California State University, Fresno

Lyles College of Engineering

Electrical and Computer Engineering Department

TECHNICAL REPORT

Experiment Title: Culminating Assignment

Instructor: Dr. Nagy Bengiamin

Course Title: ECE 173

Date Submitted: 12/15/2020

Prepared By:

Nirmala Sinha

INSTRUCTOR SECTION

Comments: _____

_____

_____

_____

Table of Contents

Section **Page #**

1. Statement Of Objectives

The purpose of this assignment is to build, analyze, and examine the working of a quadcopter in the lab environment. The mechanism of the quadcopter has to be implemented in such a way that it doesn't crash into any walls or ceiling of the lab. The quadcopter will be built using Simulink and its motion constrained by attaching it to a robotic arm. The motion of the quadcopter is used to calculate the arm's corresponding joint angles.

2. Background Information

The concept of a robot was first introduced to the public in 1920 by Czech writer Karel Čapek[1]. The introduction of the robot was made through a play where a chemical factory that produced robots. In this play, the robot was shown as artificial human beings built to reduce human efforts. People acknowledge this concept of robots. Now, robots are considered to be machines that can be programmed easily and are used to reduce human efforts. These machines are beneficial for various industries, robots can be programmed and built for different sectors of society. Robots are designed in a way that they can work efficiently 24X7 without getting exhausted. There are many advantages of a robot. Robots can be very productive, work on multiple tasks, and work in unhealthy environments.

Figure 1: Robots of different types

This project will be focusing on one of the classes of robot i.e. Quadcopter.

A quadcopter is considered to be a helicopter with four propellers [1]. These four propellers work the same way as a fan does. When a fan rotates, stress is created at the

joint where the fan is attached. The rotation of the fan blades generates a thrust, which builds a force on the ceiling. A quadcopter works with the same concept where the four propellers together generate an upward thrust. This thrust force compensates the weight allowing it to lift the quadcopter.

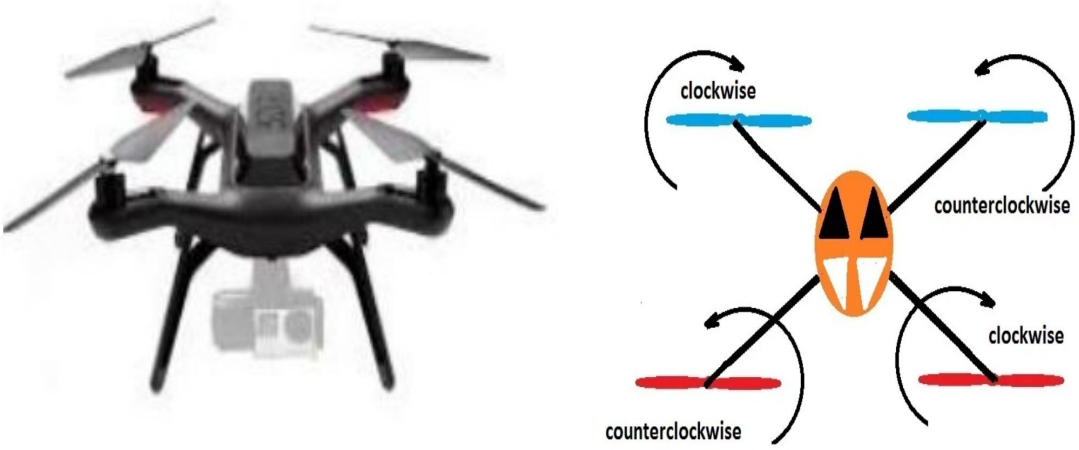

Figure 2: Quadcopter

A quadcopter can have six degrees of freedom. The right side of Figure 2 shown above demonstrates the working of the quadcopter. The two opposite propellers of the quadcopter always rotate in the opposite direction of the other two propellers. This helps in balancing and hovering the quadcopter. It is important to note that the thrust force generated by the speed of the propeller should always be greater than the gravitational force. This allows the quadcopter to lift. Once all the propellers are rotating at the same speed the operator can change the positioning and the orientation of the quadcopter according to their wish. A quadcopter can move vertically as shown in Figure 3.

Figure 3: Vertical movement

To move the quadcopter vertically is quite the same as discussed above. In vertical motion, the two propellers as shown in Figure 3 are more dominant than the other two. If

the dominant propeller speed is higher than the other two propellers. Then the propeller will be lifted upwards. If their speed is decreased gradually then the quadcopter starts to land on the ground.

To move the quadcopter horizontally in any direction one of the propeller speeds is dominant over the others. The direction where the other propeller bends or tilts the quadcopter it moves to that direction. This can be seen in Figure 4 shown below.

Figure 4: Horizontal movement

A quadcopter can also perform different orientations as well. A quadcopter can roll. To make a quadcopter roll one of the side propellers moves faster while the other two opposite propellers are dominant. This allows a quadcopter to roll the right or left side. This can be observed in Figure 5 and Figure 6 below.

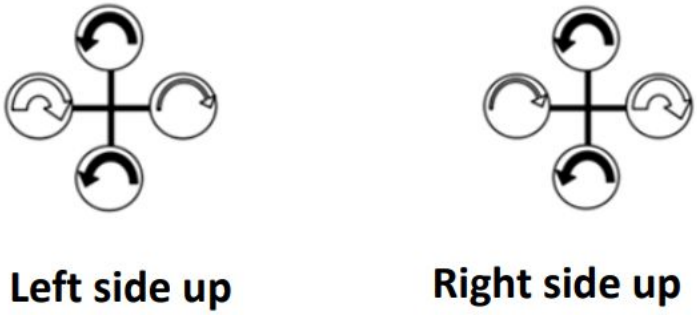

Figure 5: Working for roll

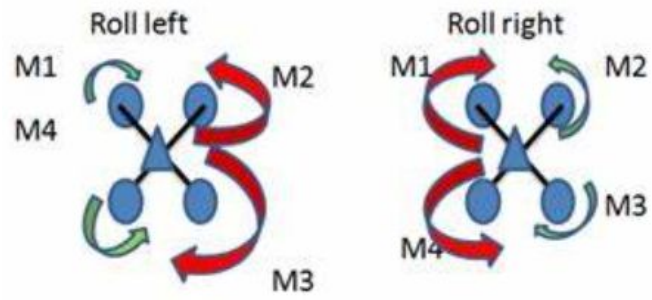

Figure 6: Quadcopter performing rolling motion

In the pitch motion, a quadcopter moves forward or backward. In this motion, if the quadcopter has to move the bottom side as shown in Figure 7. Then one of the opposite propellers moves at a higher speed. The other two propellers move at the equal speed allowing the quadcopter to move forward or backward.

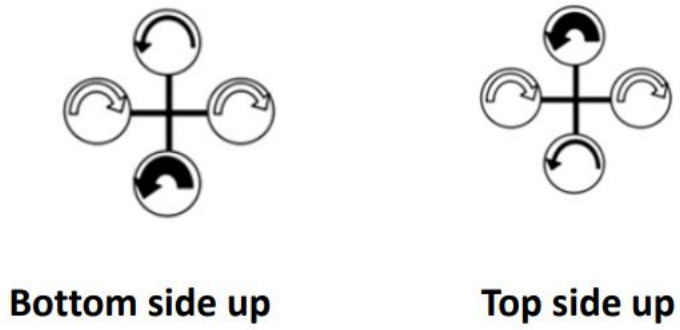

Figure 7: Working for pitch

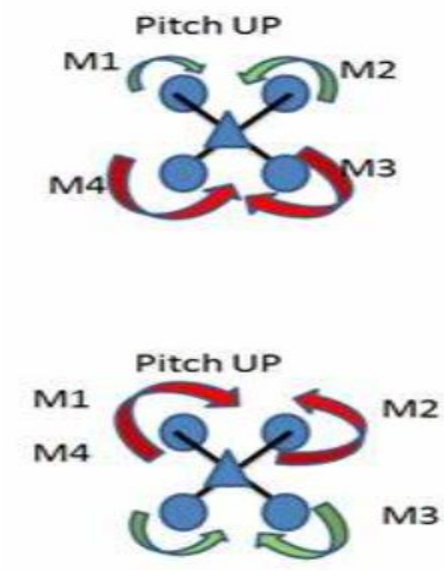

Figure 8: Quadcopter performing pitch motion

In yaw motion, the quadcopter can either move clockwise or counterclockwise. It is very straightforward to understand it is the movement along the vertical axis.
This can be seen in Figure 9 and Figure 10. When the two opposite side propellers move at the same speed and the other two propellers move at little speed.

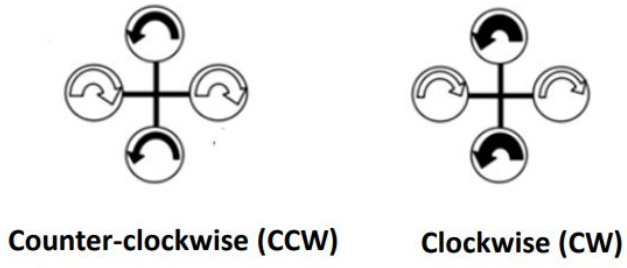

Figure 9: Working of yaw

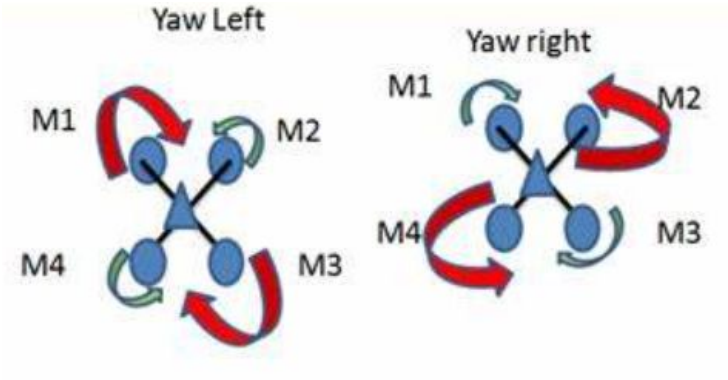

Figure 10: Quadcopter performing yaw motion

3. Experimental Procedure

a. Equipment Used

i. Simulink
ii. Matlab
iii. Google docs

b. Procedure

This project is divided into four sections. In the first section of the report students have to use simulink to create a quadcopter . In the second part students have to use DH tables and inverse kinematics to find angles of joints. In the third part students have to find the positioning of the quadcopter.

In this part of the report students have to create a simulink model. To do this first click on the simulink library choose the step function to create the yaw_command and lift_comand as shown in the Figure 11 below. To set them up double click on the step function set the step time to 0.33. For yaw_command make sure the final value and initial value is set to be 0. This can be seen in the Figure 11 below. Then click on apply and then okay.

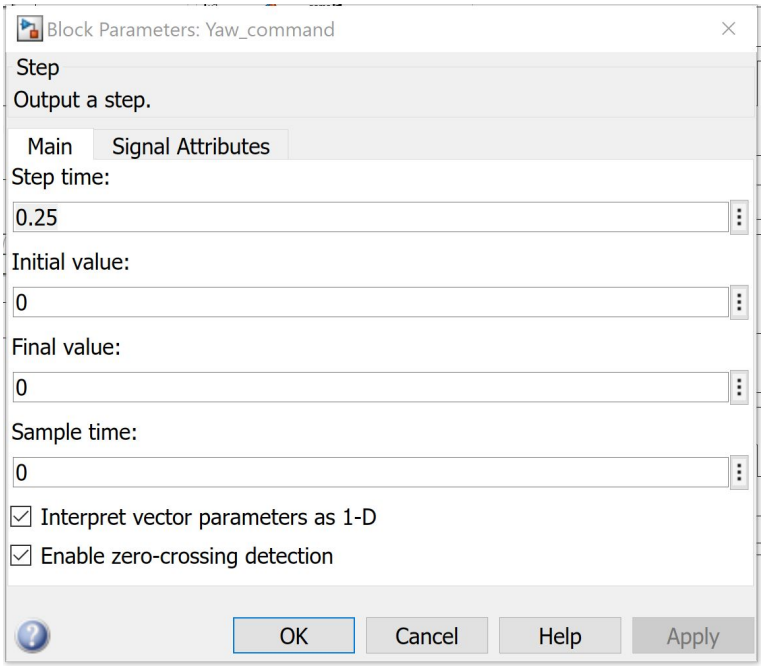

Figure 11: yaw_command setting

Set the lift_command similarly to yaw_command. Once this is done then use a function block to create angle function. Connect the roll_command to u2 of the angle function block. Pitch_command to u3 and then connect yaw_command

This angle function is built to output the angles. It generates the value of different omega's,gama double dot, alpha double dot, alpha double .This can be seen in the Figure 12 below. Connect the output of the angle function y to a demux.

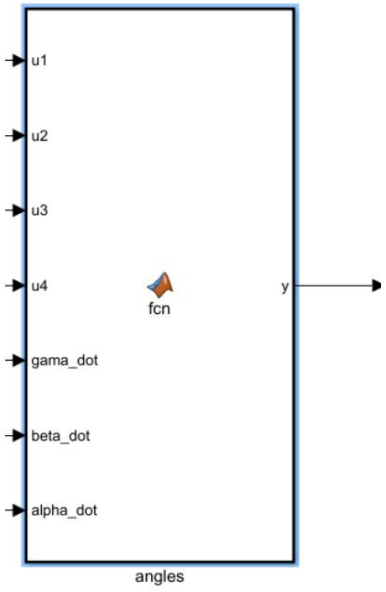

Figure 12 : angle block

The angle function can be connected as shown in the figure 13 below. A feedback loop is used to connect the gamma dot, bet dot and alpha dot. Once this is done connect the one of the branches of the demux with integrator block and absolute block. The output of the integrator and absolute block is connected to a switch as shown below in the figure 13. The switch block has three inputs and one output so the third input that should enter a switch block is a constant (value of constant =0) block. Do the same for the rest of the blocks. Then set the switch condition as shown in the figure 14 below

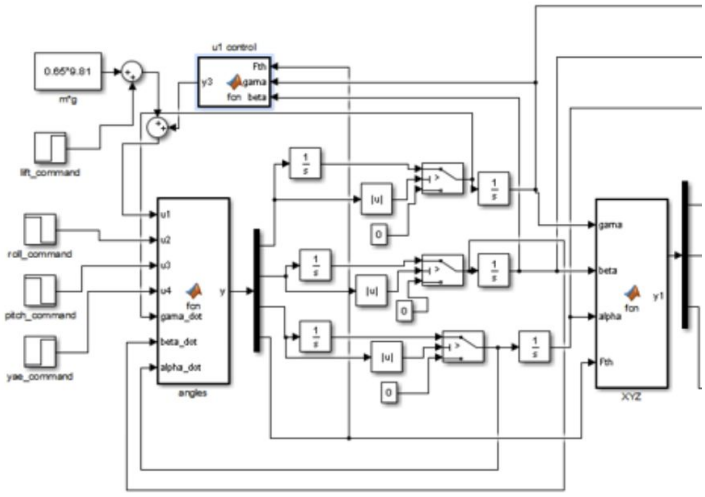

Figure 13: Connection of angle blocks

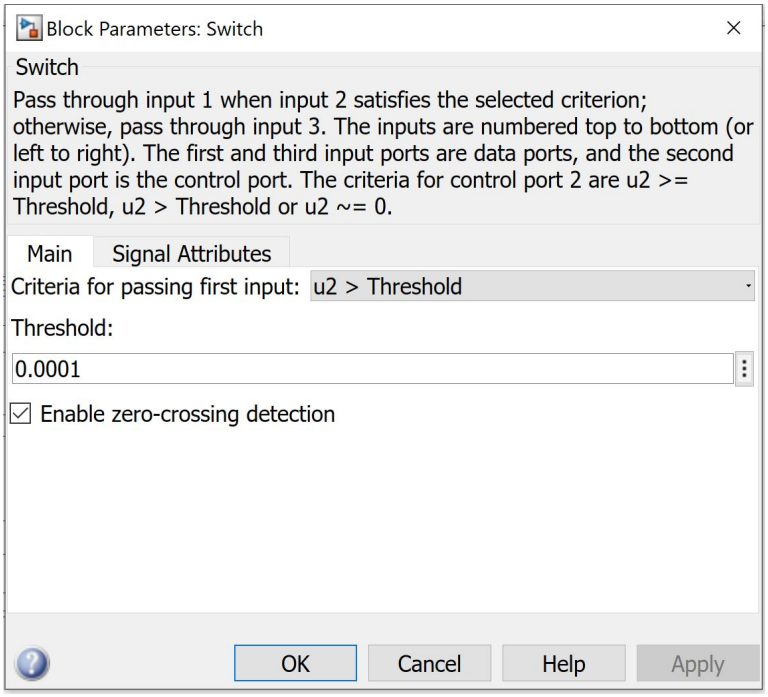

Figure 14 : Switch block setting

Then build a u1_control by using another function block. This function block will allow the as shown in the figure 15 below. The main purpose of making this function is to compensate for the lift force. Connect the output of this function with the m*g and lift_command block to the u1 on the angle block.

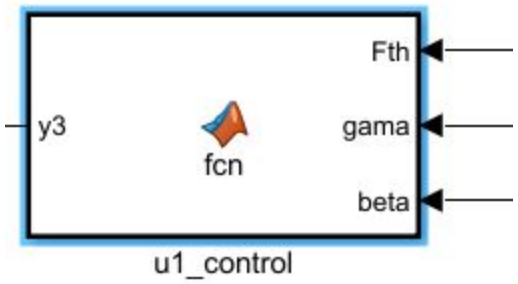

Figure 15: U1 control

Connect the u1_control block as shown in the Figure above. Then use the feedback loop to connect the thrust force, gamma and beta. These are the outputs of the angle block.

Once this is done, create another function block called XYZ, this produces the x double dot, y double dot, z double dot as shown in figure 16. The angle block is used to generate x, y and z.

Connect the output of XYZ function to demux then connect output branch of the demux to integrate block, absolute block. The output of integrate block, absolute block, is then connected to a switch. The third input of the switch is connected to a constant value (i.e) is zero. Then connect the output of this switch with an integration block. Connect the output integration block to gain. This generates x. This can be seen in figure 16.

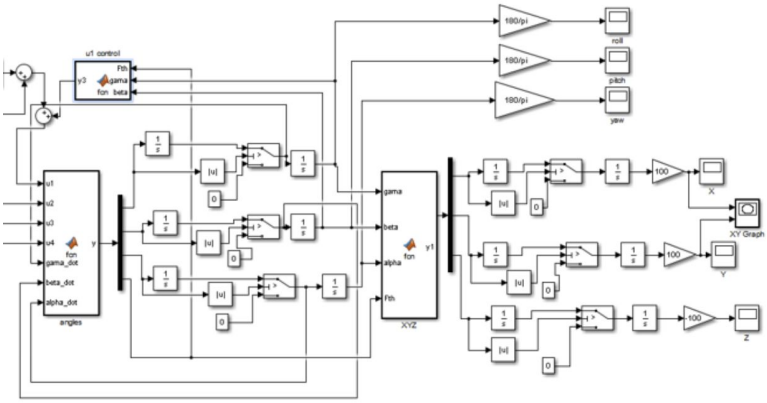

Figure 16: XYZ connection

To check the graphs built add a scope to the output of the gain block. Once all the blocks are connected the system should look like the figure 17 shown below.

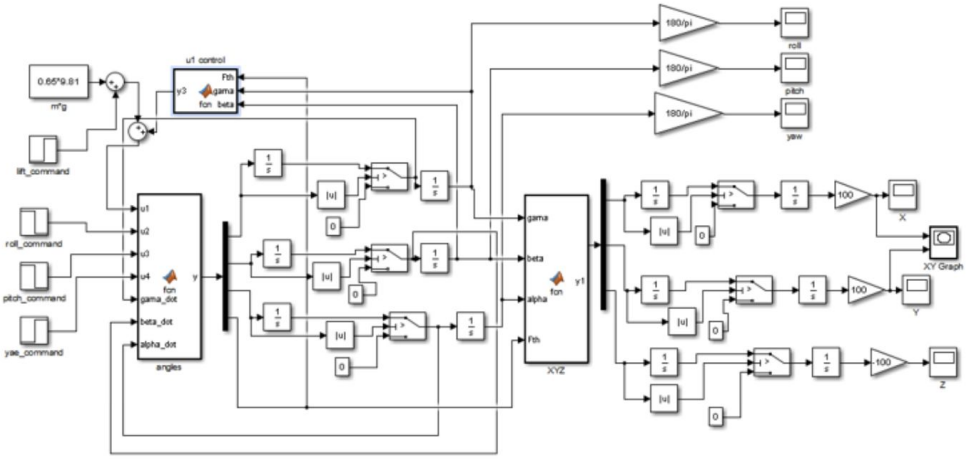

Figure 17: Add scope

These scopes allow the user to see the result obtained when the function starts to run. Once this is done set the modeling on the top bar of the matlab then click on the modeling settings. Click on the Solver and set it up to Ode 113 (Adams). Set the max step size to 0.005, relative tolerance to 1e-3 and min step size to 0.0001. Also set up the stop time to 1second. Once all the settings are set up then click on

the run. This is a basic model for creating a simulink. Once this is done students can jump onto the first part of the assignment.

i. ***Part 1- simulink***

For this project the students have to build a simulink model that produces a circular function. There are various ways to do it.

First students need to decide the value of x" (x double dot) and using this value, students can find the y" (y double dot) value. For this project a circle's equation can be used as shown in the figure18 below. Through these equations students can find y".

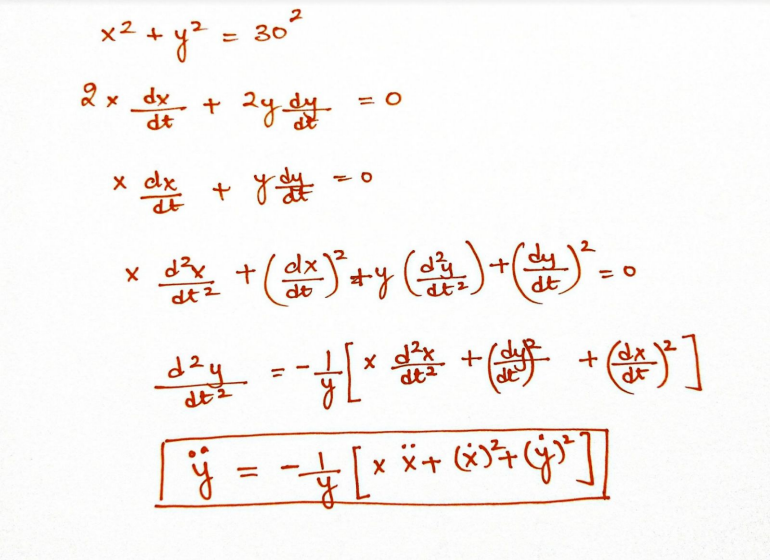

Figure 18: Equation used to derive the y"

Once this is done, the above equation shown in the figure 18 above has to be implemented in the simulink. Use a feedback loop to substitute the value of x. Take the product block, connect the feedback loop for x and connect it with x" block. This can be seen in figure 19.

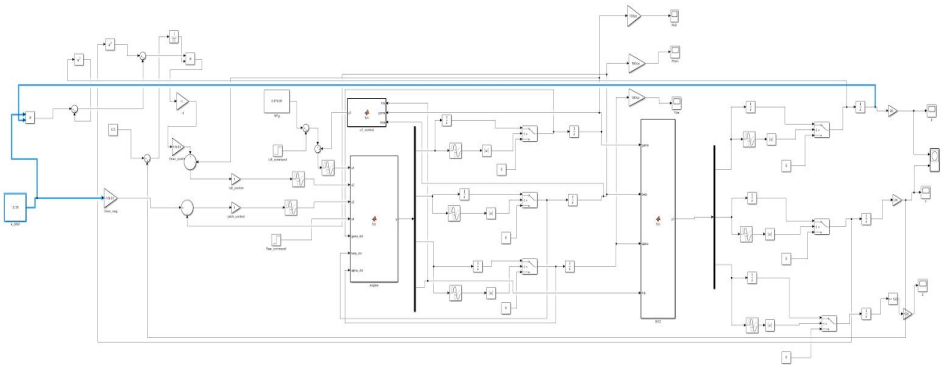

Figure 19: Building equation block

Once this is done add the x' square and y' square to the block. Then divide the whole equation with y feedback. After completing this step add a gravitational gain block to x" and y". Then connect the values to the u2 and u3.

The entire system should look like the figure 20 shown below.

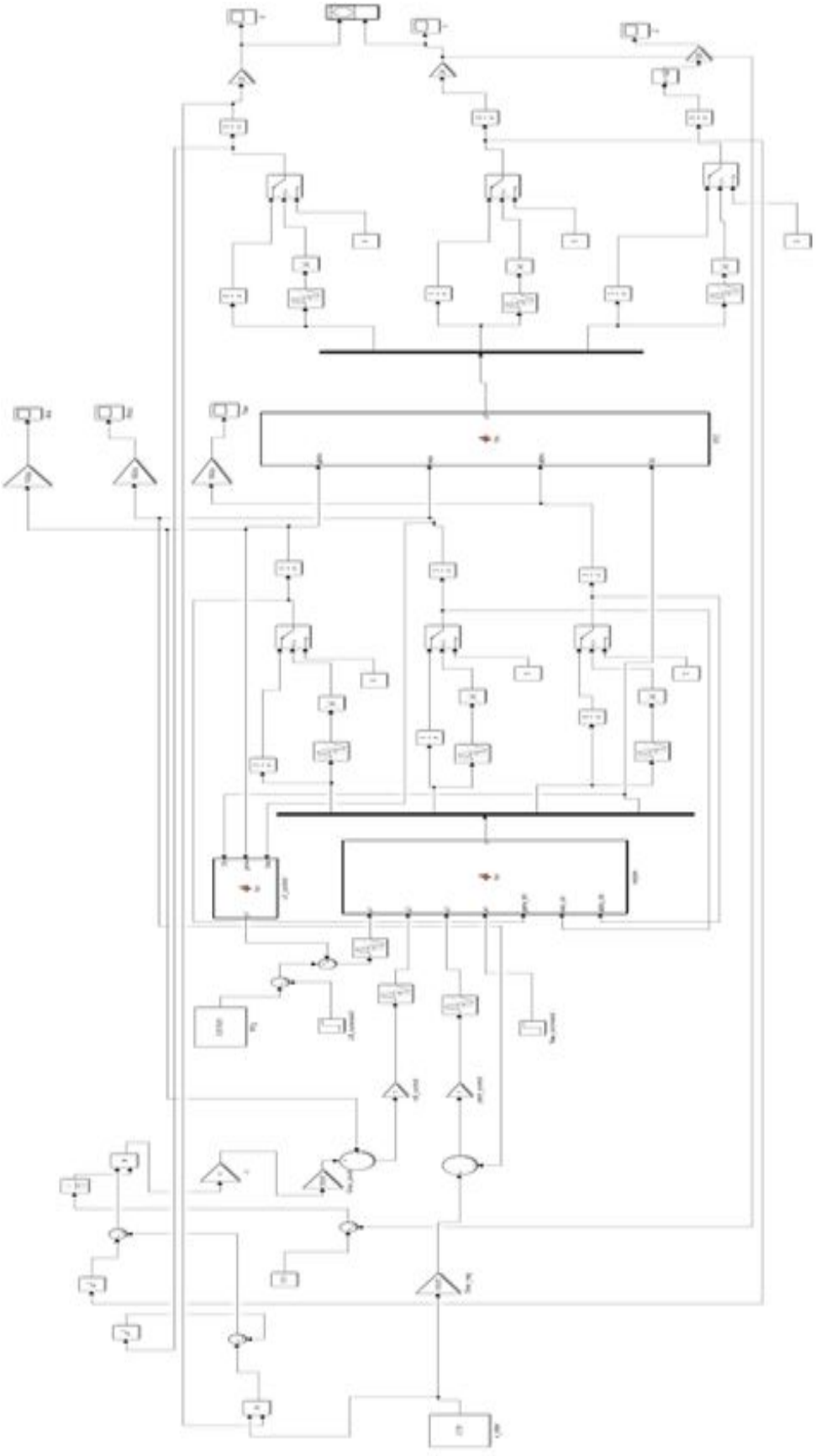

Figure 20: Simulink model

Once this is done adjust the modelling settings as discussed above. Then run the simulation and wait for the result to build up.

ii. Part 2- inverse kinematics

In this part of the project the first step is to import the data of x,y and z generated to matlab workspace. This can be done by just clicking on the x scope and making changes to the settings. In the settings, click on logging check Log data to workspace box, name the variable and change the save format of the x scope to array. This can be seen in the figure 21 as shown below.

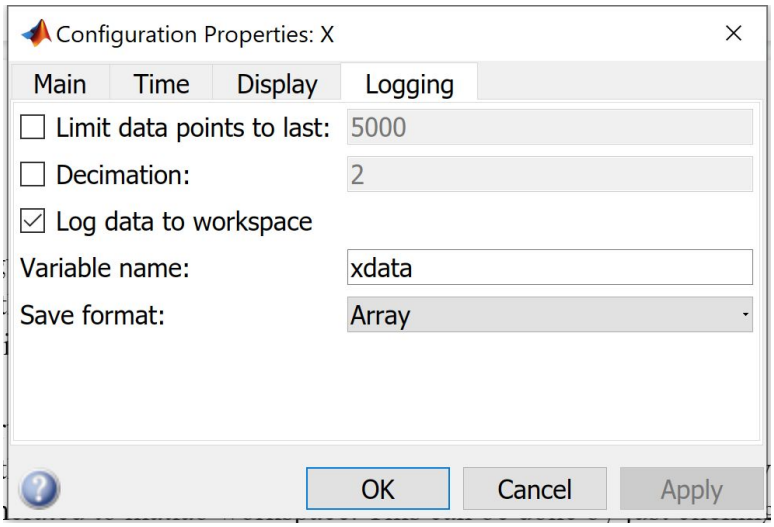

Figure 21: Export data to workspace

After this step click on run. In the matlab workspace the data for x,y, and z will be obtained.

Once the data is obtained. It becomes easier for the student to find the angles. After obtaining the data of x,y,and z. Students need to build a DH table on the robot arm. Be careful in choosing the axis. The axis for DH table can be seen in the figure 22 below. Also the DH table build using the axises can be seen in the figure 23 below

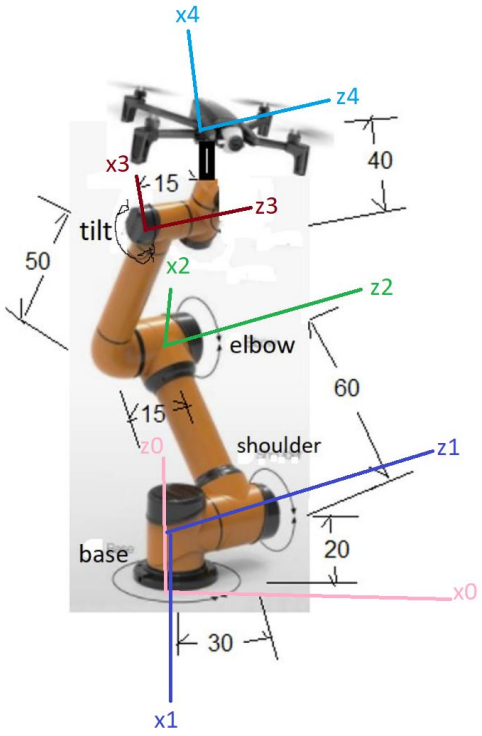

Figure 22: Robot axis

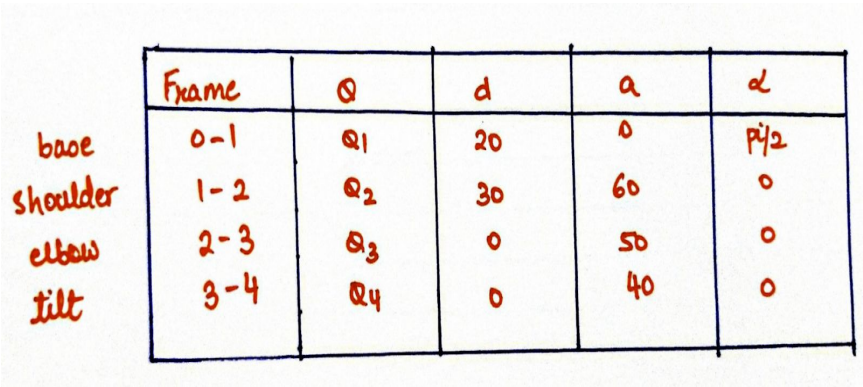

Figure 23: DH table

After this students have to use the inverse kinematics concept of finding the angles. For this part students can use the code implemented for the previous assignment. First start building the DH table in the code. Use the link function to make the dh table. In the code shown in figure 24 below the first column represents the theta. This angle is the rotation around the z axis. Second column represents the distance between two z's. Third column represents the distance between the two x's and the last angle is

the alpha angle that explains about the angle between x's along the current z.

```
%% Part #2: Use Inv Kinematics to find joint angles of the robot
%-------------------------------------------------------------
%Serial Link Structure of Robot
%-------------------------------------------------------------
L1=Link([0 20 0 pi/2],'standard');     % base
L2=Link([0 30 60 0],'standard');       % shoulder
L3=Link([0 0 50 0],'standard');        % elbow
L4=Link([0 0 40 0],'standard');        % tilt

%-------------------------------------------------------------
%Home Position = [pi/2 pi/2 0 0] - robot along x and pointing up
%End-effector will be at [30, 0, 170]
%-------------------------------------------------------------

r = SerialLink([L1 L2 L3 L4]);
```

Figure 24: DH table in code

Connect the links using the serial link function. Then set the home position of the robot arm and guess the angle at which it should be at home position.

```
%-------------------------------------------------------------


r = SerialLink([L1 L2 L3 L4]);
T=[0 0 1 30;0 -1 0 0;1 0 0 120;0 0 0 1];    % home position
guess = [pi/2 pi/2 pi/4 -pi];               % angles guessed

%-------------------------------------------------------------
```

Figure 25: home position and guess angles

Then the next step is to set the degree of freedoms for this robot. Created a different array to store the data. Then set the data extracted to different variables as shown in figure 26 below.

```
%-------------------------------------------------------------
%Initial Parameters
%-------------------------------------------------------------
M = [1 1 1 1 0 0];                    % degree of freedom
x1=[]; x2=[]; x3=[];x4=[];            % store data
x = out.xdata(:,2);                       % x data exctracted
y = out.ydata(:,2);                        % y data exctracted
z = 120 + zeros(1,size(x,1));

%-------------------------------------------------------------
```

Figure 26: Initial parameter

```
for n =1:size(x)
    Ti=r.ikine(T,guess,M,'tol',0.001,'pinv','alpha',1.0);
    x1(n)=Ti(1)*(180/pi);
    x2(n)=Ti(2)*(180/pi);
    x3(n)=Ti(3)*(180/pi);
    x4(n) = Ti(4)*(180/pi);
    guess = [Ti(1) Ti(2) Ti(3) Ti(4)];
    T(1,4)=x(n);T(2,4)=y(n);T(3,4) = z(n);
end
```

Figure 27 : For loop for inverse kinematics

Then create a for loop starting from one to the size of x. Then use the inverse kinematics command to find the angles as shown in the figure 27 above. Change the angles from radians to degrees. After implementing the for loop for inverse kinematics, plot the graphs. Using the commands shown below plot the path of the quadcopter in x, y and z plane.

```
%-------------------------------------------------------------
%Plot trajectory and joint angles from inverse kinematics
%-------------------------------------------------------------
subplot(1,3,1)
plot3(x,y,z,'LineWidth',2.5);
title('Path of quadcopter in x-y plane');
xlabel('x'); ylabel('y');zlabel('height(z)');
xlim([0 35]);ylim([0 35]);zlim([119.99 120.01]);

subplot(1,3,2)
plot(x1,'LineWidth',0.5);hold on;
plot(x2,'LineWidth',0.5); hold on;
plot(x3,'LineWidth',0.5); hold on;
plot(x4,'LineWidth',0.5);
title('Joint angles of the robot');
legend('joint 1','joint 2','joint 3','joint 4','Location','northwest');
xlabel('# of points on the quadcopter path');ylabel('joint angle (degrees)');
set(gca,'ytick',(-210:30:210));
grid;
```

Figure 28: Plotting

Also plot the joints of the robot in the same graph. After plotting the graphs the next thing to do is the forward kinematics. This can be seen figure 28.

iii. ***Part 3- forward kinematics***

In this part of the assignment students have to use forward kinematics to locate the positioning of the end-effector of the quadcopter.

```
%-------------------------------------------------------------
%Use joint angles and fwd kinematics to find end-effector pos in base frame
%-------------------------------------------------------------------------
% Q = [30;90;0;180].*pi/180;
Q_fwd = [x1; x2; x3; x4]'.*(pi/180);
x_fwd=[];y_fwd=[];z_fwd=[];

for i=1:size(Q_fwd)
    T = r.fkine(Q_fwd(i,:));
%     if (i==1) disp(T); end
    x_fwd(i) = T(1,4);
    y_fwd(i) = T(2,4);
    z_fwd(i) = T(3,4);
end
% T_new = r.fkine(Q);
% disp(T_new);
subplot(1,3,3)
plot3(x_fwd,y_fwd,z_fwd,'LineWidth',2.5,'color','r');
title('Retraced path using fwd kinematics');
xlim([0 35]);ylim([0 35]);zlim([119.99 120.01]);
xlabel('x');ylabel('y');zlabel('height(z)');
%=================================END=================================%
```

Figure 29: Forward kinematics

In this part make another array for the x,y, and z. Then create a for loop and perform the forward kinematics similar to the assignments done before. Plot the graph for the forward kinematics as shown in the figure 29 above.

4. Data Analysis

i. ***Result of Part 1***

Once the simulink is created as shown in the figure 20, simulink model, run the simulation for 1 second. The following figure shows the result of part 1. These results were obtained using simulink.

By using the equation shown in figure above. The xy graph should plot a half sinusoidal wave as shown in the Figure below. The desired path is x^2+y^2 =30^2 . With initial condition of quadcopter located at (30,0) and flying at a constant height of 120 units. This equation produces the quadcopter path as shown in the figure 30 below.

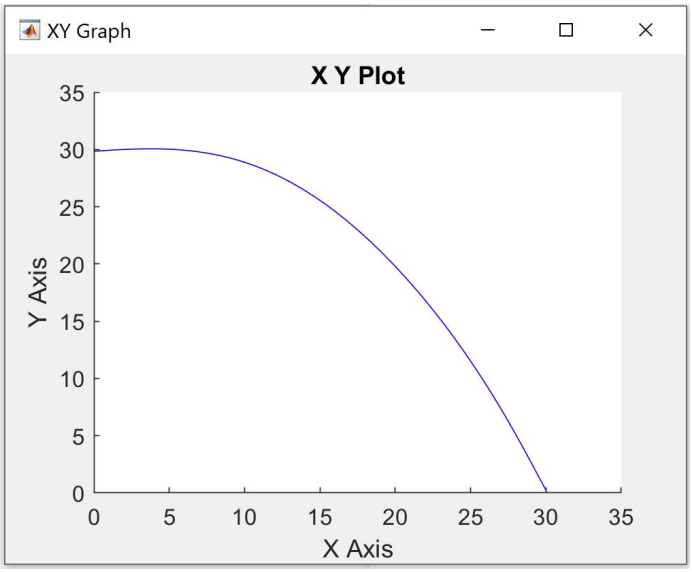

Figure 30: Quadcopter path

In the following figures 31 students will observe the change in the roll angle which shows its movement along the x-axis.

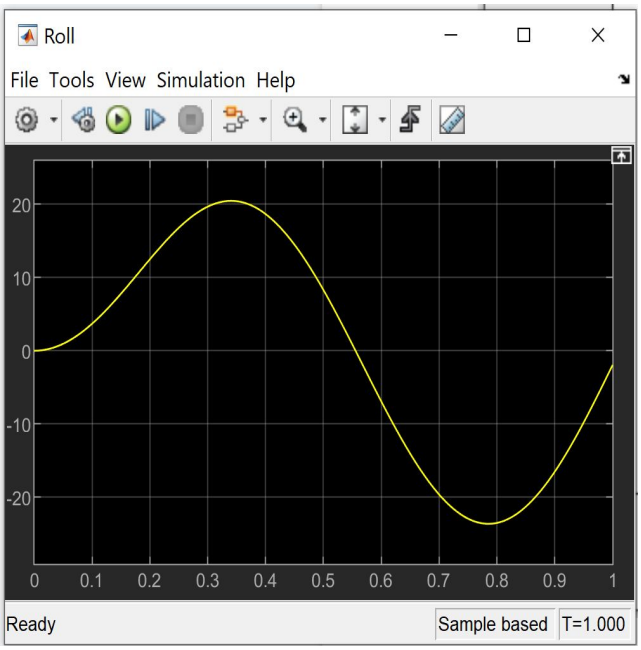

Figure 31: Graph for Roll

In the figure 32 shown below is the rotational movement of pitch along the y axis.

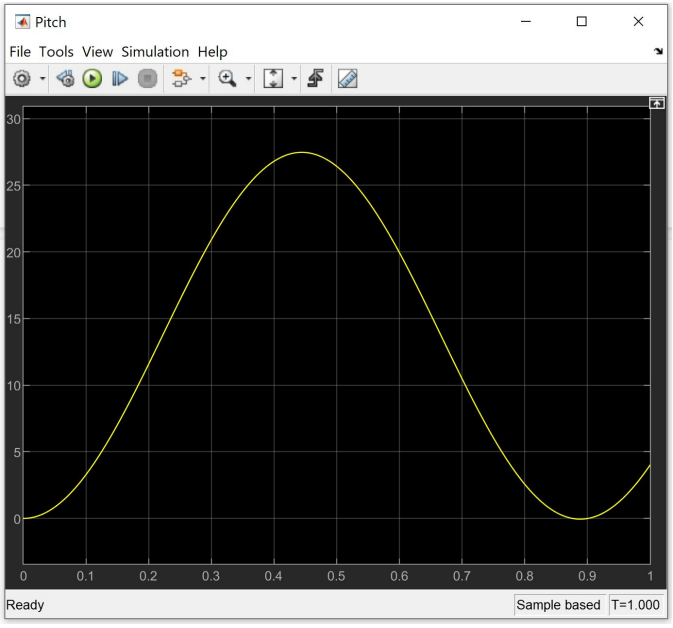

Figure 32: Graph for Pitch

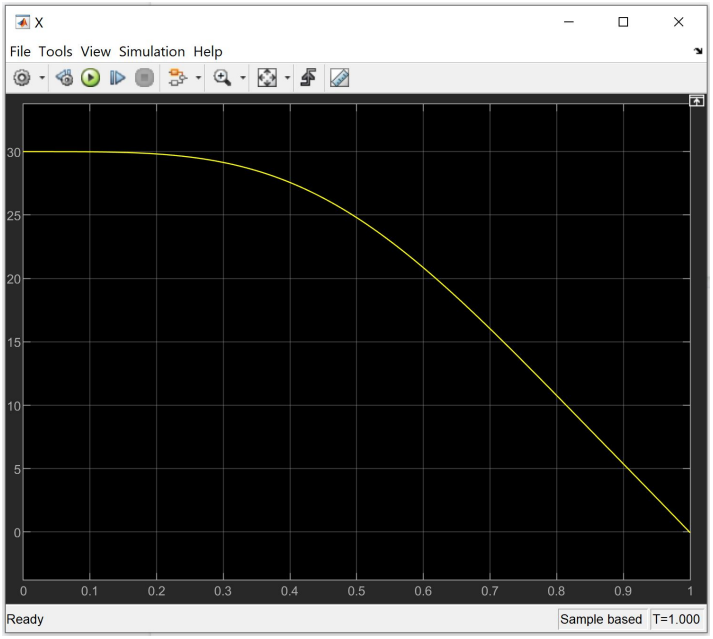

Figure 33: Graph for X

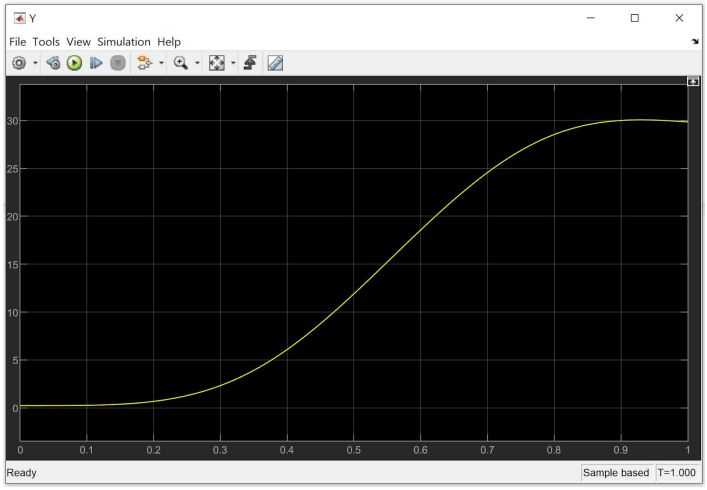

Figure 34: Graph for Y

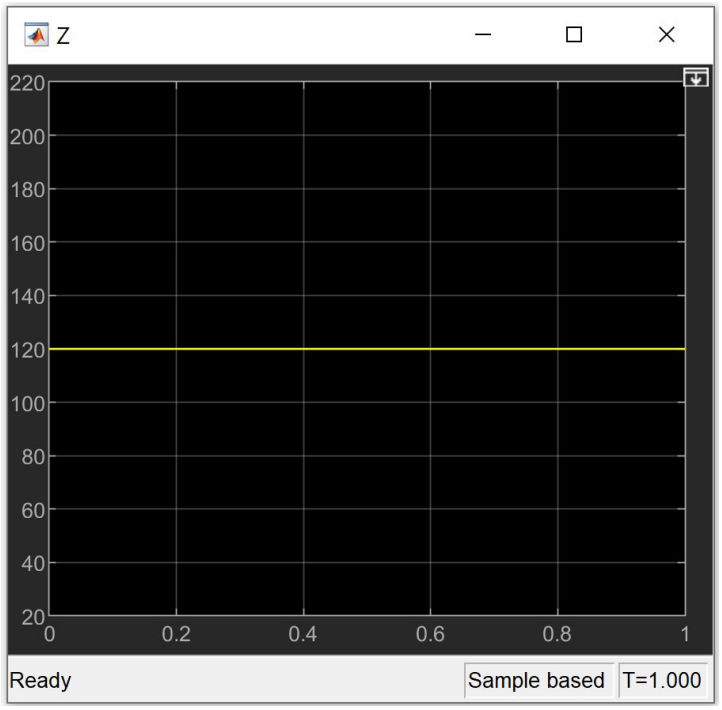

Figure 35: Graph for Z

The graph of z is a constant as it is flying at a fixed height of 120 units.

ii. ***<u>Result of Part 2</u>***

In this part of the report results of inverse kinematics are shown from figure 36 to figure 37. The figure 36 represents the path of the quadcopter in the x-y plane and figure 37 represents Joint angles of the robot. To get the desired result the x,y, and z data was imported from the simulink. Once the data was imported into worked space. It was used in the inverse kinematic equation as explained in the figure 27.

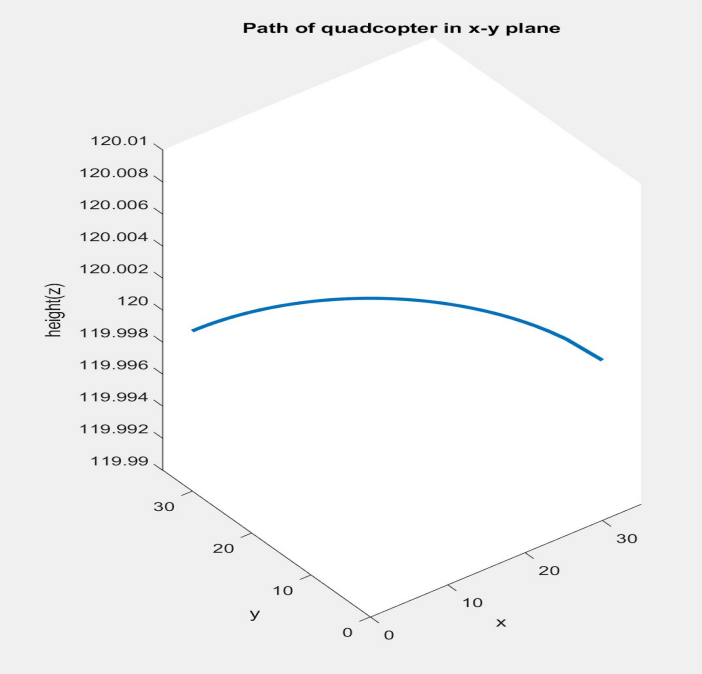

Figure 36: Result for inverse kinematics

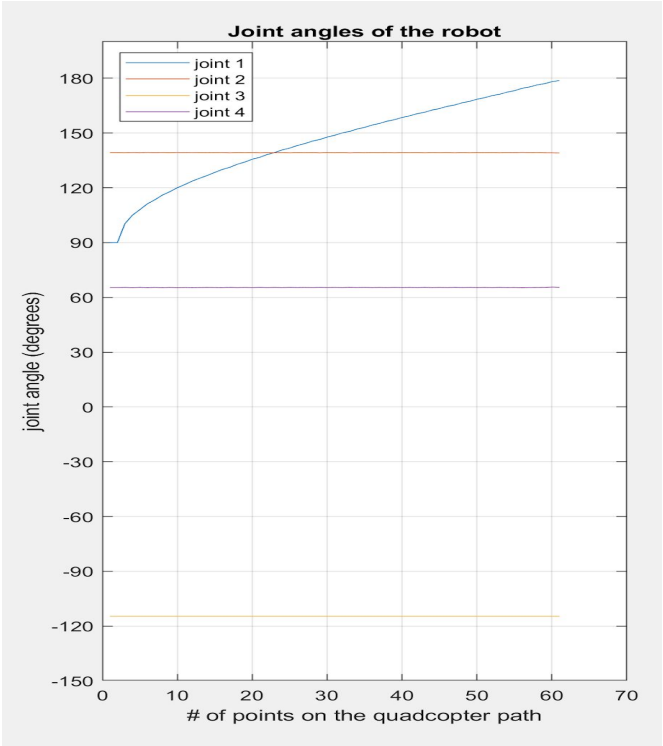

Figure 37: Result for joints plotted

iii. Result of Part 3

Figure 38 represents the result for the forward kinematics. The path is not smooth as the angles found from inverse kinematics are numerical approximations of the actual joint angle values.

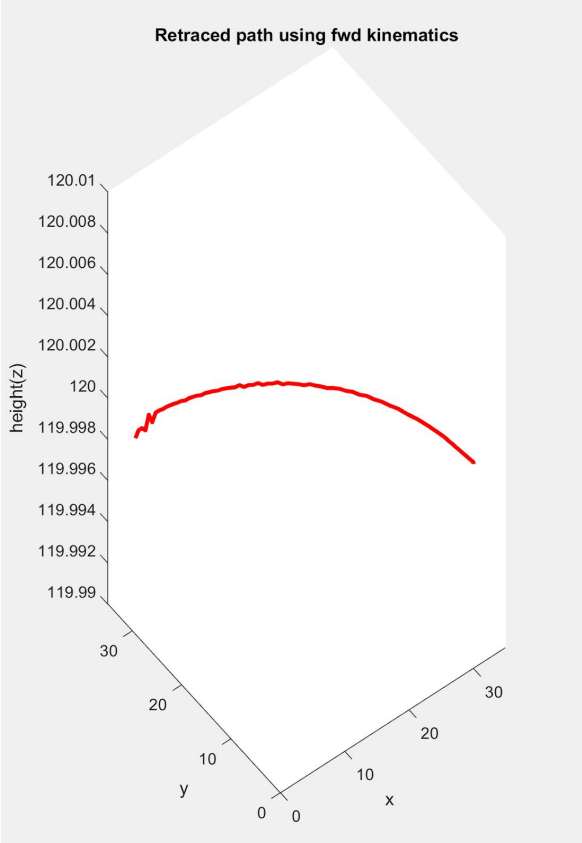

Figure 38: Result for forward kinematics

The following figure 39 shows the combined result of the inverse and forward kinematics.

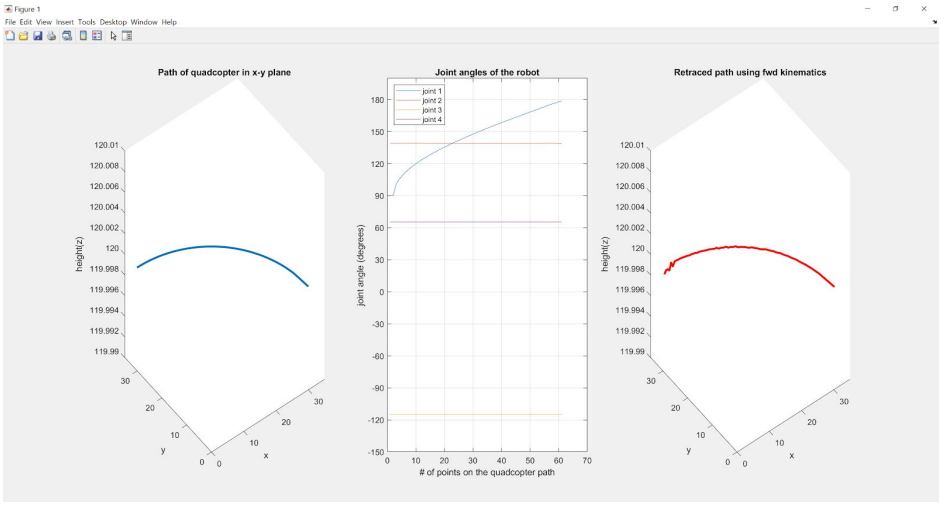

Figure 39 : Final result for inverse and forward kinematics

From this result students can understand the mechanism of inverse and forward kinematics. Using inverse kinematics we obtained the path of the quadcopter in the x-y plane which is similar to the result for forward kinematics.

iv. ***Bonus question***

This part of the report explains if we increase the weight on the center (middle links) of the drone what will be its effects.

For this question it is important to understand the working of a drone. The four propellers running at the same speed each propeller produce a thrust as shown in figure 40.

Each thrust can be calculated using the equation shown in figure 41 below. Where in the equation kf is a component of an aerodynamic force and omega is for rotation. Once force on each propeller is calculated the total thrust force can be calculated using the figure 42 shown below.

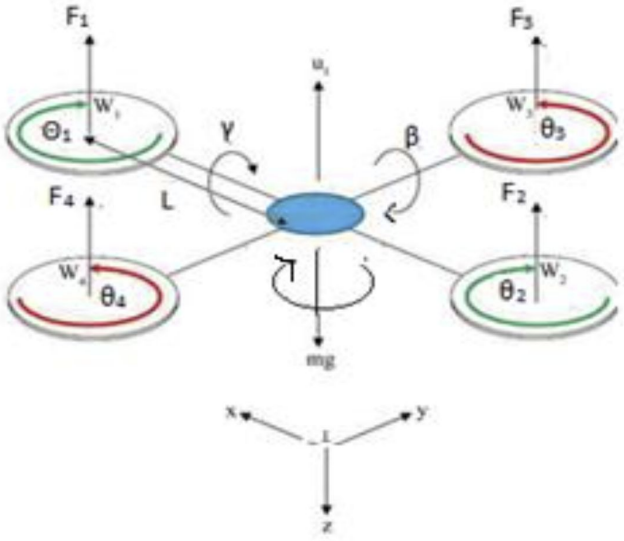

Figure 40 : Increased weight on middle link

Thrust of each propeller: $F_i = k_f \omega_i^2$

where $\omega_i = \frac{d\theta_i}{dt}$

Figure 41: Calculate thrust

Total thrust force:

$$F_{th} = F_1 + F_2 + F_3 + F_4$$

Figure 42: Total thrust

After calculating the total thrust force. If the thrust force is greater than the the mass of quadcopter * gravitational force (i.e mass force). The quadcopter gets lifted vertically.

In this case students have to increase the weight on the quadcopter by accounting the weight of links and sensors on the middle link. This will increase the mass force. To compensate this force students would need to increase the speed of each propeller to a certain limit where thrust force can be greater than mass force. Once the thrust force is greater than the mass force the quadcopter will be able to lift in a vertical direction.

For each joint position it will require an effective thrust to overcome the weight of the links which will change according to the position. Therefore the thrust generated by each propeller has to be actively controlled in order to maintain a constant height and the trajectory planned to follow. As a result this will require a more proper controlled design than the one used for this project.

5. Conclusion

The main aim of this project was to build and test a quadcopter using simulink. Through this assignment students will also get a chance to work with simulinks and explore the different tools, operations, function, etc in simulink. The other parts of this assignment allows the students to understand how to derive the angles of the robot using inverse kinematics and position of the robot using forward kinematics. The overall assignment summarizes the concepts that were learned in the class. This assignment also allows the students to design quadcopter in simulink and then think how different objectives of this assignment can be met. Overall, the experience gained from this project helps the students to get a better understanding of robotics and its fundamental concepts.

6. References

a. "R.U.R." *Wikipedia*, Wikimedia Foundation, 8 Dec. 2020, en.wikipedia.org/wiki/R.U.R.
b. Lecture Notes
c. Previous homeworks

7. Appendix

Appendix A

```
clc;
addpath 'C:\Users\sinha\Documents\MATLAB\Robotics\rvctools\rvctools'
startup_rvc

%% Part #2: Use Inv Kinematics to find joint angles of the robot
%-------------------------------------------------------------
%Serial Link Structure of Robot
%-------------------------------------------------------------
L1=Link([0 20 0 pi/2],'standard');     % base
L2=Link([0 30 60 0],'standard');       % shoulder
L3=Link([0 0 50 0],'standard');        % elbow
L4=Link([0 0 40 0],'standard');        % tilt

%-------------------------------------------------------------
%Home Position = [pi/2 pi/2 0 0] - robot along x and pointing up
%End-effector will be at [30, 0, 170]
%-------------------------------------------------------------

r = SerialLink([L1 L2 L3 L4]);
T=[0 0 1 30;0 -1 0 0;1 0 0 120;0 0 0 1];    % home position
guess = [pi/2 pi/2 pi/4 -pi];               % angles guessed

%-------------------------------------------------------------
%Initial Parameters
%-------------------------------------------------------------
M = [1 1 1 1 0 0];                  % degree of freedom
x1=[]; x2=[]; x3=[];x4=[];           % store data
x = out.xdata(:,2);                   % x data exctracted
y = out.ydata(:,2);                    % y data exctracted
z = 120 + zeros(1,size(x,1));

%-------------------------------------------------------------
%Find inverse kinematic solution
%-------------------------------------------------------------
for n =1:size(x)
   Ti=r.ikine(T,guess,M,'tol',0.001,'pinv','alpha',1.0);
   x1(n)=Ti(1)*(180/pi);
   x2(n)=Ti(2)*(180/pi);
   x3(n)=Ti(3)*(180/pi);
   x4(n) = Ti(4)*(180/pi);
```

```
   guess = [Ti(1) Ti(2) Ti(3) Ti(4)];
   T(1,4)=x(n);T(2,4)=y(n);T(3,4) = z(n);
end

% disp(x1(1));disp(x2(1));disp(x3(1));disp(x4(1));
%-------------------------------------------------------------
%Plot trajectory and joint angles from inverse kinematics
%-------------------------------------------------------------
subplot(1,3,1)
plot3(x,y,z,'LineWidth',2.5);
title('Path of quadcopter in x-y plane');
xlabel('x'); ylabel('y');zlabel('height(z)');
xlim([0 35]);ylim([0 35]);zlim([119.99 120.01]);

subplot(1,3,2)
plot(x1,'LineWidth',0.5);hold on;
plot(x2,'LineWidth',0.5); hold on;
plot(x3,'LineWidth',0.5); hold on;
plot(x4,'LineWidth',0.5);
title('Joint angles of the robot');
legend('joint 1','joint 2','joint 3','joint 4','Location','northwest');
xlabel('# of points on the quadcopter path');ylabel('joint angle (degrees)');
set(gca,'ytick',(-210:30:210));
grid;

%% Part #3: Use Fwd Kinematics to verify the solution
%-----------------------------------------------------------------
%Use joint angles and fwd kinematics to find end-effector pos in base frame
%-----------------------------------------------------------------
% Q = [30;90;0;180].*pi/180;
Q_fwd = [x1; x2; x3; x4]'.*(pi/180);
x_fwd=[];y_fwd=[];z_fwd=[];

for i=1:size(Q_fwd)
   T = r.fkine(Q_fwd(i,:));
%     if (i==1) disp(T); end
   x_fwd(i) = T(1,4);
   y_fwd(i) = T(2,4);
   z_fwd(i) = T(3,4);
end
% T_new = r.fkine(Q);
% disp(T_new);
subplot(1,3,3)
plot3(x_fwd,y_fwd,z_fwd,'LineWidth',2.5,'color','r');
```

```
title('Retraced path using fwd kinematics');
xlim([0 35]);ylim([0 35]);zlim([119.99 120.01]);
xlabel('x');ylabel('y');zlabel('height(z)');
%==================================END=================================
===%
```

Simulink rotated for clearity.