
  const double h,
  const double v_0
);

double G(
  const double ksi,
  const double R,
  const double h,
  const double v_0
);

void show(
  const double ksi,
  const double R,
  const double E_0,
  const double h
);
```