MODELING AND REAL-TIME CONTROL SYSTEM IMPLEMENTATION FOR A STEWART PLATFORM

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ONUR ALBAYRAK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
MECHANICAL ENGINEERING

NOVEMBER 2005

Approval of the Graduate School of Natural and Applied Sciences

Prof. Dr. Canan ÖZGEN
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Kemal İDER
Head of the Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Prof. Dr. M. A. Sahir ARIKAN
Supervisor

Prof. Dr. Tuna BALKAN
Co-Supervisor

Examining Committee Members

Asst. Prof. Dr. İlhan KONUKSEVEN (METU,ME)

Prof. Dr. M. A. Sahir ARIKAN (METU,ME)

Prof. Dr. Tuna BALKAN (METU,ME)

Asst. Prof. Dr. Buğra KOKU (METU,ME)

Burak GÜRCAN; Mech. Eng, MSc. (ASELSAN)

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Onur Albayrak

ABSTRACT

MODELING AND REAL-TIME CONTROL SYSTEM IMPLEMENTATION FOR A STEWART PLATFORM

Albayrak, Onur
M.Sc., Department of Mechanical Engineering
Supervisor: Prof. Dr. M. A. Sahir ARIKAN
Co-Supervisor: Prof. Dr. Tuna BALKAN

November 2005, 105 pages

This work focuses on modeling and real-time control of a motion simulator for dynamic testing of a two-axis gyro-stabilized head mirror used in modern tanks. For this purpose, a six-degree-of freedom Stewart Platform which can simulate disturbances on the stabilized head mirror during operation of the tank is employed.

Mathematical models of the Stewart Platform are constructed using MATLAB® and ADAMS®. Control system infrastructure is constructed and real-time control system elements are employed. Controller tuning is achieved by using the developed mathematical models in MATLAB®. These parameters are applied in the real-time control system and fine tuning is achieved. Accuracy of the motion simulator is tested by mounting an Inertial Measurement Unit on the Stewart Platform. Further control system strategies are discussed by means of simulation.

Keywords: Stewart Platform, Modeling, Controller Tuning, Real-Time Control, MATLAB®

ÖZ

BİR STEWART PLATFORMU'NUN MODELLENMESİ VE GERÇEK ZAMANLI KONTROLÜ

Albayrak, Onur
Yüksek Lisans, Makina Mühendisligi Bölümü
Tez Yöneticisi: Prof. Dr. M. A. Sahir ARIKAN
Ortak Tez Yöneticisi: Prof. Dr. Tuna BALKAN

Kasım 2005, 105 sayfa

Bu çalışmada, modern tanklarda kullanılan iki eksende stabilize alın aynasının performans testlerinin yapılabilmesi için tasarlanan hareket simülatörünün modellenmesi ve kontrolü gerçekleştirilmiştir. Bu amaçla altı serbestlik derecesine sahip, tankın işletimi sırasında alın aynasının maruz kalacağı bozucu etkilerin benzetimini sağlamak amacıyla bir Stewart Platformu geliştirilmiştir.

Stewart Platformu'nun matematiksel modelleri MATLAB® ve ADAMS® yazılımları kullanılarak oluşturulmuştur. Denetim sistemi altyapısı kurulmuş, gerçek zamanlı denetim sistemi elemanları kullanıma alınmıştır. Denetim sistemi ölçütleri MATLAB®'da geliştirilen modeller kullanılarak bulunmuştur. Bu ölçütler gerçek sistem üzerinde denenmiş ve nihai haline getirilmiştir. Hareket benzetim platformunun hareketinin doğruluğu bir Ataletsel Ölçüm Cihazı kullanılarak test edilmiştir. Simulasyonda farklı kontrol teknikleri de irdelenmiştir.

Anahtar Kelimeler: Stewart Platform, Modelleme, Denetim Birimi Ölçütlerinin Ayarlanması, Gerçek Zamanlı Denetim, MATLAB®

To My Lovely Family

ACKNOWLEDGMENTS

I would like to express my gratitude and appreciation to my thesis supervisor Prof. Dr. M. A. Sahir Arıkan and co-supervisor Prof. Dr. Tuna Balkan for their continuous support and supervision throughout the completion of this work.

I would like to thank to my colleagues in Aselsan Inc and my friend Deniz Yücel for their endless support and valuable comments all through this hard work.

Finally, I would like to thank my family for their love and support during preparation of this thesis.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS

l	Number of links
j	Number of joints
f_i	Degree-of-freedom of the ith joint
λ	Degree-of-freedom of space
F	Degree-of-freedom of the mechanism
a_i	ith point on the base
b_i	ith point on the moving platform
F_a	Fixed coordinate system
F_b	Body coordinate system attached to the moving platform
$\vec{u}_1^{(a)}$, $\vec{u}_2^{(a)}$, $\vec{u}_3^{(a)}$	Unit basis vectors of F_a
$\vec{u}_1^{(b)}$, $\vec{u}_2^{(b)}$, $\vec{u}_3^{(b)}$	Unit basis vectors of F_b
α	Offset angle from 120° spacing at the base
β	Offset angle from 120° spacing at the upper platform
$\overline{p}$	Position vector of F_b with referenece to F_a
${}^{b}R_a$	Rotation matrix mapping F_b to F_a
l_i	Length of the ith leg
l_i^{ini}	Length of the ith leg in initial configuration
l_i^{ch}	Length change of the ith leg
$\overline{q}$	Generalized coordinate vector
ψ, θ, φ	Euler angles of the moving platform
K_p	Proportional gain
T_i	Integral time constant
v	Translational velocity of the leg

m	Equivalent mass of the leg
x	Translational distance of the leg
F_m	Actuator force
F_d	Disturbance force to actuator
$F_m^{\max}$	Maximum control input
F_m^{noise}	System noise in control input
F_d^{noise}	System noise in disturbance force
x^{noise}	Measurement noise
q_{00}	Control force covariance value
q_{11}	Disturbance force covariance value
r_{00}	Measurement noise covariance value
$P_{k,l}$	Estimated variance matrix
$Var(x)$	Variance of the random variable x
$E(x)$	Expectation of the random variable x
G_k	Kalman gain matrix
Q_k	Variance matrix of the random vector ξ_k
R_k	Variance matrix of the random vector η_k
u_{k-1}	Control input in the previous sampling period

CHAPTER 1

INTRODUCTION

1.1 Overview

One of the most dominant instincts of human beings, the desire to dominate, has resulted in many contentions among people, races, fractions and eventually countries throughout the history of mankind. Starting from the primitive utilization of muscular power, the means to battle have come up to a point that includes the deployment of many complex war machines. The requirements for the advancing of the technology in this area have created a source of motivation for science to come up with skills to overcome the opponent.

In the last century that has faced two great world wars; the armored land forces have proved their significance in combat. The mobility supported with a firing power they provide on almost every type of terrain, has constituted the difference among battling armies. From the engineering point of view, these machines are nothing but dynamic systems that require further attention on several issues.

The comprehensive process of developing a dynamic system includes several design, implementation and testing stages. However a time period of diligence and fastidious labor is spent on design and implementation, the real outcome of the work can be assessed only after a successful testing procedure. Of the many testing procedures in dynamic system development studies, performance testing of the prototype is a critical issue. The environment in which the dynamic system is used can impose disturbances to the dynamic system and these disturbances should be considered in the early development stages of the system to achieve a design that takes the actual behavior of the system into account. If the dynamic system is going to be used in a

moving vehicle, then the motion of the vehicle should be simulated in order to see the effects on the dynamic system.

As mentioned before, armored land forces are good examples to dynamic systems. Tanks, which are extensively utilized for many types of ground missions, constitute the largest portion of these forces. Today, modern tanks are equipped with thermal and day TV cameras to provide the gunner with perfect vision capabilities in every night and day conditions. However, the movement of the tank on rough terrain disturbs the provided vision as the motion of the tank results in relative motion of these devices with respect to the ground. Head mirrors, with their ability to partially even out these relative motions, have been used in military applications to provide the gunner with a stable vision to attain advantage on battlefield.

The motion of the tank disturbs the head mirror by means of rotational velocities and translational accelerations. To define these velocities and accelerations, a motion terminology is used that comprises 3 linear and 3 rotational axes as shown in Figure 1.1. The stabilization of the head mirror is termed as the compensation of these disturbing effects on the predefined axes. Performance tests are used to evaluate the accuracy of the stabilization process.

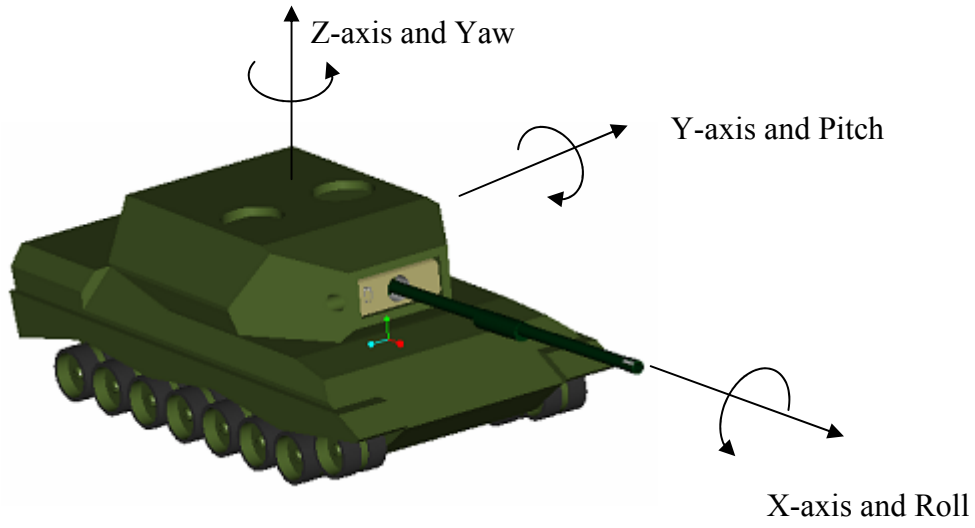


Figure 1.1: Tank motion terminology

Performance tests are usually done on a special track called APG (Aberdeen Proving Ground) on which different barriers with different heights are placed at predetermined positions. Distance between barriers, height of the barriers and velocity of the tank on APG track are determined according to the military standards. On these tests, the rotational velocities and translational accelerations on each axis due to the motion of the tank is recorded. The performance of the system is mostly affected by the motion of the tank in pitch axis. A typical pitch motion data of the tank on APG track collected by sensors is shown in Figure 1.2.

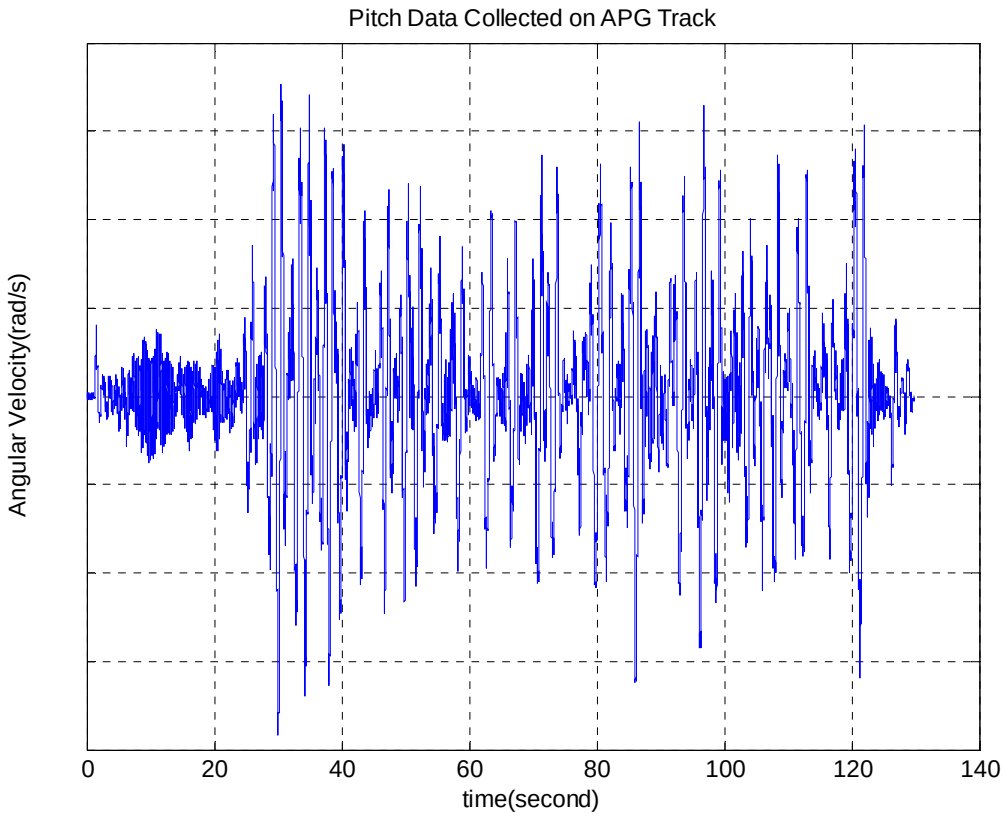


Figure 1.2: An APG track data in pitch axis collected on the tank turret

Aselsan Inc., developing a stabilized head mirror, requires performance tests of the system. These performance tests should be originated from a common tank motion that can be comparable with other related studies. Tests on APG track provide the required data source for rotational velocities and translational accelerations, which

can be used to define a tank motion. Tank motion on APG track in all six axes is recorded once by sensors and by simulating this motion in laboratory environment; the troublesome and expensive replications of the tests on APG track can be avoided. There are a number of different ways to carry the realization of tank motion into laboratory environment. The common point of all those realizations is to come up with a motion simulator that performs dynamic movements of the tank to observe the effects of these motions on the system. Of the many available solutions to this problem that enable the rapid development and verification of systems, Stewart Platforms are unique with their enabling of motion simulation in all six axes.

1.2 Stewart Platform

Stewart Platform was originally designed in 1965 as a flight simulator, and it is still commonly used for that purpose. A wide variety of applications have benefited from this design since then. Stewart Platform has been used in many industries including automotive, defense, transportation and machine tool technology. Motion simulators and machining tools are the most common ways of employing a Stewart Platform. A typical motion simulator is shown in Figure 1.3 [1]; and an example of a machine tool based on the Stewart Platform is shown in Figure 1.4 [2].

Figure 1.3: A typical Stewart Platform as motion simulator

Figure 1.4: Stewart Platform as a machine tool

Stewart-Platforms can also be used to replace conventional crane technology. A crane that utilizes a Stewart Platform provides the crane operator with greater control of the crane hoist mechanism. The National Institute of Standards and Technology has developed a crane, known as ROBOCRANE, utilizing Stewart-Platform technology [15]. Additionally, Stewart-Platforms can replace the conventional single cable hoisting technology currently being used on helicopters for use as an air crane or in air-to-sea rescue.

Stewart Platforms are parallel manipulator based systems which provide six degrees-of-freedom. They provide high rigidity for a given structural mass enabling the Stewart Platform system with high positional accuracy. They can handle relatively higher loads and provide higher speeds than serial manipulator based systems. On the other hand; Stewart Platforms have smaller workspace areas in comparison with serial manipulator systems.

Considering the effectiveness of a Stewart Platform system as a motion simulator; Aselsan tank simulator has been designed and constructed based on a Stewart Platform. The designed platform consists of a lower platform (base), upper platform,

and six linear actuators as shown in Figure 1.5. Between the base and the stationary part of the actuators, universal joints are used whereas gimbal joints are employed between the upper platform and the moving part of actuators. Stabilized head mirror and a thermal camera are mounted to the upper platform as test equipment. The main task here is to maintain the desired trajectory for the upper platform and therefore for the test equipment, by manipulating the lengths of the six linear actuators.

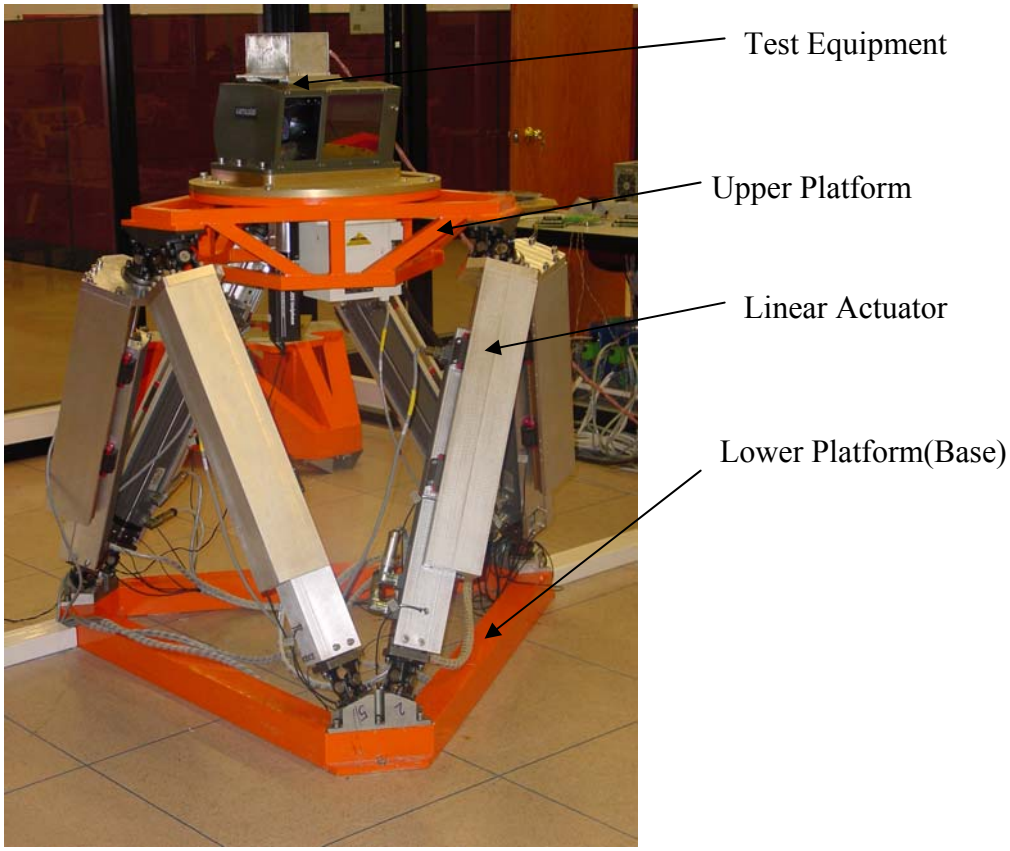


Figure 1.5: Aselsan Tank Simulator

1.3 Contents and Organization

The scope of the study comprises the mathematical modeling and control system design for the Stewart Platform as well as the real-time control of the system.

In Chapter 2, kinematics of the system including mechanism identification, kinematical identification and inverse kinematics of the platform are examined. MATLAB® modules and ADAMS® are used for modeling and simulation and outputs of these software are compared with each other.

Chapter 3 is dedicated to the discussion of dynamics of the system, including identification of dynamic parameters of the parts, forward and inverse dynamics analysis. MATLAB® and its modules and ADAMS® are used for simulations and obtained results are used for comparison.

In Chapter 4, control system strategies and the developed real-time control system are discussed. Real-time control system elements such as actuators, drivers, and control software are explained and technical data for these elements are given. The designed control system based on the kinematic and the dynamic models developed using MATLAB® and its modules is described in detail. The scope of the chapter extends to the stages of implementation of the models and verification of the generated platform motions. An additional control system strategy based on feedforward technique is the last subject of this chapter.

In Chapter 5, summary of the work done and conclusions about them are presented. Some additional work that can be done in future is also explained in this chapter.

Simulation software codes, specifications of the control system elements, real-time control system models and dynamic identification of the mechanical parts are presented in Appendices.

CHAPTER 2

KINEMATICS OF THE STEWART PLATFORM

2.1 Verification of the Number of Degrees-of-Freedom of the Stewart Platform

The developed Stewart Platform has six degrees-of-freedom and this should be verified by the general degree-of-freedom equation. The system can be represented by means of joints and links as shown in Figure 2.1. The lower platform is fixed to ground and is counted as a *link*; the upper platform, on which the test equipment is mounted, is also counted as a *link*. Each actuator is mounted to the lower platform by a universal joint and to the upper platform by a gimbal joint. Universal joints provide two rotational degrees-of-freedom whereas gimbal joints have three rotational degrees-of-freedom. Stationary and moving parts of the actuators are connected by prismatic joints. From the mechanisms point of view; for an actuator,

# of links $= l = 2$

# of joints $= j = 3$

# of joint freedom $= f_i = 5 \cdot R + P = 6$

For the whole system including six actuators, the base and the upper platform,

Total # of links $= 6 \cdot l + 2 = 14$

Total # of joints $= 6 \cdot j = 18$

Total # of joint freedom $= 6 \cdot f_i = 36$

General degree of freedom equation can be shown as [3],

$$F = \lambda.(l - j - 1) + \sum f_i \qquad (2.1)$$

in which $\lambda = 6$ for spatial space.

Substituting the parameters into the equation, it is verified that the platform has six degrees-of-freedom.

$$F = 6.(14 - 18 - 1) + 36 = 6$$

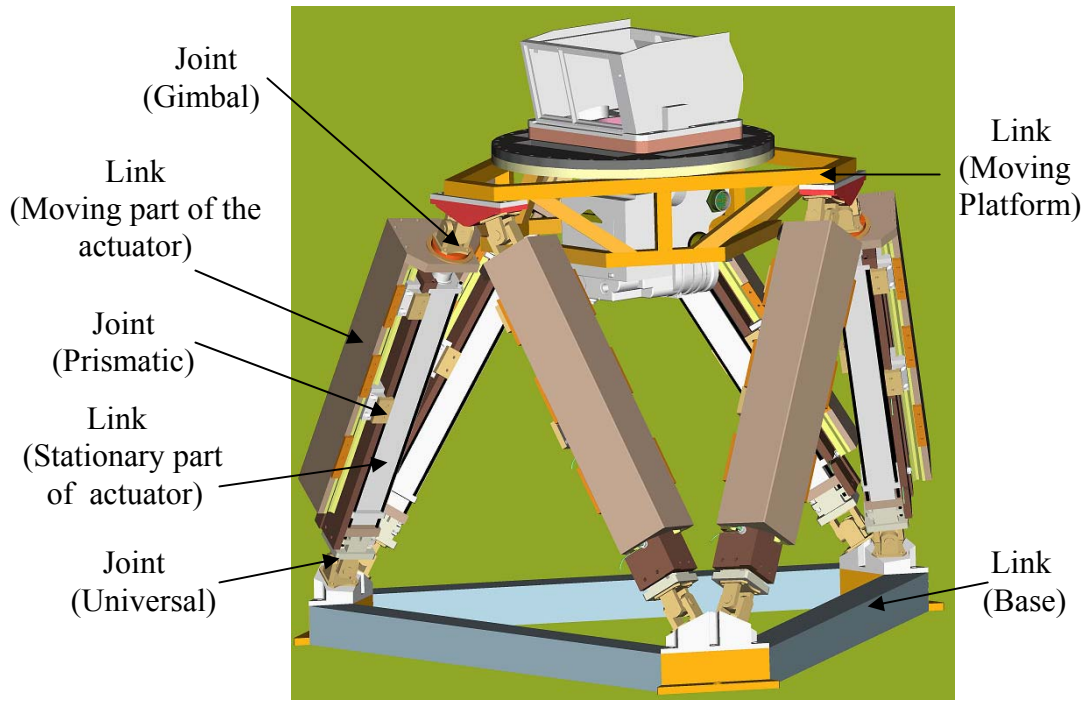


Figure 2.1: Mechanism identification of the Stewart Platform

2.2 Kinematic Identification of the Stewart Platform

Kinematic parameters from the 3-D CAD model should be obtained in order to create kinematic and dynamic models of the Stewart Platform. The sketch representing the kinematic parameters of the system is shown in Figure 2.2.

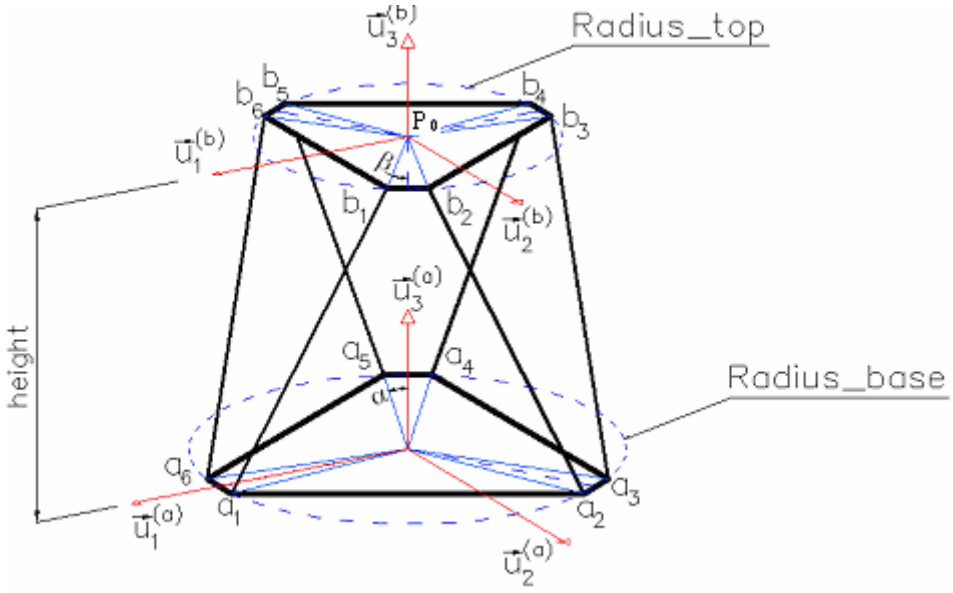


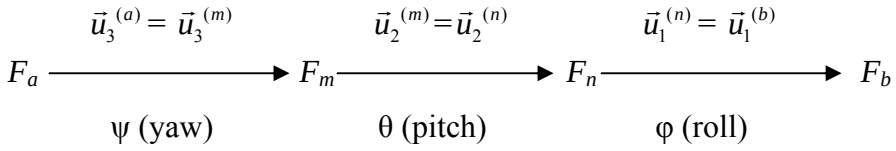
Figure 2.2: Kinematical identification of the Stewart Platform

As shown in this figure; the Stewart Platform consists of six stationary points at the base designated by $a_{1..6}$ and six variable points at the upper platform designated by $b_{1..6}$. These points are located such that 120° spacing occurs both on the upper platform and on the base. "α" is the offset angle from 120° spacing at the base whereas "β" is the offset angle from 120° spacing at the upper platform. Points on the upper platform are 60° offset from the points at the base. Unit base vectors $\vec{u}_1^{(a)}$, $\vec{u}_2^{(a)}$ and $\vec{u}_3^{(a)}$ form a fixed coordinate system F_a while unit base vectors $\vec{u}_1^{(b)}$, $\vec{u}_2^{(b)}$ and $\vec{u}_3^{(b)}$ form a body coordinate system F_b. The distances of the points at the base from the origin of F_a are "Radius_base" while the distances of the points at the upper platform from the origin of F_b are "Radius_top". In initial configuration, the distance between the fixed and the body coordinate system is "height" in the direction of $\vec{u}_3^{(a)}$.

The Stewart Platform has six degrees-of-freedom; therefore six parameters are necessary to describe its position and orientation. Three of these parameters represent the translational displacements, describing the position of a reference point

on the moving platform with reference to F_a. The other three parameters are angular displacements that describe the orientation of F_b with reference to F_a.

The position of the coordinate system F_b is specified with reference to F_a by a vector $\bar{p}$ =(X, Y, Z) ^{T}, which gives the coordinates of the reference point "P_o" with reference to F_a. Orientation of F_b with reference to F_a is described by 3-2-1 (yaw-pitch-roll) sequence of Euler angles. F_m and F_n are the intermediate coordinate systems in mapping of these coordinate systems.



Mapping between F_b and F_a is achieved through a rotation matrix based on 3-2-1 sequence of Euler angles. Rotation matrix representing the mapping between these coordinate systems is shown in equation (2.2) [4].

$$ {}^{b}R_a = \begin{pmatrix} c\psi.c\theta & s\psi.c\theta & -s\theta \\ -c\phi.s\psi + c\psi.s\theta.s\phi & c\psi.c\phi + s\psi.s\theta.s\phi & c\theta.s\phi \\ s\psi.s\phi + c\psi.s\theta.c\phi & -c\psi.s\phi + s\psi.s\theta.c\phi & c\theta.c\phi \end{pmatrix} \tag{2.2} $$

where c denotes cosine and s denotes sine.

Therefore, a generalized coordinate vector $\bar{q} = (\psi, \theta, \phi, X, Y, Z)^T$ can be defined describing the position and the orientation of the moving platform with reference to the fixed coordinate frame.

2.3 Inverse Kinematics

The study on inverse kinematics is concerned with the determination of the six leg lengths corresponding to the position and the orientation of the moving platform.

Referring to Figure 2.2, attachment points on the moving platform denoted by b_i are specified with reference to the body coordinate system as ${}^{b}\overline{b_i} = (x_{b_i}, y_{b_i}, z_{b_i})^T$, and these attachment points can be obtained with reference to the fixed coordinate system F_a by using the equation

$$\overline{b_i} = \overline{p} + {}^{b}R_a \, {}^{b}\overline{b_i} \qquad [14] \tag{2.3}$$

Once the position of attachment b_i is determined with reference to the fixed coordinate system, the leg vector can be obtained as

$$\overline{L_i} = \overline{b_i} - \overline{a_i} \tag{2.4}$$

where $\overline{a_i}$ represents the coordinates of base points with reference to the fixed coordinate system [14]. The lengths of the legs can be found as

$$l_i = \sqrt{\overline{L_i} \cdot \overline{L_i}} \tag{2.5}$$

The leg lengths in initial configuration of the platform can be found by using the equations (2.1) to (2.5) and denoted as l_i^{ini}. Required changes for leg lengths can be obtained as

$$l_i^{ch} = l_i - l_i^{ini} \tag{2.6}$$

2.3.1 Inverse Kinematics Solution in MATLAB®

Inverse kinematics solution is implemented in MATLAB® by using the equations from (2.2) to (2.6). In order to determine the leg displacements, a code is generated

for inverse kinematics, using three Euler angles and moving body translational displacements as inputs. This code in the format of m-file is given in Appendix A.

The model file for inverse kinematics operations is shown in Figure 2.3. As shown in figure below, desired platform motion is the input to "inverse_kinematics" function while actuator displacements are the outputs of the function. At the start of the simulation, simulation parameters are selected. A fixed-step solver ode5 (Dormand-Prince) is selected with a step size of 1/1000 seconds. 1 kHz sample interval is small enough to deal with the tank motion. All simulations in MATLAB® environment are done with this solver throughout the study.

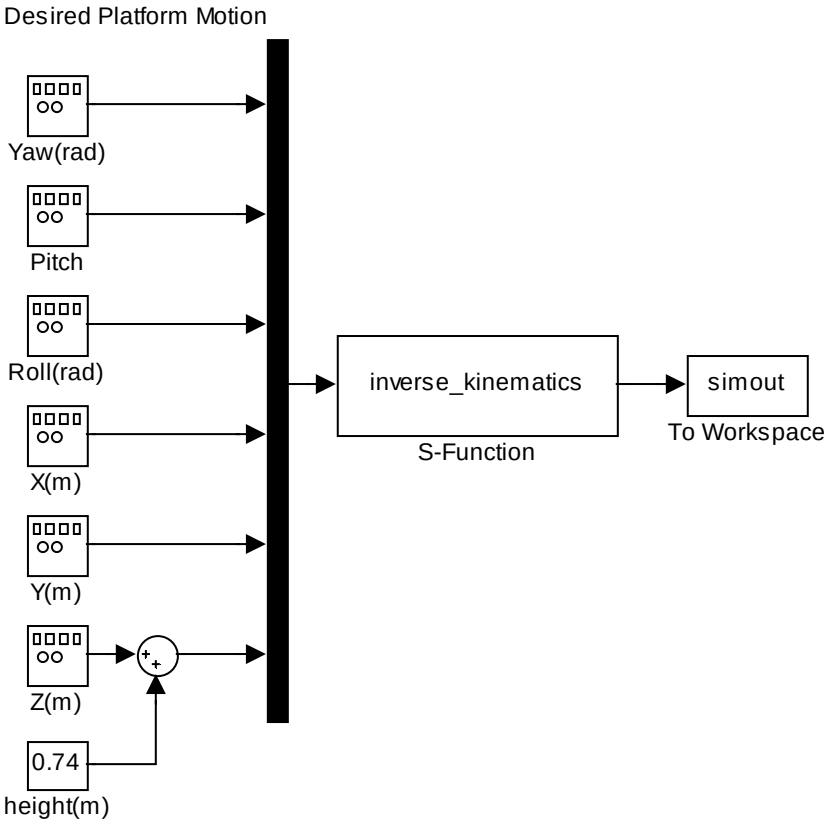


Figure 2.3: Simulink® inverse kinematics model

Tank motion in pitch axis is given as the desired platform motion and the desired actuator displacements are found by inverse kinematics.

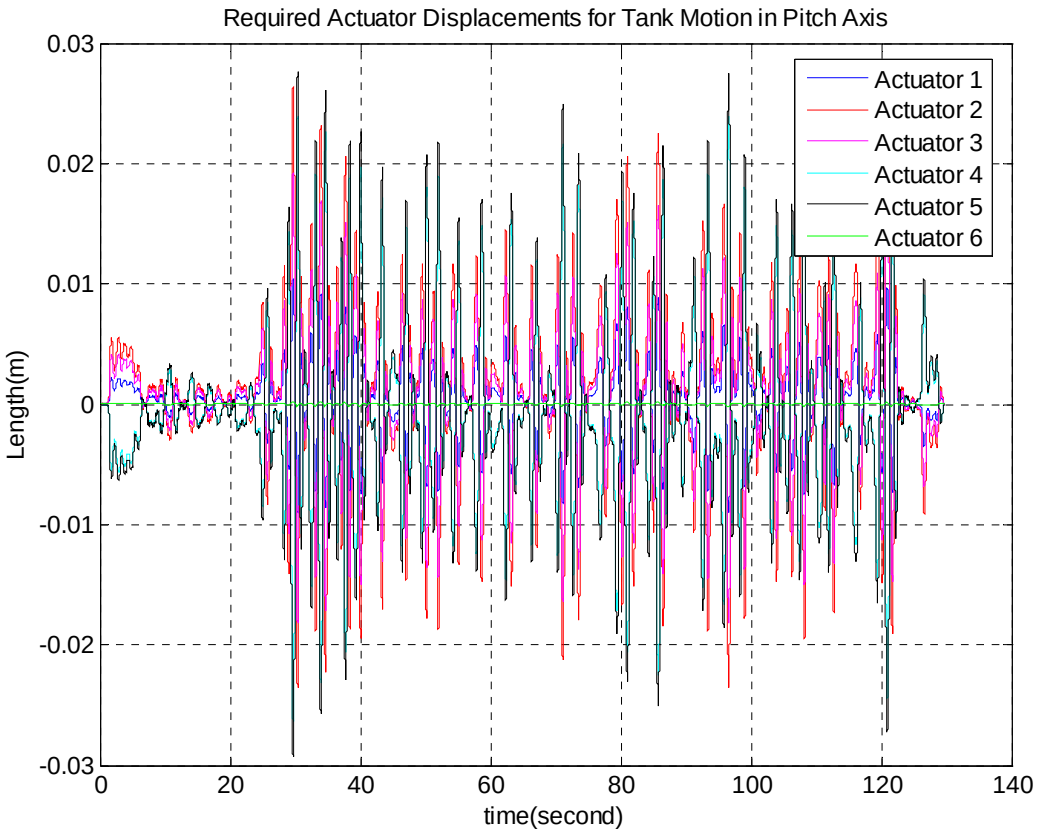


Figure 2.4: Simulink® inverse kinematics results for tank motion in pitch axis

2.3.2 Verification of MATLAB® Inverse Kinematics Solution by ADAMS®

3-D CAD model of the Stewart Platform is transformed from Pro/Engineer® to ADAMS® for kinematic analysis. ADAMS® minimizes kinematical modeling errors since the software directly uses 3-D model of the system. It also enables the visual observation of the motion of the platform. Another advantage of using ADAMS® is the opportunity it provides for the verification of the kinematic model and inverse kinematics solution [5].

3-D model of the Stewart Platform is simplified by omitting actuator profiles, test equipment, screws and nuts etc. in order to simplify the analysis. The simplified ADAMS® model is shown in Figure 2.5. All joints including universal, gimbal,

prismatic and fixed joints are created in this model and body coordinate systems are defined. A motion is created between moving platform and ground (referring to fixed coordinate system) and actuator displacements are measured in prismatic joint coordinates.

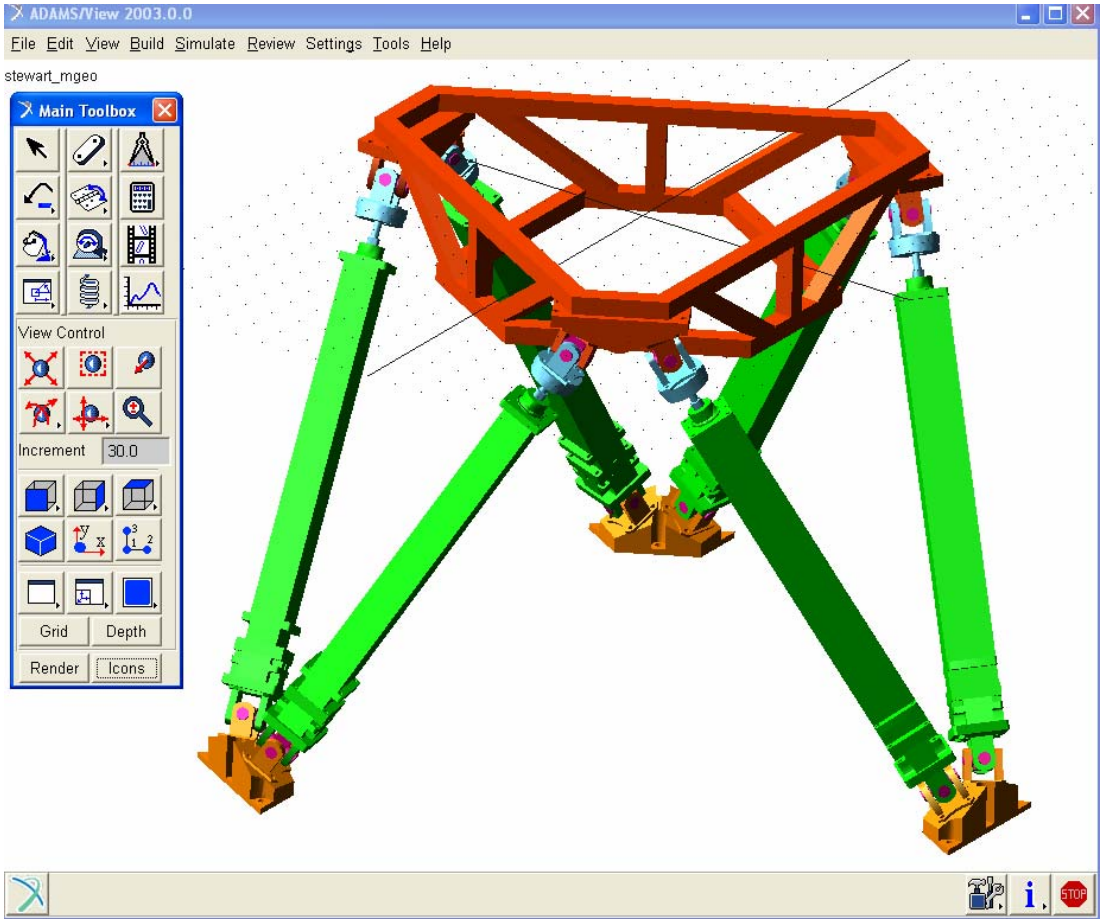


Figure 2.5: ADAMS® inverse kinematics model

All simulations in ADAMS® are done with a fixed step solver with 1/1000 as the step size. Specifically, a motion of $\bar{q} = (0, 3\,pi/180 \sin(2\pi t) rad, 0, 0, 0, 0.74m)^T$ that represents a motion in pitch axis is supplied to the moving platform and the required leg displacements are measured for this motion. In order to compare the results of ADAMS® with the solutions obtained from MATLAB®, the same motion is also

provided to the model developed in Simulink®. The obtained results from MATLAB® are exported to ADAMS®. Leg displacements obtained from both ADAMS® and MATLAB® are shown in Figure 2.6.

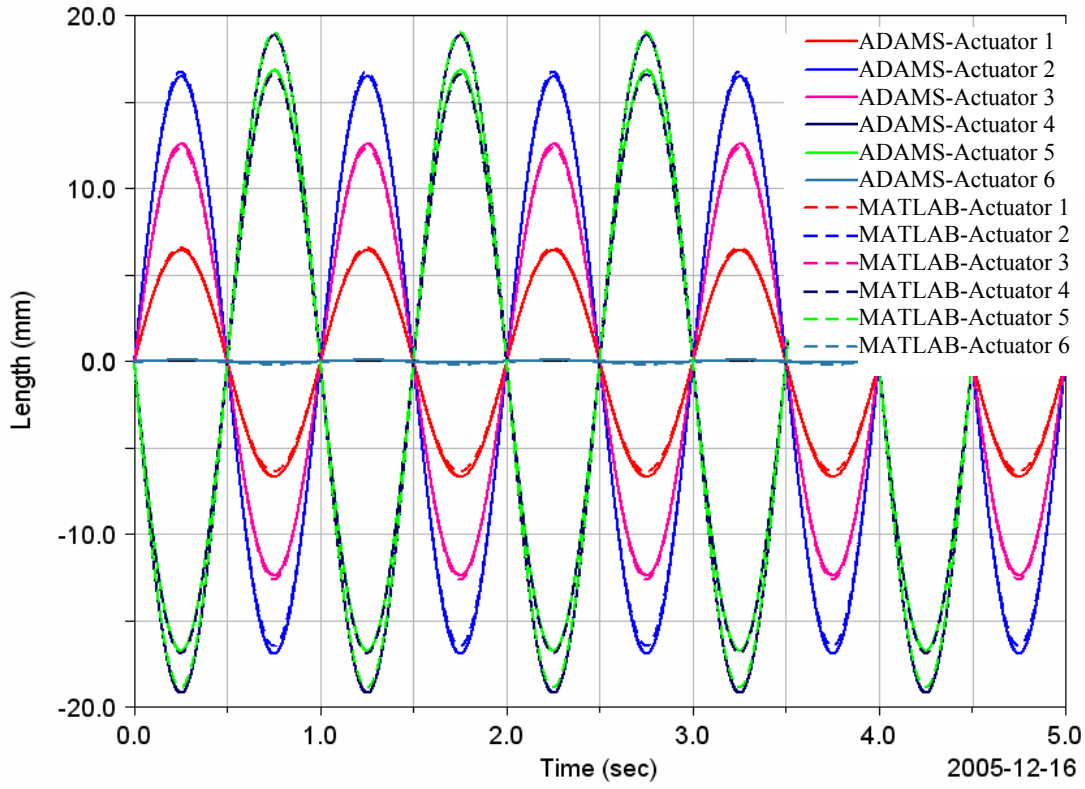


Figure 2.6: ADAMS® and MATLAB® inverse kinematics results for the motion $\bar{q} = (0, 3\,pi/180 \sin(2\pi t) rad, 0, 0, 0, 0.74m)^T$

In this figure, it is observed that inverse kinematics results of the models in ADAMS® and MATLAB® are consistent. Therefore, geometric identification of the Stewart Platform and inverse kinematics solution in MATLAB® is verified by ADAMS®.

CHAPTER 3

DYNAMICS OF THE STEWART PLATFORM

In order to have control on the Stewart Platform, a satisfactory dynamic model combined with the kinematic model should be obtained. This model enables the user understand the characteristics of the system and thus have an intuition about the control system.

Dynamics of a Stewart Platform is more complex than dynamics for a serial manipulator type system and has not been fully developed. Developed models for Stewart Platforms have used simplifying assumptions. Stewart Platforms show highly non-linear characteristics so the linear models developed for them do not show the characteristics of the original system, accurately.

In this work, dynamic modeling of the Stewart Platform is done by two commercial software; MATLAB® using its module SimMechanics® and ADAMS®. Modeling with SimMechanics® does not require three dimensional drawing of the system, but requires kinematic and dynamic parameters. In SimMechanics®, it is easier to model the system and simulations can be done faster. In ADAMS®, 3-D CAD model of the system is required; simulations are done rather slowly but visual support is provided.

Two kinds of dynamic analysis, forward dynamics and inverse dynamics are done for the Stewart Platform. In forward dynamics; forces are applied as the input signal between stationary and moving parts of the actuators, while position and velocity of the actuators and platform motion are obtained as the output signals. In inverse dynamics, upper platform trajectory is provided as the input signal, and required actuator forces for this trajectory are obtained as the output signals.

In mathematical modeling of the Stewart Platform, stiffness of the parts and friction at the joints are ignored. Distributed mass model is used for the mechanical parts of the system. In this aspect, mass and inertia parameters and the centers of gravity of the moving links should be identified accurately, in order to have a satisfactory mathematical model.

3.1 Modeling of the Stewart Platform with SimMechanics®

The Stewart Platform can be barely identified as being composed of links and joints. The upper and the lower platforms, the moving and the stationary parts of the actuators are the links in the platform. These links are defined by their mass and inertia tensor values with references to the body coordinate systems at their centers of gravity. These body coordinate systems should be referenced to the inertial coordinate system. The centers of gravity of the links are also defined with reference to their body coordinate systems. The joints connecting the links are represented by defining their motion axes (translation and/or rotation). They can be actuated dynamically as force/torque as well as kinematically (position/velocity/acceleration). Sensors can be connected to the joints and to the bodies so that output signals of the motion can be observed. A controller can be created by Simulink®; sensor measurements can be fed to this controller; control signals can be generated and given to the actuators as the input signals [6].

Dynamic modeling is done parametrically. Kinematical parameters explained in Section 2.1 are directly used. Mass and inertia values of the related parts are found and used in modeling. By using the 3-D model in Pro/Engineer®, the mass and the inertia tensors, the centers of gravity of these links are identified. Detailed dynamic parameter identification of these links is explained in Appendix D. A code in the format of m-file defines all kinematic and dynamic parameters needed for SimMechanics® model, is shown in Appendix A.1. General view of the model is represented in Figure 3.1. The blocks need some related kinematical and dynamical parameters which are stored in this code.

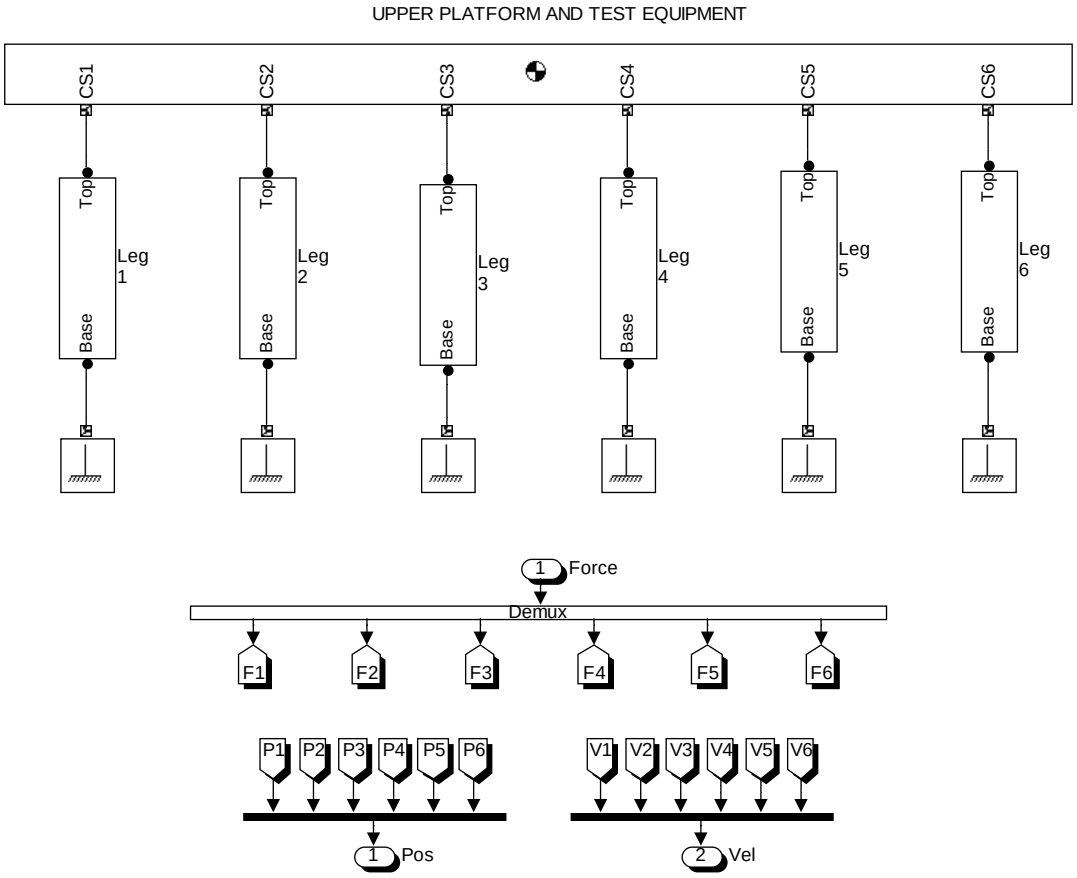


Figure 3.1: Model of the Stewart Platform in SimMechanics®

3.1.1 Forward Dynamics Model in SimMechanics®

This is the model called “forward dynamics” in which forces are the input signals to the system simulating actuator thrusts, while position and velocity of the actuators and the platform motion are the output signals. This model is used in control system implementation; since positions and velocities of the actuators will be feedback signals, while actuator thrusts will be the control signals. The model is shown in Figure 3.2.

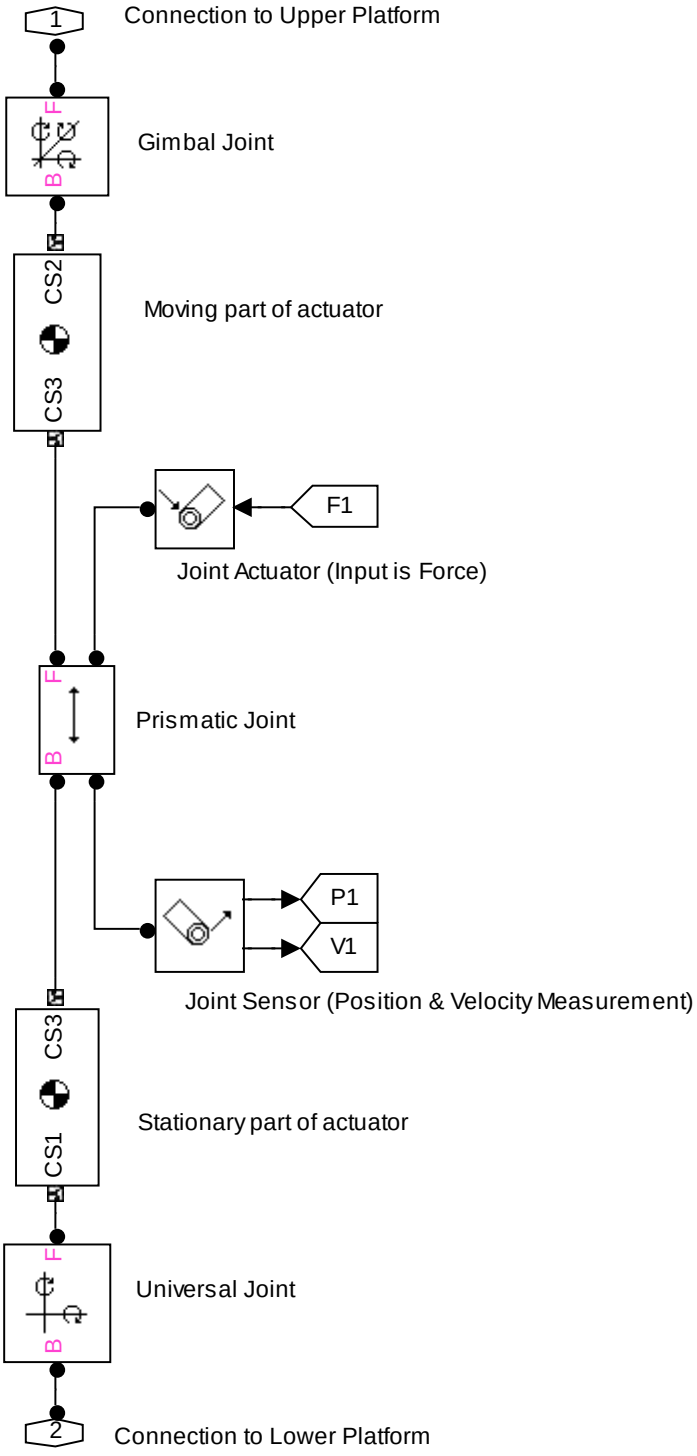


Figure 3.2: Model of one of the legs for forward dynamics model in SimMechanics®

3.1.2 Inverse Dynamics Model in SimMechanics®

Inverse dynamics model is somewhat different than the forward dynamics model because actuators are driven by position, velocity and acceleration signals rather than a force signal. Required actuator thrusts can be found for a given upper platform trajectory by combining the inverse dynamics model with the inverse kinematics model. General view of this model is shown in Figure 3.3.

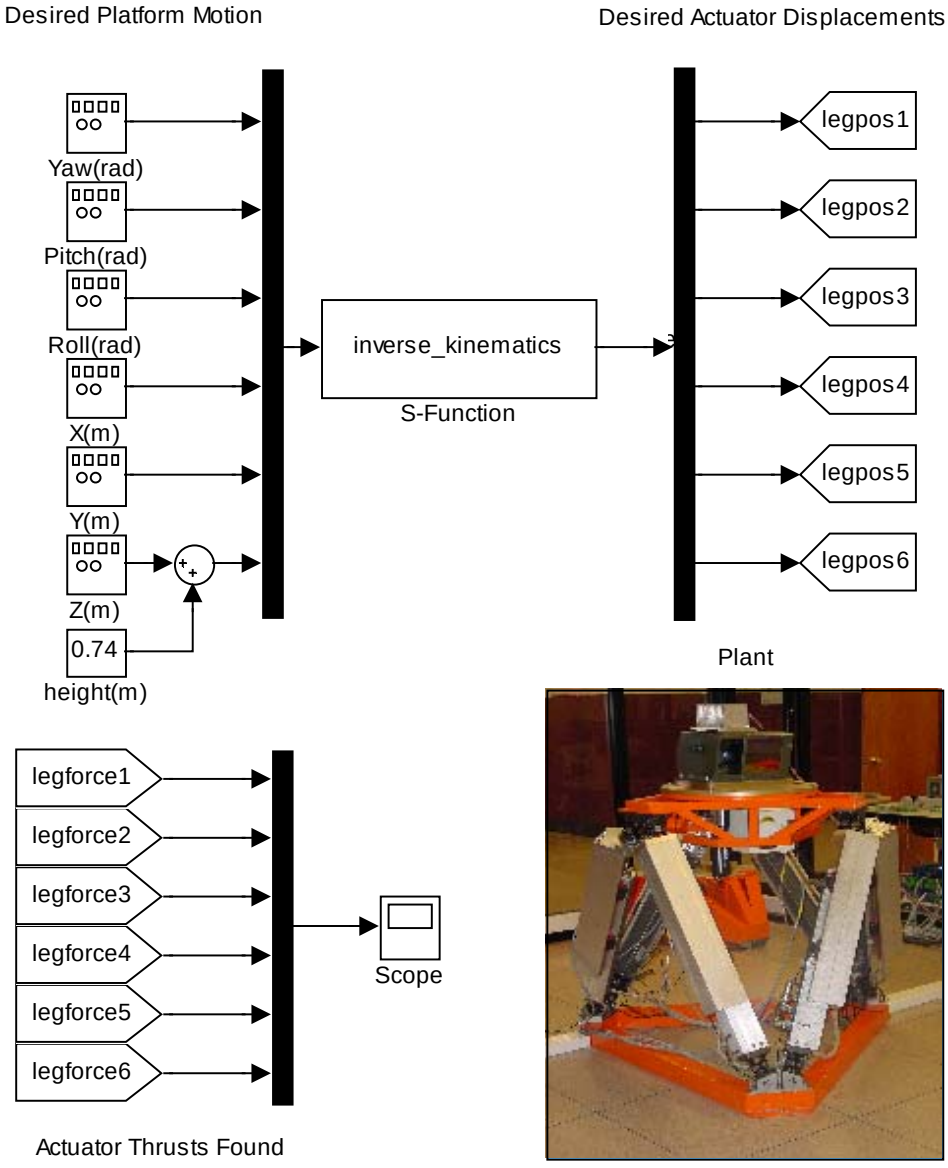


Figure 3.3: Inverse dynamics model in Simulink®/ SimMechanics®

The plant general view is the same as in Figure 3.1, however the difference between forward dynamics and inverse dynamics models can be observed in Figure 3.5. “Joint Actuator” block is utilized in order to get actuator motion as the input signal; while “Joint Sensor” block is used in the measurement of the required force signal.

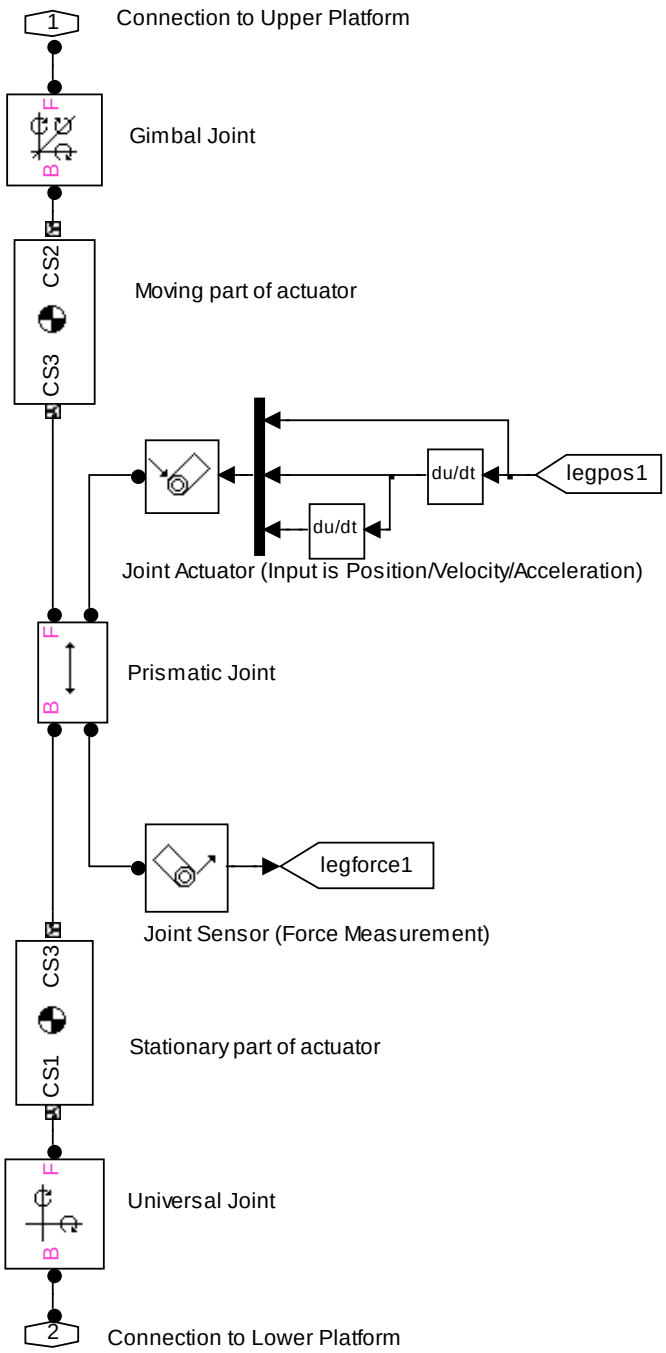


Figure 3.4: Model of one of the legs for inverse dynamics in SimMechanics®

Simulations for the inverse dynamics are done for the actual configuration of the system. Required actuator thrusts are represented as the output signals for the input signal of tank motion in pitch axis in Figure 3.5.

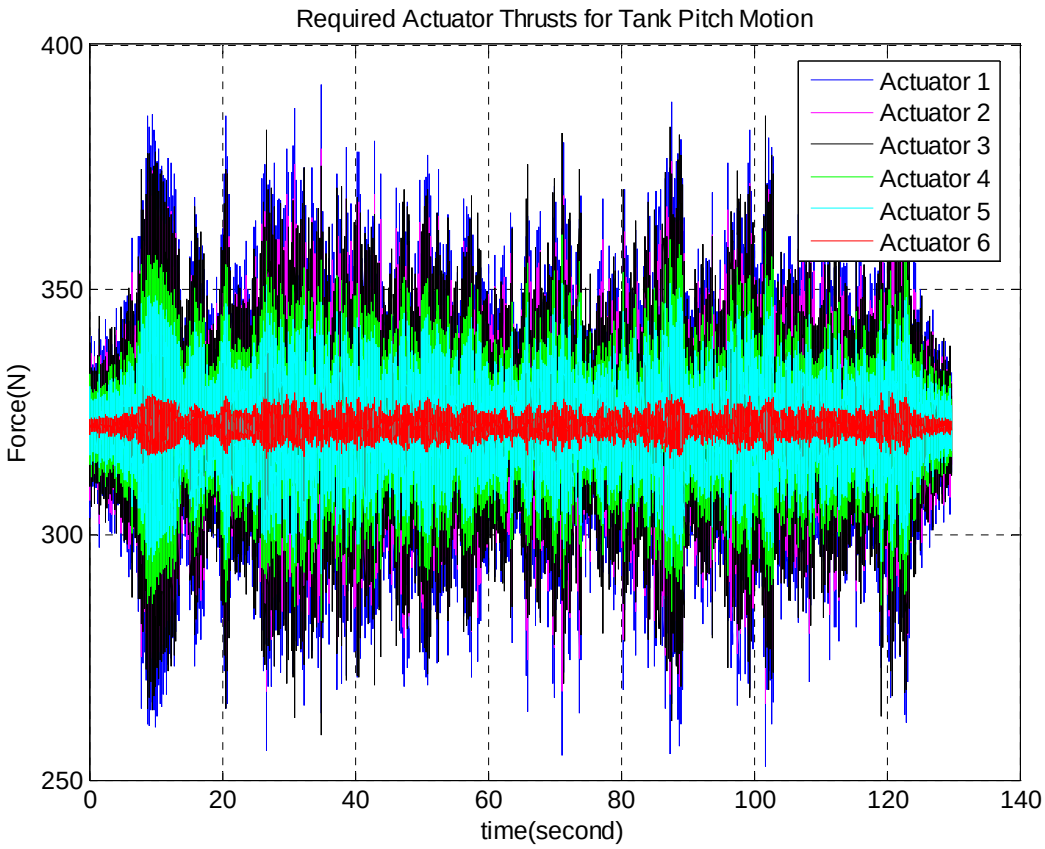


Figure 3.5: Required actuator thrusts for the tank motion in pitch axis

3.2 Verification of the Dynamic Model in SimMechanics® by ADAMS®

In order to implement inverse dynamics solution, inverse kinematics results for a defined moving platform trajectory are used in ADAMS®. These results representing actuator displacements are given as the input signals to the prismatic joints and required actuator forces are found with reference to the body coordinate systems of the actuators.

Mass and inertia tensor values for the mechanical parts can be found automatically by specifying density of them. However this model depends on a simplified 3-D CAD model of the Stewart Platform; therefore mass and inertia tensor values are specified with reference to the body coordinate system manually by using the mass and inertia tensor values of the links represented in Appendix D.

Required actuator thrusts for both ADAMS® and MATLAB® are shown for a motion of $\bar{q} = (0, 3\pi/180 \sin(2\pi t) rad, 0, 0, 0, 0.74m)^T$ which represents a sinusoidal motion in pitch axis in Figure 3.6.

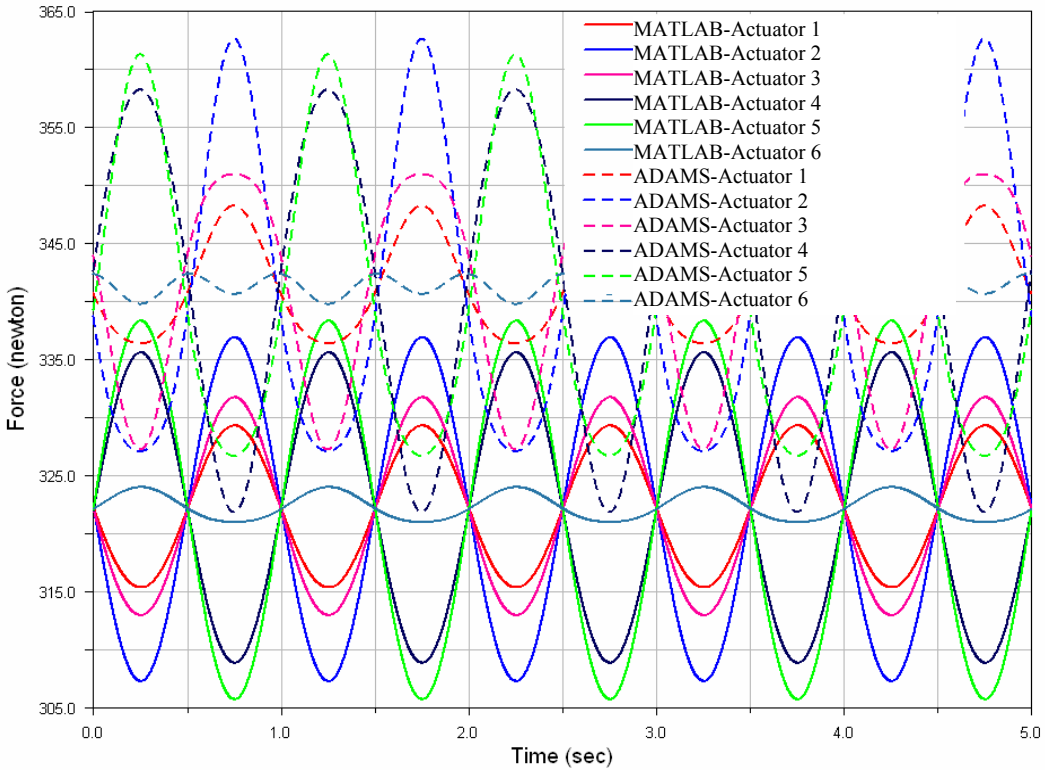


Figure 3.6: Required actuator thrusts for $\bar{q} = (0, 3\pi/180 \sin(2\pi t) rad, 0, 0, 0, 0.74m)^T$ motion input in pitch axis for both MATLAB® and ADAMS®

Comparing the results of ADAMS® with the ones in SimMechanics®, it's observed that there are differences in the results. The major difference is an offset force value

at start-up of the simulation. These differences may arise from the usage of a simplified 3-D CAD model of the Stewart Platform in ADAMS®. In simplification, stiffeners, rails etc. which are making the upper and the lower legs unsymmetrical about the linear motion axes are removed but their mass and inertia values are contributed to upper and lower legs. Although mass and inertia values of the links are reflected, centers of gravity of the upper and the lower legs differs from the one in ADAMS; because in ADAMS model, linear actuators are symmetric about their linear motion axis. Consequently, drawback of model simplification can cause these differences as well as discrepancies in the solver types of these software.

CHAPTER 4

CONTROL OF THE STEWART PLATFORM

The aim of the control system is to provide desired upper platform motion in six axes. Desired platform motion is achieved by driving the six linear actuators. The main task is to find the required actuator displacements for a defined platform motion trajectory and to provide the six actuator displacements in their own motion axes independently and in a closed-loop manner.

First of all, real-time control system elements are explained since controller tuning is attained concerning the real-time system. After introducing real-time control system elements, maximum payload that the Stewart Platform can perform is investigated. Then, the control strategy that is used in the control of the Stewart Platform is explained. Next step is the controller tuning based on the control strategy.

Primary controller tuning is critical, since it will be unsafe and harmful working directly by trial-and-error approach in real-time. For this reason, initial controller parameters are decided by using the models in Simulink®/SimMechanics®. Forward dynamics model created with SimMechanics® is used as the plant model and control actions are constructed in Simulink®; considering the limitations, constraints on the real system. Optimum control parameters found in simulations are used in the real-time control system and fine-tuning of the controller is done in real-time. After having a properly operating system, real-time response characteristics of the Stewart Platform both in time and frequency domain are identified and checked with the outputs of the simulated models in MATLAB® environment. Finally, a further control strategy based on feedforward control concept is discussed by means of simulation.

4.1 Real-Time Control System Configuration and Devices

Real-time control system of the Stewart Platform consists of several units such as computer hardware and software, data acquisition boards, drivers, electric motors, sensors etc. Major control setup is given in Figure 4.1.

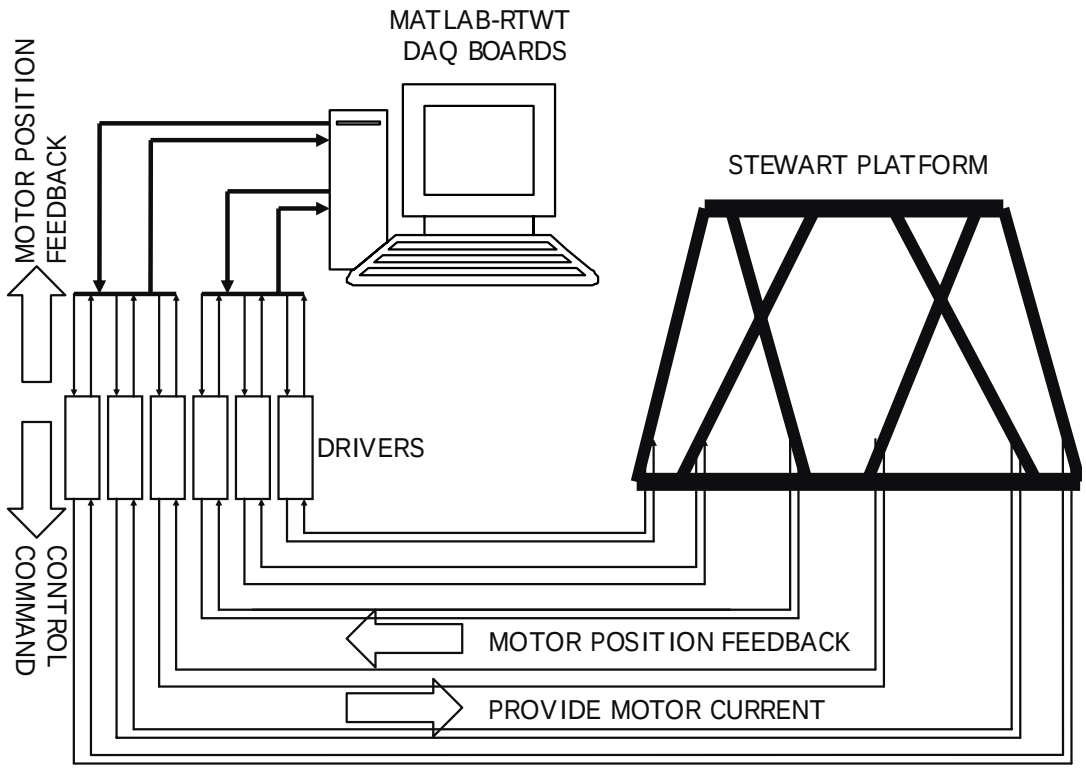


Figure 4.1: Real-time control system devices and configuration

In this figure, data transmission lines are shown schematically between hardware components. In computer, MATLAB®/Simulink® and its specific modules are used for real-time control computation, data acquisition and data sending purposes. Two data acquisition boards which are compatible with MATLAB® are used in transmission of the control system input and output signals between MATLAB® and

the drivers. The six drivers for the six linear actuators not only drive the electrical motors but also deliver motor position data to the MATLAB®.

4.1.1 Actuators

The linear actuators consist of an AC-brushless type electrical motor, a special rotary to linear motion conversion mechanism, a resolver for motor commutation purposes, and an aluminum housing. Three dimensional sectional view of the actuator is given in Figure 4.2.

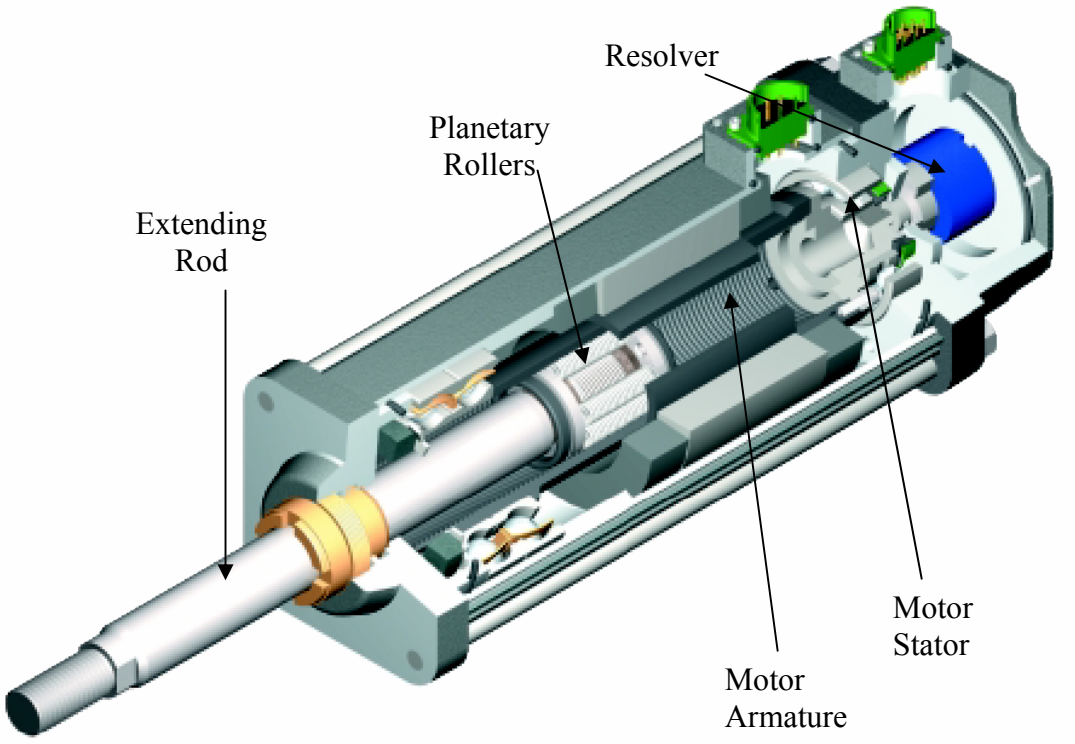


Figure 4.2: Detailed view of the linear actuator

Electric power is converted to linear motion by an electric motor and a roller-screw mechanism. Planetary rollers assembled around the actuator's extending rod follows threads on the inside surface of actuator's hollow armature [7].

Figure 4.3: Linear actuator used in the tank simulator

Driving AC-brushless motors are more complex than DC motors. They are operating with a default feedback device for commutation purposes and require digital circuitry for operation. For this specific motor, a resolver measuring the position of the electric motor is used for commutation purposes and it is processed in the driver.

Linear actuator's specifications and performance characteristics of them are shown in Appendix B.1.

4.1.2 Drivers

Electric motors are driven by servo drivers. Drivers process resolver data and use them for commutation purposes. They drive the electric motors by digital circuitries. Drivers also convert analog resolver data into digital incremental encoder data by resolver-to-digital-converter (RDC) modules. These RDC modules provide 12-bit position data as the output signal from the drivers. RDC modules also create velocity information from the resolver data and use them in feedback control loops. Drivers have current controller and speed controller options. Controller parameters of the drivers are selectable by a Windows-based parameterization program WMEMOC®. Drivers permit for both torque control and speed control as in proportional + integral

(PI) controller format. They have analogue voltage inputs corresponding to desired torque in torque control mode; or desired speed in speed control mode. In the speed control mode, a speed controller and a current controller operate simultaneously in a cascaded manner. In current control mode, only the current controller operates [8].

Parameterization program WMEMOC® enables the user to configure all the control parameters and displays the operation parameters to the user.

Figure 4.4: Motor driver used in the Stewart Platform

4.1.3 MATLAB® and Data Acquisition Hardware

Real-time control of the system is achieved by MATLAB® environment through data acquisition (DAQ) boards. MATLAB® toolbox Real-Time Windows Target® (RTWT) provides an interface for physical DAQ boards and enables hardware-in-the-loop simulation. DAQ boards are selected such that they are compatible with RTWT®. Measurement from the sensors, creating platform motion demand, inverse kinematics solution, generating and delivering control output signals to the drivers are done by MATLAB® in real-time.

Two DAQ boards each enabling 4 analog outputs and 4 encoder inputs are used for feedback measurement and control commanding [13]. Further information about the DAQ boards is given in Appendix B.4.

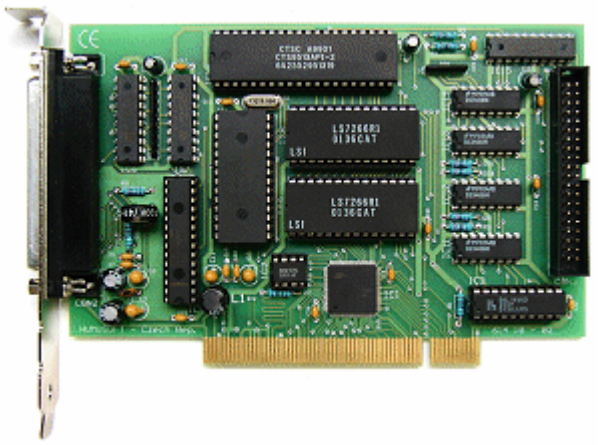


Figure 4.5: Data acquisition board used in the Stewart Platform

4.2 Maximum Payload Satisfying the Performance Criterion

As mentioned before, the Stewart Platform is designed and constructed in order to simulate the tank motion on the present payload, i.e., the head mirror and the thermal camera. However, the Stewart Platform can be used for other test equipments in order to simulate vehicle motion. The characteristics of the vehicle motion can differ according to the application; nevertheless the Stewart Platform can be used in simulation of many vehicles incorporated in land, naval or air systems.

The payload of the simulated system depends not only on the characteristics of the imposed motion but also on the dynamic load carrying capacities of the present actuators. Actuator thrust is limited by the root-mean-square (RMS) value of the motor current; which corresponds to 3073 N approximately. Therefore, a set of maximum acceleration values for a specific frequency range should be specified for the Stewart Platform; so that maximum payload for it can be found. If the

acceleration values are selected lower than the maximum accelerations, payload of the system that the Stewart Platform can handle increases. As the acceleration values become zero, i.e. when the Stewart Platform is stationary, maximum payload of the system is limited by the limits of the actuator thrusts. In this case, a static force analysis is executed in order to find the maximum payload criteria.

4.2.1 Maximum Payload that the Stewart Platform Can Handle Statically

The force analysis is carried out by using the inverse dynamics model in SimMechanics® in order to find the maximum payload that the Stewart Platform can handle statically. With an initial guess of mass and inertia tensor values of the payload, required actuator forces are found for many possible combinations of the six actuator positions in their upper and lower position limits. If no actuators goes beyond its actuator thrust limit, then mass and inertia tensor of the payload are increased. When any of the actuators exceeds its thrust limit, analysis is stopped and the payload in that iteration is recorded in order to have the maximum payload for the static case. The procedure is summarized by the flowchart shown in Figure 4.6, and the code in the format of m-file governing the analysis is shown in Appendix A.5.

In this analysis, mass and inertia tensor elements are increased by the same rate over the original payload, which means to assume that the geometry of the payload does not change. Mass increment unit is selected as 1 kg and inertia tensor is normalized according to this mass unit. Therefore; a payload increment unit consisting of a 1 kg mass and a normalized inertia tensor is obtained in order to be used in the analysis.

Initial value for the payload is given as 600 payload increment units, which correspond to 600 kg mass with the same geometry of the original payload. Payload increments are done by adding 50 payload increment units over the previous payload value.

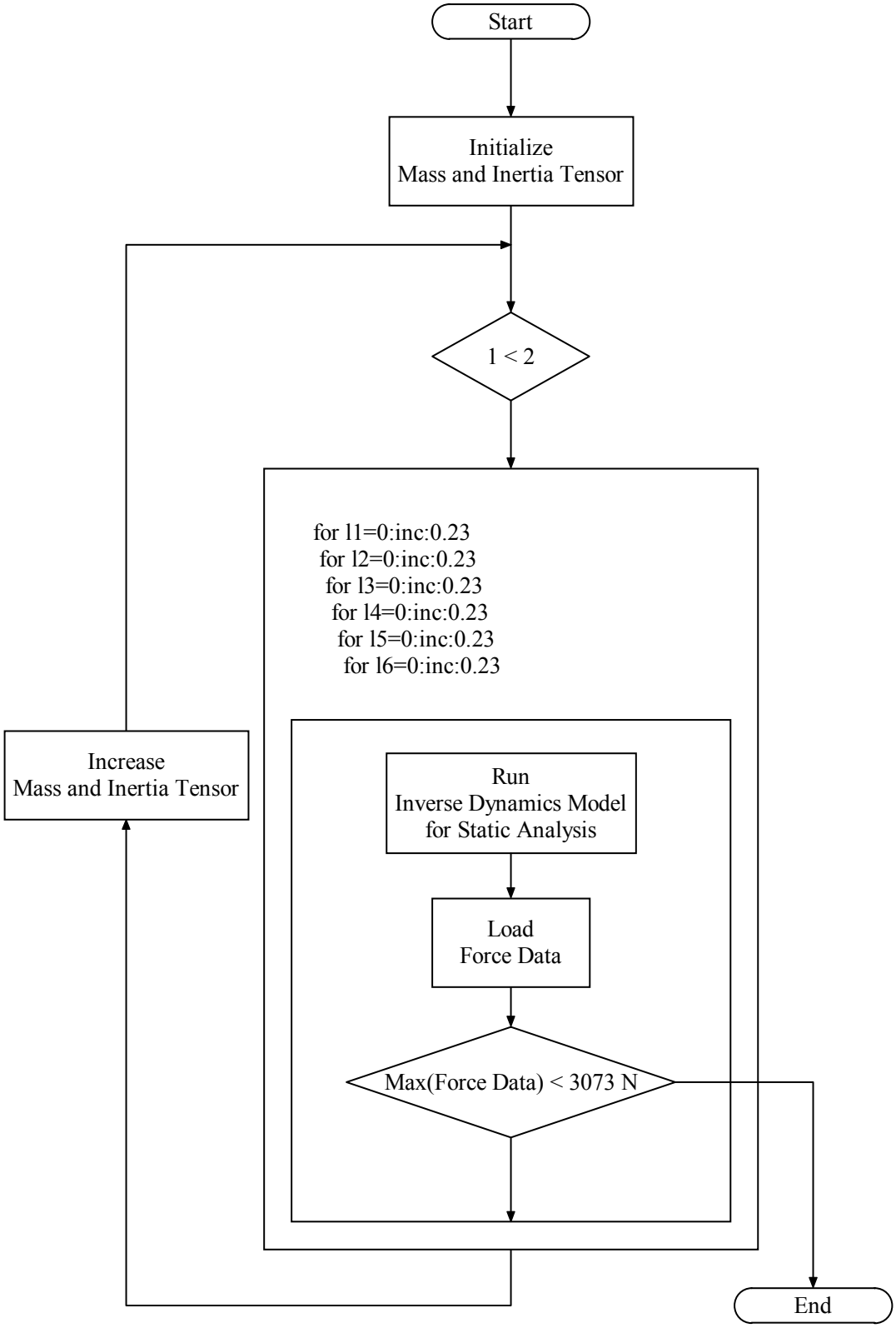


Figure 4.6: Flowchart for performance criterion (static analysis)

Actuators have position ranges of 230 mm and various combination of actuator positions are tried in their position limits. Actuator position increment is selected as one-third of the actuator position range which equals to approximately 76.7 mm. Then, each actuator position is varied in a nested-for-loop and checked whether one of the actuator is over the thrust limit.

The static analysis results in a mass of 750 kg with an actuator position set of $pos = [0 \quad 0 \quad 0.0767 \quad 0 \quad 0.23 \quad 0.23]^T$ m. As a result, the Stewart Platform can handle 700 kg mass statically at most.

As the position and mass increment values are more finely selected, more accurate results are expected but the simulation time increases dramatically.

4.2.2 Maximum Payload Satisfying the Specified Platform Motion

A typical set of acceleration values in six axes and a frequency range that the Stewart Platform operates should be specified in order to have a performance criterion. This specification should be verified kinematically before starting to the dynamic analysis, since the actuators may saturate due to their position and velocity limits. The lowest frequency in the frequency range specified is the most critical frequency in this case. Therefore, an input signal of platform motion in terms of accelerations in that frequency should be given to the inverse kinematics function and it should be verified that the actuators do not saturate due to their position and velocity limits.

A set of peak acceleration values as,

Yaw :1 rad/s^2
Pitch : 3 rad/s^2
Roll : 0.5 rad/s^2
X : 2 m/s^2
Y : 2 m/s^2

$\underline{Z}$: 2.5 m/s^2

and a frequency range of 2-70 Hz in the sinusoidal motion format are selected as the dynamic motion in order to check whether the actuators exceed their position or velocity limits. The verification is done by the inverse kinematics model in MATLAB®; in which the specified acceleration values are given as the input signals by integrating them twice, in the frequency of 2 Hz. The positions of the actuators are located at the midpoints of their position range initially; which assures that these accelerations can be achieved at least one set of actuator positions. Under these inputs, one of the actuator approaches to its velocity limit 0.254 m/s. Therefore, the general performance criterion for the Stewart Platform is finalized by verifying these accelerations and the frequency range kinematically. Figure 4.7 shows the required velocities of the actuators for the specified inputs.

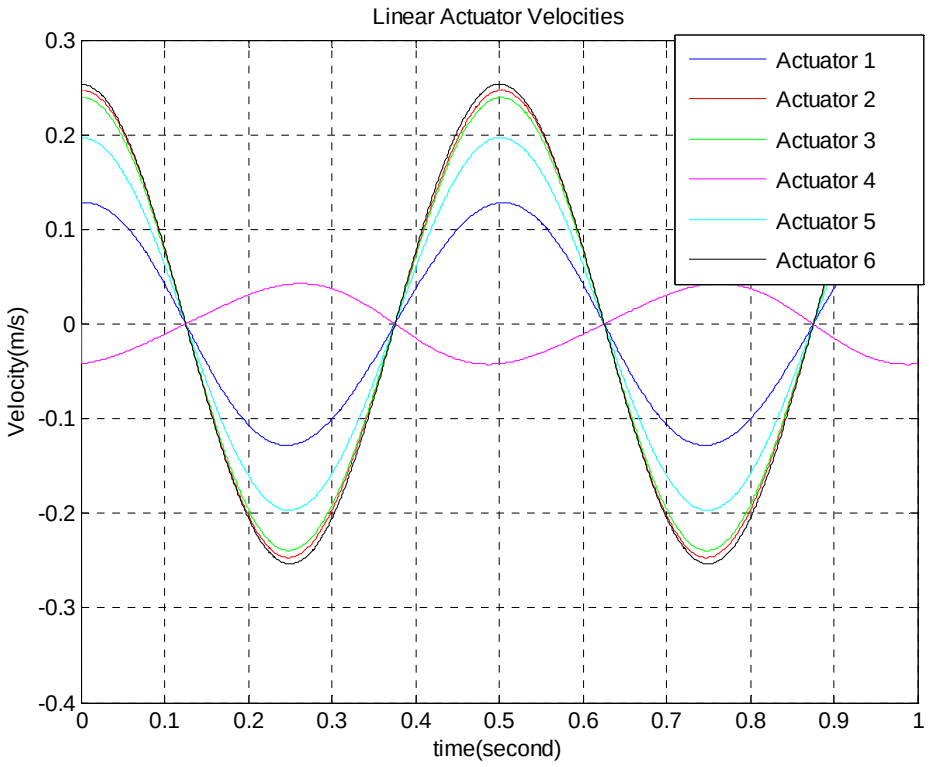


Figure 4.7: Required linear actuator velocities for the motion input that is used in performance criterion

The next aim is to find the maximum payload that the Stewart Platform can achieve these accelerations within the specified frequency range. A similar approach to the static analysis is achieved for the dynamic analysis. An initial mass and an inertia tensor value are defined and specified accelerations are given in sinusoidal motion format as inputs to the inverse dynamics model. Required forces of the actuators for these accelerations are found for many possible combinations of actuator positions and frequencies in the specified frequency range. Position and velocity limit checks are done in each run. If one of the limits is exceeded, force check is not done for that run. If the required forces do not exceed the thrust limit, mass and inertia tensor values are increased till one of the actuators exceeds its thrust limit. The flowchart summarizing the dynamic analysis is represented in Figure 4.8.

Initial value for the payload is given as 250 payload increment units which correspond to 250 kg mass with the same geometry of the original payload. Payload increments are done by adding 50 payload increment units iteratively.

Positions of the actuators start with an initial value of 0.05 m in order to have enough space for the dynamic motions. With an actuator position increment of 0.05 m, each actuator position is varied in a nested for-loop. Frequency variation is also done in the most inner loop by changing the frequency from 2 Hz to 70 Hz with a frequency increment of 10 Hz.

The dynamic analysis results in a 550 kg mass with an actuator position set of $pos = [0.15 \quad 0.05 \quad 0.05 \quad 0.15 \quad 0.05 \quad 0.05]^T$ m at 22 Hz. As a result; the Stewart Platform can achieve the specified acceleration values in the specified frequency range for a maximum payload of 500 kg mass.

If the mass, frequency and position increment values in this iterated simulation are decreased, then more accurate results can be obtained; however the simulation time increases dramatically.

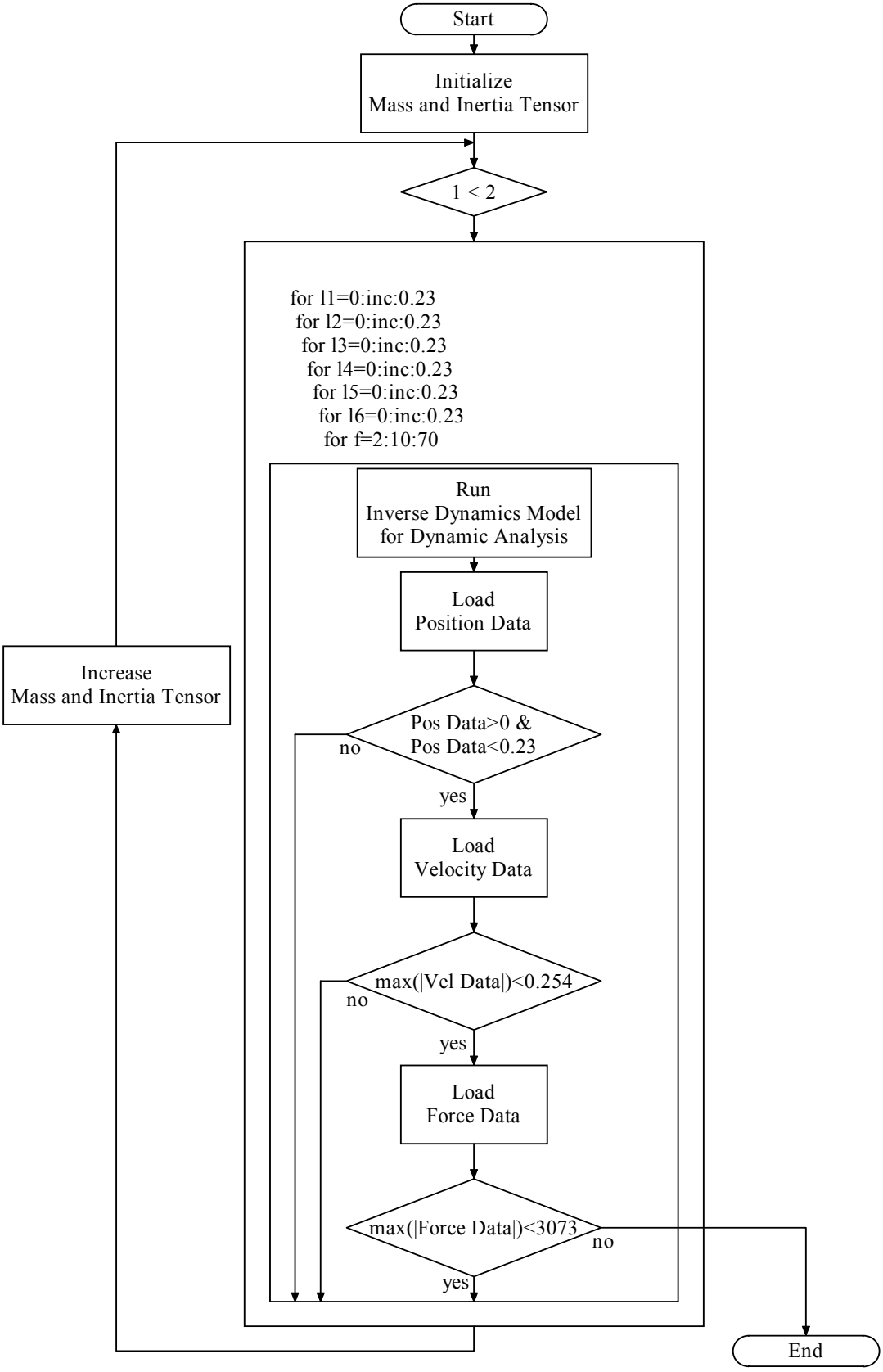


Figure 4.8: Flowchart for performance criterion (dynamic analysis)

4.3 Control Strategy

The major control strategy in the Stewart Platform is to find the required linear actuator movements for a desired platform motion by inverse kinematics and achieve these linear motions by closed-loop control in actuator motion axes. Considering the capabilities of the control system elements, major controlling action in real-time is summarized in Figure 4.9. As seen in figure below, real-time platform motion definition, real-time inverse kinematics solution, position control of the actuators and velocity demand delivery to the drivers are decided to be done in MATLAB®. Drivers are decided to be work in speed control mode closing speed and current control loops in a cascaded manner. They are driving electric motors in order to supply desired torque at electric motors. Kinematics and dynamics of the real system generates electric motor position and velocity as the output signals. They are used as the feedback signals for the position and the velocity control loops. An Inertial Measurement Unit (IMU) is used in order to verify the actual platform motion with the desired platform motion.

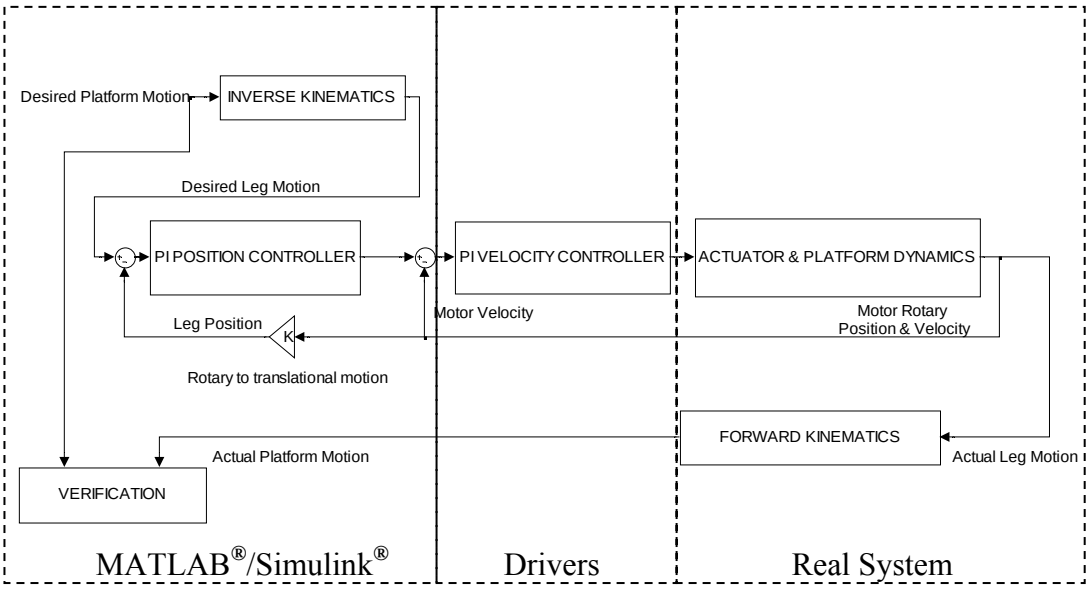


Figure 4.9: Real-time control (detailed)

Velocity control is much more critical than position control in this Stewart Platform because the major kinematical feelings are the rotational velocities and translational accelerations in three axes. Feedback signals of the velocity controllers are the angular velocities of the electrical motors. As mentioned before, a resolver is mounted to the back of the electric motor measuring electric motor's position. This resolver data is converted to incremental encoder data by the RDC module. Derivative operation of this encoder data is also done by this module in drivers, thus velocity is obtained and can be used as the velocity feedback signal. Assuming rigid structure of the linear actuators, one full revolution of the electric motor corresponds to the linear motion as "lead" of the roller-screw mechanism. This conversion is done in MATLAB® and actual position data of the linear actuators are obtained by this way. Current control loop is not shown here and taken as unity since its bandwidth is much higher than the velocity and position control loops.

To apply the control strategy, suitable controller parameters should be selected. To start controller tuning in real-time would be hard and unsafe for a Stewart Platform mechanism; therefore initial control system parameters are found by using the model in SimMechanics®.

4.3.1 Controller Tuning

The model constituted in SimMechanics® is used as the plant model including real-system parameters, constraints, conversions and actuators' torque and velocity limitations. The six linear actuators are controlled independent of each other. The general control strategy is shown Figure 4.10.

The SimMechanics® model is the forward dynamics model created in Chapter 3. As mentioned before, position and velocity control loops are working in a cascaded manner. First of all, velocity controller tuning will be done since it's the inner control loop.

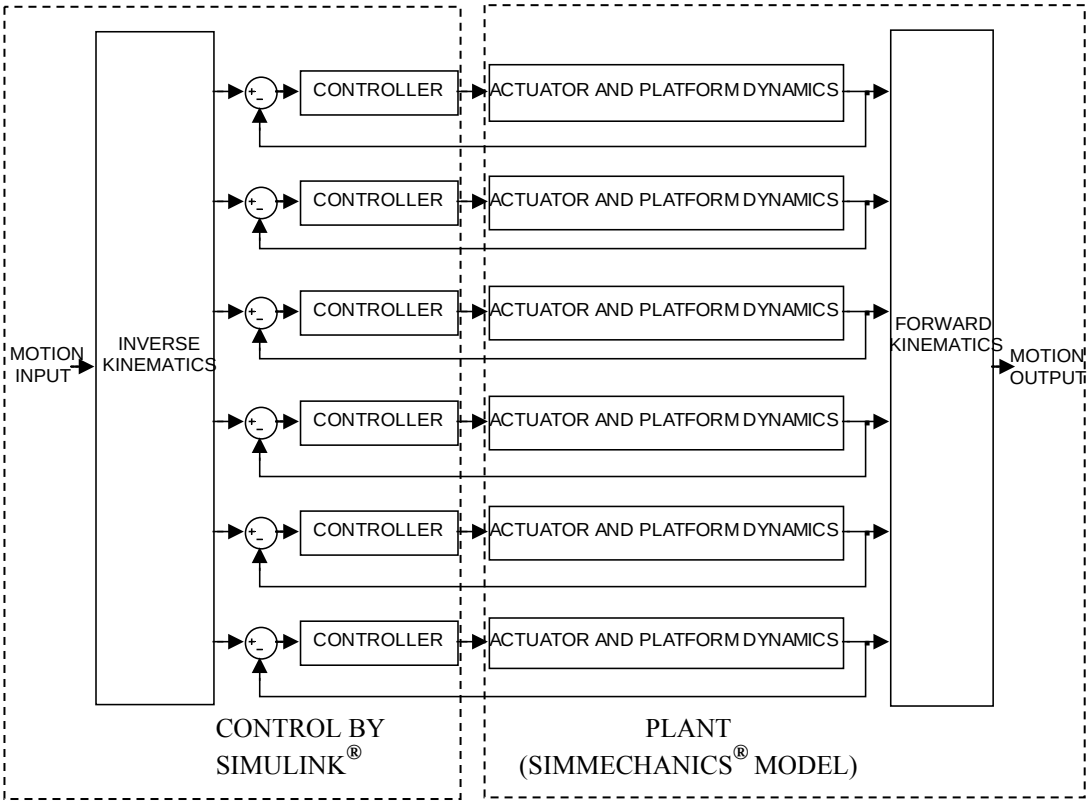


Figure 4.10: Major control system diagram in tuning

4.3.1.1 Velocity Controller Tuning

Velocity controller tuning is done by specifying and observing transient-response characteristics of the actuators. Step responses of the actuators when the platform acts in vertical direction is accomplished. As long as step responses of the actuators are investigated, it will not differ much whether the platform acts in vertical direction or in any other directions. Tuning starts with the trial and error approach and finalizes with the MATLAB® module Simulink Response Optimization® (SRO).

4.3.1.1.1 Velocity Controller Tuning by Trial and Error Approach

PI velocity controller is constituted in MATLAB® using the SimMechanics® model as a plant. General velocity controller tuning model is shown in Figure 4.11 and the velocity controller model is shown in Figure 4.12.

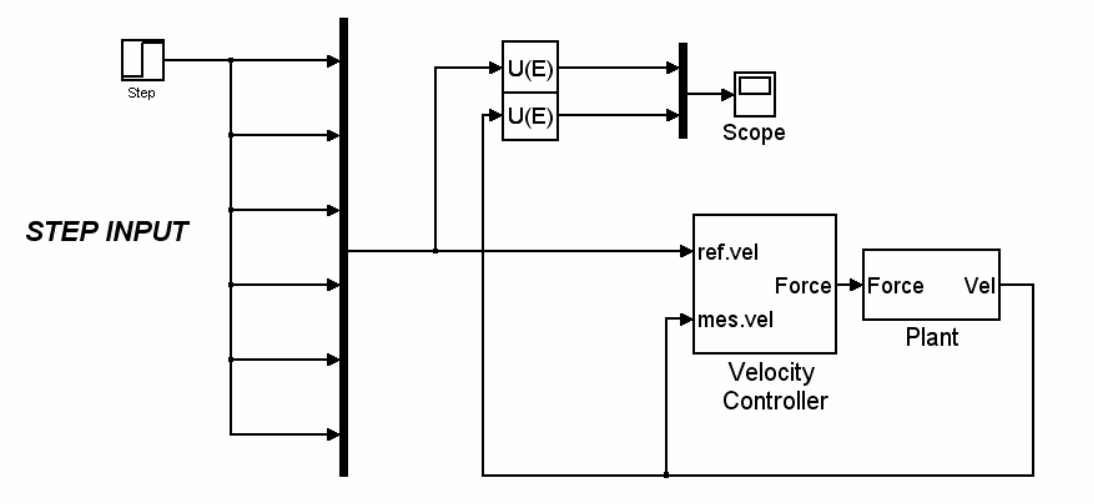


Figure 4.11: Model of velocity controller tuning in Simulink®

In simulation, step velocity demand is given to all of the actuators so that the platform motion in z-axis is expected. PI Controller is in the form of $K_p(1+\frac{1}{T_i s})$ where K_p is the proportional gain and T_i is the integral time constant [9]. Closed-loop velocity step response of one of the actuators is observed by the scope.

Various K_p and T_i values are tried in order to achieve desired velocity step response characteristics. Several K_p values are tried with a T_i value of 0.003 and the step responses are drawn for these cases on Figure 4.13. On the other hand, various T_i values are tried with a K_p value of 15 and the step responses are for these set of values are shown in Figure 4.14.

Practical experiences indicate that $K_p \approx 30$ amplifies the noise in the actuators obviously when the system is stationary but active. This is an undesirable case so $K_p = 30$ is specified as a limit for the proportional gain.

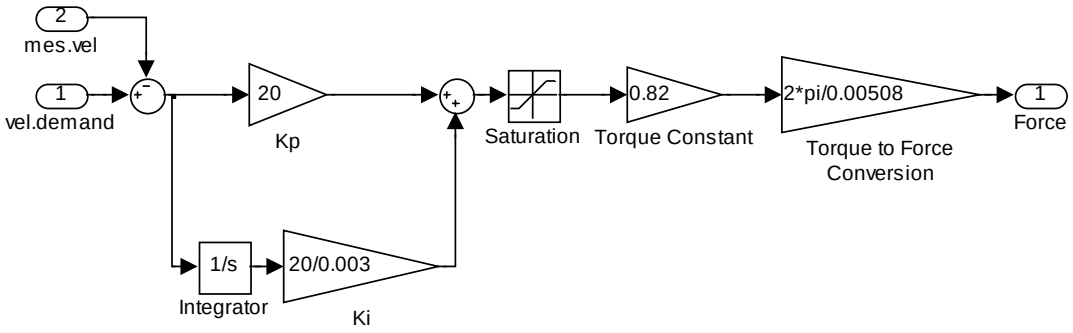


Figure 4.12: Velocity controller model in Simulink®

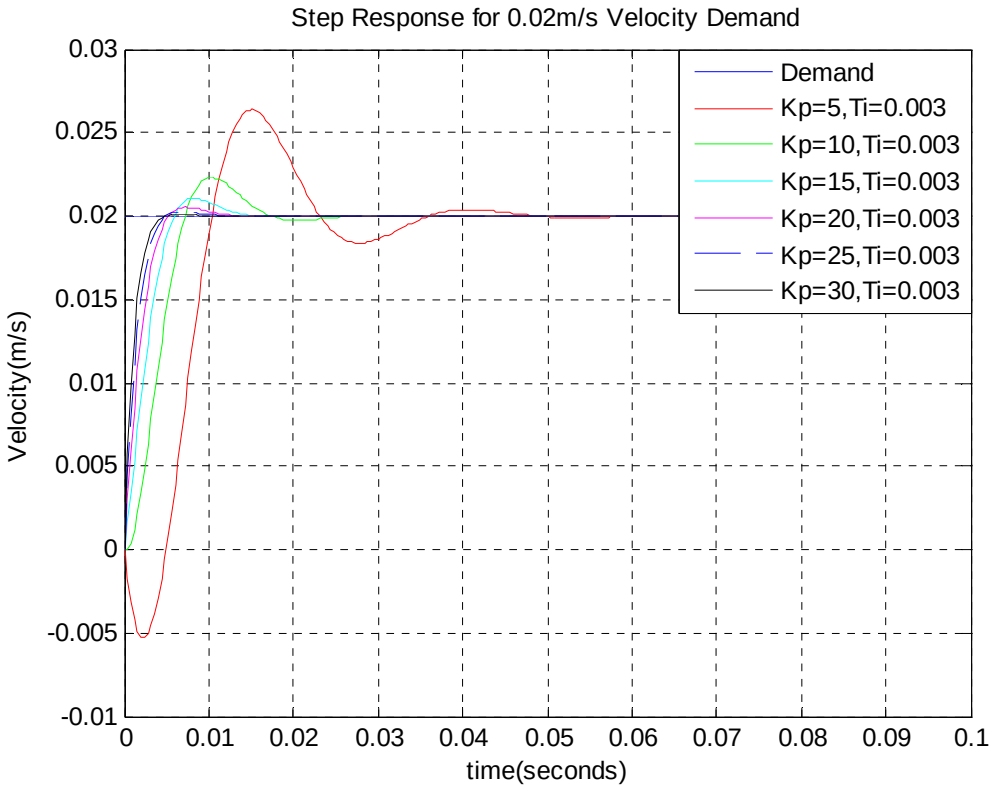


Figure 4.13: Step velocity response of one of the actuators (K_p variant)

Higher K_p values decreases rise time and maximum overshoot with a T_i value of 0.003 as seen in Figure 4.13.

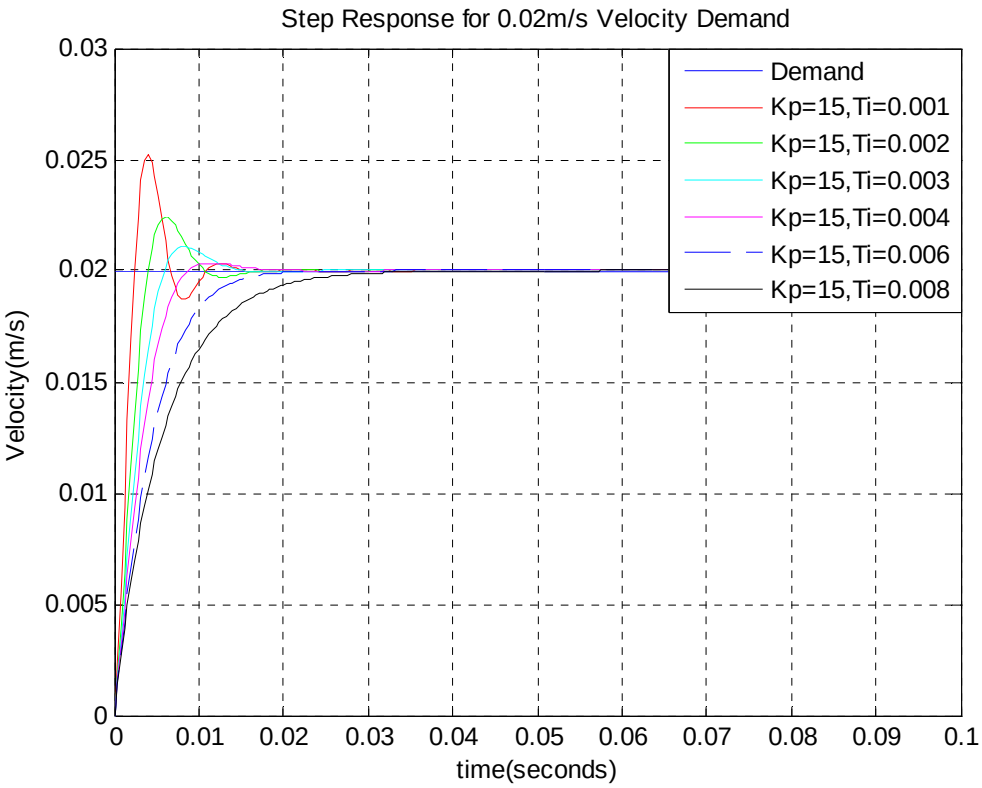


Figure 4.14 Step velocity response of one of the actuators (T_i variant)

Higher values of T_i decreases maximum overshoot but increases settling time and rise time as expected, which is shown in Figure 4.14.

In these simulations, K_p and T_i values are considered as independently; specifications can not be clearly assigned and seen on the figures, so an optimum solution may not be obtained. In order to achieve optimum parameters, Simulink Response Optimization® Tool is used.

4.3.1.1.2 Velocity Controller Tuning by Using Simulink Response Optimization® Tool

Simulink Response Optimization® is a tool that assists to tune and optimize physical systems which are modeled in Simulink environment. Parameters which can be

scalars, vectors, matrices can be tuned and optimized by defining constraints on the time-domain signals by graphically on the model. A signal constraint block, the only required block, is connected to the signal that the user wants to add constraint on it. Tuned parameters are selected and initial values are entered for these. Lower and upper bounds for the tuned parameters can also be added in this block. Simulink Response Optimization® tool then automatically converts the time domain constraints into an optimization problem; solves the problem using optimization routines taken from Optimization Toolbox® or the Genetic Algorithm and Direct Search Toolbox®. The constrained optimization problem formulated by Simulink Response Optimization iteratively calls for simulations of the Simulink® system, compares the results of the simulations with the constraint objectives, and uses gradient methods to adjust tuned parameters to better meet the objectives [10].

A signal constraint block is connected to one of the actuator's velocity output. Updated Simulink® model is shown in Figure 4.15. Since velocity step responses of the actuators are optimized, desired step responses characteristics can be reflected by simply moving the constraint bounds and segments in this signal constraint block. Desired response sub-tool under signal constraint window shown in Figure 4.16 can also be used in order to specify the step response characteristics. Specifications are identified without need to move constraint bounds by using the desired response sub-tool. Rise time for the step response is defined as the time that 99.5% of the step input is achieved in the response. Settling time is defined as the time that the output signal enters 0.5% bound of the input. Maximum overshoot is defined as 10% of the desired input. Settling time should be a maximum value of 0.02 seconds and rise time should be a maximum value of 0.006 seconds. After specifying these specifications, K_p and T_i are specified as tuned parameters and their initial values are entered as $K_p = 5$, T_i=0.005. Lower and upper bounds for K_p are assigned as 5 and 30 respectively. Lower and upper limits are assigned as 0.001 and 0.008 for T_i respectively, depending on the experiences in Section 4.3.1.1.1.

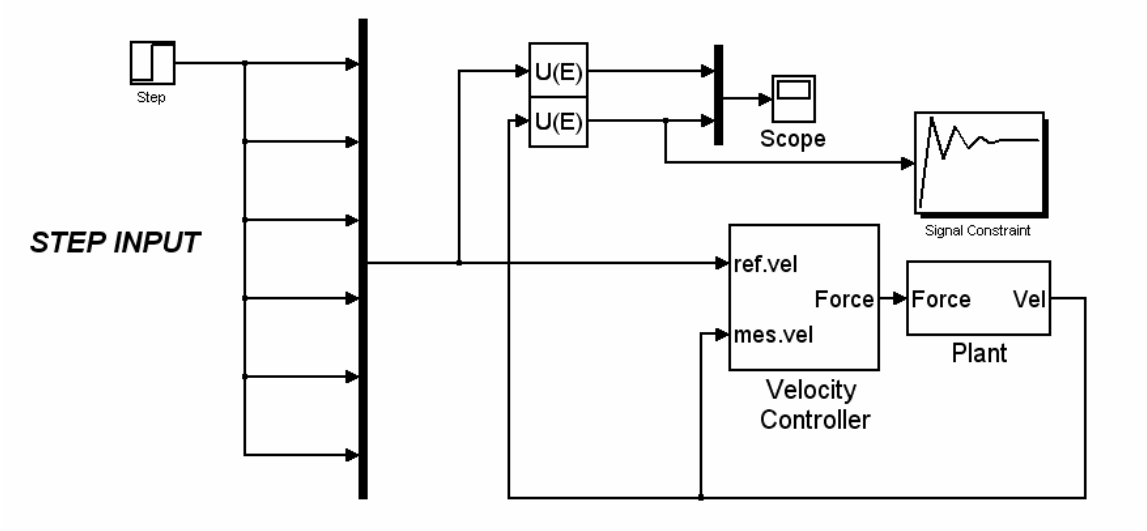


Figure 4.15 Updated Simulink® model for SRO® tool

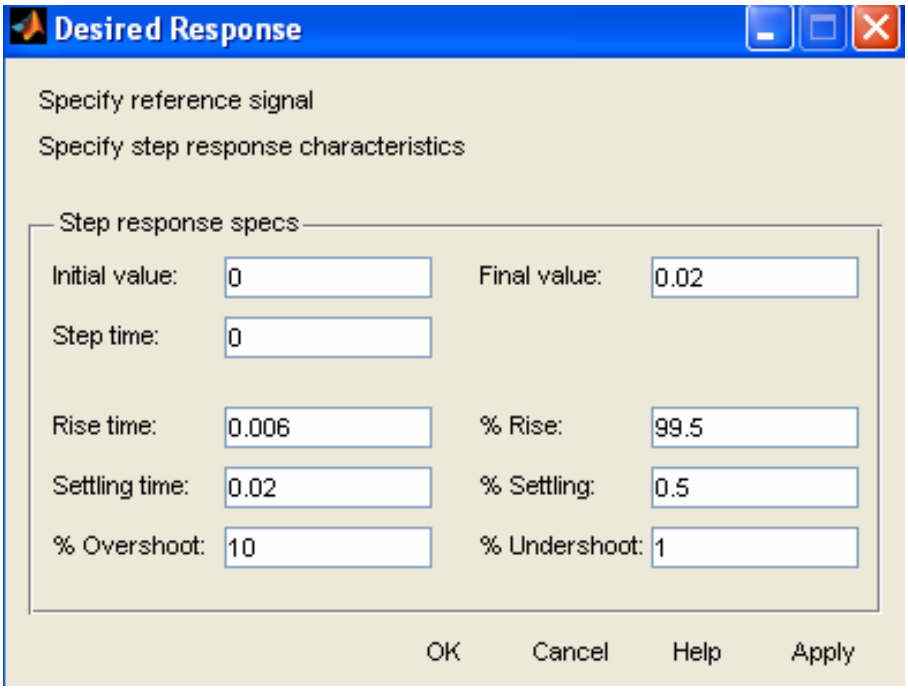


Figure 4.16: Desired response subtool

The signal bounds representing the step response specifications; initial step response, intermediate step responses and thus the optimum response are shown in Figure

4.17. A window that SRO® module generates during optimization process is also shown in Figure 4.18.

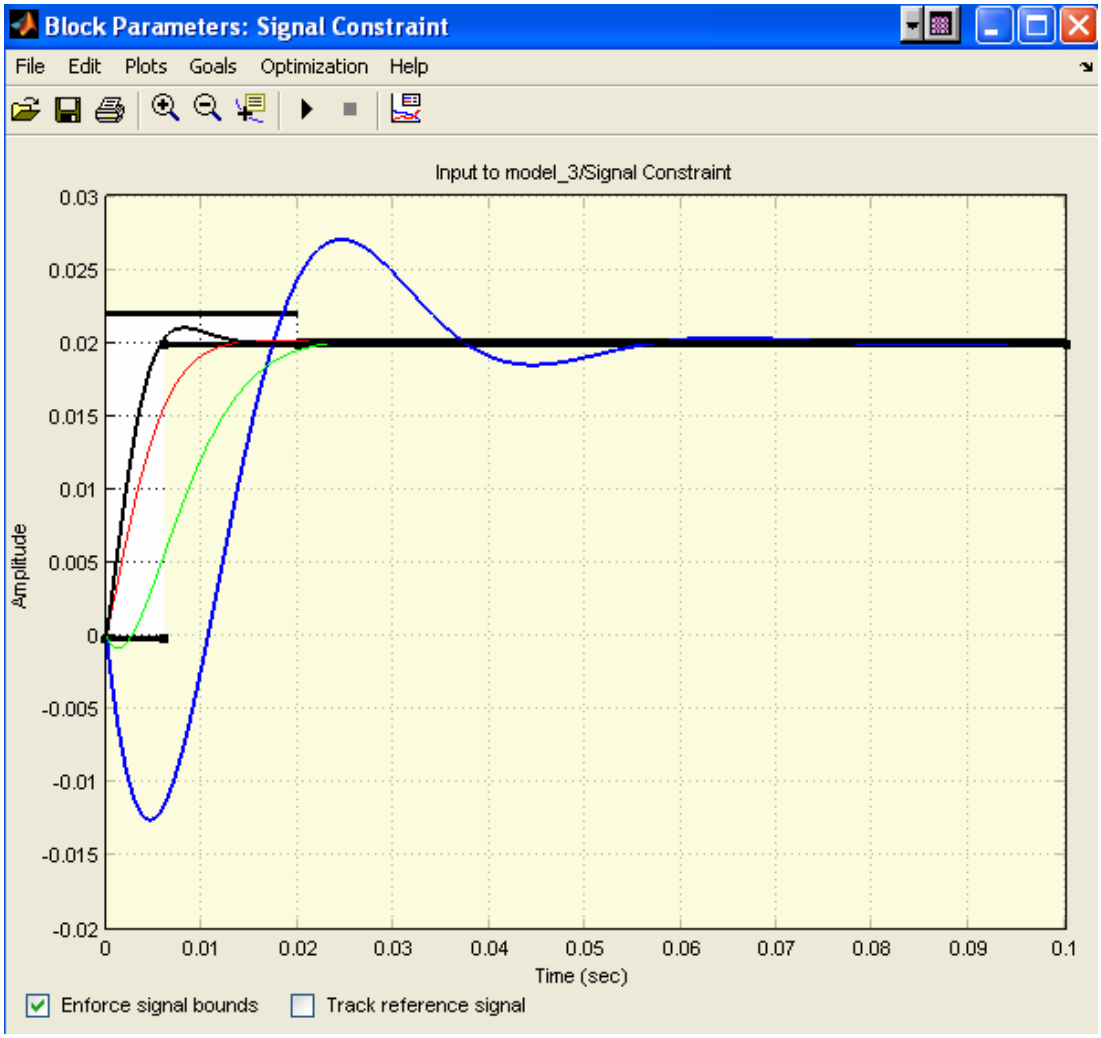


Figure 4.17: Signal constraint block

As seen in figure above, desired response specification is achieved after 3 iterations. Optimum values are found as $K_p \approx 21.6$ and $T_i \approx 0.003$. $K_p = 20$ and $Ti = 0.003$ are selected depending on the SRO® results and these values are going to be used in position controller tuning and they are also applied in the real-time control system as the parameters of the velocity controllers.

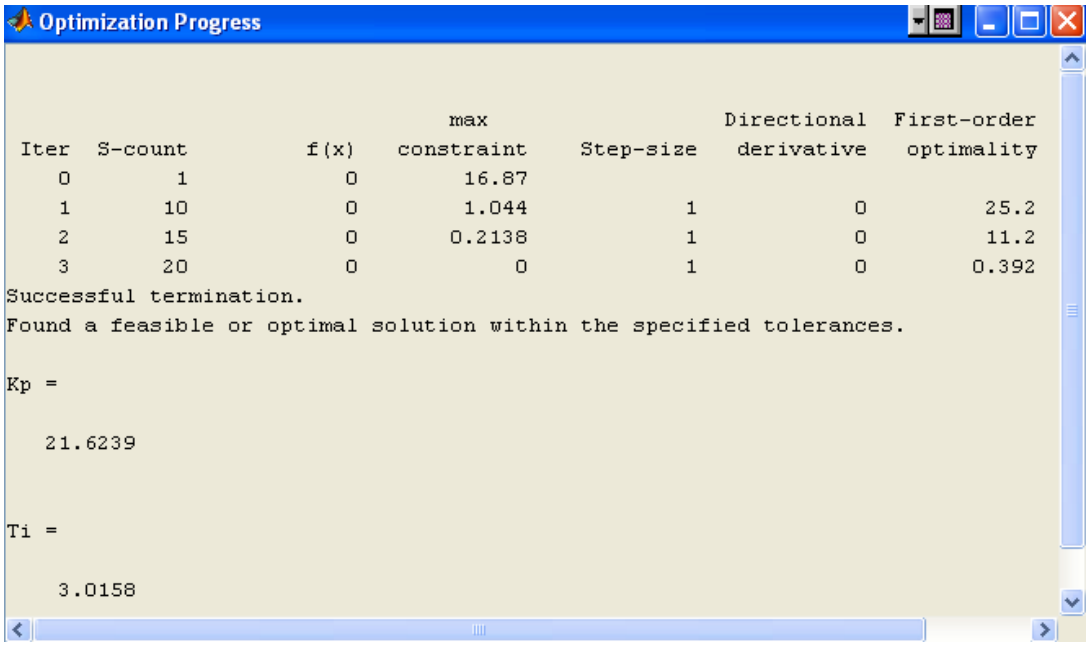


Figure 4.18: Optimization progress window

4.3.1.2 Position Controller Tuning

The next step is to tune the position controller parameters. Position and velocity loops are cascaded as seen in Figure 4.19. In position controller tuning model, the previously designed velocity loop is used as an inner loop. In this case, a step position demand for the upper platform is given in z-direction (vertical motion) and optimum position controller parameters are selected according to step response characteristics of the actuators.

Actual position controller structure is given in the Figure 4.20. Numerical differentiation of position demand is accomplished and fed as velocity feedforward. Velocity feedforward is applied in order to maintain high bandwidth since the platform's velocity and acceleration outputs are much more important. Integral control is added to proportional gain in order to eliminate steady-state position error although it is not very critical. PI controller is in the form of $K_p + \frac{K_i}{s}$.

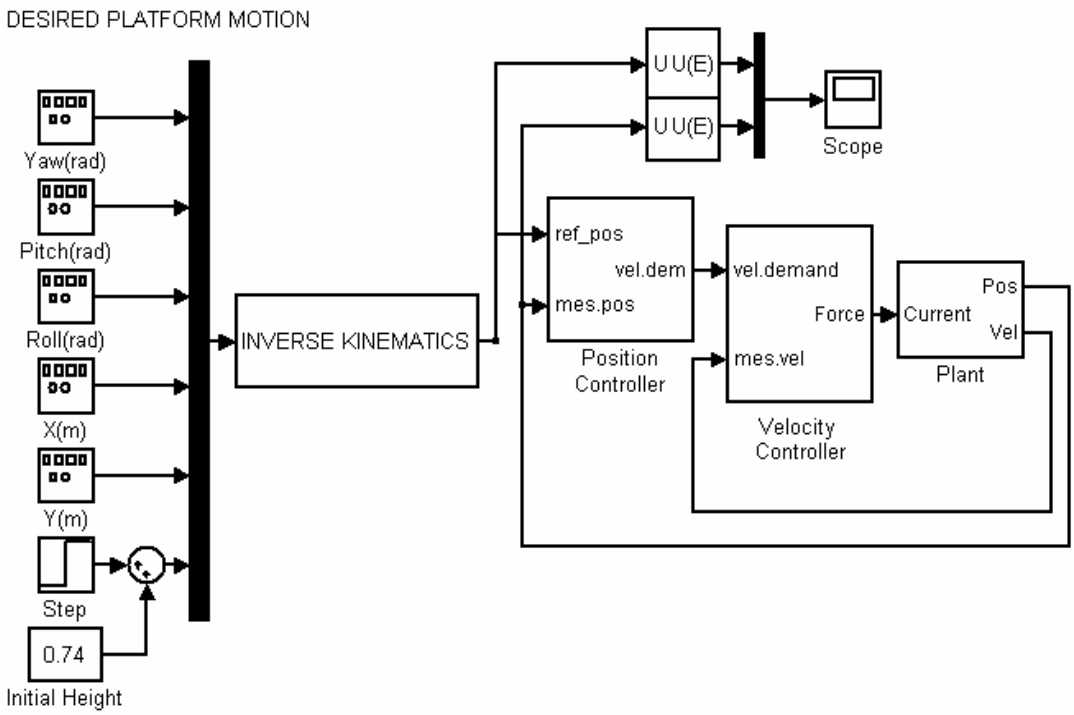


Figure 4.19: PI position controller tuning model in Simulink®

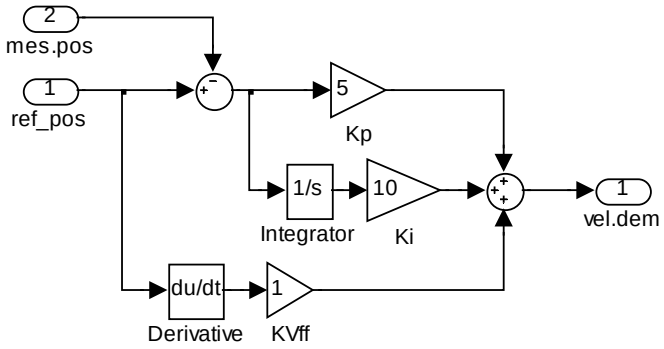


Figure 4.20: Position controller model

Step position response is given in the Figure 4.21. In this figure, the actuator becomes to the desired position about 2 seconds, while making an overshoot of 20% which is acceptable by means of a position controller. The settling time value is acceptable in terms of positioning since critical issue is to attain dynamic motions on

the platform, as mentioned before. Therefore, $K_p = 5$ and $K_i = 10$ is decided to be used in the real-time control system.

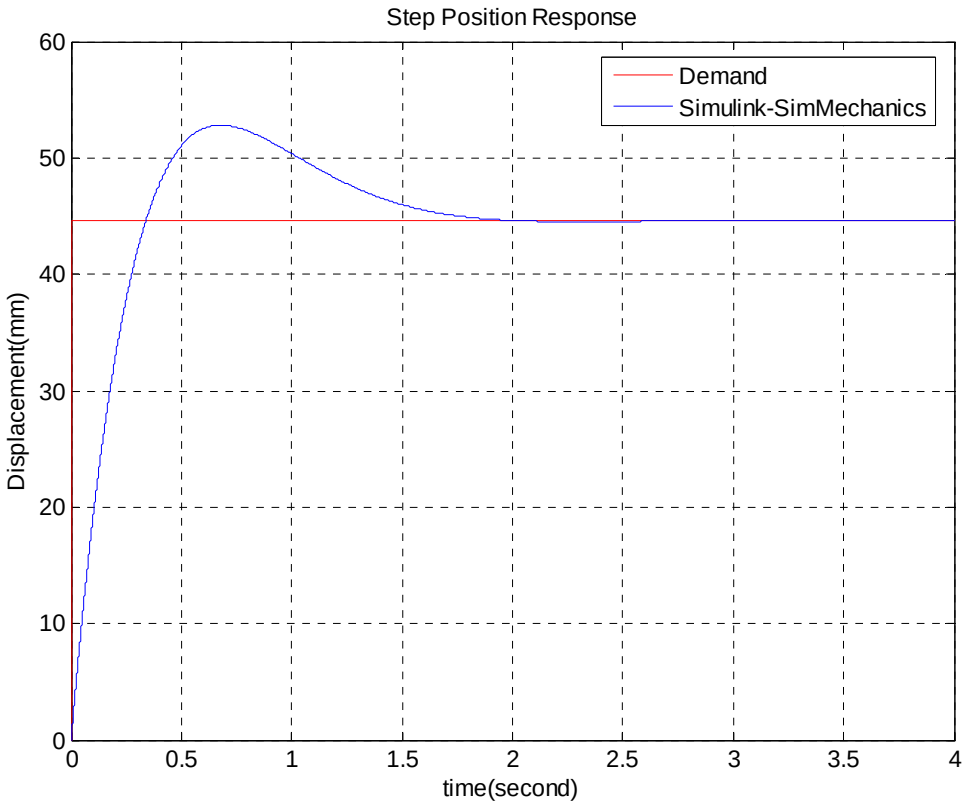


Figure 4.21: Step position response of one of the actuators

4.3.2 Real-Time Control

In the real-time control system, main operations are carried out in MATLAB® environment such as operation start and stop, position control and such arrangement facilities; besides this, drivers takes velocity demand, i.e., output of position controller as analogue input and actuates the actuators by performing velocity and current control loops in a cascaded manner. Drivers also play role on security facilities involving limit switches and mechanical brake issues.

Real-time control operations such as platform motion demanding, conversion of platform motion to actuator displacements, position controlling and velocity demanding to each driver are fulfilled by the major Simulink® model which is shown in Figure 4.22. Each subsystem of this model is deeply discussed in Appendix C. These operations are based on the flowchart given in Figure 4.23. MATLAB operations start with a homing process; which is represented by the flowchart represented in Figure 4.24.

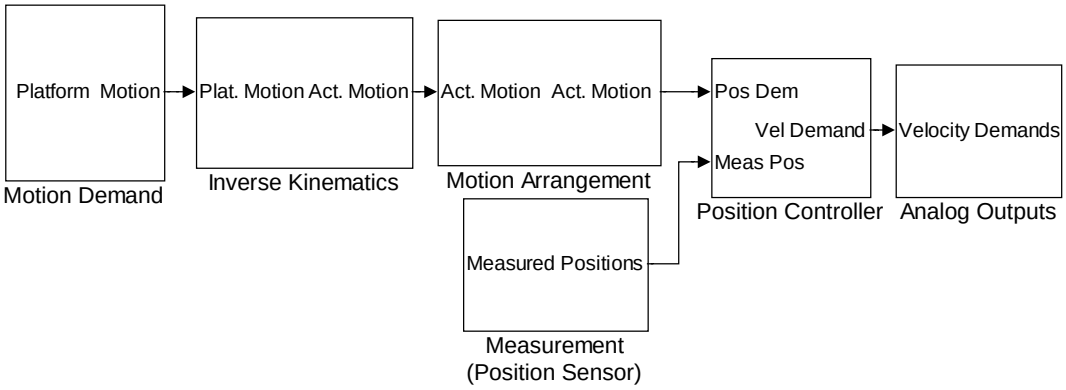


Figure 4.22: Major real-time control system blocks in Simulink®

As mentioned before, drivers fulfill the velocity controller processes combined with the security facilities. Operations are done based on the flowchart shown in Figure 4.25.

Limit switches are used in the Stewart Platform for security purposes as well as referencing. They are integrated with the drivers. If the actuator is in lower/upper limit, actuator is not permitted to go down/up further by automatically giving zero velocity demand but it is allowed to accept positive/negative velocity demands from MATLAB®.

Meanwhile, it should be mentioned that mechanical brakes are used for each actuator for additional security purposes although it is not stated in flowcharts. By an on/off

switch, user can lock any actuator in any time mechanically, which is governed by the drivers.

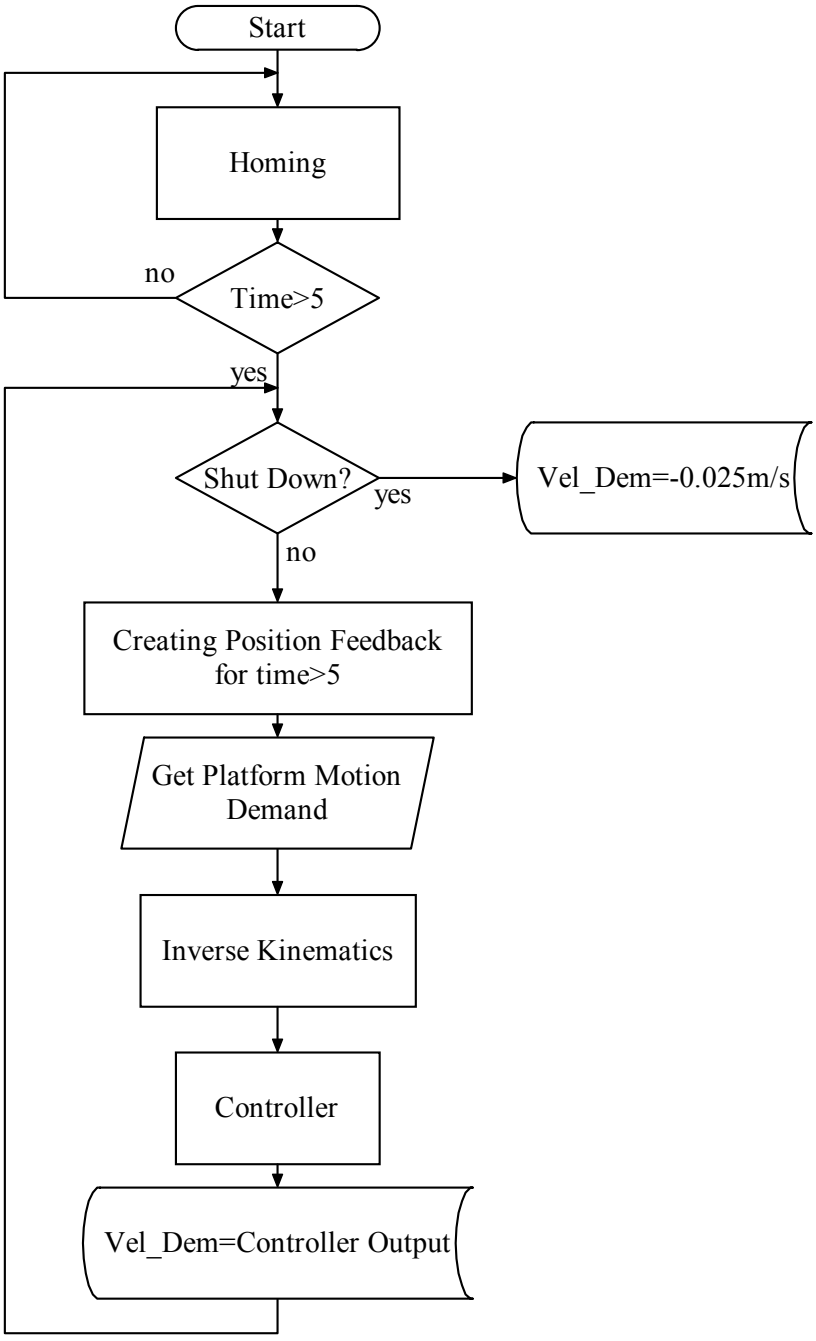


Figure 4.23: Flowchart representing the real-time control system regarding to MATLAB®

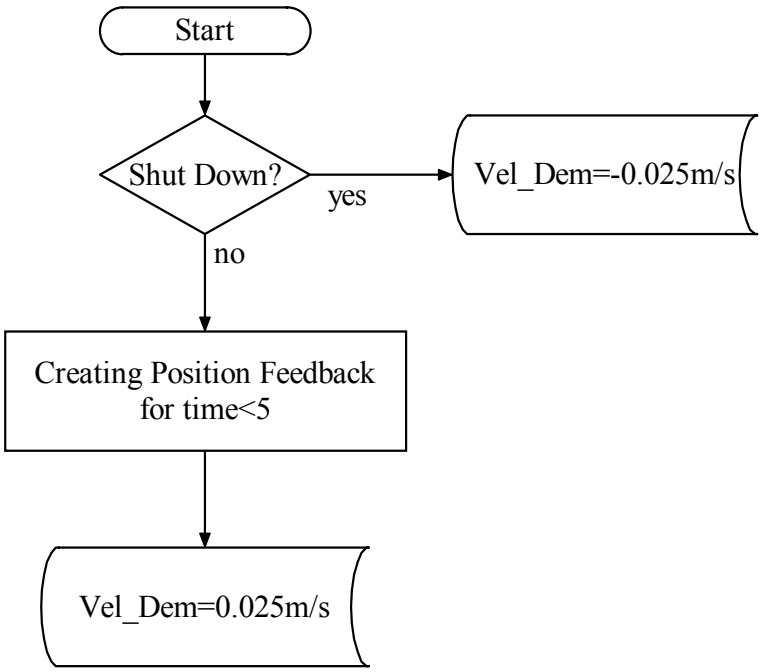


Figure 4.24: Flowchart representing homing process

As stated before, incremental encoder data are available for the actuators and these encoder data are read by the DAQ boards. These boards accept quadrature and index signals which are related with the incremental encoder data. Encoder steps are obtained from these input signals by these boards. One full revolution of the motor, which means a linear displacement of 5.08 mm of the actuator, corresponds to 4096 encoder steps. This is so, since incremental encoder data are obtained as 12-bit by RDC modules. As the index signals are read by the boards, encoder step values turn out to zero at that instant. At start-up of the operation, as the electric motors rotate in one direction, encoder step values increase up to some values and fall down to zero as the index signals arise. These values depend on the relative places of the electric motor shafts. After encoder steps fall down to zero, if the user continues to operate the actuators in the same direction, encoder step values again fall down to zero after 4096 encoder steps occur, as the following index signals arise. This information results in two different encoder process algorithms for both homing and simulation cases; which are represented in detail by flowcharts in Figure 4.26 and Figure 4.27 respectively.

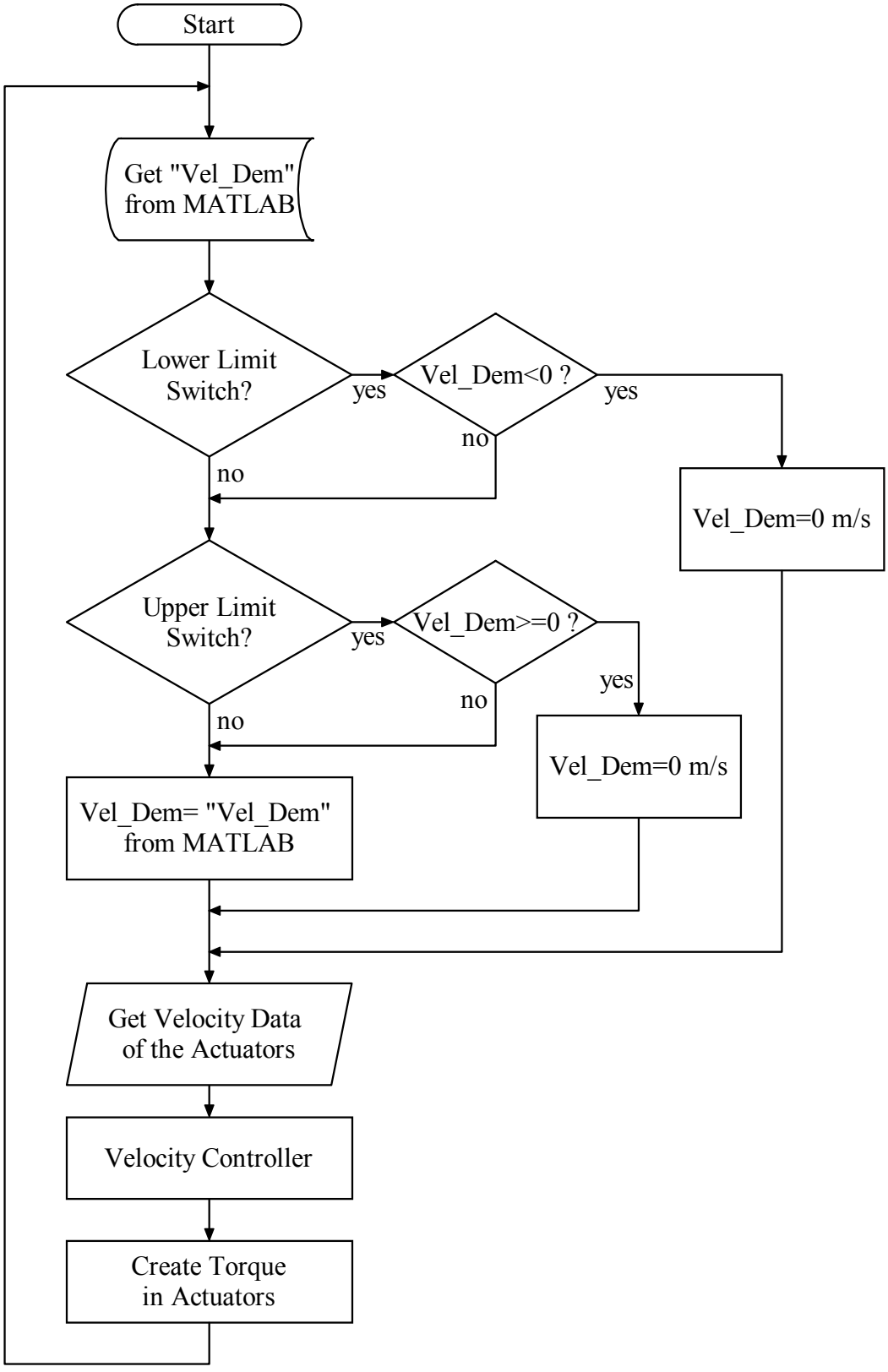


Figure 4.25: Flowchart representing the operations in drivers

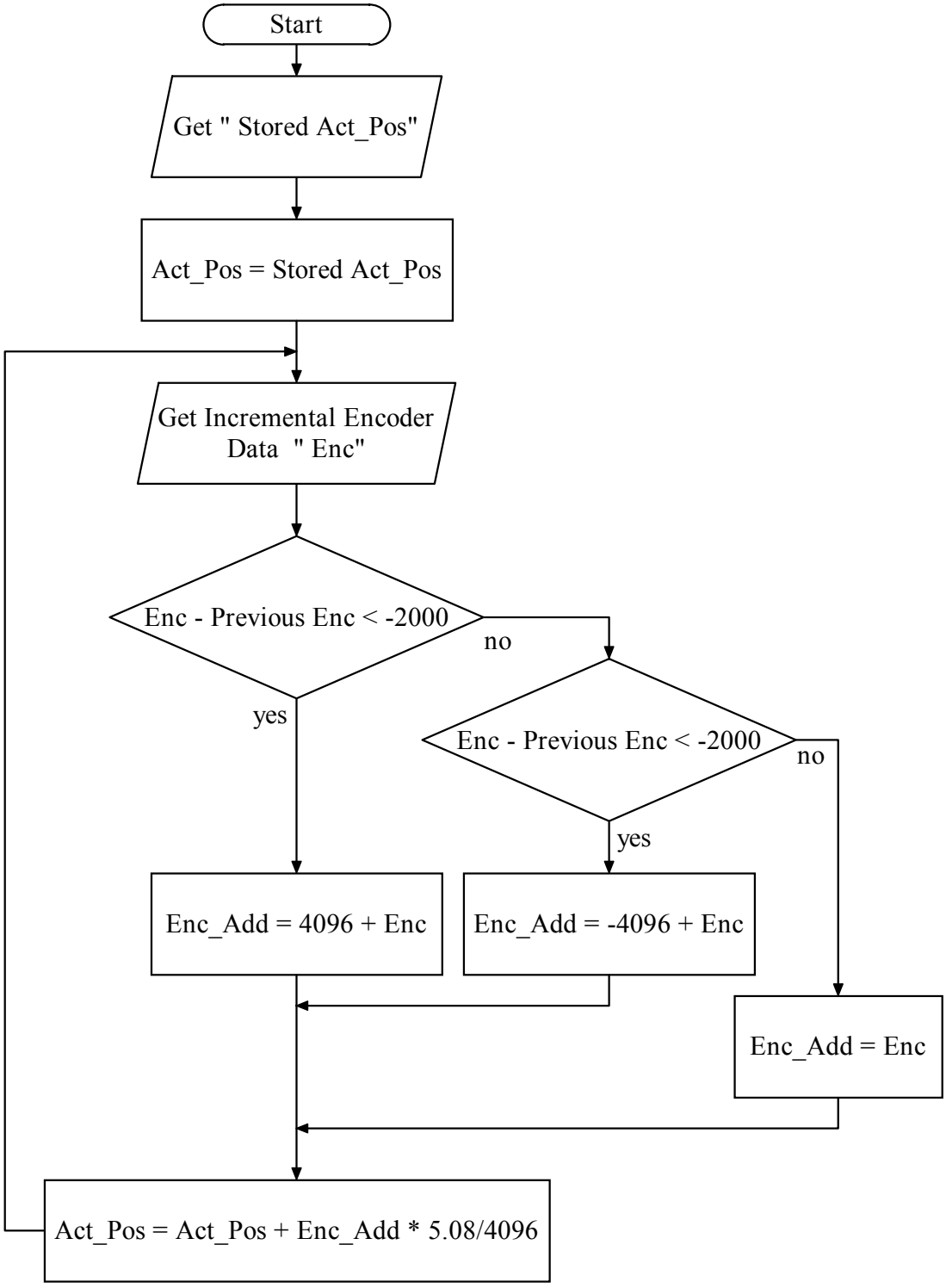


Figure 4.26: Flowchart representing position feedback acquisitions in homing process

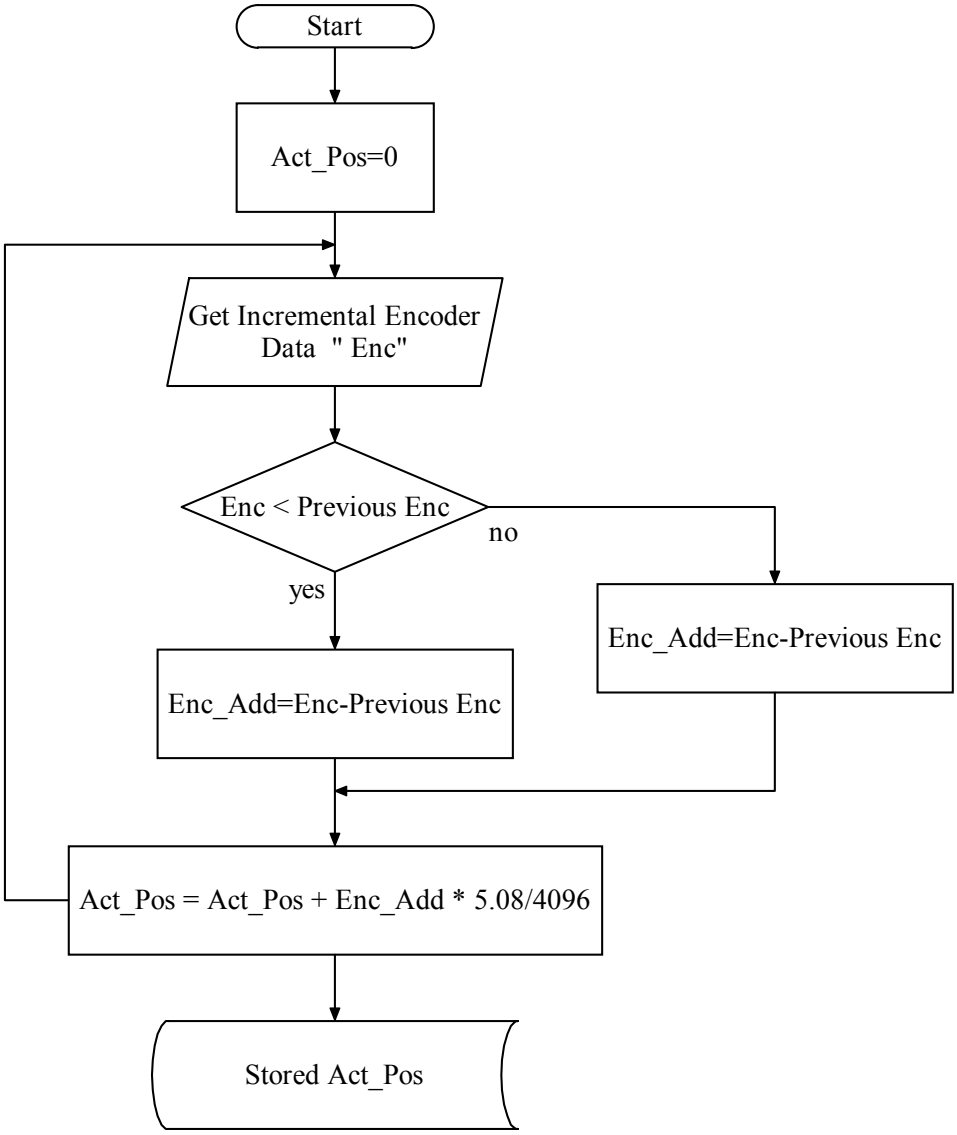


Figure 4.27: Flowchart representing position feedback acquisitions in simulation mode of the Stewart Platform

4.3.2.1 Real-Time Velocity Controller Tuning

Controller structure and parameters found in section 4.3.1.1.2 are used in velocity controller of the real system. A step velocity demand of 20 mm/sec in actuator axis is created and actual velocities of actuators are measured. Velocity acquisition is done by taking derivative of the encoder data of the electric motors delivered from the

drivers. In Figure 4.28, both real-time step response and the step response found in Section 4.3.1.1.2 are shown.

With a parameter set of $K_p = 20$ and $T_i = 0.003$, it is observed that the platform shows no overshoot performing an over damped system behavior. It has pure time delay of about 2 ms which is possibly originated from electronic hardware and stiction due to friction. Real system is rather damped than the one in simulation since viscous friction factor which is not accounted in the simulation, arises. Linear guide ways and linear bearings on the actuators are the major sources for the viscous friction. On the other hand, the rise time and the settling time of the step response are satisfactory, which are almost the same as the ones in the simulation.

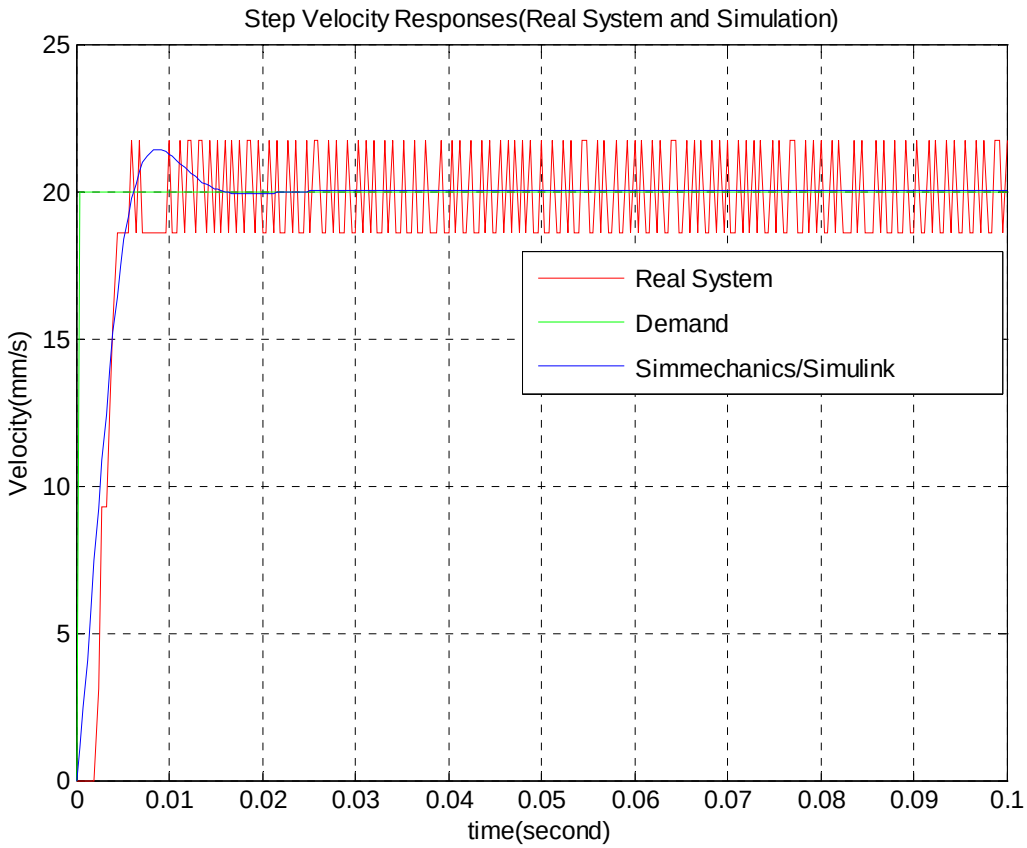


Figure 4.28 Step velocity responses of one of the actuators for the model in Simulink® / SimMechanics® and the real system

In Figure 4.28, oscillations are observed on velocity response of the actuator in real system but actually these oscillations come from the numerical derivative operation of the encoder data, performed in MATLAB®. Encoder data is in digital form and one bit uncertainty of the encoder data can create these oscillations, which should be ignored.

4.3.2.2 Real-Time Position Controller Tuning

In the real-system, position controller structure and controller parameters decided in Section 4.3.1.2 are used. A step position demand of 50 mm in z-axis is given for the upper platform motion and required actuator leg lengths are found by inverse kinematics. Desired actuator displacements and measured actuator displacements are shown in Figure 4.29.

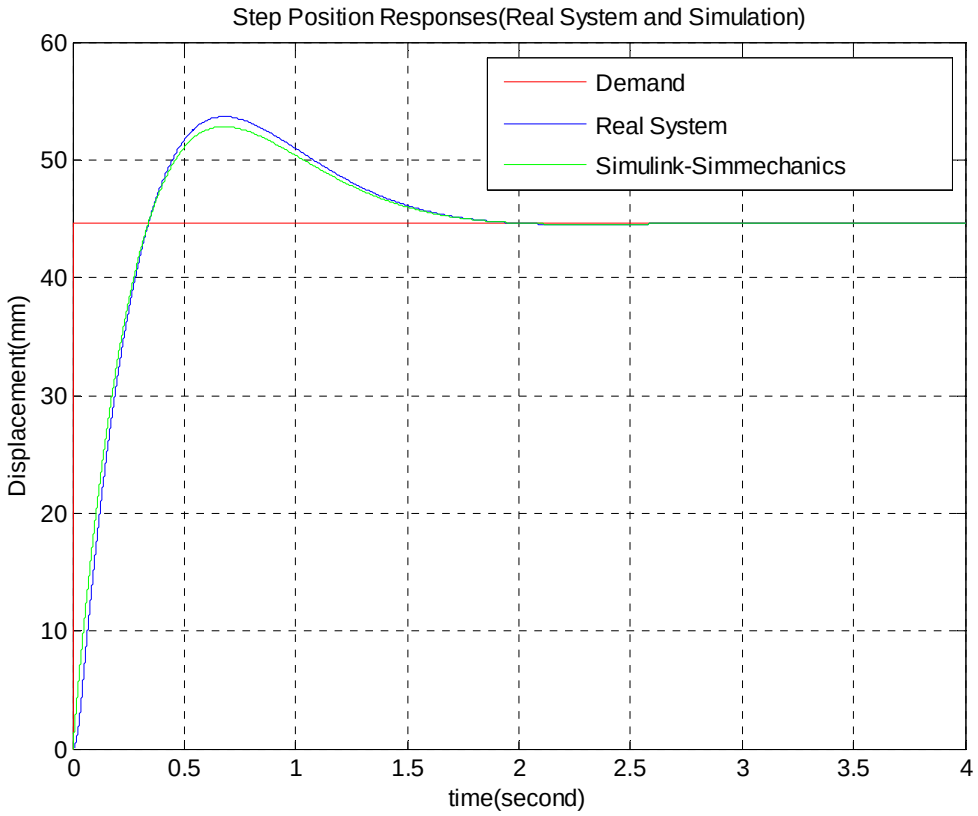


Figure 4.29 Step position response of one of the actuators for the real system

In Figure 4.29, it is observed that the actuator settles down about 2 seconds, which is acceptable by means of a position controller. Comparing the position step response of the real system with the SimMechanics® model in Section 4.2.1.2; it is observed that they show the same characteristics with a small difference in settling time.

4.3.2.3 Actuator Responses for Various Sinusoidal Inputs

In frequency domain, response of the Stewart Platform is important considering the simulation of the tank motion on APG track. The platform has to perform a motion in pitch axis having frequency content from 0.5 Hz up to 70 Hz; therefore performance of the Stewart Platform for sinusoidal responses has to be observed. This performance depends on strength and stiffness of parts, non-linearity as well as controller performance.

The controller designed in the previous sections are checked by moving the platform with various sinusoidal inputs in pitch axis up to 300 Hz. Bode magnitude plot in actuator axis is obtained by dividing the measured actuator displacement with the desired actuator displacement at various frequencies, experimentally. Flexible modes of some of the parts such as the upper platform, the upper and the lower legs may not be seen or may partially seen in this test, since measurement is done by the encoder data which is originated from the back of the electric motor. Effects of nonlinearities (clearances in bearings, backlashes etc.) in the system may not be seen or may partially be seen in this test. As far as controller performance is considered, this test gains importance.

Looking at the Figure 4.30, it is seen that the major characteristic of the platform is a second-order system. Some partial structural mode contributions to this second-order behavior are observable on the measured signal about 40 Hz and 70 Hz, specifically for motion in pitch axis. Control system has a bandwidth of about 220 Hz where -3 dB cut off occurs. It makes the peak value of 7 dB about 150 Hz; which is not critical as far as the main concern is up to 70 Hz. An amplification of 4 dB occurs at 70 Hz, which is the most amplification ratio up to 70 Hz.

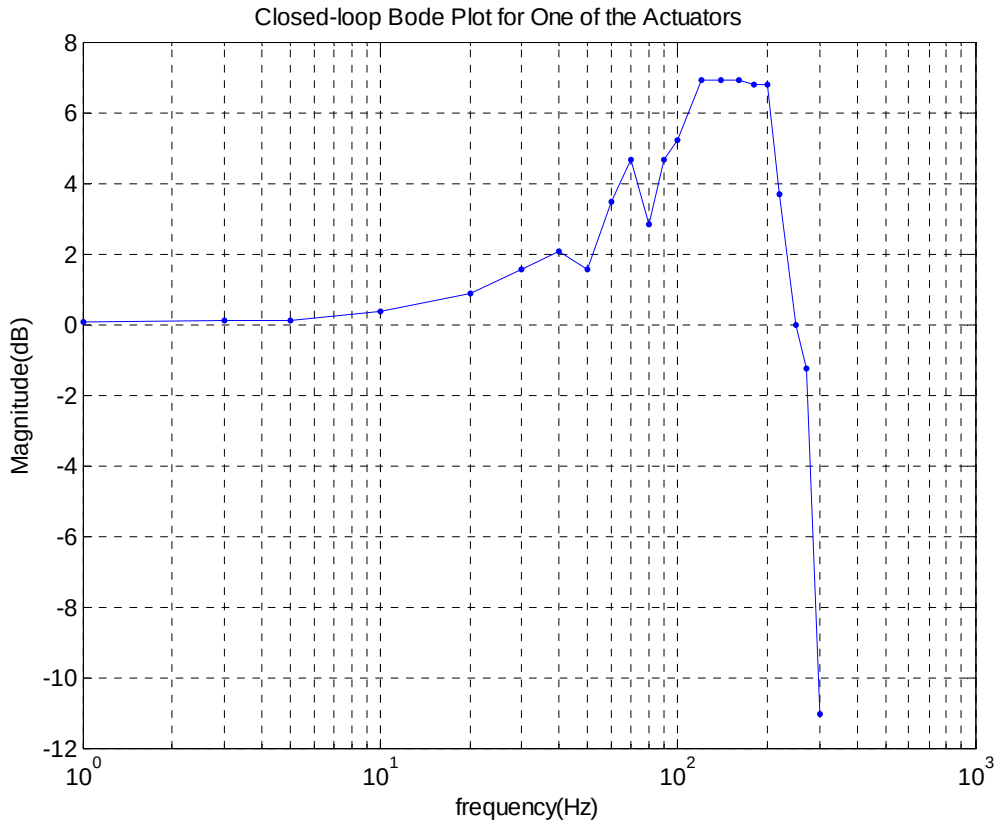


Figure 4.30: Closed-loop bode-magnitude plot for one of the actuators

4.3.2.4 Verification of Platform Motion Using Inertial Measurement Unit (IMU)

An external sensor IMU measuring three angular velocity and three translational accelerations, is used in order to check the accuracy of platform motion. IMU has a bandwidth of 75 Hz being capable of giving idea about the motion of platform. It has Micro Electromechanical System (MEMS) technology. It gives serial data in RS-422 (differential) format. This data is converted to RS-232 (single-ended) and read by computer serial connection [11]. Then, data acquired are taken into MATLAB®, analyzed and checked with the desired platform motion.

An m-file governing the analysis of IMU data is shown in Appendix A.4. IMU is shown in Figure 4.31 and IMU measurement setup is shown in Figure 4.32.

Figure 4.31: Inertial measurement unit

The accuracy of platform motion mainly depends on real-time inverse kinematics solution as well as controller performance. Inverse kinematics solution errors probably come from wrong kinematical identification because of manufacturing and assembly tolerances. Controller performance can be affected by flexible modes of the platform at relatively high frequency motions. Affects of flexible modes of the parts between the electric motors and the upper platform may not be seen on feedback measurement. 0.1 mm backlash in actuators is also one of the main sources deteriorating platform motion accuracy.

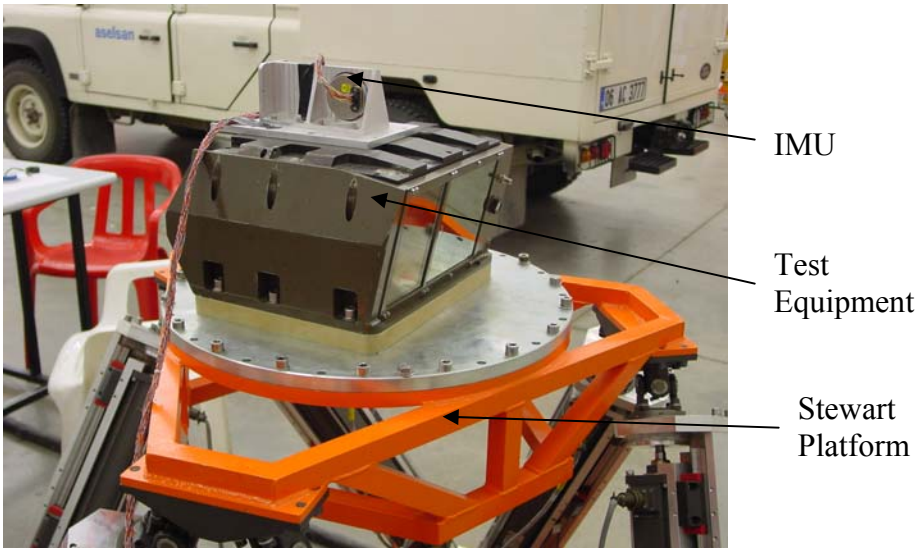


Figure 4.32: IMU measurement setup

For a tank motion in pitch axis, platform motion is measured by the IMU and checked with the desired platform motion. Desired and real motion of the platform in pitch axis is shown in Figure 4.33.

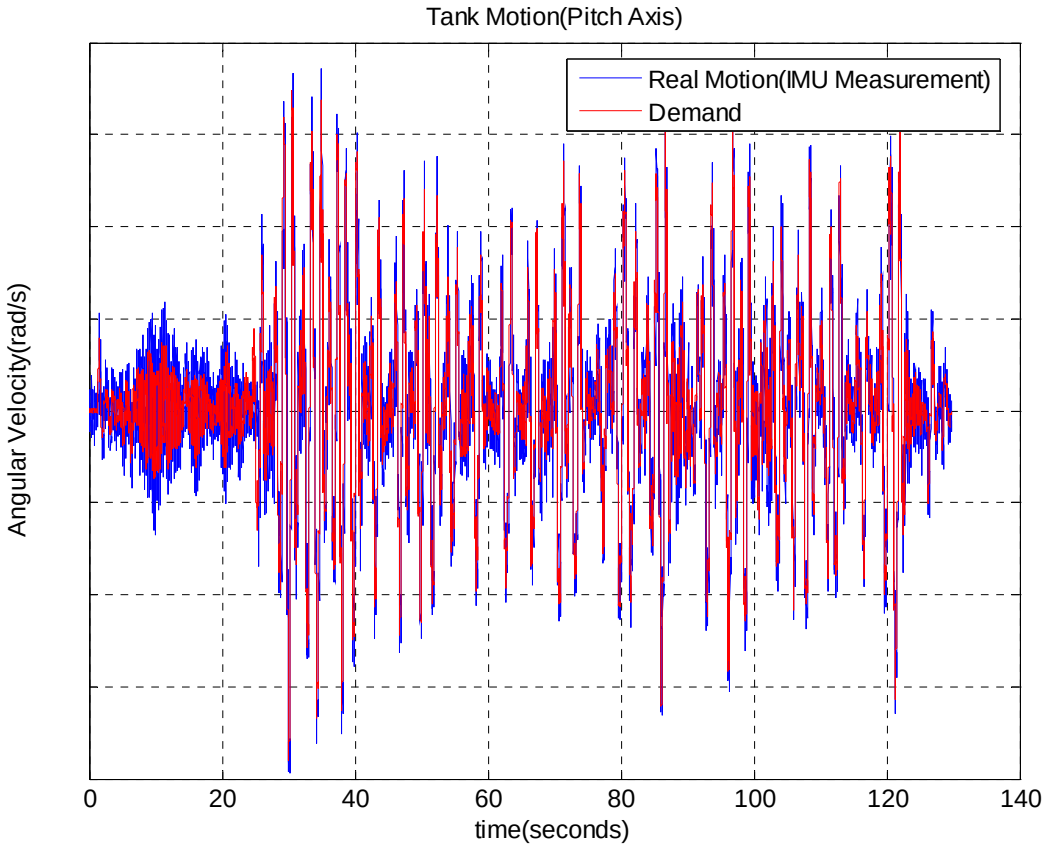


Figure 4.33: IMU measurements for tank motion in pitch direction

In the figure above, it is observed that the desired motion is amplified by the platform, preserving the frequency content of the response. Especially higher frequency demands are amplified, which can be shown by referring to Figure 4.34.

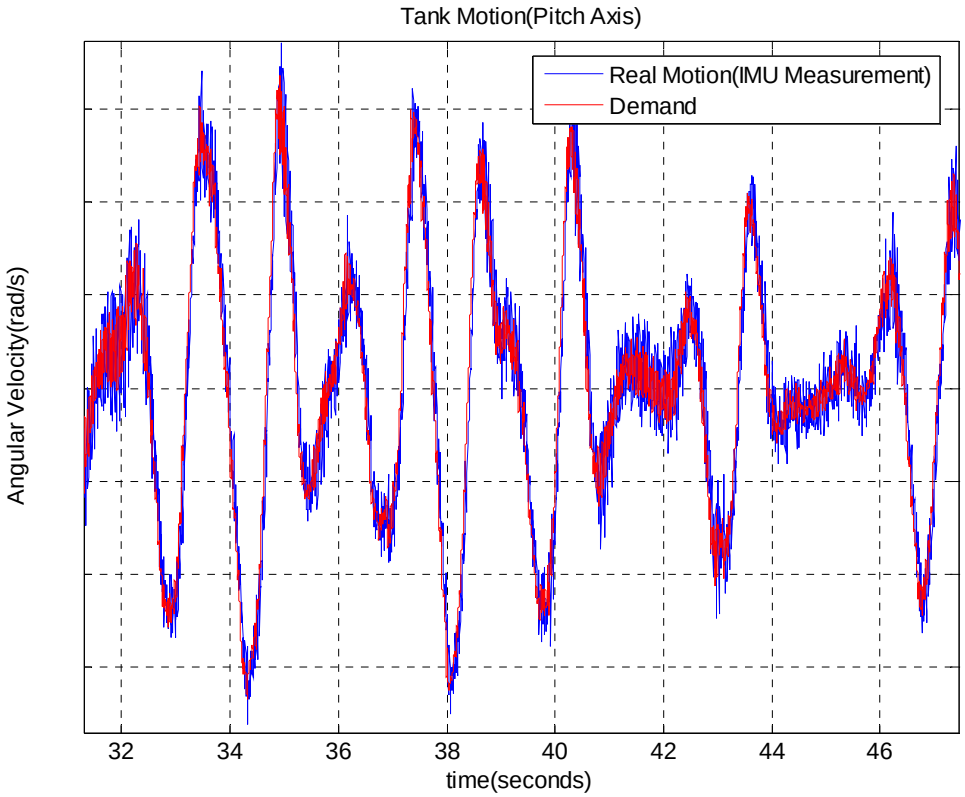


Figure 4.34: IMU measurements for tank motion in pitch axis (a closer view)

4.4 Feedforward Control Concepts

The platform can be controlled by implementing feedforward control strategies combined with feedback control. As mentioned before, drivers can work both in speed control mode or torque control mode. In torque control mode, feedforward techniques are possible since torque command is given from MATLAB®. However; in speed control mode, this is not possible since torque command will be generated by a feedback controller in PI format in drivers. On the other hand, working in torque control mode is not practical and safe in the real-time control system because of the nature of the incremental encoder data taken from the drivers. As far as absolute linear measurement is not done, two options arise in order to have torque control on the platform. One of them is to make the homing procedure by open-loop torque control; however it is not preferred since realization of homing procedure is

unsafe by open-loop control. The other alternative is to make the homing procedure by velocity control and to switch the working mode of the driver from velocity control to torque control after homing but this is time consuming and not practical. Therefore, to work in torque control mode is not preferred. Feedforward control techniques can not be implemented in real-time as far as torque control is not managed. However, in simulation, a feedforward technique is applied considering the realities of the real-system.

To work in torque control mode implies that only actuator displacement is available for feedback, so in simulations only position feedback signal is considered. Control algorithm and structure is re-designed including feedforward concepts.

4.4.1 Control Structure with Feedforward Control

Dynamics of the Stewart Platform is discussed in a different point of view in order to implement the feedforward technique. In this point of view, each leg is modeled independent of each other and as a mass driven by the electric motor where its measurement of rotation is available by incremental encoder data. This mass is the sum of the moving mass of the actuator in its actuator-axis and one-sixth of the upper platform and test equipment. The remaining dynamics (gravity, coriolis forces, centrifugal forces, friction etc.) is defined as disturbance to this lumped-mass model. The highly non-linear system model is converted to a linear model; however it's no longer a pure servo problem but a servo and a regulator problem. The non-linear terms in equations of motion are in "Disturbance Force" term.

For dynamics of one of the legs, the following equations can be written:

$$m\frac{d\dot{v}}{dt} = F_m - F_d \tag{4.1}$$

$$\frac{dx}{dt} = v \tag{4.2}$$

$$\frac{dF_d}{dt} = 0 \tag{4.3}$$

in which

v	Translational velocity of the leg
m	Equivalent mass of the leg
x	Translational distance of the leg
F_m	Actuator force
F_d	Disturbance force to actuator

It is assumed that the disturbance force is constant and its derivative is zero. This is a reasonable assumption since the sampling frequency of the controller is much higher than that of the disturbance force variation.

In state-space representation,

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned} \tag{4.4}$$

where

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -1/m & -1/m \\ 0 & 0 & 0 \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} 0 \\ 1/m \\ 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix},\ x = \begin{bmatrix} x & v & F_d \end{bmatrix}^T \tag{4.5}$$

The input variable u is the control force F_m, state variables are the position of upper_leg x, the velocity of the upper_leg v and the disturbance load force F_d; the output variable is the position of upper_leg x.

Speed v and disturbance F_d can be estimated by a Kalman filter based on the state equations represented above.

4.4.2 Speed and Disturbance Estimation Using a Kalman Filter

The Kalman filter is a set of mathematical equations that provides an efficient computational (recursive) means to estimate the state of a process, in a way that minimizes the mean of the squared error. The filter is very powerful in several aspects [13]:

It supports estimations of past, present, and even future states, and it can do so even when the precise nature of the modeled system is unknown.

State equations can be rewritten including a system noise ξ with a system noise matrix Γ and a measurement noise η into the system model as

$$\begin{aligned} \dot{x} &= Ax + Bu + \Gamma\xi \\ y &= Cx + \eta \end{aligned} \tag{4.6}$$

Both ξ and η are assumed to be zero-mean white Gaussian noise inputs.

Γ, ξ and η are written as

$$\Gamma = \begin{bmatrix} 0 & 0 \\ 1/m & 0 \\ 0 & F_m^{\max} \end{bmatrix} \tag{4.7}$$

$$\xi = \begin{bmatrix} F_m^{noise} & F_d^{noise} \end{bmatrix}, \quad \eta = x^{noise} \tag{4.8}$$

where

$F_m^{\max}$	Maximum control input
F_m^{noise}	System noise in control input
F_d^{noise}	System noise in disturbance force
x^{noise}	Measurement noise

System noises come from the non-ideal characteristics of the current controller and the actuator. The measurement noise arises from the encoder data of the motor and quantization of the leg position. Thus the variance matrix of the system noise vector Q and the variance matrix of the noise vector R are written as

$$Q = \begin{bmatrix} q_{00} & 0 \\ 0 & q_{11} \end{bmatrix}, \; R = \begin{bmatrix} r_{00} \end{bmatrix} \tag{4.9}$$

where

q_{00}	Control force covariance value
q_{11}	Disturbance force covariance value
r_{00}	Measurement noise covariance value

Design of a Kalman filter can be defined as selection of Q and R matrices. These matrices can be found experimentally, or by means of mathematical calculations as well as by trial-and error approach. Lower value selection for q_{00} and q_{11} means to thrust the model more. Similarly, lower value selection for r_{00} means thrusting the measurements more.

For digital measurements this r_{00} value must be selected lower rather analog measurements. However it should never be selected as too small since it causes some problems in Kalman filtering process and accurate estimates can not be gathered.

Actually, selection of Q matrix components is not straightforward. Non-diagonal elements indicating the coupling between system noise and modeling errors are practically coupled but it's hard to know the values of these elements. In literature, these elements are assumed as zero generally. "q_{00}" element shows the errors in the hardware in practice. Noises generated by PWM inverter and motor may cause some errors. Dynamic identification errors should also be counted as errors in the process. "q_{11}" element shows the modeling errors on disturbance. In our case, disturbance change rate is taken as zero, which is a reasonable assumption considering small sampling frequencies of controller; nevertheless it may not reflect real process since highly dynamic movements occurs in the Stewart Platform. Therefore, relatively higher q_{11} values should be chosen regarding to highly rated disturbances on the actuators.

The discrete form of Kalman filter is

$$\begin{aligned} x_{k+1} &= A_k x_k + B_k u_k + \Gamma_k \xi_k \\ y_k &= C_k x_k + \eta_k \end{aligned} \tag{4.10}$$

A_k, B_k, C_k, Γ_k are discrete forms of system matrices [12]. These discrete matrices can be found approximately by using the formulas shown below [16].

$$\begin{aligned} A_k &= I + A.(I.T + \frac{A.T^2}{2!} + \frac{A^2.T^3}{3!} + ... + \frac{A^{n-1}.T^n}{n!} + ...) \\ S &= (I.T + \frac{A.T^2}{2!} + \frac{A^2.T^3}{3!} + ... + \frac{A^{n-1}.T^n}{n!} + ...) \\ A_k &= I + A.S \\ B_k &= S.B \\ C_k &= C \\ \Gamma_k &= S.\Gamma \end{aligned} \tag{4.11}$$

where T stands for the sampling time. The larger order "n" results in more accurate results for the matrices as expected. Specifying "n" value as 1, the matrices are constituted by using the equations set 4.13 and shown below.

$$A_k = \begin{bmatrix} 1 & T & 0 \\ 0 & 1-T/m & -T/m \\ 0 & 0 & 1 \end{bmatrix}$$

$$B_k = \begin{bmatrix} 0 & T/m & 0 \end{bmatrix}^T$$

$$C_k = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

$$\Gamma_k = \begin{bmatrix} 0 & 0 \\ T/m & 0 \\ 0 & T.F_m^{\max} \end{bmatrix}$$

Kalman filter process is given by the equation set as shown in below:

$$\begin{aligned} P_{0,0} &= Var(x_0) \\ P_{k,k-1} &= A_{k-1} P_{k-1,k-1} A_{k-1}^T + \Gamma_{k-1} Q_{k-1} \Gamma_{k-1}^T \\ G_k &= P_{k,k-1} C_k^T (C_k P_{k,k-1} C_k^T + R_k)^{-1} \\ P_{k,k} &= (I - G_k C_k) P_{k,k-1} \\ \hat{x}_{0,0} &= E(x_0) \\ \hat{x}_{k,k-1} &= A_{k-1} \hat{x}_{k-1,k-1} + B_{k-1} u_{k-1} \\ \hat{x}_{k,k} &= \hat{x}_{k,k-1} + G_k (y_k - C_k \hat{x}_{k,k-1}) \end{aligned} \qquad [12] \qquad (4.12)$$

and

$P_{k,l}$	Estimated variance matrix;
$Var(x)$	Variance of the random variable x;
$E(x)$	Expectation of the random variable x;
G_k	Kalman gain matrix;
Q_k	Variance matrix of the random vector ξ_k

R_k Variance matrix of the random vector η_k

u_{k-1} Control input in the previous sampling period

Figure 4.35 shows the controller structure including feedforward control techniques. Speed estimate is used in the velocity controller and force disturbance estimation is used as a feedforward torque. In ideal case, disturbance observer cancels disturbance force.

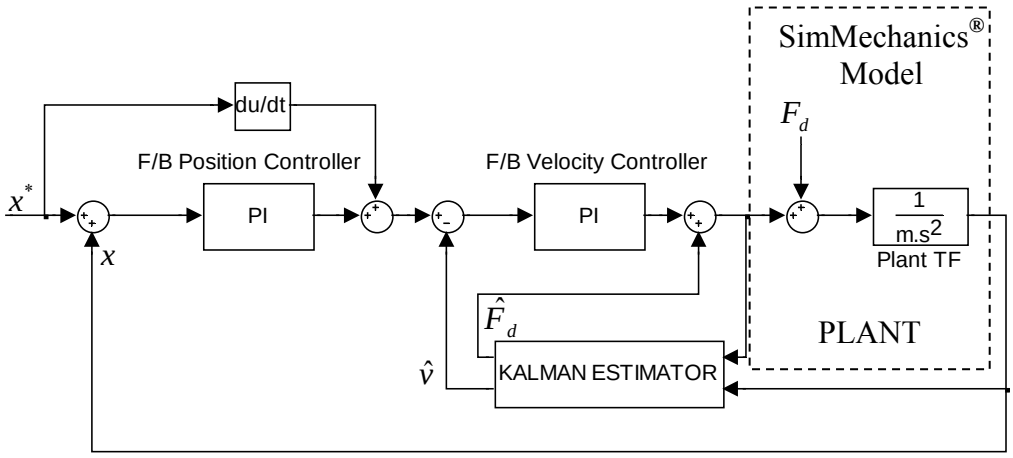


Figure 4.35: Control structure with the estimator

In the simulations, the forward dynamic model in SimMechanics® is used as the plant model. Position and velocity controller structure and parameters are the same as in the previously designed feedback controllers. "Kalman function" of MATLAB® is used for the estimator. An m-file, showing inputs and functions for this filter, is given in Appendix A.3.

"q_{00}" element is chosen as zero; since there is no non-ideal current controller and everything is clearly identified in simulation. On the other hand, higher values should be selected for "q_{00}" element in real system since dynamic identification errors will occur and non-ideal torque and motor characteristics will account. A large number is selected for "q_{11}" because of assumption of constant disturbance force on

the legs. "r_{00}" element is selected as very small because there is no noise in simulation. Small values should be selected for "r_{00}" in real system also, since encoder is a digital sensor showing noiseless characteristics. Therefore, Q and R matrices used in simulation are constituted as:

$$Q = \begin{bmatrix} 0 & 0 \\ 0 & 10000000 \end{bmatrix} \quad , \quad R = \begin{bmatrix} 0.0001 \end{bmatrix}$$

Updated Simulink® model with the utilization of feedforward control is shown in Figure 4.36. One of the differences from the previous control models is the "Velocity and Disturbance Estimator" subsystem in which velocity and disturbance forces are estimated. Another difference is that position data measured at actuators is discretized by a "Zero order hold" block; since encoder data is available in real system. This is important to see the performance of the Kalman filter accurately because derivative operation of discrete signals is not straightforward to achieve. Inner view of the "Velocity and Disturbance Estimator" subsystem is shown in Figure 4.37. Here, an LTI (Linear time invariant) system called "Observer" is used for each of the legs.

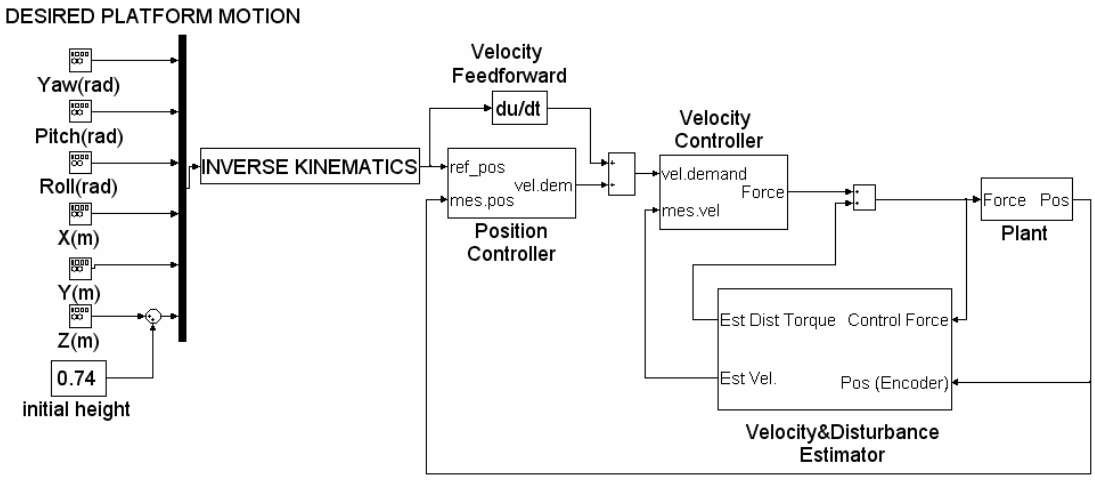


Figure 4.36: Simulink® model with feedforward control

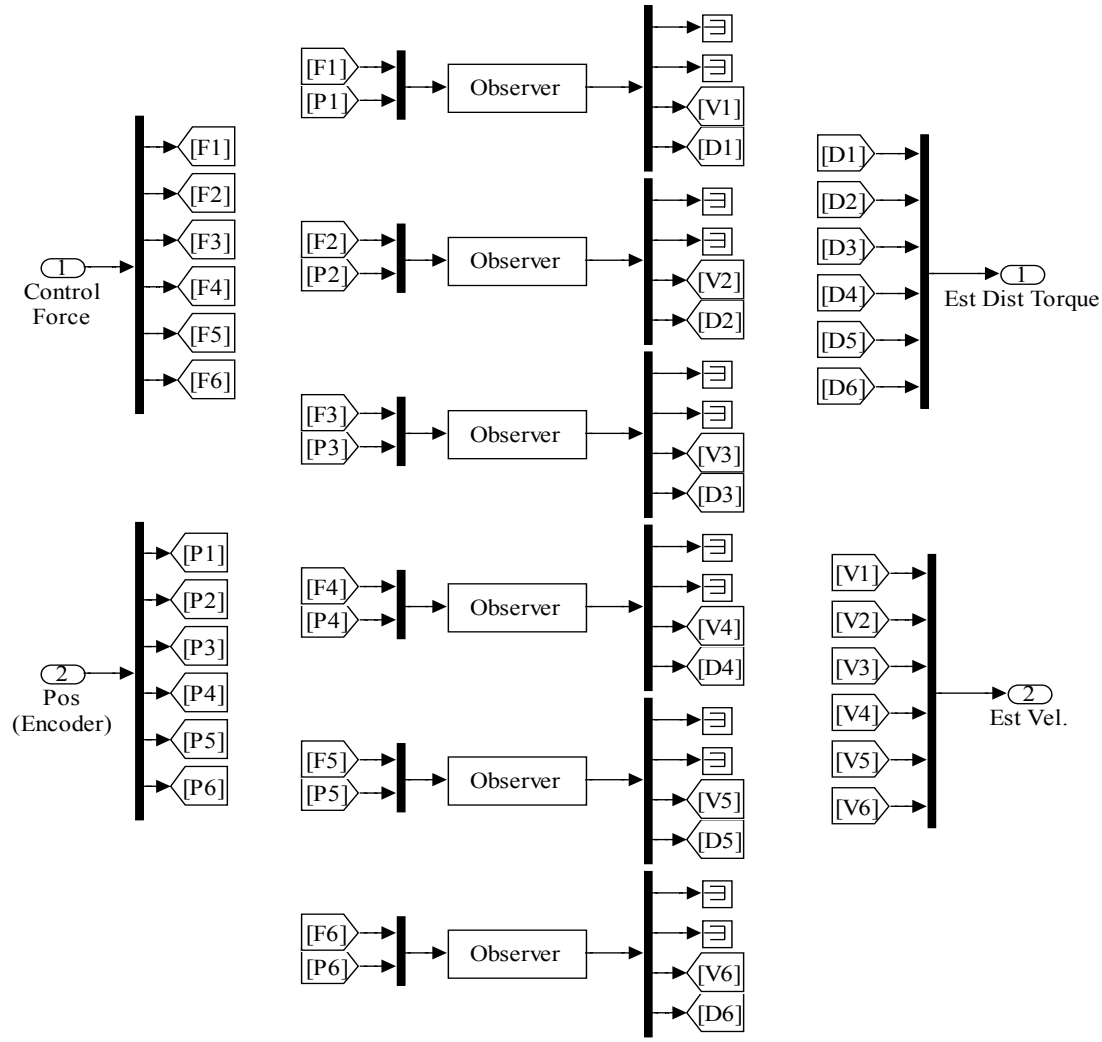


Figure 4.37: Velocity and disturbance estimator subsystem

For a tank motion input in pitch axis, desired platform motion and measured platform motions are plotted in Figure 4.38. As seen in Figure 4.38, platform almost follows the desired motion. In Figure 4.39, platform tracking at high frequencies is shown and platform tracking error is shown in Figure 4.40.

Estimated velocity by the Kalman filter is compared with the one that SimMechanics® gives as output. Actual and estimated velocity is shown in Figure 4.41 and estimation of higher frequency velocity values is seen on Figure 4.42. The error plot for the velocity estimation is also shown in Figure 4.43. Estimated disturbance force on the actuators is shown in Figure 4.44 for all the six actuators.

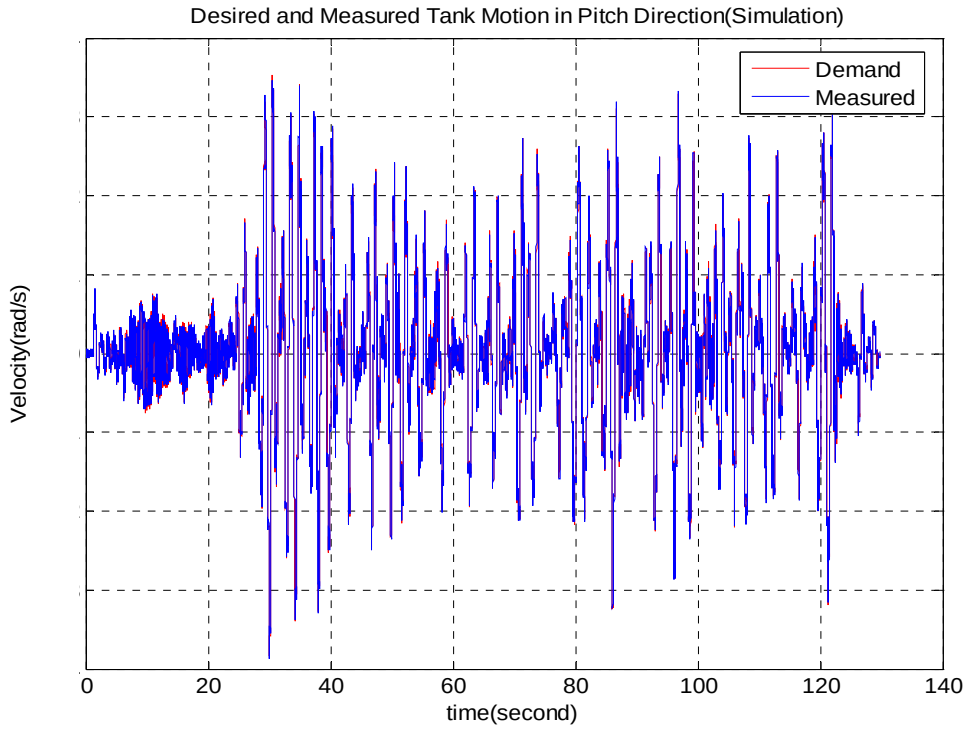


Figure 4.38: Desired and measured platform motion for tank motion in pitch axis

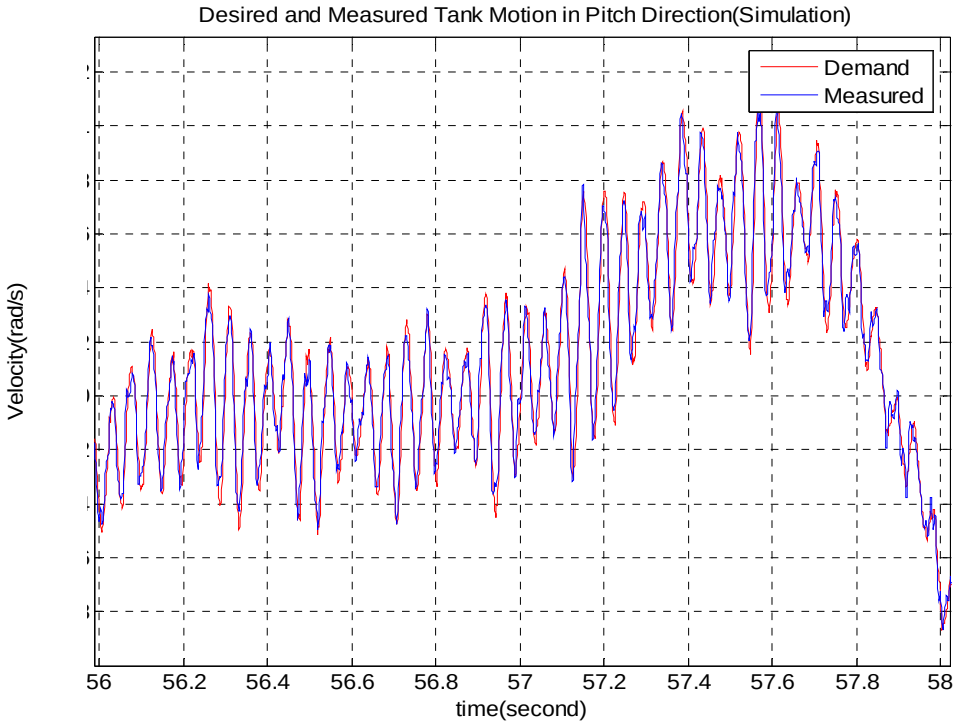


Figure 4.39: Desired and measured platform motion for tank motion in pitch axis (A closer view)

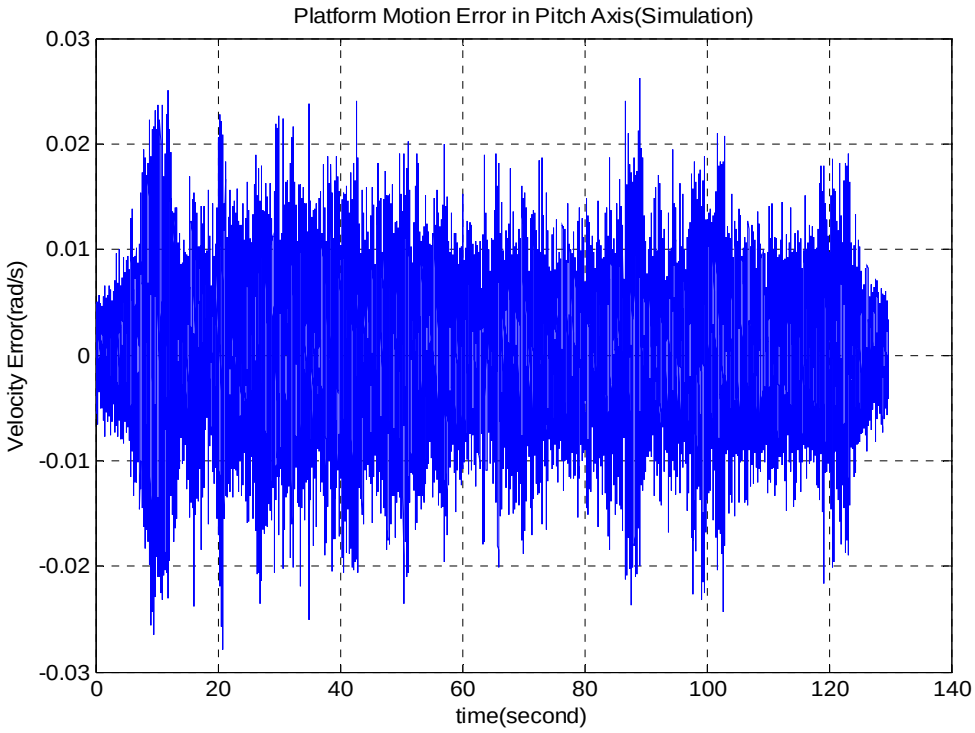


Figure 4.40: Platform motion error in pitch axis (simulation)

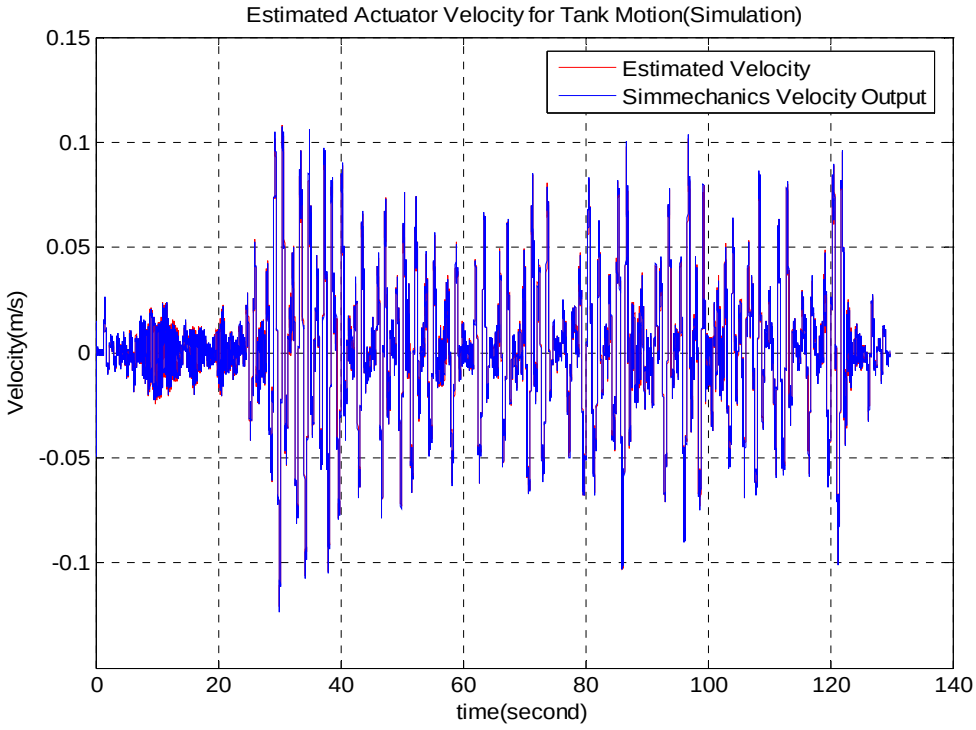


Figure 4.41: Actual and estimated actuator velocities for tank motion in pitch axis

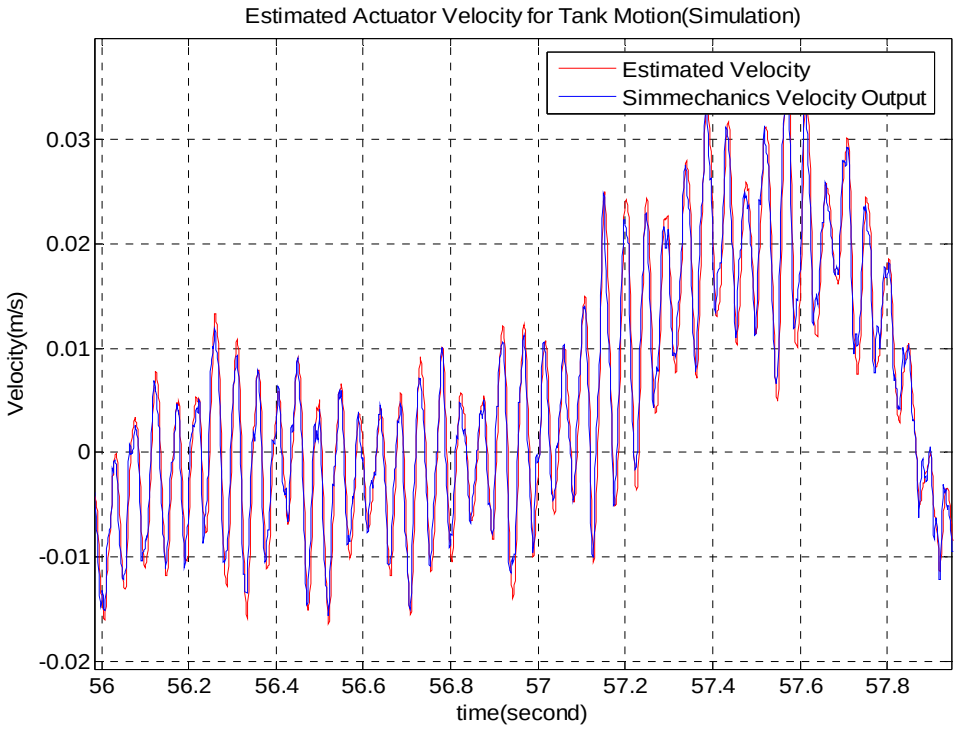


Figure 4.42: Actual and estimated actuator velocities for tank motion in pitch axis (A closer view)

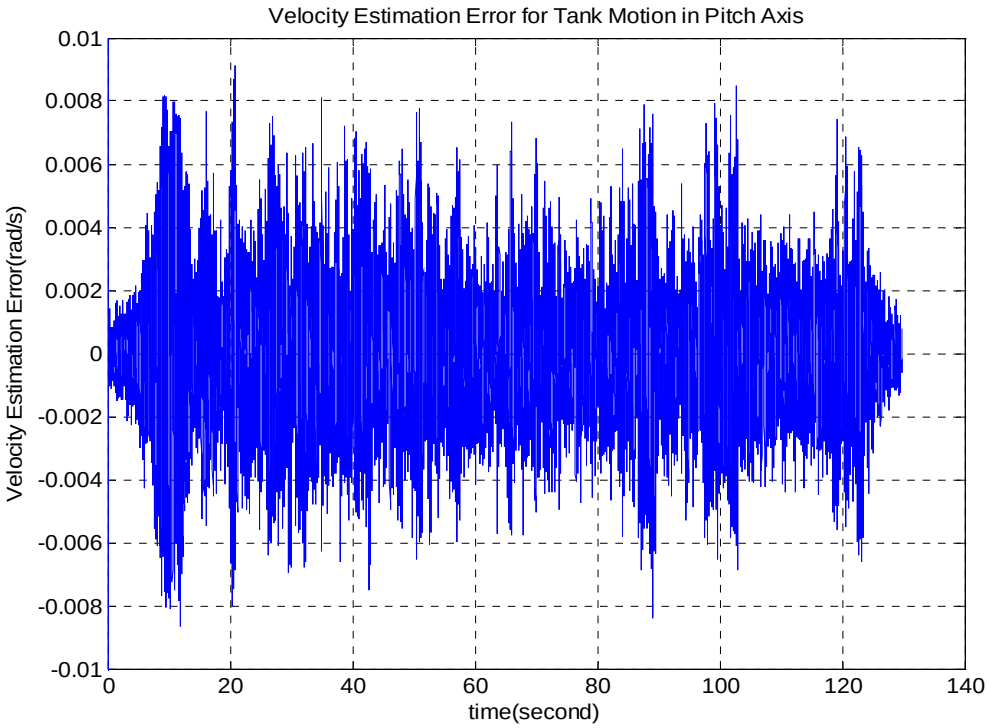


Figure 4.43: Velocity estimation error

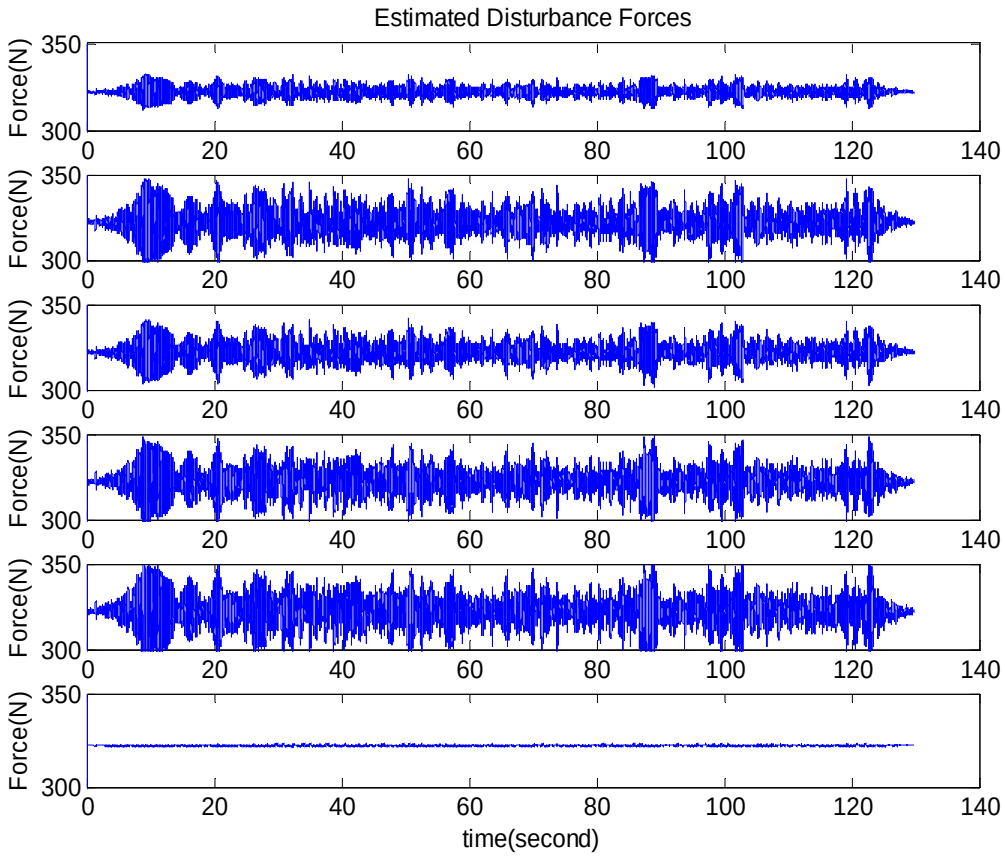


Figure 4.44: Estimated disturbance forces for all of the actuators

There is not a distinct progress in the platform motion accuracy by using the feedforward controller with the feedback controller, in comparison to using only the feedback controller. This is so, since some of the non-linear effects on the real system such as friction, flexibility of the mechanical parts etc. are not considered in the simulation. However, in the real-time control of the system, the feedforward controller can overcome these non-linear effects and improve the platform motion accuracy.

CHAPTER 5

DISCUSSION AND CONCLUSIONS

5.1 Summary and Conclusions

This thesis is focused on mathematical modeling and real-time control system design and implementation of a Stewart Platform. Aselsan Inc. requires a six degree-of freedom motion simulator which should simulate dynamic motions of the tank on APG track in terms of rotational velocities and linear accelerations. In order to fulfill this requirement, a Stewart Platform is designed and constructed. Mathematical modeling, control system design and implementation are the important steps of this work; and they are discussed in detail throughout the study.

Mathematical modeling is mainly done by two software; MATLAB® / Simulink® / SimMechanics® and ADAMS®. Accuracy of Simulink® / SimMechanics® model is important because control system design is accomplished by using the Simulink® / SimMechanics® model. ADAMS® model is used for verification of the Simulink® / SimMechanics® model. In mathematical modeling, only distributed mass model is used for the parts of the system; friction at the joints and stiffness of the parts are neglected.

Control system design takes place after mathematical modeling of the system. It is achieved by using the derived mathematical models in Simulink® / SimMechanics® and realized by components such as linear actuators, drivers, DAQ hardware, and MATLAB® software. Controller structure used in the Stewart Platform is composed of two PI controllers as position and velocity controllers with velocity feedforward. Velocity control is done by motor drivers whereas position control is accomplished by MATLAB®. Velocity controller parameters are the most critical ones since

rotational velocities and translational accelerations of the tank should be simulated. In order to find the optimum parameters; one of the MATLAB® modules, Simulink Response Optimization®, is used as a tool. Position controller parameters which are not as important as velocity controller are found by trial and error approach.

Although drivers have the ability to accept torque command from outside, this is not preferred because of incremental encoder data usage as position feedback signals. In homing process, linear actuators should be driven by velocity loop since it is not safe to make the homing procedure by open-loop torque control for this Stewart Platform. After homing process, working mode of the drivers should be changed from velocity control mode to torque control model if torque control is desired to be employed. On the other hand, this is very time-consuming and not practical for the operation of the Stewart Platform. As far as torque control is not preferable by this setup, friction and stiffness models' importance declines as they can not be accounted for in the control system. Therefore, stiffness and friction model development studies are discarded.

Directly the tank motion or a combination of inputs having various frequency contents can be applied as disturbances to the test equipment by the Stewart Platform. For inputs with high frequency, platform motion accuracy decreases as the platform amplifies the motion input, however this is not critical as far as the amplification ratio is known. Once this ratio is known in advance, the motion demand can be decreased at the same ratio to compensate for the difference between the demanded and the measured platform motions.

In stabilized head mirror tests, this Stewart Platform has been widely used not only in control system design of the head mirror but also structural dynamics testing of it. The Stewart Platform enables the design engineers to see the shortcomings of their design and observe the problems in their prototypes in advance and in laboratory conditions. In this aspect, the efficient utilization of the Stewart Platform decreases development period of the head mirror and reduces the associated development cost significantly.

5.2 Future Scope

In electric motors, resolvers which are located at the back of the motors are used as default sensors for proper operation of the electric motors. RDC modules integrated to digital circuitry of drivers convert resolver data into 12-bit incremental encoder data. They also achieve derivative operation of incremental encoder data and obtain velocities of the electric motors. As far as there is no backlash between the electric motor and the roller-screw mechanism and in joints, and as far as the parts are rigid; utilization of incremental encoder data is not critical. However, there is 0.1 mm backlash between the electric motors and the roller-screws and flexibility of the parts shows their effect on the upper platform at high frequencies. These effects can not be compensated by incremental encoder data completely because incremental encoder data shows the motion of the electric motor. Therefore, there will be some error in platform motion as far as incremental encoder data are used. As the frequency increases, this error will be more apparent. Utilization of resolvers as position feedback devices was a cheap solution for this system; but as mentioned before, platform motion accuracy decreases as the frequency of the motion input increases. At this point as a future work, absolute linear encoders can be adapted to linear actuators in order to increase the platform motion accuracy. Stiffeners enable absolute linear encoders to be mounted on them and to be used as position feedback devices. In this case, actual positions of the linear actuators can be measured by considering all the effects such as backlashes, flexibility of the parts etc. Although it is a costly solution, higher platform motion accuracy can be satisfied by using absolute linear encoders.

If absolute linear encoders are used in this system, torque control is enabled. Torque control enables various control algorithms to be used in the Stewart Platform. The suggested feedforward control technique discussed in Chapter 4 can be implemented in real-time. This method would compensate for all the non-linear forces including friction forces and forces generated from stiffness of the parts. As long as torque control is enabled, friction and stiffness of the parts can be added to the

mathematical models. Thus, various compensation techniques can be developed on these models and then they can be implemented to real-time control system.

Finally, a graphical user interface (GUI) can be developed for Stewart Platform controlling operations such as motion demanding, controller setup etc. At present, these operations are done in the model in Simulink® but use of a suitable GUI can make the real-time control operations more practical and manageable.

REFERENCES

[1] Lee, W.; Kim, J.; Cho, J., "A Driving Simulator as a Virtual Reality Tool", Kookmin University

[2] Heger, M. R., Wiens, G. J.; "Dynamic System Identification of Parallel Kinematic Machines", University of Florida

[3] Söylemez, E., (1999), "Mechanisms", Middle East Techical University

[4] Özgören, K., (2002) "Advanced Dynamics Lecture Notes", Middle East Techical University, (Unpublished)

[5] MSC.Software Corporation, (2003), "MSC.ADAMS® Basic Full Simulation Package Training Guide", Version 1.0

[6] The Mathworks Inc., (2005), "SimMechanics, User's Guide", Version 2

[7] Exlar Inc., (2005), "Exlar Product Catalog", URL: http://www.exlar.com, Last Update, 12.08.2005

[8] Metronix Inc. (1997), "Product Manual, ARS-310/xx", Version 1.0e, 30.06.1997

[9] Ogata, K., (1997), "Modern Control Engineering", Prentice Hall Inc., 3rd ed.

[10] The Mathworks Inc., (2005), "Simulink Response Optimization, User's Guide", Version 2

[11] BAE Systems Inc., (2003), "Product Datasheet for SiIMU01", Issue 3, 20.01.03

[12] Kim, H., Sul, S., (1996), "A New Motor Speed Filter in Low-Speed Range", IEEE Transactions on Industrial Electronics, Vol. 43, No.4

[13] Humusoft Inc, (2002), "MF 614, Multifunction I/O Card, User's Manual"

[14] Özdağlar, M., (2001), "Dynamic Modeling and Control of a Stewart Platform Type Motion Simulator", M.S. Thesis, METU Library

[15] Bostelman, R., Albus, J., Dalagalis, N., Jacoff, A., Gross, J., "Application of the NIST ROBOCRANE", National Institute of Standards and Technology (NIST)

[16] Haugen, F., (2005), "Discrete-time signals and systems", URL: http://techteach.no

APPENDIX A

MATLAB CODES

A.1 Code for Kinematic and Dynamic Identification of the Stewart Platform

```
deg2rad = pi/180;
x_axis = [1 0 0];
y_axis = [0 1 0];
z_axis = [0 0 1];
pos_base = [];
pos_top = [];
alpha_b = 5.275*deg2rad;
alpha_t = 9.642*deg2rad;
height = 0.740;
radius_b = 0.6304;
radius_t = 0.430;
for i = 1:3,
  angle_m_b = (2*pi/3)* (i-1) - alpha_b-20*deg2rad;
  angle_p_b = (2*pi/3)* (i-1) + alpha_b-20*deg2rad;
  pos_base(2*i-1,:) = radius_b* [cos(angle_m_b), sin(angle_m_b), 0.0];
  pos_base(2*i,:) = radius_b* [cos(angle_p_b), sin(angle_p_b), 0.0];
  angle_m_t = (2*pi/3)* (i-1) - alpha_t + 2*pi/6-20*deg2rad;
  angle_p_t = (2*pi/3)* (i-1) + alpha_t + 2*pi/6-20*deg2rad;
  pos_top(2*i-1,:) = [radius_t*cos(angle_m_t),radius_t*sin(angle_m_t), height];
  pos_top(2*i,:) =  [radius_t*cos(angle_p_t),radius_t*sin(angle_p_t), height];
end
pos_top = [pos_top(6,:); pos_top(1:5,:)];
```

```
body_pts = pos_top' - height*[zeros(2,6);ones(1,6)];
legs = pos_top - pos_base;
leg_length = [ ];
leg_vectors = [ ];
for i = 1:6,
  leg_length(i) = norm(legs(i,:));
  leg_vectors(i,:)  = legs(i,:) / leg_length(i);
end
lower_leg = struct('origin', [0 0 0], 'rotation', eye(3), 'end_point', [0 0 0]);
upper_leg = struct('origin', [0 0 0], 'rotation', eye(3), 'end_point', [0 0 0]);
for i = 1:6,
  rev1(i,:) = cross(leg_vectors(i,:), z_axis);
  rev1(i,:) = rev1(i,:) / norm(rev1(i,:));
  rev2(i,:) = - cross(rev1(i,:), leg_vectors(i,:));
  rev2(i,:) = rev2(i,:) / norm(rev2(i,:));
  cyl1(i,:) = leg_vectors(i,:);
  rev3(i,:) = rev1(i,:);
  rev4(i,:) = rev2(i,:);
end
for i = 1:2:5,
  lower_leg(i).origin = pos_base(i,:) + (352.67+41.3)/1000*cyl1(i,:)-
  9.81/1000*rev1(i,:)-9.96/1000*rev2(i,:);
  lower_leg(i).end_point = pos_base(i,:) + (719.5+41.3)/1000*cyl1(i,:);
  lower_leg(i).rotation = [rev1(i,:)', rev2(i,:)', cyl1(i,:)'];
   upper_leg(i).origin = pos_base(i,:) + 430/1000*cyl1(i,:)-55/1000*rev1(i,:)-
   55/1000*rev2(i,:);
  upper_leg(i).end_point = pos_base(i,:) + (96.75+41.3)/1000*cyl1(i,:);
  upper_leg(i).rotation = [rev1(i,:)', rev2(i,:)', cyl1(i,:)'];
end
for i = 2:2:6,
  lower_leg(i).origin=pos_base(i,:)+
(352.67+41.3)/1000*cyl1(i,:)+9.81/1000*rev1(i,:)-9.96/1000*rev2(i,:);
```

```
 lower_leg(i).end_point = pos_base(i,:) + (719.5+41.3)/1000*cyl1(i,:);
 lower_leg(i).rotation = [rev1(i,:)', rev2(i,:)', cyl1(i,:)'];
 upper_leg(i).origin=pos_base(i,:)+430/1000*cyl1(i,:)+55/1000*rev1(i,:)-
55/1000*rev2(i,:);
 upper_leg(i).end_point = pos_base(i,:) + (96.75+41.3)/1000*cyl1(i,:);
 upper_leg(i).rotation = [rev1(i,:)', rev2(i,:)', cyl1(i,:)'];
end
 lower_leg_mass= 18.43;
 lower_leg_inertia=[88.875 0.0642 0.934;0.0642 88.56 0.901;0.934 0.901 2.97]/100;
 lower_leg_inertia1=[88.56 0.0642 0.901;0.0642 88.875 0.934;0.934 0.901
2.97]/100;
 upper_leg_mass=11.7;
 upper_leg_inertia=[40.2 0.088 -0.448;0.088 40.2 -0.448;-0.448 -0.448 3.24]/100;
 upper_leg_inertia1=[40.2 -0.088 0.448;-0.088 40.2 -0.448;0.448 -0.448 3.24]/100;
 top_mass=97;
 top_inertia=[1.05 -3.4e-3 4.54e-4;-3.37e-3 1.05 1.34e-2;4.54e-4 1.34e-2 1.95];
```

A.2 Code for Inverse Kinematics

```
function [sys,x0,str,ts]=inverse_kinematics(t,x,u,flag)
switch flag,
   case 0
      [sys,x0,str,ts]=mdlInitializeSizes;
   case 3
      sys=mdlOutputs(t,x,u);
   case { 1, 2, 4, 9 }
      sys=[];
   otherwise
      error(['Unhandled flag=',num2str(flag)]);
end;
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
```

```
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 6;  % dynamically sized
sizes.NumInputs      = -1;  % dynamically sized
sizes.DirFeedthrough = 1;   % has direct feedthrough
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
str = [];
x0  = [];
ts  = [-1 0];   % inherited sample time
function sys = mdlOutputs(t,x,u)
body_pts=[0.1445    0.3710    0.2785   -0.3737   -0.4230    0.0027
  -0.4050    0.2173    0.3277    0.2127    0.0773   -0.4300
       0         0         0         0         0         0];
leg_length=[0.8644    0.8644    0.8644    0.8644    0.8644    0.8644]';
pos_base= [ 0.5701    0.6097   -0.0519   -0.1661   -0.5181   -0.4436
  -0.2692   -0.1602    0.6283    0.6081   -0.3591   -0.4479
       0         0         0         0         0         0];
ksi=u(1);teta=u(2);phi=u(3);
cp1=u(4);cp2=u(5);cp3=u(6);
% Transformation Matrix %
c11=cos(teta)*cos(ksi);
c12=cos(teta)*sin(ksi);
c13=-sin(teta);
c21=sin(teta)*sin(phi)*cos(ksi)-cos(phi)*sin(ksi);
c22=sin(teta)*sin(phi)*sin(ksi)+cos(phi)*cos(ksi);
c23=sin(phi)*cos(teta);
c31=sin(teta)*cos(phi)*cos(ksi)+sin(phi)*sin(ksi);
c32=sin(teta)*cos(phi)*sin(ksi)-sin(phi)*cos(ksi);
c33=cos(phi)*cos(teta);
Cba=[c11 c12 c13;c21 c22 c23;c31 c32 c33]; % Transformation Matrix
cp=[cp1 cp2 cp3]'; % position of the upper platform wrt base
```

```
Cp_Matrix=[cp cp cp cp cp cp]; % for six legs
legs=(Cba*body_pts)+Cp_Matrix-pos_base;
for i = 1:6;
leg_change(i)=norm(legs(:,i))-leg_length(i);
end
sys=[leg_change(1)  leg_change(2)  leg_change(3)  leg_change(4)  leg_change(5)
leg_change(6)]';
```

A.3 Code for the Kalman Filter Used as Speed and Disturbance Observer

```
T = 0.001;
m=18.8;
A=[0 1 0;0 0 -1/m;0 0 0];
eig(A)
B=[0 1/m*0.001 0]';
C=[1000 0 0];
D=[0];
Da=[0;0;1000]*T;
plant = ss(A, B, C, D);
discrete_plant= c2d(plant,T,'tustin');
[Phi, Gamma, Cd, Dd] = ssdata(discrete_plant);
Rv = 0.00001;
Rw = [100000000];
sensors = [1];
known = [1];
P = ss(Phi,[Gamma Da], Cd, [Dd 0],T);
[Observer, Ko] = kalman(P, Rw, Rv, [], sensors, known);
```

A.4 Code for IMU Analysis

```
function data=imu(sp,apg,X_acc,Y_acc,Z_acc);
close all;
```

```
load imuData.txt;
[m,n] = size(imuData);
count = imuData(:,1);
vR    = imuData(sp:m-1,5);
t=0:1/200:(m-1-sp)/200;
dr=(vR-mean(vR));
Figure,plot(t,dr);grid on;
```

A.5 Code for Finding the Maximum Payload that the Stewart Platform Can Handle Statically

```
top_mass_unit=97/97;
top_inertia_unit=[1.05 -3.4e-3 4.54e-4;-3.37e-3 1.05 1.34e-2;4.54e-4 1.34e-2
1.95]/97;
top_mass=top_mass_unit*600;
top_inertia=top_inertia_unit*600;
max_force=0;
f=3;
while max_force < 3073
 for i=1:f+1;
  for j=1:f+1;
   for k=1:f+1;
    for a=1:f+1;
     for b=1:f+1;
      for c=1:f+1;
       ll1 = 0.23/f*(i-1);
       ll2 = 0.23/f*(j-1);
       ll3 = 0.23/f*(k-1);
       ll4 = 0.23/f*(a-1);
       ll5 = 0.23/f*(b-1);
       ll6 = 0.23/f*(c-1);
       sim('stewart_inverse_dynamics_static');
```

```
      load forces.mat;
      [m,n] = size(data);
      req_forces = data(2:7,:);
      max_force = max(max(req_forces));
      if max_force > 3073
       return
      end
     end
    end
   end
  end
 end
end
 top_mass = top_mass+50;
 top_inertia=top_inertia+top_inertia_unit*50;
end
```

A.6 Code for Finding the Maximum Payload Satisfying the Specified Platform Motion

```
top_mass_unit=97/97;
top_inertia_unit=[1.05 -3.4e-3 4.54e-4;-3.37e-3 1.05 1.34e-2;4.54e-4 1.34e-2
1.95]/97;
top_mass=top_mass_unit*250;
top_inertia=top_inertia_unit*250;
max_force=0;
while max_force < 3073
inc=0.05;
 for ll1=0.05:inc:0.23;
  for ll2=0.05:inc:0.23;
   for ll3=0.05:inc:0.23;
    for ll4=0.05:inc:0.23;
```

```
    for ll5=0.05:inc:0.23;
     for ll6=0.05:inc:0.23;
      for f=2:10:69;
       sim('stewart_inverse_dynamics_for_dynamic_analysis');
       load poses_din.mat;
       poses=poses_din(2:7,:);
       if min(poses)>0 & max(poses)<0.23
        load vel_din.mat
        vels=vel_din(2:7,:);
        if max(max(abs(vels)))<0.254
         load forces_din.mat;
         [m,n] = size(forces_din);
         req_forces = forces_din(2:7,30:250);
         max_force = max(max(req_forces));
         if max_force > 3073
          return
         end
        end
       end
      end
     end
    end
   end
  end
 top_mass = top_mass+50;
 top_inertia=top_inertia+top_inertia_unit*50;
end
```

APPENDIX B

HARDWARE SPECIFICATIONS

B.1 Linear Actuator Specifications

GSX30 & GS30 Performance Specifications

Model	Frame Size in (mm)	Stroke in (mm)	Screw Lead in (mm)	Force* Rating lb (N) 1/2/3 stack	Max Velocity in/sec (mm/sec)	Continuous Motor Torque lb-in (N-m)	Maximum Static Load lb (N)	Armature Inertia lb-in-s² (Kg-m²)	Dynamic Load Rating lb (N)	Weight (approx.) lb (Kg)
GSX30-0302	3.125 (79)	3 (76)	0.2 (5.08)	415/674/NA (1846/2998/NA)	10 (254)	16.5/26.8/NA (1.86/3.03/NA)	2000 (8896)	0.00319 (0.00036)	5800 (25798)	9.5 (4.3)
GSX30-0305	3.125 (79)	3 (76)	0.5 (12.7)	166/269/NA (738/1197/NA)	25 (635)	16.5/26.8/NA (1.86/3.03/NA)	2000 (8896)	0.00319 (0.00036)	4900 (21795)	9.5 (4.3)
GSX30-0602	3.125 (79)	5.9 (150)	0.2 (5.08)	415/674/905 (1846/2998/4026)	10 (254)	16.5/26.8/36 (1.86/3.03/4.07)	2000 (8896)	0.00361 (0.000408)	5800 (25798)	11.5 (5.2)
GSX30-0605	3.125 (79)	5.9 (150)	0.5 (12.7)	166/269/362 (738/1197/1610)	25 (635)	16.5/26.8/36 (1.86/3.03/4.07)	2000 (8896)	0.00361 (0.000408)	4900 (21795)	11.5 (5.2)
GSX30-1002	3.125 (79)	10 (254)	0.2 (5.08)	415/674/905 (1846/2998/4026)	10 (254)	16.5/26.8/36 (1.86/3.03/4.07)	2000 (8896)	0.00416 (0.00047)	5800 (25798)	19 (8.6)
GSX30-1005	3.125 (79)	10 (254)	0.5 (12.7)	166/269/362 (738/1197/1610)	25 (635)	16.5/26.8/36 (1.86/3.03/4.07)	2000 (8896)	0.00416 (0.00047)	4900 (21795)	19 (8.6)
GSX30-1402	3.125 (79)	14 (356)	0.2 (5.08)	415/674/905 (1846/2998/4026)	10 (254)	16.5/26.8/36 (1.86/3.03/4.07)	2000 (8896)	0.00473 (0.000534)	5800 (25798)	22 (10)
GSX30-1405	3.125 (79)	14 (356)	0.5 (12.7)	166/269/362 (738/1197/1610)	25 (635)	16.5/26.8/36 (1.86/3.03/4.07)	2000 (8896)	0.00473 (0.000534)	4900 (21795)	22 (10)
GSX30-1802	3.125 (79)	18 (457)	0.2 (5.08)	415/674/905 (1846/2998/4026)	10 (254)	16.5/26.8/36 (1.86/3.03/4.07)	2000 (8896)	0.00533 (0.000602)	5800 (25798)	25 (11.3)
GSX30-1805	3.125 (79)	18 (457)	0.5 (12.7)	166/269/362 (738/1197/1610)	25 (635)	16.5/26.8/36 (1.86/3.03/4.07)	2000 (8896)	0.00533 (0.000602)	4900 (21795)	25 (11.3)
GS30-0302	3.125 (79)	3 (76)	0.2 (5.08)	503 (2237)	10 (254)	20 (2.26)	2000 (8896)	0.00319 (0.00036)	5800 (25798)	9.5 (4.3)
GS30-0305	3.125 (79)	3 (76)	0.5 (12.7)	201 (894)	25 (635)	20 (2.26)	2000 (8896)	0.00319 (0.00036)	4900 (21795)	9.5 (4.3)
GS30-0602	3.125) (79)	5.9 (150)	0.2 (5.08)	503 (2237)	10 (254)	20 (2.26)	2000 (8896)	0.00361 (0.000408)	5800 (25798)	11.5 (5.2)
GS30-0605	3.125 (79)	5.9 (150)	0.5 (12.7)	201 (894)	25 (635)	20 (2.26)	2000 (8896)	0.00361 (0.000408)	4900 (21795)	11.5 (5.2)
GS30-1002	3.125 (79)	10 (254)	0.2 (5.08)	503 (2237)	10 (254)	20 (2.26)	2000 (8896)	0.00416 (0.00047)	5800 (25798)	19 (8.6)
GS30-1005	3.125 (79)	10 (254)	0.5 (12.7)	201 (894)	25 (635)	20 (2.26)	2000 (8896)	0.00416 (0.00047)	4900 (21795)	19 (8.6)
GS30-1402	3.125 (79)	14 (356)	0.2 (5.08)	503 (2237)	10 (254)	20 (2.26)	2000 (8896)	0.00473 (0.000534)	5800 (25798)	22 (10)
GS30-1405	3.125 (79)	14 (356)	0.5 (12.7)	201 (894)	25 (635)	20 (2.26)	2000 (8896)	0.00473 (0.000534)	4900 (21795)	22 (10)
GS30-1802	3.125 (79)	18 (457)	0.2 (5.08)	503 (2237)	10 (254)	20 (2.26)	2000 (8896)	0.00533 (0.000602)	5800 (25798)	25 (11.3)
GS30-1805	3.125 (79)	18 (457)	0.5 (12.7)	201 (894)	25 (635)	20 (2.26)	2000 (8896)	0.00533 (0.000602)	4900 (21795)	25 (11.3)

Figure B.1: Performance specifications of the linear actuators

GSX30 & GS30 Mechanical and Electrical Specifications

		GSX30									GS30			
Nominal Backlash	in (mm)	0.004 (.10)									0.004 (.10)			
Maximum Backlash (pre-loaded)	in (mm)	0.0									0.0			
Lead Accuracy	in/ft (mm/300 mm)	0.001 (.025)									0.001 (.025)			
Maximum Radial Load	lb (N)	30 (134)									30 (134)			
Brake Holding Torque - Dry	lbf-in (Nm)	78 (8.81)									78 (8.81)			
Brake Holding Torque - Oil Lubricated	lbf-in (Nm)	26 (2.94)									26 (2.94)			
Environmental Rating: Standard / Optional		IP65/67									IP65/67			
Motor Stator		118	138	168	218	238	268	318*	338*	368*	L6	M6	M6-DS	H6*
Trapezoidal Commutation														
Continuous Motor Torque	lbf-in	15.9	15.8	15.0	25.6	25.6	25.5	37.0	36.6	34.7	18.8	19.4	27.5	19.3
	(Nm)	(1.79)	(1.78)	(1.70)	(2.89)	(2.89)	(2.88)	(4.18)	(4.13)	(3.92)	(2.13)	(2.19)	(3.11)	(2.18)
Torque Constant (Kt)	lbf-in/A	3.4	6.8	13.6	3.4	6.8	13.6	3.5	6.8	13.7	3.44	6.90	7.21	13.79
	(Nm/A)	(0.39)	(0.77)	(1.54)	(0.39)	(0.77)	(1.54)	(0.39)	(0.76)	(1.55)	(0.39)	(0.78)	(0.81)	(1.56)
Continuous Current Rating: Greased (IG)	A	5.2	2.6	1.2	8.4	4.2	2.1	11.9	6.0	2.8	5.47	2.81	3.82	1.40
Oiled (IL)	A	10.4	5.2	2.5	16.8	8.4	4.2	23.9	12.1	5.7	10.94	5.63	7.63	2.80
Peak Current Rating	Amps	10.4	5.2	2.5	16.8	8.4	4.2	23.9	12.1	5.7	10.94	5.63	7.63	2.80
RMS Sinusoidal Commutation														
Continuous Motor Torque	lbf-in	16.6	16.5	15.7	26.8	26.8	26.7	38.7	38.3	36.3	19.7	20.3	28.8	20.2
	(Nm)	(1.88)	(1.87)	(1.78)	(3.03)	(3.03)	(3.01)	(4.38)	(4.33)	(4.10)	(2.23)	(2.30)	(3.26)	(2.28)
Torque Constant (Kt)	lbf-in/A	4.4	8.7	17.5	4.4	8.7	17.5	4.4	8.7	17.6	4.41	8.85	9.25	17.68
	(Nm/A)	(0.49)	(0.99)	(1.98)	(0.49)	(0.99)	(1.98)	(0.50)	(0.98)	(1.98)	(0.50)	(1.00)	(1.04)	(2.00)
Continuous Current Rating: Greased (IG)	A	4.2	2.1	1.0	6.9	3.4	1. 7	9.7	4.9	2.3	4.47	2.30	3.12	1.14
Oiled (IL)	A	8.5	4.2	2.0	13.7	6.8	3.4	19.5	9.9	4.6	8.93	4.60	6.23	2.29
Peak Current Rating	Amps	8.5	4.2	2.0	13.7	6.8	3.4	19.5	9.9	4.6	8.93	4.60	6.23	2.29
Motor Stator Data														
Voltage Constant (Ke)	Vrms/krpm	29.9	59.7	119.5	29.9	59.7	119.5	30.3	59.2	119.9	31.2	62.4	63.5	124.8
	Vpk/krpm	42.2	84.5	169.0	42.2	84.5	168.9	42.9	83.8	169.6	45.0	90.0	91.5	180.0
Pole Configuration		8	8	8	8	8	8	8	8	8	6	6	6	6
Resistance (L-L)	Ohms	2.8	11.2	49.6	1.1	4.5	18.0	0.65	2.6	11.6	2.37	8.96	4.87	36.17
Inductance (L-L)	mH	7.7	30.7	123.0	3.7	14.7	58.7	2.5	9.5	38.8	3.92	15.72	9.62	62.92
Brake Current @ 24 Vdc	A	.72	.72	.72	.72	.72	.72	.72	.72	.72	.72	.72	.72	.72
Brake Engage/Disengage Time	ms	250/50	250/50	250/50	250/50	250/50	250/50	250/50	250/50	250/50	250/50	250/50	250/50	250/50
Mechanical Time Constant (tm),ms	min	6.5	6.5	7.2	2.6	2.6	1.5	1.5	1.7	1.6	5.12	4.82	2.47	4.87
	max	10.8	10.9	12.0	4.3	4.3	4.4	2.5	2.5	2.8	8.55	8.06	4.12	8.14
Electrical Time Constant (te)	ms	2.8	2.7	2.5	3.3	3.3	3.3	3.8	3.7	3.3	1.65	1.75	1.98	1.74
Damping Constant	lbf-in/krpm	1.23	1.23	1.23	1.23	1.23	1.23	1.23	1.23	1.23	1.23	1.23	1.23	1.23
	(Nm/krpm)	(.14)	(.14)	(.14)	(.14)	(.14)	(.14)	(.14)	(.14)	(.14)	(.14)	(.14)	(.14)	(.14)
Friction Torque	lbf-in	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00
	(Nm)	(0.23)	(0.23)	(0.23)	(0.23)	(0.23)	(0.23)	(0.23)	(0.23)	(0.23)	(0.23)	(0.23)	(0.23)	(0.23)
Bus Voltage	Vrms	115	230	460	115	230	460	115	230	460	115	230	230	460
Speed @ Bus Voltage	rpm	3000	3000	3000	3000	3000	3000	3000	3000	3000	3000	3000	3000	3000
Motor Wire Insulation		Class 180 H									Class H			
Motor Stator Rating		Class 180 H									Class F			
Thermal Switch Case Temperature	°C	100									100			
Standard Connectors (O-option)	Motor	MS-3112-E16-8P									MS-3112-E16-8P			
	Feedback	MS-3112-E14-18P									MS-3112-E14-18P			
	Brake/Limit Sw.	MS-3112-E12-8P									MS-3112-E12-8P			
End Switches (optional)		NC,NPN 9-24Vdc 20mA									NC,NPN 9-24Vdc 20mA			

All ratings at 25 degrees Celsius
For amplifiers with peak sinusoidal commutation Kt = Ktrms(0.707), Ic = Icrms/(0.707), Ipk = Ipkrms/(0.707)
*The 3 stack lamination fits only the 6 inch and longer GSX30. The GSX30-03 can only accommodate the 1 or 2 stack.
*The H6 option is not available in the 3" stroke GS30 actuator.

Specifications subject to change without notice.

Figure B.2: Mechanical and electrical specifications of the linear actuators

B.2 Driver Specifications

	ARS-310/5	ARS-310/10
Power supply	1 x 230 V AC or 310 V DC 24 V DC (0.5 A)	3 x 230 V AC or 310 V DC 24 V DC (0.5 A)
Rated current I_{Rate}	5 A_{rms}	10 A_{rms}
Peak current I_{Max}	10 A_{rms}	20 A_{rms}
Rated power at DC supply	1500 VA	3000 VA
Rated power at AC supply	1000 VA	2500 VA
Dimensions in mm (w x h x d) (without counterplug)	70 x 215 x 215	70 x 230 x 215
Encoder evaluation (plug-in modules)	Resolver, incremental encoder, Stegmann encoder with HIPERFACE, Heidenhain encoder	
Interfaces	RS 232 optional: CAN bus or PROFIBUS-DP	

Figure B.3: Technical data for the driver (ARS-310/5)

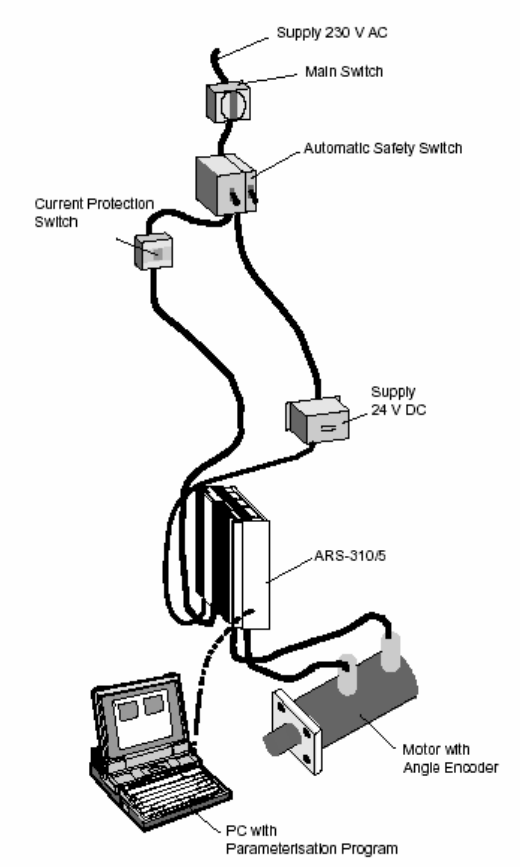


Figure B.4: Complete construction of the driver with the electric motor

B.3 Data Acquisition Hardware Specifications

Humusoft MF614 multifunction I/O card

Analog Input:

Channels:	8 single-ended
A/D Converter:	12-bit, 10microsecond conversion time
Input Ranges:	±10V, ±5V, 0-10V, 0-5V
Trigger Mode:	Software
Over voltage:	±16V

Analog Output:

Channels:	4 channels
Output Range:	±10V
Output Current:	10mA max.

Digital I/O:

Input lines:	8, TTL compatible
Output lines:	8, TTL compatible

Timer/Counter:

Timer chip:	9513
Timer resolution:	50 ns

Encoder Inputs:

Input channels:	4, single ended or differential
Inputs:	A, B, Index

Input frequency: max 2.5 MHz

General:

Power consumption: 100 mA @ +5 V, 50 mA @ +12 V, 50 mA @ -12 V

Operating temperature: 0 to 50 °C

Connector: 2 x DB-37

Interface: PCI

B.4 Inertial Measurement Unit Technical (IMU) Specifications

	Angular	Linear
Measurement Range	±600 deg/sec to ±1000 deg/sec	±50g
Scale Factor	500 ppm 1σ	2000 ppm 1σ
Bias		
Instability	5 deg/hr 1σ	-
Repeatability	100 deg/hr 1σ	10 mg 1σ
Random Walk	1.0 deg/√hr	1.0 m/s/√hr
g Sensitivity	1 deg/hr/g	
Axis Alignment	400 µrad 1σ	400 µrad 1σ
Bandwith (-90° phase shift) *other bandwidths available*	75 Hz	75 Hz
Noise (in band)	0.5 deg/sec rms	5 mg rms
Environment		
Operating Temperature	-40 deg C to +75 deg C	
Relative Humidity	100%	
Vibration (operational)	18 g rms (20 Hz to 2 kHz)	
Shock	250 g	
Altitude	Up to 25,000 m	
Mass	250 grammes	
Electrical		
Supply Voltage +/-Vs	±15V DC and +5V DC	
Power Consumption	5 VA	
General		
Built In Test	Command BIT	
Start-up Time	500 ms	
Interface	Programmable RS422 interface: 'AMRAAM', PC, High Rate	

Figure B.5: Specifications of the Inertial Measurement Unit (IMU)

APPENDIX C

SIMULINK® MODEL FOR THE REAL-TIME CONTROL SYSTEM

Real-time control system model subsystems are explained in detail in this appendix. Figure C.1 shows the blocks for platform motion demanding. In Figure C.2, the subsystem for motion demanding in pitch axis is represented. As seen, step input, sinusoidal input, tank motion etc can be given at any time.

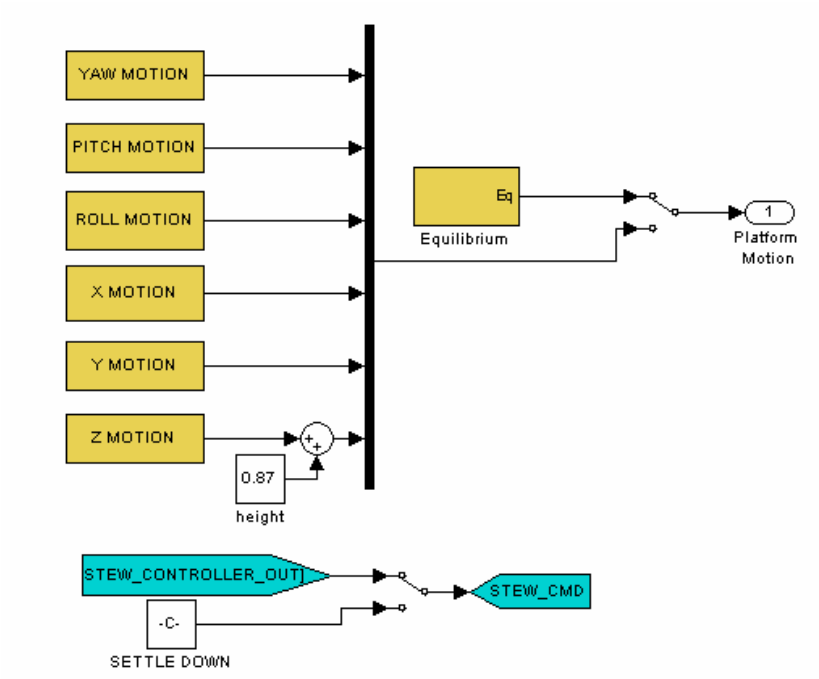


Figure C.1: Platform motion demanding subsystem

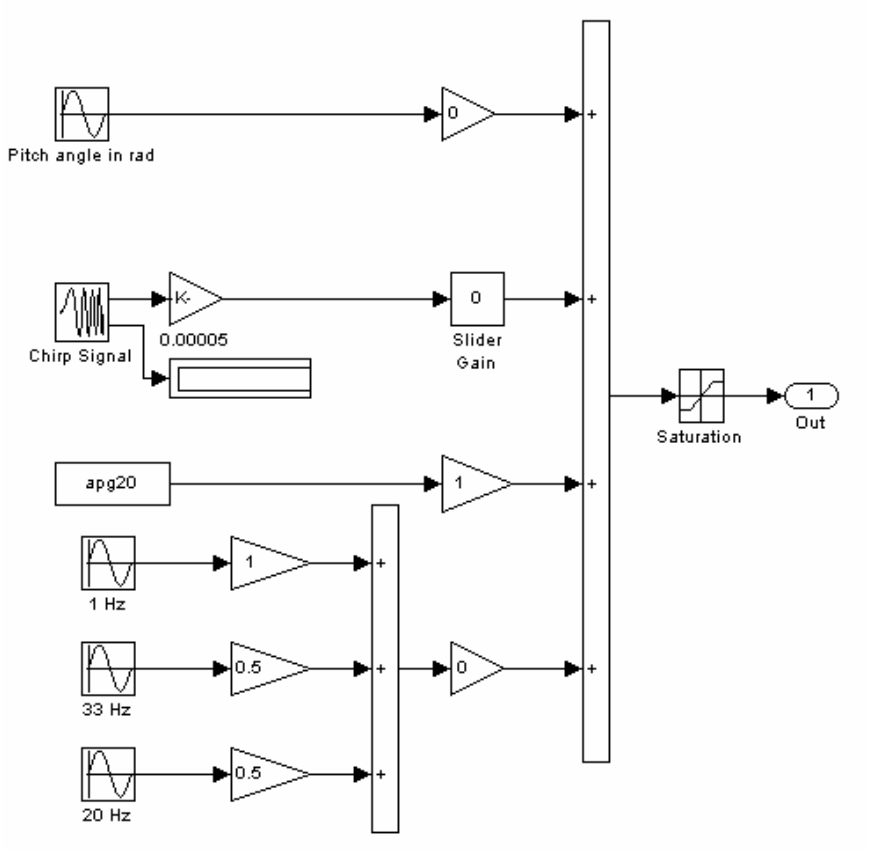


Figure C.2: Motion input blocks for pitch axis

Inverse kinematics solution takes place in the next block. Inverse kinematics code is written to s-function builder block. This block automatically creates the s-function of inverse kinematics function which is a must for MATLAB® to use a function in real-time applications. Figure C.3 shows related subsystem.

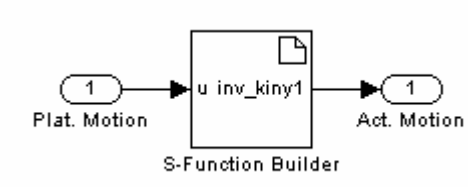


Figure C.3: Inverse kinematics function in real-time

Measurement block reads incremental encoder data from encoder input block in which DAQ board is installed. It converts incremental encoder data into position feedback signal. In Figure C.4, major measurement block is shown. Figures C.5 to C.10 represent the related submodels for encoder reading and conversion to actuator displacements according to the flowcharts given in Section 4.3.2.

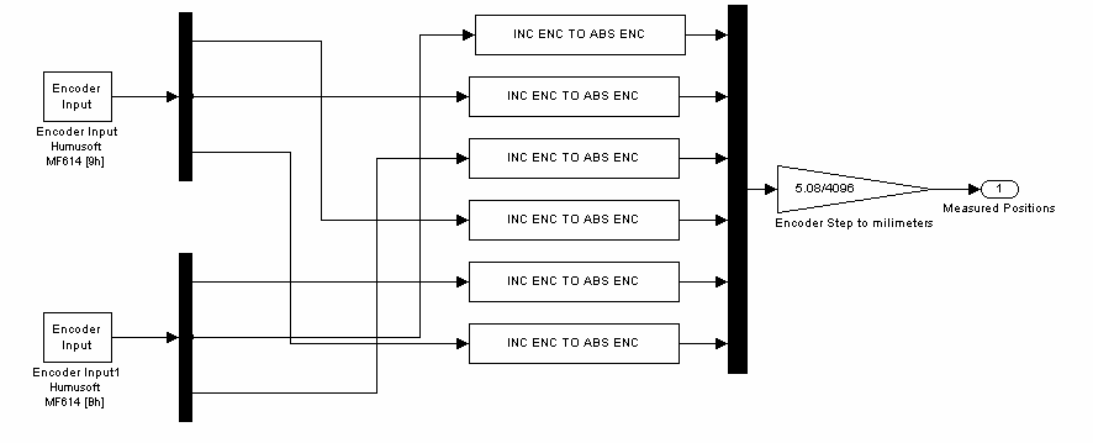


Figure C.4: General model for encoder reading

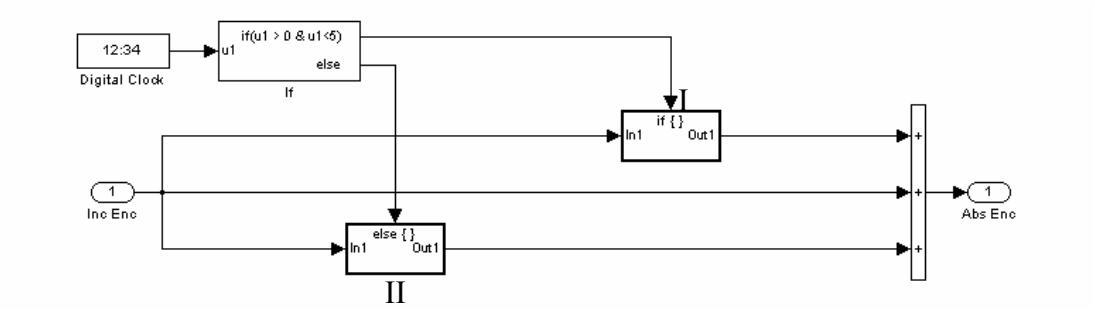


Figure C.5: One of the blocks for converting incremental encoder to actuator displacement

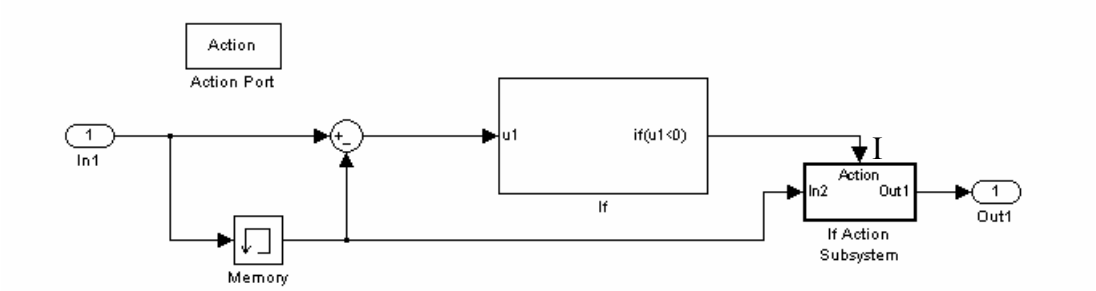


Figure C.6: Block I of the model in Figure C.5

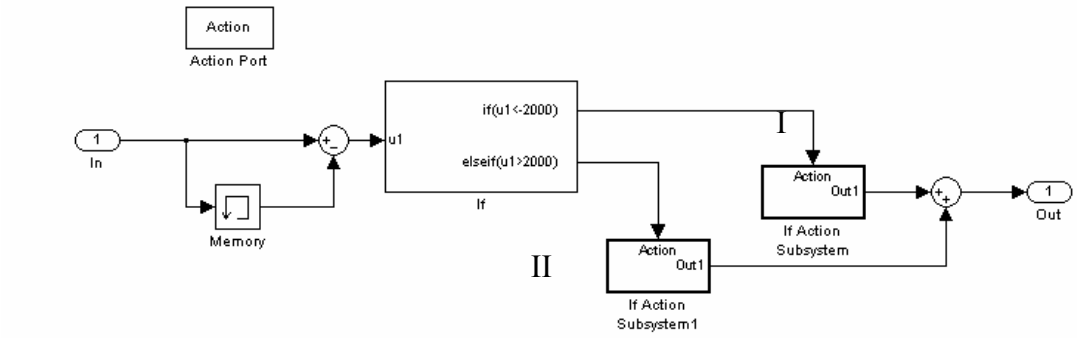


Figure C.7: Block II of the model in Figure C.5

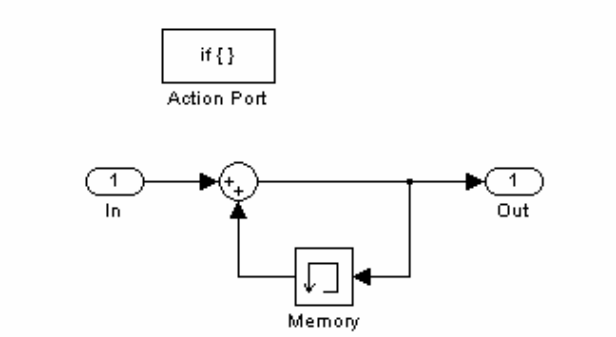


Figure C.8: Block I of the model in Figure C.6

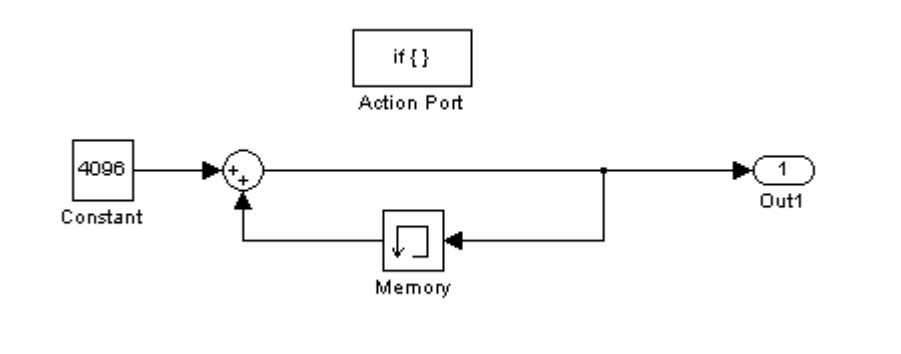


Figure C.9: Block I of the model in Figure C.7

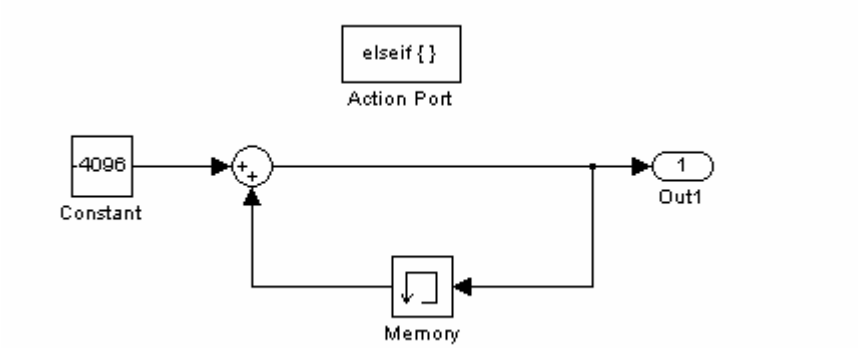


Figure C.10: Block II of the model in Figure C.7

In motion arrangement block, timing facilities for the Stewart Platform to operate on an equilibrium point are done as shown in Figure C.11.

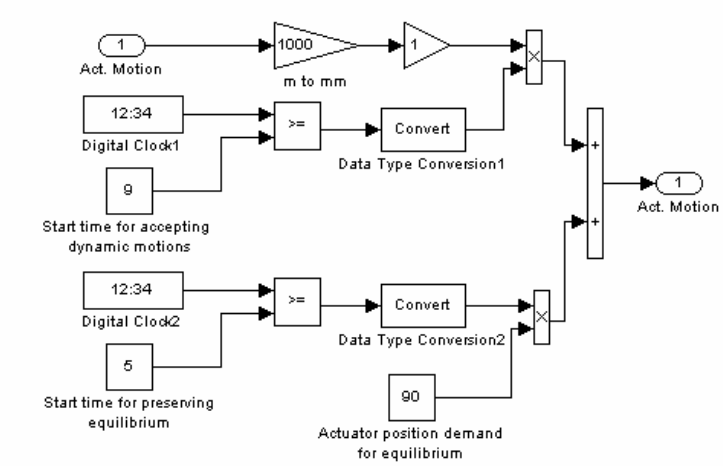


Figure C.11: Motion arrangement subsystem

In position controller block, PI position control operations with velocity feedforward are done as shown in Figure C.12. Start of the integral operation is the time for beginning of the position controller. Velocity feedforward starts a little time after PI loop begins preventing abrupt changes and harsh movements at the very beginning of the motion.

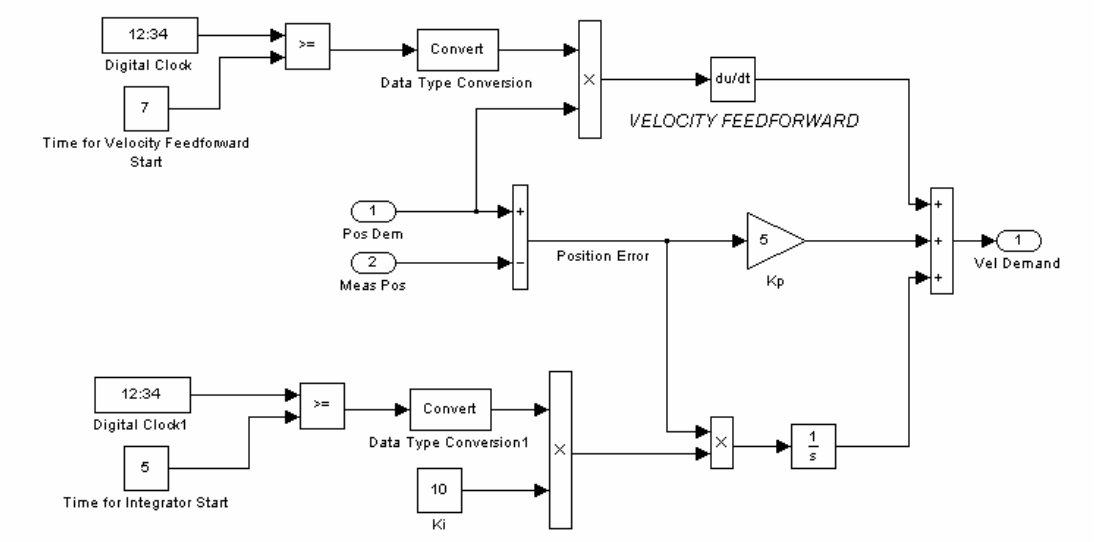


Figure C.12: Position controller subsystem

In analog output subsystem, velocity demands in voltages are given to drivers via analog output blocks addressing to DAQ boards as shown in Figure C.13. This subsystem also involves timing arrangements between velocity control and position control.

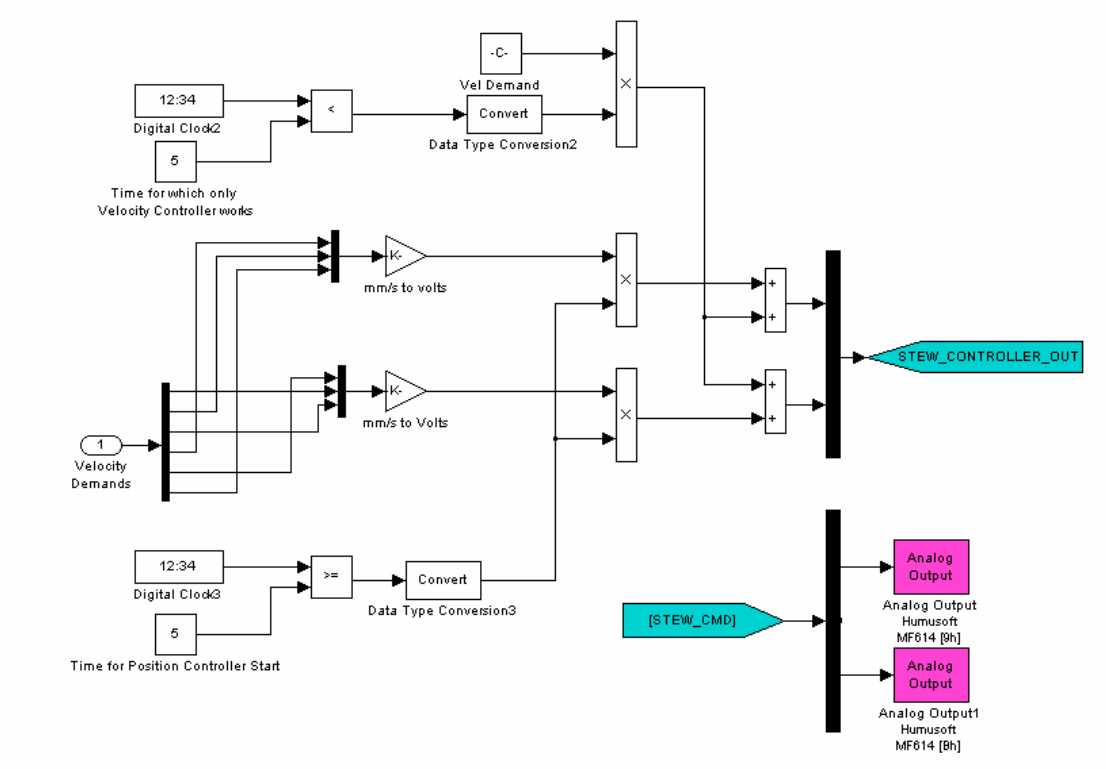


Figure C.13: Analog output subsystem

APPENDIX D

DYNAMIC IDENTIFICATION OF THE RELATED LINKS IN PRO/ENGINEER®

D.1 Dynamic Identification of the Moving Part of the Actuator

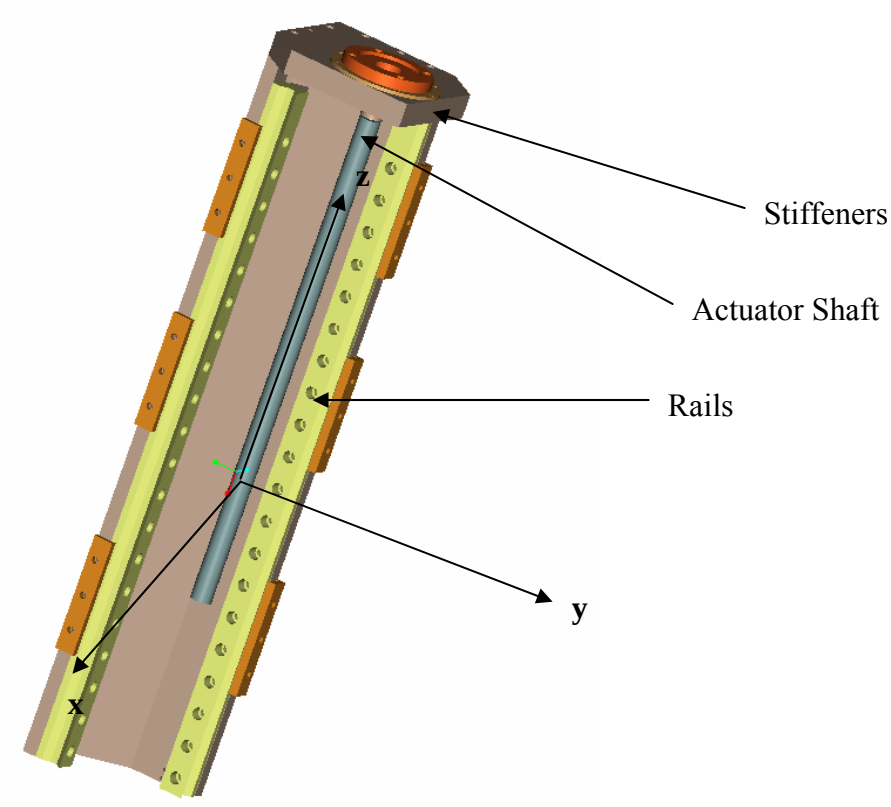


Figure D.1: Moving part of the actuator (the upper leg)

A Cartesian coordinate system is created where z axis is in the direction of motor shaft. X and y axes of this coordinate system form a plane in which center of gravity

of the part exists and this plane is perpendicular to the z-axis. Center of gravity of the part is found and reflected to m-file in Appendix A.1. Mass and inertia tensors found shown below are also added to this m-file.

$$Mass = 11.7kg$$

Inertia Tensor with reference to Center of Gravity:

$$I = \begin{bmatrix} 0.402 & 0.0009 & -0.0045 \\ 0.0009 & 40.2 & -0.0045 \\ -0.0045 & -0.0045 & 0.0324 \end{bmatrix} kg.m^2$$

D.2 Dynamic Identification of the Stationary Part of the Actuator

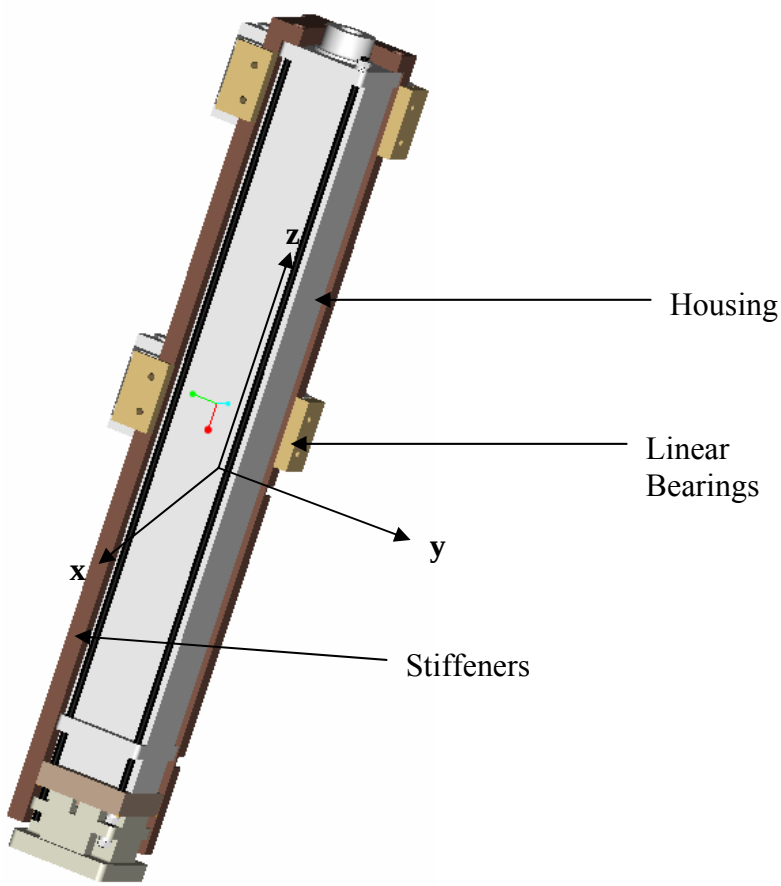


Figure D.2: Stationary part of the actuator (the lower leg)

A Cartesian coordinate system is created where z axis is in the direction of motor shaft. X and y axes of this coordinate system form a plane in which center of gravity of the part exists and this plane is perpendicular to the z-axis. Center of gravity of the part is found and reflected to m-file in Appendix A.1. Mass and inertia tensors, which are shown below, are also added to this m-file.

$$Mass = 18.43kg$$

Inertia Tensor with reference to center of gravity:

$$I = \begin{bmatrix} 0.8888 & 0.0006 & 0.0093 \\ 0.0006 & 0.8856 & 0.009 \\ 0.0093 & 0.009 & 0.0297 \end{bmatrix} kg.m^2$$

D.3 Dynamic Identification of the Moving Platform and the Test Equipment

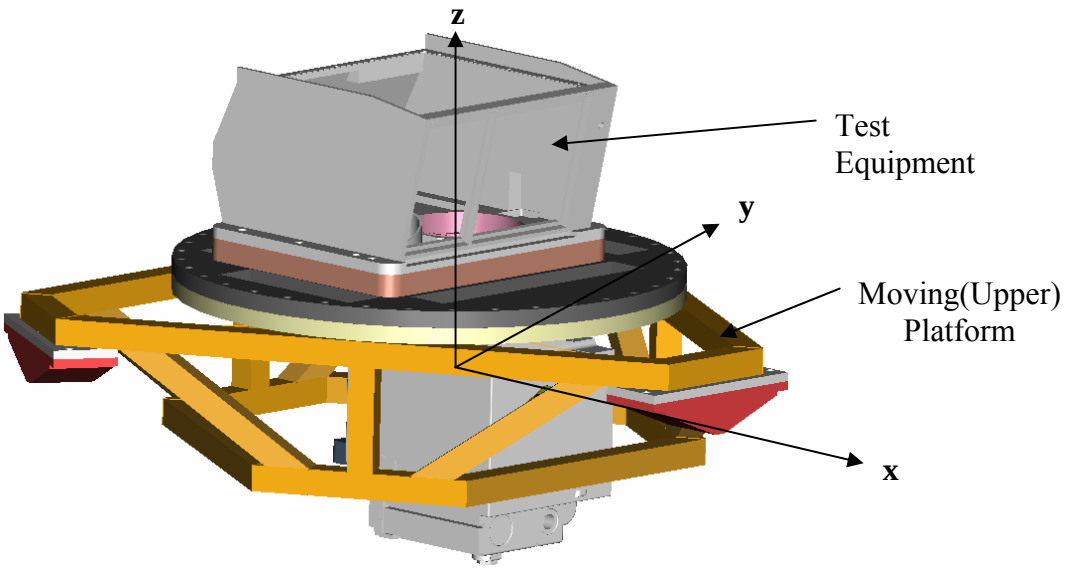


Figure D.3: The moving platform and the test equipment

A Cartesian coordinate system is created, in which z axis is in the direction shown in Figure D.3; x and y axes form a plane, in which center of gravity of the part exists and perpendicular to z-axis. Center of gravity of the part is found and reflected to m-file in Appendix A.1. Mass and inertia tensors, which are shown below, are also added to this m-file.

$$Mass = 97kg$$

Inertia Tensor with reference to Center of Gravity:

$$I = \begin{bmatrix} 1.0500 & -0.0034 & 0.0005 \\ -0.0034 & 1.0500 & 0.0134 \\ 0.0005 & 0.0134 & 1.9500 \end{bmatrix} kg.m^2$$