UNIVERSITETET I AGDER

Design of a Heave Compensation System with a Redundant Hydraulic Manipulator

by

Magnus Berthelsen Kjelland

May 25, 2011

Supervisors:
Morten Ottestad
Geir Hovland
Michael Rygaard Hansen

Department of Engineering

Faculty of Technology and Science

Preface

This report is the documentation of a master thesis concerning heave compensation in an offshore environment.
The Agder University is financing all expenses as sensors, hydraulic, electrical components and control systems.
The report contains a literature survey over ides of ways to design and control heave compensating equipment.
The kinematic models of the Stewart platform and a hydraulic manipulator arm is studies and simulated.
The hydraulic manipulator's dynamic is simulated with the control system in a heave compensating motion and it's also tested through a real experiment.

A thanks to Eivind for his help with both mechanic and hydraulics in the machine workshop.

The work in this report was done during the spring of 2011.

Abstract

Waves cause problems when operating in big sea because of their constant movement. Many operations, as moving cargo from ship to ship or from ship to a fixed platform, demands stable working conditions, and this is often difficult to achieve when the waves becomes too big.

The use of active heave compensation can greatly reduce this disturbance. To overcome the movement of the waves, different applications can be applied. The 6 degrees of freedom Stewart platform can be used to compensate, as it can compensate for all motions, translative and orientational. It can also be used as a test platform to create artificial waves/movements which can be used to test/develope new heave compensating equipment.

By solving the kinematics for both the Stewart platform and the hydraulic manipulator, creating a dynamic model for the manipulator using simulation software, the machines are analyzed.

They are modeled and simulated in different software as SimulationX and MATLAB/Simulink. A model of a hydraulic manipulator is created to compensate wave motion when we place the manipulator on top of the Stewart platform. The manipulators has a redundant configuration and the properties of this is explore in regards to heave compensation.

A physical experiments shows that the end effector of the hydraulic manipulator follows a straight line as if it heave compensating.

In order to detect changes in position and orientations, an Inertial Measurement Unit is designed. This can detect fast angles changes in roll and pitch due to sensor fusion using an accelerometer and a gyroscope. A solution for removing position drifting by introducing a Spring-Damper-System is also presented. Further research concerning wave prediction may increase the accuracy of measuring position using the accelerometer.

By creating these models and test them in real life, we will have a great tool for further research concerning heave compensation. Our new simulation of compensation indicates that many different applications for wave compensation can be developed and tested using the Stewart platform, which gives a real life testing environment. By having this complete heave compensation laboratory many new ideas and product can be created.

Contents

1. Introduction

In order to compensate for constant change of position caused by waves many different solutions has been invented. A ship mounted crane with a controllable winch can keep a payload compensated by turning the winch back and forth, making the payload standing almost at rest [12] even if big waves are affecting the ship. This is a approach used in many application today, but it may be problematic if the payload is to be transported through air and not in water, due to almost non friction in the air compare to the water. This may cause the payload to swing to the sides if the ship/vessel containing the crane is exposed to roll or pitch.

There has been research in many field of heave compensation. Another application there has been research on is a compensated helideck by use of the Stewart platform [6]. The general use of Stewart platform can be used in many application.
An example of application using the Stewart platform is the 'Heave Compensated Walkway' made by the Netherland based company Ampelmann. This machine uses the Stewart platform to compensate a walkway that goes from one vessel to another [2] and is shown in figure 1.1 where the walkway goes from one ship to another.

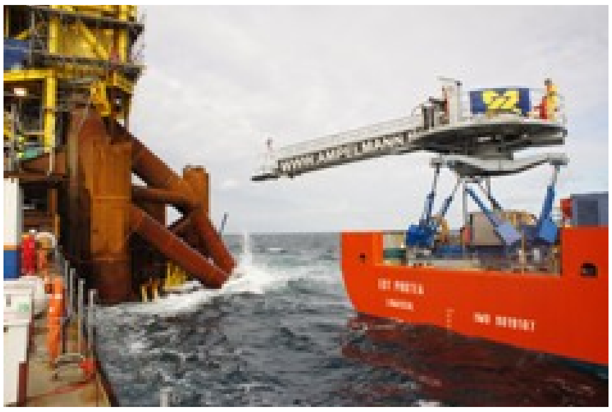


Figure 1.1: *Ampelmann's Heave Compensated Walkway [2]*

Heave Compensation Laboratory

The Stewart Platform, a 6 degrees of freedom (DOF) platform was bought to Agder University by the Norwegian Center of Offshore Wind Energy (NORCOWE) in cooperation with Christian Mikkelsen Research (CMR) as a part of a bigger project concerning offshore wind turbines.
The project is split in work packages were each group is studying different aspects of subjects within the offshore wind energy area. It covers all from design and construction of the turbines as well as installation and maintenance. This project report as a master thesis is a part of the 'Marine Operations' were operations offshore is studied.
In the offshore industry, heave motion created from waves is a problem in offshore handling, drilling and sub-sea operations. Research and knowledge in the heave compensating field of technology is huge benefit

for the industry dealing with there problems.
With the Stewart platform at the university, many different experiments can be done. Testing heave compensation with either manipulator arms or compensating winches are examples of opportunities that comes with the stewart platform.
While testing different heave compensating equipments, the machines are often placed on top of the platform, which gives a good testing environment for simulation wave motion. But i can also be turned around, such as the compensating equipment are mounts on solid ground, while the payload is picked up/placed down on a moving platform.
Ideally there could be used two Stewart platforms, each having different position and velocities to give the ultimate test environment for heave compensating equipments.

Project Description

A crane with an end effector can be controlled to compensate for the motion crated by the waves if the position reference to a fixed point is known. If such a crane is to be designed, a good test environment will give a huge advantage. In this project such an environment will be created having a Stewart platform works as a wave generator while a hydraulic manipulator will try compensate this motion.

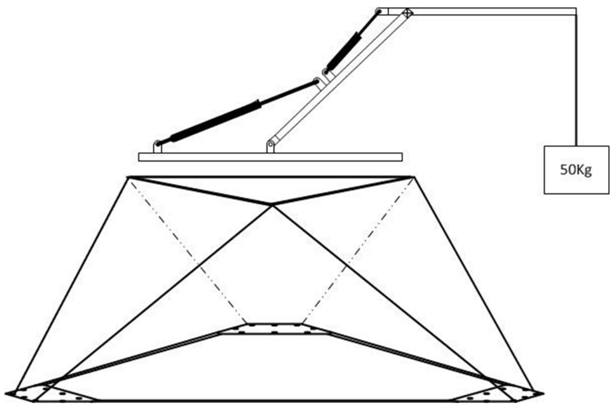


Figure 1.2: *Heave Compensation Environment [13]*

The the mechanical and kinematic aspect of the Stewart platform will be analyzed where the focus will be on the inverse kinematic. An animated model is created to verify the kinematic model.
A redundant hydraulic manipulator will be modeled and simulated, where both the dynamic and kinematics for the redundant and non-redundant configuration. There will also be a physical experiment where the kinematics is tested.
This will show how the pseudo inverse Jabobian solve the problem with the redundancy.
To be able to detect motion and rotation, an inertial measurement unit is analyzed. Problems concerning position drift in introduced and a solution for the problem is proposed. Sensor fusion, combining angle measurement from both an accelerometer and gyroscope is introduced.

2. Stewart Platform

The Stewart Platform 2.1 was shipped from the Netherlands during the autumn of 2010, and arrives at Agder University shortly after.
Since the platform was shipped separately from the control system, some work on the platform was necessary to complete before the it could be used.

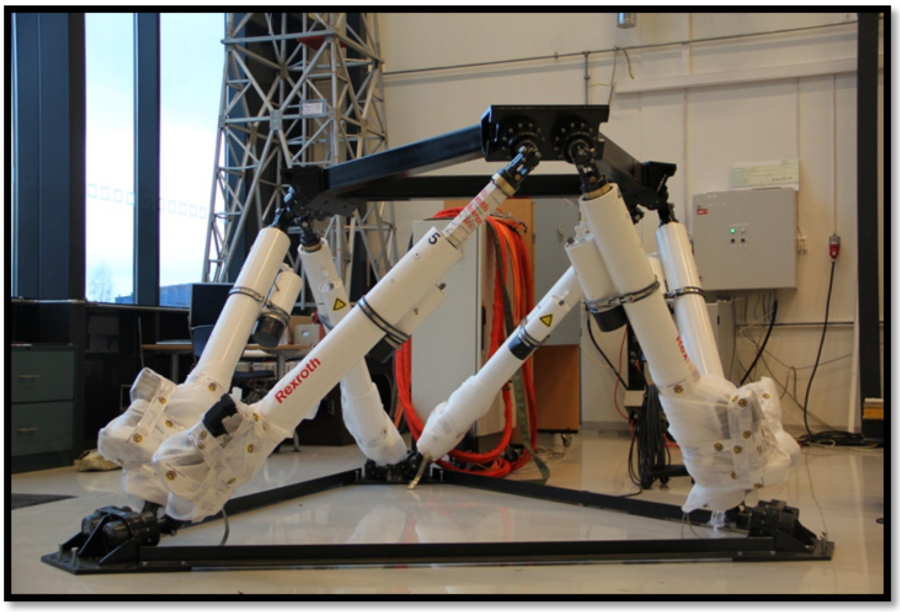


Figure 2.1: *The Stewart Platform in the machine workshop at Agder University*

This machine can move its top platform in 6 degrees of freedom, so all possible motion can be archived with this machine. This motion is moving it translative and rotational in all X-Y-Z axes as in figure 2.2.

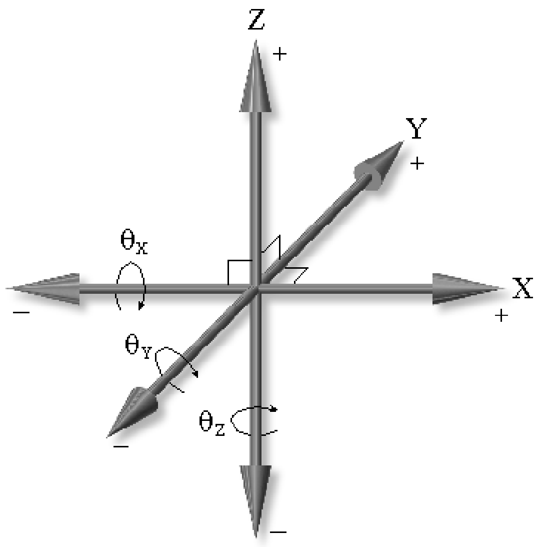


Figure 2.2: *The Six Degrees of Freedom[10]*

2.1 Mechanical Installation

The maximum payload of the platform is 1500kg and it is capable of moving this at a high acceleration. This demands that the platform is firmly connected to the fundament.

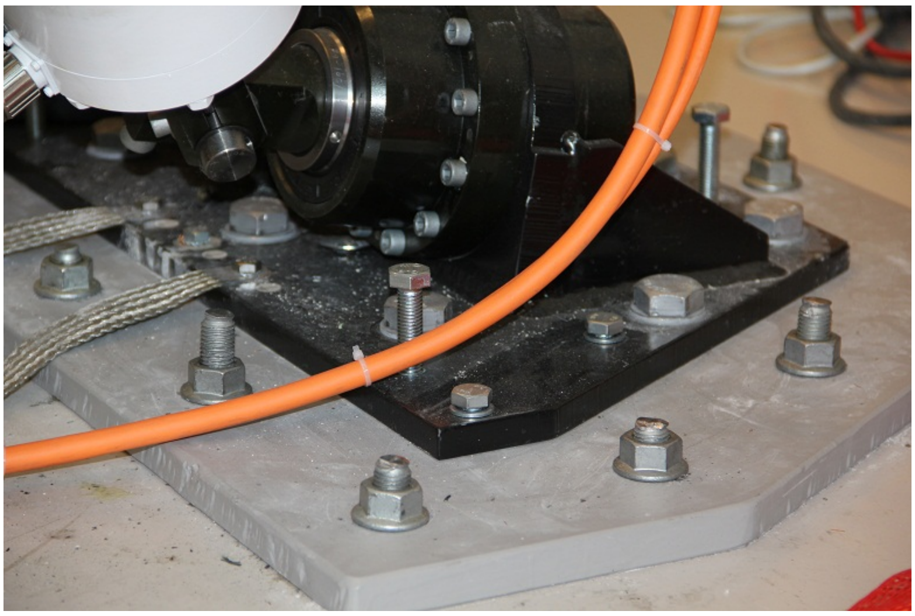


Figure 2.3: *Connection between the Stewart Platform and the Foundation*

2.2 Electrical Installation

The platform was complete with control system and electrical control cabinet. The only remaining job before the platform could be operated was to connect the power and signal cables from each leg of the platform to the control cabinet, and from the cabinet to the control system / PC, figure 2.4.

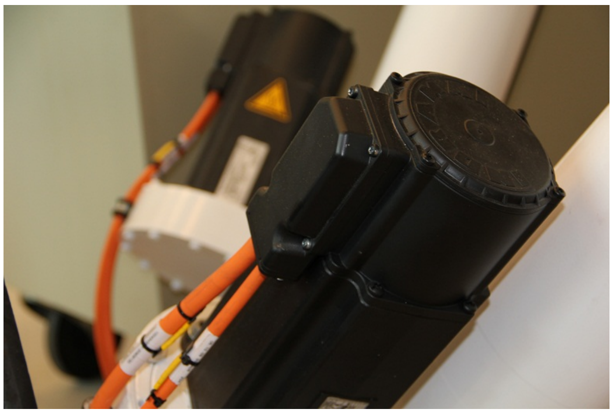


Figure 2.4: *Power and Signal Cable to each Servo Motor*

Emergency and Safety Stop

The Stewart platform is a big machine with quick movements and can cause damage to humans or equipment. Because of this safety aspect, emergency stop functions are included as a part of the control system. It consists of one emergency stop switch and one soft stop switch. By pushing the emergency stop switch, the entire machine loses its power from the control cabinet, making it stop instantly. While moving the machine in full speed, this function shouldn't be used except when there is a chance for human or major equipment damage.
If the machine is in motion and it need to be stopped quickly, but not instantly, the soft stop switch can be used. This switch makes the machine slow down and move to its neutral position.
In figure 2.5 the circuits of the stop functions are showed. The emergency stop has a closed loop from and to the control cabinet, and will execute the emergency stop if the signal in this wire is broken. The soft stop need a power supply signal of 24 volts that need to go into the control cabinet. If it loses the 24 volt signal, the soft stop function is executed.

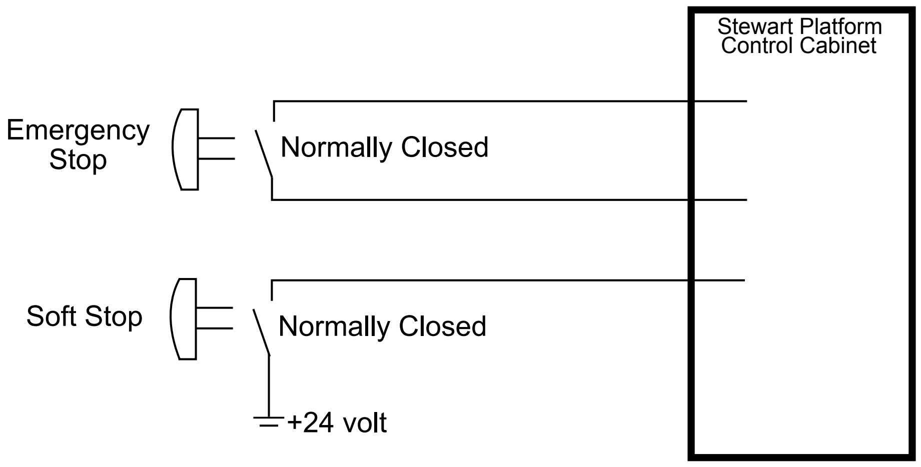


Figure 2.5: *Emergency and Soft stop circuits*

2.3 Inverse Kinematics

The inverse kinematics problem is solved by giving each joint of the Stewart platform a coordinate point in space. The base joints are stationary while the top joint are a function of the translation variables X,Y and Z and the rotational variables roll, pitch and yaw of the platform.
To begin calculating the kinematics it is necessary to find each the position of each joint on both the base and the platform. The length of each of the six legs is based on the base and platform and can be found using pythagoras sentence.

Base and Platform

The base and the platform has the same configuration of the position of the joints, but they differs in size, were the base is much larger than the platform. This is to get good control of the rotation of the platform (pitch yaw) rather than bigger workspace in form of (surge sway).
The position of the joints P and B [1..6] are shown in 2.6.

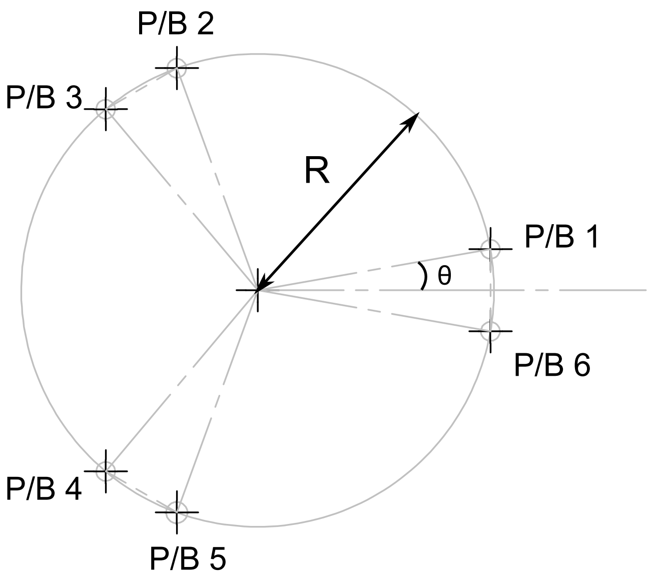


Figure 2.6: *Position of the base and platform joints*

The radius of the circle defines the distance from the center of the platform to joints and is equal for each joint. Every joint pair is located 120 degrees from each other to split them evenly over the circle [11],[7]. There is also an offset angle θ separates the two legs from each other.

Each point as a X and a Y coordinate. With the points lying on the circle it's easy to find its coordinates:

$$P1(X,Y) = \quad (R \cdot \cos(0^o + \theta), R \cdot \sin(0^o + \theta)) \tag{2.1}$$
$$P2(X,Y) = (R \cdot \cos(120^o - \theta), R \cdot \sin(120^o - \theta)) \tag{2.2}$$
$$P3(X,Y) = (R \cdot \cos(120^o + \theta), R \cdot \sin(120^o + \theta)) \tag{2.3}$$
$$P4(X,Y) = (R \cdot \cos(240^o - \theta), R \cdot \sin(240^o - \theta)) \tag{2.4}$$
$$P5(X,Y) = (R \cdot \cos(240^o + \theta), R \cdot \sin(240^o + \theta)) \tag{2.5}$$
$$P6(X,Y) = \quad (R \cdot \cos(0^o - \theta), R \cdot \sin(0^o - \theta)) \tag{2.6}$$
$$\tag{2.7}$$

Figure 2.7 shows the position of both the platform and base joints created in MATLAB.

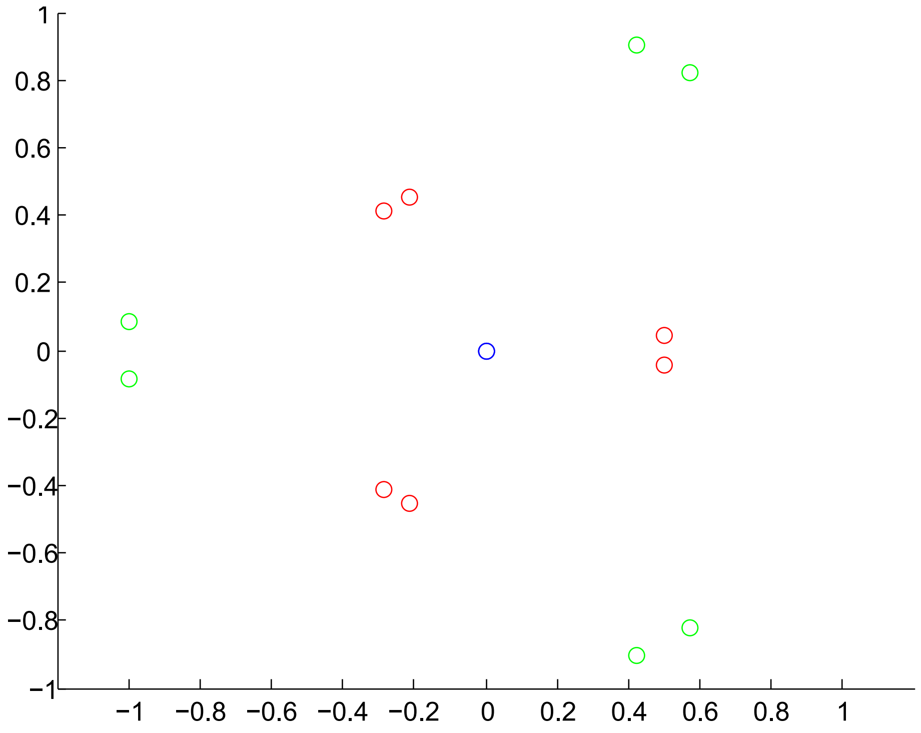


Figure 2.7: *Position of the base and platform joints from MATLAB, Green = Base, Red = Platform and Blue = Origin(0,0)*

A good tool for solving kinematics is the Denavit-Hartenberg Parameters technique [9],[13] which uses a rotational and translational matrix to do transformation from the global to the local coordinate system. The rotational matrix:

$$Rotational = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.8}$$

The translational matrix:

$$Translation = \begin{bmatrix} 1 & 0 & X \\ 0 & 1 & Z \\ 0 & 0 & 1 \end{bmatrix} \tag{2.9}$$

By multiplying in the series can we can build up the kinematics: Transformation Matrix=RotZ*TransZ*TransX*RotX. To get from the origin, which is the center of the base platform the following series of transformation is followed in figure 2.8

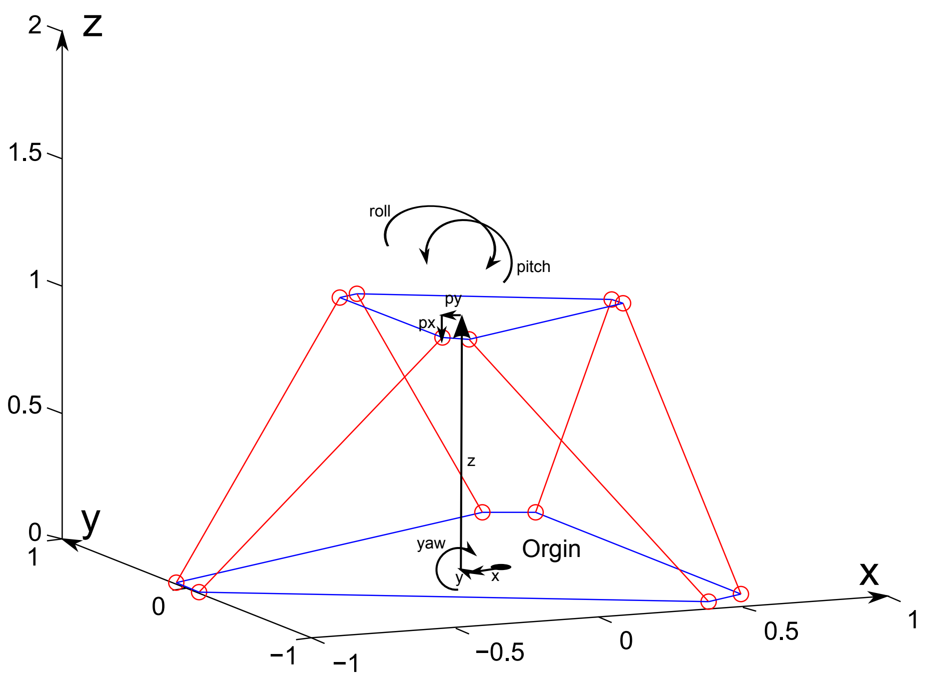


Figure 2.8: *Use of Denavit-Hartenberg Parameters to solve the Kinematics:X->Y->Yaw->Z->Roll->Pitch->Px->Py*

Actuating Legs

To find the length of the actuating legs L_n, where n is the leg number, a simple calculation is performed:

$$L_n = \sqrt{(P_x - B_x)^2 + (P_y - B_y)^2 + (P_z - B_z)^2} \tag{2.10}$$

$$Where:\ n = [1 \dots 6] \tag{2.11}$$

The angle of the legs can be described by:

$$\varphi_n = \tan^{-1}\left(\frac{Py - By}{Px - Bx}\right) \tag{2.12}$$

$$\psi_n = \cos^{-1}\left(\frac{Pz - Bz}{L_n}\right) \tag{2.13}$$

$$\tag{2.14}$$

2.4 Verification of the Kinematics

In figure 2.9 the kinematics for the Stewart platform is verified. A structural animation is created using MATLAB. In this figure the position and orientation of the Stewart platform change, making the platform move.
Only one of the position or orientation variables are change at a time. The default values of the variables are Z=1,X=0,Y=0,Yaw=90^o,Pitch=0^o and Roll=0^o.

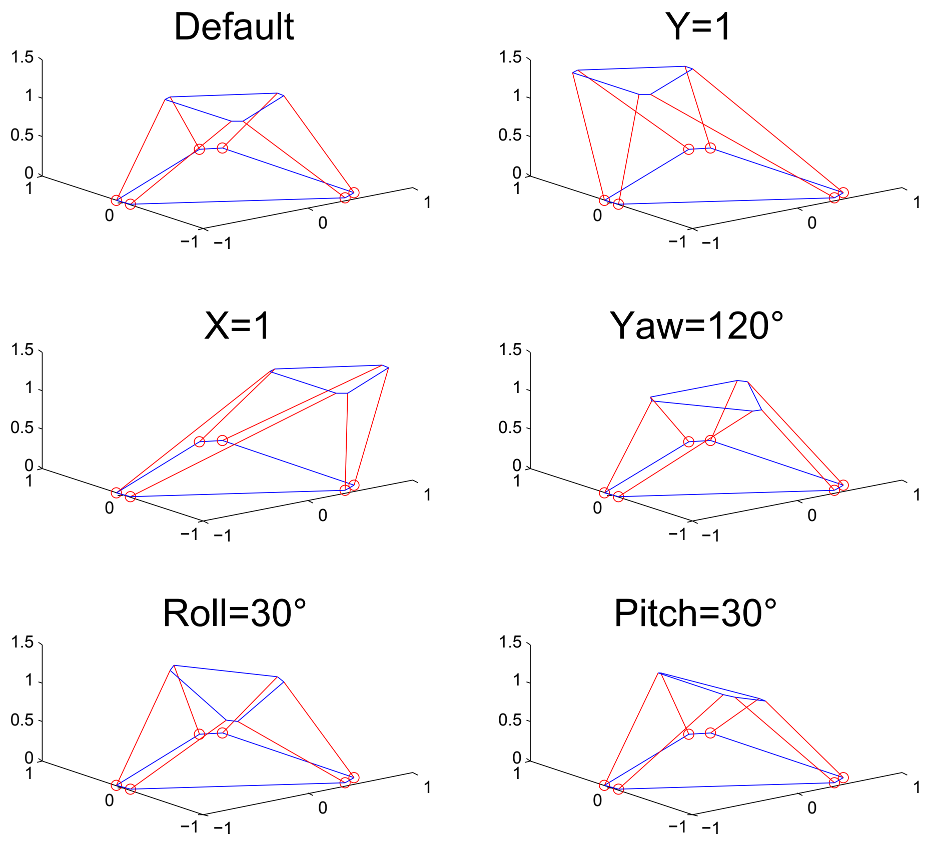


Figure 2.9: *Verification of the Kinematics for the Stewart Platform*

2.5 Control System

There are six servo drives connected to each servo motor. The control system gives each leg a position reference and at the same time controls the acceleration and velocity. The servo drives then moves the leg to the desired position. Each servomotor sends a feedback signal of its position to the drive.

Motion Control

The Stewart platform delivered by Rexroth comes complete with electrical cabinets and control system. The control system runs on PC with real-time Linux and if only connected to the Electrical cabinet by a fiber optical loop going through the six servo drives.
The control system lets the operator choose different ways of controlling the platform. There are two main modes: "Automatic" which controls the platform over Ethernet from a remote pc. Flight simulators can be used on this remote machine and all flight information is sent through Ethernet to the motion control system.
The other mode i "Manual" and lets the operator control the platform either by controlling each leg, or by the degrees of freedom of the platform.
In manual mode, recordings of a movement can also be played and give the platform a pre-defined trajectory.

3. Hydraulic Manipulator

3.1 Background

The hydraulic manipulator arm has been used in many bachelor projects at Agder University.[14] The mechanical and hydraulic has been done, but there were some issues with the electrical cabling that had to be fixed before the machine could be operated.
Some new pressure sensors for the hydraulic oil pressure was also mounted on the machine, and new cables had to be connected the control system.

3.2 Mechanical

The manipulator arm is made by three bars connected with three joints. Two rotational and one translative. Each joint is actuated by a hydraulic cylinder. The joints itself is assumed to have low friction.
The tool point position of this manipulator is redundant only if the X and Y position and not the angle of rotation is used[4]:

$$3\ Bars \cdot 3\ dof - 3\ Joints \cdot 2\ dof - Angle\ of\ rotation\ for\ toolpoint = 2 \tag{3.1}$$

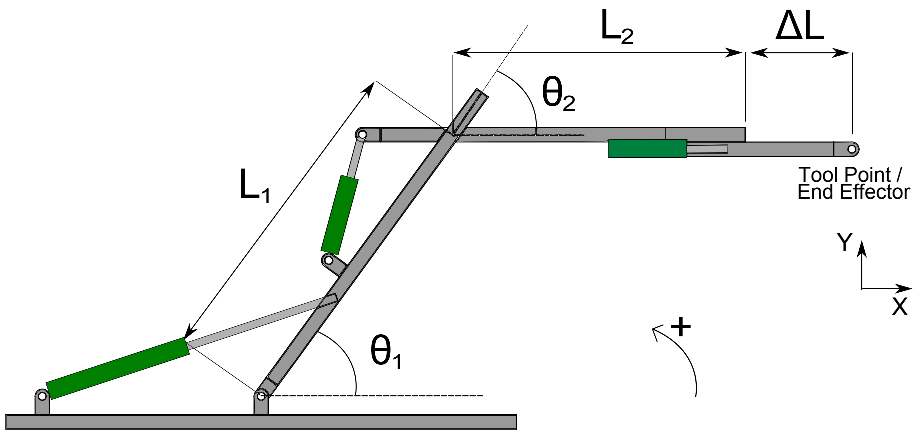


Figure 3.1: *Hydraulic Manipulator Arm*

L_1 *and* L_2 is the length of the two first bars. Since the last actuated bar works as a elongation of L_2, it's denoted as $\Delta\ L$. Making the total length of the bar $L_2 + \Delta L$
The two angles θ_1 *and* θ_2 is the angle between each bar compared to where it is connected. As θ_2 is the angle between the bar L_1 *and* L_2.

Rotation Joint to Translation Cylinder

Since the relationship between the angular velocity and the hydraulic cylinder's velocity isn't linear, a mathematical function is needed on order to move the angles θ_1 *and* θ_2 at the correct velocity based on the cylinder speed.

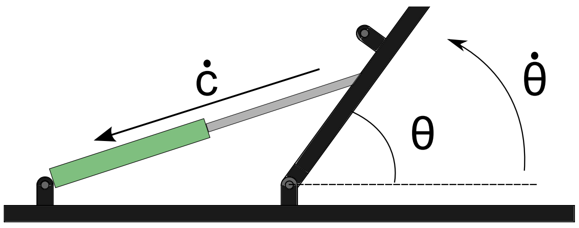


Figure 3.2: *Rotation to translation*

In order to find this relationship the "Cosine Sentence" is used. This sentence describe the lengths and angles of a triangle.

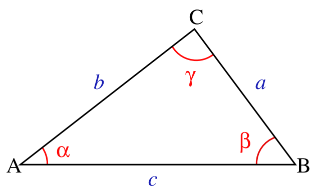


Figure 3.3: *Figure for describing the Cosine Sentence*

Derivation of the length "c" with respect to the angle γ the relationship is found.

$$c^2 = a^2 + b^2 - 2 \cdot a \cdot b \cdot \cos(\gamma) \tag{3.2}$$

$$c = \pm\sqrt{a^2 + b^2 - 2 \cdot a \cdot b \cdot \cos(\gamma)} \tag{3.3}$$

$$\frac{\delta c}{\delta \gamma} = \pm\frac{a \cdot b \cdot \sin(\gamma)}{\sqrt{(a^2 - 2 \cdot \cos(\gamma) \cdot a \cdot b + b^2)}} \tag{3.4}$$

First Cylinder - θ_1

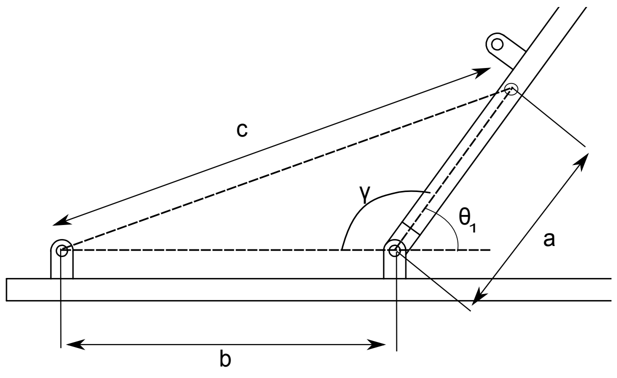


Figure 3.4: θ_1 - *Angular to Translative Velocity*

Since all the interfacing joint are lying on the same horizontal and rotational angle, it can be describes as a perfect triangle. The lengths a and b is measured directly from the machine.

Second Cylinder - θ_2

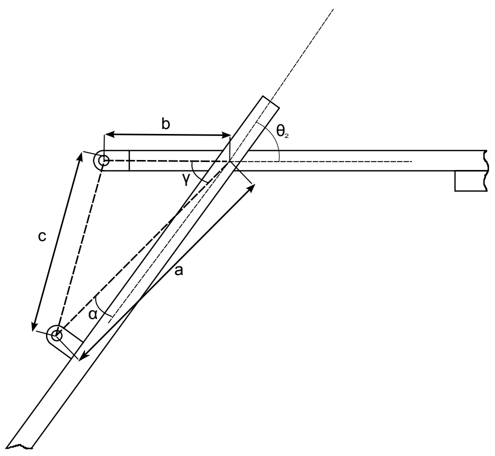


Figure 3.5: θ_2 - *Angular to Translative Velocity*

Here the lower interface point of the hydraulic actuator doesn't lie on the reference angle of θ_2. This causes the offset angle α to be used. The lengths a and b is measured directly from the machine.

3.3 Hydraulics

Servo Valves

The servo valve can electrically be controlled to let hydraulic fluid pas through it by adjusting the area of orifices inside the valve. Because of this the amount fluid passing through the valve can be described by the orifice equation. This equation uses a constant K_v which describes relationship between the opening of the spool inside the valve and the flow that passes through.
The value K_v can be obtained by connecting the valve in the following circuit:

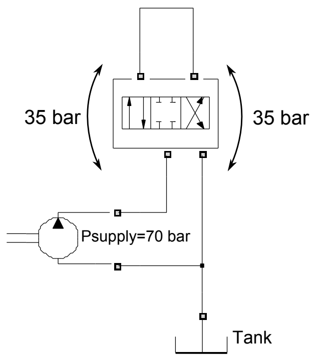


Figure 3.6: *Hydraulic Circuit - No Load Flow*

By supplying the valve 70 Bar and having the valve at full opening the No load Flow can be acquired. K_v can then be found by:

$$K_v = \frac{Q_{NL}}{\sqrt{\frac{Ps}{2}}} \tag{3.5}$$

With K_v known, the equation describing the relationship between fluid flow and opening signal can be set up:

$$Q = K_v \cdot u \cdot \sqrt{P_2 - P_1} \tag{3.6}$$

Hydraulic Cylinders

There are 3 hydraulic cylinders on the manipulator. Two are connected to a revolute joint while the last one i connected to a prismatic joint 3.7.

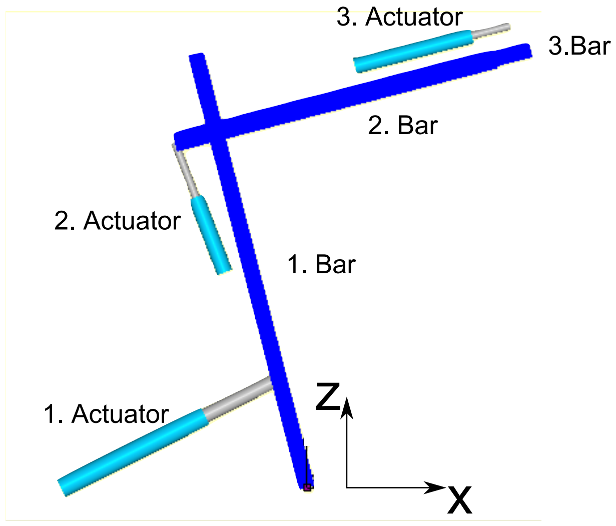


Figure 3.7: *Picture of Manipulator and its cylinders*

Dynamic

The dynamical properties of the manipulator and its components are described by [3]. The following equations describe the properties of the cylinders

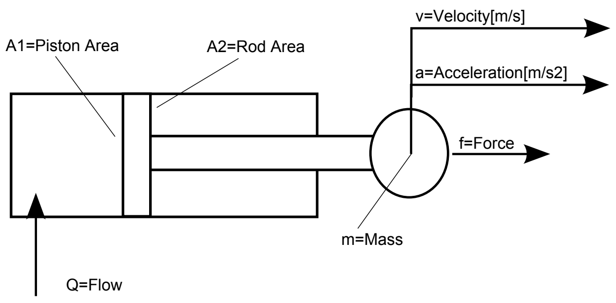


Figure 3.8: *Cylinder Dynamic*

Force:

$$F = m \cdot a = P_1 \cdot A_1 - P_2 \cdot A_2 \tag{3.7}$$
$$F = P_1 \cdot A_1 - P_2 \cdot A_1 * \varphi \tag{3.8}$$
$$F = A_1(P_1 - \varphi \cdot P_2) \tag{3.9}$$
$$Where : \varphi = \frac{A_1}{A_2} \tag{3.10}$$

Velocity:

$$v_1 = \frac{Q}{A_1} \tag{3.11}$$

$$v_2 = \frac{Q}{A_1 \cdot \varphi} \tag{3.12}$$

$$\tag{3.13}$$

Hydraulic Circuit

The hydraulic of the manipulator is connected such as the 'rod side' of the cylinders is directly connected to the pressure side, while the 'piston side' is connected to the servo valve.
Since the area of the 'piston side' is larger than the area of the 'rod side', more force is created on the 'piston side' with the same pressure on both sides. Making the cylinder attract when directing oil flow from the hydraulic pressure unit.
To retract the cylinder, the oil from the 'piston side' is led to the tank though the servo valve. And since there always is oil pressure on the 'piston side' the cylinder will retract.

The configuration of the hydraulic components is connected as in figure 3.9.

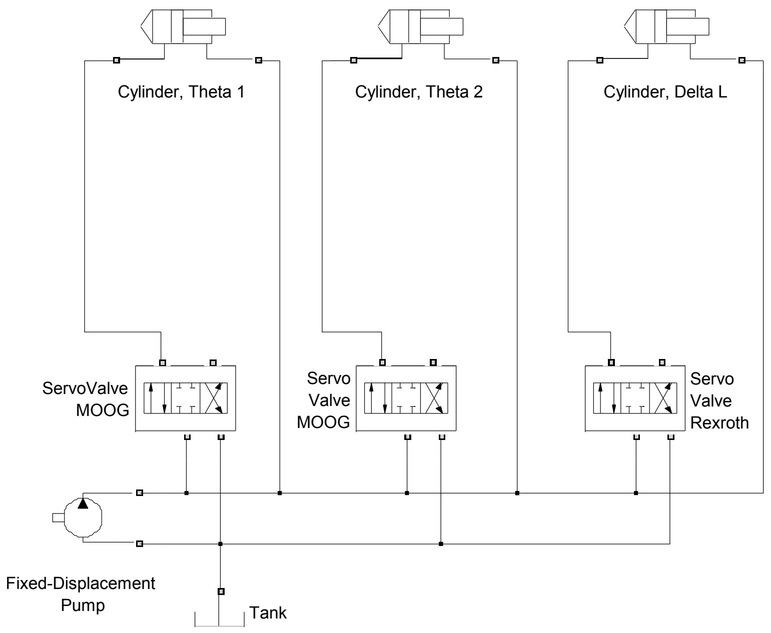


Figure 3.9: *Hydraulic Circuit*

3.4 Kinematics

Without Redundancy

If only the two angles θ_1 *and* θ_2 are used and the displacement ΔL is fixed, making the manipulator arm not redundant. This make it easier to find the forward/inverse kinematics and inverse Jabobian for the system. The forward kinematics can be describes as: "The forward kinematics problem is concerned with the relationship between the individual joints of the robot manipulator and the position and orientation of the tool

or end effector" [15]. The forward kinematics equations will solve the end effector coordinate as a function of the joint variables.
Inverse kinematics will do the opposite. Instead of solving the end effector, it will give a solution for the joint position or orientation as a function of the position or orientation of the end effector.

Forward Kinematics

The DH parameters technique is used to find the forward kinematics. The coordinates for the tip of the manipulator, X and Y:

$$X = (L_2 + \Delta) \cdot \cos(\theta_1 + \theta_2) + L_1 \cdot \cos(\theta_1) \tag{3.14}$$
$$Y = (L_2 + \Delta) \cdot \sin(\theta_1 + \theta_2) + L_1 \cdot \sin(\theta_1) \tag{3.15}$$

Inverse Kinematics

If the tip point of the manipulator is to be controlled, the inverse kinematics needs to be found. The result will be two equations where the two angles will be a function of the tip coordinate[15].

$$\theta_1 = f(X, Y) \tag{3.16}$$
$$\theta_2 = f(X, Y) \tag{3.17}$$

By knowing the coordinates the distance to the tip can be found by using Pythagoras. When this distance is known the angle θ_2 can be found 3.18.
Now there are only one unknown, θ_1, that must be solved 3.19.

$$\theta_2 = -cos^{-1}(\frac{X^2 + Y^2 - L_1^2 - L_2^2}{2 * L1 * L2}) \tag{3.18}$$
$$\theta_1 = \tan^{-1}(\frac{(A \cdot Y - B \cdot X)}{(A \cdot X + B \cdot Y)}) \tag{3.19}$$
$$where : A = L_1 + L_2 \cdot \cos(\theta_2) \tag{3.20}$$
$$B = L_2 \cdot \sin(\theta_2) \tag{3.21}$$

Jabobian Matrix

The inverse Jabobian will tell us what velocity of the two joints we need in order to move the tip at a given velocity 3.29. The Jabobian is a n(number of links) by m(numbers of degrees of freedom of the end effector) matrix that links the relationship for the velocity of the end effector and the joint velocities [15]
The matrix 3.22 is square for non-redundant manipulators and non-square if the manipulator is redundant.

$$J = \begin{bmatrix} \frac{\delta X}{\delta \theta_1} & \frac{\delta X}{\delta \theta_2} \\ \frac{\delta Y}{\delta \theta_1} & \frac{\delta Y}{\delta \theta_2} \end{bmatrix} \tag{3.22}$$

For the hydraulics manipulator in a non-redundant configuration the velocity relationship is a square matrix. While the Jabobian matrix gives a solution of the end effector position velocity, the inverse Jabobian matrix will give a solution of the joint velocity as a function of the end effector velocity. This can be used if the end effector need to follow a line with a certain velocity reference, and it's the inverse Jabobian that is used in 3.23. Since the Jabobian matrix is square, it can easily be inverted.

$$\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} = J^{-1} \cdot \begin{bmatrix} \dot{X} \\ \dot{Y} \end{bmatrix} \tag{3.23}$$

To find the jacobain, the MATLAB function 'Jabobian()' solve is automatically.

$$J^{-1} = \begin{bmatrix} -L_2 \cdot \sin(t_1 + t_2) - L_1 \cdot \sin(t_1), & -L_2 \cdot \sin(t_1 + t_2) \\ L_2 \cdot \cos(t_1 + t_2) + L_1 \cdot \cos(t_1), & L_2 \cdot \cos(t_1 + t_2) \end{bmatrix} \tag{3.24}$$

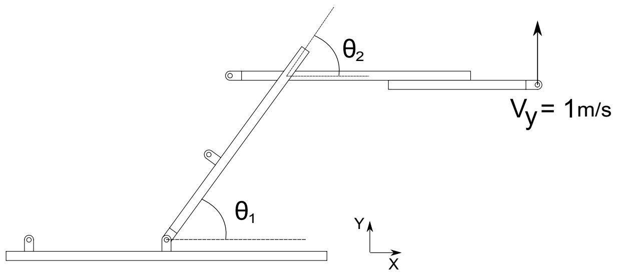


Figure 3.10: *Moving the tool point based on the inverse Jabobian*

If in figure 3.10, the desired velocity is 1 meter per second in Y direction the inverse Jabobian will give the angular velocity of each joint in order to move the tool point at the desired speed.
Substituting the velocity reference for the tool point into 3.29 and initial values for $\theta_1 = 30^o, \theta_2 = -60^o, L_1 = 1 \; and \; L_2 = 1$, the angular velocity of each joint is found:

$$\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} = J^{-1} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.5500 \\ 0.9526 \end{bmatrix} \tag{3.25}$$

Note that the inverse Jabobian is a function of both the tool point velocity and the actual rotation of each joint, $\theta_1 \; and \; \theta_2$. Since the inverse Jabobian is a function of θ and as the manipulator is moving, a new angular velocity for $\theta_1 \; and \; \theta_2$ must constantly be calculated [9].

3.4.1 Redundancy

Forward Kinematics

The end points of the tip of the manipulator X and Y is described by a function of the three actuated joins.

$$X = L_2 * cos(\theta_1 + \theta_2) + \Delta_1 * cos(\theta_1 + \theta_2) + L_1 * cos(\theta_1) \tag{3.26}$$

$$Y = L_2 * sin(\theta_1 + \theta_2) + \Delta_1 * sin(\theta_1 + \theta_2) + L_1 * sin(\theta_1) \tag{3.27}$$

Inverse Kinematics

The inverse kinematics gives the two angles and the displacement based on the tip of the manipulator. Inverse of the forward kinematics.
To solve the redundant inverse kinematics there has to exists three equations for the angle and displacements, since there are three unknowns.

Jabobian

To find the relationship between the velocity of the tip of the manipulator and the velocities of each joint, the Jabobian matrix J is found.

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \end{bmatrix} = J \cdot \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\Delta}_1 \end{bmatrix} \tag{3.28}$$

It consists of the derivative of X and Y depending on the derivative of each joint. Note that in 3.29 the Jabobian matrix is not square. This is a result of the redundancy of the manipulator

$$J = \begin{bmatrix} \frac{\delta X}{\delta \theta_1} & \frac{\delta X}{\delta \theta_2} & \frac{\delta X}{\delta \Delta_1} \\ \frac{\delta Y}{\delta \theta_1} & \frac{\delta Y}{\delta \theta_2} & \frac{\delta Y}{\delta \Delta_1} \end{bmatrix} \tag{3.29}$$

Rather than finding the velocities on the tip of the manipulator based on knowing the velocities in the joints, the inverse Jabobian matrix will do the opposite. It will tell us what joint velocities we need in order to make the tip move at a desired velocity.
Because of the redundancy of the manipulator, the Jabobian matrix 3.29 isn't square and thus not directly invertible, and the Inverse Jabobian cannot easily be found.

Pseudo-inverse Jabobian

A way to solve the inverse 2x3 matrix to use the Moore–Penrose pseudo inverse Method [5] such as

$$\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\Delta}_1 \end{bmatrix} = J^{\dagger} \cdot \begin{bmatrix} \dot{X} \\ \dot{Y} \end{bmatrix} \tag{3.30}$$

Where $J^{\dagger}$ is the pseudo inverse of J. The pseudo inverse method can be used even if the matrix isn't square or not in full row rank. It is not an exact solution to inverse Jabobian, but it will try to give the best solution to equation 3.28
The pseudo inverse Jabobian can be found by:

$$J^{\dagger} = J^T \cdot (J \cdot J^T)^{-1} \tag{3.31}$$

Damped Least Square Jabobian

Another option is to use the DLS method [5]. This method finds the values of $\dot{\theta}_1, \dot{\theta}_2, \dot{\Delta}_1$ that minimize the function:

$$||J \cdot \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\Delta}_1 \end{bmatrix} - \begin{bmatrix} \dot{X} \\ \dot{Y} \end{bmatrix} ||^2 + \lambda^2 \cdot || \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\Delta}_1 \end{bmatrix} ||^2 \tag{3.32}$$

Where λ is non-zero damping constant.
This equation can be rewritten to be:

$$\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\Delta}_1 \end{bmatrix} = J^T (J \cdot J^T + \lambda^2 \cdot I)^{-1} \cdot \begin{bmatrix} \dot{X} \\ \dot{Y} \end{bmatrix} \tag{3.33}$$

4. Inertial Measurement Unit

4.1 Background

In order to measure movement of ships, oil rigs or offshore wind turbines, either a reference data from a solid point or a Inertial Measurement Unit (IMU) is used. Often, no solid reference point is available, so the IMU is a device much used for the purpose of finding position and angle.
The IMU is used in various applications as ROV, UAV, Heave compensation equipment, ship motion monitoring and ocean wave measurement.

4.2 Movements of the Stewart Platform

Since the Stewart platform is a 6 degrees of freedom platform it is able to move in all directions and all orentations. These motions are the same as the motion of what a ship would experience while out on the ocean. While the gravity reference together with an angular velocity sensor will tell us the pitch (ϕ) and roll(θ) angle of the ship, it's more difficult to measure heave motion. By using accelerometers to measure heave motion, equation 4.1 shows the error sources while measuring heave motion.

$$\ddot{x} = a - g\cos\phi \cdot \cos\theta - b - n \tag{4.1}$$

Here the $\ddot{x}$ is the actual acceleration of the ship, b is a constant or slowly changing bias and n is the white Gaussian noise. a is the acceleration measured from the accelerometer mounted vertical from the ship horizontal line.
Since the position is the double integral of the acceleration, much of the high frequency noise is reduced in this process, but due to the bias, the signal will experience drift. This drift can be reduced if a second order high pass filter is introduced [1].
Another approach which further may reduce the effect of this drift is to introduce a center point that the heave motion moves around. In this point the position is zero. To compensate for the drift the position of the ship is introduced to a spring-damper system shown in figure 4.1. A frequency of 0.16 Hz, bias of -0.03 m/s^2, white Gaussian noise of N(0,0.2), spring coefficient at 0.3 N/m and a damping coefficient at 1 Ns/m is set as parameters.

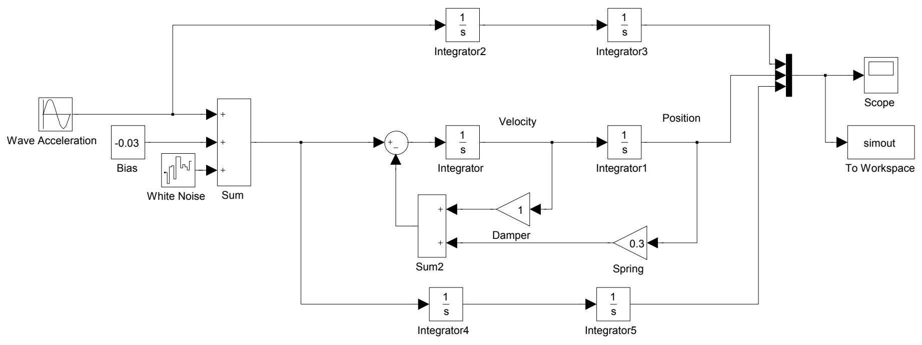


Figure 4.1: *Spring-Damper Compensating System*

By simulation the acceleration and analyzing the results shown in figure 4.2, it's clear that the spring-damper system reduces the drifting problem, but it introduces phase shift for the estimated position.

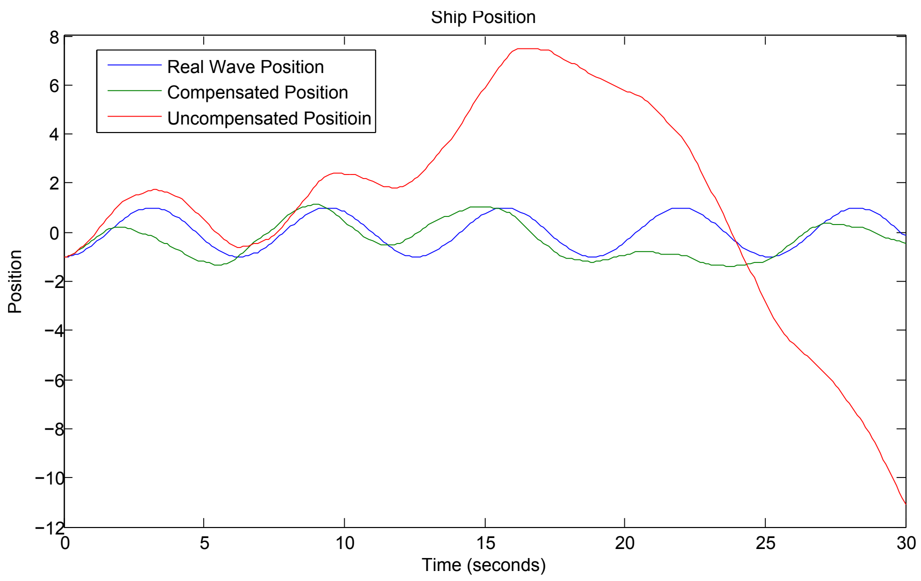


Figure 4.2: *Ship Position, Integrated from Acceleration*

ADIS 16350 High Precision, Tri-Axis Inertial Sensor

A way to measure both movements and rotation is to use a inertial sensor that consists of both a tri-axis accelerometer and a tri-axis gyroscope.
The ADIS 16350 has these properties and is ideal to use as an IMU.
This sensor communicated using the serial peripheral bus SPI and can deliver data from the sensor at a high frequency. The properties of the sensors are:

- $\pm 10g's$
- $\pm 300/150/75 \frac{degrees}{sec}$
- 14-bit resolution

4.3 1. Order Complementary Filter

Accelerometer

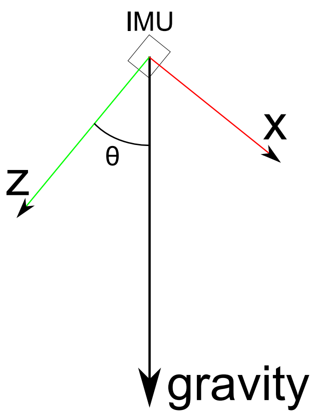


Figure 4.3: *IMU-Measurement of angle using Accelerometer*

Using the vector of the earth's gravity shown in figure 4.3, [17],[16] and [8] the accelerometer can determine the stationary angle of the sensor reference to the earth's gravity field as described in equation 4.2. This gives an accurate measurement, but gives some noise and are not very good at measuring angular velocity.

$$\theta_{accelerometer} = \arctan \frac{-Z}{X} \tag{4.2}$$

Gyroscope

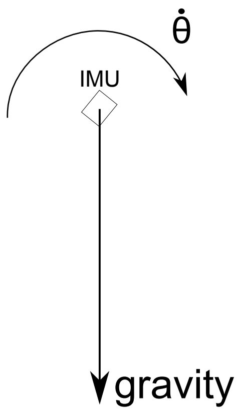


Figure 4.4: *IMU-Measurement of angle using gyroscope*

On the other hand, the gyroscope gives a very fast and accurate measurement of the angular velocity, but if it's used to find the stationary angle, integration of the signal is needed 4.4. This causes the signal to drift because of there will always be a small bias signal that it integrated towards infinity. So seen separately each sensor has its positive and negative sides.

$$\theta_{gyroscope} = \int \dot{\theta} \cdot dt \tag{4.3}$$

Sensor Fusion

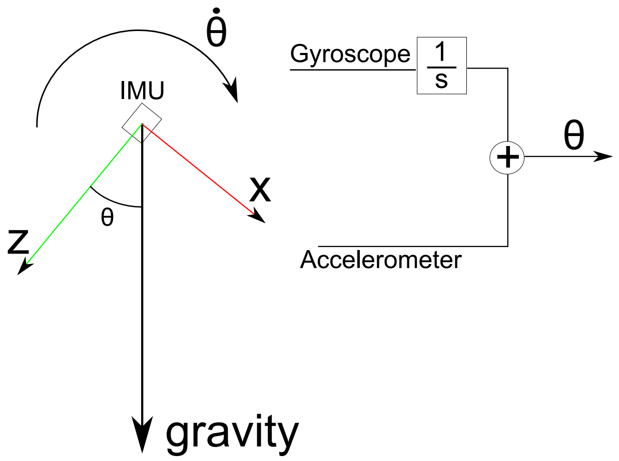


Figure 4.5: *IMU-Complementary Filter*

However, using the positive prosperities of both sensors, they can be combined, making it good on angular velocity as well as giving the stationary angle. To do this, the 1. order complementary filter is used 4.4. This algorithm first integrates the angle measured from the gyroscope, since the gyroscope only measures angular velocity. Then the two angles from the accelerometer4.2 and gyroscope 4.3 are added together with a weighting factor ω. The sum of the weighting factor must be equal to 1 since if more or less would influence the angle making it larger or smaller than the actual angle. Program for the 1. order complementary filter is shown in appendix B.8.

$$\theta_{complemntaryfilter} = \omega \cdot \theta_{accelerometer} + (1 - \omega) \cdot \theta_{gyroscope} \tag{4.4}$$

4.4 Testing the IMU

To verify that the measurement of the angle is correct a small test bench was setup. It consists of the IMU connected to an encoder. This gives a very accurate reference signal that can be used to verify the signal from the sensor. On the rotational axis there is attached a rod, increasing the inertia of the rotating element. The program controlling the inertial sensor is shown in appendix B.7.

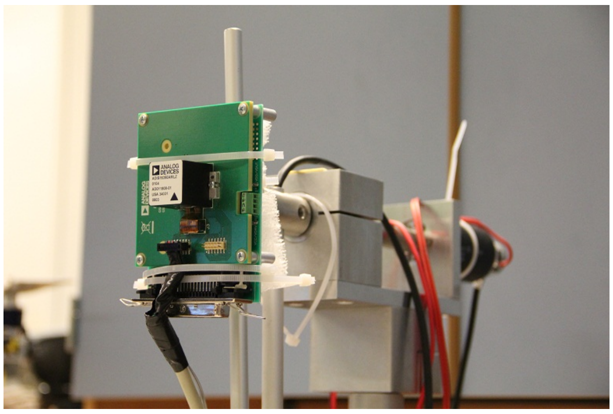


Figure 4.6: *Test Bench for Inertial Measurement Unit*

When using the test bench, the rod is turned so i stand 90 degrees from the pointing downwards. Then it's dropped and will swing back and forth until it's in stable downwards position. By recording the signal from both the encoder and the estimated angle, the sensor can be benchmarked to see how fast it can observe the changing angle.
In figure 4.7 the recorded values are plotted against each other from a 90 degrees fall. The results show that the estimated angle follows the reference encoder angle very good. It is also shown in the plot that the angle settles at approximate 4.7 degrees, only 2.5 degrees error from the encoder angle.

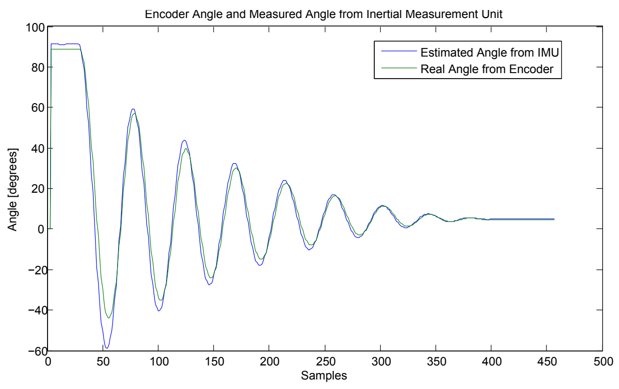


Figure 4.7: *IMU-Experiment, 90 degrees fall*

Another test was done, testing an impact to see the reaction of hit. The data was recorded and is displayed in figure 4.8. As shown in the plot, the estimated angle is not as quick as the encoder.

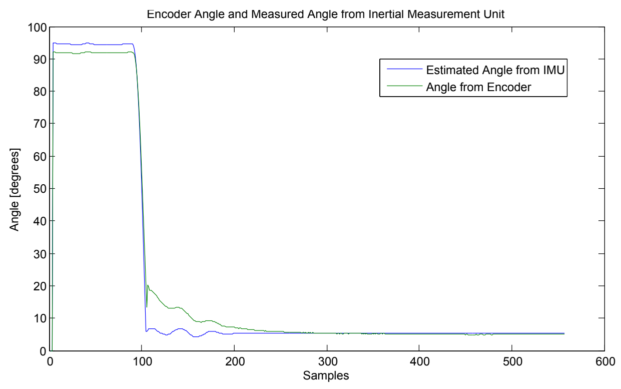


Figure 4.8: *IMU-Experiment, Impact Hit from 90 degrees*

5. Electrical and Control System

5.1 Electrical

Current Supply Circuit

In order to control the hydraulic servo valves, a voltage to current transformer is used. The maximum output current of the National Instruments Analog Output module is limited, and could not be used as a controller directly.
This circuit was designed and created by a bachelor project group in 2010 [14]. It consists of an operational amplifier and resistors in the following circuit:

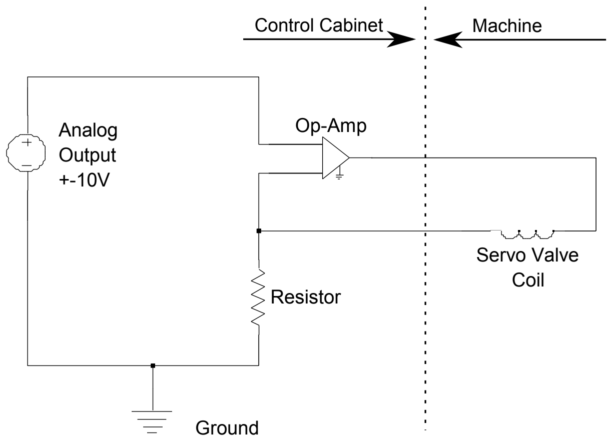


Figure 5.1: *OPMAMP Circuit*

The circuit is controlled by a ± 10 V input signal and gives ± 100 mA output current. To test the linearity and offset of this circuit a digital multi meter was connected in substitute of the servo valve and the current passing through the multi meter was recorded and plotted in 5.2.

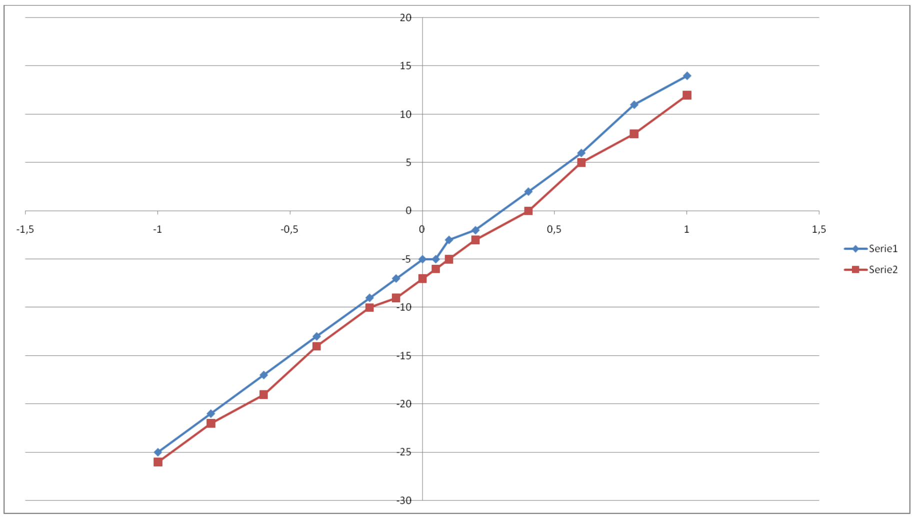


Figure 5.2: *Output Current vs. Input Voltage*

End stop

There are mounted four end stops on the manipulator. This is to protect the structure and cylinders when the manipulator reaches out of its limits. The end stops are connected as digital inputs and prevents the control signal to move the cylinders if the end stop is reached. It's only possible to move the cylinders back again and away from the end stop.

Encoders

To measure the angle of the two manipulator arms, two encoders are mounted on the supporting axels of each arm. The encoder measures the relative angle on the axel on where they are mounted
The encoders are capable of measuring 7200 steps each rotation [14]. This gives the encoders a resolution of:

$$\frac{360\,\text{deg}}{7200} = 0.05\,\text{deg} \tag{5.1}$$

Pressure Sensor

The pressure sensor can measure hydraulic oil pressure in a range from 0 to 400 bars. The output signal is from 0 to 10 volts making 0 bar = 0 volts and 400 bars = 10 volts.

Distance Sensor

To measure the distance of the prismatic joint (appendix D.1), a linear potentiometer is connected to both ends of the joint. The sensors will give a voltage signal linear to the displacement of the prismatic joint described by the simplified equation 5.2 based on the calculation in [14].

$$Displacement_{meter} = Voltage_{analoginputpotentiometer} \cdot 0.0739 \tag{5.2}$$

5.2 Control System

Feed Forward with compensation

The control system gives a control command to the servo vales based on a desired position or velocity of each of the joints on the manipulator. In the P or PI regulator, the exist an error that is the desired set point minus the actual point. This error is then amplified and sent as a control signal to the desired actuator. Because this error is moving toward zero, no command signal will be sent when it finally reaches zero. This causes a problem when controlling the velocity of the actuator. Since the servo valves doesn't give any flow then the control signal is zero it will force the actuator to stop when the control system has zero error on the velocity.

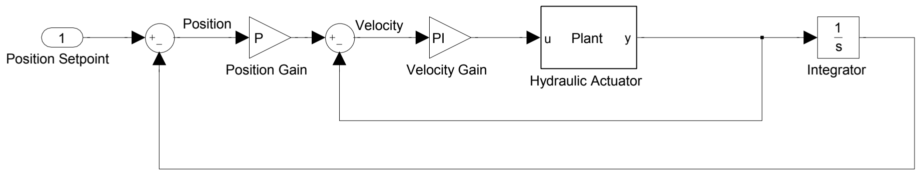


Figure 5.3: *Regulator with P and PI Gain*

To compensate for this, a feed forward signal is used. This signal will not be affected by the actual velocity of the actuator. When combining the two signals the feed forward signal will make the actuator move, while the PI regulated signal will make small adjustment to make the actuator move at the desired velocity.

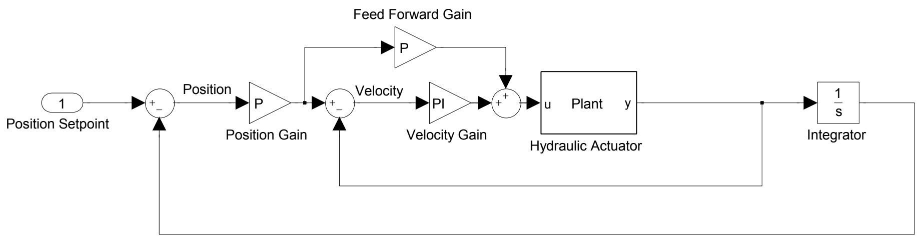


Figure 5.4: *Regulator with P, PI and Feed Forward Gain*

Model Based Control

This way to control uses the theoretical properties of the hydraulic components while measuring real live data in order to know the status of the machine.
It is used to calculate the servo valves opening signal (u) based on what flow we wants to pass through it (Q). The orifice equation states that the flow passing through the valve is dependent of the pressure on both sides (P_1 *and* P_2), the control signal and the characteristics of the valve (K_v). By measuring these two pressures and knowing what flow we want, the control signal can be calculated.

$$Q = K_v \cdot u \cdot \sqrt{P_2 - P_1} \tag{5.3}$$

$$u = \frac{Q}{K_v \cdot \sqrt{P_2 - P_1}} \tag{5.4}$$

Since the flow Q is directly connected to the velocity (v) and area (A) of the cylinders Q can be substituted, making it:

$$u = \frac{v \cdot A}{K_v \cdot \sqrt{P_2 - P_1}} \tag{5.5}$$

The velocity v is can also be substituted with the angular velocity ($\delta\gamma$) of the joints θ_1 *and* θ_2:

$$u = \frac{\pm \frac{a \cdot b \cdot \sin(\gamma)}{\sqrt{(a^2 - 2 \cdot \cos(\gamma) \cdot a \cdot b + b^2)}} \cdot \delta\gamma \cdot A}{K_v \cdot \sqrt{P_2 - P_1}} \tag{5.6}$$

The total regulator will have the model based control signal as an input just before the signal is sent to the actuator. By having a good model based regulator and a good reference position, the output of each of the P and PI gain should lay close to zero. This model is not implemented in the physical experimentation.

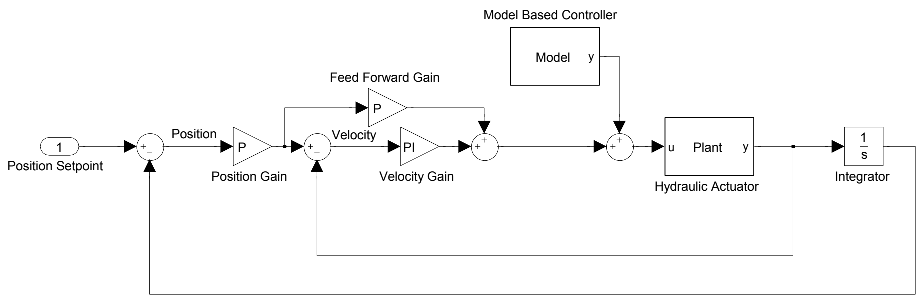


Figure 5.5: *Regulator with Model Based Gain*

5.3 Kinematic Control

Non Redundant

The kinematic control for the non-redundant manipulator is based on the inverse kinematic control. By having a set point for the tool point in the XY-plane the angles for the two joints are found. The velocity reference for each of the two joint can be calculated as an error of the angle set point versus the actual angle of the joint, figure 5.6.

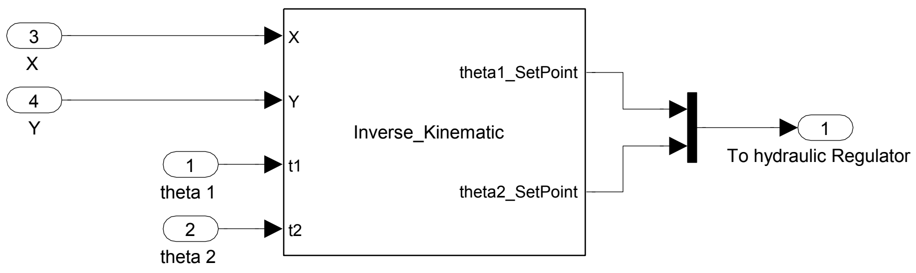


Figure 5.6: *Kinematic Control - Inverse Kinematics*

By including the inverse Jabobian also velocity reference for the tool point can be used. This control will result in a velocity reference of each hydraulic actuator to give the correct tool point velocity, figure 5.7.

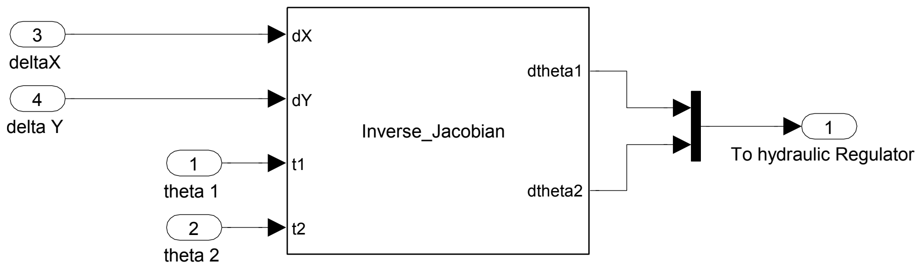


Figure 5.7: *Kinematic Control - Inverse Jabobian*

Redundant Manipulator

There are different ways of creating a control system for the redundant manipulator. In this report, two main solutions are presented.
The first uses a condition based equation to control the redundant actuator. This condition is based on the limits of each joint. The control goes through an algorithm which checks if the joint are within its limitations. If they are, the displacement of the redundant joint is minimized. If not, then the redundant joint is changed to make the two joints go within its limits (appendix A.6).
This procedure is gives a good solution for the redundant manipulator problem, but it will demand more capacity from the control system because there is more program code that has to be processed.
This control has the control topology seen in figure 5.8.

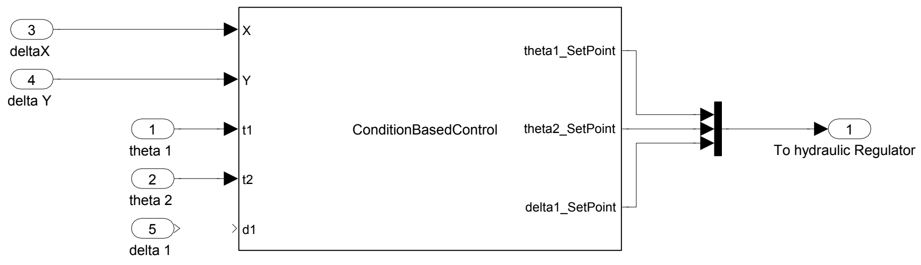


Figure 5.8: *Kinematic Control - Condition Based*

Pseudo Inverse Control

The other solution for the redundant control problem is the Pseudo Inverse Jabobian solution 3.30 and the Damped Least Square method 3.33, (appendix A.5).
These two methods are very similar based on the control topology as seen in figure 5.9.

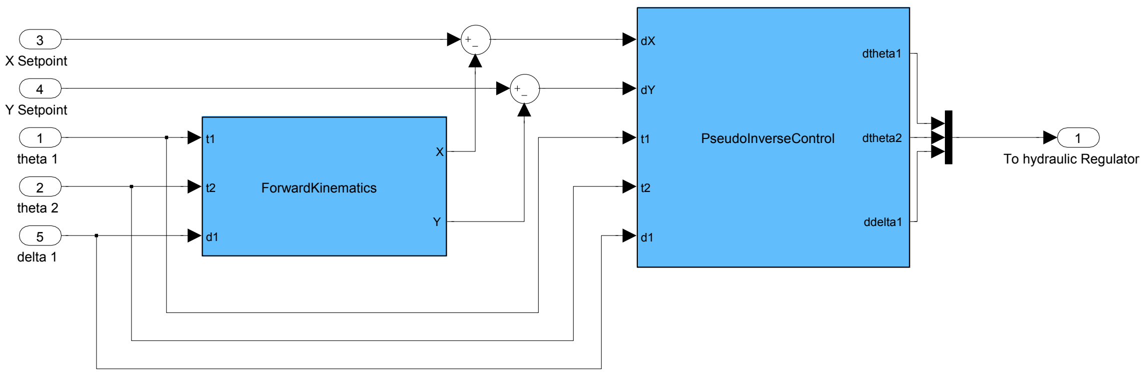


Figure 5.9: *Kinematic Control - Pseudo Inverse Jabobian*

When using the pseudo inverse Jabobian the tool tip position must be calculated from the joint angles and displacement. This is done in the 'forward kinematics' block in the control topology 5.9.

5.4 Real-Time Control System

The control system used to control the manipulator is a CompactRIO delivered by National Instruments. It consists of a chassis which has an can store a controller and 4 input, output modules. The controller is a real-time module NI cRIO-9022 with a built in FPGA controller. 1 analog input NI 9201, 1 analog output NI 9263 and 2 digital input/output NI 9401 modules is installed in the chassis.
This make is a complete control system able to control the whole manipulator and also having an input for the inertial measurement unit. The connection of the control system is described in figure 5.10

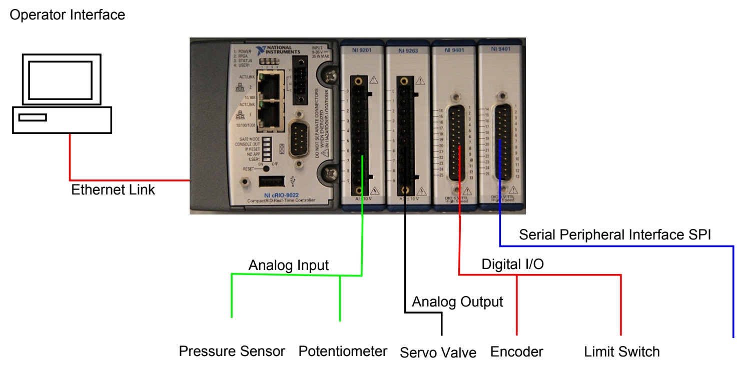


Figure 5.10: *Complete Control System*

The design of the control system was done using LabView. It's also delivered by National Instruments, and is used to create a graphical interface and control for the cRIO and FPGA (appendix B.1 and B.2). The human machine interface created to control the hydraulic manipulator is shown in appendix B.5 and B.4.

6. Modeling and Simulation

6.1 Dynamics - SimulationX

Hydro-Mechanical

SimulationX is a good software to model and simulate dynamical systems. It has a huge library of mechanical, electrical hydraulics, etc. items and has the possibility of running 3D animations.
In this report, the hydro-dynamics of the manipulator arm is simulated.
The model is based on an old bachelor project involving the same manipulator [14]. The dimensions used in this model are the same as in the old one, but a new control system is implemented using inverse kinematics to make the tool point of the arm following a path.

3 solid elements are representing each bar of the manipulator. Attached to each solid element is a joint attached. The hydraulic actuators are connected to each bar using a 'force interface point'. This lets the actuators to be connected at the middle of a bar instead of at the ends.

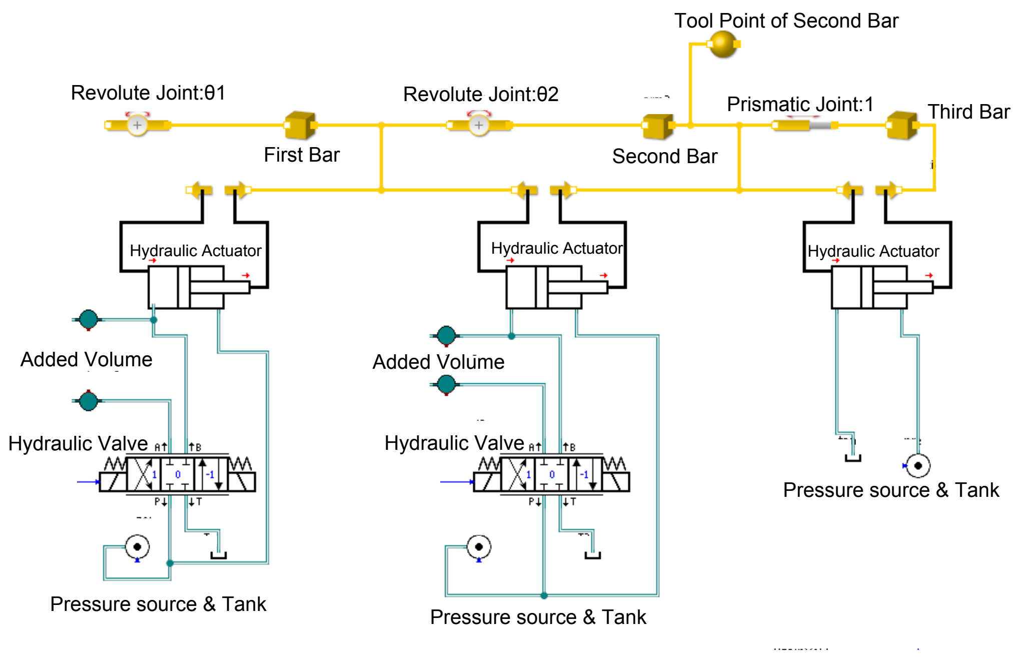


Figure 6.1: *SimulationX - Hydro-Mechanical Mechanism*

Each hydraulic actuator is connected to a proportional directional valve. The last actuator is for simplicity

connected directly to a pressure source. This is because the redundancy isn't modeled in SimulationX, only in Matlab/Simulink.
A pressure supply @ 70 bars is again connected to the P side of the actuator as well as a tank is connected to the T side.
In figure 6.2 the complete manipulators is displayed as a 3D animation.

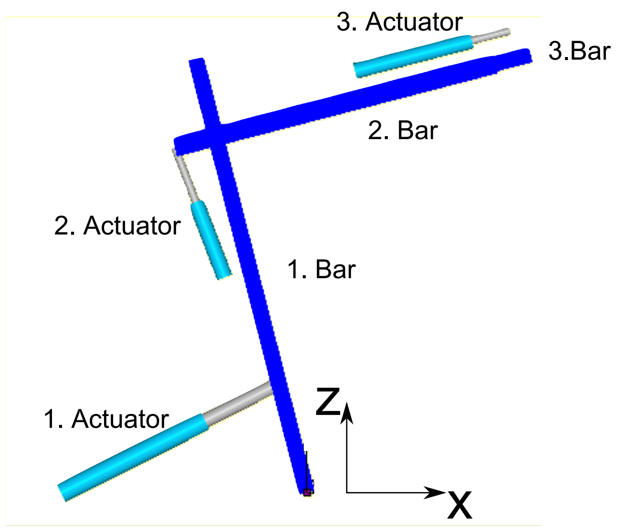


Figure 6.2: *SimulationX - 3D Animation*

Control System

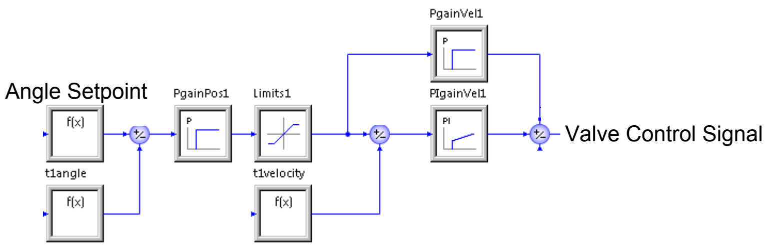


Figure 6.3: *SimulationX - Control System*

The proportional directional valve regulator has the same configuration described in figure 5.4. It used a set point for the angles θ_1 *and* θ_2 which generate a velocity reference if there is an angle error.
The limitation block is used to limit the velocity reference in order to make angular velocity following the reference. If this is not included, this reference would be too high for the manipulator to follow.

The inverse kinematics is included as decried in equations 3.18,3.19,3.20 and 3.21.

Simulation Results - SimulationX

In the simulation, the manipulator is controlled to move the tool point of the second bar in a sinusoid path in the Z axis. In the X axis the tool point is held at stable point. As seen in the figure 6.4 the manipulator tires to follow the Z set point of $1.1 \pm 0.3 meters$ at a frequency of 0.1 Hz while the X set point lies at a constant 0.2 meters.
At the beginning of the plot the manipulator moves from it's initial stating point towards the moving Z path.

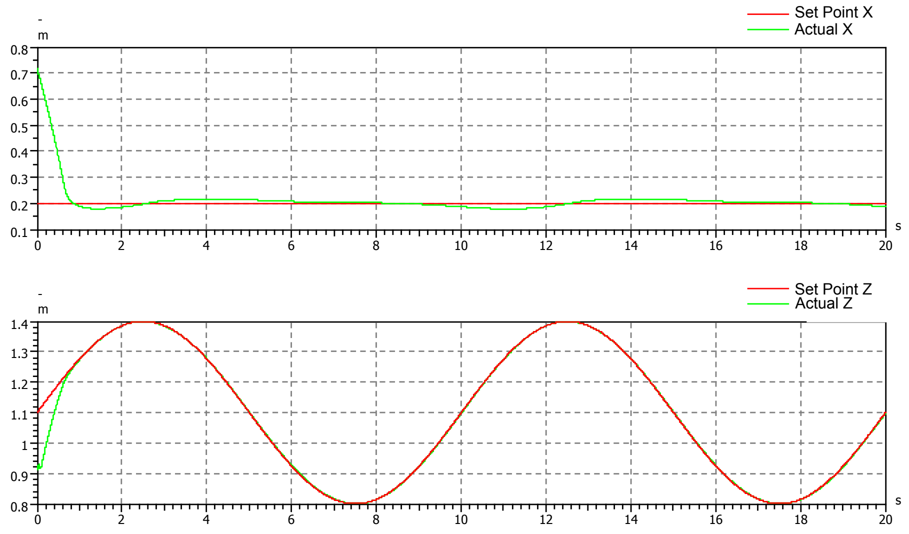


Figure 6.4: *SimulationX Results - Tool Point Set Point vs. Actual Values [m]*

The pressure in the piston side in the actuators is shown in the figure 6.5. The simulation is the same as in 6.4.
Since the pressure supply is directly connected to the rod side will lay at 70 bars. Only the simulation time of 2-20 seconds is included display the stabile pressure.

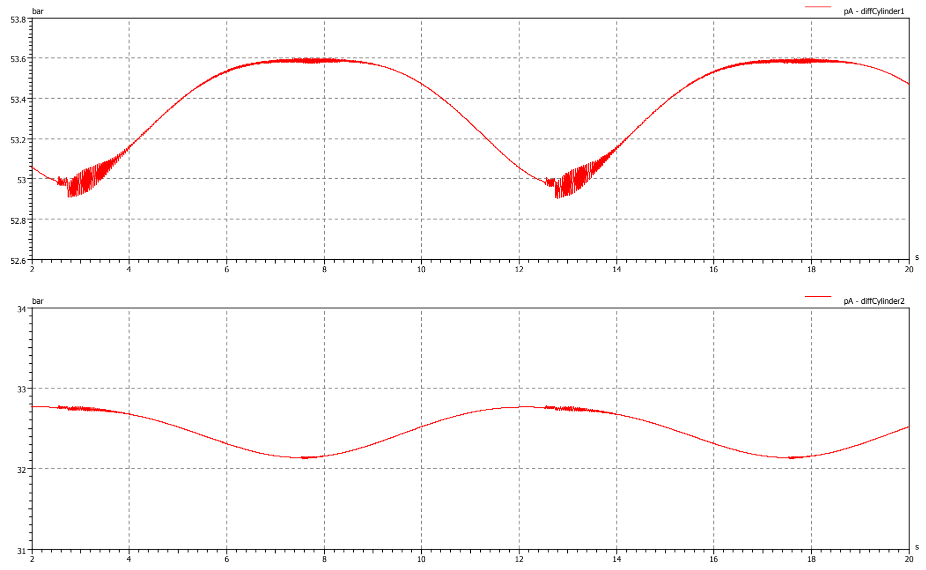


Figure 6.5: *SimulationX Results - Oil Pressure [bar]*

Since there isn't any heavy load at the tip of the manipulator the oil pressure won't increase much before the cylinders start moving. The biggest force to overcome will be the constant 70 bars pressure at the rod side.

The hydraulic oil flow is an important variable in moving the manipulator. The oil flow is plotted in figure 6.6 and are showing the oil flow to and from the two chambers of each of the two controlled cylinders.

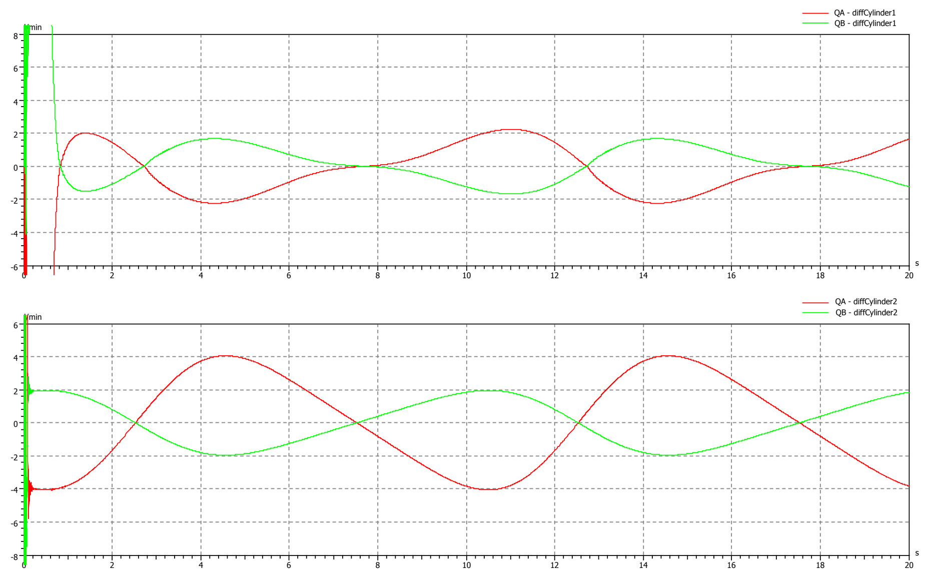


Figure 6.6: *SimulationX Results - Oil Flow [liter/min]*

In order to move the tool point as simulated, the hydraulic power unit that delivers oil to the system must be able to at least give 4+6=10 liter pr minute.

6.2 Kinematics - Matlab/Simulink

To verify the kinematics a model of the hydraulic manipulator is created in Matlab/Simulink. Here the different controls are tested and plotted.

Tool Point Control

Redundancy

The condition based kinematics described in figure 5.8 is modeled and simulated. There are several constraints included while simulating. These constraints limits the angle of the joints and are the same as on the physical manipulator A.6.
$\theta_{1min} = 60^o, \theta_{1max} = 120^o, \theta_{2min} = -120^o, \theta_{2max} = -60^o, \Delta_{1min} = 0.05 meter and \Delta_{1max} = 0.35 meter$

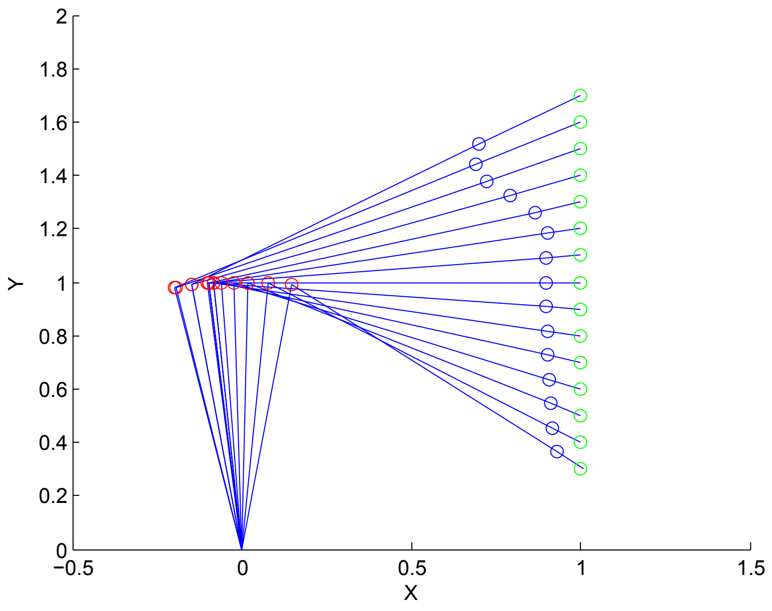


Figure 6.7: *Redundant Control - Condition Based*

Redundancy - Pseudo-Inverse Jabobian

The Pseudo-Inverse Jabobian solution is first tested without any constraints. This means that the two rotational joints can move in 360^o and the translative joint can move from $\pm\infty$. The simulation is done by trying to move the tool point in a circle and is shown in figure 6.8. As the figure shows, the manipulator reaches it's set point without any problems.

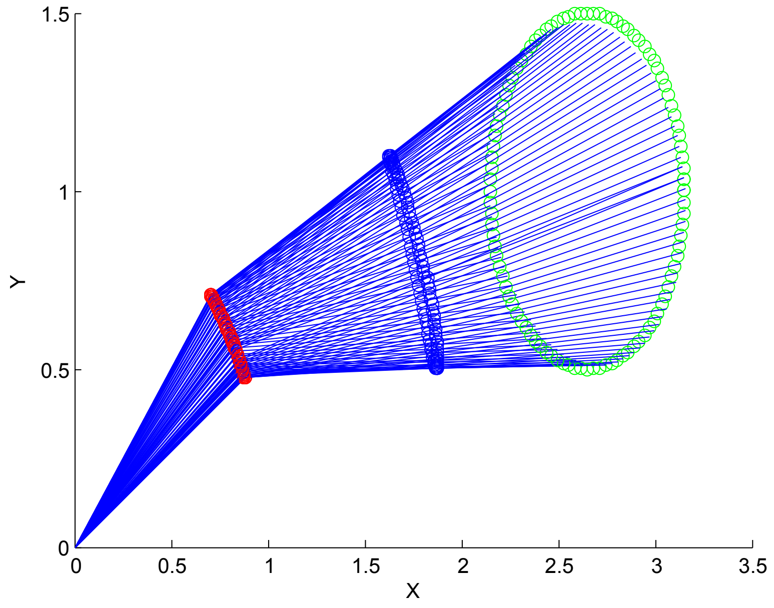


Figure 6.8: *Pseudo-Inverse Jabobian, no Constraints*

When including the constraints, the pseudo inverse Jabobian hits some problems. It shown in figure 6.9

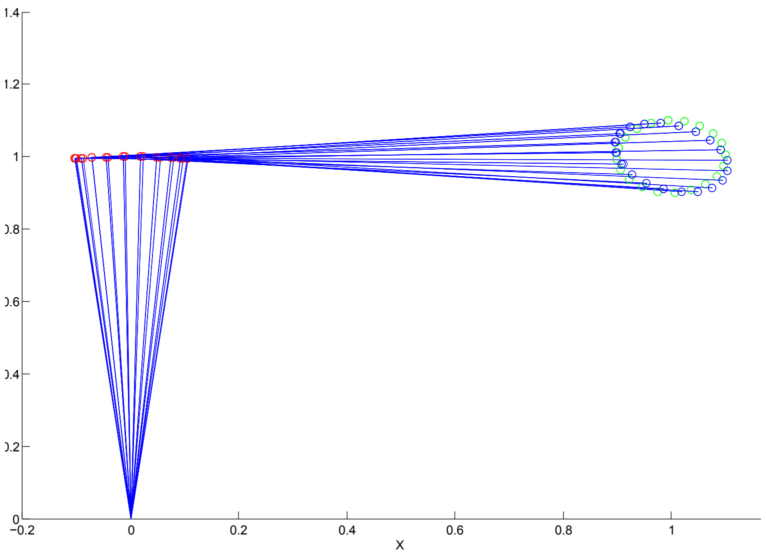


Figure 6.9: *Pseudo-Inverse Jabobian, with Constraints*

To get a better view of the circle the tool point is trying to follow, figure 6.10 shows a zoomed view of the tool point. The end effector can reach its set points.

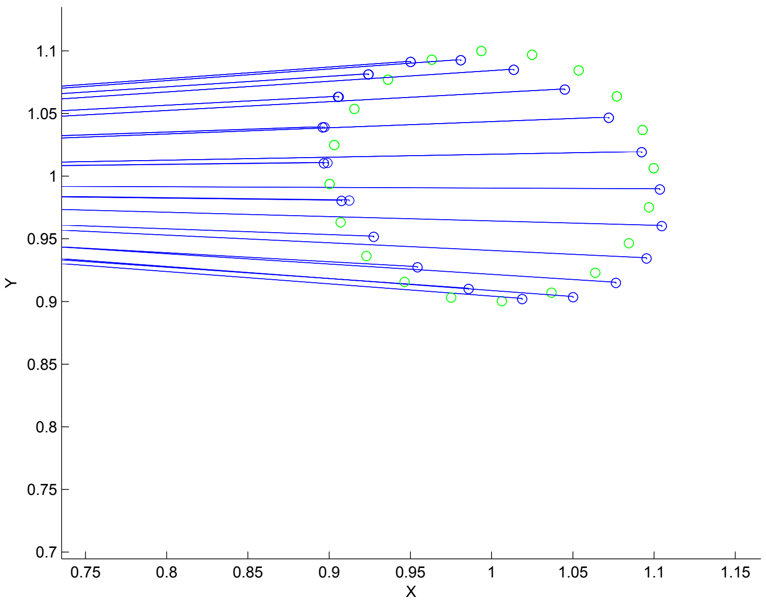


Figure 6.10: *Pseudo-Inverse Jabobian, with Constraints(Zoomed)*

7. Experiments

Testing and experiments is a good way to test if both the regulator / control system and dynamic of the system. The hydraulic manipulator is a complete working machine and only needs a hydraulic power unit, shown in figure 7.1, to be able to operate.

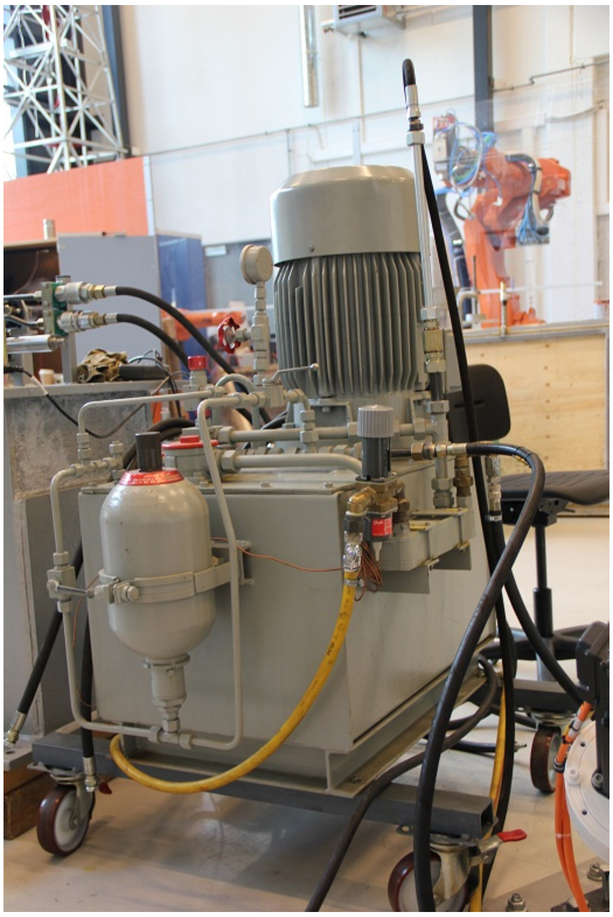


Figure 7.1: *Transportable Hydraulic Power Unit*

The experiments done with the manipulator consists of various testing each manipulator as well as the complete tool point control, both with and without redundancy.

7.1 Test Bench

The manipulator was mounted on the Stewart platform to give it a stable fundament. It is much energy in the manipulator when it's moving and is can cause fatal accidents if it should fall while moving.

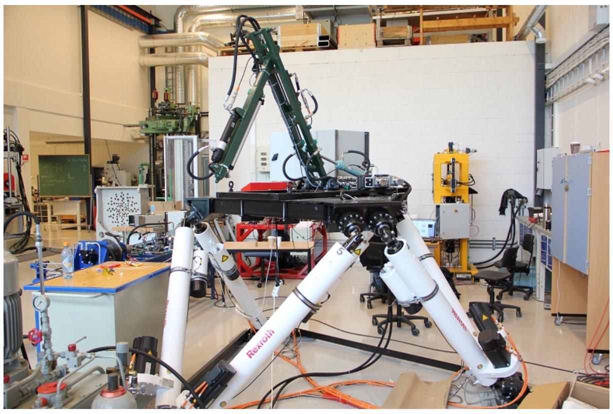


Figure 7.2: *Test Bench with the Manipulator on top of the Stewart Platform*

7.2 Without Redundancy

Since the dynamic and kinematic is less complex when working without redundancy this part of the testing in done before introducing the redundancy.
An experiment on the manipulator is to have the tool point move in a square. This is done by giving set points on each of the squares four corners.

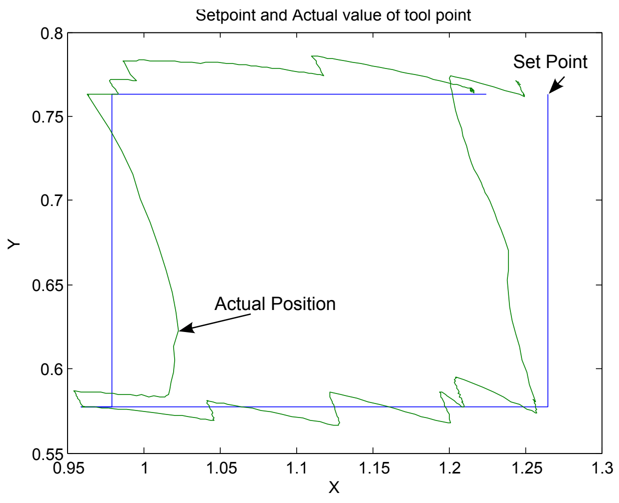


Figure 7.3: *Experiment Plot of the Set Point vs. the Actual Position in a square motion*

In figure 7.3 the blue line represents the set point of the tool point while the green line represents the actual position of the tool point.
The actual value follows the blue square, but it's some error. This is could be a result of bad tuning of the regulators controlling the hydraulic servo valves as the P and PI regulator gains.
In figure 7.4 the actual encoder values are recorded under the same experiment as in 7.3. The values are imported in the simulation in Matlab/Simulink such as the movement of the manipulator can be shown.

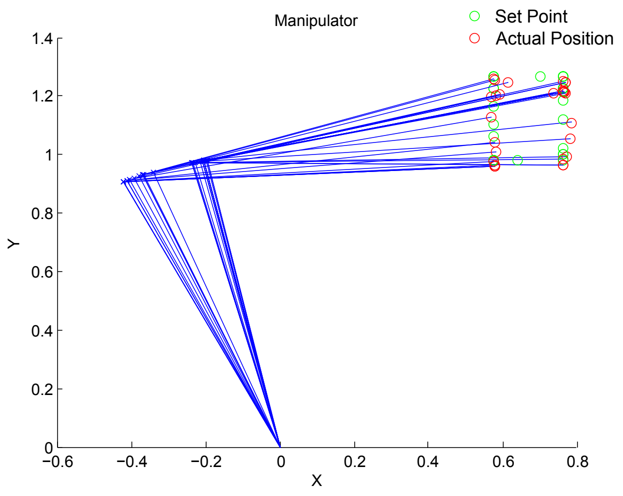


Figure 7.4: *Experimental data showing movements of the Manipulator*

7.3 With Redundancy

The testing of the manipulator with the last redundant joint is similar to the one without redundancy. In this test the redundant problem was solved by using the pseudo inverse Jabobian. In figure 7.5 the manipulators tool point had to follow a circular set point curve.

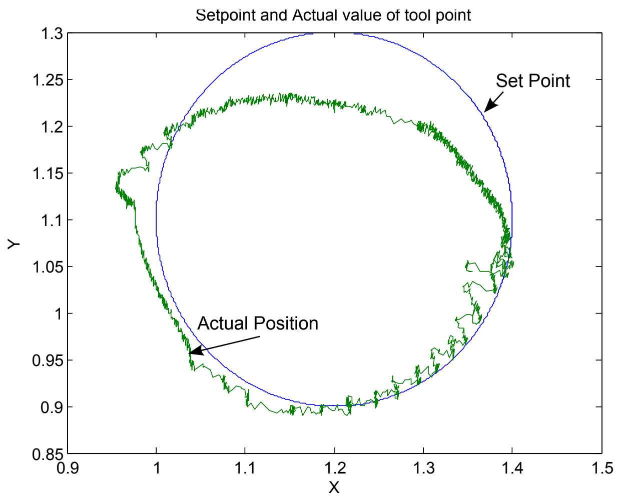


Figure 7.5: *Experiment Plot of the Set Point vs. the Actual Position in a circular motion*

In figure 7.6 the angle are imported to the simulation, same as in 7.4.

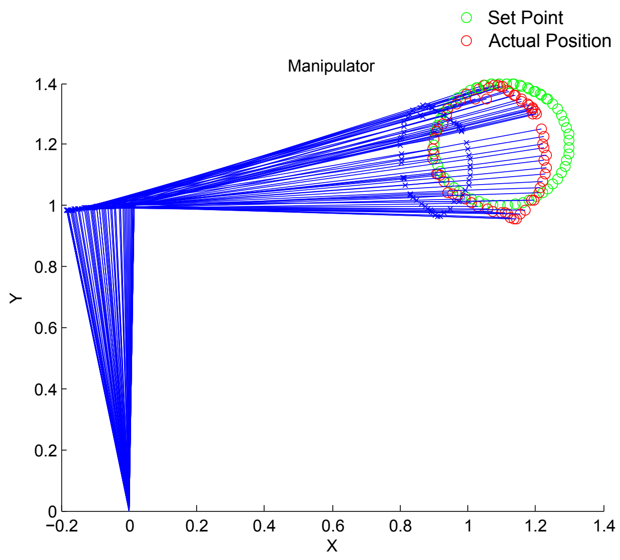


Figure 7.6: *Experimental data showing movements of the Manipulator*

I may look like the last joint doesn't seem to move, but in figure 7.7 all of the actuators are moving. Even though it looks like the third actuator hits its limits.

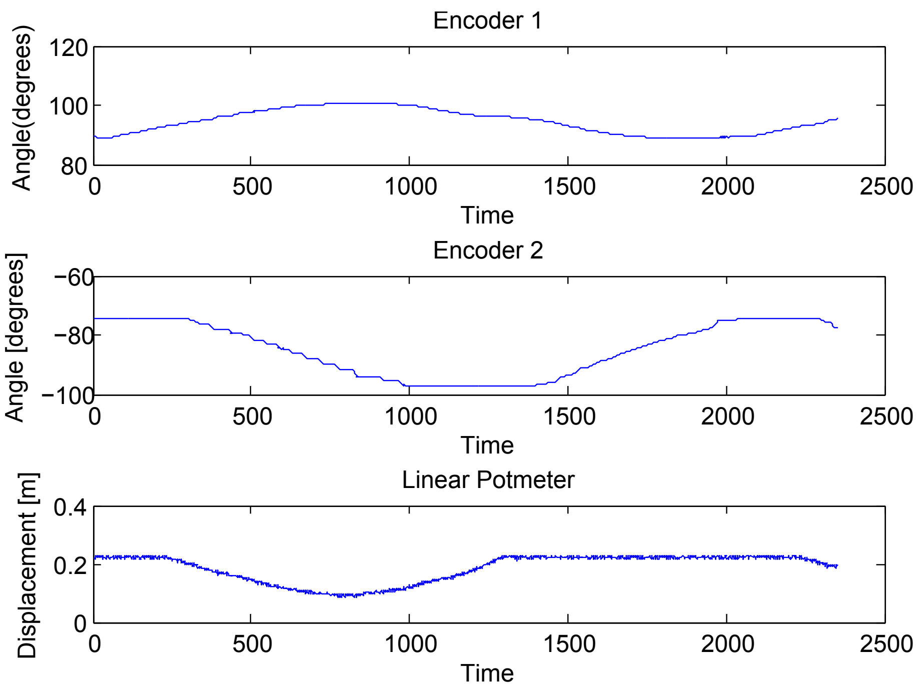


Figure 7.7: *Encoder and Linear Potentiometer*

Straight Line

The second part of the testing was done with the redundant configuration. The set point moves now in a straight line. First in the Y direction, Up/Down. Than it moves in X direction Left/Right. In figure 7.8 the angle are imported to the simulation, same as in 7.4.

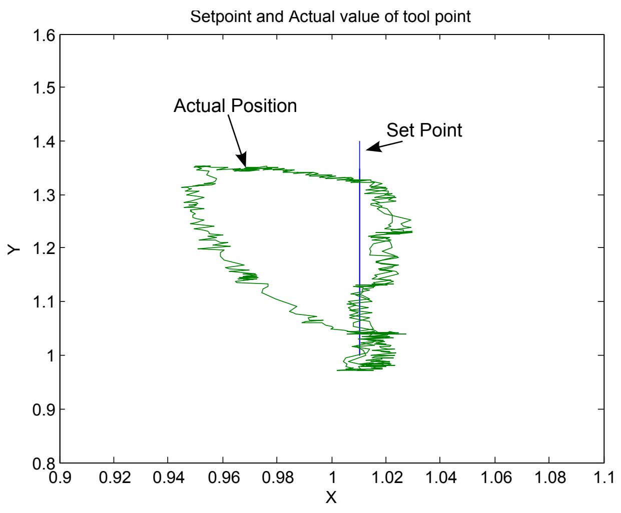


Figure 7.8: *Experiment Plot of the Set Point vs. the Actual Position in a straight line motion*

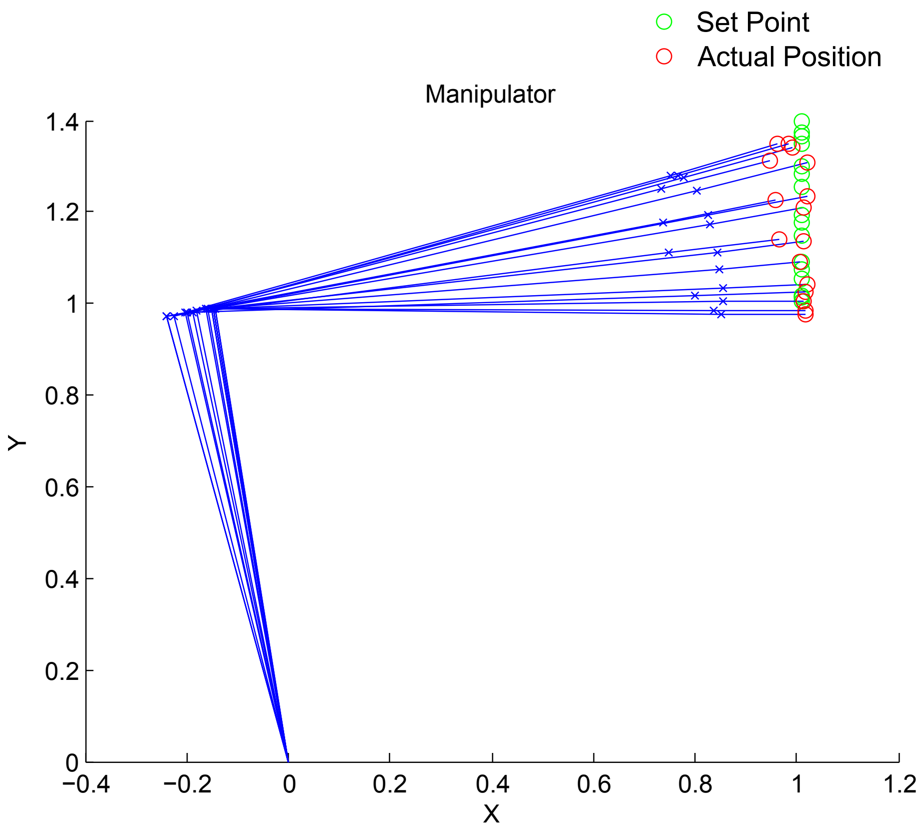


Figure 7.9: *Experimental data showing movements of the Manipulator*

The next test is the X direction, figure 7.10.

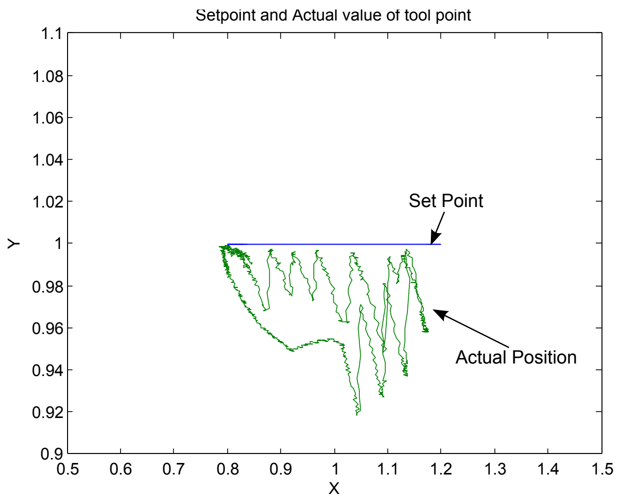


Figure 7.10: *Experiment Plot of the Set Point vs. the Actual Position in a straight line motion*

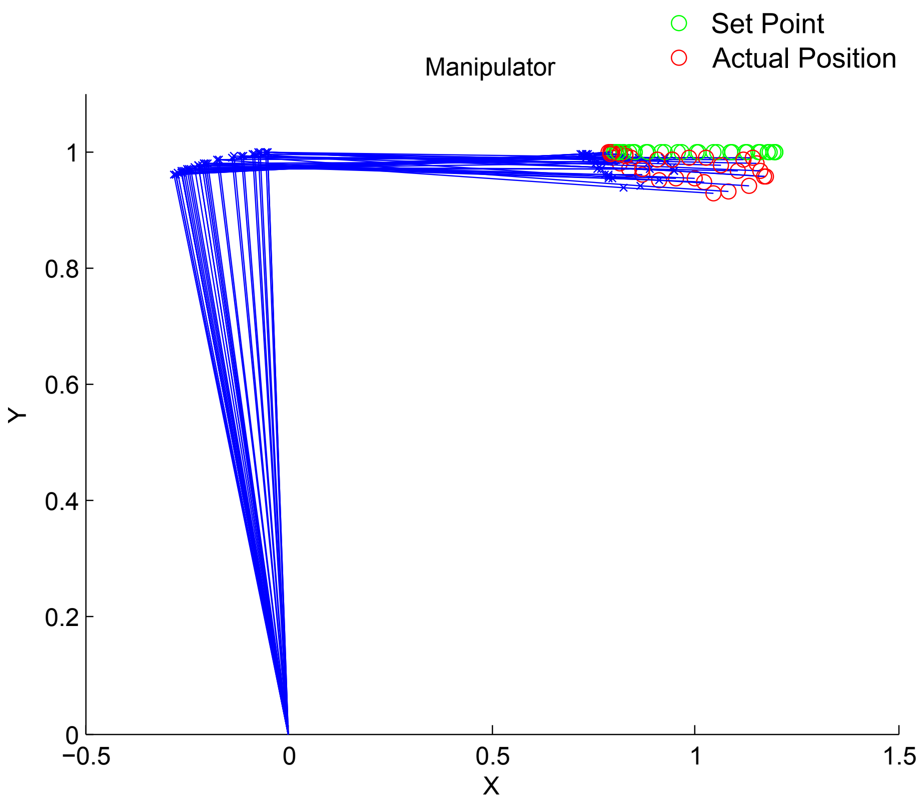


Figure 7.11: *Experimental data showing movements of the Manipulator*

8. Conclusion

This report shows the design of the control system for a heave compensated manipulator. With the control system the manipulator is able to follow a path with some errors. The manipulator was mounted on top of the Stewart platform, but the manipulator was not tested together with the Stewart platform due to lack of time.
The kinematics for both the Stewart platform and the hydraulic manipulator was successfully included in both the simulation, and only for the manipulator, in a real experiment. The redundancy was solved in different ways, but some problems were experienced when constrains were introduced when dealing with the pseudo-inverse Jabobian matrix as shown in the simulation and the extermination.
A model based controller for the hydraulic manipulator was proposed, but not included in any simulation or experiments.
In the simulation of the dynamics the desired set point for both position and velocity was reach with satisfying results for the hydraulic manipulator, but when experimenting in real life it didn't work that good. Especially the 2. actuator experienced some problem in the middle of the experimental period, but suddenly the problem disappeared. Perhaps errors with the controller unit for the servo valves were present.
Even though much time was used to fine tune the P and PI gains for the regulator, more works could be done in this area to improve the machine further.
The estimation for the roll and pitch angle, using the Inertial Measurement Unit was also a success. The sensor fusion using the 1. order complementary filter worked great. It gave a good result and even when doing the drop test it quickly corrected the error from the fall.
The wave measurement simulation gave good result when introducing the Spring-Damper system. Instead of having the position drifting away, it follow the correct reference with some error. Although the simulation gave a better result when introducing the Spring-Damper system, this way of measurement should not be used as the only one. There were too much error.

8.1 Further Work

Further work on this project includes research due to the model based controller. Since this was not applied to the project, the regulation of the position and velocities for the manipulator could be greatly improved by including this in the control system. The pressure sensors installed on the machine could be used for this purpose.
Due to irregularities in the unit that delivers current for the servo valves, the performance of the servo valve could be increased by introducing a new servo controller, like an industrial controller.
More research concerning solutions for the redundant problem should also be done in further work, as this is a very interesting field for robot kinematics and may offers benefits for heave compensating equipment.
Real wave data and data of the position of vessels operating in the North Sea may create a wave specter that can be used to generate realistic waves using the Stewart platform. This wave specter will give an accurate impression of what kind of environment the North Sea has to offer.

Bibliography

[1] Yang Wenlin ; Wei Sufen ; Zhang Zhuying ; Zhang Aiqun. Numerical simulation and testing analysis of adaptive heave motion measurements. 2009.

[2] Ampelmann. *Web page*, http://www.ampelmann.nl/.

[3] Michael Rygaard Hansen. Torben Ole Andersen. *Hydraulic Components and Systems.* 2009.

[4] S. Bandyopadhyay. A novel characterisation of spatial manipulators based on their degrees-of-freedom. 2009.

[5] Samuel R. Buss. Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. Technical report, Department of Mathematics University of California, San Diego, 2009.

[6] J. ; Saltaren R. ; Ferre M. ; Aracil R. Campos, A. ; Quintero. An active helideck testbed for floating structures based on a stewart-gough platform. 2008.

[7] S.E. Salcudean D. Li. Modeling, simulation and control of a hydraulic stewart platform.

[8] Michael ; Torres Javier ; Barton John ; O'Flynn Brendan ; O'Mathuna Cian Girardin, Yoan ; Walsh. Accounting for sensor drift in miniature, wireless inertial measurement and positioning systems: An extended kalman filtering approach. 2010.

[9] Geir Hovland. Mas 407, robot-dynamics. 2008. Lecture notes.

[10] http://img443.imageshack.us/img443/7374/6dof.gif.

[11] Anindito Santoso Indrawanto. Design and control of the stewart platform robot. 2009.

[12] Berstrand Haessig Oliver Sawodny Klaus Schneider Jörg Neypert, Tobias Mahl. A heave compensation approach for offshore cranes. 2008.

[13] Magnus B. Kjelland. Mini project - heave compensation. 2010.

[14] Brettvik Majken Aziz Soma Lørum Amund, Larssen Christian. Hydraulisk aktuert maipulator med 2d toolpoint kontroll. Technical report, Agder University, 2010.

[15] M. Vidyasagar Mark Spong, Seth Hutchinson. *Robot Modeling and Control.* John Wiley and Sons, Inc, 2006.

[16] T.A. ; Ravani B. Parsa, K. ; Lasky. Design and implementation of a mechatronic, all-accelerometer inertial measurement unit. 2007.

[17] B. ; Silvestre C. ; Oliveira P. ; Batista P. Vasconcelos, J.F. ; Cardeira. Discrete-time complementary filters for attitude and position estimation: Design, analysis and experimental validation. 2011.

List of Figures

A. MATLAB/Simulink

A.1 Position and plot of joints of Stewart Platform

```
close all;
clc;
X=0;
Y=0;
Z=1;
yaw=90*pi/180;
roll=0*pi/180;
pitch=0;
%syms X Y Z yaw roll pitch
Px1=X+(3140116564492417*cos(roll)*cos(yaw))/72057594037927936+(4486462071114449*cos↙
(pitch)*sin(yaw))/9007199254740992+(4486462071114449*cos(yaw)*sin(pitch)*sin(roll))↙
/9007199254740992;
Py1=Y-(4486462071114449*cos(pitch)*cos(yaw))/9007199254740992+(3140116564492417*cos↙
(roll)*sin(yaw))/72057594037927936+(4486462071114449*sin(pitch)*sin(roll)*sin(yaw))↙
/9007199254740992;
Pz1=Z+(3140116564492417*sin(roll))/72057594037927936-(4486462071114449*cos(roll)*sin↙
(pitch))/9007199254740992;
Px2=X+(8163294823962471*cos(roll)*cos(yaw))/18014398509481984-(7613213784381523*cos↙
(pitch)*sin(yaw))/36028797018963968-(7613213784381523*cos(yaw)*sin(pitch)*sin(roll))↙
/36028797018963968;
Py2=Y+(7613213784381523*cos(pitch)*cos(yaw))/36028797018963968+(8163294823962471*cos↙
(roll)*sin(yaw))/18014398509481984-(7613213784381523*sin(pitch)*sin(roll)*sin(yaw))↙
/36028797018963968;
Pz2=Z+(8163294823962471*sin(roll))/18014398509481984+(7613213784381523*cos(roll)*sin↙
(pitch))/36028797018963968;
Px3=X+(7378265682839367*cos(roll)*cos(yaw))/18014398509481984-(645789656254767*cos↙
(pitch)*sin(yaw))/2251799813685248-(645789656254767*cos(yaw)*sin(pitch)*sin(roll))↙
/2251799813685248;
Py3=Y+(645789656254767*cos(pitch)*cos(yaw))/2251799813685248+(7378265682839367*cos↙
(roll)*sin(yaw))/18014398509481984-(645789656254767*sin(pitch)*sin(roll)*sin(yaw))↙
/2251799813685248;
Pz3=Z+(7378265682839367*sin(roll))/18014398509481984+(645789656254767*cos(roll)*sin↙
(pitch))/2251799813685248;
Px4=X-(7378265682839367*cos(roll)*cos(yaw))/18014398509481984-(645789656254767*cos↙
(pitch)*sin(yaw))/2251799813685248-(645789656254767*cos(yaw)*sin(pitch)*sin(roll))↙
/2251799813685248;
Py4=Y+(645789656254767*cos(pitch)*cos(yaw))/2251799813685248-(7378265682839367*cos↙
(roll)*sin(yaw))/18014398509481984-(645789656254767*sin(pitch)*sin(roll)*sin(yaw))↙
/2251799813685248;
Pz4=Z-(7378265682839367*sin(roll))/18014398509481984+(645789656254767*cos(roll)*sin↙
(pitch))/2251799813685248;
Px5=X-(8163294823962471*cos(roll)*cos(yaw))/18014398509481984-(7613213784381523*cos↙
(pitch)*sin(yaw))/36028797018963968-(7613213784381523*cos(yaw)*sin(pitch)*sin(roll))↙
/36028797018963968;
Py5=Y+(7613213784381523*cos(pitch)*cos(yaw))/36028797018963968-(8163294823962471*cos↙
(roll)*sin(yaw))/18014398509481984-(7613213784381523*sin(pitch)*sin(roll)*sin(yaw))↙
/36028797018963968;
Pz5=Z-(8163294823962471*sin(roll))/18014398509481984+(7613213784381523*cos(roll)*sin↙
(pitch))/36028797018963968;
Px6=X-(3140116564492417*cos(roll)*cos(yaw))/72057594037927936+(4486462071114449*cos↙
(pitch)*sin(yaw))/9007199254740992+(4486462071114449*cos(yaw)*sin(pitch)*sin(roll))↙
/9007199254740992;
Py6=Y-(4486462071114449*cos(pitch)*cos(yaw))/9007199254740992-(3140116564492417*cos↙
(roll)*sin(yaw))/72057594037927936+(4486462071114449*sin(pitch)*sin(roll)*sin(yaw))↙
/9007199254740992;
Pz6=Z-(3140116564492417*sin(roll))/72057594037927936-(4486462071114449*cos(roll)*sin↙
(pitch))/9007199254740992;
Bz(1:6)=0;
%Calculated Length of each Leg
```

```matlab
L1=sqrt((Px1-Bx(6))^2+(Py1-By(6))^2+(Pz1-Bz(6))^2)
L2=sqrt((Px2-Bx(1))^2+(Py2-By(1))^2+(Pz2-Bz(1))^2)
L3=sqrt((Px3-Bx(2))^2+(Py3-By(2))^2+(Pz3-Bz(2))^2)
L4=sqrt((Px4-Bx(3))^2+(Py4-By(3))^2+(Pz4-Bz(3))^2)
L5=sqrt((Px5-Bx(4))^2+(Py5-By(4))^2+(Pz5-Bz(4))^2)
L6=sqrt((Px6-Bx(5))^2+(Py6-By(5))^2+(Pz6-Bz(5))^2)

%To find the jacobian the syms must be used..
% j=simple(jacobian([L1 L2 L3 L4 L5 L6],[X Y Z yaw roll pitch]))
axis([-1 1 -1 1 0 2])

hold on
%plotting lines between joints of the base
line([Bx(1) Bx(2)],[By(1) By(2)],[Bz(1) Bz(2)]);
line([Bx(2) Bx(3)],[By(2) By(3)],[Bz(2) Bz(3)]);
line([Bx(3) Bx(4)],[By(3) By(4)],[Bz(3) Bz(4)]);
line([Bx(4) Bx(5)],[By(4) By(5)],[Bz(4) Bz(5)]);
line([Bx(5) Bx(6)],[By(5) By(6)],[Bz(5) Bz(6)]);
line([Bx(6) Bx(1)],[By(6) By(1)],[Bz(6) Bz(1)]);
%plotting lines between joints of the platform
line([Px1 Px2],[Py1 Py2],[Pz1 Pz2]);
line([Px2 Px3],[Py2 Py3],[Pz2 Pz3]);
line([Px3 Px4],[Py3 Py4],[Pz3 Pz4]);
line([Px4 Px5],[Py4 Py5],[Pz4 Pz5]);
line([Px5 Px6],[Py5 Py6],[Pz5 Pz6]);
line([Px6 Px1],[Py6 Py1],[Pz6 Pz1]);
%plotting Joints of platform
plot3(Px1,Py1,Pz1,'ro');
plot3(Px2,Py2,Pz2,'ro');
plot3(Px3,Py3,Pz3,'ro');
plot3(Px4,Py4,Pz4,'ro');
plot3(Px5,Py5,Pz5,'ro');
plot3(Px6,Py6,Pz6,'ro');
%plotting Joints of base
plot3(Bx(1),By(1),Bz(1),'ro');
plot3(Bx(2),By(2),Bz(2),'ro');
plot3(Bx(3),By(3),Bz(3),'ro');
plot3(Bx(4),By(4),Bz(4),'ro');
plot3(Bx(5),By(5),Bz(5),'ro');
plot3(Bx(6),By(6),Bz(6),'ro');
%plotting LEGS
line([Px1 Bx(6)],[Py1 By(6)],[Pz1 Bz(6)],'color','r');
line([Px2 Bx(1)],[Py2 By(1)],[Pz2 Bz(1)],'color','r');
line([Px3 Bx(2)],[Py3 By(2)],[Pz3 Bz(2)],'color','r');
line([Px4 Bx(3)],[Py4 By(3)],[Pz4 Bz(3)],'color','r');
line([Px5 Bx(4)],[Py5 By(4)],[Pz5 Bz(4)],'color','r');
line([Px6 Bx(5)],[Py6 By(5)],[Pz6 Bz(5)],'color','r');
```

A.2 Structure and plot of joints of Stewart Platform

```
clear all
close all
clc
%%MAIN SETUP: Two parameters.
%Parameters for the Platform
%Radius of the joints in meters/cm/mm ??
%Offset of the joint in degrees
radiusP=0.5;
offsetP=5;
%Parameters for the Base
%Radius of the joints in meters/cm/mm ??
%Offset of the joint in degrees
radiusB=1;
offsetB=5;
%%Building...
axis([-1.2 1.2 -1 1])
hold on
Px(1)=cosd(0+offsetP)*radiusP
Py(1)=sind(0+offsetP)*radiusP

Px(2)=cosd(120-offsetP)*radiusP
Py(2)=sind(120-offsetP)*radiusP

Px(3)=cosd(120+offsetP)*radiusP
Py(3)=sind(120+offsetP)*radiusP

Px(4)=cosd(240-offsetP)*radiusP
Py(4)=sind(240-offsetP)*radiusP

Px(5)=cosd(240+offsetP)*radiusP
Py(5)=sind(240+offsetP)*radiusP

Px(6)=cosd(0-offsetP)*radiusP
Py(6)=sind(0-offsetP)*radiusP

Bx(1)=cosd(60+offsetB)*radiusB
By(1)=sind(60+offsetB)*radiusB

Bx(2)=cosd(180-offsetB)*radiusB
By(2)=sind(180-offsetB)*radiusB

Bx(3)=cosd(180+offsetB)*radiusB
By(3)=sind(180+offsetB)*radiusB

Bx(4)=cosd(300-offsetB)*radiusB
By(4)=sind(300-offsetB)*radiusB

Bx(5)=cosd(300+offsetB)*radiusB
By(5)=sind(300+offsetB)*radiusB

Bx(6)=cosd(60-offsetB)*radiusB
By(6)=sind(60-offsetB)*radiusB
%%Plotting
%Platform
plot(Px(:),Py(:),'ro')
plot(0,0,'bo')
%Base
plot(Bx(:),By(:),'go')
plot(0,0,'bo')
```

```
%%               DH TABLE
% |---------------------------------------|
% |  RotZ   |  TransZ   |  TransX   |  RotX  |
% |---------------------------------------|
% |         |           |    X      |  -90   |
% |         |     Y     |           |  +90   |
% |  yaw    |     Z     |           |  +-90  |
% |  roll   |           |           |  pitch |
% |         |   -0.45   |  -0.7794 |        |
% |---------------------------------------|
syms X Y Z yaw roll pitch

%%Leg1
TransX0=[1,0,0,X;0,1,0,0;0,0,1,0;0,0,0,1];RotX0=[1,0,0,0;0,0,1,0;0,-1,0,0;0,0,0,1];
TransZ1=[1,0,0,0;0,1,0,0;0,0,1,Y;0,0,0,1];RotX1=[1,0,0,0;0,0,-1,0;0,1,0,0;0,0,0,1];
RotZ2=[cos(yaw),-sin(yaw),0,0;sin(yaw),cos(yaw),0,0;0,0,1,0;0,0,0,1];
TransZ2=[1,0,0,0;0,1,0,0;0,0,1,Z;0,0,0,1];
RotX2=[1,0,0,0;0,0,-1,0;0,1,0,0;0,0,0,1];
RotZ3=[cos(roll),-sin(roll),0,0;sin(roll),cos(roll),0,0;0,0,1,0;0,0,0,1];
RotX3=[1,0,0,0;0,cos(pitch),-sin(pitch),0;0,sin(pitch),cos(pitch),0;0,0,0,1];
TransZ4=[1,0,0,0;0,1,0,0;0,0,1,Px(1);0,0,0,1];TransX4=[1,0,0,Py(1);0,1,0,0;0,0,1,0;↙
0,0,0,1];

G=TransX0*RotX0*TransZ1*RotX1*RotZ2*TransZ2*RotX2*RotZ3*RotX3*TransZ4*TransX4;
Px1=simple(G(1,4))
Py1=simple(G(2,4))
Pz1=simple(G(3,4))
%%               DH TABLE
% |---------------------------------------|
% |  RotZ   |  TransZ   |  TransX   |  RotX  |
% |---------------------------------------|
% |         |           |    X      |  -90   |
% |         |     Y     |           |  +90   |
% |  yaw    |     Z     |           |  +-90  |
% |  roll   |           |           |  pitch |
% |         | Px        |  Py       |        |
% |---------------------------------------|


%%Leg2
TransX0=[1,0,0,X;0,1,0,0;0,0,1,0;0,0,0,1];RotX0=[1,0,0,0;0,0,1,0;0,-1,0,0;0,0,0,1];
TransZ1=[1,0,0,0;0,1,0,0;0,0,1,Y;0,0,0,1];RotX1=[1,0,0,0;0,0,-1,0;0,1,0,0;0,0,0,1];
RotZ2=[cos(yaw),-sin(yaw),0,0;sin(yaw),cos(yaw),0,0;0,0,1,0;0,0,0,1];
TransZ2=[1,0,0,0;0,1,0,0;0,0,1,Z;0,0,0,1];
RotX2=[1,0,0,0;0,0,-1,0;0,1,0,0;0,0,0,1];
RotZ3=[cos(roll),-sin(roll),0,0;sin(roll),cos(roll),0,0;0,0,1,0;0,0,0,1];
RotX3=[1,0,0,0;0,cos(pitch),-sin(pitch),0;0,sin(pitch),cos(pitch),0;0,0,0,1];
TransZ4=[1,0,0,0;0,1,0,0;0,0,1,Px(2);0,0,0,1];TransX4=[1,0,0,Py(2);0,1,0,0;0,0,1,0;↙
0,0,0,1];

G=TransX0*RotX0*TransZ1*RotX1*RotZ2*TransZ2*RotX2*RotZ3*RotX3*TransZ4*TransX4;
Px2=simple(G(1,4))
Py2=simple(G(2,4))
Pz2=simple(G(3,4))


%%               DH TABLE
% |---------------------------------------|
% |  RotZ   |  TransZ   |  TransX   |  RotX  |
% |---------------------------------------|
% |         |           |    X      |  -90   |
% |         |     Y     |           |  +90   |
% |  yaw    |     Z     |           |  +-90  |
% |  roll   |           |           |  pitch |
% |         | Px        |  Py       |        |
% |----------------------------------------|

%%Leg3
TransX0=[1,0,0,X;0,1,0,0;0,0,1,0;0,0,0,1];RotX0=[1,0,0,0;0,0,1,0;0,-1,0,0;0,0,0,1];
TransZ1=[1,0,0,0;0,1,0,0;0,0,1,Y;0,0,0,1];RotX1=[1,0,0,0;0,0,-1,0;0,1,0,0;0,0,0,1];
RotZ2=[cos(yaw),-sin(yaw),0,0;sin(yaw),cos(yaw),0,0;0,0,1,0;0,0,0,1];
TransZ2=[1,0,0,0;0,1,0,0;0,0,1,Z;0,0,0,1];
RotX2=[1,0,0,0;0,0,-1,0;0,1,0,0;0,0,0,1];
RotZ3=[cos(roll),-sin(roll),0,0;sin(roll),cos(roll),0,0;0,0,1,0;0,0,0,1];
RotX3=[1,0,0,0;0,cos(pitch),-sin(pitch),0;0,sin(pitch),cos(pitch),0;0,0,0,1];
TransZ4=[1,0,0,0;0,1,0,0;0,0,1,Px(3);0,0,0,1];TransX4=[1,0,0,Py(3);0,1,0,0;0,0,1,0;↙
0,0,0,1];

G=TransX0*RotX0*TransZ1*RotX1*RotZ2*TransZ2*RotX2*RotZ3*RotX3*TransZ4*TransX4;
Px3=simple(G(1,4))
Py3=simple(G(2,4))
Pz3=simple(G(3,4))

%%                DH TABLE
% |----------------------------------------|
% |  RotZ   |  TransZ   |  TransX   |  RotX   |
% |----------------------------------------|
% |         |           |    X      |  -90   |
% |         |     Y     |           |  +90   |
% |  yaw    |     Z     |           |  +-90  |
% |  roll   |           |           |  pitch |
% |         | Px        |  Py       |        |
% |----------------------------------------|

%%Leg4
TransX0=[1,0,0,X;0,1,0,0;0,0,1,0;0,0,0,1];RotX0=[1,0,0,0;0,0,1,0;0,-1,0,0;0,0,0,1];
TransZ1=[1,0,0,0;0,1,0,0;0,0,1,Y;0,0,0,1];RotX1=[1,0,0,0;0,0,-1,0;0,1,0,0;0,0,0,1];
RotZ2=[cos(yaw),-sin(yaw),0,0;sin(yaw),cos(yaw),0,0;0,0,1,0;0,0,0,1];
TransZ2=[1,0,0,0;0,1,0,0;0,0,1,Z;0,0,0,1];
RotX2=[1,0,0,0;0,0,-1,0;0,1,0,0;0,0,0,1];
RotZ3=[cos(roll),-sin(roll),0,0;sin(roll),cos(roll),0,0;0,0,1,0;0,0,0,1];
RotX3=[1,0,0,0;0,cos(pitch),-sin(pitch),0;0,sin(pitch),cos(pitch),0;0,0,0,1];
TransZ4=[1,0,0,0;0,1,0,0;0,0,1,Px(4);0,0,0,1];TransX4=[1,0,0,Py(4);0,1,0,0;0,0,1,0;↙
0,0,0,1];

G=TransX0*RotX0*TransZ1*RotX1*RotZ2*TransZ2*RotX2*RotZ3*RotX3*TransZ4*TransX4;
Px4=simple(G(1,4))
Py4=simple(G(2,4))
Pz4=simple(G(3,4))

%%                DH TABLE
% |----------------------------------------|
% |  RotZ   |  TransZ   |  TransX   |  RotX   |
% |----------------------------------------|
% |         |           |    X      |  -90   |
% |         |     Y     |           |  +90   |
% |  yaw    |     Z     |           |  +-90  |
% |  roll   |           |           |  pitch |
% |         | Px        |  Py       |        |
% |----------------------------------------|
%%Leg5
TransX0=[1,0,0,X;0,1,0,0;0,0,1,0;0,0,0,1];RotX0=[1,0,0,0;0,0,1,0;0,-1,0,0;0,0,0,1];
TransZ1=[1,0,0,0;0,1,0,0;0,0,1,Y;0,0,0,1];RotX1=[1,0,0,0;0,0,-1,0;0,1,0,0;0,0,0,1];
RotZ2=[cos(yaw),-sin(yaw),0,0;sin(yaw),cos(yaw),0,0;0,0,1,0;0,0,0,1];
TransZ2=[1,0,0,0;0,1,0,0;0,0,1,Z;0,0,0,1];
RotX2=[1,0,0,0;0,0,-1,0;0,1,0,0;0,0,0,1];
RotZ3=[cos(roll),-sin(roll),0,0;sin(roll),cos(roll),0,0;0,0,1,0;0,0,0,1];
RotX3=[1,0,0,0;0,cos(pitch),-sin(pitch),0;0,sin(pitch),cos(pitch),0;0,0,0,1];
TransZ4=[1,0,0,0;0,1,0,0;0,0,1,Px(5);0,0,0,1];TransX4=[1,0,0,Py(5);0,1,0,0;0,0,1,0;↙
0,0,0,1];

G=TransX0*RotX0*TransZ1*RotX1*RotZ2*TransZ2*RotX2*RotZ3*RotX3*TransZ4*TransX4;
Px5=simple(G(1,4))
Py5=simple(G(2,4))
Pz5=simple(G(3,4))
%%                DH TABLE
% |---------------------------------------|
% |  RotZ   |  TransZ   |  TransX   |  RotX  |
% |---------------------------------------|
% |         |           |    X      |  -90   |
% |         |     Y     |           |  +90   |
% |  yaw    |     Z     |           |  +-90  |
% |  roll   |           |           |  pitch |
% |         | Px        |  Py       |        |
% |---------------------------------------|

%%Leg6
TransX0=[1,0,0,X;0,1,0,0;0,0,1,0;0,0,0,1];RotX0=[1,0,0,0;0,0,1,0;0,-1,0,0;0,0,0,1];
TransZ1=[1,0,0,0;0,1,0,0;0,0,1,Y;0,0,0,1];RotX1=[1,0,0,0;0,0,-1,0;0,1,0,0;0,0,0,1];
RotZ2=[cos(yaw),-sin(yaw),0,0;sin(yaw),cos(yaw),0,0;0,0,1,0;0,0,0,1];
TransZ2=[1,0,0,0;0,1,0,0;0,0,1,Z;0,0,0,1];
RotX2=[1,0,0,0;0,0,-1,0;0,1,0,0;0,0,0,1];
RotZ3=[cos(roll),-sin(roll),0,0;sin(roll),cos(roll),0,0;0,0,1,0;0,0,0,1];
RotX3=[1,0,0,0;0,cos(pitch),-sin(pitch),0;0,sin(pitch),cos(pitch),0;0,0,0,1];
TransZ4=[1,0,0,0;0,1,0,0;0,0,1,Px(6);0,0,0,1];TransX4=[1,0,0,Py(6);0,1,0,0;0,0,1,0;↙
0,0,0,1];

G=TransX0*RotX0*TransZ1*RotX1*RotZ2*TransZ2*RotX2*RotZ3*RotX3*TransZ4*TransX4;
Px6=simple(G(1,4))
Py6=simple(G(2,4))
Pz6=simple(G(3,4))
```

A.3 Animation of joints of Stewart Platform

```
close all
clear all
clc
fig=figure;
aviobj = avifile('test.avi');
aviobj.fps = 20;
X=0;Y=0;Z=1;yaw=pi/4;roll=0;pitch=0;
Bx1=0.1;By1=0;Bz1=0;Bx2=-0.1;By2=-0.1;Bz2=0;Bx3=-0.1;By3=0.1;Bz3=0;
for n=1:10
    Px1 =X - (cos(roll)*cos(yaw))/20 - (cos(pitch)*sin(yaw))/20 - (cos(yaw)*sin(pitch)↙
*sin(roll))/20;Py1 =Y + (cos(pitch)*cos(yaw))/20 - (cos(roll)*sin(yaw))/20 - (sin↙
(pitch)*sin(roll)*sin(yaw))/20;Pz1 =Z - sin(roll)/20 + (cos(roll)*sin(pitch))/20;Px2 =X↙
+ (cos(roll)*cos(yaw))/20 - (cos(pitch)*sin(yaw))/20 - (cos(yaw)*sin(pitch)*sin(roll))↙
/20;Py2 =Y + (cos(pitch)*cos(yaw))/20 + (cos(roll)*sin(yaw))/20 - (sin(pitch)*sin(roll)↙
*sin(yaw))/20;Pz2 =Z + sin(roll)/20 + (cos(roll)*sin(pitch))/20;Px3 =X + (cos(pitch)↙
*sin(yaw))/20 + (cos(yaw)*sin(pitch)*sin(roll))/20;Py3 =Y - (cos(pitch)*cos(yaw))/20 +↙
(sin(pitch)*sin(roll)*sin(yaw))/20;Pz3 =Z - (cos(roll)*sin(pitch))/20;axis([-0.2 0.2↙
-0.2 0.2 0 1.2]);plot(0.1,0.1);hold on;line([Bx1 Bx2],[By1 By2],[Bz1 Bz2]);line([Bx2↙
Bx3],[By2 By3],[Bz2 Bz3]);line([Bx1 Bx3],[By1 By3],[Bz1 Bz3]);line([Px1 Px2],[Py1 Py2],↙
[Pz1 Pz2]);line([Px2 Px3],[Py2 Py3],[Pz2 Pz3]);line([Px1 Px3],[Py1 Py3],[Pz1 Pz3]);line↙
([Px1 Bx3],[Py1 By3],[Pz1 Bz3],'color','r');line([Px1 Bx2],[Py1 By2],[Pz1↙
Bz2],'color','r');plot3(Px1,Py1,Pz1,'ro');line([Px3 Bx2],[Py3 By2],[Pz3↙
Bz2],'color','b');line([Px3 Bx1],[Py3 By1],[Pz3 Bz1],'color','b');plot3(Px3,Py3,↙
Pz3,'bo');line([Px2 Bx1],[Py2 By1],[Pz2 Bz1],'color','g');line([Px2 Bx3],[Py2 By3],[Pz2↙
Bz3],'color','g');plot3(Px2,Py2,Pz2,'go');hold off;axis([-0.2 0.2 -0.2 0.2 0 1.2]);↙
frame = getframe(fig);aviobj = addframe(aviobj,frame);
    yaw=yaw-0.1;
end
for n=1:20
    Px1 =X - (cos(roll)*cos(yaw))/20 - (cos(pitch)*sin(yaw))/20 - (cos(yaw)*sin(pitch)↙
*sin(roll))/20;Py1 =Y + (cos(pitch)*cos(yaw))/20 - (cos(roll)*sin(yaw))/20 - (sin↙
(pitch)*sin(roll)*sin(yaw))/20;Pz1 =Z - sin(roll)/20 + (cos(roll)*sin(pitch))/20;Px2 =X↙
+ (cos(roll)*cos(yaw))/20 - (cos(pitch)*sin(yaw))/20 - (cos(yaw)*sin(pitch)*sin(roll))↙
/20;Py2 =Y + (cos(pitch)*cos(yaw))/20 + (cos(roll)*sin(yaw))/20 - (sin(pitch)*sin(roll)↙
*sin(yaw))/20;Pz2 =Z + sin(roll)/20 + (cos(roll)*sin(pitch))/20;Px3 =X + (cos(pitch)↙
*sin(yaw))/20 + (cos(yaw)*sin(pitch)*sin(roll))/20;Py3 =Y - (cos(pitch)*cos(yaw))/20 +↙
(sin(pitch)*sin(roll)*sin(yaw))/20;Pz3 =Z - (cos(roll)*sin(pitch))/20;axis([-0.2 0.2↙
-0.2 0.2 0 1.2]);plot(0.1,0.1);hold on;line([Bx1 Bx2],[By1 By2],[Bz1 Bz2]);line([Bx2↙
Bx3],[By2 By3],[Bz2 Bz3]);line([Bx1 Bx3],[By1 By3],[Bz1 Bz3]);line([Px1 Px2],[Py1 Py2],↙
[Pz1 Pz2]);line([Px2 Px3],[Py2 Py3],[Pz2 Pz3]);line([Px1 Px3],[Py1 Py3],[Pz1 Pz3]);line↙
([Px1 Bx3],[Py1 By3],[Pz1 Bz3],'color','r');line([Px1 Bx2],[Py1 By2],[Pz1↙
Bz2],'color','r');plot3(Px1,Py1,Pz1,'ro');line([Px3 Bx2],[Py3 By2],[Pz3↙
Bz2],'color','b');line([Px3 Bx1],[Py3 By1],[Pz3 Bz1],'color','b');plot3(Px3,Py3,↙
Pz3,'bo');line([Px2 Bx1],[Py2 By1],[Pz2 Bz1],'color','g');line([Px2 Bx3],[Py2 By3],[Pz2↙
Bz3],'color','g');plot3(Px2,Py2,Pz2,'go');hold off;axis([-0.2 0.2 -0.2 0.2 0 1.2]);↙
frame = getframe(fig);aviobj = addframe(aviobj,frame);
    yaw=yaw+0.1;
end
for n=1:10
    Px1 =X - (cos(roll)*cos(yaw))/20 - (cos(pitch)*sin(yaw))/20 - (cos(yaw)*sin(pitch)↙
*sin(roll))/20;Py1 =Y + (cos(pitch)*cos(yaw))/20 - (cos(roll)*sin(yaw))/20 - (sin↙
(pitch)*sin(roll)*sin(yaw))/20;Pz1 =Z - sin(roll)/20 + (cos(roll)*sin(pitch))/20;Px2 =X↙
+ (cos(roll)*cos(yaw))/20 - (cos(pitch)*sin(yaw))/20 - (cos(yaw)*sin(pitch)*sin(roll))↙
/20;Py2 =Y + (cos(pitch)*cos(yaw))/20 + (cos(roll)*sin(yaw))/20 - (sin(pitch)*sin(roll)↙
*sin(yaw))/20;Pz2 =Z + sin(roll)/20 + (cos(roll)*sin(pitch))/20;Px3 =X + (cos(pitch)↙
*sin(yaw))/20 + (cos(yaw)*sin(pitch)*sin(roll))/20;Py3 =Y - (cos(pitch)*cos(yaw))/20 +↙
(sin(pitch)*sin(roll)*sin(yaw))/20;Pz3 =Z - (cos(roll)*sin(pitch))/20;axis([-0.2 0.2↙
-0.2 0.2 0 1.2]);plot(0.1,0.1);hold on;line([Bx1 Bx2],[By1 By2],[Bz1 Bz2]);line([Bx2↙
Bx3],[By2 By3],[Bz2 Bz3]);line([Bx1 Bx3],[By1 By3],[Bz1 Bz3]);line([Px1 Px2],[Py1 Py2],↙
[Pz1 Pz2]);line([Px2 Px3],[Py2 Py3],[Pz2 Pz3]);line([Px1 Px3],[Py1 Py3],[Pz1 Pz3]);line↙
```

```
([Px1 Bx3],[Py1 By3],[Pz1 Bz3],'color','r');line([Px1 Bx2],[Py1 By2],[Pz1↙
Bz2],'color','r');plot3(Px1,Py1,Pz1,'ro');line([Px3 Bx2],[Py3 By2],[Pz3↙
Bz2],'color','b');line([Px3 Bx1],[Py3 By1],[Pz3 Bz1],'color','b');plot3(Px3,Py3,↙
Pz3,'bo');line([Px2 Bx1],[Py2 By1],[Pz2 Bz1],'color','g');line([Px2 Bx3],[Py2 By3],[Pz2↙
Bz3],'color','g');plot3(Px2,Py2,Pz2,'go');hold off;axis([-0.2 0.2 -0.2 0.2 0 1.2]);↙
frame = getframe(fig);aviobj = addframe(aviobj,frame);
    yaw=yaw-0.1;
end
yaw=pi/4;
for n=1:10
    Px1 =X - (cos(roll)*cos(yaw))/20 - (cos(pitch)*sin(yaw))/20 - (cos(yaw)*sin(pitch)↙
*sin(roll))/20;Py1 =Y + (cos(pitch)*cos(yaw))/20 - (cos(roll)*sin(yaw))/20 - (sin↙
(pitch)*sin(roll)*sin(yaw))/20;Pz1 =Z - sin(roll)/20 + (cos(roll)*sin(pitch))/20;Px2 =X↙
+ (cos(roll)*cos(yaw))/20 - (cos(pitch)*sin(yaw))/20 - (cos(yaw)*sin(pitch)*sin(roll))↙
/20;Py2 =Y + (cos(pitch)*cos(yaw))/20 + (cos(roll)*sin(yaw))/20 - (sin(pitch)*sin(roll)↙
*sin(yaw))/20;Pz2 =Z + sin(roll)/20 + (cos(roll)*sin(pitch))/20;Px3 =X + (cos(pitch)↙
*sin(yaw))/20 + (cos(yaw)*sin(pitch)*sin(roll))/20;Py3 =Y - (cos(pitch)*cos(yaw))/20 +↙
(sin(pitch)*sin(roll)*sin(yaw))/20;Pz3 =Z - (cos(roll)*sin(pitch))/20;axis([-0.2 0.2↙
-0.2 0.2 0 1.2]);plot(0.1,0.1);hold on;line([Bx1 Bx2],[By1 By2],[Bz1 Bz2]);line([Bx2↙
Bx3],[By2 By3],[Bz2 Bz3]);line([Bx1 Bx3],[By1 By3],[Bz1 Bz3]);line([Px1 Px2],[Py1 Py2],↙
[Pz1 Pz2]);line([Px2 Px3],[Py2 Py3],[Pz2 Pz3]);line([Px1 Px3],[Py1 Py3],[Pz1 Pz3]);line↙
([Px1 Bx3],[Py1 By3],[Pz1 Bz3],'color','r');line([Px1 Bx2],[Py1 By2],[Pz1↙
Bz2],'color','r');plot3(Px1,Py1,Pz1,'ro');line([Px3 Bx2],[Py3 By2],[Pz3↙
Bz2],'color','b');line([Px3 Bx1],[Py3 By1],[Pz3 Bz1],'color','b');plot3(Px3,Py3,↙
Pz3,'bo');line([Px2 Bx1],[Py2 By1],[Pz2 Bz1],'color','g');line([Px2 Bx3],[Py2 By3],[Pz2↙
Bz3],'color','g');plot3(Px2,Py2,Pz2,'go');hold off;axis([-0.2 0.2 -0.2 0.2 0 1.2]);↙
frame = getframe(fig);aviobj = addframe(aviobj,frame);
    pitch=pitch-0.1;
end
for n=1:20
    Px1 =X - (cos(roll)*cos(yaw))/20 - (cos(pitch)*sin(yaw))/20 - (cos(yaw)*sin(pitch)↙
*sin(roll))/20;Py1 =Y + (cos(pitch)*cos(yaw))/20 - (cos(roll)*sin(yaw))/20 - (sin↙
(pitch)*sin(roll)*sin(yaw))/20;Pz1 =Z - sin(roll)/20 + (cos(roll)*sin(pitch))/20;Px2 =X↙
+ (cos(roll)*cos(yaw))/20 - (cos(pitch)*sin(yaw))/20 - (cos(yaw)*sin(pitch)*sin(roll))↙
/20;Py2 =Y + (cos(pitch)*cos(yaw))/20 + (cos(roll)*sin(yaw))/20 - (sin(pitch)*sin(roll)↙
*sin(yaw))/20;Pz2 =Z + sin(roll)/20 + (cos(roll)*sin(pitch))/20;Px3 =X + (cos(pitch)↙
*sin(yaw))/20 + (cos(yaw)*sin(pitch)*sin(roll))/20;Py3 =Y - (cos(pitch)*cos(yaw))/20 +↙
(sin(pitch)*sin(roll)*sin(yaw))/20;Pz3 =Z - (cos(roll)*sin(pitch))/20;axis([-0.2 0.2↙
-0.2 0.2 0 1.2]);plot(0.1,0.1);hold on;line([Bx1 Bx2],[By1 By2],[Bz1 Bz2]);line([Bx2↙
Bx3],[By2 By3],[Bz2 Bz3]);line([Bx1 Bx3],[By1 By3],[Bz1 Bz3]);line([Px1 Px2],[Py1 Py2],↙
[Pz1 Pz2]);line([Px2 Px3],[Py2 Py3],[Pz2 Pz3]);line([Px1 Px3],[Py1 Py3],[Pz1 Pz3]);line↙
([Px1 Bx3],[Py1 By3],[Pz1 Bz3],'color','r');line([Px1 Bx2],[Py1 By2],[Pz1↙
Bz2],'color','r');plot3(Px1,Py1,Pz1,'ro');line([Px3 Bx2],[Py3 By2],[Pz3↙
Bz2],'color','b');line([Px3 Bx1],[Py3 By1],[Pz3 Bz1],'color','b');plot3(Px3,Py3,↙
Pz3,'bo');line([Px2 Bx1],[Py2 By1],[Pz2 Bz1],'color','g');line([Px2 Bx3],[Py2 By3],[Pz2↙
Bz3],'color','g');plot3(Px2,Py2,Pz2,'go');hold off;axis([-0.2 0.2 -0.2 0.2 0 1.2]);↙
frame = getframe(fig);aviobj = addframe(aviobj,frame);
    pitch=pitch+0.1;
end
for n=1:10
    Px1 =X - (cos(roll)*cos(yaw))/20 - (cos(pitch)*sin(yaw))/20 - (cos(yaw)*sin(pitch)↙
*sin(roll))/20;Py1 =Y + (cos(pitch)*cos(yaw))/20 - (cos(roll)*sin(yaw))/20 - (sin↙
(pitch)*sin(roll)*sin(yaw))/20;Pz1 =Z - sin(roll)/20 + (cos(roll)*sin(pitch))/20;Px2 =X↙
+ (cos(roll)*cos(yaw))/20 - (cos(pitch)*sin(yaw))/20 - (cos(yaw)*sin(pitch)*sin(roll))↙
/20;Py2 =Y + (cos(pitch)*cos(yaw))/20 + (cos(roll)*sin(yaw))/20 - (sin(pitch)*sin(roll)↙
*sin(yaw))/20;Pz2 =Z + sin(roll)/20 + (cos(roll)*sin(pitch))/20;Px3 =X + (cos(pitch)↙
*sin(yaw))/20 + (cos(yaw)*sin(pitch)*sin(roll))/20;Py3 =Y - (cos(pitch)*cos(yaw))/20 +↙
(sin(pitch)*sin(roll)*sin(yaw))/20;Pz3 =Z - (cos(roll)*sin(pitch))/20;axis([-0.2 0.2↙
-0.2 0.2 0 1.2]);plot(0.1,0.1);hold on;line([Bx1 Bx2],[By1 By2],[Bz1 Bz2]);line([Bx2↙
Bx3],[By2 By3],[Bz2 Bz3]);line([Bx1 Bx3],[By1 By3],[Bz1 Bz3]);line([Px1 Px2],[Py1 Py2],↙
```

```
[Pz1 Pz2]);line([Px2 Px3],[Py2 Py3],[Pz2 Pz3]);line([Px1 Px3],[Py1 Py3],[Pz1 Pz3]);line↙
([Px1 Bx3],[Py1 By3],[Pz1 Bz3],'color','r');line([Px1 Bx2],[Py1 By2],[Pz1↙
Bz2],'color','r');plot3(Px1,Py1,Pz1,'ro');line([Px3 Bx2],[Py3 By2],[Pz3↙
Bz2],'color','b');line([Px3 Bx1],[Py3 By1],[Pz3 Bz1],'color','b');plot3(Px3,Py3,↙
Pz3,'bo');line([Px2 Bx1],[Py2 By1],[Pz2 Bz1],'color','g');line([Px2 Bx3],[Py2 By3],[Pz2↙
Bz3],'color','g');plot3(Px2,Py2,Pz2,'go');hold off;axis([-0.2 0.2 -0.2 0.2 0 1.2]);↙
frame = getframe(fig);aviobj = addframe(aviobj,frame);
    pitch=pitch-0.1;
end
pitch=0;
for n=1:10
    Px1 =X - (cos(roll)*cos(yaw))/20 - (cos(pitch)*sin(yaw))/20 - (cos(yaw)*sin(pitch)↙
*sin(roll))/20;Py1 =Y + (cos(pitch)*cos(yaw))/20 - (cos(roll)*sin(yaw))/20 - (sin↙
(pitch)*sin(roll)*sin(yaw))/20;Pz1 =Z - sin(roll)/20 + (cos(roll)*sin(pitch))/20;Px2 =X↙
+ (cos(roll)*cos(yaw))/20 - (cos(pitch)*sin(yaw))/20 - (cos(yaw)*sin(pitch)*sin(roll))↙
/20;Py2 =Y + (cos(pitch)*cos(yaw))/20 + (cos(roll)*sin(yaw))/20 - (sin(pitch)*sin(roll)↙
*sin(yaw))/20;Pz2 =Z + sin(roll)/20 + (cos(roll)*sin(pitch))/20;Px3 =X + (cos(pitch)↙
*sin(yaw))/20 + (cos(yaw)*sin(pitch)*sin(roll))/20;Py3 =Y - (cos(pitch)*cos(yaw))/20 +↙
(sin(pitch)*sin(roll)*sin(yaw))/20;Pz3 =Z - (cos(roll)*sin(pitch))/20;axis([-0.2 0.2↙
-0.2 0.2 0 1.2]);plot(0.1,0.1);hold on;line([Bx1 Bx2],[By1 By2],[Bz1 Bz2]);line([Bx2↙
Bx3],[By2 By3],[Bz2 Bz3]);line([Bx1 Bx3],[By1 By3],[Bz1 Bz3]);line([Px1 Px2],[Py1 Py2],↙
[Pz1 Pz2]);line([Px2 Px3],[Py2 Py3],[Pz2 Pz3]);line([Px1 Px3],[Py1 Py3],[Pz1 Pz3]);line↙
([Px1 Bx3],[Py1 By3],[Pz1 Bz3],'color','r');line([Px1 Bx2],[Py1 By2],[Pz1↙
Bz2],'color','r');plot3(Px1,Py1,Pz1,'ro');line([Px3 Bx2],[Py3 By2],[Pz3↙
Bz2],'color','b');line([Px3 Bx1],[Py3 By1],[Pz3 Bz1],'color','b');plot3(Px3,Py3,↙
Pz3,'bo');line([Px2 Bx1],[Py2 By1],[Pz2 Bz1],'color','g');line([Px2 Bx3],[Py2 By3],[Pz2↙
Bz3],'color','g');plot3(Px2,Py2,Pz2,'go');hold off;axis([-0.2 0.2 -0.2 0.2 0 1.2]);↙
frame = getframe(fig);aviobj = addframe(aviobj,frame);
    roll=roll-0.1;
end
for n=1:20
    Px1 =X - (cos(roll)*cos(yaw))/20 - (cos(pitch)*sin(yaw))/20 - (cos(yaw)*sin(pitch)↙
*sin(roll))/20;Py1 =Y + (cos(pitch)*cos(yaw))/20 - (cos(roll)*sin(yaw))/20 - (sin↙
(pitch)*sin(roll)*sin(yaw))/20;Pz1 =Z - sin(roll)/20 + (cos(roll)*sin(pitch))/20;Px2 =X↙
+ (cos(roll)*cos(yaw))/20 - (cos(pitch)*sin(yaw))/20 - (cos(yaw)*sin(pitch)*sin(roll))↙
/20;Py2 =Y + (cos(pitch)*cos(yaw))/20 + (cos(roll)*sin(yaw))/20 - (sin(pitch)*sin(roll)↙
*sin(yaw))/20;Pz2 =Z + sin(roll)/20 + (cos(roll)*sin(pitch))/20;Px3 =X + (cos(pitch)↙
*sin(yaw))/20 + (cos(yaw)*sin(pitch)*sin(roll))/20;Py3 =Y - (cos(pitch)*cos(yaw))/20 +↙
(sin(pitch)*sin(roll)*sin(yaw))/20;Pz3 =Z - (cos(roll)*sin(pitch))/20;axis([-0.2 0.2↙
-0.2 0.2 0 1.2]);plot(0.1,0.1);hold on;line([Bx1 Bx2],[By1 By2],[Bz1 Bz2]);line([Bx2↙
Bx3],[By2 By3],[Bz2 Bz3]);line([Bx1 Bx3],[By1 By3],[Bz1 Bz3]);line([Px1 Px2],[Py1 Py2],↙
[Pz1 Pz2]);line([Px2 Px3],[Py2 Py3],[Pz2 Pz3]);line([Px1 Px3],[Py1 Py3],[Pz1 Pz3]);line↙
([Px1 Bx3],[Py1 By3],[Pz1 Bz3],'color','r');line([Px1 Bx2],[Py1 By2],[Pz1↙
Bz2],'color','r');plot3(Px1,Py1,Pz1,'ro');line([Px3 Bx2],[Py3 By2],[Pz3↙
Bz2],'color','b');line([Px3 Bx1],[Py3 By1],[Pz3 Bz1],'color','b');plot3(Px3,Py3,↙
Pz3,'bo');line([Px2 Bx1],[Py2 By1],[Pz2 Bz1],'color','g');line([Px2 Bx3],[Py2 By3],[Pz2↙
Bz3],'color','g');plot3(Px2,Py2,Pz2,'go');hold off;axis([-0.2 0.2 -0.2 0.2 0 1.2]);↙
frame = getframe(fig);aviobj = addframe(aviobj,frame);
    roll=roll+0.1;
end
for n=1:10
    Px1 =X - (cos(roll)*cos(yaw))/20 - (cos(pitch)*sin(yaw))/20 - (cos(yaw)*sin(pitch)↙
*sin(roll))/20;Py1 =Y + (cos(pitch)*cos(yaw))/20 - (cos(roll)*sin(yaw))/20 - (sin↙
(pitch)*sin(roll)*sin(yaw))/20;Pz1 =Z - sin(roll)/20 + (cos(roll)*sin(pitch))/20;Px2 =X↙
+ (cos(roll)*cos(yaw))/20 - (cos(pitch)*sin(yaw))/20 - (cos(yaw)*sin(pitch)*sin(roll))↙
/20;Py2 =Y + (cos(pitch)*cos(yaw))/20 + (cos(roll)*sin(yaw))/20 - (sin(pitch)*sin(roll)↙
*sin(yaw))/20;Pz2 =Z + sin(roll)/20 + (cos(roll)*sin(pitch))/20;Px3 =X + (cos(pitch)↙
*sin(yaw))/20 + (cos(yaw)*sin(pitch)*sin(roll))/20;Py3 =Y - (cos(pitch)*cos(yaw))/20 +↙
(sin(pitch)*sin(roll)*sin(yaw))/20;Pz3 =Z - (cos(roll)*sin(pitch))/20;axis([-0.2 0.2↙
-0.2 0.2 0 1.2]);plot(0.1,0.1);hold on;line([Bx1 Bx2],[By1 By2],[Bz1 Bz2]);line([Bx2↙
Bx3],[By2 By3],[Bz2 Bz3]);line([Bx1 Bx3],[By1 By3],[Bz1 Bz3]);line([Px1 Px2],[Py1 Py2],↙
[Pz1 Pz2]);line([Px2 Px3],[Py2 Py3],[Pz2 Pz3]);line([Px1 Px3],[Py1 Py3],[Pz1 Pz3]);line↙
([Px1 Bx3],[Py1 By3],[Pz1 Bz3],'color','r');line([Px1 Bx2],[Py1 By2],[Pz1↙
Bz2],'color','r');plot3(Px1,Py1,Pz1,'ro');line([Px3 Bx2],[Py3 By2],[Pz3↙
Bz2],'color','b');line([Px3 Bx1],[Py3 By1],[Pz3 Bz1],'color','b');plot3(Px3,Py3,↙
Pz3,'bo');line([Px2 Bx1],[Py2 By1],[Pz2 Bz1],'color','g');line([Px2 Bx3],[Py2 By3],[Pz2↙
Bz3],'color','g');plot3(Px2,Py2,Pz2,'go');hold off;axis([-0.2 0.2 -0.2 0.2 0 1.2]);↙
frame = getframe(fig);aviobj = addframe(aviobj,frame);
    roll=roll-0.1;
end
roll=0;

for n=1:10
    Px1 =X - (cos(roll)*cos(yaw))/20 - (cos(pitch)*sin(yaw))/20 - (cos(yaw)*sin(pitch)↙
*sin(roll))/20;Py1 =Y + (cos(pitch)*cos(yaw))/20 - (cos(roll)*sin(yaw))/20 - (sin↙
(pitch)*sin(roll)*sin(yaw))/20;Pz1 =Z - sin(roll)/20 + (cos(roll)*sin(pitch))/20;Px2 =X↙
+ (cos(roll)*cos(yaw))/20 - (cos(pitch)*sin(yaw))/20 - (cos(yaw)*sin(pitch)*sin(roll))↙
/20;Py2 =Y + (cos(pitch)*cos(yaw))/20 + (cos(roll)*sin(yaw))/20 - (sin(pitch)*sin(roll)↙
*sin(yaw))/20;Pz2 =Z + sin(roll)/20 + (cos(roll)*sin(pitch))/20;Px3 =X + (cos(pitch)↙
*sin(yaw))/20 + (cos(yaw)*sin(pitch)*sin(roll))/20;Py3 =Y - (cos(pitch)*cos(yaw))/20 +↙
(sin(pitch)*sin(roll)*sin(yaw))/20;Pz3 =Z - (cos(roll)*sin(pitch))/20;axis([-0.2 0.2↙
-0.2 0.2 0 1.2]);plot(0.1,0.1);hold on;line([Bx1 Bx2],[By1 By2],[Bz1 Bz2]);line([Bx2↙
Bx3],[By2 By3],[Bz2 Bz3]);line([Bx1 Bx3],[By1 By3],[Bz1 Bz3]);line([Px1 Px2],[Py1 Py2],↙
[Pz1 Pz2]);line([Px2 Px3],[Py2 Py3],[Pz2 Pz3]);line([Px1 Px3],[Py1 Py3],[Pz1 Pz3]);line↙
([Px1 Bx3],[Py1 By3],[Pz1 Bz3],'color','r');line([Px1 Bx2],[Py1 By2],[Pz1↙
Bz2],'color','r');plot3(Px1,Py1,Pz1,'ro');line([Px3 Bx2],[Py3 By2],[Pz3↙
Bz2],'color','b');line([Px3 Bx1],[Py3 By1],[Pz3 Bz1],'color','b');plot3(Px3,Py3,↙
Pz3,'bo');line([Px2 Bx1],[Py2 By1],[Pz2 Bz1],'color','g');line([Px2 Bx3],[Py2 By3],[Pz2↙
Bz3],'color','g');plot3(Px2,Py2,Pz2,'go');hold off;axis([-0.2 0.2 -0.2 0.2 0 1.2]);↙
frame = getframe(fig);aviobj = addframe(aviobj,frame);
    X=X-0.01;
end
for n=1:20
    Px1 =X - (cos(roll)*cos(yaw))/20 - (cos(pitch)*sin(yaw))/20 - (cos(yaw)*sin(pitch)↙
*sin(roll))/20;Py1 =Y + (cos(pitch)*cos(yaw))/20 - (cos(roll)*sin(yaw))/20 - (sin↙
(pitch)*sin(roll)*sin(yaw))/20;Pz1 =Z - sin(roll)/20 + (cos(roll)*sin(pitch))/20;Px2 =X↙
+ (cos(roll)*cos(yaw))/20 - (cos(pitch)*sin(yaw))/20 - (cos(yaw)*sin(pitch)*sin(roll))↙
/20;Py2 =Y + (cos(pitch)*cos(yaw))/20 + (cos(roll)*sin(yaw))/20 - (sin(pitch)*sin(roll)↙
*sin(yaw))/20;Pz2 =Z + sin(roll)/20 + (cos(roll)*sin(pitch))/20;Px3 =X + (cos(pitch)↙
*sin(yaw))/20 + (cos(yaw)*sin(pitch)*sin(roll))/20;Py3 =Y - (cos(pitch)*cos(yaw))/20 +↙
(sin(pitch)*sin(roll)*sin(yaw))/20;Pz3 =Z - (cos(roll)*sin(pitch))/20;axis([-0.2 0.2↙
-0.2 0.2 0 1.2]);plot(0.1,0.1);hold on;line([Bx1 Bx2],[By1 By2],[Bz1 Bz2]);line([Bx2↙
Bx3],[By2 By3],[Bz2 Bz3]);line([Bx1 Bx3],[By1 By3],[Bz1 Bz3]);line([Px1 Px2],[Py1 Py2],↙
[Pz1 Pz2]);line([Px2 Px3],[Py2 Py3],[Pz2 Pz3]);line([Px1 Px3],[Py1 Py3],[Pz1 Pz3]);line↙
([Px1 Bx3],[Py1 By3],[Pz1 Bz3],'color','r');line([Px1 Bx2],[Py1 By2],[Pz1↙
Bz2],'color','r');plot3(Px1,Py1,Pz1,'ro');line([Px3 Bx2],[Py3 By2],[Pz3↙
Bz2],'color','b');line([Px3 Bx1],[Py3 By1],[Pz3 Bz1],'color','b');plot3(Px3,Py3,↙
Pz3,'bo');line([Px2 Bx1],[Py2 By1],[Pz2 Bz1],'color','g');line([Px2 Bx3],[Py2 By3],[Pz2↙
Bz3],'color','g');plot3(Px2,Py2,Pz2,'go');hold off;axis([-0.2 0.2 -0.2 0.2 0 1.2]);↙
frame = getframe(fig);aviobj = addframe(aviobj,frame);
    X=X+0.01;
end
for n=1:10
    Px1 =X - (cos(roll)*cos(yaw))/20 - (cos(pitch)*sin(yaw))/20 - (cos(yaw)*sin(pitch)↙
*sin(roll))/20;Py1 =Y + (cos(pitch)*cos(yaw))/20 - (cos(roll)*sin(yaw))/20 - (sin↙
(pitch)*sin(roll)*sin(yaw))/20;Pz1 =Z - sin(roll)/20 + (cos(roll)*sin(pitch))/20;Px2 =X↙
+ (cos(roll)*cos(yaw))/20 - (cos(pitch)*sin(yaw))/20 - (cos(yaw)*sin(pitch)*sin(roll))↙
/20;Py2 =Y + (cos(pitch)*cos(yaw))/20 + (cos(roll)*sin(yaw))/20 - (sin(pitch)*sin(roll)↙
*sin(yaw))/20;Pz2 =Z + sin(roll)/20 + (cos(roll)*sin(pitch))/20;Px3 =X + (cos(pitch)↙
*sin(yaw))/20 + (cos(yaw)*sin(pitch)*sin(roll))/20;Py3 =Y - (cos(pitch)*cos(yaw))/20 +↙
```

```
(sin(pitch)*sin(roll)*sin(yaw))/20;Pz3 =Z - (cos(roll)*sin(pitch))/20;axis([-0.2 0.2↙
-0.2 0.2 0 1.2]);plot(0.1,0.1);hold on;line([Bx1 Bx2],[By1 By2],[Bz1 Bz2]);line([Bx2↙
Bx3],[By2 By3],[Bz2 Bz3]);line([Bx1 Bx3],[By1 By3],[Bz1 Bz3]);line([Px1 Px2],[Py1 Py2],↙
[Pz1 Pz2]);line([Px2 Px3],[Py2 Py3],[Pz2 Pz3]);line([Px1 Px3],[Py1 Py3],[Pz1 Pz3]);line↙
([Px1 Bx3],[Py1 By3],[Pz1 Bz3],'color','r');line([Px1 Bx2],[Py1 By2],[Pz1↙
Bz2],'color','r');plot3(Px1,Py1,Pz1,'ro');line([Px3 Bx2],[Py3 By2],[Pz3↙
Bz2],'color','b');line([Px3 Bx1],[Py3 By1],[Pz3 Bz1],'color','b');plot3(Px3,Py3,↙
Pz3,'bo');line([Px2 Bx1],[Py2 By1],[Pz2 Bz1],'color','g');line([Px2 Bx3],[Py2 By3],[Pz2↙
Bz3],'color','g');plot3(Px2,Py2,Pz2,'go');hold off;axis([-0.2 0.2 -0.2 0.2 0 1.2]);↙
frame = getframe(fig);aviobj = addframe(aviobj,frame);
    X=X-0.01;
end
X=0;
for n=1:10
    Px1 =X - (cos(roll)*cos(yaw))/20 - (cos(pitch)*sin(yaw))/20 - (cos(yaw)*sin(pitch)↙
*sin(roll))/20;Py1 =Y + (cos(pitch)*cos(yaw))/20 - (cos(roll)*sin(yaw))/20 - (sin↙
(pitch)*sin(roll)*sin(yaw))/20;Pz1 =Z - sin(roll)/20 + (cos(roll)*sin(pitch))/20;Px2 =X↙
+ (cos(roll)*cos(yaw))/20 - (cos(pitch)*sin(yaw))/20 - (cos(yaw)*sin(pitch)*sin(roll))↙
/20;Py2 =Y + (cos(pitch)*cos(yaw))/20 + (cos(roll)*sin(yaw))/20 - (sin(pitch)*sin(roll)↙
*sin(yaw))/20;Pz2 =Z + sin(roll)/20 + (cos(roll)*sin(pitch))/20;Px3 =X + (cos(pitch)↙
*sin(yaw))/20 + (cos(yaw)*sin(pitch)*sin(roll))/20;Py3 =Y - (cos(pitch)*cos(yaw))/20 +↙
(sin(pitch)*sin(roll)*sin(yaw))/20;Pz3 =Z - (cos(roll)*sin(pitch))/20;axis([-0.2 0.2↙
-0.2 0.2 0 1.2]);plot(0.1,0.1);hold on;line([Bx1 Bx2],[By1 By2],[Bz1 Bz2]);line([Bx2↙
Bx3],[By2 By3],[Bz2 Bz3]);line([Bx1 Bx3],[By1 By3],[Bz1 Bz3]);line([Px1 Px2],[Py1 Py2],↙
[Pz1 Pz2]);line([Px2 Px3],[Py2 Py3],[Pz2 Pz3]);line([Px1 Px3],[Py1 Py3],[Pz1 Pz3]);line↙
([Px1 Bx3],[Py1 By3],[Pz1 Bz3],'color','r');line([Px1 Bx2],[Py1 By2],[Pz1↙
Bz2],'color','r');plot3(Px1,Py1,Pz1,'ro');line([Px3 Bx2],[Py3 By2],[Pz3↙
Bz2],'color','b');line([Px3 Bx1],[Py3 By1],[Pz3 Bz1],'color','b');plot3(Px3,Py3,↙
Pz3,'bo');line([Px2 Bx1],[Py2 By1],[Pz2 Bz1],'color','g');line([Px2 Bx3],[Py2 By3],[Pz2↙
Bz3],'color','g');plot3(Px2,Py2,Pz2,'go');hold off;axis([-0.2 0.2 -0.2 0.2 0 1.2]);↙
frame = getframe(fig);aviobj = addframe(aviobj,frame);
    Y=Y-0.01;
end
for n=1:20
    Px1 =X - (cos(roll)*cos(yaw))/20 - (cos(pitch)*sin(yaw))/20 - (cos(yaw)*sin(pitch)↙
*sin(roll))/20;Py1 =Y + (cos(pitch)*cos(yaw))/20 - (cos(roll)*sin(yaw))/20 - (sin↙
(pitch)*sin(roll)*sin(yaw))/20;Pz1 =Z - sin(roll)/20 + (cos(roll)*sin(pitch))/20;Px2 =X↙
+ (cos(roll)*cos(yaw))/20 - (cos(pitch)*sin(yaw))/20 - (cos(yaw)*sin(pitch)*sin(roll))↙
/20;Py2 =Y + (cos(pitch)*cos(yaw))/20 + (cos(roll)*sin(yaw))/20 - (sin(pitch)*sin(roll)↙
*sin(yaw))/20;Pz2 =Z + sin(roll)/20 + (cos(roll)*sin(pitch))/20;Px3 =X + (cos(pitch)↙
*sin(yaw))/20 + (cos(yaw)*sin(pitch)*sin(roll))/20;Py3 =Y - (cos(pitch)*cos(yaw))/20 +↙
(sin(pitch)*sin(roll)*sin(yaw))/20;Pz3 =Z - (cos(roll)*sin(pitch))/20;axis([-0.2 0.2↙
-0.2 0.2 0 1.2]);plot(0.1,0.1);hold on;line([Bx1 Bx2],[By1 By2],[Bz1 Bz2]);line([Bx2↙
Bx3],[By2 By3],[Bz2 Bz3]);line([Bx1 Bx3],[By1 By3],[Bz1 Bz3]);line([Px1 Px2],[Py1 Py2],↙
[Pz1 Pz2]);line([Px2 Px3],[Py2 Py3],[Pz2 Pz3]);line([Px1 Px3],[Py1 Py3],[Pz1 Pz3]);line↙
([Px1 Bx3],[Py1 By3],[Pz1 Bz3],'color','r');line([Px1 Bx2],[Py1 By2],[Pz1↙
Bz2],'color','r');plot3(Px1,Py1,Pz1,'ro');line([Px3 Bx2],[Py3 By2],[Pz3↙
Bz2],'color','b');line([Px3 Bx1],[Py3 By1],[Pz3 Bz1],'color','b');plot3(Px3,Py3,↙
Pz3,'bo');line([Px2 Bx1],[Py2 By1],[Pz2 Bz1],'color','g');line([Px2 Bx3],[Py2 By3],[Pz2↙
Bz3],'color','g');plot3(Px2,Py2,Pz2,'go');hold off;axis([-0.2 0.2 -0.2 0.2 0 1.2]);↙
frame = getframe(fig);aviobj = addframe(aviobj,frame);
    Y=Y+0.01;
end
for n=1:10
    Px1 =X - (cos(roll)*cos(yaw))/20 - (cos(pitch)*sin(yaw))/20 - (cos(yaw)*sin(pitch)↙
*sin(roll))/20;Py1 =Y + (cos(pitch)*cos(yaw))/20 - (cos(roll)*sin(yaw))/20 - (sin↙
(pitch)*sin(roll)*sin(yaw))/20;Pz1 =Z - sin(roll)/20 + (cos(roll)*sin(pitch))/20;Px2 =X↙
+ (cos(roll)*cos(yaw))/20 - (cos(pitch)*sin(yaw))/20 - (cos(yaw)*sin(pitch)*sin(roll))↙
/20;Py2 =Y + (cos(pitch)*cos(yaw))/20 + (cos(roll)*sin(yaw))/20 - (sin(pitch)*sin(roll)↙
*sin(yaw))/20;Pz2 =Z + sin(roll)/20 + (cos(roll)*sin(pitch))/20;Px3 =X + (cos(pitch)↙
```

```
*sin(yaw))/20 + (cos(yaw)*sin(pitch)*sin(roll))/20;Py3 =Y - (cos(pitch)*cos(yaw))/20 +↙
(sin(pitch)*sin(roll)*sin(yaw))/20;Pz3 =Z - (cos(roll)*sin(pitch))/20;axis([-0.2 0.2↙
-0.2 0.2 0 1.2]);plot(0.1,0.1);hold on;line([Bx1 Bx2],[By1 By2],[Bz1 Bz2]);line([Bx2↙
Bx3],[By2 By3],[Bz2 Bz3]);line([Bx1 Bx3],[By1 By3],[Bz1 Bz3]);line([Px1 Px2],[Py1 Py2],↙
[Pz1 Pz2]);line([Px2 Px3],[Py2 Py3],[Pz2 Pz3]);line([Px1 Px3],[Py1 Py3],[Pz1 Pz3]);line↙
([Px1 Bx3],[Py1 By3],[Pz1 Bz3],'color','r');line([Px1 Bx2],[Py1 By2],[Pz1↙
Bz2],'color','r');plot3(Px1,Py1,Pz1,'ro');line([Px3 Bx2],[Py3 By2],[Pz3↙
Bz2],'color','b');line([Px3 Bx1],[Py3 By1],[Pz3 Bz1],'color','b');plot3(Px3,Py3,↙
Pz3,'bo');line([Px2 Bx1],[Py2 By1],[Pz2 Bz1],'color','g');line([Px2 Bx3],[Py2 By3],[Pz2↙
Bz3],'color','g');plot3(Px2,Py2,Pz2,'go');hold off;axis([-0.2 0.2 -0.2 0.2 0 1.2]);↙
frame = getframe(fig);aviobj = addframe(aviobj,frame);
    Y=Y-0.01;
end
Y=0;
for n=1:10
    Px1 =X - (cos(roll)*cos(yaw))/20 - (cos(pitch)*sin(yaw))/20 - (cos(yaw)*sin(pitch)↙
*sin(roll))/20;Py1 =Y + (cos(pitch)*cos(yaw))/20 - (cos(roll)*sin(yaw))/20 - (sin↙
(pitch)*sin(roll)*sin(yaw))/20;Pz1 =Z - sin(roll)/20 + (cos(roll)*sin(pitch))/20;Px2 =X↙
+ (cos(roll)*cos(yaw))/20 - (cos(pitch)*sin(yaw))/20 - (cos(yaw)*sin(pitch)*sin(roll))↙
/20;Py2 =Y + (cos(pitch)*cos(yaw))/20 + (cos(roll)*sin(yaw))/20 - (sin(pitch)*sin(roll)↙
*sin(yaw))/20;Pz2 =Z + sin(roll)/20 + (cos(roll)*sin(pitch))/20;Px3 =X + (cos(pitch)↙
*sin(yaw))/20 + (cos(yaw)*sin(pitch)*sin(roll))/20;Py3 =Y - (cos(pitch)*cos(yaw))/20 +↙
(sin(pitch)*sin(roll)*sin(yaw))/20;Pz3 =Z - (cos(roll)*sin(pitch))/20;axis([-0.2 0.2↙
-0.2 0.2 0 1.2]);plot(0.1,0.1);hold on;line([Bx1 Bx2],[By1 By2],[Bz1 Bz2]);line([Bx2↙
Bx3],[By2 By3],[Bz2 Bz3]);line([Bx1 Bx3],[By1 By3],[Bz1 Bz3]);line([Px1 Px2],[Py1 Py2],↙
[Pz1 Pz2]);line([Px2 Px3],[Py2 Py3],[Pz2 Pz3]);line([Px1 Px3],[Py1 Py3],[Pz1 Pz3]);line↙
([Px1 Bx3],[Py1 By3],[Pz1 Bz3],'color','r');line([Px1 Bx2],[Py1 By2],[Pz1↙
Bz2],'color','r');plot3(Px1,Py1,Pz1,'ro');line([Px3 Bx2],[Py3 By2],[Pz3↙
Bz2],'color','b');line([Px3 Bx1],[Py3 By1],[Pz3 Bz1],'color','b');plot3(Px3,Py3,↙
Pz3,'bo');line([Px2 Bx1],[Py2 By1],[Pz2 Bz1],'color','g');line([Px2 Bx3],[Py2 By3],[Pz2↙
Bz3],'color','g');plot3(Px2,Py2,Pz2,'go');hold off;axis([-0.2 0.2 -0.2 0.2 0 1.2]);↙
frame = getframe(fig);aviobj = addframe(aviobj,frame);
    Z=Z-0.05;
end
for n=1:20
    Px1 =X - (cos(roll)*cos(yaw))/20 - (cos(pitch)*sin(yaw))/20 - (cos(yaw)*sin(pitch)↙
*sin(roll))/20;Py1 =Y + (cos(pitch)*cos(yaw))/20 - (cos(roll)*sin(yaw))/20 - (sin↙
(pitch)*sin(roll)*sin(yaw))/20;Pz1 =Z - sin(roll)/20 + (cos(roll)*sin(pitch))/20;Px2 =X↙
+ (cos(roll)*cos(yaw))/20 - (cos(pitch)*sin(yaw))/20 - (cos(yaw)*sin(pitch)*sin(roll))↙
/20;Py2 =Y + (cos(pitch)*cos(yaw))/20 + (cos(roll)*sin(yaw))/20 - (sin(pitch)*sin(roll)↙
*sin(yaw))/20;Pz2 =Z + sin(roll)/20 + (cos(roll)*sin(pitch))/20;Px3 =X + (cos(pitch)↙
*sin(yaw))/20 + (cos(yaw)*sin(pitch)*sin(roll))/20;Py3 =Y - (cos(pitch)*cos(yaw))/20 +↙
(sin(pitch)*sin(roll)*sin(yaw))/20;Pz3 =Z - (cos(roll)*sin(pitch))/20;axis([-0.2 0.2↙
-0.2 0.2 0 1.2]);plot(0.1,0.1);hold on;line([Bx1 Bx2],[By1 By2],[Bz1 Bz2]);line([Bx2↙
Bx3],[By2 By3],[Bz2 Bz3]);line([Bx1 Bx3],[By1 By3],[Bz1 Bz3]);line([Px1 Px2],[Py1 Py2],↙
[Pz1 Pz2]);line([Px2 Px3],[Py2 Py3],[Pz2 Pz3]);line([Px1 Px3],[Py1 Py3],[Pz1 Pz3]);line↙
([Px1 Bx3],[Py1 By3],[Pz1 Bz3],'color','r');line([Px1 Bx2],[Py1 By2],[Pz1↙
Bz2],'color','r');plot3(Px1,Py1,Pz1,'ro');line([Px3 Bx2],[Py3 By2],[Pz3↙
Bz2],'color','b');line([Px3 Bx1],[Py3 By1],[Pz3 Bz1],'color','b');plot3(Px3,Py3,↙
Pz3,'bo');line([Px2 Bx1],[Py2 By1],[Pz2 Bz1],'color','g');line([Px2 Bx3],[Py2 By3],[Pz2↙
Bz3],'color','g');plot3(Px2,Py2,Pz2,'go');hold off;axis([-0.2 0.2 -0.2 0.2 0 1.2]);↙
frame = getframe(fig);aviobj = addframe(aviobj,frame);
    Z=Z+0.05;
end
for n=1:10
    Px1 =X - (cos(roll)*cos(yaw))/20 - (cos(pitch)*sin(yaw))/20 - (cos(yaw)*sin(pitch)↙
*sin(roll))/20;Py1 =Y + (cos(pitch)*cos(yaw))/20 - (cos(roll)*sin(yaw))/20 - (sin↙
(pitch)*sin(roll)*sin(yaw))/20;Pz1 =Z - sin(roll)/20 + (cos(roll)*sin(pitch))/20;Px2 =X↙
+ (cos(roll)*cos(yaw))/20 - (cos(pitch)*sin(yaw))/20 - (cos(yaw)*sin(pitch)*sin(roll))↙
/20;Py2 =Y + (cos(pitch)*cos(yaw))/20 + (cos(roll)*sin(yaw))/20 - (sin(pitch)*sin(roll)↙
*sin(yaw))/20;Pz2 =Z + sin(roll)/20 + (cos(roll)*sin(pitch))/20;Px3 =X + (cos(pitch)↙
*sin(yaw))/20 + (cos(yaw)*sin(pitch)*sin(roll))/20;Py3 =Y - (cos(pitch)*cos(yaw))/20 +↙
(sin(pitch)*sin(roll)*sin(yaw))/20;Pz3 =Z - (cos(roll)*sin(pitch))/20;axis([-0.2 0.2↙
-0.2 0.2 0 1.2]);plot(0.1,0.1);hold on;line([Bx1 Bx2],[By1 By2],[Bz1 Bz2]);line([Bx2↙
Bx3],[By2 By3],[Bz2 Bz3]);line([Bx1 Bx3],[By1 By3],[Bz1 Bz3]);line([Px1 Px2],[Py1 Py2],↙
[Pz1 Pz2]);line([Px2 Px3],[Py2 Py3],[Pz2 Pz3]);line([Px1 Px3],[Py1 Py3],[Pz1 Pz3]);line↙
([Px1 Bx3],[Py1 By3],[Pz1 Bz3],'color','r');line([Px1 Bx2],[Py1 By2],[Pz1↙
Bz2],'color','r');plot3(Px1,Py1,Pz1,'ro');line([Px3 Bx2],[Py3 By2],[Pz3↙
Bz2],'color','b');line([Px3 Bx1],[Py3 By1],[Pz3 Bz1],'color','b');plot3(Px3,Py3,↙
Pz3,'bo');line([Px2 Bx1],[Py2 By1],[Pz2 Bz1],'color','g');line([Px2 Bx3],[Py2 By3],[Pz2↙
Bz3],'color','g');plot3(Px2,Py2,Pz2,'go');hold off;axis([-0.2 0.2 -0.2 0.2 0 1.2]);↙
frame = getframe(fig);aviobj = addframe(aviobj,frame);
    Z=Z-0.05;
end
Z=0;


aviobj = close(aviobj)
```

A.4 Kinematics, Jacobian Non Redundant Manipulator

```
%                DH TABLE
% |---------------------------------------|
% |  RotZ  |  TransZ  |  TransX  |  RotX  |
% |---------------------------------------|
% |        |          |          |   -90  |
% |    t1  |          |     L1   |        |
% |    t2  |          |     L2   |        |
% |---------------------------------------|

clear all
% Calcutating G matrix for point P
clc
close all
syms t1 t2 L1 L2 Xdot Ydot

RotX1=[1,0,0,0;0,0,-1,0;0,1,0,0;0,0,0,1];
RotZ2=[cos(t1),-sin(t1),0,0;sin(t1),cos(t1),0,0;0,0,1,0;0,0,0,1];
TransX2=[1,0,0,L1;0,1,0,0;0,0,1,0;0,0,0,1];
RotZ3=[cos(t2),-sin(t2),0,0;sin(t2),cos(t2),0,0;0,0,1,0;0,0,0,1];
TransX3=[1,0,0,L2;0,1,0,0;0,0,1,0;0,0,0,1];


P1=[0,0]
G1=(RotX1)*(RotZ2*TransX2);
P2=[G1(1,4),G1(3,4)]
G2=(RotX1)*(RotZ2*TransX2)*(RotZ3*TransX3);
P3=[G2(1,4),G2(3,4)]
G3=(RotX1)*(RotZ2*TransX2)*(RotZ3*TransX3);
P4=[G3(1,4),G3(3,4)]
%Forward Kinematics
 X=P4(1);
 Y=P4(2);
 %% JACOBIAN
  J=jacobian([X Y],[t1 t2])
  Jinv=inv(J)
  %Including Xdot and Ydot
  Qd=Jinv*[Xdot;Ydot]
```

A.5 Kinematics, Jacobian, Pseudo Inverse Redundant Manipulator

```
%                DH TABLE
% |---------------------------------------|
% |  RotZ  |  TransZ  |  TransX  |  RotX  |
% |---------------------------------------|
% |        |          |          |   -90  |
% |    t1  |          |     L1   |        |
% |    t2  |          |     L2   |        |
% |        |          |     d1   |        |
% |---------------------------------------|

clear all
% Calcutating G matrix for point P
clc
close all
syms t1 t2 L1 L2 d1 Xdot Ydot

RotX1=[1,0,0,0;0,0,-1,0;0,1,0,0;0,0,0,1];
RotZ2=[cos(t1),-sin(t1),0,0;sin(t1),cos(t1),0,0;0,0,1,0;0,0,0,1];
TransX2=[1,0,0,L1;0,1,0,0;0,0,1,0;0,0,0,1];
RotZ3=[cos(t2),-sin(t2),0,0;sin(t2),cos(t2),0,0;0,0,1,0;0,0,0,1];
TransX3=[1,0,0,L2;0,1,0,0;0,0,1,0;0,0,0,1];
TransX4=[1,0,0,d1;0,1,0,0;0,0,1,0;0,0,0,1];

P1=[0,0]
G1=(RotX1)*(RotZ2*TransX2);
P2=[G1(1,4),G1(3,4)]
G2=(RotX1)*(RotZ2*TransX2)*(RotZ3*TransX3);
P3=[G2(1,4),G2(3,4)]
G3=(RotX1)*(RotZ2*TransX2)*(RotZ3*TransX3)*(TransX4)
P4=[G3(1,4),G3(3,4)]
%% Kinematics
 X=P4(1);
 Y=P4(2);
 %% Jacobian
  J=jacobian([X Y],[t1 t2 d1])
  %psudoinverse J+
  Jp=transpose(J)*inv(J*transpose(J));%
  Qd=Jp*[Xdot;Ydot]
  lamda=1.5;
Jdls=transpose(J)*inv(J*transpose(J)+(lamda^2)*eye(2));%
  Qdls=Jdls*[Xdot;Ydot]
  lamda=0.5;
Jdlsnull=transpose(J)*inv(J*transpose(J)+(lamda^2)*eye(2));%
  Qdlsnull=Jdlsnull*[Xdot;Ydot]
```

A.6 Redundant Control based on Condition Algorithm Manipulator

```
close all
clear all
clc
L1=1;
L2=1;
J=0.4;
t1=pi/4;
t1min=60*pi/180;
t1max=120*pi/180;
t2min=-120*pi/180;
t2max=-60*pi/180;
t2=-pi/4;

P1 =[0,0];
P2 =[ L1*cos(t1), L1*sin(t1)];
P3 =[ L1*cos(t1) + L2*cos(t1)*cos(t2) - L2*sin(t1)*sin(t2), L1*sin(t1) + L2*cos(t1)*sin↙
(t2) + L2*cos(t2)*sin(t1)];
P4 =[ J*(cos(t1)*cos(t2) - sin(t1)*sin(t2)) + L1*cos(t1) + L2*cos(t1)*cos(t2) - L2*sin↙
(t1)*sin(t2), J*(cos(t1)*sin(t2) + cos(t2)*sin(t1)) + L1*sin(t1) + L2*cos(t1)*sin(t2) +↙
L2*cos(t2)*sin(t1)];
axis([-0.5 1.5 0 2]);
% hold on;
% plot(P1(1),P1(2),'ro');
% plot(P2(1),P2(2),'ro');
% plot(P3(1),P3(2),'ro');
% plot(P4(1),P4(2),'ro');
% line([P1(1) P2(1)],[P1(2) P2(2)]);
% line([P2(1) P3(1)],[P2(2) P3(2)]);
% line([P3(1) P4(1)],[P3(2) P4(2)]);

% X=f1(t1,t2,J)
% Y=f2(t1,t2,J)
% f3(t1,t2,J)=0



grid on
Y=1.8;
X=1;
for n=1:15
    Y=Y-0.1;
%Y=Y-0.1;


   d=sqrt(X^2+Y^2)

    J=d-1.5
    if J>0.35
        J=0.35;
    end
    if J<0.1
        J=0.1;
    end

t2=-acos((X^2+Y^2-L1^2-(L2+J)^2)/(2*L1*(L2+J)));
if (t2<t2min)
    t2=t2min;
elseif (t2>t2max)
    t2=t2max;
end
```

```
A=(L1+(L2+J)*cos(t2));
B=(L2+J)*sin(t2);
s1=(Y-(B/A)*X)/((A+(B^2)/A));
c1=(Y+(A/B)*X)/((B+(A^2)/B));
t1=atan2(s1,c1);
%t1=atan(s1/c1);

if (t1<t1min)
    t1=t1min;
    d1=sqrt((X-1)^2+(Y)^2)
    J=d1-L2;
    t2=-acos((X^2+Y^2-L1^2-(L2+J)^2)/(2*L1*(L2+J)));
elseif (t1>t1max)
    t1=t1max;

    d1=sqrt(X^2+(Y-1)^2)
    J=d1-L2;
    t2=-acos((X^2+Y^2-L1^2-(L2+J)^2)/(2*L1*(L2+J)));
end

hold on
line([0 cos(t1)*L1],[0 sin(t1)*L1])
line([cos(t1)*L1 (L2)*cos(t1 + t2) + L1*cos(t1)],[sin(t1)*L1 ((L2)*sin(t1 + t2)+L1*sin↙
(t1))])
line([(L2)*cos(t1 + t2) + L1*cos(t1) (L2+J)*cos(t1 + t2) + L1*cos(t1)],[((L2)*sin(t1 +↙
t2)+L1*sin(t1)) ((L2+J)*sin(t1 + t2)+L1*sin(t1))])
plot(X,Y,'go')
plot(cos(t1)*L1,sin(t1)*L1,'ro')
plot((L2)*cos(t1 + t2) + L1*cos(t1),((L2)*sin(t1 + t2)+L1*sin(t1)),'bo')
hold on
end
grid off
xlabel('X')
ylabel('Y')
```

A.7 Simulink System of Heave Compensation

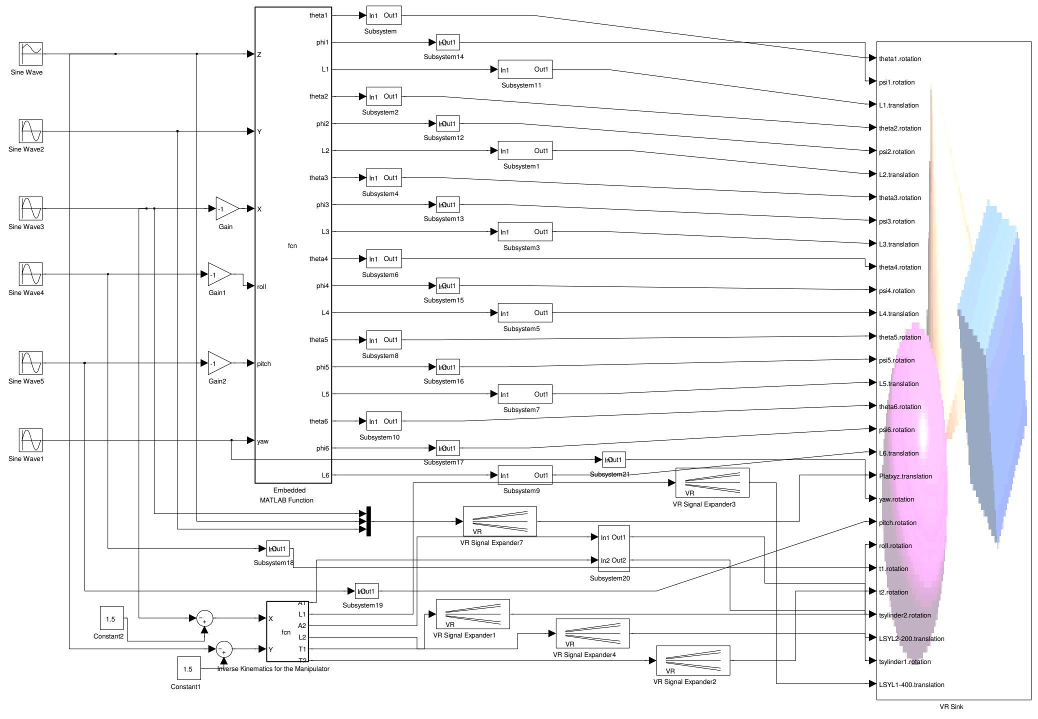


Figure A.1: *Simulink System of Heave Compensation*

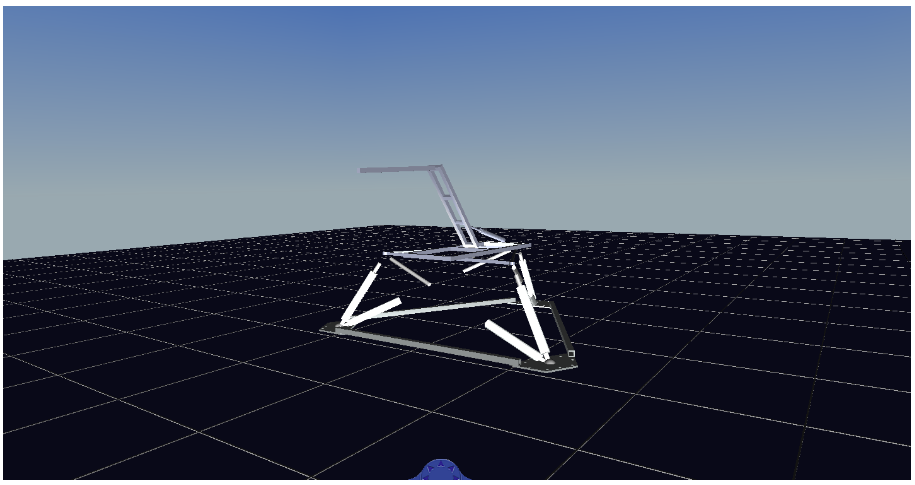


Figure A.2: *Animated Model of Heave Compensation System*

B. LABVIEW

B.1 Hydraulic Manipulator - LabView Diagrams

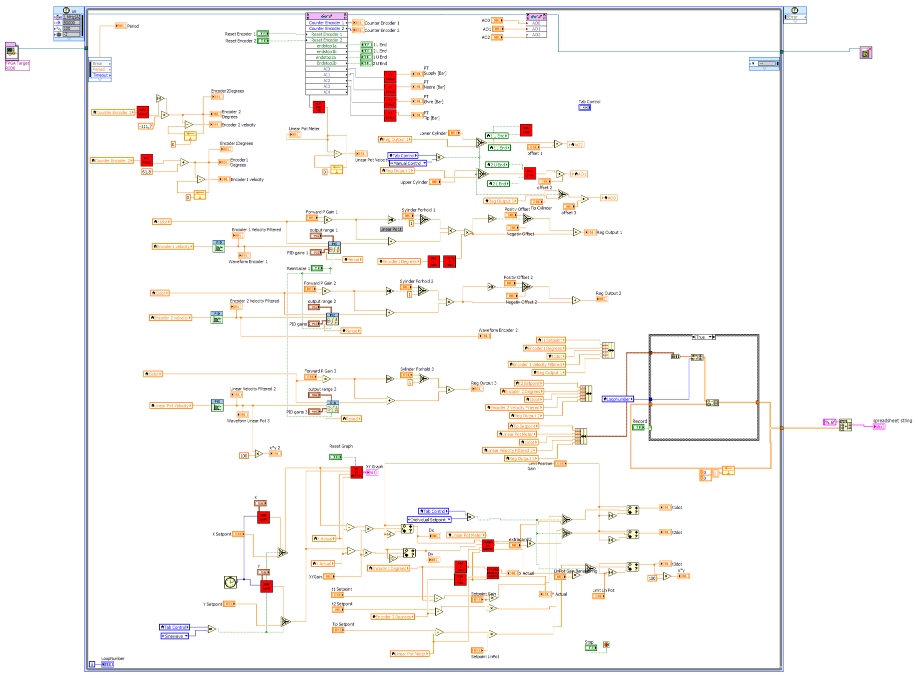


Figure B.1: *Main Program for Controlling the Hydraulic Manipulator*

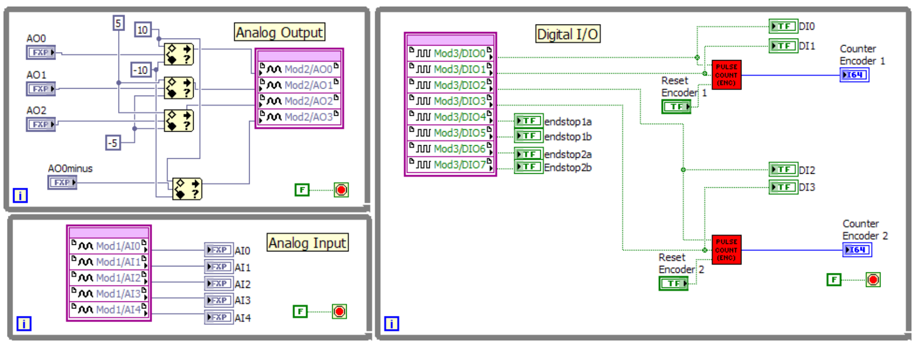


Figure B.2: *FPGA program for Controlling the Hydraulic Manipulator*

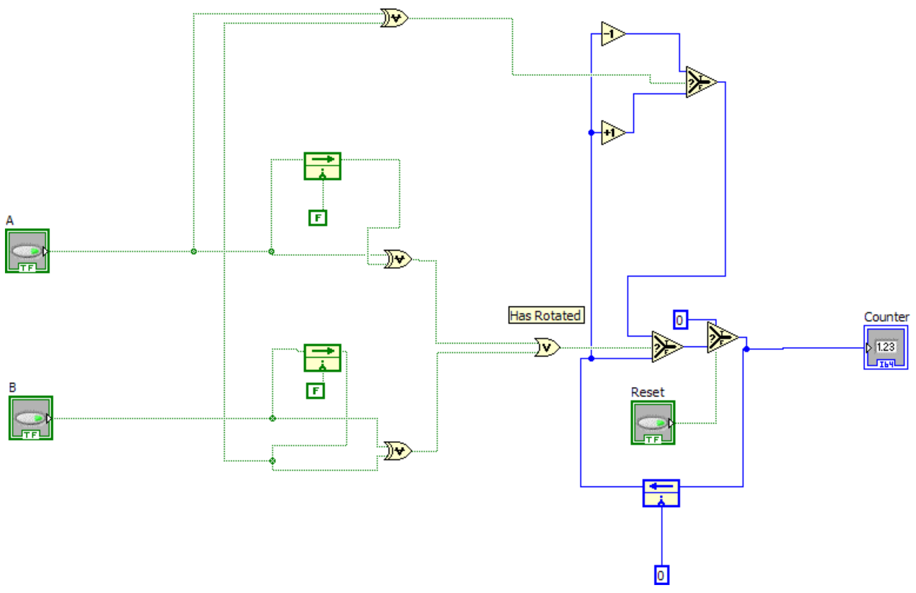
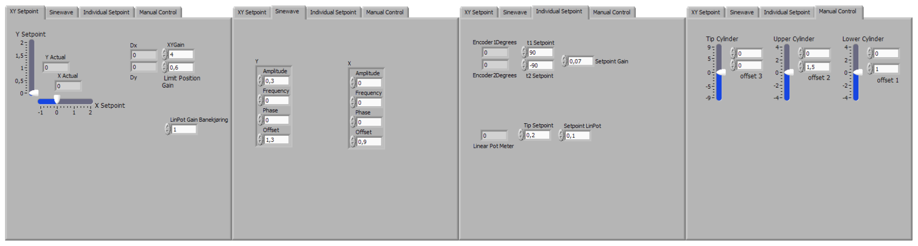


Figure B.3: *FPGA program for Counting the Steps for the Encoders*



Y Setpoint
Y Actual 0
X Actual 0
X Setpoint
Dx 0
Dy 0
XYGain 4
Limit Position Gain 0,6
LinPot Gain Banekjøring 1

Y
Amplitude 0,3
Frequency 0
Phase 0
Offset 1,3

X
Amplitude 0
Frequency 0
Phase 0
Offset 0,9

Encoder1Degrees 0
Encoder2Degrees 0
t1 Setpoint 90
t2 Setpoint -90
Setpoint Gain 0,07
Linear Pot Meter 0
Tip Setpoint 0,2
Setpoint LinPot 0,1

Tip Cylinder 0
offset 3 0
Upper Cylinder 0
offset 2 1,5
Lower Cylinder 0
offset 1 1

Figure B.4: *Control Interface for the Hydraulic Manipulator*

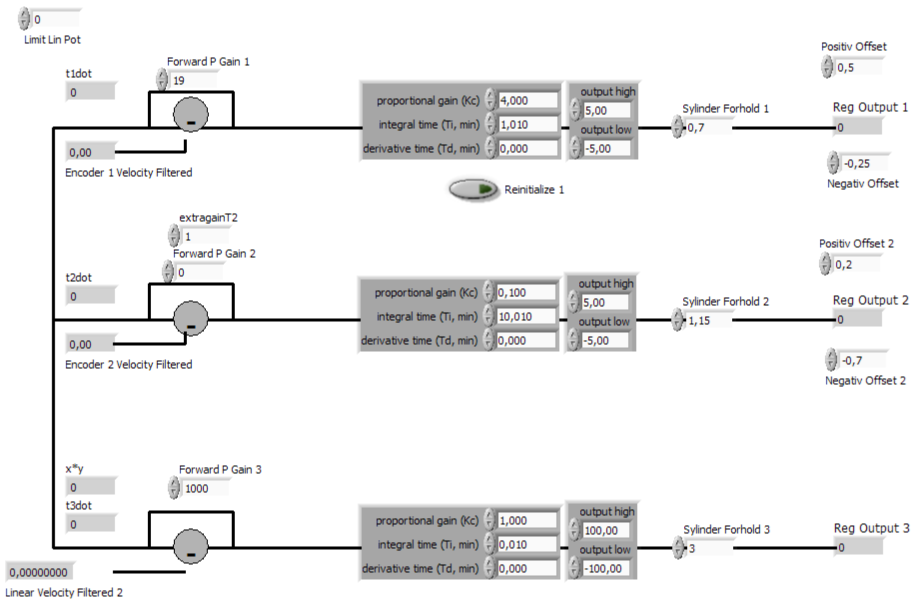
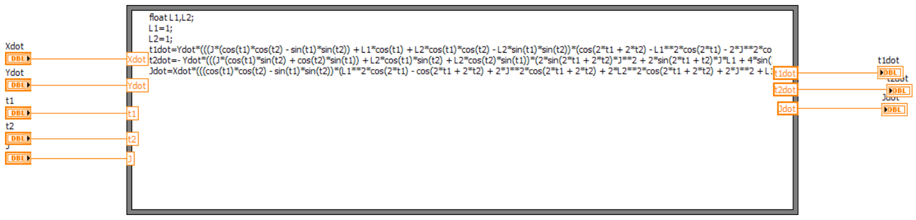


Figure B.5: *Regulator Interface for the Hydraulic Manipulator*



```
float L1,L2;
L1=1;
L2=1;
t1dot=Ydot*(((J*(cos(t1)*cos(t2) - sin(t1)*sin(t2)) + L1*cos(t1) + L2*cos(t1)*cos(t2) - L2*sin(t1)*sin(t2))*(cos(2*t1 + 2*t2) - L1**2*cos(2*t1) - 2*J**2*co
t2dot=- Ydot*(((J*(cos(t1)*sin(t2) + cos(t2)*sin(t1)) + L2*cos(t1)*sin(t2) + L2*cos(t2)*sin(t1))*(2*sin(2*t1 + 2*t2)*J**2 + 2*sin(2*t1 + t2)*J*L1 + 4*sin(
Jdot=Xdot*(((cos(t1)*cos(t2) - sin(t1)*sin(t2))*(L1**2*cos(2*t1) - cos(2*t1 + 2*t2) + 2*J**2*cos(2*t1 + 2*t2) + 2*L2**2*cos(2*t1 + 2*t2) + 2*J**2 + L
```

Figure B.6: *Pseudo-Inverse Jacobian in Formula Node*

B.2 Inertial Measurement Unit - LabView Diagrams

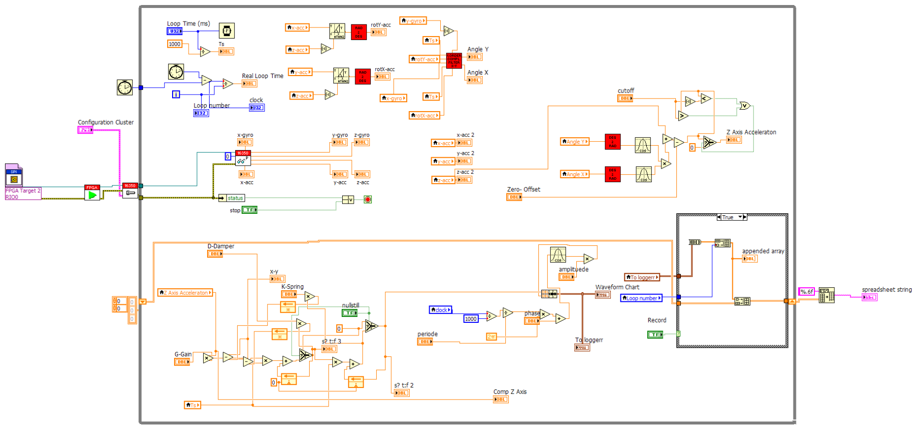


Figure B.7: *Program for Estimating angles from the IMU*

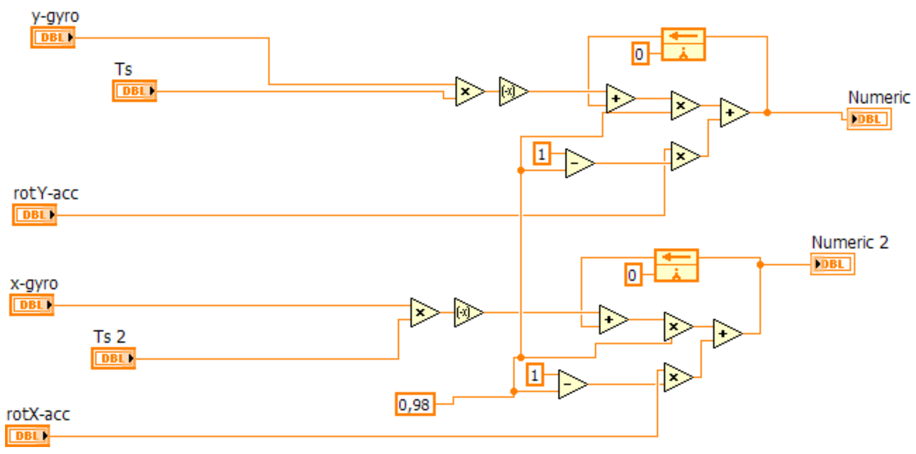


Figure B.8: *1.Order Complementary Filter*

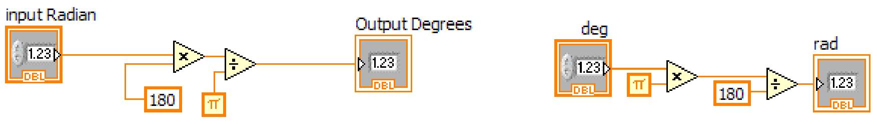


Figure B.9: *Degrees to Radians - Radians to Degrees*

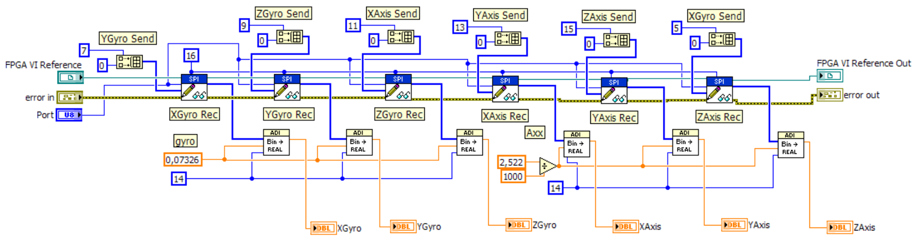


Figure B.10: *Read measurements from ADIS16350 by use of SPI*

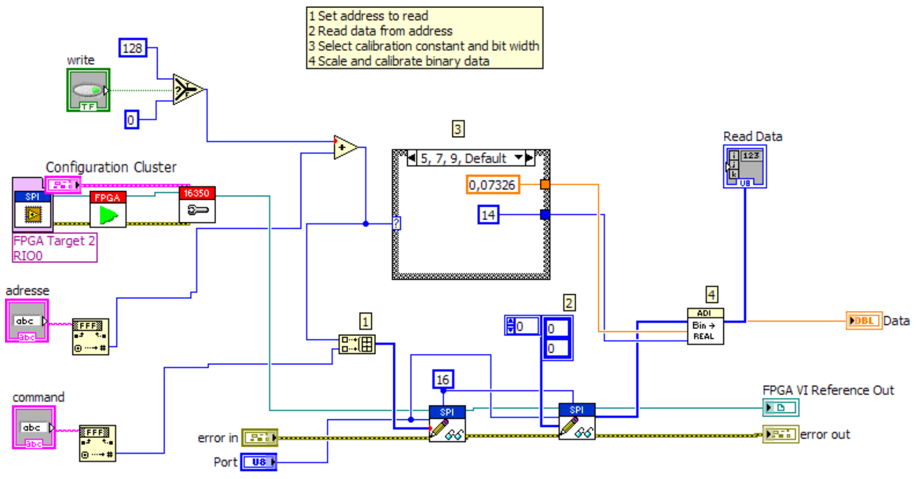


Figure B.11: *Setup Program for controlling the registers of the ADIS16350*

C. Hydraulic Components - Manipulator

1. **Hydraulic Actuator**
   - Type: LJM - NH30-SD-40/20 X 300-S-(TV)
   - Max Pressure: 250 bar
   - Piston Diameter: 40 mm
   - Rod Diameter: 20 mm
   - Stroke: 300 mm

2. **Hydraulic Actuator**
   - Type: Faroil - FA 40/20-100-125/125
   - Max Pressure: 250 bar
   - Piston Diameter: 40 mm
   - Rod Diameter: 20 mm
   - Stroke: 100 mm

3. **Hydraulic Actuator**
   - Type: LJM - NH30-S-SD- 40/ 20x 200-S
   - Max Pressure: 250 bar
   - Piston Diameter: 40 mm
   - Rod Diameter: 20 mm
   - Stroke: 200 mm

3. **Hydraulic Actuator**
   - Type: LJM - NH30-S-SD- 40/ 20x 200-S
   - Max Pressure: 250 bar
   - Piston Diameter: 40 mm
   - Rod Diameter: 20 mm
   - Stroke: 200 mm

D. Datasheets

D.1 Novo Technik - Position Transducer

novotechnik
Siedle Group

Position Transducers up to 750 mm

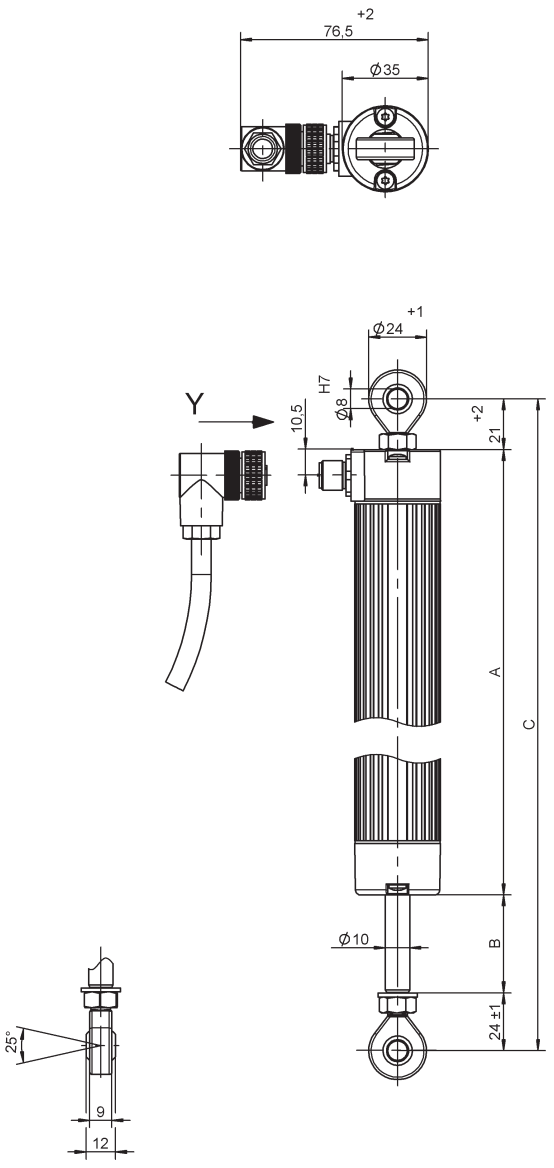
LWG Series

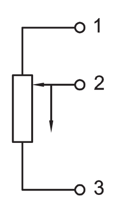
Special features

- twin-bearing actuating rod
- mountable over backlash-free pivot heads with a large angle of free movement (up to ±12.5°)
- outstanding linearity
- resolution better than 0.01 mm
- life (depending on application) up to 50 million movements
- protection class IP 65
- M12 Connector

The LWG series was designed for a wide range of demanding applications in the mechanical, vehicle, automation and robotics industries. Outstanding linearity and accurate measurement are achieved with a resistance element made of conductive plastic melded to a glass-fiber reinforced substrate.

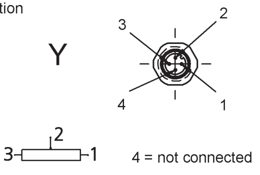
The wiper system is mounted on the actuating rod and coupled free-of-backlash for a long life and trouble-free operation.





Schematic

Connection pin assigment
Section



4 = not connected

Novotechnik U.S., Inc.
155 Northboro Road
Southborough, MA 01772

Phone: 508-485-2244
Fax: 508-485-2430
Email: info@novotechnik.com

Description	
Housing	aluminum, anodized
Fixings	see drawing
Actuator	stainless steel, rotatable
Bearings	sleeve bearing
Resistance element	conductive plastic
Wiper assembly	precious metal multi-finger wiper, elastomer damped
Electrical connections	4-pin M12 connector

Environmental Data		
Temperature range	-30...+100	°C
Vibration	5...2000 A_{max} = 0.75 a_{max} = 20	Hz mm g
Shock	50 11	g ms
Life	50 x 10^6 (typical)	movem.
Operating speed	5	m/s max.
Protection class	IP 65 (DIN 400 50 / IEC 529)	

Type designations	**LWG 75**	**LWG 100**	**LWG 150**	**LWG 225**	**LWG 300**	**LWG 360**	**LWG 450**	**LWG 500**	**LWG 600**	**LWG 750**	
Electrical Data											
Defined electrical range	75	100	150	225	300	360	450	500	600	750	mm
Electrical stroke	76	102	152	228	304	366	457	508	610	762	mm
Nominal resistance	3	3	5	5	5	5	5	5	5	10	kΩ
Resistance tolerance	20										±%
Independent linearity	0.1	0.1	0.08	0.07	0.06	0.05	0.05	0.05	0.05	0.04	%
Repeatability	< 0.01										mm
Recommended operating wiper current	≤ 1										µA
Max. wiper current in case of malfunction	10										mA
Max. permissible applied voltage	42										V
Effective temperature coefficient of the output-to-applied voltage ratio	typical 5										ppm/K
Insulation resistance (500 VDC, 1 bar, 2 s)	≥ 10										MΩ
Dielectric strength (50 Hz, 2 s, 1 bar, 500 VAC)	≤ 100										µA
Mechanical Data											
Body length (dimension A)	201	227	277	354	430	505	619	684	810	994	±2 mm
Mechanical stroke (dimension B)	79	105	155	231	307	368	460	510	612	764	±2 mm
Minimum distance between pivot heads (dimension C)	247	273	323	400	476	551	665	730	856	1040	mm
Operating force horizontal	3.6	3.7	4.0	4.5	4.9	5.2	5.7	6	6.6	7.5	N
Operating force vertical	7.4	7.6	8.0	8.7	9.3	9.8	10.6	11	11.9	13.2	N

Order designations

Type	**Art. no.**
LWG 75	026103
LWG 100	026104
LWG 150	026106
LWG 225	026109
LWG 300	026112
LWG 360	026114
LWG 450	026118
LWG 500	026120
LWG 600	026124
LWG 750	026130

Other lengths on request

Recommended accessories

Process-controlled indicators MAP...with display,
Signal conditioner MUP.../MUK… for standardized output signals

Important

All values given for this series – including linearity, lifetime, microlinearity, resistance to external disturbances and temperature coefficient in voltage dividing mode – are quoted for the device operating with the wiper voltage driving an operational amplifier working as a voltage follower where virtually no load is applied to the wiper (Ie ≤ 1 µA).

Subject to changes