MECHATRONICS INTEGRATION FOR A VEHICLE SIMULATOR

Taahir Kader

Student Number: 210513192

Submitted in fulfilment of the academic requirements for the degree of
Master of Science in Engineering,
College of Agriculture, Engineering and Science,
University of KwaZulu-Natal

June 2016

Supervisor: Professor Riaan Stopforth

Co-Supervisor: Professor Glen Bright

As the candidate’s supervisor I agree to the submission of this dissertation.

Date of Submission: __________________________

Supervisor: __________________________

Professor Riaan Stopforth

Declaration

I, Taahir Kader, declare that

1. The research reported in this dissertation, except where otherwise indicated, is my original research.

2. This dissertation has not been submitted for any degree or examination at any other university.

3. This dissertation does not contain other persons’ data, pictures, graphs or other information, unless specifically acknowledged as being sourced from other persons.

4. This dissertation does not contain other persons' writing, unless specifically acknowledged as being sourced from other researchers. Where other written sources have been quoted, then:

a. Their words have been re-written but the general information attributed to them has been referenced

b. Where their exact words have been used, then their writing has been placed in italics and inside quotation marks, and referenced.

5. This dissertation does not contain text, graphics or tables copied and pasted from the Internet, unless specifically acknowledged, and the source being detailed in the dissertation and in the References sections.

Signed: __________________________

Publications

The work in this dissertation has been presented in the following publications:

Publication 1:

Kader, T., Stopforth, R. and Bright, G. (2016) 'Simulation System to Aid in Vehicle Simulator Design', Research and Development (R&D) Journal of The South African Institution of Mechanical Engineering (SAIMechE), [Under review], February 2016. This journal paper contains work done in chapter 2, chapter 4 and chapter 5 of this dissertation.

Signed: _________________________

Acknowledgements

I would like to take this opportunity to express my sincere gratitude and appreciation to my supervisor Prof Riaan Stopforth for all the help, support, advice and guidance whilst on this journey. I wish to also thank Prof Glen Bright for the support, advice and suggestions throughout my masters which benefitted immensely.

I would also like to express gratitude to the following people and organisations:

- To my family for all the support and especially to my parents for the encouragement and commitment shown throughout my life.

- CSIR for the financial support provided to pursue this research work.

- Mechanical engineering workshop staff and admin staff for all their assistance.

- To my friends for providing motivation and assistance during this research work.

Abstract

This dissertation presents the research and integration of a mechatronics system to be used in a vehicle simulator. The vehicle simulator is comprised of a 3-DOF platform which is used to provide motion cues to the driver. Kinematic analysis is performed on the 3-DOF system and this analysis assists in implementing platform motion control. To recreate the motion sensations experienced in an actual vehicle while respecting the platform workspace limits the classical washout algorithm is implemented in the vehicle simulator. A novel simulation system was contributed in Matlab/Simulink to aid in vehicle simulator design. This simulation setup incorporates all the motion cueing aspects; these aspects include input vehicle data scaling, the classical washout algorithm and inverse kinematic analysis. The developed simulation system was used to adjust the motion cueing parameters to ensure motion that respects the actuator motion constraints. These constraints ensure the vehicle simulator is operated safely.

A second contribution used the developed simulation system in Matlab/Simulink and the human vestibular system models. A performance evaluation was performed on the 3-DOF system against the traditional 6-DOF system. The results highlight the benefits of the 3-DOF system in replication of certain motion cues. Software was developed to receive input game data and output actuator stroke lengths to the motion control system. Limitations in the motion control system were found when testing was done on the vehicle simulator. These limitations led to a modified partial 2-DOF vehicle simulator. The motion control hardware is able to replicate actuator motion well. The final vehicle simulator system is a partial 2-DOF system that provides visual and motion cues that create a realistic driving experience. The developed system is suitable for applications with cost constraints and reasonable performance requirements.

Table of Contents

List of Figures

List of Tables

List of Acronyms

3-D	Three Dimensional
AHS	Automated Highway System
CAD	Computer-Aided Design
CSV	Comma-Separated Values
DOF	Degree of Freedom
DOFs	Degrees of Freedom
DLL	Dynamic Link Library
HP	High-Pass
ITS	Intelligent Transportation Systems
kg	Kilogram
LAN	Local Area Network
LED	Light-Emitting Diode
LP	Low-Pass
m	Metre
NADS	National Advanced Driving Simulator
PID	Proportional–Integral–Derivative
PLC	Programmable Logic Controller
RTOS	Real-Time Operating System
s	Second
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VR	Virtual Reality
X-Sim	Cross-Simulator Software

1 Introduction

The following chapter provides details on the research conducted on the various aspects involved in a vehicle simulator system and explains the research objectives for this dissertation. The sections in this chapter aim to highlight all the aspects that a vehicle simulator is comprised of.

The field of robotic manipulators is explained and the use of these manipulators in a vehicle simulator system is discussed. The various literatures on vehicle simulators are discussed, and this research assists in categorising these systems based on cost and fidelity. Applications for vehicle simulators are also discussed.

A literature review of motion cueing provides insight into the methods used to increase the vehicle simulator fidelity. This section also highlights how the human vestibular system functions and how this system is exploited in a vehicle simulator.

The various techniques in research for motion control of robotic manipulators, specifically parallel manipulators which form the basis of a vehicle simulator system, is then discussed.

Using the research knowledge gained the motivation for this study, scientific contributions and research objectives are presented.

1.1 Robotic Manipulators

In robotics, there are two main types of manipulators which are used to create a robotic system that performs a certain number of tasks. These manipulators include the serial and parallel manipulators.

A serial manipulator is an open chain kinematic mechanism which comprises a fixed base, series of links attached together by joints and an end-effector. The motion of these manipulators is achieved by actuating individual joints. By controlling the motion of the joints either the position and/or the orientation of the end-effector is manipulated to perform a specific task (Ghosal, no date). The serial manipulator, illustrated in figure 1-1 (Ghosal, no date), is referred to as an open loop manipulator. This is because none of the links form closed kinematic chains.

The serial manipulator has a large workspace and is dexterous. The disadvantages of the serial manipulator are that the cantilever-like structure is not rigid and it has poor dynamic performance at high speed (Lee and Shah, 1988).

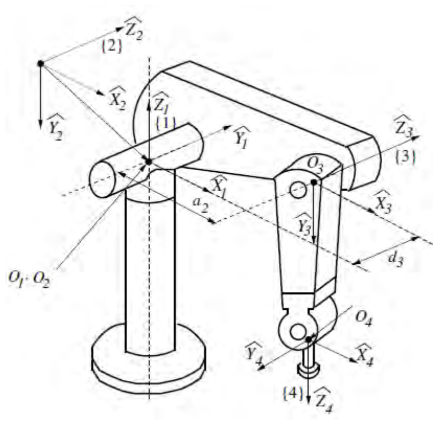


Figure 1-1 Schematic of a PUMA 560 Serial Manipulator

Parallel manipulators consist of a fixed base and a number of independent kinematic chains connected to a moving platform or end-effector. Parallel manipulators have a greater load carrying capacity due to the many parallel links distributing the load. The actuator locations in a parallel manipulator are near the base, this location results in low inertia of the parts in motion. Parallel manipulators do not suffer from accumulation of errors along a kinematic chain and have a higher stiffness. The disadvantages of these manipulators include smaller workspace due to the link interference among kinematic chains, physical constraints introduced by universal and spherical joints, complex forward kinematics, platform singularities and motion actuator range limits (Patel and George, 2012).

In 1942 a patent was filed, in the US, for a parallel robot to control the movement and positioning of a spray gun (Pollard, 1942). Development of parallel manipulators date back to the 1960s during which a universal tyre test machine was developed (Gough and Whitehall, 1962); this manipulator is a six-linear jack system. Later, Stewart developed a 6-DOF parallel manipulator to be used as a flight simulator (Stewart, 1965). Due to the complex forward kinematics and difficulty to manufacture precise spherical joints at low cost (Tsai et al., 1996), the development of lower DOF parallel manipulators has been researched extensively.

Analysis of a 3-DOF parallel manipulator, which has two rotational degrees of freedom and one translational degree of freedom, was performed (Lee and Shah, 1988). Solutions for forward and inverse kinematics were derived for this 3-DOF parallel manipulator. A 3-DOF purely translational parallel robot, known as the DELTA robot was developed (Clavel, 1988). Forward and inverse kinematics for the DELTA robots was presented (Sternheim, 1987). The DELTA robot can operate with high speed and accuracy. This led to high usage of the DELTA robot in the medical, pharmaceutical and packaging industry (Patel and George, 2012). Kinematic analysis was performed on a purely translational 3-DOF parallel manipulator made entirely from revolute joints (Tsai et al., 1996).

Recent research has focused on the development of hybrid manipulators; these manipulators combine both serial and parallel manipulators and aim to benefit from the advantages of both. An 8-DOF hybrid manipulator was developed and aimed to combine the high rigidity of a parallel robot and large workspace of a serial robot (Mohammadipanah and Zohoor, 2009). A 3-DOF hybrid parallel manipulator which is modular was created. This manipulator is aimed at being reconfigurable, either manually or automatically, as well as self-repairing (Ng et al., 2006).

Applications for parallel manipulators include motion simulators, precise machine tools and micro mechanisms. Figure 1-2 (Patel and George, 2012) depicts a 6-DOF Stewart platform parallel manipulator which is typically used in motion simulators.

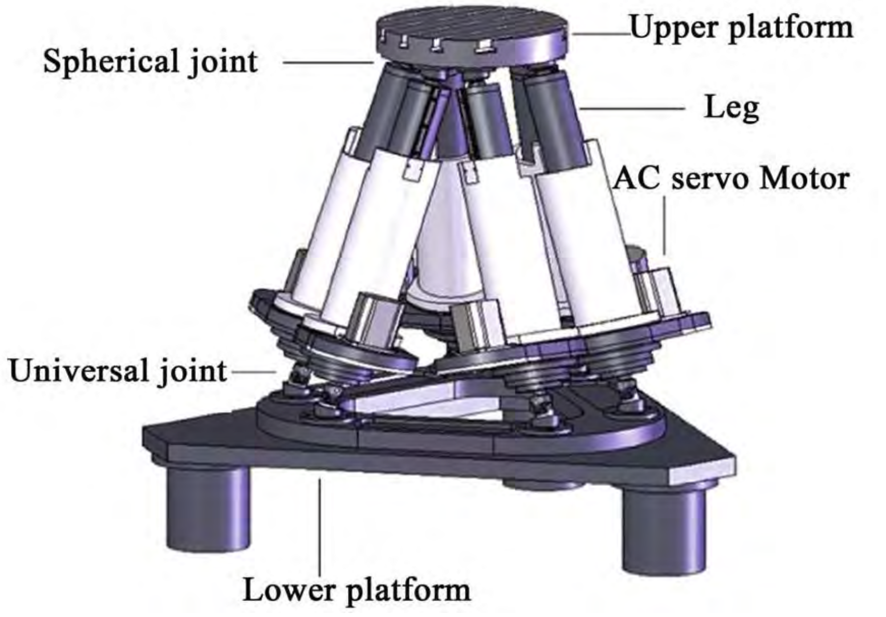


Figure 1-2 Stewart Platform Parallel Manipulator

In robotics an important area of interest is the kinematics of the manipulator. The kinematics of a robotic manipulator aims to provide a relationship linking the motion of the end-effector to the motion of the joint variables. These joint variables could be joint angles for revolute joints or joint lengths for prismatic joints.

The two main aspects in kinematics is the forward and inverse kinematics. In forward kinematics the joint variables are known and the aim is to work out the position and orientation of the end-effector. Inverse kinematics gives us the position and orientation of the end-effector and requires the computation of joint variables to achieve such a position and orientation.

The research carried out focused on parallel manipulator kinematics, these robot manipulators have fairly straight forward inverse kinematics but have complex forward kinematic equations. Kinematic analysis of the 6-DOF motion platform robotic wrist was performed, the inverse kinematics solution was presented in close form and the Newton-Rhapson method was used to iteratively solve the forward kinematic solution (Nguyen et al., 1991). Closed form forward and inverse kinematic solutions for a 3-DOF parallel manipulator were developed (Lee and Shah, 1988). Simulations, using MATLAB/Simulink and SimMechanics toolbox, were used to verify the inverse kinematic equations derived for a similar 3-DOF platform and visually analyse platform motion (Yu et al., 2010). The research knowledge gained is used to perform kinematic analysis on the motion platform used for the vehicle simulator system.

1.2 Vehicle Simulators

Motion simulators originated from the development of flight simulators. The first vehicle simulators started to appear in the 1970s and featured improved fidelity with advancements in computer technology. In the early 1980s Daimler-Benz created a high fidelity vehicle simulator (Drosdol and Panik, 1985). Subsequent high-fidelity simulators have since been created by General Motors (Bertollini et al., 1994), University of IOWA (Freeman et al., 1996) and Toyota (Toyota, 2007). Renault initially developed a 6-DOF motion platform to be used as a vehicle simulator (Reymond and Kemeny, 2000). Renault currently have the ULTIMATE simulator which is capable of 8-DOF, it consists of a 6-DOF Stewart platform combined with an XY motion system (Colombet et al., 2008). Figure 1-3 (Colombet et al., 2008) illustrates the Renault ULTIMATE driving simulator, the XY table is added to better replicate sustained longitudinal and lateral acceleration.

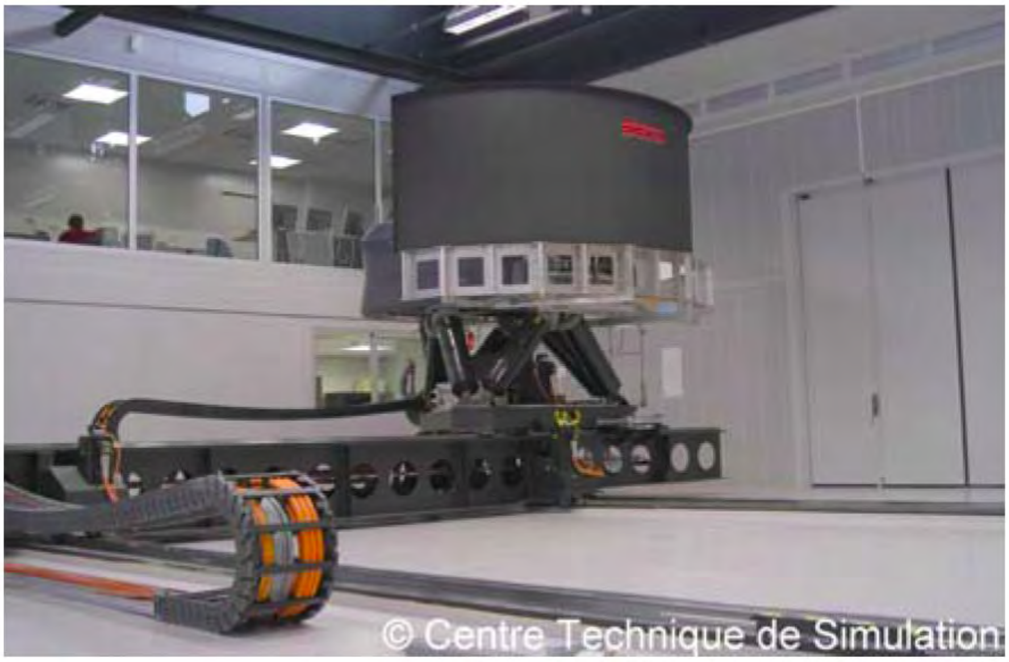


Figure 1-3 Renault ULTIMATE Driving Simulator

The National Advanced Driving Simulator (NADS), illustrated in figure 1-4 (Chen et al., 2001), which was funded by the National Highway Traffic Safety Administration (NHTSA) is a driving simulator operated at the University of Iowa. The 9-DOF system consists of a turntable that rotates $\pm$330 degrees and a 6-DOF Stewart platform which moves on a XY motion system (Chen et al., 2001). The XY motion system is added to better replicate sustained longitudinal and lateral acceleration similar to the Renault ULTIMATE driving simulator.

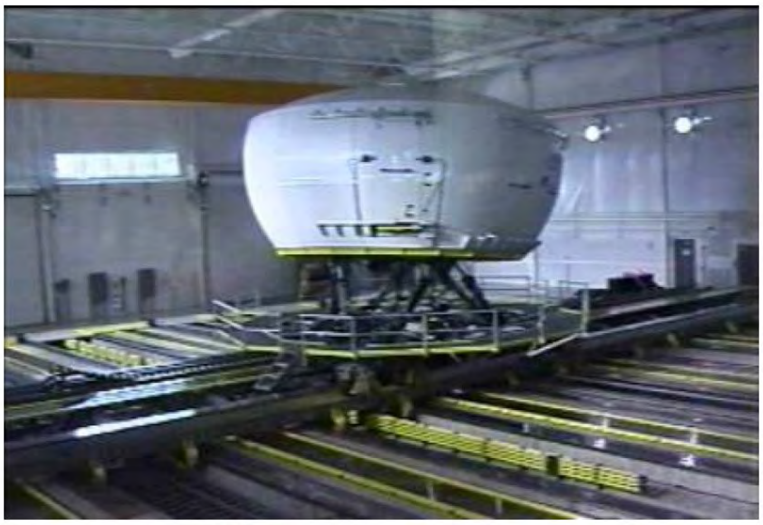


Figure 1-4 The NADS at the University of Iowa

Lee et al. developed an effective and economical driving simulator based on the 6-DOF platform and this simulator is a scaled down version intended for usage in human factor studies and evaluation of full-scaled motion simulators. This work also highlighted the various subsystems and how these are put together to create a high fidelity driving simulator system (Lee et al., 1998). A motion control system for this driving simulator, which is hydraulically driven, was subsequently developed (Kim et al., 1997). Universiti Teknologi Malaysia (UTM) presented a conceptual design for a 6-DOF platform. This study performed simulations, using Matlab/Simulink and the SimMechanics toolbox. The simulations allowed for visualisation of the motion platform and by integrating with the inverse kinematics the user is provided with a graphical display of motion cues. This type of simulation allows testing and verification of the vehicle simulator platform to be performed before an actual system is constructed (Shiong et al., 2009).

Several studies have looked at alternative vehicle simulator systems due to the excessive cost of the traditional 6-DOF Stewart platform used to provide motion cues. A low-cost 2-DOF motion platform was developed and it allowed for the recreation of longitudinal and yaw motion. This particular simulator is a compromise between motion replication quality, cost and compactness. It is intended to be used in driving schools, hospitals and other areas (Arioui et al., 2009). The 2-DOF motion platform performance evaluation and experiments were also performed. Using the classical washout algorithm to replicate motion cues the platform showed acceptable driving realism (Arioui et al., 2011).

In the leisure industry low-cost motion systems are the most common systems used. It is adequate to represent just rotational motion along the x-axis (Roll), rotational motion along the y-axis (Pitch) and translation motion along the z-axis (Heave). The increased availability of these lower-cost systems will allow for third world countries to adopt them in civil and military applications (Denne, 1986). A 3-DOF motion simulator was developed and a study concluded that the participants feel the vehicle simulator system does well to replicate vehicle motion (Capustiac et al., 2011).

In a vehicle simulator a number of sub-systems exist, as illustrated in figure 1-5 adapted from (Taikui and Jianmin, 2011). These sub-systems interact in cohesion to provide a high fidelity simulator. These sub-systems include:

- The automobile cab system, which is used to provide the inputs from the driver. This system uses component such as a steering wheel, gear knobs, acceleration and brake pedals.

- The visual system is used to provide visual cues to the driver based on the inputs received from the automobile cab system. This system also provides car kinematic parameters, such as linear accelerations and angular velocities.
- A computer control system is used to transform the input linear accelerations and angular velocities of the vehicle into actuator stroke lengths of the motion platform.
- The motion platform system provides the dynamic control of the motion platform and subsequent motion cues experienced by the driver in the simulator.

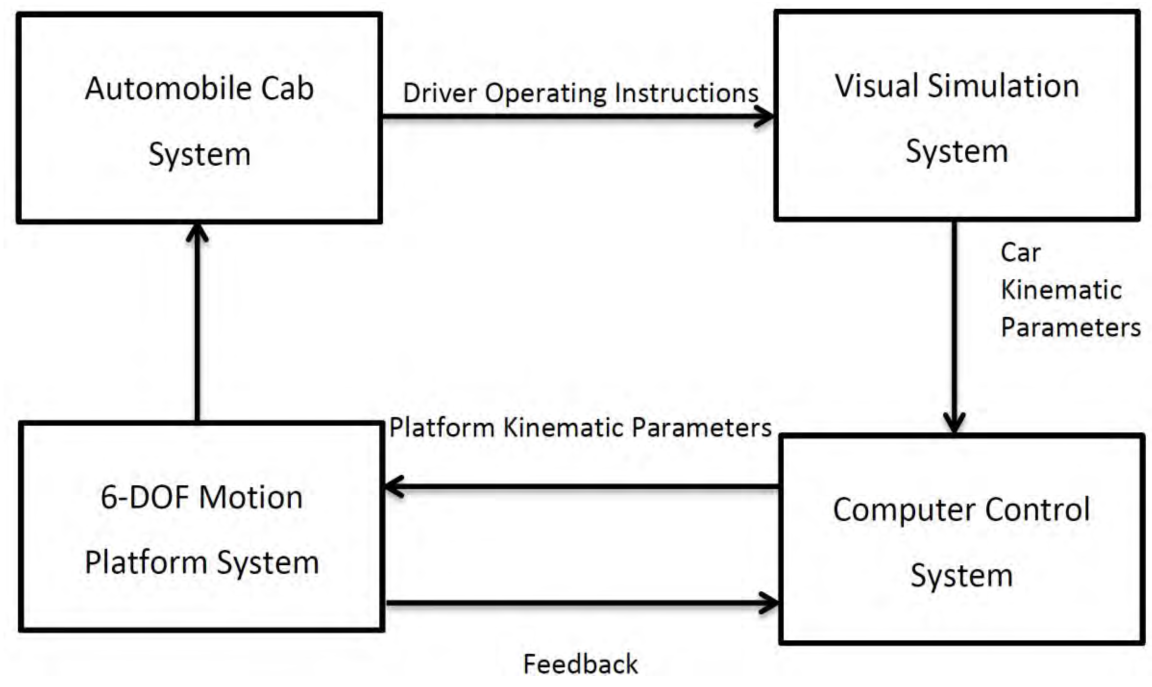


Figure 1-5 Vehicle Simulator Sub-systems

A study performed classifies vehicle simulators into low, medium or high-cost simulators as follows (Blana, 1996):

- Low-cost simulators are now available with improvements in computer technology which have enabled the ability of creating reasonable fidelity motion systems. These simulators are useful for dissertation related research and vehicle manufactures with limited budget for research.
- Medium-cost simulators provide a large projection screen, advanced imaging techniques and a complete vehicle. These systems can be comprised of a fixed-base or a motion platform.
- A high-cost simulator provides a 360 degree field of view and is located in an enclosed cabin. They are usually made from a 6-DOF motion platform.

Applications for vehicle simulators include:

- Research related to human factor studies – The Kookmin University fixed-based driving simulator was used to test driver reactions in the recreation of an accident scenario. It was designed to ascertain to what level driver carelessness or absentmindedness contributed to traffic accidents. Unfortunately due to the simulator being fixed-based realism was compromised and no significant conclusions could be drawn (Lee et al., 2001). The NADS operated at the University of Iowa was designed to test a host of human factors in contributing to traffic accidents (Chen et al., 2001). It is also intended to use the NADS to test Intelligent Transportation Systems (ITS) and Automated Highway System (AHS) technologies in a safe and controlled environment, with one of the outcomes being to assess how these system impact overall driver performance (Stall and Bourne, 1996).

- To validate vehicle dynamic models, test car prototypes and new features - Renault used a 6-DOF motion simulator to test an adaptive cruise control system. Testing in the simulator allowed for critical and even dangerous scenarios to be tested. The simulator is seen as a prototyping tool which allows for the adaptive cruise control system to be tested before it is integrated into an actual vehicle (Raymond et al., 2000).

- To facilitate training of the vehicle driver - The Arizona Department of Transportation (ADOT) have used fixed-base vehicle simulators to train snowplow operators. Two types of training were performed, the first was to teach drivers how to react to potential hazard while operating a snowplow and the second taught drivers proper driving techniques to increase fuel efficiency. It was concluded that both types of training have value and improvements to the training was planned (Kihl and Wolf, 2007). Utah Department of Transportation (UDOT) in collaboration with University of Utah and General Electric Driver Development (GEDD) developed a similar training program for snowplow operators, which included training in both a fixed-base and motion simulator. The motion simulator was used to help operators prepare for issues critical to the safe and efficient operation of the snowplow. Fixed-based simulators were used to teach driving techniques to improve fuel efficiency. It was concluded that operators who received training had lower odds of being involved in an accident than the control group who were not trained. Fuel efficiency was also greater for trained drivers (Strayer et al., 2004). Fuel efficiency training for truck drivers in a truck fleet operation concluded that drivers with the poorest fuel efficiency benefitted significantly from the training. Proper driving techniques learnt during training were shown to be retained for this group (Strayer and Drews, 2003).

Figure 1-6 illustrates the designed vehicle simulator system; this particular system is a lower-cost system. It is comprised of a 3-DOF motion platform capable of providing rotational motion about the x-axis (Roll), rotational motion about the y-axis (Pitch) and translational motion along the z-axis (Heave). This motion capability allows the platform to replicate sustained longitudinal and lateral accelerations, through a technique called tilt coordination. Inability to replicate sustained longitudinal accelerations results in poor simulation of maneuverers such as emergency braking (Arioui et al., 2009). The designed vehicle simulator systems performance is evaluated against the traditional 6-DOF Stewart platform using the human vestibular system models, to highlight the benefits of such a system in replicating certain motion cues.

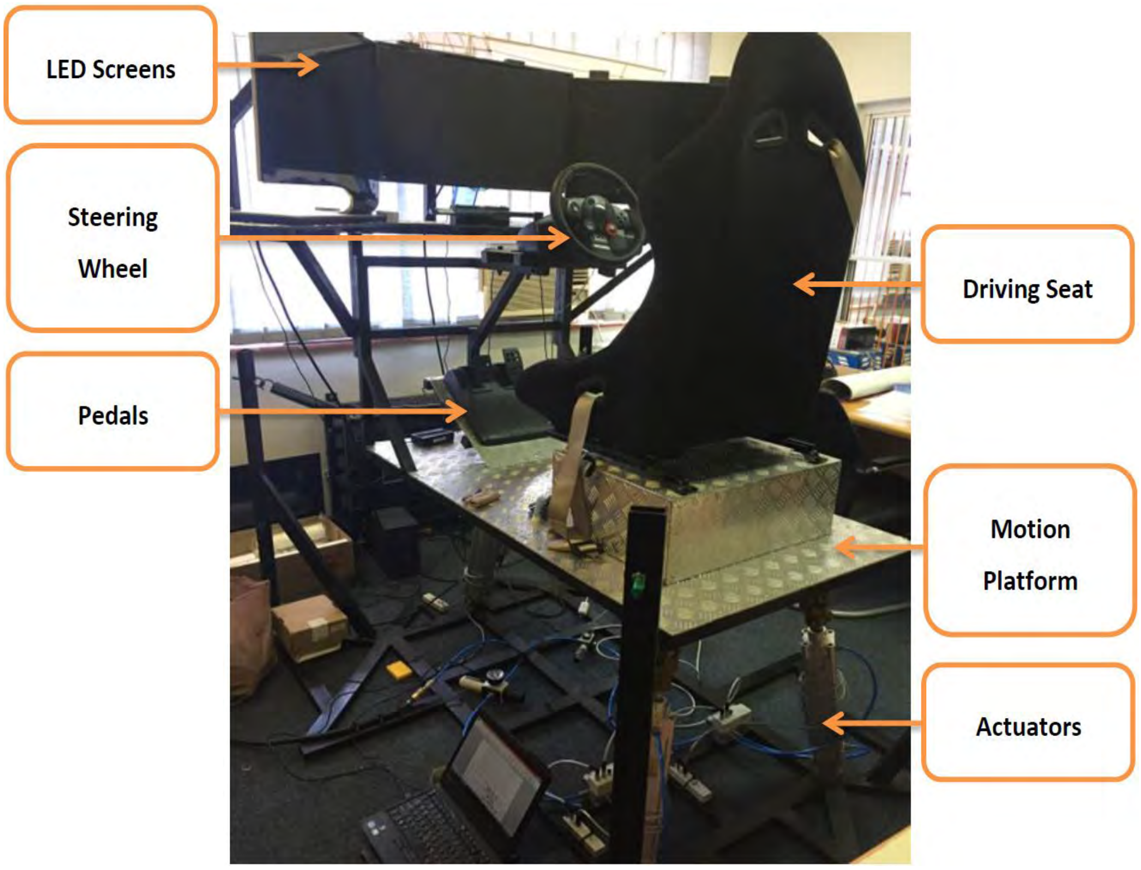


Figure 1-6 Designed Vehicle Simulator System

Applications for such a lower-cost system include human factor studies in scenarios which do not have much transient accelerations e.g. highway studies. In terms of testing car prototypes this system could be used to test adaptive cruise control systems which generally have smooth sustained accelerations. For driving training this platform can be adopted as a first contact for new drivers to provide experience, in heavy machinery systems which do not undergo severe accelerations, in general leisure environments and fuel efficiency training since proper shifting techniques do not produce too many transient acceleration signals.

1.3 Motion Cueing

Vehicle driving was a task thought to be mainly facilitated through visual information. Recent research has shown that other sensory information, such as the vestibular and proprioceptive channels also contribute to motion perception (Kemeny and Panerai, 2003).

The human being senses motion via the vestibular system. The vestibular system consists of the semi-circular canal and otolith. The otolith senses linear acceleration and the semi-circular canal senses angular velocity. Additionally the otolith senses head tilt, which is the rotation of the head relative to gravity (Kemeny and Panerai, 2003).

Figure 1-7 (Kemeny and Panerai, 2003) illustrates the human vestibular system. The semi-circular canals are in red, orange and pink; these canals sense the angular acceleration of the head. The otolith receptors in blue and green, sense both linear acceleration and tilt of the head.

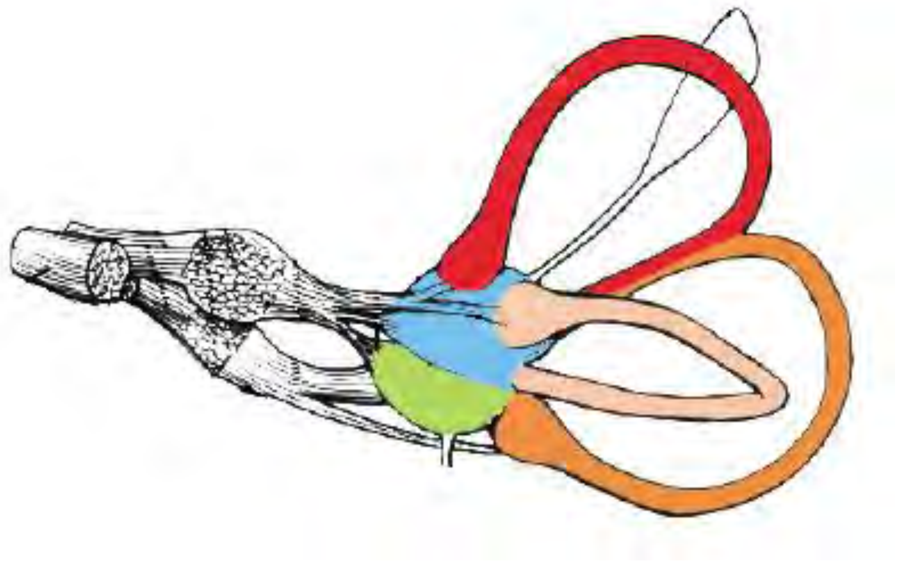


Figure 1-7 Human Vestibular System

The aim of the motion platform in a vehicle simulator is to replicate the motion sensations experienced in a real vehicle, as accurately as possible. The main issue with the motion platform is the limited workspace; this makes it difficult to recreate the motion sensations felt in a real vehicle. Various motion cueing algorithms have been developed to try and replicate as closely as possibly the sensations felt in a real vehicle (Taikui and Jianmin, 2011). The motion cueing algorithm research aims to develop techniques to exploit the human vestibular system and aid in replication of real vehicle motion sensations within the limited motion platform workspace.

Motion cueing algorithms consist of two aspects:

- Washout - Replicating of the transient accelerations is achieved by high-pass filtering; this signal is integrated to obtain a position or orientation output. To prevent actuator saturation additional high-pass filtering is added to return the platform back to neutral position (washout). This return motion should go undetected to the human vestibular system to avoid being detected as false motion cues (Reymond et al., 2000).

- Tilt coordination - Replication of sustained horizontal accelerations is achieved by first low-pass filtering the acceleration signal. Tilting of the motion platform to make use of a component of the gravity vector is then used to replicate these sustained accelerations. The rate of tilting must be done under the detectable threshold of the vestibular system to prevent false motion cues (Reymond et al., 2000).

Various types of motion cueing algorithms have been proposed in literature, these include:

- The classical washout algorithm, first proposed (Schmidt and Conrad, 1970), employs fixed parameters in the filter designs. High-pass filters are used to extract the transient components of the translational and rotational channels, the filter parameters are chosen to keep the motion platform within the workspace. A low-pass filter is used to represent the sustained translational acceleration through tilt coordination. The equations for the classical washout algorithm were initially developed (Reid and Nahon, 1985) and a method to select the filter parameters was subsequently proposed (Reid and Nahon, 1986).

- The adaptive washout algorithm is seen as an improvement to the classical washout algorithm; the filter parameters are updated in real time with the aim of minimising a cost function (Arioui et al., 2005). The classical washout algorithm suffers from false motion cues for transient motion and the adaptive algorithm was designed with false cue reduction in mind (Ariel and Sivan, 1984). A theoretical evaluation of the adaptive algorithm was done using the vestibular system model; it concluded that the adaptive algorithm provides motion sensations closer to the actual vehicle as compared to the classical washout algorithm (Taikui and Jianmin, 2011).

- The optimal washout algorithm aims to minimise the sensation errors between the physiological outputs of the vestibular system in an actual vehicle and the motion platform. An optimal control problem is developed to generate the input to the motion platform based on the input to the actual vehicle; this process is done such that the error between the outputs in the vehicle and motion platform is minimised (Sivan et al., 1982).

- Model predictive control is a model-based technique that allows the ability to include workspace constraints and to make use of future references signals. Baseggio et al. employed a technique that makes use of the detailed model of the human vestibular system and a predictive strategy based on a virtual driver (Baseggio et al., 2011).

Figure 1-8 (Beghi et al., 2012) shows an implementation of the classical washout algorithm. The linear acceleration is filtered into the transient and sustained components, using high-pass and low-pass filters respectively. The high-pass filter signal is integrated twice to give platform position and the washout filter is used to return the platform to neutral position. The low-pass filter signal is transformed via tilt coordination; this tilting is interpreted as a sustained acceleration by the human vestibular system. The angular velocity is also high-pass filtered and integrated to give an output for platform orientation. The signal is added with the tilt coordination signal to provide the final platform orientation signal.

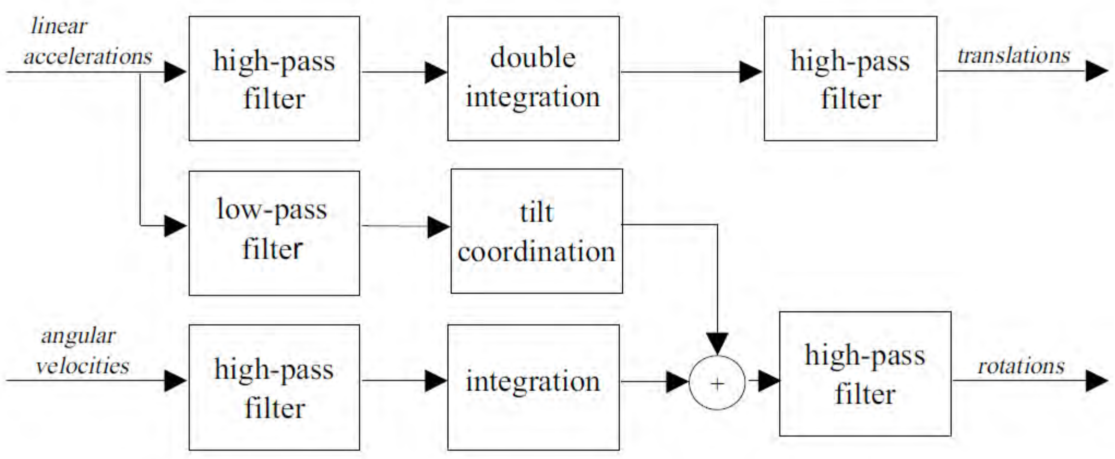


Figure 1-8 Classical Washout Algorithm

Figure 1-9 (Beghi et al., 2012) illustrates the model-predictive control scheme. The actual vehicle translational accelerations and rotational velocities are obtained from simulation software. These are scaled and the perceived accelerations, *r*, is obtained by filtering these values via the vestibular system model. This signal becomes the reference for the MPC algorithm. Using the MPC algorithm the displacement signals, *p*, are passed to the platform control system.

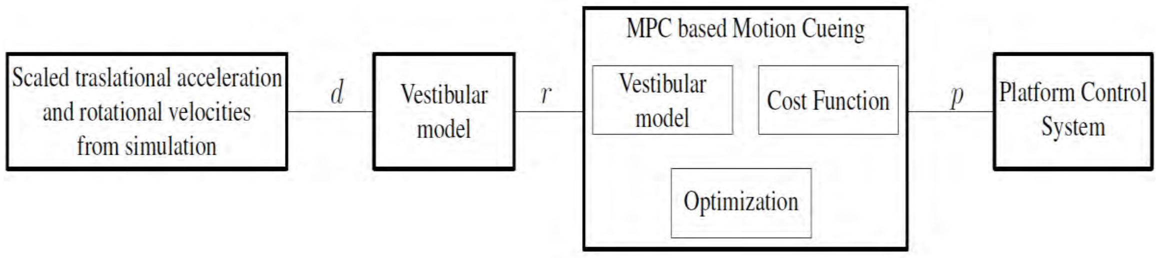


Figure 1-9 Scheme for Model Predictive Control Strategy

The MPC technique is shown to make better use of workspace, eliminate false cues and has better performance than the classical washout algorithm (Baseggio et al., 2011). The subsequent research conducted makes use of an optimisation algorithm to tune the MPC algorithm with regard to platform workspace constraints and tilt coordination (Beghi et al., 2012).

1.4 Motion Control

Motion control of parallel manipulators deals with the trajectory control of parallel manipulators. The main focus is to minimize the error between the desired end-effector position and orientation and the actual end-effector position and orientation. Motion control of parallel manipulators can be classified as either model-based control or performance based control approaches (He et al., 2007).

Traditionally PID controllers, which are performance based controllers, are often applied to the control of parallel manipulators. PID controllers are easy to implement but are known to have steady state errors. A model based control strategy, PID control with gravity compensation, was designed to mitigate the steady state errors due to gravity. This controller was shown to have faster response than the traditional PID controller and suffered from no steady state errors (Yang et al., 2008).

Results are favourable for model based control schemes however it is difficult to implement for parallel manipulators due to the high nonlinearity of parallel manipulator systems. This has led to research into performance based control strategies (He et al., 2007).

Adaptive control is a nonlinear, performance based, control strategy. It aims to identify and optimize parameters of the dynamic model online. This type of control scheme requires significant computational power. An implementation of the nonlinear adaptive control on the real-time operating system (RTOS) called XOberon was implemented and achieved better performance than a traditional linear controller (Honegger et al., 2000). The adaptive controller is able to achieve good control performance in situations with model parameter uncertainties but control performance can suffer due to unknown disturbances. An adaptive control scheme which incorporates disturbance rejection capabilities, to reject leg coupling disturbances, for a 6-DOF parallel manipulator performed well in normal and extreme conditions (Qinglong and Wenjie, 2011).

Robust control is another control strategy designed for plants with parameter uncertainties and disturbances. A 2-DOF QFT robust controller was designed in the joint space for a 6-DOF parallel robot. The single channel mathematical model of an electro-hydraulic system was defined and a robust controller with pre-filter was designed. The controller was tested and the experimental results show strong robustness against parameter variations, good disturbance rejection and precise trajectory tracking (Wu et al., 2010).

Due to the reasonable performance requirements for this vehicle simulator an industrial motion control system is used. The motion control system, by Festo, is a pneumatic system and it is designed to perform position control of each actuator leg in the 3-DOF motion platform. Figure 1-10 (Festo, 2009) illustrates the position control system for a single linear pneumatic actuator; the vehicle simulator currently has 3 position control systems for the 3 linear pneumatic actuators. Inverse kinematics is used to supply the CMAX controllers with the desired actuator stroke length, the CMAX controller then adjusts the output from the double acting directional proportional control valve (VPWP). The proportional control valve drives the linear pneumatic actuator which has a built in incremental position encoder which feeds the actual position back to the CMAX controller. This creates a closed loop feedback position control system. Each position control system has to be configured with suitable parameters to achieve good tracking performance.

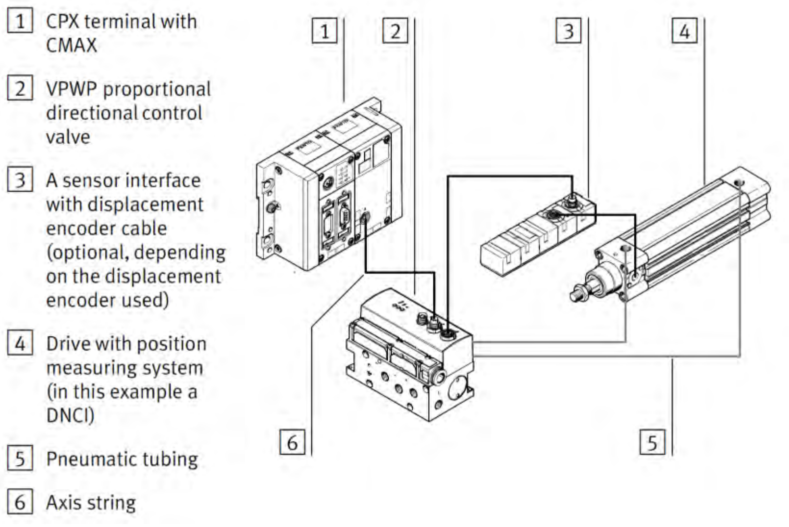

1 CPX terminal with CMAX
2 VPWP proportional directional control valve
3 A sensor interface with displacement encoder cable (optional, depending on the displacement encoder used)
4 Drive with position measuring system (in this example a DNCI)
5 Pneumatic tubing
6 Axis string
1
2
3
4
6
5

Figure 1-10 Festo Single Actuator Position Control System

1.5 Motivation for Study

The University of Kwa-Zulu Natal has a vehicle simulator, illustrated in figure 1-11, which was designed as a final year project in 2012. Currently the system only makes use of visual cues, which it receives from 3 x 27 inch LED monitors. The vehicle simulator is low-cost compared to other simulator systems mentioned in the literature review, such as the Daimler-Benz vehicle simulator, General Motors vehicle simulator and the NADS.

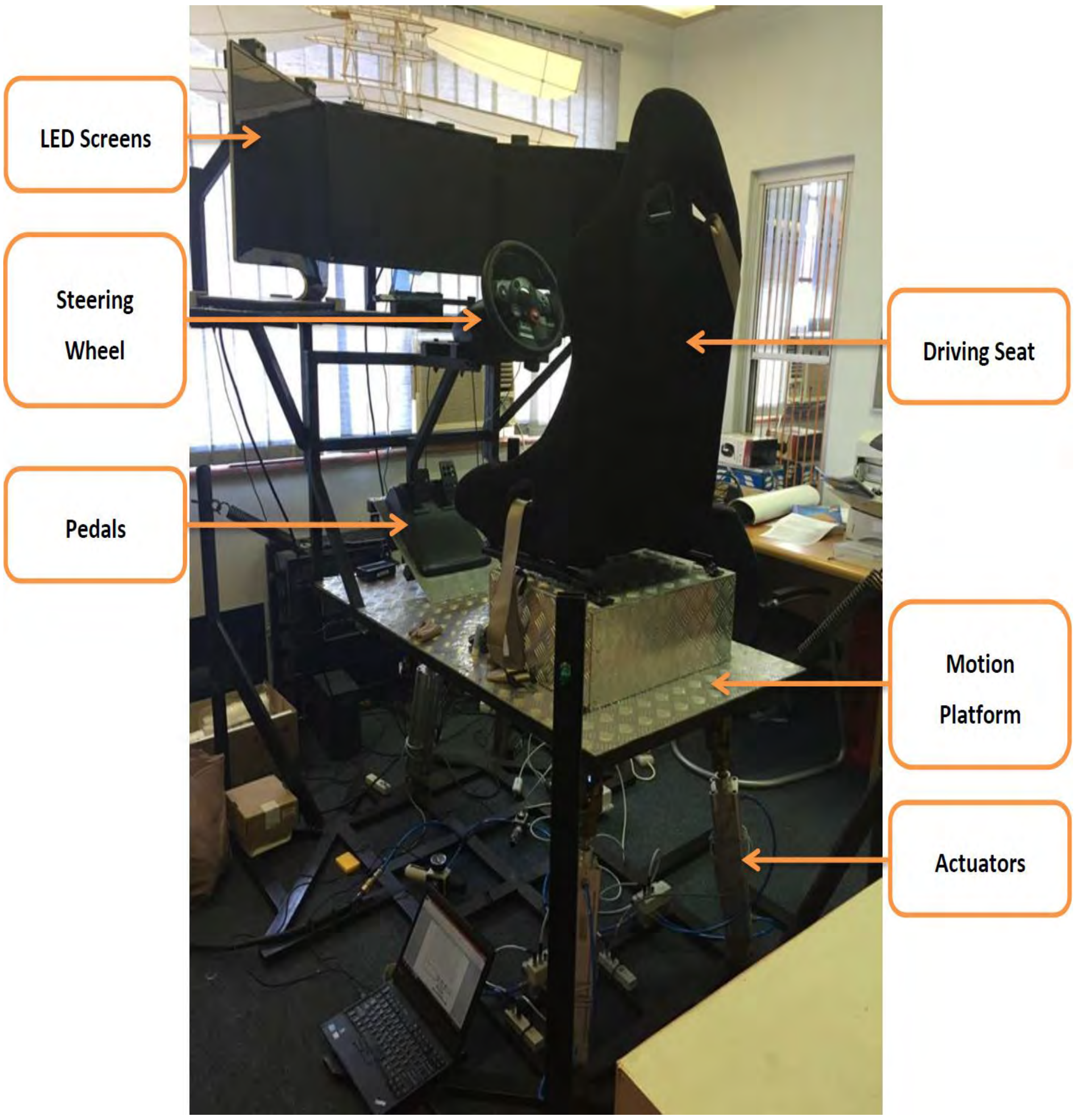


Figure 1-11 Vehicle Simulator System

The motion platform for this simulator is a 3-DOF platform as compared to the traditional 6-DOF Stewart platform. The reasons for choosing a 3-DOF system were the lower manufacturing costs involved and relatively simple manufacturing of such a system.

The motivation of this research is to research, design and implement motion cues for the 3-DOF motion platform. The aim is to create the best possible fidelity in the vehicle simulator system by creating realistic motion cues that work in cohesion with visual cues. Evaluation is performed against the traditional 6-DOF motion platform using the human vestibular system models, highlighting the benefit of such a system in certain applications.

1.6 Scientific Contribution

The research contributes the following aspects:

- A performance evaluation of the 3-DOF motion platform which was designed. The performance of the 3-DOF motion platform is evaluated against the traditional 6-DOF motion platform in the Matlab/Simulink environment. By observing the outputs of the human vestibular system models the fidelity of both systems is assessed. The results conclude on the benefit of the 3-DOF motion platform in replication of certain motion sensations and in certain applications, especially those which have cost constraints and reasonable performance requirements.

- A simulation system developed in Matlab/Simulink to aid in the design of a vehicle simulator is contributed. The position control system used, in the vehicle simulator, is a pneumatic system and comprises of 3 linear pneumatic actuators. Limits were imposed on the position, velocity and acceleration values of each actuator. These limits were imposed to guarantee safety of the user in the vehicle simulator and safety of the mechanical structure of the vehicle simulator. The simulation system developed comprises of all the aspects involved in the motion cueing process, which includes input vehicle data scaling, implementing of the washout algorithm and performing inverse kinematics for the 3-DOF motion platform. By incorporating the various motion cueing aspects in the Matlab/Simulink environment the parameters of the various aspects are varied until performance that adheres to the actuator motion limits is achieved. Using the developed simulation system also aids in ensuring the motion cueing aspects are tested before they can be implemented on the actual vehicle simulator.

1.7 Research Objectives

The research objectives are as follows:

- Investigate and understand current mechanical framework.
- Perform kinematic analysis, simulation and testing of the 3-DOF motion platform and traditional 6-DOF motion platform.
- Investigate, design, implement and test the classical washout algorithm for use in both the 3-DOF motion platform and traditional 6-DOF motion platform.

- Implement the novel simulation system, developed in Matlab/Simulink, to aid in the vehicle simulator design. This system is used to adjust various parameters in the motion cueing process to ensure actuator motion constraints are respected.
- Evaluate the performance of the 3-DOF motion platform against the traditional 6-DOF motion platform using human vestibular system models in Matlab/Simulink.
- Write software a software plugin, written in C++, to interface between the physics engine of a game and the position control system on the 3-DOF motion platform. The software plugin implements the various motion cueing aspects in the C++ language.
- Configure and test the position control system hardware.
- Develop PLC software to perform actuator position control.
- Evaluate the position control system performance to provide motion cues against the performance of the Matlab/Simulink simulation system results.
- Integrate and test the entire vehicle simulator system. Evaluate the fidelity of the vehicle simulator.

1.8 Dissertation Outline

The next chapter in the dissertation is the mechanical system design and analysis. The mechanical system is presented and the various components for the vehicle simulator are discussed. Kinematic solutions for the 3-DOF platform are derived and a simulation system is developed, in the Matlab/Simulink environment, to validate the derived inverse kinematic equations. Similar kinematic equations are presented for a known 6-DOF motion platform and Matlab/Simulink simulations are used to validate these equations.

Chapter 3 presents the Festo position control system used in the vehicle simulator for motion control. The various components in the position control system are discussed and the configuration of this position control system is explained. Setup and basic testing of the position control system is performed. The chapter concludes by explaining the PLC software algorithm and how this algorithm is used in the vehicle simulator.

Chapter 4 presents the motion cueing strategy used for the vehicle simulator. The classical washout algorithm is selected to be implemented on the vehicle simulator. An implementation of the classical washout algorithm is presented for the motion platform. Simulations are performed on the classical washout algorithm in the Matlab/Simulink environment. The simulations test the algorithm to ensure the washout process is effective. It also evaluates the ability of the classical washout algorithm to replicate the motion sensations experienced in an actual vehicle; this process makes use of the human vestibular system models.

Chapter 5 presents the software used to provide visual cues and telemetry data. The X-Sim software is used to provide visual cues and telemetry data via the games build in physics engine. A novel simulation system developed, in Matlab/Simulink, is used to adjust the motion cueing parameters to ensure motion that adheres to the actuator motion constraints; this guarantees safe performance. Fidelity of the 3-DOF motion platform is evaluated using the human vestibular system models. The 3-DOF motion platform motion cues are evaluated against the sensations felt in an actual vehicle and the 6-DOF motion platform; this testing highlights the benefits of the 3-DOF motion platform, especially in scenarios with cost constraints and reasonable performance requirements. Once this testing is complete a software plugin is developed, in C++, to interface between the X-Sim software and the Festo position control system. Motion cueing that was implemented and tested in the Matlab/Simulink environment was written in the C++ language to be used in the software plugin. The actuator stroke length outputs for the software plugin and the simulation system, in Matlab/Simulink, were also compared to ensure the C++ plugin implementation is correct.

In Chapter 6 the entire vehicle simulator is presented. The various components used to provide visual and motion cues are discussed. These visual and motion cues are integrated to create a vehicle simulator with the best possible fidelity. Position control testing with live game data is performed and this testing highlights the issue of control system instability on the back actuators. A modification is done to the 3-DOF system to provide motion cues through the front actuator only. The modified system is a partial 2-DOF system and is able to provide translational motion along the z-axis (Heave) and rotational motion along the y-axis (Pitch). The modified partial 2-DOF system is compared to the initially designed 3-DOF system and the results are favourable. The chapter is concluded by evaluating the position control system response in the vehicle simulator against the results from the Matlab/Simulink simulations.

The final chapter presents the conclusions of this research and highlights possible future work which could be undertaken.

1.9 Chapter Summary

This chapter presents research conducted in the field of vehicle simulators and the various components of these simulators.

The importance of parallel manipulators, which have a high load carrying capacity, in vehicle simulators, was discussed. The complexities and limitations of these robot manipulators were highlighted. Research developments in the field of motion cueing algorithms began due to the limited workspace of parallel manipulators; these motion cueing algorithms aims to maximise platform utilisation and provide realistic motion sensations.

The history of vehicle simulators and various commercial vehicle simulators were mentioned. It was shown that the vehicle simulator is comprised of several sub-systems and these sub-systems act in cohesion to provide high fidelity. The classification of these vehicle simulators based on cost was also presented; this showed that costing has a direct impact on fidelity of the vehicle simulator. The higher costing systems tend to provide the highest simulator fidelity. Several applications which used both higher and lower cost systems were discussed.

Research has classified motion control of parallel manipulators as either model based control or performance base control. Various types of model based control strategies were discussed; these strategies are difficult to implement due to non-linear characteristics of the parallel manipulator model. Subsequent research into performance based control highlights the benefits of this control strategy. Performance based control however requires excessive computational power. Robust control strategies were also researched and these aim to provide good tracking while rejecting disturbances. In the 3-DOF motion platform an industrial control system, which is a pneumatic system from Festo, was selected to perform motion control and this control systems components were explained.

The motivation of this study was highlighted and this aims create the best possible fidelity by creating realistic motion cues that work in cohesion with visual cues. Implementation of the various components used in the vehicle simulator system is performed on the 3-DOF motion platform. Performance of the 3-DOF motion platform is evaluated against the 6-DOF motion platform, using the human vestibular system models, to highlight the benefits of such a system.

Scientific contributions were presented for this research. The first is the performance evaluation of the 3-DOF motion platform against the traditionally used 6-DOF motion platform. Results are aimed to show the benefit of the 3-DOF system especially in applications which have cost constraints and reasonable performance requirements. The second contribution is through the novel simulation system developed which aids in the vehicle simulator design and evaluating of the selected position control system, in terms of trajectory tracking and accurate replication of motion cues. This simulation system, developed in Matlab/Simulink, ensures actuator motions that adhere to the actuator motion constraints are achieved. This guarantees safety of the user of the vehicle simulator and safety of the mechanical structure of the vehicle simulator.

The chapter concludes with the research objectives, showing the various aspects that need to be performed to create a vehicle simulator system, and dissertation outline for the rest of this dissertation.

2 Mechanical System Design and Analysis

The following chapter describes the mechanical system used for the vehicle simulator. It provides a detailed description of the 3-DOF motion platform used for the vehicle simulator and presents the kinematic analysis for this platform. Simulations are performed, in Matlab using Simulink and the SimMechanics toolbox. SimMechanics is a toolbox that provides a multibody simulation environment which allows for the modelling and simulation of mechanical systems using their geometrical layout and structural properties. It provides a simulation environment were kinematic and dynamic analysis can be performed on multibody systems (The MathWorks Inc., 2007). The SimMechanics toolbox creates a 3-D model of the 3-DOF motion platform based on the geometrical layout of the motion platform. This model can be used to simulate kinematic and dynamic analysis for the motion platform.

The simulation system developed is used verify the derived inverse kinematic equations by comparing them to the results output from the structural model of the platform created using the SimMechanics toolbox. A similar simulation system is developed for the 6-DOF motion platform and the inverse kinematic equations for this platform are also verified. The 3-DOF motion platforms performance is evaluated against the traditionally used 6-DOF motion platform in subsequent chapters.

2.1 Vehicle Simulator Framework

The University of KwaZulu-Natal has a vehicle simulator which was designed to be used in the Department of Mechanical Engineering. The simulator comprises of a mechanical framework, a 3-DOF motion platform and three linear pneumatic actuators.

Figure 2-1 illustrates the mechanical framework which was designed to support the following components for the vehicle simulator:

- 3 x 27 inch LED monitors
- Steering wheel
- Vehicle seat
- Pedals

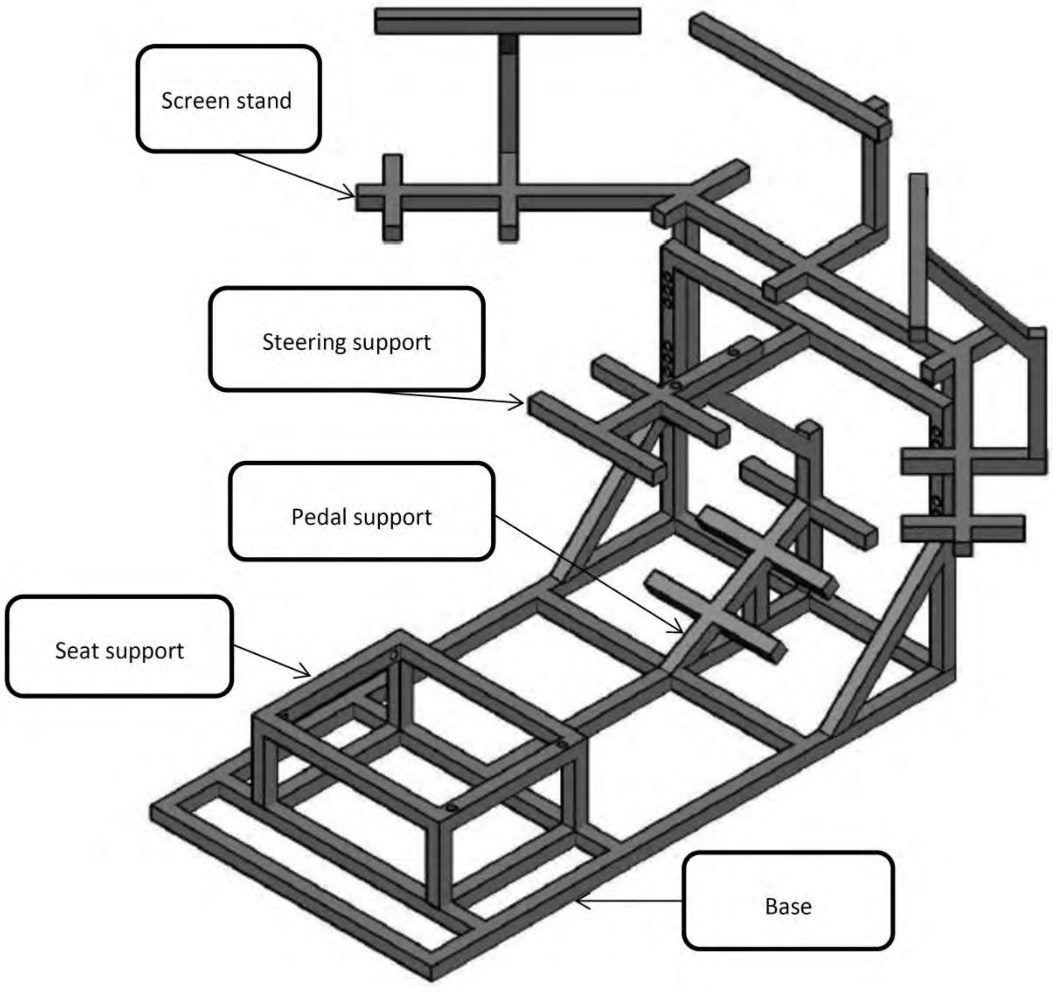


Figure 2-1 Mechanical Framework of the Vehicle Simulator

The mechanical framework was constructed from steel tubing, which is lower cost in comparison to aluminium. To reduce the mass of the structure hollow steel tubing was used. The steel tubing used is 32 mm x 32 mm x 2 mm for all members of the mechanical framework.

2.2 Vehicle Simulator Motion Platform

Figure 2-2 illustrates the vehicle simulator motion platform which was designed to be a 3-DOF platform. It is designed to provide translation motion along the z-axis (Heave) and rotational motion about the x-axis (Roll) and y-axis (Pitch) respectively.

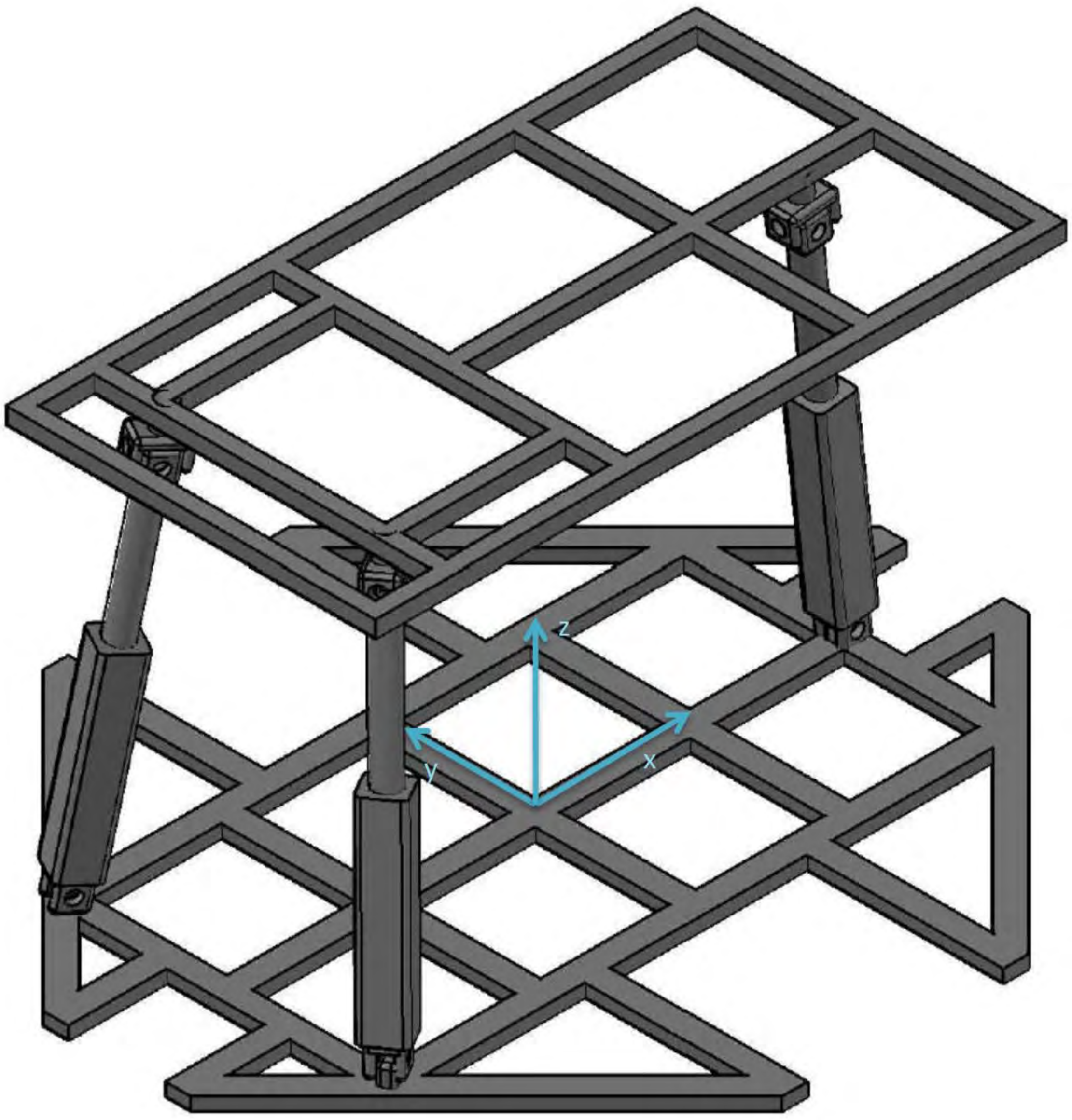


Figure 2-2 3-DOF Platform

The individual kinematic leg illustrated in figure 2-3, which forms part of the platform, is comprised of the following:

- A passive revolute joint which attaches the base to the first link.
- An actuated prismatic joint which connects the first and second link.
- A passive universal joint which attaches the second link to the moving platform.

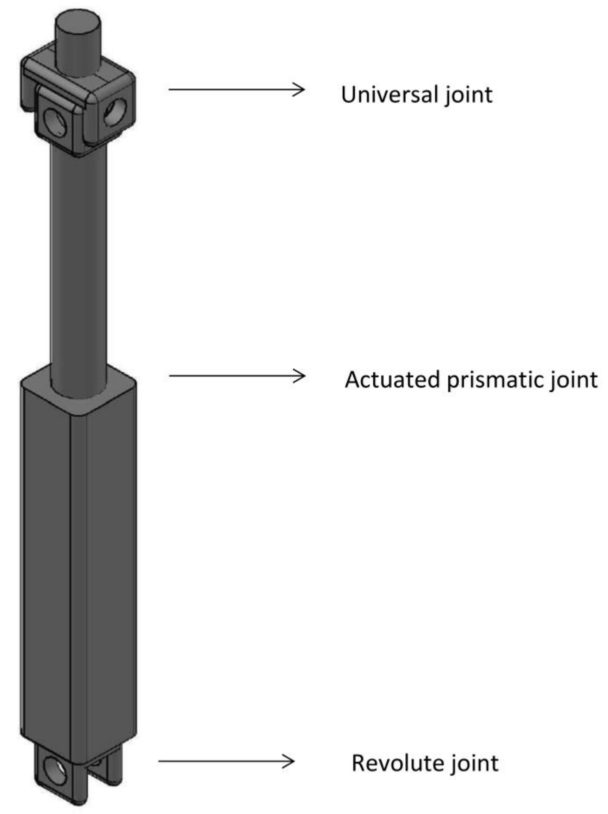


Figure 2-3 Individual Kinematic Leg for the 3-DOF Platform

2.3 Motion Platform Kinematics

The following section provides the kinematic analysis for the 3-DOF motion platform used in the vehicle simulator. A solution for the inverse kinematic problem is presented in closed-form and an iterative method is used to solve the forward kinematics.

2.3.1 Inverse Kinematics

The inverse kinematics of a robot manipulator aims to find the actuator stroke lengths for a particular end-effector position and orientation. The 3-DOF parallel manipulator motion can be specified with three independent end-effector parameters; these parameters include the translational motion along the z-axis (Heave), the rotational motion about the x-axis (Roll) and rotational motion about y-axis (Pitch) respectively.

A complete kinematic analysis of a symmetric 3-DOF parallel manipulator was initially performed (Lee and Shah, 1988). The inverse kinematic solution for the current platform being used was developed using a geometrical approach similar to the one presented for a 6-DOF Stewart platform-based robotic wrist (Nguyen et al., 1991). The aim was to develop equations which are used to find the actuator stroke lengths for a desired end-effector position and orientation. Figure 2-4 illustrates the motion platform for the vehicle simulator with the various coordinate systems used.

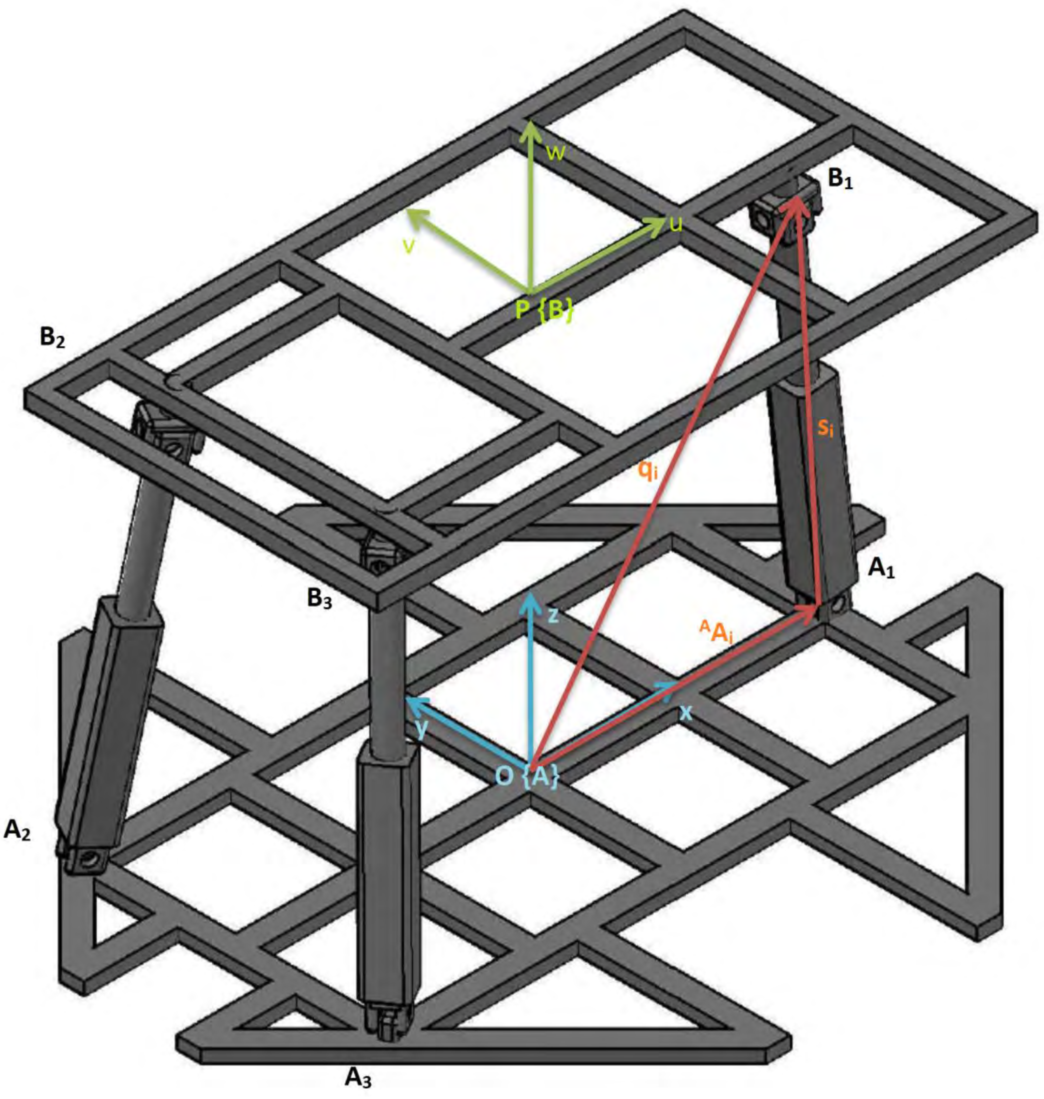


Figure 2-4 3-DOF Platform with Coordinate Systems

Coordinate frame **A**(x, y, z) is attached to the centroid, **O**, of the base of the motion platform and coordinate frame **B**(u, v, w) is attached to the centroid, **P**, of the moving platform.

The x-y plane contains revolute joints A_i, $i = 1$ to 3, and the u-v plane contains universal joints $B_i, i = 1$ to 3.

A point moving from the moving coordinate system **B**, to the fixed base coordinate system **A**, can be described fully by a translational component and a rotational component. Unit vectors **u**, **v**, **w**, are defined along the u, v, w axes of the moving coordinate system **B**.

The rotation matrix from coordinate frame **B** to coordinate frame **A** is defined as:

$$^{A}R_{B} = \begin{bmatrix} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{bmatrix} \; ... \; (2.1)$$

The rotation matrix is an orthogonal matrix and satisfies the following conditions:

$$u_x^2 + u_y^2 + u_z^2 = 1, \; ... \; (2.2)$$

$$v_x^2 + v_y^2 + v_z^2 = 1, \; ... \; (2.3)$$

$$w_x^2 + w_y^2 + w_z^2 = 1, \; ... \; (2.4)$$

$$u_x v_x + u_y v_y + u_z v_z = 0, \; ... \; (2.5)$$

$$u_x w_x + u_y w_y + u_z w_z = 0, \; ... \; (2.6)$$

$$v_x w_x + v_y w_y + v_z w_z = 0 \; ... \; (2.7)$$

The vector $^{A}A_i = [a_{ix} \; a_{iy} \; a_{iz}]^T$ is the position of the revolute joint A_i with respect to the frame **A** and vector $^{B}B_i = [b_{iu} \; b_{iv} \; b_{iw}]^T$ is the position of the universal joint B_i with respect to the frame **B**.

The three revolute joints, in the base coordinate frame **A,** are given by the following coordinates in metres:

$$^{A}A_1 = [0.6 \quad 0 \quad 0]^T \; ... \; (2.8)$$

$$^{A}A_2 = [-0.6 \quad 0.25 \quad 0]^T \; ... \; (2.9)$$

$$^{A}A_3 = [-0.6 \quad -0.25 \quad 0]^T \; ... \; (2.10)$$

The three universal joints, in the moving platform coordinate frame **B**, are given by the following coordinates in metres:

$$^{B}B_1 = [0.5 \quad 0 \quad 0]^T \; ... \; (2.11)$$

$$^{B}B_2 = [-0.5 \quad 0.15 \quad 0]^T \ \ldots \ (2.12)$$

$$^{B}B_3 = [-0.5 \quad -0.15 \quad 0]^T \ \ldots \ (2.13)$$

The position of point **P**, the centroid of the moving platform, with respect to fixed base frame **A** is given by:

$$^{A}P = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \ \ldots \ (2.14)$$

The position vector q_i of B_i with respect to coordinate frame **A** is given by the following transformation:

$$q_i = {}^{A}P + {}^{A}R_B{}^{B}B_i \ \ldots \ (2.15)$$

The coordinates of the universal joints with respect to coordinate frame **A** is given by the following:

$$q_1 = \begin{bmatrix} p_x + 0.5u_x \\ p_y + 0.5u_y \\ p_z + 0.5u_z \end{bmatrix} \ \ldots \ (2.16)$$

$$q_2 = \begin{bmatrix} p_x - 0.5u_x + 0.15v_x \\ p_y - 0.5u_y + 0.15v_y \\ p_z - 0.5u_z + 0.15v_z \end{bmatrix} \ \ldots \ (2.17)$$

$$q_3 = \begin{bmatrix} p_x - 0.5u_x - 0.15v_x \\ p_y - 0.5u_y - 0.15v_y \\ p_z - 0.5u_z - 0.15v_z \end{bmatrix} \ \ldots \ (2.18)$$

The motion of each limb is constrained by the revolute joints, which attaches the limb to the fixed base. The motion is constrained in one of the following three planes:

$$q_{1y} = 0 \text{ for } i = 1 \ \ldots \ (2.19)$$

$$q_{2y} = -\frac{0.25}{0.6} q_{2x} \text{ for } i = 2 \ \ldots \ (2.20)$$

$$q_{3y} = \frac{0.25}{0.6} q_{3x} \text{ for } i = 3 \ \ldots \ (2.21)$$

Using the above results of equation 2.16 to equation 2.21 gives the following:

$$p_y + 0.5u_y = 0 \ \ldots \ (2.22)$$

$$p_y - 0.5u_y + 0.15v_y = -\frac{0.25}{0.6}(p_x - 0.5u_x + 0.15v_x) \ \ldots \ (2.23)$$

$$p_y - 0.5u_y - 0.15v_y = \frac{0.25}{0.6}(p_x - 0.5u_x - 0.15v_x) \ \ldots \ (2.24)$$

Adding Eq. 2.23 to Eq. 2.24 gives the following motion constraint for the y-axis translational motion:

$$p_y = \frac{1}{2}u_y - \frac{0.0375}{0.6}v_x \ \ldots \ (2.25)$$

Subtracting Eq. 2.24 from Eq. 2.23 gives the following motion constraint for the x-axis translational motion:

$$p_x = 0.5u_x - 0.36v_y \ \ldots \ (2.26)$$

The Roll-Pitch-Yaw angles of orientation for the moving platform are defined as a rotation of α about the x-axis, followed by a rotation of β about the y-axis and a rotation of γ about the z-axis. The platform has two rotational degrees of freedom, a rotation about the x-axis (Roll) and a rotation about the y-axis (Pitch), implying that $\gamma = 0$ and the rotation matrix is given by:

$$^{A}R_B = R_Z(0)R_Y(\beta)R_X(\alpha) = \begin{bmatrix} cos\,\beta & sin\,\beta\, sin\,\alpha & sin\,\beta\, cos\,\alpha \\ 0 & cos\,\alpha & -sin\,\alpha \\ -sin\,\beta & \cos\beta\, sin\,\alpha & cos\,\beta\, cos\,\alpha \end{bmatrix} \ldots \ (2.27)$$

Using the above rotation matrix the three motion constraints can be expressed as follows:

$$\gamma = 0 \ \ldots \ (2.28)$$

$$p_y = -\frac{0.0375}{0.6} sin\,\beta\, sin\,\alpha \ \ldots \ (2.29)$$

$$p_x = 0.5\cos\beta - 0.36\cos\alpha \ \ldots \ (2.30)$$

From figure 2-4 the leg vector $s_i = [s_{ix}\ s_{iy}\ s_{iz}]^T$ with respect to frame **A**, is given by:

$$s_i = {}^{A}P + {}^{A}R_B{}^{B}B_i - {}^{A}A_i$$

$$= \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} + \begin{bmatrix} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{bmatrix} \begin{bmatrix} b_{iu} \\ b_{iv} \\ b_{iw} \end{bmatrix} - \begin{bmatrix} a_{ix} \\ a_{iy} \\ a_{iz} \end{bmatrix} \ldots \ (2.31)$$

For the above equation the revolute joints A_i, $i = 1$ to 3 are contained within the x-y plane, resulting in $a_{iz} = 0$. The individual leg vectors are thus given by:

$$s_1 = \begin{bmatrix} p_x + 0.5u_x - 0.6 \\ p_y + 0.5u_y \\ p_z + 0.5u_z \end{bmatrix} \;...\; (2.32)$$

$$s_2 = \begin{bmatrix} p_x - 0.5u_x + 0.15v_x + 0.6 \\ p_y - 0.5u_y + 0.15v_y - 0.25 \\ p_z - 0.5u_z + 0.15v_z \end{bmatrix} \;...\; (2.33)$$

$$s_3 = \begin{bmatrix} p_x - 0.5u_x - 0.15v_x + 0.6 \\ p_y - 0.5u_y - 0.15v_y + 0.25 \\ p_z - 0.5u_z - 0.15v_z \end{bmatrix} \;...\; (2.34)$$

The magnitude of each leg vector gives the leg length of each leg. Taking the magnitude of each leg vector gives the following leg length equations:

$$l_i = \sqrt{{s_{ix}}^2 + {s_{iy}}^2 + {s_{iz}}^2} \text{ for } i = 1 \text{ to } 3 \;...\; (2.35)$$

$$l_1 = \sqrt{(p_x + 0.5u_x - 0.6)^2 + (p_y + 0.5u_y)^2 + (p_z + 0.5u_z)^2}$$

$$= \sqrt{(p_x + 0.5\cos\beta - 0.6)^2 + {p_y}^2 + (p_z - 0.5\sin\beta)^2} \;...\; (2.36)$$

$$l_2$$
$$= \sqrt{(p_x - 0.5u_x + 0.15v_x + 0.6)^2 + (p_y - 0.5u_y + 0.15v_y - 0.25)^2 + (p_z - 0.5u_z + 0.15v_z)^2}$$

$$= \sqrt{\begin{matrix}(p_x - 0.5\cos\beta + 0.15\sin\beta\sin\alpha + 0.6)^2 + \left(p_y + 0.15\cos\alpha - 0.25\right)^2 \\ +(p_z + 0.5\sin\beta + 0.15\cos\beta\sin\alpha)^2\end{matrix}} \;...\; (2.37)$$

$$l_3$$
$$= \sqrt{(p_x - 0.5u_x - 0.15v_x + 0.6)^2 + (p_y - 0.5u_y - 0.15v_y + 0.25)^2 + (p_z - 0.5u_z - 0.15v_z)^2}$$

$$= \sqrt{\begin{matrix}(p_x - 0.5\cos\beta - 0.15\sin\beta\sin\alpha + 0.6)^2 + \left(p_y - 0.15\cos\alpha + 0.25\right)^2 \\ +(p_z + 0.5\sin\beta - 0.15\cos\beta\sin\alpha)^2\end{matrix}} \;...\; (2.38)$$

The leg length equations derived above is used to determine the actuator stroke lengths for a particular trajectory of the end-effector. These equations are dependent on the 3 independent end-effector parameters (Roll, Pitch and Heave) and the 3 constraint equations 2.28, 2.29 and 2.30.

2.3.2 Forward Kinematics

The forward kinematics for a robotic manipulator deals with finding the end-effector position and orientation for a particular set of joint variables. In general for parallel manipulators the equations for solving the forward kinematic problem are highly non-linear and in many instances no closed-form solution exists (Nguyen et al., 1991).

The technique below, used to solve the forward kinematic problem is a numerical method known as the Newton method, it is generally simpler and more computationally efficient than the exact solution (Smit, 2010). This technique has good convergence for a solution.

The system of non-linear equations can be written as a function of p_z ,α and β as follows for $i = 1$ to 3:

$$f_1(p_z,\alpha,\beta) = (p_x + 0.5\,cos\,\beta - 0.6)^2 + {p_y}^2 + (p_z - 0.5 sin\,\beta)^2 - l_1^2 = 0 \;\ldots\; (2.39)$$

$$f_2(p_z,\alpha,\beta) = \; (p_x - 0.5\,cos\,\beta + 0.15\,sin\,\beta\,sin\,\alpha + 0.6)^2 + \left(p_y + 0.15\,cos\,\alpha - 0.25\right)^2 + (p_z + 0.5\,sin\,\beta + 0.15\cos\beta\,sin\,\alpha)^2 - l_2^2 = 0 \;\ldots\; (2.40)$$

$$f_3(p_z,\alpha,\beta) = \; (p_x - 0.5\,cos\,\beta - 0.15\,sin\,\beta\,sin\,\alpha + 0.6)^2 + \left(p_y - 0.15\,cos\,\alpha + 0.25\right)^2 + (p_z + 0.5\,sin\,\beta - 0.15\cos\beta\,sin\,\alpha)^2 - l_3^2 = 0 \;\ldots\; (2.41)$$

The iterative solution for the Newton method, for p_z ,α and β is given by:

$$\begin{bmatrix} p_z \\ \alpha \\ \beta \end{bmatrix}^{n+1} = \begin{bmatrix} p_z \\ \alpha \\ \beta \end{bmatrix}^{n} - \mathrm{J}_n^{-1} \begin{bmatrix} f_1({p_z}^n,\alpha^n,\beta^n) \\ f_2({p_z}^n,\alpha^n,\beta^n) \\ f_3({p_z}^n,\alpha^n,\beta^n) \end{bmatrix} \;\ldots\; (2.42)$$

n denotes the iteration number and J_n is called the Jacobian matrix and is given by:

$$\mathrm{J}_n = \begin{bmatrix} \frac{\partial f_1({p_z}^n,\alpha^n,\beta^n)}{\partial p_z} & \frac{\partial f_1({p_z}^n,\alpha^n,\beta^n)}{\partial \alpha} & \frac{\partial f_1({p_z}^n,\alpha^n,\beta^n)}{\partial \beta} \\ \frac{\partial f_2({p_z}^n,\alpha^n,\beta^n)}{\partial p_z} & \frac{\partial f_2({p_z}^n,\alpha^n,\beta^n)}{\partial \alpha} & \frac{\partial f_2({p_z}^n,\alpha^n,\beta^n)}{\partial \beta} \\ \frac{\partial f_3({p_z}^n,\alpha^n,\beta^n)}{\partial p_z} & \frac{\partial f_3({p_z}^n,\alpha^n,\beta^n)}{\partial \alpha} & \frac{\partial f_3({p_z}^n,\alpha^n,\beta^n)}{\partial \beta} \end{bmatrix} \;\ldots\; (2.43)$$

The initial approximation for p_z ,α and β for $n = 0$, is given by:

$$\begin{bmatrix} p_z \\ \alpha \\ \beta \end{bmatrix}^{n=0} = \begin{bmatrix} \frac{(l_1+l_2+l_3)}{3} \\ 0 \\ 0 \end{bmatrix} \; ... \; (2.44)$$

For each iteration of Eq. 2.42 an improved approximation is obtained. The technique will continue to iterate until the convergence criteria is satisfied:

$$\sqrt{f_1({p_z}^n, \alpha^n, \beta^n)^2 + f_2({p_z}^n, \alpha^n, \beta^n)^2 + f_3({p_z}^n, \alpha^n, \beta^n)^2} < \varepsilon \; ... \; (2.45)$$

ε is a small positive quantity set by the user.

The technique above allows for the 3 independent platform parameters to be determined for a particular set of stroke lengths. The 3 dependent parameters can then be determined from the constraint equations 2.28, 2.29 and 2.30. These 6 parameters give the platform position and orientation.

2.4 Inverse Kinematics Simulation

The 3-DOF motion platform inverse kinematics was simulated in Matlab/Simulink. The Simulink modelling package was used to recreate the structural model of the motion platform and simulate the results. Simulink contains a toolbox, which is called SimMechanics, which provides the components used to model the motion platform. SimMechanics allows for kinematic and dynamic analysis to be performed on the designed mechanical system.

The simulation has the following objectives:

- Compare the calculated actuator stroke lengths, based on the derived leg length equation 2.36 to equation 2.38, with the actuator stroke lengths that are output from the structural model.
- Observe how well the input platform trajectory is replicated at the output of the structural model.

2.4.1 Inverse Kinematics Simulation System

The mobility criterion for the 3-DOF platform is given using the Grubler formula:

$$F = \lambda(n - j - 1) + \sum_{i=1}^{j} f_i \; ... \; (2.46)$$

Where $\lambda = 6$ for spatial manipulators, n is the number of links, j is the number of joints, f_i is the number of degrees of freedom of the ith joint. The 3-DOF platform has 3 universal joints, 3 prismatic joints and 3 revolute joints. This configuration gives the following result for the mobility criterion:

$$F = 6(8 - 9 - 1) + (3 + 3 + 6)$$

$$F = 0$$

If the same configuration is used in Matlab/Simulink it results in an overconstrained system and a simulation error when attempts are made to actuate the prismatic joints. To be able to simulate the 3-DOF motion platform the universal joints in the system are replaced by spherical joints. This configuration gives the following result for the mobility criterion:

$$F = 6(8 - 9 - 1) + (3 + 3 + 9)$$

$$F = 3$$

This configuration allows for the 3 prismatic joints to be actuated and the simulation system can function correctly. The rotational motion about the z-axis (Yaw) introduced by the spherical joints should be minimal and can be neglected because the actual 3-DOF motion platform uses universal joints.

Figure 2-5 illustrates the individual kinematic leg which is used. This model was created as a library package in Simulink to facilitate re-usability. The individual kinematic leg is made up of a revolute joint at the base, prismatic joint in the middle and spherical joint at the top. The spherical joint is used in place of the universal joint used on the actual 3-DOF motion platform.

The 3-DOF motion platform contains three individual kinematic legs; these legs connect the base to the motion platform. The prismatic joints stroke lengths are varied according to the inverse kinematic calculations. The aim is to calculate the individual stroke length of each leg for a particular platform end-effector position and orientation. The PVA block in Figure 2-5 is used to input the position, velocity and acceleration changes that each prismatic joint will undergo. A joint sensor block is attached to each prismatic joint to measure changes in stroke length.

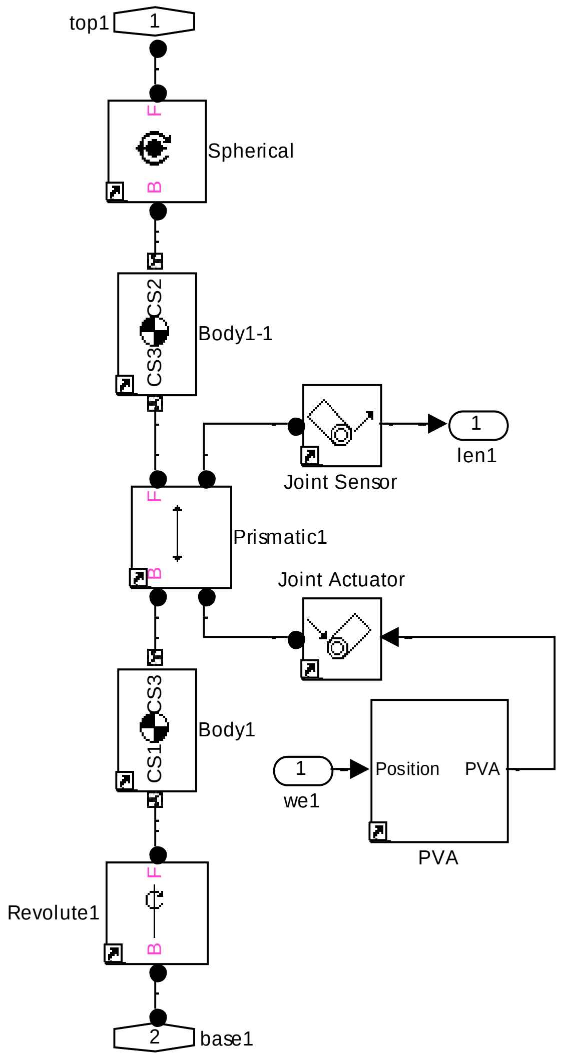


Figure 2-5 Branch Model for the Individual Kinematic Leg for the 3-DOF Platform

Figure 2-6 shows the structural model for the 3-DOF motion platform. The geometrical layout of the individual kinematic legs for the motion platform in Simulink is based on the CAD model of the actual 3-DOF motion platform. The individual kinematic legs are attached to the base via the revolute joints and to the top platform via the spherical joints.

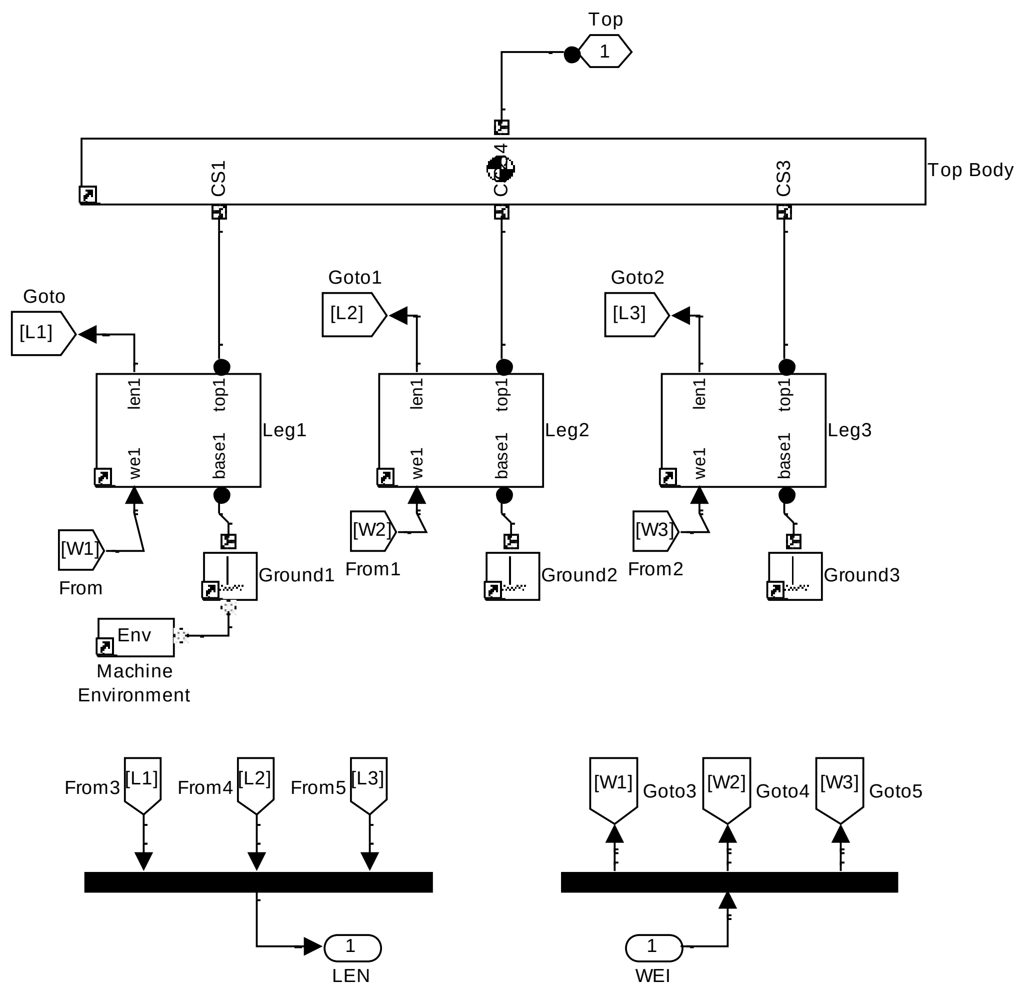


Figure 2-6 Structural Model for the 3-DOF Platform

2.4.2 Inverse Kinematics Simulation Results

In order to validate the derived leg length equation 2.36 to equation 2.38 for the inverse kinematics, a trajectory test was performed. The fundamental idea behind this test is to specify a path in time that the end-effector of the motion platform will follow. The leg lengths are then determined based on inverse kinematics equation 2.36 to equation 2.38. These leg length values are input to the system and are compared to the leg lengths from the output of the structural model of the motion platform. The end-effector output trajectory from the structural model was also compared to the input trajectory. Using the results obtained the accuracy of the derived leg length equation 2.36 to equation 2.38 for the inverse kinematics was determined.

The desired trajectory input for the end-effector of the motion platform is based on the 3 independent parameters below:

$$\alpha = 15\sin(\frac{2\pi}{3}t) \ \ldots \ (2.47)$$

$$\beta = -15\sin\left(\frac{2\pi}{3}t\right) \ \ldots \ (2.48)$$

$$p_z = 0.05\sin\left(\frac{2\pi}{3}t\right) + 0.74 \ \ldots \ (2.49)$$

The independent constraint parameters are α which specifies a rotation about the x-axis (Roll), β which specifies a rotation about the y-axis (Pitch), p_z which specifies translation motion along the z-axis (Heave). The roll and pitch angles have a sinusoidal input, with amplitude of 15 degrees and angular frequency of $\frac{2\pi}{3}$ rad/s. The heave motion starts with a 0.74 m height bias at rest. This height value is based on the structural height above the ground of the actual 3-DOF motion platform. The heave motion is varied using a sinusoidal input with maximum amplitude of 0.05 m which is added to the height bias.

Figure 2-7 illustrates the orientation of the end-effector input trajectory. It shows the sinusoidal input signal for the rotation about the x-axis (Roll) and y-axis (Pitch). There is no rotation about the z-axis (Yaw) in the system because of the constraint introduced by the universal joints used on the actual motion platform; therefore this parameter is set to zero for all instances in time.

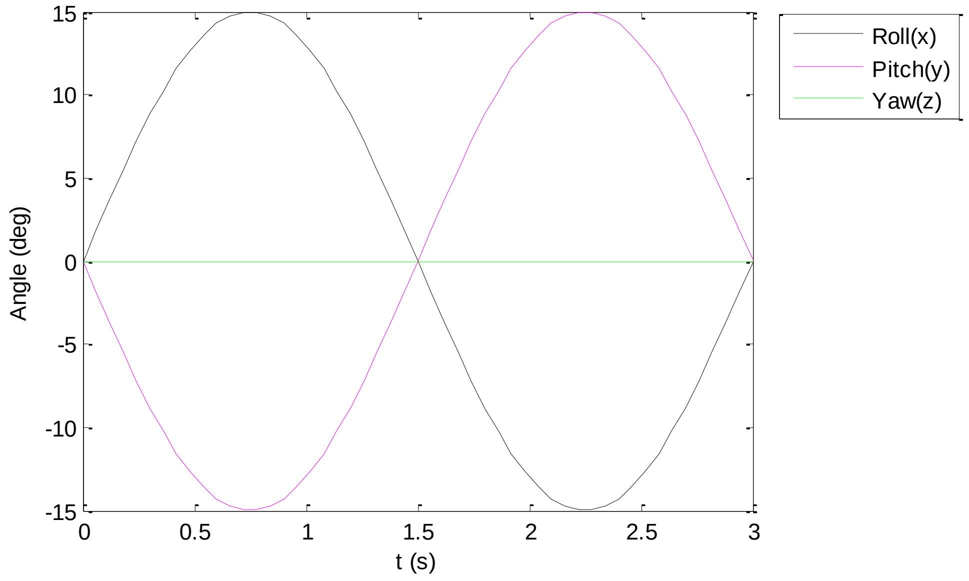


Figure 2-7 Desired Platform Orientation for the 3-DOF Platform

Figure 2-8 illustrates the position of the end-effector input trajectory. The translational motion of the end-effector is a change in motion about the z-axis (Heave). This input is a sinusoidal signal with a maximum change in height of 0.05 m. From the results it can be seen that the constraint equation 2.29 and equation 2.30 introduces constrained translational motion along the x and y axes. The x-axis motion is fairly large but does not change significantly with a minimum value of 0.1352 m and maximum value of 0.1400 m, therefore this motion will not affect the overall motion, in terms of changing actuator stroke of the motion platform significantly. Constrained motion along the y-axis is minimal and this motion is ignored.

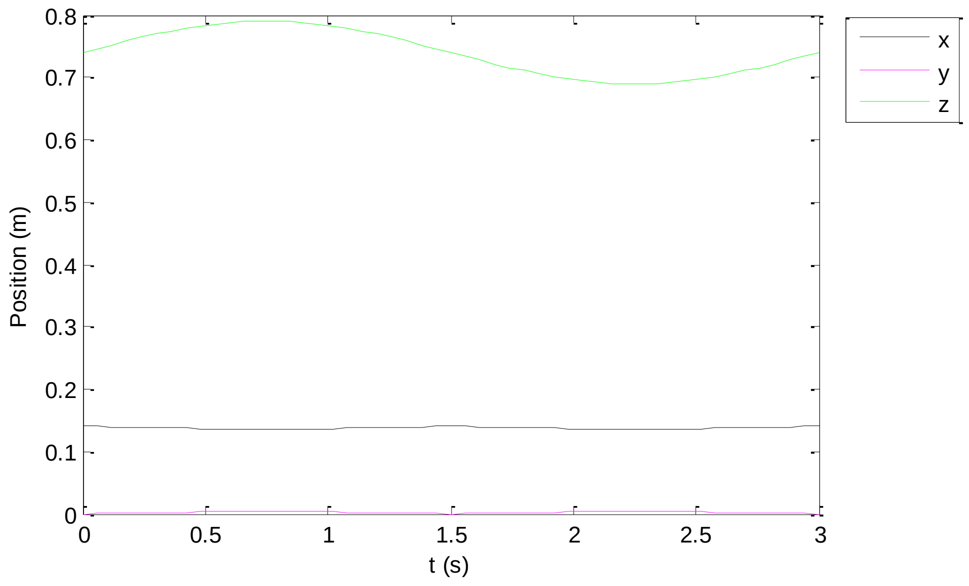


Figure 2-8 Desired Platform Position for the 3-DOF Platform

The simulation model illustrated in figure 2-9 is used to predict the motion of the end-effector. The end-effector motion from the structural model, designed using SimMechanics, should be similar to the desired input trajectory. The analysis of this result will determine the accuracy of the derived leg length equation 2.36 to equation 2.38 for the inverse kinematics.

The leg trajectory block uses the desired trajectory input equation 2.47 to equation 2.49 to determine the change in stroke length of the prismatic joints, for each individual kinematic leg, using the derived leg length equation 2.36 to equation 2.38. The changing stroke lengths of the prismatic joints are input into the structural model.

The first output of the structural model is the simulated change in stroke lengths of the prismatic joints; these values are measured using the joint sensor block attached to each prismatic joint shown in figure 2-5. The second output provides the body position and orientation for the end-effector. This output provides both the change in translational motion and change in Roll-Pitch-Yaw angles of the end-effector. The translational motion and the orientation of the end-effector should be close to the desired input trajectory equation 2.47 to equation 2.49; this result will validate the derived leg length equation 2.36 to equation 2.38 for the inverse kinematics.

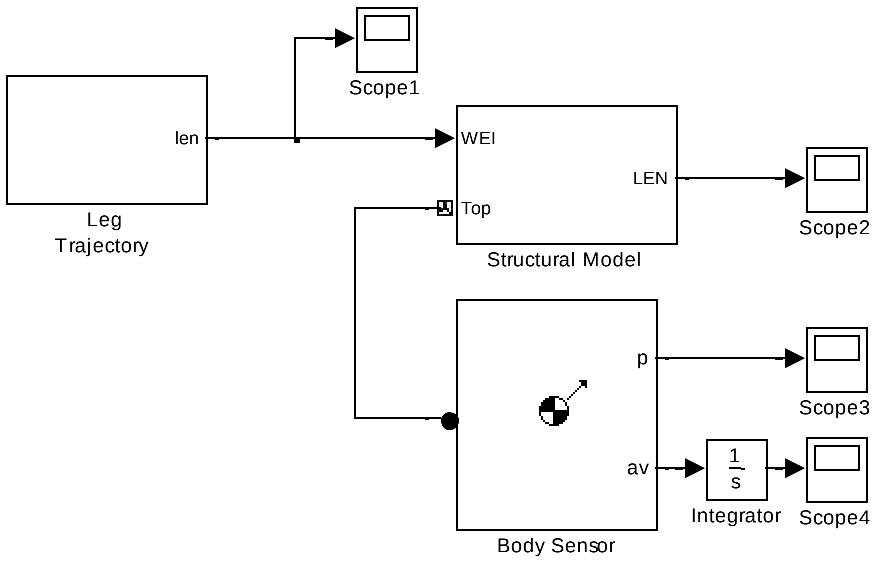


Figure 2-9 Simulation Model for the 3-DOF Platform

Figure 2-10 and figure 2-11 illustrate the outputs from Scope 1 and Scope 2. The output illustrated in figure 2-10 shows the change in stroke length of the individual kinematic legs, which is based on the derived leg length equation 2.36 to equation 2.38. The output illustrated in figure 2-11 is the change in kinematic stroke length from the output of the structural model of the motion platform. Comparing the two results it can be seen that the calculated change in stroke lengths, based on derived leg length equation 2.36 to equation 2.38, matches the change in stroke lengths of the structural model.

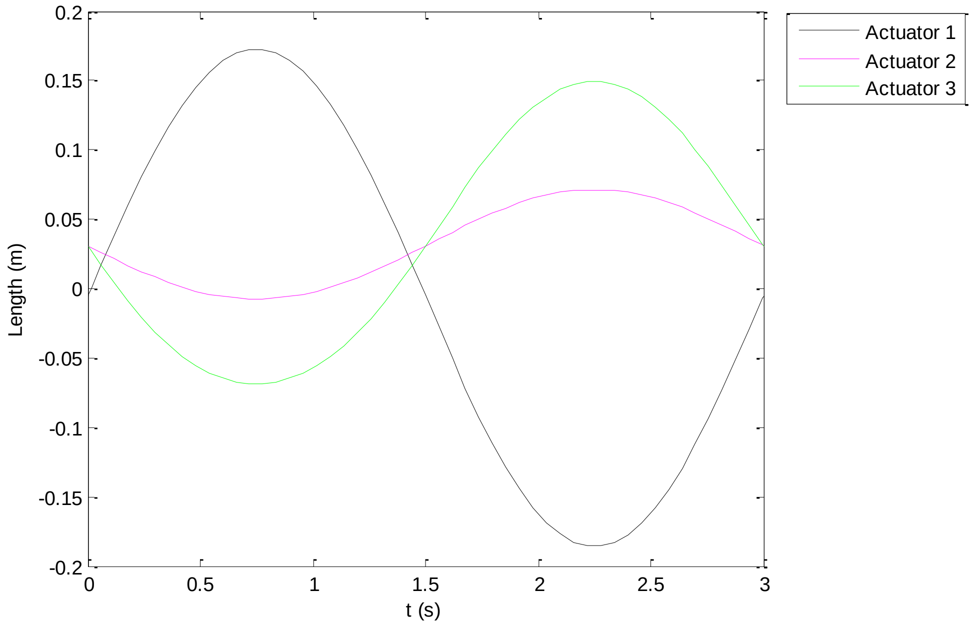


Figure 2-10 Calculated Stroke Lengths for the 3-DOF Platform

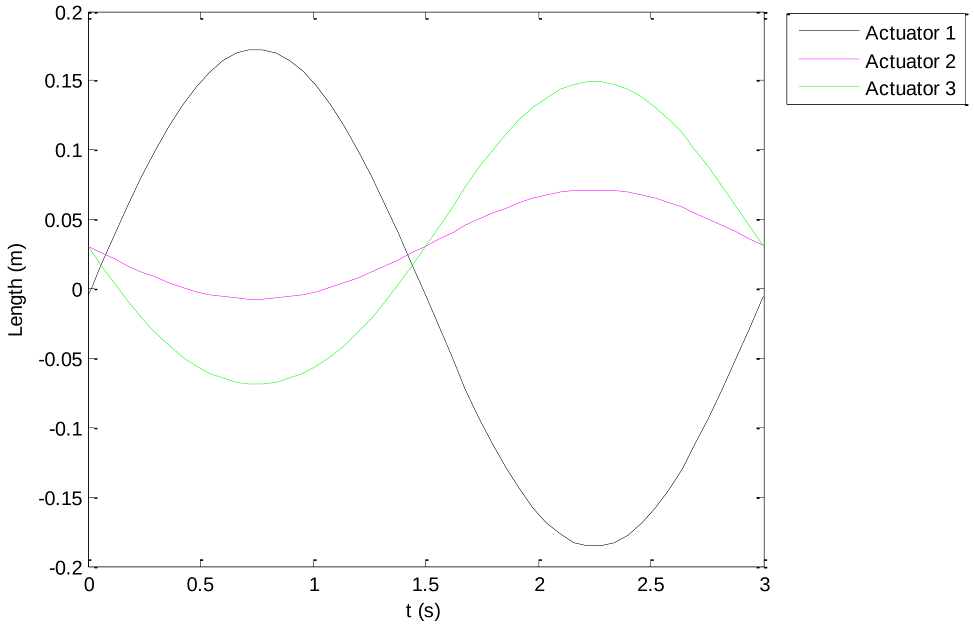


Figure 2-11 Simulated Stroke Lengths for the 3-DOF Platform

Figure 2-12 illustrates the result of the end-effector orientation output from the structural model. The orientation output for the end-effector of the structural model is reasonably similar to the orientation of the desired input trajectory equation 2.47 and equation 2.48, the sinusoidal signals for the x-axis (Roll) and y-axis (Pitch) is repeated almost identically. The rotational motion about the z-axis (Yaw) is negligible for the 3-DOF motion platform which uses universal joints in place of spherical joints; hence it can be ignored.

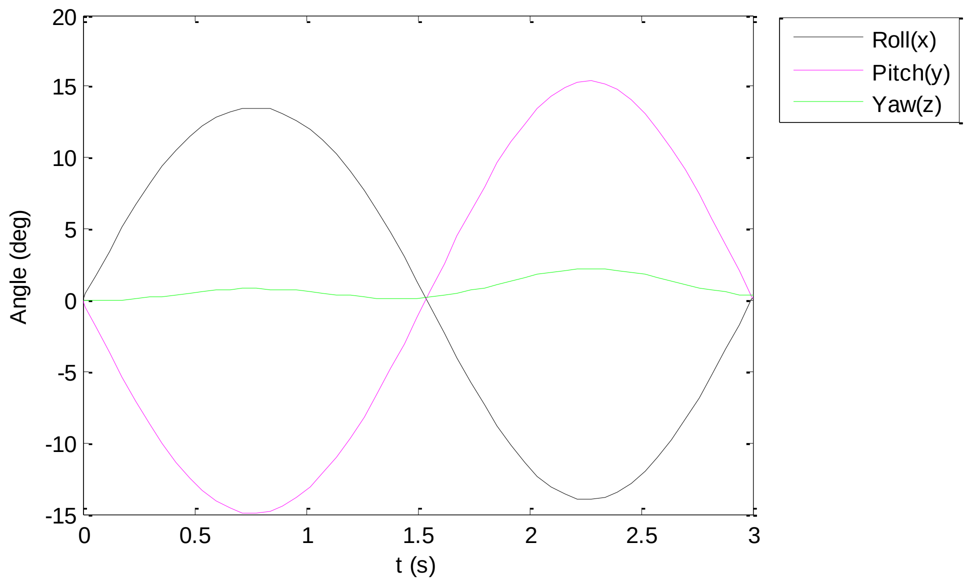


Figure 2-12 Simulated Platform Orientation for the 3-DOF Platform

Figure 2-13 illustrates the result of the end-effector position output from the structural model. The position of the end-effector of the structural model is in agreement with the position of the desired input trajectory equation 2.49; the change in translational motion along the z-axis (Heave) is in agreement with the input trajectory motion. Constrained translational motion about the x-axis is similar to the constraint motion from the input trajectory with a minimum value of 0.1206 m and a maximum value of 0.1401 m, therefore this motion will not affect the overall motion on the platform significantly, in terms of changing actuator stroke lengths. Constrained motion along the y-axis is minimal, similar to the input trajectory case, and is ignored.

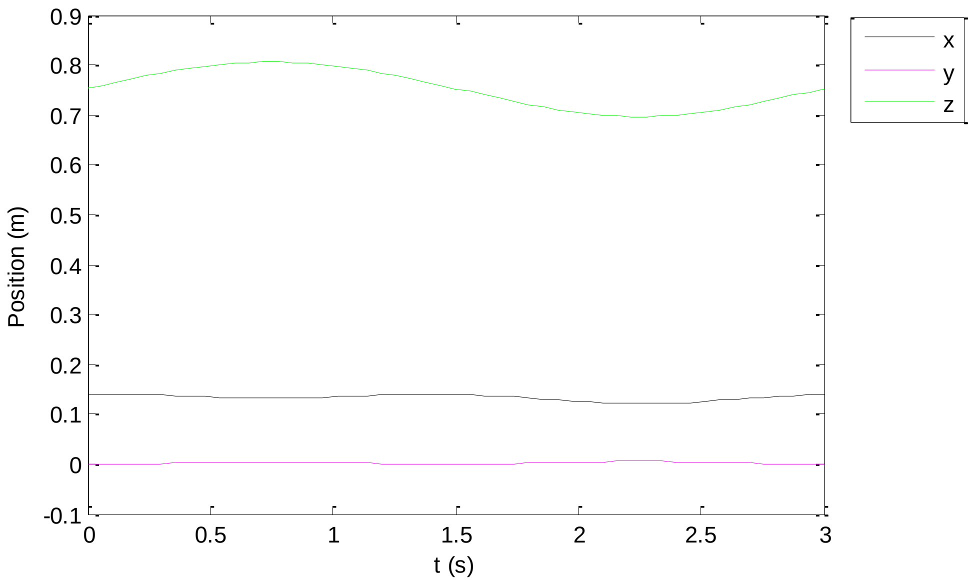


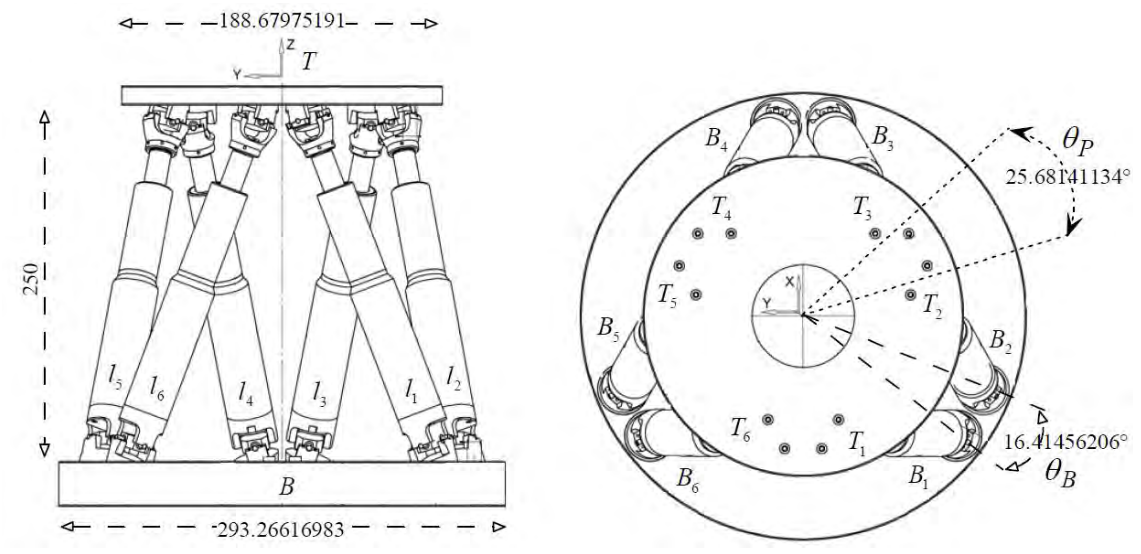
Figure 2-13 Simulated Platform Position for the 3-DOF Platform

The results show that the output trajectory from the structural model matches the required input trajectory. Actuator stroke length outputs from the structural model also match the calculated actuator stroke lengths based on the derived leg length equation 2.36 to equation 2.38. This result shows that the derived leg length equation 2.36 to equation 2.38 for the inverse kinematics is valid and acceptable to be used for the 3-DOF motion platform.

2.5 Inverse Kinematics Simulation for the 6-DOF Motion Platform

Simulations performed for the 3-DOF motion platform inverse kinematics is repeated, in Matlab/Simulink, for the 6-DOF motion platform. The aim was similarly to validate the inverse kinematic equations below which were derived previously (Bingul and Karahan, 2012).

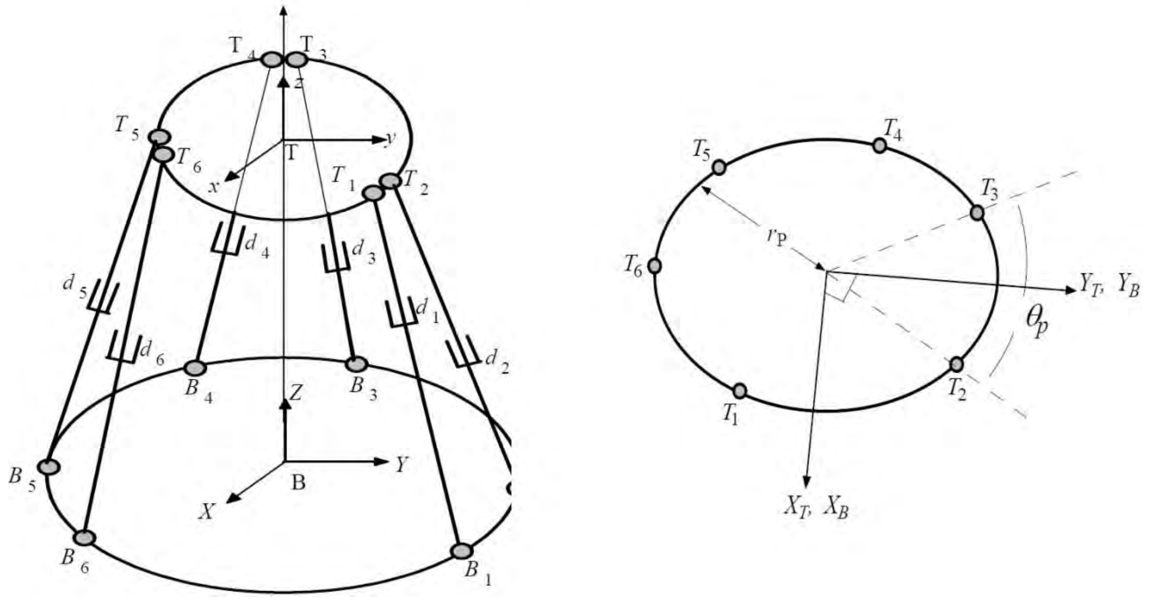
Figure 2-14 (Bingul and Karahan, 2012) illustrates the 6-DOF motion platform geometrical layout which is used. θ_p represents the angle between top joints (T_2 and T_3 , T_4 and T_5 , T_1 and T_6) and θ_b represents the angle between bottom joints (B_1 and B_2 , B_3 and B_4 , B_5 and B_6).



188.67975191
250
293.26616983
T
B
l_1 l_2 l_3 l_4 l_5 l_6
B_1 B_2 B_3 B_4 B_5 B_6
T_1 T_2 T_3 T_4 T_5 T_6
θ_P
25.68141134°
16.41456206°
θ_B

Figure 2-14 6-DOF Platform Model

Figure 2-15 (Bingul and Karahan, 2012) illustrates the 6-DOF motion platform with joint and coordinate system labels.



T_1 T_2 T_3 T_4 T_5 T_6
d_1 d_2 d_3 d_4 d_5 d_6
B_1 B_3 B_4 B_5 B_6
T
B
x y z
X Y Z
r_P
θ_p
Y_T, Y_B
X_T, X_B

Figure 2-15 6-DOF Platform with Joint and Coordinate System Labels

Equation 2.50 to equation 2.54, which is used here for the 6-DOF motion platform inverse kinematics, was derived previously (Bingul and Karahan, 2012). The top universal joints in the motion platform are represented by the following coordinates:

$$GT_i = \begin{bmatrix} GT_{xi} \\ GT_{yi} \\ GT_{zi} \end{bmatrix} = \begin{bmatrix} r_p\cos(\lambda_i) \\ r_p\sin(\lambda_i) \\ 0 \end{bmatrix} \begin{cases} \lambda_i = \frac{i\pi}{3} - \frac{\theta_p}{2} & i = 1,3,5 \\ \lambda_i = \lambda_{i-1} + \theta_p & i = 2,4,6 \end{cases} \quad ... \text{(2.50)}$$

with r_p the radius of the moving platform.

The bottom universal joints are represented by the following coordinates:

$$B_i = \begin{bmatrix} B_{xi} \\ B_{yi} \\ B_{zi} \end{bmatrix} = \begin{bmatrix} r_{base}\cos(\nu_i) \\ r_{base}\sin(\nu_i) \\ 0 \end{bmatrix} \begin{cases} \nu_i = \frac{i\pi}{3} - \frac{\theta_b}{2} & i = 1, 3, 5 \\ \nu_i = \nu_{i-1} + \theta_b & i = 2, 4, 6 \end{cases} \quad \text{. . . (2.51)}$$

with r_{base} the radius of the fixed base.

The rotation matrix for the 6-DOF motion platform is given by:

$$^{B}R_T = R_z(\gamma)R_Y(\beta)R_X(\alpha) = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \quad \text{. . . (2.52)}$$

$$= \begin{bmatrix} \cos\beta\cos\gamma & \cos\gamma\sin\alpha\sin\beta - \cos\alpha\sin\gamma & \sin\alpha\sin\gamma + \cos\alpha\cos\gamma\sin\beta \\ \cos\beta\sin\gamma & \cos\alpha\cos\gamma + \sin\alpha\sin\beta\sin\gamma & \cos\alpha\sin\beta\sin\gamma - \cos\gamma\sin\alpha \\ -\sin\beta & \cos\beta\sin\alpha & \cos\alpha\cos\beta \end{bmatrix}$$

The position of the centroid of the moving platform is given by:

$$P = [P_x \quad P_y \quad P_z]^T \quad \text{. . . (2.53)}$$

The leg length equations (inverse kinematics) for each leg are given by:

$$l_i = \sqrt{\begin{matrix} \left(P_x - B_{xi} + GT_{xi}r_{11} + GT_{yi}r_{12}\right)^2 \\ +\left(P_y - B_{yi} + GT_{xi}r_{12} + GT_{yi}r_{22}\right)^2 \\ +\left(P_z + GT_{xi}r_{31} + GT_{yi}r_{32}\right)^2 \end{matrix}} \quad \text{for } i = 1 \text{ to } 6 \quad \text{. . . (2.54)}$$

Figure 2-16 illustrates the individual kinematic leg used for the 6-DOF motion platform. It consists of a passive universal joint connecting the base to the lower leg, an actuated cylindrical joint that connects the lower leg to the upper leg and a passive universal joint that connects the upper leg to the top platform. As in the 3-DOF motion platform case the input to the joint actuator is the PVA block; this block inputs the position, velocity and acceleration that the cylindrical joint will undergo. The joint sensor block is attached to each cylindrical joint to measure changes in stroke length.

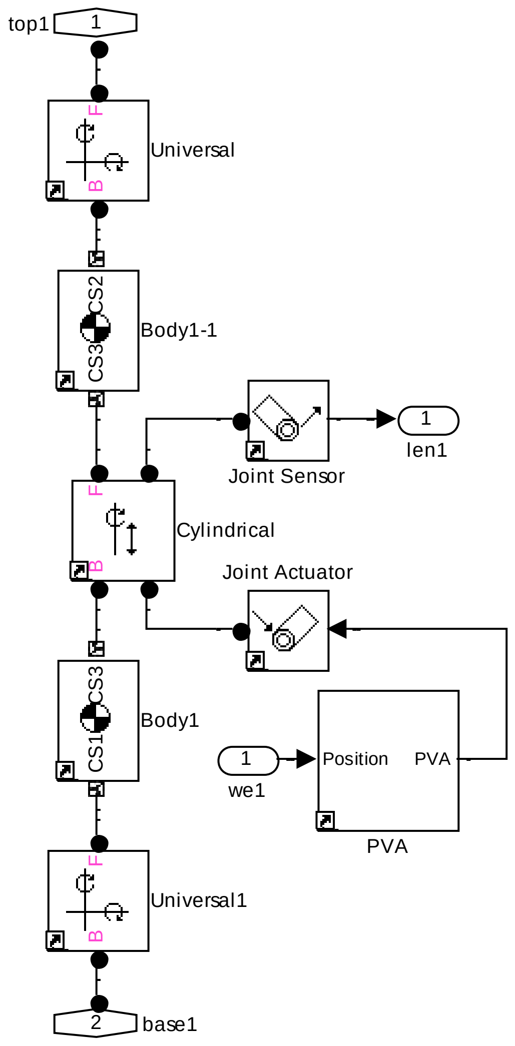


Figure 2-16 Branch Model for the Individual Kinematic Leg for the 6-DOF Platform

Figure 2-17 illustrates the structural model for the 6-DOF motion platform. The individual kinematic legs are attached to the base and top platform by universal joints. The geometric configuration is based on the standard 6-DOF platform layout used previously (Bingul and Karahan, 2012).

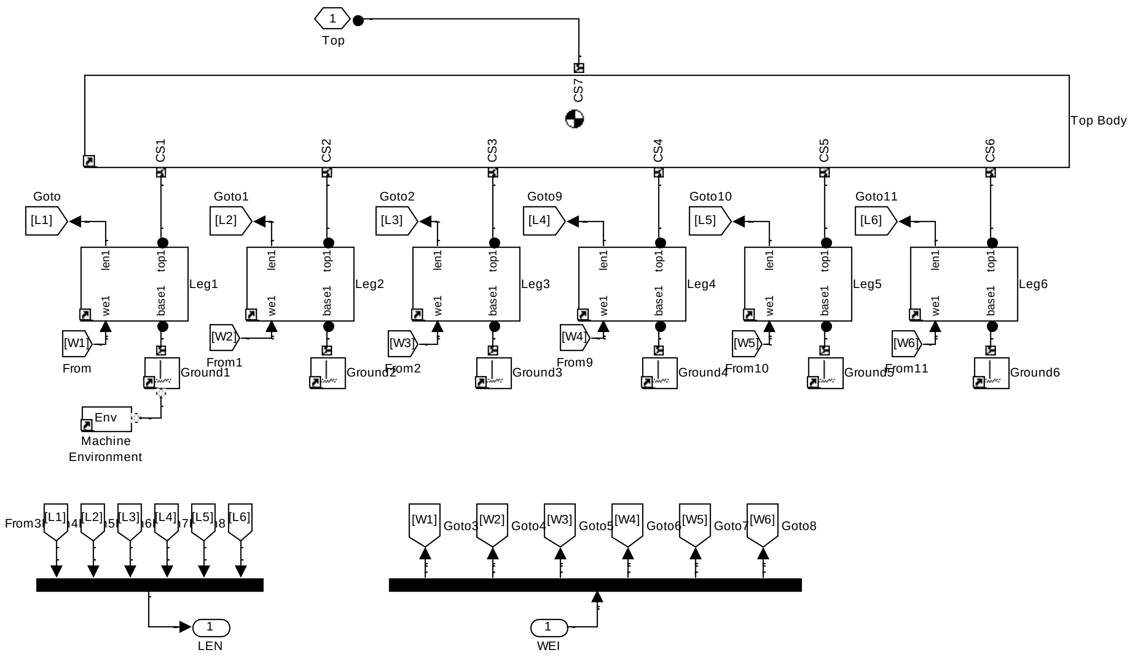


Figure 2-17 Structural Model for the 6-DOF Platform

The input trajectory for the 6-DOF motion platform uses the desired trajectory input equation 2.47 to equation 2.49 used for the 3-DOF motion platform, Eq. 2.49 uses a height bias of 2.5 m; additionally the following movements are added:

$$\gamma = 5\sin(\frac{2\pi}{3}t) \ \ldots \ (2.55)$$

$$p_x = 0.05\sin(\frac{2\pi}{3}t) \ \ldots \ (2.56)$$

$$p_y = 0.10\sin(\frac{2\pi}{3}t) \ \ldots \ (2.57)$$

The 3 additional parameters are added for the 6-DOF motion platform since these parameters are also independent constraint parameters. γ specifies a rotation about the z-axis (Yaw) with amplitude of 5 degrees and angular frequency of $\frac{2\pi}{3}$ rad/s. p_x specifies translational motion along the x-axis (Surge) with amplitude of 0.05 m and angular frequency of $\frac{2\pi}{3}$ rad/s. p_y specifies translational motion along the y-axis (Sway) with amplitude of 0.10 m and angular frequency of $\frac{2\pi}{3}$ rad/s.

Figure 2-18 illustrates the orientation of the end-effector for the 6-DOF motion platform input trajectory. It shows the sinusoidal input signals for rotation about the x-axis (Roll), y-axis (Pitch) and z-axis (Yaw). These signals are the orientation desired to be replicated by the structural model.

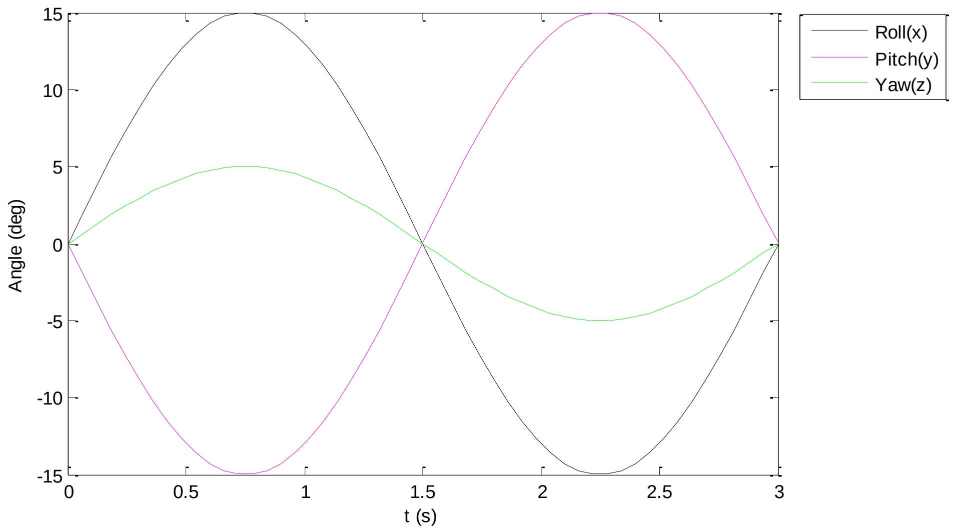


Figure 2-18 Desired Platform Orientation for the 6-DOF Platform

Figure 2-19 illustrates the position of the end-effector for the 6-DOF motion platform input trajectory. Translational motion along the x-axis (Surge) is a sinusoidal signal with a maximum change in motion of 0.05 m. Translational motion along the y-axis (Sway) is a sinusoidal signal with a maximum change in motion of 0.10 m. The translational motion about the z-axis (Heave) is a sinusoidal signal with a maximum change in height of 0.05 m. This signal starts from a height bias of 2.5 m, which represents the 6-DOF motion platforms height above the ground.

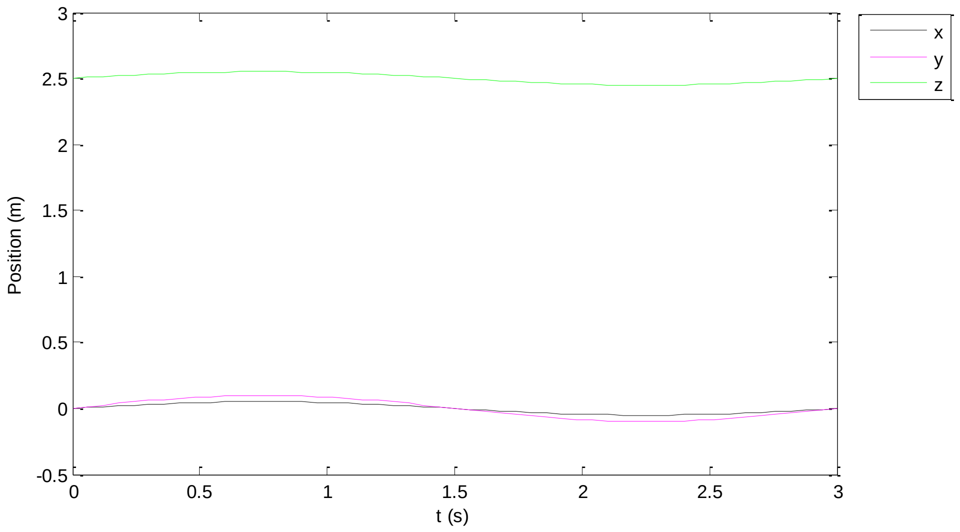


Figure 2-19 Desired Platform Position for the 6-DOF Platform

The simulation model illustrated in figure 2-20 is used to verify the inverse kinematics Eq. 2.54 for the 6-DOF motion platform. The leg trajectory block uses the desired trajectory input equation 2.47 to equation 2.49 and equation 2.55 to equation 2.57 to output the change in stroke lengths of the cylindrical joints. These values are based on the inverse kinematic equations for the 6-DOF motion platform. Changes in the stroke lengths of the cylindrical joints are input into the structural model for the 6-DOF motion platform.

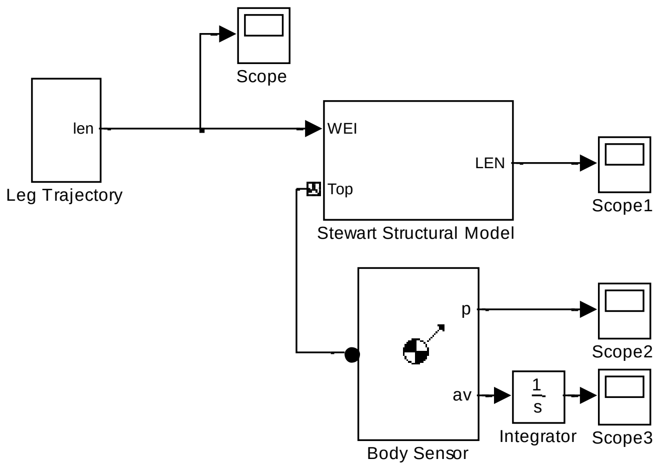


Figure 2-20 Simulation Model for the 6-DOF Platform

Figure 2-21 and figure 2-22 illustrate the outputs from Scope and Scope 1 respectively. Figure 2-21 illustrates the change in stroke lengths of the individual kinematics legs, which is based on the inverse kinematics Eq. 2.54 for the 6-DOF motion platform. Figure 2-22 illustrates the change in stroke lengths from the output of the structural model of the 6-DOF motion platform. It can be seen that the calculated change in stroke lengths, based on the inverse kinematics Eq. 2.54 for the 6-DOF motion platform, is in agreement with the change in stroke length output from the structural model.

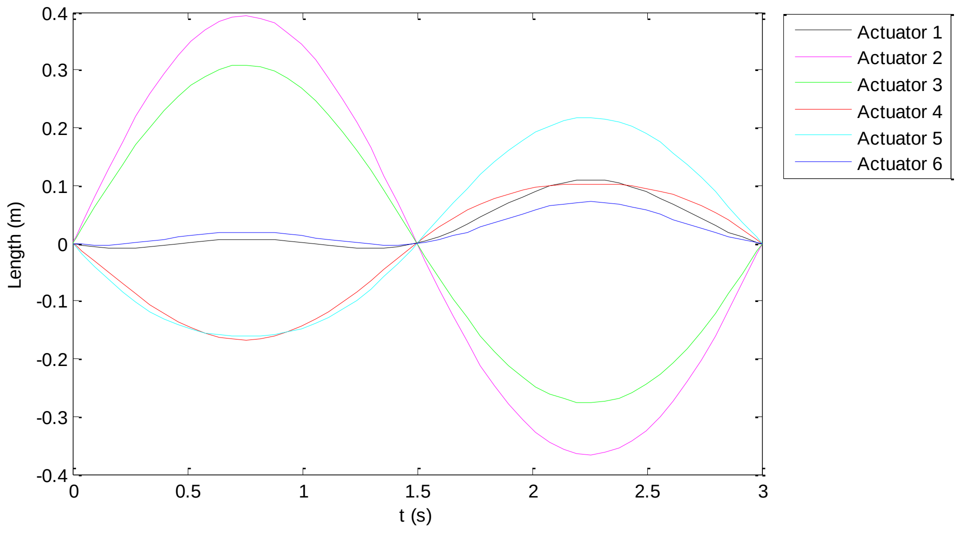


Figure 2-21 Calculated Stroke Lengths for the 6-DOF Platform

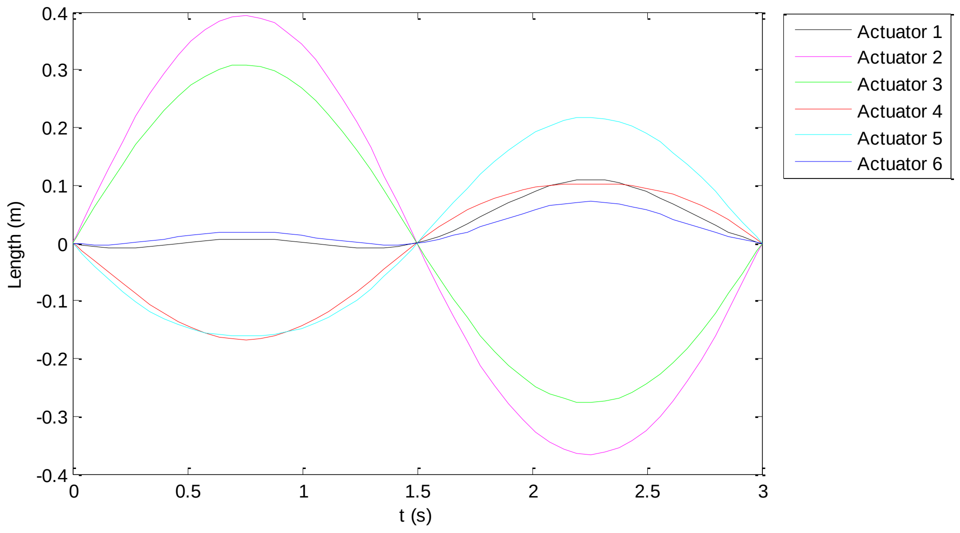


Figure 2-22 Simulated Stroke Lengths for the 6-DOF Platform

Figure 2-23 illustrates the end-effector orientation output from the structural model of the 6-DOF motion platform. The orientation output for the end-effector of the structural model is similar to the orientation of the desired input trajectory equation 2.47 to equation 2.48 and equation 2.55.

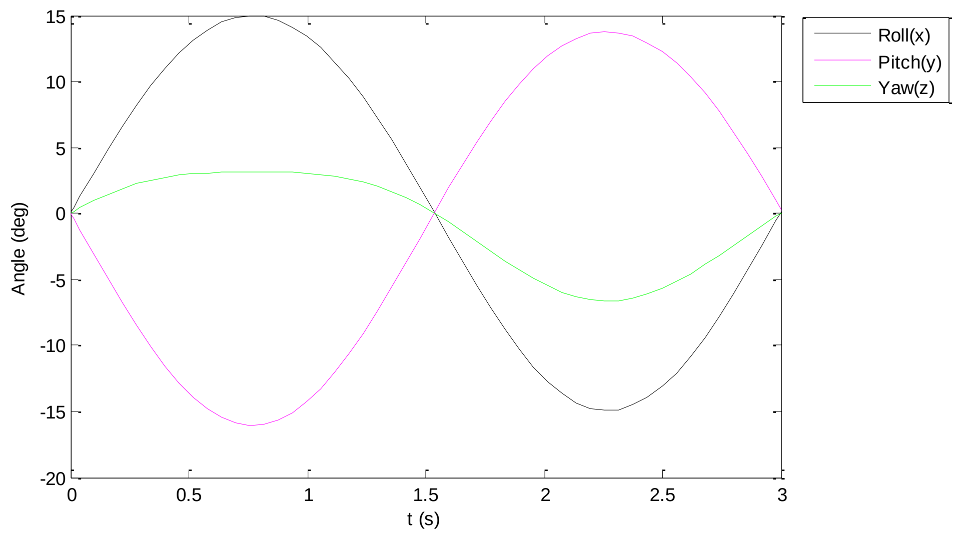


Figure 2-23 Simulated Platform Orientation for the 6-DOF Platform

Figure 2-24 illustrates the end-effector position output from the structural model for the 6-DOF motion platform. The position of the end-effector output from the structural model is in agreement with the position of the desired input trajectory equation 2.49 and equation 2.56 to equation 2.57.

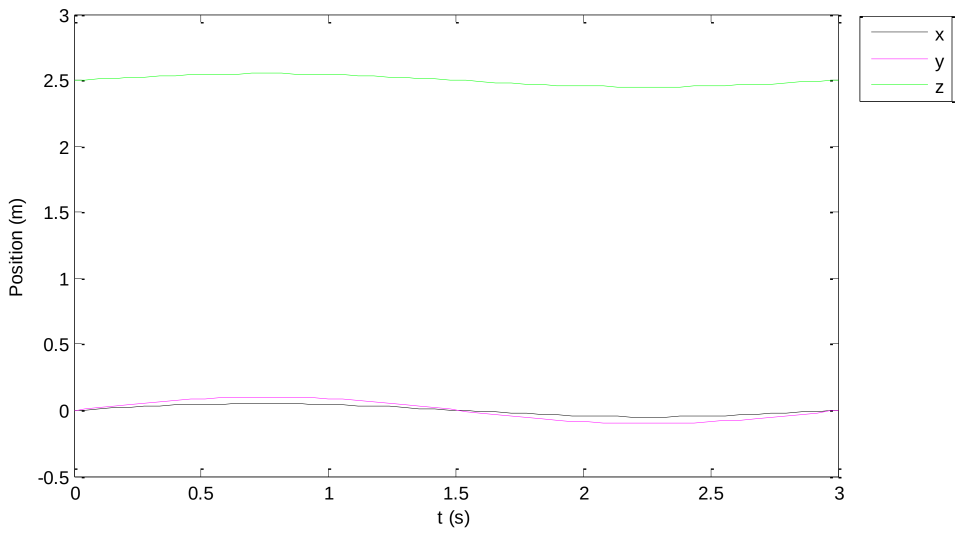


Figure 2-24 Simulated Platform Position for the 6-DOF Platform

The result shows that the output trajectory from the structural model for the 6-DOF motion platform follows the desired input trajectory. Actuator stroke lengths output from the structural model for the 6-DOF motion platform matches the calculated actuator stroke lengths based on the inverse kinematics Eq. 2.54 for the 6-DOF motion platform. Results indicate that the structural model is able to replicate the inverse kinematics and that the inverse kinematics Eq. 2.54 is acceptable to be used.

2.6 Chapter Summary

The mechanical system for the vehicle simulator was discussed, highlighting and describing the various components that the vehicle simulator is comprised of. The closed form solution for the inverse kinematics of the 3-DOF motion platform used for the vehicle simulator was presented in detail. This solution highlighted the fact that at any time the actuator stroke lengths can be determined using just 3 independent parameters (Roll, Pitch and Heave) and constraint equations 2.28, 2.29 and 2.30. In the next section the iterative solution for the forward kinematic was presented; this solution is known as the Newton method. The method is simpler and more computational efficient than an exact solution to the forward kinematics problem. Simulations were then performed, in Matlab/Simulink, to verify the derived leg length equation 2.36 to equation 2.38 for the inverse kinematics. SimMechanics was used to create the structural model of the 3-DOF motion platform using the geometrical structure of the platform. The simulation results, output from the structural model, showed that the change in stroke lengths for the actuators in both the derived leg length case and the structural model output were in agreement. It was also shown that the output for the platform end-effector trajectory is similar to the desired input trajectory equation 2.47 to equation 2.49. Based on this result the derived leg length equation 2.36 to equation 2.38 for the inverse kinematics is accepted.

A similar setup was developed for the traditional 6-DOF motion platform. It was shown that the inverse kinematics Eq. 2.54, which was derived previously (Bingul and Karahan, 2012), is acceptable to be used for this motion platform. The 3-DOF motion platform, used for the vehicle simulator, is evaluated against the 6-DOF motion platform in the chapters to follow. This study will highlight the benefits of the 3-DOF motion platform in replication of certain motion cues and this platforms application in certain scenarios.

3 Motion Control System

This chapter presents the motion control system used in the vehicle simulator system. It explains the Festo position control system used to perform position tracking for each actuator in the system. The various components used are described and details of how these components interact to perform position control are discussed. Setup with parameters used for each of the actuators in the motion control system is then presented. Testing of each actuator in the system is performed and the performance of the position control system tracking is analysed.

The PLC software algorithm is then explained, this algorithm provides the link between the X-Sim Convertor software plugin output, explained in chapter 5, and the Festo position control system. The data transfer, processing and transmission is explained in detail. This algorithm shows how the various aspects involved in the vehicle simulator are integrated to achieve the desired performance.

3.1 Motion Control System Overview

Figure 3-1 illustrates the various components of the position control system for a single linear pneumatic actuator. The detailed hardware architecture and electrical schematic is attached in appendix A and appendix B respectively. The PLC device is the programmable device which is used to obtain the transferred position data from the X-Sim Universal Serial Output (USO) interface, transmitted via UDP, and transmit this position data to each of the 3 axis controllers. The axis controller is the device which performs the control system tracking by controlling the double acting directional proportional control valve. The linear drive provides position feedback through the position sensor interface; this position feedback is transferred to the proportional control valve and through to the axis controller. Based on the difference between the required position value and actual position value the axis controller controls the direction and flow of air output from the double acting directional proportional control valve. This control signal causes the output linear drive position to track the required input position, achieving feedback control.

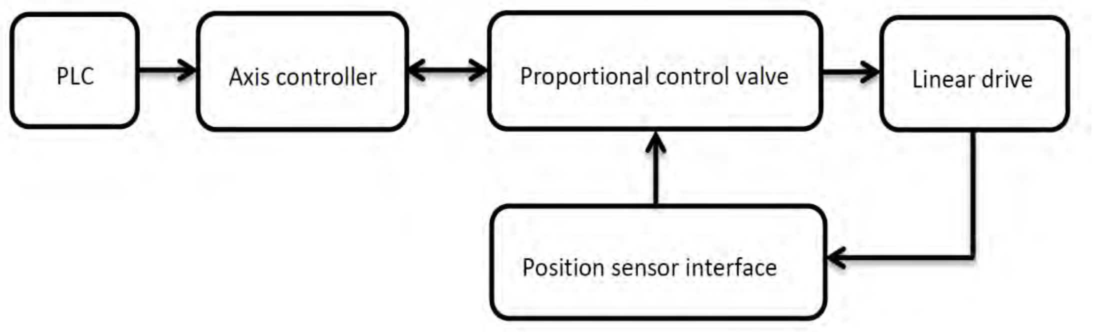


Figure 3-1 Single Axis Pneumatic Position Control System

The entire position control system used was obtained from Festo. The linear pneumatic actuator used is illustrated in figure 3-2 (Festo, 2014a) and contains an integrated displacement encoder. The 3 cylinders used all have a piston diameter of 63 mm and an actuator stroke length of 250 mm. Each actuator is capable of lifting a maximum mass of 60 kg at 6 bar of pressure.

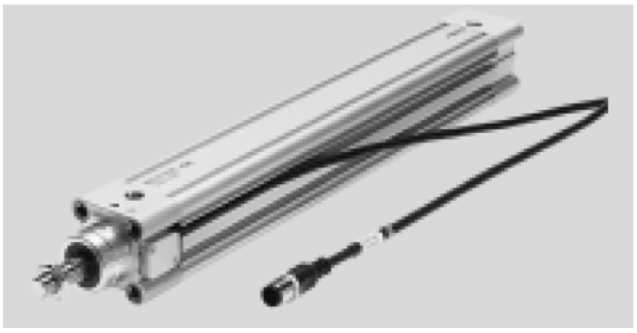


Figure 3-2 Linear Pneumatic Actuator with Integrated Displacement Encoder

The sensor interface shown in figure 3-3 (Festo, 2014b) is designed to interface the displacement encoder to the double acting directional proportional control valve. The actual position value from the displacement encoder is passed all the way through to the axis controller.

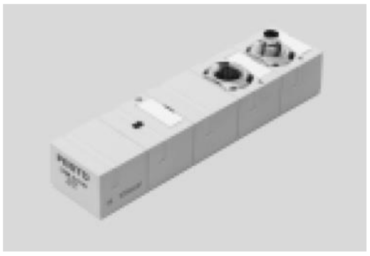


Figure 3-3 Digital Incremental Sensor Interface

Figure 3-4 (Festo, 2015a) illustrates the double acting directional proportional control valve used. The double acting valve is able to control the forward and backward strokes on the linear pneumatic actuator by applying the appropriate pressure to the appropriate end of the linear pneumatic actuator.

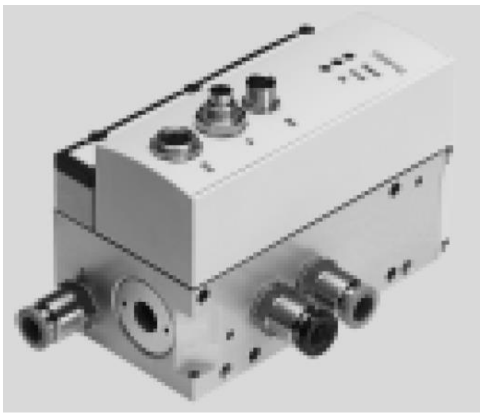


Figure 3-4 Double Acting Directional Proportional Control Valve

The CMAX axis controller illustrated in figure 3-5 (Festo, 2015b) is the intelligence of the position control system. It detects the various hardware components in the system and determines if all the components are functioning correctly. The controller can be used as either a position controller or force controller, for this application it is used as a position control system. It provides feedback control by adjusting the control signal to the directional proportional control valve; this signal adjustment is based on the error signal between the desired and actual position of the linear pneumatic actuator.

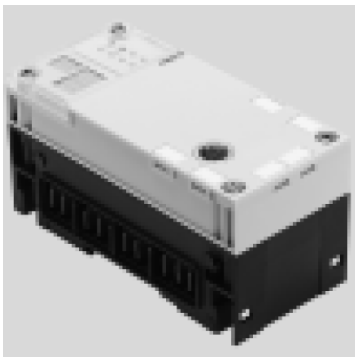


Figure 3-5 CMAX Axis Controller

The CPX programmable logic controller, illustrated in figure 3-6 (Festo, 2015c), is the device used to interface with the X-Sim Convertor software plugin and the 3 axis controllers. Software is written in the PLC environment to read in data, via UDP, and extract the position output for each actuator. This position output is passed to the appropriate axis controller to perform position control.

Figure 3-6 CPX Programmable Logic Controller

3.2 Motion Control System Setup and Testing

The pneumatic position control system components were connected together onto the vehicle simulator system, illustrated in figure 3-7. The system was configured and tested using the Festo Configuration Tool. Basic motion was performed, without any simulator driver, on each actuator to test the functionality and evaluate the position control systems performance.

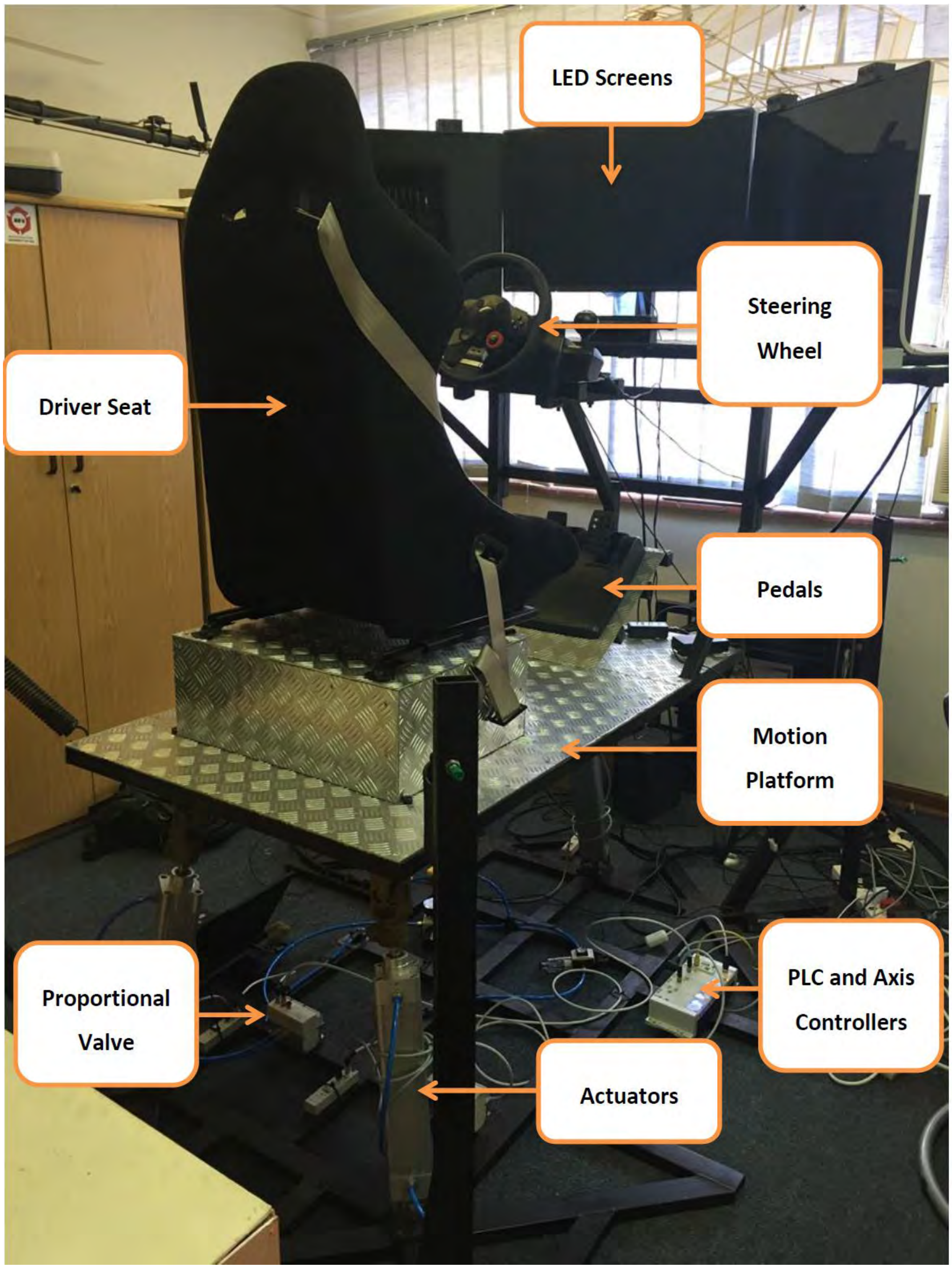


Figure 3-7 Vehicle Simulator with Position Control System

Before any motion could be performed each system is configured with basic parameters, shown in table 3-1, that aid in position control. Figure 3-8 illustrates the actuator labelling used; this labelling of actuators is used in subsequent chapters to ensure correct motion data for each of the actuators.

Table 3-1 Actuator Position Control Parameters

Actuator	1	2	3
Mass	25 kg	25 kg	25 kg
Supply Pressure	3 bar	3 bar	3 bar
Fitting Position	90°	90°	90°
Velocity	0.2 m/s	0.2 m/s	0.2 m/s
Acceleration	2.0 m/s^2	2.0 m/s^2	2.0 m/s^2
Deceleration	2.0 m/s^2	2.0 m/s^2	2.0 m/s^2
Position Tolerance	1.0 mm	1.0 mm	1.0 mm

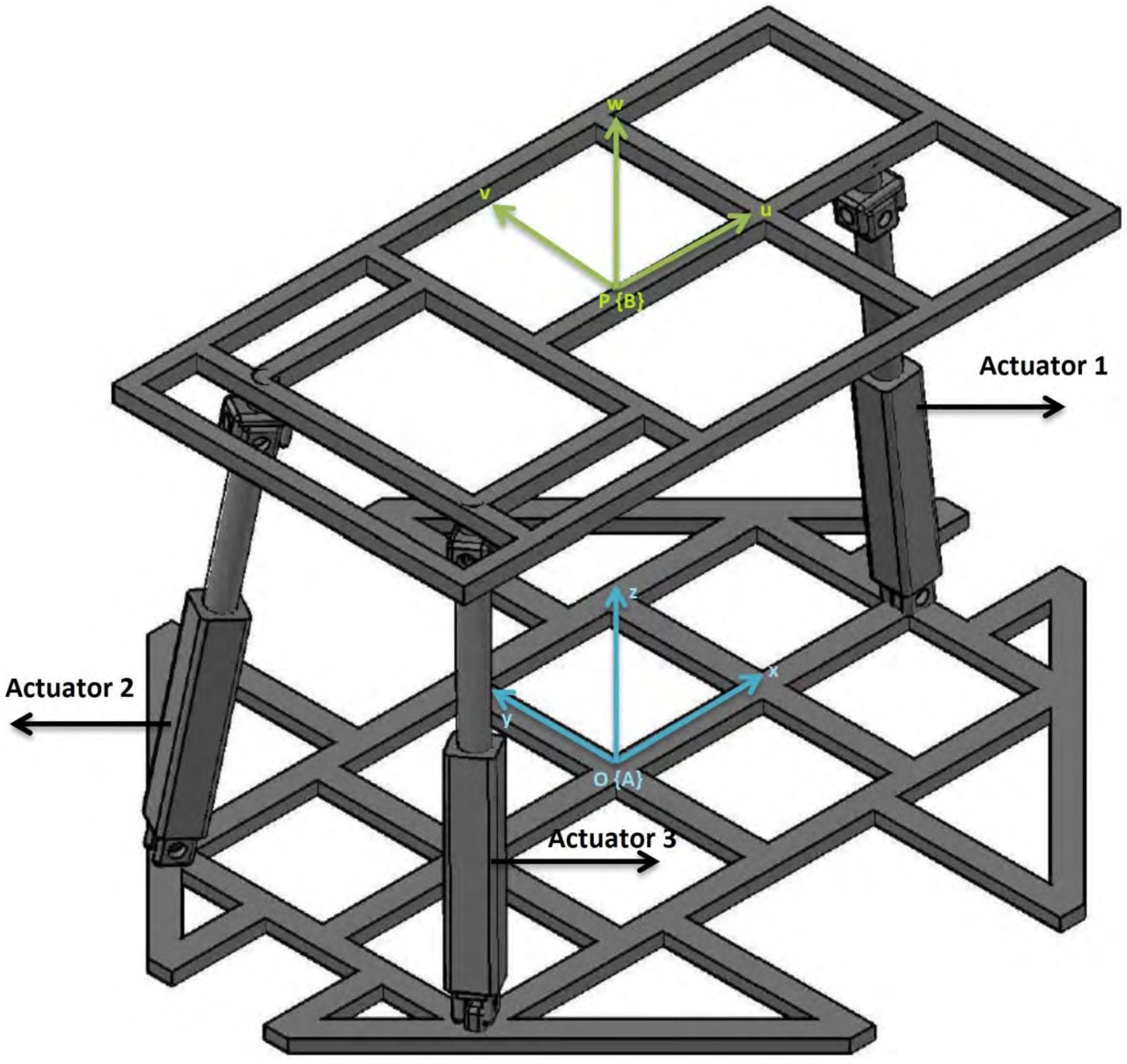


Figure 3-8 3-DOF Platform with Actuator Labels

The next step was to calibrate each displacement encoder that is built into each of the actuators. The method is called homing and retracts each actuator until the mechanical end point is reached; this position becomes the zero reference point for the displacement encoder.

Basic motion was then performed on each of the 3 actuators in the system, the results are reported below.

Figure 3-9 illustrates the position control tracking performance of actuator 1 in the vehicle simulator. Input to the system is a step motion change from 10 mm to 70 mm. The graph illustrates the tracking performance showing the actual position of the actuator stroke length tracking the nominal value (output from the controller) which is desired. Transient response of the system is within the 1 mm error tolerance throughout and the response time is 1.4 s. The system tracks well until the end position is reached. Steady state actual value is 69.21 mm and the nominal value is 69.99 mm. The error value is -0.78 mm which is within the 1 mm position tolerance set for the system.

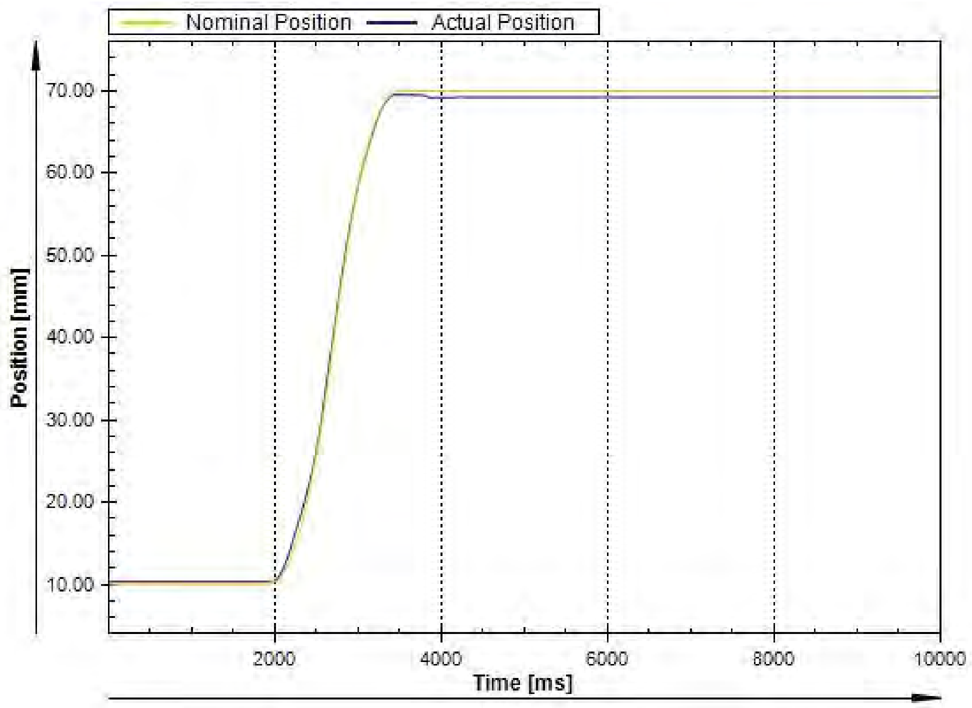


Figure 3-9 Actuator 1 Position Control Tracking

Figure 3-10 illustrates the position control performance of actuator 2 in the vehicle simulator. The input used is again a step change from 10 mm to 70 mm. The graph illustrates the tracking performance showing the actual position of the actuator stroke length tracking the nominal value (output from the controller) which is desired. The system tracks within 1 mm error tolerance for the transient response. The response time for this step change input is 1.4 s. Steady state actual value is 69.15 mm for this actuator and the nominal value which is desired is 69.99 mm. The error for this actuator is -0.84 mm and is within the 1 mm error tolerance used for the system.

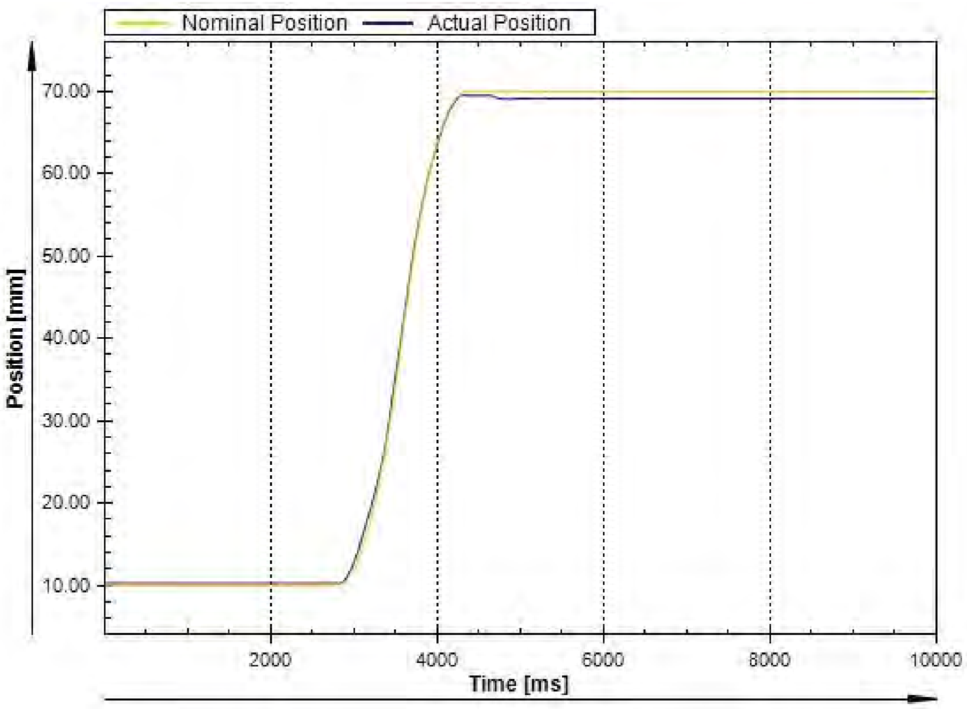


Figure 3-10 Actuator 2 Position Control Tracking

Figure 3-11 illustrates the position control tracking performance of actuator 3 in the vehicle simulator. The input is a step motion change from 10 mm to 70 mm. The graph illustrates the tracking performance showing the actual position of the actuator stroke length tracking the nominal value (output from the controller) which is desired. The system tracks well within the 1 mm tolerance until the end position is reached. It can be seen that final actual value is 69.18 mm and the nominal value, which is desired, is 69.99 mm. The error is within the tolerance of 1 mm used for testing. The response time for actuator 3 is 1.4 s.

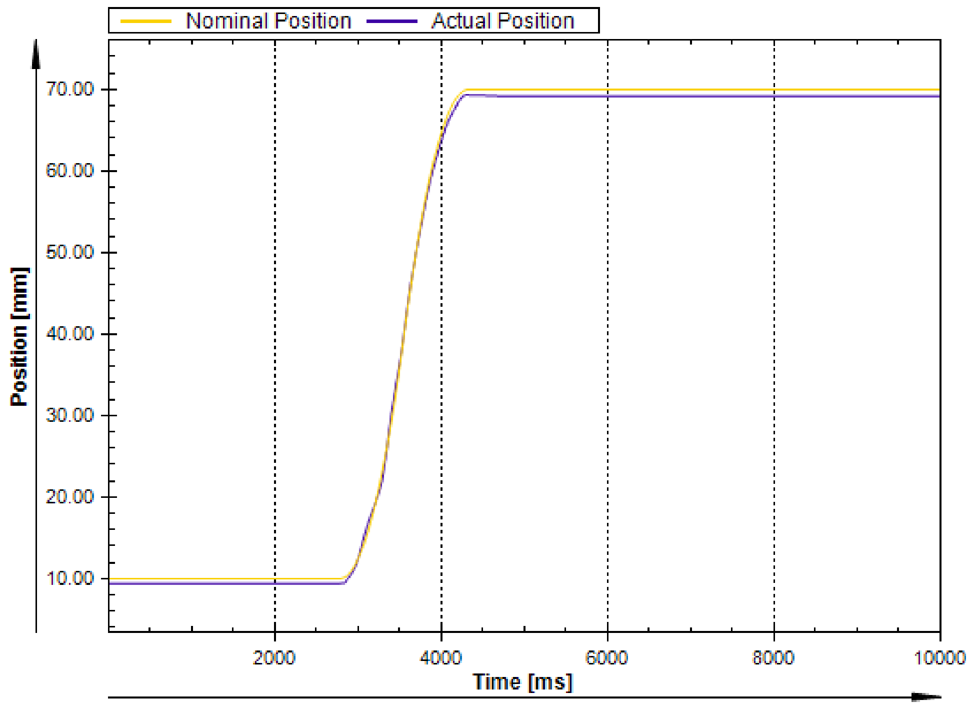


Figure 3-11 Actuator 3 Position Control Tracking

3.3 PLC Software Programming

The PLC is programmed in the Festo Software Tool program and uses the structure text syntax for code writing. The CPX PLC is designed to interface with the X-Sim Convertor software plugin, discussed in chapter 5, and receives data over a UDP network connection. Figure 3-12 illustrates the PLC program UML activity diagram illustrating the software algorithm; the full code for this algorithm is attached in appendix C. The PLC receives the actuator position string which is transferred via UDP; this data contains the required positions of each of the 3 actuators to be able to replicate the vehicle motion on the vehicle simulator motion platform. The PLC separates and extracts each actuators individual axis string which contains the individual actuators required position value. The actuator string is then converted to an integer values since the CMAX controller accepts the required actuator position in integer format. The integer actuator position value is written to the CMAX controller which controls the directional proportional control valve to achieve the desired position value.

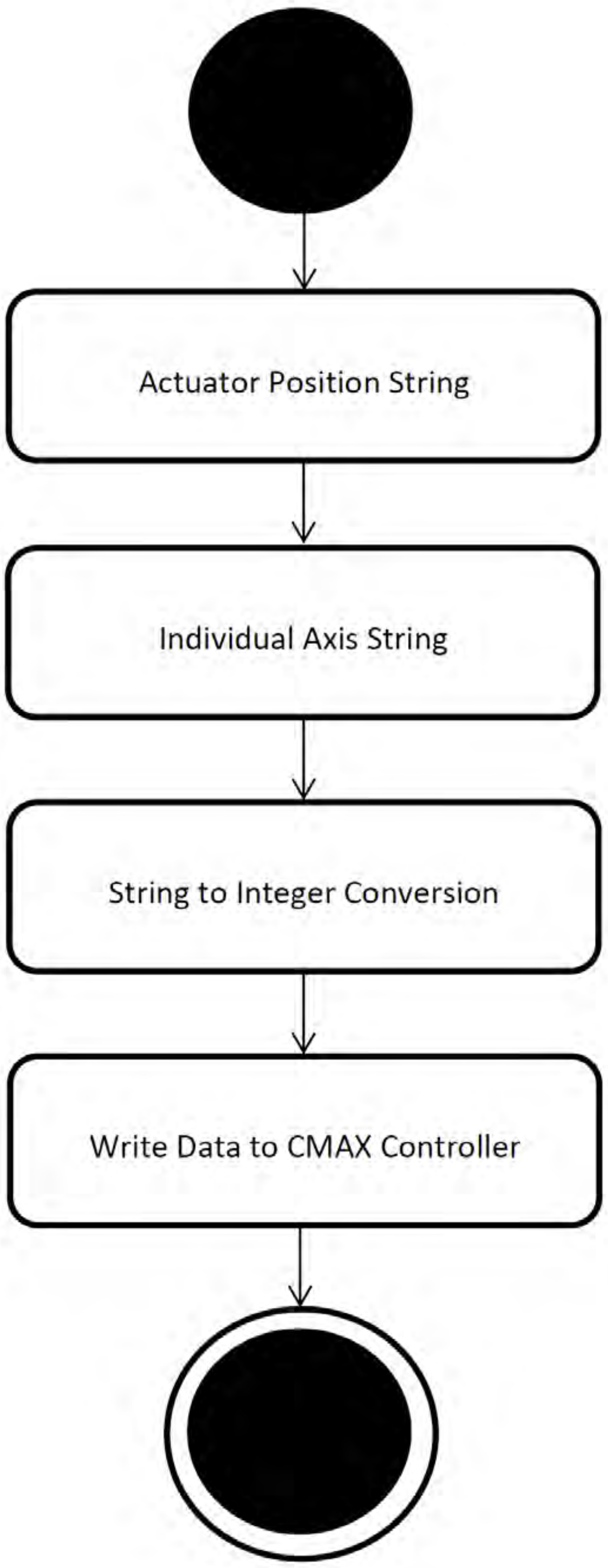


Figure 3-12 PLC Program UML Activity Diagram

3.4 Chapter Summary

This chapter presented the motion control system used in the vehicle simulator system. It explained the Festo position control system, which is a pneumatic system. The various components in the position control system are explained and details of how these components interact to achieve position control are discussed.

Motion control hardware was added and configured onto the vehicle simulator system. Parameters for the system, shown in table 3-1 were then added. Basic motion tasks were performed for each of the 3 actuators in the vehicle simulator system; the results showed excellent tracking performance and good control system response time.

The final part of this chapter explained the PLC software algorithm which is written to provide the interface between the X-Sim Convertor software plugin, explained in chapter 5, and the position control hardware. This software algorithm is used in chapter 6 to obtain results using live data from the games physics engine and evaluate the vehicle simulators performance.

4 Motion Cueing

Motion cueing aims to recreate the motion sensations experienced in a vehicle within the confines of a motion simulator platform. The main problem with the replication of this motion is the limited workspace of the motion platform. The classical washout algorithm (Schmidt and Conrad, 1970), adaptive washout algorithm (Ariel and Sivan, 1984), optimal washout algorithm (Sivan et al., 1982) and model predictive control techniques (Baseggio et al., 2011) aim to recreate these motion sensations, by exploiting the human vestibular system, and try to maximise workspace utilisation.

In human beings the vestibular system is responsible for providing motion cues. The full functioning of the vestibular system models and its limitations is presented in this chapter. The various coordinate systems used in motion cueing are discussed with the aim of highlighting which coordinate system is best to implement the motion cueing strategy. The classical washout algorithm is designed be used for both the 3-DOF motion platform and the 6-DOF motion platform cases. It was decided to use the classical washout algorithm for the vehicle simulator due to its low computational requirements and ease of implementation, in comparison to other motion cueing strategies. Finally simulations in Matlab/Simulink are conducted to evaluate the performance of the classical washout algorithm against the human vestibular system models.

4.1 Vestibular System

The vestibular system is the sensory system used to provide motion cues. It consists of the otolith and semi-circular canal. The semi-circular canal senses angular velocity and the otolith is used to sense linear motion via specific force.

In figure 4-1 the semi-circular canal model used is illustrated, this model was developed by Young and Oman and was subsequently reported on (Zacharias, 1978). It can be seen that the term δTH represents a detection threshold of angular velocity in the semi-circular canal system. Motion below this threshold will go undetected to the human observer. The parameter values used in the semi-circular canal model are taken from (Reid and Nahon, 1985) and shown in Table 4-1.

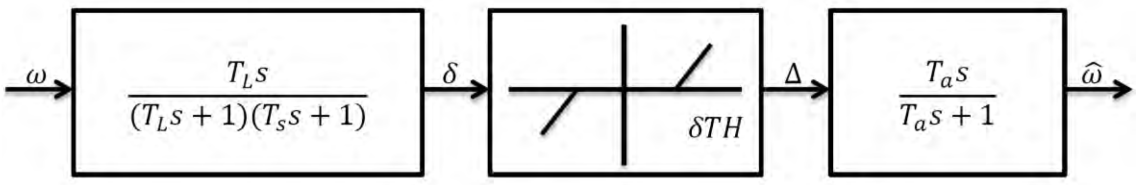


Figure 4-1 Semi-Circular Canal Model

The detection threshold output Δ is represented by:

$$\Delta = 0 \text{ for } |\delta| < \delta TH \ ... \ (4.1)$$

$$\Delta = \delta - SGN(\delta)\delta TH \text{ for } |\delta| > \delta TH \ ... \ (4.2)$$

The semi-circular canal model is used to evaluate the sensed angular velocity $\widehat{\omega}$ for the three axes of motion, with the actual vehicle angular velocity ω as the input. This model applies to rotation about the x-axis (Roll), rotation about the y-axis (Pitch) and rotation about the z-axis (Yaw), with different parameter values.

The otolith contained in the vestibular system is used to sense the translational motion. It senses specific force, the vector difference between translational inertial acceleration and gravitational acceleration. It is represented by:

$$\vec{f} = \vec{a} - \vec{g} \ ... \ (4.3)$$

Figure 4-2 illustrates the model for the otolith system which is used; this model was developed by Meiry and Young and was subsequently reported on (Zacharias, 1978). It can be seen that the term dTH represents a detection threshold of specific force motion in the otolith system. Motion below this threshold will go undetected to the human observer. The parameter values used in the otolith model are taken from (Reid and Nahon, 1985) and shown in Table 4-2.

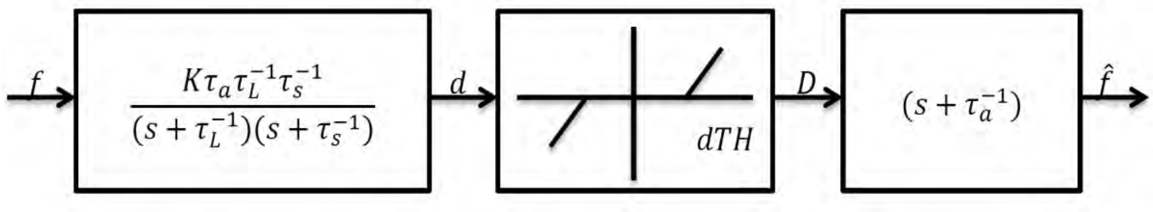

f
$\frac{K\tau_a\tau_L^{-1}\tau_s^{-1}}{(s+\tau_L^{-1})(s+\tau_s^{-1})}$
d
dTH
D
$(s+\tau_a^{-1})$
$\hat{f}$

Figure 4-2 Otolith Model

The detection threshold output D is represented by:

$$D = 0 \text{ for } |d| < dTH \ ... \ (4.4)$$

$$D = d - SGN(d)dTH \text{ for } |d| > dTH \ \ldots \ (4.5)$$

The otolith model is used to evaluate the sensed specific force $\hat{f}$ for the three axes of motion, with the actual vehicle specific force f as the input. This model applies to translational motion along the x-axis (Surge), the y-axis (Sway) and the z-axis (Heave), with different parameter values.

Parameter values for the semi-circular canal and otolith model are taken from (Reid and Nahon, 1985) and are shown in table 4-1 and table 4-2.

Table 4-1 Model Parameters for Rotational Motion

	Roll (x-axis)	Pitch (y-axis)	Yaw (z-axis)
$T_L(s)$	6.1	5.3	10.2
$T_s(s)$	0.1	0.1	0.1
$T_a(s)$	30	30	30
δTH (°/s)	3.0	3.6	2.6

Table 4-2 Model Parameters for Translational Motion

	Surge (x-axis)	Sway (y-axis)	Heave (z-axis)
$\tau_L(s)$	5.33	5.33	5.33
$\tau_s(s)$	0.66	0.66	0.66
$\tau_a(s)$	13.2	13.2	13.2
K	0.4	0.4	0.4
dTH (m/s^2)	0.17	0.17	0.28

4.2 Coordinate Systems

The motion cueing techniques aim to replicate the motion sensations felt in a real vehicle within the workspace of the motion simulator platform. Based on research done previously (Reid and Nahon, 1985) the following coordinate systems are chosen. Coordinate system **{B}** is located at the centroid of the moving platform and coordinate system **{A}** is located at the centroid of the base.

Coordinate system **{B}** illustrated in figure 4-3 represents the point where the specific forces and angular velocity inputs to the vehicle are used, in (Reid and Nahon, 1985) it was shown that this location is the best choice since it minimises actuator movement. Coordinate system **{A}** illustrated in figure 4-3 represents the inertial coordinate frame. It is the coordinate system in which the platform motion is evaluated.

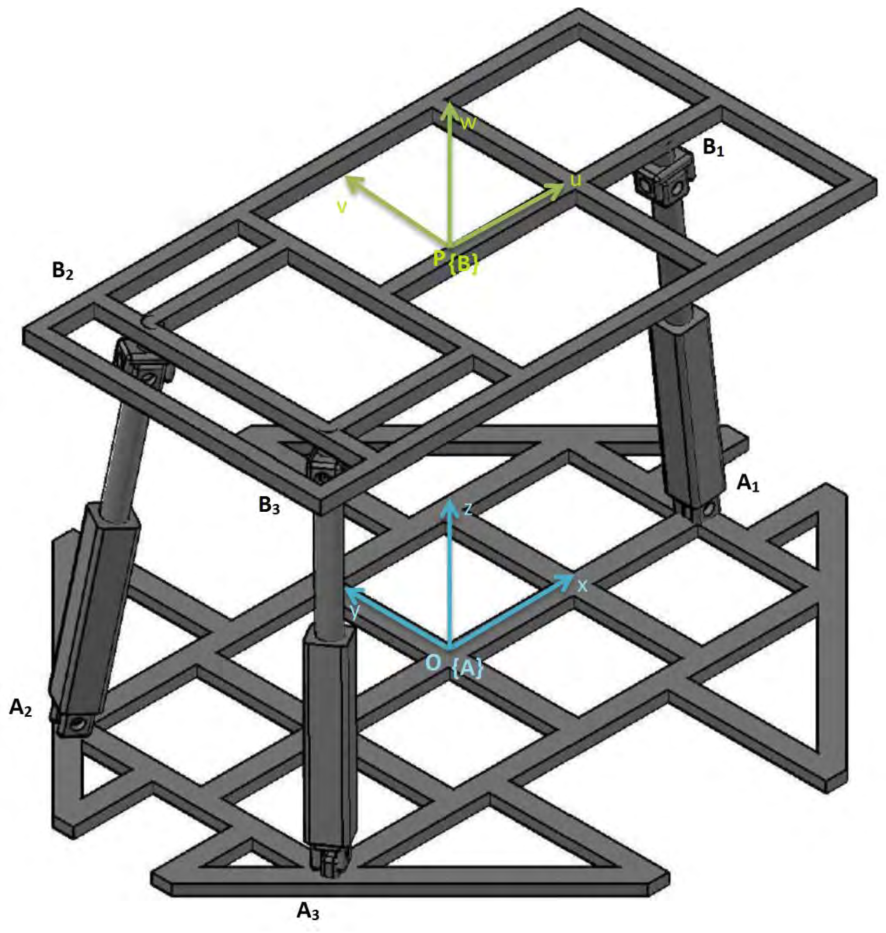


Figure 4-3 Motion Cueing Coordinate Systems for the 3-DOF Platform

The inputs to the motion cueing strategy are the specific force vector $\vec{f}_{veh}$ and angular velocity vector $\vec{\omega}_{veh}$ experienced in the real vehicle. The motion cueing strategy aims to replicate these signals, within the vehicle simulator, as closely as possible.

$$\vec{f}_{sim} \approx \vec{f}_{veh} \;\ldots\; (4.6)$$

$$\vec{\omega}_{sim} \approx \vec{\omega}_{veh} \;\ldots\; (4.7)$$

$\vec{f}_{sim}$ and $\vec{\omega}_{sim}$ represent the specific force vector and angular velocity vector experienced at the centroid of the moving platform coordinate system **{B}**.

4.3 Classical Washout Algorithm

The classical washout algorithm is a motion cueing strategy first implemented in flight simulators (Schmidt and Conrad, 1970). It is designed to replicate the motion sensations felt in an actual vehicle without breaching the platform workspace constraints. The implementation is divided into two channels, the first channel is used for translational motion and the second is used for rotational motion.

4.3.1 Translational Motion

Figure 4-4 illustrates the translation channel for the classical washout algorithm. The translational channel is used to replicate the transient component of the vehicle specific force vector $\vec{f}_{veh}$. The vehicle specific force vector $\vec{f}_{veh}$ is input into the system. This signal is scaled to help constrain platform motion. The scaled specific force vector $\vec{f}_1$ is then used to generate the acceleration vector $\vec{a}_1$ for the centroid of the moving platform **{B}**. The acceleration vector is given by:

$$\vec{a}_1 = \vec{f}_1 + \vec{g}_1 \ \ldots \ (4.8)$$

According to (Reid and Nahon, 1985) the gravitational vector signal $\vec{g}_1$ is given by:

$$\vec{g}_1 = {}^{A}R_B^T \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} \ldots \ (4.9)$$

The result of matrix multiplying by the transpose of the rotation matrix from **{B}** to coordinate frame **{A}** is expressed as:

$$\vec{g}_1 = \begin{bmatrix} g \sin\beta \\ -g \sin\alpha \cos\beta \\ -g \cos\alpha \cos\beta \end{bmatrix} \ldots \ (4.10)$$

The vector $\vec{a}_1$ is then transformed into coordinate frame **{A}**. This transformation is done by multiplying by the rotation matrix as follows:

$$\vec{a}_2 = {}^{A}R_B \vec{a}_1 \ \ldots \ (4.11)$$

Filtering of the vector $\vec{a}_2$ in the fixed based coordinate system **{A}** is done to perform washout. The washout process is used to return the simulator motion platform back to the neutral (centre) position. This process helps in preventing steady state motion errors on the actuator legs. The output acceleration vector ${}^{A}\vec{a}$ is then integrated twice to produce the platform position vector ${}^{A}P = [p_x \quad p_y \quad p_z]^T$, in the inertial coordinate frame **{A}**. This signal is used in the inverse kinematic analysis to obtain the actuator stroke lengths.

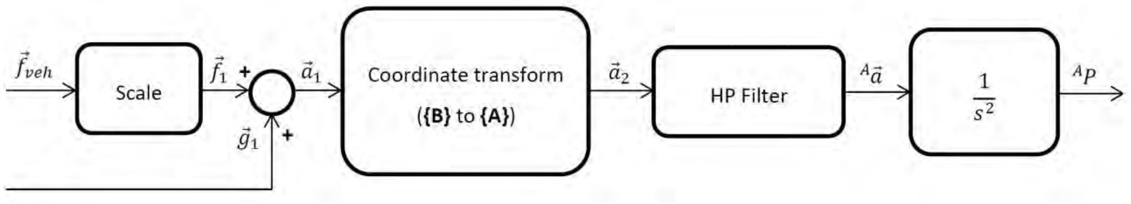


Figure 4-4 Translational Channel for the Classical Washout Algorithm

4.3.2 Rotational Motion

Figure 4-5 illustrates the rotational channel for the classical washout algorithm. The rotational channel is composed of two parts, which together produce the rotation (Roll-Pitch-Yaw) angles for the motion platform.

The first part involves a process called tilt coordination. Tilt coordination is used to replicate the sustained component of the vehicle specific force vector $\vec{f}_{veh}$ via tilt of the motion platform. It aims to use a component of the gravity vector to simulate a sustained specific force. This component is interpreted by the otolith as a sustained linear acceleration. It is important to note that the tilt rate should be kept below δTH, the angular velocity motion detection threshold, to prevent false rotational cues from being detected by the semi-circular canal.

The process starts with the signal $\vec{f}_{veh}$ which is scaled and passed through a low-pass filter. This filter extracts the low frequency component vector $\vec{f}_L$ of the vehicles specific force.

In the absence of rotational motion from the angular velocity component, the Roll-Pitch-Yaw angles can be represented by:

$${}^{A}\vec{\varphi} = {}^{A}\vec{\varphi}_L \ ... \ (4.12)$$

The Roll-Pitch-Yaw angles based on the sustained specific force vector $\vec{f}_L$ were approximated previously (Reid and Nahon, 1985) and is given by:

$$\alpha_L \approx \frac{f_L^y}{g} \ ... \ (4.13)$$

$$\beta_L \approx -\frac{f_L^x}{g} \ ... \ (4.14)$$

$$\gamma_L = 0 \ ... \ (4.15)$$

The second component of the rotational channel is used in the replication of the transient component of the vehicle angular velocity vector $\vec{\omega}_{veh}$. The vehicle angular velocity vector $\vec{\omega}_{veh}$ is scaled to ensure the platform rotational motion limits are not reached. The next step involves transforming the angular velocity vector $\vec{\omega}_1$ to the Roll-Pitch-Yaw angle rate vector, $\vec{\dot{\varphi}}_1$, which is required to be able to perform inverse kinematic analysis. The transformation is given by:

$$\vec{\dot{\varphi}}_1 = {}^{A}T_{B}\vec{\omega}_1 \ \ldots \ (4.16)$$

With ${}^{A}T_{B}$ given by:

$${}^{A}T_{B} = \begin{bmatrix} 1 & \sin\alpha\tan\beta & \cos\alpha\tan\beta \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha\sec\beta & \cos\alpha\sec\beta \end{bmatrix} \ldots \ (4.17)$$

The signal $\vec{\dot{\varphi}}_1$ is then high-pass filtered to ensure platform washout. This filtering will ensure the platform returns back to its neutral position (centre) once the rotational motion is complete. It has the same effect as the translation channel washout filter by preventing steady state motion errors on the actuator legs. The signal $\vec{\dot{\varphi}}_H$ is integrated to give the high frequency signal for Roll-Pitch-Yaw angles below:

$${}^{A}\vec{\varphi}_H = \int \vec{\dot{\varphi}}_H \ dt \ \ldots \ (4.18)$$

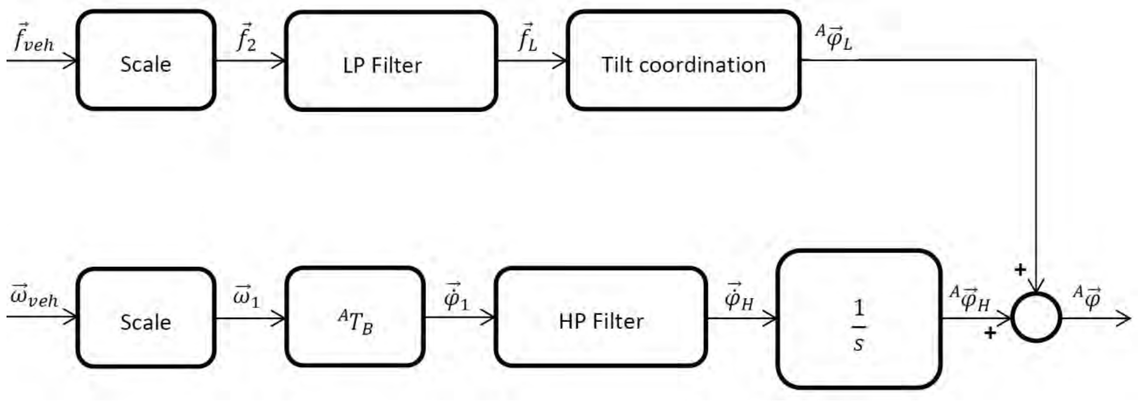


Figure 4-5 Rotational Channel for the Classical Washout Algorithm

The Roll-Pitch-Yaw angle components for the sustained specific force vector signal (Eq. 4.12) and the angular velocity vector signal (Eq. 4.18) are combined to give the Roll-Pitch-Yaw angle values that are used in the inverse kinematic analysis. The Roll-Pitch-Yaw angle values are given by:

$$^{A}\vec{\varphi} = {}^{A}\vec{\varphi}_L + {}^{A}\vec{\varphi}_H \ \ldots \ (4.19)$$

4.3.3 Filter Selection

The filter selection for the flight simulator in (Reid and Nahon, 1986) was chosen to be 2nd order for transient translational acceleration and 1st order for transient angular velocity. This selection was due to modest motions experienced in a flight simulator. A vehicle in general has more demanding acceleration manoeuvres, leading to the usage of a 3rd order filter for transient translational acceleration and a 2nd order filter for the transient angular velocity.

The transient translational acceleration filter, HP Filter in figure 4-4, implemented in coordinate frame **{A}** is given by:

$$HP_{Translational}(s) = \frac{s^3}{(s^2+2\zeta\omega_n s+\omega_n^2)(s+\omega_b)} \ \ldots \ (4.20)$$

The filter parameters used for $HP_{Translational}$ in the rest of this chapter are given in table 4-3

Table 4-3 Parameter Values for the Translational Channel Filter

ζ	1
ω_n	3.1 rad/s
ω_b	0.2 rad/s

The transient angular velocity filter, HP Filter in figure 4-5, implemented in coordinate frame **{A}** is given by:

$$HP_{Angular}(s) = \frac{s^2}{(s+\omega_n)^2} \ \ldots \ (4.21)$$

The filter parameters used for $HP_{Angular}$ in the rest of this chapter are given in table 4-4

Table 4-4 Parameter Values for the Transient Angular Velocity Filter

ζ	1
ω_n	1 rad/s

The low-pass filter, LP Filter in figure 4-5, used in the tilt coordination process is given by:

$$LP_{Tilt}(s) = \frac{\omega_n^2}{s^2+2\zeta\omega_n s+\omega_n^2} \ \ldots \ (4.22)$$

The filter parameters used for LP_{Tilt} in the rest of this chapter are given in table 4-5

Table 4-5 Parameter Values for the Low-Pass Tilt Coordination Filter

ζ	1
ω_n	6.2 rad/s

4.4 Motion Cueing Simulation

A simulation setup was created in Matlab/Simulink to evaluate the effectiveness of the classical washout algorithm. The purpose of this simulation setup was to evaluate the effectiveness of the classical washout algorithm to:

- Effectively return the platform to neutral position using washout.
- Replicate sustained translational accelerations via tilt coordination.
- Replicate the sensations experienced in a vehicle as closely as possible within the motion simulator platform.

4.4.1 Motion Cueing Simulation Setup

The simulation setup was divided into libraries for the various subsystems. Libraries help in creating an easy to understand and modular system. Using libraries also facilitates reusability and allows for modification to be done easily.

The first subsystem created was for the translational motion channel, illustrated in figure 4-6, this subsystem aims to replicate the transient component of the vehicle specific force vector $\vec{f}_{veh}$ in the simulation setup. The inputs to this subsystem are the vehicle specific force signals $\vec{f}_{veh}$ and the Roll-Pitch-Yaw angles. These inputs are used to produce the acceleration signal $\vec{a}_1$ given by Eq. 4.8. The translational filter block process aims to extract the acceleration signals that are used at the centroid of the motion simulator platform **{B}**.

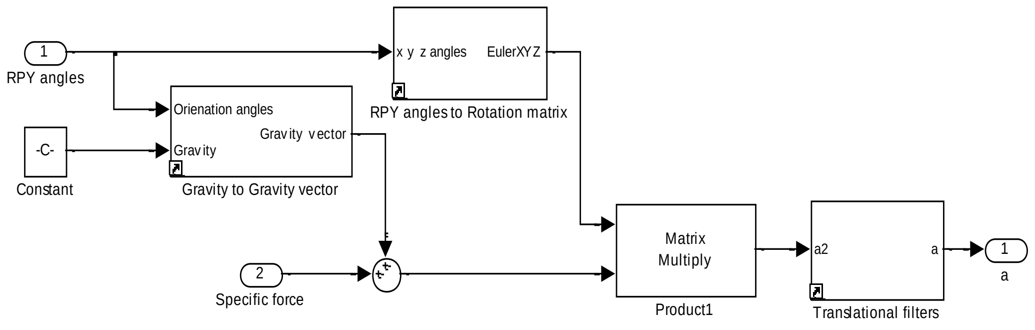


Figure 4-6 Translational Channel Subsystem

Figure 4-7 illustrates the tilt coordination subsystem which takes in the vehicle specific force vector signal $\vec{f}_{veh}$. The low-pass filter is used to extract the sustained specific force vector $\vec{f}_L$. The low frequency Roll-Pitch-Yaw angle vector,${}^{A}\vec{\varphi}_L$, is generated using equations 4.13 and 4.14. These signals are rate limited to 3 °/s for the x-axis (Roll) and 3.6 °/s for the y-axis (Pitch), which is the motion perception threshold values for the semi-circular canal δTH. The z-axis (Yaw) component has no contribution to the tilt coordination process and is set to zero.

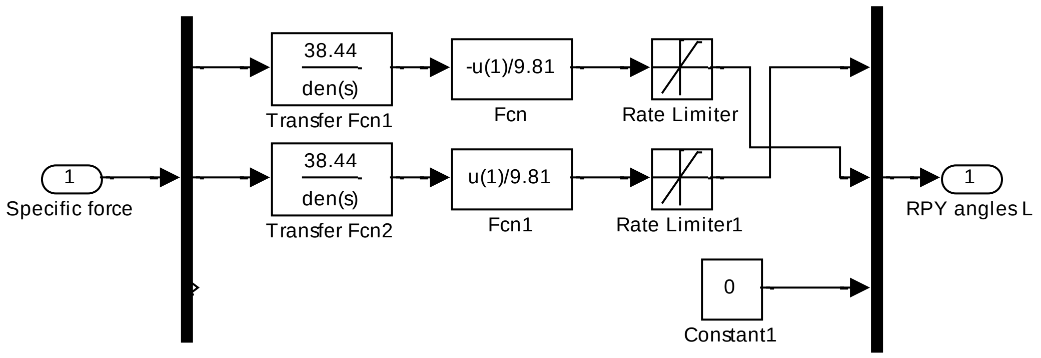


Figure 4-7 Tilt Coordination Subsystem

The angular velocity subsystem, illustrated in figure 4-8, aims to replicate the transient component of the vehicle angular velocity vector $\vec{\omega}_{veh}$. It takes in the Roll-Pitch-Yaw angles and the vehicle angular velocity vector $\vec{\omega}_{veh}$. The Roll-Pitch-Yaw angle rates (Eq. 4.16) are then formed and this signal is filtered to extract the high frequency component. The high frequency Roll-Pitch-Yaw angle rate signal is integrated to give the Roll-Pitch-Yaw angles,${}^{A}\vec{\varphi}_H$, which is output from this subsystem.

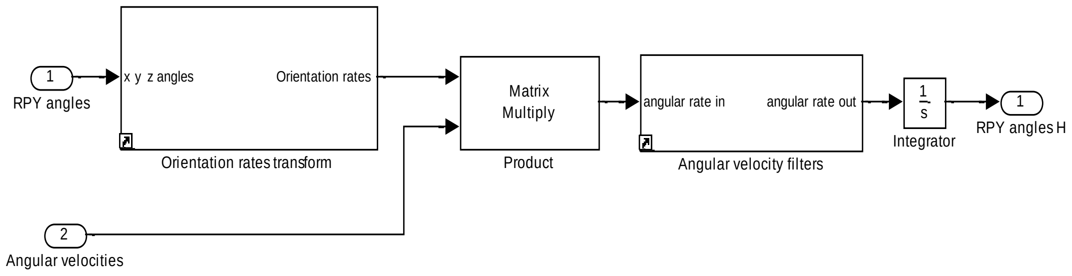


Figure 4-8 Angular Velocity Subsystem

The human vestibular system models are used to evaluate the effectiveness of the classical washout algorithm in replication of motion sensations experienced in a real vehicle. The models for the otolith and the semi-circular canal are used to evaluate the classical washout algorithm in Simulink.

The otolith model illustrated in figure 4-9 is used to evaluate the effectiveness of the classical washout algorithm in replication of the vehicle specific force vector $\vec{f}_{veh}$ experienced.

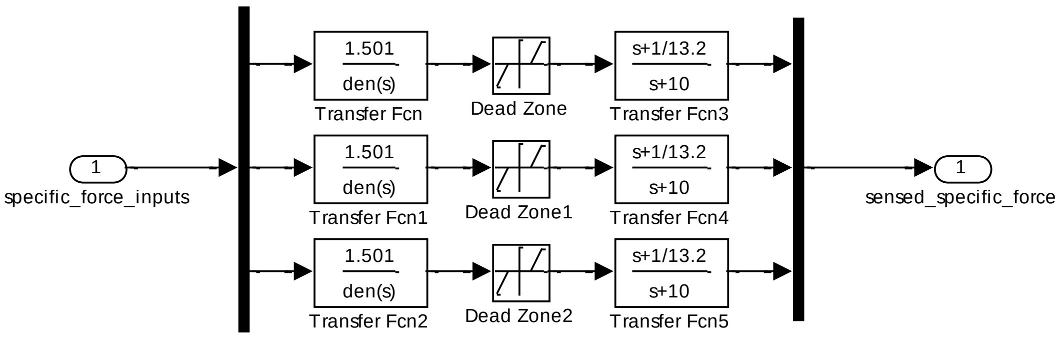


Figure 4-9 Otolith Simulation Model

The semi-circular canal model illustrated in figure 4-10 is used to evaluate the effectiveness of the classical washout algorithm in replication of the vehicle angular velocity vector $\vec{\omega}_{veh}$ experienced.

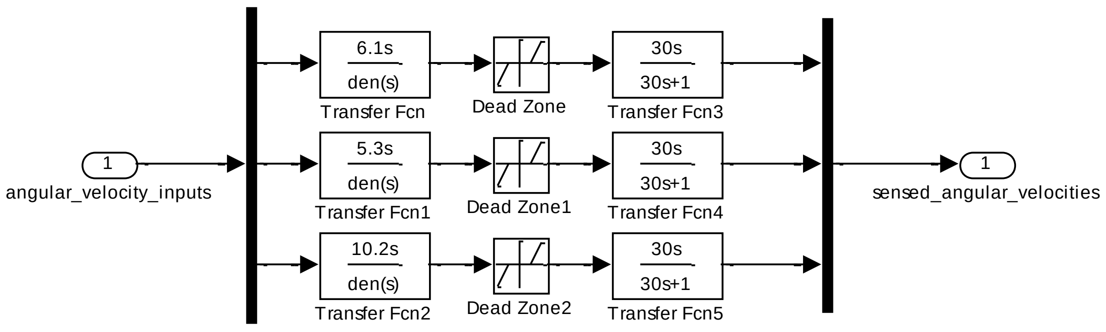


Figure 4-10 Semi-circular Canal Simulation Model

4.4.2 Motion Cueing Simulation Results

The simulations presented in this section were performed to verify the correct functioning of the classical washout algorithm in replication of both translational acceleration and angular velocity motion cues. A series of tests were conducted and these tests are explained together with the results.

4.4.2.1 Translational Motion Test

The translational motion test was used to verify the correct functioning of the classical washout algorithm in replication of the vehicle specific force vector $\vec{f}_{veh}$. The human vestibular system, using the otolith model, is used to show how well the vehicle specific force vector $\vec{f}_{veh}$ is replicated in the motion simulator by $\vec{f}_{sim}$ the specific force vector at the centroid of the moving platform coordinate system **{B}**.

Figure 4-11 illustrates the specific force input signal used for the system testing. The input signal is a unit step response with an initial value of 2 m/s^2 along the x-axis, the signal lasts for a period of 10 s.

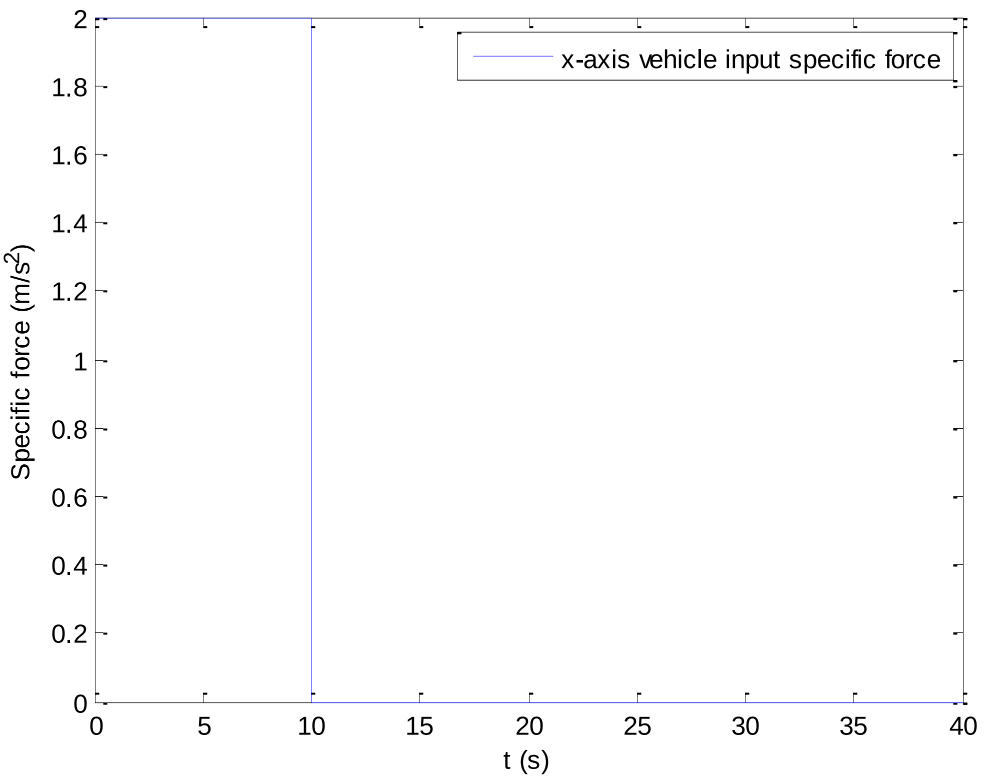


Figure 4-11 Specific Force Input for the x-axis

Figure 4-12 illustrates the platform acceleration along the x-axis. This acceleration represents the transients extracted from the high-pass filter along the translational motion channel. The first transient occurs initially when the acceleration goes from 0 m/s^2 to 2 m/s^2. The next transient occurs at 10 seconds when the acceleration drops from 2 m/s^2 to 0 m/s^2. It can be seen that the high-pass filter attempts to return the platform back to neutral (centre) position after both the transient acceleration periods. This process is known as washout and it aids in preventing steady state motion errors on the actuator legs. It can be seen that the washout process also creates some acceleration in the opposite direction to the intended acceleration; these accelerations can be seen just after the start and at 10 seconds. These signals are known as a false cue if the acceleration is above the otolith systems motion detection threshold dTH. It is known that the classical washout algorithm does let through some false cues due to the fixed filter parameters employed. During selection of these filter parameters there is a trade-off between optimal workspace utilisation and minimization of false motion cues.

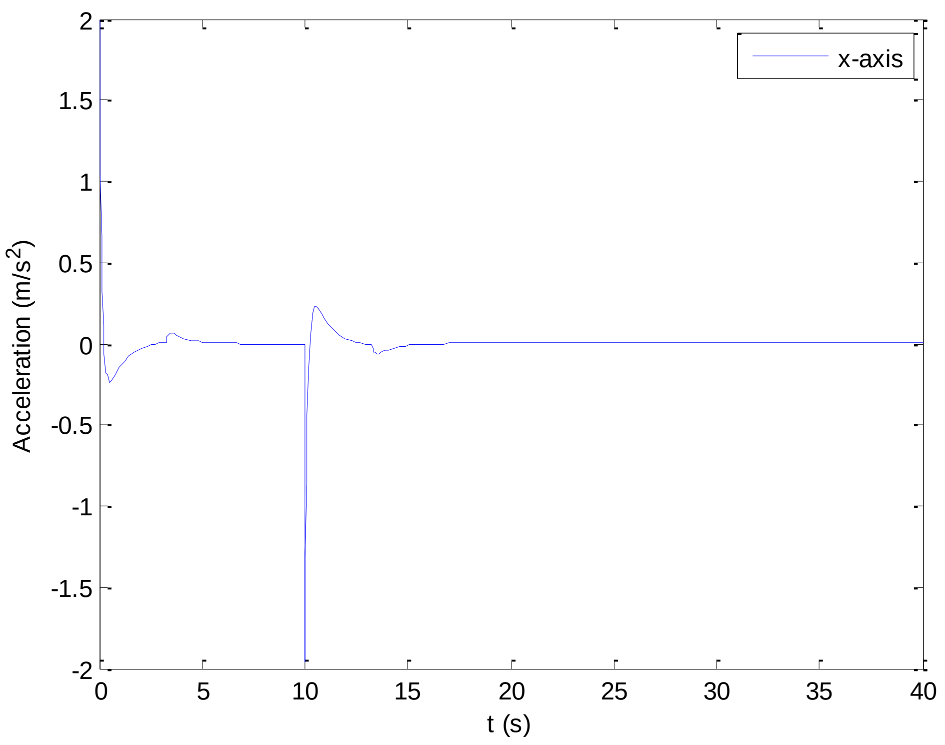


Figure 4-12 Platform Acceleration for the x-axis

In figure 4-13 the position of the platform along the x-axis is shown. Initial transient acceleration creates motion in the positive direction, washout then occurs causing the platform to return to the neutral (centre) position. The washout process is not optimal due to the fixed filter parameters, which are designed for worse case motion. This results in the platform taking additional time to stop motion completely. The washout process is a trade-off between optimal workspace utilisation and the prevention of false motion cues. The advantages of using the classical washout algorithm are the minimal implementation complexity and low processing performance requirements of this algorithm.

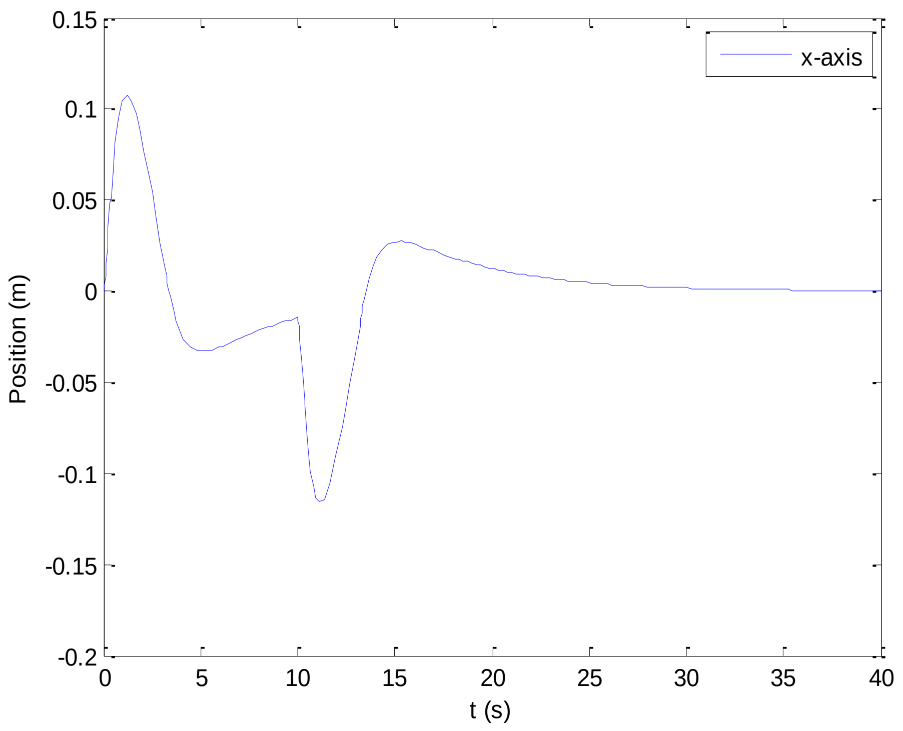


Figure 4-13 Platform Position for the x-axis

Figure 4-14 shows the replication of the sustained component of the vehicle specific force vector $\vec{f}_{veh}$ via tilt coordination. This replication is done via tilting of the platform about the y-axis (Pitch). The tilting is done with the tilt rate limit set to the motion detection threshold value of 3.6 °/s for rotations about the y-axis. The tilting of the motion platform is an attempt to replicate the sustained acceleration, along the x-axis, of 2 m/s^2 that occurs during the initial 10 seconds. The maximum tilt angle achieved for this motion is -11.68 degrees which gives the perception of accelerating constantly at 1.99 m/s^2 based on Eq. 4.14.

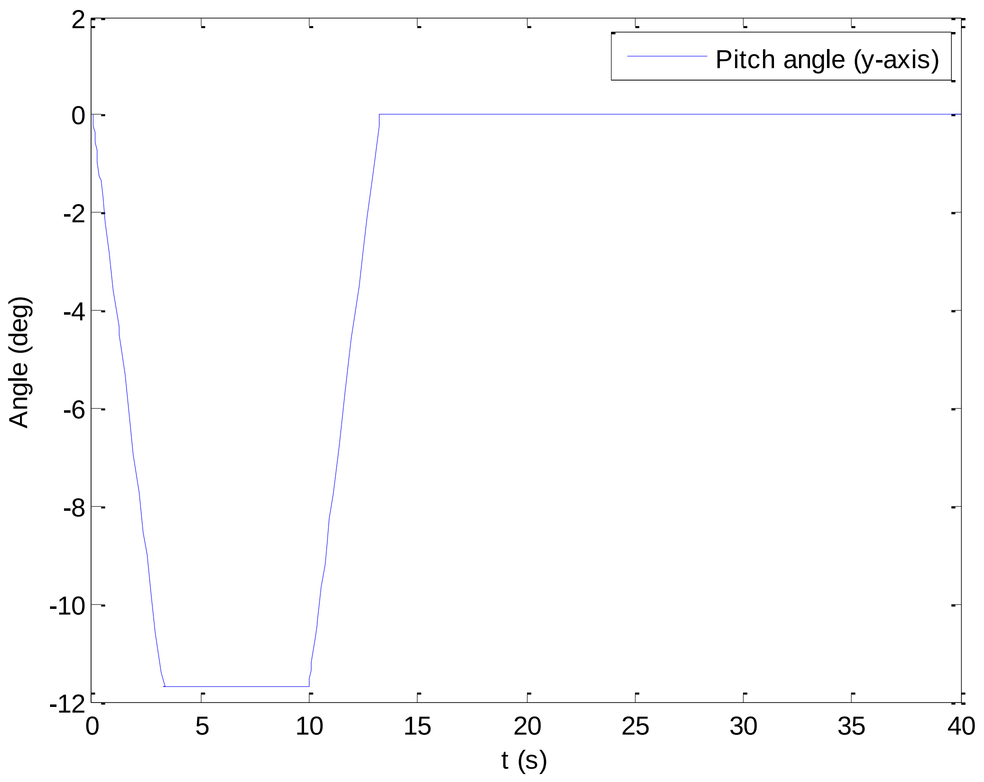


Figure 4-14 Platform Pitch Angle for the y-axis

Illustrated in figure 4-15 is the actual output from the otolith model for the vehicle specific force vector $\vec{f}_{veh}$ for the x-axis component and the output from the otolith model for the simulator specific force vector $\vec{f}_{sim}$ for the x-axis component. It can be seen that the classical washout filter provides a reasonably good result for the replication of translational motion sensations within the motion simulator platform. The washout filter is also effective in ensuring the platform returns to the neutral position after the motion input is complete. The classical washout filter, through the usage of tilt coordination, is also able to replicate sustained translational accelerations successfully.

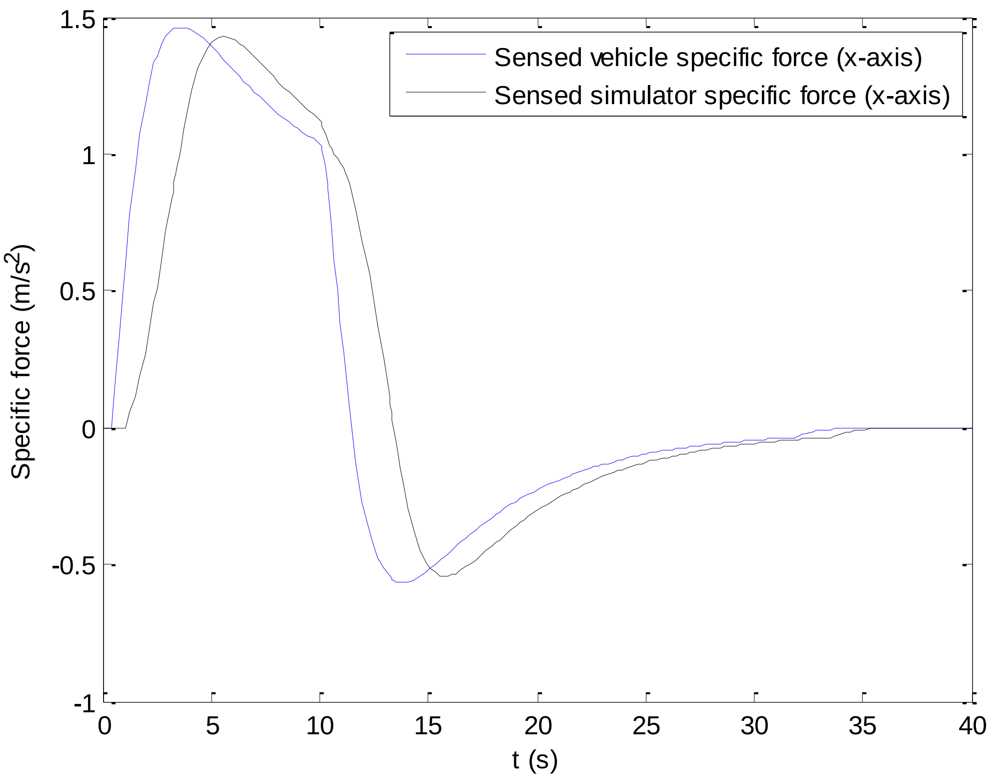


Figure 4-15 Sensed Vehicle and Simulator Specific Force for the x-axis

4.4.2.2 Rotational Motion Test

The rotational motion test was used to verify the correct functioning of the classical washout algorithm in replication of the vehicle angular velocity vector $\vec{\omega}_{veh}$. The human vestibular system, using the semi-circular canal model, is used to show how well the vehicle angular velocity vector $\vec{\omega}_{veh}$ is replicated in the motion simulator by $\vec{\omega}_{sim}$ the angular velocity vector at the centroid of the moving platform coordinate system **{B}**.

In figure 4-16 illustrated, the angular velocity positive and negative ramp input test signal is shown. The slope rate for the positive slope is set at 0.1 rad/s and -0.1 rad/s for the negative slope. The positive and negative slope input each run for 0.125 seconds.

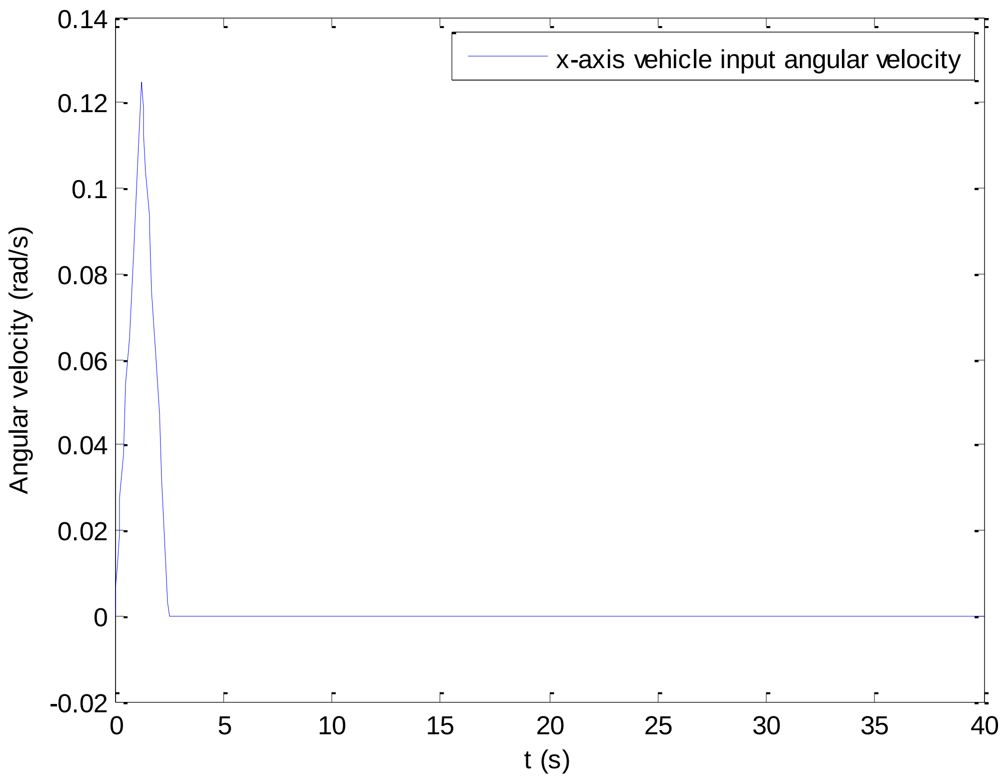


Figure 4-16 Angular Velocity Input for the x-axis

Illustrated in figure 4-17 is the sensed angular velocity signal, along the x-axis, of the actual vehicle for the vehicle angular velocity input shown in figure 4-16. The semi-circular canal model is used to demonstrate the feeling felt by the vehicle user. It can be noted that the human semi-circular system attenuates the input angular velocity signal. Figure 4-18 shows the sensed angular velocity along the x-axis within the simulator, this signal matches closely the sensed angular velocity of the vehicle signals general shape in figure 4-17 but it is an attenuated signal. There is also some sensed angular velocity in the opposite direction of motion due to the washout process attempting to return the platform to the neutral position. Ideally this motion should not occur, but the classical washout filter is known to let through such false cues due to the fixed filter parameter values.

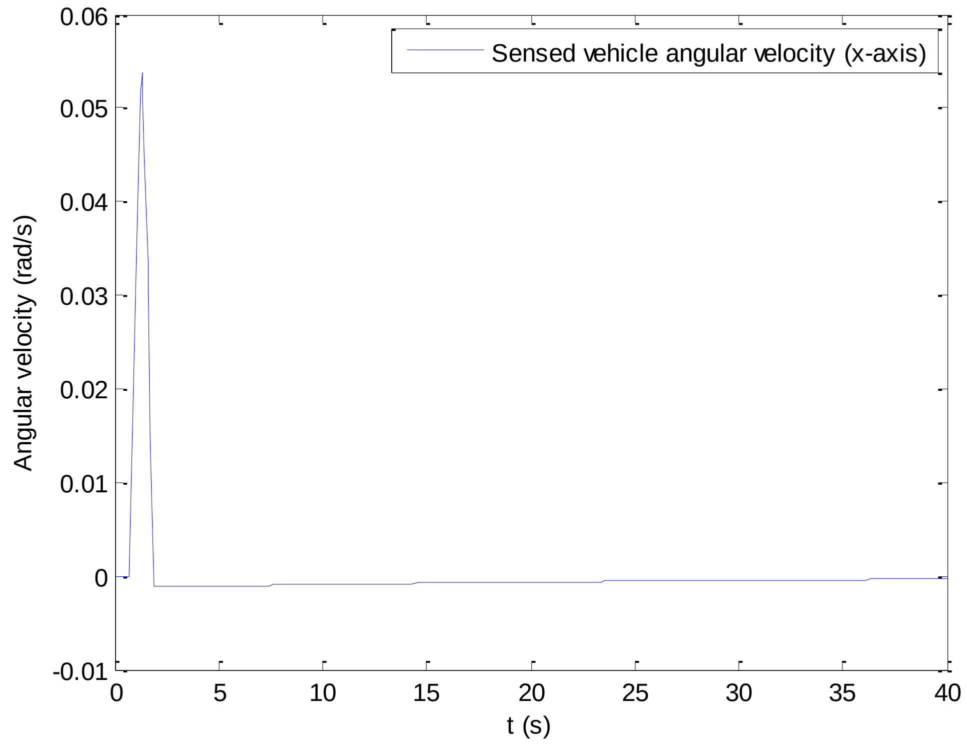


Figure 4-17 Sensed Vehicle Angular Velocity for the x-axis

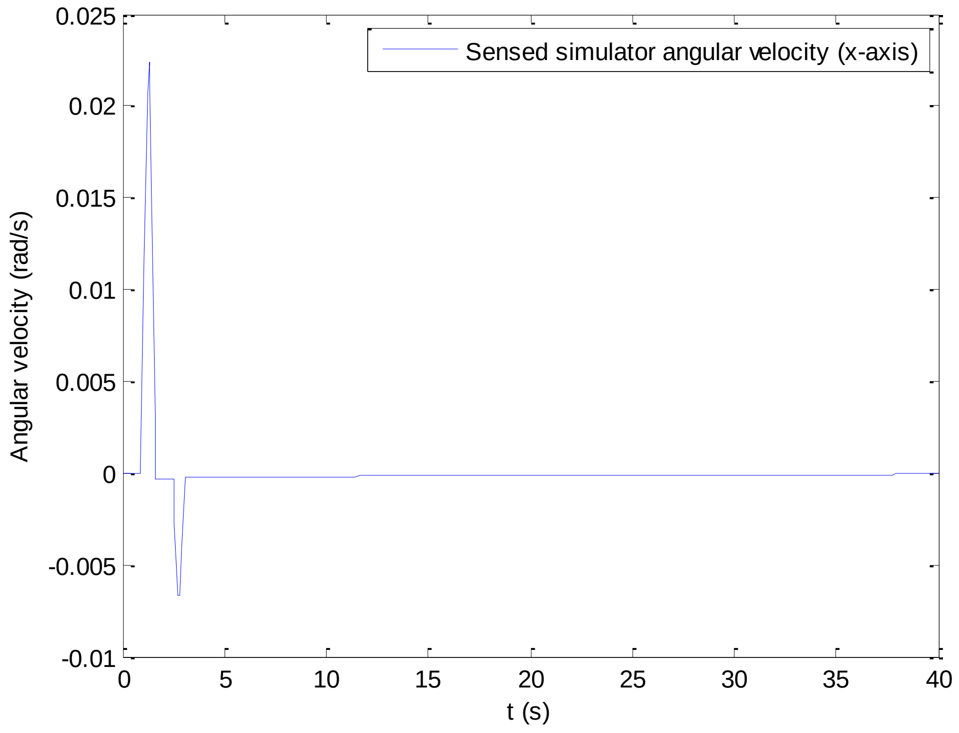


Figure 4-18 Sensed Simulator Angular Velocity for the x-axis

Figure 4-19 illustrates the platform orientation along the x-axis. It can be seen that the platform reaches a Roll angle (x-axis) of about 5.5 degrees. The platform attempts to return to neutral (centre) position after the applied signal goes to zero but there is some platform motion caused in the opposite direction. This motion is explained again by the washout process being is a trade-off between optimal workspace utilisation and prevention of false motion cues. The performance seems poor since the applied motion cues lasted just 0.25 seconds but it should be noted that the applied test signal of dual positive and negative ramp is fairly challenging. A signal of this nature is unlikely to be encountered in the actual output of the vehicle angular velocity vector $\vec{\omega}_{veh}$.

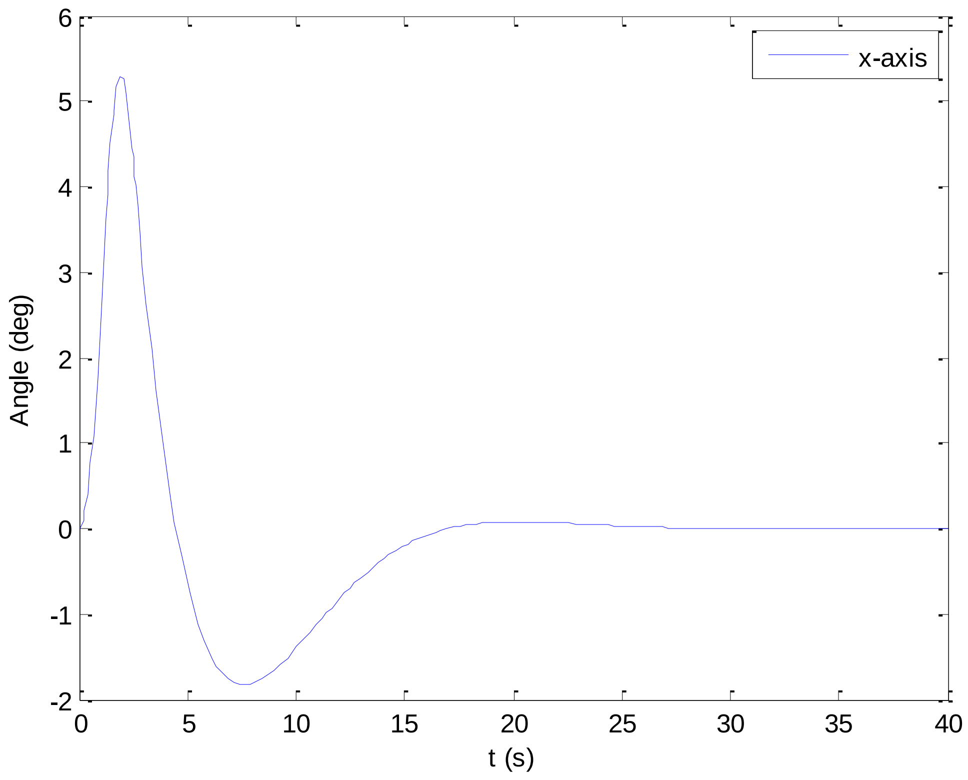


Figure 4-19 Platform Orientation for the x-axis

4.5 Chapter Summary

The technique of motion cueing and its usage in the replication of vehicle motion sensations was discussed. The chapter highlighted the functioning of the human vestibular system. The human vestibular system is able to sense translational and rotational motion sensations via the otolith and semi-circular canal systems respectively. Limitations in motion detection threshold of both the otolith dTH and semi-circular δTH were highlighted. These limits are used in the motion cueing algorithm to perform washout, which aims to return the platform to the neutral position without being detected by the human observer, this washout process aids in preventing steady state motion errors on actuator legs.

The classical washout algorithm was described in terms of the translational and rotational components. The transient specific force and angular velocity signals are obtained by filtering of the input signals of the vehicle specific force vector $\vec{f}_{veh}$ and angular velocity vector $\vec{\omega}_{veh}$. The filtering is performed in the inertial coordinate frame **{A}** which prevents the accumulation of motion errors on the actuator legs and successfully performs platform washout. The replication of the sustained component of the vehicle specific force vector $\vec{f}_{veh}$ is performed via platform tilt. This technique aims to exploit the otolith system, which senses linear motion via specific force. By tilting the platform, a component of the gravity vector can be used to replicate sustained specific force signals. The platform tilting was kept below the rotational channel motion detection threshold δTH; limiting this tilt ensures that the motion is not interpreted as false rotational motion.

The simulation system was setup to test the ability of the classical washout algorithm to effectively recreate vehicle motion sensations within the simulator environment and return the platform to the neutral position by performing washout. It was shown that the translation motion channel is able to replicate the motion sensations experienced in the vehicle fairly well with no false motion cues; according to the otolith model output in figure 4-15. The rotational channel is able to replicate the motion sensations experienced in the vehicle but suffers from some false cues when performing platform washout. Alternative motion cueing algorithms, such as the adaptive washout algorithm (Ariel and Sivan, 1984), optimal washout algorithm (Sivan et al., 1982) and MPC (Baseggio et al., 2011), could be used in future to mitigate false cues. It is observed that the washout process in both the translational and rotational channels were performed successfully and the platform was able to return to the neutral position after the input motion signal subsided. The washout filter parameters will need to be adjusted in the next chapter based on motion signals received from the software system and the position control system motion constraints, to ensure optimal performance on the vehicle simulator.

5 Software

This chapter discusses the software implementation for the vehicle simulator. The software package, X-Sim, which is used in the interfacing between telemetry data from the games physics engine and the motion control system, is explained. The software plugin, which is written in C++, is designed to process input game data into actuator stroke lengths and this data is sent to the actuator position control system.

A novel simulation setup in Matlab/Simulink, using the SimMechanics toolbox, is developed. This setup is used to adjust the input data scaling and filter parameters on the classical washout algorithm. The simulation setup is used for the following:

- Test that motion data from the game is replicated on the simulator platform in the Matlab/Simulink environment.
- Ensure that the position, velocity and acceleration constraints imposed on the actuators in the system are not violated.
- Test the fidelity of the system in replicating the input game data.
- Evaluate the fidelity performance of the 3-DOF system against the traditional 6-DOF system using the human vestibular system models.

The Matlab/Simulink setup is implemented in C++ for the X-Sim software package. The C++ software implementation results for actuator stroke lengths are tested against the results from the Matlab/Simulink setup to ensure the C++ software plugin implementation is correct.

5.1 X-Sim Software

In the vehicle simulator, visual cues are passed to the driver via the 3 monitors that are mounted on the platform. The game Dirt 3 is used to generate visual cues for the vehicle simulator. The game renders visual cues and provide motion cues via the built in physics engine.

X-Sim is the middleware package which is used to extract and interpret the data from the games physics engine. The X-Sim package consists of two software packages, the Extractor and Convertor. These packages are designed to run independently and could possibly run on separate machines.

The X-Sim Extractor software is used to communicate with the games physics engine and receive the telemetry data. This data includes the lateral force, longitudinal force, vertical force, roll angle, pitch angle and yaw angle. The physics data from most games are read either from shared memory or via a localhost network connection.

Input telemetry data is relayed from the X-Sim Extractor software to the X-Sim Convertor software via a TCP network connection. The use of a network connection between the two software packages allows them to run on independent machines. This network connection may be required depending on the capabilities of the machine running the game. When a lower performance machine is used to run the game then it is intuitive to run the X-Sim Convertor software on a separate machine. This setup reduces any processing bottlenecks which may affect the relaying of data to the motion simulator, resulting in delayed motion cues. When using a high performance machine, both software packages could run on the same machine. Figure 5-1 illustrates the network configuration used for this particular motion simulator application; it can be seen that both the TCP connections are done on the localhost machine.

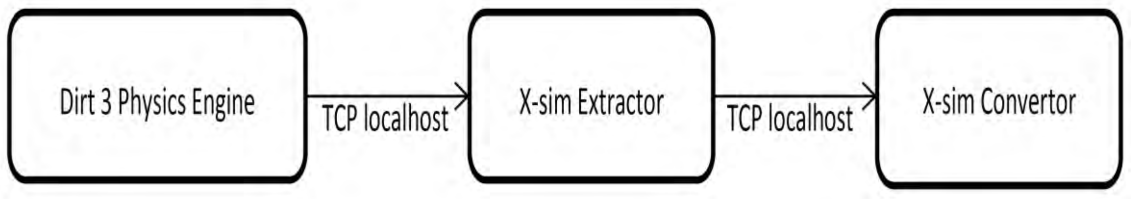


Figure 5-1 X-Sim Software Setup

The X-Sim Convertor software is able to receive telemetry data from the X-Sim Extractor software in real time. The X-Sim Convertor software is then able to perform maths functions on the data. This data processing is used in applications were the data needs to be scaled or filtered and kinematic analysis needs to be performed.

The X-Sim Convertor software is able to output data in three separate modes as follows (X-Sim, no date):

- The Universal Serial Output (USO) – Processed data is transferred to the position control system hardware via an RS232/RS485 serial interface or a network connection.
- Synaptrix Interface – This mode provides a motion control system that is able to control certain hardware systems, allowing for sensing and controlling the hardware directly.

- CSV file – Data is logged to a CSV file and allows the user to perform data analyses. The data logged could be input game data or data which has been processed by the X-Sim Convertor software.

5.1.1 Game Telemetry Data

The testing of the input data received from the game was done by logging the data to CSV file for the 3 translational values (Lateral, Longitudinal and Vertical specific forces) and 3 rotational values (Roll, Pitch and Yaw angles). The output values from the X-Sim Extractor software have a 32-bit signed range. Figure 5-2 illustrates the game data for the longitudinal specific force for a single lap of the game, it highlights the various instances in time were the vehicle undergoes acceleration and deceleration during the lap.

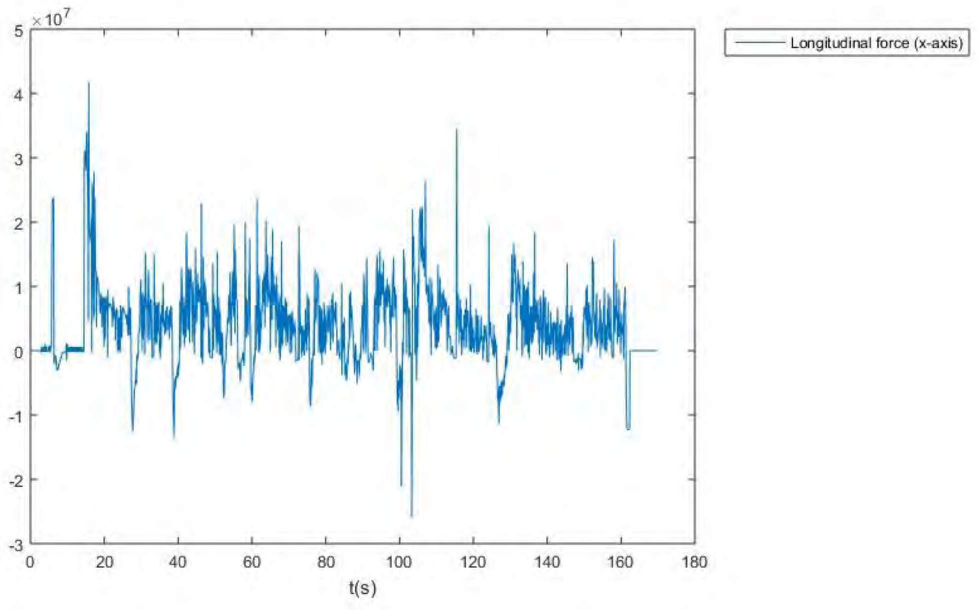


Figure 5-2 Longitudinal Force Game Data

Figure 5-3 illustrate the game data for the pitch angle values for a single lap of the game, it shows the various motions of the car rotating about the y-axis.

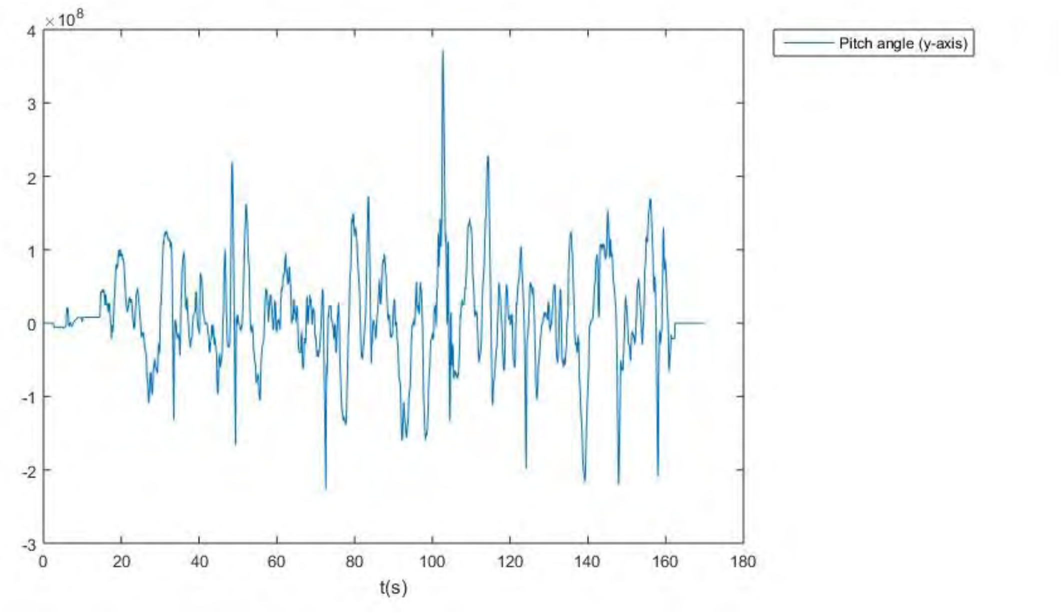


Figure 5-3 Pitch Angle Game Data

Angular velocity values are required in the classical washout algorithm for the rotational channel. The Roll, Pitch and Yaw angle values from the input game data had to be scaled and then transformed into angular velocity signals. The following transformation was used:

$$\vec{\omega}_{veh} = S\dot{\vec{\varphi}}_{veh} \; ... \; (5.1)$$

with S the transformation matrix from RPY angle rates $\dot{\vec{\varphi}}_{veh}$ into vehicle angular velocity $\vec{\omega}_{veh}$ given by:

$$S = \begin{bmatrix} 1 & 0 & -\sin\beta \\ 0 & \cos\alpha & \sin\alpha\cos\beta \\ 0 & -\sin\alpha & \cos\alpha\cos\beta \end{bmatrix} \; ... \; (5.2)$$

RPY angle rates $\dot{\vec{\varphi}}_{veh}$, used above, are obtained from RPY angles $\vec{\varphi}_{veh}$ by the following transfer function which was shown previously (Reid and Nahon, 1985):

$$\dot{\vec{\varphi}}_{veh} = \frac{12.5}{s+12.5}\vec{\varphi}_{veh} \; ... \; (5.3)$$

5.1.2 Software Plugin

The X-Sim software allows for the implementation of a dll (dynamic linked library) software plugin which is used to perform processing on the input game data. The plugin is written in C++ and it is incorporated in the X-Sim Convertor software. The plugin allows for various forms of data processing such as scaling, motion cueing and inverse kinematics to be performed. Figure 5-4 illustrates the software plugin UML activity diagram which was developed to be used for the 3-DOF motion platform. The software plugin, which is shown in appendix D, was written to extract and scale the input data received from the game via the X-Sim Extractor software package. The implementation of the classical washout filter is performed in the discrete domain. The translation motion along the z-axis (Heave), rotational motion about the x-axis (Roll) and rotational motion about the y-axis (Pitch) output from the classical washout algorithm is used in the inverse kinematics analysis. Using the derived leg length equation 2.36 to equation 2.38 and constraint equation 2.28 to equation 2.30 the actuator stroke lengths for each actuator is derived and passed through to the motion control system, over a LAN, via UDP.

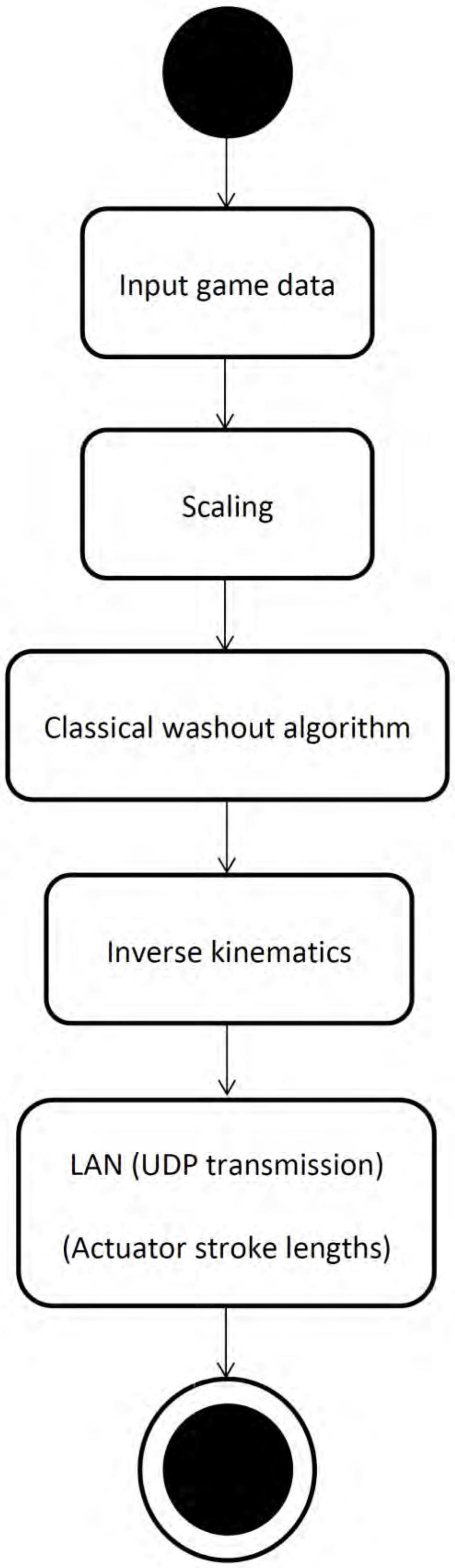


Figure 5-4 Software Plugin UML Activity Diagram

5.2 Simulations

The motion control system, discussed in chapter 3, is a pneumatic system. It employs 3 pneumatically driven linear actuators and performs position control on each of these actuators. The system itself is limited to the following actuator position, velocity and acceleration values. Table 5-1 shows the limits of the actuators.

Table 5-1 Actuator Motion Limits

Position	$\pm$ 0.1 m
Velocity	$\pm$ 0.2 m/s
Acceleration	$\pm$ 2 m/s^2

It is important to adhere to the actuator motion constraints imposed to ensure safe operation of the vehicle simulator. A failure to adhere to these limits could result in mechanical damage to the structure of the motion simulator platform and injury to the user in the vehicle simulator. Based on these constraints it is important that the vehicle input data scaling and classical washout filter parameters are selected to ensure motion that does not violate the actuator motion constraints. The objective is to get the actuator motion to be within the position, velocity and acceleration limits imposed.

In the previous work in chapters 2 and chapter 4 two simulation systems were created in the Matlab/Simulink environment. Chapter 2 created a structural model of the 3-DOF motion platform by using the SimMechanics toolbox, this toolbox creates a model of the 3-DOF motion platform based on the geometrical properties of the platform. The system was designed to perform verification of the derived inverse kinematic equation 2.36 to equation 2.38 for the 3-DOF motion platform. Kinematic analysis was performed and results from the output of the structural model were compared to the derived inverse kinematic equation 2.36 to equation 2.38, this chapter concluded by confirming the derived inverse kinematic equation 2.36 to equation 2.38 were correct. The work in chapter 4 created and implemented the motion cueing strategy in the Matlab/Simulink environment. Results from this chapter showed that the classical washout algorithm has reasonable performance in replication of motion sensations experienced in a real vehicle.

The simulation setup used for this particular chapter combines the previous two simulation setups from chapter 2 and chapter 4. A novel simulation setup is created, illustrated in figure 5-5, which is able perform the entire vehicle simulator data processing. The vehicle simulator data processing includes input data scaling, classical washout algorithm implementation and inverse kinematic analysis. The setup is used to adjust the scaling of the input data and the classical washout filter parameters to ensure the actuator motions adhere to constraints imposed on them by the selected position control system. Additionally SimMechanics creates a 3-D visual display of the motion platform, allowing for the platform motion to be viewed in real time.

Input game data received from the games physics engine (Specific forces and RPY angles) are input directly into the simulation system. These values are scaled and the RPY angles are transformed into the vehicle angular velocity vector $\vec{\omega}_{veh}$. The classical washout algorithm takes in the vehicle specific force vector $\vec{f}_{veh}$ and vehicle angular velocity vector $\vec{\omega}_{veh}$ to be used in the translational and rotational channel. Independent motion parameters (Roll, Pitch and Heave) are output from the classical washout algorithm. Independent parameter values are fed into the leg trajectory block which computes the derived leg length equation 2.36 to equation 2.38. These values are input into the structural model, which is designed using the SimMechanics toolbox. Motion from the output of the structural model is then analysed to ensure that the actuator motions do not violate the imposed motion constraints.

The simulation setup developed allows for input data scaling and filter parameters to be adjusted easily and safely in the Matlab/Simulink environment. Adjustments are made till the actuator motion constraints, in table 5-1, are adhered to. The simulation setup can then be implemented as a software plugin in the X-Sim Convertor software. Since the software has been tested, in Matlab/Simulink, it is guaranteed that the motion constraints will be adhered to by the software plugin. Adherence to these constraints ensures safety of the user and safety of the mechanical structure of the vehicle simulator.

A fidelity study is also performed in Matlab/Simulink to highlight how well the vehicle simulator, using the 3-DOF motion platform, replicates the sensations felt in a vehicle for the selected motion cueing parameters. The 3-DOF motion platform fidelity is also evaluated against the traditional 6-DOF motion platform.

The simulation setup is also used to test and verify the correct functioning of the various components in the C++ software plugin developed. The Matlab/Simulink setup was modified to be implemented in the discrete domain, since the X-Sim software samples input game data at discrete time intervals of 0.01 second.

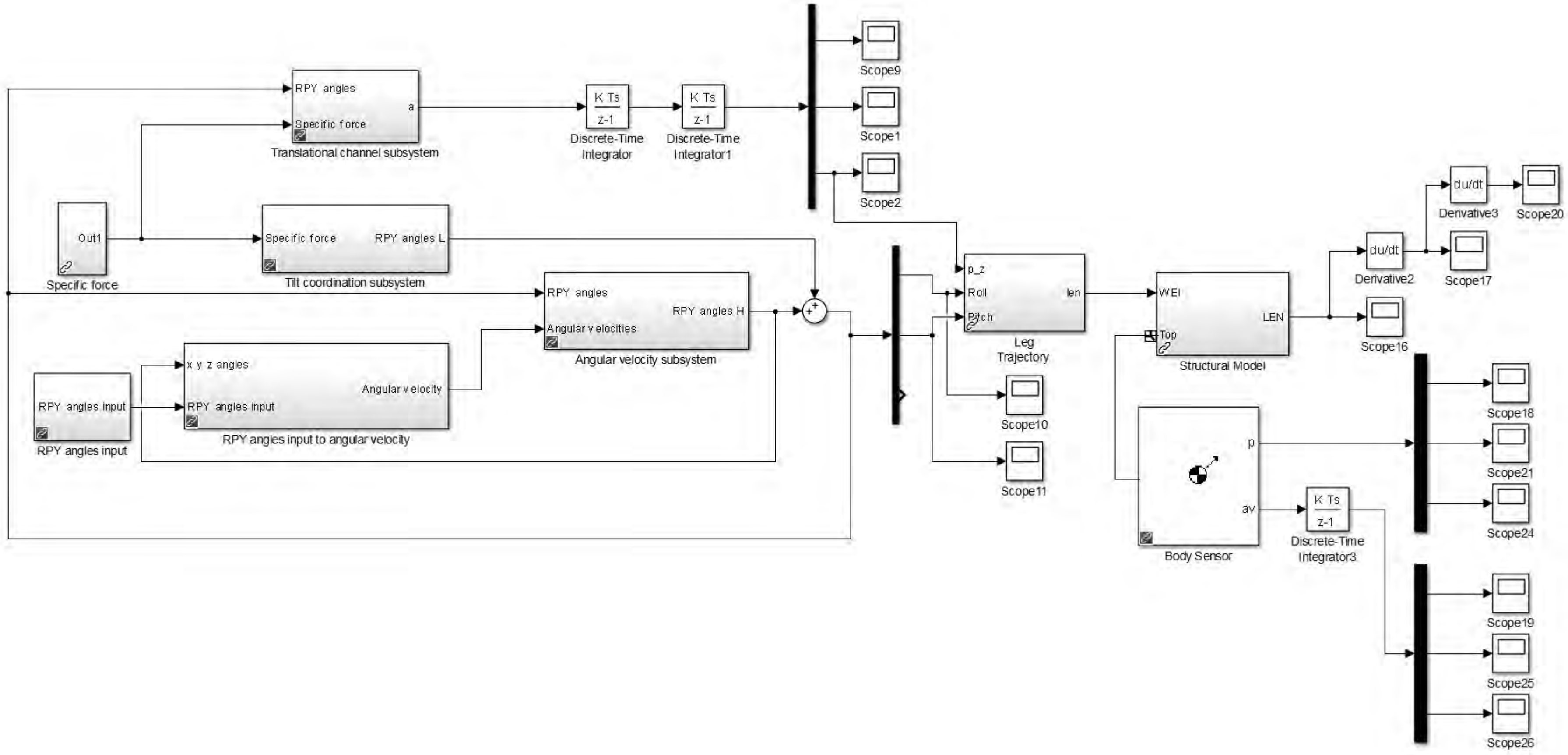


Figure 5-5 Vehicle Simulator Matlab/Simulink Simulation Setup

5.2.1 Filter Parameter Selection

The filter parameters used in the classical washout algorithm were adjusted to not violate the actuator motion constraints in table 5-1. The transient translational acceleration filter, HP Filter in figure 4-4, has the following form:

$$HP_{Translational}(s) = \frac{s^3}{(s^2+2\zeta\omega_n s+\omega_n^2)(s+\omega_b)} \quad ... \ (5.4)$$

Table 5-2 shows the filter parameters selected for $HP_{Translational}$

Table 5-2 Translational Channel Filter Parameters

ζ	1
ω_n	3.1 rad/s
ω_b	0.2 rad/s

The transient angular velocity filter, HP Filter in figure 4-5, has the following form:

$$HP_{Angular}(s) = \frac{s^2}{(s+\omega_n)^2} \quad ... \ (5.5)$$

The parameters values for $HP_{Angular}$ are shown in table 5-3.

Table 5-3 Transient Angular Velocity Filter Parameters

ζ	1
ω_n	1 rad/s

Upon initial evaluation it was found that the output signals from the transient angular velocity filters were creating large actuator accelerations. It was decided to attenuate some of the high frequency signals by passing the output signal from the transient angular velocity filters through a low-pass filter. The low-pass filter has a break frequency of ω_b = 3.1 rad/s and has the following form:

$$LP_{Angular}(s) = \frac{\omega_b}{(s+\omega_b)} \quad ... \ (5.6)$$

The low-pass filter, LP Filter in figure 4-5, used in the tilt coordination process has the following form:

$$LP_{Tilt}(s) = \frac{\omega_n^2}{s^2+2\zeta\omega_n s+\omega_n^2} \quad ... \ (5.7)$$

The parameter values for LP_{Tilt} are given in table 5-4.

Table 5-4 Tilt Coordination Filter Parameters

ζ	1
ω_n	1 rad/s

5.2.2 Discrete Filter Implementation

To be able to implement the classical washout algorithm in the C++ software plugin the filter design had to be done in the discrete domain. The sampling time for the X-Sim software is 0.01 second per output for each reading of actuator stroke lengths. The Matlab/Simulink libraries were modified to implement the simulation in the discrete domain.

The bilinear transform was used with the following approximation for continuous time to discrete time conversion:

$$s \approx \frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}} \;... \; (5.8)$$

With T the sampling time of 0.01 second.

The transient translational acceleration filter in the discrete domain is designed using filter parameters in table 5-2 and with a sample time of 0.01 second. The transfer function in the discrete domain is given by:

$$HP_{Translational}(z) = \frac{Y(z)}{X(z)} = \frac{0.004844z^3-0.004844z^2-0.004844z+0.004844}{z^3-2.937z^2+2.875z-0.938} \;... \; (5.9)$$

Implementing this transfer function in the form of a difference equation yields the following:

$$y[n] = 0.938y[n-3] - 2.875y[n-2] + 2.937y[n-1] + 0.004844x[n-3] - 0.004844x[n-2] - 0.004844x[n-1] + 0.004844x[n] \;... \; (5.10)$$

The transient angular velocity filter in the discrete domain is designed using filter parameters in table 5-3 and with a sample time of 0.01 second. The transfer function in the discrete domain is given by:

$$HP_{Angular}(z) = \frac{Y(z)}{X(z)} = \frac{0.9901z^2-1.98z+0.9901}{z^2-1.98z+0.9802} \;... \; (5.11)$$

Implementing this transfer function in the form of a difference equation yields the following:

$$y[n] = -0.9802y[n-2] + 1.98y[n-1] + 0.9901x[n-2] - 1.98x[n-1] + 0.9901x[n] \ \ldots \text{ (5.12)}$$

The low-pass filter added to the rotational channel in the discrete domain is given by:

$$LP_{Angular}(z) = \frac{0.01526z+0.01526}{z-0.99} \ \ldots \text{ (5.13)}$$

Implementing this filter in the form of a difference equation yields the following:

$$y[n] = 0.99y[n-1] + 0.01526x[n] + 0.01526x[n-1] \ \ldots \text{ (5.14)}$$

The low-pass filter used in the tilt coordination process in the discrete domain is designed using filter parameters in table 5-4 and with a sample time of 0.01 second. The transfer function in the discrete domain is given by:

$$LP_{Tilt}(z) = \frac{2.475x10^{-5}z^2+4.95x10^{-5}z+2.475x10^{-5}}{z^2-1.98z+0.9802} \ \ldots \text{ (5.15)}$$

Implementing this filter in the form of a difference equation yields the following:

$$y[n] = -0.9802y[n-2] + 1.98y[n-1] + 2.475x10^{-5}x[n-2] + 4.95x10^{-5}x[-1] + 2.475x10^{-5}x[n] \ \ldots \text{ (5.16)}$$

5.2.3 Kinematic Analysis Results

The first part of the testing used the Matlab/Simulink setup. The input game is scaled and processed with the classical washout algorithm. Upon completion the three independent parameters, translational motion along the z-axis (Heave) and rotation about the x-axis (Roll) and y-axis (Pitch) are input into the inverse kinematics system. Using the three independent parameters and the constraint equation 2.28 to equation 2.30 the derived leg length equation 2.36 to equation 2.38 are formed. As in chapter 2 the output trajectory from the structural model can be compared to the input trajectory to determine the accuracy of the derived leg length equation 2.36 to equation 2.38 and the effectiveness of the classical washout algorithm can also be determined from the output trajectory.

Figure 5.6 illustrates the roll angle (x-axis) comparison; this graph shows the comparison between the input and output roll angle trajectory. It can be seen that the output trajectory, for the roll, replicates the input trajectory extremely well. The maximum roll angle achieved for the selected motion cueing parameters in the system is 4.5 degrees. Figure 5-7 illustrates the error between the input and output trajectory for the roll angle. The maximum error is 0.2863 degrees and the mean error is 0.0274 degrees.

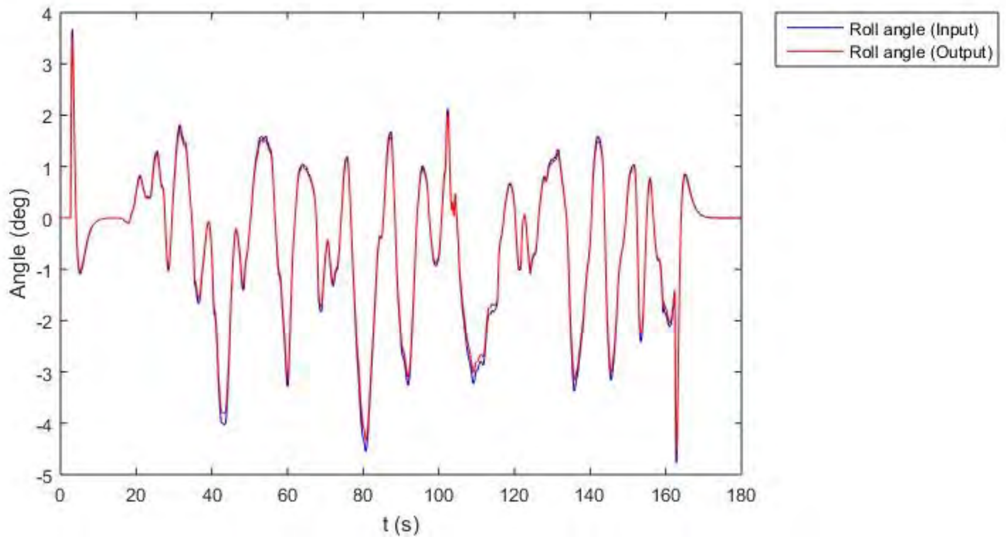


Figure 5-6 Roll Angle Comparison

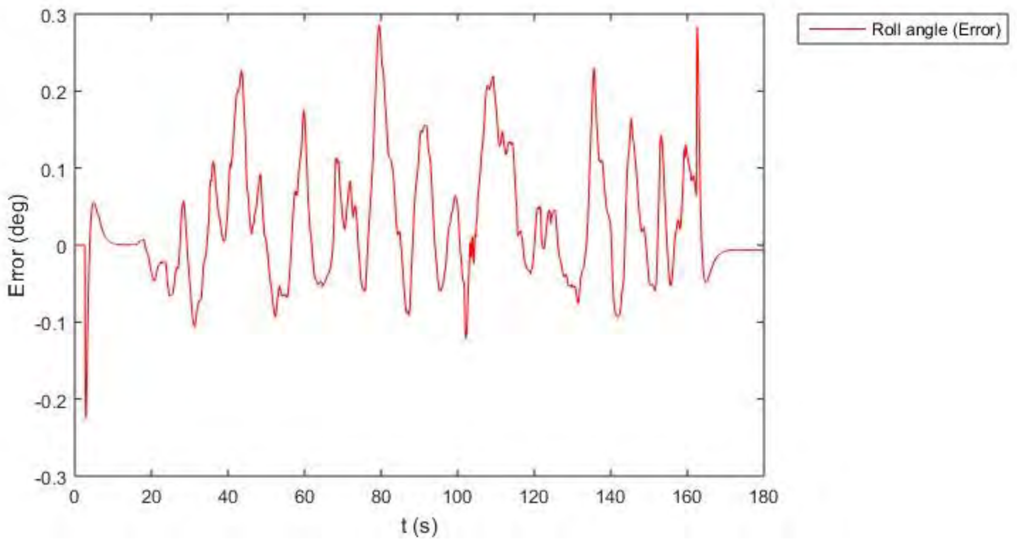


Figure 5-7 Roll Angle Error

Figure 5-8 illustrates the pitch angle (y-axis) comparison. It can be noted that the output trajectory, for the pitch angle, replicates the input trajectory extremely well. The maximum pitch angle achieved for the selected motion cueing parameters in the system is 5.4 degrees. Figure 5-9 illustrates the error between the input and output trajectory for the pitch motion. The maximum error is 0.2170 degrees and the mean error is 0.0290 degrees.

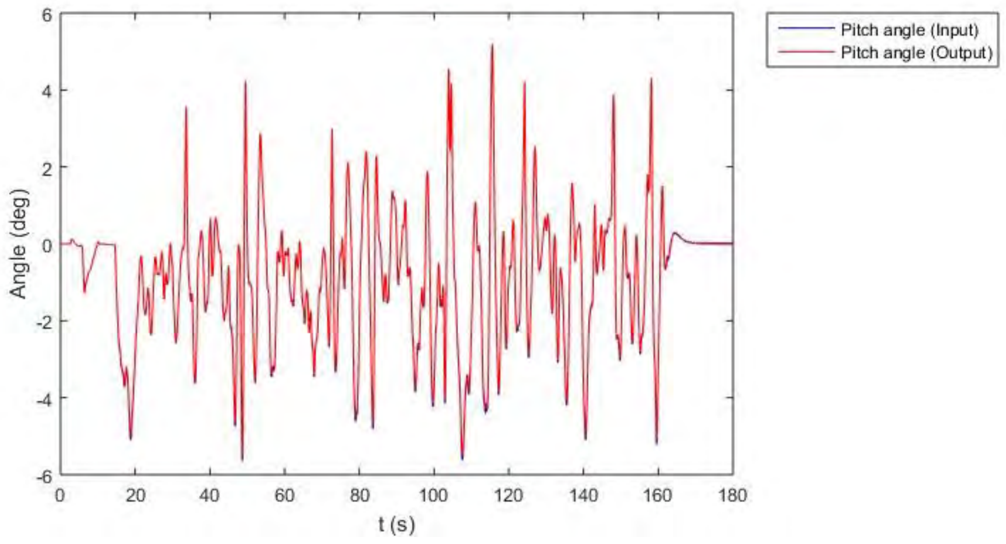


Figure 5-8 Pitch Angle Comparison

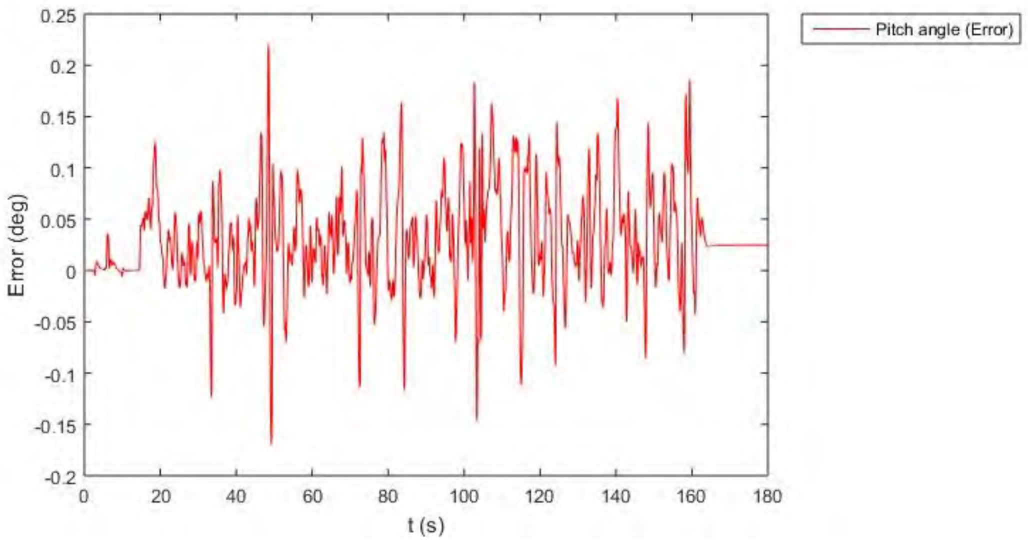


Figure 5-9 Pitch Angle Error

Figure 5-10 illustrates the heave (z-axis) motion comparison. The heave motion of the output replicates the heave input well; there is some difficulty in replicating the larger heave motions. The maximum heave motion achieved, at the output, for the selected motion cueing parameters in the system is 0.0083 m. Figure 5-11 illustrates the error between the input and output trajectories for the heave motion. The maximum error is 0.0013 m and the mean error is 0.00025 m.

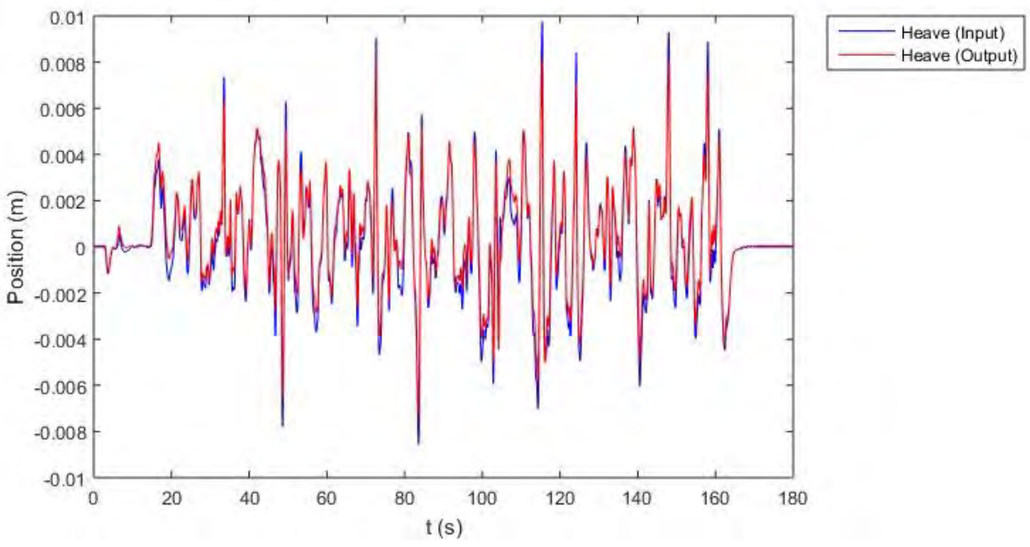


Figure 5-10 Heave Comparison

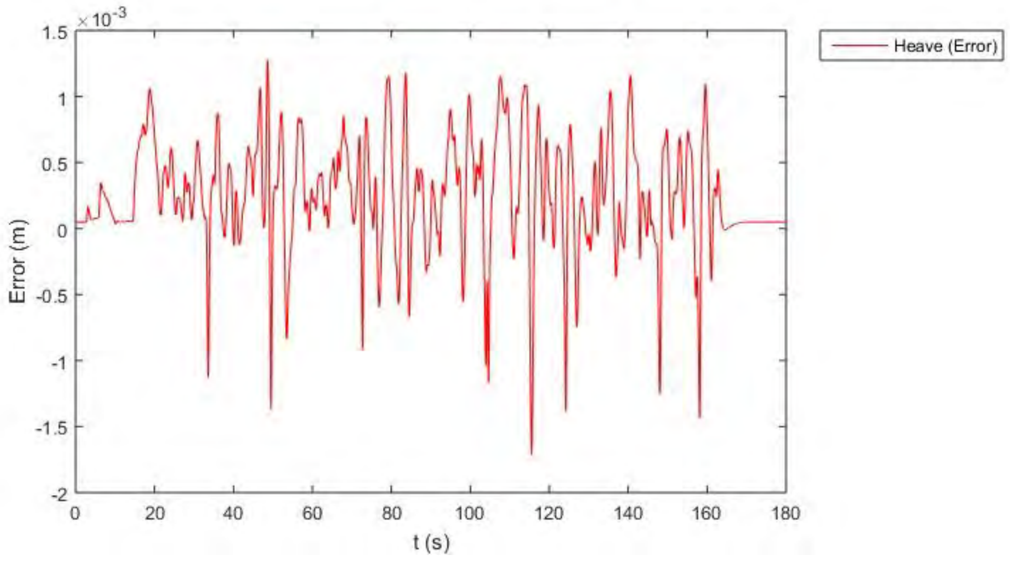


Figure 5-11 Heave Error

The errors between the input and output trajectory are reasonable and not extreme. All the outputs track their respective inputs well. These results validate the derived leg length equation 2.36 to equation 2.38. It can also be seen that the classical washout algorithm is effective in ensuring the platform returns to neutral position (washout) for the entire duration of the lap. Roll, pitch and heave motions all return to zero when the applied motion cue is complete. The maximum values achieved for the roll angle, pitch angle and heave motion represent reasonable performance for a low-cost simulator used for research purposes.

Figure 5-12 and figure 5-13 illustrates the 3-D model for the 3-DOF motion platform for two instances in the simulation; this system, using SimMechanics, provides a visualisation tool of expected platform motions. It can also verify platform motion is correct, for e.g. the platform should tilt backwards about the y-axis (Pitch) to replicate a positive acceleration about the x-axis as in figure 5-12. In figure 5-13 the driver is going around a curve and turning left, therefore the driver is tilted to the right to replicate the motion sensations experienced in the vehicle.

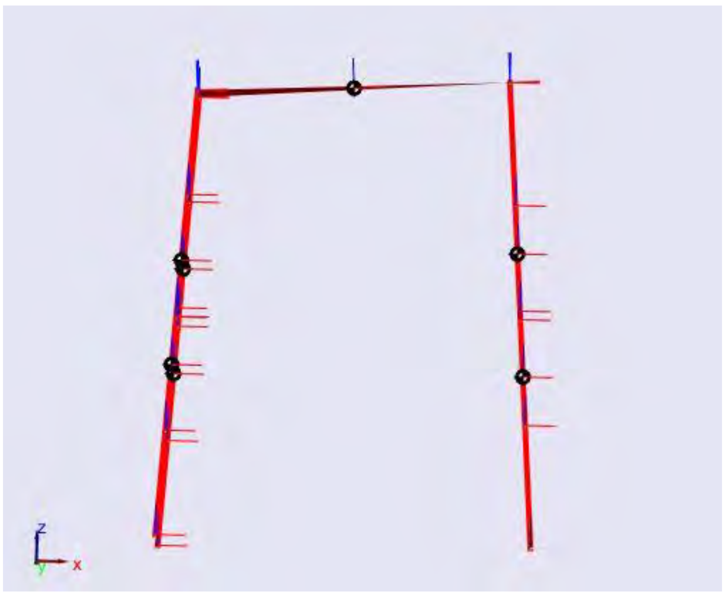


Figure 5-12 3-DOF Platform Backward Tilt

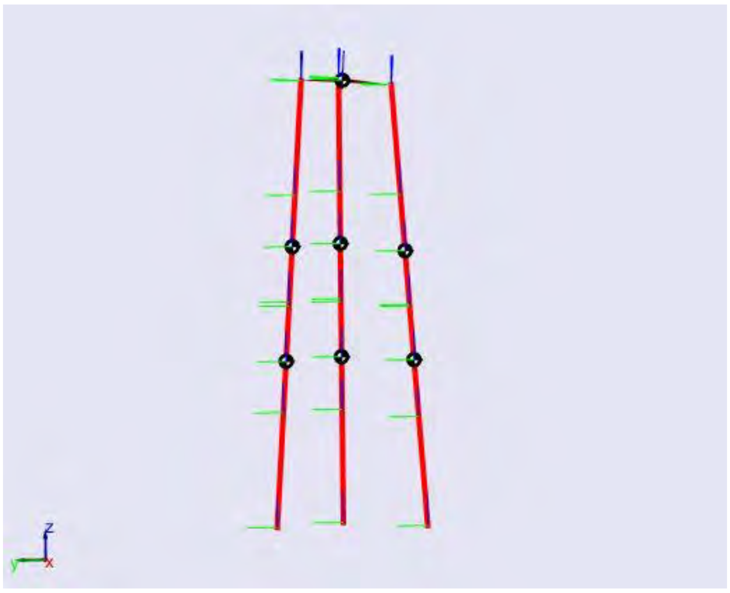


Figure 5-13 3-DOF Platform Tilt to Right

5.2.4 Motion Limit Results

The next aspect of the testing evaluated the actuator motion limits from the structural model to determine if the actuator motions adhere to the constraints. The filter parameters for all the channels in the washout algorithm were selected to ensure that the actuator motion limits in table 5-1 is not violated. The results from this test helped ensure a successful implementation of the motion scaling, classical washout algorithm and inverse kinematics on the actual 3-DOF motion platform. It also guarantees safety of the user and the mechanical structure of the vehicle simulator.

Figure 5-14 illustrates the actuator stroke length output from the structural model for each of the actuators. Actuator 1 has a maximum motion change of 0.0458 m, actuator 2 has a maximum motion change of 0.0766 m and actuator 3 has a maximum motion change of 0.0848 m. Based on these results it can be concluded that all three actuators adhere to the maximum possible actuator position limit of 0.1 m.

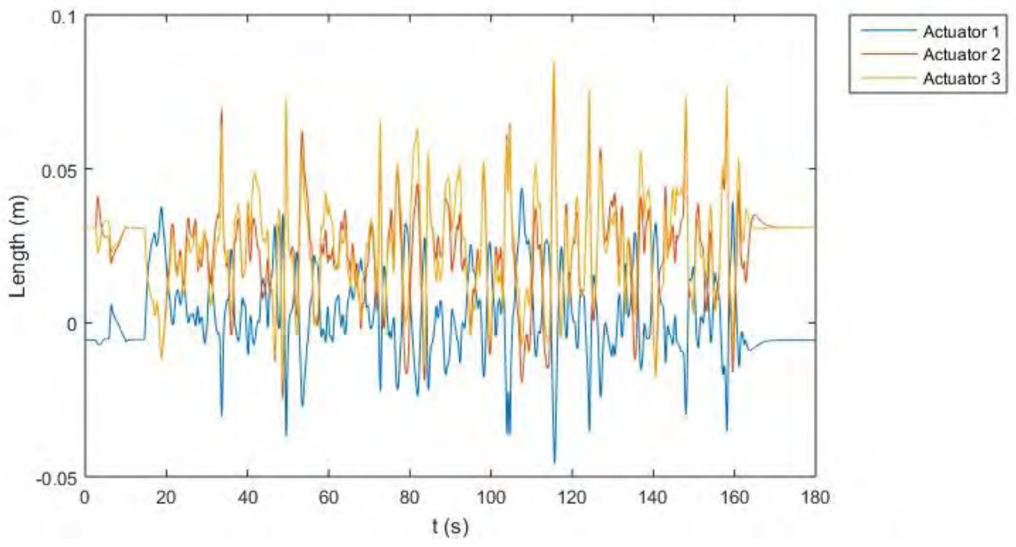


Figure 5-14 Structural Model Length Output of Actuators

Figure 5-15 illustrates the velocity output of each actuator from the structural model. Actuator 1 has a maximum velocity of 0.1178 m/s, actuator 2 has a maximum velocity of 0.1580 m/s and actuator 3 has a maximum velocity of 0.1570 m/s. The actuator velocity values lie within the maximum permissible actuator velocity of 0.2 m/s.

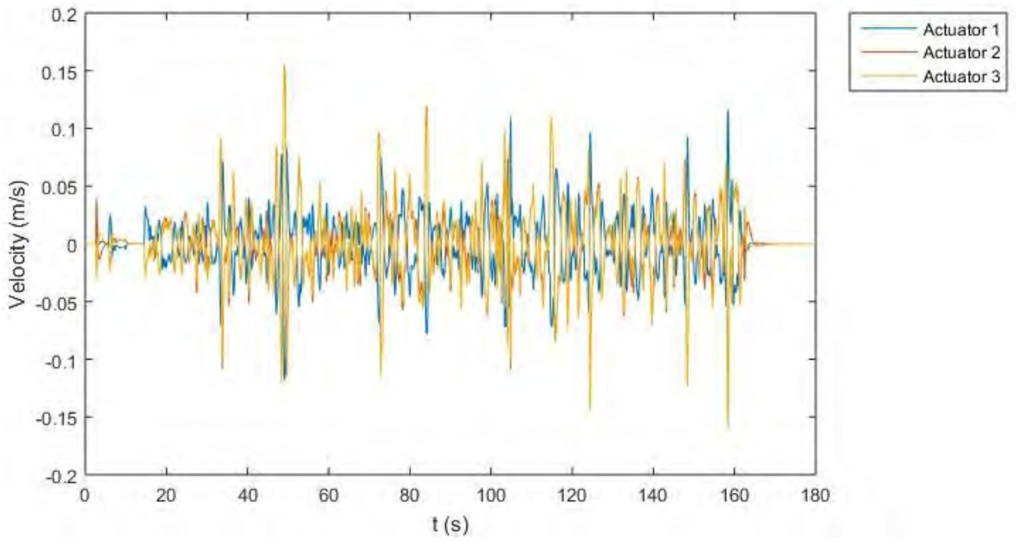


Figure 5-15 Structural Model Velocity Output of Actuators

Figure 5-16 illustrates the acceleration output of each actuator from the structural model. Actuator 1 has a maximum acceleration of 1.1643 m/s^2, actuator 2 has a maximum acceleration of 1.5375 m/s^2 and actuator 3 has a maximum acceleration of 1.5230 m/s^2. The accelerations of all actuators lie within the maximum acceleration value of 2 m/s^2.

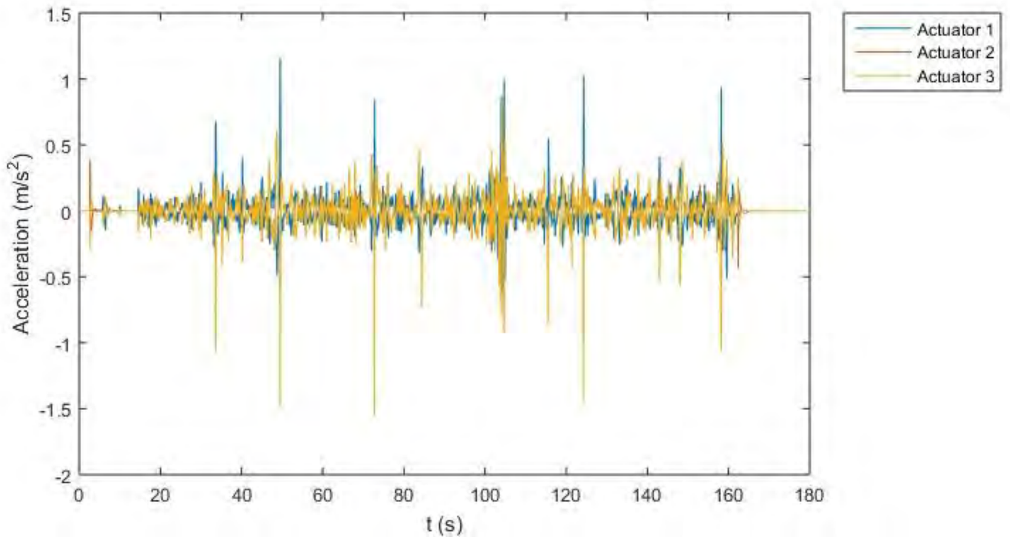


Figure 5-16 Structural Model Acceleration Output of Actuators

The input data scaling and filter parameters selected ensured that none of the actuator motion constraints, in table 5-1, are violated. These constraints ensure safe operation for the driver in the vehicle simulator and safety of the mechanical structure for the vehicle simulator. It can be seen from the acceleration of the actuator legs that the actuators reach maximum acceleration for very short periods in time. This platform acceleration is due to the fixed filter parameters of the classical washout algorithm which is designed for the worst motion case. Therefore platform workspace usage is not optimised for the classical washout algorithm.

5.2.5 Simulator Fidelity Results

Simulator fidelity is the next aspect which is evaluated for the vehicle simulator. To evaluate the vehicle simulator fidelity the otolith model is used for the translational motion and the semi-circular canal model is used for the rotational motion.

Figure 5-17 illustrates the sensed specific forces in the vehicle and the vehicle simulator along the x-axis. It can be seen that the vehicle simulator does reasonably well to replicate the motion sensations felt in the vehicle, with some false motion cues. For the vehicle simulator used, which is a 3-DOF system, only the sustained specific forces along the x-axis will be felt. This sensation is represented by tilt coordination along the y-axis (Pitch); here the motion is kept below the y-axis semi-circular canal motion detection threshold δTH to prevent false rotational motion cues. The transient component of vehicle specific force along the x-axis is not used in the 3-DOF vehicle simulator.

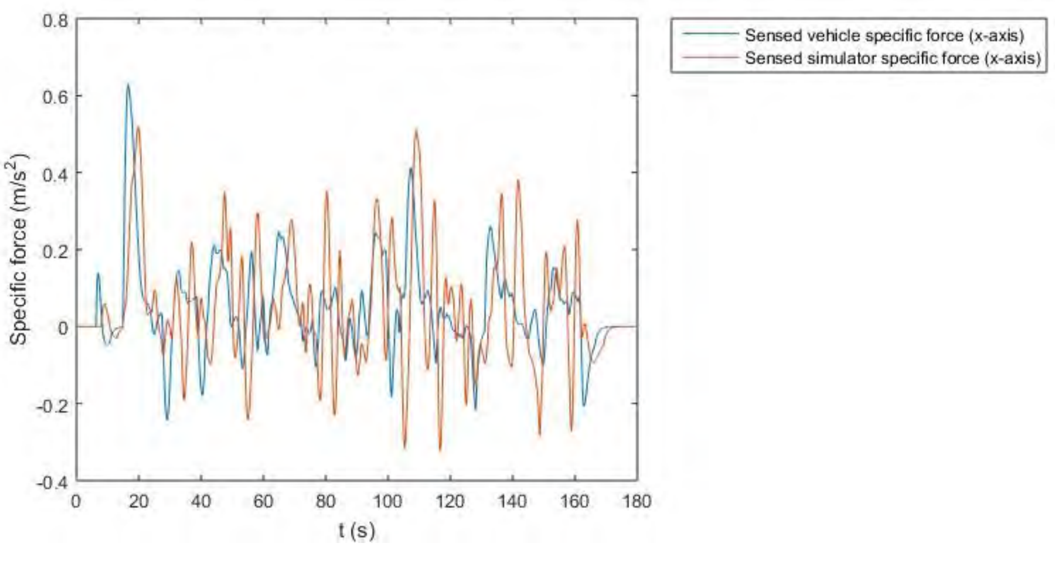


Figure 5-17 Sensed Vehicle and Simulator Specific Force by the Otolith Model (x-axis)

Figure 5-18 illustrates the sensed specific forces in the vehicle and the vehicle simulator along the y-axis. It can be seen that the vehicle simulator replicates, extremely well, the motion sensations felt in the vehicle for this channel. For the vehicle simulator used, which is a 3-DOF system, only the sustained specific forces along the y-axis will be felt. The sustained specific forces along the y-axis are represented by tilt of the motion platform along the x-axis (Roll).

Tilting along the x-axis is kept below the semi-circular canal motion detection threshold δTH to prevent false rotational motion cues along the x-axis (Roll). The transient component of vehicle specific force along the y-axis is not used in the 3-DOF vehicle simulator.

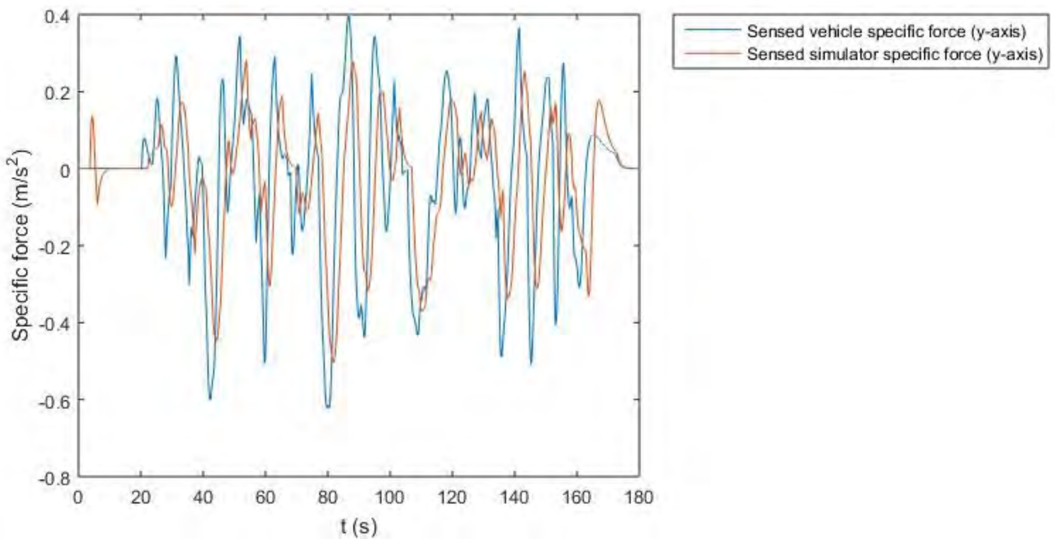


Figure 5-18 Sensed Vehicle and Simulator Specific Force by the Otolith Model (y-axis)

Figure 5-19 illustrates the sensed specific forces in the vehicle and the vehicle simulator along the z-axis. It can be seen that the vehicle simulator does well to replicate the motion sensations felt in the vehicle. The transient component of vehicle specific force along the z-axis (Heave) will be felt in the vehicle simulator.

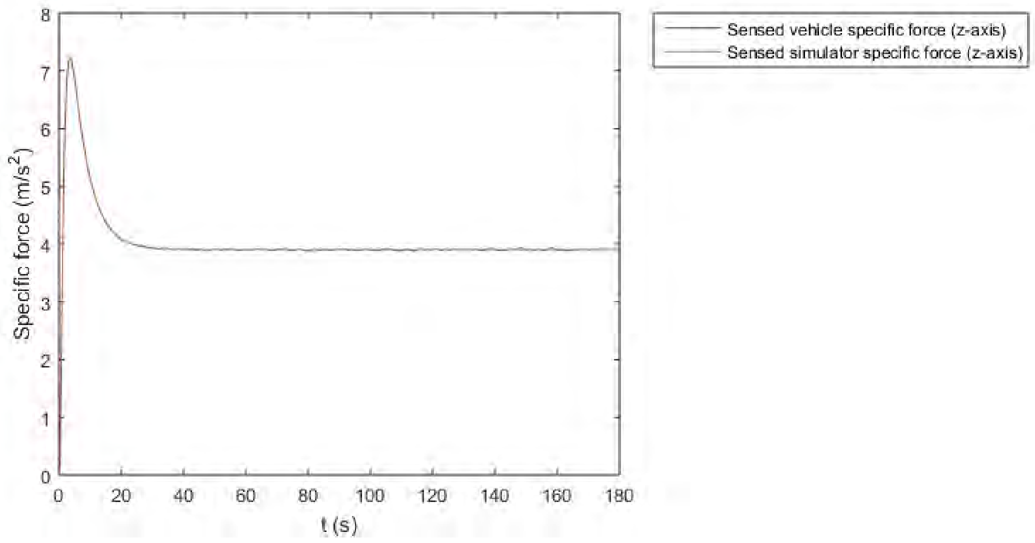


Figure 5-19 Sensed Vehicle and Simulator Specific Force by the Otolith Model (z-axis)

Figure 5-20 illustrates the sensed angular velocity in the vehicle and the vehicle simulator along the x-axis. The motion sensations along this channel are minimal and the vehicle simulator replicates an attenuated version of vehicle angular velocity along the x-axis (Roll).

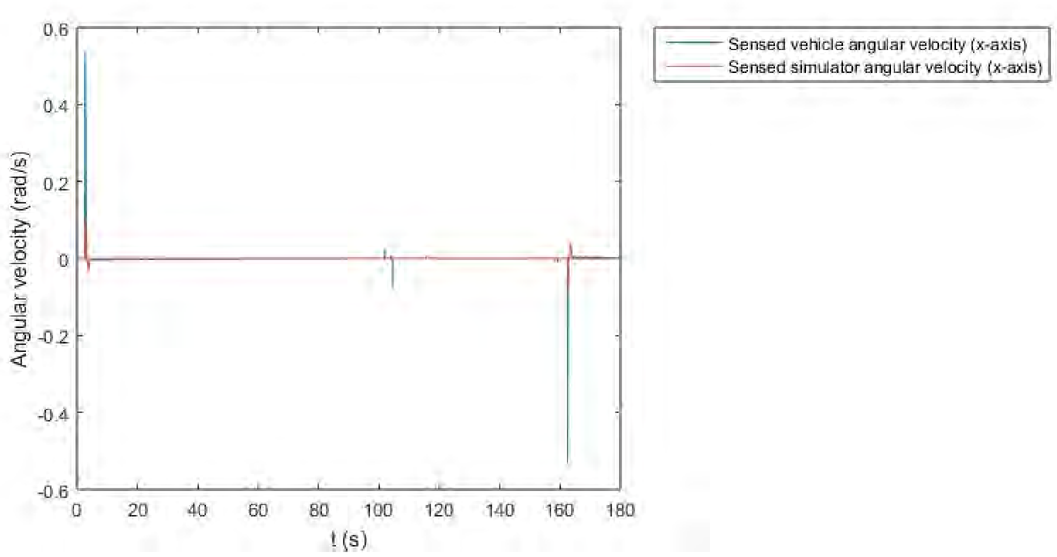


Figure 5-20 Sensed Vehicle and Simulator Angular Velocity by the Semi-circular Canal Model (x-axis)

Figure 5-21 illustrates the sensed angular velocity in the vehicle and the vehicle simulator along the y-axis. The motion sensations in the vehicle simulator are attenuated but represent the motion sensations felt in the vehicle reasonably well.

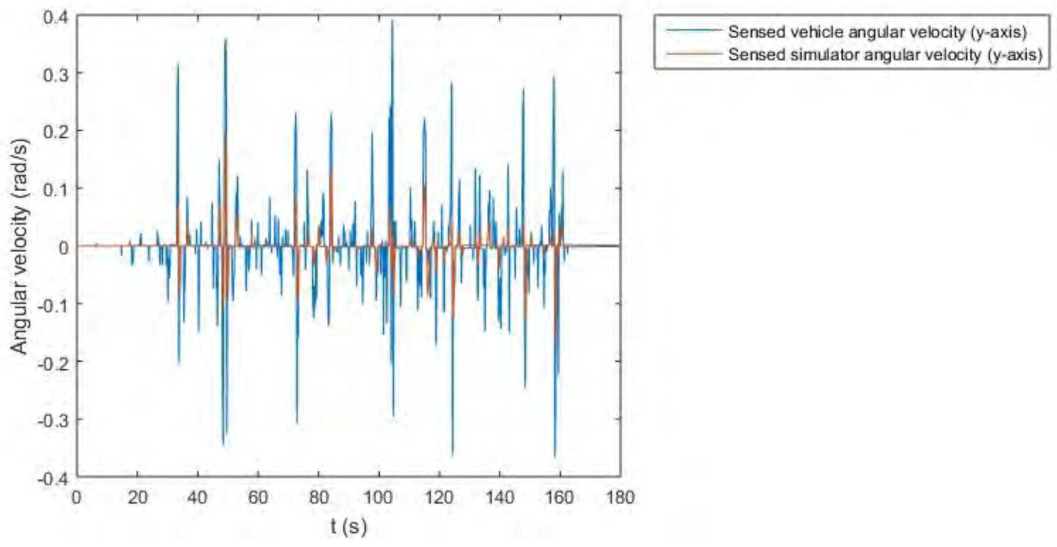


Figure 5-21 Sensed Vehicle and Simulator Angular Velocity by the Semi-circular Canal Model (y-axis)

The vehicle simulator designed is a 3-DOF system. It is used to replicate sustained translational accelerations along the x-axis and y-axis; this replication is done via tilt coordination. It is also used to replicate the transient translational acceleration along the z-axis (Heave) and transient rotational motion about the x-axis (Roll) and y-axis (Pitch) in the vehicle simulator.

5.2.6 Performance Evaluation of 3-DOF Motion Platform

The 3-DOF vehicle simulator used is a system that is not commonly used for vehicle simulators. Generally the traditional 6-DOF motion platform is used for vehicle simulators. However it is fairly well known that these systems have excessive costs attached to them. These systems have typically been funded by car manufacturers and transport departments to perform various research and training; these developers typically have excessive budgets making the 6-DOF systems feasible. By evaluating the performance of the 3-DOF system against the 6-DOF motion platform developed in chapter 2 the benefits of such a system becomes apparent, particularly in scenarios were cost is a major factor and reasonable performance is needed.

Figure 5-22 shows the sensed specific force, along the x-axis, by the otolith model for both 3-DOF and 6-DOF systems. These signals are compared to the sensed specific force, along the x-axis, in the actual vehicle. Both systems perform reasonably well with a fair amount of false motion cues. The 3-DOF system actually has smaller sensations felt for false motion cues, which is attributed to the lack of transient translational acceleration for the x-axis component.

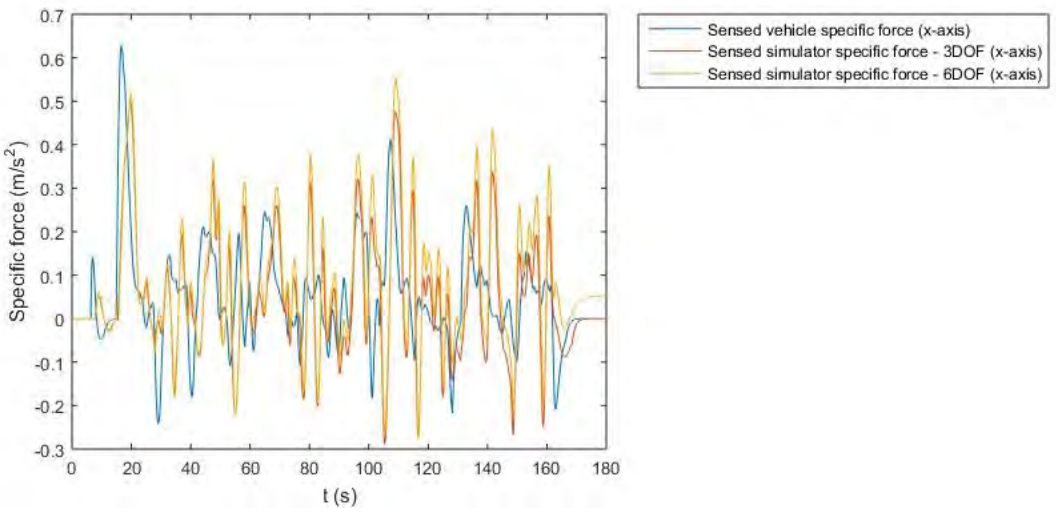


Figure 5-22 Specific Force Comparison (x-axis)

Figure 5-23 shows the sensed specific force, along the y-axis, by the otolith model for both 3-DOF and 6-DOF systems. These signals are compared to the sensed specific force, along the y-axis, in the actual vehicle. Both systems perform extremely well with minimal amount of false motion cues. The 6-DOF system performs slightly better, specifically at replicating the transient accelerations since the 3-DOF system lacks the transient component.

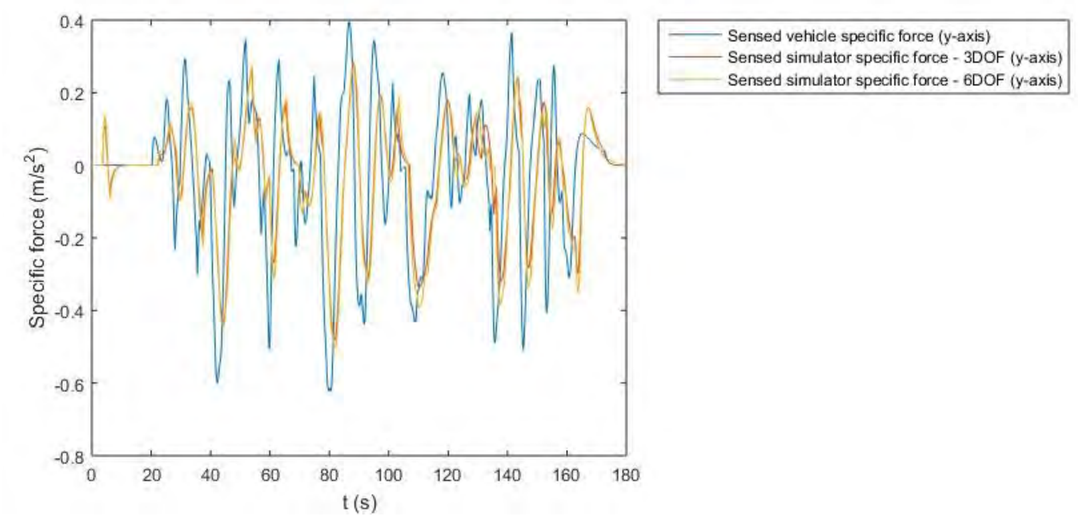


Figure 5-23 Specific Force Comparison (y-axis)

Figure 5-24 shows the sensed specific force, along the z-axis, by the otolith for both 3-DOF and 6-DOF systems. These signals are compared to the sensed specific force, along the z-axis, in the actual vehicle. Both systems perform well in replication of motion sensations felt along the z-axis.

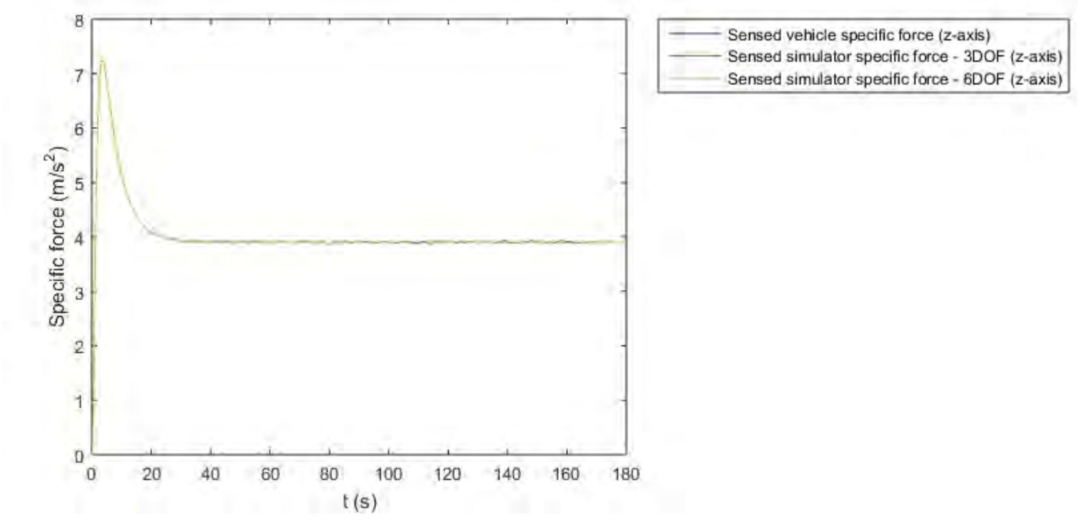


Figure 5-24 Specific Force Comparison (z-axis)

Figure 5-25 shows the sensed angular velocity, along the x-axis, by the semi-circular canal model for both 3-DOF and 6-DOF systems. These signals are compared to the sensed angular velocity, along the x-axis, in the actual vehicle. The motion for this component is minimal and both systems replicate an attenuated signal with some small false motion sensations. Figure 5-26 illustrates the motion cues experienced in both systems for a small time period. It shows that both components replicate an attenuated signal, the 6-DOF system performs slightly better.

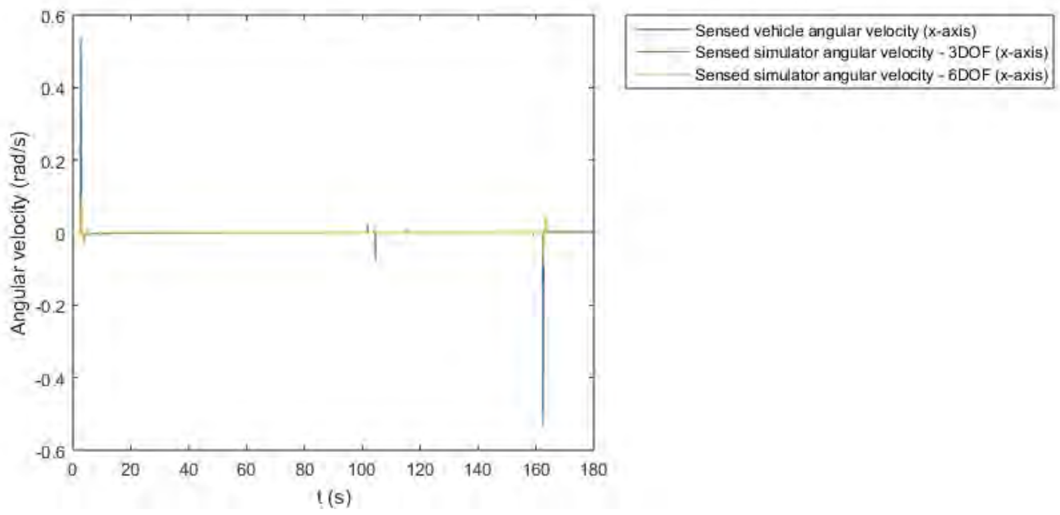


Figure 5-25 Angular Velocity Comparison (x-axis)

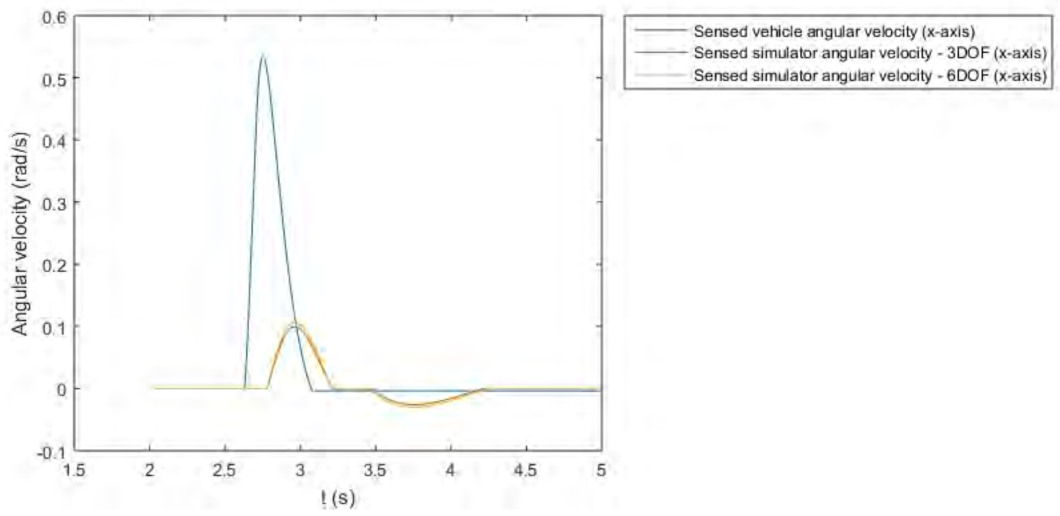


Figure 5-26 Angular Velocity Sample Comparison (x-axis)

Figure 5-27 shows the sensed angular velocity, along the y-axis, by the semi-circular canal model for both 3-DOF and 6-DOF systems. These signals are compared to the sensed angular velocity, along the y-axis, in the actual vehicle. The performance of motion replication for this component is modest in both systems. Figure 5-28 illustrates the motion cues experienced in both systems for a small time period. It shows that both components replicate an attenuated signal, the 6-DOF system performs slightly better. There are some motion cues which are missed completely in both systems.

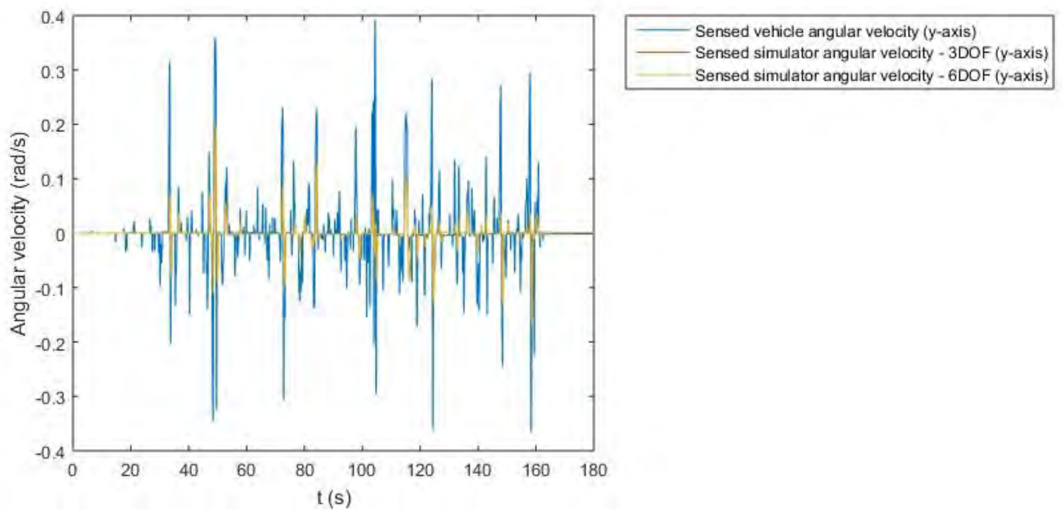


Figure 5-27 Angular Velocity Comparison (y-axis)

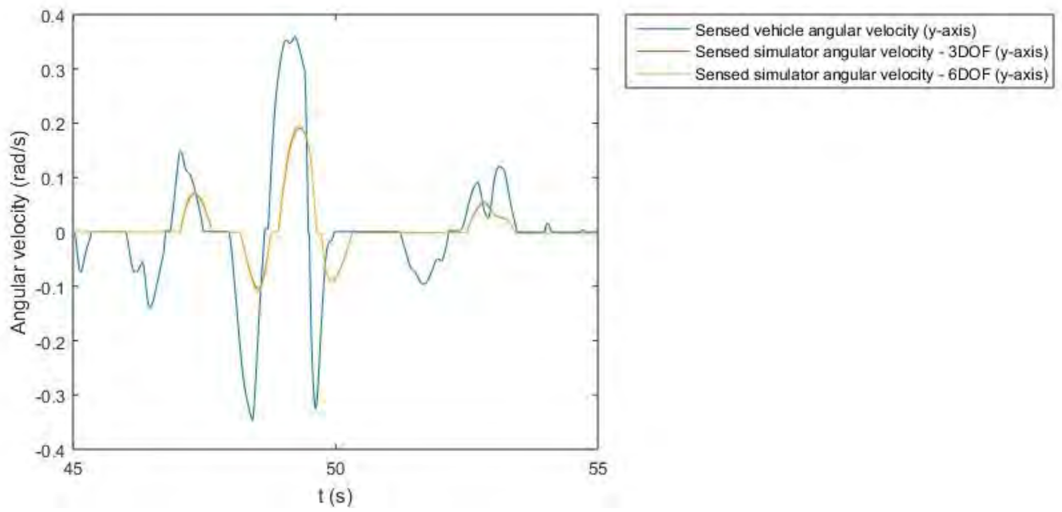


Figure 5-28 Angular Velocity Sample Comparison (y-axis)

Figure 5-29 shows the sensed angular velocity, along the z-axis, for both 3-DOF and 6-DOF systems. These signals are compared to the sensed angular velocity, along the z-axis, in the actual vehicle. The 3-DOF system has no motion for this component while the 6-DOF system replicates an attenuated signal for the sparse motion sensations felt for this component. Figure 5-30 illustrates the motion cues experienced in both systems for a small time period. It shows the 6-DOF system replicates an attenuated signal for the motion cues felt in the actual vehicle. There are some motion cues which are missed completely in the 6-DOF system.

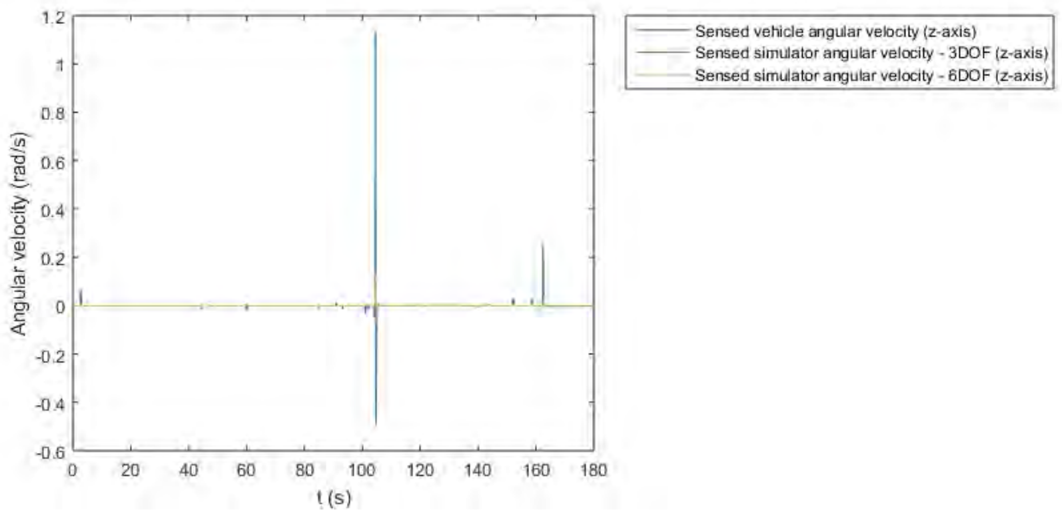


Figure 5-29 Angular Velocity Comparison (z-axis)

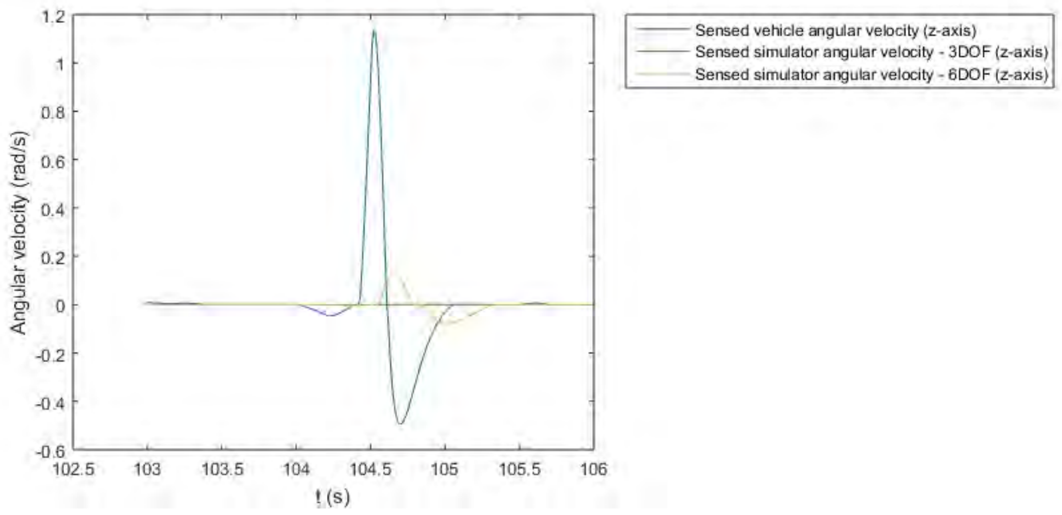


Figure 5-30 Angular Velocity Sample Comparison (z-axis)

For the modest motions required by the motion control system, in table 5-1, the classical washout filter parameters were optimised. Using this classical washout algorithm implementation on both the 3-DOF and 6-DOF systems the performance output shows that there is merit for the 3-DOF motion system. For modest motion requirements the 3-DOF system is more than capable of replicating the more costly 6-DOF system. It only ever fails to replicate rotational motion along the z-axis (Yaw) at all and from figure 5-29 this motion cue is of not much significance. When a low-cost solution with modest performs requirements is desired then the 3-DOF motion platform would be the best choice.

5.2.7 Software Plugin Testing

The final part of the testing was used to verify the correct functioning of the C++ software plugin in implementing the input data scaling, classical washout algorithm and inverse kinematics. The software plugin actuator outputs, which were saved to CSV file, are shown.

Figure 5-31 illustrates the actuator 1 stroke length output for the software plugin and the Matlab/Simulink simulation. The software plugin output for actuator 1 does well to replicate the output from the Matlab/Simulink simulation. The maximum error value between the actuator 1 value in the software plugin and in the Matlab/Simulink simulation is -0.0134 m.

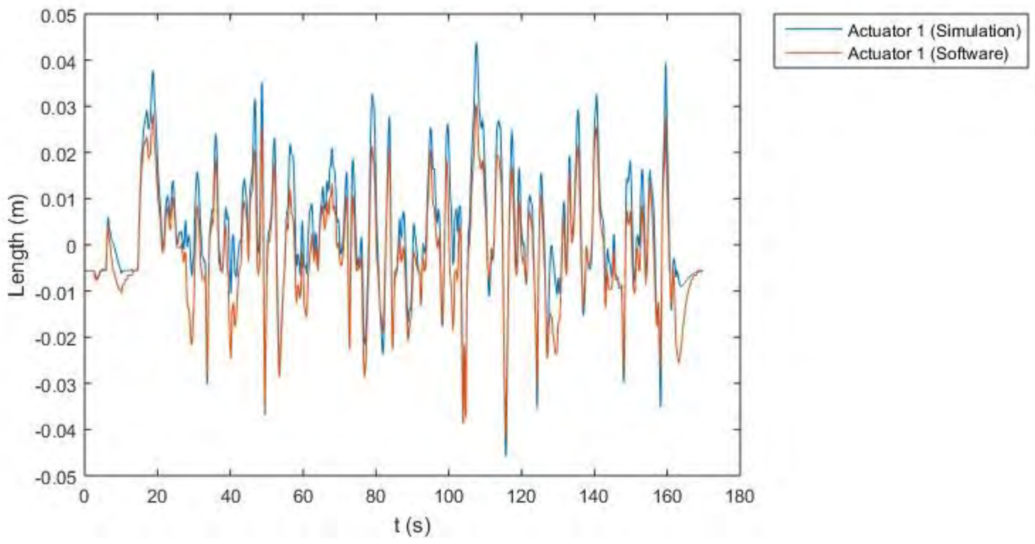


Figure 5-31 Actuator 1 Output Comparison

Figure 5-32 illustrates the actuator 2 stroke length output for the software plugin and the Matlab/Simulink simulation. The software plugin output for actuator 2 replicates the output from the Matlab/Simulink simulation well. The maximum error value between the actuator 2 value in the software plugin and in the Matlab/Simulink simulation is 0.0035 m.

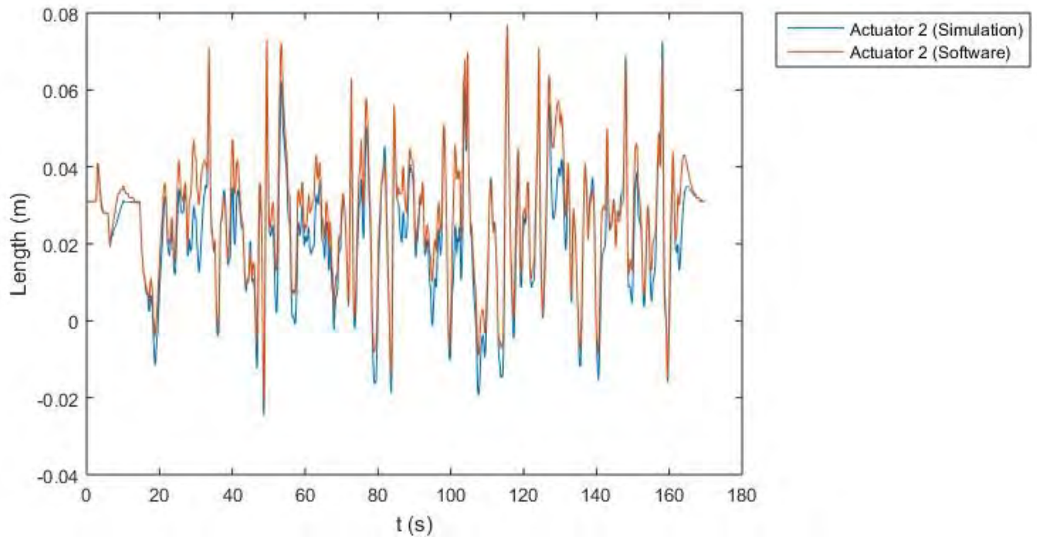


Figure 5-32 Actuator 2 Output Comparison

Figure 5-33 illustrates the actuator 3 stroke length output for the software plugin and the Matlab/Simulink simulation. The software plugin output for actuator 3 replicates the output from the Matlab/Simulink simulation well. The maximum error value between the actuator 3 value in the software plugin and in the Matlab/Simulink simulation is 0.0010 m.

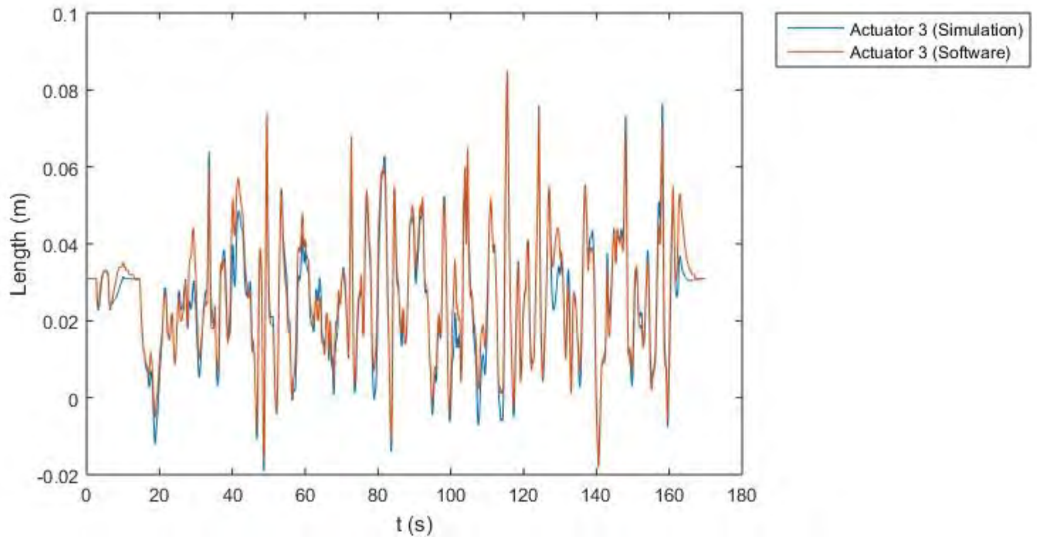


Figure 5-33 Actuator 3 Output Comparison

The results from the software plugin match the output from the structural model really well for all the actuators. The maximum errors for the actuators are acceptable and none of the actuators have significant errors between the software plugin output and the Matlab/Simulink simulation results. These results conclude that the software plugin implementation of data scaling, classical washout algorithm and inverse kinematics calculations are correct. It can also be noted that the washout process works well in returning the platform actuators back to zero positions for the entire duration of the lap. The software plugin is used in the next chapter to transfer actuator stroke length data, via UDP, to the motion control system.

5.3 Chapter Summary

This chapter discussed the software package, X-Sim, which is used to interface with the games physics engine and obtain telemetry data. A software plugin, written in C++ and found in appendix D, is used to perform the data processing and transmit the actuator stroke lengths to the position control system via a LAN connection that uses UDP. The data processing includes input data scaling, the classical washout algorithm and inverse kinematics.

A novel simulation setup developed was used to test the various data processing steps; this testing was done using telemetry data logged from the game. This simulation setup allowed for the input data scaling and filter parameters in the classical washout algorithm to be adjusted. The filter parameters for the various aspects of the classical washout algorithm were selected to ensure the actuator motion constraints, in table 5-1, is not violated. The results from the simulation setup show that the input platform trajectory, into the structural model, is replicated at the output of the structural model. It also confirms that the filter parameters selected ensured the constraints in table 5-1, for the actuator position control system, is respected. Adherence to these constraints ensures safety of the user in the vehicle simulator and safety of the mechanical structure of the vehicle simulator. Simulator fidelity was then evaluated, using the human vestibular system models. The results indicate that the classical washout algorithm does very well in replication of translational motion sensed by the otolith model and the classical washout algorithm does reasonably in replication of the rotational motion sensed by the semi-circular canal model. Evaluating the performance of the 3-DOF system against the traditional 6-DOF system it was found that the 3-DOF system performs better than the 6-DOF system in replication of some motion sensations. This result highlights the benefit of such a platform were low-cost and reasonable performance requirements are needed. The various data processing steps in Matlab/Simulink were implemented in C++ and were used in the software plugin for the X-Sim software package. The results, of the actuator stroke lengths, from the output of the software plugin were compared against the output from the Matlab/Simulink setup. The results show that the software plugin implementation is correct and acceptable to be used to transmit data to the motion control system.

6 Results and Discussion

This chapter discusses and evaluates the performance of the position control system on the vehicle simulator. Details on how the entire system functions and a description on how the various components come together to create a vehicle simulator is presented. Results from the position control system are evaluated and compared to the results in the Matlab/Simulink simulations.

6.1 Vehicle Simulator System Overview

Figure 6-1 illustrates the entire integrated vehicle simulator system, this figure shows the various components and how these combine to achieve the best possible motion simulator fidelity. The game (Dirt 3) provides visual cues to the simulator driver via the PC monitors. Telemetry data is transferred to the software plugin using the X-Sim software package. Position data for each actuator is transferred from the software plugin into the Festo PLC, via a UDP network interface. The position data for each actuator is passed onto the axis controllers which provide actuator motion. Motion cues are transferred to the simulator driver via this actuator motion. Through seamless synchronisation of visual and motion cues a vehicle simulator system with the best possible fidelity is achieved.

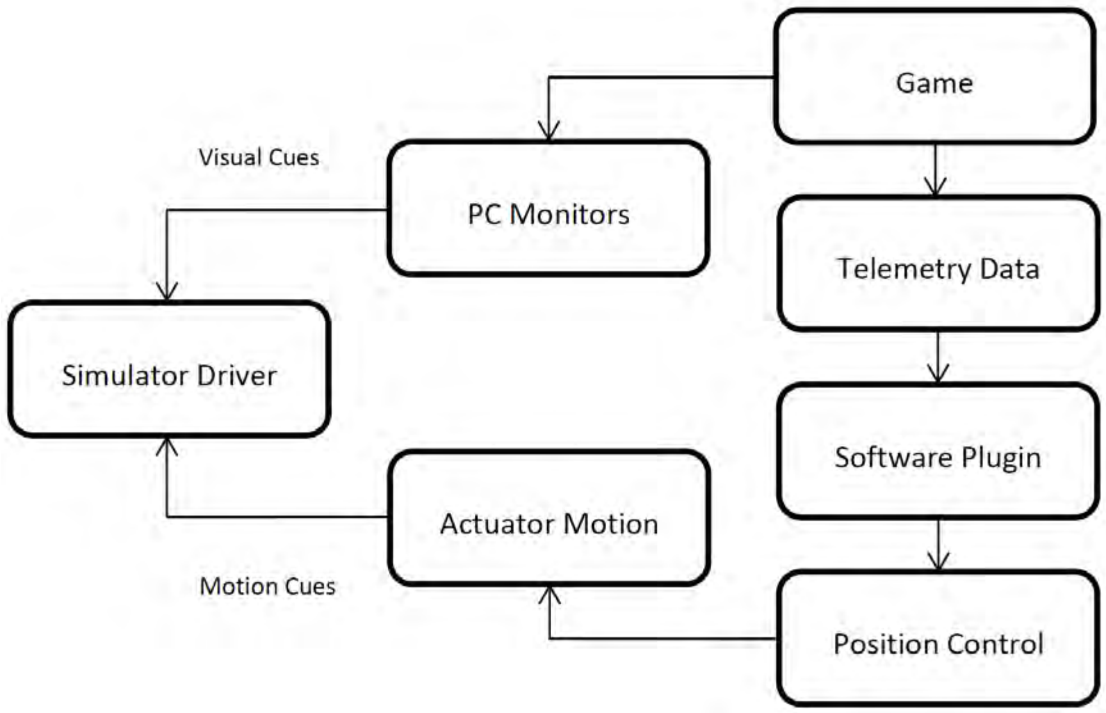


Figure 6-1 Vehicle Simulator System Overview

6.2 Position Control System Limitations

During integration and testing of the entire system it was found that the back actuators in the system were creating excessive coupling forces between each other. These coupling forces affected the position control system performance and led to the control system on each of these actuators becoming unstable. Independently these actuator position control systems were found to track the required input correctly, however when simultaneous motion was required the position control system on each actuator became unstable. Figure 6-2 to figure 6-4 illustrates the position control system performance for each of the 3 actuators. It shows the actual position tracking the reference input, which is the nominal position supplied to the CMAX controller, from the software plugin.

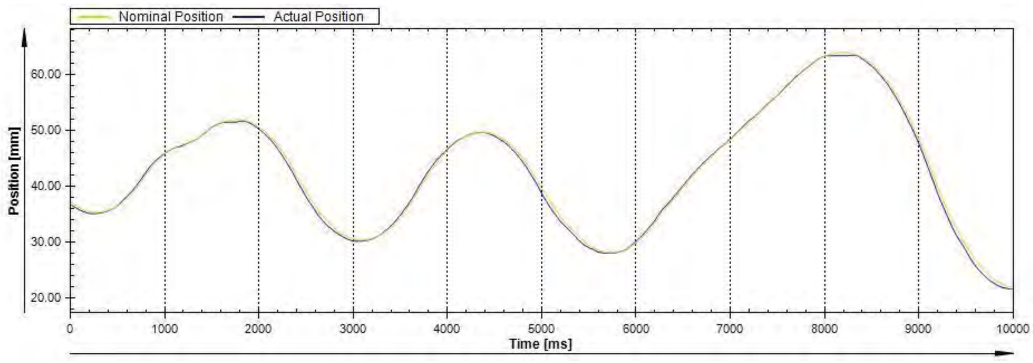


Figure 6-2 Actuator 1 Position Control Tracking with Live Game Data

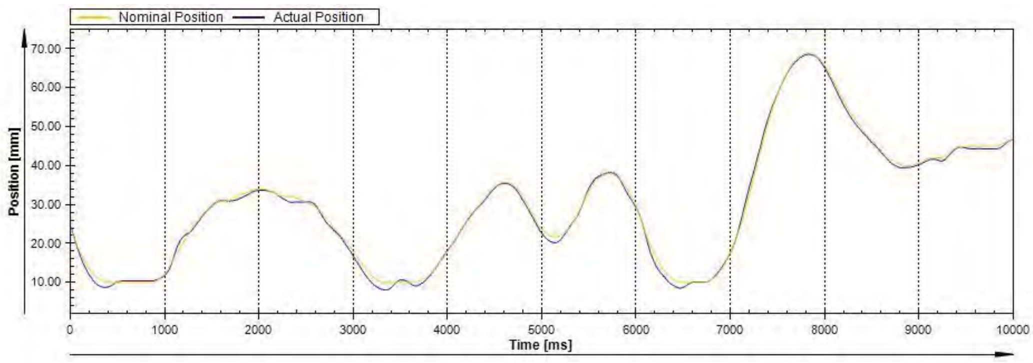


Figure 6-3 Actuator 2 Position Control Tracking with Live Game Data

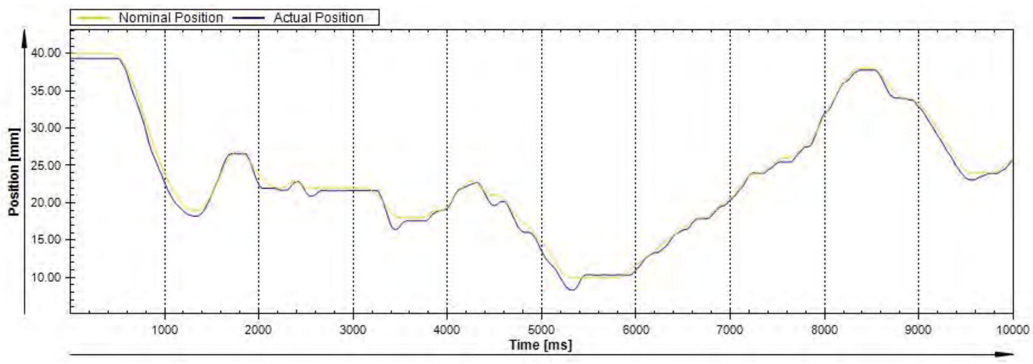


Figure 6-4 Actuator 3 Position Control Tracking with Live Game Data

The excessive coupling on the position control system could not be mitigated with the selected position control system from Festo, despite numerous efforts to adjust the control system parameters on each of the axis controllers. It was decided to modify the system by not implementing position control on the back actuators in the vehicle simulator system. This modification will eliminate the rotational motion about the x-axis (Roll) from the vehicle simulator system completely. Removing the rotational motion about the x-axis (Roll) will eliminate the transient x-axis angular velocity component and the sustained translational motion along the y-axis (which is done through platform tilt). Figure 6-5 illustrates the modified 3-DOF motion platform with the front actuator (Actuator 1) used to provide motion. The new vehicle simulator system is modified into a partial 2-DOF system which is able to produce translational motion along the z-axis (Heave) and rotational motion about the y-axis (Pitch).

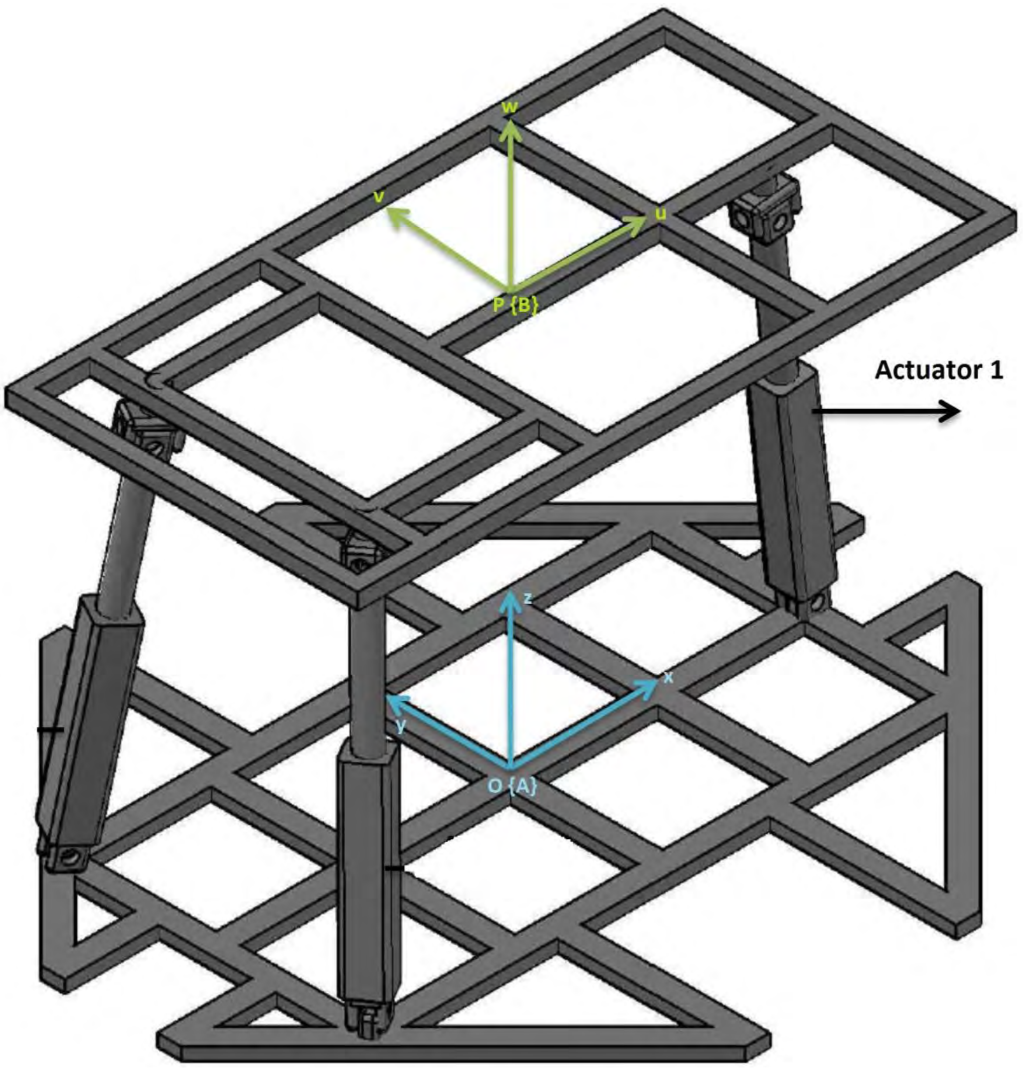


Figure 6-5 Modified Partial 2-DOF Motion Platform with Single Motion Actuator

Simulations in Matlab/Simulink were conducted using the Vehicle Simulator Simulation Setup, illustrated in figure 5-5, to evaluate the modified vehicle simulator systems performance using just the front actuator for motion. The new systems motion was compared to the motion for the 3-DOF motion platform, highlighting the effect on motion cues for the modified system. The partial 2-DOF systems input data scaling factors used were all increased since actuator 1 is only used and this actuator had modest motions initially for the 3-DOF system. Figure 6-6 illustrates the z-axis translational motion (Heave) for the partial 2-DOF system and the initially designed 3-DOF system. It can be seen that with a larger scaling factor, increased from 1 m/s^2 to 2 m/s^2 for the specific force z-axis game input, there is more motion achieved for this motion cue in the partial 2-DOF system. The maximum heave motion achieved in the partial 2-DOF system is 0.0307 m and the maximum heave motion achieved in the 3-DOF system is 0.0083 m.

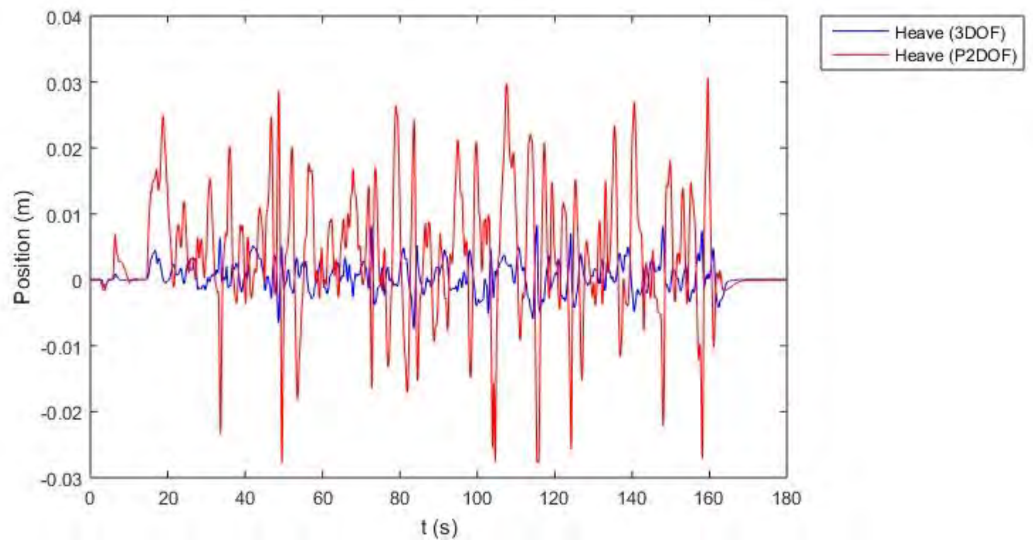


Figure 6-6 Heave Comparison (Modified System)

Figure 6-7 illustrates the rotational motion about the x-axis (Roll) for the partial 2-DOF system and the initially designed 3-DOF system. The partial 2-DOF system has no rotational motion about the x-axis (Roll). A maximum roll value of 4.5313 degrees is achieved for the 3-DOF system.

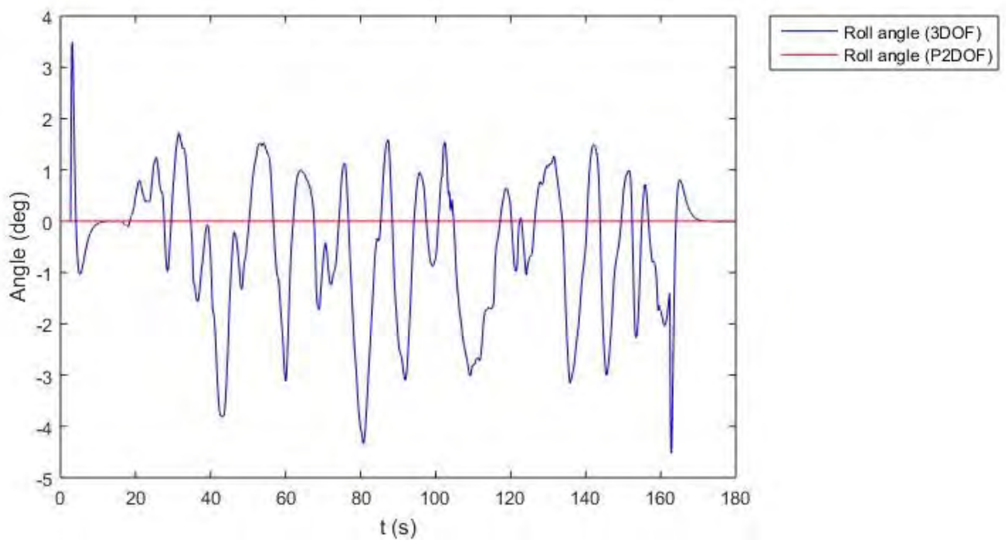


Figure 6-7 Roll Angle Comparison (Modified System)

Figure 6-8 illustrates the rotational motion about the y-axis (Pitch) for the partial 2-DOF system and the 3-DOF system. The scaling factor for the game input data for the pitch component was increased from 10 degrees to 15 degrees. The partial 2-DOF system provides an attenuated signal that matches the 3-DOF motion systems cues. A maximum pitch value of 3.5140 degrees is achieved for the partial 2-DOF system and a maximum pitch value of 5.4235 degrees is achieved in the 3-DOF system.

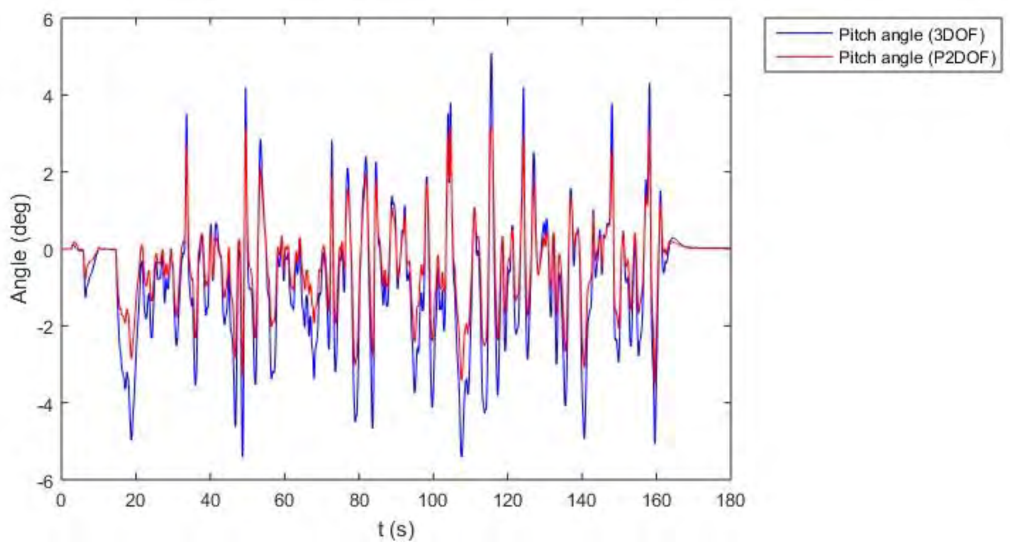


Figure 6-8 Pitch Angle Comparison (Modified System)

Figure 6-9 illustrates the actuator stroke length output for actuator 1 in the vehicle simulator system for the partial 2-DOF system. A maximum platform motion value of 0.6 m was achieved. Simulation results proved that the motion constraints, in table 5-1, were still respected in the modified system, even with the increased input data scale factors.

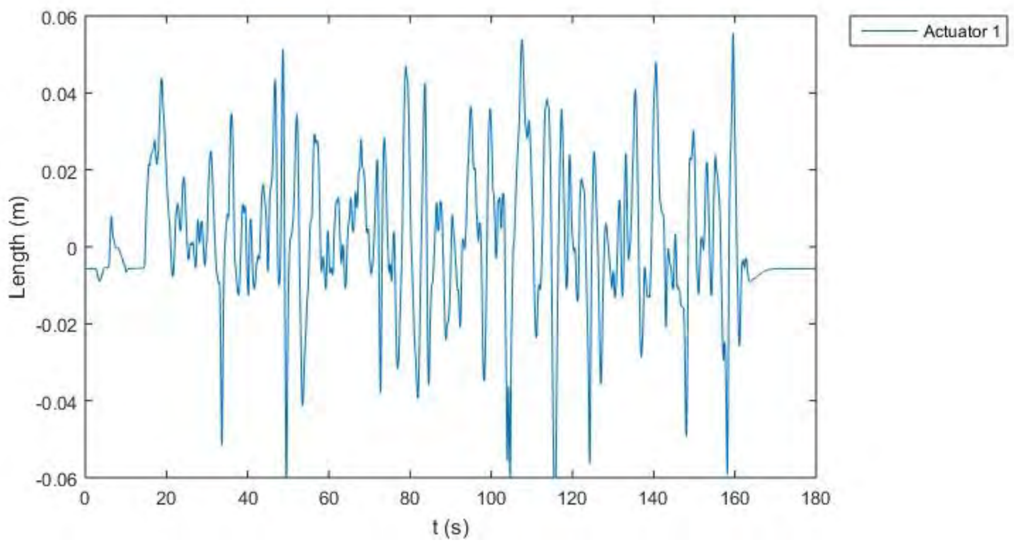


Figure 6-9 Actuator 1 Matlab/Simulink Simulation Output (Modified System)

The modified partial 2-DOF system is able to achieve more significant motion cues for the translation motion along the z-axis (Heave) than the 3-DOF system. No rotational motion about the x-axis (Roll) occurs in the partial 2-DOF system and the 3-DOF system provides significant cues for this component. Rational motion along the y-axis (Pitch) is replicated well in the partial 2-DOF system, this systems replicates attenuated cues that are felt in the 3-DOF system. The partial 2-DOF systems actuator motion respects the actuator motion constraints in table 5-1.

Figure 6-10 shows the sensed specific force, along the x-axis, by the otolith model for both 3-DOF and the modified partial 2-DOF system. These signals are compared to the sensed specific force, along the x-axis, in the actual vehicle. Both systems perform reasonably well with a fair amount of false motion cues. The partial 2-DOF system outperforms the 3-DOF system by having far less false motion cues. It provides motion sensations closer to that felt within the actual vehicle.

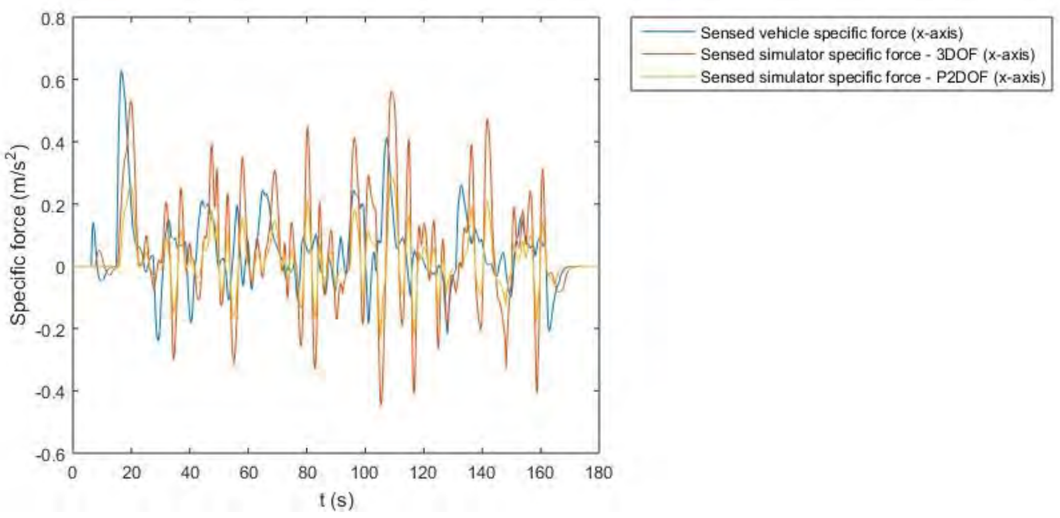


Figure 6-10 Specific Force Comparison – Modified System (x-axis)

Figure 6-11 shows the sensed specific force, along the z-axis, by the otolith for both 3-DOF and the modified partial 2-DOF system. These signals are compared to the sensed specific force, along the z-axis, in the actual vehicle. Both systems perform well in replication of motion sensations felt along the z-axis.

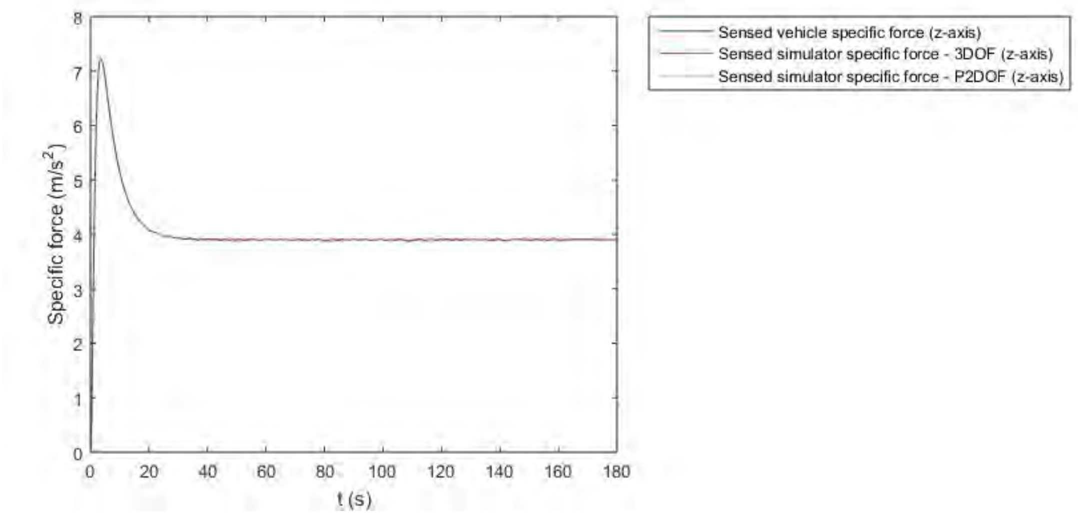


Figure 6-11 Specific Force Comparison – Modified System (z-axis)

Figure 6-12 shows the sensed angular velocity, along the y-axis, by the semi-circular canal model for both 3-DOF and the modified partial 2-DOF system. These signals are compared to the sensed angular velocity, along the y-axis, in the actual vehicle. The performance of motion replication for this component is modest in both systems; the 3-DOF system outperforms the modified partial 2-DOF system for this motion cue. Figure 6-13 illustrates the motion cues experienced, for sensed angular velocity along the y-axis, in both systems for a small time period. It shows that both components replicate an attenuated signal, the 3-DOF system performs better than the partial 2-DOF system in motion replication. There are some motion cues which are missed completely in both systems.

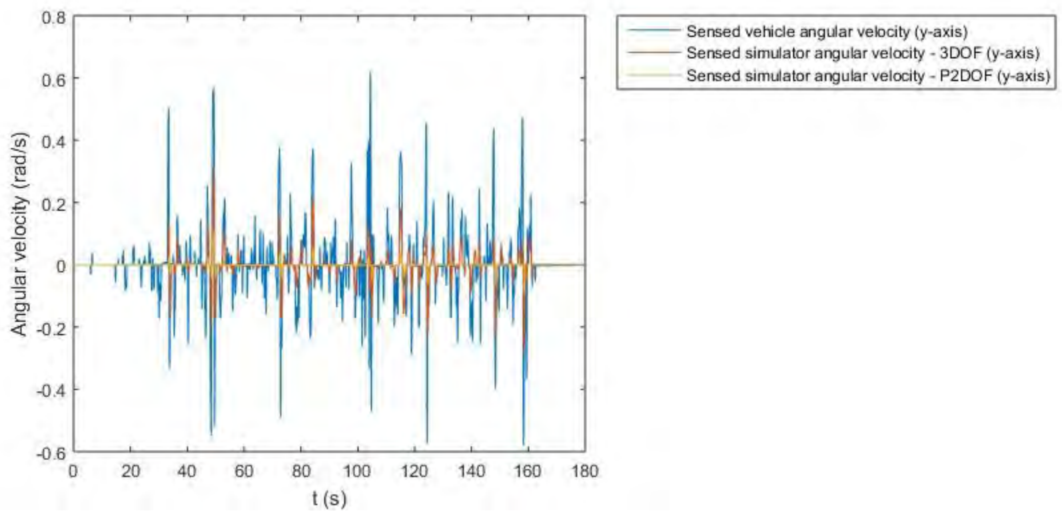


Figure 6-12 Angular Velocity Comparison – Modified System (y-axis)

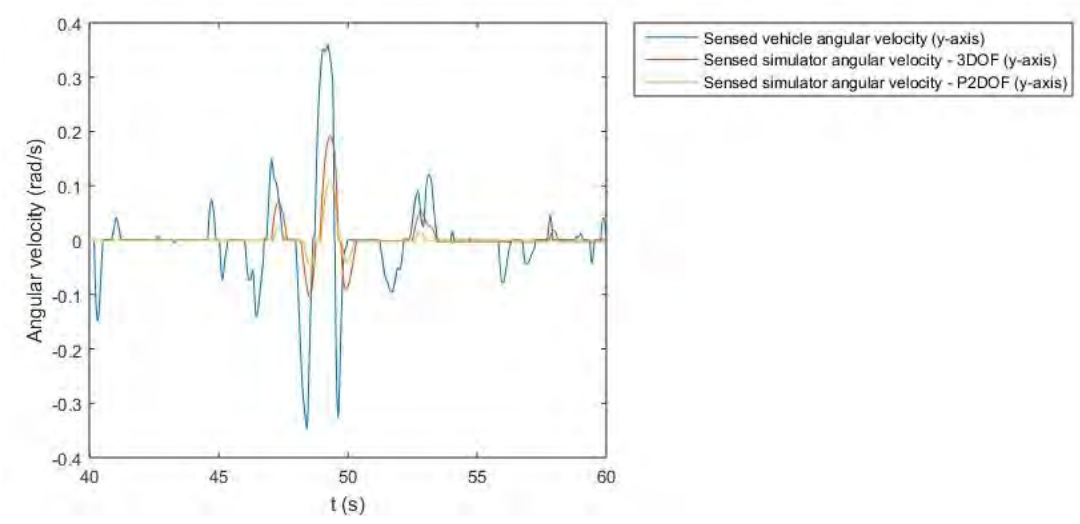


Figure 6-13 Angular Velocity Sample Comparison – Modified System (y-axis)

The modified system which is a partial 2-DOF system is able to outperform the initially designed 3-DOF motion system in replication of translation motion along the x-axis (Surge). Both systems are able to replicate the translation motion along the z-axis (Heave). Rotational motion along the y-axis (Pitch) is replicated poorly in both systems, with the 3-DOF system performing better. This modified partial 2-DOF system has applications were sustained translation motion along the x-axis is required. Inability to replicate sustained longitudinal accelerations results in poor simulation of maneuverers such as emergency braking (Arioui et al., 2009). Applications for the partial 2-DOF system include human factor studies in scenarios which do not have much transient accelerations e.g. highway studies. In terms of testing car prototypes this system could be used to test adaptive cruise control systems which generally have smooth sustained accelerations. For driving training this platform can be adopted as a first contact for new drivers to provide experience, in heavy machinery systems which do not undergo severe accelerations, in general leisure environments and fuel efficiency training since proper shifting techniques do not produce too many transient acceleration signals.

During testing of the position control system it was found that stopping at the mechanical endpoints was abrupt. This method of stopping could eventually cause damage to the vehicle simulator and causes discomfort for the user in the vehicle simulator. Upper and lower software end positions were set to ensure the actuator does not stop abruptly at the mechanical end positions. Illustrated in figure 6-14 is the lower software end position (**LP**) and the upper software end position (**UP**), which was set at 10 mm and 130 mm respectively, these values were based on the maximum and minimum position values of actuator 1 from figure 6-

9. These values give a working stroke length of 120 mm for the actuator. This cylinder stroke length was limited, by the software end positions; based on the workspace usage from the Matlab/Simulink simulation results. In addition to these limits the actuator needed to have a reference zero point at which to start from and return to when motion is complete. Figure 6-14 illustrates the actuator reference zero point and software position limits selected. The reference zero point (**ZP**) for the actuator was set at 70 mm in height, resulting in maximum motion in both the forward and backward strokes.

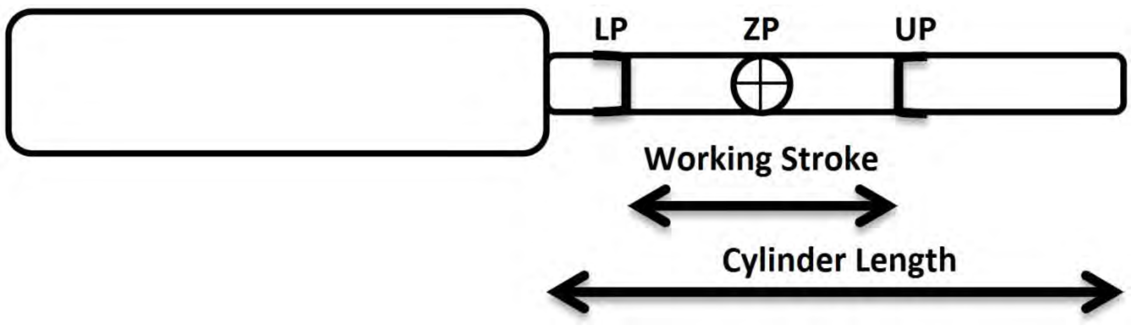


Figure 6-14 Actuator Reference Point and Position Limits

6.3 Position Control System Results

Position control tracking performance for the modified partial 2-DOF system is logged using the Festo Control Software. Live data from the game is sent to the position control hardware and the tracking performance is evaluated. Ideally the actual position should track the nominal position (this is the value sent to the controller from the written software plugin) value perfectly.

Figure 6-15 illustrates the tracking performance for the modified partial 2-DOF system using live game data. The position control system hardware tracks the nominal position well with minimum deviation from the setpoint value and almost perfect response time.

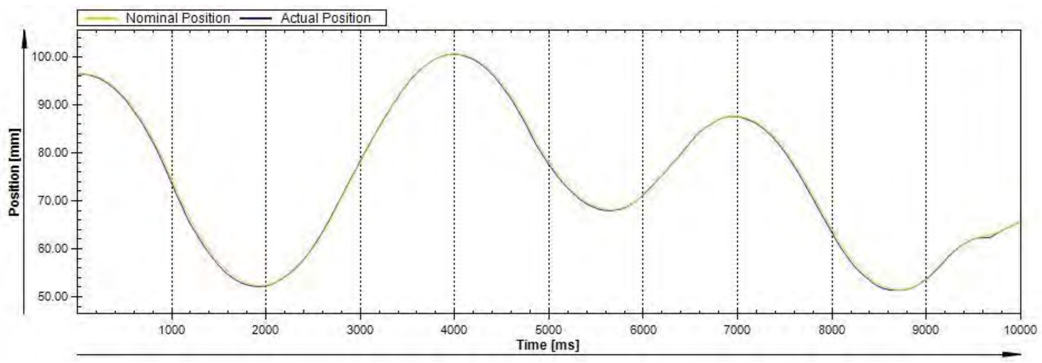


Figure 6-15 Actuator 1 Position Control Tracking with Live Game Data (Modified System)

Figure 6-16 illustrates the simulation results in Matlab/Simulink for the modified partial 2-DOF system. Comparing this result to the actual tracking performance of the position control hardware in figure 6-15 it can be noted that the position control hardware response performs well. It can be seen that there is limitation in fully reproducing some transient components for this system. The fidelity while driving in the actual vehicle is realistic. Motion and visual cues tie in to create an immersive driving experience in the current vehicle simulator system. The overall performance is a vehicle system with good fidelity and the modified partial 2-DOF system is a low-cost alternative to the traditional 6-DOF system.

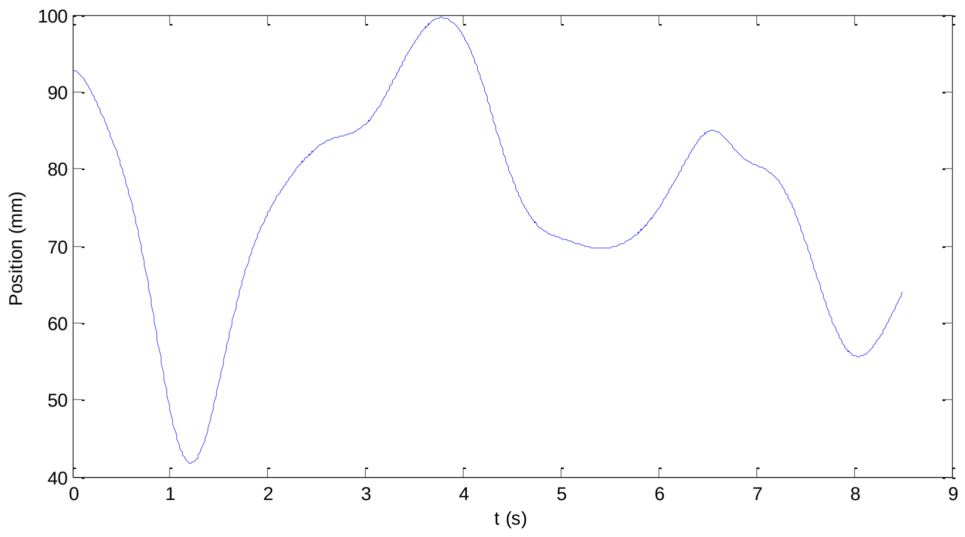


Figure 6-16 Actuator 1 Matlab/Simulink Simulation Output with Live Game Data

6.4 Chapter Summary

The vehicle simulator with the various components was presented in this chapter. It was shown how the motion and visual cues tie in together to achieve the best possible vehicle simulator fidelity.

Position control system testing found limitation in the selected Festo position control system. It was decided to modify the system by eliminating rotational motion about the x-axis (Roll) completely from the system. The modified system is a partial 2-DOF system and this system was shown to outperform the 3-DOF system, initially chosen, in replication of translational motion along the x-axis (Surge).

Testing and comparing the position control hardware tracking performance was then performed. It was shown that the position control system does extremely well to track the nominal position value. Comparing the position control system tracking performance to the Matlab/Simulink system shows that replication of actuator motion is replicated well. The position control hardware does come short in replicating of some transient motions. The overall system provides motion and visual cues that tie in together to create an immersive driving experience. This modified partial 2-DOF system is seen as a low-cost alternative in scenarios with cost constraints and reasonable performance requirements.

7 Conclusions and Future Work

The purpose of this research was to research, design and implement motion cues for the 3-DOF motion platform. This motion platform is able to provide translational motion along the z-axis (Heave), rotational motion about the x-axis (Roll) and rotational motion about the y-axis (Pitch). The research aimed to create the best possible fidelity in the vehicle simulator system by creating realistic motion cues that work in cohesion with visual cues.

The motion platform for this simulator is a 3-DOF platform as compared to the traditional 6-DOF Stewart platform. It is known that the 6-DOF motion platform is used extensively in vehicle simulators around the world. The 6-DOF motion platform has complex forward kinematics and excessive manufacturing costs (Tsai et al, 1996). Lower DOF motion platforms are a compromise between motion replication quality and cost (Arioui et al., 2009).

A kinematic analysis was performed on the 3-DOF system to facilitate in replication of vehicle motion cues. This kinematic analysis presented a closed form solution to the inverse kinematics and a numerical approximation for the forward kinematics. The derived inverse kinematics equations were validated in the Matlab/Simulink environment, using the SimMechanics toolbox. A similar simulation setup and testing was implemented for the traditional 6-DOF system.

The classical washout algorithm was the motion cueing algorithm selected to be used on the 3-DOF platform in the vehicle simulator. This particular motion cueing algorithm is simple to implement and has lower performance requirements than the other motion cueing algorithms. A disadvantage of the classical washout algorithm is the false motion cues due to the fixed filter parameters used in the design process.

A novel simulation system was developed in the Matlab/Simulink environment, illustrated in figure 5-5, to aid in the design of a vehicle simulator. The simulation system developed comprises of all the aspects involved in the motion cueing process which includes input vehicle data scaling, implementing of the classical washout algorithm and performing inverse kinematics.

The first research contribution showed how the developed simulation system was used to aid in ensuring the actuator motion constraints, in table 5-1, are respected. The input data scaling and filter parameters were adjusted to ensure that these actuator motion constraints are respected. These constraints ensure safe operation for the driver in the vehicle simulator and ensure safety of the mechanical structure of the vehicle simulator.

It can be noted from the acceleration of the actuator legs, illustrated in figure 5-16, that the actuators reach a maximum acceleration for very short periods of time. This platform acceleration is due to the fixed filter parameters of the classical washout algorithm which is designed for the worst case motion. Therefore platform workspace usage is not optimised. In future to improve the workspace usage and motion sensations in the vehicle simulator system alternative washout algorithms will be implemented. These alternatives include the adaptive washout algorithm (Ariel and Sivan, 1984), optimal washout algorithm (Sivan et al., 1982) and model predictive control techniques (Baseggio et al., 2011).

The next research contribution evaluated the performance of the 3-DOF system against the traditional 6-DOF system, in the Matlab/Simulink environment, using the human vestibular system models. Specific force comparison along the x-axis shows that the 3-DOF system performs better at motion replication and has smaller motion sensations for false motion cues than the 6-DOF system. Specific force comparison along the y-axis showed the 6-DOF system performing better at motion replication for this component. Both systems do well in replication of specific force along the z-axis. In terms of the angular velocity components, both systems replicate attenuated motion cues for the angular velocity along the x-axis with some small false motion cues. Angular velocity along the y-axis is replicated modestly in both systems. In terms of angular velocity along the z-axis the 3-DOF system has no motion sensation for this component and the 6-DOF system does reasonably well in replication of the sparse motion sensations felt. These results highlight the benefits of the 3-DOF system in certain applications. These systems are applicable to scenarios with cost constraints and which have reasonable performance requirements. The reasons for choosing a 3-DOF system were the lower manufacturing costs involved and relatively simple manufacturing of such a system.

Testing of the position control system on the 3-DOF motion platform highlighted instability in the back actuators. This instability led to the elimination of motion in the back actuators and this change removed the rotational motion along the x-axis (Roll). The final system developed was a partial 2-DOF system; this system is capable of partial restitution of translational motion along the z-axis (Heave) and rotational motion along the y-axis (Pitch). This system was evaluated against the initially designed 3-DOF system. Results from this evaluation showed that the partial 2-DOF system did better in replication of translational motion along the x-axis (Surge).

The position control system tracking results for the partial 2-DOF system showed excellent tracking of the nominal position value (this is the value sent to the controller from the written software plugin). Comparing the position control system tracking performance to the Matlab/Simulink system shows that the replication of the actuator motion is replicated well. The position control hardware does come short in replication of some transient motions. Final testing of the system shows that it provides visual and motion cues that tie in together to create an immersive driving experience.

The following research objectives were met:

- The current mechanical framework was investigated and understood.
- Kinematic analysis, simulation and testing of the 3-DOF motion platform and traditional 6-DOF motion platform were performed.
- The classical washout algorithm for use in both the 3-DOF motion platform and traditional 6-DOF motion platform was investigated, designed and implemented.
- A novel simulation system, developed in Matlab/Simulink, to aid in the vehicle simulator design was developed. This system was successfully used to adjust various parameters in the motion cueing process to ensure actuator motion constraints are respected.
- The performance of the designed 3-DOF motion platform was evaluated against the traditional 6-DOF motion platform using human vestibular system models in Matlab/Simulink.
- A software plugin, developed in C++, was used to interface between the physics engine of a game and the position control system on the 3-DOF motion platform. The software plugin implemented the various motion cueing aspects in the C++ language.
- The Festo position control system hardware was configured and tested.
- PLC software to perform actuator position control was developed and tested.
- The position control system performance was evaluated. This evaluation highlighted the instability of the selected position control system. This instability led to a modification in which rotational motion along the x-axis (Roll) was removed from the vehicle simulator. The vehicle simulator was modified into a partial 2-DOF system. This modified systems position control performance in replication of motion cues was

subsequently evaluated against the results from the developed Matlab/Simulink simulation system.

- The entire vehicle simulator system was integrated and tested. A vehicle simulator with good fidelity was achieved. The initially designed 3-DOF vehicle simulator and the final partial 2-DOF vehicle simulators performance highlight the benefits of lower-cost systems. These systems have merit in applications with cost constrains and reasonable performance requirements.

Table 7-1 adapted from (Cheng et al., 2006) shows how the two systems designed, the initial 3-DOF system and final partial 2-DOF system, perform against other simulators. These two systems are not able to produce the same large motions cues as other alternative simulators. These alternative simulators have excessive costs as compared to both systems designed, even without accounting for inflation costs on these alternative systems. The alternative simulators are viable to transport departments and car manufactures which have excessive budgets. The 3-DOF system and final partial 2-DOF system are seen as low-cost alternative in scenarios with cost constraints and reasonable performance requirements.

The partial 2-DOF system is able to replicate transient translational motion along the z-axis, transient rotational motion about the y-axis and sustained translation motion along the x-axis (which is done through platform tilt about the y-axis). Inability to replicate sustained translation motion along the x-axis results in poor simulation of maneuverers such as emergency braking (Arioui et al., 2009).

Applications for the developed system includes human factor studies in scenarios which do not have much transient accelerations e.g. highway studies. In terms of testing car prototypes this system could be used to test adaptive cruise control systems which generally have smooth sustained accelerations. For driving training this platform can be adopted as a first contact for new drivers to provide experience, in heavy machinery systems which do not undergo severe accelerations, in general leisure environments and fuel efficiency training since proper shifting techniques do not produce too many transient acceleration signals.

Future work for this vehicle simulator is evident in the shortcomings of the selected motion control system from Festo. Unfortunately at present a modification to the control system could not be performed due to the closed source nature of the Festo system. New hardware would need to be implemented and an improved control system could be designed to be able to create a fully functional 3-DOF vehicle simulator. Improved control strategies such as model based control or performance based control would be selected. It is more appropriate to select a performance based control strategy for this system due to non-linear dynamics of parallel manipulators. Controllers with good disturbance rejection such as an adaptive control scheme which incorporates disturbance rejection (Qinglong and Wenjie, 2011) and robust control (Wu et al., 2010) are desirable to mitigate the disturbances introduced by the leg coupling forces on the back actuators.

In terms of the provision of visual cues future work would see replacing the current 3x27 inch LED monitors with a VR technology device to render visual cues. This technology will render more immersive visual cues and a better field of view.

Table 7-1 Vehicle Simulator Systems Performance/Cost Comparisons

	Kookmin University Simulator	Vision Light (MotionBase 3D150)	NADS (University of Iowa)	3-DOF System	2-DOF System
Pitch angle (Degrees)	25	18	25	5.4	3.5
Roll angle (Degrees)	20	15	25	4.5	0
Heave motion (m)	-	-	0.6	0.0083	0.0307
Load (Newtons)	1970	1500	-	1200	1200
DOFs	6	6	13	3	2 (Partial)
No. of Actuators	3	3	10	3	1
Acceleration (m/s^2)	6	7	10	2	2
Price ($)	-	$16990	$50 million	$9500	$5700

The dissertation presented the mechatronics integration for a vehicle simulator. The developed system is lower-cost and has application in scenarios with cost constraints and reasonable performance requirements.

References

Ariel, D. and Sivan, R. (1984) ‘False cue reduction in moving flight simulators’, IEEE Transactions on Systems, Man and Cybernetics, 14(4), pp.665-671.

Arioui, H., Hima, S. and Nehaoua, L. (2009) ‘2 DOF low cost platform for driving simulator: design and modeling’, 2009 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Singapore, July 2009, pp.1206-1211.

Arioui, H., Hima, S., Nehaoua, L., Bertin, R.J.V. and Espié, S. (2011) ‘From design to experiments of a 2-DOF vehicle driving simulator’, IEEE Transactions on Vehicular Technology, 60(2), pp.357–368.

Arioui, H., Nehaoua, L. and Amouri, A. (2005) ‘Classic and adaptive washout comparison for a low cost driving simulator’, 13th IEEE Mediterranean Conference on Control and Automation (MED 2005), Limassol, June 2005, pp.586-591.

Baseggio, M., Beghi, A., Bruschetta, M., Maran, F. and Minen, D. (2011) ‘An MPC approach to the design of motion cueing algorithms for driving simulators’, 2011 14th International IEEE Conference on Intelligent Transportation Systems, Washington-DC, October 2011, pp.692-697.

Beghi, A., Bruschetta, M. and Maran, F. (2012) ‘A real time implementation of MPC based motion cueing strategy for driving simulators’, 51st IEEE Conference on Decision and Control, Maui, Hawaii, December 2012, pp.6340-6345.

Bertollini, G.P., Johnston, C.M., Kuiper, J.W., Kukula, J.C., Kulczycka, M.A. and Thomas, W.E. (1994) ‘The general motors driving simulator’, SAE Technical Paper No. 940179.

Bingul, Z. and Karahan, O. (2012) ‘Dynamic modeling and simulation of stewart platform’, Serial and Parallel Robot Manipulators - Kinematics, Dynamics, Control and Optimization, Dr. Serdar Kucuk (Ed.), InTech, Available at: http://www.intechopen.com/books/serial-and-parallel-robot-manipulators-kinematics-dynamics-control-and-optimization/dynamic-modelling-of-stewart-platform (Accessed 22nd May 2016).

Blana, E. (1996) ‘A survey of driving research simulators around the world’, Institute of Transport Studies, University of Leeds, Working Paper 481.

Capustiac, A., Banabic, D., Schramm, D. and Ossendoth, U. (2011) 'Motion cueing: From design until implementation', Proceedings of the Romanian Academy, Series A, 12(3), pp.249-256.

Chen, L.D., Papelis, Y., Waston, G. and Solis, D. (2001) 'NADS at the University of IOWA: A tool for driving safety research', Proceedings of the 1st Human-Centered Transportation Simulation Conference, Iowa, November 2001.

Cheng, M., Irawan, P., Kwak, S, and Putra, P. (2006) 'Motion base driving simulator', Available at: https://deepblue.lib.umich.edu/bitstream/handle/2027.42/49562/me450?sequence=2 (Accessed 31st May 2016)

Clavel, R. (1988) 'Delta, a fast robot with parallel geometry', Proceedings of International Symposium on Industrial Robots, Lausanne, April 1988, pp.91-100.

Colombet, F., Dagdelen, M., Reymond, G., Pere, C., Merienne, F. and Kemeny, A. (2008) 'Motion cueing: What is the impact on the driver's behaviour?', In Driving Simulator Conference 2008 (DSC), Monaco, January 2008, pp.171-182.

Denne, P. (1986) 'Simulators for leisure–A new industry', IEE Conference on Simulation, November 1986.

Drosdol, J. and Panik, F. (1985) 'The Daimler-Benz driving simulator a tool for vehicle development', SAE Technical Paper No. 850334.

Festo (2009) 'System description CMAX axis controller', Available at: https://www.festo.com/net/SupportPortal/Files/379484/CPX-CMAX-SYS_2015-05a_559751g1.pdf (Accessed 22nd May 2016).

Festo (2014a) 'Standard cylinder DNCI, with integrated displacement encoder', Available at: https://www.festo.com/cat/en-gb_gb/data/doc_ENGB/PDF/EN/DNCI_EN.PDF (Accessed 22nd May 2016).

Festo (2014b) 'Sensor interface CASM', Available at: https://www.festo.com/cat/en-gb_gb/data/doc_ENGB/PDF/EN/CASM_EN.PDF (Accessed 22nd May 2016).

Festo (2015a) 'Proportional directional control valves VPWP', Available at: https://www.festo.com/cat/de_de/data/doc_engb/PDF/EN/VPWP_EN.PDF (Accessed 22nd May 2016).

Festo (2015b) 'Axis controller CPX-CMAX', Available at: https://www.festo.com/cat/en-us_us/data/doc_ENUS/PDF/US/CPX-CMAX_ENUS.PDF (Accessed 22nd May 2016).

Festo (2015c) 'Control blocks CPX-CEC', Available at: https://www.festo.com/cat/en-gb_gb/data/doc_ENGB/PDF/EN/CPX-CEC_EN.PDF (Accessed 22nd May 2016).

Freeman, J.S., Watson, G., Papelis, Y.E., Lin, T.C., Tayyab, A., Romano, R.A. and Kuhl, J.G. (1996) 'The Iowa driving simulator: An implementation and application overview', SAE Technical Paper 950174.

Ghosal, A. (no date) 'Kinematics of serial manipulators'.

Gough, V.E. and Whitehall, S.G. (1962) 'Universal tyre test machine', Proceedings of 9th International Congress FISITA, pp.117-137.

He, J.F., Jiang, H.Z., Cong, D.C., Ye, Z.M. and Han, J.W. (2007) 'A survey on control of parallel manipulator', Key Engineering Materials, 339, pp.307-313.

Honegger, M., Brega, R. and Schweiter, G. (2000) 'Application of a nonlinear adaptive controller to a 6 dof parallel manipulator', Proceedings of the 2000 IEEE International Conference on Robotics & Automation, San Francisco-CA, April 2000, pp.1930-1935.

Kemeny, A. and Panerai, F. (2003) 'Evaluating perception in driving simulation experiments', Trends in Cognitive Sciences, 7(1), pp.31-37.

Kihl, M. and Wolf, P.J. (2007) 'Using driving simulators to train snowplow operators: The Arizona experience', Proceedings of the 2007 Mid-Continent Transportation Research Symposium, Iowa, August 2007.

Kim, J.H., Lee, W.S., Park, I.K., Park, K.K. and Cho, J.H. (1997) 'A design and characteristic analysis of the motion base for vehicle driving simulator', IEEE International Workshop on Robot and Human Communication, Sendai, September 1997, pp.290-294.

Lee, W.S., Jung, S.K., Cho, J.H., Shin, S.I. and Bag, J.C. (2001) 'A sudden acceleration study using the Kookmin University driving simulator', Proceedings of the 1st Human-Centered Transportation Simulation Conference, Iowa, November 2001.

Lee, W.S., Kim, J.H. and Cho, J.H. (1998) ‘A driving simulator as a virtual reality tool’, Proceedings of the 1998 IEEE International Conference on Robotics & Automation, Leuven, May 1998, pp.71-76.

Lee, K.M. and Shah, D.K. (1988) ‘Kinematic analysis of a three-degrees-of-freedom in-parallel actuated manipulator’, IEEE Journal on Robotics and Automation, 4(3), pp.354–360.

Mohammadipanah, H. and Zohoor, H. (2009) ‘Design and analysis of a novel 8-DOF hybrid manipulator’, International Journal of Mechanical, Aerospace, Industrial, Mechatronic and Manufacturing Engineering, 3(10), pp.1158-1164.

Ng, C.C., Ong, S.K. and Nee, A.Y.C. (2006) ‘Design and development of 3–DOF modular micro parallel kinematic manipulator’, The International Journal of Advanced Manufacturing Technology, 31(1-2), pp.188–200.

Nguyen, C.C., Antrazi, S.C., Zhou, Z.-L. and Campbell, C.E. (1991) ‘Analysis and implementation of a 6 DOF Stewart platform-based robotic wrist’, Computers & Electrical Engineering, 17(3), pp.191–203.

Patel, Y.D. and George, P.M. (2012) ‘Parallel manipulators applications—a survey’, Modern Mechanical Engineering, 2(3), pp.57-64.

Pollard, W.L.V. (1942) ‘Position controlling apparatus’ US Patent No. 2286571.

Qinglong, J. and Wenjie, C. (2011) ‘Adaptive control of 6-DOF parallel manipulator’, Proceedings of the 30th Chinese Control Conference, Yantai, July 2011, pp.2440-2445.

Reid, L.D. and Nahon, M.A. (1985) ‘Flight simulation motion-base drive algorithms: Part 1, Developing and testing equations’, UTIAS Report No. 296, December 1985.

Reid, L.D. and Nahon, M.A. (1986) ‘Flight simulation motion-base drive algorithms: Part 2, Selecting the system parameters’, UTIAS Report No. 307, May 1986.

Reymond, G., Heidet, A., Canry, M. and Kemeny, A. (2000) ‘Validation of Renault’s dynamic simulator for adaptive cruise control experiments’, In Proceedings of the Driving Simulator Conference (DSC00), Paris, September 2000, pp.181-191.

Reymond, G. and Kemeny, A. (2000) ‘Motion cueing in the Renault driving simulator’, Vehicle System Dynamics: International Journal of Vehicle Mechanics and Mobility, 34(4), pp.249-259.

Schmidt, S.F. and Conrad, B. (1970) ‘Motion drive signals for piloted flight simulators’, in Contractor Report NASA CR-1601, Washington, May 1970.

Shiong, C.Y., Jalil, M.K.A. and Hussein, M. (2009) ‘Motion visualisation and control of a driving simulator motion platform’, Proceedings of the 6th International Symposium on Mechatronics and its Applications (ISMA09), Sharjah, March 2009, pp.1-5.

Sivan, R., Ish-Shalom, J. and Huang, J.K. (1982) ‘An optimal control approach to the design of moving flight simulators’, IEEE Transactions on Systems, Man and Cybernetics, 12(6), pp.818-827.

Smit, P.E. (2010) ‘Development of a 3-DOF motion simulation platform’, Doctoral dissertation, Stellenbosch University.

Stall, D. and Bourne, S. (1996) ‘The national advanced driving simulator: potential applications to ITS and AHS research’, In Proceeding of the 6th Annual meeting of the Intelligent Transportation Society, April 1996, pp.700-710.

Sternheim, F. (1987) ‘Computation of the direct and inverse geometric models of the delta 4 parallel robot’, Robotersysteme, 3(4), pp.199-203.

Stewart, D. (1965) ‘A platform with six degrees of freedom’, Proceedings of the Institution of Mechanical Engineers, 180(1), pp.371-386.

Strayer, D.L. and Drews, F.A. (2003) ‘Simulator training improves driver efficiency: Transfer from the simulator to the real world’, Proceedings of the Second International Driving Symposium on Human Factors in Driver Assessment, Training and Vehicle Design, Utah, July 2003, pp.190-193.

Strayer, D.L., Drews, F.A. and Burns, S. (2004) ‘The development and evaluation of a high-fidelity simulator training program for snowplow operators’, Proceedings of the Third International Driving Symposium on Human Factors in Driver Assessment, Training and Vehicle Design, pp.464-470.

Taikui, W. and Jianmin, D. (2011) ‘Research on motion control system of driving simulator and control algorithm’, Third International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC), Zhejiang, August 2011, pp.61-65.

The MathWorks, Inc. (2007) ‘SimMechanics 2 Users Guide’, Available at: https://mecanismos2mm7.files.wordpress.com/2011/09/tutorial-sim-mechanics.pdf (Accessed 22nd May 2016).

Toyota (2007) ‘Toyota Develops World – Class Driving Simulator’, Available at: http://www.toyota.co.jp/en/news/07/1126_1.html (Accessed 27th September 2015).

Tsai, L.W., Walsh, G.C. and Stamper, R.E. (1996) ‘Kinematics of a novel three DOF translational platform’, Proceedings of the 1996 IEEE International Conference on Robotics and Automation, Minneapolis- Minnesota, April 1996, pp.3446-3451.

Wu, B., Wang, Z., Wang, F., Liu, X. and Wu, S. (2010) ‘QFT robust controller design and experiment research of electro-hydraulic driving parallel robot’, 2010 International Conference on Computer Application and System Modeling (ICCASM), Taiyuan, October 2010, pp.V4-653-V4-657.

X-Sim (no date), ’Manual’, Available at: http://www.x-sim.de/manual/index.html (Accessed 22nd May 2016).

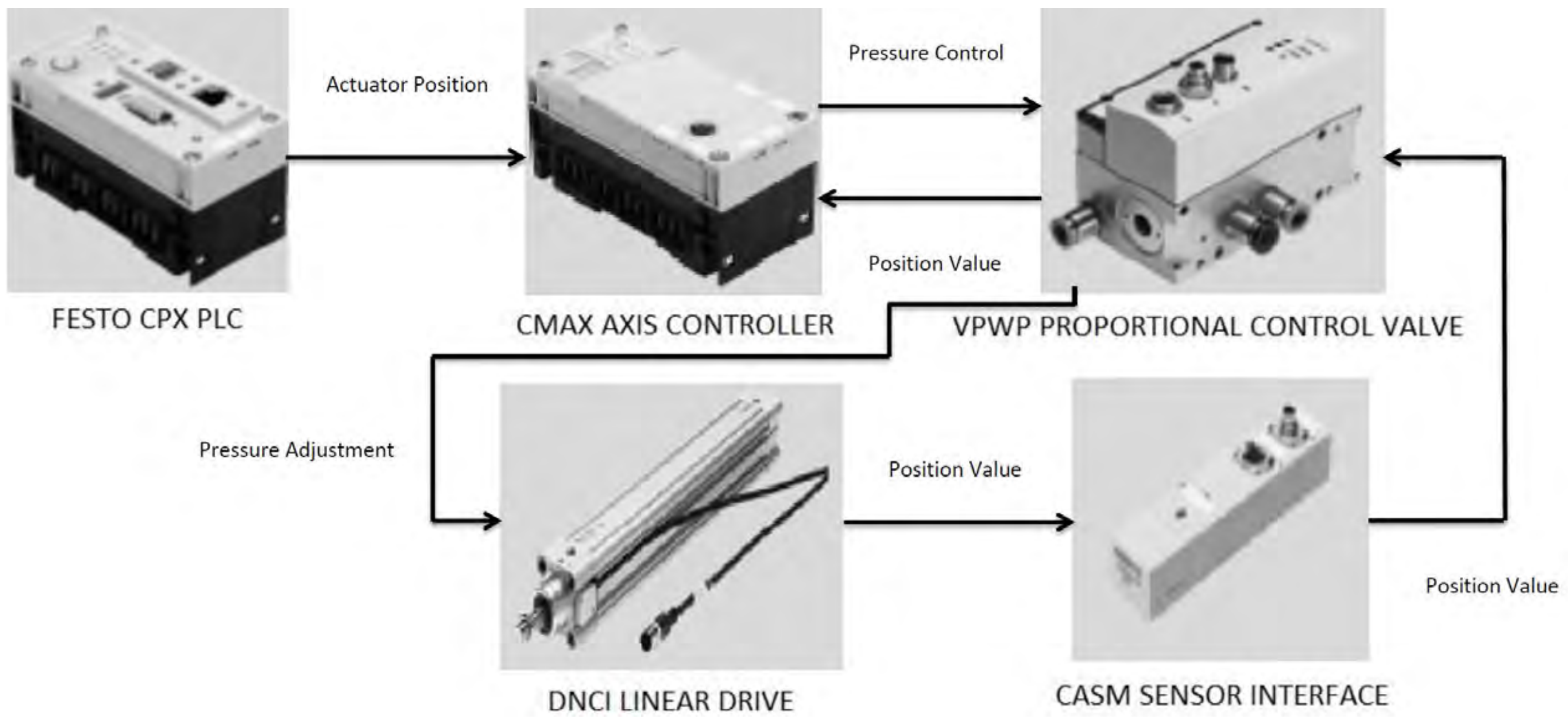
Yang, C., He, J., Jiang, H. and Han, J. (2008) ‘Modeling and simulation of 6-DOF parallel manipulator based on PID control with gravity compensation in Simulink/ADAMS’, 2008 International Workshop on Modelling, Simulation and Optimization, Hong Kong, December 2008, pp.391-395.

Yu, L., Zhang, L., Zhang, N., Yang, S. and Wang, D. (2010) ‘Kinematics simulation and analysis of 3-RPS parallel robot on SimMechanics’, Proceedings of the 2010 IEEE International Conference on Information and Automation (ICIA), Harbin, June 2010, pp.2363-2367.

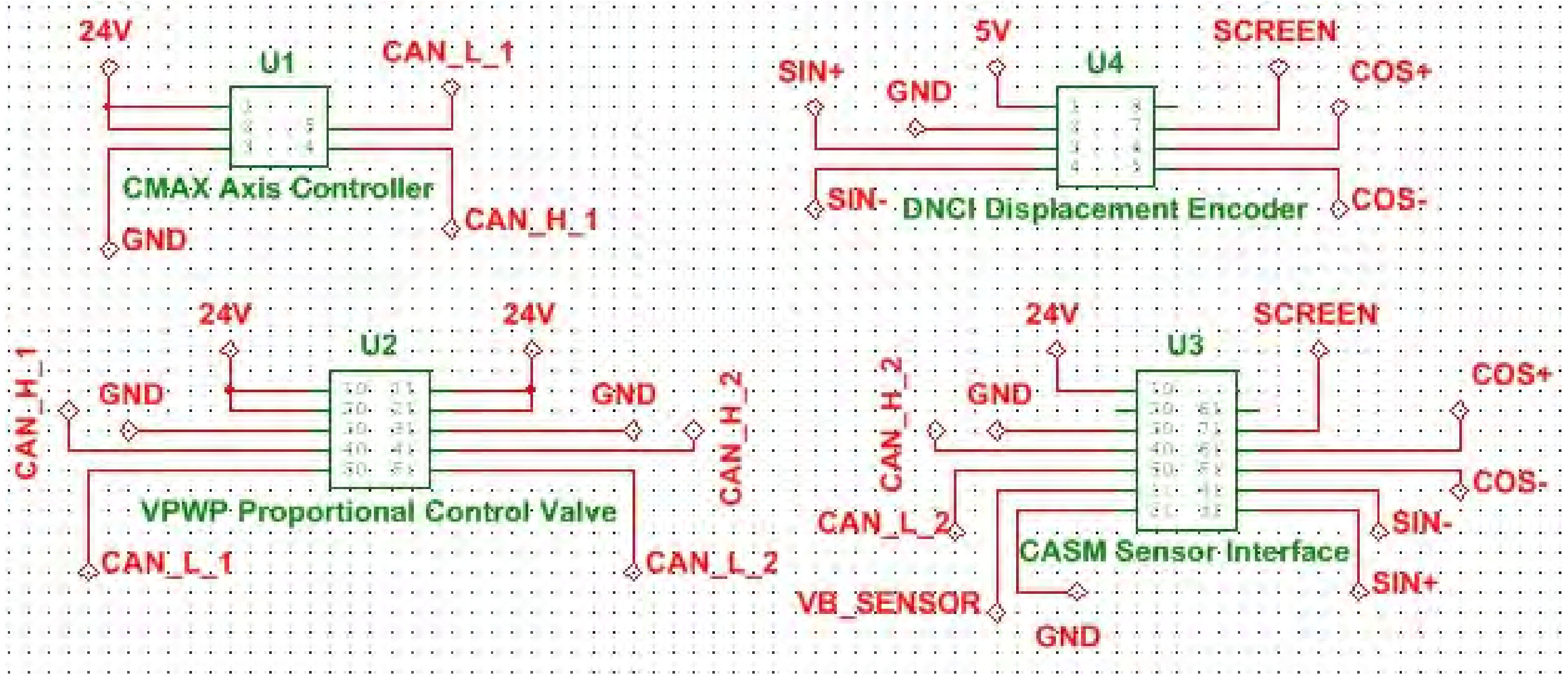
Zacharias, G.L. (1978) ‘Motion cue models for pilot-vehicle analysis’, BOLT BERANEK AND NEWMAN INC CAMBRIDGE MA CONTROL SYSTEMS DEPT, AMRL-TR-78-2, May 1978

Appendices

Appendix A – Hardware Architecture



Appendix B – Electrical Layout



Appendix C – PLC Software

Codeblock P0

```
STEP 0
 THEN  SET             P1              " AXIS 1 SETUP

STEP 1
 IF                    NOP
 THEN  SET             P3              " UDP HANDLER

STEP 2
 IF                    NOP
 THEN  SET             P4

STEP 3
 IF                    NOP
 THEN  JMP TO 1
```

Codeblock P1

```
STEP 0
 THEN  SET             O128.0          " ENABLE DRIVE AXIS V1

STEP 1
 IF                    I128.0          " DRIVE ENABLED
 THEN  SET             O128.0          " ENABLE DRIVE
       SET             O128.1          " STOP
       SET             O128.2          " BRAKE
       RESET           O128.3          " RESET (RISING EDGE ACK ERROR)
       RESET           O128.4
       RESET           O128.5          " LOCK (FCT ENABLED = V0)
       SET             O128.6          " OPM1
       RESET           O128.7          " OPM2
       SET             O129.6          " FAST STOP
       RESET           O129.0          " ABS
       RESET           O129.1          " COM1
       RESET           O129.2          " COM2
 IF                    I128.0          " DRIVE ENABLED
       AND             I128.1          " OPEN
       AND      N      I128.3          " FAULT
       AND             I128.4          " 24V
       AND             I128.6          " OPM1
       AND      N      I128.7          " OPM2
 THEN  SET             O128.10         " HOME

STEP 2
 IF                    I128.15
 THEN                  NOP

STEP 3
 IF                    I128.10
 THEN                  NOP

STEP 4
 IF                    I128.10
 THEN  LOAD            V1100
         TO            OW130
       SET             O129.10         " sec. setpoint
       SET             O129.13         " sec. setpoint
       SET             O129.14
       SET             O129.3          " CONT
       RESET           O128.10         " HOME
```

```
STEP 5
 IF                     I128.10
 THEN  SET              O128.9          " START
       SET              T51             " DELAY START
         WITH           0.1s

STEP 6
 IF                     NOP
 THEN  JMP TO 6
```

Codeblock P3

```
STEP 0
 THEN  CFM 2                            " Install UDP handler
         WITH           V1024           " Local port number (>=1024)
         WITH           V4              " Number of first flagword for data
       LOAD             FU32            " 0 if successful, otherwise error
         TO             R103
 IF                     NOP
 THEN  LOAD             FW15
         SWAP
         TO             R103
       LOAD             FW18
         SWAP
         TO             R104
       LOAD             FW21
         SWAP
         TO             R105

STEP 1
 IF                     NOP
 THEN  LOAD       (     R103
       AND              V255      )
         TO             R106
 IF                     NOP
 THEN  LOAD       (     R104
       AND              V255      )
         TO             R107
 IF                     NOP
 THEN  LOAD       (     R105
       AND              V255      )
         TO             R108
```

Codeblock P4

```
STEP 0
 IF                     I128.9          "ACK
 THEN                   NOP

STEP 1
 IF               ( (   R106
         >=             V10       )
       AND        (     R106
         <=             V130      ) )
 THEN  LOAD             R106
         *              V100
         TO             OW130           "AXIS 1
 OTHRW                  NOP
```

Appendix D – X-Sim Software Plugin Code

```
// Rotation motion globals
double d_rpy_pre_1[3] = { 0, 0, 0 };
double rotation_angles_pre_1[3] = { 0, 0, 0 };
double rpy_1_rate_hf_pre_1[3] = { 0, 0, 0 };
double rpy_2_rate_hf_pre_1[3] = { 0, 0, 0 };
double rpy_3_rate_hf_pre_1[3] = { 0, 0, 0 };
double rpy_rate_pre_1[3] = { 0, 0, 0 };
double rpy_pre_1[3] = { 0, 0, 0 };

// Translational motion globals
double a_XYZ_pre_1[3] = { 0, 0, 0 };
double a_H1_XYZ_pre_1[3] = { 0, 0, 0 };
double a_H2_XYZ_pre_1[3] = { 0, 0, 0 };
double a_H2_XYZ_pre_2[3] = { 0, 0, 0 };
double p_XYZ_pre_1[3] = { 0, 0, 0 };
double p_XYZ_pre_2[3] = { 0, 0, 0 };

// Tilt coordination motion globals
double fl_1_xy_pre_1[2] = { 0, 0 };
double fl_2_xy_pre_1[2] = { 0, 0 };
double f_xy_pre_1[2] = { 0, 0 };

// Euler angles (high plus low)
double euler_angles_pre_1[3] = { 0, 0, 0 };

// Translational motion codeblock

double rotation_matrix[3][3] = { { cos(euler_angles_pre_1[1])*cos(euler_angles_pre_1[2]),
sin(euler_angles_pre_1[0])*sin(euler_angles_pre_1[1])*cos(euler_angles_pre_1[2]) - cos(euler_angles_pre_1[0])*sin(euler_angles_pre_1[2]),
cos(euler_angles_pre_1[0])*sin(euler_angles_pre_1[1])*cos(euler_angles_pre_1[2]) + sin(euler_angles_pre_1[0])*sin(euler_angles_pre_1[2]) }, {
cos(euler_angles_pre_1[1])*sin(euler_angles_pre_1[2]), sin(euler_angles_pre_1[0])*sin(euler_angles_pre_1[1])*sin(euler_angles_pre_1[2]) +
cos(euler_angles_pre_1[0])*cos(euler_angles_pre_1[2]), cos(euler_angles_pre_1[0])*sin(euler_angles_pre_1[1])*sin(euler_angles_pre_1[2]) -
sin(euler_angles_pre_1[0])*cos(euler_angles_pre_1[2]) }, { -sin(euler_angles_pre_1[1]), sin(euler_angles_pre_1[0])*cos(euler_angles_pre_1[1]),
cos(euler_angles_pre_1[0])*cos(euler_angles_pre_1[1]) } };

        double f_x, f_y, f_z = 0;
        double g = -9.81;
        double max_longitudinal_acceleration = 2128990;
        double max_lateral_acceleration = 3454338;
        double max_vertical_acceleration = 6725081;
        double a_xyz[3] = { 0, 0, 0 };
        double a_XYZ[3] = { 0, 0, 0 };
        double a_H1_XYZ[3] = { 0, 0, 0 };
        double a_H2_XYZ[3] = { 0, 0, 0 };
        double p_XYZ[3] = { 0, 0, 0 };
        f_x= 2*((valuearray[sixdofarray::longitudenal_acceleration].inputvalue)/max_longitudinal_acceleration);
        // Positive x(mine) for positive z
        f_y = -2*((valuearray[sixdofarray::lateral_acceleration].inputvalue)/max_lateral_acceleration);
        // Negative y(mine) for positive x
        f_z = -1*((valuearray[sixdofarray::vertical_acceleration].inputvalue)/max_vertical_acceleration);
        // Negative z(mine) for positive y


        if ((f_x - 2.0) > EPSILON) { f_x = 2.0; }
        else if ((f_x + 2.0) < (-EPSILON)) { f_x = -2.0; }
        if ((f_y - 2.0) > EPSILON) { f_y = 2.0; }
        else if ((f_y + 2.0) < (-EPSILON)) { f_y = -2.0; }
        if ((f_z - 1.0) > EPSILON) { f_z = 1.0; }
        else if ((f_z + 1.0) < (-EPSILON)) { f_z = -1.0; }
        // Set f_z offset of 9.81
        f_z = f_z - g;
        a_xyz[0] = f_x - g*sin(euler_angles_pre_1[1]);
        a_xyz[1] = f_y + g*sin(euler_angles_pre_1[0])*cos(euler_angles_pre_1[1]);
        a_xyz[2] = f_z + g*cos(euler_angles_pre_1[0])*cos(euler_angles_pre_1[1]);
        for (int i = 0; i < 3; ++i) {
                for (int j = 0; j < 3; ++j) {   a_XYZ[i] += rotation_matrix[i][j] * a_xyz[j];}}
                for (int i = 0; i < 3; ++i) {   a_H1_XYZ[i] = 0.9695*a_H1_XYZ_pre_1[i] - 0.9847*a_XYZ_pre_1[i] + 0.9847*a_XYZ[i];}
                for (int i = 0; i < 3; ++i) {   a_H2_XYZ[i] = 0.9695*a_H2_XYZ_pre_1[i] - 0.9847*a_H1_XYZ_pre_1[i] + 0.9847*a_H1_XYZ[i];}
                for (int i = 0; i < 3; ++i) {   p_XYZ[i] = -p_XYZ_pre_2[i] + 2*p_XYZ_pre_1[i] + 0.000025*a_H2_XYZ_pre_2[i] +
        0.00005*a_H2_XYZ_pre_1[i] + 0.000025*a_H2_XYZ[i];}
```

```
// Tilt coordination codeblock

double f_xy[2] = {0, 0};
double fl_1_xy[2] = {0, 0};
double fl_2_xy[2] = {0, 0};
double fl_xy_g[2] = {0, 0};
double fl_xy_g_pre_1[2] = {0, 0};
double euler_angles_l[3] = {0, 0, 0};
f_xy[0] = f_x;
f_xy[1] = f_y;

for (int i = 0; i < 2; ++i) { fl_1_xy[i] = 0.99*fl_1_xy_pre_1[i] + 0.004975*f_xy[i] + 0.004975*f_xy_pre_1[i];}
for (int i = 0; i < 2; ++i) { fl_2_xy[i] = 0.99*fl_2_xy_pre_1[i] + 0.004975*fl_1_xy[i] + 0.004975*fl_1_xy_pre_1[i];}
fl_xy_g[0] = fl_2_xy[1] / 9.81;                                    // Tilt rate limiting
fl_xy_g[1] = -fl_2_xy[0] / 9.81;
fl_xy_g_pre_1[0] = fl_2_xy_pre_1[1] / 9.81;
fl_xy_g_pre_1[1] = -fl_2_xy_pre_1[0] / 9.81;
                  if (((fl_xy_g[0] - fl_xy_g_pre_1[0])/0.01)>(0.0524)) { fl_xy_g[0] = fl_xy_g_pre_1[0] + 0.0524*0.01;}
                  else if(((fl_xy_g[0] - fl_xy_g_pre_1[0])/0.01)<(-0.0524)) { fl_xy_g[0] = fl_xy_g_pre_1[0] - 0.0524*0.01;}
                  if (((fl_xy_g[1] - fl_xy_g_pre_1[1])/0.01)>(0.0628)) { fl_xy_g[1] = fl_xy_g_pre_1[1] + 0.0628*0.01;}
                  else if (((fl_xy_g[1] - fl_xy_g_pre_1[1])/0.01)<(-0.0628)){ fl_xy_g[1] = fl_xy_g_pre_1[1] - 0.0628*0.01;}
euler_angles_l[0] = fl_xy_g[0];
euler_angles_l[1] = fl_xy_g[1];

// Rotational motion codeblock

double max_roll_angle = 0.262;
double max_pitch_angle = -0.262; // Negative y for positive x
double max_yaw_angle = -0.262;
double max_roll_value = 10501891;
double max_pitch_value = 906402;
double max_yaw_value = 17997544;
double scaled_rpy[3] = { 0, 0, 0 };
double d_rpy[3] = { 0, 0, 0 };
double angular_velocity[3] = { 0, 0, 0 };
double rpy_rate[3] = { 0, 0, 0 };
double rpy_1_rate_hf[3] = { 0, 0, 0 };
double rpy_2_rate_hf[3] = { 0, 0, 0 };
double rpy_3_rate_hf[3] = { 0, 0, 0 };
double rpy[3] = { 0, 0, 0 };

scaled_rpy[0] = max_roll_angle*((valuearray[sixdofarray::roll_angle].inputvalue) / max_roll_value);
scaled_rpy[1] = max_pitch_angle*((valuearray[sixdofarray::pitch_angle].inputvalue) / max_pitch_value);
scaled_rpy[2] = max_yaw_angle*((valuearray[sixdofarray::yaw_angle].inputvalue) / max_yaw_value);
for (int i = 0; i < 3; ++i) {    if ((scaled_rpy[i] - 0.262) > EPSILON) {
                                                scaled_rpy[i] = 0.262;}
                                    else if ((scaled_rpy[i] + 0.262) < (-EPSILON)) {
                                                scaled_rpy[i] = -0.262;}}
for (int i = 0; i < 3; ++i) { d_rpy[i] = 0.8824*d_rpy_pre_1[i] + 11.76*scaled_rpy[i] - 11.76*rotation_angles_pre_1[i];}
angular_velocity[0] = d_rpy[0] - d_rpy[2]*sin(rotation_angles_pre_1[1]);
angular_velocity[1] = d_rpy[1]*cos(rotation_angles_pre_1[0]) + d_rpy[2]*sin(rotation_angles_pre_1[0])*cos(rotation_angles_pre_1[1]);
angular_velocity[2] = -d_rpy[1]*sin(rotation_angles_pre_1[0]) + d_rpy[2]*cos(rotation_angles_pre_1[0])*cos(rotation_angles_pre_1[1]);
rpy_rate[0] = angular_velocity[0] + angular_velocity[1]*sin(euler_angles_pre_1[0])*tan(euler_angles_pre_1[1]) +
angular_velocity[2]*cos(euler_angles_pre_1[0])*tan(euler_angles_pre_1[1]);
rpy_rate[1] = angular_velocity[1]*cos(euler_angles_pre_1[0]) - angular_velocity[2]*sin(euler_angles_pre_1[0]);
rpy_rate[2] = angular_velocity[1]*(sin(euler_angles_pre_1[0])/cos(euler_angles_pre_1[1])) +
angular_velocity[2]*(cos(euler_angles_pre_1[0])/cos(euler_angles_pre_1[1]));
for (int i = 0; i < 3; ++i) { rpy_1_rate_hf[i] = 0.99*rpy_1_rate_hf_pre_1[i] + 0.995*rpy_rate[i] - 0.995*rpy_rate_pre_1[i];}
for (int i = 0; i < 3; ++i) { rpy_2_rate_hf[i] = 0.99*rpy_2_rate_hf_pre_1[i] + 0.995*rpy_1_rate_hf[i] - 0.995*rpy_1_rate_hf_pre_1[i];}
for (int i = 0; i < 3; ++i) { rpy_3_rate_hf[i] = 0.9695*rpy_3_rate_hf_pre_1[i] + 0.01526*rpy_2_rate_hf[i] + 0.01526*rpy_2_rate_hf_pre_1[i];}
for (int i = 0; i < 3; ++i) { rpy[i] = rpy_pre_1[i] + 0.005*rpy_3_rate_hf[i] + 0.005*rpy_3_rate_hf_pre_1[i];}

double euler_angles[3] = { 0, 0, 0 };
euler_angles[0] = euler_angles_l[0] + rpy[0];
euler_angles[1] = euler_angles_l[1] + rpy[1];
euler_angles[2] = rpy[2];
double actuator_1_length = 0;
double actuator_2_length = 0;
double actuator_3_length = 0;
double alpha, beta = 0;
double p_x, p_y, p_z = 0;
alpha = euler_angles[0];
beta = euler_angles[1];
p_x = 0.5*cos(beta) - 0.36*cos(alpha);
p_y = (-0.0375/0.6)*sin(beta)*sin(alpha);
p_z = p_XYZ[2] + 0.74;
```

```
actuator_1_length = sqrt(pow((p_x + 0.5*cos(beta) - 0.6), 2.0) + pow(p_y, 2.0) + pow((p_z - 0.5*sin(beta)), 2.0)) - 0.74;
actuator_2_length = sqrt(pow((p_x - 0.5*cos(beta) + 0.15*sin(beta)*sin(alpha) + 0.6), 2.0) + pow((p_y + 0.15*cos(alpha) - 0.25), 2.0) + pow((p_z +
0.5*sin(beta) + 0.15*cos(beta)*sin(alpha)), 2.0)) - 0.74;
actuator_3_length = sqrt(pow((p_x - 0.5*cos(beta) - 0.15*sin(beta)*sin(alpha) + 0.6), 2.0) + pow((p_y - 0.15*cos(alpha) + 0.25), 2.0) + pow((p_z +
0.5*sin(beta) - 0.15*cos(beta)*sin(alpha)), 2.0)) - 0.74;

int actuator_length_1 = (actuator_1_length * 1000);
int actuator_length_2 = (actuator_2_length * 1000);
int actuator_length_3 = (actuator_3_length * 1000);
        if (actuator_length_1 < -60) {actuator_length_1 = -60;}
        else if (actuator_length_1 > 60) {actuator_length_1 = 60;}
        if (actuator_length_2 < -60) {actuator_length_2 = -60;}
        else if (actuator_length_2 > 60) {actuator_length_2 = 60;}
        if (actuator_length_3 < -60) {actuator_length_3 = -60;}
        else if (actuator_length_3 > 60) { actuator_length_3 = 60;}

//Now set the output to all actuators simultaneous
        valuearray[sixdofarray::roll_angle].resultvalue = int(euler_angles[0]*1000);
        valuearray[sixdofarray::pitch_angle].resultvalue = int(euler_angles[1]*1000);
        valuearray[sixdofarray::yaw_angle].resultvalue = int(p_XYZ[2]*1000);
        valuearray[sixdofarray::longitudenal_acceleration].resultvalue = int((actuator_length_1 + 71));
        valuearray[sixdofarray::lateral_acceleration].resultvalue = int((actuator_length_2 + 71));
        valuearray[sixdofarray::vertical_acceleration].resultvalue = int((actuator_length_3 + 71));

// Translational reset

for (int i = 0; i < 3; ++i) {   a_XYZ_pre_1[i] = a_XYZ[i];
                                a_H1_XYZ_pre_1[i] = a_H1_XYZ[i];
                                a_H2_XYZ_pre_2[i] = a_H2_XYZ_pre_1[i];
                                a_H2_XYZ_pre_1[i] = a_H2_XYZ[i];
                                p_XYZ_pre_2[i] = p_XYZ_pre_1[i];
                                p_XYZ_pre_1[i] = p_XYZ[i];}
// Tilt reset

for (int i = 0; i < 2; ++i) {   f_xy_pre_1[i] = f_xy[i];
                                fl_1_xy_pre_1[i] = fl_1_xy[i];
                                fl_2_xy_pre_1[i] = fl_2_xy[i];}

// Rotational reset

for (int i = 0; i < 3; ++i) {   d_rpy_pre_1[i] = d_rpy[i];
                                rotation_angles_pre_1[i] = scaled_rpy[i];
                                rpy_1_rate_hf_pre_1[i] = rpy_1_rate_hf[i];
                                rpy_2_rate_hf_pre_1[i] = rpy_2_rate_hf[i];
                                rpy_3_rate_hf_pre_1[i] = rpy_3_rate_hf[i];
                                rpy_rate_pre_1[i] = rpy_rate[i];
                                rpy_pre_1[i] = rpy[i];
                                euler_angles_pre_1[i] = euler_angles[i];}
```