Lebanese American University

School of Engineering

Department of Industrial and Mechanical Engineering

Augmented Reality

Final year Project

Spring 2020

Youssef Jammal
Charles Hanna
Mazen Koteich
Abbas Fayyad

Project Advisor: Dr. Elias Brassitos.

Table of Contents

Table of Figures

Abstract:

The world is constantly evolving, may it be through invention of new solutions to upcoming problems or by improving old solutions of contemporary issues.

This is mainly achieved through experimenting, scientists do so in order to ascertain the results of their studies, but these experiments are not always safe. According to a study done by Richard Van Noorden (lab and workplace researcher) 86% of experimenters claim to have been injured during their careers.

This is where augmented reality steps in. augmented reality provides us with a new type of testing which leaves the experimenter far away from danger that might occur throughout the experiment. In addition to that, it provides experimenters very accurate results.

Augmented reality has uses out of the experimentation domain as well; it can be introduced in the world of gaming as well since it surpasses virtual reality therefore giving the users a closer “realer” experience and therefore a more enjoyable and enthusiastic one.

Introduction:

The main concept of the project is to demonstrate augmented reality through designing a prototype of driving a car.

The user will be driving a remote controlled car while sitting on a chair that is controlled by actuators. The goal of the project is for the user to have the experience of being inside the RC car while sitting on the chair.

For example, if the RC car drives through a bumpy road, tilts or goes down/up a steep path the user will be experiencing the feeling of being inside the RC car as it goes through these paths. We might even make it more realistic by installing a GoPro camera on the RC car and have the user see what he would see if he was inside of the car.

Problem Statement and Formulation:

As mentioned before, safety is a main constraint in experiments. In addition to that, some experiments cost a lot of money due to expensive equipment that constitute the prototype being tested, and some have to be repeated multiple times in order to achieve desired results which adds up to the cost of the experiment. Experiments can also be time-depleting since it would take experimenters time to rebuild a new prototype after every failed experiment and reset the experiment conditions.

Augmented reality is a solution to all the problems mentioned before.

It provides maximum safety for the user, it is very economical in terms of cost since the prototype is only built once and can be used in multiple experiments which also solves the time issues. In addition to that, on a social level, augmented reality can introduce gamers to a whole new experience in playing games since it makes it feel even more realistic than virtual reality which is considered the most advanced gaming technology nowadays.

Summary of Literature Review and Background Research:

In order to understand the concept of our project, background research concerning augmented reality is in order. Some of the main augmented reality uses will be discussed in the following pages:

AR Driving simulation:

Figure 1: AR Driving simulation

Augmented reality is used in teaching new drivers without taking the risk of getting into the car and putting oneself in danger. It help drivers experience different driving situations they might face in real life. Driving simulations are also being used in military vehicle trainings where it is more useful since the military is a very demanding job when it comes to accuracy and speed, not to mention the expensive and hard to control vehicles in the military such as tanks, Humvees… (Virtual Reality Society)

It is also being used in controlling real vehicles rather than only controlling a virtual part, the user could be driving a tank in a war zone while being away from danger.

AR driving is also getting popular in racing games, users can now race each other in real life; controlling real cars while sitting on a chair and getting the feeling of being inside the car.

AR Flight and Ship simulations:

Figure 2:Flight Simulators

Figure 3: Bridge Simulators

Augmented reality is also being used in the aviation and marine fields being an efficient and affordable solution for training pilots, pilots now do not have to get into the struggle of flying the actual planes or drive the ships which will take much time to prepare and also a lot of time to train several pilots. Due to these AR simulators, more pilots can be trained in a shorter time. In addition to that, AR simulations can have their own custom made tracks and obstacles which can be much more difficult than what the pilots would face in real life therefore equipping them with the necessary experience to deal with real life obstacles as they travel the world.

AR in the Medical field:

Figure 4: Surgery using AR

Since mistakes are not allowed in the medical field, AR helps train surgeons by providing a virtual platform to practice. Surgeons can now practice and improve their skills, as their jobs have no room for errors. AR helps surgeons perform surgeries much more accurately through haptics. They way haptics help them is by reducing the movement of the surgeon's hand and making it steadier. The surgeon will be moving his arm on a platform while a robotic arm is moving over the patient's body, for every 10 cm moved by the surgeon's arm the robotic arm moves 1cm therefore increasing the steadiness and accuracy of the surgeon which will increase surgeries' success rates.

Concept development:

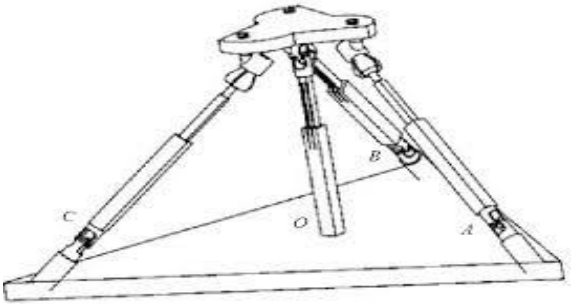


Figure 5. 3DOF parallel manipulator

The 3dof parallel manipulator would be the backbone of our project. It is a mechanical system that uses multiple actuators to control a single platform, which in our case is the chair. Parallel manipulators are commonly used with six actuators but we will be using a three legged parallel manipulator due to budget constraints. The top platform represents the chair, the three degrees of freedom

are pitching, rolling and vertical elevation, there is also yawing but it will not but taken into consideration.

The sole purpose of the actuators is to get the platform in the desired position, at the desired acceleration, and with as little time lag as possible in reference to the RC car. The prototype can be built in many different ways. The main difference between all of them is the choice of actuators that are going to control the chair, so we looked into different types of suiting actuators which would help us decide on which is the most convenient for our application.

Candidate actuators:

Hydraulic pistons:

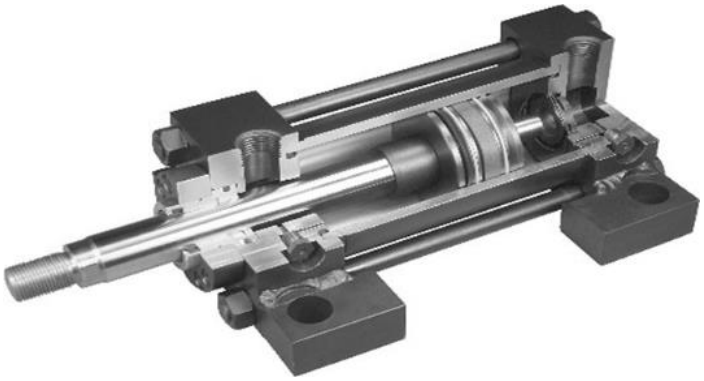


Figure 6: Hydraulic piston

The hydraulic pistons use the pressure of oil inside and convert it to mechanical work through the piston. A pump pressurizes the oil inside the piston therefore pushing the piston linearly therefore obtaining the mechanical work output.

Hydraulic pistons are used when high speeds and forces are needed, they have a very quick response time and have the peak power compared to other candidate actuators.

But on the other hand, the pistons are very expensive and complex when it comes to implementation since we have to account for the piping system and the pump that would pressurize oil in the pistons.

Pneumatic pistons:

Figure 7: Pneumatic piston

Pneumatic pistons use the kinetic and potential energy of air and transform it into linear mechanical work. These pistons are cheaper than other actuators, they have good response time, but they are very hard to control, the pneumatic pistons are most useful in on and off application which is not desirable in our project. In order to be able to control the pistons, we would have to introduce control valves and then have these control valves controlled by electric motors which is a hassle that might not pay off eventually.

Electric pistons:

Figure 8: Electric piston

Electric pistons convert the rotational force of the motor into linear movement through the screw rod. These pistons have very high efficiencies and are very environmentally friendly. They are very complex in terms of control due to multiple configurations and the need to use multiple control systems but on the other hand these pistons have an extremely high accuracy and the ability of synchronization which makes it able of synchronizing with the other two legs in case it was chosen. Such actuators have high cost due to their huge motor.

Pulley systems:

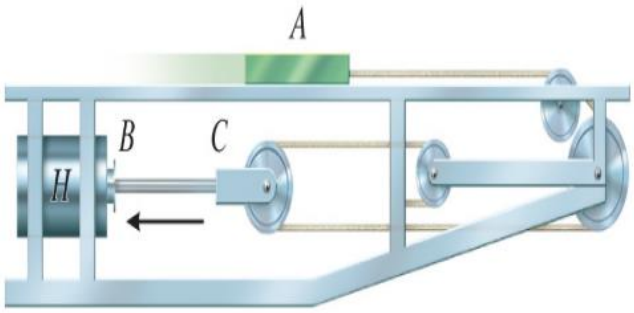
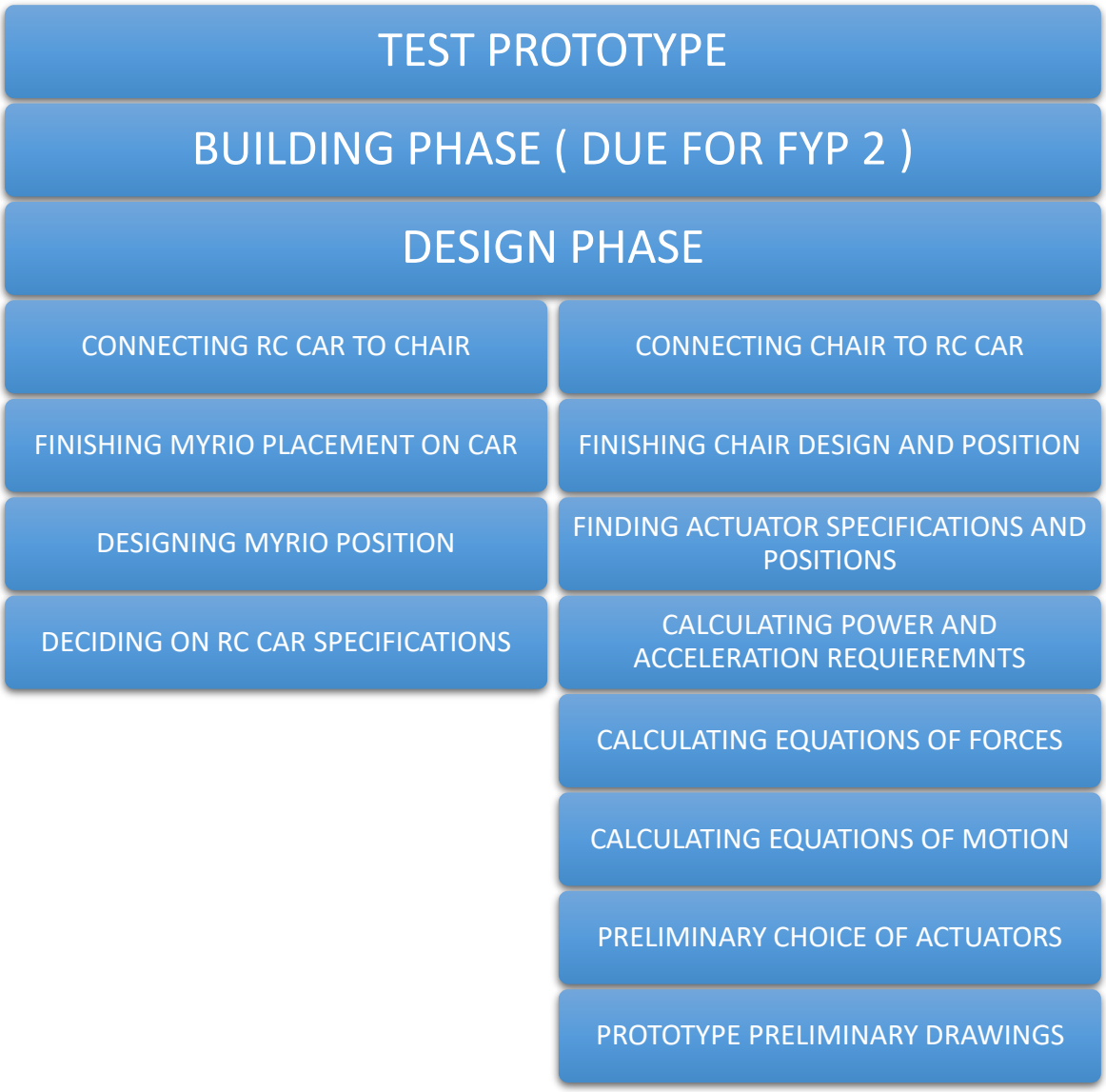


Figure 9: Pulley System

Pulley systems are by far the most efficient when it comes to cost versus heavy lifting, these systems are very cheap but are capable of exerting extreme large forces especially when they are combined. These systems are not very accurate and are hard to control because they work on friction which means there is slipping and energy dissipated as heat. In addition to the extremely slow response time due to expansion of the cables and fatigue. There if we were to adopt the pulley systems we would have to make a trade-off between heavy lifting capability and speed of response, heavy lifting requires combining pulleys which in turn slows the system's response time.

Methodology:

Preliminary WBS



The preliminary WBS is represented in a hierarchical manner.

In order to get a clearer idea of the project, we start by drawing a preliminary figure of the prototype, we then, having done the literature review, move to deciding on actuators. Having done so, we can now find the equations of motion of the system, in other words, how would the position of the actuators affect the position of the platform, and we will find that using inverse kinematics, which means we set the position of the platform first and deduce the position of the actuators accordingly. After finding the equations of motion we would be able to

find the forces exerted by each actuator and therefore get the power requirements of each actuator by implementing the velocity that they should attain. We then work simultaneously on deciding RC car specifications; whether it should be 3D printed or manufactured or maybe a standard toy RC car that could be purchased from any store and at the same time we would be defining the actuator specifications now that we know their positions at all times and the power and speed that they should exert and be looking to buy suitable ones.

Keeping in mind that if we ever believe that the chosen type of actuators is not practical for some reason we will have to redo all steps from “preliminary choice of actuators” to “calculating power and acceleration requirements”.

We finally move to finding the perfect placement of the myrio on the car as well as on the chair system and would be able to connect the two myrios together.

Having finished these, we would basically be done with the design phase of the prototype and can move to the building phase.

Preliminary drawings:

Below is a preliminary drawing of the prototype on Solidworks.

Figure 10: Preliminary design

Deciding on actuators:

Having done the preliminary drawings, we then decided on the actuators.

Possible actuators were the DC motors and hydraulic pistons.

DC motors are more expensive than hydraulic pistons since we will be needing Motors of big sizes but they do not require a lot of implementation steps.

But hydraulic pistons, although they require a lot of implementation steps but they are more effective in terms of power delivery and are relatively cheaper than DC motors.

So we have decided on DC motors as actuators for our project.

Differing form the preliminary drawings, we will only be designing our project with three legs instead of four, and that is for economic purposes.

Mechanical Design:

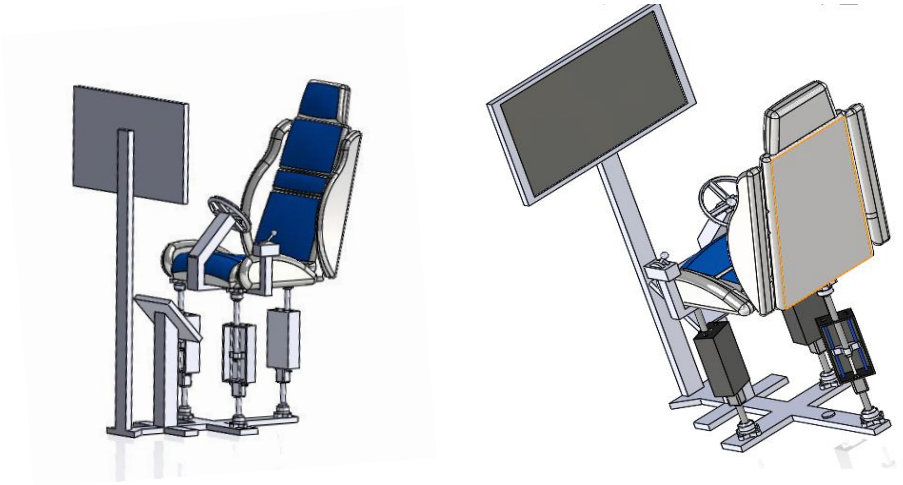


Figure 11: SolidWorks mechanical design

This is the final design of the project; parts were made to scale in order to get them 3D printed. The prototype has 3 actuators as decided, the steering wheel and pedals are inserted to simulate the remote driving. TV screen for the user to watch the car and get the experience of being in it.

Choice of actuators:

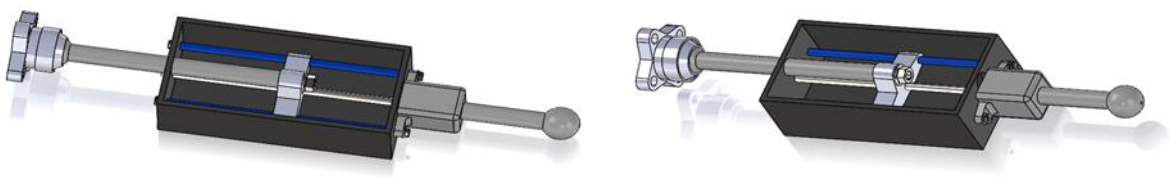


Figure 12: Actuators

The actuators that were used are screw motor actuators, they are very similar to electric pistons. The grey rod is a screw rod, it is connected to the motor from one end and inserted into the nut. The nut is connected to two rods (blue rods) in order to prevent it from rotating with the screw rod therefore limiting its movement to back and forth according to the motor's rotation. Two universal ball joints are connected to both ends of the actuators in order to have the required flexibility by the system.

Variables used in the MATLAB code:

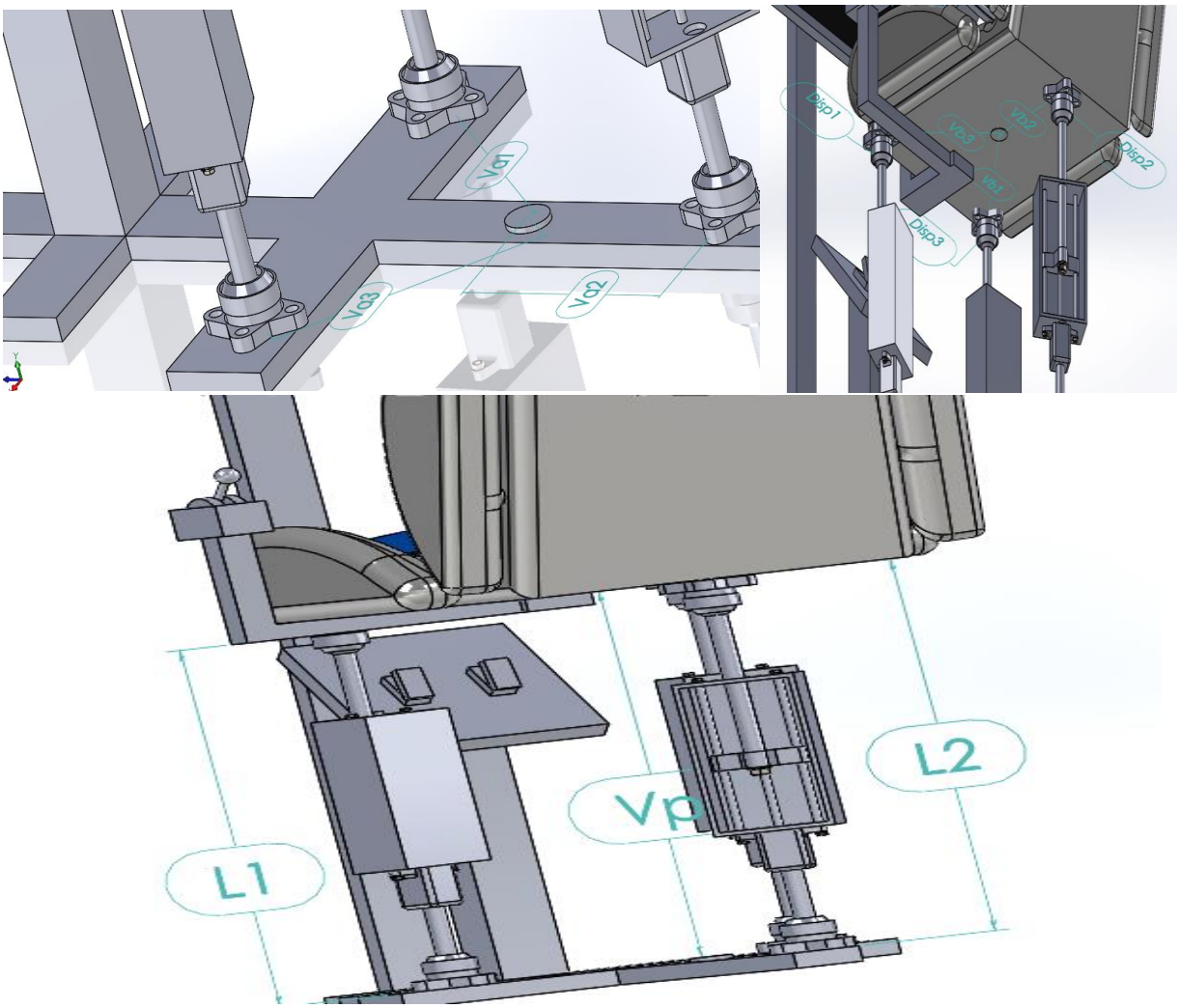


Figure 13: Variables used in the MATLAB code shown in SolidWorks

- Va: coordinates of the actuator's base with respect to ground origin.
- Vb: coordinates of the actuator's tip with respect to the moving frame origin.

- Disp: coordinates of actuator tip with respect to ground origin.
- L: length of actuators
- Vp : coordinates of frame origin with respect to ground origin.

MATLAB code:

The MATLAB code is divided into 2 parts, the optimization of the leg angle, and the second part the detailed analysis of the optimized system, the first part consists of 3 nested while loop, the outer loop is the incrementing of r2, the middle loop is the increment of r1, and the inner loop is the cycle simulation and finally outputs the graph representing the motor power requirements with respect to r1 and r2. The second part is the detailed analysis and outputs 21 graphs which will later be shown, each leg has been analyzed and its, displacement, velocity, acceleration, force, power, rpm, and engine torque requirements graphed.

Legend:

acceleration1matrix: the actual acceleration matrix of leg 1 with 200 values

acceleration1matrixt: the acceleration matrix of leg 1 after differentiating the velocity matrix, leaving the matrix with 199 and not 200 values

acceleration1spline: the spline structure which converts acceleration1matrixt to acceleration1matrix

acceleration2matrix: the actual acceleration matrix of leg 2 with 200 values

acceleration2matrixt: the acceleration matrix of leg 2 after differentiating the velocity matrix, leaving the matrix with 199 and not 200 values

acceleration2spline: the spline structure which converts acceleration1matrixt to acceleration1matrix

acceleration3matrix: the actual acceleration matrix of leg 3 with 200 values

acceleration3matrixt: the acceleration matrix of leg 3 after differentiating the velocity matrix, leaving the matrix with 199 and not 200 values

acceleration3spline: the spline structure which converts acceleration1matrixt to acceleration1matrix

add_on: an dummy variable used to manipulate the MAX_Matrix

base: length of the platform's triangular shape base

coefficients: matrix containing the force coefficients

cos_alpha_link_1: angle between the x axis and the leg 1

cos_alpha_link_2: angle between the x axis and the leg 2

cos_alpha_link_3: angle between the x axis and the leg 3

cos_beta_link_1: angle between the y axis and the leg 1

cos_beta_link_2: angle between the y axis and the leg 2

cos_beta_link_3: angle between the y axis and the leg 3

cos_delta_link_1: angle between the z axis and the leg 1

cos_delta_link_2: angle between the z axis and the leg 2

cos_delta_link_3: angle between the z axis and the leg 3

density: density of the material being used for the platform

displacement1matrix: actual displacement of the leg 1 with initial and final derivatives equal to 0 after cubic splines

displacement2matrix: actual displacement of the leg 2 with initial and final derivatives equal to 0 after cubic splines

displacement3matrix: actual displacement of the leg 3 with initial and final derivatives equal to 0 after cubic splines

Displacement_of_link_1: coordinate of the leg 1's tip

Displacement_of_link_2: coordinate of the leg 2's tip

Displacement_of_link_3: coordinate of the leg 3's tip

elongation1: maximum elongation of leg 1

elongation2: maximum elongation of leg 2

elongation3: maximum elongation of leg 3

enginetorque1: motor 1 torque matrix

enginetorque2: motor 2 torque matrix

enginetorque3: motor 3 torque matrix

force1: force matrix of leg 1 before cubic splines

force1matrix: force matrix of leg 1 after cubic splines with initial and final derivative equal to 0

force2: force matrix of leg 2before cubic splines

force2matrix: force matrix of leg 2 after cubic splines with initial and final derivative equal to 0

force3: force matrix of leg 3 before cubic splines

force3matrix: force matrix of leg 3 after cubic splines with initial and final derivative equal to 0

force_resultants: matrix containing the force resultans

forces: matrix containing force 1, force 2 and force 3

forcesforspline1: force matrix of leg 1 for the cubic splines

forcesforspline2: force matrix of leg 2 for the cubic splines

forcesforspline3: force matrix of leg 3 for the cubic splines

forcespline1: input of initial and final derivatives to be 0 for leg 1

forcespline2: input of initial and final derivatives to be 0 for leg 2

forcespline3: input of initial and final derivatives to be 0 for leg 3

height: length of the platform's triangular shape height

iteration: number of iterations that have been done in the simulation loop

iteration_calculation: finding the number of iterations required to do 1 full cycle for the slowest components between pitch, roll and elevation

iterationrounding: iteration_calculation rounded

L1: length of leg 1 at time t

L2: length of leg 2 at time t

L3: length of leg 3 at time t

Length1: length of leg 1 matrix before cubic splines

Length2: length of leg 2 matrix before cubic splines

Length3: length of leg 3 matrix before cubic splines

Mall: combined rotational matrix

mass: mass of the platform

MATRIXPOWER: dummy matrix for monitoring r1t2, r2t2 and the powermatrix

MAX_Matrix: matrix of containing the “t” of every iteration in the simulation loop

MAX_P1: sum of the power used for the leg 1 over the course of 1 simulation with a specific r1 and r2

MAX_P2: sum of the power used for the leg 2 over the course of 1 simulation with a specific r1 and r2

MAX_P3: sum of the power used for the leg 3 over the course of 1 simulation with a specific r1 and r2

maxlength1: maximum base to tip length of leg 1

maxlength2: maximum base to tip length of leg 2

maxlength3: maximum base to tip length of leg 3

MaxP1t: matrix containing the sums of the power used for the leg 1 over the course of 1 simulation, r1 changes and r2 in constant when r2 changes MaxP1t gets emptied

MaxP1t2: matrix containing the sums of the power used for the leg 1 over the course of 1 simulation, along the entire optimization process

MaxP2t: matrix containing the sums of the power used for the leg 2 over the course of 1 simulation, r1 changes and r2 in constant when r2 changes MaxP2t gets emptied

MaxP2t2: matrix containing the sums of the power used for the leg 1 over the course of 1 simulation, along the entire optimization process

MaxP3t: matrix containing the sums of the power used for the leg 3 over the course of 1 simulation, r1 changes and r2 in constant when r2 changes MaxP3t gets emptied

MaxP3t2: matrix containing the sums of the power used for the leg 1 over the course of 1 simulation, along the entire optimization process

mazen: the matrix to be sent for the CAD simulation

minPower: variable used to find the optimal values

minPowerindex: variable indexing the optimal r1 and r2 in the r1t2 and r2t2 matrices

norm_Va: coefficient to multiply the Va coordinates

norm_Vb: coefficient to multiply the Vb coordinates

Number_of_iterations: number of iterations to be done until a simulation is terminated

numberofcycles: how many cycles are desired throughout 1 simulation

omega_phi: frequency of the roll

omega_theta: frequency of the pitch

omega_Zcenter: frequency of the elevation

optimumr1value: optimal r1 value

optimumr2value: optimal r2 value

phi: roll angle at time t

phi_dot: roll angular velocity at time t

phi_double_dot: roll angular acceleration at time t

phi_sign: direction of roll

plate_value: in case a prototype is being done with small sized, the plate value can be set to any number to calculate the required motor power for a normal sized plate

platform_pitch_matrix: rotation matrix for the pitch

platform_roll_matrix: rotation matrix for the roll

power1matrix: matrix containing the power used by leg 1 after cubic splines

power2matrix: matrix containing the power used by leg 2 after cubic splines

power3matrix: matrix containing the power used by leg 3 after cubic splines

Powermatrix: sum of Max_P1t2, Max_P2t2, and Max_P3t2,

r1: Vax/Vbx

r1t: contains values of r1 until r2 changes

r1t2: contains r1 values all along the optimization process

r2: Vay/Vby

r2t: contains values of r2 until r2 changes (dummy variable)

r2t2: contains r2 values all along the optimization process

ratio: similar to the plate value, but this is a division instead of a multiplication, if ratio and plate value are equal nothing would change

rotationalvelocity1: RPM leg 1

rotationalvelocity2: RPM leg 2

rotationalvelocity3: RPM leg 3

spline1: spline structure for the displacement of leg 1

spline2: spline structure for the displacement of leg 2

spline3: spline structure for the displacement of leg 3

splines: number of splines to be done

t: time “t”

theta: pitch angle at time t

theta_dot: pitch angular velocity at time t

theta_double_dot: pitch angular acceleration at time t

theta_sign: direction of pitch

thickness: thickness of the plate

Va1: explained in the previous report section

Va2: explained in the previous report section

Va3: explained in the previous report section

Vb1: explained in the previous report section

Vb2: explained in the previous report section

Vb3: explained in the previous report section

velocity1matrix: the actual velocity matrix of leg 1 with 200 values

velocity1matrixt: the velocity matrix of leg 1 after differentiating the displacment matrix, leaving the matrix with 199 and not 200 values

velocity1spline: the spline structure which converts velocity1matrixt to velocity1matrix

velocity2matrix: the actual velocity matrix of leg 2 with 200 values

velocity2matrixt: the velocity matrix of leg 2 after differentiating the velocity matrix, leaving the matrix with 199 and not 200 values

velocity2spline: the spline structure which converts velocity1matrixt to velocity1matrix

velocity3matrix: the actual velocity matrix of leg 3 with 200 values

velocity3matrixt: the velocity matrix of leg 3 after differentiating the velocity matrix, leaving the matrix with 199 and not 200 values

velocity3spline: the spline structure which converts velocity1matrixt to velocity1matrix

Vp: coordinates of the center point of the plate, it is not the centroid, it is located in the middle of the height and base

x: time increment

xq: number of values to be interpolated using splines

xq1: dummy variable used on the velocity and acceleration matrices to give them back their number of variables after differentiating which removed 1 value

Zcenter: elevation angle at time t

Zcenter_dot: elevation velocity at time t

Zcenter_double_dot: elevation acceleration at time t

Zcenter_sign: direction of elevation

Zcentervalue: amplitude of elevation

```
clc
clear
clf

k2=1;
r1t2=[];
r2t2=[];
MaxP1t2=[];
MaxP2t2=[];
MaxP3t2=[];


% number of cycles to simulate is the
 numberofcycles=2;

%iterations
plate_value=1;
ratio=1;
norm_Vb=plate_value/ratio;
norm_Va=norm_Vb;

% center of frame to leg in frame 1
Vb1= -norm_Vb.*[-0.165; 0.17775; 0];
Vb2= -norm_Vb.*[0.165; 0.17775; 0];
Vb3= -norm_Vb.*[0; -0.17775; 0];

% parameters and properties of the frame
height= ratio*2*(abs(Vb3(2)) + abs(Vb1(2)));
base= ratio*(abs(Vb1(1)) + abs(Vb2(2)));

splines=10000;

% input the thickness and density
thickness=0.05;
density= 2700;

%total mass of the frame plus the person on it
mass =base*height*thickness*2700/2;

%initialisation of variables
x=0.01;

% these are the variables and their initial conditions

Zcentervalue=0.1;

Zcenter_sign=1;
omega_Zcenter=4.8;

theta_sign=1;
omega_theta=5.2;

phi_sign=1;
omega_phi=5;
```

```
iteration_calculation=min([omega_Zcenter, omega_theta, omega_phi]);

Number_of_iterations=(2*pi()/iteration_calculation)/x;

iterationrounding=round(Number_of_iterations);

if iterationrounding<Number_of_iterations
   Number_of_iterations=iterationrounding+1;
else
   Number_of_iterations=iterationrounding;
end

 Number_of_iterations= Number_of_iterations*numberofcycles;

while r2<=2
r1=1;
r1t=[];
r2t=[];
MaxP1t=[];
MaxP2t=[];
MaxP3t=[];
while r1<=2
   MAX_P1=0;
   MAX_P2=0;
   MAX_P3=0;

% center of frame to leg in frame 0
Va1= -norm_Va.*[r1*0.165; r2*-0.17775; 0];
Va2= -norm_Va.*[r1*-0.165; r2*-0.17775; 0];
Va3= -norm_Va.*[0; r2*0.17775; 0];

%matrix initialization
force1=[];
force2=[];
force3=[];
add_on=[];
MAX_Matrix=[];
Length1=[];
Length2=[];
Length3=[];

%time initialization
t=0;
iteration=0;
while iteration<Number_of_iterations
% height, roll and pitch of the frame

Zcenter= Zcenter_sign*Zcentervalue * cos (t * omega_Zcenter);
Zcenter_dot= Zcenter_sign*Zcentervalue*omega_Zcenter*(-1)*sin(t*omega_Zcenter);
Zcenter_double_dot= Zcenter_sign*Zcentervalue*(omega_Zcenter^2)*(-1)*cos(t*omega_Zcenter);

theta= theta_sign*(pi/6)*cos (t * omega_theta);
theta_dot= theta_sign*(pi/6)*omega_theta*(-1)*sin(t*omega_theta);
theta_double_dot=  theta_sign*(pi/6)*(omega_theta^2)*(-1)*cos(t*omega_theta);

phi= phi_sign*(pi/4)*sin (t * omega_phi);
```

Figure 14: Pages 1 and 2 of the MATLAB code

```
phi_dot= phi_sign*(pi/4)*omega_phi*(-1)*sin(t*omega_phi);
phi_double_dot= phi_sign*(pi/4)*(omega_phi^2)*(-1)*cos(t*omega_phi);

% center of the frame coordinate
Vp=[0; 0; Zcenter+0.5];

% rotational vector
platform_pitch_matrix= [ 1, 0, 0; 0, cos(theta), sin(theta); 0, -sin(theta), cos(theta) ];
platform_roll_matrix= [ cos(phi), 0, -sin(phi); 0, 1, 0; sin(phi), 0, cos(phi) ];
Mall= platform_pitch_matrix*platform_roll_matrix;

% leg at frame 0 to leg at frame 1
Displacement_of_link_1= Vp - Mall * Vh1;
Displacement_of_link_2= Vp - Mall * Vh2;
Displacement_of_link_3= Vp - platform_pitch_matrix * Vh3;

% angles of the links
cos_alpha_link_1= -(Displacement_of_link_1(1)-Va1(1))/(sum((Displacement_of_link_1-Va1).^2));
cos_beta_link_1= -(Displacement_of_link_1(2)-Va1(2))/(sum((Displacement_of_link_1-Va1).^2));
cos_delta_link_1= -(Displacement_of_link_1(3)-Va1(3))/(sum((Displacement_of_link_1-Va1).^2));

cos_alpha_link_2= (Displacement_of_link_2(1)-Va2(1))/(sum((Displacement_of_link_2-Va2).^2));
cos_beta_link_2= (Displacement_of_link_2(2)-Va2(2))/(sum((Displacement_of_link_2-Va2).^2));
cos_delta_link_2= (Displacement_of_link_2(3)-Va2(3))/(sum((Displacement_of_link_2-Va2).^2));

cos_alpha_link_3= (Displacement_of_link_3(1)-Va3(1))/(sum((Displacement_of_link_3-Va3).^2));
cos_beta_link_3= (Displacement_of_link_3(2)-Va3(2))/(sum((Displacement_of_link_3-Va3).^2));
cos_delta_link_3= (Displacement_of_link_3(3)-Va3(3))/(sum((Displacement_of_link_3-Va3).^2));

%resultant of forces
force_resultants= [(density*thickness*height*(base^3)/48)*phi_double_dot ;
            (density*thickness*(height^3)*base/9)*theta_double_dot;
             mass*(Zcenter_double_dot + 9.81)                    ];

%coefficients
coefficients=
[cos_delta_link_1*(Displacement_of_link_1(1))+cos_alpha_link_1*(Displacement_of_link_1(3)-
Zcenter),cos_delta_link_2*(Displacement_of_link_2(1))+cos_alpha_link_2*(Displacement_of_link_2(3)-
Zcenter),0;
          cos_delta_link_1*(Displacement_of_link_1(2))-cos_beta_link_1*(Displacement_of_link_1(3)-
Zcenter),cos_delta_link_2*(Displacement_of_link_2(2))-cos_beta_link_2*(Displacement_of_link_2(3)-
Zcenter),cos_delta_link_3*(Displacement_of_link_3(2))+cos_beta_link_3*(Displacement_of_link_3(3)-
Zcenter);
          cos_delta_link_1,cos_delta_link_2,cos_delta_link_3];

%force exerted by leg
forces= inv(coefficients)*force_resultants;

%force matrices
force1=[force1, forces(1)];
force2=[force2, forces(2)];
force3=[force3, forces(3)];

% leg length
L1=sqrt(sum((Displacement_of_link_1-Va1).^2));
L2=sqrt(sum((Displacement_of_link_2-Va2).^2));
```

```
L3=sqrt(sum((Displacement_of_link_3-Va3).^2));

Length1=[Length1,L1];
Length2=[Length2,L2];
Length3=[Length3,L3];

MAX_Matrix=[MAX_Matrix,t];
t=t+x;
add_on=[add_on,1];
iteration=iteration+1;
end

Length1=[0.5, Length1, 0.5];
Length2=[0.5, Length2, 0.5];
Length3=[0.5, Length3, 0.5];

MAX_Matrix=MAX_Matrix+1.*add_on;
MAX_Matrix=[0, MAX_Matrix, max(MAX_Matrix)+1];

%Displacement

spline1=spline(MAX_Matrix, [0 Length1 0]);
spline2=spline(MAX_Matrix, [0 Length2 0]);
spline3=spline(MAX_Matrix, [0 Length3 0]);
xq=linspace(min(MAX_Matrix), max(MAX_Matrix), splines);

displacement1matrix= ppval(spline1, xq);
displacement2matrix= ppval(spline2, xq);
displacement3matrix= ppval(spline3, xq);

%velocity

velocity1matrixt=diff(displacement1matrix);
velocity2matrixt=diff(displacement2matrix);
velocity3matrixt=diff(displacement3matrix);

xq1=linspace(min(MAX_Matrix), max(MAX_Matrix), splines-1);

velocity1spline= spline(xq1, [0 velocity1matrixt 0]);
velocity2spline= spline(xq1, [0 velocity2matrixt 0]);
velocity3spline= spline(xq1, [0 velocity3matrixt 0]);

velocity1matrix= ppval(velocity1spline, xq);
velocity2matrix= ppval(velocity2spline, xq);
velocity3matrix= ppval(velocity3spline, xq);

%acceleration

acceleration1matrixt= diff(velocity1matrix);
acceleration2matrixt= diff(velocity2matrix);
acceleration3matrixt= diff(velocity3matrix);

acceleration1spline= spline(xq1, [0 acceleration1matrixt 0]);
acceleration2spline= spline(xq1, [0 acceleration2matrixt 0]);
acceleration3spline= spline(xq1, [0 acceleration3matrixt 0]);
```

Figure 15: Pages 3 and 4 of the MATLAB code

```
acceleration1matrix= ppval(acceleration1spline, xx);
acceleration2matrix= ppval(acceleration2spline, xx);
acceleration3matrix= ppval(acceleration3spline, xx);

%forces

forcesforspline1=[0, force1,0];
forcesforspline2=[0, force2,0];
forcesforspline3=[0, force3,0];

forcespline1=spline(MAX_Matrix, [0, forcesforspline1, 0]);
forcespline2=spline(MAX_Matrix, [0, forcesforspline2, 0]);
forcespline3=spline(MAX_Matrix, [0, forcesforspline3, 0]);

force1matrix= ppval(forcespline1, xx);
force2matrix= ppval(forcespline2, xx);
force3matrix= ppval(forcespline3, xx);

%power

power1matrix= abs(force1matrix.*velocity1matrix);
power2matrix= abs(force2matrix.*velocity2matrix);
power3matrix= abs(force3matrix.*velocity3matrix);

MAX_P1=sum(power1matrix);
MAX_P2=sum(power2matrix);
MAX_P3=sum(power3matrix);

r1t=[r1t;r1];

r2t=[r2t;r2];

MaxP1t=[MaxP1t; MAX_P1];
MaxP2t=[MaxP2t; MAX_P2];
MaxP3t=[MaxP3t; MAX_P3];

r1=r1+0.03;
end
r1t2=[r1t2;r1t];

r2t2=[r2t2;r2t];

MaxP1t2=[MaxP1t2; MaxP1t];
MaxP2t2=[MaxP2t2; MaxP2t];
MaxP3t2=[MaxP3t2; MaxP3t];

r2=r2+0.03;
end
Powermatrix= MaxP1t2+MaxP2t2+MaxP3t2;
figure(1)
plot3(r1t2,r2t2,Powermatrix)
minPower=min(Powermatrix);

minPowerindex= find(Powermatrix==minPower);

optimumr1value= r1t2(minPowerindex)
```

```
optimumr2value= r2t2(minPowerindex)

MATRIXPOWER= [r1t2, Powermatrix, r2t2];

r1= min(optimumr1value);

r2= min(optimumr2value);

%step 2:simulating using the optimal values

% center of frame to leg in frame 0
Va1= -norm_Va.*[r1*0.165; r2*-0.17775; 0];
Va2= -norm_Va.*[r1*-0.165; r2*-0.17775; 0];
Va3= -norm_Va.*[0; r2*0.17775; 0];

%matrix initialization
force1=[];
force2=[];
force3=[];
add_on=[];
MAX_Matrix=[];
Length1=[];
Length2=[];
Length3=[];

%time initialization
t=0;

Number_of_iterations=Number_of_iterations*5;

x=x/5;

iteration=0;

while iteration<Number_of_iterations
% height, roll and pitch of the frame


Zcenter= Zcenter_sign*Zcentervalue * sin (t * omega_Zcenter);
Zcenter_dot= Zcenter_sign*Zcentervalue*omega_Zcenter*cos(t*omega_Zcenter);
Zcenter_double_dot= Zcenter_sign*Zcentervalue*(omega_Zcenter^2)*(-1)*sin(t*omega_Zcenter);

theta= theta_sign*(pi/6)*sin (t * omega_theta);
theta_dot= theta_sign*(pi/6)*omega_theta*cos(t*omega_theta);
theta_double_dot=  theta_sign*(pi/6)*(omega_theta^2)*(-1)*sin(t*omega_theta);

phi= phi_sign*(pi/4)*sin (t * omega_phi);
phi_dot= phi_sign*(pi/4)*omega_phi*cos(t*omega_phi);
phi_double_dot= phi_sign*(pi/4)*(omega_phi^2)*(-1)*sin(t*omega_phi);

% center of the frame coordinate
Vp=[0; 0; Zcenter+0.5775];

% rotational vector
platform_pitch_matrix= [ 1, 0, 0; 0, cos(theta), sin(theta); 0, -sin(theta), cos(theta) ];
```

Figure 16: Pages 5 and 6 of the MATLAB code

```
platform_roll_matrix= [ cos(phi), 0, -sin(phi); 0, 1, 0; sin(phi), 0, cos(phi) ];
Mall= platform_pitch_matrix*platform_roll_matrix;

% leg at frame 0 to leg at frame 1
Displacement_of_link_1= Vp - Mall * Vb1;
Displacement_of_link_2= Vp - Mall * Vb2;
Displacement_of_link_3= Vp - platform_pitch_matrix * Vb3;

% graph of the leg movement

figure(25)
line1=
line([Va1(1),Displacement_of_link_1(1)],[Va1(2),Displacement_of_link_1(2)],[Va1(3),Displacement_of_lin
k_1(3)]);
line2=
line([Va2(1),Displacement_of_link_2(1)],[Va2(2),Displacement_of_link_2(2)],[Va2(3),Displacement_of_lin
k_2(3)]);
line3=
line([Va3(1),Displacement_of_link_3(1)],[Va3(2),Displacement_of_link_3(2)],[Va3(3),Displacement_of_lin
k_3(3)]);
line4=
line([Displacement_of_link_1(1),Displacement_of_link_2(1)],[Displacement_of_link_1(2),Displacement_of_
link_2(2)],[Displacement_of_link_1(3),Displacement_of_link_2(3)]);
line5=
line([Displacement_of_link_1(1),Displacement_of_link_3(1)],[Displacement_of_link_1(2),Displacement_of_
link_3(2)],[Displacement_of_link_1(3),Displacement_of_link_3(3)]);
line6=
line([Displacement_of_link_3(1),Displacement_of_link_2(1)],[Displacement_of_link_3(2),Displacement_of_
link_2(2)],[Displacement_of_link_3(3),Displacement_of_link_2(3)]);

set (line1,'lineWidth',3,'color','red')
set (line2,'lineWidth',3,'color','red')
set (line3,'lineWidth',3,'color','red')

hold all
grid on
pause (0.03);
xlim([-1,1]);
ylim([-1,1]);
zlim([-1,3]);
delete(line1);
delete(line2);
delete(line3);
delete(line4)
delete(line5)
delete(line6)
view([1,1,1])

% angles of the links
cos_alpha_link_1= -(Displacement_of_link_1(1)-Va1(1))/(sum((Displacement_of_link_1-Va1).^2));
cos_beta_link_1= -(Displacement_of_link_1(2)-Va1(2))/(sum((Displacement_of_link_1-Va1).^2));
cos_delta_link_1= -(Displacement_of_link_1(3)-Va1(3))/(sum((Displacement_of_link_1-Va1).^2));

cos_alpha_link_2= (Displacement_of_link_2(1)-Va2(1))/(sum((Displacement_of_link_2-Va2).^2));
cos_beta_link_2= (Displacement_of_link_2(2)-Va2(2))/(sum((Displacement_of_link_2-Va2).^2));
cos_delta_link_2= (Displacement_of_link_2(3)-Va2(3))/(sum((Displacement_of_link_2-Va2).^2));
```

```
cos_alpha_link_3= (Displacement_of_link_3(1)-Va3(1))/(sum((Displacement_of_link_3-Va3).^2));
cos_beta_link_3= (Displacement_of_link_3(2)-Va3(2))/(sum((Displacement_of_link_3-Va3).^2));
cos_delta_link_3= (Displacement_of_link_3(3)-Va3(3))/(sum((Displacement_of_link_3-Va3).^2));

%resultant of forces
force_resultants= [(density*thickness*height*(base^3)/48)*phi_double_dot ;
          (density*thickness*(height^3)*base/9)*theta_double_dot;
           mass*(Zcenter_double_dot + 9.81)              ];

%coefficients
coefficients=
[cos_delta_link_1*(Displacement_of_link_1(1))+cos_alpha_link_1*(Displacement_of_link_1(3)-
Zcenter),cos_delta_link_2*(Displacement_of_link_2(1))+cos_alpha_link_2*(Displacement_of_link_2(3)-
Zcenter),0;
        cos_delta_link_1*(Displacement_of_link_1(2))-cos_beta_link_1*(Displacement_of_link_1(3)-
Zcenter),cos_delta_link_2*(Displacement_of_link_2(2))-cos_beta_link_2*(Displacement_of_link_2(3)-
Zcenter),cos_delta_link_3*(Displacement_of_link_3(2))+cos_beta_link_3*(Displacement_of_link_3(3)-
Zcenter);
        cos_delta_link_1,cos_delta_link_2,cos_delta_link_3];

%force exerted by leg
forces= inv(coefficients)*force_resultants;

%force matrices
force1=[force1, forces(1)];
force2=[force2, forces(2)];
force3=[force3, forces(3)];

% leg length
L1=sqrt(sum((Displacement_of_link_1-Va1).^2));
L2=sqrt(sum((Displacement_of_link_2-Va2).^2));
L3=sqrt(sum((Displacement_of_link_3-Va3).^2));

Length1=[Length1,L1];
Length2=[Length2,L2];
Length3=[Length3,L3];

MAX_Matrix=[MAX_Matrix,t];
t=t+x;
add_on=[add_on,1];
iteration=iteration+1;
end

Length1=[0.5, Length1, 0.5];
Length2=[0.5, Length2, 0.5];
Length3=[0.5, Length3, 0.5];

MAX_Matrix=MAX_Matrix+1.*add_on;
MAX_Matrix=[0, MAX_Matrix, max(MAX_Matrix)+1];

%Displacement

spline1=spline(MAX_Matrix, [0 Length1 0]);
spline2=spline(MAX_Matrix, [0 Length2 0]);
spline3=spline(MAX_Matrix, [0 Length3 0]);
```

Figure 17: Page 7 and 8 of the MATLAB code

```
xq=linspace(min(MAX_Matrix), max(MAX_Matrix), 200);

displacement1matrix= ppval(spline1, xq);
displacement2matrix= ppval(spline2, xq);
displacement3matrix= ppval(spline3, xq);

figure (2)
plot(xq, displacement1matrix)
figure (3)
plot(xq, displacement2matrix)
figure (4)
plot(xq, displacement3matrix)

%velocity

velocity1matrixt=diff(displacement1matrix);
velocity2matrixt=diff(displacement2matrix);
velocity3matrixt=diff(displacement3matrix);

xq1=linspace(min(MAX_Matrix), max(MAX_Matrix), 200-1);

velocity1spline= spline(xq1, [0 velocity1matrixt 0]);
velocity2spline= spline(xq1, [0 velocity2matrixt 0]);
velocity3spline= spline(xq1, [0 velocity3matrixt 0]);

velocity1matrix= ppval(velocity1spline, xq);
velocity2matrix= ppval(velocity2spline, xq);
velocity3matrix= ppval(velocity3spline, xq);

figure (5)
plot(xq, velocity1matrix)
figure (6)
plot(xq, velocity2matrix)
figure (7)
plot(xq, velocity3matrix)

%acceleration

acceleration1matrixt= diff(velocity1matrix);
acceleration2matrixt= diff(velocity2matrix);
acceleration3matrixt= diff(velocity3matrix);

acceleration1spline= spline(xq1, [0 acceleration1matrixt 0]);
acceleration2spline= spline(xq1, [0 acceleration2matrixt 0]);
acceleration3spline= spline(xq1, [0 acceleration3matrixt 0]);

acceleration1matrix= ppval(acceleration1spline, xq);
acceleration2matrix= ppval(acceleration2spline, xq);
acceleration3matrix= ppval(acceleration3spline, xq);

figure (8)
plot(xq, acceleration1matrix)
figure (9)
plot(xq, acceleration2matrix)
figure (10)
plot(xq, acceleration3matrix)
```

```
%forces

forcesforspline1=[0, force1,0];
forcesforspline2=[0, force2,0];
forcesforspline3=[0, force3,0];

forcespline1=spline(MAX_Matrix, [0, forcesforspline1, 0]);
forcespline2=spline(MAX_Matrix, [0, forcesforspline2, 0]);
forcespline3=spline(MAX_Matrix, [0, forcesforspline3, 0]);

force1matrix= ppval(forcespline1, xq);
force2matrix= ppval(forcespline2, xq);
force3matrix= ppval(forcespline3, xq);

figure (11)
plot(xq, force1matrix)
figure (12)
plot(xq, force2matrix)
figure (13)
plot(xq, force3matrix)

%power

power1matrix= abs(force1matrix.*velocity1matrix);
power2matrix= abs(force2matrix.*velocity2matrix);
power3matrix= abs(force3matrix.*velocity3matrix);

figure (14)
plot(xq, power1matrix)
figure (15)
plot(xq, power2matrix)
figure (16)
plot(xq, power3matrix)

rotationalvelocity1=(100*60/(2.31*2*pi())).*velocity1matrix;
rotationalvelocity2=(100*60/(2.31*2*pi())).*velocity2matrix;
rotationalvelocity3=(100*60/(2.31*2*pi())).*velocity3matrix;

figure (17)
plot(xq, rotationalvelocity1)
figure (18)
plot(xq, rotationalvelocity2)
figure (19)
plot(xq, rotationalvelocity3)

enginetorque1=(100/2.31).*force1matrix;
enginetorque2=(100/2.31).*force2matrix;
enginetorque3=(100/2.31).*force3matrix;

figure (20)
plot(xq, enginetorque1)
figure (21)
plot(xq, enginetorque2)
figure (22)
plot(xq, enginetorque3)
```

Figure 18: Pages 9 and 10 of the MATLAB code

```
mazen=[displacement1matrix; displacement2matrix; displacement3matrix; xq];

maxlength1=max(displacement1matrix);
maxlength2=max(displacement2matrix);
maxlength3=max(displacement3matrix);

elongation1=max(displacement1matrix)-min(displacement1matrix);
elongation2=max(displacement2matrix)-min(displacement2matrix);
elongation3=max(displacement3matrix)-min(displacement3matrix);
```

Figure 19: Page 11 of the MATLAB code

The following graphs will be used for the sizing of the linear actuator, in case the linear actuator was bought from a manufacturer, we would look at the velocity, displacement and power required and make sure the linear actuator is in compliance, but in case the actuator is to be built, the motor is to be sized first with the rpm and power as a reference, and the body of the actuator will be sized using the maximum displacement and maximum length of the actuator

In the following graphs, the x-axis is time and the y-axis is the component of interest.

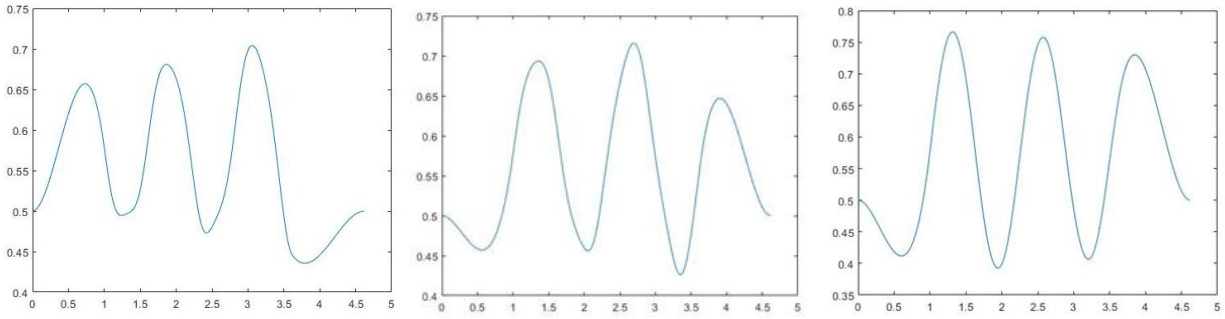


Figure 20: Displacement of legs 1,2, and 3(MATLAB graphs)

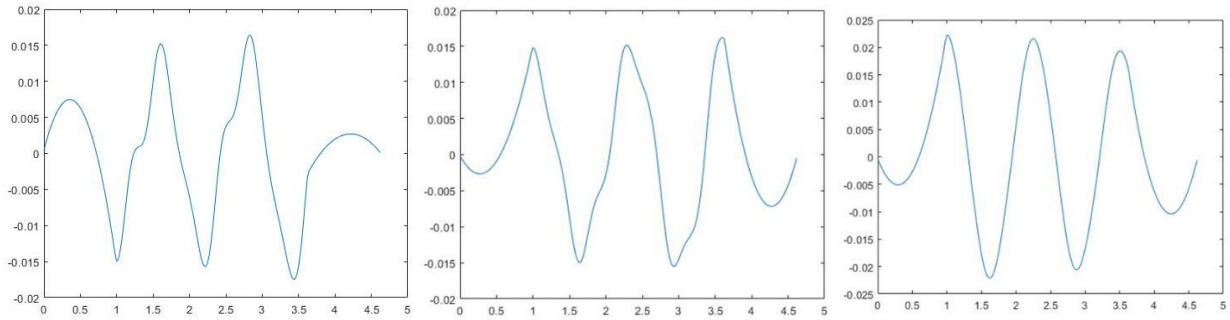


Figure 21: Velocity of legs 1,2, and 3 (MATLAB graphs)

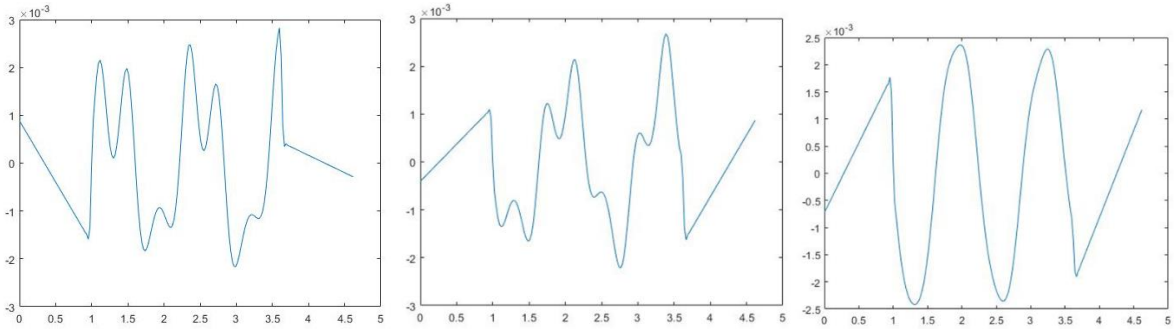


Figure 22: Acceleration of Legs 1,2, and 3 (MATLAB graphs)

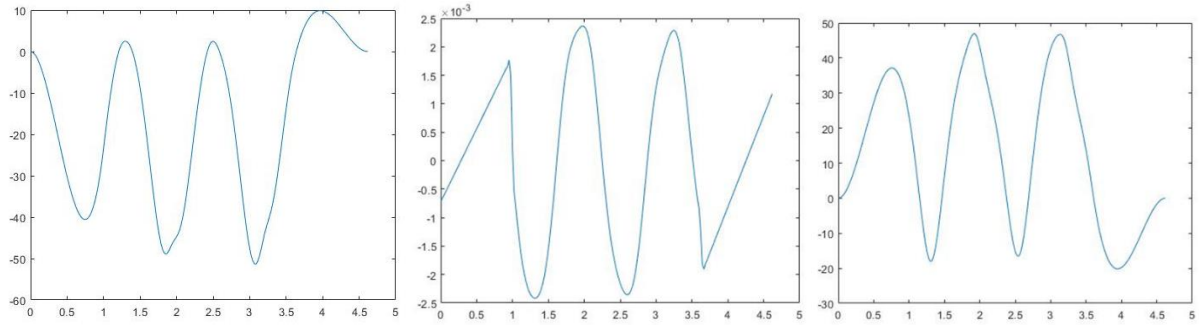


Figure 23: Force on legs 1,2, and 3 (MATLAB graphs)

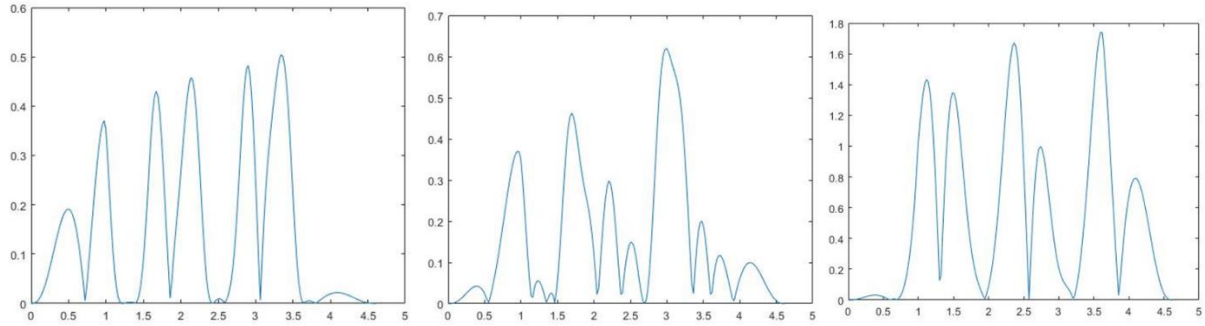


Figure 24: Power used by legs 1,2, and 3 (MATLAB graphs)

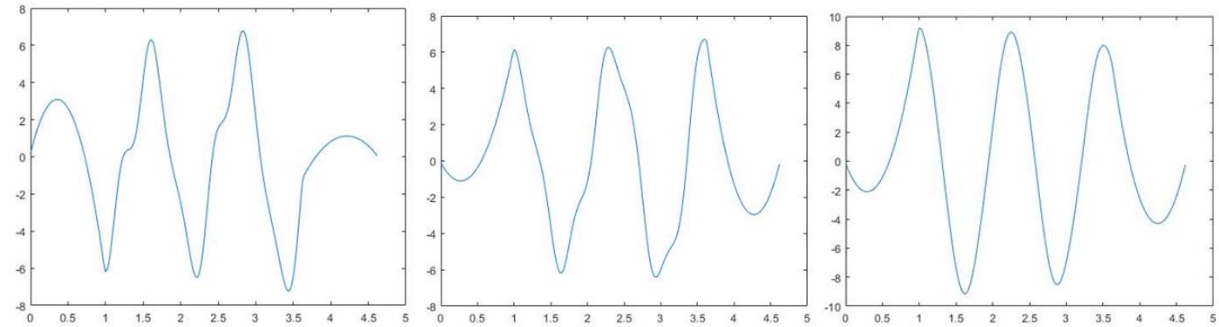


Figure 25: RPM of motor 1,2, and 3 (MATLAB graphs)

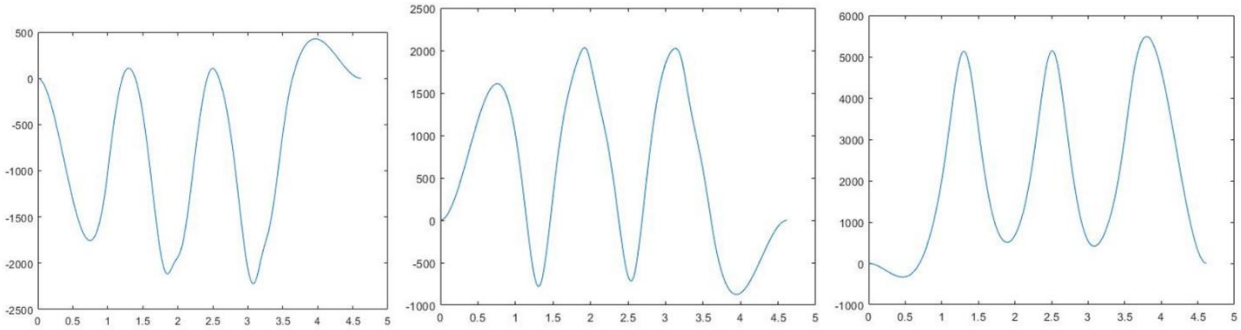


Figure 26: Torque of motors 1,2, and 3 (MATLAB graphs)

LabVIEW Code:

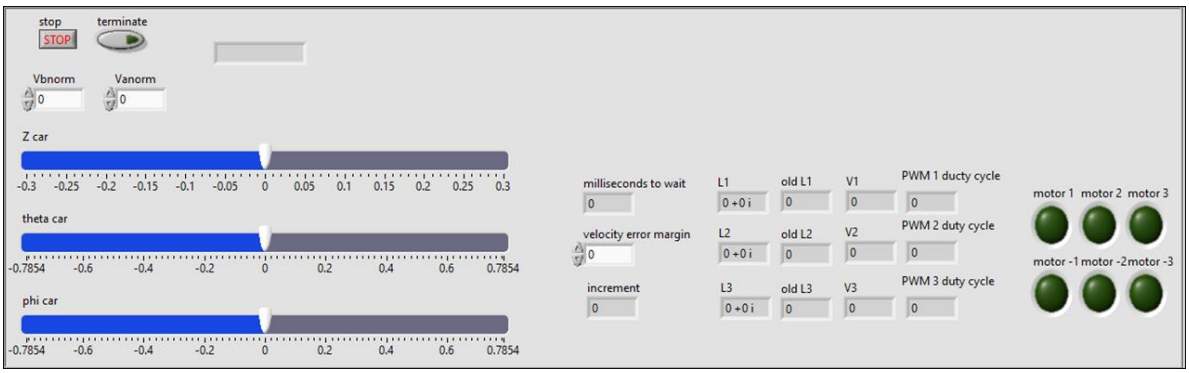


Figure 27: LabVIEW front panel

In the following picture we can see the front panel of the LabVIEW program, due to the circumstances, a MyRio could not be used and therefore 3 sliding bars were used to simulate the movement of the vehicle, which in the case of a MyRio, a gyroscope's output would be the values replacing the sliding bars. Another input are the Vb and Va as previously discussed in the MATLAB code and the SolidWorks design, are inputted to create an initial angle, being part of the optimization process. For the outputs, we can see the different lengths L1, L2 and L3, and their respective velocities, V1, V2 and V3, the sign of the velocities are to control which direction the motor would be running, motor 1, motor 2, and motor 3 being in the positive direction, and motor -1, motor -2 and motor -3 in the negative direction and finally the amplitude of the velocity is what will control the speed which will further be discussed in the block diagrams.

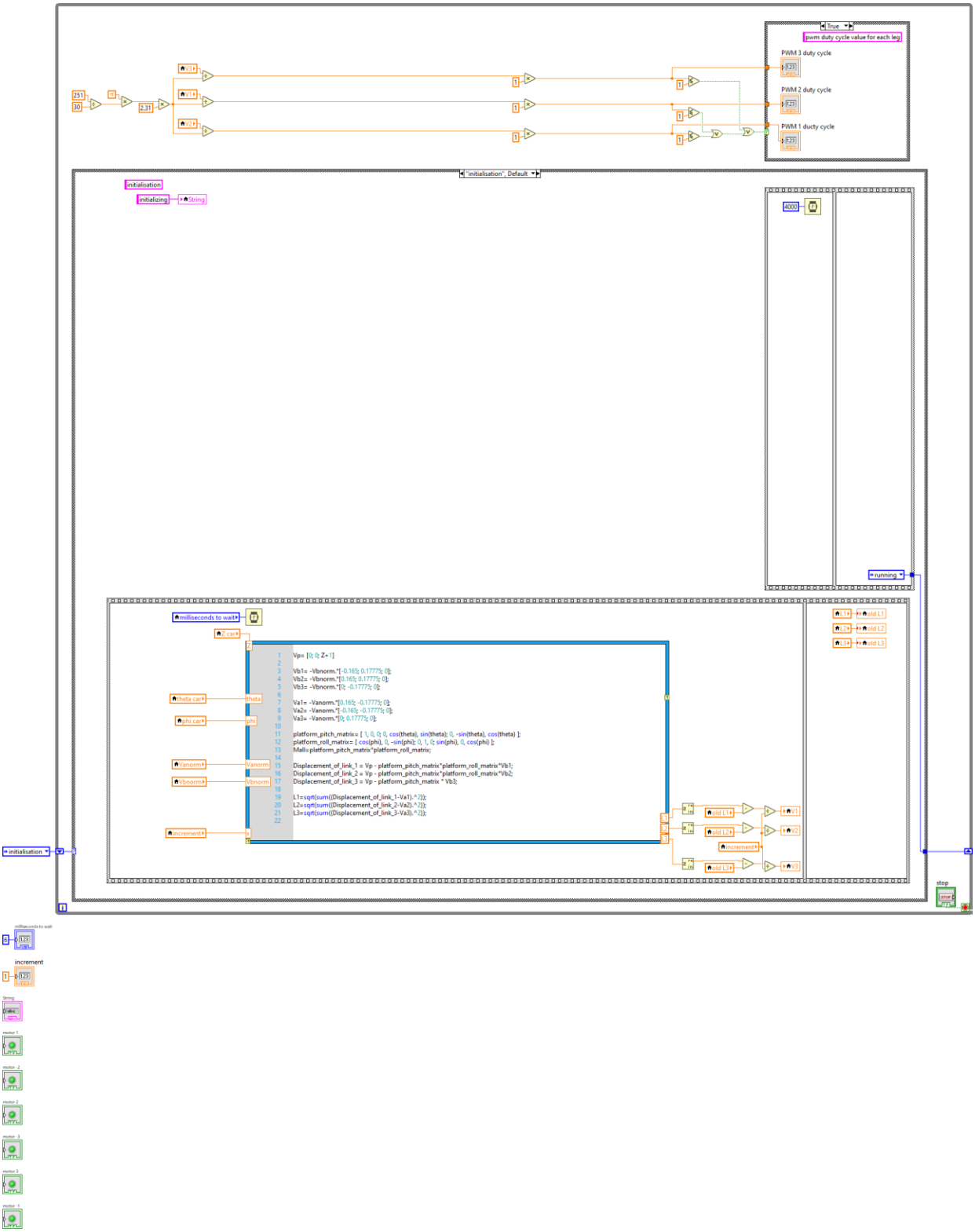


Figure 28: Initialization

The program is being initialization to not create an infinite velocity, the program will insure it has run for at least 4 seconds, and the moves to the running phase.

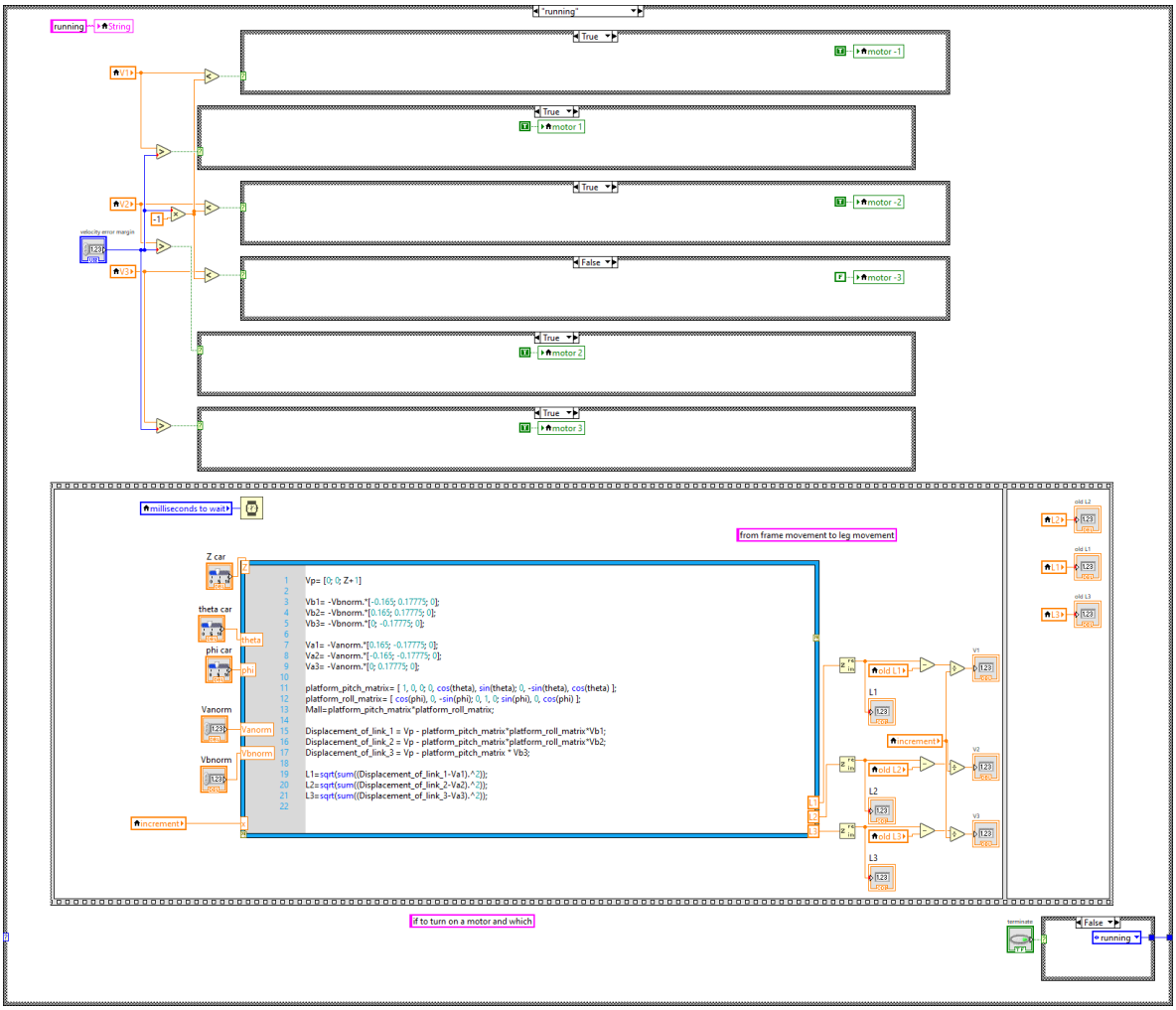


Figure 29: LabVIEW code running

During the running phase, the velocity is constantly being calculated, then the information is read and the program checks in which direction should the motor run.

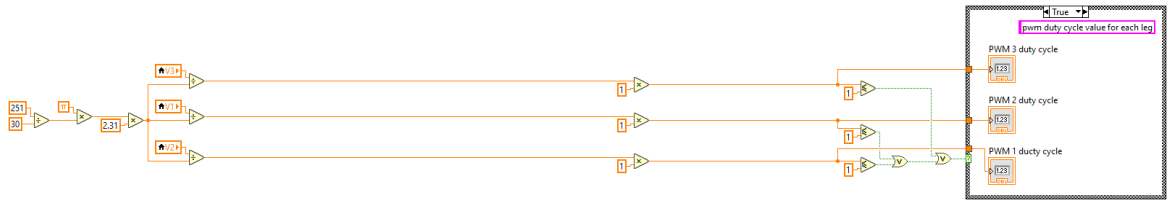


Figure 30: LabVIEW code running

Simultaneously, while the motor direction is being checked, the motor speed is also being fed to the PWM as a duty cycle, by dividing the desired motor speed by its maximum speed.

Results:

Having finished the code, we should be able to see the pistons positions and elongation for every position of the platform (chair). We can now find the forces exerted by each actuator and manage to get the exact positions and elongations of pistons, we were able to find the speed and acceleration of every piston at every single position which will help us analyze the forces and find them more easily. We also solved the problem of elevation, the chair elevates correctly and very accurately as expected.

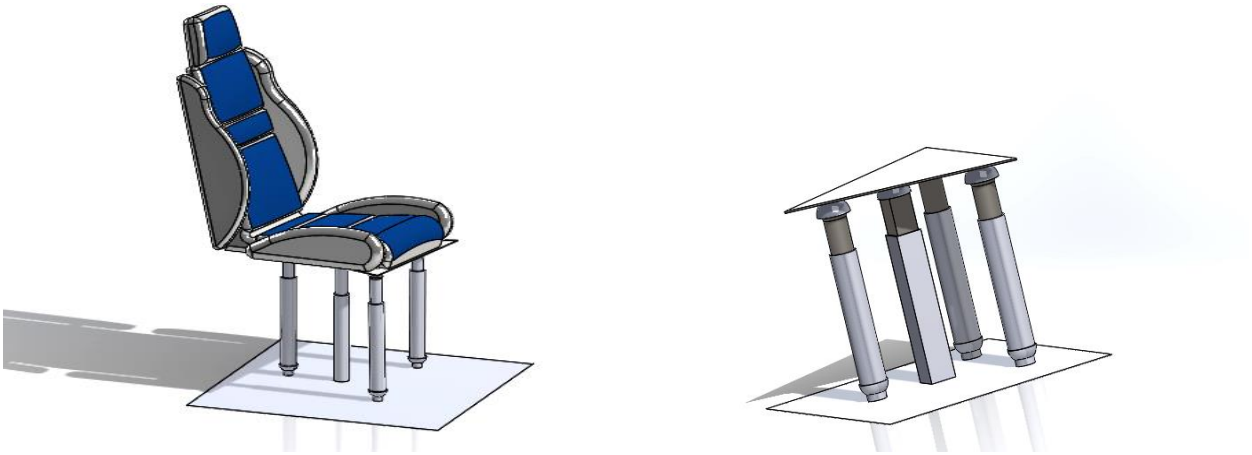


Figure 31: Motion analysis model

This simple model was done for the sake of the motion analysis, and it is designed exactly like the MATLAB parameters and dimensions. The purpose of this process was to get a more accurate and realistic view of the model's motion, and to plot the stress and deformation on the pistons and the platform they are connected to. The above figures connected each piston to the ground and to the platform using ball joints and a fixed piston in the middle to prevent rotation and make the motion more stable. The origins of the platform and the base were mated to one axis.

After getting the displacement data for each leg from MATLAB, the results were imported as text files to linear motors placed at the origin of each piston. The

figures below show the displacement imported to SolidWorks for Leg1, Leg2, and Leg 3, in addition to the velocity, acceleration, and jerk that were traced according to the displacement given.

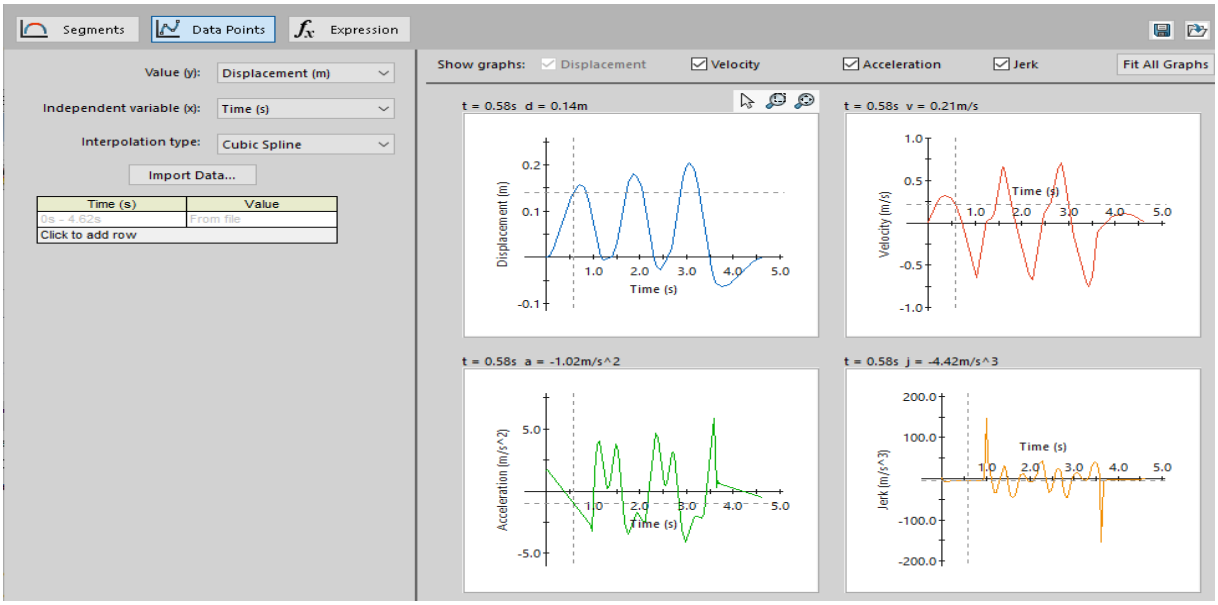


Figure 32: Graphs for Leg1

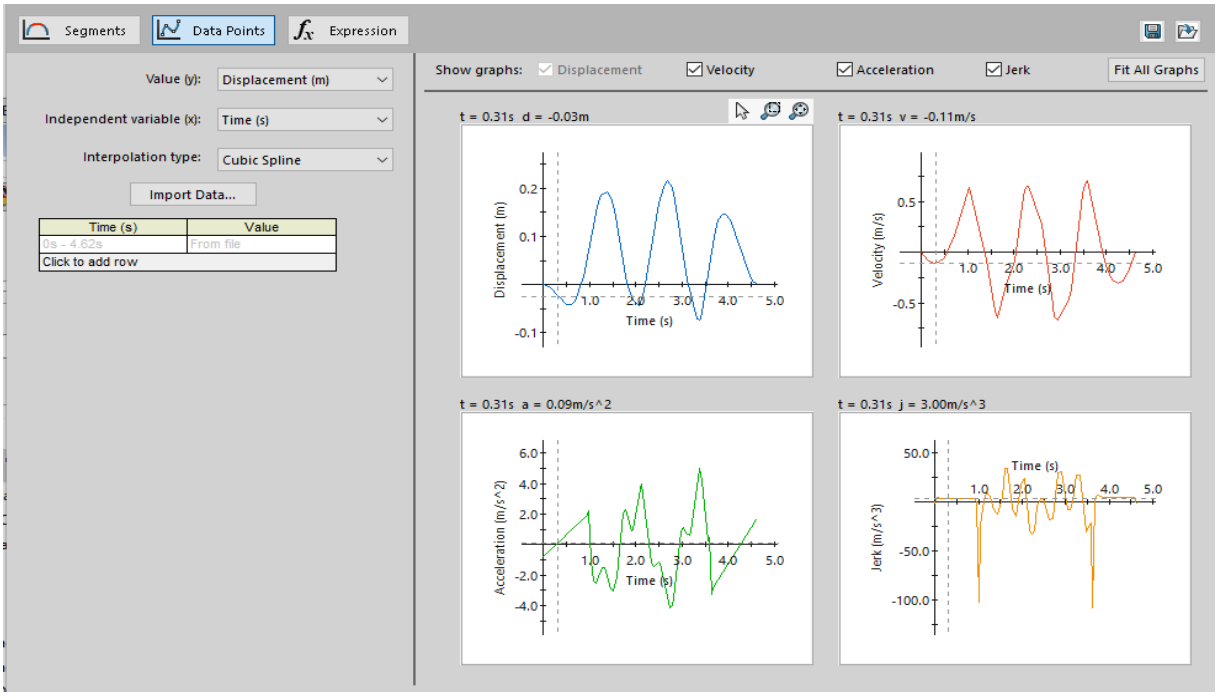


Figure 33: Graphs for leg2

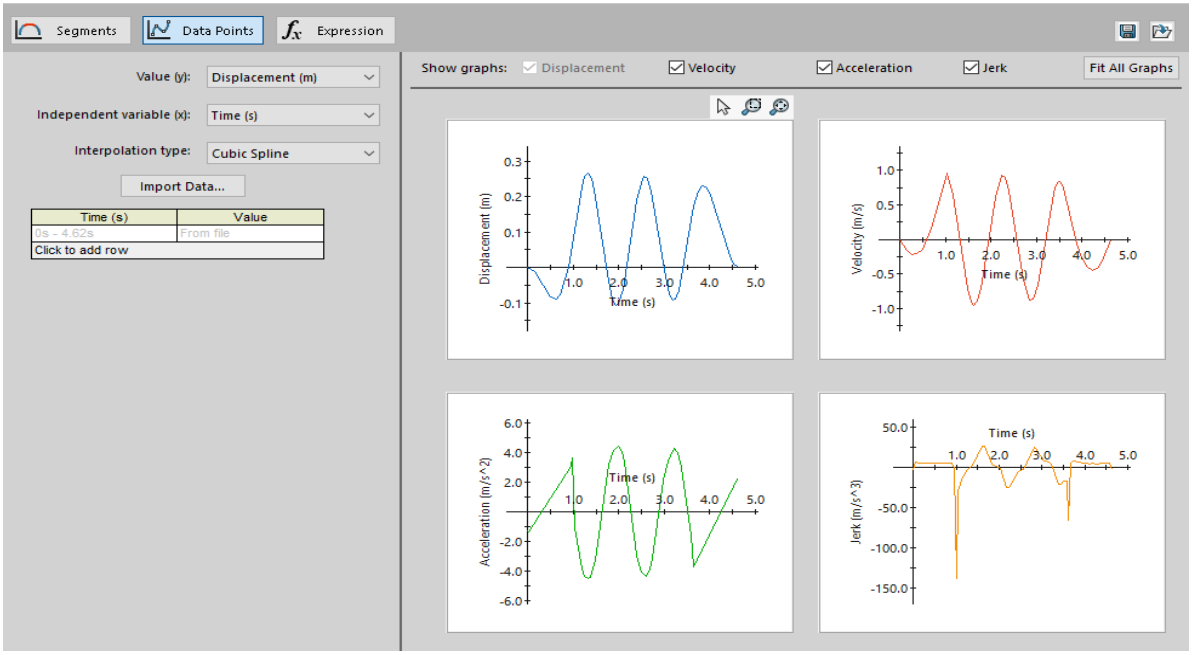


Figure 34: Graphs for Leg3

The graphs show that all legs start from their initial positions at 0 with 0 velocity, and then start to move according to the MATLAB data to get back to 0 with a full cycle at 4.62 seconds.

After running the motion analysis, a simulation setup was designed on each piston and the platform. The material chosen for the simulation was Alloy Steel SS, which has a yield strength of 620421997.8 N/m^2. The figures below show the stress plots of the model on each piston and the platform.

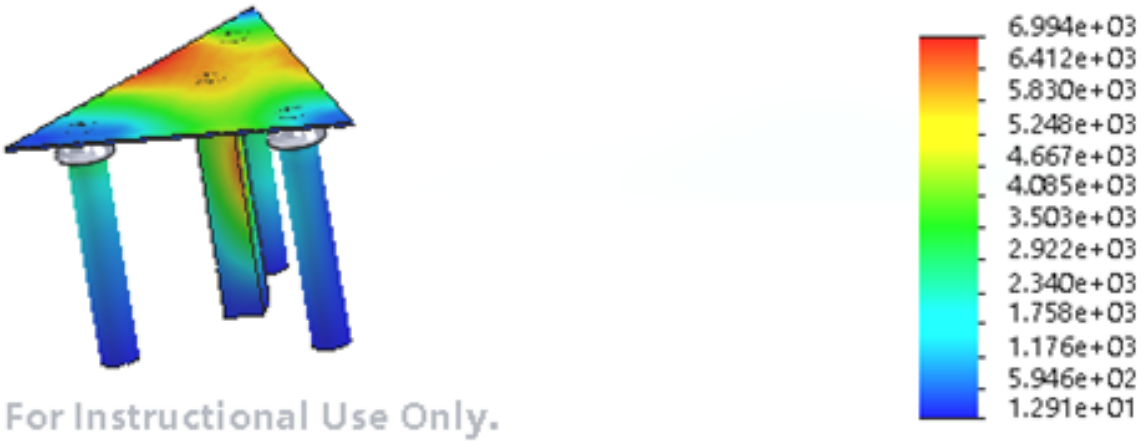


Figure 35: Stress plot view 1

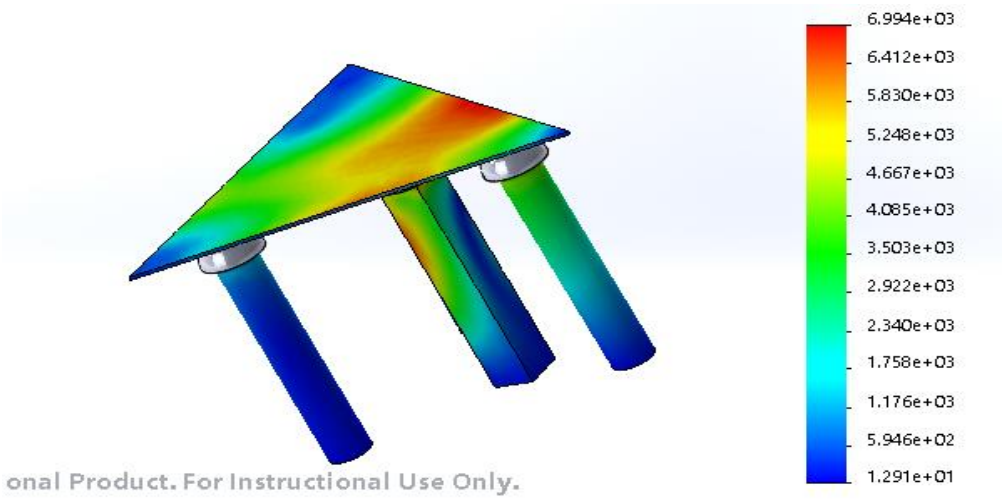


Figure 36: Stress plot view 2

The figures show that the stress was maximum at the middle leg reaching approximately 6.8x10^3 N/m^2, since it is preventing the platform from rotating while the legs are moving. Moreover, it is shown that the maximum stress at the platform was around 6.85x10^3 on the front size between the position of Leg1 and Leg2, due to the forces exerted by both legs.

The following figure shows the deformation plot on the pistons and the platform:

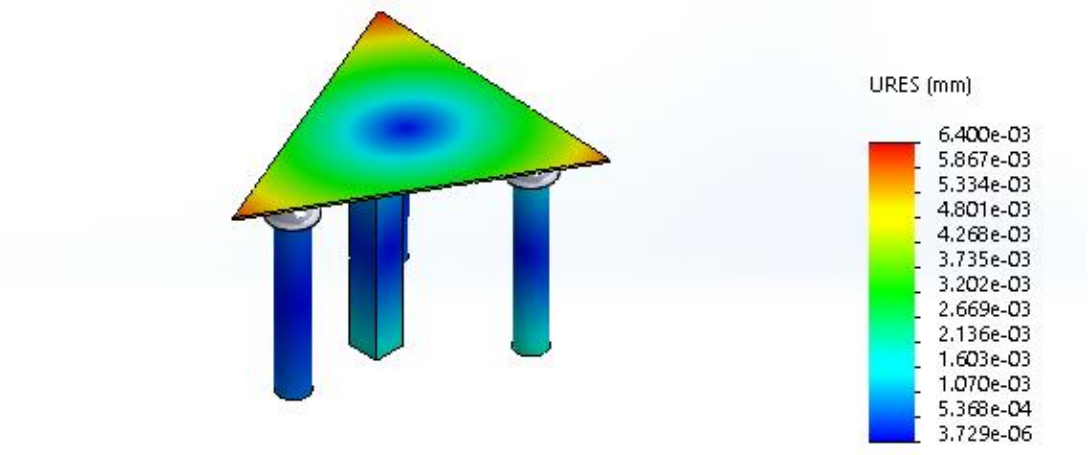


Figure 37: Deformation plot

The plot shows that the maximum deformation was approximately 5.876x10^-3 mm at the free tips of the platform since they don't have any suppor. However, the plot also shows that the minimum deformation was very close to 0 mm at the

middle of the platform due to the fact that the middle leg acting as a support, prevented it from any deformation.

Conclusion:

Augmented reality is still a new domain, but it is a perfect investment since it influences many fields such as medicine, engineering, gaming…

The project designed will ease experiments, and make things seem closer to reality with accurate results and without putting anyone in harm's way. In addition to being an unprecedented technology in the world of science, the fact that the user is controlling a real car while sitting on a chair and getting the experience of being inside of the car is simply fascinating which gives it a huge edge in the world of gaming as well.

References:

- https://www.nature.com/news/safety-survey-reveals-lab-risks-1.12121
- https://www.sciencedirect.com/science/article/pii/S0094114X06001108
- https://link.springer.com/chapter/10.1007/978-3-319-00398-6_12
- https://www.roemheld-gruppe.de/fileadmin/user_upload/downloads/technische_informationen/Wissenswertes_Hydraulikzylinder_en_0212.pdf
- https://ieeexplore.ieee.org/document/770430
- https://www.raconteur.net/digital-transformation/ar-vr-healthcare-surgery
- https://elearningindustry.com/augmented-reality-in-aviation-changing-face-sector-training-simulated-experience
- https://www.aviationtoday.com/2019/08/01/training-brain-mind/
- https://www.technologyrecord.com/Article/how-the-motor-racing-industry-is-leveraging-augmented-reality-61595
- https://www.rivieramm.com/news-content-hub/news-content-hub/simulator-training-reduces-collision-risk-55804