Development of operator interfaces for a heavy maintenance manipulator

A dissertation submitted by

Peter Milani

BE(Mech), GradCert (EngTech), Dip. Pers Man

For the award of

Master of Engineering Research

2014

i. Abstract

This dissertation details the development of an intuitive operator interface for a complex serial manipulator, to be used in heavy maintenance tasks. This interface allows the operator to control the manipulator in the 'task-space', with software handling the conversion to 'joint-space'. Testing of the interfaces shows operator task-space control to be most effective in reducing operator workload and improving the ease of use of a complex machine. These methods are applicable in concept, to a wider range of manipulators and other machines.

A number of operator interfaces were developed: a Joystick Interface, a Master Arm interface and a 6-D Mouse Interface. The Joystick Interface made use of a task space to joint space transformation implemented in software. The Master Arm utilised a scale model to conduct the transformation. Finally, a 3D mouse Interface utilised sensors in an Android Device with a software based task to joint space transformation. These interfaces were tested and the Joystick Interface proved most suitable according to the operator's subjective opinion. Quantitative measurement also showed that it accurately reproduced the operator's commands.

The software transformation developed for the Joystick and 6-D Mouse interfaces utilised the Jacobian Matrix to complete the task-space to joint-space conversion. However, since the manipulator contained a redundant joint, an additional algorithm was required to handle the redundancy. This additional algorithm also improved manipulator safety, as it navigated the arm away from singularities which could result in large joint movement. The novelty of this algorithm is based on its pragmatic approach, and could be modified to achieve a number of safety or performance goals.

The control strategy centred on the operator specifying commands to the arm in the frame of the task. The developed algorithm enabled the control strategy by ensuring that viable solutions for joint velocity could be found in a manipulator that has redundant joints. Furthermore, this algorithm utilised a cost function that minimised the chances of large joint movements due to singularities, improving the safety of the device.

Overall, the project has delivered a viable operator interface for controlling a complex, redundant manipulator. This interface was tested against a number of alternate operator interfaces. The contrasting results of the strengths and weaknesses of various interfaces meant that a number of key insights were gained, and a pragmatic approach to redundancy management was developed.

ii. Certification

I hereby certify that the work contained in the dissertation is the bonafide work of myself and that the work has not been previously submitted for an award. To the best of the candidate's knowledge and belief, the dissertation contains no material previously published or written by another person except where due acknowledgement and reference has been made in the dissertation to that work.

Peter Milani

Date:

Supervisor Signature:

John Billingsley

Date:

iii. Acknowledgements

The following people contributed to the development of the project:

Terry Milani, for assistance in the construction of the prototype and troubleshooting problems along the way.

David Patch, for educating the team about Dragline Maintenance and the initial proposer of the manipulator idea.

Brad Glenwright, for facilitating the site visits and assisting the team with experimental audiences and providing good feedback for the project.

Noel Zahra for support and championing of the project within BMA, particularly with the transition of the project from Norwich Park to Peak Downs Mine.

Lauren Milani, Alan Milani for insight into preparation of the dissertation.

iv. Glossary

The following terms acronyms are used in this dissertation:

2D – two dimensional, describing a position on a plane

3D – three dimensional, describing a position in a volume

6D – six dimensional, describes position in a volume as well as orientation

ADC – Analog to Digital Converter

CLK – Clock signal line in an SPI communications channel

CPU – Central Processing Unit

CS, CSN – Chip Select line in an SPI communications channel

GAZEBO – the Gazebo robot simulation program by the Open Source Robotics Foundation

HMI – Human Machine Interface

HSEC – Health, Safety, Environment, and Community

IOIO – ioio digital acquisition board for java programs

JAVA – Java programming language

LCD – Liquid Crystal Display

MISO – Master In Slave Out line in an SPI communications channel

MOSI – Master Out Slave In line in an SPI communications channel

PID – Proportional Integral Derivative control methodology

PLC – Programmable Logic Controller

PWM – Pulse Width Modulation

ROS – Robot Operating System

SPI – Serial Peripheral Interface, a communications standard

UI – User Interface

USB – Universal Serial Bus

WIFI – Wireless LAN network

Table of Contents

University of Southern Queensland....1
i. Abstract....2
ii. Certification....4
iii. Acknowledgements....5
iv. Glossary....6
1 Introduction....10
2 Background....12
3 Simulation....18
3.1 Hydraulic Actuator Modelling....18
3.2 Hydraulic Actuator Simulation and Control....18
3.3 Simulation of Kinematic Arms....19
4 Interface Design and Development....20
4.1 Introduction....20
4.2 Intuition – Task-space to Joint-space conversion....20
4.2.1 Common Mine Site Interfaces....20
4.3 Solving the Jacobian Matrix....25
4.4 Novel Heavy Maintenance Manipulator Interfaces....29
4.4.1 Novel Method for Resolving Manipulator Joint Redundancies....30
4.4.2 Application of Task space conversion on real interfaces....32
4.4.3 Master Arm Interface....32
4.4.4 Joystick Interface....32
4.4.5 Sensor Interface....33
4.5 Feedback....33
4.5.1 Joystick Interface Force Feedback....34
4.5.2 Master Arm Interface....34
4.5.3 Video Feedback....35
4.6 Design and Ergonomics....40
4.6.1 Common Electrical Framework....40
4.6.2 Master Arm Interface....44
4.6.3 Joystick Interface....47
4.6.4 Sensor Interface....50
4.7 Interface Testing and Results....53
4.7.1 Aim....53
4.7.2 Method....53
4.7.3 Results....57
4.7.4 Discussion....64
4.7.5 Conclusions from Interface Testing....65
5 Conclusion....66
6 Bibliography....68
Appendix A Hydraulic Actuator Modelling....71
A.1. Simple Models – no pressure term....71
A.1.1. Hydraulic Motors versus Hydraulic Cylinders....72
A.2. Simple Models – Pressure and Force....73
A.3. Hydraulic Valve Dynamics....74
A.3.1.3. Simplification – Valve symmetry and matching (Merritt, 1967, p. 82)....79
A.3.2. Valve connected to Actuator combined (Merritt, 1967, pp. 145–148)…82

A.3.3. Complete Hydraulic Cylinder Model....84
A.4. Conclusion....85
Appendix B Simulation Results of Hydraulic Actuators and control systems....87
B.1. Discrete simulation of linear and non-linear models....87
B.1.1. Method....87
B.1.2. Results....90
B.1.3. Discussion....91
B.1.4. Conclusion....92
B.2. Hydraulic cylinder joint servo simulation....92
B.2.1. Geometric Arrangement....92
B.2.2. Results....95
B.2.3. Discussion....97
B.3. Hydraulic motor joint servo simulation....98
B.3.1. Arrangement....98
B.3.2. Response....98
B.3.3. Discussion....99
B.4. Hydraulic Actuator Control....99
B.4.1. PID Control....100
B.4.2. State Space Feedback....108
B.4.3. Variable Structure Control....111
B.4.4. Sliding Mode Control....111
B.4.5. Bang Bang Control....115
B.4.6. Discussion....116
Appendix C Forward Kinematics....119
C.1. Basis for use....119
C.2. Method....119
C.3. Forward Kinematic Example:....120
C.4. Conclusion....124
Appendix D Arm Dynamics - Recursive Newton Euler Algorithm....125
D.1. Incorporating the RNEA into Actuator Modelling....128
Appendix E Custom Manipulator Arm Simulations....129
E.1. Visualising the Manipulator with Processing....129
E.2. Visualising the Arm with Open Scene Graph....130
E.2.1. What is Open Scene Graph....130
E.2.2. Key Concepts in OSG....132
E.2.3. Importing of geometric models in OSG....132
E.2.4. Blender....133
E.3. Conclusion....134
Appendix F Application of ROS and GAZEBO in simulation of manipulators....135
F.1. Robot Operating System....135
F.1.1. ROS Message Framework....136
F.1.2. Developing Manipulator Dynamics....137
F.1.3. Plugins....137
F.2. Conclusion....138

1 Introduction

This dissertation will detail the results of developing and testing an intuitive operator interface for controlling a seven axis, six degree of freedom serial manipulator for use in heavy maintenance tasks. This project is a subset of a larger project to develop the said manipulator and is currently in the process of active development. An intuitive operator interface is important because a large, seven axis, serial manipulator is inherently dangerous. The danger is due to the actuated joints not physically aligning with the intended tasks of the device. The interfaces that were developed in this project allow the operator to issue directions in the task-space, with the interface handling the conversion to the machine's joint-space. The methods used to achieve this are applicable to a wide range of manipulators and other machines.

This dissertation will firstly look at the background problem regarding a manipulator designed to separate workers from the dangers of heavy maintenance. The heavy maintenance environment will be examined, along with the manipulator, in brief. The next section will examine simulation environments developed to better understand hydraulic actuators and closed-loop control methods. A realistic simulation supports the project as this will be the environment used to test the suitability of the interface.

The project utilised the jacobian inverse to transform operator task-space commands into the joint-space of the manipulator. Additionally, because the manipulator contains seven axes, the inverse of the jacobian contains a nullspace which allows an infinite number of solutions. An algorithm is developed to handle the effect of this null space on the possible joint solutions. The algorithm also allows the ability to enhance some performance or safety criteria. In this case, it is to minimise the chance of singularity within the solution. The method for redundancy selection is a pragmatic approach which could be used to select the redundant joint using any manner of criteria.

The final section will cover the development of the three interfaces and the results from testing them on the simulations. The algorithm's effectiveness is demonstrated, and the section outlines the factors that make one interface more intuitive than another interface. The testing regime included results and feedback from the target mine-site operators. Their feedback regarding the interface's suitability, ergonomics and ease of use is detailed and was invaluable in improving interface functionality.

Overall, the project has delivered a functional operator interface for controlling a complex manipulator. A number of key insights into operator interfaces were discovered along with a novel and pragmatic approach to deal with the effects of redundant joints in a serial manipulator. The interface faithfully reproduced the operator's task-space input in the movement of the manipulator's tool.

2 Background

In Open Cut Strip Mining, there are a variety of machines that perform large volume overburden removal in order to expose coal seams for mining. The economic depth of these seams can be between 40 – 80 metres below the surface, as a result, the machines that perform these tasks are uncommonly large. These machines are known as draglines, and the smallest operational size has a approximate weight of about 3000 tons. They are mobile and utilise a walking motion to change position, moving up and down the pit removing overburden. They have been in operation in Queensland since the 1970s. A small dragline digging in its natural environment is

Illustration 1: A Marion 8050 Dragline in operation, conducting pre-strip of overburden. To estimate scale, note the size of the standard walkways on the side of the machine.

shown in Illustration 1.

Most major components on a Dragline are large and heavy. The machine at peak capacity consumes about 1MW of energy and operates at close to a 100% Duty cycle, in normal operation, being shut down for only 12 hours every three weeks. The focus of this energy is the digging bucket, which has to be heavily built to withstand the energies upon it. Typical load capacities are about 50 – 60 tons of material, and a fully laden bucket with supporting rigging is about 160 tons. A map of a standard rigging setup is located in Illustration 2. Note the prevalence of pin connected joints containing double clevises to secure components together.

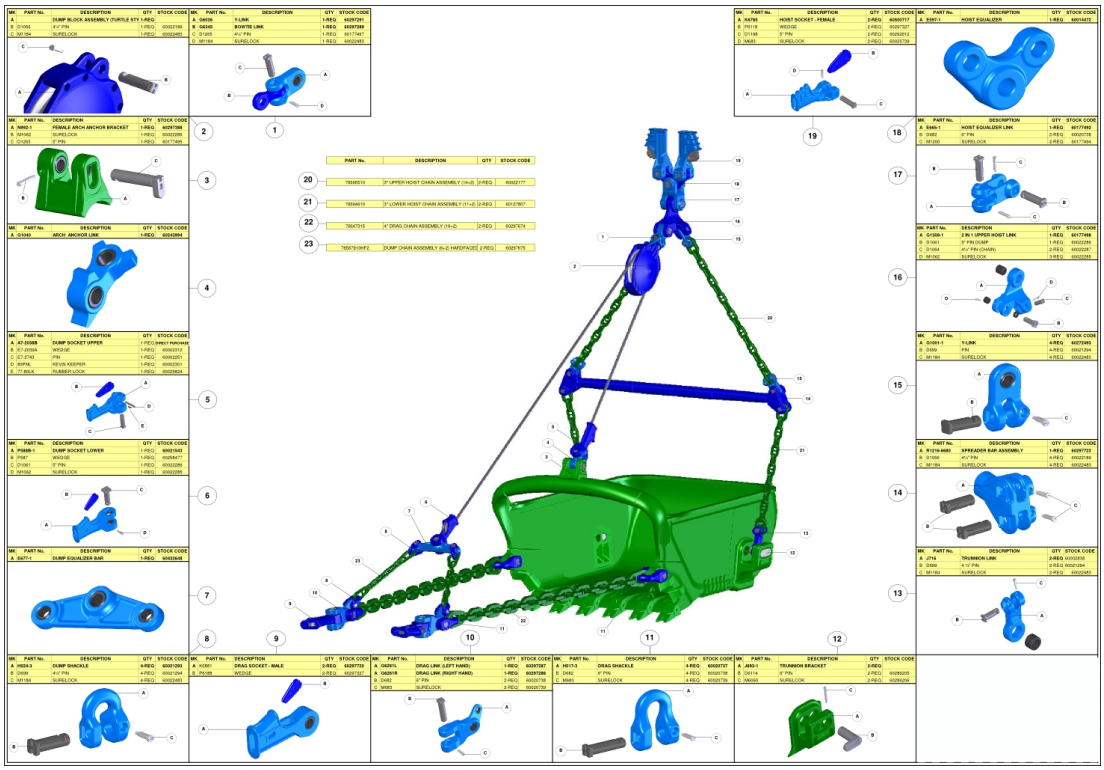


Illustration 2: Rigging Map for a standard rock digging bucket. Note the number of pins and clevises involved in securing the rigging together.

Despite being built strongly, the rigging connecting the bucket to the Hoist or Drag Ropes of the Dragline wears out quickly and is subject to a rigorous maintenance schedule.

Illustration 3: Dragline rigging repair in action. Due to weight the majority of repair actions require use of a 20 ton crane for load support. Many standard rigging elements such as captive hooks are not used in rigging maintenance.

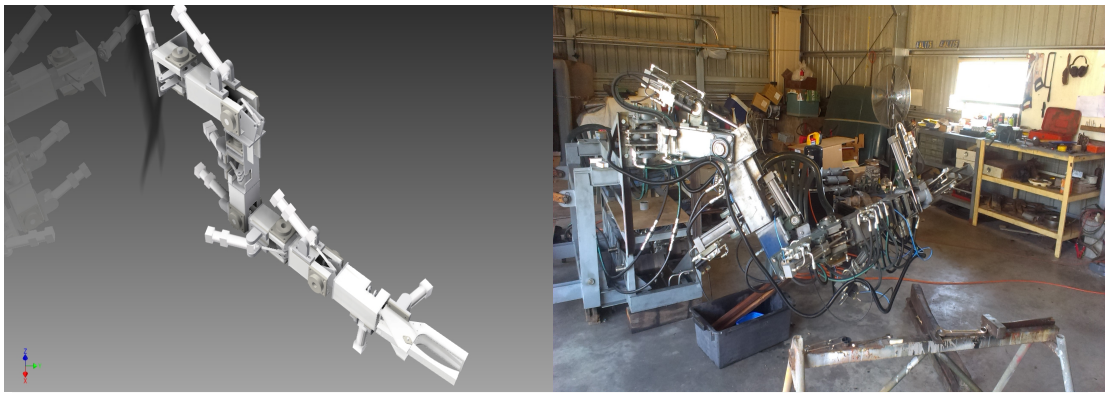
The majority of dragline rigging components are subject to uneven wear as elements rub against each other. As a result, some of the most common rigging maintenance tasks require disassembly, rotation and reassembly so that opposite faces of a component are exposed to wear. Other maintenance tasks simply require part replacement. Some components weigh in excess of half a ton with most weighing greater than 50kg for a small machine. The Lower Hoist Chains, shown being lifted in Illustration 3, weigh 680 kg and the drag socket weighs 830 kg. There is considerable stored energy within dragline parts, and uncontrolled movement in the vicinity of humans easily results in crushing injuries. Other dangers include interaction with 85mm wire rope, which can contain considerable spring tension and has also been responsible for serious injuries.

Illustration 4: Rotation of Lower Hoist Chains requires workers to be exposed to the risk of catastrophic injury in the event of a falling hoist chain.

One of the common dangerous maintenance tasks that are performed is the rotation of Lower Hoist chains. It requires personnel to get beneath a suspended hoist chain in order to remove the chain from the clevis on the trunion link, rotate 180 degrees and place it back into the trunion link. The usual arrangement is shown below in Illustration 4, and has two workers performing the task. Note the full 680kg chain is suspended directly above them by a 20 ton crane. If there is a rigging failure or a failure in the crane, the workers would be directly in the path of the falling chain.

These are the risks that maintenance workers face on a weekly basis. It represents a high risk to mining companies, and as such there is a desire to create a manipulator to separate workers from these risks. This dissertation describes just one part of the overall project to design the manipulator, in particular the design of the operator's control interface that would make control intuitive. This objective was successful in designing an intuitive interface for a seven axis, six degree of freedom manipulator.

The manipulator to be controlled is a hydraulically actuated, seven axis manipulator modelled loosely on the kinematics of a human arm. The first three joints act as a



Drawing 1: Seven Axis, six degree of freedom manipulator arm to be controlled. The prototype built to reflect the modelling.

shoulder, the fourth as an elbow and with the final three joints performing wrist functions. The Gripper is a simple open/close mechanism. The joints are actuated by cylinders fitted with proportional valves, and have ranges of movement from 90 – 140 degrees. The manipulator is shown in Drawing 1.

3 Simulation

Simulation was an important part of the project, and forms the basic framework for the validation of the test results and also to understand the dynamics of the control plant. However, as it does not directly answer the question before us, much of the work in developing the simulations have been moved to the Appendices.

3.1 Hydraulic Actuator Modelling

Three models were developed to simulate hydraulic cylinders and motors. The models attempt to reconcile the actuator pressure, displacement and speed with the valve setting applied. Three models were derived and tested. The first model assumes that pressure remains constant, regardless of load, and that the valve regulates flow through the actuator. The second model assumes that the valve regulates the pressure in the cylinder, and that the flow is dependent on the dynamic movement of the load. In the third model, actuator speed, displacement and internal pressure are all treated as system states and valve displacement has a non-linear relationship with internal pressure and oil flow. The governing equations for the models are derived located in Appendix A.

3.2 Hydraulic Actuator Simulation and Control

The most accurate hydraulic model was simulated to test for accuracy of response. The simulation results for a valve-cylinder combination is contained in Appendix B.1. The preferred model is extended to include the states of a rotary joint being driven by a valve-cylinder combination as described in Appendix B.2 and the results are contained in Appendix B.3. Finally, we examined some linear and non-linear control regimes and measured their performance with respect to gain and also sampling frequency in Appendix B.4.

3.3 Simulation of Kinematic Arms

The actuators above need to be tested on a model of a manipulator arm, so that the actuator dynamics, arm kinematics and ultimately operator interfaces could be assessed. A number of simulation environments were developed that progressively improved the capabilities of the simulation. These simulations included kinematic models, the derivation of which is contained in Appendix C, force-dynamic factors and actuators detailed in Appendix D, visual environments and serial/wireless interfaces are detailed in Appendix E. Finally, the adoption of GAZEBO simulation environment, which allowed all of the above and included accurate collision modelling, meant that the objectives of simulation could be met. The adoption of GAZEBO and Robot Operating System (ROS) is detailed in Appendix F.

4 Interface Design and Development

4.1 Introduction

Most machines have operator interfaces which control the joint space. This is done to simplify the hydraulic control and results in one joint being moved by a single actuator, which is energised by a single control signal. Any additional control signals are usually present to meet some safety requirement. Regardless of whether there is a PLC or computer in the loop, it rarely closes it. Most machines require the operator to close the loop and conduct any transformations between the task action and the joint-space control. The level to which the joint space aligns with the task space, directly corresponds to the ease and intuitiveness of the interface.

4.2 Intuition – Task-space to Joint-space conversion

The level of intuition in an interface depends on how well the interface controls task space coordinates. *Task-space* coordinates are those directions and movements that directly support the completion of a task. These may be the same or at odds to the *Joint-space* coordinates. The Joint-space coordinates are directions and movements achievable by the joints of the machine. In most cases, the interface only controls the joint-space in a machine. The level to which those joints align with the task being done, will indicate how easy it is for the operator to complete a task with the machine. In rare cases, the operator interface will support the conversion between the task-space and joint-space if they are not aligned. If the interface does not support this conversion, and the operator only controls the joint-space of a machine whose joints do not align with the task, operation is only possible with great difficulty, risk, or requirement for a great deal of training and experience.

4.2.1 Common Mine Site Interfaces

In order to illustrate the points above, the dissertation will now examine three mining related machines and their interfaces, contrasting them by their intended task and by the interface provided to achieve it. Their ease of use will be evaluated by studying the length of operator training and importance of operator experience on effective and safe operation. The implications of this study on the Heavy Maintenance Manipulator's interface will be inferred and applied to the interfaces successfully developed and tested on operators.

4.2.1.1 Forklift

A forklift typically has five axes to control: forward, reverse, turning, mast elevation, tilt and width adjustment of the forks as in Illustration 6. These axes are a combination of prismatic and revolute joints, controlled directly by an accelerator, steering wheel and three levers corresponding directly to the joint space. When the task space is analysed for a forklift, four degrees of freedom are identified. A forklift's task requires it to pickup and move pallets from one flat (two dimensional) space to another flat space at any particular height. The control interface consists of the steering wheel and accelerator to control the forklift's 2D position and yaw of the forks. These three degrees of freedom form a *non-holonomic* relationship when

compared to the two controllable axes, steering and acceleration. This relationship means that any position and orientation can be achieved, subject to the path taken (S.M. LaValle, 2006). As a result, an operator is required to perform path planning.

The mast elevation enables movement of the forks in the third dimension, so this expands the task space to include the placing of items at height. A forklift, therefore, has four degrees of freedom in its task space, three position and one orientation. This allows a forklift to complete a task such as picking up a pallet from the back of a truck, taking it into a warehouse and setting it on some pallet racking around 4-5m in height.

The tilt function could be described as a fifth degree of freedom, however, it is probably more accurate to describe it as an axis used in gripping the pallet. It allows the mast to be tilted back so the load does not slip off the forks. Similarly, the fork width function, common on forklifts, is more of a gripper function that allows the safe support of objects that may not all be a standard width. In this way the joint-space is highly aligned with the task-space of the machine, therefore, direct control of joint-space by an operator is valid and intuitive.

Illustration 5: Axes present in typical forklift ("Forklift," n.d.)

The formal training requirements for a forklift are three days("Training and Licencing," 2013)("Forklift Training," 2013), but require the operator to be at least 18 years of age and presumably competent at path planning.

4.2.1.2 Excavator with Backhoe

An excavator conducting backhoe operations should also be examined. For fixed operation, the operator has four axes to control: swing, boom raise and lower, arm raise and lower and bucket flex, as shown in Illustration 6 . The four axes are partially aligned to the task-space. Bucket digging requires the bucket to subscribe an arc in order for it to fill with material.

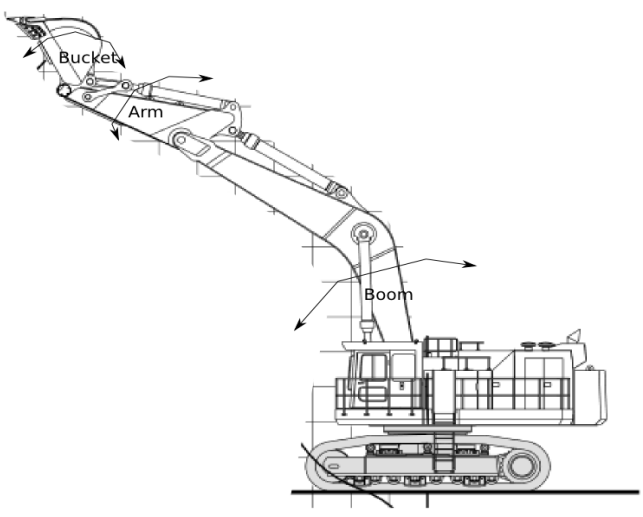


Illustration 6: Excavator axes with backhoe attachment

The digging arcs are supported by the machine's revolute joints. However, the joints are also required to position the bucket in both the radial and vertical space from the swing axis. The position of the bucket is defined in cylindrical task-space coordinates. Therefore, unlike the forklift, an excavator's task space does not correspond directly with the joint-space of the boom-arm-bucket arrangement. In excavator/backhoe operations, the operator controls only joint space. However, the system is determinant, there are four joints, and four degrees of freedom in the task-space. There is only one joint-space solution for a given bucket position and pitch.

As the operator drives the machine in a joint-space which is not aligned with a task space, he/she has to convert between joint-space and task-space. To know what combination of boom angle, arm angle and bucket angle will be required to achieve a given bucket position and pitch requires training and practice. Typical excavator training times are five days for a full course certification(“Excavator Training,” 2013) (“Excavator Course - RIIMPO320B,” 2013). Additionally, it has been shown that operator experience has a significant effect on the productivity of an excavator, due to an experienced operator activating more than one joint at a time. It seems that the ability to transform between task-space and joint-space improves with experience(K. Hughes and X. Jaing, 2010, p. 419).

4.2.1.3 Dragline

A third and final example of user interface which includes different characteristics is that of a dragline. For fixed digging, the operator has three controls, Drag, Hoist and Swing. The task-space of the machine consists of four degrees of freedom, three of position and one of bucket carry angle. The Bucket potentially has a full six degrees of freedom, if tensions in the twin ropes is ever unequal, but the task space of the machine is only defined in four dimensions: three cylindrical position coordinates and the bucket carry angle. As the operator only has three controls of a joint-space that consists of Drag, Hoist and Swing, the system is under-determined (P. Ridley, 2004, p. 17). This means the operator does not have full control of all four task-space degrees of freedom at any operating point. The under-determined nature of the system manifests itself as lines of constant carry angle that are relatively fixed (depending on bucket load distribution(P. Ridley and R. Algra, 2004, p. 1001)) as shown in Illustration 7. Without using bucket inertia to expand the task-space, operators have a fixed envelope of where they can dump the bucket, i.e. achieve a carry angle of -90deg by releasing tension on the Dump Ropes. In order to conduct digging, the bucket must be constrained by the ground and allow the Drag tension to pull the bucket full.

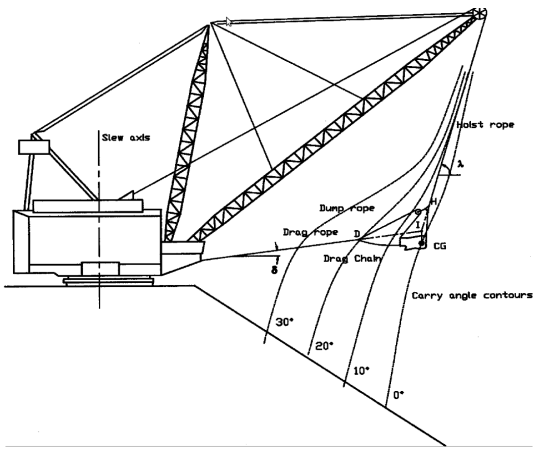


Illustration 7: Representation of lines of constant carry angle for a Dragline Bucket

The position of the bucket radially and vertically is dependent on a combination of Drag and Hoist Tensions, which are also tied to the carry angle of the bucket. It is only through

the design of the Dragline system that the carry of the bucket corresponds to the intended method of operation or task-space of the machine. This system requires transformation by the operator from the Drag and Hoist joint-spaces to a non-linear position and indeterminate carry-angle task-space. Of the three machines examined, due to the operator interface complexity, dragline operators can be expected to require the most training and experience for effective production.

4.2.1.4 Summary of common mine site machine interfaces

Machine	Operator Interface	Joint-space/Task space correspond?	Operator Controls	Task Space DOF	Indeterminate system?	Order of difficulty in operation
Forklift	Joint Space	Partially	3	4	Yes, common non-holonomic for navigation	minimum
Excavator -fixed	Joint Space	No	4	4	No	medium
Dragline -fixed	Joint Space	No	3	4	Yes	maximum

Table 1: Summary of common mine site machinery task-space, joint-space and operator interface relationships against difficulty of operation.

A summary of common site operator interfaces that have been examined is contained in Table 1. It is obvious that the ease of effective machine operation is due, in part at least, to how well the operator interface corresponds to the task being undertaken. If the task-space and the operator input correspond, a large number of degrees of freedom may be manipulated relatively easily. Where a machine is controlled in joint-space, particularly one which does not correspond to its task-space, the operator is required to close the loop and perform the conversions between the joint-space of the controls and the task-space of the machine. The conversion requires training and experience. Typically, the less the joints correspond to the task, the more training is required for effective and safe operation.

The next section will examine a serially-linked manipulator with a tool at the end. The relationship between the position of the joints (the joint-space), and the position of the tool in six degrees of freedom (the task-space) will also be evaluated. This relationship is determined by forward kinematics when converting from joint-space to task-space, and the change in the task-space can be estimated from the change in the joint-space through a mathematical device known as the jacobian.

4.3 Solving the Jacobian Matrix

When trying to control a manipulator arm, the typical problem solved by forward kinematics is: “What is the tool position and orientation for a given set of joint angles?”. This is usually straightforward to solve. For a treatment of this problem see Appendix C. In most cases an operator can see where the tool is and also knows where it needs to be. The problem for the operator is in deciding the changes in joint angles that are required to allow a desired change in tool position. Solving this problem mathematically is a more interesting question. The problem is best stated as: “Given an arm arrangement, how must the joint angles change to achieve a desired

change in tool position or orientation?" Expressed differently: "What "twitches" in joint angle are required to change tool position or orientation by a small amount?" The answer lies in the Jacobian Matrix.

For the next section, it will be assumed that the arm has only six joints. This is the exact number of joints to ensure that there is only one solution to our query. A full description of position consists of three values; x, y, and z, and full orientation description consists of three values; θ, φ, γ. If there are six joints, any change in tool position or orientation must be the sum of tool changes due to variations in each joint. This creates six equations describing that relationship, one for each joint. This relationship is also known as the Jacobian. The changes in the tool become the six unknowns. As there are six equations and six unknowns, the system is determinate and there is only one solution for the unknowns.

The Jacobian is the matrix of first order partial derivatives of a vector valued function ("Jacobian Matrix," 2013). In this case, it is a matrix of how each of the tool orientation and position coordinates change with slight changes in each of the joint angles. Tool change over joint change becomes the partial derivative. As stated above, it describes the change of tool position as the sum of the tool position/orientation changes due to changes in each joint. It is expressed below:

$$J=\begin{vmatrix} \frac{\partial x}{\partial \theta_1} & \frac{\partial x}{\partial \theta_2} & \frac{\partial x}{\partial \theta_3} & \frac{\partial x}{\partial \theta_4} & \frac{\partial x}{\partial \theta_5} & \frac{\partial x}{\partial \theta_6} \\ \frac{\partial y}{\partial \theta_1} & \frac{\partial y}{\partial \theta_2} & \frac{\partial y}{\partial \theta_3} & \frac{\partial y}{\partial \theta_4} & \frac{\partial y}{\partial \theta_5} & \frac{\partial y}{\partial \theta_6} \\ \frac{\partial z}{\partial \theta_1} & \frac{\partial z}{\partial \theta_2} & \frac{\partial z}{\partial \theta_3} & \frac{\partial z}{\partial \theta_4} & \frac{\partial z}{\partial \theta_5} & \frac{\partial z}{\partial \theta_6} \\ \frac{\partial \beta}{\partial \theta_1} & \frac{\partial \beta}{\partial \theta_2} & \frac{\partial \beta}{\partial \theta_3} & \frac{\partial \beta}{\partial \theta_4} & \frac{\partial \beta}{\partial \theta_5} & \frac{\partial \beta}{\partial \theta_6} \\ \frac{\partial \varphi}{\partial \theta_1} & \frac{\partial \varphi}{\partial \theta_2} & \frac{\partial \varphi}{\partial \theta_3} & \frac{\partial \varphi}{\partial \theta_4} & \frac{\partial \varphi}{\partial \theta_5} & \frac{\partial \varphi}{\partial \theta_6} \\ \frac{\partial \gamma}{\partial \theta_1} & \frac{\partial \gamma}{\partial \theta_2} & \frac{\partial \gamma}{\partial \theta_3} & \frac{\partial \gamma}{\partial \theta_4} & \frac{\partial \gamma}{\partial \theta_5} & \frac{\partial \gamma}{\partial \theta_6} \end{vmatrix}$$

It can be used to determine the small changes in tool position and orientation δx given small changes in joint angle $\delta\theta$, both 6x1 vectors as shown below:

$$\delta x = J\,\delta\theta$$

More importantly, the expression can be inverted to determine the machine operator's query: "What changes in joint angle are required for small changes in position or orientation?"

$$\delta\theta = J^{-1}\delta x$$

If a vector representing small changes in orientation and position is multiplied by the inverse of the Jacobian, the solution is the required small changes in the joints $\delta\theta$. This works quite well, the inverse of the 6x6 matrix can be computed using *Gaussian Elimination* ("Gaussian Elimination," 2013) or evaluated analytically using

the *shur's complement* (“Shur Complement,” 01 Sep 13). Care has to be taken when dealing with a determinate system to avoid singularities. Singularities are parts of the joint space for which a large change in joint angle only causes a small change in tool location. They are difficult to detect and often rely on a careful off-line analysis of all spaces where the singularities occur, either through analytical or numerical means.

4.3.1.1 Contemporary way for Dealing with Redundant Manipulators

The case has been examined where there are six axes for the six position and orientation components. However, a six axis manipulator can be limited if a particular joint angle cannot be achieved. As we have already examined, there is only one set of joint solutions for a given position and orientation, so if a solution cannot be achieved because a joint cannot change as desired, the possibility of achieving the tool target position and orientation would be undermined. By adding an additional joint to the arm, the number of viable solutions for position and orientation can be increased, however the system is now over-determined, and one of the possible solutions must be chosen. In fact, there is an infinite number of angle changes over two or more axes (for a manipulator without limits) that can completely cancel each other out, this is known as the *null* space of the manipulator. It exists whenever there is a non-square matrix with full rank that requires inversion. Therefore, if it is not dealt with appropriately, the manipulator may not move at all!

This project developed a novel procedure for avoiding the null-space and attaining other desirable attributes. Firstly, we will examine how redundancy is handled in contemporary manipulators. A jacobian can still be constructed for a seven axis system, it is now becomes a rectangular 6x7 matrix:

$$J=\begin{vmatrix} \frac{\partial x}{\partial \theta_1} & \frac{\partial x}{\partial \theta_2} & \frac{\partial x}{\partial \theta_3} & \frac{\partial x}{\partial \theta_4} & \frac{\partial x}{\partial \theta_5} & \frac{\partial x}{\partial \theta_6} & \frac{\partial x}{\partial \theta_7} \\ \frac{\partial y}{\partial \theta_1} & \frac{\partial y}{\partial \theta_2} & \frac{\partial y}{\partial \theta_3} & \frac{\partial y}{\partial \theta_4} & \frac{\partial y}{\partial \theta_5} & \frac{\partial y}{\partial \theta_6} & \frac{\partial y}{\partial \theta_7} \\ \frac{\partial z}{\partial \theta_1} & \frac{\partial z}{\partial \theta_2} & \frac{\partial z}{\partial \theta_3} & \frac{\partial z}{\partial \theta_4} & \frac{\partial z}{\partial \theta_5} & \frac{\partial z}{\partial \theta_6} & \frac{\partial z}{\partial \theta_7} \\ \frac{\partial \beta}{\partial \theta_1} & \frac{\partial \beta}{\partial \theta_2} & \frac{\partial \beta}{\partial \theta_3} & \frac{\partial \beta}{\partial \theta_4} & \frac{\partial \beta}{\partial \theta_5} & \frac{\partial \beta}{\partial \theta_6} & \frac{\partial \beta}{\partial \theta_7} \\ \frac{\partial \varphi}{\partial \theta_1} & \frac{\partial \varphi}{\partial \theta_2} & \frac{\partial \varphi}{\partial \theta_3} & \frac{\partial \varphi}{\partial \theta_4} & \frac{\partial \varphi}{\partial \theta_5} & \frac{\partial \varphi}{\partial \theta_6} & \frac{\partial \varphi}{\partial \theta_7} \\ \frac{\partial \gamma}{\partial \theta_1} & \frac{\partial \gamma}{\partial \theta_2} & \frac{\partial \gamma}{\partial \theta_3} & \frac{\partial \gamma}{\partial \theta_4} & \frac{\partial \gamma}{\partial \theta_5} & \frac{\partial \gamma}{\partial \theta_6} & \frac{\partial \gamma}{\partial \theta_7} \end{vmatrix}$$

A pseudo inverse can be calculated for this matrix and full rank of the matrix can be expected as each of the arm joints are physically independent elements and do not drive each other. As a result the Moore-Penrose pseudo inverse $J^{\dagger}$ can be calculated directly from:

$$J^{\dagger}=J^{T}\left(JJ^{T}\right)^{-1}$$ (“Matrix PseudoInverse,” 2013, p. 2)

where J^T is the matrix transpose. This pseudo inverse can be used directly, however it is usually combined with a weighting matrix W to deal with singularities and to minimise performance metrics. There is no set formula for determining the weighting matrix and it selection is rather arbitrary. If W is to be used then it is typically used to weight the pseudo inverse in the method shown below:

$$J_W^\dagger = W^{-1} J^T \left(J W^{-1} J^T \right)^{-1}$$ (L. Beiner and J. Mattila, 1999, p. 176)

This weighting matrix adjusts the null space of the jacobian. It adjusts these in order to achieve some performance metric, such as lower joint torque, kinetic energy, avoiding joint limits and so on. Typically, developing this weighting matrix is specific to the manipulator and its current configuration, therefore can be somewhat computationally involved in its determination and requires updating as the manipulator moves.

4.3.1.2 Conclusion

Understanding the kinematics of an arm is useful for both simulation and control. The Forward Kinematics converts the joint space to the world coordinate system task space. The inverse kinematics is much harder to calculate, but if a starting orientation is known, then joint space can be modified to achieve a task space goal. When dealing with redundant joints, the contemporary methods require a weighting matrix which can be computationally intensive to determine as it changes when the manipulator moves. This dissertation will now propose a more pragmatic approach to the elimination of the redundancy.

4.4 Novel Heavy Maintenance Manipulator Interfaces

The Heavy Maintenance Manipulator has seven joint axes that position the tool with six degrees of freedom, in order to maximise flexibility and range of movement. As it is desired that the operator's input describes the task-space, the interface is required to convert between this input and the joint-space of the machine. A pragmatic algorithm for dealing with the redundant joint will be proposed.

4.4.1 Novel Method for Resolving Manipulator Joint Redundancies

```
Redundant Joint Pseudo Code

//Make each joint redundant in turn and examine the solution to the changes in
//Joint angle to find the solution which has the smallest maximum change. This
//becomes the desired joint to make redundant.

float tempmax =0
number_of_joints =7

for i =0 to  (number_of_joints-1) then

    matrix6x7 J = 0
    matrix6x6 Jshort =0
    vector6x1 dtheta=0
    vector6x1 dx
    vector7x1 dthetafull =0;
    vector<vector7x1> jointsolutions;

    //develop the full 6x7 Jacobean
    J = return_full_jacobean()

    //remove the column corresponding to redundant joint
    Jshort = remove_redundant(i)

   //invert and multiply with desired task space to get joint space
   //changes
   dtheta = Jshort.invert.mult(dx)

  //adjust dtheta to reinsert the joint that is redundant, making
  //it equal to zero and thus a representation of all seven joints
  dthetafull = insertRedundancy(dtheta, i)
  jointsolutions.add(dthetafull);


  //check the max of the resulting vector of seven joints
  // choose the one with the smallest maximum
  if (max(jointsolutions.get(i))<tempmax) then

      tempmax = max(jointsolutions.get(i));
  end
end
```

Illustration 8: Pseudo code for the determination of redundant joint and avoidance of singularities.

In this project, redundancy was handled in a novel way, avoiding the complexity of an overly analytical weighting matrix. The goals of this novel method were primarily to minimise the chance of singularities due to HSEC consequences of large, impromptu joint changes and to provide workable solutions to the problem.

The method treated each joint as being fixed in turn, and calculated a square jacobian for each of these cases using the remaining six joints. The joint-space changes were calculated for each of these cases, inserting a zero joint change for the joint that is held fixed. Finally, the set of seven joint changes results are compared and the case that has the smallest, maximum joint change is chosen. This ensures to steer us around the singularities which have been identified as large joint changes for a given tool change. This method is shown in Illustration 8.

Where seven different joints can be made redundant, there will be seven possible solutions (cases). It is important to note that once the seven possible solutions are calculated, the method for selecting the redundant joint can be based on what ever criteria the designer desires. In this case, the redundant joint was chosen to minimise the size of the joint changes required (thus singularities), but other criteria may be chosen to minimise joint torques, avoid limits or avoid obstacles. As the manipulator only has position sensors fitted, such alternative criteria could not be implemented.

In the general case, where we have a manipulator with *n* redundant joints, where:

$$n = j - m$$

where *j* is the number of joints and *m* is the number of degrees of freedom. The operational complexity increases exponentially with the number of possible solutions equal to:

$$(m+n)^n$$

This does not include the computation required to implement the selection criteria. Thus the computational cost can dramatically increase with every additional redundant joint. This may not be such a problem as the cost of computation is relatively cheap.

4.4.2 Application of Task space conversion on real interfaces

Two methods for calculating the joint-space of the manipulator from the user inputs were used in the three main interfaces successfully developed in this project:

1. Master Arm Interface,
2. Joystick Interface, and
3. Sensor Interface.

4.4.3 Master Arm Interface

The Master Arm Interface Interface consists of a miniature model of the arm that was kinematically equivalent to the manipulator. As the operator moves the Master Arm in what is task-space control of the tool, the joints angles on the Master Arm are measured and are applied as reference positions to servo joints on the manipulator, or Slave Arm. Thus the Slave Arm follows where the Master Arm leads. The operator focuses on moving the Master Arm, so as to control the tool position in task-space, as the Master moves, the joint-space solution is solved by mechanical movement of its joints. Thus we have a user interface which allows control in task-space, the transformation to joint-space is achieved by the mechanical replication between the Master and the Slave Arm. The software to achieve this is relatively minimal and acts mainly as a transport medium for the information from the Master to the Slave.

4.4.4 Joystick Interface

In the Joystick Interface, joysticks are utilised to gather user task space input. The joystick axes were aligned to the direction of the task-space axes being controlled. For example, if the operator wants to move the tool left, a joystick will need to be pushed left. If an up movement is required, a joystick is pushed upwards.

In order to achieve this effect, software implements our task-space to joint-space conversion. It treats the measurement of the joystick as a task-space change and calculates the appropriate changes in joint-space as described. Task-space control of this sort has been used on simpler manipulators before and has shown to improve the performance of novice and experienced operators (A. Hansson and M. Servin, 2010, p. 1074).

4.4.5 Sensor Interface

In the Sensor Interface, an acceleration sensor and a magnetic sensor were used to determine the orientation of a handheld device (android phone). The touchscreen interface implemented methods for indicating task-space position and orientation change of the tool. Once these commands were input, the same software framework as per the Joystick Interface was used to determine the appropriate conversion to joint-space.

4.5 Feedback

An important aspect of operator interfaces is the provision of feedback to the operator. In most existing cases this requirement includes a visual reference, either a light or display, for load and equipment state. In hydraulically actuated machines, where the operating levers directly actuate the hydraulic valves, the operator can feel and hear the amount of fluid flow through the valve. This assists the operator to assess the required spool displacement required to achieve a particular level of joint movement, resulting in a slower, smoother joint movement (K. Hughes and X. Jaing, 2010, p. 419).

In electrical interfaces, force feedback is typically not incorporated into the interface. This results in lower operator fatigue and faster manipulation of the joints (K. Hughes and X. Jaing, 2010, p. 417). As a consequence with no other controls in place, electronic joint control interfaces can result in a higher level of wear and fatigue on the mechanical structure of the machine.

For the three successful interfaces developed, feedback consisted of:

1. lights on the interface to show the status of the interface, and
2. lights on the Slave Arm to indicate the state of the Arm.

During the course of the project, a number of other methods of feedback were investigated but were not ultimately incorporated into the interface at this time. These require additional work to be successfully incorporated into the operator interface. These methods of feedback were:

1. force feedback on Joystick Interface;
2. force feedback on Master Arm Interface; and

3. video Feedback of arm movement.

These methods of feedback will be examined further.

4.5.1 Joystick Interface Force Feedback

The provision of force feedback for the Joystick Interface was applied through the direct connection of motors to the axes of a joystick. A PID signal adjusted the motor voltage, based on the position of the joystick. If a larger force were required, the reference for the PID would be moved so as to increase the motor signal and hence the force in a particular direction.

The configuration ran into a number of issues, particularly tuning the PID to overcome the large amounts of backlash and friction in the gears of joystick. The motor, even when geared, produced only a minimal level of force that was hardly felt by the operator. A key issue was the mechanical cradling of a 2-D gimbal that described x and y movement, in a structure that allowed movement in the z direction. The third joystick axis introduced additional mechanical complexity which was difficult to solve.

Overall, the use of a PID position-controlled motor in this way was not an effective option for achieving force feedback. A better option would have included a current sensor and direct control of the motor torque. A force controlled actuator would have been beneficial as long as the max force of the actuator could be felt by the operator. This requires larger motors, heavier gearing or greater mechanical advantage, all equalling a larger, bulkier interface. The use of low pressure hydraulics could have been an option as well, but at significant cost of actuators, valves and pressure sensors.

4.5.2 Master Arm Interface

A version of the Master Arm Interface was built that incorporated servo motors linked to actuate the joint through the use of tension springs. Position sensors were utilised to measure the joint angle, and an algorithm was arranged so that the force required on the joint was a function of the difference in angle between the joint and the Servos. When there is a difference, a spring would be extended creating a torque on the joint. The joint actuated in this way would have a limit of about 90 degrees before the springs would start to experience extension regardless of the difference in servo/joint angle. This was not such a problem as most joints were about 120 degrees in range and the force due to the spring extension past 90 degrees would serve to apply limits to the Master Arm.

Additionally, to combat the friction issues encountered with the force feedback Joystick, the joints were supported by small bearings. Overall, the system was let down by the size of the electric servos and the weight of the arm. The arm was built heavier than usual to support the bearings and the servos. As a result, the servos working against gravity were overloaded and not effective. To apply force feedback on a Master Arm in this way would require both a very light arm and stronger servos in the heaviest loaded joints. Unfortunately, time ran out before this was achieved.

4.5.3 Video Feedback

Video feedback is included in this section since the preferred arrangement has not been conclusively tested, though there are some aspects that will be shared here. The physical implementation of video feedback was achieved with the ROS *image_transport* and *uvc_camera* library allowing output of video feed to a normal computer screen or to an Android Tablet. It was found during simulation that a single camera is less effective because there could be no perception of depth thus tool position and orientation could only partially be evaluated by an operator. A two camera arrangement improves performance with depth much easier to perceive. A wider question emerges with multiple cameras: How should multiple cameras be mounted to best support the operator? This question is affected by a multitude of variables. In this we trialled two, twin video cameras in two configurations:

1. Configuration 1: A Head Camera and a Tool Camera,
2. Configuration 2: Two Head Cameras.

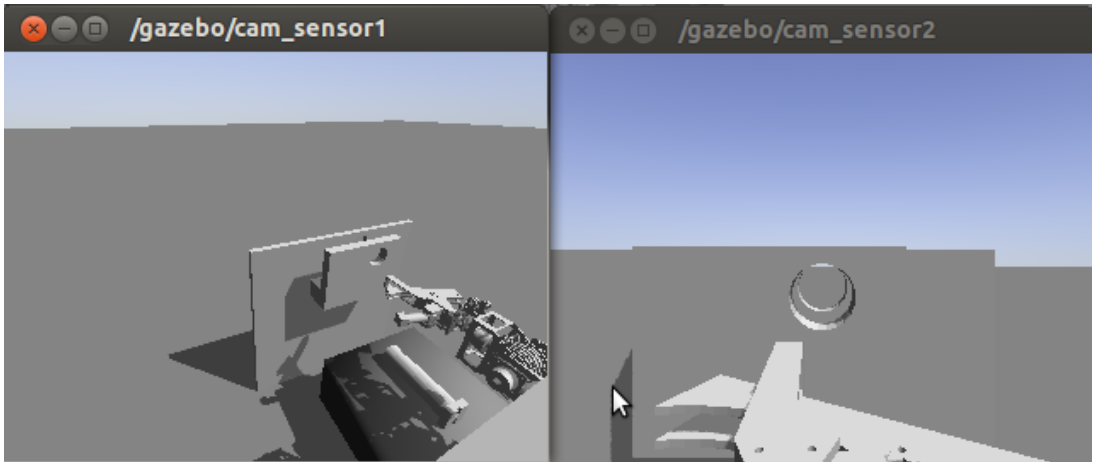


Illustration 9: Output of Camera Configuration 1, A Head Camera and Tool Camera

Configuration 1 has some problems, particularly when there is little variation in background and when the Hand Camera shows objects which are not visible by the Head Camera. This can lead to disorientation of the operator that can not be rectified by an of the information presented by the cameras. If the alignment axes are coaxial to the camera, lining up of the object by Configuration 1 is supported. This is dependent on the grip that can be achieved on the object to be lined up. If it does not result in coaxial alignment between the camera axis and the task axis, the Tool Camera's utility is reduced.

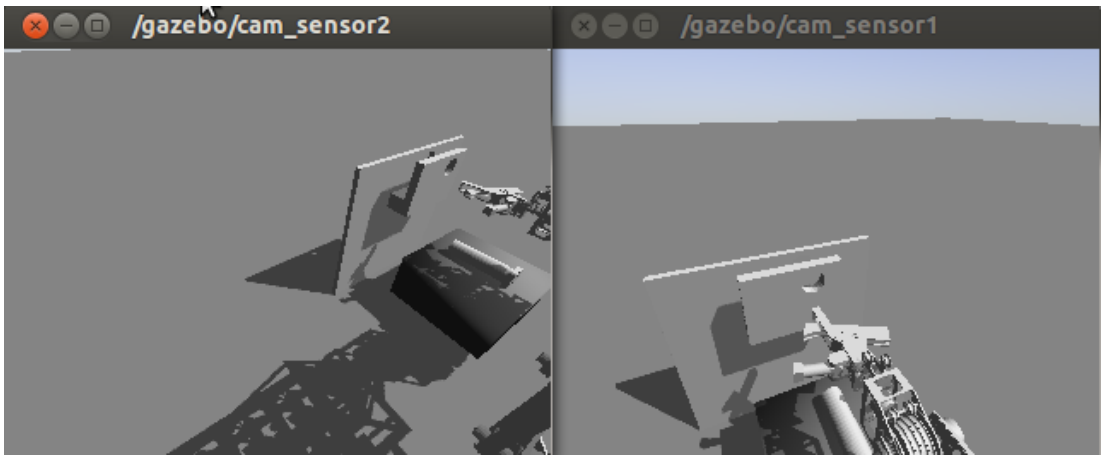


Illustration 10: Camera Configuration 2, two head cameras providing some depth perception

In Camera Configuration 2, two head cameras providing some level of depth and height perception via a stereoscopic arrangement. The left-hand camera's field of view is orientated more towards the intended workspace, and provides a view that assists the estimation of vertical position and forward position. The right-hand camera is better suited for estimation of lateral position. These two views highlight one of the major problems with video feedback via cameras, namely the narrow field of view that does not cover the whole possible workspace. A solution would be to either install more cameras or develop a camera mount that revolves around the x and z axes (altitude-azimuth mount) to allow the camera to focus on some moveable point of interest.

Using multiple additional cameras would likely introduce bandwidth problems if they were streaming concurrently. This could be contrasted with extra control required of the camera platform. Bandwidth issues could be mitigated by automatic control of which cameras are streaming, and a moveable camera platform could be controlled to automatically focus on the tool.

Due to the relatively little evidence supporting one camera arrangement and the restrictions on fixed cameras, the video feedback question cannot be resolved without further testing. The preferred method of visual feedback at this stage will be to have the operator on the ground, out of harm's way, controlling the position and orientation of the manipulator without the help of a video feed.

4.5.3.1 System State Feedback

A light system was introduced on the operator interface and on the slave manipulator in order to provide direct feedback as to the state of the object. The operator interface light system is shown in Table 2.

Light	State	Interface Active	Output Published
Green	On	Yes	No
Orange	Actuated	Yes	Yes

Table 2: Light-based feedback for operator interface.

On the Slave Manipulator, the light system also identifies the system's state, but specifically for the manipulator. The Manipulator light system is shown in Table 3.

Light	State	Valves Active	Joint State Evaluated
Red	Error	No	Yes – errors found
Blue	On	No	Yes
Amber	Joints Active, Valves Active	Yes	Yes

Table 3: Light based feedback for Slave Manipulator.

Overall, the light system is very effective in allowing understanding of a system's state, based on a quick glance at the light and an understanding of the state of the machine.

4.5.3.2 Conclusion

Feedback in an interface has important implications with respect to the overall effectiveness of the system. Like most existing machines, the level of feedback achieved by this interface consisted of purely visual references such as lights. Force feedback was attempted but was undermined by the mechanical implementation factors. These factors could be alleviated through better tooling and construction of the interfaces. Video feedback was also investigated and holds great promise, particularly if stereoscopic sensors on a movable altitude-azimuth base are implemented. More work is required to achieve an adequate level of sensor based feedback.

4.6 Design and Ergonomics

4.6.1 Common Electrical Framework

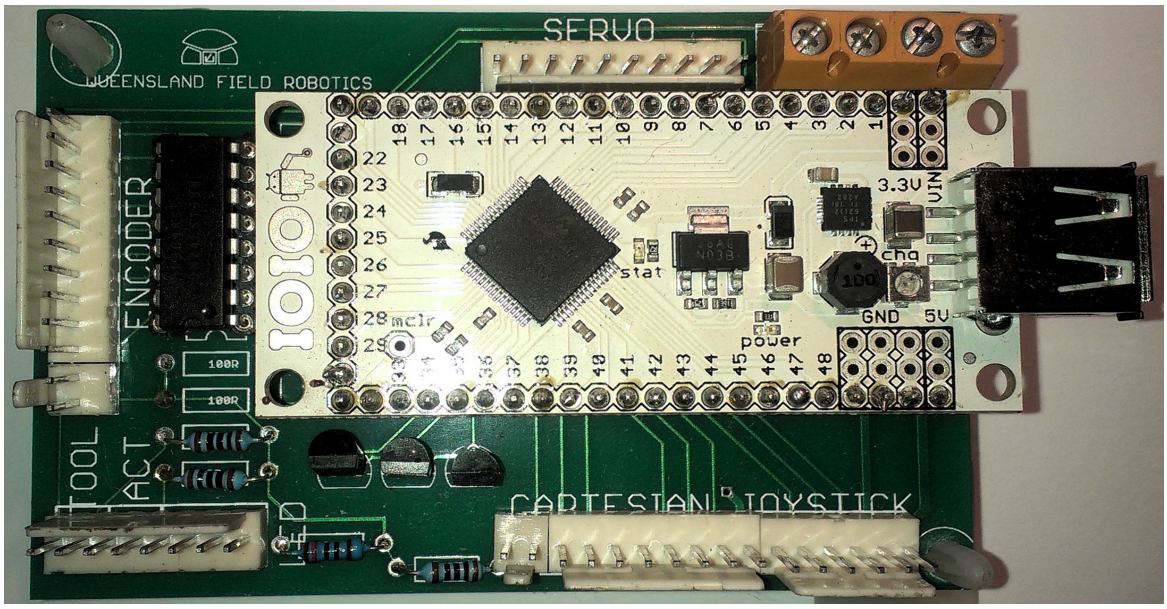


Illustration 11: Common Interface Development Circuit

For the electrical and computational support of all interfaces, a common electrical circuit was designed. This circuit supported both the Master Arm Interface and the Joystick Interface, as well as the common Actuation Button, Tool Button and Status Light. The circuit was centred around the input/output device IOIO, which is a digital acquisition board designed for use with JAVA Android devices. It provides digital, analog and serial interfaces to an Android application via a USB connection. Due to the USB connection, there is a sizeable lag penalty compared to a typical embedded application for every input-output operation. There is however, a massive computational benefit in incorporating an Android platform, or any high-speed 32-bit device into an embedded application due to their additional computational power when dealing with floating point arithmetic. The Android platform provides many connection options utilising Bluetooth™ and WIFI, though not normally a wired connection. Additionally, Android provides a ready made visual user interface through the LCD touchscreen.

It was a very flexible development arrangement. The circuit provides up to six analog inputs for reading thumb joysticks, a Serial Peripheral Interface (SPI) for reading up to seven joint sensors, and seven Pulse Width Modulation (PWM) outputs for application in interfaces incorporating force feedback. It can drive a tricolour LED with common anode, and has inputs for an Actuation Button and Tool Button. The power supply is handled by the IOIO which provides 5V and 3.3V power to the interfaces from a 12V feed. It also charged the Android device via the USB connection. The interfaces were connected via polarised 0.1” headers. Overall, the common circuit for interfaces improved electrical reliability and explored packaging

options for the future. The block diagram for the circuit is shown in Illustration 12.

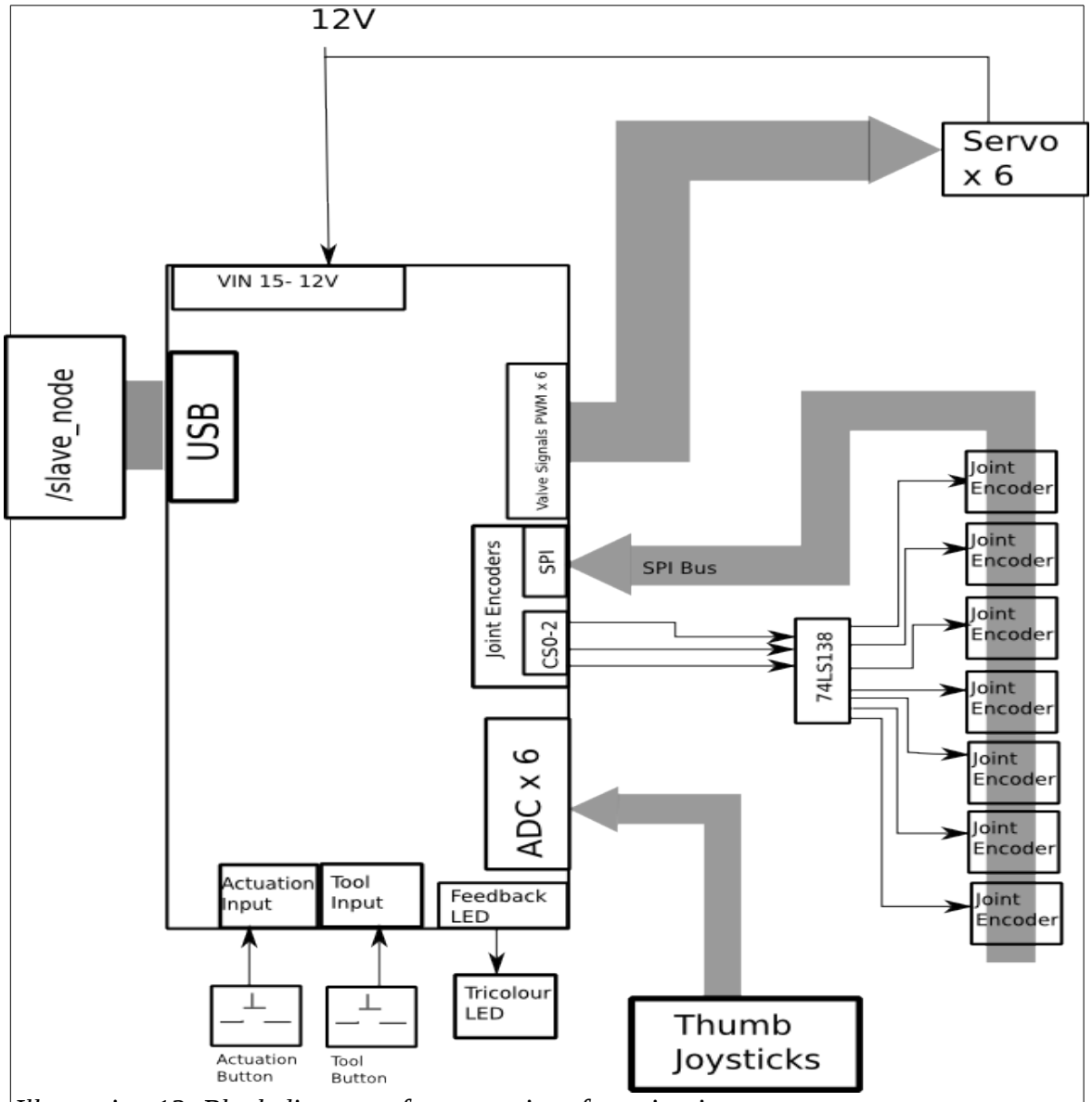


Illustration 12: Block diagram of common interface circuit

4.6.2 Master Arm Interface

4.6.2.1 Mechanical Design Considerations

The Master Arm Interface replicates the kinematics of the slave arm such that for any position or orientation pose, if the joint angles in the Master Arm are replicated in the slave arm, then a similar pose and orientation is expected from the Slave Arm. The test model was built out of acrylic plastic, and utilised 10 bit position controllers for joint angle reading. A grip was provided to reduce fatigue on the operator's hand. The Arm was fixed so that it would be close to the operator's left hand when seated. The operator could control the arm, minimising the level of interference with the operator's body. The Master Arm was modelled as 1:1 to a human arm.

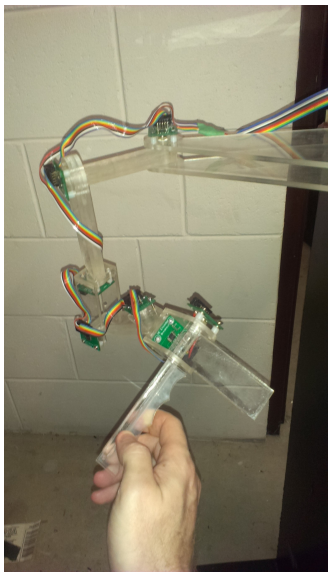


Illustration 13: Master Arm Interface showing Joints

The larger the Master Arm, the lower the gain of the system. That is, the further the joint is from the tool, the less it will move for a given task-space movement of the tool. The effect of this is that larger models will have a much higher level of precision compared to those which are miniaturised. This in turn impacts on the space required to utilise one of these interfaces, as longer link lengths require more movement to achieve a given output on the Slave Arm.

Each joint consists of a single or pair of non-lubricated plain bearings, with a loose fit tolerance. This allows relative ease of movement, and had an acceptable level of positional accuracy. It provided a very good precision in the joint sensors. Tool button, actuation button and status light were placed in a console for operation by the right hand.

Overall the mechanical design used for testing of the Master Arm Interface was capable and achieved a good response when tested on Coal Mining Workers for both intuitiveness and accuracy.

4.6.2.2 Electrical Design Considerations

Circuits specific to the Master Arm Interface are the joint encoders. These are the same encoders that are used on the Slave Arm. They provide a variety of outputs, but the SPI interface was specifically chosen as it provided measurements of absolute position. This eliminated the requirement for a high frequency update loop to poll the output of an Encoder or interrupt driven interfaces. Additionally there was no requirement for post-processing to produce an angle measurement which could experience drift from a zero position if measurements were lost. It also minimised the number of wires required as a common data, and clock bus could be shared between all encoders. The output from the encoders was very stable with only small changes to the Lower Significant Bit of around 0.1-0.2%.

The full SPI interface requires four wires to achieve full-duplex communications: Master In Slave Out (MISO), Master Out Slave In (MOSI), Clock (CLK) and a Chip Select ($\overline{CS}$) for every slave device. For these encoders, the MOSI line was not required as the encoder didn't receive any information during normal operation and the MOSI line could be dropped.

To save data lines on the IOIO, the $\overline{CS}$ was provided by a 74LS138 3 to 8 line decoder/multiplexer to service seven slave devices using only three lines from the IOIO. The normal Chip Select pin in the software SPI module was not utilised. The arrangement used is shown in Illustration 12. The MISO and CLK lines were maintained on a common bus for all sensors. The hardware to connect the buses consisted of a single polarised header. Ribbon cable was used to provide a single line of cabling down the arm. Each $\overline{CS}$ had its own separate line down the arm.

The SPI output from an encoder is shown in Illustration 14. For each joint sensor the transmission begins with CSN switching LOW output. Data is then output from the sensor with each falling edge of an inverted CLK signal provided by the IOIO as master. This is known as SPI mode 3, and various other configurations of clock polarity and reading phase are possible in the protocol. The data consists of joint angular position, D9-D0, followed by six status bits. The status bits highlight any errors in the joint encoder including mag strength, linearity, and chip readiness. A parity bit to ends the transmission to assist in detection of transmission errors. The transmission finishes when CSN is made HIGH, the next transmission can begin within 0.5µs with CSN being dropped LOW.

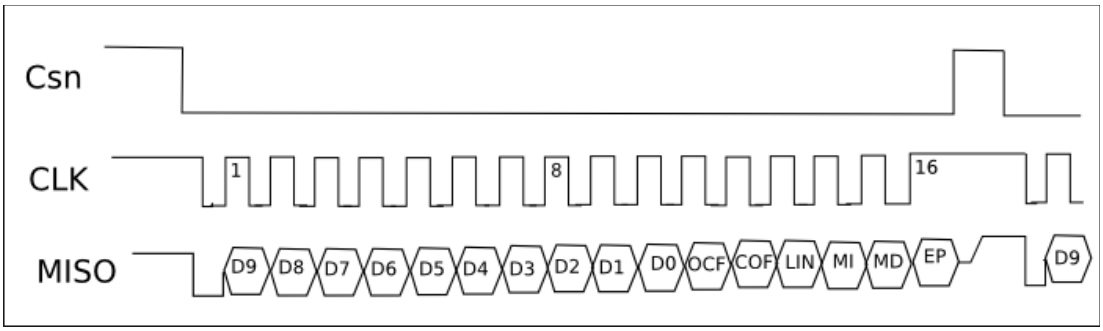


Illustration 14: Voltage signals along the Chip Select, Clk and MISO lines for the Joint Encoder SPI ("AS5040 Datasheet," 2009, p. 33)

4.6.2.3 Software Design

For this interface to work, two software nodes are utilised, one to read the Joint Sensors on the Master Arm, and the other to apply position control to the Slave Arm. For the master arm the software runs a simple setup routine and then engages in a loop. In the setup, the software reads the sensors, to determine the zero position for the Master Arm. Therefore, the Arm must be positioned at start time to the zero position of the Slave Arm. The software then follows its loop of:

1. reading its sensors,
2. applying the zero position to get angle change from a starting pose, and
3. publishing these positions to a joint controller as reference positions.

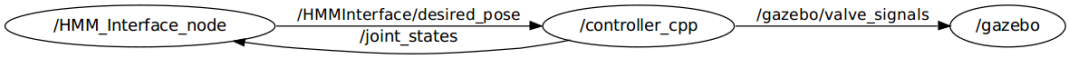


Illustration 15: Software nodes of the Master Arm Interface. The /gazebo node stands in for the /Slave_node. Note that the joint state feedback for the controller node is done by a Service call and therefore not shown.

The Joint Controller */controller_cpp* subscribes to these reference positions and also to Joint Data from the Slave Arm. It applies PID control and publishes valve commands to the slave_node, controlling the actuators as shown in Illustration 15. This forms a non-realtime control loop and is typically run on the same computer as the slave_node in order to avoid connection issues.

4.6.3 Joystick Interface

4.6.3.1 Mechanical Design Considerations

The Joystick Interface utilised a number of twin axis thumb joysticks to receive operator input. The Thumb Joystick is utilised on a number of commercial interfaces, predominately game controllers. They are spring loaded, always returning to centre, can be manipulated by one finger, giving the possibility of manipulating more than one thumb joystick with one hand. The spring also gives them a good 'feel' so that even if there is no direct feedback, the operator still has a sense when the joystick is close to its limits. The Joysticks are small and light, making them suitable for portable interfaces and also to mount in a variety of configurations. This is important so that the joystick is aligned to the task-space of the user.

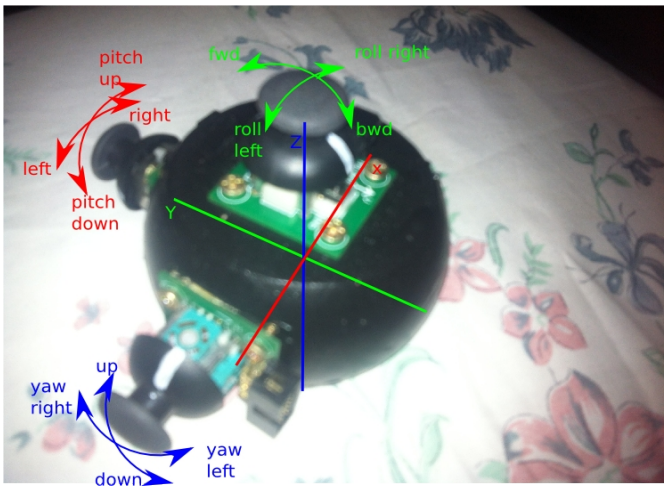


Illustration 16: Joystick Interface Mk I, three thumb joysticks are mounted orthogonally in the same direction as the task space axes, giving control over the six degrees of freedom.

Two options were tested for aligning the joysticks with the task axes. Remembering that the six task degrees of freedom are left/right, forward/backward, up/down, roll, pitch and yaw. They represent the translational and rotational movement about three axes x, y and z. The first option aligned three joysticks orthogonally to each other around a central mount. Each joystick axis therefore represented either a translation or a rotation about the axis the joystick was aligned to as shown in Illustration 16. The three joysticks were designed to be operated by one hand, leaving the other hand free for other tasks. This arrangement had poor ergonomics, in particular due to the limitations of reach of an operators hand. Also, it didn't equally divide the workload between the available hands, making a five fingered hand control six axes!

The second option divided the workload between both hands. The left hand controlled orientation, and the right hand controlled position. To achieve this, four thumb joysticks, two for position and two for orientation, were used to replicate the six task degrees of freedom. This gave four axes for three position axes, and four axes for the three orientation axes. The joystick pairs were mounted orthogonally to each other and the joystick axes controlled the aligned position or orientation axis. It allowed the operator to directly actuate his task-space intentions on the interface. It also divided the workload, allowing control to be achieved by only two fingers on each hand leaving the others to push buttons and support the interface. The interface had similar ergonomics as a game controller, except that it has four joysticks instead of two. The interface is shown in Illustration 17.

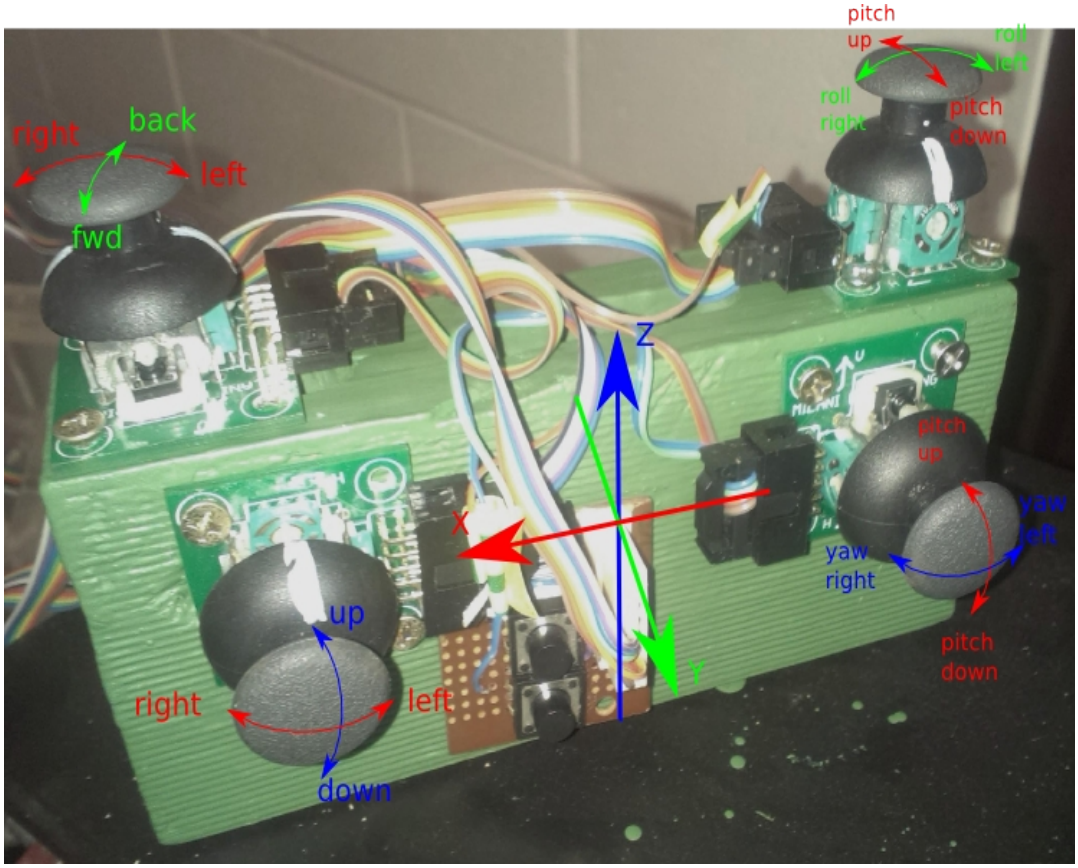


Illustration 17: Joystick Interface Mk II, arranged to be used with two hands, the right hand controls position and left hand controls orientation.

4.6.3.2 Electrical Design Considerations

The electrical design is relatively simple with the thumb joysticks forming a linear voltage divider, read by the Analog to Digital Converters (ADC) on the IOIO. The IOIO has sixteen ADCs available and six are reqired to read the Thumb Joystick position. The joysticks are supplied with the ADC ref voltage of 3.3V.

4.6.3.3 Software Design

As the software is required to conduct the transformations between the task-space input and the joint-space control of the machine. The preferred software arrangement splits the hardware interfacing and the transformation algorithm into separate nodes. This allows the same transformation algorithm to be used with a variety of interfaces that publish the operator's task-space intentions.

The */Joystick_node* does the hardware interfacing and runs a setup routine followed by a loop. During the setup routine, the thumb joysticks remain in a the neutral position so that this reference position may be read and recorded to become the zero point for the joystick axis. In the loop, the thumb joysticks are read and the zero point subtracted from them to get a signed value indicating a positive or negative direction in the task-space axis. The range of joystick movement was broken up into three bands representing a Deadband, Fine and Coarse movement. This minimised the occurrence of drift in the results and made the input more robust. The difference

in gain from fine to coarse movement was a factor of five. The measurements of joystick position were then published to the */HMM_Cartesian_Interface_node* over the */HMMInterface/dtask_space* topic.

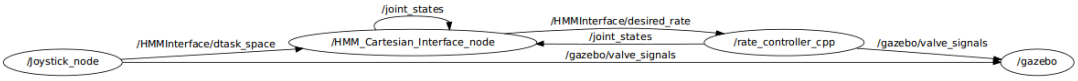


Illustration 18: Software node arrangement of the Joystick Interface. The /gazebo node replicates the /Slave_node. The joint feedback is carried out via a Service call and therefore not shown.

The */HMM_Cartesian_Interface_node* calculated the jacobian and dealt with the redundancy as shown in Section 4.3. The node subscribed to the */joint_states* msg that was being published by the */rate_controller_cpp*. The */joint_states* topic described the position, velocity and torque of each of the joints. This was used to update the kinematic model in the */HMM_Cartesian_Interface_node* to the same position state as the real arm, hence ensure the inverse jacobian algorithm was accurate for the current pose of the arm. The node calculated the joint changes as per the redundancy algorithm and published them to */rate_controller_cpp* which was an open loop velocity controller of the joint actuators. Rate_controller_cpp applied the joint change twitches as valve signals to the Slave Manipulator.

4.6.4 Sensor Interface

4.6.4.1 Mechanical Considerations

The sensor interface utilises an Android device equipped with a gravity and magnetic sensor which allows the measurement of all the orientation axes. The touchscreen provides three buttons for control of the input: a forward arrow, a backward arrow and a button labelled “Orientation” as shown in Illustration 19.

To achieve position movement, the operator aligns the arrows with the intended direction for movement and then presses the arrow that corresponds to the direction required. Software measures the orientation of the device and determines the positional changes due to a movement along the line of action indicated by the orientation of the device.

When the Orientation Button is pressed, any subsequent change in orientation of the device becomes the change in orientation task-space. This amount is proportional to that physical change in orientation. A gain is applied to improve the sensitivity of the input.

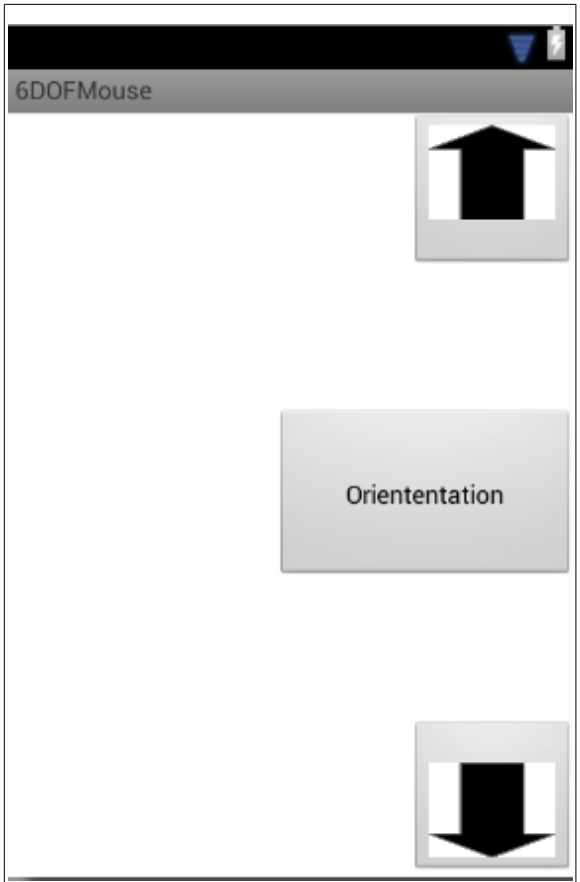


Illustration 19: Screenshot of Sensor Interface, showing forward, reverse button and orientation button

The only Android device that was available with the appropriate sensors was a 10inch tablet. This tablet was somewhat heavier than the 4 inch device that was desired. A 4 inch device could be supported in the palm of the operator's hand and easily manipulated in order to change orientation. An additional problem identified with this interface was that the Android User Interface (UI) does not record the pressing of multiple buttons at the same time. As a result, a physical Tool Open Button and Actuation Button were still required. It would be envisaged that the additional buttons would be integrated into the single device.

4.6.4.2 Electrical Considerations

As the interface is already packaged in an Android Device, only the Interface Status Light, Tool Open Button and Actuation Button were required and they were implemented as part of the common electrical arrangement.

4.6.4.3 Software Considerations

Whereas the software for previous interfaces ran on one main thread, this interface required two threads to share data. A user interface thread which measured sensor output and a ROS thread that handled ROS communications. The sensor packages were read each time a change was detected. Two sensor packages were read: The Gravity sensor and the Orientation sensor.

The Gravity sensor measured the gravitational acceleration along three orthogonal axes which correspond to the x, y and z directions. The directions are relative to the Android device, hence, if the device is held flat, then the readings would be x=0, y =0 and z=-9.8. The units of the sensor are m/s^2. Pitch forward increases the y value and roll to the right or left increases x value. These sensor values are equal to $g \sin\theta$, where θ is the angle around an axis. Hence there will be a sinusoidal output in the axes as the tablet is rotated at a constant rate. This means that the value from the sensor can be used directly and no further geometric functions are required.

The Orientation Sensor also measures yaw based on the relative position of magnetic north to the tablet, and the output from the gravity sensor ("Position Sensors," 2013). The units are in radians. The Sensor Interface establishes a zero direction of the Orientation Sensor on startup which is taken as the reference for the positive y-axis (forward face) of the Slave Manipulator. The direction the top of the Android device is facing at application initialisation becomes the positive y-axis for

the interface. The difference from this reference direction is then included in the amount of x and y task-space movement as per below (for a forward arrow push):

$$dx = \left(1 - \frac{g_y}{g}\right) \sin\left(mag_z - mag_{z0}\right)$$

$$dy = \left(1 - \frac{g_y}{g}\right) \cos\left(mag_z - mag_{z0}\right)$$

$$dz = \frac{g_y}{g}$$

Where g_y is the acceleration value along the y-axis, mag_z is the current yaw angle and mag_{z0} is the reference yaw angle when the application was first started. For determining task-space orientation change, the reading when the Orientation Button is pressed is stored. Task space orientation change becomes the change in orientation from that initial value.

$$droll = mag_{y2} - mag_{y1}$$
$$dpitch = mag_{x2} - mag_{x1}$$
$$dyaw = mag_{z2} - mag_{z1}$$

The subscript 1 refers to the reading when the Orientation Button is first pressed and subscript 2 refers to the sensor value at some time after with the Orientation Button pressed. The change in orientation, *droll*, *dpitch* and *dyaw* stop being evaluated once the Orientation Button is released and reset to new values.

4.6.4.3.1 ROS Thread.

Once the task-space directions have been evaluated, they are passed to */Sensor_node* in a separate CPU thread via a shared memory application object. Once transferred to the */Sensor_node* the operator's task-space control signals are published to the same */HMM_Cartesian_Interface_node* that is used in the Joystick Interface. The remainder of the software is as per the Joystick Interface.

4.7 Interface Testing and Results

In order to evaluate the current range of human machine interfaces (HMI) to allow tele-operation of the Heavy Maintenance Manipulator, testing was conducted to evaluate their effectiveness with the target operators. The three interfaces were the Master Arm Interface, Joystick Interface, and the Sensor Interface. These were tested on Mine Employees using two simulated maintenance tasks.

4.7.1 Aim

The aim of these experiments is to gather subjective and objective evidence regarding the suitability and ease of use of the three interfaces.

4.7.2 Method

4.7.2.1 Simulating Tasks with ROS and GAZEBO

Having a model that works within a Force-Dynamic simulation, which includes collision modelling, allowed some of the tasks of the arm to be examined. These tasks include: picking up a pin and inserting it into a double clevis, and rotating a Hoist Chain. These task-based simulations allowed the study of: the arm range of movement, the requirements for effective grasping, and the ability of having only a single arm to perform the required tasks.

4.7.2.1.1 Pin Place Task

One of the dragline rigging maintenance tasks that was focussed on was inserting a pin. This is a common task and was replicated by a simulation of inserting a drag pin into a hole in a vertical wall. This task demonstrated a number of key capabilities in the simulation: how an interface can accurately position and orientate the gripper, and how an interface can then align the gripped pin to the target hole. This task required a couple of extra models beyond that of the manipulator. These included a 6 inch pin, and a wall with a double clevis, which is similar in arrangement to a trunion mount on a dragline bucket. The starting layout is shown in Illustration 6.

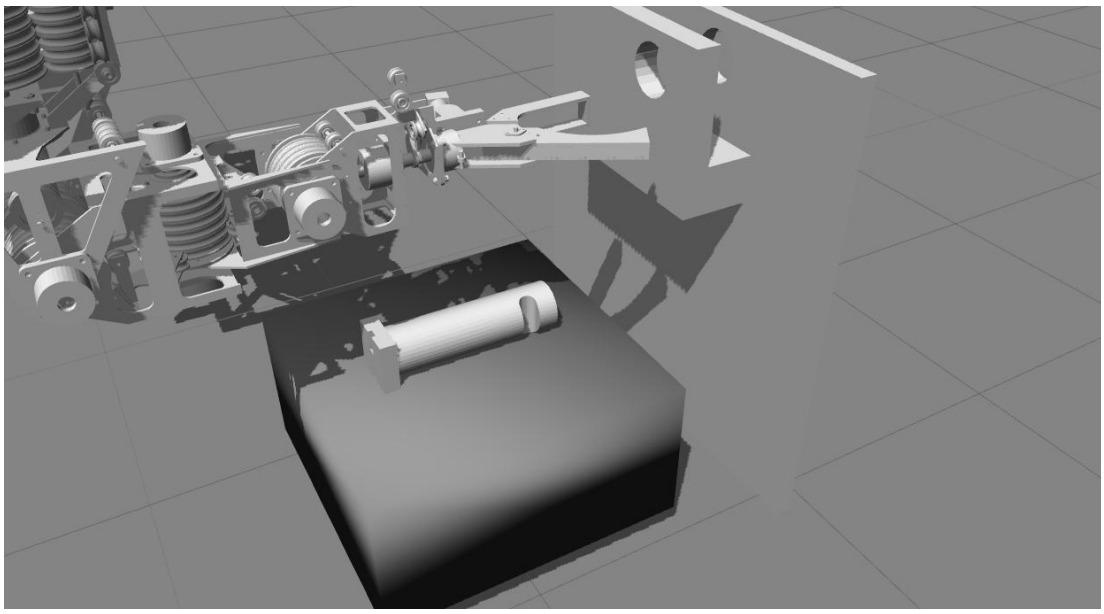


Illustration 20: Starting layout for a pin placement task showing a 6" pin on a platform ready to be inserted into the double clevis on the wall

4.7.2.1.2 Hoist Chain Rotation Task

This task attempted to replicate the rotation of a Hoist Chain which was being held in a double clevis. This required the chain to be removed from the clevis, rotated 180 degrees and reinserted into the clevis. The task tested the interfaces and grippers as before, but also tested to see if the manipulator has the range of movement capable to complete the task using just a single arm. The layout is shown in Illustration 7. An interesting point to note is in the simulation of chain.

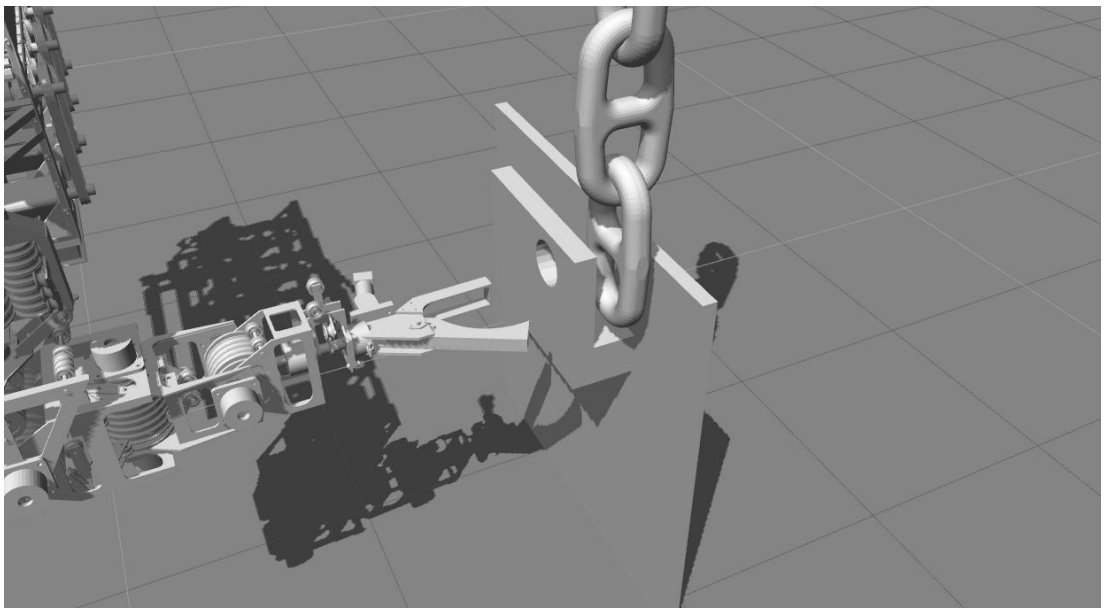


Illustration 21: Starting layout of Hoist Chain Rotation Task, showing manipulator, Trunion Wall, and suspended Hoist Chain.

Normal chain-links have six degrees of freedom, however to simplify simulation, and to improve simulation performance, the joints between the links were modelled with only two axes. This saves computational power and allows the simulation to run closer to real time. This gave the chain some of its more common movement, but not the full movement that would be observed by letting the chain fall in a heap. Illustration 22 shows the joint axes used in the simplified hoist chain model. In all links other than the top link, both the green and red axes are modelled as rotational joints. The top link rotates to the world via the blue axis.

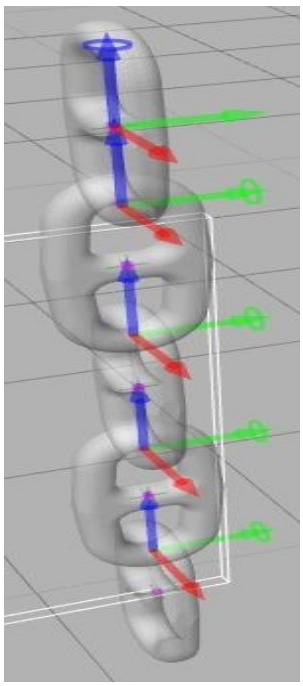


Illustration 22: Hoist Chain model showing joints

If the full six degree of freedom movement of chain-links relative to each other is desired, a hoist chain can be modelled without any joints but with self-collision on. This meant that the simulator dynamics engine would have to determine the movement of the links based on collision modelling. This successfully modelled the chain with behaviour that was very realistic but at a greater computational cost. The cost of modelling the links using collision modelling made the

simulation run about 250 times slower than the simplified version. The simulation ran at about 2% of real time, which was not suitable if being used as an environment for testing of operator interfaces.

4.7.2.1.3 Process

Each of the interfaces was applied to the two maintenance situations outlined above:

- the task to place a pin into a double clevis, and
- the task to remove and rotate a suspended hoist chain and place into the clevis.

The Joystick Interface during this testing was the Mk I joystick, with the Mk II being developed as a result of the test results. Operators were asked to complete the task and questioned for subjective information regarding the ease of use of the different interfaces. The comments regarding each interface were recorded after each test and a comparison between the interfaces was conducted using a questionnaire.

After the intitial round of testing was completed, the results were used to redesign and debug the interfaces and there was a second retest of the interfaces.

4.7.3 Results

The five individuals were made available for the initial testing of the interfaces. Their qualifications and experience are as follows:

Serial	**Operator Trade**	**Machinery Qualifications**	**Computer Literacy**
1	Fitter	Nil.	Nil
2	Boilermaker	Forklift, Overhead Crane, Telehandler	Game Consoles
3	Fitter	20ton Crane, Bucket Truck, Forklift	Minimal
4	Rigger Supervisor	20ton Crane, Elevated Work Platform, Grader	Office suite, GSAP
5	Fitter	20 Ton Crane.	Minimal

Comments Regarding the Interfaces

Interface:	Positives	Negatives
Master Arm	• Wasn't too bad to control.	• Could be improved by aligning the Master Arm to the

	• Read the signal and did what you wanted. • The whole concept is very good and would be useful • Interface was pretty responsive. • Gripper open button, could be a gripper close button. But having it as a gripper open button is safer. Probably having it as a toggle open, toggle closed would be more suitable. • Easy enough. • May help if the gripper button were on the master arm. • It did what you wanted it to do. • Working off your arm movements is the biggest positive. • Was able to complete the pin-place task. •	dominant hand. • The dead-man (actuation switch) could be improved by possibly being actuated by foot. • Didn't know where the limits were. • Delay – but can get used to it • Needs a grip to handle the master arm. • Delay, if it could be eliminated that would be handier. • Hand grip needs to be in place. • Had difficulty with Joint 5, getting it to move. • Getting the 5th Joint to move was difficult • Size of arm could be improved, something could attach to your arm. • Gripper could be redesigned, grab once and that's it. A Magnet would be a good alternate gripper or some sort of pincer • Clamping arrangement. Need a cradle and a ram setup for the tool. • Controls not to be too touchy, gradual control that you can react to.
Joystick Interface	• Doesn't have to be held in place • Would be good if you could get it to be responsive. • Park a tele-handler and use, and possibly remove from the cab to stand next to rigger for better communications. • The interface wasn't too bad in finding a solution it was just that I would like the ability to switch between the computer controlling joint space and the ability for the operator to	• Has a bit of a longer training liability • Interface is a bit tricky in having enough fingers to operate the joysticks. Could be fixed by having more thumbsticks for two hands. • Controls not responsive. • The joystick arrangement is awkward. • Sometimes the controls don't do what you want them to do [due to dynamics and joint limits]

	control joint space. • Joystick would be better if it did what you wanted it to do. • Felt more responsive over the amount of control operator felt he had.	• Hardest thing is reaching all the joysticks. • Allow the use of two hands and mount on block of wood with all joysticks in a straight line. • Too busy on one hand. Split it up with max 4 functions on once hand. Possibly a two handed interface with 3 functions each side, position in one hand and orientation in the other. • If controlling two arms, only control one at a time, with a switch between them • Joystick mounting ergonomics would be better if mounted side by side.
Android Interface	• Nil positives	• Not responsive, not doing what I want. • Dead man difficult to control, needed to be better integrated or with the foot. • Could not get it to rotate and assume the position. • No limits or restraints. • Wasn't responsive in doing what you wanted it to do. • Getting the hang of it is hard [unfamiliar industrial interface] • Touchscreen stopped receiving commands half-way through. • Controlling joints would be more beneficial • Difficult due to interface.

Overall Questions

The following questions were asked regarding the interfaces:

1. Which Interface was simplest to understand?
2. Which interface gave you more accurate control?
3. Which interface do you prefer and why?
4. Which interface do you least prefer.
5. If you had two features you would change on the preferred interface what

would they be?

Run	Answers to Questions.
1.	1. Master Arm because one hand is doing most of the action. 2. Joystick Interface if you could use two hands like a game controller. 3.Master arm because operator movements are followed by the slave arm. Use of body movements is beneficial 4. Sensor Interface because the size of the tablet makes it hard to use, better if smaller.
2.	1. Master Arm, or if inverse kinematics were working then the Joystick as it is familiar to Tele-handler Operator. 2. Master Arm 3. Master Arm because it does what you want.
3.	1. Joystick Interface was easiest to use and most practical, you can take it with you for greater versatility 2. Master Arm but Joystick Interface is better if it stayed still [note at this time there was at time drift in the response of the signal. 3. Joystick Interface preferred for ease of use. 4. Sensor Interface due to delay in response. 5.Would change the joystick arrangement, such as lining elements up in a row if possible.
4.	1. Master Arm [note this run was successful in placing pin in hole] 2. Master Arm because it followed what you were doing in the physical space. 3. Master Arm preferred as it was much easier to control. Maybe the Joystick Interface if there may be joysticks split between task space and and Joint-space control. 4. Sensor Interface – did not have a sense of movement or sense of feel. 5. Master Arm, handle closer to the movement of the arm.
5.	1. Joystick Interface if it worked 2. Joystick Interface – difficulties with the simulation 3. Joystick Interface – more sensitive to movement 4. Sensor Interface, wasn't really responsive, a bit all over the place. 5. Have a control of speed with the Joystick Interface. [NB as the djoint is normalised, the speed correction in the joystick is removed.]

Illustration 23: Graph of Orientation response to Joystick Interface. Note that an input of the Joystick could affect one or two pose changes, this is an incorrect response

4.7.4 Discussion

Overall, the Master Arm received the best feedback during the testing due to its accuracy in following the users input, albeit with some slight delay. The delay of about 270ms was distracting, but the majority of operators were able to adapt. Measurements attributed this delay to the Software interface between the IOIO and the program, giving the SPI messaging loop a frequency of 3- 4 Hz. This was a problem with the IOIO Java libraries or firmware and not in the software developed in this project.

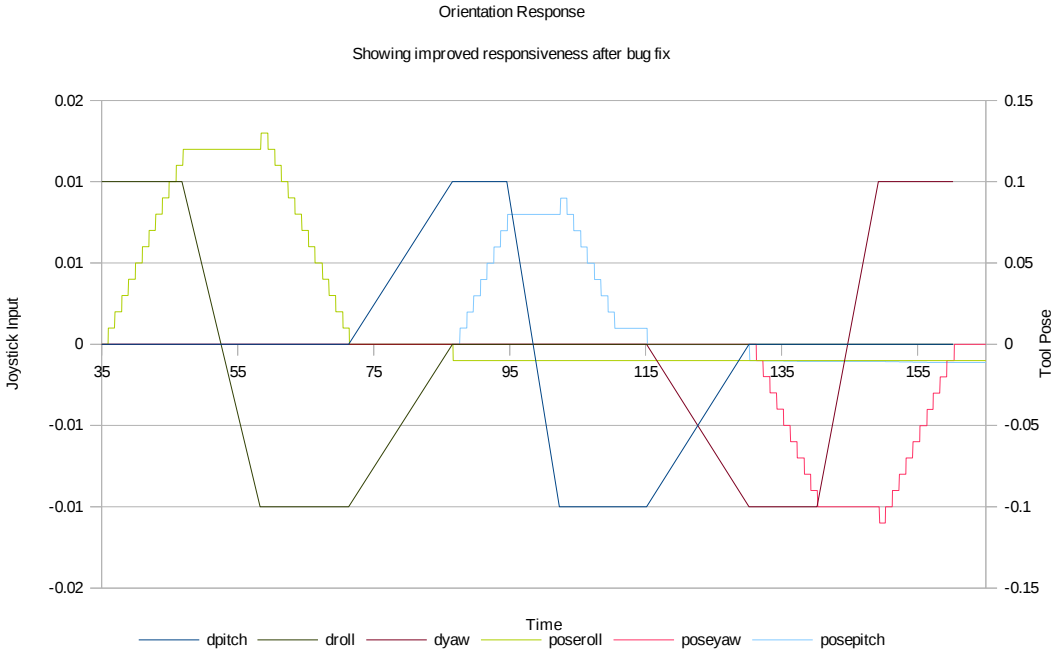


Illustration 24: Graph of Orientation response to Joystick Interface after bug fix, the output now shows a decoupling of output hence a more responsive interface

The Joystick Interface at the time of testing, had some problems with correctly evaluating the jacobian for orientation, resulting in coupling of the orientation movements as shown in Illustration 23. Note that position control was not affected. This was due to a bug in the code that was later rectified so that the operator's orientation requirement was correctly achieved by the system. The rectified output is shown in Illustration 24.

The ergonomic issues of the Joystick Interface Mk 1 were rectified to break up the workload between two hands in Mk II, as has previously been discussed. Due to these issues, the Master Arm Interface was the most preferred interface, primarily due to the relative accuracy of its task-space/joint-space transformation. Operators noted, however, if the Joystick Interface could be improved (as it subsequently was), it would be preferred to the Master Arm Interface, since its small size would be better suited to a field work profile. The small size could be operated from within a cab with minimal setup required, yet be more portable, allowing it to easily transition to outside operation. The Master Arm Interface, by contrast, took up significant space and would be cumbersome in a machine operator's cab. It would also be too unwieldy to be used in a portable capacity.

This was reflected in a second round of testing with operators, where the Joystick Interface was clearly preferred with some minor ergonomic changes relating to the positioning of the actuation and tool buttons. Also, there was a suggestion that some type of feedback be implemented to advise the operator if a joint limit were reached, as this became the main reason why the Slave Arm didn't follow the operator's

response. Feedback could be achieved through a vibrating motor.

The results for the second round of testing are not presented formally because only one operator was available for testing. That operator had previously preferred the Master Arm Interface in the initial round of testing. However, due to the Joystick Interface's improved responsiveness and ergonomic design this became the preferred interface. The Joystick Interface's small size, portability and rugged, comfortable form make it ideally suited to mobile field operations.

The Sensor Interface was consistently rejected by the experimental audience. Operators found it difficult to control the output of the interface, mainly, as it was an unfamiliar interface and took some experience to understand how it worked. The use of sensors as input within the interface was not immediately obvious like the joystick interface. This meant it was not intuitive to use as operators could not immediately understand how input was to be achieved.

4.7.5 Conclusions from Interface Testing

The Joystick Interface is the preferred method of control of the Arm, due to its intuitive control, compact size and general portability. The Master Arm interface was a close second, however the size of the interface would make it difficult to use in a field setting. The Sensor Interface was the least preferred because the actual method of input was not immediately apparent or familiar.

The response of the Joystick Interface was very accurate in replicating the operator's desired task-space movements and avoiding large joint movements due to singularities in the solution. There is no noticeable computational delay in determining the redundancy and calculating the joint twitches required. This method of determining the redundant joint is pragmatic but the computational requirements would exponentially increase with the number of redundant joints to be solved.

5 Conclusion

This dissertation details the development of an intuitive operator interface for a complex serial manipulator, to be used in heavy maintenance tasks. This interface allows the operator to control the manipulator in the 'task-space', with software handling the conversion to 'joint-space'. Testing of the interfaces shows operator task-space control to be most effective in reducing operator workload and improving the ease of use of a complex machine. These methods are applicable in concept, to a wider range of manipulators and other machines.

A number of operator interfaces were developed: a Joystick Interface, a Master Arm interface and a 6-D Mouse Interface. The Joystick Interface made use of a task space to joint space transformation implemented in software. The Master Arm utilised a scale model to conduct the transformation. Finally, a 3D mouse Interface utilised sensors in an Android Device with a software based task to joint space transformation. These interfaces were tested and the Joystick Interface proved most suitable according to the operator's subjective opinion. Quantitative measurement also showed that it accurately reproduced the operator's commands.

The software transformation developed for the Joystick and 6-D Mouse interfaces utilised the Jacobian Matrix to complete the task-space to joint-space conversion. However, since the manipulator contained a redundant joint, an additional algorithm was required to handle the redundancy. This additional algorithm also improved manipulator safety, as it navigated the arm away from singularities which could result in large joint movement. The novelty of this algorithm is based on its pragmatic approach, and could be modified to achieve a number of safety or performance goals.

The control strategy centred on the operator specifying commands to the arm in the frame of the task. The developed algorithm enabled the control strategy by ensuring that viable solutions for joint velocity could be found in a manipulator that has redundant joints. Furthermore, this algorithm utilised a cost function that minimised the chances of large joint movements due to singularities, improving the safety of the device.

Overall, the project has delivered a viable operator interface for controlling a complex, redundant manipulator. This interface was tested against a number of alternate operator interfaces. The contrasting results of the strengths and weaknesses of various interfaces meant that a number of key insights were gained, and a pragmatic approach to redundancy management was developed.

6 Bibliography

Forklift Training. *Crown Forklifts* (2013). at <http://www.crown.com/au/train/forklift_training.html>

Gaussian Elimination. *Wikipedia* (2013). at <http://en.wikipedia.org/wiki/Gaussian_elimination>

Handling Keyboard Input to Update a Callback. *openscenegraph.org* (2012). at <www.openscenegraph.org/projects/osg/wiki/Support/Tutorials/BasicKeyboardInput>

E. Guizzo & E. Ackerman. How Rethink Robotics Built its new Baxter Robot Worker. *IEEE Spectrum (2012).* at <spectrum.ieee.org/robotics/industrial-robots/rethink-robotics-baxter-robot-factory-worker>

Merritt, H. E. *Hydraulic Control Systems*. (John Wiley and Sons, 1967).

Industrial, ROS. *ROS (2013)* . at <www.ros.org/wiki/Industrial>

S. Lorenzo & M. Shaker. Integral Sliding Mode as a New Method to Control the Industrial Drives. *Record of the 1998 IEEE Conference* (1988).

Jacobian Matrix and Determinant. *Wikipedia* (2013). at <http://en.wikipedia.org/wiki/Jacobian_matrix_and_determinant>

H. Wang, W. Chen, Y. Lei & Y. Shouqian. Kinematic Analysis and Simulation of a 7 DOF Cable Driven Manipulator. *IEEE International Conference on Control and Automation* (2007).

H. Wang, W. Chen, Y. Lei & Y. Shouqian. Kinematic Control for a 7DOF Cable Driven Anthropomorphic Arm. *IEEE International Conference on Intelligent Robots and Systems* (2006).

H. Seraji, M.K. Long & T.S. Lee. Motion Control of 7 DOF Arms: The Configuration Control Approach. *IEEE Transactions on Robotics and Automation* 9**,** (1993).

Overview. *Willow Garage* (2011). at <http://www.willowgarage.com/pages/software/overview>

S.M. LaValle. Planning Algorithms. *Cambridge University Press* (2006). at <http://planning.cs.uiuc.edu/node658.html>

Position Sensors. *Android API Guides* (2013). at <http://www.developer.android.com/guide/topics/sensors/sensor_position.html>

Proportional Valve Controller - Metal Housing. Electronic Controls 3.438.8. (2013). at <www.hydraforce.com>

Quantitative PID tuning procedures. (2013). at <www.iamechatronics.com/notes/78-lessons-in-instrumentation/466-quantitative-pid-tuning-procedures>

Roy Featherstone. *Rigid Body Dynamics Algorithms*. (Springer, 2008).

Scene Graph. *openscenegraph.org* (2012). at <www.openscenegraph.org/projects/osg/wiki/Support/KnowledgeBase/SceneGraph>

A. Hansson & M. Servin. Semi-autonomouse shared control of large-scale manipulator arms. *Control Engineering Practice* **18,** 1069–1076 (2010).

Shur Complement. *Wikipedia* (01 Sep 13). at <http://en.wikipedia.org/wiki/Schur_complement>

B. Ytai. SpiMaster.Java. (2011). at <https://github.com/ytai/ioio/blob/master/software/IOIOLib/src/ioio/lib/api/SpiMaster.java>

Billingsley, J. Teaching Control Fundamentals for Mechatronics and Robotics - the use of JavaScript for simulation and animation. *ACRA* (1999).

The Moore-Penrose Pseudo Inverse. *CalTech* (2013). at <http://robotics.caltech.edu/~jwb/courses/ME115/handouts/pseudo.pdf>

Training and Licencing. *Blueprint Forklift Training* (2013). at <http://blueprint.org.au/page/23/Training_and_Licensing>

Tuning PID Controls. (2013). at <www.efitechnology.com/tech_articles_PIDTuning.htm>

K. Hughes & X. Jaing. Using Discrete Event Simulation to Model Excavator Operator Performance. *Human Factors and Ergonomics in Manufacturing and Service Industries* **20,** 419 (2010).

Bengiamin, N. N. & Kauffman, B. Variable Structure Control. *IEEE Control Systems Magazine* (1984).

Welcome to OpenSceneGraph website. *openscenegraph.org* (2012). at <www.openscenegraph.org/projects/osg>

Appendix A Hydraulic Actuator Modelling

There are a number of published ways to model hydraulic systems, each varying in complexity and accuracy. For this application, the more complex models were chosen because they could relate fluid flow and pressure to actuator position, velocity and force as a function of valve spool displacement. In this section we will examine three models of Hydraulic Actuators and discuss their strengths and weaknesses.

A.1. Simple Models – no pressure term

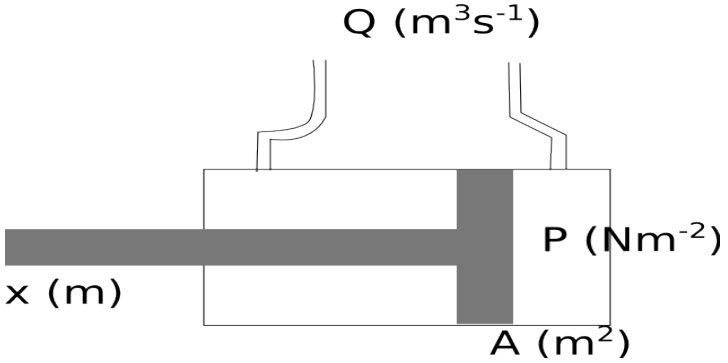


Illustration 25: Model actuator layout

In the simplest hydraulic models, actuating pressure (P_L) and force in a cylinder (F) is assumed to be constant and equal to the supply pressure of the fluid acting on the actuator's working area. The working area can be the area of a piston (A) in a cylinder. The relationship between actuating pressure and cylinder force is given below:

$$P_L = \frac{F}{A}$$

As pressure does not change, the dynamic relationship is purely between the flow rate of the hydraulic liquid (Q) and the resulting physical movement of the actuator. This is linear displacement (x) for a cylinder. The relationship is given by:

$$Q = A\dot{x}$$

for a cylinder where $\dot{x}$ is linear velocity("Engineering System Models," 14 Jul 13). The speed of the actuator is therefore proportional to the flow. Typically the flow is controlled by a valve, in our case a proportional valve as shown in Illustration 26 such that the flow is proportional to the position of the valve spool x_v related by a constant k_v .

$$Q = k_v x_v$$

All flow produces movement of the actuator due to the continuity of the system. Therefore flow out of the valve can be made equal to the the flow actuating the cylinder:

$$k_v x_v = Q = A\,\dot{x}$$

The cylinder's position acts as an integrator for the valve spool position. A step input in spool location would produce a ramp output of cylinder position, thus the actuator's velocity is proportional to the spool displacement.

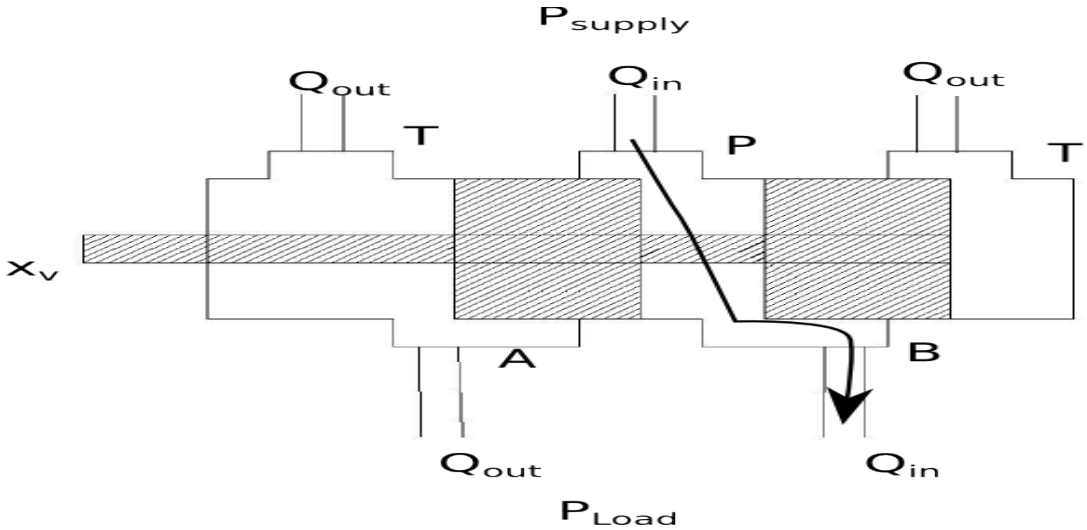


Illustration 26: A typical 3 position 4-Way valve with flow, pressure, Pump (P), Tank (T) and A and B ports marked

This model is suitable for lightly loaded conditions where the load on the cylinder is not going affect the pressure actuating the cylinder. In this case, the movement of the cylinder is going to be equal to the flow rate. This model starts to break down when the pressure due to the load on the cylinder starts to approach the supply pressure of the oil. This model also does not take into account leakage or the compressibility of the fluid.

A.1.1. Hydraulic Motors versus Hydraulic Cylinders

Before we move on, it is important to understand the difference between hydraulic motors and hydraulic cylinders in the simple case outlined in Section A.1. A motor relies on a nominal displacement, broadly equivalent to working area in a piston. Its units are $m^3 radian^{-1}$. Additionally the pressure is now proportional to the Torque (T) on the motor. The angular term θ replaces the linear term x. Therefore the relationship between pressure and torque is:

$$P_L = \frac{T}{Dm}$$

and the relationship between oil flow and angular movement is:

$$Q = D_m \dot{\theta}$$

The remaining relationships between oil flow and valve spool position remain unchanged.

Most hydraulic motors vary from cylinders in that they have a greater internal leakage, particularly in Gear Motors and Char-lynn motors. This manifests itself as a volumetric efficiency which becomes worse at low angular velocity. Volumetric efficiency affects the value of nominal displacement making it larger at low speeds. The value can be determined by examining the performance data published by motor manufacturers.

A.2. Simple Models – Pressure and Force

A modification of the above models attempts to incorporate the dynamic response of the cylinder load on the model. It would now be assumed that the valve spool controls pressure instead of metering oil:

$$P = k_v x_v$$

This model then equates the pressure on the cylinder and the force on the cylinder due to the mass (m) and damping (d) of the load such that:

$$PA = m\ddot{x}_c + d\dot{x}_c$$

where x_c is the linear movement of the cylinder with respect to time, $\ddot{x}_c$ is therefore acceleration of the cylinder, and $\dot{x}_c$ is the velocity of the cylinder. When equated a relationship is generated between the valve spool position and the position of the cylinder(“Engineering System Models,” 14 Jul 13):

$$k_v x_v A = m\ddot{x}_c + \dot{x}_c$$

This is a great equation for a mathematical study as the relationship between x_v x_c and *P* is linear. However there is a serious flaw in studying this equation is that the central assumption is wrong. Pressure is not proportional to valve spool position. The relationship will be derived in the next section.

A.3. Hydraulic Valve Dynamics

The relationship between flow (subject to load) Q_L , spool position x_v and pressure due to the load P_L is a non-linear one. The relationship has been derived from pressure-flow curves and it has been shown that flow is a function of spool position and load on the cylinder:

$$Q_L = Q_L(x_v P_L)$$ (Merritt, 1967, p. 81)

The relationship between these flows requires an examination of the flows within the valve and is based on the fundamental relationships in fluid dynamics, flow through a pipe or orifice.

A.3.1.1. Flow through a pipe

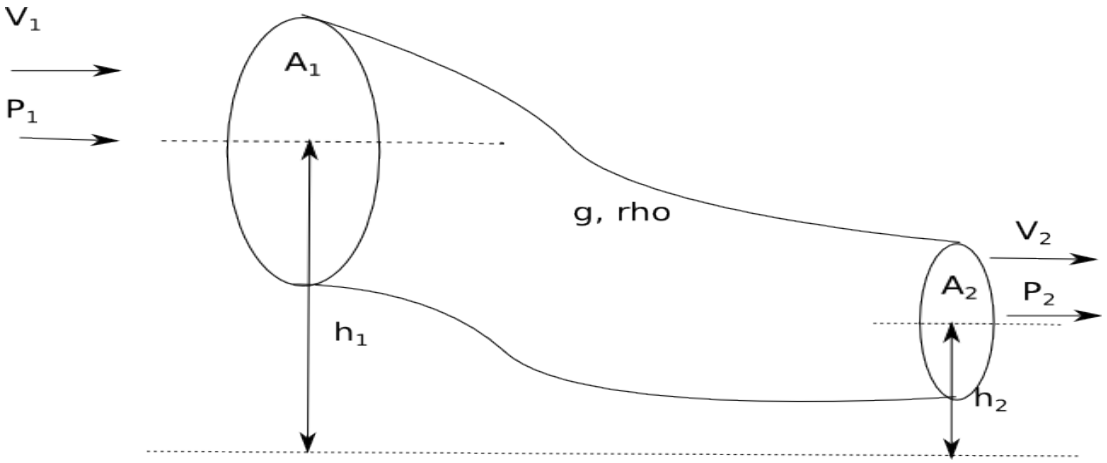


Illustration 27: Laminar incompressible flow through a pipe, showing all major parameters of the governing equations

The laminar, incompressible, inviscid flow through a horizontal pipe whose end conditions can be described by the fluid's pressure, speed, density, pipe cross-sectional area and relative height is given by Bernoulli's Equation:

$$P_1+\rho g h_1+\frac{1}{2}\rho v_1^2=P_2+\rho g h_2+\frac{1}{2}\rho v_2^2$$

which, if we assume the pipe is horizontal (ie $h_1=h_2$) can be simplified to:

$$P_1+\frac{1}{2}\rho v_1^2=P_2+\frac{1}{2}\rho v_2^2$$

As there is only flow through the orifices at both ends of the pipe the following Continuity Equation applies:

$$A_1 v_1=Q=A_2 v_2$$

these two equations can be combined and rearranged to give a pressure and orifice-shape dependent relationship for flow:

$$Q=A_2\sqrt{\frac{1}{1-(\frac{A_2}{A_1})^2}}\sqrt{\frac{2(P_1-P_2)}{\rho}}$$

$\sqrt{\frac{1}{1-(\frac{A_2}{A_1})^2}}$ is unchanging (providing the geometry is fixed) therefore is often treated as a metering coefficient C_d . Therefore the flow equation through an orifice is simplified to:

$$Q=A_2 C_d\sqrt{\frac{2(P_1-P_2)}{\rho}}$$

So it can be seen from this equation that the flow is non-linearly dependent on the pressures at both ends of the pipe. This is not considered in the simple hydraulic model, and is linearly dependent on the size of A_2 which if controlled or selected suitably in a valve could mean that flow is proportional to spool displacement x_v if $A_2 = wx_v$ where w is a constant. This is how spool position is handled in the simple hydraulic model. At no stage does it seem possible to make the spool displacement linearly proportional to pressure so the simple pressure model seems the least accurate. However, a typical 3-position, 4 way valve is a little more complicated than this, which we will examine in the next section.

A.3.1.2. Flows within a valve

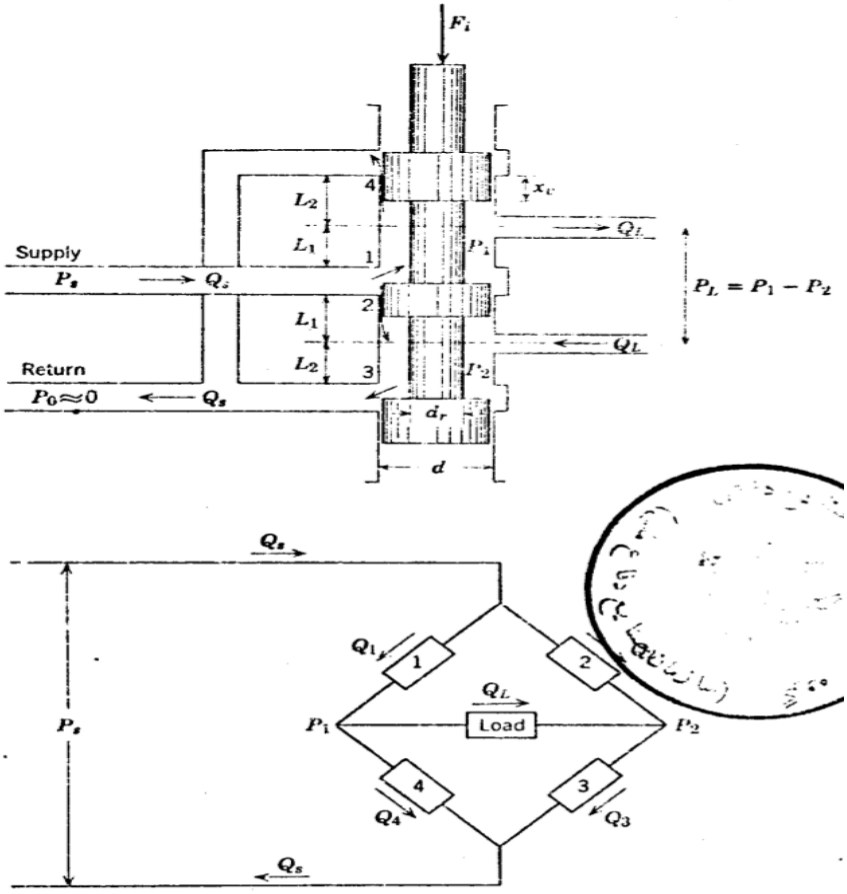


Illustration 28: 4 way, 3 position spool valve. The static flows are indicated by arrows. (Merritt, 1967, fig. 5–1)

Illustration 28 shows the flows within the valve in a static sense, by not including flows due to compressibility of the fluid. These are better handled when the valve is combined with an actuator. The main flows are Q_s, Q_L which are flows external to the valve, and Q_1, Q_2, Q_3, Q_4 which are internal flows, Q_1 and Q_3 are the intended paths for oil flow with Q_2 and Q_4 are flows due to leakage around the spool. The continuity of the flows actuating the load Q_L can be derived as:

$$Q_L = Q_1 - Q_4$$
$$Q_L = Q_3 - Q_2$$

The pressure drop P_L across the load is by definition:

$$P_L = P_1 - P_2$$

The flows 1 to 4 are all treated as flows through an orifice which was derived previously and are equal to:

$$Q_1 = C_d A_1 \sqrt{\frac{2(P_s - P_1)}{\rho}}$$
$$Q_2 = C_d A_2 \sqrt{\frac{2(P_s - P_2)}{\rho}}$$
$$Q_3 = C_d A_3 \sqrt{\frac{2(P_2)}{\rho}}$$
$$Q_4 = C_d A_4 \sqrt{\frac{2(P_1)}{\rho}}$$

Here the tank return pressure is usually 0 and can be neglected in Q_3 and Q_4 . If the return line is pressurised then it can be incorporated as being the pressure difference between the supply pressure and the tank pressure (Merritt, 1967, pp. 79–81).

As discussed in the previous section, the areas of the orifices can be functions of the valve spool position $A = A(x_v)$ for each of the orifices. All of these equation can be solved to determine the relationship between the load flow through the actuator Q_L and the pressure across the actuator P_L which are the key elements for determining what an actuator can push and at what speed.

A.3.1.3. Simplification – Valve symmetry and matching (Merritt, 1967, p. 82)

Determining the functions of the orifices with valve-spool position, can be simplified by design of the valve to ensure that the ports are matched, symmetrical and as linear as possible. Matching of the ports ensures that for a given displacement, the areas of forward and return flow are the same, that is:

$$A_1 = A_3$$
$$A_2 = A_4$$

Symmetry means that the area of the opposing orifices are the same for a given spool stroke from null. Meaning the area exposed is the same when the valve operates in a forward direction to the same when it operates the opposite direction:

$$A_1(x_v) = A_2(-x_v)$$
$$A_3(x_v) = A_4(-x_v)$$

If the valve is matched and symmetrical then all orifice areas are the same given a spool stroke in the actuating direction of the orifice (as opposed to when the orifice is acting as a leak flow).

Linearisation means that the valve slot is kept at a constant width throughout the spool stroke, ensuring that the Area of the orifice increases proportionally with spool stroke, such as:

$$A = A(x_v) = wx_v$$

If the valves are matched and symmetrical it can be shown that:

$$Q_1 = Q_3$$

$$Q_2 = Q_4$$

And by substituting the relevant flow equations into the above, and taking into account the matched and symmetrical nature of the orifices, it can be shown that:

$$P_2 = P_s - P_1$$

Incorporating the definition of load pressure drop $P_L = P_1 - P_2$, the relationship between load drop, supply pressure and pressure in the cylinder lines becomes:

$$P_1 = \frac{P_s + P_L}{2}$$

$$P_2 = \frac{P_s - P_L}{2}$$

Using the above relationships, we can then redefine the flows through the various orifices in terms of supply pressure and load pressure drop:

$$Q_1 = C_d A_1 \sqrt{\frac{(P_s - P_L)}{\rho}}$$

$$Q_4 = C_d A_4 \sqrt{\frac{2(P_s + P_L)}{\rho}}$$

and the flow through the valve that moves the load $Q_L = Q_1 - Q_4$ becomes:

$$Q_L = C_d A_1 \sqrt{\frac{(P_s - P_L)}{\rho}} - C_d A_4 \sqrt{\frac{(P_s + P_L)}{\rho}}$$

This demonstrates that the flow through the larger orifice A_1 minus the flow through the smaller orifice A_4 determines the flow that moves the load. The other load flow equation, $Q_L = Q_3 - Q_2$ can also be treated in a similar manner. These flows will both be equal providing the valve is matched and symmetrical.

Now we will consider further simplification specific to our 4 way, 3 position, closed centre valve that due to the closed centre nature of the valve, the leakage flows can be neglected as being very tiny(Merritt, 1967, p. 85), such that:

$$A1 = A4 = A2 = A3 = 0 \quad \text{, when} \quad x_v = 0$$

$$A1 = A3 = wx_v \quad , \quad A_4 = A_2 = 0 \quad \text{, when} \quad x_v > 0$$
$$A2 = A4 = wx_v \quad , \quad A_1 = A_3 = 0 \quad \text{, when} \quad x_v < 0$$

and consequently:

$$Q_L = C_d w x_v \sqrt{\frac{(P_s - P_L)}{\rho}} \quad \text{, when} \quad x_v > 0$$
$$Q_L = C_d w x_v \sqrt{\frac{(P_s + P_L)}{\rho}} \quad \text{, when} \quad x_v < 0$$

or combined using the sign of the valve spool displacement:

$$Q_L = C_d w x_v \sqrt{\frac{1}{\rho}\left(P_s - \frac{x_v}{|(x_v)|} P_L\right)}$$

This demonstrates the relationship between the flow going to the actuator versus the valve spool position and the pressure drop due to load in the actuator. It supports the simplification offered in the simple model, that flow is proportional to spool position, which is only valid when the pressure drop due to load is small. It does not support the claim that pressure is proportional to spool position highlighting that to be an inaccurate simplification.

A.3.2. Valve connected to Actuator combined (Merritt, 1967, pp. 145–148)

In the previous section we ignored transient compressibility due to the limited contained volume of fluid and negated the leakage internal to the valve due to the very small orifice through which leakage could occur. However, when considering the valve combined with an actuator, the volume of the fluid in the cylinder and pipes is significant and therefore has a transient compressibility. Also important is the leakage between the cylinders chambers, particularly if considering a gear motor as they typically have a high internal port to port flow. As a result, these factors must be incorporated into our valve-actuator model and are shown in Illustration 29.

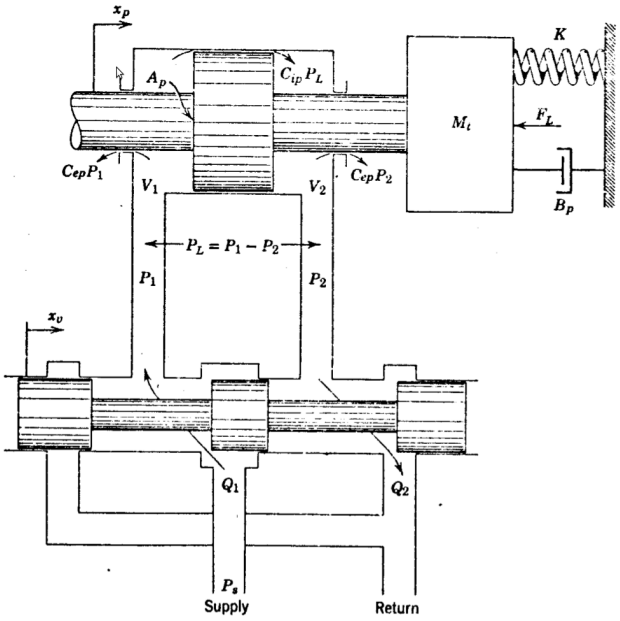


Illustration 29: Hydraulic valve with loaded actuator attached (Merritt, 1967, fig. 6–6)

The flow from the ports of the valve to the lines to the actuator has already been derived:

$$Q_L = C_d w x_v \sqrt{\frac{1}{\rho}\left(P_s - \frac{x_v}{|(x_v)|} P_L\right)}$$

which is rearranged to:

$$Q_L = C_d \sqrt{\left(\frac{P_s}{\rho}\right)} w x_v \sqrt{\left(1 - \frac{x_v}{|(x_v)|}\frac{P_L}{P_s}\right)}$$

This flow is accounted for in continuity as a flow due to compression of the fluid, Q_c , a leakage flow, Q_l and the flow to move the piston or work flow, Q_w . The compression flow is the change in volume due to the change in pressure :

$$Q_c = \frac{dV}{dt} = \frac{V_t}{4\beta}\frac{dP_L}{dt}$$

It is proportional to the change in pressure, by a constant related by the total volume V_t over the Bulk Modulus of Elasticity of a fluid β .

The leakage flow consists of leakage between the chambers, C_i and leakage lost external to the cylinder C_e both represented by a coefficient and proportional to the difference in chamber pressures:

$$Q_l = \left(C_i + \frac{C_e}{2}\right) P_L$$

which if the coefficients of leakage are combined into a coefficient of total leakage C_t lead to:

$$Q_l = C_t P_L$$

Finally, the flow that does work on the cylinder moves the cylinder piston of area A a certain speed $\dot{x}_p$:

$$Q_w = A\dot{x}_p$$

By combining these elements, such that Q_w equals the load flow minus the losses due to compressibility and leakage we get:

$$Q_w = Q_L - Q_l - Q_c \text{ , or rearranged}$$

$$Q_L = Q_w + Q_l + Q_c$$

This leads to our second equation defining the motion of a cylinder coupled with a valve:

$$Q_L = A\dot{x}_p + C_t P_L + \frac{V_t}{4\beta}\frac{dP_L}{dt}$$

The final equation of interest is Newtons second law which will describe what external loads cause a pressure drop across the cylinder chambers. In a mechanical system of a mass, spring and damper this could be related as:

$$P_L A = m\ddot{x}_p + d\dot{x}_p + k x_p + F_L$$

It is worth noting that for clarity focus is on an actuator being a cylinder as it is easier to visualise. However these relationships remain applicable to hydraulic motors, with the modification that D_m replace A and that the angular variable θ_m replace x_p .

A.3.3. Complete Hydraulic Cylinder Model.

Utilising the three equations of dynamics with respect to a hydraulic cylinder with valve dynamics, we will rearrange into a state-space representation describing the movement of the cylinder piston with respect to the valve spool movement. This relationship takes into account the loading on the cylinder which has been shown earlier to be:

$$Q_L = C_d\sqrt{\left(\frac{P_s}{\rho}\right)}wx_v\sqrt{\left(1-\frac{x_v}{|(x_v)|}\frac{P_L}{P_s}\right)}$$

$$Q_L = A\dot{x}_p + C_t P_L + \frac{V_t}{4\beta}\frac{dP_L}{dt}$$

$$P_L A = m\ddot{x}_p + d\dot{x}_p + k x_p + F_L$$

Firstly the cylinder piston has a velocity. The velocity is equal to the first derivative of position, hence we will choose our first two states being piston position and velocity:

$$\dot{x}_p = v_p$$

such that the last equation becomes:

$$P_L A = m\dot{v}_p + d v_p + k x_p + F_L$$

which when rearranged so that the $\dot{v}_p$ is to the left of the equation:

$$\dot{v}_p = \frac{A}{m} P_L - \frac{d}{m} v_p - \frac{k}{m} x_p + \frac{F_L}{m}$$

This gives the three states, P_L , v_p , x_p , plus a non-linear term $\frac{F_L}{m}$. Now solving for Q_L , eliminating that term to determine $\dot{P}_L$:

$$A\,\dot{x}_p + C_t P_L + \frac{V_t}{4\beta}\frac{dP_L}{dt} = C_d \sqrt{\left(\frac{P_s}{\rho}\right)} w x_v \sqrt{\left(1 - \frac{x_v}{|(x_v)|}\frac{P_L}{P_s}\right)}$$

The relationship equating $\dot{P}_L$:

$$\dot{P}_L = \frac{-4\beta A}{V_t} v_p - \frac{4\beta C_t}{V_t} P_L + \frac{4\beta}{V_t} C_d \sqrt{\left(\frac{P_s}{\rho}\right)} w x_v \sqrt{\left(1 - \frac{x_v}{|(x_v)|}\frac{P_L}{P_s}\right)}$$

Here the change in pressure drop is linearly dependent on piston velocity and leakage while accounting for the relationship with the valve spool position, x_v our system input. Our flow into the system is also affected by the cylinder internal pressure resisting the supply pressure.

Having derived the necessary state space equations our goal is not to linearise the equations in order to apply linear control theory, but to simulate this system as it has been derived, then use linear and non-linear control techniques to achieve position control of the system. Our governing equations are listed below:

$$\dot{x}_p = v_p$$

$$\dot{v}_p = \frac{A}{m} P_L - \frac{d}{m} v_p - \frac{k}{m} x_p + \frac{F_L}{m}$$

$$\dot{P}_L = \frac{-4\beta A}{V_t} v_p - \frac{4\beta C_t}{V_t} P_L + \frac{4\beta}{V_t} C_d \sqrt{\left(\frac{P_s}{\rho}\right)} w x_v \sqrt{\left(1 - \frac{x_v}{|(x_v)|}\frac{P_L}{P_s}\right)}$$

These dynamic equations and state variables will be used to facilitate simulation in the next section.

A.4. Conclusion

In this section three models of a hydraulic system that can be used for simulation have been examined. The most simple hydraulic system, is applicable if the loads on the actuator are small with the assumption that the supply pressure will not be challenged by the loads on the cylinder. The second model used an overly simplistic approach to dealing with variations in pressure, where the relationship between the valve stroke and the pressure in the cylinder was not accurate and not supported by the physics. Finally, the third model contained some approximations, but nevertheless remained consistent to the governing relationships within fluid dynamics. It successfully related cylinder pressure, position and velocity to the movement of the valve spool. These are the definitive factors to be included if studying models of hydraulic systems is to have relevance.

Appendix B Simulation Results of Hydraulic Actuators and control systems.

B.1. Discrete simulation of linear and non-linear models

This section discusses the simulation of the different models. The method of simulation relies on Euler integration of the state variables (Billingsley, 1999, p. 11). This method consists of estimating the next state value based on the expected change in that value over the simulation time period, that is:

$$x_k = x_{k-1} + \dot{x}_{k-1} * dt$$

The change in state is derived directly from the state equations, furthermore this presents an elegance that lifts the restriction on non-linear relationships in the state-space. Our non-linear terms derived in the previous chapter can be used directly to simulate the dynamic changes in the state variables of the cylinder. These state variables are cylinder displacement x_p , cylinder velocity v_p and the pressure difference between actuator chambers due to the load on the actuator P_L .

B.1.1. Method

To demonstrate the effectiveness of the models, the task of propelling a mass against gravity by means of a hydraulic cylinder was simulated. The arrangement is shown in Illustration 30. Three models were compared: the simple model, the pressure model and the full derived model. The state equations are shown in Table 1.

Each model is simulated with each run varying the amount of mass being supported. The Area of the piston is 0.002 m^2 and the supply pressure is 105 bar, giving a theoretical static maximum supported weight of 2142kg.

Table 1

Model	Description	State Equations
Simple Model	This model assumes that flow is a function of spool stroke and that pressure in the cylinder remains constant at supply pressure.	$P_L = P_s$ $\dot{x}_p = \frac{K}{A_p} x_v$
Pressure Model	This model assumes that Pressure in the cylinder is a function of spool stroke.	$P_L = K x_v$ $\dot{v}_p = P_L \frac{A_p}{m} - g$ $\dot{x}_p = v_p$
Fully Derived Model	This model assumes that flow is a function of spool stroke and pressure in cylinder.	$\dot{P}_L = \frac{-4\beta A}{V_t} v_p - \frac{4\beta C_t}{V_t} P_L + \frac{4\beta}{V_t} C_d \sqrt{(\frac{P_s}{\rho})} w x_v \sqrt{(1 - \frac{x_v}{\lvert(x_v)\rvert} \frac{P_L}{P_s})}$ $\dot{v_p} = \frac{A_p}{m} P_L - g$ $\dot{x}_p = v_p$

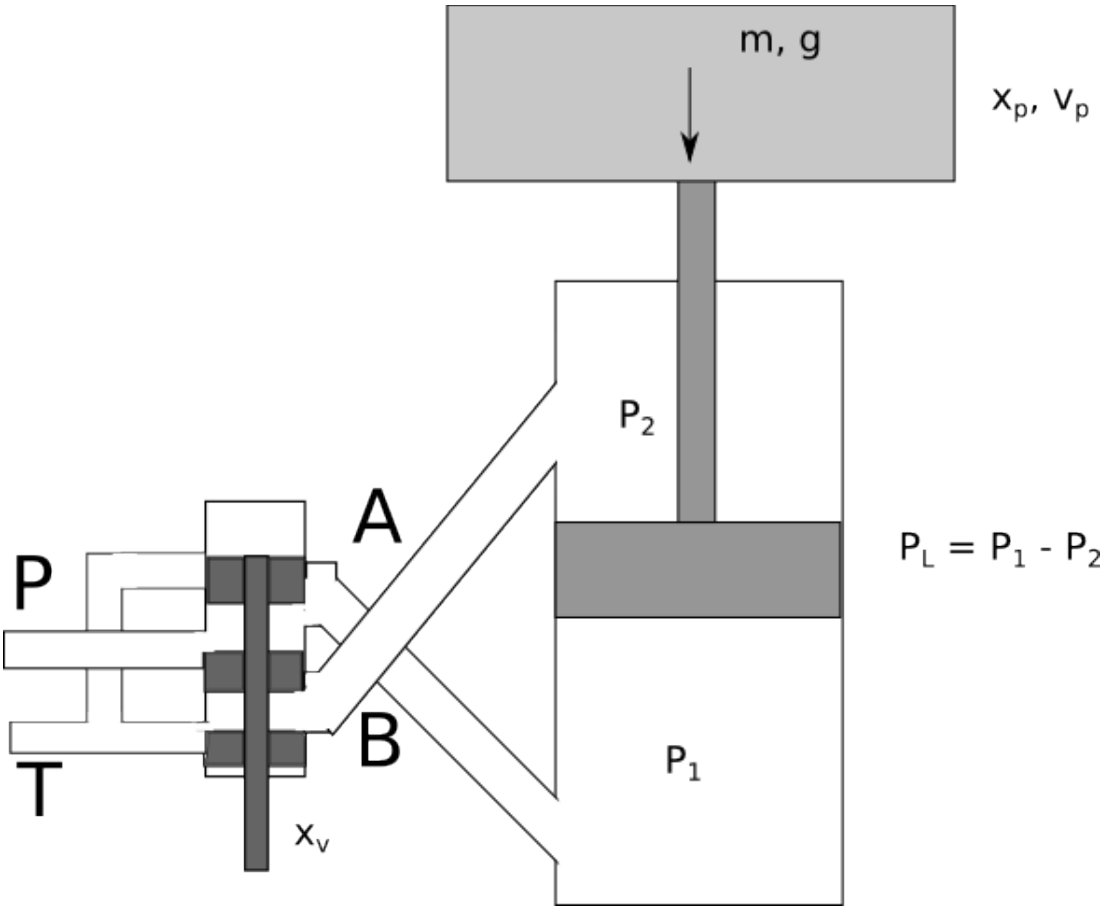


Illustration 30: Arrangement for comparison between Hydraulic actuator models

B.1.2. Results

A comparison between the different simulated models and their response to variations in the supported load is shown in Illustration 7.

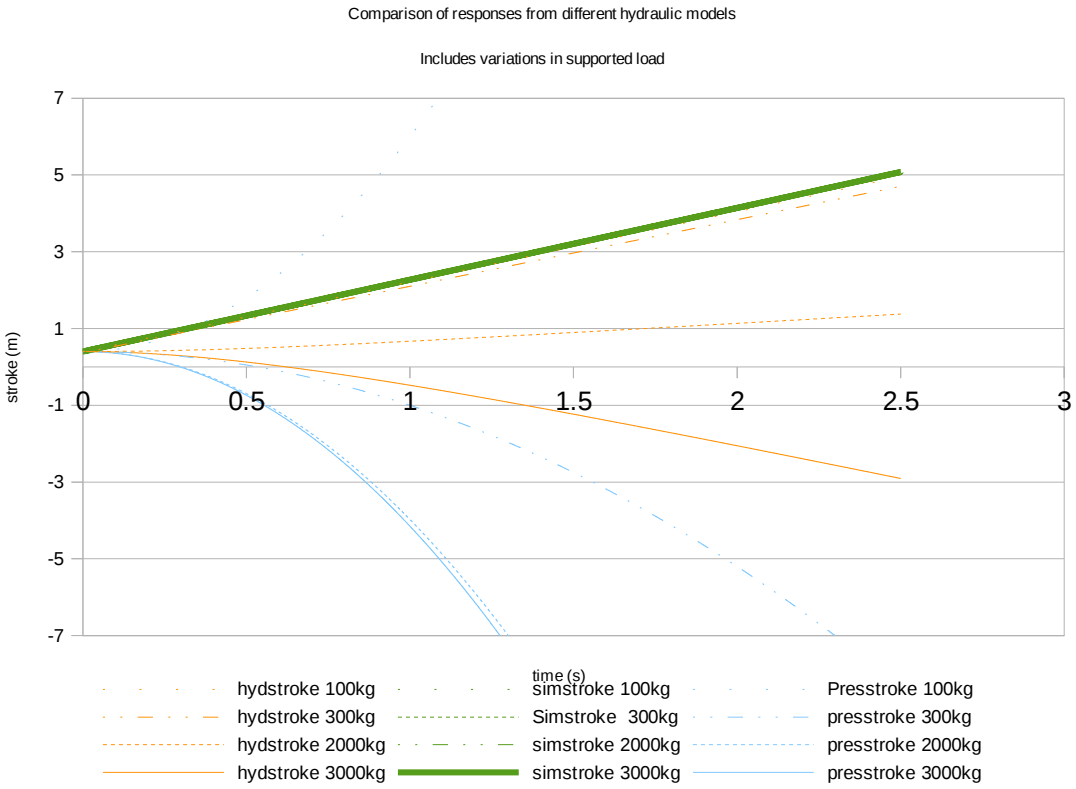


Illustration 31: Comparision of cylinder stroke response across different models and loads, for a step input.

B.1.3. Discussion

The Simple Model's (simstroke) response did not change as the modelled load was varied. All four simulation runs lie along the same line. It is worth noting that this line shows a strong correlation to the line of our Fully Derived Model (hydstroke) at low loading values.

The Pressure Model (presstroke) exhibits non-hydraulic behaviour, that is, it shows constant acceleration of the load in all cases, regardless of any ability of the value to supply the hydraulic fluid to support the subsequent cylinder movement. The Pressure Model would lead to an incorrect understanding of Hydraulic systems and highlights the dangers of choosing models which attempt to over-simplify a real system so that linear mathematics can be applied. Hydraulic valves and cylinder arrangements do not continuously accelerate their loads, typically they get to a top speed and stay at that speed for a given spool displacement.

This behaviour is shown by the fully derived model. In fact its response agrees with both the simple model for low loading conditions, and also show a realistic outcome for the condition of overloading. The condition of overloading occurs at 2142kg in a static sense. The simulation confirms that, with the 2000kg loaded condition showing a sluggish response, and the 3000kg showing an overloaded response where the fluid is pushed back against the intended direction.

B.1.4. Conclusion

The fully derived model, allows us to monitor the force driving the load as well as the speed of the stroke, both of which are affected by valve effects. The model recreates overloaded conditions and also models well the slightly loaded conditions. The simple model only has relevance in slightly loaded conditions and the pressure based model should not be used at all. It will be the preferred method for modelling hydraulics from now on. The derived model of an actuator and valve will be applied to an arrangement for changing a revolute axis.

B.2. Hydraulic cylinder joint servo simulation

It is common for hydraulically actuated manipulators to utilise a cylinder's stroke to vary the angle of a revolute joint. Typically these machines are not automated due to the difficulties in modelling and control as discussed in this section.

B.2.1. Geometric Arrangement

The geometric arrangement of the cylinder model is that of a three bar linkage with one of the bars having variable length. Two of the bars, *a* and *b* retain a fixed length, but the side opposite the joint has a length of $x+d$. Where x is the stroke and d the minimum length of the cylinder. The angle of the joint, θ is related to the cylinder stroke by the cosine rule:

$$(x+d)^2 = a^2 + b^2 - 2ab\cos(\theta)$$

hence the relationship between the cylinder stroke and the angle of the joint is non-linear due to the square relationship and the cosine.

The following values are applied in the analysis:

- a= 0.2 m,
- b= 0.4 m,
- d = 0.3 m;

additionally the stroke of the cylinder is limited to 0.3 m max.

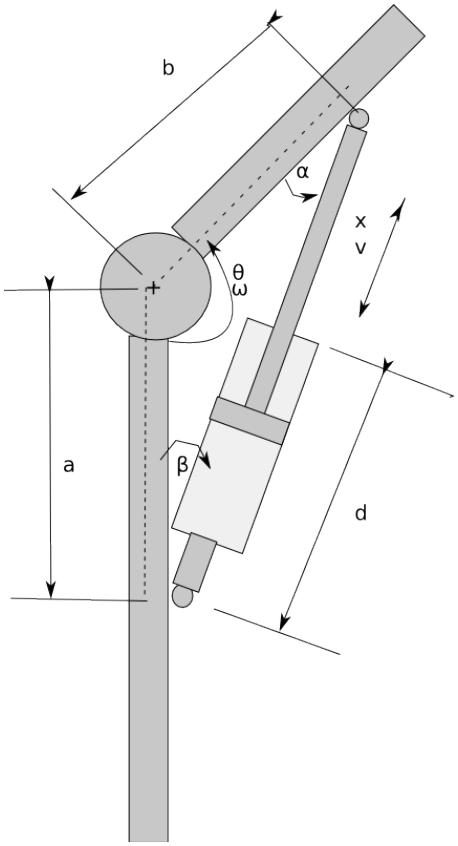


Illustration 32: Cylinder Joint Servo Arrangment

This means that the geometric relationship between stroke and joint angle will always remain within geometric possibilities. To ensure this is the case, Boundary conditions are applied to the stroke length such that:

$x_{max} = 0.3$, $v_{x=0.3} = 0$ and

$x_{min} = 0$, $v_{x=0} = 0$

When dealing with force in this example, a simplifications is that constant force is experienced regardless of the joint angle. However, if this simplification is not suitable, the actual derivation of force for this arrangement is now given.

To assess the effect of a load on a cylinder driven joint, the angular position and torques must be related to the force some way. Given the arrangement in Illustration 32, if a mass is located at the connection of the cylinder rod and link *b*, with gravity *g* acting straight down, a relationship between the torque due the weight of the mass and the mass rotational inertia can be developed that acts on the cylinder piston.

The torque by the cylinder on the joint is the product of the force of the cylinder by an effective lever arm *l* which is the perpendicular distance from the line of action of the cylinder to the centre of the joint. The relationship is:

$$l = a\sin\beta$$

Therefore the torque on the joint from the cylinder, T_1 is given by:

$$T_1 = AP_L l$$
$$T_1 = AP_L a\sin\beta$$

Similarly when incorporating the torque from the inertia of the mass T_2 the relationship between joint angular acceleration and linear acceleration of the cylinder must be determined. This is again dependent on the effective lever arm and given by:

$$\dot{\omega} = \dot{v}\, a\sin\beta$$

And incorporating the expression for angular inertia:

$$T_2 = mb^2\dot{\omega}$$
$$T_2 = mb^2\dot{v}\, a\sin\beta$$

The torque due to the gravitational acceleration of the mass, T_3, is defined, dependent entirely on the joint angle:

$$T_3 = mgb\sin\theta$$

The sum of all torques should be zero, but noting that T_3 and T_2 act against the cylinder extending (ie its positive axis).

$$0 = T_1 + T_2 + T_3$$
$$mb^2\dot{v}\, a\sin\beta = AP_L a\sin\beta - mgb\sin\theta$$
$$\dot{v} = \frac{AP_L a\sin\beta}{mb^2 a\sin\beta} - \frac{mgb\sin\theta}{mb^2 a\sin\beta}$$
$$\dot{v} = \frac{AP_L}{mb^2} - \frac{gsin\theta}{b\, a\sin\beta}$$

now $\sin\beta = 1-\cos\beta$ and $\cos\beta = \frac{b^2-(x+d)^2-a^2}{-2(x+d)a}$ so substituting to remove $a\sin\beta$:

$$\dot{v} = \frac{AP_L}{mb^2} - \frac{(gsin\theta)(-2(x+d)a)}{(ba)(-2(x+d)a - b^2 + (x+d)^2 + a^2)}$$

This replaces the third state equation for finding $\dot{v}$.

B.2.2. Results

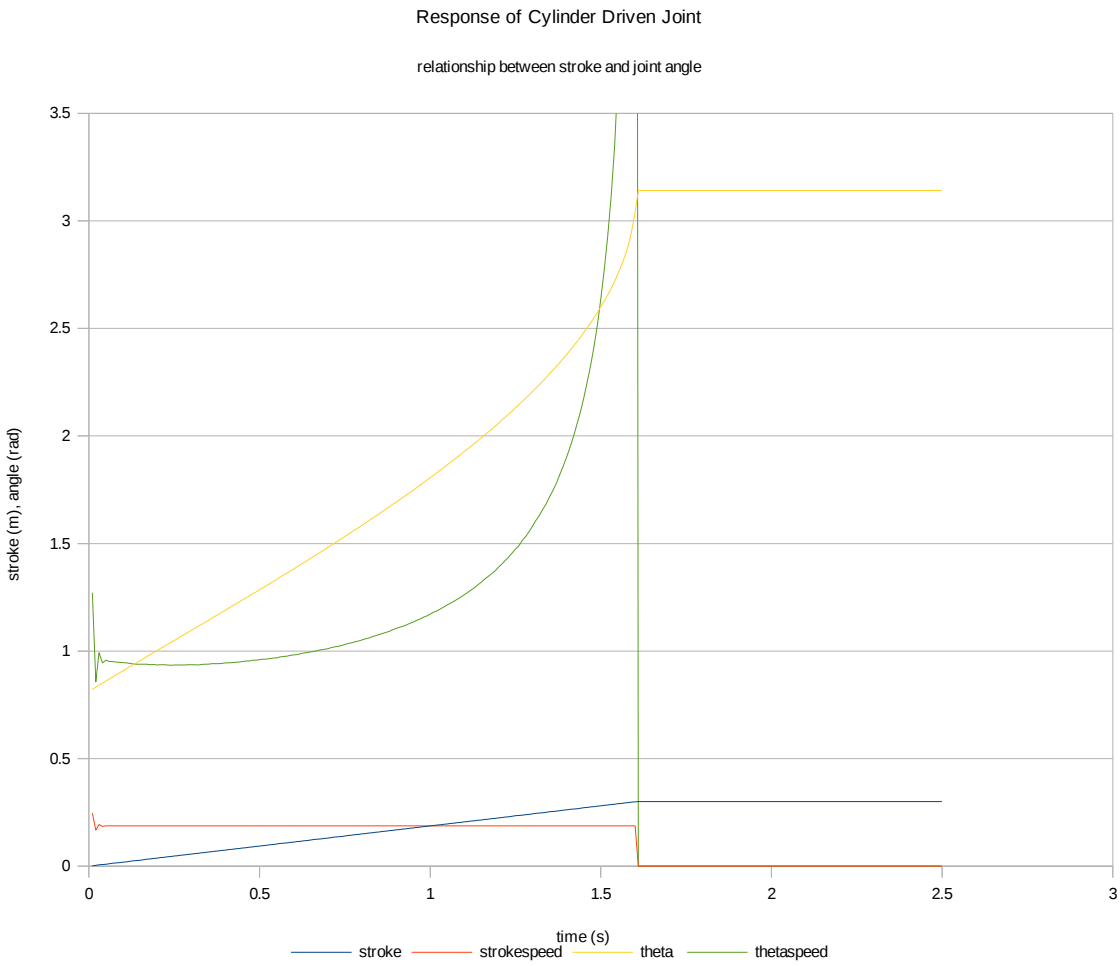


Illustration 33: Response to a step input of a cylinder driven joint

Illustration 33 shows the relationship between cylinder stroke and speed and the resulting angle created by the geometry. The valve setting is a step input. The max stroke in this case is 0.3m, at which point the cylinder stops moving, stroke and angle are held constant and velocity terms change to zero.

B.2.3. Discussion

The stroke response curves are the same as those in Illustration 31. The cylinder stroke response is fairly linear with time, as the cylinder is not overloaded. The joint angle response, however shows that the change in joint angle increases with stroke length. This effectively is a variable gain that is dependent on stroke length. Also angular velocity is accelerating with increasing stroke length. This is due to the non-linear geometric arrangement between stroke length and joint angle. This makes

sense as a small change in stroke length will make a large change in angle when the angle α is small.

B.3. Hydraulic motor joint servo simulation

In this section we will look at the effect of coupling a hydraulic motor to a revolute joint via a pulley or gearbox.

B.3.1. Arrangement

The hydraulic motor is coupled via a gearbox, to the joint. The arrangement of the joint is shown in Illustration 34. Now disregarding the issues of backlash, and assuming all components are rigid, the position and velocity output of the motor should be linear with the position and velocity output of the joint.

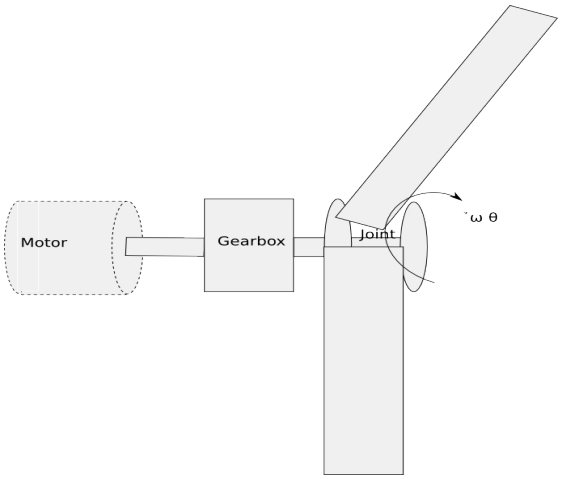


Illustration 34: Hydraulic motor driven joint

The motor angular position and velocity are θ_m, ω_m and are related to joint angular position and velocity θ_j, ω_j by the gear ratio of the gearbox by a proportional amount. This is reflected in the simulated response of the system. The loading of the joint is purely inertia with a constant element representing rolling friction of the bearing. However it is much easier to transfer joint loading to the motor as both are angular in nature with the conversion simply a proportional relationship governed by the gear ratio.

B.3.2. Response

Illustration 35 shows the response curves for a motor driven joint, and it clearly shows the linear relationship between the actuator states and the joint states. This is worth noting because motor torque is also linearly dependent on the joint torque. This was of particular use when modelling joint torques in the simulations.

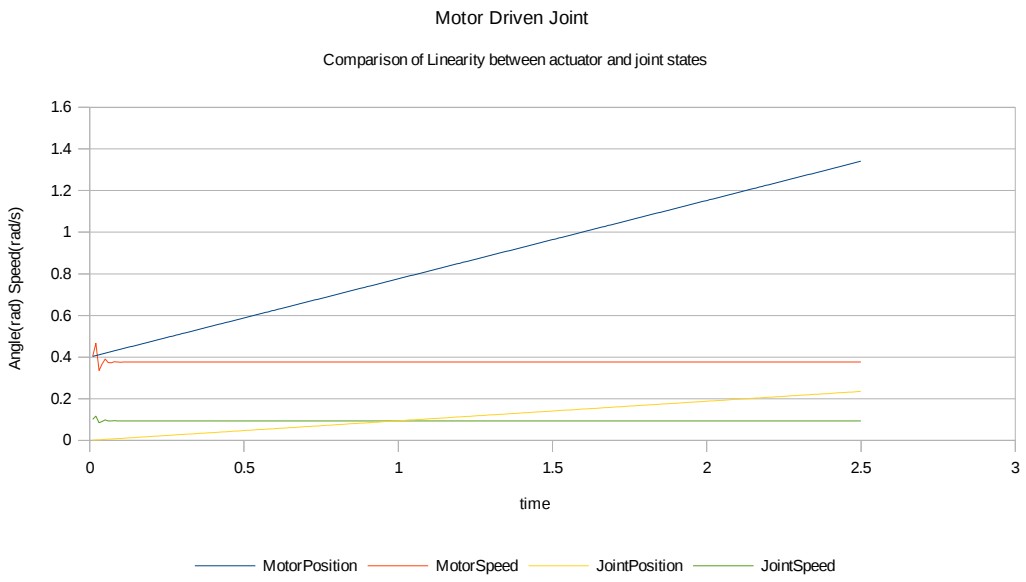


Illustration 35: Response to a step input of a motor driven joint. Gear ratio of 4:1

B.3.3. Discussion

This result is expected as the mathematics show a linear relationship between motor states and joint states in a motor driven joint. However in industry, the cylinder driven joint is most common in many manipulator tasks, from HIAB cranes to earth excavators and backhoes. These applications require positioning of joints, and whilst they are open loop controlled it is the joint position which is important not the cylinder stroke. This task is handled by human operators who normally require training to ensure safe operation. If closed loop control is required, then the non-linear relationship between the cylinder stroke and the joint angle will need to be handled.

In a motor driven joint, the joint's angle is linearly dependent on the angle of the actuator. Joint torque is linearly dependent on motor torque. This potentially means that the control of the joint could be more easily simulated and more easily controlled. This will be apparent when we examine the manipulator simulation.

B.4. Hydraulic Actuator Control

This section will examine a variety of control regimes on our cylinder driven joint model. Linear control through PID and State-space and non-linear Sliding Mode Control and Bang-Bang control will be simulated and discussed. The objective of all control regimes will be joint position control. We will compare rise time, settling time, steady state error and the effect of control loop sampling.

Linear control parameters will be determined through trial and error tuning of the situation. The tuning rules will be explained alongside each control regime. This method is chosen because we are not linearising the governing equations, hence the methods from linear control theory are difficult to implement. The linear control examples will be tuned qualitatively and simulated.

Once the control regimes have been simulated, the effect of sampling in the control loop will be examined. It will determine whether the control regime requires a high performance control system, or whether an agricultural pace of control can be permitted.

B.4.1. PID Control

In PID control, the output of the system $y(t)$ is compared to the input reference $r(t)$ to determine an error $e(t)$:

$$e(t)=r(t)-y(t)$$

This error is then manipulated to determine the control signal to go into the system, in our case the signal that actuates the valves on the hydraulic actuator. The manipulation of the error signal is what gives the control regime its name. The control signal $u(t)$ is made up of a linear combination of the error (proportional), the rate of change of the error (derivative) and the sum of all previous errors (integral). The relationship is described below:

$$u(t)=pe(t)+i\int e(t)dt+d\frac{de(t)}{dt}$$

where p is the proportional gain, i is the integral gain and d the derivative gain. To determine the best combination of these gains analytically relies on linearising the governing equations and then modifying them using pole placement techniques. However with an inherently non-linear system the resulting configuration still has to be tuned to the hardware as there is no linearising a fixed geometric configuration. Tuning is completed via some rules of thumb in particular Zeiger-Nichol's Quarter Wave Damping (“Quantitative PID tuning procedures,” 2013). This is a common PID control technique and consists of the following steps (“Tuning PID Controls,” 2013):

1. Turn the integral and derivative modes off and enable the closed loop control with proportional mode only;
2. Increment the proportional gain by a quarter of the previous value and monitor the output for oscillations;
3. Continue until the outputs successive peaks are about one quarter of in amplitude of the previous peak;
4. Enable the integral mode and increment the gain to eliminate the output error;
5. Finally enable and add derivative control only if the control loop output needs better performance.

So we will follow the steps provided and tune our response cylinder driven servo joint:

B.4.1.1. PID Tuning Steps 1-3 – Setting Proportional Gain

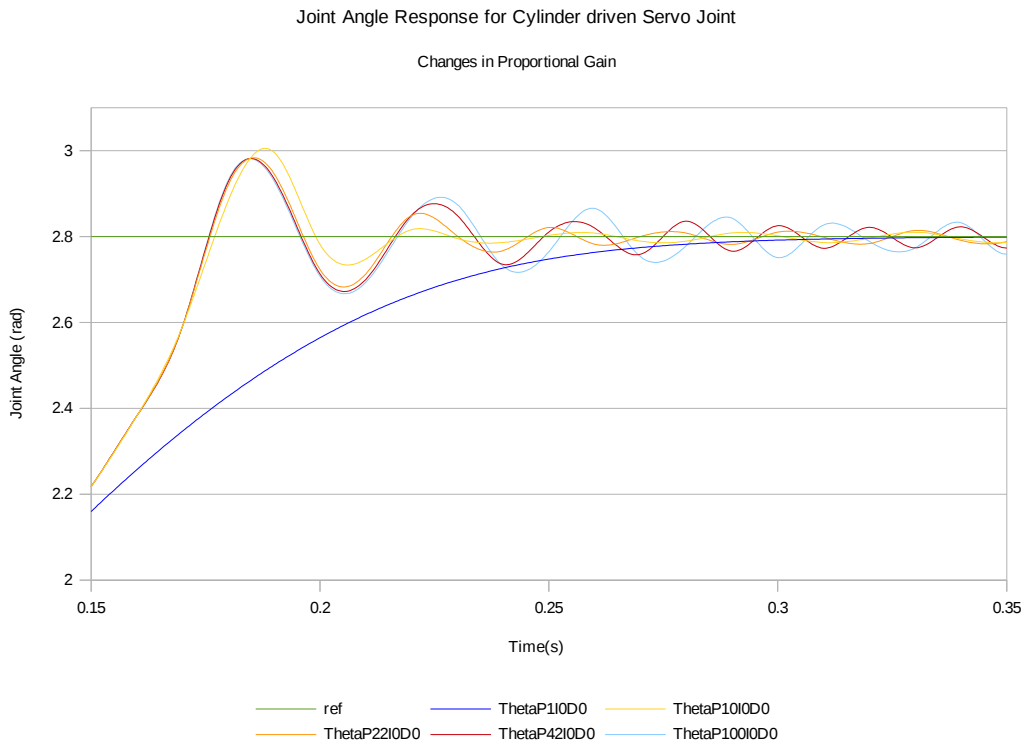


Illustration 36: Joint Angle Response to a step input, for differing levels of proportional gain in a cylinder driven Servo Joint

In Illustration 36 the effect of changing the proportional gain against the desired response is demonstrated. An explanatory note of the chart; this diagram will need colour graphics to show up. In the legend the code P1I0D0 regard the different levels of gain for the PID controller, in this case proportional is given a level of 1 with zero integral and derivative gain. The control response that has overshoot whose second and third oscillations are about a quarter of the previous peak is desired. It was assessed that this was achieved at the P=42 mark. Note that the P=22 response, the second oscillations are only abour 10% of the previous peak. In contrast the P=100 response is a little too energetic with the subsequent peaks being about 50%. The problem with too much gain is when the drive no longer shortens the rise time, but only serves to increase the settling time and and additional proportional gain will only require greater additional derivative gain to dampen. This will decrease stability. So the quarter amplitude selection of P42 seems to meet the quarter amplitude decay requirements without excessively contributing to settling time. After about the third peak the oscillations no longer decrease in amplitude but oscillate about the reference.

B.4.1.2. PID Tuning Steps 4 – Setting Integral Gain

In this step, the steady state error is examined to see if any integral gain could eliminate this error. The steady state error for this system, is very small due to the stiffness of the system. The system essentially gets maximum drive for very small values of control signal. In this case Integral gain is set to a very small amount of I=1e-4. This leaves a tiny bit of error which is slowly eliminated, that is difficult to see on the scale of the graph.

This is proven in Illustration 37, where a larger integral gain of I = 1e-2, causes a decrease in the response due to the integral of the initial error. This eventually drives itself back to zero error. The flat part of the response is due to the cylinder reaching its limits. Overall, the integral error is really not needed in this situation and is better if it is left extremely small or zero. For the remainder of the testing we have settled on an integral gain value of I=1e-5.

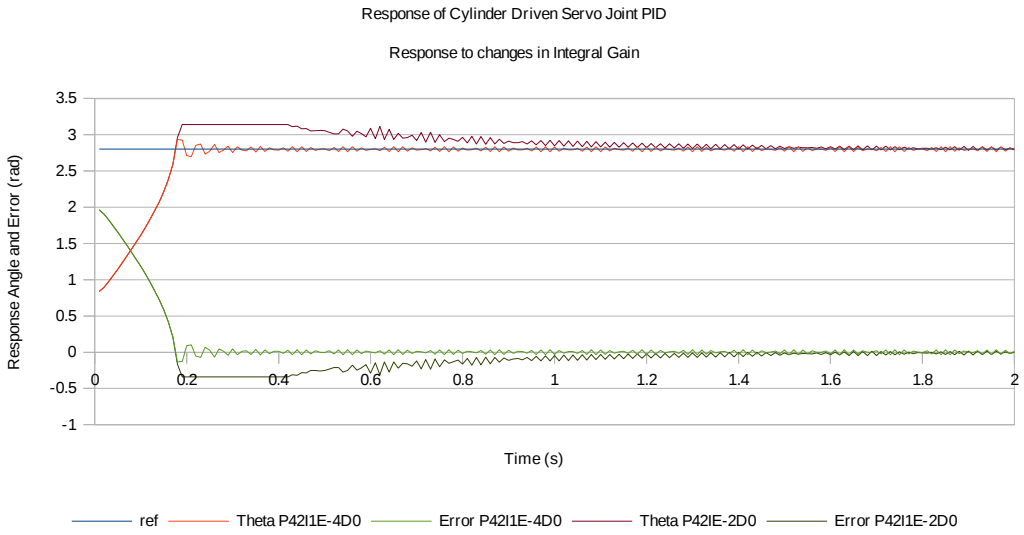


Illustration 37: Response to a step input for different levels of Integral Gain

B.4.1.3. PID Tuning Step 5 – Adding Derivative Gain to improve response

Derivative gain has the effect of dampening oscillations. Our response is affected by oscillations and could be considered underdamped. Adding derivative gain can be used to limit the oscillations seen above and prevent overshoots without greatly affecting rise time. This is the critically damped case. If used in excess the system will become sluggish and is known as overdamped.

During tuning, Derivative Gain was increased to a point where there was no overshoot of the reference. As is shown in Illustration 38, with D=0.1, there is still overshoot, the overshoot isn't eliminated until D=1 and this could be considered critically damped. As an example of over damping the system, the derivative gain is

increased to D=10.

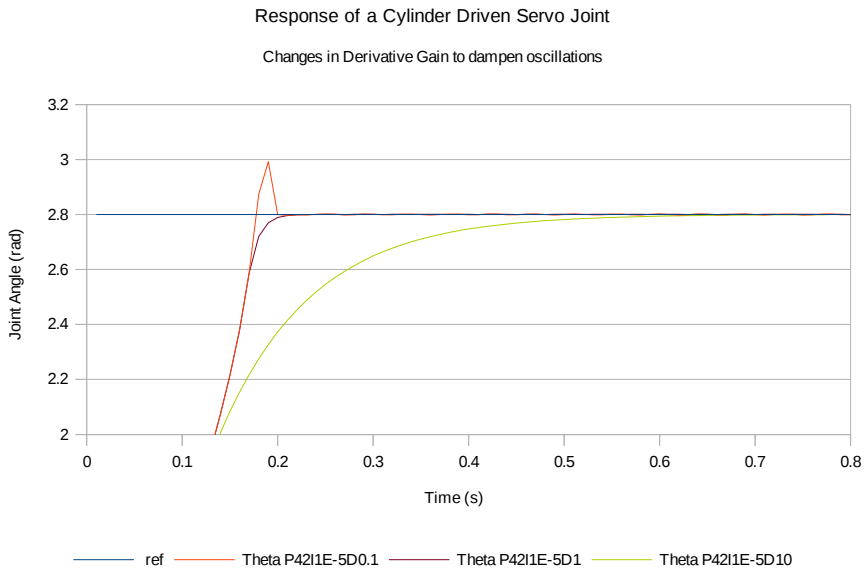


Illustration 38: Effect of Derivative Gain on Joint Response to step input

Overall, the PID tuning of the system is effective at controlling a hydraulically driven servo joint when using a continuous control signal. The response to a ramp input is also very good, primarily due to the high level of gain. However, there is one effect that hasn't been considered and that is the effect of sampling rate on the control system. As this control system will be digitally implemented on a a computer, the computer will only have discrete time instances when it will receive the feedback and adjust the control signal Effectively due to the time step of our simulation so far, the controller sampling rate is 10kHz as the controller is updated every simulation time step. What will happen when the sampling rate is reduced to an agricultural 10Hz?

B.4.1.4. Effect of Sampling time on Joint Response.

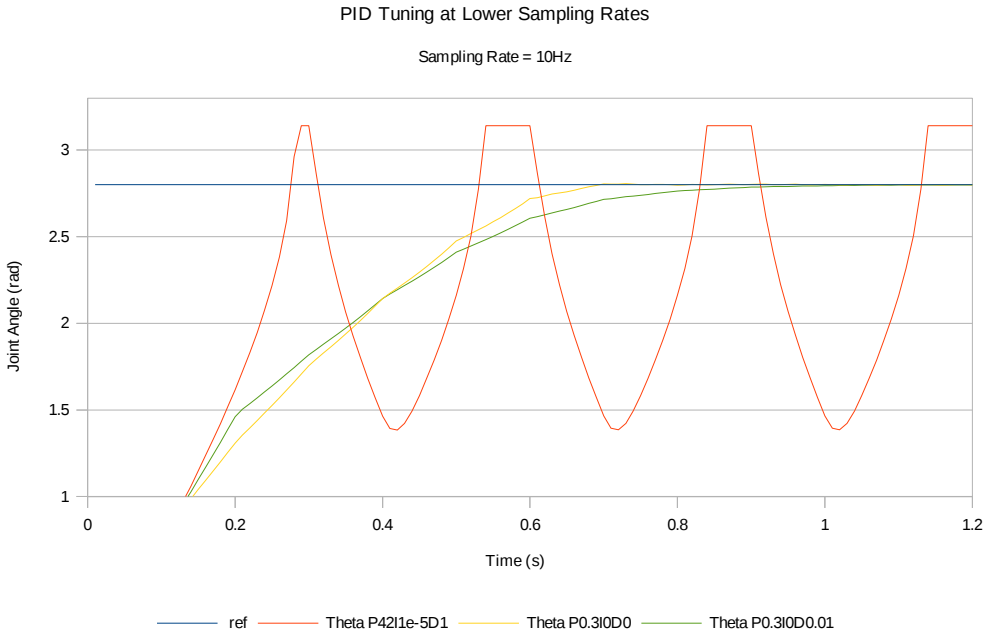


Illustration 39: Response to a step input of a PID tuned system at a sampling rate of 10Hz

Illustration 39 shows our original 10kHz system tuning under a 10Hz sampling regime. The highly oscillatory nature of the response indicates that the proportional gain is too high. The selection of a lower gain seems to make the system more controllable, however the effect of disturbance can upset systems that rely on high proportional gain tempered with decent derivative or integral gain.

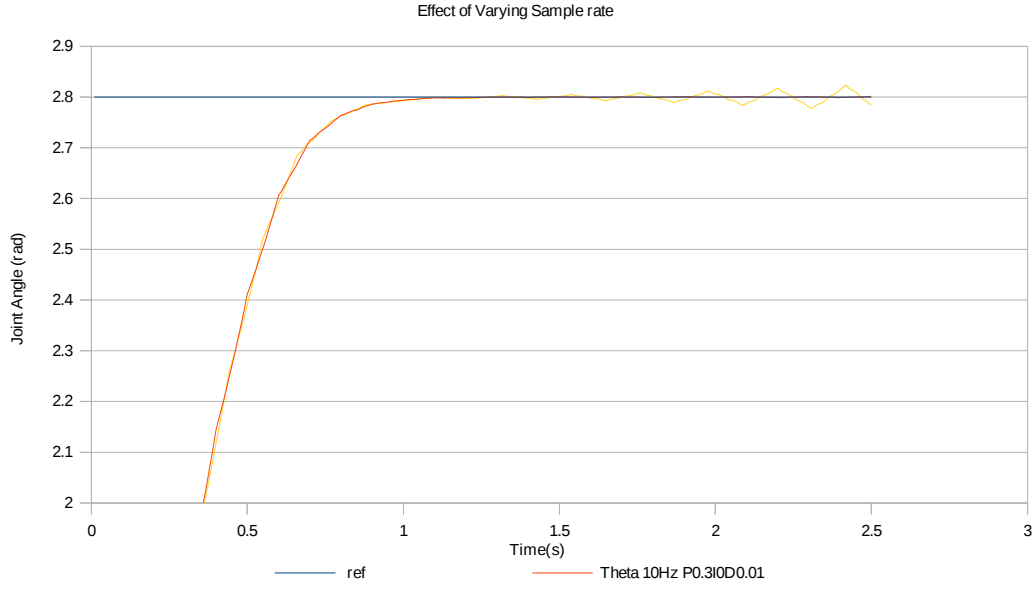


Illustration 40: Effect of changing sample frequency for a PD system with derivative gain

This is exhibited by the change in response of the system tuned with P=0.3, I=0 and D =0.01, when sampled at different rates as per Illustration 40. This response curve shows that by reducing the sample frequency, performance is degraded and becomes unstable as indicated by the increasing oscillation around the 2 second mark. This can be contrasted with the response of a system that has no derivative gain and only proportional gain. It maintains its stability in even lower sampling environments as is shown in Illustration 41. Thus, for a lower sampled system, maintaining a control loop with only proportional control is a way of ensuring that the system remains stable under the effect of a variety of noise and disturbance.

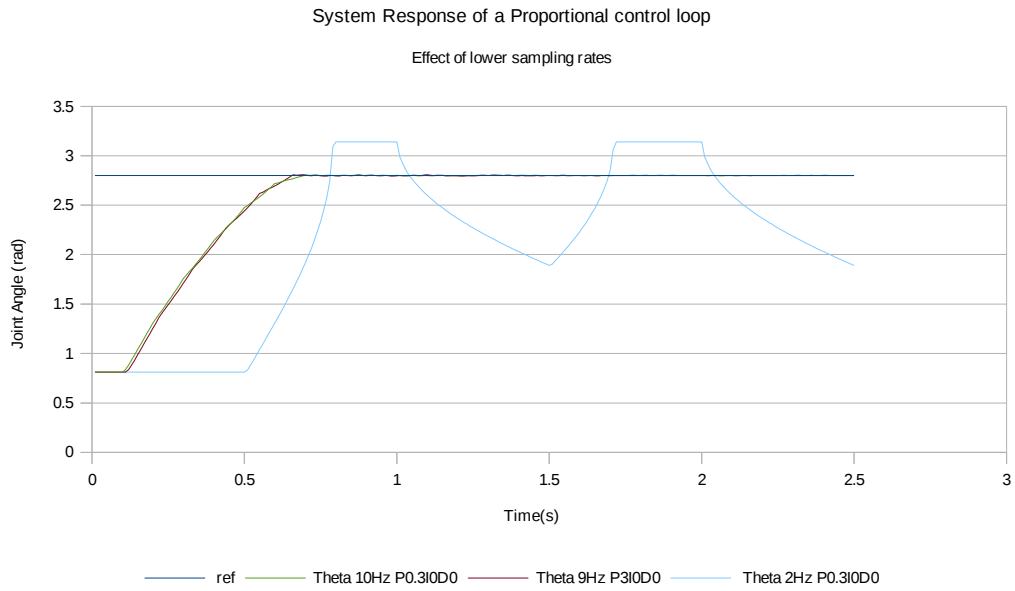


Illustration 41: Effect of reducing the sampling rate on a control loop with only proportional gain

Ultimately, the performance of the control loop is dependent on the sampling frequency of the system with higher frequencies being preferred. Additionally 10Hz which is quite low is shown to be suitable as a sampling rate. The effect of low sampling rates is to degrade servo performance. It is therefore important to ensure that the sampling rate is sufficiently high to allow higher gains to be used in order to maintain the required performance of the system.

B.4.2. State Space Feedback

In State Space Feedback the system's states are measured or approximated and a gain matrix is applied before they are compared to the reference. This is shown in Illustration 42 which depicts a system where state rate of change $\dot{x}$ is a linear combination of input δ adjusted by the input matrix *B* and the states transformed by the plant matrix *A*. The output θ is also dependent on state values modified by

an output matrix *C*. In this way the system's states must describe all aspects of the system. In a closed loop system the states are transformed by a feedback gain matrix *K* and linearly combined with a reference to determine the input term. Therefore the open loop state space response is given by linear control representation as:

$$\dot{x} = Ax + B\delta$$

$$\theta = Cx$$

In the closed loop, the states are multiplied by the gain matrix K, and then subtracted from the reference value to arrive at δ , such that:

$$\delta = \theta_{des} - Kx \quad ,$$

therefore the open loop response then becomes:

$$\dot{x} = Ax + B(\theta_{des} - Kx)$$

This is not entirely accurate for the system given that it contains non-linear components for working with the inputs. *B* would be non-linear and a combination of both inputs and states. Additionally, the model terms describing the load has some non-linear elements, such as friction, gravity that are not dependent on any state. Additionally the joint state variables are related to the cylinder state variables via non-linear geometric terms as described in Section B.2

Nevertheless our model does define a relationship between states x and their rate of change $\dot{x}$.

If the state to be controlled is not a derivative of another state, for example, position in a second order system where the next state is velocity. Then the equivalent PID gains would only be proportional and derivative with no integral gain being possible. Hence, elimination of steady state error, for which integral control is used, is achieved through a scaling of the reference which is applied before the summation of the feedback(“State Space Controller,” 2013).

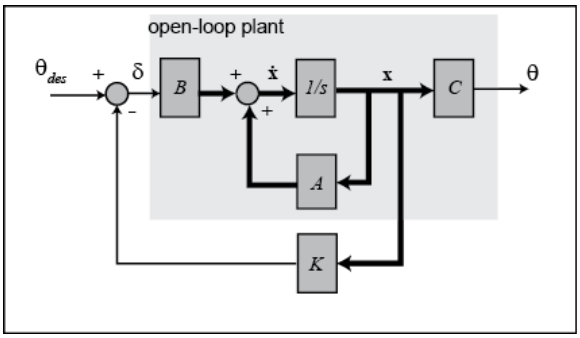


Illustration 42: Generic State Space system with feedback

Given the manipulator's dynamic equations, a canonical form cannot be arranged without linearising the system about some point in the operating space. As a result the state equations are:

$$\dot{x}_p = v_p$$

$$\dot{v}_p = \frac{A_p}{m} P_L - g$$

$$\dot{P}_L = \frac{-4\beta A}{V_t} v_p - \frac{4\beta C_t}{V_t} P_L + \frac{4\beta}{V_t} C_d \sqrt{\left(\frac{P_s}{\rho}\right)} w x_v \sqrt{\left(1 - \frac{x_v}{|(x_v)|} \frac{P_L}{P_s}\right)}$$

where the states are: $x_1 = x_p$, $x_2 = v_p$, $x_3 = P_L$ and $u = x_v$. Also to be included is the output relationship (roughly equivalent to what the output matrix *C* does) which is also non-linear:

$$\theta = \cos^{-1}\left(\frac{(x+d)^2 - a^2 - b^2}{2ab}\right) \quad \text{and}$$

$$\dot{\theta} = \frac{\theta_k - \theta_{k-1}}{dt}$$

As this state-space representation is non-linear, the typically state-space control techniques of applying the feedback gains to the characteristic equation is not possible without extensive linearisation, the motive for not doing this has been discussed previously. Therefore the state feedback gains need to be tuned until a desired response is achieved.

The aim is to control joint angle, building on the work of the PID control section, quarter amplitude tuning is again implemented. Starting with zero gain for feedback of all states, the proportional gain (ie the gain for the position state) is increased until the system overshoots and dies off with each successive peak 25% of the preceeding one. Now there is no integral adjustment to be made, if needed steady state error can be eliminated by scaling the reference as detailed above. The final element to adjust is the feedback gain of the velocity state to eliminate oscillation.

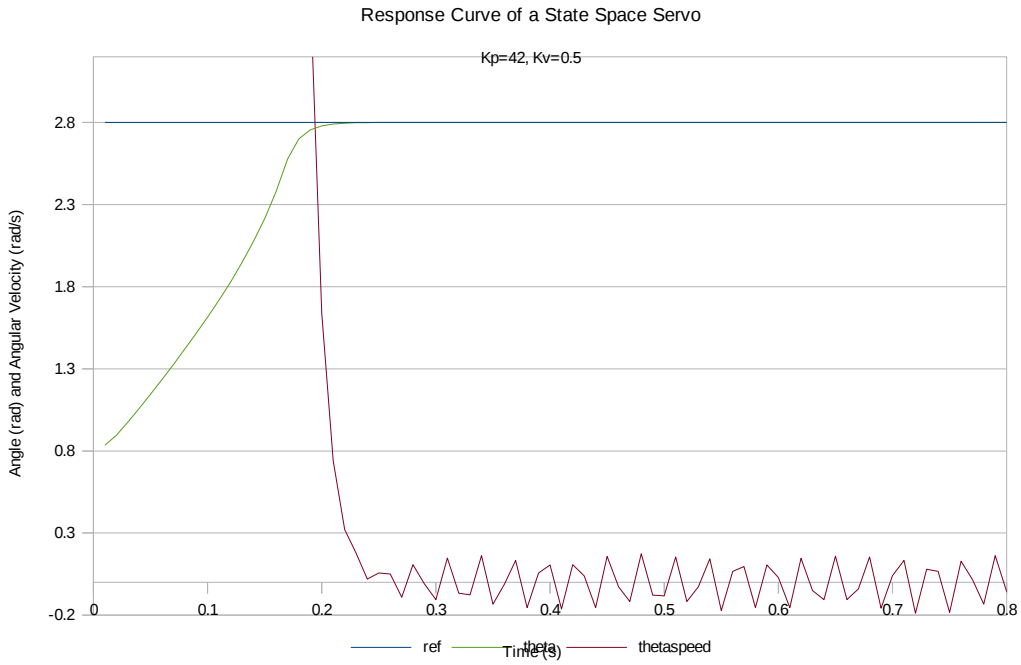


Illustration 43: Response curve showing angle, velocity and reference, with gains similar to those in the PID critically damped system

When the proportional and integral gains of the PID system are applied as state feedback gains for position and velocity the curves are very similar. Taking $k_1 = 42$ and $k_2 = 0.5$ the simulation outputs the curve shown in Illustration 43, which is very similar to that in Illustration 38 showing critical damping.

B.4.2.1. Discussion

Overall, the response of the state-space system is very similar to that of the PID control system. When comparing both systems there are significant differences in implementation, yet there is strong agreement between the responses of the servo system for similar levels of gain.

B.4.3. Variable Structure Control

Variable Structure Control was a control regime developed in the 1950s, where a number of control methods are used in different parts of the error space, with non-linear transitions between them. For example, in a system where position control is desired, the drive signal may be maximum at large errors, towards zero error. At some pre-determined point the signal may change to a PID control regime or other control regime which may have more desirable properties such as stability, settling time, or steady-state error. Hence, as a whole, the system is able to enjoy the best of both worlds; maximum drive and accuracy(Bengiamin and Kauffman, 1984, p. 3). Two control regimes derived from Variable Structure Control are Sliding Mode Control and Bang-Bang Control.

Both Sliding Mode Control and Bang-Bang Control are non-linear control systems that rely on a switching regime to change between different control configurations states. These states are typically driven close to their maximum input. As a result show little response change to variations in the disturbances on the system and can handle noise quite well. This quality makes them useful in real world control arrangements where *robustness* – the ability to maintain a response in the face of disturbance, is key. They are discussed together because the switching mode is the same, only the magnitude of drive that differs. Due to the rapid propagation of force throughout a hydraulic system, variable structure controllers of such systems can exhibit very good response characteristics.

B.4.4. Sliding Mode Control

Sliding Mode Control drives itself backwards and forwards across a particular switching line to drive to a zero error. The switching line is the mode and the system appears to slide down when the response is plotted on a phase plane (x versus v). Control regimes that rely on this effect to ensure zero error are known as Sliding Mode Controllers.(S. Lorenzo and M. Shaker, 1988, p. 1059) The drive in a sliding mode controller is proportional to the error, but changes sign according to whether it is greater or less than the switching function, typically a linear function with a negative gradient passing through the origin of the phase plane.

Sliding Mode Controllers exhibit a phenomenon known as “chattering” that occurs near the origin, it is apparent when the drive has driven to a significantly small error, that it begins to oscillate across the switching line. This can be eliminated by changing structure at small errors to include integral values(Bengiamin and Kauffman, 1984, p. 3). When using actuators with suitable load holding characteristics at zero drive, deadband at a suitable error level can be applied to the signal in order to eliminate chattering. This requires deadband to be suitably small to ensure accurate positioning of the joint, and a sufficient holding torque to ensure the joint is held true.

B.4.4.1. Control Algorithm

The control regime is best illustrated on what is known as the *Phase Plane*. This is a graph which the velocity state is plotted on the y-axis and the position state is plotted on the x axis. The switching line is represented as a line through the origin that has a negative slope as shown in Illustration 44. The system is driven by two gains either side of the switching line k1, and k2. The gains operate to position the response toward the nearest switching line.

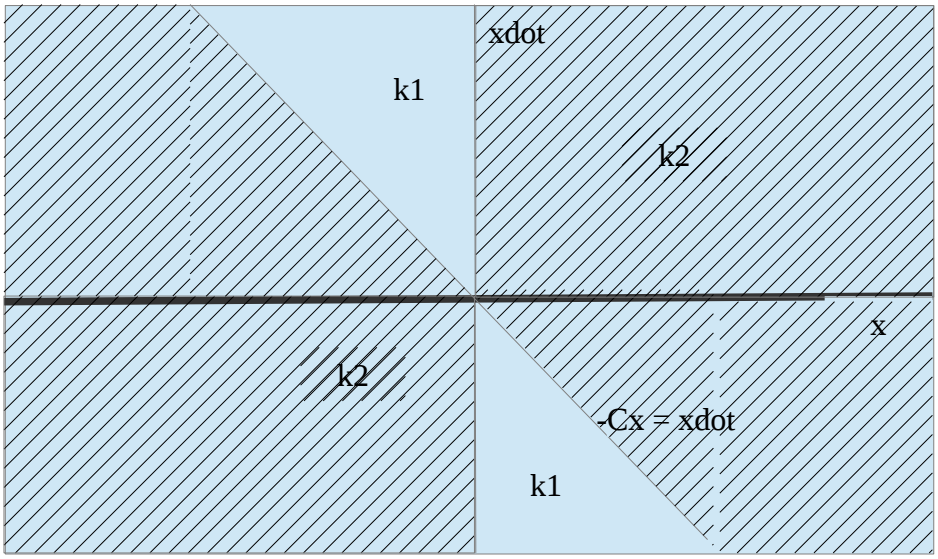


Illustration 44: The phase plane

The switching line is given by:

$$\sigma = Cx + \dot{x}$$

when $\sigma = 0$. Therefore the drive, *u*, from the system is given by

$$u = \psi x$$

where

$$\psi = \begin{cases} k_1, x\sigma > 0 \\ k_2, x\sigma < 0 \end{cases}$$ (Bengiamin and Kauffman, 1984, p. 3)

The response of the system can be manipulated by changing the slope of the switching line C, the forward gain k_1 and the reverse gain k_2 . By having a steeper switching line the response is faster, however the response may not travel down that line if the switching gains are not increased sufficiently. Examples of the system switching down different switching lines is shown in Illustrations 45 to 48. The sampling rate doesn't have too great an effect on the viability of the sliding mode controller any thing faster than 10 Hz will result in a reasonably effective control situation.

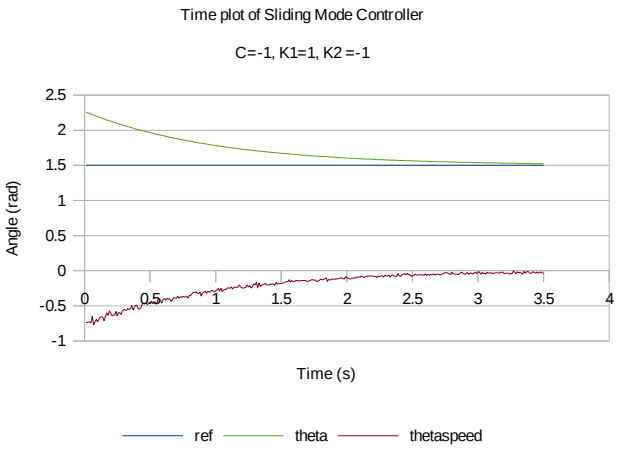


Illustration 45: Time plot of system switching down a sliding mode of -1 with small gains 1 and -1.

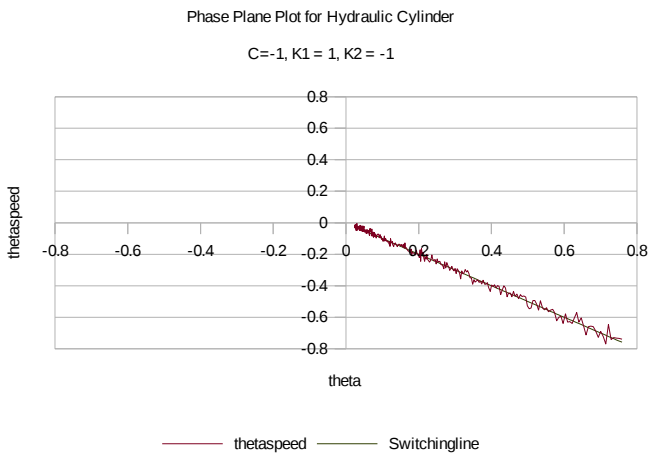


Illustration 46: Phase Plane Plot of system switching down a sliding mode of -1 with small gains 1 and -1

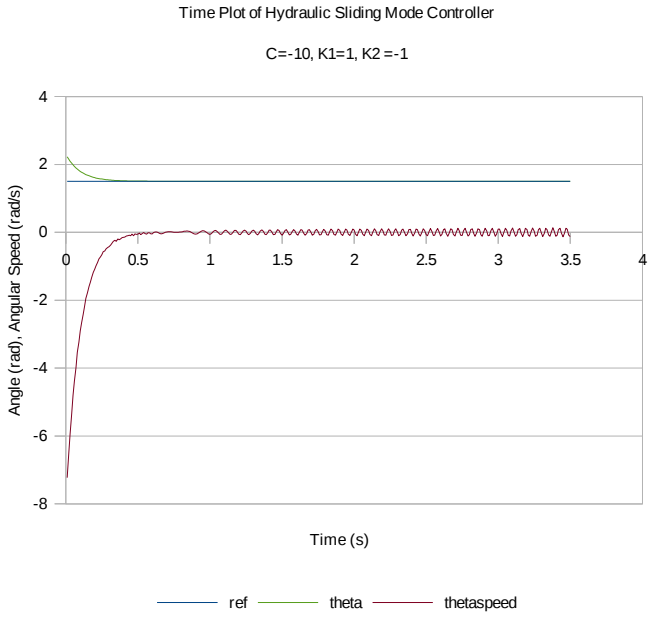


Illustration 47: Time plot of system switching down a sliding mode of -10 with small gains 1 and -1

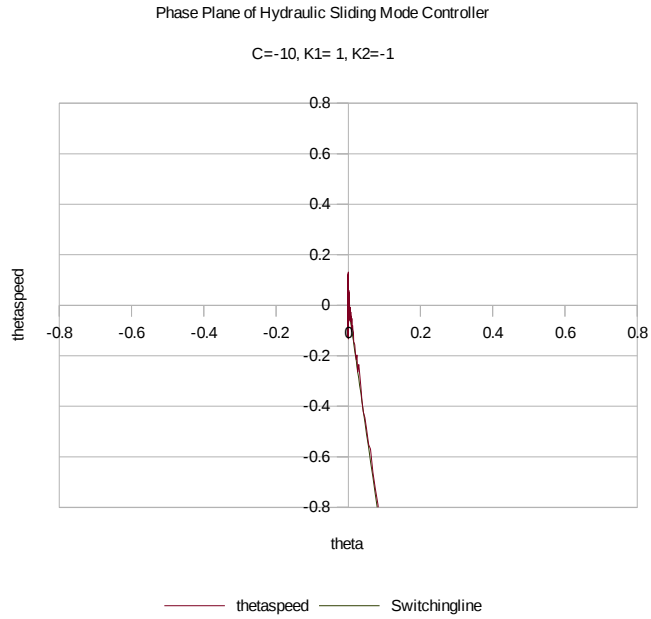


Illustration 48: Phase Plane Plot of system switching down a sliding mode of -10 with small gains 1 and -1

One of the key reasons why 10Hz control loops can be tolerated is because of the lower gains that are being used to set the drive, $k_1 = 1$, $k_2 = -1$. When the gains approach infinity, such that the drive is either 1 or -1 then the Sliding Mode Controller changes into a Bang-Bang Controller and the effect of sampling period is not nearly so tolerant.

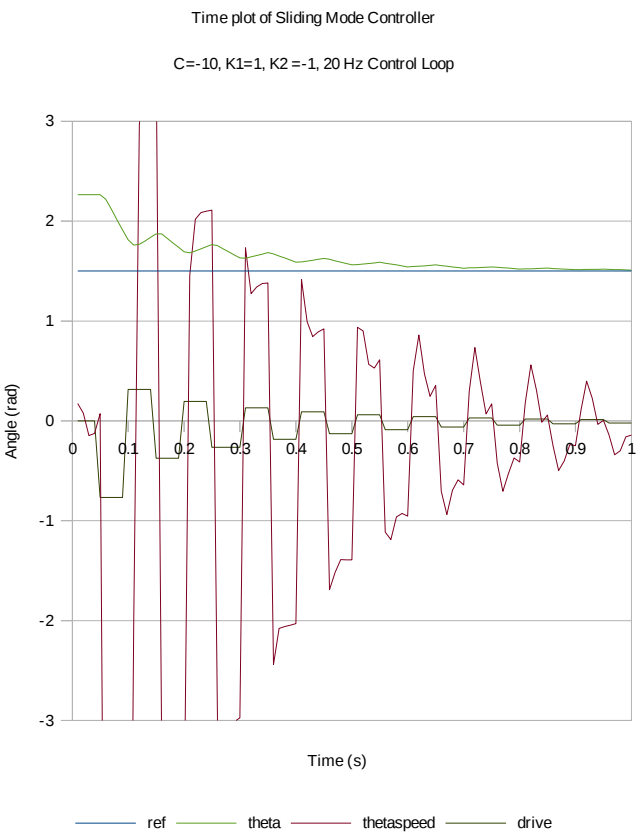


Illustration 50: Sliding Mode Controller operating at 20Hz control rate.

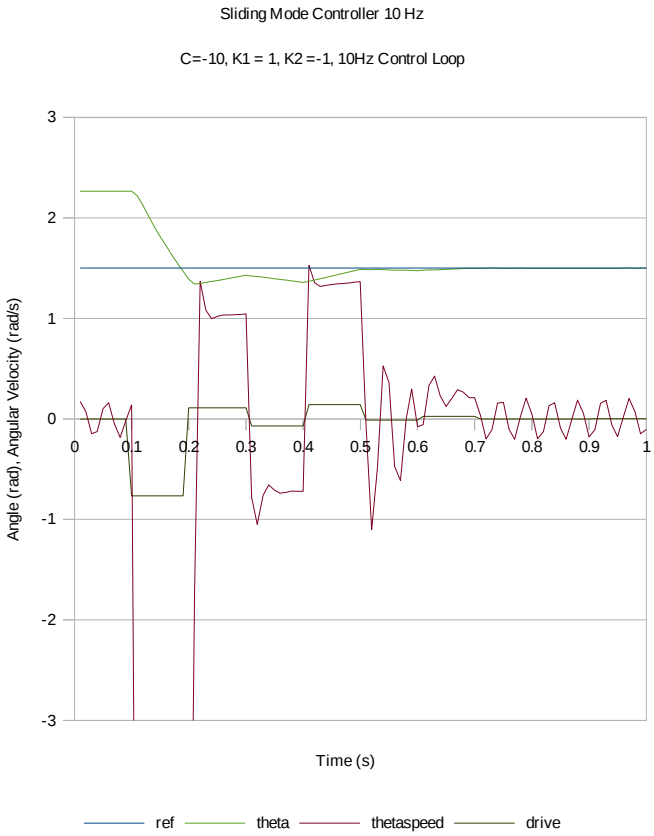


Illustration 49: Sliding Mode Controller operating at 10Hz control rate

B.4.5. Bang Bang Control

As previously stated, the Bang Bang controller is a Sliding Mode Controller, with infinite gain. When sliding down the mode the controller uses the full drive available. The benefits of a Bang-Bang Controller are a simpler physical setup and a very robust control regime. This regime rejects noise and disturbances well due to the maximum drive applied by the actuators. For example Illustration 51 shows a Bang-Bang Controller operating down a sliding mode with C = 100.

The sliding mode is defined as before:

$$\sigma = Cx + \dot{x}$$

but now the drive is not proportional to the error, but full drive, *u* as given below:

$$u = \begin{Bmatrix} 1, x\sigma > 0 \\ -1, x\sigma < 0 \end{Bmatrix}$$

The response curves of Bang-Bang Controllers are very similar to that of Sliding Mode Controllers, however steeper sliding modes can be achieved ensuring the responses are very good. The penalty for working with such high gains, is that the control loop is more critical. The effects of control loop update rate on performance

using the Sliding Mode are examined from Illustrations 50 and 49.

In Illustration 51, we can see that the Bang-Bang controller on our system is quite sensitive to the sampling frequency of the control loop, note the changing scale on these graphs. Note that this relationship is particular to the choice of valve size. In order to accept a lower sampling requirement, a smaller valve should be chosen. This is particularly important as at 200Hz the dynamic limits of a typical solenoid valve driver are approached.

B.4.6. Discussion

In examining sliding mode controllers we have looked at two basic implementations of them. The Sliding Mode Controller with a proportional gain, and a Bang-Bang Controller with infinite gain. They are both robust as long as the sampling requirements can be met due to their fundamental instability(J. Billingsley, 2009, p. 74). Each of these examples suffer from “chattering” which can be seen in the angular velocity time plots, but there are a couple of methods to eliminate this problem. Chattering and the effect it has mechanically on the actuators, is dependent on sampling frequency, the system's response times, and the drive value applied.

When applied directly to hydraulic proportional valves, sliding mode controllers can result in a high number of actuation cycles, if the sampling loop frequency is low. These cycles can quickly use valve life. If the sampling frequency is higher than the valve response frequencies, this effect can be reduced. For common valves these start at around 200Hz.

The drive value can also have an effect, the full drive of a Bang-Bang controller, will

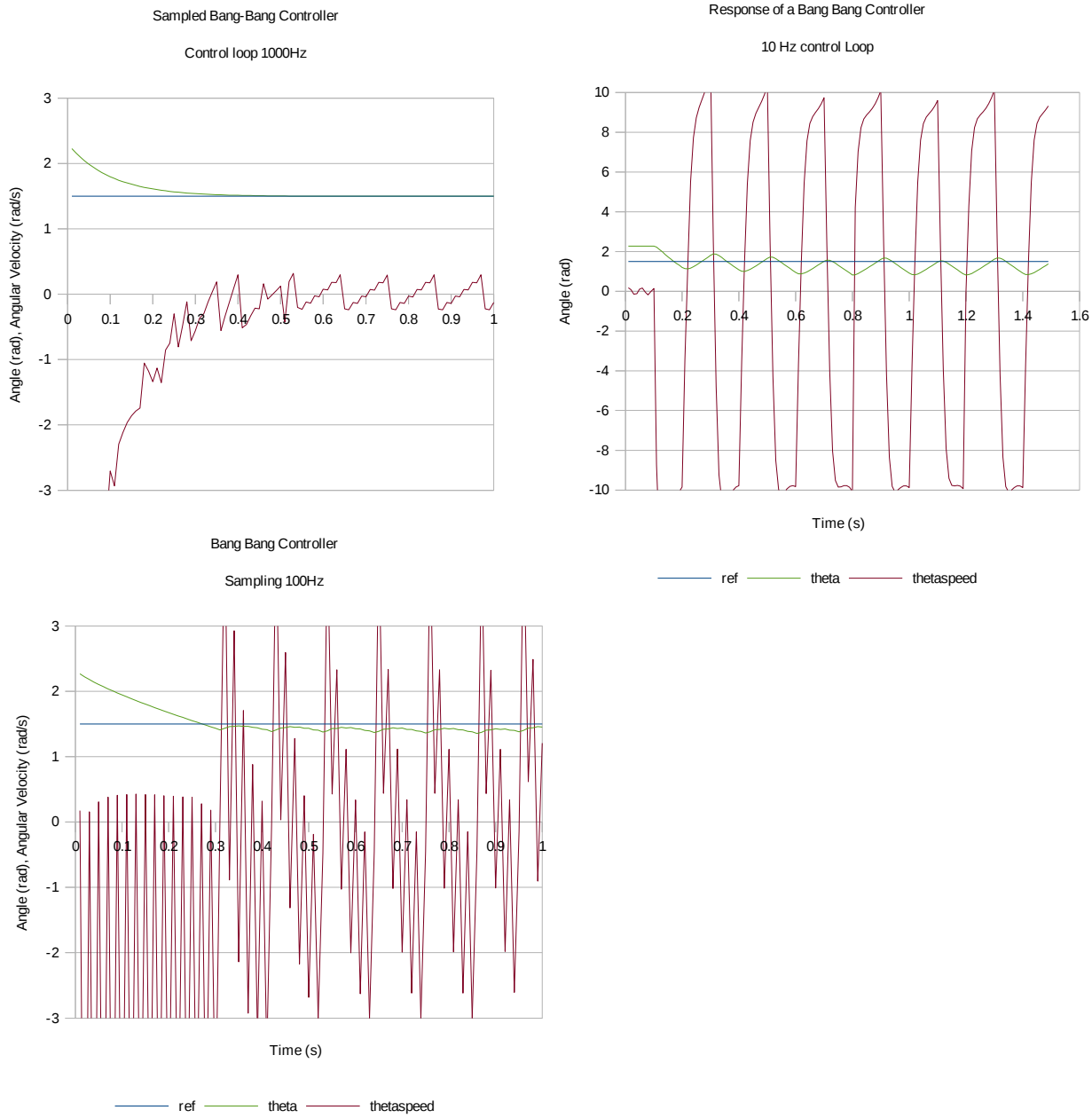


Illustration 51: Effect of changing sampling rates on a Bang Bang Controller. For this system with drives at this gain would require a control loop towards 1KHz.

ensure that the valve opens completely if the control frequency is within the valve's dynamic range. However when using the proportional drive of a sliding mode controller, as small error values, the drive signal may not be sufficient to ensure that the valve opens at all, due to the mechanical deadband within the valve due to friction and profile design. This has the effect of imposing a deadband on the valve.

Deadband, particularly that imposed in software, is probably the most effective method for avoiding chatter at low frequency. It does require some consideration of design specifications such as acceptable joint position error and holding torque. Both of which depend on a suitably fast control loop to ensure that deadband is applied within the appropriate limits.

Appendix C Forward Kinematics

C.1. Basis for use

Forward kinematics describes the methods for determining the position and orientation of a particular point on a jointed mechanism in respect to another point. Given the joint positions, lengths of links and types of joints involved, the difference in position and orientation can be determined. It forms the basis for much of the simulation and control algorithms that are developed to work with manipulator arms.

C.2. Method

Assuming the only place an arm can bend is at its joints, calculating the forward kinematics for an arm consists of determining the transformation matrices that transform the coordinate frame of the previous link, to the coordinate frame of the current link, and continuing this for every link of the manipulator from the Base to the Tool. The Transformation from the Base Frame (which can be the world coordinate system) to the Tool is then the product of every link transformation matrix from the Base to the Tool. We will have a look at a worked example.

In calculating the forward kinematics of links connected by revolute joints, there are two types of transformations that allow the conversion of one link's coordinate system into the frame of an adjoining link. These transformations include a Translation, and a Rotation. Both of these can be expressed in a 4x4 Transformation Matrix, where the top-left 9 elements are the Rotation Matrix *R*, a 3x3 Matrix. The three righthanded elements are the Translation Vector *x*, a 3x1 vector which determines the amount of displacement in the *x*, *y* and *z* directions a point is translated. The bottom row is always {0,0,0,1} as shown below:

$$^{n}T_{m}=\begin{bmatrix} R & x \\ 0 & 1 \end{bmatrix}$$

A Transformation matrix which only describes translation, would keep *R* as identity and would thus be represented as:

$$^{n}T_{m}=\begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The rotation matrix describes rotations about roll, pitch and yaw (RPY). RPY can become problematic and suffer from *gimbal lock* when they are used to describe rotations around all three axes which share an origin. In forward kinematics this isn't such a problem as each joint only consists of a single axis.

The three rotation transformation matrices that will be encountered will describe rotations about the x-axis, y-axis and z -axis:

$$T_{\theta_x} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta_x) & -\sin(\theta_x) & 0 \\ 0 & \sin(\theta_x) & \cos(\theta_x) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{\theta_y} = \begin{bmatrix} \cos(\theta_y) & 0 & \sin(\theta_y) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta_y) & 0 & \cos(\theta_y) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{\theta_z} = \begin{bmatrix} \cos(\theta_z) & -\sin(\theta_z) & 0 & 0 \\ \sin(\theta_z) & \cos(\theta_z) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Each link coordinate frame is typically differs from the previous frame by a translation from the previous joint to the new joint and a rotation through the joint. There are many systems in place to maintain a coherent transformation structure down the arm. Denavit-Hartenberg (DH) methods are one, but it is often easier to choose a resting pose where all the angles in the arm are considered at their “zero” position. By maintaining orthogonality of this chosen resting pose with the world coordinate frame, the above transformations can be used directly. The transformation used is the one corresponding to the world axis the joint axis is parallel to. As the joint angles are all 0, the rotation transformations are straightforward to calculate.

C.3. Forward Kinematic Example:

In this example the forward kinematics will be calculated from the world coordinate frame, located top right in Illustration 8 down the arm through the next seven joints (there are eight shown in the diagram, we will neglect the one at the gripper). Each axes glyph shows the intended rotation with a small circle around one of the axes in the same colour of the axis. Red being the x-axis, green for y-axis and blue for z-axis. Due to the choice of “rest” position, all of the joints are aligned with the axes of the world coordinate frame, for all joints equal to zero.

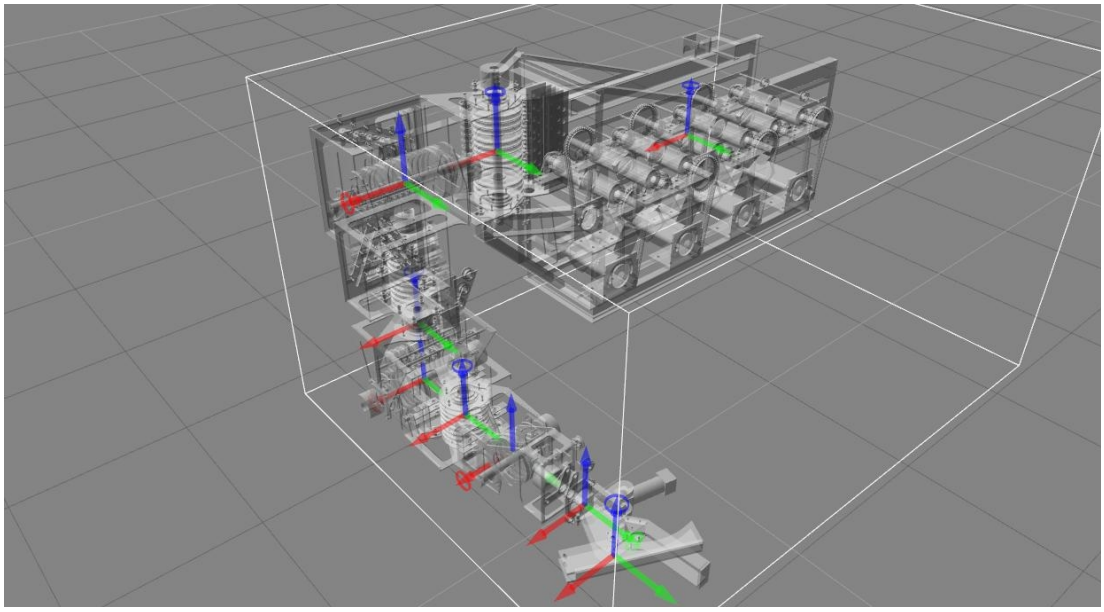


Illustration 52: Forward Kinematic example, showing a Base with a seven axis manipulator attached showing axis directions. Z-blue, Y-green, X-red.

For the First Transformation, from the World Frame to the First Link translation is applied from the origin to the first Joint. This translation is 900mm down the x-axis, 0mm along the y-axis and 279 mm higher on the z-axis. The translation Matrix is therefore:

$$ {}^{world}T_1 = \begin{bmatrix} 1 & 0 & 0 & 900 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 279 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

The Second Transformation is for any rotation through the first joint, of an angle θ_1 , whose axis is aligned with the z-axis. To calculate the transformation the rotation transformation matrix for rotation about the z-axis is used:

$$ {}^{1}T_{Shoulder} = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 & 0 \\ \sin(\theta_1) & \cos(\theta_1) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

So the coordinates have been translated from the origin to the location at the centre of the first axis, and subsequently been rotated through that axis. The coordinates are now in the frame of Link 1. The total transformation from world frame to the Link1 Frame, ${}^{World}T_{Shoulder}$ is the product of the two:

$$ {}^{World}T_{Shoulder} = {}^{World}T_1 \, {}^{1}T_{Shoulder} $$

$$ {}^{World}T_{Shoulder} = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 & 900 \\ \sin(\theta_1) & \cos(\theta_1) & 0 & 0 \\ 0 & 0 & 1 & 279 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

This method can be continued to determine the transformations between a link and its preceding link. For the next link a translation 422mm in the x direction, 0mm in the y-direction and 0mm in the z-direction is required. The joint at that point rotates about an x-axis by an angle θ_2 the transformation

For an arm of seven joints the result will comprise of seven such transformation matrices. Each made up of a translation and rotation that transform the coordinate system between link frames. Therefore to determine the relationship of the tool to the world coordinate frame would require multiplying all the transformations from the world frame to the tool frame.

$$ {}^{World}T_{Tool} = {}^{World}T_{Shoulder}\,{}^{Shoulder}T_{LShoulder}\,{}^{LShoulder}T_{UpprArm}\,{}^{UpprArm}T_{ForeArm}\,{}^{ForeArm}T_{WristZ}\,{}^{WristZ}T_{WristX}\,{}^{WristX}T_{Tool} $$

The transformation matrix ${}^{World}T_{Tool}$ will be a 4x4 matrix whose values correspond with world position and orientation according to the equivalent matrix elements listed below.

$$ {}^{World}T_{Tool} = \begin{bmatrix} c_\gamma c_\varphi & -s_\gamma c_\beta + c_\gamma s_\varphi s_\beta & s_\gamma s_\beta + c_\gamma s_\varphi c_\beta & x \\ s_\gamma c_\varphi & c_\gamma c_\beta + s_\gamma s_\varphi s_\beta & -c_\gamma s_\beta + s_\gamma s_\varphi c_\beta & y \\ -s_\varphi & c_\varphi s_\beta & c_\varphi c_\beta & z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

where x, y, z is the position of the tool, and the rotation matrix a combination of world coordinate, roll β, pitch φ, and yaw γ. There are many ways to extract the roll pitch and yaw from the rotation matrix, but the best ones rely on the use of the inverse tangent function, atan2(). Atan2() is beneficial as it will also determine the quadrant of the angle lies in. Roll β is determined below:

$$ \beta = \tan^{-1}\left(\frac{m_{32}}{m_{33}}\right) $$

$$ \beta = \tan^{-1}\left(\frac{c_\varphi s_\beta}{c_\varphi c_\beta}\right) $$

$$ \beta = \tan^{-1}\left(\frac{s_\beta}{c_\beta}\right) $$

Yaw γ is determined by a similar formula:

$$\gamma = \tan^{-1}\left(\frac{m_{32}}{m_{33}}\right)$$

$$\gamma = \tan^{-1}\left(\frac{c_\varphi s_\gamma}{c_\varphi c_\gamma}\right)$$

$$\gamma = \tan^{-1}\left(\frac{s_\gamma}{c_\gamma}\right)$$

To determine Pitch requires the use of the pythagorean identity $\sin^2\theta + \cos^2\theta = 1$. There a two good options for extracting pitch:

option 1:

$$\varphi = \tan^{-1}\left(\frac{-m_{31}}{m_{11} * \cos\gamma + m_{21} * \sin\gamma}\right)$$

$$\varphi = \tan^{-1}\left(\frac{s_\varphi}{c_\varphi c_\gamma c_\gamma + c_\varphi s_\gamma s_\gamma}\right)$$

$$\varphi = \tan^{-1}\left(\frac{s_\varphi}{c_\varphi\left(c_\gamma{}^2 + s_\gamma{}^2\right)}\right)$$

$$\varphi = \tan^{-1}\left(\frac{s_\varphi}{c_\varphi}\right)$$

option 2:

$$\varphi = \tan^{-1}\left(\frac{-m_{31}}{\sqrt{m_{11}^2 + m_{21}^2}}\right)$$

$$\varphi = \tan^{-1}\left(\frac{s_\varphi}{\sqrt{c_\varphi^2 c_\gamma^2 + c_\varphi^2 s_\gamma^2}}\right)$$

$$\varphi = \tan^{-1}\left(\frac{s_\varphi}{c_\varphi\sqrt{c_\gamma^2 + s_\gamma^2}}\right)$$

$$\varphi = \tan^{-1}\left(\frac{s_\varphi}{c_\varphi}\right)$$

So now the coordinate transformation can be calculated from the world to the end effector using the joint angles. The position coordinates and RPY angles for the tool in the world frame can be extracted.

C.4. Conclusion

Hence the location of the tool frame can be determined if the position of the joints are available for calculation. This is forward kinematics, determining the position of the tool given the joint positions, lengths of links and types of transforms. It forms the basis for much of the simulation and control algorithms that are developed to work with manipulator arms.

Appendix D Arm Dynamics - Recursive Newton Euler Algorithm

Previous sections have focused on the determination of what position and orientation a tool will have for a given joint angle (Appendix C), and also what change in joint angle will allow us to achieve a given change in tool position and orientation (Section 4.3). However for a more comprehensive simulation that incorporates forces and weights, arm dynamics need to be incorporated into the simulation. The inverse dynamics algorithm, determining the joint forces from the orientation and speed of the kinematic chain was taken from Featherstone's published version of the RNEA (R. Featherstone, 2008, chap. 1–5). This version differs from many classical versions in that it utilises spatial vectors for force and speed that combine both the angular and linear components into 6D vectors. The benefit of this is that all the classical angular and linear calculations can be represented in a single line of code. The entire algorithm, if there is access to suitable Matrix libraries, can be represented in as little as 22 lines of source code(R. Featherstone, 2010, p. 93). The algorithm in pseudocode is shown in Figure 2.

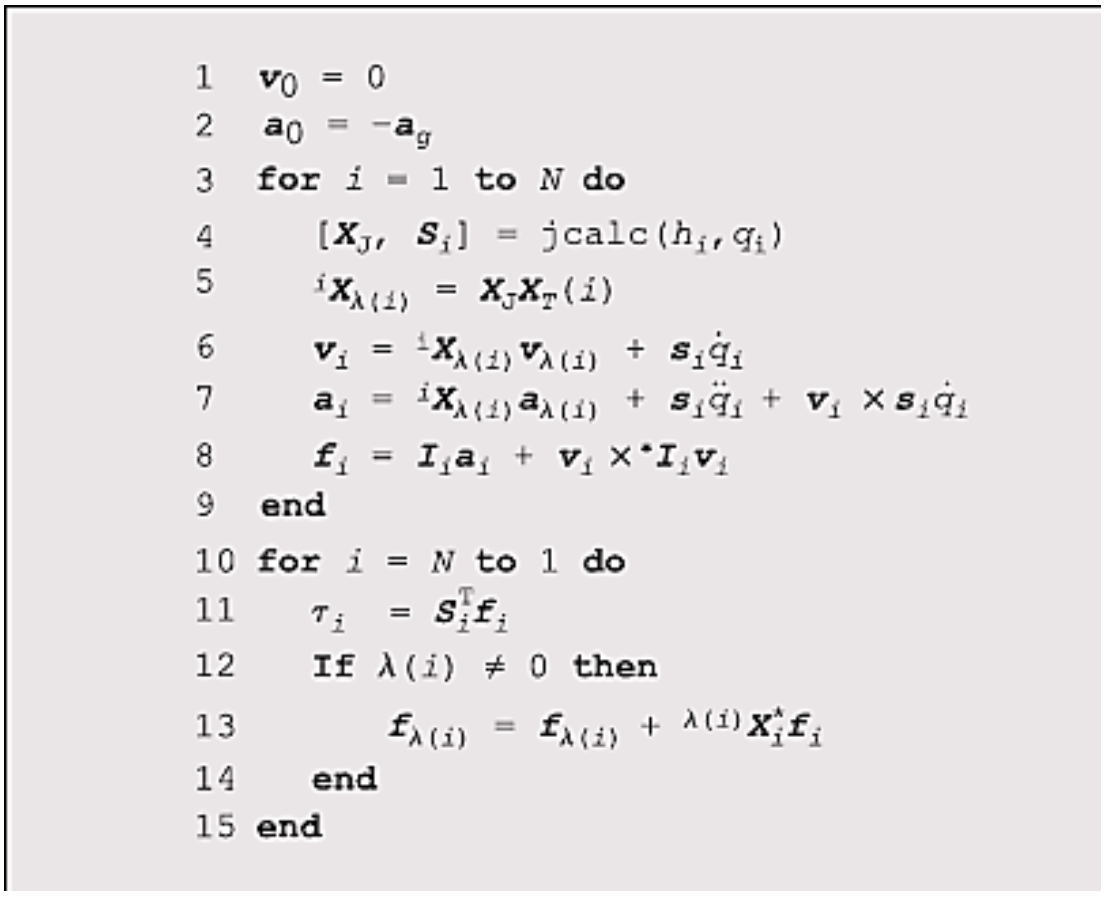


Illustration 53: The Recursive Newton Euler Algorithm Pseudocode. (R. Featherstone, 2010, p. 93)

A full explanation of the algorithm is probably best left to a reference on the subject. However a brief overview of the algorithm is presented below.

The algorithm works on a kinematic tree that represents N_B bodies and N_J joints where $N_B = N_J$ for a tree with no closed loops. The tree can be represented by a graph which shows the bodies as dots and the joints as linkages such as below:

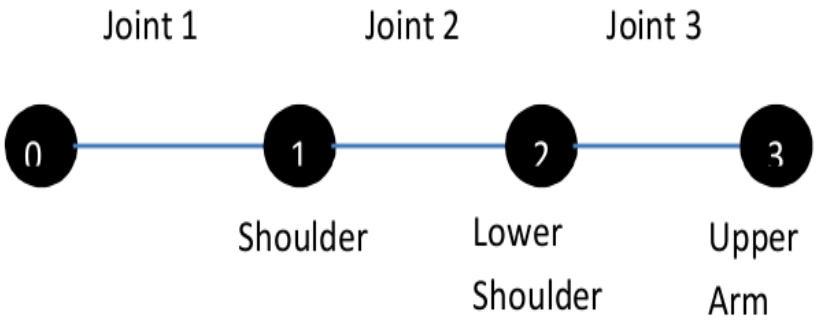


Illustration 54: Tree Graph of serial manipulator

In this representation the bodies are numbered 0 to N_B, with 0 being the base providing absolute reference. And the linkages are similarly numbered 1 to N_J such that the joint "1" links the body "1" to its "parent", 0. In a serial kinematic tree with no closed loops a body can have only one parent, but a body can have any number of children. In the algorithm the parent of Body *i* is *λ(i)*. Other variables such as X_J and X_T which are spatial transformation matrices, and I_i is the spatial inertia tensor (includes both angular and linear inertia terms). A graphical representation of this relationship is shown below.

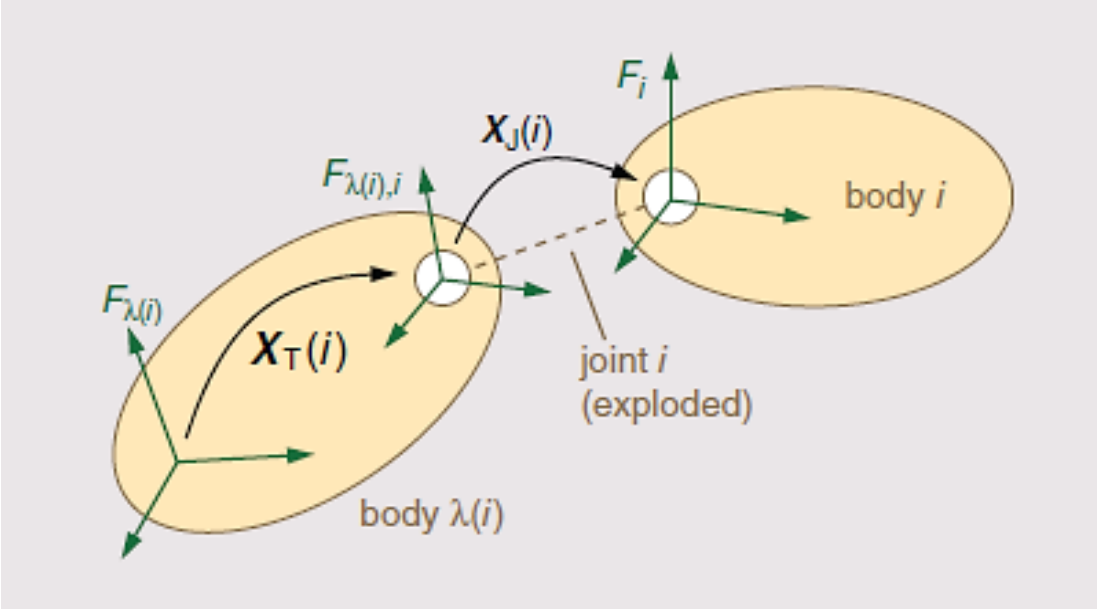


Illustration 55: Coordinate Frames and Transforms associated with Joint i. (R. Featherstone, 2010, p. 90)

Algorithm Flow

From the program flow it can be seen that the program first starts at the base and works its way out to primarily evaluate the positions, velocities and accelerations on the joints, which are in turn based on the parent's position, velocity and acceleration. As the program steps through this loop it is utilising data already evaluated in the previous iteration (for the body's parent marked with a subscript $\lambda(i)$). This is the recursive aspect of the algorithm and the reason for its efficiency and low complexity. The forces calculated in this loop are calculated for the forces on this body alone and include centripetal, coriolis and other external forces. Gravity was modelled by an acceleration of g in the negative z direction acting on the base therefore (Illustration 53, line 2) would be included in all acceleration calculations due to the recursive calculation of acceleration. However it should be noted that the forces are not recursively generated in this loop.

The second loop utilises the forces already generated in the previous loop. It works from the endmost body to the base and adds up the forces on each joint including the forces on the child(s) joint. This is where the force determination is recursive. The force for the joint is calculated in the six spatial dimensions of the body coordinates. This term is therefore useful in design of the joints as it includes all the loads that will be exerted on the bearings. The item τ_i isolates the force that is acting in the direction that the joint is not constrained. This will be the force that will load the actuator.

D.1. Incorporating the RNEA into Actuator Modelling

As the actuator modelling determines what angular joint states are achieved, and the RNEA simply takes those states and returns the torque on each joint, incorporating the joint torques into our actuator models was about extracting the information from one of the models and feeding it into the other to get solutions that would better model the situation as shown in Illustration 8. Firstly the Joint States of position, velocity and acceleration were substituted into the RNEA. The RNEA was computed and the Joint Torques were computed. These Torques were substituted into the actuator dynamics as an external force on the actuator. Note that in doing this, we used the Joint Servo with motor actuators, as the relationship between joint torque and actuator torque was more linear. Finally, the Joint Servo actuators were updated and new angular joint states were evaluated.

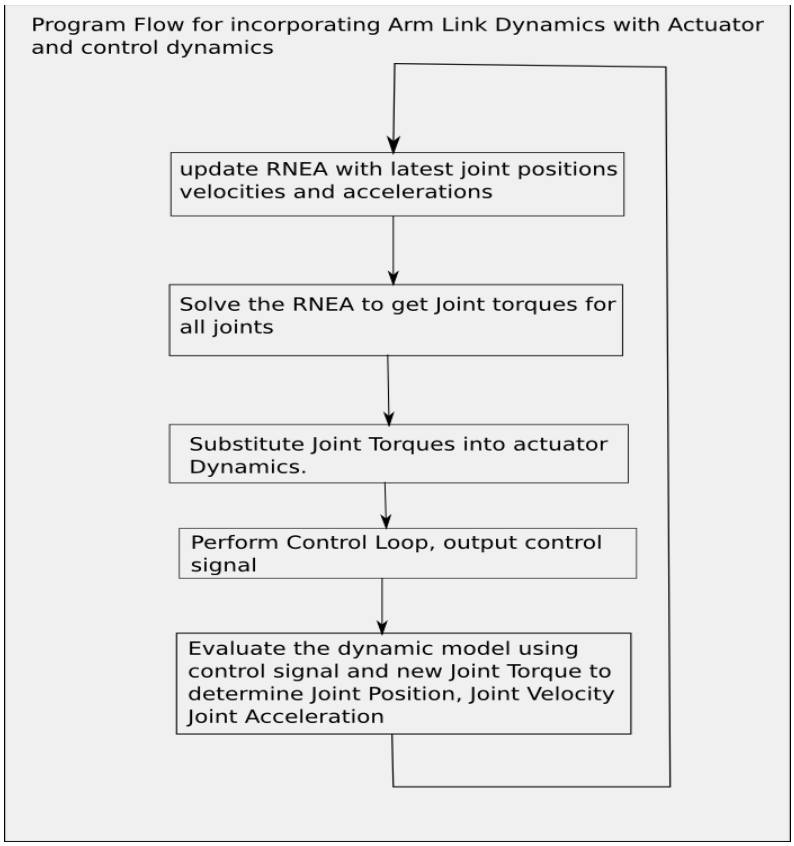


Illustration 56: Program Flow for incorporating RNEA dynamics into the Actuator and Control Loop modelling

Appendix E Custom Manipulator Arm Simulations

E.1. Visualising the Manipulator with Processing

Processing is an open source development environment and programming language that is supported by a large community of users to develop software that has a strong graphical context. The language is based on Java and it gives instant access to 2D and 3D renderers based on openGL (open Graphics Library) libraries for modern hardware-accelerated graphics. As a result it is easy to get into processing and develop visually complex programs, which was what was needed to start visualising this manipulator.

A screenshot from one of the processing simulations is shown in Illustration 24. It shows a model of the arm, floating above a workspace which consists of a ground plane, and three walls, painted with bitmaps. The forward kinematics are calculated to give the position of the tool in relation to the centre of the base.

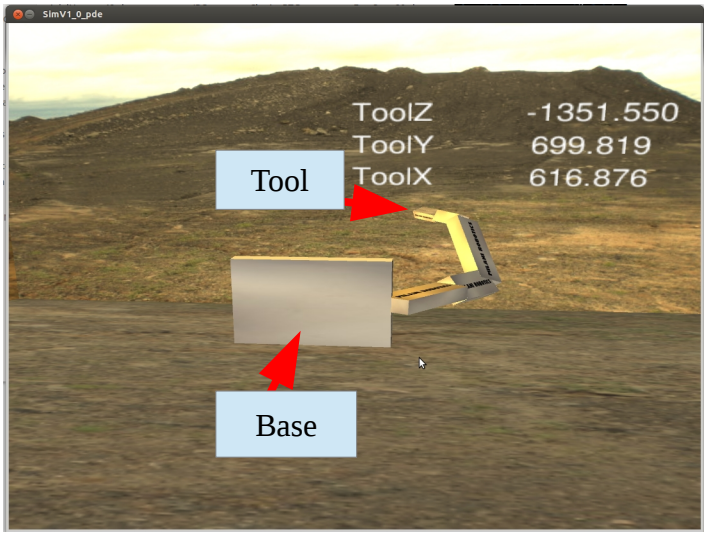


Illustration 57: Screenshot from Processing Simulation

There were about ten *Processing* simulation sketches developed throughout the project, prior to the ACARP funding. Some of the simulations received input from sensors, such as accelerometers over an RS232 interface, others used keyboard and mouse input to change the model. It is a very effective environment for small programs but can get a little difficult to handle when a program becomes much larger say, over 300 lines. Additionally debugging of errors is extremely difficult as there is no debug feature in processing to find errors in code.

One of the key aspects to test was the visual feedback interface was the effect of distributing cameras about the machine and using these as feedback to an operator, in separate viewports. Whilst this may have been possible using third-party libraries, there were difficulties in getting the libraries to work.

The models only had kinematics simulated and the jacobian evaluated. There was no dynamics of any type incorporated into the models. The forward kinematics was simulated using the jacobian to implement task space movement of the tool. Noting at this stage the arm was only a six axis representation of a human arm. The shoulder

had three joints, an elbow and a wrist that only had two joints.

Technically, six axes is the minimum required for a manipulator to assume any position in any orientation. Determining the end-position from the angles of the joints is termed Forward – Kinematics, and is relatively straightforward. Determining the joint angles required for a particular end-position is a bit more difficult and is termed inverse kinematics.

E.2. Visualising the Arm with Open Scene Graph

E.2.1. What is Open Scene Graph

Open Scene Graph is a commercial grade, open source, graphics simulation environment that is built on openGL but incorporates a number of abstractions so that users can concentrate on the simulation and sidestep the detail about the render. It enjoys widespread use across industrial, gaming and virtual reality industries(“Welcome to OpenSceneGraph website,” 2012). This essay will not go into the detail of OSG, as there are sufficient tutorials on the website and across the web to do that, but will instead just outline some of the fundamental concepts that were important to understand in the development of the simulation for the HMM.

The first key concept is that of a scene graph itself. The scene graph represents a number of *nodes* that are used to describe the scene to render. A node may be a drawable object (such as a geometric model of an aircraft), a primitive such as a pyramid, or a lightsource. Nodes also cover the various transformation objects, such as a transformation matrix or simply a scaling or translation object. The scene graph is represented by a tree, with the root world at the top, working through the transformations as the branches and the objects to be rendered as the leaves(“Scene Graph,” 2012)
of the tree. An example of a scene graph is shown below in Illustration 6.

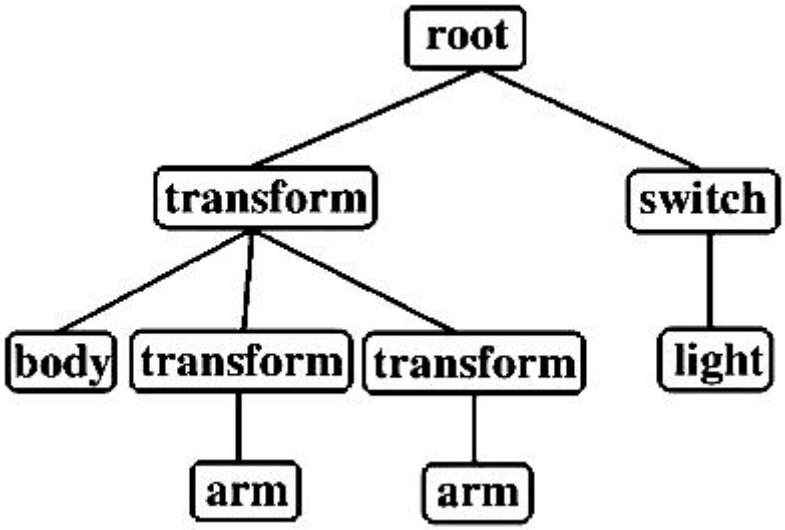


Illustration 58: A Tree representing a simple scene graph(“Scene Graph,” 2012)

Program Flow in OSG
Typically the scene graph is setup prior to render and is then rendered using the view command. To get the scene to actually do something, then the scene graph must be changed, usually by modifying one of the nodes. The view command incorporates a loop which continously calls various callbacks that must be specified prior to starting the viewer loop.

Two of the key callbacks OSG implements are an Update Callback and an Event Callback that are called at each frame render and traverses the graph from root to leaves. The Update callback gives the program the opportunity to identify each node and modify the node if required. This callback was especially useful for rendering the kinematics of the HMM arm by applying the new transform to the arm,[1] thus changing the visual scene in the render.

The Update callback is also useful for modifying user data such as the hydraulic actuator modelling, and the Arm Inverse Dynamic modelling using RNEA. To do this the classes and types of user data were applied to the UserData node of the root node. This node gets traversed like all the rest, and identifed by its class type, (determined by a successful cast to the ArmDataType), changes to the UserData and to the transformation nodes dependent on it were applied.(“Changing Models,” 2012)

The Event Callback was used to apply a keyboard input to the system. It works by detecting a keypress or mouse movement since the last call to the event callback. By interrogating the type of input into the system, the appropriate action can be taken. This appropriate action was communicated to the Update Callback (which operates separately) through a pointer to information within the Update Callback (“Update Callback,” 2012)
This was used to allow keystrokes to set the arm position for arm dynamics testing, and setting of the desired arm position for actuator modelling.

E.2.2. Key Concepts in OSG.

There were some key issues experienced whilst rendering to OSG, such as which way is up? The Default viewer will initially render a scene with z-axis up, x-axis to the right and y-axis into the screen. This corresponded well with the model which also has z-axis towards the sky, x- axis to the RHS and the front towards the positive y axis.

As already mentioned, normal transformation matricies have to be transposed prior to submitting them to an osg::MatrixTransform transformation. This was completed by incorporating a transpose() method with my matrixes which were of the osg::Matrixf type.

1A point to note was that the transform matrix that is used in OSG is the transpose of the transformation matrix usually encountered. (This appears to be a decision of the original OSG developer.)

E.2.3. Importing of geometric models in OSG

OSG primarily sources its drawable nodes from geometric models imported into the scene graph rather than developed from primitives. The import file type, .osg, can be exported from some CAD programs. As a result, and in order to utilise the productivity benefits of developing the models in a commercial modelling program, Blender™ with an .osg plugin was utilised in order to correctly reference the models exported from the CAD program as an .STL file, apply textures and then export from blender as an .osg file. The referencing in Blender consisted of ensuring the origin was in the correct place, (such as at the parent joint) and that the model was orientated to the global axes correctly. This ensured that the models would show up at the right place with the correct orientation, when the transformation matrix was applied to them.

Enabling Multiple View Ports with OSG

OSG enables the creation of multiple viewports which can display different aspects of the scene graph. These were enabled by defining an osg::view object, complete with Camera Object, including projection maxtrix, position and orientation, and adding these to the default osg::viewer, which renders all view objects. This allowed the rendering of a world scene, a head camera angle, and a hand camera angle. As shown in Illustration 6.

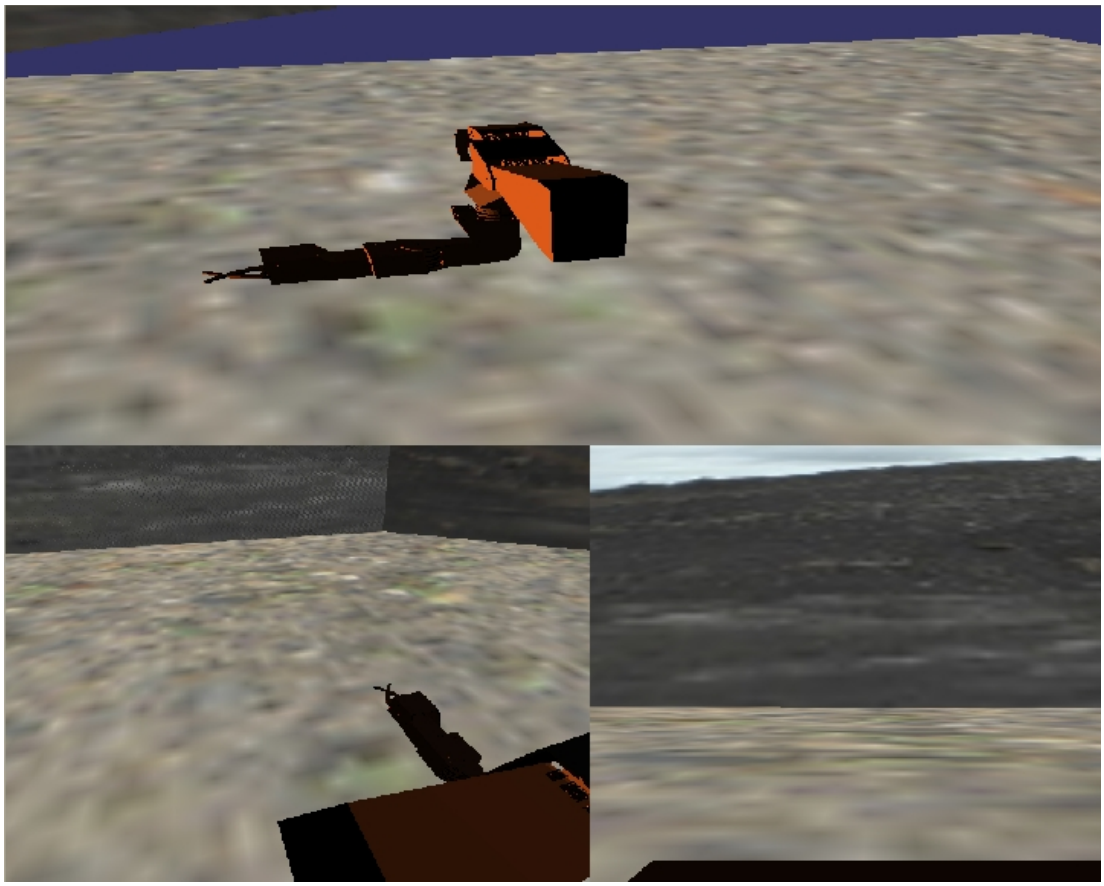


Illustration 59: Screenshot from OSG-based simulator showing three viewports, Overview (top), Head (bottom left) and Tool (bottom right)

E.2.4. Blender

Blender is another open source CAD modelling program that is primarily targeted toward animation. As such its mechanical design capabilities are almost zero in comparison to the latest professional CAD packages available. The interface is not intuitive as it has little resemblance to the typical user interfaces, but once it is mastered it can provide some powerful animation capabilities.

However for this simulation development Blender was used solely to reference models developed in a commercial CAD program to the appropriate origin and to ensure the orientation was correct. It was also possible to apply materials and textures to the models that would show up in OSG. Blender could import the .STL file developed in CAD, which is a vertex representation of the surfaces of the model, reference it, and export as an .osg file for use in a scene graph in OSG. This is how Blender was used in this simulation.

E.3. Conclusion

OSG simulation effectively rendered the scene and allowed interfacing with a variety of user interfaces. It also allowed dynamic simulation of arm actuators loaded by the forces produced on the arm. It was used to develop and demonstrate the human interfaces to be used for operator control and feedback. However because collision modelling was not implemented, it was difficult to accurately recreate a work task environment. This limitation was removed through the adoption of Robot Operating System as a software framework.

Appendix F Application of ROS and GAZEBO in simulation of manipulators

The dynamics of hydraulic servos has been examined, in order to test the effects of different control regimes. When the hydraulic servos are substituted into a serially linked arm on a manipulator, it important to know how controllable the arm becomes and whether a human operator can handle it. To answer this question and to provide a feedback mechanism for testing interfaces without a real arm, a variety of simulation environments were developed. Initially some custom environments were developed that incorporated arm dynamics. The methodology for implementing manipulator dynamics is listed in Appendix D There were two custom visual simulations developed, these are detailed Appendix E Finally, the custom simulations were dropped in favour of the GAZEBO 3D robotic simulator. A very sophisticated, open source simulator with dynamic and collision modelling, interfaces directly with Robot Operating System (ROS). It is a highly customisable environment and is suitable for all types of Robotic Simulation.

F.1. Robot Operating System

Robot Operating System (ROS) is a world class, open software tool set for developing complex code arrangements as collections of simpler programs. It was developed by the incubator company Willow Garage to drive their PR1, PR2 and TurtleBot platforms. It has since been taken up by many robotics companies as the software framework including traditional NC robot-building companies like ABB(“ROS Industrial,” 2013), iRobot and Rethink Robotics. It is an open-source software framework, whose development was transferred to the Open Source Robotics Foundation (OSRF) early in 2013.

ROS works by linking multiple small programs, known as nodes, via formatted messages over defined topics carried by any TCP network. Messages sharing information are published and subscribed to between nodes in a synchronous method. This allows powerful computers in a cloud network to interact and support mobile and remote robots with low internal computational power. The scope of ROS is immense and can support any conceivable architecture of intelligent robotics in development today. Best of all, open source and available for commercial application. This is inline with Willow Garage's objectives to make intelligent robotics development widespread (“Overview,” 2011).

ROS comes packaged with a 3D dynamics simulator known as GAZEBO. However the latest versions have split the two environments into standalone but mutually supporting programs. It handles a far greater range of simulation capabilities than was developed internally, as it includes a true dynamic environment for modelling of collisions and joint torques. The programming overhead for using GAZEBO is not high, as the same nodes that communicate to GAZEBO can be reused to communicate to actual hardware. The key to using GAZEBO is the robot model. This is described in an Universal Robot Description Format (URDF) , which is a ROS format used by a variety of other ROS packages. URDF use has recently been depreciated in GAZEBO, for Simulation Description Format (SDF) preferred as it

includes many features for building environments not available in URDF. Both are written in XML and are typically human readable. They describe everything relating to the model, including joint positions, collision models, 3D models, inertia and mass, sensors and plug-ins programs that allows software to be run on the simulated model.

F.1.1. ROS Message Framework

In ROS, nodes are linked together via topics and services. Each passes a formatted message, but fundamentally have different message flows. Over a topic there is usually one publisher and one or more subscribers. Messages are published over a topic at a set rate typically 1 – 100Hz, though the actual throughput, depends on the size of the message and the bandwidth of the link. Messages sent over a topic usually represent a stream of information, such as joint states, or video. Services on the other hand, are typically asynchronous and follow a request – reply sequence. The initiating node is the service client that issues the request that is processed by the receiving node which contains the service server. When developing code the names of the topics and services are used to control which nodes are talking to each other, as a particular message type may be used over a number of different topics or servers. ROS message traffic is not real-time, that is message publication may be delayed depending on the work load of the issuing computer, router bandwidth and message size, as a result it is good practice to ensure that time-stamps are applied to messages and servers, so that the temporal validity of the information can evaluated by the subscriber.

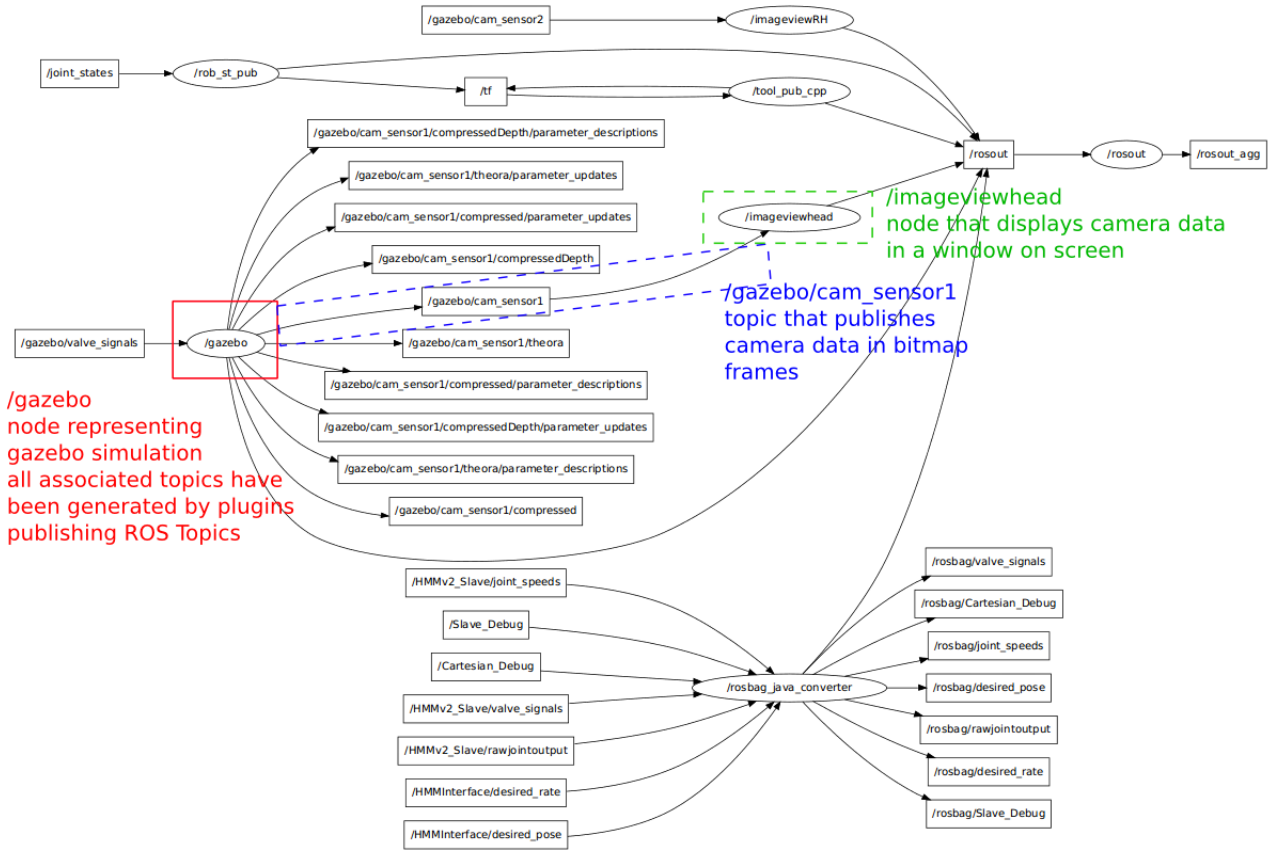


Illustration 60: Example of a ROS communications network showing all nodes (ellipses) all topics available (rectangles) and messages currently being sent (arrows)

Illustration 60 gives an example of a ROS network which includes a GAZEBO simulation. The ellipses represent software nodes and the boxes represent message topics. The arrows represent messages, so where two nodes are linked by arrows through a topic, illustrates a publisher/subscriber link. The direction of the arrows shows which node publishes a message and which node subscribes (consumes) the message. Service Servers and Service Clients are not illustrated here as their links are usually not permanent. Note that the */gazebo* node represents all the plugins that are attached to the models being simulated. In this case, it includes the program describing actuator dynamics and message interface of the slave manipulator arm, and two camera sensors, one of which is publishes to a node that outputs the image data to screen.

F.1.2. Developing Manipulator Dynamics

Developing a manipulator model with ROS and GAZEBO is a combination of describing the manipulator with a URDF file and writing a plugin program which will provide software control over the model. The plugin can communicate directly with the simulated model, whilst providing the ROS interfaces that external nodes can communicate with as shown in Illustration 60. Such a setup, allows a GAZEBO model to present the same interface as a real device, allowing external elements, in our case the Human Interface devices, to remain unchanged whether working with a simulated model or with a real manipulator. Whilst this could be achieved with a normal programming environment such as OSG, the standardised and modular nature of the software make it as easy to specify the output from a joystick, the joint state of a manipulator or streaming video from a camera sensor. It also supports distributing the software network over multiple computers and processing cores.

The key parts of the model to ensure the arm is correctly simulated involve:

1. specifying a visual reference for the links of the arm;
2. specifying the kinematic tree, in terms of joints and links which is taken directly from the kinematics already developed.
3. Specifying the inertia and centre of gravity values for the links of the arm.
4. Specifying the collision models for the links of the arm. To reduce computational complexity, collision models for most links are simply boxes around the links, except for the grippers where the collision model is a 3D object.
5. The gripper links (left, right and palm) are specified as a gripper object, which allows simplification of the modelling by treating gripper pickups not wholly through collisions but through the implementation of a joint between the gripper and object, given a certain number of collisions have occurred between the gripper elements and the object.
6. The plugin specifying the interface between the simulation and the ROS system, as well as actuator dynamic modelling of the hydraulic actuators.

F.1.3. Plugins

In the simulated model developed in GAZEBO, the ROS plugin did a number of things:

1. implemented the ROS Interface for the slave device;
2. interfaced actuator dynamics with the simulated model.

The ROS interface consisted of a subscriber and an a service server. The subscriber over topic */gazebo/valve_signals* of messages of type *motorcomd. Motorcomd* messages contained eight 32-bit float values which specifying the directional valve setting. They had a range from 0 to 1, with 0.5 specifying a closed valve, for each of the actuators including those on the gripper. This replicated the signal that will be applied to the proportional valve driver as a pulse width modulated signal.

The Service Server allows querying of joint position by a string and returns a list of joint positions expressed as integers ranging from 0 – 1023, with a zero position of 512. It forms the feedback link for the system control system. This is representative of the output that is presented from the 10 bit, absolute rotary encoders, and allows us to test how best to deal with the data presented from the sensors.

The plug-in interacts with the simulation by receiving a software object describing the simulated model's attributes. This gives the plugin access to the simulated model's attributes and provides functions to query and change them within the program. Objects representing the model's joints are extracted from this model object, and are used to query joint angle and velocity measurements and to apply joint torques. The same actuator software objects developed in the custom simulations are reused. Hydraulic motor objects are used as actuators due to the linear relationship between joint torque and motor torque. Velocity values are taken from the Joint objects and applied to the hydraulic motor models. Also applied is the valve spool position, taken from the */gazebo/valve_signals* topic and the actuator model is computed. The result of the computation is an actuator torque which is applied directly to the model's joint objects.

F.2. Conclusion

Use of GAZEBO and ROS represents worlds best practice in the use of simulation modelling. The result contains everything that is required to successfully test operator interfaces, to examine key aspects of the Manipulator design, particularly the control and software modelling, and range of movement and gripper design. GAZEBO's modelling of maintenance tasks provided a safe environment for the evaluation of the operator interfaces and led to very constructive feedback. In Part 4 our 3-dimensional dynamic simulation, powered by GAZEBO, will be used to support the development of the operator interfaces to be used with this manipulator, by providing a safe environment for testing.