



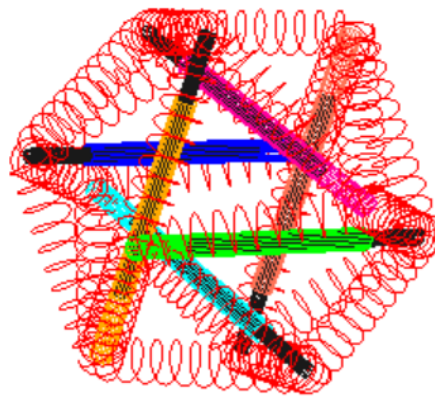
university of  
 groningen

faculty of science  
 and engineering

# Tensegrity Robot Design and Control

## Bachelor Integration Project

Industrial Engineering and Management



George Gittins

S3319040

Supervisor 1:

Zhiyuan Liu

Prof. Dr. Ir. Ming Cao

Supervisor 2:

Mozhdeh Taheri, PhD



# Contents

<b>Abstract</b>	<b>3</b>
<b>Introduction</b>	<b>4</b>
<b>Goals and Objectives</b>	<b>5</b>
<b>Research Questions</b>	<b>5</b>
<b>Problem Analysis</b>	<b>6</b>
Social Aspects	7
System Description	7
Stakeholder analysis	7
Reflection on Research and Design Methodology	9
Technical Aspects	10
Mechanical and Physical Design	10
Mathematical Modelling and Control System	13
Model Testing and Evaluation	22
<b>Discussion and Optimizations</b>	<b>30</b>
Lean	30
<b>Conclusion</b>	<b>33</b>
<b>Bibliography</b>	<b>34</b>
<b>Appendix</b>	<b>37</b>

# Abstract

This report describes the digital creation process of a tensegrity robot and dissects each stage: mechanical and physical design, mathematical modelling and control system, and evaluation and testing. Additionally, for each stage, optimizations and improvements have been identified and are described. These optimizations for the creation procedure help by promoting and aiding further research of tensegrity robots.

The tensegrity robot of this project has been designed in ADAMS, a physical system simulation software. Using a plugin for control and operation, the model has been connected to a Matlab Simulink control system. The tensegrity robot consists of 6 bars and moves by changing the length of the bars, displacing the centre of mass and causing the robot to roll over. The input data of the Simulink control system consists of the x, y and z coordinates of each of the 6 ends of the bars. The output data is the velocity at which each bar changes length.

The robot can roll at a speed of around 0.25m/s, which can be further improved by adjusting the desired lengths of the rods at each time step and the velocity at which they change. As an additional task, an obstacle detection and avoidance function was made that provides the control system with desired coordinates. Given that the coordinates of the obstacles are known, the function can decide whether to move past the left or right side of the obstacle. Eventually reaching the final target destination. Recordings of the simulations have been made, most notably that of the obstacle avoidance function in action.

The improvements include saving time spent on determining parameter values and setting values in the simulation software, with a method for selecting control system inputs and outputs, and adopting best practices for defining performance indicators for specific purposes of a tensegrity robot. The potential uses and hence the market for tensegrity robots is growing, that is why research is needed to develop the creation and operation of these robots.

# Introduction

For this project, a 6-rod-24-string tensegrity robot has been built in ADAMS and controlled using Simulink. (Liu, Z. 2021) ADAMS is a program that facilitates 3d designing of various products and tools that can be used to simulate real-world environments to test the capabilities of the robot model. (MSC Software. 2021) Simulink is an extension of Matlab that allows for visual programming and simulation of physical models. (Mathworks. 2021) During the creation, control, and evaluation design stages of the tensegrity robot, optimizations were made and have been described in this report. The optimizations were discovered by iteratively improving the model according to key performance indicators; velocity, acceleration and force dissipation. These KPIs have been derived from real-world applications, basic capabilities and stakeholder desires for the tensegrity robot.

The tensegrity robot has a spherical shape; its structure consists of rods and cables that are under compression and tension respectively. (Lenarčič, J. et al. 2019) Movement is generated by adjusting the length of the cables or rods thereby shifting its centre of mass and rolling over. (Zheng, Y. et al. 2020) As opposed to more common robots, tensegrity robots are resilient against deformation, able to change shape, and have a high strength-to-weight ratio. (Fivat S. et al 2010) Researchers have managed to optimize the movement speed with artificial learning to 15cm/s for a 13cm diameter robot. However, the resultant movement was not by rolling but with small steps. (Rieffel, J. et al. 2018) The model that has been made in this report has not been operated with artificial learning and moves by rolling.



Figure 1: Example of a tensegrity robot consisting of cables and rods (Fivat S. et al 2010)

There is a large volume of literature on tensegrity robots aimed at optimizing their design and operation (Bagheri, H. 2020) as well as finding potential applications. (Bruce, J. 2016)(Rieffel, J. 2018) For example, since 2013, the Dynamic Tensegrity Robotics Lab at NASA has been developing a tensegrity robot named Superball to safely land on Mars and explore the surface. (Caluwaerts, K. et al. 2014)(Vespignani, M. et al. 2018) By optimizing the stages of the tensegrity model, the progress of these researchers is encouraged. To locate and optimize these stages, the design method, operation method, performance evaluation, and tensegrity robots' applications are observed and discussed.

# Goals and Objectives

This project aims to optimize the design of a rolling tensegrity robot in terms of creating, controlling, and evaluating stages. The first step of this objective is to create a tensegrity robot in ADAMS, then write code to control it with Matlab and Simulink, and consequently attempt to discover the best practices for doing so. The mathematical models, control models, and physical models will be improved by iteratively testing the model. The performance of the tensegrity robot will be evaluated on key performance indicators.

# Research Questions

Since the main objective is to find optimizations in the design of the model, the design process will play a large role. Therefore the main research question should be:

**What steps should be taken to develop a fully optimized model of a rolling tensegrity robot in regards to the creating, controlling, and evaluating stages?**

The main question will require more research and knowledge to strengthen the final solution, this can be achieved by studying and indirectly addressing the following sub-questions:

1. What is the design procedure of a stable tensegrity structure in ADAMS?
2. What calculations should be done to find the optimal tensegrity robot bar change for each desired movement?
3. What should be coded in Matlab Simulink to ensure a direct translation from calculation to bar length change?
4. Which key performance indicators (KPIs) should be used to improve the properties of the tensegrity robot? (such as movement speed, energy cost, and amplitude of vibration)

The above questions were designed to steer the project and are answered in this report. By answering the main research question, a method for creating a tensegrity robot will be developed. This is supported by an explanation of the applied optimizations to the method and design. Therefore readers will be able to easily extract potential optimizations for their methods and designs.

The planning and steps taken for this project were organized and tracked in an Excel sheet. This sheet is shown in Appendix G.

# Problem Analysis

The entire project will be completed digitally; simulations will be run to ensure the tensegrity robot operates successfully in the real world. Each of the design stages will have opportunities for improvement and optimization. To better understand the tensegrity robot, each of the stages have been discussed in the following sections. Figure 2 shows the interrelations between the mechanical design, control system, testing and evaluation and how they contribute to the tensegrity robot model. To optimize this system, especially for a real physical construction, each part must be polished but also the interconnections must be strong such that information can be easily translated between one another. This will make problem-solving much easier.

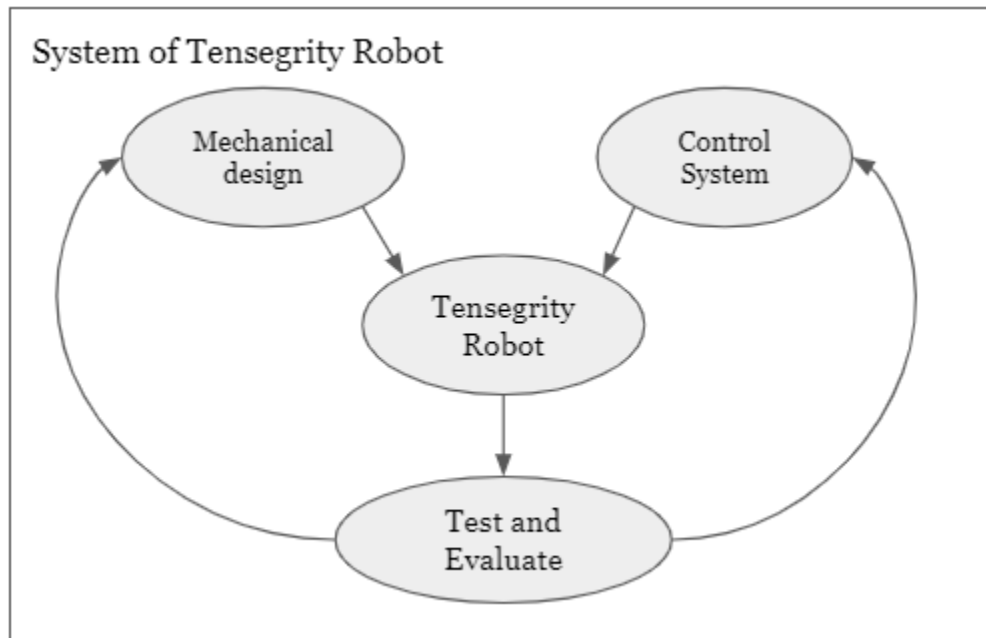


Figure 2: System representation of tensegrity robot design stages

These stages are focused on the technical aspects, from the mechanical and physical design, the mathematics and programming, to the testing and evaluation procedure. However, the social aspects must also be considered. What impact does this project have on the world? In the discussion of the social aspects, the interrelations of the stakeholders will be discussed. This discussion is then used to derive what qualities the tensegrity model must possess.

# Social Aspects

## System Description

The student has taken information from supervisors, researchers, companies, and organizations related to tensegrity robots. This information is used to develop the project. The progress can be tracked and observed by the supervisors, the University of Groningen, and all other parties. Should the project attract interest from external parties such as NASA, CNSA (Bruce, J. 2016), or the local military, they should be able to access this information. This system is further elaborated on with the Mendelow diagram of figure 3.

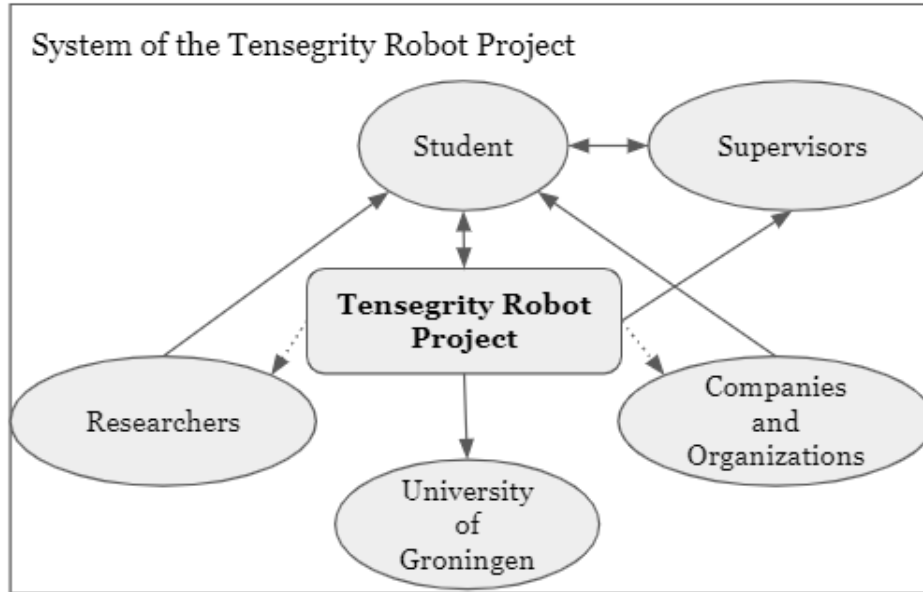


Figure 3: System of the Tensegrity Robot Project

## Stakeholder analysis

The key player for this project is the supervisor, Zhiyuan Liu, for his PhD in automation & control systems he researches this type of robot. The findings of this project will be useful to Zhiyuan for developing his research. For example, Zhiyuan may benefit from the mathematical verifications in Matlab regarding the static and locomotive operation of the tensegrity robot. (Liu, Z. 2021)

The University of Groningen, companies, and organizations, and researchers are external stakeholders. The University of Groningen will take an interest in the work of its students. The crowd of researchers, companies, and organizations working on their tensegrity robots aim to utilize them for labour that requires safe transportation of goods over irregular terrains. Such as planetary exploration, military transportation, and medical assistance. (Kim, K. et al. 2017) A recent development is the combination of a tensegrity robot and a drone. (Liu, Z. 2021) The drone is protected by the tensegrity structure and after landing, the tensegrity can realign the drone for take-off.



The stakeholders help drive the project towards particular goals. Therefore, the project quality requirements of M. Taheri, prof. Dr. Ir. M. Cao and Z. Liu are very important to consider. To further establish quality goals, the interest of the University of Groningen and the relevant companies and organizations is desired. If the project is completed it may attract their attention and result in their decision to fund, support, or take over for further research and implementation. This is why this project should aim to impact the field of robotics; improvements to general code, mathematical interpretations, and optimal bar length changes of tensegrity robots. The stakeholders are presented in the following figure:

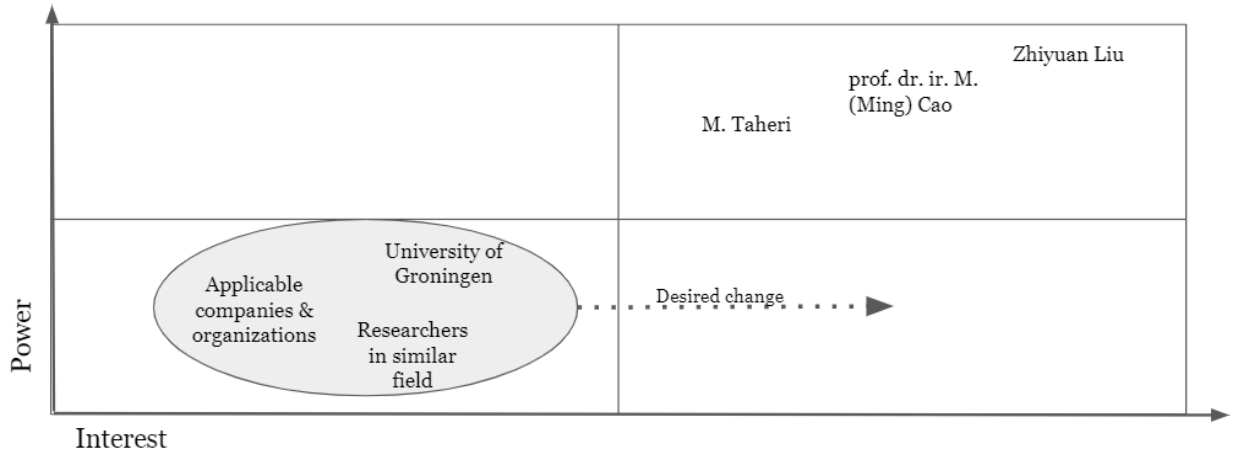


Figure 4: Mendelow Diagram of Ba IP Tensegrity Robot Design and Control

From the desires of the stakeholders, we can conclude that the robot should be able to roll in the desired direction quickly and smoothly. Additionally, the robot should be able to survive an impact such as from a drop test of twice its diameter. Therefore, to ensure the robot rolls fast, velocity will be monitored. To ensure rolling is smooth and any payload is transported safely, acceleration will be monitored. Lastly, strength and safety will be tested by monitoring the forces acting on the springs.

Bar-driven tensegrity robots are not widely used or researched. Cable-driven robots are far more common although bar-driven is easier to model and control. For example, the Superball by NASA is Cable driven. (Vespignani, M. et al. 2018) Due to this, research and papers on bar driven models were harder to find. Additionally, the work that has been completed in three months has the potential to be compressed into a conference paper, this is because the work is integral, from background, to tensegrity modeling, to tensegrity robot locomotion analysis, to locomotion simulation, and to obstacle avoidance.

## Reflection on Research and Design Methodology

The conclusions and decisions made in preparation of this report were made using the problem analysis, stakeholders analysis and system description. These have been further discussed in the research and design plan report submitted before the start of this project. The system description presented in figures 2 and 3 has served as a basis for the procedure of completing this report. As mentioned, each part had to be connected sufficiently to ensure a good outcome. Understanding the role that the project plays in the real world is both a motivating and steering factor. For example, by analysing the implementations of the tensegrity robot, the capabilities of the robot were determined. Additionally, by discussing the stakeholders and their role in the project it was made clear which parties set the standards of the project and which sources of information would be best for the various problems encountered for the project. The first supervisor, Liu, Z., was best to approach regarding technical issues and it became clear that they valued developments in the technical aspects more than social aspects. As opposed to the second supervisor, Taheri, M., who was a better source for social aspects.

Although a large part of the planning made in the research and design plan has remained unchanged, specific decisions had to be made as the project progressed. For example, once the robot was able to roll the following step was to make new environments to test the robot in. Once this step was completed, a decision had to be made: either sloped environments or obstacle-filled environments. The outcome was to design obstacles and create a route planner for the robot to follow and avoid the obstacles. Logistically and in terms of utility, this was a more productive next step due to the larger impact on the overall ability of the tensegrity robot. Additionally, the performance indicators suggested in the research design plan were changed as the project progressed and the final goal became clearer. Other than that, the design plan was followed and consistently reviewed for reference during both the technical design and writing of this report.

Overall, the process of dissecting literature and writing a research and design plan has helped give meaning to the project as a whole. Understanding the potential impacts of the project results in a motivationally driven project with set expectations and standards. The following sections should help support the argument that these expectations have been met. The expectation that a tensegrity robot has been created, simulated and controlled successfully and is supported by a report that clarifies both the procedure of creating and improving the robot towards specific targets.

# Technical Aspects

## Mechanical and Physical Design

First, the tensegrity structure needed to be created in ADAMS, this required practice in using the program by following tutorials. (Norton, R. 2020) The tensegrity robot has a defined mechanical structure; springs with dampers, displacement of joints, mass, gravity, contact force, and friction with the ground; Each of which can be defined in ADAMS. The model consists of exactly 6 rods and 24 cables, resulting in an icosahedron shape, as shown in figure 1 and 5. (Rhodes, T. et al. 2019) To organize the model the rods will be labelled with a number from 1 to 6. With each end of the rod labelled either A or B depending on the radius at that end.

The only change that can be made in simulations is the length of the rods, which are limited to a minimum and maximum length, and the cables connecting the rods will have a fixed length. (Liu, Z. 2021) The selection of the bar length range is an arbitrary choice in the context of this project, however, the NASA Superball has a diameter of around 1.5m (Bruce, J. 2016) so a maximum bar length of 1.6m will be selected. To be more specific, each bar will consist of two concentric parts 0.8m in length that expand and contract at a range of 0.85 to 1.55m. The radii of the concentric rods are 0.02m and 0.03m and have been determined in combination with the mass due to the direct relationship between the two properties.

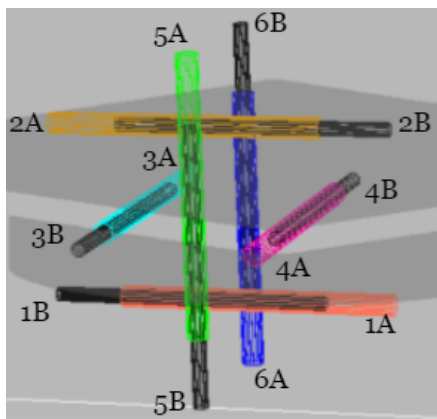


Figure 5: Tensegrity robot rods with coloured and labelled ends

Each outer end of the bar is connected to four springs (interchangeable with cables) that provide tension between the bars. Each spring will have the same properties. These include the stiffness coefficient, damping coefficient and preload. These values cannot be selected arbitrarily because they individually influence the behaviour of the model. Additionally, they are dependent on the mass and length of the 6 bars.

For the selection of these parameters, a comparable study was observed. (Luo, A. et al 2016) Their selected parameters are shown in table 1 below and compared to those of this project.

Table 1: Bar and spring parameters selected for comparable tensegrity robot design projects.

Parameters	Luo, et al 2016	This project
Bar Length (m)	0.2	1
Bar Weight (kg)	0.15	1
Stiffness Coefficient (N/m)	500	1000
Spring Damping coefficient (Ns/m)	50	100
Spring Preload (N)	10	20

Note that the bar length and mass are at least five times larger for this project, however, the remaining values are doubled. Using the comparative study these values were initially set and experimentally adjusted. By simulating the model falling and analysing the behaviour of the forces, the coefficients and preload were determined. The preload is the tension in the spring due to the current displacement from the rest position. A preload of 20N means that the spring is pulling axially with a force of that magnitude. The values were selected such that the magnitude of the force acting on the springs was minimized as well as the rate at which the force dissipated was suitable for the protection of a payload.

The tensegrity robot needs to interact with the floor. This means that dynamic and static friction constants must be determined for the contact between the bars and the ground. Research on common surface interactions and their coefficients showed that the static friction values for metal or plastic in contact with common ground types ranged between 0.3 and 0.8 and the dynamic friction values ranged between 0.1 and 0.5. (Engineering ToolBox, 2004) Therefore the selected values for these simulations would be 0.5 and 0.3 for static and dynamic friction respectively. Earth's gravity of  $9.807 \text{ m/s}^2$  is used for the simulations, although that of Mars,  $3.711 \text{ m/s}^2$ , is also used for double-checking the stability of the structure. (MSC Software. 2021) These are predefined values that can be selected in ADAMS simulation, however, in most cases, behaviour does not change much and result in the same changes to the model.

Next, the concentric translational joints between each of the bar pairs had to be defined. The motion of the bars would be limited to these joints. Additionally, the tension due to the springs must not result in motion along these joints. In other words, the length of the bars shouldn't change unless requested by the control system. Therefore the joints require a high enough static friction value such that the tension of the eight springs does not overcome the friction. The tension will be at maximum when the bars are fully extended. To not spend time calculating the minimally required friction coefficient, this value can be set arbitrarily high. In the simulations of this project, the static and dynamic friction values have been set to 2.0, which is twice as large as rubber in contact with asphalt. (Engineering ToolBox, 2004) For this project the value has no impact on other aspects, therefore further consideration is not necessary. The friction is

dependent on the actuator system used in the bars. This decision will be further addressed in a later section.

The current physical model of this project is shown in Figure 6. There are three pairs of bars in parallel. The outer concentric bars are each coloured differently whilst the inner bars are all coloured black. This was done for visual aid in the simulations. The springs are coloured red and connect bars to the ends of the nearest four orthogonal bars.

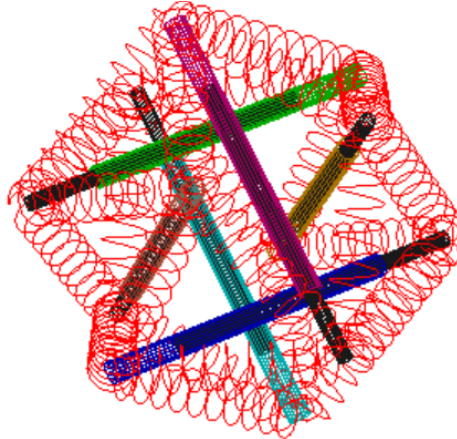


Figure 6: Tensegrity robot of IEM Integration project “Tensegrity Robot Design and Control” design in ADAMS

The following table 2 lists each of the rods and their respective colours:

Table 2: Tensegrity structure rod labels and respective colours

Rod	Colour (Based on ADAMS)
1A	Coral
2A	Maize
3A	Cyan
4A	Magenta
5A	Green
6A	Blue

As mentioned earlier, each rod has two concentric parts, the larger part is coloured and labelled with an A. The thinner parts are black and labelled with a B.

## Mathematical Modelling and Control System

The control system has been built in Matlab Simulink. Simulink is used to generate control inputs to control the length of the rods of the tensegrity robot in ADAMS. See Appendix B for an example of a Simulink control model. The model then outputs values as feedback to Simulink. Based on the performance of the robot model in simulated scenarios, the model can be evaluated to improve the system.

The structure of the robot model can be represented with a graph  $G$ , consisting of vertices and edges which are used as both input and output data for the model.  $G$  is graphically represented as shown in Figure 7. The forces and positions at the ends of the six rods will be represented by matrix values:  $P$  and  $F$ . There will be 12 respective points that exhibit these values along the x, y, and z-axis. The position, velocity and acceleration of these points can be tracked. (Wang, M. et al. 2020) (Connely R. 2002) These values are then interpreted by the Simulink control system to formulate the next movement and translate it to input data for the robot model. These values can also be used to calculate the centre of mass. For example, using the average position of the rod ends.

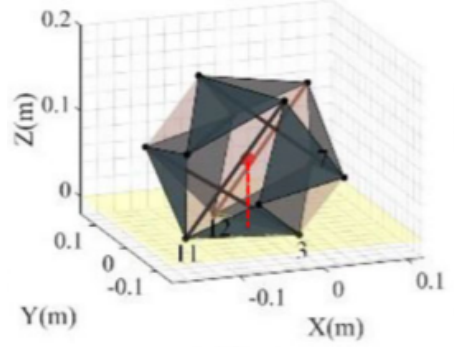


Figure 7: Graphical representation of tensegrity structure (Zheng, Y. Et al 2020)

It may not be clear at first how the tensegrity robot maintains its structure and does not collapse.

The most straightforward explanation of how equilibrium is maintained by the tensegrity structure would be how the resultant force at each rod endpoint,  $F_i$ , is kept equal to zero.

$$F_i = \sum_{j=1}^4 (T_{i,j}) + R_i + f_{ext} \quad \text{Eq. 1}$$

Where  $F_i$  is the resultant force at any of the 12 rod ends,  $T_{i,j}$  is the tension force from any of the four cables attached to that rod,  $R_i$  is the reaction force from the rod and  $f_{ext}$  is any external force. All values are vectors in 3D space and the external force could be weight due to gravity or contact force.

$T_{i,j}$  would be calculated for each cable according to Hooke's Law. The force is parallel to the cable and proportional to the change in length due to stress.

$$T_{i,j} = k(l_1 - l_0) \quad \text{Eq. 2}$$

Where  $k$  is Hooke's coefficient that relates to the elastic properties of the cable material and cross-sectional area. The change in length is calculated by subtracting the length at rest from the current length. These lengths are found by calculating the distance between the related points  $p_i$  and  $p_j$  from the aforementioned matrix  $P$ . For each point to be in equilibrium  $F_i$  must be equal to zero. This condition is only possible if  $R_i$  is equal and opposite to the remainder of equation 1:

$$-R_i = \sum_{j=1}^4 (T_{i,j}) + f_{ext} \quad \text{Eq. 3(1)}$$

The force  $R_i$  is always parallel to the length of the respective rod. Therefore, the forces at point  $p_i$  that are orthogonal to the rod length must add up to zero. As mentioned in an earlier section, the rod lengths are the only part of the model that can be manipulated. All other changes are caused by the change in rod length. (Liu, Z. 2021)(Wang, R. et al. 2020)

The tensegrity robot's movement is described by its 'gait'. (Zheng, Y. et al. 2020) It will be important to ensure that, while rolling, the robot has a good gait. Not every rod length and structure will keep the robot in equilibrium. The structure must remain stable therefore every change in rod length must be compensated for by simultaneously changing other rod lengths. (Liu, Z. 2021)

ADAMS has an inbuilt plugin that connects the model to Simulink. To create this connection, inputs and outputs needed to be selected. This stage is important because it will have an impact on the design of the control system. As mentioned earlier, the location of each rod end can be tracked as a point in three dimensions. Therefore, it would be crucial for this information to transfer from ADAMS to Simulink. The information Simulink requires includes the location of the centre of mass of the tensegrity model, the length of the rods and the location of the ends of rods. These are calculated using the coordinates of each of the bar ends. The locations are vector coordinates from a fixed origin point. In total, this produces 36 output values from ADAMS (in meters). This data is sufficient for finding the displacement, velocity and acceleration of each of the parts. The following step is to determine the best input data for ADAMS. The only change that can be made is the length of the rods, this can be done by requesting a specific length, velocity, acceleration of the rod. ADAMS has a feature named 'motions' where this physical change is defined. All of the possible change requests are viable. For example, in the current design and research of Liu Z. ( as of 2021), the motion is based on an axial force applied to the end of the rod.

The input data was selected by picturing how the tensegrity robot bars would operate in the real world. As an example, the concentric bars will contract and expand using a motor attached to one end with a rod threaded through the two bars. (Bagheri, H. et al 2020) When the motor is active, the thread rotates and adjusts the distance between the two ends of the bar at a constant speed. Based on this actuator model, the best length change measure would be the velocity. The information the motor would receive would only be to turn on or off and in which direction to turn. Acceleration, force or desired location would be less intuitive. This decision to use velocity also eliminates further need to consider the friction in the bars when changing length. Therefore, the Simulink model will take the location data of the tensegrity robot and inform the robot of the velocity at which each bar should contract or extend.

The Simulink model developed for manual operation is shown in Figure 8.

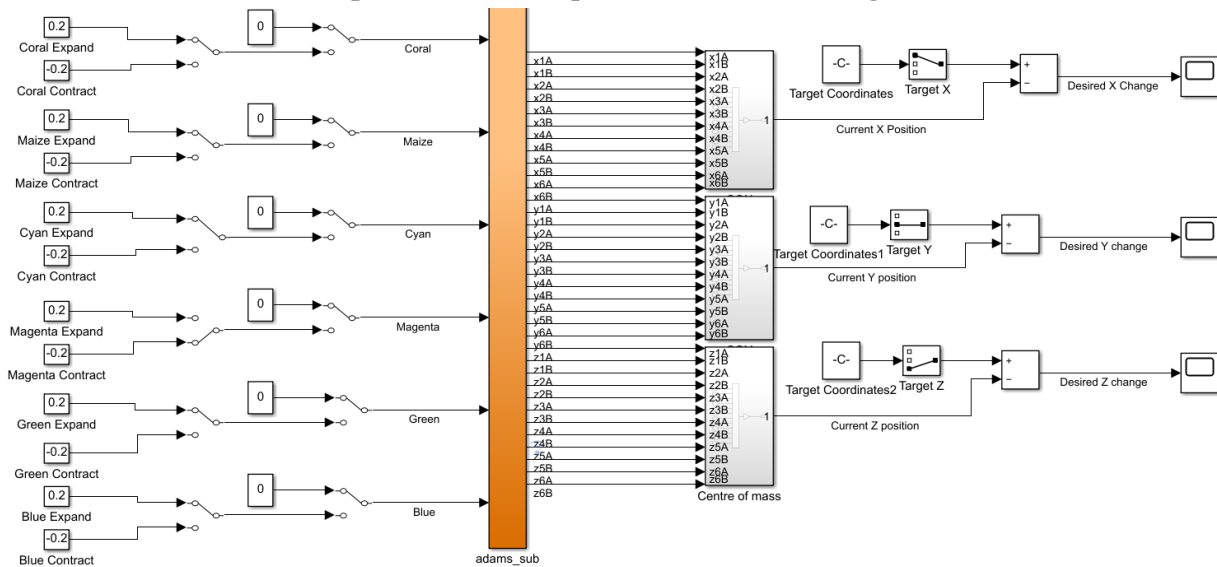


Figure 8: Simulink manual control system of tensegrity robot bars length change rate and calculating distance from the desired centre of mass coordinates

Figure 10 consists of the control model that has been made in Simulink up to this point. The orange bar represents the tensegrity robot model in ADAMS. On the left side of the bar, 6 inputs equal the desired velocity of the respectively colour labelled tensegrity bar. When the Simulink model runs, the simulation in ADAMS starts and the robot's motion is visualized. The switches can be changed during the simulation to change the input velocity. As shown in figure 10, the system starts with all the switches set to zero. On the right side of the orange bar, the centre of mass is calculated and related to a pre-defined desired location for the tensegrity robot to roll towards.



This manual control allows for the testing of theory. Once the correct combination of rod movements has been derived in theory, this can be applied to automate the manual control system. The following step in this project will be to connect the right and left side of the orange bar in the control model to make the tensegrity robot automatically roll towards the desired position. This will require using the location data to select the best rods to adjust and subsequently rolling towards the target location.

The following step is to find the right rods to adjust. At any point, the model is expected to be standing on three different rods that form a triangle. If the centre of mass can be shifted past the edge of this triangle the tensegrity robot will roll.

This triangle is visualized in figure 9, with three rod ends in contact with the ground, the centre of mass and the projection of the centre of mass to the ground. (Liu, Z. 2021)

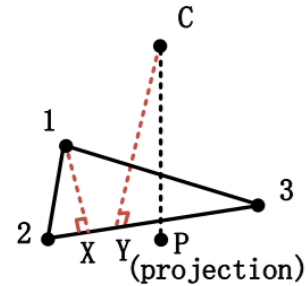
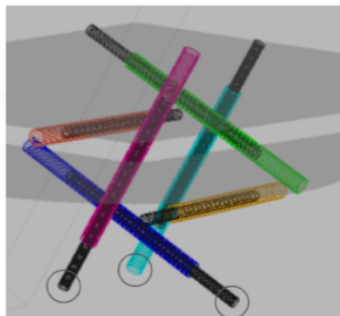


Figure 9: Projection of tensegrity robot contact with ground and centre of mass (Liu, Z. 2021)

Two triangle types are formed; A closed face that resembles an equilateral triangle and an open face that resembles an isosceles triangle. Examples of these two types are shown below as well as a list of all 20 triangles formed by the tensegrity robot structure.

Open Face:

1. 1A - 5B - 6A
2. 1B - 5B - 6A
3. 2A - 5A - 6B
4. 2B - 5A - 6B
5. 3A - 1B - 2A
6. 3B - 1B - 2A
7. 4A - 1A - 2B
8. 4B - 1A - 2B
9. 5A - 3A - 4B
10. 5b - 3A - 4B
11. 6A - 3B - 4A
12. 6B - 3B - 4A



Closed Face:

1. 1A - 5B - 6A
2. 1B - 5B - 6A
3. 2A - 5A - 6B
4. 2B - 5A - 6B
5. 3A - 1B - 2A
6. 3B - 1B - 2A
7. 4A - 1A - 2B
8. 4B - 1A - 2B

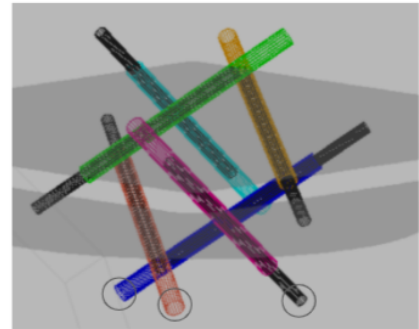


Figure 10: Lists and example diagrams of the tensegrity robot's triangles formed by the rod ends

The manual operation tests helped discover that the robot was able to roll in the desired direction by extending one rod and contracting two. More specifically, in figure 9, if point 3 is the furthest from the desired direction then the rod at point 3 should extend and rods at point 1 and 2 should contract. This concept was a successful base model for automatic locomotion. The manual Simulink model was then adjusted for automation. Given the locations of the rod ends and the coordinates of the target location, the model could define desired lengths of the rods at any time.

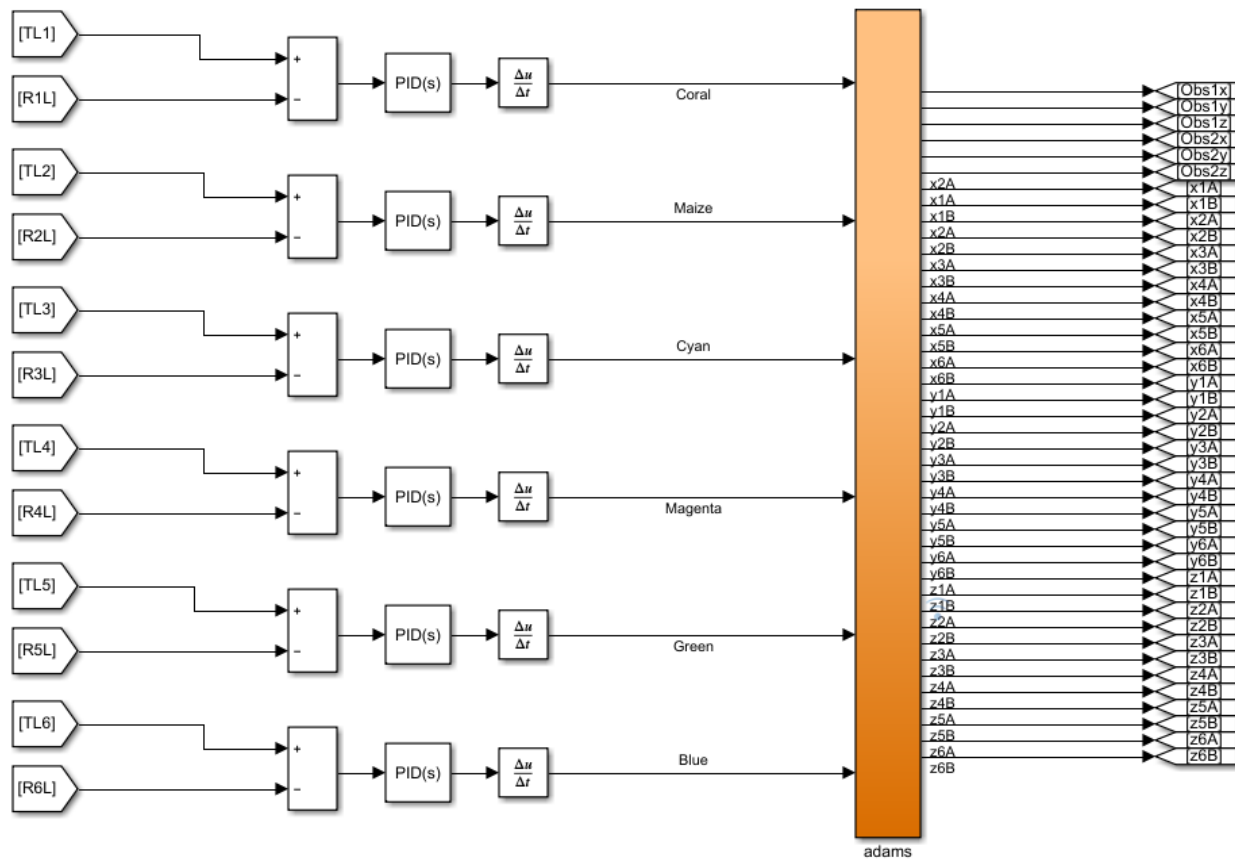


Figure 11: Simulink automatic control system of desired tensegrity robot bars length

In this case, the switches have been replaced with PID controllers. The model detects a difference between the current length and desired length, this difference is used by the PID controller to determine a smooth transition towards the desired length. The PID controller requires values for P, proportion, I, integral and D, derivative. To explain briefly, the value for proportion determines how quickly the desired length should be reached, the value for integral reduces undershooting and ensures the desired length is reached, and the value for derivative lowers the chance of overshooting and going past the desired length. These values were found experimentally. In the final simulations, the values selected for the PID controller were 2, 1 and 0 respectively. There were no cases of overshooting, therefore making it a PI controller. (Douglas, B. 2018) The output of the PI controller is a length, to change it to a velocity a derivative block is used. That value is used as the input for each rod's velocity.

The output taken from ADAMS is the coordinates of all the rod ends and that of the obstacles used for later simulations. To determine the location of the tensegrity robot, all 36 coordinates are needed. The coordinates are organized into vector sets of 12 X-values, 12 Y-values and 12 Z-values. This organization is done using the block shown below:

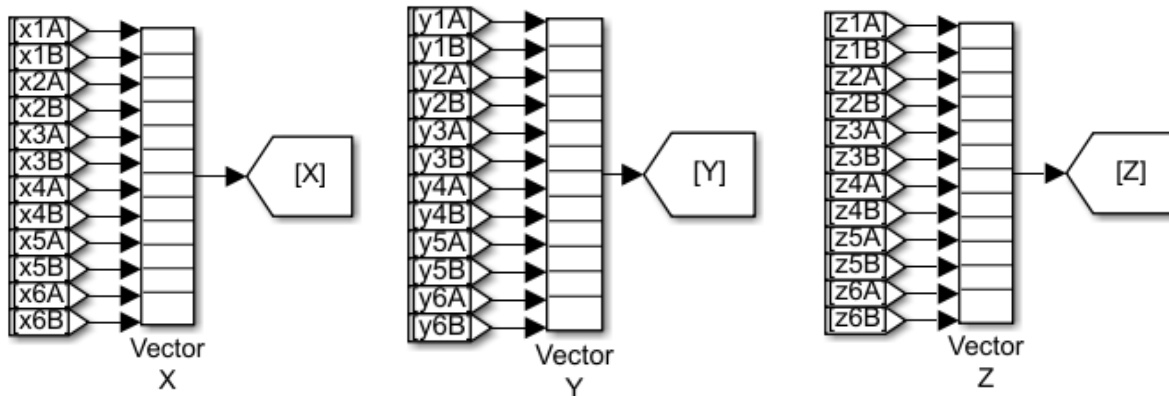


Figure 12: Simulink block used to organize tensegrity x, y and z-values into one matrix array

The locations of the rod ends were used to determine the length of the rods at any time using the Pythagorean theorem. A useful tool in Simulink is the Matlab function block where the operation on any variables can be manually defined, such as the one in the figure below.

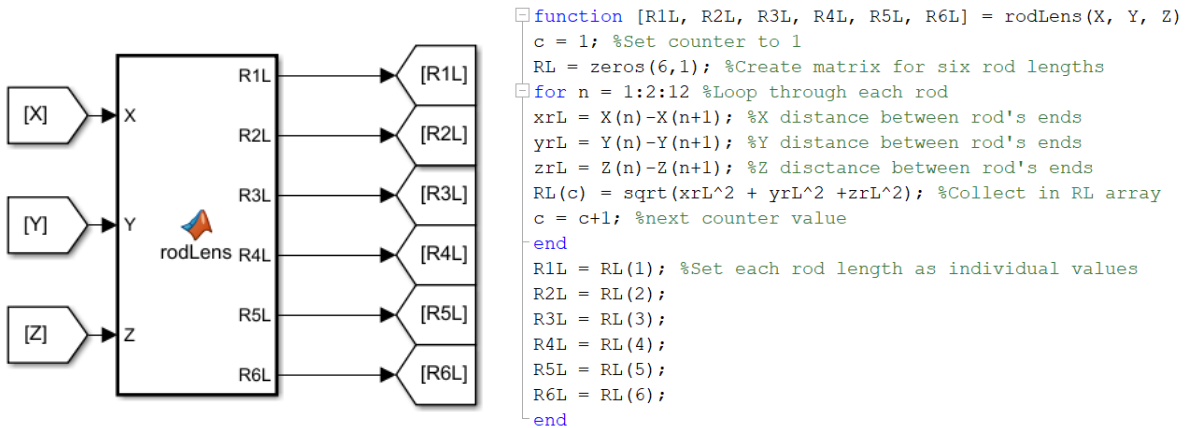


Figure 13: Simulink function for determining the length of the 6 tensegrity robot rods

A Matlab function was also used to determine the desired lengths of each of the rods based on the current triangle on the ground and the orientation of the robot towards the target location.

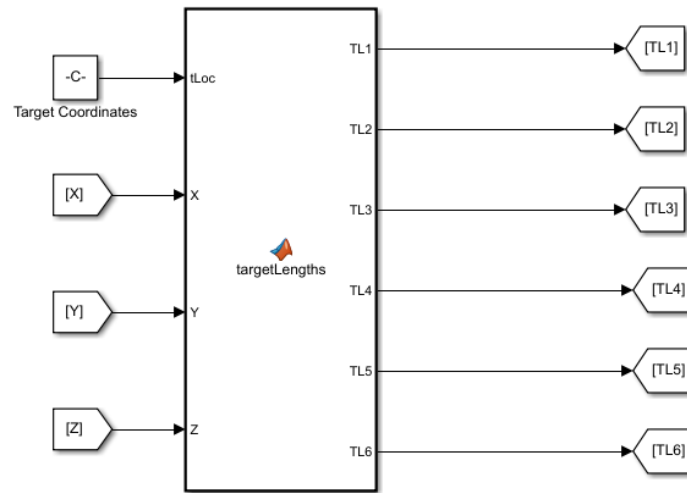


Figure 14: Simulink model using the rod end coordinates and target coordinates to determine the desired rod lengths

The ‘targetLengths’ function consists of using the location data to determine which three ends were in contact with the ground and which of those was furthest from the target location. The code of this function can be found in Appendix E. The three ends on the ground were determined by finding which ends had the lowest y-coordinate values. The x and z coordinates were used to find the distance each end had from the target x and z coordinates. This information was enough to tell which direction the robot had to roll and which rods had to be changed in length to do so. Once the robot reaches the

As mentioned earlier, there are two types of triangles formed by the rods in contact with the ground. A closed face is an equilateral triangle, therefore no matter which direction the robot needs to roll, a similar grouping or rods needs to be changed. Open face is an isosceles triangle, in this case, there are two different types of rolling, over one of the two the longer sides or the shorter side. However, in every case, the rods in contact with the ground would be adjusted. The furthest rod on the ground from the desired location would extend and the other two would contract. However, rolling over the shorter side of the open face triangle was the most challenging movement, requiring a total of five rods to adjust. This situation is best described visually with the following figure.

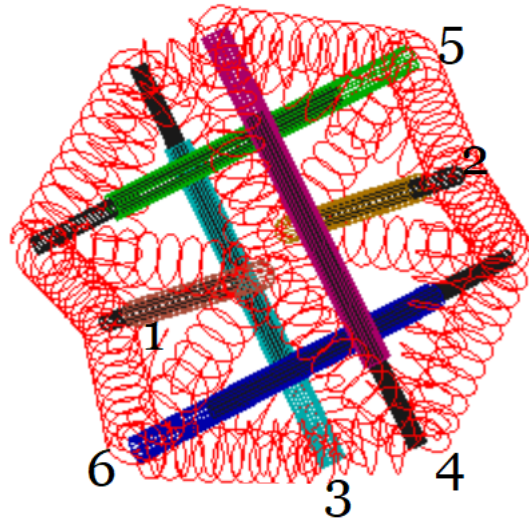


Figure 15: Tensegrity robot on open face triangle with numbered rods

In this figure, the tensegrity stands on rods 6, 3 and 4. For the tensegrity to roll over, the ends of rod 3 and 4 and subsequently stand on the other end of rod six. Rods 3 and 4 would have to contract and rod 6 would have to extend. In some cases, this was sufficient for rolling but if the robot already had momentum, more adjustments were needed. The two rods perpendicular to the desired direction, rods 1 and 2, were also adjusted. Rod 2 should be extended and rod 1 should contract.

The exact desired lengths were also determined through testing. The resting length is 1.2m, the starting length is 1.0m, extending length is 1.55m and the contracting length is 0.85m. It was not favourable to extend the rods to their maximum length of 1.6m as this resulted in instability.

Once operational length changes were selected, the model was tested and improved to ensure effective and controlled rolling. The Simulink block diagram was organized and any unnecessary calculations and blocks were removed. The Matlab function block was used as much as possible due to its flexibility and ability to reduce clutter on the block diagram.

The final result is a tensegrity robot that can roll towards any desired coordinates on a flat plane. Once the destination has been reached within 0.5m, the tensegrity robot adjusts all the rods to a length of 1.2m and stops moving. This means that it can follow any route input given to it in coordinate form. This is arguably a highly compatible model, with for example the problem described in the following paragraph.

An additional step was to find out if the model was able to avoid obstacles. Instead of rolling in a straight line towards the desired location, the robot would have to plan a route around obstacles, eventually reaching the target. The obstacle course has been designed in ADAMS, is shown in the following section and the coordinates of the obstacles are defined. Given that the system knows the coordinates of the obstacles in advance, the tensegrity can roll towards the target location until the distance to the object reaches a threshold and roll around it, maintaining a fixed distance from the obstacle. Alternatively, a path planning algorithm can be implemented

that determines a route around the obstacles to reach the target location. For this method, it is important to ensure the inputs are compatible with the algorithm and the algorithm provides compatible outputs for the tensegrity robot. The output would have to be checkpoint coordinates for the tensegrity to roll towards.

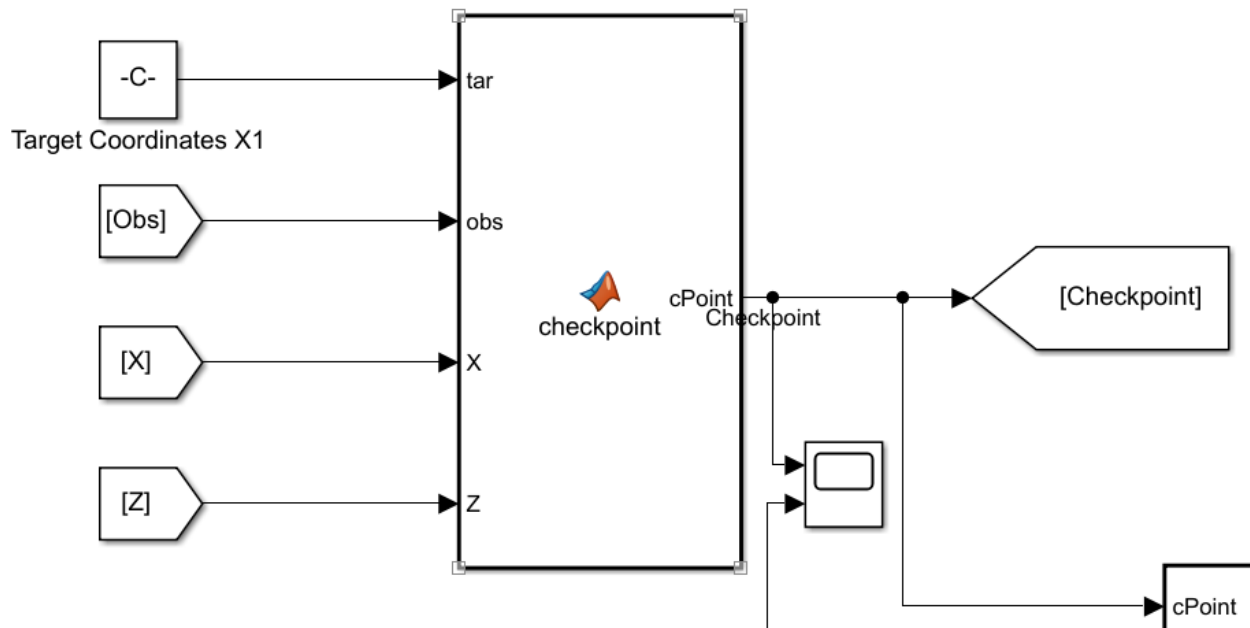


Figure 16: Simulink model section for determining the next checkpoint to roll tensegrity towards

The obstacle detection and avoidance function, named 'checkpoint', can be found in Appendix F. The function uses the target location, the known coordinates of all obstacles and the coordinates of the tensegrity robot as inputs. These inputs are used to produce checkpoint coordinates which the control system uses as a direction to roll towards. The function finds out which obstacles are between the tensegrity and the target, then it subsequently finds the nearest obstacle to the tensegrity and picks either the left or right side, 1.75m to either side, of that obstacle as the next checkpoint. This final decision is made by selecting the side which is nearest to the final target. The obstacles have a cuboid shape with a width and height of 1m and a thickness of 0.2m. Once the tensegrity is within 0.5m of the checkpoint, it moves on to the next checkpoint.

## Model Testing and Evaluation

For the iterative testing and improvement process, key performance indicators (KPI) were needed. Considerations for creating the KPIs included the fields that tensegrity robots are being developed for, the current capabilities of tensegrity robots, and potential implementations of tensegrity robots. For example, velocity should be maximized whilst considering safety. Safety can be ensured by making the acceleration as low and consistent as possible. The force due to impacts on the tensegrity should be minimal and quickly dissipate throughout the structure. This will help protect a payload. The performance of the robot in terms of the KPIs will provide crucial information on which aspects of the robot model need to be improved.

Firstly, iterative testing was performed for the drop test and manual control of the bars. The drop test was repeated every time the simulation started. The robot is placed 1 meter above the ground, falls and collides with the ground due to gravity. This test helps to ensure that the model is strong enough to handle large impacts. This testing procedure was used to ensure that the stiffness and damping coefficients of the springs were optimized to hold the structure together and dissipate the kinetic energy. (Goyal, R. et al 2019)

The force in each of the springs was measured by ADAMS and was displayed and analysed in graphs such as in Figure 17 below:

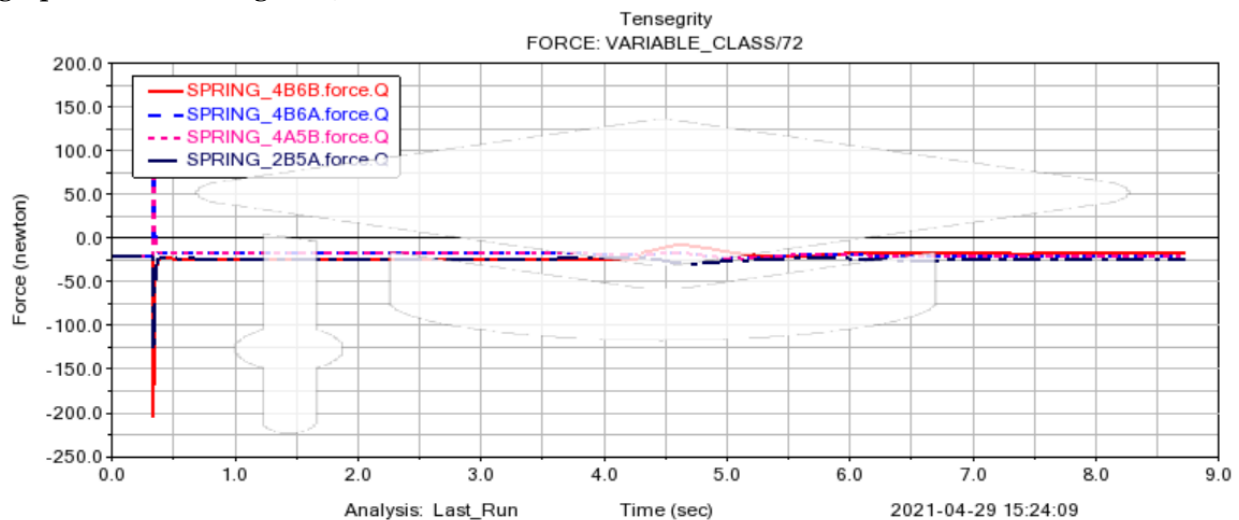


Figure 17: Force magnitudes applied to four selected springs of the tensegrity robot

The springs are labelled to refer to the rods that they are connected to. In this simulation, the robot was dropped from a 1m height above the ground. The collision with the ground can be observed to be just before the 0.5-second mark due to the large peak at that time. The maximum force applied to the springs in this simulation reaches 200N upon impact which is dampened very quickly due to the damping coefficient of the springs and the stable structure of the tensegrity. Appendix I shows the most recent iteration of the force simulation from a height of 3m.

The top view of the simulation environment is shown below.

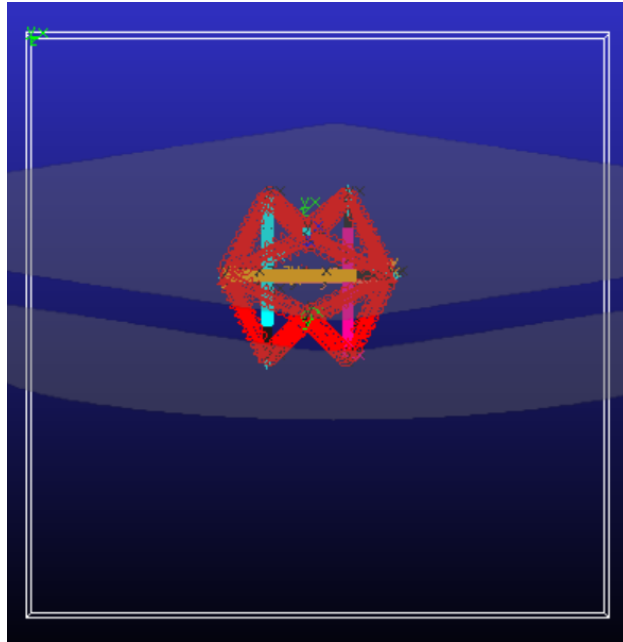


Figure 18: Top view of the ADAMS simulation environment with the ground surface of 4x4m

This surface of 4 by 4 meters has been utilized as the environment that the robot can roll across. Given a desired target location's coordinates, which was set using an initialisation function, the robot rolls towards that location. An example of such a simulation is shown using the following figures:

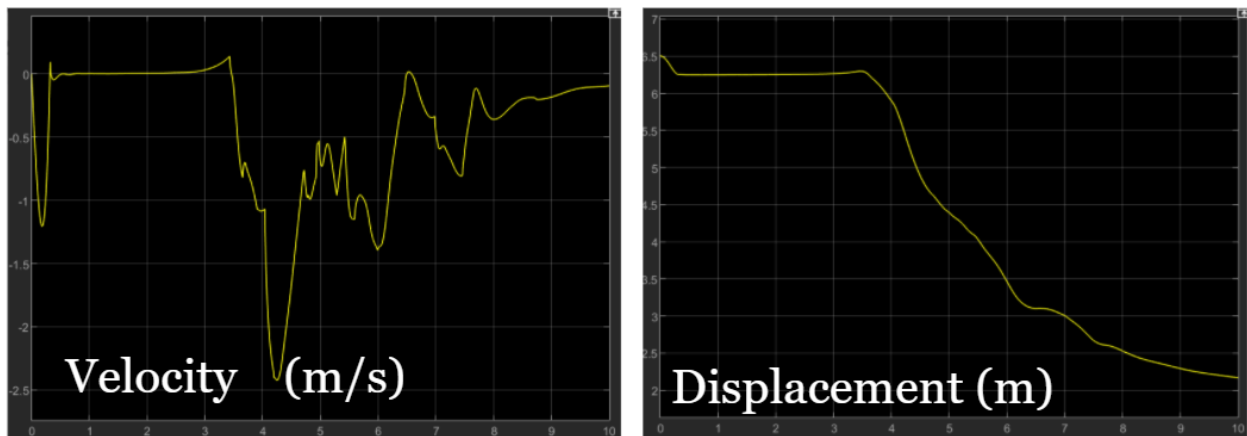


Figure 19: Velocity and displacement of tensegrity robot relative to target coordinates [0, 0.5, 3]

For the first second of simulation, the robot drops onto two rod endings, after four seconds the robot has rolled onto the third rod end. At this point, the robot starts to adjust the rod lengths to start rolling which is confirmed by the reduction in displacement. Velocity ranges from 0 to 2.5 m/s towards the target location. Once the robot has started rolling there is never an instance when the robot rolls in the wrong direction.



The related rod length changes are displayed in the following figure:

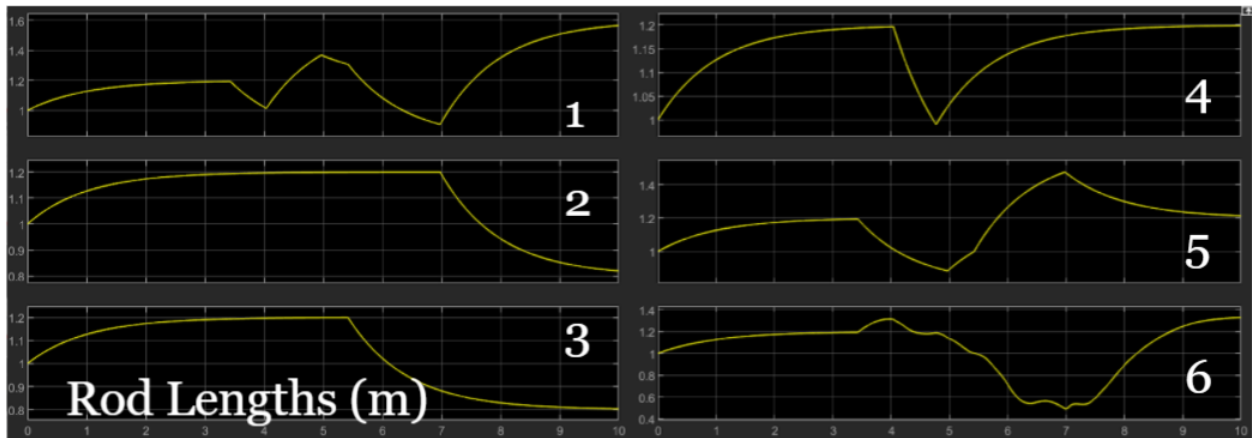


Figure 20: Lengths of the six tensegrity rods for simulation of rolling towards target coordinates [0, 0.5, 3]

At the start of every simulation, all the rod lengths are changed to 1.2. The first rod length changes for rolling occur at the 3.5-second mark. Rods 1 and 5 are contracted whilst rod 6 extends, resulting in a roll in the desired direction. However, the behaviour shown by rod 6 is irregular, which had to be fixed by correcting the physical model in ADAMS. The coordinates were not correctly measured, resulting in an incorrect relationship between desired length and actual length. This issue was fixed by recalibrating the rod measurement. This change resulted in the following improved figure:

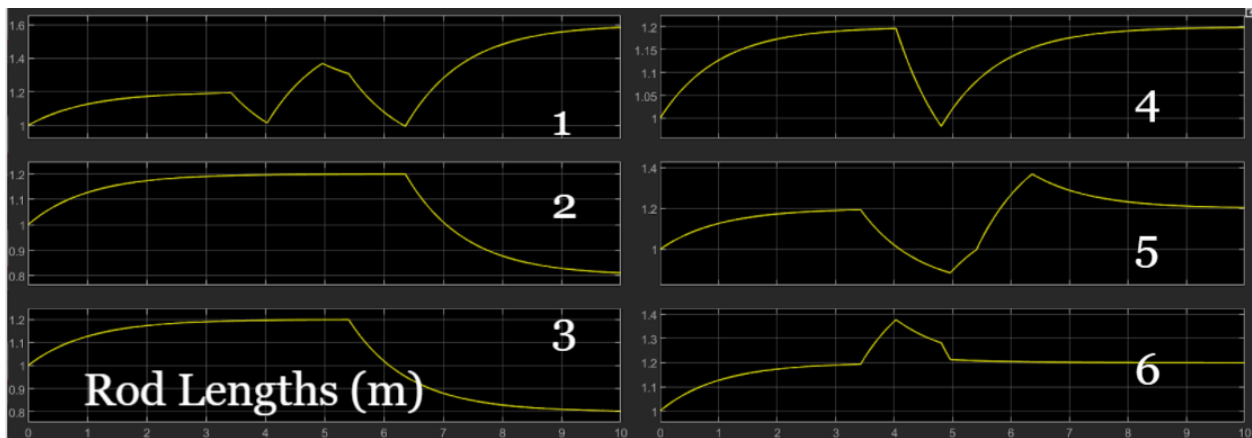


Figure 21: Improved lengths of the six tensegrity rods for simulation of rolling towards target coordinates [0, 0.5, 3]

As can be seen, the behaviour of rod six is now far more regular. As for the velocity and displacement, these had also become slightly more controlled.

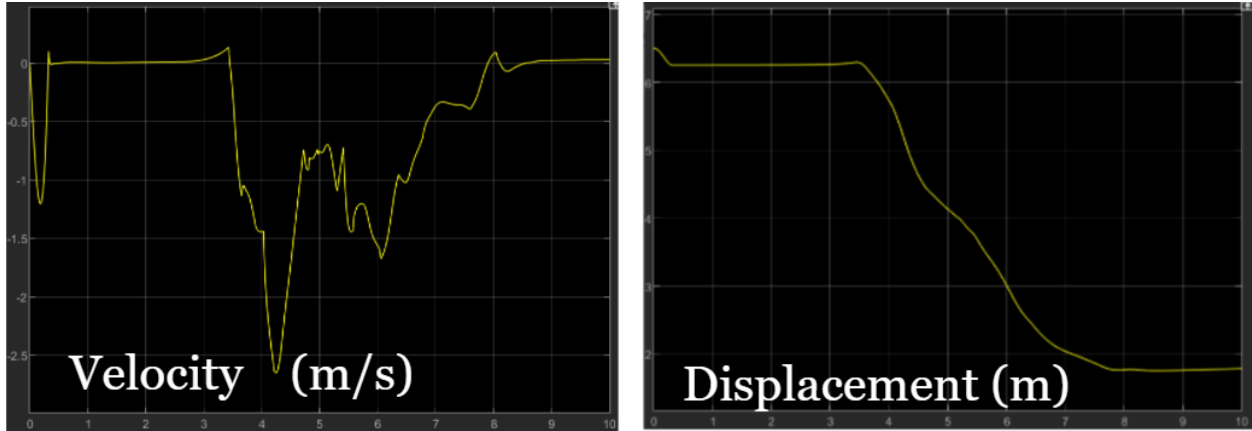


Figure 22: Post-improvement velocity and displacement of tensegrity robot relative to target coordinates [0, 0.5, 3]

This iterative improvement method continued. Resulting in changes to the desired rod lengths for each possible orientation and desired direction. Various starting orientations and target coordinates were implemented to ensure rigorous testing.

Later linear rolling simulation results are shown by the displacement and acceleration in the following figure:

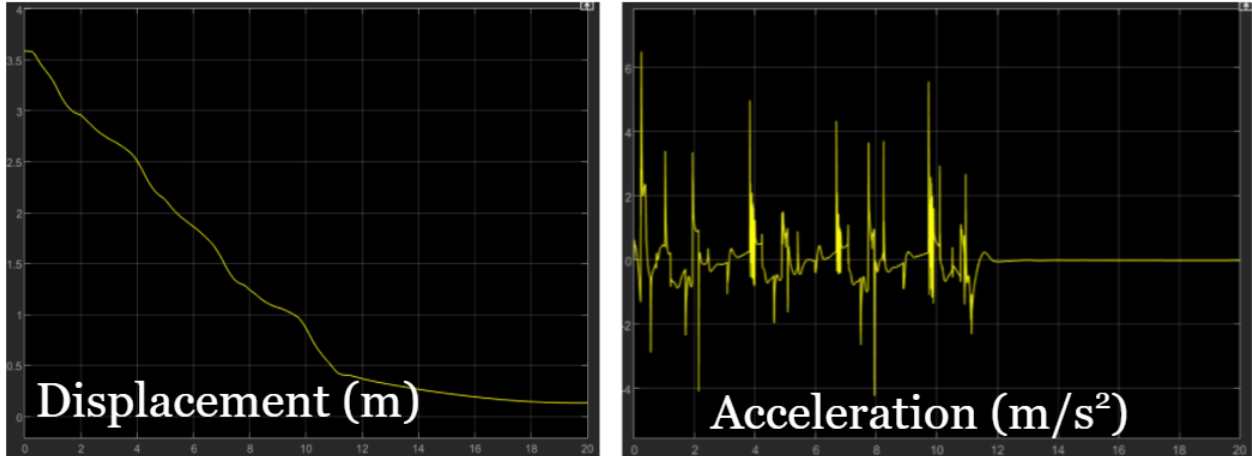


Figure 23: Displacement and acceleration of tensegrity robot of simulation 20/05/2020 13:00

In this simulation, the robot had to travel a distance of 3.5m, within 11 seconds the distance covered was 3m. This means that the average velocity was just above 0.27m/s. The robot has an average diameter of 1.2m, this ratio of diameter to velocity is low relative to the speed of the machine learning method with 15cm/s for a 13cm diameter robot. (Rieffel, J. et al 2018) Further improvement to this model would be necessary to come closer to their achievement. As for the acceleration, the absolute maximum reached is 6.5m/s<sup>2</sup>. This can be related to the gravitational constant of 9.81m/s<sup>2</sup> as a reference to the impact on the payload that might be carried by the tensegrity. Therefore, an aim should be to minimize this absolute value, even though currently this is an acceptable value.

Once the control system was sufficient enough to reach any location on the 4x4m surface, the next step was to provide the robot with obstacles and ensure it was able to plan a route around them. The 4x4m surface was expanded to a 10x10 area and two obstacles were added. The new environment is shown below.

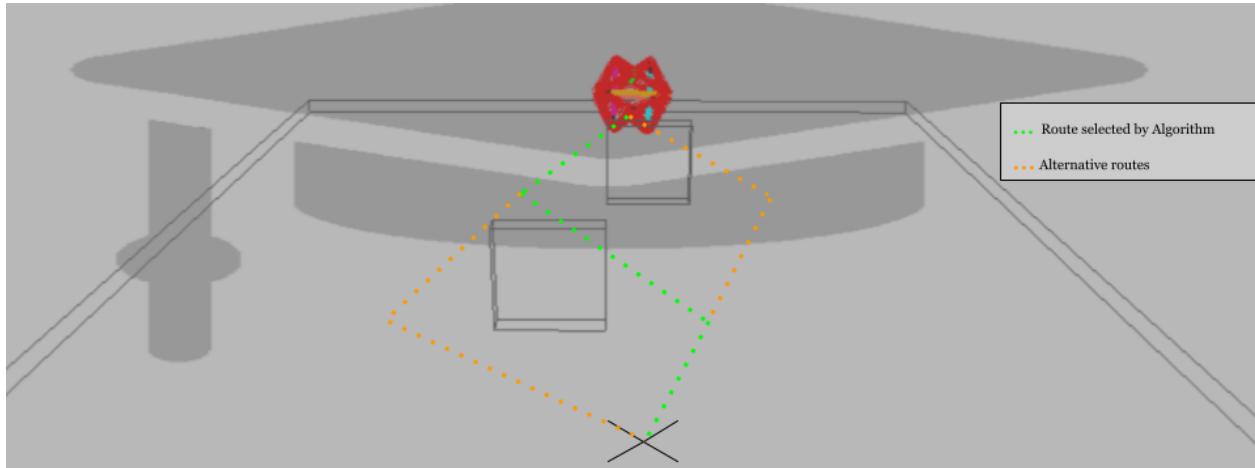


Figure 24: Tensegrity robot testing environment with obstacles and possible routes towards target objective

In this scenario, the tensegrity robot is expected to roll around both obstacles and reach the bottom of the figure. The code that is used to detect and avoid these obstacles can be found in Appendix F. In the most recent simulation of the scenario, it took the tensegrity robot around 10 seconds to pass the first obstacle. This occasion is shown in the figure below.

Based on the logic of the code, the robot is expected to roll to the other side of the following obstacle and continue to the target coordinates. This is indeed accomplished within a minute of simulation time with an average velocity of 0.2m/s. This code may need more improvement for robustness, although it did meet the requirements initially set for it. The simulation is described alongside the graphs on the following pages.

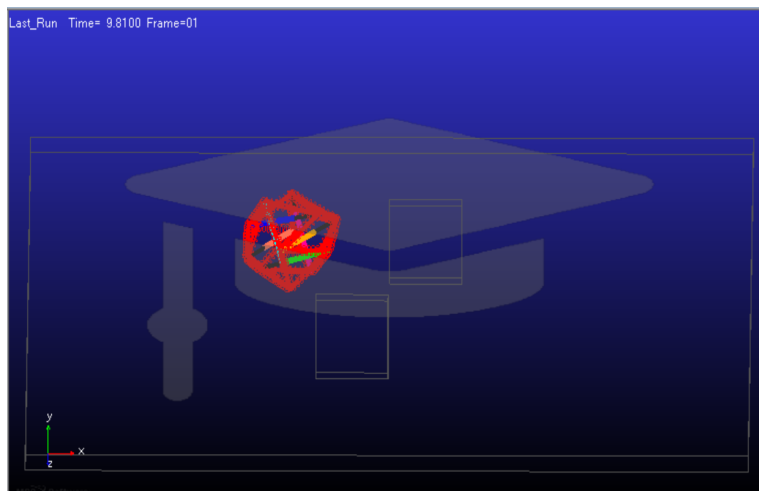


Figure 25: Tensegrity robot passing the first obstacle after 9 seconds, using the current obstacle detection and avoidance function

The final target coordinates are (7.5, 0.5, 0). Starting from (0, 0.5, 0), the final destination is reached after 27 seconds. The first graph to observe is the displacement, velocity and acceleration of the tensegrity robot in relation to the current checkpoint.

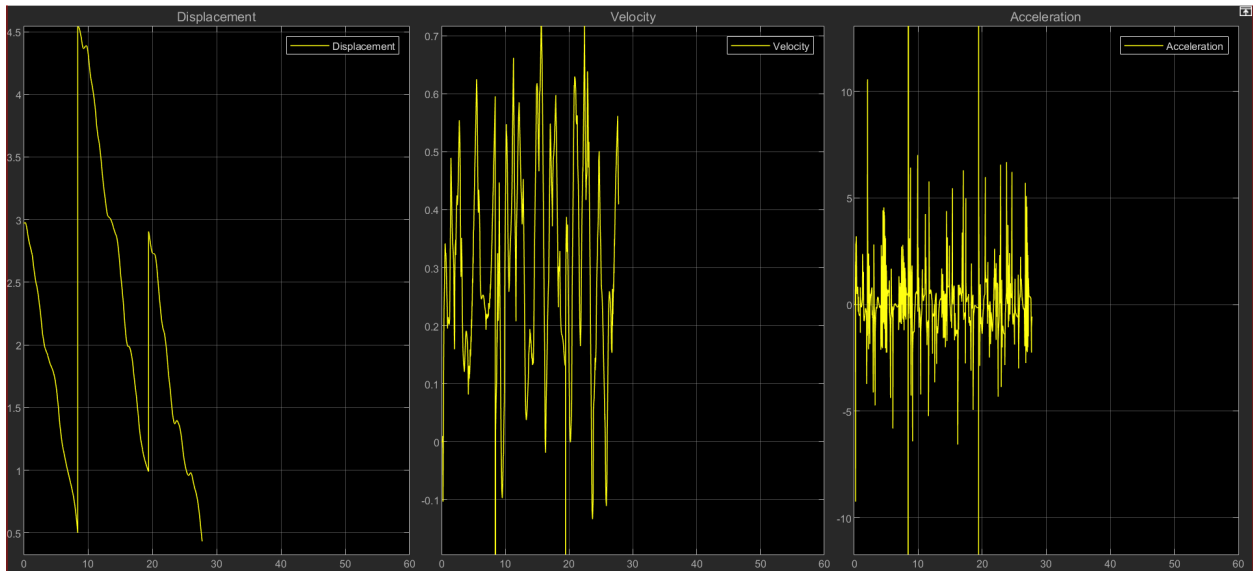


Figure 26: Displacement (m), velocity (m/s) and Acceleration (m/s<sup>2</sup>) in the latest simulation of the tensegrity robot avoiding two obstacles

Firstly, the peaks in the displacement graph represent a change in the checkpoint. The robot rolls towards each checkpoint and once it reaches a 0.5m distance to the checkpoint the next checkpoint is targeted which increases the displacement from the desired goals. Every time this change occurs there are spikes in the velocity and acceleration graphs. These occur at the 9 and 19-second marks. The velocity of the robot averages at about 0.275m/s which can be derived from the slope of the displacement graph or estimated from the velocity graph. The velocity graph shows heavy fluctuation between 0 and 0.7m/s. Given the motion of the tensegrity robot, this is fitting. The acceleration graph shows that 6m/s<sup>2</sup> and -6m/s<sup>2</sup> are maximum and minimum accelerations experienced by the robot and any package it may be carrying. Using gravitational acceleration on earth as a frame of reference helps to understand what the robot or package might be experiencing. Seeing as a value of 9.81m/s<sup>2</sup> is never exceeded, there should be no harm caused to the package.

The following graph used for the tests and simulations is the rod end heights and is shown in the following figure.

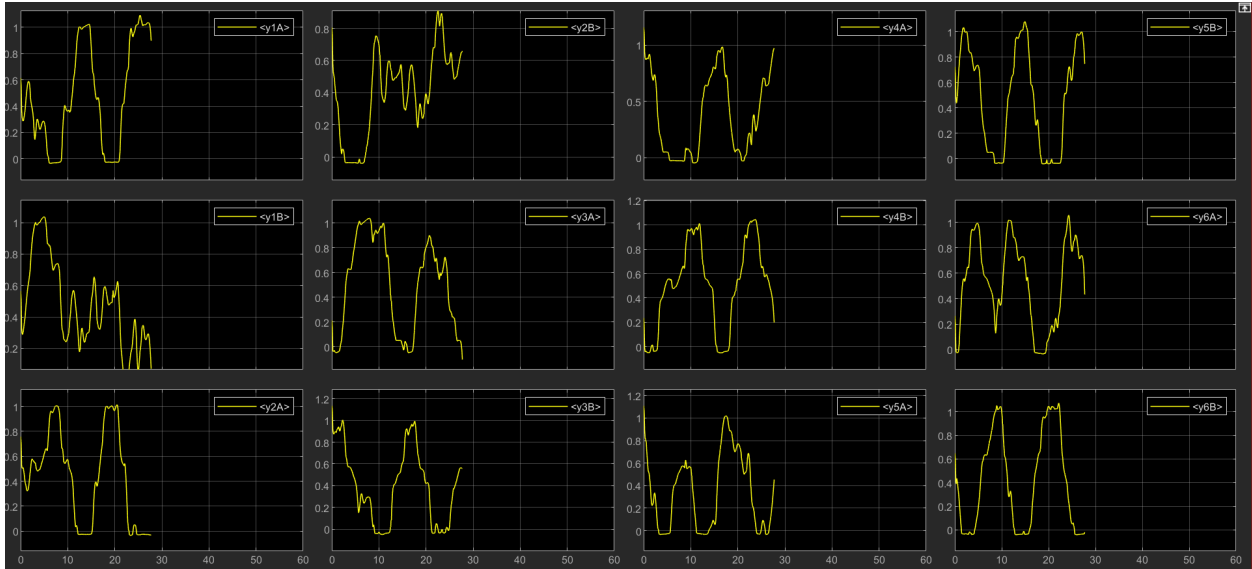


Figure 27: Rod end heights (m) in the latest simulation of the tensegrity robot avoiding two obstacles

This graph shows which rod is in contact with the ground at any point. Smooth sinusoidal curves are desired for these graphs to ensure regular, controlled movement. As the project went on and improvements were made to the model, the rod end heights graphs became smoother and smoother. However, it is not yet perfect leaving room for improvement as can be seen in the occasional sudden changes in height. This final result is sufficient.

The movement of the tensegrity robot compared to the checkpoints can also be tracked in terms of the x, y and z coordinates. A graph for this was also produced during simulation in Simulink.

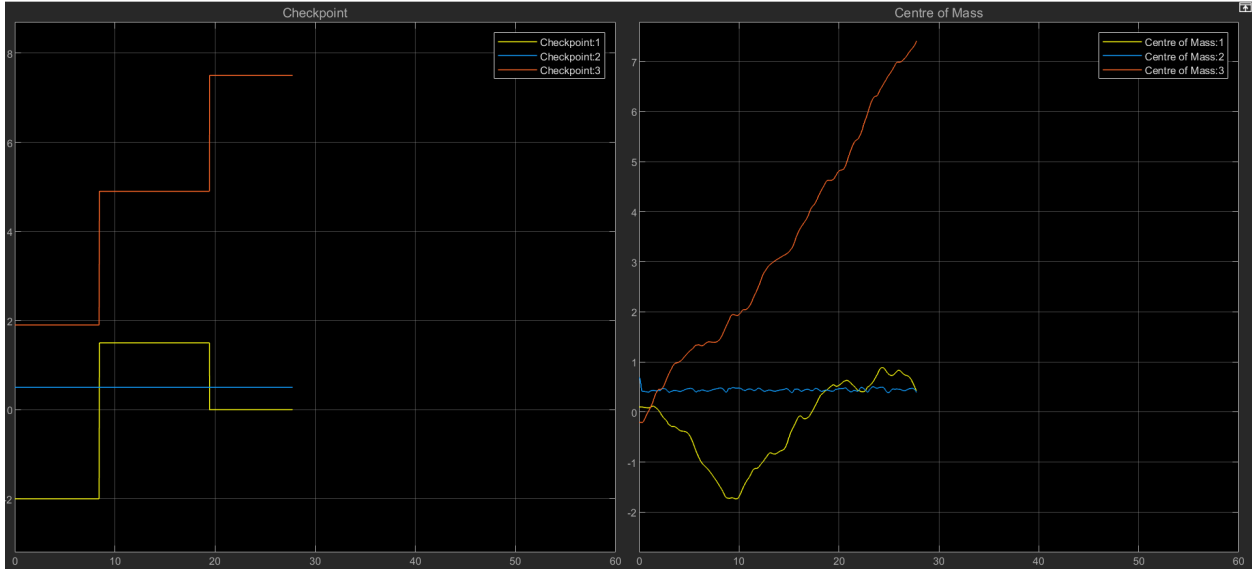


Figure 28: Checkpoint coordinates (m) and current coordinates (m) in the latest simulation of the tensegrity robot avoiding two obstacles

The graph on the left shows the current coordinates of the checkpoint that changes when the checkpoint has been reached. The graph on the right shows the current coordinates of the tensegrity. On both graphs the yellow line represents the x value, the blue line represents the y-value and the red line represents the z value. The final target coordinates are (7.5, 0.5, 0), the graph on the right shows that the tensegrity was always heading in the positive z-direction but had to move in the negative x-direction to avoid the obstacles.

The final graph is the lengths of the rods at any point. As can be seen in the graph below the lengths range between 0.85m and 1.55m as expected. There is however a variation in large length changes and smaller movements. Some movements may be unnecessary for the desired rotation of the robot but are caused by the logic of the code. A further improvement would require that any unnecessary rod adjustments are removed to reduce energy usage by the robot.

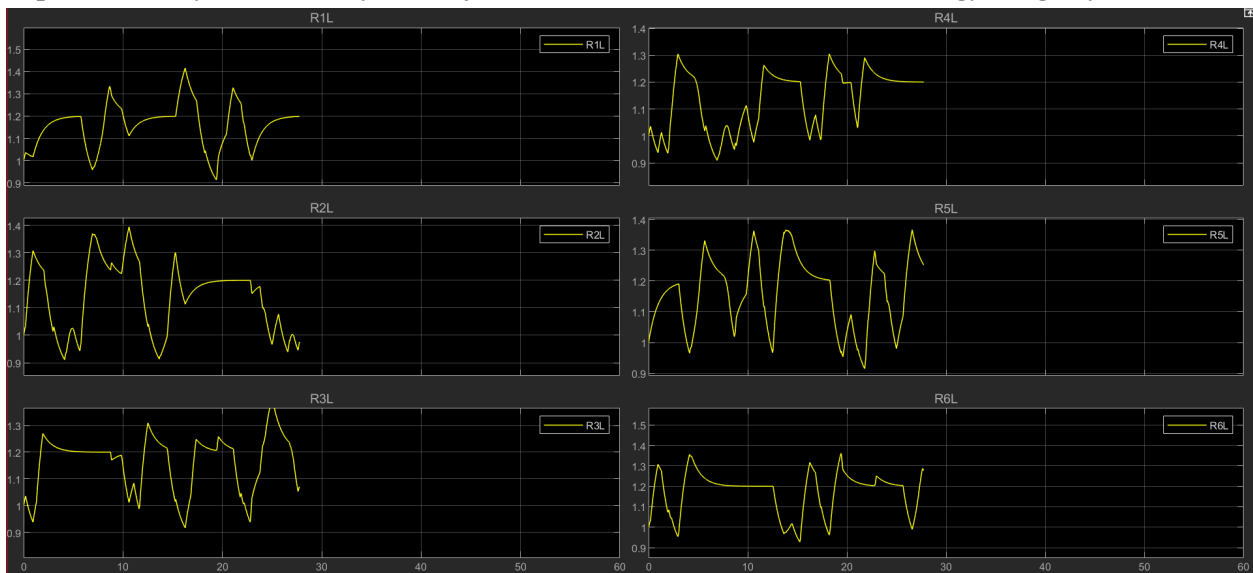


Figure 29: Rod lengths (m) in the latest simulation of the tensegrity robot avoiding two obstacles

The graphs of a previous, unsuccessful simulation with the same directions can be seen in Appendix H.

The following possible steps, should this project be continued, would be programming locomotion up and down slopes. Currently, the model detects the rods on the ground by their y coordinates. If the robot is at a different elevation or stands at an angle, the function for determining the desired rod lengths would not work.

However, The robot is successfully able to roll to any desired location within the simulation environment. Every orientation is accounted for and the system is compatible with any algorithm or code that outputs desired coordinates. The strengths and limitations of the project, Simulink block diagram and functions are discussed in the following section.

# Discussion and Optimizations

As can be seen from the previous sections, this project requires many steps that vary from being quick, easy, time-consuming and challenging. This is why optimizing the process is important, both for the outcome of this project as well as for researchers looking into developing their tensegrity robot.

## Lean

Lean manufacturing is a concept first developed by Toyota's manufacturing operations. One concept from Lean is the three M's, schematically shown in Figure 30. Muri, defined as overburdened; Where has the workload been too high, or where is there an overload of calculations or data? Mura, defined as unevenness; Are there any imbalances in the workflow or model? Muda, defined as waste; Where are unnecessary steps taken or calculations made that can be removed? (Marchwinski, C., et al. 2014)

This concept can be applied to the creation process of the tensegrity robot and will help find optimizations for the procedure and design.

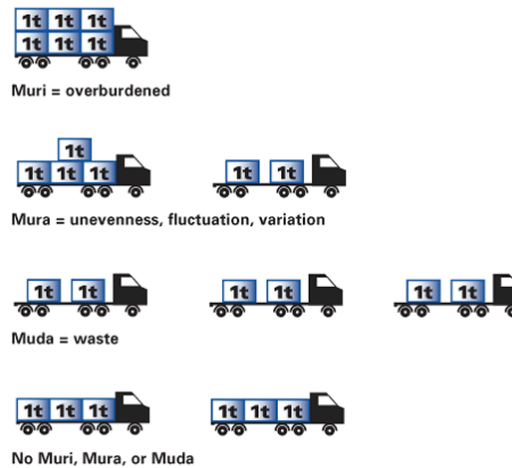


Figure 30: A schematic of the three M's of lean manufacturing (Marchwinski, C., et al. 2014)

Designing the robot in ADAMS has been a time-consuming yet easy task. A large amount of time was spent creating each component of the robot. As the design of the tensegrity becomes more widespread perhaps there will be templates available online to test and adapt for personal use. Before this happens, however, developments in the control and uses of the robots need to be made for widespread use to become more attractive.

Another time-consuming part was setting all the parameter values for the springs and friction values, and having to individually change these every time a new value needed to be tested. To improve this stage an optimization was found which was to set a single variable value and make all relevant parameters equal to that variable. This ensured that if a new parameter value needed to be tested, only one variable had to be changed. This solution can be applied to many other values of the repetitive tensegrity robot design.

Next, the best values for the stiffness, damping and preload of the springs were found experimentally. This method can be improved with a mathematical relationship between the size, starting shape of the structure, and spring values. This relationship can then be used by researchers hoping to design their tensegrity structure.

Selecting the right inputs and outputs for the control system is highly dependent on the researcher and their preferences. Arguably, the method implemented in this report leading to the selection of output rod velocity was quite successful. Visualizing the product in the real world and evaluating the behaviour of the actuators helps to find what information is most compatible with the system. If perhaps a hydraulic actuator had been visualised, then the outcome variable would have been the force instead of velocity.

The desired rod lengths are found using twenty-two separate if statements that check for every possible orientation. Twenty of the if statements are for the twenty faces of the icosahedron tensegrity shape. Then there is a statement for when the robot is transitioning between faces and standing on two legs. Lastly, there is a statement for when the robot is close enough to the desired position to stop moving. Without this final statement, the robot will reach a point where two orientations are closest and fluctuate between them. Consideration for a loop was made to reduce the length of the code, however, the use of 22 if statements results in a relatively short code. The orientations are specific, therefore, a loop may not be as compatible as the currently used if statements.

The improvements and optimizations for the control system should be quite straightforward, the model must be cleaned up and any unnecessary calculations can be removed. At every stage of designing the control system, it should be analysed holistically in case some shortcut calculations can be found. Currently, a weak point with the control system is a large number of input values (36 coordinate values), this perhaps could be reduced by calculating the necessary values in ADAMS instead of in Simulink. At this stage, the robot can move linearly at an average velocity slightly below 0.3m/s. With the right adjustments to the rod length changes, this velocity can be increased.

Simulink is a block diagram simulation tool. Developing a system for the tensegrity robot to derive desired rod lengths was not a straightforward task, especially not with the sole use of blocks. The best solution that was found for this was a specific block named 'Matlab function'. This block allowed for manual coding of an individual block. For example, the blocks used to find the actual length of the rods and the block used to conditionally derive the target rod lengths were Matlab function blocks. The discovery of this block type helped overcome a large hurdle and reduce the overall size and complexity of the Simulink block diagram. A noticeable trend in this project was the increased use of Matlab function blocks and the simplicity of the diagram.

The simulation is a relatively demanding process for hardware. For efficient development and improvement of the model, an Intel i7 processor would be recommended in combination with at least 8GB of RAM. Anything below that results in limiting simulation times of 10 mins for a 20-second simulation. To overcome this issue, the ADAMS plugin provides the option to either



display a live simulation or only calculate values. The latter is less demanding and halves simulation time. Therefore, if the hardware in use does not meet the recommended standard, multiple scopes can be placed in the block diagram to analyse the behaviour of the model. The use of these scopes is recommended regardless since quick simulations and analyses help to discover and solve smaller issues. An example of a useful scope is the height of each of the rod ends. This helped to visualize which rod ends were forming a triangle. The scope is shown in the figure below.

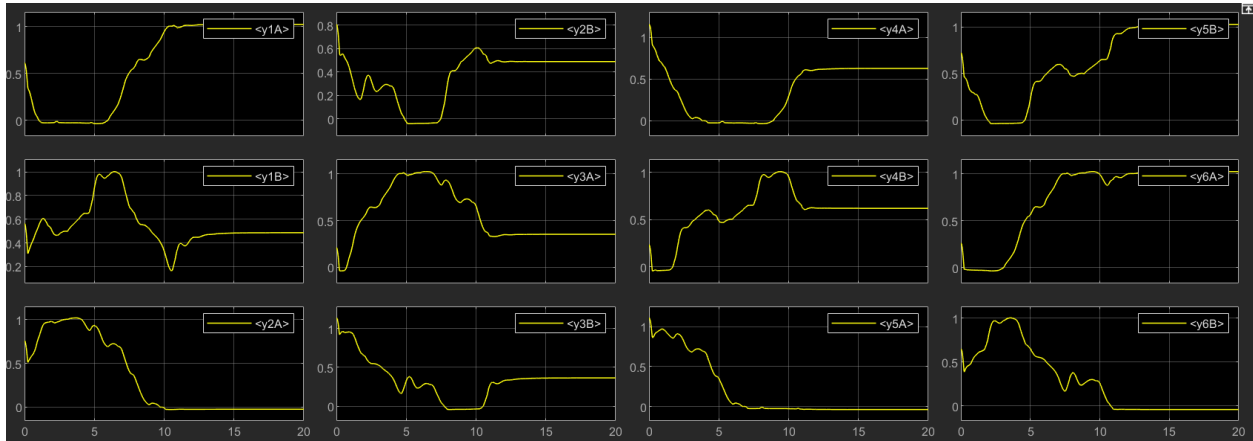


Figure 31: Rod end heights (m) of tensegrity robot simulation 20/05/2020 13:00

In regards to the obstacle detection and avoidance algorithm, there are limitations. The code requires that the coordinates of the obstacles are known. It isn't possible to derive these without a sensor. Therefore, the coordinates of the obstacles were selected as outputs from ADAMS and used for the Simulink function. Additionally, the function assumes that all obstacles are similarly shaped with a width of 1m and are relatively thin. Varying shapes and sizes would require a much longer code and far more ADAMS outputs to visualize the obstacles. The function also is at risk of not selecting the most optimal route because the only consideration it makes for the next checkpoint is the nearest obstacle between the tensegrity and the target. This function provides a clear basis for improvement, although it works given the current scenario. It would also work if more obstacles of the same size were placed because it ignores all obstacles that are not in the way. To further improve this function, a path planning algorithm could be applied. For example, the path can be iteratively improved as time passes with the use of model predictive control (Borrelli, F. 2016), to ensure the overall route taken is indeed the shortest.

Developing evaluation methods for the tensegrity robot requires both research and stakeholder analysis to determine the most important qualities of a tensegrity robot. The stakeholders help discover what the expectations of the product are and what capabilities it must possess. The research helps to find comparable products and evaluate what they perform well at. This helps to create targets for the design. It helps to take a holistic approach when evaluating the simulations and respective graphs. Finding interconnections and root causes takes critical thinking and often it turns out that multiple issues are solved by making a single small change. For example, testing out a new maximum rod length or adjusting the values of the PID controller can lead to a much higher velocity but may stop the tensegrity from rolling in some orientations.

In summary, at each stage, it is important to focus on reducing waste. Finding out which long tasks can be made shorter and which difficult tasks can be made easier. This ensures that more time can be spent on the more difficult tasks and a clearer path can be made for preceding research.

## Conclusion

In conclusion, the original goal, which was to design a tensegrity robot and make it roll in the desired direction, has been met. This task was completed at the preliminary stage near the end of May. As an additional task, the robot would have to automatically avoid obstacles to reach a final desired destination. In the final weeks of the project, the goals set for this task were also met and the robot was able to move a 7-meter distance while avoiding two obstacles in 27 seconds. This simulation has been recorded and the video should be available together with this report and the project files.

This project aims to optimize the design stages of a rolling tensegrity robot model. The final deliverables will be a report that discusses the method, explains each of the improvements and optimizations that have been and need to be made, and addresses the research questions. The report is supported by the design files, control files, and video recordings of the simulations. The literature, system analysis, and planning have been curated to challenge the student as well as create a rigorously tested tensegrity robot. The main research question asks for fully optimized creation, control, and evaluation stages. Therefore the final deliverable can be critiqued on whether any of the stages still have room for improvement.

In the stakeholders and system analysis, a desire was expressed to make external stakeholders interested in the findings. Rather than merely creating a model, by further optimizing the model and discussing the improvements, the stakeholders can apply similar optimizations to their research. This project should also benefit the research of supervisor Zhiyuan similarly. The goals of this project, set by the student and supervisor, have been completed successfully and the findings could be of benefit to soft robot researchers.

Even at this stage, there are improvements to be made to the model. Firstly, different desired rod lengths could be even further tested to improve velocity and control. Secondly, the Simulink block diagram, which can be seen in Appendix B, can always be cleaned up and made more efficient. Thirdly, the obstacle detection and avoidance function can be further improved for better obstacle detection and optimal route planning.

Zhiyuan Liu, the supervisor of this project, has been furthering his work on tensegrity structures alongside this project. For example, the combination of drones and tensegrity robots, a combination that works well because of its ability to move swiftly in the air, as well as protect and reorient itself when landing on the ground. The work described in this report should hopefully help find new perspectives and solutions for similar further study.

# Bibliography

Bagheri, H., Ahmed, R., Skowronek, E. O., Marvi, H. (2020) Materials, Actuators, and Sensors for Soft Bioinspired Robots., Advanced Materials, Wiley 2020-12-21, 0935-9648

Borrelli, F., Bemporad, A., Morari, M. (2016) Predictive Control for linear and hybrid systems.

Bruce, J. (2016) Design, building, testing, and control of superball: a tensegrity robot to enable new forms of planetary exploration. dissertation. University of California, Santa Cruz.

Caluwaerts K., Despraz J., Işçen A., Sabelhaus A. P., Bruce J., Schrauwen B. and SunSpiral Vytas. (2014). Design and control of compliant tensegrity robots through simulation and hardware validation. Soc. Interface.1120140520. <http://doi.org/10.1098/rsif.2014.0520>

Connelly R. (2002) Tensegrity Structures: Why are They Stable? Rigidity Theory and Applications (pp.47-54), DOI: 10.1007/0-306-47089-6\_3

Connelly R. (2013). Tensegrities and Global Rigidity. In: Senechal M. (eds) Shaping Space. Springer, New York, NY. [https://doi.org/10.1007/978-0-387-92714-5\\_21](https://doi.org/10.1007/978-0-387-92714-5_21)

Douglas, B. (2018) 'Understanding PID control: What is PID control?' MATLAB Tech Talks. Mathworks, Inc. Video lecture. <https://youtu.be/wkfEZmsQqiA>

Engineering ToolBox, (2004). Friction and Friction Coefficients. [online] Available at: [https://www.engineeringtoolbox.com/friction-coefficients-d\\_778.html](https://www.engineeringtoolbox.com/friction-coefficients-d_778.html)

Fivat S., Rieffel J., Paul C., Valero Cuevas F., Lipson H. (2010). Tensegrity Robots. Creative Machines Lab. Columbia University. Web. <https://www.creativemachineslab.com/tensegrity.html>

Goyal, R. and Skelton, R. E. (2019) Tensegrity system dynamics with rigid bars and massive strings. Multibody Syst Dyn 46, 203–228. <https://doi.org/10.1007/s11044-019-09666-4>

Kim, K. Moon, D. Bin, J. Y. and Agogino, A. M. (2017) "Design of a spherical tensegrity robot for dynamic locomotion," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) pp. 450-455, doi: 10.1109/IROS.2017.8202192.

Lenarčič, J. and Parenti-Castelli, V. (2019) International Symposium on Advances in Robot Kinematics. Springer (Springer proceedings in advanced robotics, 8). doi: 10.1007/978-3-319-93188-3.

Liu, Z. (2021). Tensegrity Robot Design and Control Project Description and Hints. Industrial Engineering and Management Bachelor Integration Project. University of Groningen.

Luo, A. Xin, H. Cao, P. Hao, X. Yu, Y. Sun, P. Tian, W. (2016) "Motion simulation of six-bar tensegrity robot based on ADAMS," 2016 IEEE International Conference on Mechatronics and Automation, 2016, pp. 264-269, doi: 10.1109/ICMA.2016.7558572.

Marchwinski, C., & Lean Enterprise Institute. (2014). Lean lexicon: A graphical glossary for lean thinkers. Brookline, MA: Lean Enterprise Institute.

Mathworks. (2021). Simulation and Model-Based Design. Matlab & Simulink. Web. <https://nl.mathworks.com/products/simulink.html>

MSC Software. (2021). ADAMS -The Multibody Dynamics Simulation Solution. Hexagon. Tutorials and Manuals. Web. <https://www.mscsoftware.com/product/ADAMS>

Norton, R. (2020) ADAMS Tutorial Kit for Mechanical Engineering Courses. Design of Machinery. Web. <https://www.mscsoftware.com/page/ADAMS-tutorial-kit-mechanical-engineering-courses>

Rhodes, T., Gotberg, C., & Vikas, V. (2019). Compact shape morphing tensegrity robots capable of locomotion. *Frontiers in Robotics and Ai*, 6, 111–111. <https://doi.org/10.3389/frobt.2019.00111>

Rieffel, J., Mouret, J. (2018) Adaptive and Resilient Soft Tensegrity Robots. *Soft Robotics*. DOI: 10.1089/soro.2017.0066

Chopra, S., & Meindl, P. (2016). Supply Chain Management: Strategy, Planning, and Operation. (6th ed.) Pearson Education.

Vespignani, M., Sunspiral, V., Bruce, J., Friesen, J. M., & 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems. (2018). Design of superball v2, a compliant tensegrity robot for absorbing large impacts. *Ieee International Conference on Intelligent Robots and Systems*, 2865-2871, 2865–2871. <https://doi.org/10.1109/IROS.2018.8594374>

Wang, M., Xu, X., Luo, Y. (2020). A General Model for Both Shape Control and Locomotion Control of Tensegrity Systems, *FRONTIERS IN BUILT ENVIRONMENT* 6.

Wang, R., Goyal, R., Chakravorty, S., & Skelton, R. E. (2020). Model and data based approaches to the control of tensegrity robots. *Ieee Robotics and Automation Letters*, 5(3), 3846–3853. <https://doi.org/10.1109/LRA.2020.2979891>

Zheng, Y., Cai, H., Wang, M., Yao, J., Xu, X., Zhou, C., & Luo, Y. (2020). Rolling gaits of a strut-actuated six-strut spherical tensegrity. *International Journal of Advanced Robotic Systems*, 17(5), 172988142096090–172988142096090. <https://doi.org/10.1177/1729881420960904>

Zheng, Y., Li, Y., Lu, Y., Wang, M., Xu, X., Zhou, C., & Luo, Y. (2021) "Robustness Evaluation for Rolling Gaits of a Six-Strut Tensegrity Robot," *International Journal of Advanced Robotic Systems*, 18(1), pp. 172988142199363–172988142199363. doi: 10.1177/1729881421993638

# Appendix

## A. Project description by supervisor

### Tensegrity Robot Design and Control

Research group: Prof. Ming Cao, Zhiyuan Liu

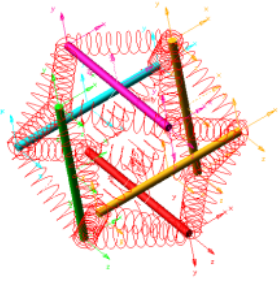

**Project description:**

Tensegrity structures are lightweight, resistant to impacts and possible to control their stiffness and shapes. A well designed and widely used tensegrity framework is six-bar-twenty-four-cable structure. By controlling lengths of bars or cables, tensegrity robots could change one-dimensional actions into rolling movements. Learning and controlling tensegrity robots could bring about fresh information about soft robots, as well as solid knowledge on control theories and methods.

**Subject of design:**

This project is designed for learning a new type of robot --- tensegrity robot and carry out simple control strategies to make it move. Build your own tensegrity robot in Adams and then manipulate it from Simulink. Add in more details to make the simulation realistic and challenging.

**Field of knowledge:** control theory, graph theory





++

Figure A1: Tensegrity project description

## B. Basic Simulink control model of a tensegrity robot and actual final designs

A basic control system is shown here where the desired state is translated into a change in structure, implemented by the robot that outputs its current state.

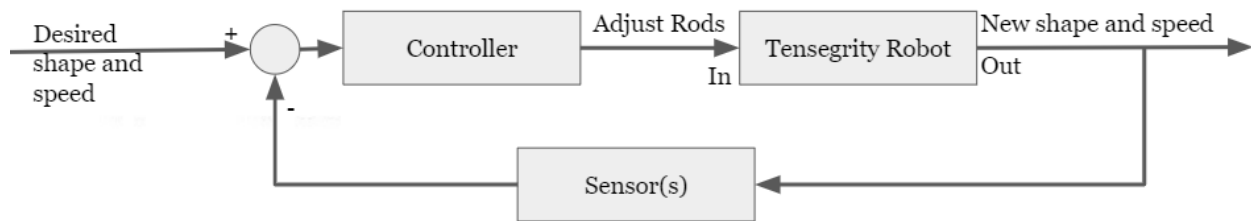


Figure B1: Original basic schematic of Simulink control model

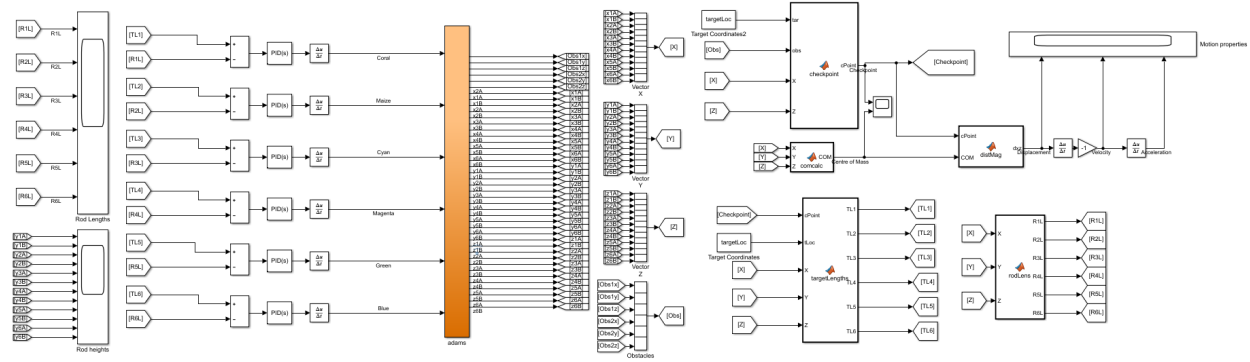


Figure B2: Final version on 4/6/2021

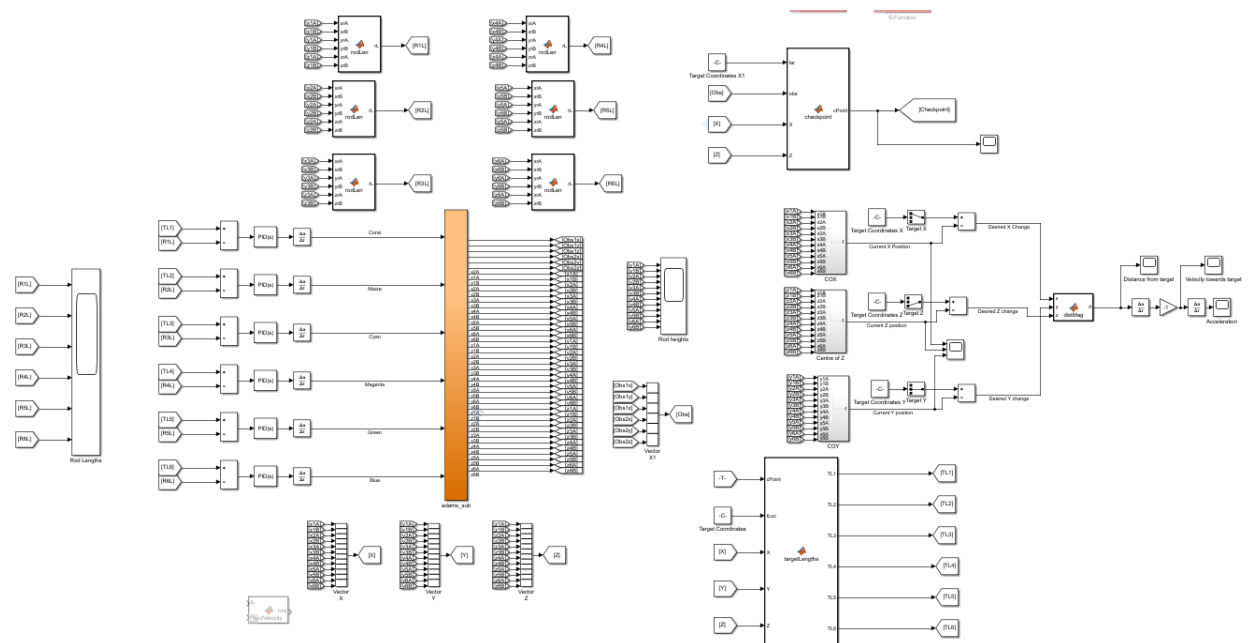


Figure B3: Preliminary version on 21/05/2021

### C. Sub goals set by student and supervisor in the first meeting

To learn about: Basic control algorithms, closed-loop systems, basic formation control algorithms, tensegrity frameworks and tensegrity robots, ADAMS and its real-time simulation along with Simulink.

To discover: How to build mechanical structures in ADAMS, how to construct closed-loop control systems in Simulink, how to stabilize closed-loop systems and further improve performance, how to make a tensegrity robot change its shape, how to decide which bars to manipulate and what the optimal length change is.

To build: A tensegrity robot and corresponding simulation environment ADAMS and a bar-length control system in Matlab and Simulink.

To test: Make the tensegrity robot roll to a specific target via manual or automatic control of its shape then evaluate performance based on KPIs.

#### **D. Additional Questions**

Background knowledge:

What are tensegrity structures and robots?

What tasks would this kind of robot be able to do?

What would a tensegrity robot be able to do better than other robots?

What are the advantages of using ADAMS and Matlab for this project?

What does control theory and graph theory have to do with designing and operating a Tensegrity robot?

Physics, maths and programming knowledge:

What calculations should be done to find the optimal tensegrity robot shape change for each desired movement?

What should be coded in Matlab to ensure a direct translation from calculation to bar length change?

What evaluation KPIs should be used to improve the properties of the tensegrity robot ( e.g. moving speed, energy cost, amplitude of vibration)?

ADAMS design and material selection knowledge:

What are the necessary properties of a tensegrity robot?

What are the necessary design steps of a Tensegrity robot using the programme ADAMS?

What are the material requirements of a Tensegrity robot given its uses and applied fields?

#### **E. targetLengths: Matlab function for deriving desired rod lengths**

```
function [TL1,TL2,TL3,TL4,TL5,TL6] = targetLengths(cPoint, tLoc, X, Y, Z)
```

```
%Function for choosing the length of each of the rods based on which
```

```
%triangle of rod ends is in contact with the ground
```

```
%Using the triangles orientation relative to targetLoc from initTense.m
```

```
%Define the three possible states of a rod and respective length
```

```
rest = 1.2;
```

```
extend = 1.55;
```

```
startlen = 1;
```

```
contract = 0.85;
```

```
%Define initial rod lengths
```

```
TL1 = rest;
```

```
TL2 = rest;
```

```
TL3 = rest;
```

```
TL4 = rest;
```

```
TL5 = rest;
```

```
TL6 = rest;
```



```

rD = zeros(1,12);
%Calculate XZ distance of each rod end to target locations
for r = 1:12
    rD(r) = sqrt((cPoint(1) - X(r))^2 + (cPoint(3) - Z(r))^2);
end

%Find mean distance of all rod ends from target
td = sqrt((tLoc(1)-mean(X))^2+(tLoc(3)-mean(Z))^2);

%Count the number of rod ends in contact with the ground
fC = sum(Y <= 0.05);

%If any booleans are met the target lengths will be redefined
if fC ~ 3 %Rod lengths will only adjust if three rods touch the ground
    TL1 = rest;
    TL2 = rest;
    TL3 = rest;
    TL4 = rest;
    TL5 = rest;
    TL6 = rest;
elseif td < 0.5 %If the robot is within half a meter of the target, stop.
    TL1 = rest;
    TL2 = rest;
    TL3 = rest;
    TL4 = rest;
    TL5 = rest;
    TL6 = rest;
%Open Triangles
elseif Y(1)<= 0.05 && Y(10) <= 0.05 && Y(11) <= 0.05 %1A-5B-6A
    if rD(1) >= rD(10) && rD(1) >= rD(11) %If 1A is the furthest from the target make this
adjustment
        TL1 = extend;
        TL5 = contract;
        TL6 = contract;
        TL3 = extend;
        TL4 = startlen;
    elseif rD(10) >= rD(1) && rD(10) >= rD(11) %If 5A is the furthest from the target make this
adjustment
        TL1 = contract;
        TL5 = extend;
        TL6 = contract;
    else %If last rod on ground is the furthest from the target make this adjustment
        TL1 = contract;
        TL5 = contract;
        TL6 = extend;

```

```

end
elseif Y(2) <= 0.05 && Y(10) <= 0.05 && Y(11) <= 0.05 %1B-5B-6A
  if rD(2) >= rD(10) && rD(2) >= rD(11)
    TL1 = extend;
    TL5 = contract;
    TL6 = contract;
    TL4 = extend;
    TL3 = startlen;
  elseif rD(10) >= rD(2) && rD(10) >= rD(11)
    TL1 = contract;
    TL5 = extend;
    TL6 = contract;
  else
    TL1 = contract;
    TL5 = contract;
    TL6 = extend;
  end
elseif Y(3) <= 0.05 && Y(9) <= 0.05 && Y(12) <= 0.05 %2A-5A-6B
  if rD(3) >= rD(9) && rD(3) >= rD(12)
    TL2 = extend;
    TL5 = contract;
    TL6 = contract;
    TL4 = extend;
    TL3 = startlen;
  elseif rD(9) >= rD(3) && rD(9) >= rD(12)
    TL2 = contract;
    TL5 = extend;
    TL6 = contract;
  else
    TL2 = contract;
    TL5 = contract;
    TL6 = extend;
  end
elseif Y(4) <= 0.05 && Y(9) <= 0.05 && Y(12) <= 0.05 %2B-5A-6B
  if rD(4) >= rD(9) && rD(4) >= rD(12)
    TL2 = extend;
    TL5 = contract;
    TL6 = contract;
    TL3 = extend;
    TL4 = startlen;
  elseif rD(9) >= rD(4) && rD(9) >= rD(12)
    TL2 = contract;
    TL5 = extend;
    TL6 = contract;
  else

```

```

    TL2 = contract;
    TL5 = contract;
    TL6 = extend;
end
elseif Y(5) <= 0.05 && Y(2) <= 0.05 && Y(3) <= 0.05 %3A-1B-2A
    if rD(5) >= rD(2) && rD(5) >= rD(3)
        TL3 = extend;
        TL1 = contract;
        TL2 = contract;
        TL5 = extend;
        TL6 = startlen;
    elseif rD(2) >= rD(5) && rD(2) >= rD(3)
        TL3 = contract;
        TL1 = extend;
        TL2 = contract;
    else
        TL1 = contract;
        TL3 = contract;
        TL2 = extend;
    end
elseif Y(6) <= 0.05 && Y(2) <= 0.05 && Y(3) <= 0.05 %3B-1B-2A
    if rD(6) >= rD(2) && rD(6) >= rD(3)
        TL1 = contract;
        TL3 = extend;
        TL2 = contract;
        TL6 = extend;
        TL5 = startlen;
    elseif rD(2) >= rD(6) && rD(2) >= rD(3)
        TL1 = extend;
        TL3 = contract;
        TL2 = contract;
    else
        TL1 = contract;
        TL3 = contract;
        TL2 = extend;
    end
elseif Y(7) <= 0.05 && Y(1) <= 0.05 && Y(4) <= 0.05 %4A-1A-2B
    if rD(7) >= rD(1) && rD(7) >= rD(4)
        TL4 = extend;
        TL1 = contract;
        TL2 = contract;
        TL6 = extend;
        TL5 = startlen;
    elseif rD(1) >= rD(7) && rD(1) >= rD(4)
        TL4 = contract;

```

```

    TL1 = extend;
    TL2 = contract;
else
    TL4 = contract;
    TL1 = contract;
    TL2 = extend;
end
elseif Y(8) <= 0.05 && Y(1) <= 0.05 && Y(4) <= 0.05 %4B-1A-2B
    if rD(8) >= rD(1) && rD(8) >= rD(4)
        TL4 = extend;
        TL1 = contract;
        TL2 = contract;
        TL5 = extend;
        TL6 = startlen;
    elseif rD(1) >= rD(8) && rD(1) >= rD(4)
        TL4 = contract;
        TL1 = extend;
        TL2 = contract;
    else
        TL4 = contract;
        TL1 = contract;
        TL2 = extend;
    end
elseif Y(9) <= 0.05 && Y(6) <= 0.05 && Y(7) <= 0.05 %5A-3B-4A
    if rD(9) >= rD(6) && rD(9) >= rD(7)
        TL5 = extend;
        TL3 = contract;
        TL4 = contract;
        TL1 = extend;
        TL2 = startlen;
    elseif rD(6) >= rD(9) && rD(6) >= rD(7)
        TL5 = contract;
        TL3 = extend;
        TL4 = contract;
    else
        TL5 = contract;
        TL3 = contract;
        TL4 = extend;
    end
elseif Y(10) <= 0.05 && Y(6) <= 0.05 && Y(7) <= 0.05 %5B-3B-4A
    if rD(10) >= rD(6) && rD(10) >= rD(7)
        TL5 = extend;
        TL3 = contract;
        TL4 = contract;
        TL2 = extend;
    end

```

```

    TL1 = startlen;
elseif rD(6) >= rD(10) && rD(6) >= rD(7)
    TL5 = contract;
    TL3 = extend;
    TL4 = contract;
else
    TL5 = contract;
    TL3 = contract;
    TL4 = extend;
end
elseif Y(11) <= 0.05 && Y(5) <= 0.05 && Y(8) <= 0.05 %6A-3A-4B
    if rD(11) >= rD(5) && rD(11) >= rD(8)
        TL6 = extend;
        TL3 = contract;
        TL4 = contract;
        TL2 = extend;
        TL1 = startlen;
    elseif rD(5) >= rD(11) && rD(5) >= rD(8)
        TL6 = contract;
        TL3 = extend;
        TL4 = contract;
    else
        TL6 = contract;
        TL3 = contract;
        TL4 = extend;
    end
elseif Y(12) <= 0.05 && Y(5) <= 0.05 && Y(8) <= 0.05 %6B-3A-4B
    if rD(12) >= rD(5) && rD(12) >= rD(8)
        TL6 = extend;
        TL3 = contract;
        TL4 = contract;
        TL1 = extend;
        TL2 = startlen;
    elseif rD(5) >= rD(12) && rD(5) >= rD(8)
        TL6 = contract;
        TL3 = extend;
        TL4 = contract;
    else
        TL6 = contract;
        TL3 = contract;
        TL4 = extend;
    end
end

```

%Closed triangles

```

elseif Y(1) <= 0.05 && Y(10) <= 0.05 && Y(7) <= 0.05 %1A-5B-4A

```

```

if rD(1) >= rD(10) && rD(1) >= rD(7)
    TL1 = extend;
    TL5 = contract;
    TL4 = contract;
elseif rD(10) >= rD(1) && rD(10) >= rD(7)
    TL1 = contract;
    TL5 = extend;
    TL4 = contract;
else
    TL1 = contract;
    TL5 = contract;
    TL4 = extend;
end
elseif Y(1)<= 0.05 && Y(8) <= 0.05 && Y(11) <= 0.05 %1A-4B-6A
    if rD(1) >= rD(8) && rD(1) >= rD(11)
        TL1 = extend;
        TL4 = contract;
        TL6 = contract;
    elseif rD(8) >= rD(1) && rD(8) >= rD(11)
        TL1 = contract;
        TL4 = extend;
        TL6 = contract;
    else
        TL1 = contract;
        TL4 = contract;
        TL6 = extend;
    end
elseif Y(2)<= 0.05 && Y(6) <= 0.05 && Y(10) <= 0.05 %1B-3B-5B
    if rD(2) >= rD(6) && rD(2) >= rD(10)
        TL1 = extend;
        TL3 = contract;
        TL5 = contract;
    elseif rD(6) >= rD(2) && rD(6) >= rD(10)
        TL1 = contract;
        TL3 = extend;
        TL5 = contract;
    else
        TL1 = contract;
        TL3 = contract;
        TL5 = extend;
    end
elseif Y(2)<= 0.05 && Y(5) <= 0.05 && Y(11) <= 0.05 %1B-3A-6A
    if rD(2) >= rD(5) && rD(2) >= rD(11)
        TL1 = extend;
        TL3 = contract;

```

```

    TL6 = contract;
elseif rD(5) >= rD(2) && rD(5) >= rD(11)
    TL1 = contract;
    TL3 = extend;
    TL6 = contract;
else
    TL1 = contract;
    TL3 = contract;
    TL6 = extend;
end
elseif Y(3)<= 0.05 && Y(6) <= 0.05 && Y(9) <= 0.05 %2A-3B-5A
    if rD(3) >= rD(6) && rD(3) >= rD(9)
        TL2 = extend;
        TL3 = contract;
        TL5 = contract;
    elseif rD(6) >= rD(3) && rD(6) >= rD(9)
        TL2 = contract;
        TL3 = extend;
        TL5 = contract;
    else
        TL2 = contract;
        TL3 = contract;
        TL5 = extend;
    end
elseif Y(3)<= 0.05 && Y(5) <= 0.05 && Y(12) <= 0.05 %2A-3A-6B
    if rD(3) >= rD(5) && rD(3) >= rD(12)
        TL2 = extend;
        TL3 = contract;
        TL6 = contract;
    elseif rD(5) >= rD(3) && rD(5) >= rD(12)
        TL2 = contract;
        TL3 = extend;
        TL6 = contract;
    else
        TL2 = contract;
        TL3 = contract;
        TL6 = extend;
    end
elseif Y(4)<= 0.05 && Y(7) <= 0.05 && Y(9) <= 0.05 %2B-4A-5A
    if rD(4) >= rD(7) && rD(4) >= rD(9)
        TL2 = extend;
        TL4 = contract;
        TL5 = contract;
    elseif rD(7) >= rD(4) && rD(7) >= rD(9)
        TL2 = contract;

```

```

    TL4 = extend;
    TL5 = contract;
else
    TL2 = contract;
    TL4 = contract;
    TL5 = extend;
end
elseif Y(4) <= 0.05 && Y(8) <= 0.05 && Y(12) <= 0.05 %2B-4B-6B
    if rD(4) >= rD(8) && rD(4) >= rD(12)
        TL2 = extend;
        TL4 = contract;
        TL6 = contract;
    elseif rD(8) >= rD(4) && rD(8) >= rD(12)
        TL2 = contract;
        TL4 = extend;
        TL6 = contract;
    else
        TL2 = contract;
        TL4 = contract;
        TL6 = extend;
    end
end
end
end

```

## F. Obstacle avoiding function

```

function cPoint = checkpoint(tar, obs, X, Z)
%Takes a top view of the simulation environment to find an optimal route
%around the obstacles. Graphically represented by X and Z axis.
pX = tar(1);
pZ = tar(3);
totOb = length(obs)/3; %Find total number of obstacles
disObX = zeros(1, totOb); %Create sized zero matrix for obstacles x distance from target
disObZ = zeros(1, totOb); %Create sized zero matrix for obstacles z distance from target
disOb = zeros(1, totOb); %Create sized zero matrix for obstacles distance from target
mX = mean(X); %find the mean x position of the tensegrity
mZ = mean(Z); %find the mean z position of the tensegrity
dX = tar(1) - mX; %find the x distance between the the mean point and target
dZ = tar(3) - mZ; %find the z distance between the the mean point and target
b = 1; %set counter value for upcoming loop
for a = 1:3:totOb*3 %run loop along x values of all obstacles
    disObX(b) = tar(1)-(obs(a)+0.5); %calculate x distance of all obstacles from target
    b = b+1; %add to counter for disObx position
end
end

```



```

b = 1; %set counter value for upcoming loop
for a = 3:3:totOb*3 %run loop along z values of all obstacles
    disObZ(b) = tar(3)-obs(a); %calculate z distance of all obstacles from target
    b = b+1; %add to counter for disObz position
end

disT = sqrt(dX^2+dZ^2); %Calculate the distance between tensegrity and target
b = zeros(1,totOb); %create an array of zeros to check which obstacles are in the way
for a = 1:totOb %run loop along all obstacles
    disOb(a) = sqrt(disObX(a)^2+disObZ(a)^2); %calculate distance of each obstacle from final
    target
    if disOb(a) < disT-0.2 %check if obstacle is closer than tensegrity
        if disObX(a)+1.5 > dX > disObX(a)-1.5 && dZ > 0 && disObZ(a) > 0 %check if obstacle is
            between tensegrity and target
            b(a) = 1; %set array value to 1
        elseif disObX(a)+1.5 > dX > disObX(a)-1.5 && dZ < 0 && disObZ(a) < 0 %check if obstacle
            is between tensegrity and target
            b(a) = 1; %set array value to 1
        end
    end
end
end

nearOb = 0; %set value for selecting the nearest obstacle to tensegrity
for a = 1:totOb %check each obstacle
    if b(a) == 1 %Confirm if obstacle is in the way
        if disOb(a) > nearOb %Check if obstacle is nearest to tensegrity
            nearOb = disOb(a); %set new nearest distance
            pXr = obs(3*a-2)-1.75; %Create X coordinate 1.75m to right of obstacle
            pZr = obs(3*a); %Create Z coordinate to right of obstacle
            pDr = sqrt((tar(1)-pXr)^2+(tar(3)-pZr)^2); %Calculate point distance to target
            pXl = obs(3*a-2)+2.75; %create X coordinate 1.75m to left of obstacle
            pZl = obs(3*a); %create Z coordinate to left of obstacle
            pDl = sqrt((tar(1)-pXl)^2+(tar(3)-pZl)^2); %Calculate point distance to target
            if pDr>pDl %check which point is closest to target
                pX = pXl; %set checkpoint coordinate to left side
                pZ = pZl; %set checkpoint coordinate to left side
            else
                pX = pXr; %set checkpoint coordinate to right side
                pZ = pZr; %set checkpoint coordinate to right side
            end
            disP = sqrt((pX - mX)^2+(pZ - mZ)^2); %Find distance to checkpoint
            if disP < 0.5 %check if tensegrity is at checkpoint
                nearOb = 0; %move on to next target
            end
        end
    end
end
end
end

```

end

if nearOb == 0 %if no obstacles are in the way

    pX = tar(1); %Set checkpoint to target

    pZ = tar(3);

end

cPoint = [pX, 0.5, pZ]; %Set output checkpoint

end

### G. Planning tracker used for project

Integration Project: Tensegrity Robot		April														May																																								
George Gittins S3319040		Week 14				Week 15				Week 16				Week 17				Week 18			Week 19				Week 20				Week 21			Week 22				Week 23																				
Phase	Status	5	6	7	8	9	12	13	14	15	16	19	20	21	22	23	24	27	28	29	30	3	4	5	6	7	10	11	12	13	14	17	18	19	20	21	24	25	26	27	28	31	1	2	3	4	5	6	7	8						
Follow tutorials in Adams	Completed																																																							
Practice Matlab and review existing code	Completed																																																							
Research Literature on mathematical models	Completed																																																							
Research Literature on tensegrity structure	Completed																																																							
Record progress in report	Completed																																																							
Design tensegrity robot in Adams	Completed																																																							
Create code for controlling tensegrity	Completed																																																							
Make KPIs for testing tensegrity robot	Completed																																																							
Record progress in report	Completed																																																							
Connect Matlab code to Adams Design	Completed																																																							
Create the control system in Simulink	Completed																																																							
Design simulation environments to test robot	Completed																																																							
Record progress in report	Completed																																																							
Hand in Intermediate Report (IR)	Completed																																																							
Run tensegrity roll tests	Completed																																																							
Evaluate and improve code and design	Completed																																																							
Record progress in report	Completed																																																							
Run tensegrity environment simulation test	Completed																																																							
Evaluate and improve code and design	Completed																																																							
Further research on recent literature	Completed																																																							
Hand in Preliminary report (PR)	Completed																																																							
Select image and write summary for Symposium	Completed																																																							
Complete Final Presentation	In Progress																																																							
Hand in symposium image and summary	Completed																																																							
Write up and Organize Final report	In Progress																																																							
Check English + structure	In Progress																																																							
Design poster for Symposium	In Progress																																																							
Hand in Final Report	Not started																																																							
Hand in Symposium Poster	Not started																																																							
Hand in Final presentation slides	Not started																																																							

Figure G1: Screenshot of excel planning sheet for tensegrity design and control BaIP on 3/6/2021

## H. Older test of object avoiding function and locomotion on 25/05/2021

Separated into three parts of approximately 10 seconds. This was done to speed up the simulation performance of the hardware used.

### Part 1

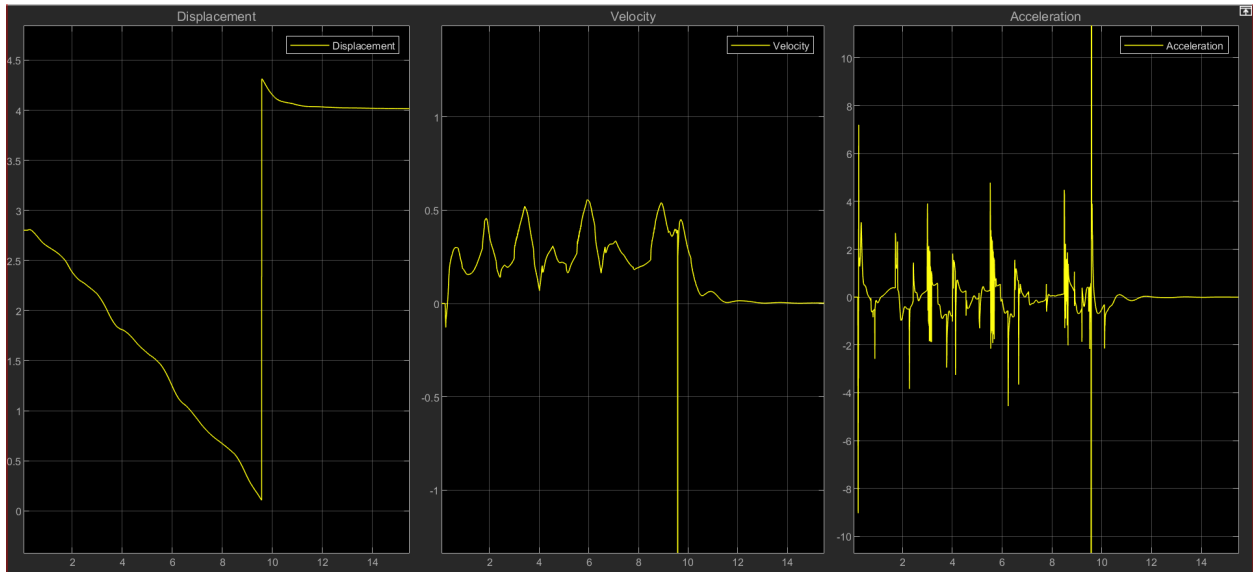


Figure H1: Displacement, velocity and acceleration

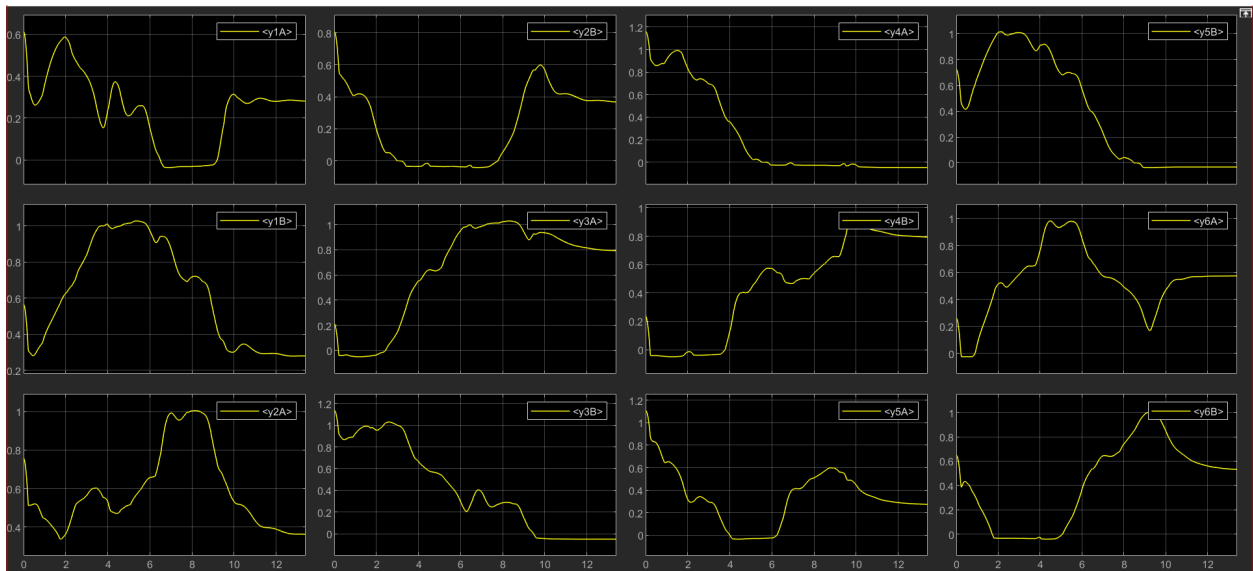


Figure H2: Rod end heights

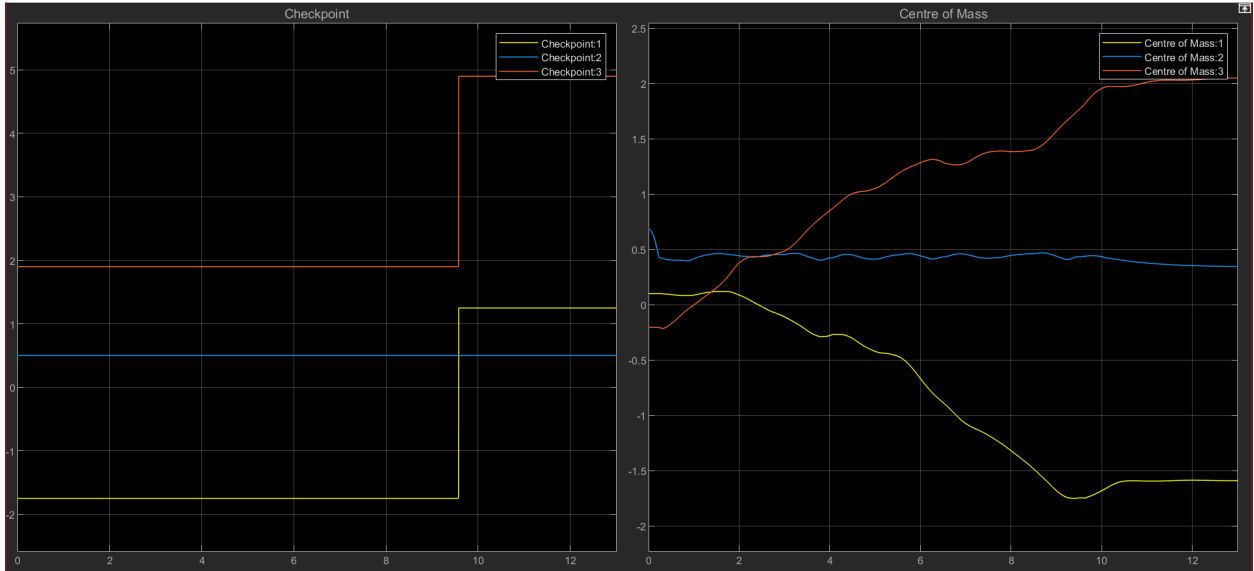


Figure H3: Checkpoint coordinates and current tensegrity coordinates

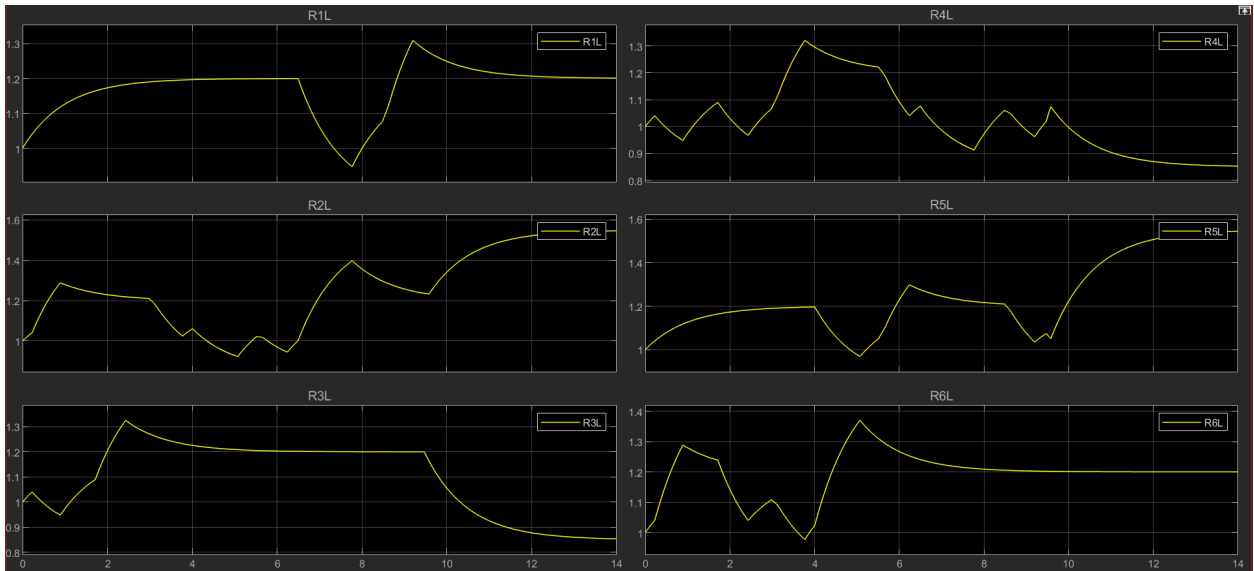


Figure H4: Rod lengths

## Part 2

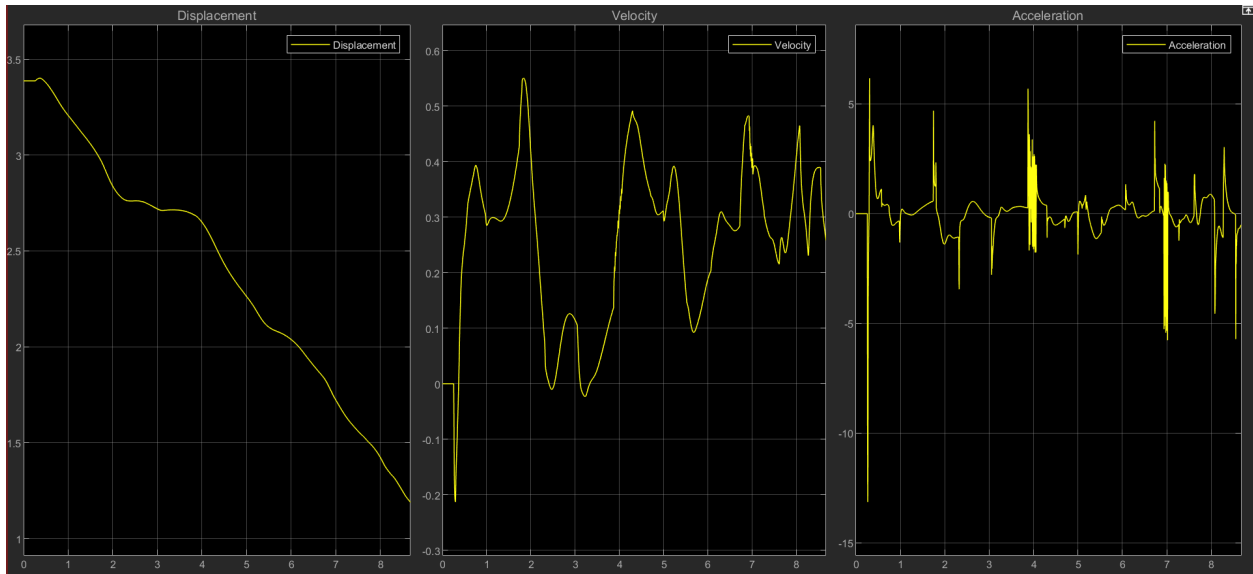


Figure H5: Displacement, velocity and acceleration

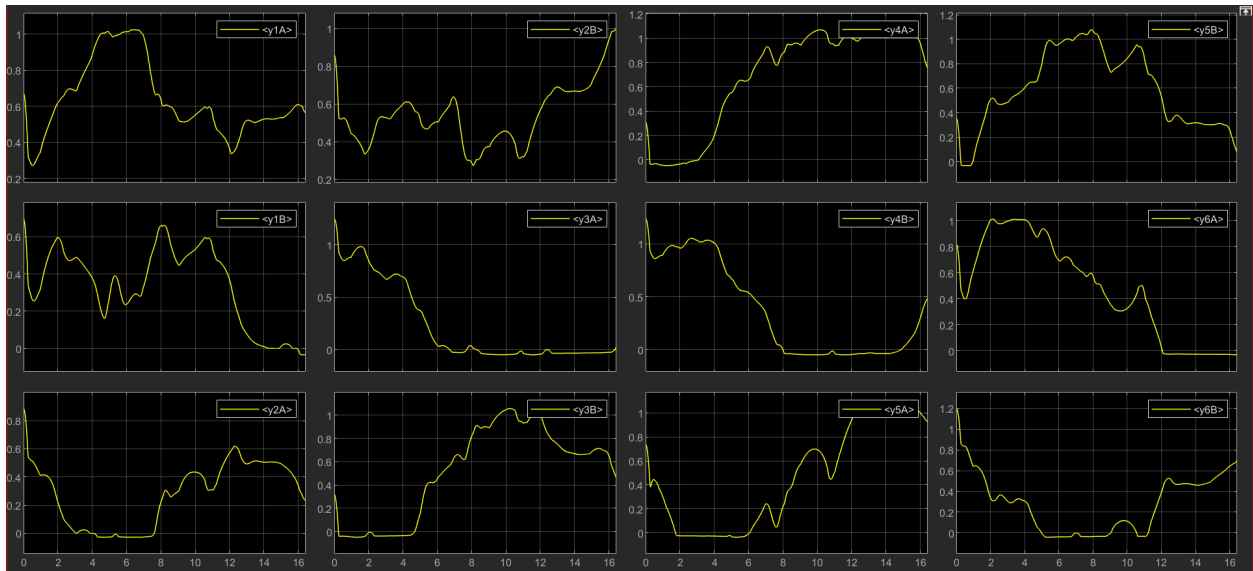


Figure H6: Rod end heights

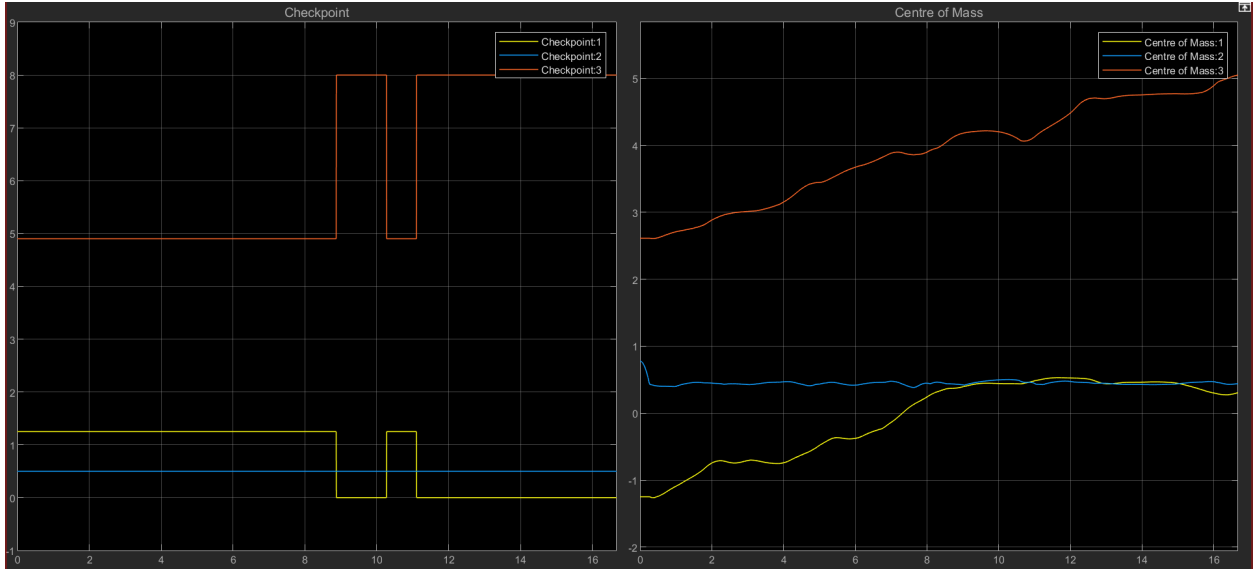


Figure H7: Target coordinates and current coordinates of the tensegrity robot

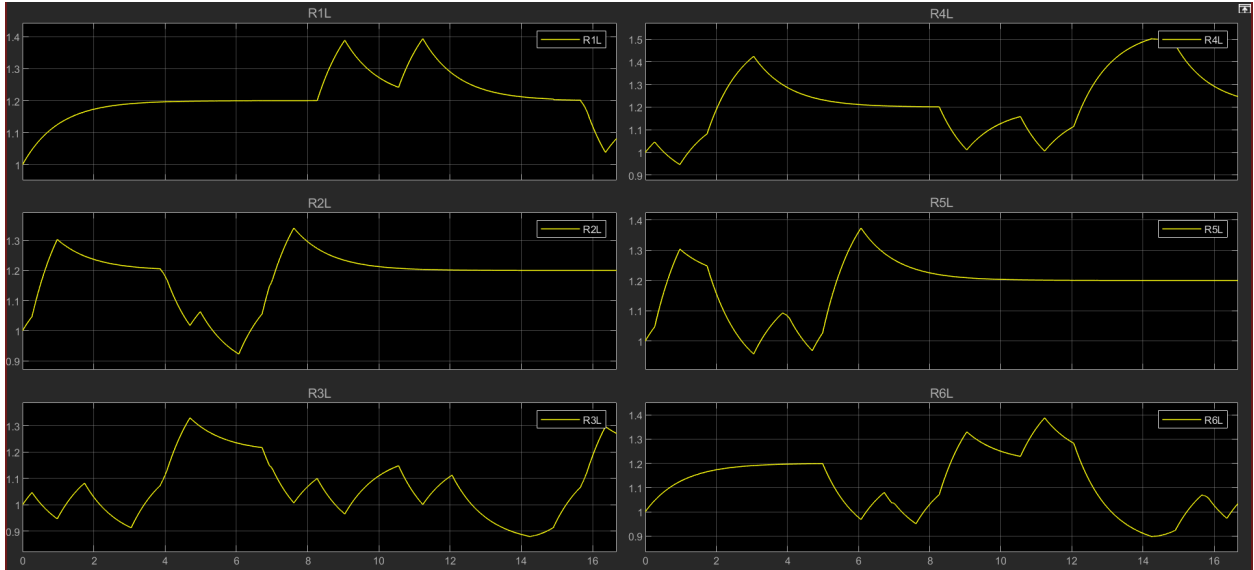


Figure H8: Rod lengths

### Part 3

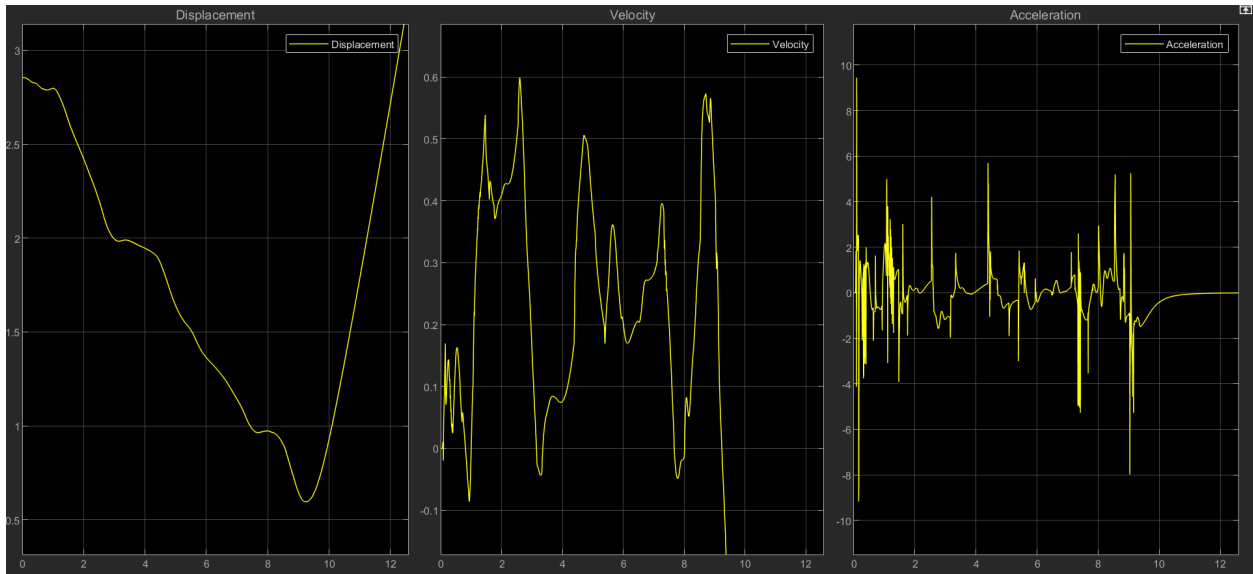


Figure H9: Displacement, velocity and acceleration

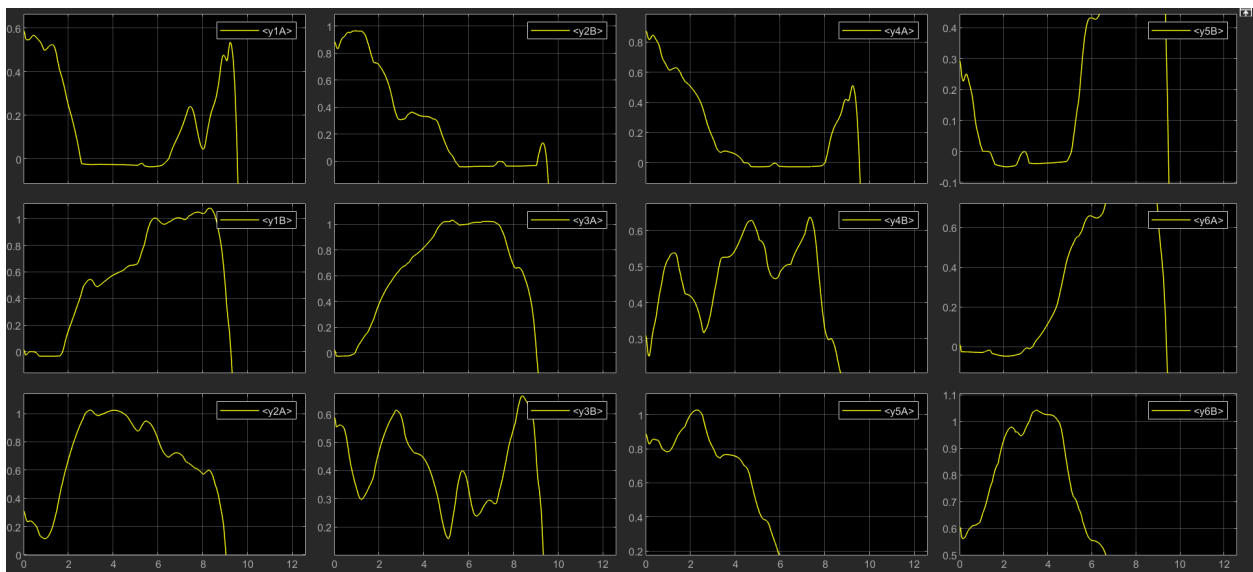


Figure H10: Rod end heights, notice that the tensegrity falls off of the simulation environment platform at 9 seconds

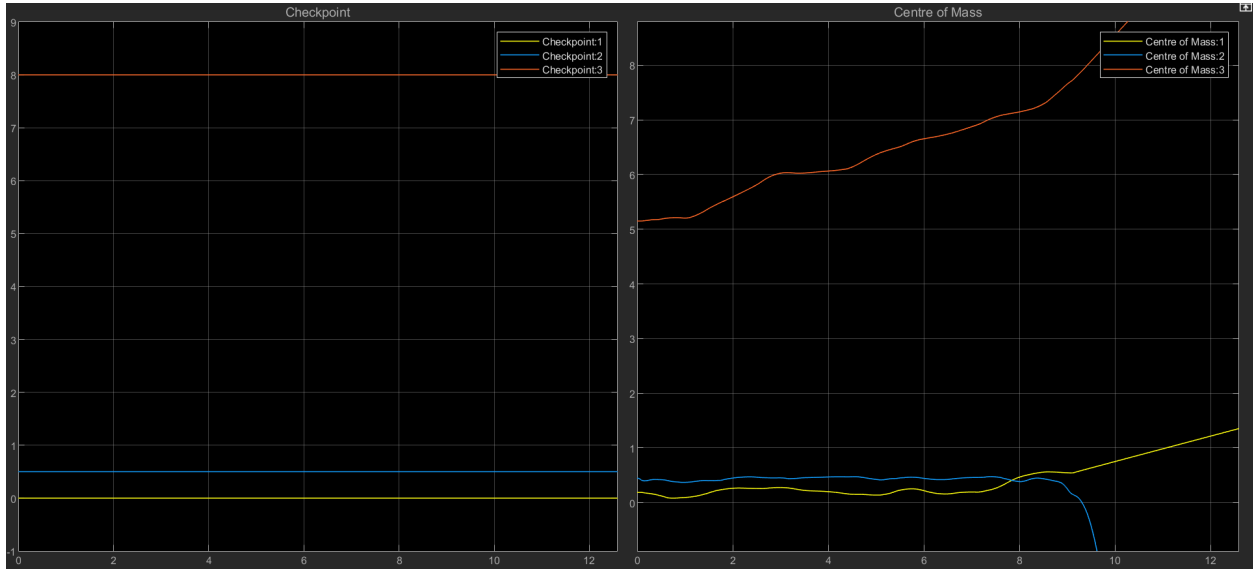


Figure H11: Target coordinates and current coordinates of tensegrity

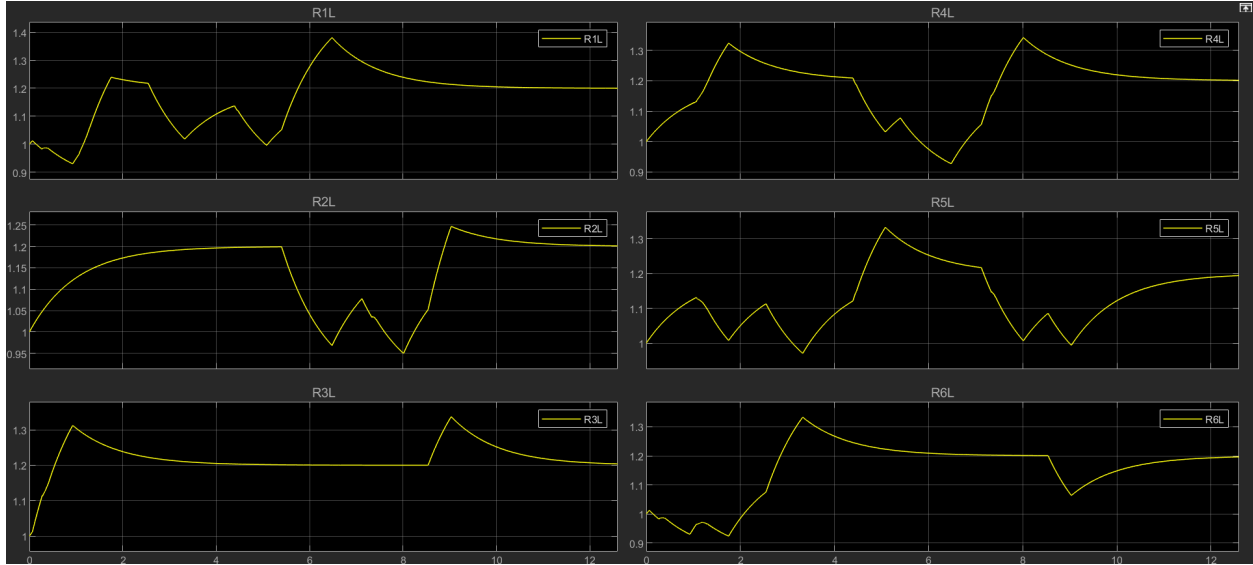


Figure H12: Rod lengths, once target location is reached (and falling off the end of the platform), all rods change their length to rest length of 1.2.

**I. Most recent Force simulation of tensegrity robot drop test**



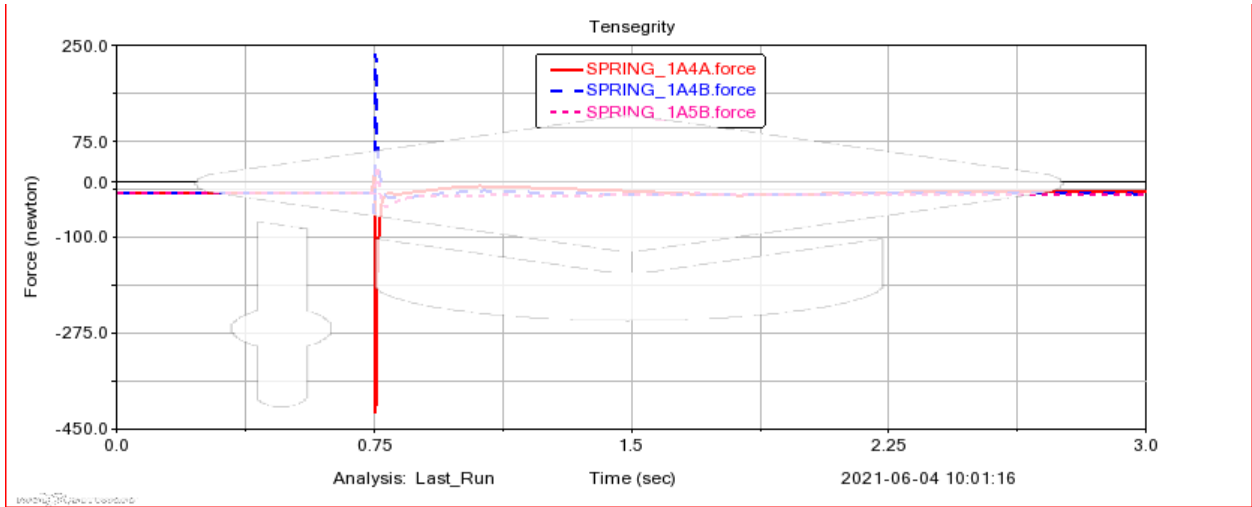


Figure I1: Force distribution test 6/4/2021

Forces reach around 440N due to drop from 3m height. The force dissipates and dampens within 1 second of landing on the ground.