UNIVERSITY OF SOUTHAMPTON

Iterative Learning Control Implemented on a Multi-Axis System

by

James David Ratcliffe

Thesis for the degree of Doctor of Philosophy

in the

Faculty of Engineering, Science and Mathematics

School of Electronics and Computer Science

June 2005

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING, SCIENCE AND MATHEMATICS
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

by James David Ratcliffe

This thesis concerns the implementation and comparison of different Iterative Learning Control (ILC) strategies on a multi-axis gantry robot. The majority of ILC research focusses on developing new algorithms for different classes of plant, then proving, by undertaking rigorous mathematical and simulation based studies, that the new algorithm will meet performance and stability requirements. The work presented here strictly concerns the performance of different ILC strategies on a physical plant by experimental methods alone, demonstrating that ILC can successfully be implemented in industrial applications. A test facility consisting of a three axis gantry robot and associated peripheral hardware is designed and built for this purpose. Four tests are developed to investigate key issues which are of particular importance to ILC implementation, minimum achievable tracking error, long-term stability, robustness to initial state error and robustness to plant modelling error. A rigorous framework for measuring quantitatively the performance of different algorithms is established to allow fair comparison. The experimental analysis is divided into two categories: basic and model-based algorithms. The work relating to basic algorithms initially uses a standard three-term PID controller to establish a benchmark performance level, to which the ILC performance can be compared. Combining a basic Proportional (P-type) ILC algorithm with the conventional PID controller to form a hybrid is found to increase significantly the performance, but at the expense of long-term stability. To remedy this, a logical progression of different filtering techniques, band-stop, low-pass, zero-phase-low-pass and a new frequency aliasing method are applied to the hybrid controller to steadily improve long-term stability and subsequently tracking performance. The work relating to model-based algorithms compares the performance of three previously developed, optimal ILC algorithms: adjoint ILC, inverse ILC and norm-optimal ILC. These algorithms operate on the plant input signal alone, without the requirement for an additional feedback controller. The three algorithms are found to produce significantly different tracking performance in response to disturbances and modelling error. The thesis concludes with an analysis of the tracking performance generated by each algorithm and a general discussion summarising algorithm attributes. For the plant used in this experimental work, it is found that the basic P-type algorithm has a slower convergence rate, but can achieve tracking error levels similar to more advanced model-based techniques.

Contents

List of Figures

List of Tables

Acknowledgements

Thanks to my family for their continuous encouragement and enthusiasm towards my research. Many a good discussion analysing the similarities between human and machine learning has proved invaluable for developing ideas on ILC implementation. Thanks particularly to my Mum, Dominique for proof reading the entire document. Of course, many thanks to my project supervisors, Paul Lewin and Eric Rogers and to the UK Engineering and Physical Sciences Research Council (EPSRC) for supporting and funding this research. Finally, special thanks to my colleagues Chris Freeman, Jari Hätönen and David Owens for their friendship and excellent advice pertaining to control systems.

Chapter 1

Introduction

1.1 Philosophy of Learning Controllers

Feedback control systems are used throughout industry in a multitude of forms and on a vast range of different tasks. Yet, no matter how well designed the feedback controller is, by the very nature in which it works, there will always be some element of error between the controller demand and the value of the actual variable being controlled as discussed in the ILC context by Amann, Owens, and Rogers (1996b); Sugie and Ono (1987). Although the number of applications is large, most control problems can in fact be classified as one of two different types; either trajectory tracking or set-point regulation (Dutton, Thompson, and Barraclough, 1998). In trajectory tracking, the controller is supplied with a varying input signal. The control objective is to make the output of the system under control (now referred to as the plant) follow the trajectory perfectly. An example is motion control of a construction robot manipulator. In set-point regulation, the input to the controller is constant and the control objective is to hold the output of the plant constant even if the plant or the surrounding environment is changing. Examples include constant temperature control of a heated room and constant speed control of a conveyor belt system carrying a variable payload.

Within both of these control types, there exists a sub-set of control problems which specifically have the nature of being cyclic or repetitive. In the trajectory tracking mode, this implies that the trajectory is of finite, constant duration and that, once the trajectory has been completed, it is supplied to the system again and this repeats ad infinitum. In the set-point regulation mode, the controller must compensate for a load which changes cyclically. If feedback control is applied to either of these special cases, the error signal generated for each cycle or trial will remain almost identical from repetition to repetition. However, Iterative Learning Control (ILC) and Repetitive Control (RC) have been designed specifically for these types of control problems. The underlying philosophy of these controllers is to find the input signal, which must be supplied to the plant,

in order to obtain zero error at the output. In effect, the input signal is progressively modified or learnt from trial to trial, based on the error information derived from previous trials, as discussed by Moore and Xu (2000). As the trial number increases, the error at each point in the trajectory is steadily reduced. Theoretically, as the number of trials tends to infinity, the error reduces to zero and perfect tracking is achieved (Kawamura, Miyazaki, and Arimoto, 1985). In practice, zero error is virtually impossible to achieve, due to the effect of random disturbances between trials. However, the error can be greatly reduced to provide a significant improvement compared to other controllers. The work in this document focusses on the implementation of ILC on the trajectory tracking type problems. The performance of different iterative learning controllers is evaluated in a practical environment and ILC is also compared to standard classical feedback control.

1.2 Motivation for Industry

There are many different areas in industry which involve tasks of a cyclic nature, for example: vehicle assembly, food processing, bulk manufacture of cheap goods and batch processing of chemicals. One factor common to all these industries is the need for the production or assembly system to operate as rapidly as possible. While a product is in the factory rather than out with the customer, the manufacturing and storage costs steadily increase and overall profits are reduced. Hence there is a serious incentive to mass produce goods as quickly as possible. Faster assembly lines also mean that less machines need to be run in parallel to produce the same quantity of goods per hour. Less machinery potentially implies reduced overhead and maintenance costs which are also very attractive to industry.

The main problems with trying to operate machinery faster are limitations on the physical parameters such as velocity, acceleration or temperature, which can be imposed on the system. In most cases, the machinery has already been designed to run at maximum rate, consequently no further increase can be obtained without significantly raising the risk of component failure. Component failure increases system maintenance or stoppage time and so negates the overall effect of running the system faster. One obvious solution to this problem is to upgrade or redesign the equipment so that it can run at a higher rate. However, this is costly and industry is reluctant to take this route. Bearing in mind the fact that existing machinery must be used, an alternative approach to increasing production rate is to modify the way in which the task is performed. This is where the control system becomes very important. If the control system can accurately follow the reference trajectory, it implies the dead-time allowed for the system to settle can be reduced (Barton, 1999). Reducing the dead-time shortens the overall time for one iteration, allowing more iterations to be performed in a set period of time, which translates into a faster production line. The reasons for initially allowing dead-time in the trajectory are twofold. Firstly, time must usually be allowed for the controller to

compensate for the transient dynamics of the plant and allow the system to reach steady-state. Secondly, the controller must reduce any steady-state error which may be present between the plant output and the reference trajectory, before critical elements of the manufacturing process need to be performed.

For example, consider a CNC milling machine which is tasked with drilling a set of holes through a steel plate as part of an assembly process. The rate at which the holes may be drilled is limited by the cutting rate of the drill bit through the steel. However, the time taken to reposition the drill bit for each hole is also a significant part of the overall machining time. The limitations on this part of the process are not only the velocity and acceleration of the machine, but also the performance of the positioning controller. The machine cannot begin drilling the hole until the drill bit is positioned within certain tolerance bounds. If a poorly designed controller is used, the initial positioning error and the transient dynamics induced by the rapid motion of the drilling head will take longer to reduce to within the tolerance band. Hence the dead-time must be longer than for a controller which can track the trajectory more accurately with less error.

The three-term, Proportional, Integral and Derivative (PID) controller is widely used throughout industry even though the initial concept dates back to 1936 (Franklin, Powell, and Emani-Nacini, 1994) and significantly more advanced controllers are now available. The PID controller has remained popular for many decades due to a number of factors which are discussed by Barton and Lewin (2000).

- There is a wide range of commercially available, low cost, PID controllers.
- The PID controller is very simple to tune and requires minimal set-up time.
- In certain industries the technology and mathematics behind more advanced controllers are not widely applied.

PID controllers are suitable for many industrial applications. However, their inability to compensate for non-linear effects makes the PID controller a poor choice in an age when requirements for accurate control are increasing (Li and Li, 1996).

Iterative learning control is a good candidate for improving tracking performance of dynamic systems and hence, has the potential to increase significantly the output rate and accuracy of industrial processes. Yet there is very little evidence of ILC actually being implemented in practice (Longman, 2000). There appear to be three main reasons for this. Firstly, the technology is not available in standard 'off-the-shelf' controllers such as Programmable Logic Controllers (PLC). Secondly, a majority of published research has concentrated on theoretical studies supported by simulation only. Thirdly, there are some problems with learning controllers, such as stability issues and robustness to non-repeating disturbances, which must be fully resolved before they can be used in industry (Kim, Chin, Lee, and Choi, 2000). Industry appears to be reluctant to implement

technology which has not been thoroughly tried and tested. One of the main objectives of this project is to demonstrate that ILC can successfully be implemented in industrial applications, in the hope that ILC will become more widely used.

1.3 Project Overview

The research undertaken in this thesis is part of a joint project, involving the University of Southampton and the University of Sheffield. The project has been sponsored by the UK Engineering and Physical Sciences Research Council (EPSRC) and aims to investigate a number of areas within iterative learning control and repetitive control. The project contains a number of work packages:

1. Design and construction of a multi-axis test facility.
2. Development and simulation-based evaluation of convergent predictive control algorithms using optimal control.
3. Design and analysis software to produce a range of ILC designs for comparison and refinement in simulations studies prior to experimental verification on the experimental test facility.
4. Nonlinear iterative learning by adaptive Lyapunov techniques.
5. Development of theoretical links between repetitive and iterative control paradigms using concepts of duality, Lyapunov methods and related techniques.
6. Experimental verification and assessment.

The fundamental goals of the project are twofold: One aspect is an analysis of existing iterative and repetitive control techniques, leading to the design of new algorithms by means of advanced control methodologies. The other aspect is an intensive, thorough experimental evaluation of new and existing algorithms, focussing particularly on comparing the performance of different techniques and investigating the robustness of controllers with respect to a number of known stability issues. The combination of theoretical and experimental analysis within one project also aims to bridge the existing gap between analysis and practical evaluation.

1.4 Research Objectives

This thesis is concerned with the experimental components of the larger project and focusses particularly on work packages 1, 3 and 6, contributing some developments to the other packages. The goals of this thesis are therefore more specific than the broad overview.

1.4.1 Experimentally implement ILC

The significant lack of practically implemented ILC is one of the main causes for the minimal application of ILC usage within industry. Simulations and mathematical analysis are vital elements of control system development. However, they are of little purpose if the resulting algorithms remain untested and unused. The difficulties associated with practical implementation, highlight areas where theory can be improved and where entirely new areas of research need to be developed. By thoroughly testing several new ILC algorithms, this project will add to the existing pool of experimental data, which can be employed to promote ILC in industry as a viable solution to high precision tracking control problems.

1.4.2 Overcome difficulties presented by multi-axis systems

The cartesian design of a gantry robot results in negligible cross-coupling between the axes, i.e. when one axis moves, the effect, measured on any of the other axes is minimal. The robot can therefore be treated as three Single Input Single Output (SISO) systems rather than one Multi Input Multi Output (MIMO) system, and the design of the control system is significantly simplified. However, three separate controllers must now be implemented simultaneously on one control platform. This poses several problems to be solved before successful implementation can be achieved. ILC is particularly demanding with respect to memory and processor speed, therefore implementation of complicated ILC algorithms must be carefully planned so as to fit within the limitations set by the control hardware. Maximising the use of controller resources may also allow the development of ILC algorithms which can be implemented on commercial control platforms, such as the Programmable Logic Controller (PLC). This would be a significant step towards achieving wide-spread use of ILC in industrial applications.

1.4.3 Develop a framework for comparing different algorithms

ILC is different from other control techniques, because the tracking performance changes as the number of iterations increases. Standard measures of performance are therefore not particularly suited to ILC. New performance techniques need to be considered specifically for ILC systems, in order to measure the key aspects which define and affect the performance of ILC.

1.4.4 Compare ILC to PID feedback control

PID feedback control is still the most popular control system used in industry. Comparing the performance which can be achieved by ILC to that achieved by PID feedback is

therefore essential, if the advantages and drawbacks of using ILC based systems are to be clearly determined.

1.4.5 Compare basic and model-based ILC techniques

The most basic ILC algorithm (P-type) uses the previous plant input vector, the previous error vector and a scalar gain to generate the next input vector. In comparison, model-based ILC requires a model of the plant and additional mathematical techniques to achieve the same objective. Regardless of the level of complexity of the algorithm, for SISO systems, ILC fundamentally generates a single input vector for each trial. The additional complexity of model-based algorithms requires significant extra resources in terms of plant modelling time, control system design time, processor speed and memory. This leads to the question: does the use of extra resources produce a control system which performs significantly better than simpler algorithms? An answer may be obtained by implementing both basic algorithms and model-based optimal algorithms on the same system and comparing their relative performance.

1.4.6 Develop algorithms which are long-term stable

ILC algorithms which become unstable are not an option for industrial applications. The loss of product and time caused by any instability is unacceptable. The causes of instability need to be investigated and possible solutions assessed.

1.4.7 Investigate the effects of disturbances

Initial state error and plant modelling error are often analysed in published ILC research by using simulation studies. However, no practical experimentation to date has actually investigated the effects of these problems on the performance of the algorithm. A systematic approach needs to be developed in order to investigate this aspect of ILC, so that future research can be properly directed at solving these problems.

Chapter 2

Literature Review

2.1 Origins of Iterative Learning Control

The subset of cyclic control problems defined previously can itself be divided into two smaller groups. Iterative learning control applies to one of these groups, while repetitive control applies to the other. As more research is performed in both these areas, an increasing number of links are being made between the two. However, the original approaches to the two control techniques were quite significantly different, as discussed by Longman (2000).

The recognised formal definition for iterative learning control was proposed in 1984 (Arimoto, Kawamura, and Miyazaki, 1984a,b), though Longman (2000) traces initial concepts back to an article published in Japanese by Uchyama (1978). Similar ideas were also independently developed in 1984 by Casalino and Bartolini (1984); Craig (1984). However, the articles proposed by Arimoto et al. (1984a,b) are the most frequently cited, so these will be discussed here.

The theory behind ILC is based on human learning. Learning can be defined as "changes in the system that are adaptive in the sense that they enable the system to do the same task more efficiently and more effectively the next time" (Oh, Bien, and Suh, 1988). Arimoto and Naniwa (2000) mathematically define concepts associated with the learnability of dynamical systems. Humans learn by practicing or repeating a task until it is perfected. During the learning process, mistakes are frequently made, yet a lot of useful information can be extracted from them. At each attempt or iteration of the task, one hopes to improve on the last iteration by using knowledge gained from previous attempts. Arimoto et al. (1984b) investigated whether this principle could be applied to machines, to give them some form of learning autonomy without the need for human intervention. At this point, it is necessary to define the main factor which distinguishes ILC from repetitive control.

- The initial conditions of the system are reset before each iteration commences.

This is particularly true for robotics applications, where the robot can be homed to a starting position before carrying out the required task (Arimoto et al., 1984a). It also implies that iterations do not need to follow each other immediately. The variable amount of time between iterations can be used to compute the next input sequence to the plant. With respect to practically implementing ILC, these characteristics have important consequences. By allowing the system time to reset between each iteration, the dynamics of the previous iteration are not carried over to the next iteration i.e. whatever happened at the end of the previous iteration does not directly affect the performance in the next. The pause between iterations allows even very complex algorithms to be implemented as there is no processing time constraint. In repetitive control, the initial conditions are not reset before each trial and there is no time between repetitions. The process is continuous, one repetition flows directly into the next and the final conditions from one repetition are the initial conditions for the next (Hara, Omata, and Nakano, 1985). This implies that data processing and computation of the next input to the plant must be continuous and performed on-line while the system is operating. There is therefore a limited time for processing complex algorithms between each sample instant.

The learning control solution which was proposed by Arimoto et al. was to use the error vector generated in the previous iteration to modify the input vector to the plant for the next iteration so as to reduce the future error vector. This was defined as

$$\mathbf{u}_{k+1}(t) = \mathbf{u}_k(t) + \beta(t)\dot{\mathbf{e}}_k(t) \tag{2.1}$$

$$\mathbf{e}_k(t) = \mathbf{r}(t) - \mathbf{y}_k(t) \tag{2.2}$$

where k is the iteration number, $\mathbf{u}_{k+1}(t)$ is the vector of inputs for the next iteration, $\mathbf{u}_k(t)$ is the vector of inputs for the current iteration, $\beta(t)$ is the learning gain matrix, $\dot{\mathbf{e}}_k(t)$ is the error derivative vector for the current iteration, $\mathbf{r}(t)$ is the reference trajectory vector and $\mathbf{y}_k(t)$ is the plant output vector for the current iteration. Vectors are used because the calculation to find the next input sequence can be performed off-line as a batch process in the time between iterations.

Having stated the algorithm, the conditions for convergence have been comprehensively studied. This can be understood by viewing the task as identifying which types of plant the algorithm can be used to control and also choosing a value of learning gain which is guaranteed to reduce the error at each iteration. As long as the convergence criteria are satisfied, theoretically the algorithm is guaranteed to converge towards zero error, as the number of iterations increases to infinity. If the convergence conditions are not met, it is highly probable that the algorithm will not reduce the error at the next iteration, but

will in fact increase it. This is known as divergence. Virtually all ILC algorithms, no matter how complicated, follow the basic format of Equation 2.1. The main variations depend on how the error vector is used to modify the current iteration input vector.

2.2 Algorithm Development

Since initial work began in iterative learning control, significant effort has been dedicated to developing new theories and algorithms. In general, the objectives of the research have included one or more of the following:

- Reduce the error over the whole cycle as close to zero as possible.
- Reduce the number of trials needed to achieve near zero error, i.e. faster convergence to zero.
- Develop algorithms for different classes of plants.
- Improve algorithm robustness with respect to modelling errors and random disturbances.
- Achieve convergence while maintaining intermediate trial performance.

Initial research looked mainly into using different combinations of proportional, integral and derivative error for learning and studied higher order systems which used data from more than one trial. The next stage in algorithm development was to include elements from other control areas into ILC. These included model based, optimal, robust and predictive approaches. These developments are reviewed in order to provide a comprehensive overview of the diverse range of research which has been performed to date. Case studies of specific algorithms are presented where applicable, to illustrate areas where there has been extensive research.

2.2.1 Basic algorithms

'Basic controllers' are iterative learning algorithms which require no knowledge of the plant. The controller is connected to an unknown plant and one or more parameters are adjusted until the desired performance is obtained. The algorithm developed by Arimoto et al. Equation 2.1 also fits into this category.

2.2.1.1 P-type

The Proportional or P-type algorithm (Arimoto, Kawamura, Miyazaki, and Tamaki, 1985b; Kawamura, Miyazaki, and Arimoto, 1988; Mita and Kato, 1985; Sugie and Ono,

1987) is very similar to the algorithm derived by Arimoto et al. (Equation 2.1) which is called the Derivative or D-type (Arimoto et al., 1984a,b; Arimoto, Kawamura, and Miyazaki, 1985a; Kawamura et al., 1985). It differs in that the error derivative used in the original algorithm has simply been replaced by the error. The next input sequence is therefore generated from the current input sequence and the current error multiplied by a learning gain. Arimoto et al. (1985b) proposed the modified algorithm:

$$\mathbf{u}_{k+1}(t) = \mathbf{u}_k(t) + \beta \mathbf{e}_k(t) \tag{2.3}$$

The main advantage gained from using the P-type algorithm is that it does not require differentiation to calculate the update. Differentiation in control systems has the potential to significantly amplify small noise signals. Even if great care is taken to screen signal wires and use noise rejecting electronics, in practice, there will always be some element of noise present in the control loop.

Although one may have expected the P-type algorithm to be developed before the D-type, this was not the case. The main reason was that, for some time, it was uncertain that the convergence proof for the P-type algorithm could be found. Sugie and Ono (1991) explain this by specifically concluding that the order of the error derivative must match the relative degree of the system. The systems used in early implementations of ILC were relative degree 1 and 2, and therefore it was necessary to use higher order derivatives. In discrete time systems, relative degree can be considered as a time delay between an input and the corresponding output (Jang, Ahn, and Choi, 1994).

2.2.1.2 D-alpha

D-alpha lies between P and D types. It uses the technique of fractional calculus to obtain a fractional derivative of the error signal (Chen and Moore, 2001). The algorithm is essentially D-type:

$$\mathbf{u}_{k+1}(s) = \mathbf{u}_k(s) + \beta s^{\alpha} \mathbf{e}_k(s) \tag{2.4}$$

where $0 < \alpha < 1$. When alpha is zero, the algorithm is P-type and when alpha is one, the algorithm is D-type. Fractional calculus has already been successfully implemented on standard feedback controllers, such as the three term controller, and has been found to improve performance. Applied by Chen and Moore (2001) to ILC, it is possible to view fractional calculus as a special type of filter which requires all of the historical error data. D-alpha ILC can improve the monotonic convergence of the learning algorithm.

2.2.1.3 Combined error derivatives and integrals

A further development from P and D-type algorithms is achieved by combining several terms together. P-D, P-I-D (where I represents the integral) and other combinations result. The objective is to use the features of each of these terms to produce an algorithm which can be applied to a wider range of plants and can improve convergence properties. A PID-type iterative learning controller has been successfully implemented by Kim and Kim (1996) on a CNC machine tool. The machine was required to cut circles of radius 29.7mm and depth 3mm from a sheet of aluminium. The tracking accuracy achieved for the first iteration was within 20.37 μm, while for the fifth iteration the error was reduced to 8.55 μm, a reduction of 11.82 μm.

2.2.1.4 Higher order

'Higher order' describes algorithms which use data from more than just one iteration (Chen, Wen, and Sun, 1997). For example, a second order algorithm may use the error data for the current iteration and the previous one. The theory behind using higher order algorithms is to make use of all the data which has been gathered since the process commenced operation. Theoretically, using more information should produce a system which is more robust to disturbances, initial state error, and can converge faster (Bien and Huh, 1989; Chen, Sun, Huang, and Dou, 1992; Chien, 1996). An example of a higher order algorithm is:

$$\mathbf{u}_{k+1}(t) = \mathbf{u}_k(t) + \sum_{n=0}^{N} \beta_n(t)\mathbf{e}_l(t) \tag{2.5}$$

where N is the system order, $l = k - n + 1$ and β_n is the learning gain matrix for the n'th iteration. The learning gain matrix need not be the same for each iteration. This allows for a more flexible algorithm, as older data can be penalised by using smaller gains. Increasing the order of the algorithm has also been found to allow convergence for systems with higher relative degree (Ahn, Choi, and Kim, 1993).

There is some debate as to whether higher order systems are more useful than first order. Sun and Wang (2001b) and Chen, Gong, and Wen (1998a) suggest that higher order systems, if designed properly, can achieve fast convergence speeds. Norrlöf and Gunnarsson (1999) also conclude that higher order algorithms can be found to be stable where first order algorithms are not. However, having performed a rigorous mathematical analysis of various order algorithms, Xu and Tan (2001, 2002) strictly conclude that first order systems converge fastest. Having compared a first order and a second order system through practical implementation, Norrlöf (2000) concludes that the second order system

does not perform better than a first order system. Another drawback to the second order system is that twice the amount of processing memory is required. An advantage may be that the second order system has potential to compensate for a plant which varies slightly between iterations by smoothing the data from more than one iteration. Overall, it is suggested that using the lowest order system is best, but higher order systems can be used if necessary to improve control of nonlinear plants.

Xu, Chen, Lee, and Yamamoto (1999) have designed a successful higher order learning controller specifically for systems where the controlled variable cannot be measured until the end of the process. This is termed terminal iterative learning control and has potential for application in microchip manufacturing processes.

Moore and Chen (2003) propose the use of a higher order ILC algorithm which operates in both the time domain and the iteration domain. The higher order algorithm achieves monotonic convergence in the time domain and can also compensate for iteration dependent disturbances in the iteration domain. For example the controller can compensate for disturbances which are repeated every two iterations rather than every iteration.

2.2.2 Variations of basic ILC

2.2.2.1 Hybrid controllers

'Hybrid' describes the combining of two different systems which work together towards a common goal. In many of the cases where ILC has been implemented on real physical systems, the learning law has been coupled to a standard feedback controller (Barton, Lewin, and Brown, 2000; Havlicsek and Alleyne, 1999; Longman, 2000; Tayebi, 2004). The objective is to improve the robustness of the controller (Doh, Moon, and Chung, 1999; Moon and Chung, 1998). The feedback controller can compensate for non-periodic disturbances, while the learning controller reduces the periodic disturbances (Kuc, Nam, and Lee, 1991).

Another advantage of using a hybrid system is that the learning controller need not operate all of the time. Once the error has been reduced to within tolerance bounds, the learning controller can be switched out and the feedback controller will continue to operate the system with the same level of accuracy (Barton et al., 2000). The learning controller can then monitor the error and can be switched in again, should the error begin to increase (due to component wear, for example). Two different arrangements of hybrid controller have been investigated. In one case, the learning controller modifies the output of the feedback controller (Jang, Choi, and Ahn, 1995; Kuc, Lee, and Nam, 1992; Moon, Lee, and Chung, 1996), while in the other case, the learning controller modifies the input to the feedback controller (Liang and Looze, 1993). Both types can be shown to satisfy convergence criteria (Longman, 2000).

2.2.2.2 Current iteration error

The first algorithms to be developed generated the next input sequence to the plant, from the input sequence and the tracking error obtained during the previous trial. This made these algorithms completely feed-forward by not taking into account the error of the current trial (Chien and Liu, 1996). The overall effect was to create a system which was not robust to random disturbances. To counteract this, it is possible to use a feedback controller in a hybrid arrangement which provides the necessary feedback element. An alternative solution is to use the Current Iteration Tracking Error law (CITE) (Owens, 1992).

$$u_{k+1}(t) = u_k(t) + (\beta e_{k+1}(t)) \tag{2.6}$$

where e_{k+1} is the current iteration error. In this example, the algorithm must be computed at each sample interval, otherwise it is fundamentally non-causal. It can be demonstrated mathematically that Equation 2.6 will converge for relative degree one MIMO systems with minimum phase. It can also be shown that the algorithm can tolerate a high learning gain which leads to faster reduction in tracking error (Chien, 1998). The CITE gain has also been found to have a direct influence on the final tracking error bound (Chen et al., 1997; Chen, Xu, and Lee, 1996c). Owens and Munde (2000) suggest that using the current error is beneficial for three main reasons.

- The most recent data reflects the current performance of the system.
- Current error feedback could stabilize unstable plants.
- The effects of noise and modelling errors can potentially be reduced.

2.2.2.3 Anticipatory

Some learning control algorithms use the concept of anticipatory control. The concept is explained in detail by Wang (1999) and can be found to apply specifically to the P-type basic ILC algorithm. In the original D-type algorithm, the control effort update, at one particular sampling interval, consists of the control effort and the tracking error derivative from the previous iteration at the corresponding sampling interval. For the D-type algorithm, this approach is correct, as the different terms do correspond exactly. However, for the P-type algorithm, the approach no longer applies. The P-type law does not capture the trend of the error from previous sampling intervals. At one particular sampling interval, the error may be recorded as zero. However, this does not guarantee that it was zero for the previous sample. In fact, the error gradient may be significantly different. The basic P-type law does not make use of this information.

Wang (2000) proposes a solution to this problem by using an anticipatory scheme. When generating the new input sequence, instead of using the error from exactly the same sample instant one trial back in time, the error from one of the following sample instants is used. Note that the shift in samples is only a small number and is usually set to 1. This approach can generate a concise mathematical proof for convergence (Wang, 1998) and has been found to improve the rate of error reduction in simulations (Ma, Low, and Tso, 1993). Sun and Wang (2001a) show that the anticipatory system can also relax the convergence conditions for systems with higher relative degree. Wang (1999) tested the anticipatory algorithm on a two-link SCARA robot system and obtained good convergence results. For a 15 iteration test, the root mean square (rms) error is reduced by approximately one order of magnitude after only one iteration. Notably, the error reduction is non-monotonic and the rms error increases between iterations 5 and 10, following the initial convergence. This is a possible indication of instability. Equation 2.7 is higher order and contains both an anticipatory and a current iteration error term.

$$\mathbf{u}_{k+1}(m) = \mathbf{u}_k(m) + \beta_1 \mathbf{e}_k(m+1) - \beta_2 e_{k+1}(m) \tag{2.7}$$

Where m is the sample interval, $\mathbf{e}_k(m+1)$ is the anticipatory error term for the last iteration and $e_{k+1}(m)$ is the current iteration error.

2.2.2.4 Forgetting factor

In general, a forgetting factor is a gain with magnitude less than one, as discussed by Wang (1995). The forgetting factor can be applied to any term of an algorithm to reduce the effect of that term on the input update. It has been used on the previous iteration input term $\mathbf{u}_k$ in an attempt to reduce the effect of high frequency noise and initial state error which can otherwise be amplified around the learning loop (Chien and Liu, 1996). Using the forgetting factor reduces the noise transmission and increases the noise immunity of the algorithm. This effect becomes obvious when considering the Nyquist plot of a system. In the cases where the poles of the system are very close to the unit circle, the forgetting factor has the ability to draw the poles away from the unit circle and reduce the risk of instability (Lewin, 1999).

2.2.2.5 Time delays

Time delays are very common in control systems. They vary in magnitude from milliseconds and less, to hours or days. Longer time delays are frequently encountered in chemical batch processing plants. But, delays of a few seconds can also be encountered

in other areas of industry. A time delay is described as the period of time which elapses between a change in plant input and the corresponding change in the plant output.

One basic approach to compensating for this time delay is to use a delay shift (Park, Bien, and Hwang, 1998). The basic P-type algorithm can be modified by adding the delay shift to the error term.

$$\mathbf{u}_{k+1}(t) = \mathbf{u}_k(t) + \beta \mathbf{e}_k(t + \tau_e) \tag{2.8}$$

where τ_e is the estimated time delay. This algorithm is successful as long as the time delay is correctly estimated. However, if the estimate is incorrect, there is no mathematical guarantee that the algorithm will converge. Using an incorrect time delay has potential to completely destabilise the system.

An alternative solution is to implement an ILC algorithm alongside a feedback controller equipped with a Smith predictor (Xu, Hu, Lee, and Yamamoto, 2001). The feedback controller and Smith predictor stabilise the plant and remove the time delay, while the learning algorithm improves the tracking accuracy. Again, an accurate estimate of the time delay and the plant model are required for the Smith predictor to work successfully. Hu, Xu, and Lee (2001) use a learning controller coupled with a feedback controller and a Smith predictor. The ILC modifies the output of the feedback controller and the signal is then fed into the Smith predictor.

2.2.2.6 Reduced sampling frequency

Implementation of learning controllers requires some form of data storage memory which holds set-point and error values from previous trials. Hence, learning control is invariably implemented in discrete time, using a microprocessor. In discrete ILC systems, the sample frequency can have an effect on the algorithm convergence. Hillenbrand and Pandit (1999) derive the convergence conditions for a P-type anticipatory system. It is found that the algorithm can behave erratically in the first few iterations if the learning criterion is simply to reduce the error as the number of iterations increases. A more successful algorithm can be produced if the criterion is to reduce the error norm at each iteration. Unfortunately, using the error norm has a greater tendency to violate the conditions for convergence and the algorithm is more likely to diverge. However, by reducing the sample frequency, the system parameters can be adjusted until the convergence conditions are satisfied in which case the algorithm will converge.

A reduced sampling frequency has also been found to limit the impact of initial state error. This is because a larger sampling time naturally averages the control output and reduces the occurrence of short-duration step inputs. At high sampling frequency, the controller can respond to initial error and transients with a series of step inputs, with significantly different amplitudes, which induce transient plant behaviour. At low

sampling frequency, if the output at the first sample interval is chosen correctly, by the second sample interval the tracking error will be small or absent. Drastic control adjustments will no longer be required and the algorithm will continue to learn correctly (Hillenbrand and Pandit, 2000).

2.2.2.7 Selection of learning gain

Irrespective of the complexity of a learning algorithm, the fundamental principle behind ILC algorithms is to update the next input sequence to the plant, by using data obtained from previous input sequences and some measure of the output tracking error, which is multiplied by a learning gain. The choice of this learning gain is therefore an issue of great importance as it has a fundamental effect on the convergence and stability of the algorithm (Glower, 1997; Hwang, Bien, and Oh, 1991). In the most basic algorithms, the learning gain is chosen by the designer. In more advanced algorithms, the selection process is automated to some extent. Moore (1998, 1999) suggests a method which can be used to select the learning gain of a simple ILC controller to obtain perfect tracking within four trials. Through calculation, simulation studies and practical implementation, there are certain trends which can be identified which relate the choice of learning gain to the performance of the learning controller. High learning gain produces faster reduction of the tracking error from trial to trial, but the steady state error is larger, while low learning gain produces a slower reduction of the tracking error, but the steady state error can be reduced further. This is because low learning gain causes less amplification of noise.

2.2.2.8 Coefficient tests for convergence

It is important to determine whether a learning control scheme will be stable, when implemented on a plant. A simple method using only the parameters of the plant and controller would be particularly useful, as it would allow initial controller designs to be evaluated and modified rapidly. Judd, Hideg, and Van Til (1991) propose such a technique. The concept is similar to that of the Routh stability test for standard feedback control systems. A characteristic polynomial is generated from the characteristics of the plant and the controller and is then tested against a set of conditions in order to determine whether the complete system will be stable.

2.2.2.9 Non-periodic repeating disturbances

ILC is traditionally associated with removing repeating disturbances from periodic systems. However, it is possible to construct an ILC style controller capable of suppressing repeatable disturbances triggered in a random fashion, for example: vibrations and

shocks in mechanical servo systems. This is an unusual application of ILC, because the reference trajectory is continuous and does not need to be repeating. The majority of control is performed by an appropriate feedback controller and the ILC element is only used when a repeating disturbance occurs. The appropriate moment to switch on the ILC is selected by a Likelihood Ratio Test which detects the start of the characteristic error waveform, associated with the disturbance. Each time the disturbance occurs, the ILC is therefore supplied with the tracking error data until the end of the repeating disturbance. The ILC can therefore learn the signal which must be applied during the disturbance to minimise the effect on the plant output. Tousain, Boissy, Norg, Steinbuch, and Bosgra (1998) have successfully simulated this type of system with the aim of rejecting shock disturbances for a hard disk drive. The format of the learning controller is represented by:

$$u_{k+1}(t) = F_1(u_k(t) + F_2 e_{k+1}(t)) \tag{2.9}$$

where F_1 and F_2 are appropriately designed filters.

2.2.3 Model-based algorithms

'Model-based algorithms' include any controller which requires prior knowledge of the plant. For these controllers to be implemented, it is necessary to generate some form of model, which describes the behaviour of the plant. This is then used either during the design process, or as part of the ILC algorithm itself.

2.2.3.1 Model based controllers

The principle behind model based controllers is to obtain an accurate model of the plant which can be used in parallel with the real plant to provide information to the controller which would otherwise not be available (De Roover and Bosgra, 1997). The model is particularly useful for providing information about unknown states, or states which cannot easily be measured. As well as feeding the control input into the real plant, it is also supplied to the model. If the model is accurate, the output of the real plant and the model should be the same. If there is a difference between the two outputs, the error can be used to update the model to make it more accurate.

Phan and Frueh (1996, 1999) propose a novel way of implementing a model based controller by learning the plant model during each iteration. As the reference trajectory is constant for each iteration, it is only necessary to learn the dynamics of the plant for this trajectory as other dynamics will not be excited. Each time an iteration is performed, a set of data for the input and the corresponding output is obtained. Using this data, a set of basis functions can be trained to emulate the behaviour of the plant and generate a

model. The model is then used for standard model based control. Effectively, this is an on-line plant identification technique. Phan and Frueh (1999) implemented this model based controller on an experimental apparatus, consisting of a number of parallel steel rods held together by a thin spring-steel wire. Actuation force was supplied to one rod, while the tip of another rod was required to follow a set trajectory. The model based controller successfully reduced the tracking error by over one order of magnitude.

2.2.3.2 Inverse plant models

The input-output relationship of a plant can be represented by Equation 2.10. If the inverse of the plant model can be derived (Equation 2.11) it can be used to directly calculate the exact input sequence which must be supplied to the plant to obtain the desired trajectory (Xu and Ji, 1998).

$$y_k(t) = G^0(u_k(t)) \tag{2.10}$$

$$u_k(t) = -G^0(r(t))^{-1} \tag{2.11}$$

where G^0 is the true non-linear model of the plant, $G^0(\cdot)^{-1}$ represents the inverse of $G^0(\cdot)$ and $r(t)$ is the reference trajectory. With this scheme, there is no requirement for any control system. However, model inaccuracies and random disturbances are not taken into account. Model inaccuracies generally arise because a linear model is used to describe a non-linear plant. Relying upon the inverse model technique is therefore not usually suitable for implementation on real systems. The inverse model can however produce a good estimate of the signal which needs to be supplied to the plant. Combining the inverse model with a learning controller allows learning control to compensate for model error (Markusson, Hjalmarsson, and Norrlöf, 2001).

2.2.3.3 Robust control

It is often possible to design a control system which will be well matched to a plant model and should theoretically provide performance within tolerance bounds. However, it is not evident that this controller will be able to cope with disturbances, variations in the plant and modelling errors. For example, it is accepted that the model is only a representation of the real plant and that it will not be perfectly accurate (De Roover and Bosgra, 2000). Yet, standard design procedure uses the model to generate the controller. The resulting controller may in fact be poorly matched to the actual plant. Under steady state conditions, this may not be noticeable. However in transient conditions the control system may become unstable. Robustness analysis can be used to establish whether

the performance of a complete system will remain within the design specification, by determining how tolerant the system is to disturbances. It is therefore necessary to design into any controller a measure of robustness, which will allow the plant to vary to some extent without the controller being significantly adversely affected.

One definition of robustness in a learning control sense is that the error sequence generated during a trial remains bounded when bounded noise is present in the system (Sogo and Adachi, 1996). Of particular interest is the magnitude of the bounds on the final steady state tracking error, as this determines to what extent the error can be reduced (Chen, Xu, and Lee, 1996b). Liang and Looze (1993) specifically derive and define robustness conditions which are specific to ILC systems. They also determine that, in the presence of modelling error, the complementary learning sensitivity must be small for input frequencies at which the model uncertainty is large. Due to the inability of learning controllers to compensate for non-repeating disturbances, it is essential to incorporate robust control techniques when designing learning control systems (Xu and Viswanathan, 2000; Xu, Viswanathan, and Qu, 2000).

Sensitivity is an important measure in robustness analysis, as it describes how one component or variable of a system is affected by a change in another component or variable. High sensitivity implies that a small change of one variable has a large effect on others. Therefore low sensitivity is generally desired when designing a robust controller.

Tayebi and Zaremba (2003) propose a generic approach to designing robust learning control systems. By using a hybrid combination of a standard feedback controller in parallel with a learning controller, stability is theoretically guaranteed, as long as the feedback controller meets a specified robustness performance condition. If this is true, then a simple calculation leads to a performance weighting function, which is applied to the input of the learning controller and guarantees overall stability.

2.2.3.4 Adaptive controllers

When a control system is initially designed and commissioned, the controller should be finely tuned to match the plant and should provide control within maximum error bounds. However, as the plant and the surrounding environment change with time, the tuning of the controller will no longer match the dynamics of the plant and the accuracy of the control will degrade. Adaptive control is concerned with monitoring the performance of the controller and adjusting its parameters to compensate for changes in the plant and maintain the performance of the overall system. From this definition, it is possible to conclude that learning controllers are, by their very nature, adaptive. However, in learning control, the whole signal is adapted, while in adaptive control, only the parameters of the controller are changed (Norrlöf and Gunnarsson, 2001). Learning controllers have an advantage over conventional adaptive controllers because the tracking

accuracy can be improved at each trial. With conventional adaptive control the tracking accuracy does not improve, it remains constant (Poo and Ma, 1995). Conversely, adaptive control is well suited to compensating for non-repeating variations in plant dynamics. Combining an iterative learning controller and an adaptive controller can therefore use the advantages of both systems (Choi and Lee, 2000).

Hätönen, Owens, and Moore (2004) clearly define the difference between a time invariant ILC law and an adaptive ILC law. This can be summarised by comparing two versions of the basic P-type anticipatory ILC algorithm.

$$\mathbf{u}_{k+1}(t) = \mathbf{u}_k(t) + \beta \mathbf{e}_k(t+1) \tag{2.12}$$

This is the time invariant version of the algorithm, compared to

$$\mathbf{u}_{k+1}(t) = \mathbf{u}_k(t) + \beta_{k+1}(t)\mathbf{e}_k(t+1) \tag{2.13}$$

which is the adaptive form. Note that the learning gain is not a constant and is updated for each iteration by another update law which must be selected appropriately.

Case study French, Munde, Rogers, and Owens (1999) propose a simple adaptive ILC algorithm which allows modification of the learning gain from iteration to iteration by means of a separate gain update algorithm. A similar technique has also been investigated by Owens and Munde (2000). The algorithm developed by French et al. is suitable for Linear plants. The main control input update algorithm follows the standard P-type format, using the current iteration error variation.

$$u_{k+1}(t) = u_k(t) + (\beta e_{k+1})(t) \tag{2.14}$$

While the learning gain β is updated between iterations by

$$\beta_{k+1} = \beta_k + c\|\mathbf{e}_k\|^2 \tag{2.15}$$

where c is, in turn, the learning gain adjustment gain. Effectively, the gain is also adjusted in an iterative manner by using the previous iteration gain term and a function of the overall error in the previous iteration. As the number of iterations heads towards infinity, the iteration error will converge to zero and the learning gain β converges to a final value which is not infinity.

The basic operation of the algorithm can be reasoned quite logically. When the tracking error is large, a smaller learning gain is required, because multiplication with the error results in a large change in control effort. However, once the error becomes small, using a lower learning gain generates a very small change in control effort. The learning ability has effectively been reduced. In the adaptive algorithm, the learning gain is gradually

increased as the error reduces so that, even with small error values, the ability to learn is not reduced and the algorithm can continue to reduce the error further. As the algorithm reaches zero tracking error, the learning gain converges to a non-zero value.

In algorithms which are unable to adapt, the selection of learning gain is a compromise between convergence rate and steady state tracking error. With the adaptive algorithm, both fast convergence and low steady state tracking error can be achieved.

2.2.3.5 2-Dimensional repetitive process analysis

Learning control algorithms are generally developed by considering the plant and the controller separately. This implies that a transfer function for the overall system is never developed. Evaluation of learning gains is through trial and improvement rather than through calculation. 2-dimensional analysis proposes several advantages over the traditional approach (Owens, Amann, Rogers, and French, 2000).

- 2-D theory offers a mathematical model to describe the entire process dynamics.
- 2-D stability theory provides a useful method to show convergence.
- Proof of the stability of a 2-D learning system will guarantee convergence.

2-D analysis is particularly applicable to learning control systems because they are naturally of a two dimensional configuration (Kurek and Zaremba, 1993). The dimensions can be considered as:

1. Time during an iteration.
2. The iteration number.

Systems of this form can be represented in discrete state space, by a local state vector of the form:

$$x(i,j) = \begin{bmatrix} x^h(i,j) \\ x^v(i,j) \end{bmatrix} \tag{2.16}$$

Where x^h and x^v are the horizontal and vertical state components and i and j are non-negative integer-valued horizontal and vertical coordinates. With respect to learning controllers, h and i represent the sample or time interval during an iteration while v and

j represent the iteration number. The ILC algorithm developed by Arimoto et al. can be concisely represented in 2-D system theory (Geng, Carroll, and Xie, 1990).

$$\mathbf{u}(i, j+1) = \mathbf{u}(i, j) + \Delta\mathbf{u}(i, j) \tag{2.17}$$

Where $\Delta\mathbf{u}(i, j)$ represents the modification of the update which, in the Arimoto equation, is derived from the error and a learning gain. The most important aspect of using 2-D systems theory is that a framework for analysing the stability of 2-D systems is already well established and can be applied directly to learning controllers (Galkowski, Rogers, and Owens, 1999). Lee and Lee (1993) have used 2-D systems theory to develop an ILC controller with feed-forward and feedback elements for use on a Video Cassette Recorder (VCR) servo system.

2.2.3.6 Optimal learning control

Optimal control is concerned with providing the 'best' solution to the control task. The solution is specified by a criterion to be optimised, which is known as the cost function. The cost function must be formulated by the designer and must describe a curve with a minimum point. In most learning control algorithms, the cost function includes a description of the tracking error, so that the optimal controller attempts to reduce it to a minimum (Chen and Fang, 2004; Gunnarsson and Norrlöf, 1999). Many strategies implement a descent gradient approach for the minimisation process (Togai and Yamano, 1985). The algorithm drives the system towards the minimum point by calculating the gradient of the cost function and then taking a step in the direction of this gradient. In some implementations, if the gradient is not heading towards the minimum point, the step is made very small or reduced to zero, because taking a step in this direction would in fact be diverging away from the minimum point. Other implementations accept that, on average, the gradient will lead towards the minimum point and therefore take the same size step, irrespective of the outcome. This implies that, depending on the style of algorithm, the size of the step can be chosen automatically by the algorithm (Amann, Owens, and Rogers, 1996a), or can be fixed by the designer.

Optimization based ILC algorithms represent some of the most computationally intensive and complicated algorithms developed to date. Yet, it is interesting to note that in the majority of cases, the fundamental structure of the learning algorithm is still based on the previous input term, together with some form of gain or filter multiplied by the error sequence from either the previous, or the current iteration. Where the optimal ILC algorithm differs from other ILC formats is that the choice of learning gain is optimally calculated by minimising the cost function (Saab, 2003). Often, the gain is in fact a

matrix or vector of gains. The performance of the resulting algorithm therefore depends fundamentally on appropriate selection of the cost function.

An example of a Linear quadratic optimal learning control algorithm has been developed by Frueh and Phan (1998, 2000). Hatzikos, Hätönen, and Owens (2004) provide a useful summary of optimal ILC development, before investigating the use of Genetic Algorithms (GA) to develop an optimal controller which can compensate for nonlinear plants.

Case study The optimal ILC algorithm developed by Amann, Owens, and Rogers (1995) is not simply aimed at optimally reducing the tracking error. The algorithm has three properties, the first of which is to reduce the norm of the tracking error at each iteration. The second is to choose automatically the step size of the cost function gradient and the third is to improve robustness by using causal feedback from the current trial and feedforward data from the previous trials. The input sequence to the next iteration is found by solving the minimum norm optimization problem

$$\mathbf{u}_{k+1} = \arg\min_{\mathbf{u}_{k+1}} \{J_{k+1}(\mathbf{u}_{k+1})\} \tag{2.18}$$

where the cost function is defined to be

$$J_{k+1}(\mathbf{u}_{k+1}) = \|\mathbf{e}_{k+1}\|_{\mathcal{Y}}^2 + \|\mathbf{u}_{k+1} - \mathbf{u}_k\|_{\mathcal{U}}^2 \tag{2.19}$$

Where $\mathcal{Y}$ and $\mathcal{U}$ are ℓ_2-spaces,

$$\mathbf{e}_{k+1} = \mathbf{r} - \mathbf{y}_{k+1} \tag{2.20}$$

and

$$\mathbf{y}_{k+1} = G\mathbf{u}_{k+1} \tag{2.21}$$

The cost function attempts to minimise the tracking error, but also attempts to minimise the change in the input sequence. The latter aims to prevent excessive control effort from being sent to the plant and helps to generate smooth inputs to actuators (Lee, Lee, and Kim, 2000). It also makes the learning somewhat conservative. The cost function can be written in more familiar form as sums.

$$J_{k+1} = \sum_{t=1}^{N} e_{k+1}^T(t)Q(t)e_{k+1}(t) + \sum_{t=0}^{N-1}[u_{k+1}(t) - u_k(t)]^T R(t)[u_{k+1}(t) - u_k(t)] \tag{2.22}$$

Where $Q(t)$ and $R(t)$ are weight matrices used to adjust the balance between optimally reducing the tracking error and limiting the change in the input sequence. By defining the inner products of $\mathcal{Y}$ and $\mathcal{U}$ in ℓ_2 space it is possible to simplify the cost function further. Using block-diagonal matrices Q and R with $Q(t)$ and $R(t)$ on the diagonal, the

definitions of the inner products $\langle \cdot, \cdot \rangle$ in $\mathcal{Y}$ and $\mathcal{U}$ are

$$\langle y_1, y_2 \rangle_{\mathcal{Y}} = y_1^T Q y_2 = \sum_{t=1}^{N} y_1(t)^T Q(t) y_2(t) \tag{2.23}$$

$$\langle u_1, u_2 \rangle_{\mathcal{U}} = u_1^T R u_2 = \sum_{t=0}^{N-1} u_1(t)^T R(t) u_2(t) \tag{2.24}$$

To minimise the cost function and hence derive the update law for $\mathbf{u}_{k+1}$, it is necessary to differentiate with respect to $\mathbf{u}_{k+1}$ which produces

$$\frac{1}{2}\frac{\partial J_{k+1}}{\partial \mathbf{u}_{k+1}} = -G^T Q \mathbf{e}_{k+1} + R(\mathbf{u}_{k+1} - \mathbf{u}_k) = 0 \tag{2.25}$$

Rearranging the differentiated cost function to solve for $\mathbf{u}_{k+1}$ leads to the optimal control law.

$$\mathbf{u}_{k+1} = \mathbf{u}_k + R^{-1} G^T Q \mathbf{e}_{k+1} \tag{2.26}$$

$R^{-1}G^TQ$ is equivalent to the adjoint operator G^* of G, so the optimal control law can be represented by

$$\mathbf{u}_{k+1} = \mathbf{u}_k + G^* \mathbf{e}_{k+1} \tag{2.27}$$

The control law, though mathematically correct, is not implementable in practice as it is non-causal. There is a requirement to know error data which is not yet available. However, it is possible to generate a causal procedure by changing the state-space parameters and noting that the transpose contains an element of time-reversal. The details of this conversion are presented by Amann et al. (1995) but will be omitted here, as the case study aims only to demonstrate how an optimal control algorithm such as Equation 2.27 can be formulated for ILC. The causal algorithm is implemented in Section 5.4.

2.2.3.7 Predictive controllers

Predictive controllers take a slightly different approach to learning control problems as compared to other types of controllers. Rather than solely using historical data to alter the plant input, the cyclic element of the disturbance is learnt by a prediction algorithm which is used to predict future error (Hätönen and Owens, 2004). The predicted error can then be used to alter the plant input. The significant advantage derived from this method is that the controller can compensate for future errors, by adjusting the control input at the current time. It allows for improved output regulation and in general the control action required to control the plant is reduced (Bone, 1995). This, in turn,

reduces stress and wear on mechanical components and actuators.

Case study Continuing their work on norm-optimal control (Amann et al., 1996b), Amann, Owens, and Rogers (1998) extended the algorithm to include future error prediction. The derivation of the predictive-optimal controller is similar to that of the original optimal controller. The main difference is that the cost function has been modified to include the predictive element.

$$J_{k+1,N}(\mathbf{u}_{k+1}) = \sum_{i=1}^{N} \lambda^{i-1}(\|\mathbf{e}_{k+i}\|^2 + \|\mathbf{u}_{k+i} - \mathbf{u}_{k+i-1}\|^2) \tag{2.28}$$

The new cost function not only includes minimisation of the error and limiting the change of the input sequence as for the optimal algorithm, but also considers the error over the future N iterations. The λ term is an extra design feature which affects the importance of more distant iterations by allowing them different weightings. Omitting several stages of the computation, the cost function can eventually be differentiated to produce

$$\frac{1}{2}J'_{k+1,N} = -G^*\mathbf{e}_{k+1} + \mathbf{u}_{k+1} - \mathbf{u}_k - \lambda G^* Q_{N-1}\mathbf{e}_{k+1} \tag{2.29}$$

which, when set equal to zero and solved for $\mathbf{u}_{k+1}$ gives the input update algorithm as

$$\mathbf{u}_{k+1} = \mathbf{u}_k + G^*(I + \lambda Q_{N-1})\mathbf{e}_{k+1} \tag{2.30}$$

where Q_N is solved recursively.

$$Q_N = L_N(I + \lambda Q_{N-1}) = [I + GG^*(I + \lambda Q_{N-1})]^{-1}(I + \lambda Q_{N-1}) \tag{2.31}$$

$$L_N := [I + GG^*(I + \lambda Q_{N-1})]^{-1} \tag{2.32}$$

In a manner similar to the original optimal algorithm the update equation is not practically implementable in this format and must be transformed into a computational procedure. However, Equation 2.30 does clearly show the inclusion of the prediction term λQ_{N-1}. The term N is of particular significance, as it adjusts how many iterations ahead the algorithm should predict. This is known as the prediction horizon. It has been shown that, as the prediction horizon increases towards infinity ($N \rightarrow \infty$), the rate of convergence of the algorithm is maximised. It is also shown that using future predicted error as well as past error data can improve the rate of convergence compared to non-predictive algorithms.

2.2.3.8 Fuzzy/Neural controllers

Learning controllers are classified under the term 'intelligent' control systems. Fuzzy logic and neural network systems also belong to this group. The key features of these controllers is that they monitor the process being controlled and attempt to improve the tracking control by learning how to modify the control input. The learning takes place automatically without any external assistance. The main disadvantage of fuzzy/neural controllers is that they are only useful for poorly defined systems. If the system is well defined, classical control techniques are likely to be more appropriate.

Seo, Park, and Lee (1999) propose a combination of different controllers which can be used for non-linear plants. The solution is a hybrid of three controller types. A feedback controller is used to stabilise the plant and keep the tracking error within bounds. Either a fuzzy logic controller or a neural network is used to learn the non-linearities of the plant and subsequently compensate for them. The non-linear compensator also allows the gains in the feedback controller to be kept reasonably low. Finally, an iterative learning controller is used to improve trajectory tracking and to update the parameters of the non-linear compensator. The combination of the three controllers is found to improve the control of the non-linear system. Hideg (1998) considers an alternative implementation of ILC which uses a neural network to reduce the computational load in the calculation of an integration routine.

2.2.3.9 Non-minimum phase systems

Non-minimum phase describes a system where any of its zeros are located in the right half of the s-plane. In the linear case at high frequencies, minimum phase zeros induce a +90 degrees phase shift while non-minimum phase zeros produce a -90 degrees phase shift. The characteristic step response of a SISO non-minimum phase system is that the output of the plant will initially peak in the direction opposite to the final steady state output. This makes non-minimum phase systems very difficult to control because the controller must anticipate that the plant will initially react in the opposite direction to what is desired. Examples of non-minimum phase systems in engineering practice are limited, but one example is that of a large ship turning at high speed (Dutton et al., 1998).

The approach to using ILC for non-minimum phase systems is to find the inverse model of the plant (Ghosh and Paden, 2001). This is not a simple technique, because inverting the transfer function of the plant by causal filtering will produce a system which is unstable (Markusson et al., 2001). The solution is to use a technique known as stable inversion (Sogo, Kinoshita, and Adachi, 2000). The basic concept is to split the plant into causal

$G_+(s)$ and anti-causal $G_-(s)$ components.

$$G(s) = G_+(s)G_-(s) \tag{2.33}$$

Each of these elements can then be treated separately. The causal element will have a stable inverse which can be found by causal filtering, while the anti-causal element will have a stable inverse which can only be found by anti-causal filtering. Ghosh and Paden (1999, 2002) also propose an alternative method for finding an approximate pseudo-inverse model for non-minimum phase plants.

Because ILC is fundamentally a batch process, anti-causal filtering can be performed on the data for a complete iteration, in the stoppage time before the next iteration begins. Having obtained the inverse plant model, it can be used to derive the input profile which must be supplied to the plant to obtain perfect tracking, assuming that the plant model is an exact representation of the plant. In practice this is not possible, as the model will always be somewhat inaccurate. However, the profile will be a good estimate which ILC can refine as tests are performed. Extensive practical implementation of iterative learning control on non-minimum phase systems has been performed by Freeman, Hätönen, Lewin, Rogers, and Owens (2004a); Freeman, Lewin, and Rogers (2004b,c).

2.3 Problems Encountered with ILC

Iterative learning control is not an ideal solution to perfect trajectory tracking. There are a number of issues surrounding the implementation of ILC which need to be addressed.

2.3.1 Initial state error

Combined with random disturbances, another critical factor, which specifically affects iterative learning control convergence, is the initial state of the plant at the start of an iteration. In many algorithms an assumption is made that, after one iteration, the system is returned to exactly the same initial conditions in preparation for the next iteration. In practice, this is never quite accurate, as there is almost always some initial state error (Park and Bien, 2000; Sun and Wang, 2003). It has been demonstrated that initial state error can have a significant effect on the stability of ILC algorithms. Much research has been performed on developing algorithms which are robust to initial state error (Chen, Wen, Gong, and Sun, 1999; Chen, Wen, Xu, and Sun, 1996a; Lee and Bien, 1996; Sun and Wang, 2001b).

A procedure for making ILC algorithms robust to initial state error is not to compensate for the error, but accept that it will be present (Jiang and Unbehauen, 2002; Sun

and Wang, 2002). For a particular algorithm, mathematical boundary conditions can be determined, within which the initial error must lie. As long as the error is within these boundaries, the algorithm will not be affected and will remain stable (Jiang and Unbehauen, 1999). It would seem logical that an algorithm with tolerance to a wide range of initial error is superior to others. However, in general, as the tolerance increases, the final tracking error also increases and the algorithm is no longer guaranteed to converge to zero tracking error.

An alternative approach to reducing the sensitivity of ILC to initial state error is to use initial state learning (Chen et al., 1999). Equations 2.34 and 2.35 give an example of a D-type ILC law combined with an initial state learning algorithm. Effectively, the process has two systems operating in parallel. The D-type law is standard ILC, while the state learning algorithm obtains the initial conditions of the system states. The initial state information can be used to modify the initial value of the D-type law to completely remove the initial state error (Chen et al., 1996a).

$$\mathbf{u}_{k+1}(t) = \mathbf{u}_k(t) + \beta \dot{\mathbf{e}}_k(t) \tag{2.34}$$

$$x_{k+1}(t_0) = x_k(t_0) + B(t_0)\beta(t_0)e_k(t_0) \tag{2.35}$$

where $x(t_0)$ are the initial states of the plant and B is the standard state space system matrix from $\dot{x} = Ax + Bu$.

A more recent solution to the initial error problem is to use rectifying action. Rather than deriving algorithms which tolerate initial error, Sun and Wang (2002) actively use rectifying action to remove the initial error in the first portion of the trajectory and lock the system onto the desired trajectory. The rectifying action generates a smooth input at the start of the iteration to compensate for the initial error. Once the error has been removed, the learning controller can function normally.

2.3.2 Long term stability

Barton et al. (2000); Gorinevsky (1999); Havlicsek and Alleyne (1999); Huang and Longman (1996); Lewin (1999); Longman, Akogyeram, Hutton, and Juang (2001); Norrlöf and Gunnarsson (2002b); Songchon and Longman (2001) amongst others, point out a very significant problem with ILC implementation. They are concerned with the long term stability of all learning algorithms. It is found, in practice that if a learning algorithm is implemented for a large number of iterations, the tracking error will begin to increase after it is initially reduced. This leads to divergence of the algorithm and total instability of the controller (Chen, Dou, and Tan, 2001). Evidently, this is unacceptable in an industrial environment. In practice, it is sometimes necessary to sacrifice perfect tracking

and fast convergence to obtain an algorithm which will not diverge (De Roover, 1996). In fact, it is more important to obtain an algorithm which does not diverge, rather than one which converges quickly but then diverges later (De Roover, Bosgra, and Steinbuch, 2000). Chen and Longman (1999) have conducted a significant amount of research into the reasons why learning control experiences long term instability. It is found that several factors can operate to destabilise a system.

2.3.2.1 Frequency domain analysis

The error signal between the reference trajectory and the actual trajectory of a feedback control system consists of a continuous spectrum of frequencies. With respect to the system being controlled, there exist low frequency and high frequency error components. In repeated trajectory tracking problems, the low frequency components are usually repeatable from trial to trial, while the high frequency components are random, often caused by measurement noise. High frequency components of the error also tend to have very small amplitude and so may be considered negligible. Therefore, in order to improve tracking, it is necessary to reduce the amplitude of the low frequency components, as these tend to have the most influence on overall tracking accuracy.

The frequency response of a stable plant generally indicates an increasing phase lag at higher frequency, which has a destabilizing effect on the controlled plant. When using standard feedback control, this is not a problem, as the amplitude of high frequency error signals remains relatively small, even as the controller actively reduces the amplitude of the low frequency components. In contrast, learning controllers, by their very nature, integrate the error signal from trial to trial and eventually the high frequency components can build up within the learning loop and drive the system unstable.

From the moment an iterative controller begins to operate, the low frequency error is reduced, while the destabilising high frequency error is increased. Initially, because the low frequency error has significantly larger amplitude, the potential for tracking improvement is large and the effect of the slowly increasing high frequency amplitude cannot be detected. Later, as the low frequency error becomes negligible, the effect of the high frequencies begins to show and the overall tracking error noticeably grows. This explains why a learning controller appears to converge to minimal error but begins to diverge after a number of iterations (Huang and Longman, 1996).

2.3.2.2 Time domain analysis

Basic ILC learns by adjusting the plant input waveform at each sample instant for the next iteration, based on the error which is recorded at the corresponding sample instant during the current iteration. At any one sample instant k, no account is taken of the

learning which occurs at previous sample instants. Yet, if the corrections made at previous samples successfully reduce the tracking error earlier in the trajectory, the corrective action at k may in fact be greater than necessary. Only the first sample instant is unaffected by this procedure, while the last sample instant is affected by the learning which occurs at all other samples. This implies that it is easier to learn the required waveform at the start of the trajectory than at the end, as there is far less interaction between samples. As the learning in early samples successfully reduces the tracking error at the start of the trajectory to near zero, the corrections made to the early part of the trajectory will become negligible. Effectively, once the early section of the trajectory has been learnt, it remains constant. Thus enabling later samples to learn more effectively as they are less affected by earlier samples. At each iteration, the learnt section of the trajectory increases further along the trajectory, until it reaches the final sample. The complete trajectory has now been learnt. However, while learning occurs in the early part of the trajectory, the error at the end of the trajectory is likely to fluctuate significantly as the corrective action at each iteration is unnecessarily abrupt.

The tracking performance for one iteration is usually given as a representation of the error recorded at each sample. Frequently, the error norm or the mean squared error are used. If we consider the effect of learning on the error norm or the mean squared error, the emerging pattern of reduction in error followed by an increase becomes observable. During the first few iterations, successful learning occurs mainly near the start of the trajectory, the error is being rapidly reduced, and as the interactions with later iterations have not built up yet, the overall error reduces quickly. As more iterations occur, the error reduction at the beginning of the trajectory is minimal, yet the control effort increases near the end of the trajectory so that the overall error appears to increase. As learning begins to occur at the end of the trajectory, the overall error reduces again and reaches a minimum. This effect does not explain long term instability of ILC, but does suggest an alternative reason for the observed decrease followed by increase in tracking error (Huang and Longman, 1996).

2.3.2.3 The waterbed effect

The waterbed effect states that, for any feedback control system, the attenuation of certain error frequencies results in the amplification of the error at other frequencies. It is derived from the sensitivity function for a discrete feedback controller relating command to error. Use of the Bode integral theorem on this sensitivity function concludes that in the logarithm space, the attenuation of certain frequencies is equal to the amplification of the frequencies which are not attenuated. Feedback control does not specifically target any frequencies for reduction. In general, dominant frequencies are attenuated and amplification occurs at frequencies which are less significant, resulting in improved control. Learning controllers are frequently feed-forward and, with correct design, can be made to bypass the waterbed effect so that attenuation of certain frequencies does not

amplify others. However, when a learning controller is coupled to a feedback controller in a hybrid arrangement, the waterbed effect will be present (Songchon and Longman, 2001).

2.3.2.4 Solutions to long-term instability

The design engineer is not completely devoid of tools to prevent long term instability. A number of solutions can be implemented to help prevent the problem from occurring.

- Switch off the learning controller when the tracking error is sufficiently reduced (Barton et al., 2000).
- Use a low-pass cut-off filter to remove high frequency components of the signal (Longman, 2000).
- Implement phase-lead compensation (Longman, 2000).

The first solution is very simple to implement and is particularly suited to a hybrid, feedback and learning controller. The controller monitors the tracking error over each trial and, once it reaches either a tolerance boundary or, better still, reaches a minimum, the learning controller is switched off. The feedback controller maintains the same level of tracking accuracy and there is no further learning. A second boundary can be used to switch the learning controller on again, if the error increases due to changes in the plant, for example due to wear. The drawback is that the ability to learn is switched off for the majority of the time the system is operating and there is no opportunity to improve system performance on a continuous basis.

The second solution involves using a low pass cut-off filter to remove the high frequency components of the error signal, so that they are not integrated around the learning loop (Chen and Longman, 1999). The filter must prevent all frequencies above a critical value from being transmitted (Lee, Bang, Yi, Son, and Yoon, 1996). The critical frequency can be determined from the point at which a Nyquist plot of the system goes outside the unit circle and is usually chosen as the Nyquist frequency. Zheng and Alleyne (2003) and Rotariu, Ellenbroek, and Steinbuch (2003) independently propose modifications to this approach, which use an adaptive filter to vary the cut-off frequency. This allows the bandwidth of the control system to be increased at certain points during the iteration period where high frequency components of the error signal need to be learnt, while allowing a lower cut-off frequency during other segments of the iteration where high frequency signals would have a destabilising effect.

Unfortunately, introducing a cut-off filter induces additional phase lag which destabilises the system. To counteract this, non-causal, zero-phase batch filtering can be performed between iterations to prevent phase lag from being introduced (Pandit and Buchheit,

1999). Because no filter is perfect, a small amplitude of high frequency signals will still remain in the learning loop. Over time, these can build up and render the controller unstable after thousands of iterations of stable operation. A solution is to use the quantization effect, inherent in digital control systems. If the quantization level is larger than the amplitude of the noise leaking through the filter, the quantization effectively removes the remaining noise because it cannot be recorded (Chen and Longman, 1999).

The third solution involves a low pass cut-off filter and a phase lead compensator (Yongqiang and Wang, 2003). It can be used when the maximum cut-off frequency using a filter alone is not high enough to remove disturbances which have significant effect on the error. The phase lead compensator shifts the Nyquist plot so that the frequency at which the plot leaves the unit circle is made higher. It allows the cut-off frequency to be raised until all the critical disturbance frequencies are passed to the learning controller and can be removed.

2.4 Previous Practical Implementation

Most published iterative learning control research tests and proves new theories by using simulation examples. The work in this thesis is strictly concerned with evaluating different ILC strategies on a real physical system. Simulations are used in this work, but only to test and optimise algorithms, before they are implemented on the real plant.

It is also important to note that ILC research is not completely devoid of practical implementation studies. A number of experiments have been performed on a variety of different dynamic systems and in all cases, ILC has been found to improve the performance of the control system. A selection of these studies is included to illustrate the diversity of applications involving ILC.

2.4.1 Application to robotics

Iterative learning control was originally introduced by Arimoto and co-workers specifically to improve the trajectory tracking capabilities of robots. Since then, many ILC simulation studies have been based around robotic applications for example Arimoto et al. (1984b, 1985a); Bondi, Casalino, and Gambardella (1988); Kawamura et al. (1985, 1988); Mita and Kato (1985); Togai and Yamano (1985); Wang and Cheah (1998).

Longman (2000) and Elci, Longman, Phan, Juang, and Ugoletti (1994) describe the practical implementation of basic ILC on a seven axis industrial robot. The robot must perform a rapid trajectory move, which maximises the interaction of the axes and induces non-linear dynamics such as Coriolis and centrifugal effects. The seven axes are each controlled separately as if the robot consisted of seven SISO systems. The non-linear

effects and interactions between axes are simply treated as repeatable disturbances. Even using a simple learning controller, the tracking error is shown to be reduced by up to a factor of 1000 in 12 iterations. This demonstrates that iterative learning control is well suited to controlling individual subsystems of a larger overall system, with little knowledge of how the subsystems interact with each other (Hwang, Kim, and Bien, 1993).

De Luca and Panzieri (1994) implement a simple ILC algorithm for controlling a two link robot arm and specifically investigate how the learning controller can compensate for the effect of gravity on a flexible link robot. The difficulty with flexible link robots is that the joint angles cannot be used to determine the exact position of the end effector as would be the case for a rigid link robot. The flexing of the robot arm during motion, results in an error between the actual position of the end effector and the position calculated from joint angle measurements. De Luca and Panzieri use an optical transducer attached to the robot flexible link to measure the deflection of the arm and feed the information into the controller. Coupled with the effect of gravity, this is a complex dynamic problem. The learning controller requires no prior knowledge of the plant and yet successfully manages to compensate exactly for the effects of gravity. Gunnarsson and Norrlöf (2001); Norrlöf (2002); Norrlöf and Gunnarsson (2001) have also performed significant practical implementation on robots particularly with respect to optimal learning control. They have compared three different algorithms by practical implementation on a robot arm (Norrlöf and Gunnarsson, 2002a). The algorithms are taken from different areas of research and include a classical P-type, a model-based design and an optimization-based technique. All algorithms are found to reduce the tracking error, as the number of iterations increases. However, the model-based and optimization algorithms can potentially result in faster convergence and smaller steady state error. The general conclusion is that, if a basic algorithm can meet performance requirements, it should be used, as simplicity is an advantage in practical implementations. If the basic algorithm performance is unsatisfactory, more advanced algorithms need to be developed, which process more information about the plant.

2.4.2 Application to chain conveyors

Conveyor systems are extensively used in industry for transporting all manner of goods between the components of a larger system. For most applications, the control system required is very basic, the system may even be operated completely open-loop or by means of limit switches, which trigger different processes. However, in processes requiring greater accuracy, the performance of the controller becomes very important.

Consider an assembly line where a conveyor moves the product through a series of workstations and different components are added at each stage. The position of the product must be controlled very accurately, so that there is minimal relative error between the

product and the work-station while the components are being added. The task can be performed in one of two ways, described as either indexing or synchronising (Barton and Lewin, 2000). In indexing mode, the work-station remains stationary, the conveyor must move the product to the work-station, then wait while the assembly task is performed before moving the product on to the next work-station. In synchronising mode, the conveyor runs at constant velocity. The work-station must match the velocity of the conveyor and perform the assembly task while moving. When the task is complete, the work-station rapidly returns to its initial position ready to synchronise velocity with the next product. Both of these implementations require accurate position control.

Barton et al. (2000) implemented learning control on a chain conveyor system with the aim of improving the tracking performance of a synchronising system. A hybrid controller was used which consisted of a learning controller coupled with a 3-term feedback controller. A series formation was used whereby the learning controller output was supplied to the feedback controller, so as to adjust the reference trajectory and achieve perfect tracking. Due to the continuous nature of the task, the implementation was, in fact, more appropriate to repetitive control. However, the formulation of the problem was from an iterative learning viewpoint. Moore (2000) explains in detail the differences between the implementations of ILC and RC and concludes that it is possible to develop what is essentially an ILC controller, specifically designed for continuous repetitive systems which do not include resetting between trials. The algorithm was a basic P-type with current iteration error feedback.

$$u_{k+1}(t) = u_k(t) + Ke_{k+1}(t) \tag{2.36}$$

Note that u and e are scalars, not vectors, to represent correctly the repetitive implementation. Three values (0.01, 0.05, 0.10) were used for the learning gain K and the mean squared tracking error was recorded for each trial. The results show that, irrespective of the value for learning gain, the performance of the hybrid controller was significantly improved over that for the 3-term controller alone and the error could be reduced to approximately 14% of the 3-term controller value. The main factor affected by changing the learning gain was the rate of error reduction. With $K = 0.1$ the error was reduced to the same level as for $K = 0.01$, but in 20 trials rather than 150. However, it is worth noting that for all values of learning gain, the system eventually became unstable if the learning controller was allowed to operate for every iteration. For higher gain, the onset of instability was much sooner than for lower gain. This problem was solved by switching off the learning controller once the error had been sufficiently reduced. The 3-term controller, then, continued to operate the system at the same error level without any further learning.

2.4.3 Liquid slosh in industrial packaging

When a package containing a fluid is accelerated, motion is induced in the fluid. This is known as slosh (Grundelius and Bernhardsson, 2000). Automated packaging machines are extensively used in industry to package fluids. The machine first builds the container, then fills it before finally sealing it. Slosh is a significant problem in this process because if the liquid experiences sufficient acceleration, it can splash out of the container or contaminate the sealing surfaces which results in incorrectly sealed packages. However, the motion of the fluid under acceleration is very repeatable and, if the correct acceleration profile can be found, the slosh can be kept to a minimum.

Grundelius (2000); Grundelius and Bernhardsson (2000) propose a solution to reduce the liquid slosh in a packaging process by using iterative learning. The task is to learn the acceleration profile which keeps the liquid slosh below a specified height up the side of the container. In the experiments, the height of the slosh is measured by using a laser displacement sensor. The feedback to the controller consists of the height of the slosh on both the front and rear faces of the container, respective to the direction of motion. Before the learning algorithm can be applied, a reasonably good estimate of the optimal acceleration profile must be generated from models of the fluid behaviour. This is the profile which is supplied to the plant for the first iteration. The learning algorithm takes the form

$$\mathbf{u}_{k+1} = Q\mathbf{u}_k + L_1\mathbf{e}_{k_1} + L_2\mathbf{e}_{k_2} \tag{2.37}$$

where Q, L_1 and L_2 are matrices which represent time varying filters, selected from process model parameters, $\mathbf{e}_{k_1}$ and $\mathbf{e}_{k_2}$ are the fluid surface elevation errors for the front and rear surfaces of the container. The filters also affect the level of learning during an iteration. There are two error terms in this algorithm taking into account both the displacement on the front surface and on the rear surface. Use of either a linear or non-linear process model for selection of the filters produces results which show that the liquid slosh can be significantly reduced by the learning controller, and particularly the slosh near the end of the iteration.

2.4.4 Control of paralyzed human limbs

Paralysis can be the results of several factors such as neurological disorders or spinal injuries. The central nervous system and the brain are no longer in control of muscles because the nerves which carry the stimulation and feedback signals are somehow damaged and do not function properly. However, the muscles are still functional and can be stimulated externally to generate movement. One method known as Functional Electrical Stimulation (FES) uses electrical impulses to stimulate the muscles. If the correct sequence of pulses is supplied to muscles, then motion can be restored in paralysed areas

of the body. The main difficulty in this process is determining the stimulation sequence required to obtain the desired motion.

Dou, Tan, Lee, and Zhou (1999) offer one solution to this problem by implementing an iterative learning controller, which learns the stimulation pattern required to obtain controlled movement of a paralysed patient's arm. The control signal to the arm is provided by means of the FES technique, while feedback is provided by means of a angle measuring transducer attached to the moving joint. Only the elbow joint is considered. The wrist and shoulder joints are held stationary. The reference trajectory is a set of angular displacements which occur over a period of two seconds. The controller implemented on the patient is of a hybrid nature using a PD-Type feedback controller and a higher order ILC controller in parallel. The ILC controller takes the form

$$\mathbf{u}_{k+1}(t) = \mathbf{u}_k(t) + \sum_{i=1}^{j} \beta_i(t)\mathbf{e}_{k-i+1}(t+1) \tag{2.38}$$

where j is the order of the controller. Results in both simulation and practical implementation demonstrate that ILC significantly improves the tracking of the paralysed arm. In the practical implementation, at the first iteration hardly any motion of the arm is perceptible as the tracking is non-existent. However, by the tenth iteration, the tracking becomes efficient. For this particular implementation, the higher order algorithm appears to be more successful than the first order algorithm.

2.4.5 Plastics manufacture - injection molding

Injection molding is a high-speed manufacturing process, which allows large quantities of products to be made at low cost. The molding process can generate very intricate shapes. However, accurate molding of intricate shapes requires the parameters of the injection molding machine such as temperature and pressure to be controlled accurately.

Havlicsek and Alleyne (1999) investigate improving the accuracy of pressure regulation within the injection and packing phases of a molding cycle. Pressure during these stages is regulated by means of electro-hydraulic valves and feedback to the controller is by means of thin-film pressure transducers. The controller used is of the Hybrid type with a feedback controller and a learning controller operating in parallel. The feedback controller stabilises the plant, while the learning controller modifies the output of the feedback controller to compensate for repeating disturbances. The output from the hybrid controller can be represented by

$$w_k(t) = G_{fb}(s)e_k(t) + \mathbf{u}_k^*(t) \tag{2.39}$$

where $w_k(t)$ is the input to the plant, $G_fb(s)e_k(t)$ is the feedback control signal and $u_k^*(t)$ is the filtered feed-forward signal from the iterative learning controller. The iterative

learning component of the controller uses the basic PD-type format.

$$\mathbf{u}_k(t) = \mathbf{u}_{k-1}(t) + Q(s)(K_p\mathbf{e}_{k-1}(t) + K_d\dot{\mathbf{e}}_{k-1}(t)) \tag{2.40}$$

$$\mathbf{u}_k^*(t) = Q(s)\mathbf{u}_k(t) \tag{2.41}$$

where K_p and K_d are respectively the proportional and derivative learning gains. $Q(s)$ is a Q-filter with zero phase lag, used to remove noise from the feed-forward output of the controller and improve robustness of the system.

Results generated by practical implementation on a molding machine demonstrate that after six iterations, the learning controller is capable of reducing the tracking error by at least an order of magnitude. It is suggested that after six iterations, the learning controller may be switched off to prevent the onset of instability. The feedback controller will then continue to operate alone with the same level of error reduction.

2.4.6 Military applications

Iterative learning is not strictly restricted to motion control systems. Chen, Wen, Xu, and Sun (1998b); Chen, Xu, and Wen (1998c) apply iterative learning-based identification to determine the drag coefficient of a projectile fired by artillery and a bomb dropped by an aircraft. Determining the drag coefficient of either of these weapons is critical to improving targeting accuracy.

In iterative learning control systems, the objective is to learn the input to the plant, which gives zero error tracking. In iterative learning-based identification, the inputs and outputs are already known, so that the task consists in learning the function which links them together. The technique involves several stages. Firstly, the number of iterations over which the identification will take place must be chosen. An arbitrary drag coefficient curve is then chosen and used to calculate the output, based on the input. The calculated output and the actual output are compared, the error is found and is used to update the drag coefficient in an iterative manner. The bomb drag coefficient is updated using a first order D-type algorithm.

$$[C_{df}(t)]_{k+1} = [C_{df}(t)]_k + \beta(t)\dot{e}_k(t) \tag{2.42}$$

where C_{df} is the drag coefficient curve. Whereas the projectile drag coefficient is updated using a higher order D-type algorithm. In both studies, the iterative identification techniques compare well to other identification techniques such as optimal dynamic fitting. The iterative technique allows less restriction on the initial estimate of the drag coefficient curve.

2.4.7 Disk drive control

Optical (CD/DVD) and hard disk drives (HDD) are used extensively in the entertainment industry, for storing music and films and in the computing industry, for storing data and programs. Data tracks are read from the disk by means of an optical or electro-magnetic read head, which must accurately follow the track to prevent read errors. Track following is performed by sensors which monitor the error between the track and the read head. The error is greatly amplified and used to control a motor which moves the read head onto the track. Tracking errors occur due to both periodic and non-periodic disturbances, the periodic disturbances having more significant effects. Periodic disturbances arise due to track decenter and track eccentricity. Track decenter occurs when the center of rotation of the tracks does not correspond with the center of rotation of the disk. Track eccentricity implies that the tracks are not perfectly round, but are somewhat elliptical.

Moon et al. (1996) have developed an iterative learning controller capable of learning the periodic disturbances and reducing the tracking error. The controller is a hybrid composed of a feedback controller and a learning controller operating in parallel. As it is not desirable for the learning controller to learn the non-periodic disturbances, a filter has been added into the learning algorithm to prevent high frequency components of the signal being used in the learning process. Results show a reasonable reduction of the periodic tracking error when the learning controller is implemented. Adaptive feed-forward and repetitive control algorithms are now standard in most DVD and HDD drives (Chew and Tomizuka, 1990; Onuki and Ishioka, 2001; Zhou, Steinbuch, and Leenknegt, 2004).

2.4.8 Controlling wafer temperature in rapid thermal processing

Microchip manufacture requires the thermal processing of a silicon wafer involving a number of techniques (including annealing, oxidation, nitridation, chemical vapour deposition and cleaning) to build up the electronic circuit. Conventional processing uses a furnace to heat the silicon wafers to the correct temperatures. A more recent technique known as Rapid Thermal Processing (RTP) has advantages over furnace processing because all of the manufacturing processes can be performed within a single unit, with a reduced thermal budget. However, RTP is not currently a popular technique as it is very difficult to maintain an even temperature over the entire surface of the silicon wafer.

Yang, Lee, Ahn, and Lee (2003) consider the temperature control of a RTP unit capable of accepting 8 inch silicon wafers. The temperature of a test wafer is measured in three locations by thermocouples and heat is supplied by tungsten-halogen lamps arranged in three arrays. The overall plant is therefore a 3 input, 3 output, MIMO system. In addition, the plant is highly nonlinear and very sensitive to noise.

Conventional PI feedback control is unable to control the wafer temperature suitably due to the highly interacting nature of the inputs and outputs. Similarly, conventional model-based control techniques are unsuitable, because the non-linearity of the plant results in large error between the model and the real plant dynamics. To overcome these problems, a Batch Linear Quadratic Gaussian (BLQRG) optimal learning controller is developed to learn the required control signal for the three arrays of lamps which will accurately control the wafer temperature. The optimal ILC algorithm is not only able to learn between iterations, but also operates real-time, so that the controller can respond to disturbances as they occur, rather than wait till the next iteration.

2.4.9 Biochemical industry

Iterative learning control has been successfully implemented in the biochemical industry. Choi, Choi, Lee, and Lee (1996) discuss the practical implementation of ILC on the fed-batch cultivation of Acinetobacter calcoaceticus RAG-1. This microorganism is used to produce emulsan, a chemical with good emulsifying properties for crude oil. Ethanol is an important factor for the microorganism as it is a carbon source aiding cell growth, cell maintenance and emulsan production. However, too much ethanol has a negative effect on the microorganism. Therefore the ethanol level must be carefully regulated.

Choi et al. (1996) implement a hybrid iterative learning controller and PI-type feedback controller to improve the control of ethanol supply to the microorganism over a number of batches. The learning controller is of basic P-type and uses the inverse plant model as the learning gain function. Results from practical implementation demonstrate that using the learning controller improves the yield from one batch by approximately 20% after only three iterations.

A more recent implementation investigated by Mezghani, Roux, Cabassud, Le Lann, Dahhou, and Casamatta (2002) uses an optimal ILC on a chemical batch reactor which can be used to manufacture pharmaceutical products. The cost function is used to minimise the error while simultaneously limiting the change in the output at the next iteration, thus restricting sudden changes in the control effort. Sliding window filtering is also performed to smooth the output and reduce the effect of extreme control effort values.

2.4.10 Torque ripple minimisation for electric motors

Permanent Magnet Synchronous Motors (PMSM) are becoming increasingly popular in industrial servo applications. However, their construction intrinsically results in torque variations which occur as the drive shaft rotates. This is particularly noticeable if the motor is required to rotate under load at low speed. Torque pulsations result in reduced

servo performance, unnecessary noise and undesirable mechanical vibrations. The main sources of torque ripple are cogging and flux harmonics. Cogging is due to the magnetic attraction between the permanent magnets on the rotor and the stator, which tries to turn the rotor so that the stator teeth and the rotor magnets are aligned. Flux harmonics occur in the air gap between stator and rotor, because a perfect sinusoidal variation in flux is difficult to achieve.

Qian, Panda, and Xu (2004) have successfully implemented two ILC schemes significantly reducing the level of torque ripple. The first scheme is a standard time-based higher order P-type algorithm which simultaneously uses data from the past iteration and also from the current iteration (CITE). A forgetting factor is implemented in order to reduce the impacts of measurement noise, initial state error and variable system dynamics. However, use of the forgetting factor limits the extent to which the repeating tracking error can be reduced. To overcome this problem, a second algorithm is developed which uses a Fourier series representation of the previous iteration input instead of the actual input used. The Fourier series representation intrinsically averages the effect of noise and non-repeating disturbances, without requiring a forgetting factor. During experimental tests, both ILC techniques produce similar performance with respect to ripple reduction, reducing it by a factor of three. The Fourier based approach performs slightly better than the time based approach.

2.4.11 Electromechanical valve control in camless engines

The ability to adjust the valve timing of an internal combustion engine can significantly improve power output, engine efficiency and reduce emissions. Valves actuated by a cam mechanism offer limited variation in valve timing. An increasing interest is noticeable in the development of electromechanically actuated valves, which can be operated at any time by a command signal from the engine management system. However, the performance requirements of these valves are very demanding. The valve must open and close extremely quickly, but the contact velocity, as the valve mechanism reaches the limit of travel in either direction, must be kept as small as possible to reduce component wear and eventual failure. These opposing criteria produce a system which is extremely difficult to control. However, because of the repeating nature of the motion which the valve must undergo, the system is well suited to ILC.

Hoffmann, Peterson, and Stefanopoulou (2003) have simulated and experimentally assessed the performance of a simple learning controller in series with a feedback controller. The feedback controller is only operated in the last stages of the valve motion, when the open loop plant is naturally unstable and requires feedback control in order to be stabilised. The learning controller is able to adjust both the plant input and the reference trajectory to the feedback controller and consequently ILC can learn the required actuation voltage to achieve a slow contact velocity. The target contact velocity of 0.1 m/s

is successfully achieved after 35 iterations.

2.5 Summary

From a review of ILC literature, it can be concluded that all ILC algorithms are based on the same fundamental principle, which uses the input from the previous trial and a modification of the error sequence to develop the input for the next trial. In basic algorithms, the error modification is generally simple and often takes the form of a scalar gain. For model-based algorithms, the model is frequently used to filter the error vector, in order to generate more rapid error reduction. Techniques from a variety of other control methodologies such as optimal, adaptive and robust have been applied to ILC with the aim of improving convergence rate and stability to unknown and un-modelled disturbances.

ILC has been successfully applied to a number of industrial applications including injection molding, robot control, disk drive control and chemical batch processes. The use of ILC in these examples has greatly improved the performance of the plant being controlled. However, in general more experimental data is required, so as to direct theoretical research towards the problems which arise when ILC is used in industrial applications.

Chapter 3

Multi-Axis Test Facility and Experimental Procedures

3.1 Introduction

It is essential to test new ILC algorithms on a plant which resembles an existing industrial process, if ILC is to have wider application in industry. The following sections describe the design and construction of the test facility, used to evaluate the ILC algorithms discussed in later chapters. Different elements of the construction: hardware, interfaces and software are discussed with sufficient detail to allow a replica of the test facility to be constructed. The plant is modelled by means of frequency response tests, which produce a continuous-time transfer function for each axis. The models are validated by comparing simulated and experimental step responses. Finally, at the end of this chapter a series of test procedures are established to allow a fair comparison of the performance produced by different algorithms.

3.2 Test Facility

3.2.1 Hardware: Gantry robot

The experimental work presented in this thesis has been performed in its entirety, on a 3-axis, industrial gantry robot as shown in Figure 3.1. The robot is a commercially available unit manufactured by Aerotech Inc. USA. and consists of three individual, linear motion axes which, when controlled simultaneously allow the robot end-effector to be positioned anywhere within a cuboid work envelope. The end-effector consists of an electromagnet on a compliant arm, the design of which is shown in Figure 3.2 (further details of additional compliant arm components are in Appendix A). The purpose of the

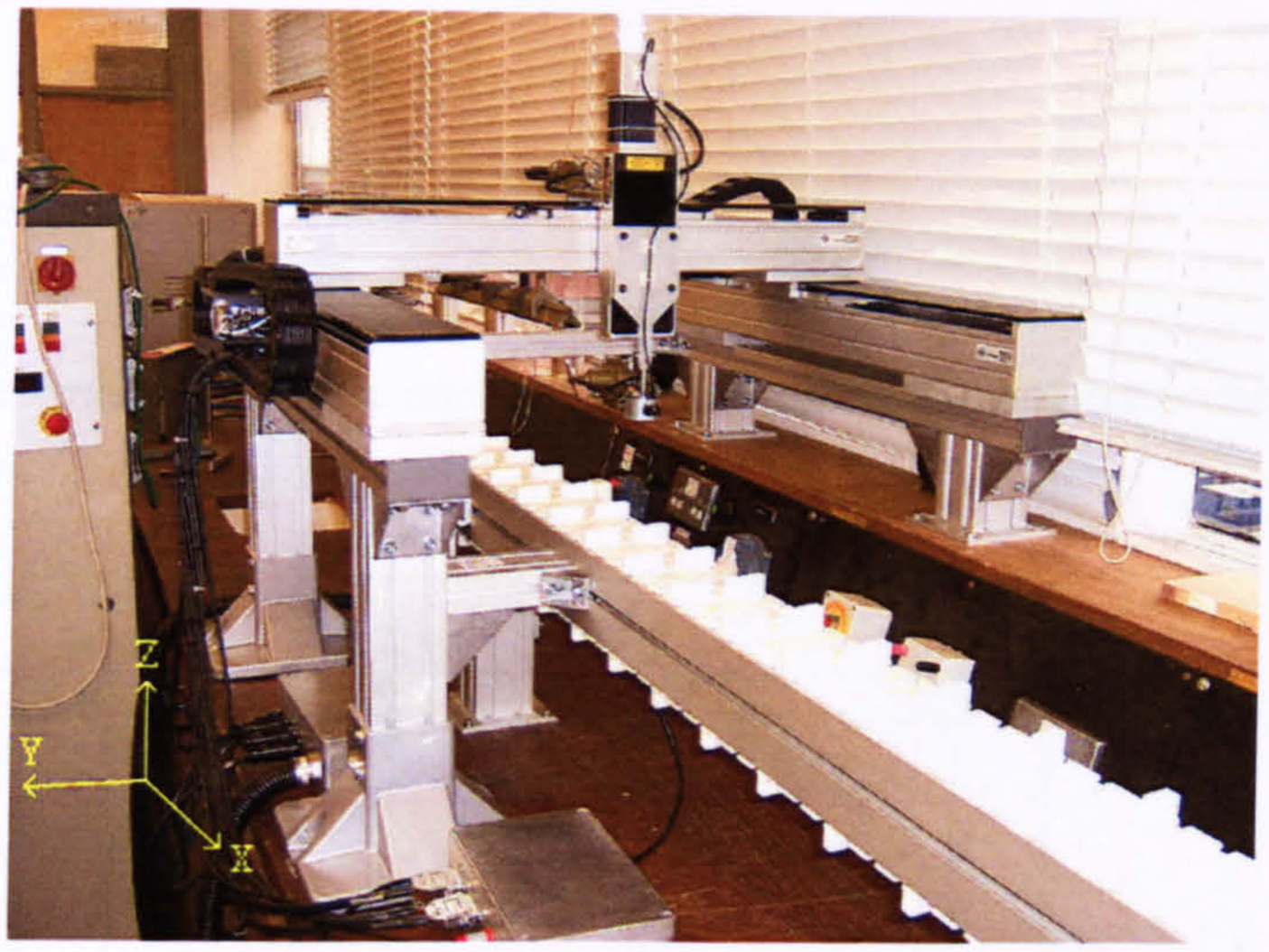


FIGURE 3.1: The Gantry Robot test facility

end-effector is to hold ferromagnetic materials, while they are moved by the robot. It has not been used within the scope of experiments performed in this thesis. In order to avoid confusion, each axis has been assigned an identifier, either X, Y, or Z, to specify which of the axes is being discussed.

The X-axis is the lowest horizontal axis which moves parallel to the conveyor located beneath the robot. The X-axis actually consists of two units, an Aerotech model ALA10064-M brushless dc, permanent magnet, linear motor of 1025mm physical length, providing 640mm of effective travel, which appears on the left-hand side of Figure 3.1 and a free running Aerotech model ALB10064-M linear slide, on the right side of Figure 3.1. The second horizontal Y-axis is mounted across the two X-axis units and is also an Aerotech model ALA10052-M brushless dc, permanent magnet, linear motor of physical length 950mm, providing 520mm of effective travel. The Y-axis moves in a direction perpendicular to the conveyor. The Z-axis is a short, vertical, linear motion stage consisting of a 2mm lead, Aerotech model ATS100-100, ball-screw slide powered by an Aerotech model BM75, brushless dc, rotary motor. The Z-axis is mounted on the carriage of the Y-axis linear motor. The resulting dimensions of the cuboid work envelope are 640 $\times$ 520 $\times$ 100mm (X-Y-Z).

The gantry robot is mounted above the conveyor on a custom built stand, which has been constructed from extruded aluminium beams and brackets, supplied by Flexlink. A three dimensional design of the stand can be seen in Figure 3.3 and a list of components is presented in Table 3.1. The Aerotech motors could not be bolted directly to the Flexlink components, because the mounting techniques are incompatible. Therefore four adapter plates were manufactured to provide a bridge between the two systems, the design can be seen in Figure 3.4. The entire assembly is bolted to a wooden bench 0.87m above the floor. The mountings are therefore not completely rigid and do have a degree of flexibility.

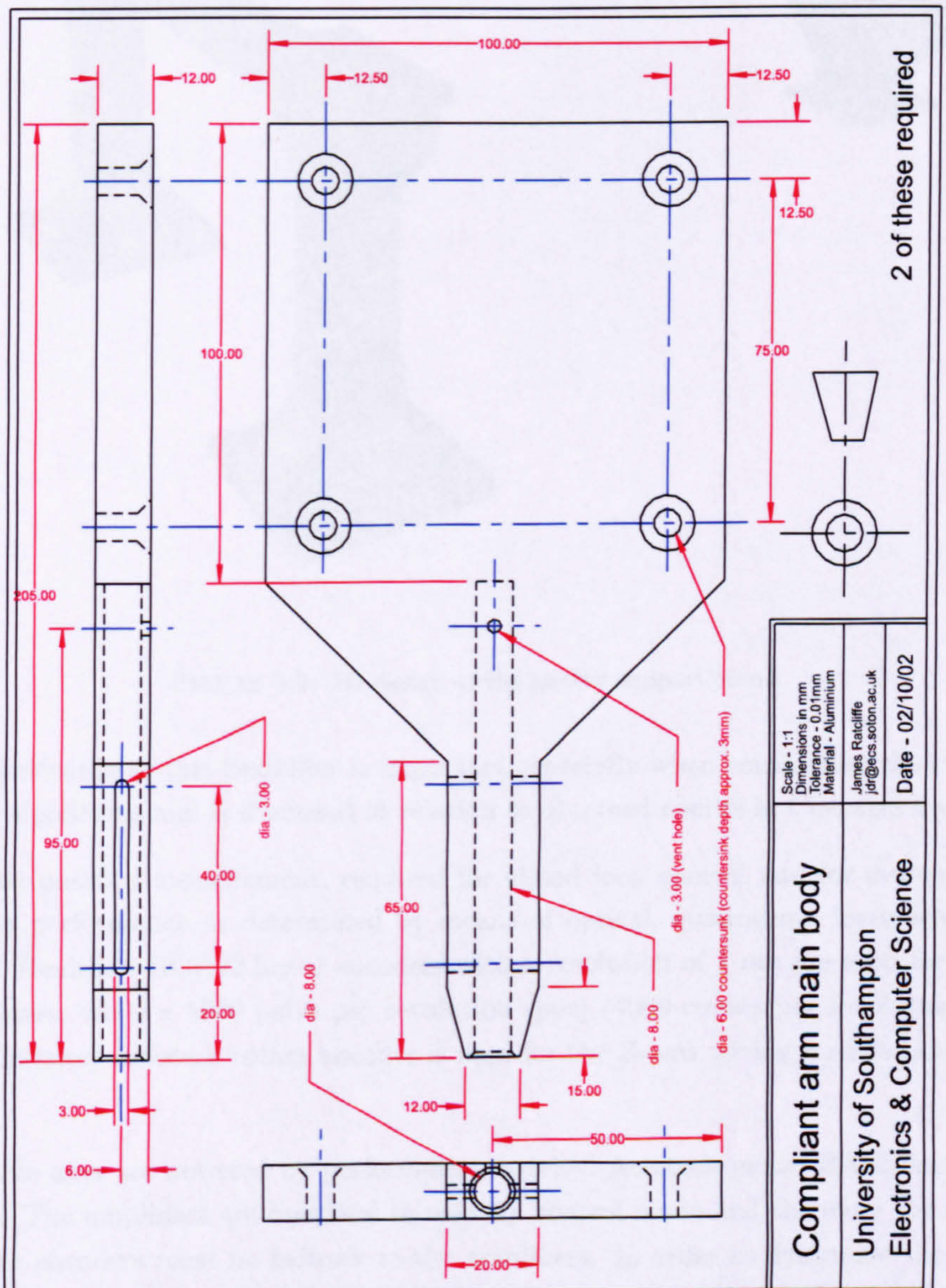


FIGURE 3.2: Compliant arm main body

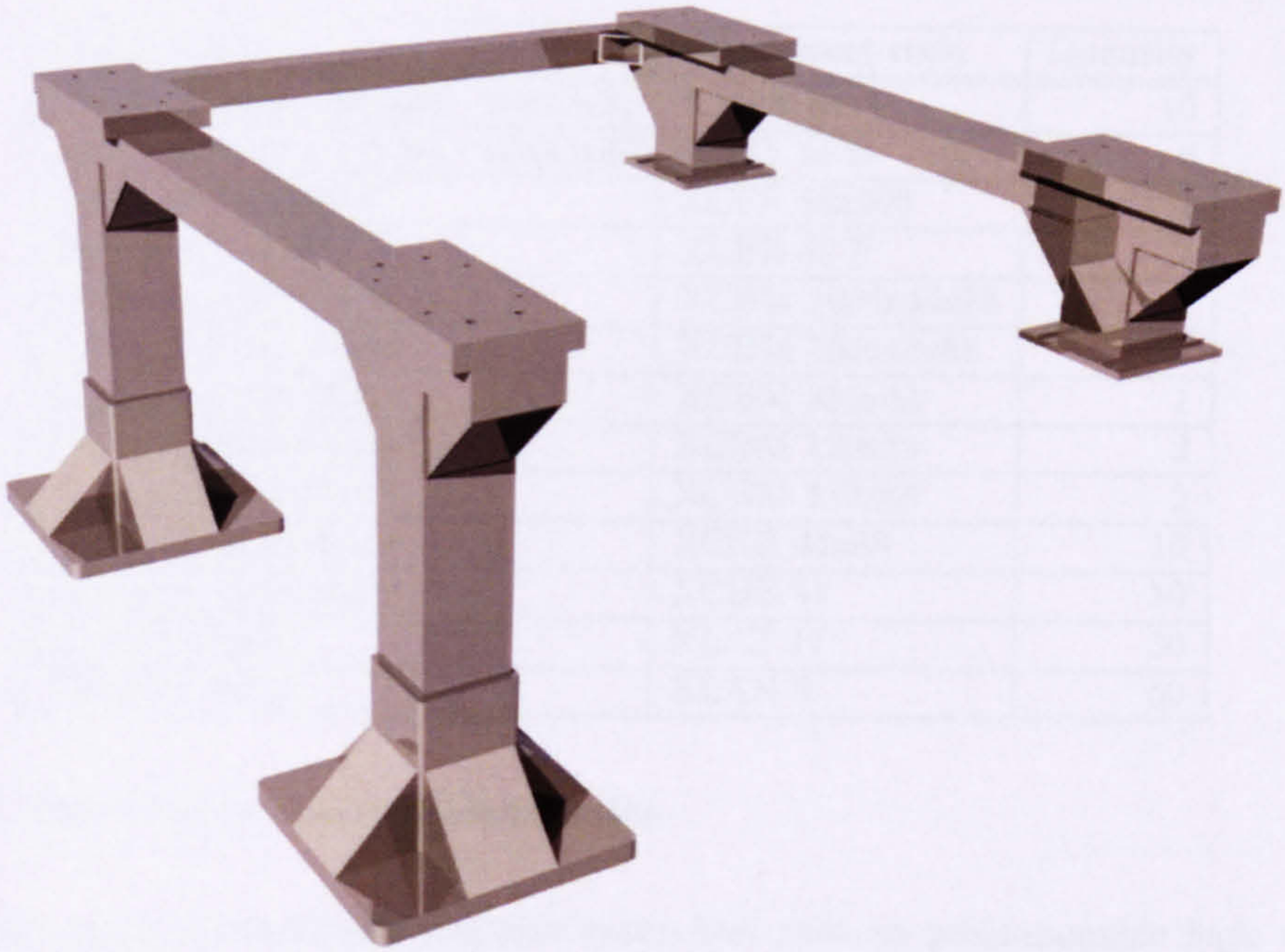


FIGURE 3.3: 3D design of the gantry support stand

The significance of this flexibility is important especially when considering the stability of ILC algorithms and is discussed in relation to obtained results in Chapters 4 and 5.

The axis position measurement, required for closed loop control and for evaluation of tracking performance, is determined by means of optical, quadrature, incremental encoders. Renishaw RGH22 linear encoders with a resolution of 1 μm are used for the X and Y-axes, while a 1000 pulse per revolution (ppr) (4000 counts per revolution (cpr) in quadrature) Aerotech rotary encoder is used for the Z-axis, giving a resolution of 0.5 μm.

The three axes are powered by performance matched Aerotech model BA10 linear amplifiers. The amplifiers are operated in velocity control mode and therefore the signals from the encoders must be fedback to the amplifiers. In order to determine the motor speed setpoint, the amplifiers require a ± 10 Volt control signal, supplied by the main control hardware. The amplitude of the control determines speed, while the sign specifies direction of motion. In effect, the complete control system consists of two closed loops operating together as shown in Figure 3.5. The controller calculates the set-point, while the drive performs local velocity control.

TABLE 3.1: Gantry support stand components - Flexlink

Description	Component code	Quantity
Angle bracket - diecast - 80x80x82	XMFA 84 A	10
Angle bracket - diecast - 42x42x39	XLFA 44 B	6
Large foot - diecast	XCFF 88x260	2
Foot plate for 88x88	XCFB 88 F	2
Support beam 44x88	XCBM 1024x44x88	2
Support beam 44x88	XCBM 150x44x88	2
Support beam 44x44	XCBM 882x44	1
Support beam 88x88	XCBM 120x88	2
Support beam 88x88	XCBM 432x88	2
End cap - polyamide	XCBE 44x88	10
End cap - polyamide	XCBE 44	10
T-bolts 17mm	XLAT 17	50
M8 Lock nuts	XLAN 8	50

3.2.2 Hardware: Control electronics

There are many commercially available controllers such as programmable logic controllers, industrial computers, digital signal processors and single chip micro-controllers, capable of controlling the gantry robot. However, these tend to have built-in control functions, typically limited to the three-term or PID controller. More advanced devices use their own programming language, consisting of basic functions, which allow the operator to program the control technique within restrictive limits. Neither of these approaches is suitable for the implementation of ILC because many required techniques (for example: matrix manipulation and dynamic memory management) are not readily available. This suggests that the algorithm should be implemented with some form of computer-based programming language, providing the flexibility to program any form of algorithm, and able to access the computer's own memory structure for saving data from previous iterations, which is a key requirement of ILC. The 'C' language is a particularly suitable choice for implementing real-time control because calculations and manipulations are highly optimised and are performed in a minimum number of processor clock cycles.

The resulting control system used during experimentation consists of a Pentium 4, Desktop PC, running the Linux operating system and using 'C' for software development. Linux is also particularly well suited to real-time control applications because interrupt handling routines generate significantly shorter delay latencies than other operating systems. The $\pm$10 Volt control set-points are generated by a 16 channel, 12-bit Digital to Analogue (D/A) expansion card, while the encoder generated signals are read by a separate 4 channel, quadrature decoder expansion card. Strict sample interval timing is achieved by means of a third, binary input expansion card with an on-board timer capable of generating interrupts at precise intervals. Interrupts are generated at the required sampling frequency. The computer must then service each interrupt and calculate the

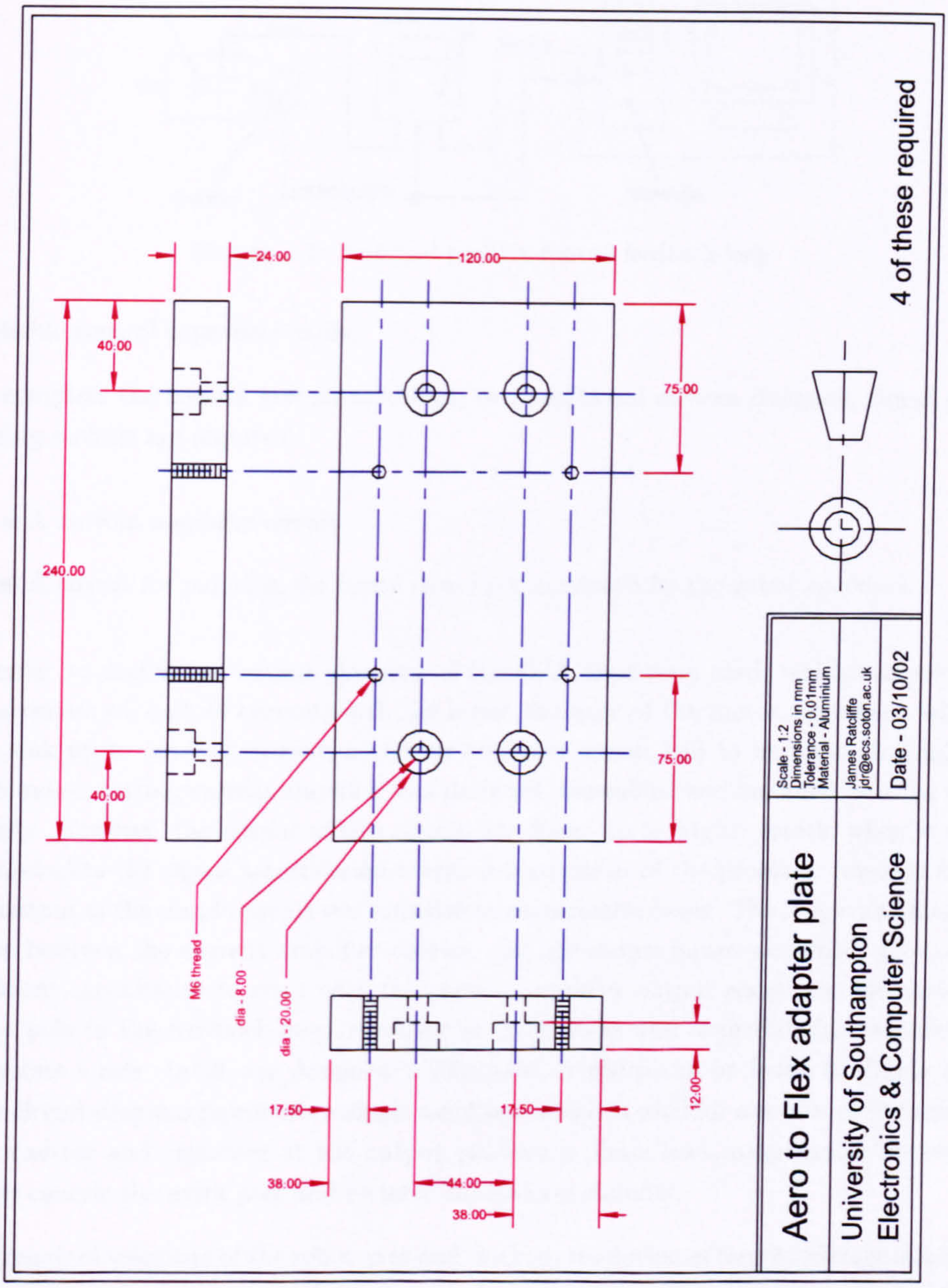


FIGURE 3.4: Aerotech to Flexlink mounting adapters

next control set-point in the time period of one sample instant, before the next interrupt occurs. This open architecture control system using separate cards for each task involved in closed-loop control, allows the computer based controller to be very flexible. The software programmer also has direct control over every aspect of the closed-loop system, rather than relying on pre-programmed routines commonly supplied with commercially

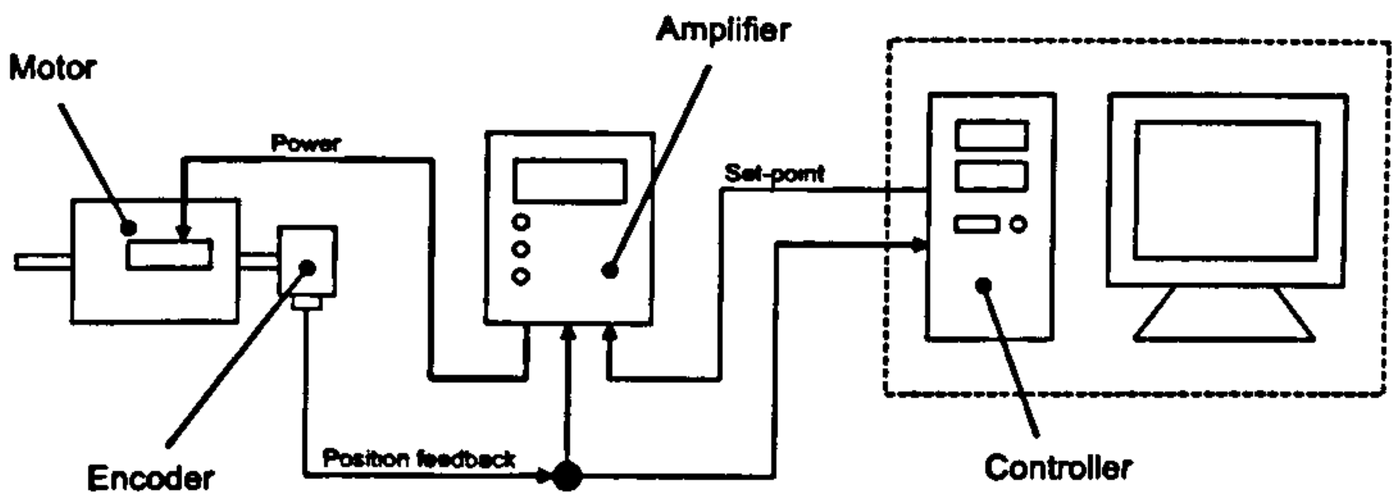


FIGURE 3.5: Standard position control feedback loop

available control expansion cards.

To complete the control system hardware, two additional custom designed, signal processing circuits are required:

- A current amplifier circuit.
- A circuit for reducing the count frequency produced by the robot encoders.

In order to match the output circuitry of the D/A expansion card, which can supply a maximum of 2mA of current, with the input circuitry of the motor amplifiers, which can sink up to 5mA of current, a current amplifier circuit had to be built. Initially a basic non-inverting current amplifier was designed, assembled and installed into the test facility. However, the output of this circuit was found to be highly erratic when it was connected to the signal transmission wires. Investigation of the problem, revealed that the output of the simple circuit was sensitive to capacitative loads. The long transmission wires between the current amplifier circuits and the motor power amplifiers produced sufficient capacitance to react with the current amplifier output resistance and add an extra pole in the feedback loop, resulting in oscillations and unpredictable behaviour. Therefore a new circuit was designed, a schematic of which can be found in Figure 3.6. This circuit diagram represents a single amplifier, which is used for one axis of the robot. The resistor and capacitor at the output produce a phase lead compensator network, which cancels the extra pole and restores closed-loop stability.

The required velocities of the robot axes and the high resolution of the encoders produce a pulse train of frequency too high for the encoder expansion card to read. In consequence, a frequency reducing circuit has been designed which reduces the count frequency. Figure 3.7 shows a schematic of the electronic circuit for one encoder. The count frequency is reduced by a factor of 32. Therefore the position resolution, as seen by the computer, is 32 times less detailed than the encoder resolution, resulting in 32μm for the X and Y-axes and 16μm for the Z-axis. The main difficulty which arose in designing this circuit was to reduce the count frequency while preserving the A and B channel phase difference. The phase information is used by the expansion card to determine which direction the motor is travelling in, and is therefore an essential element of the feedback loop. The differential

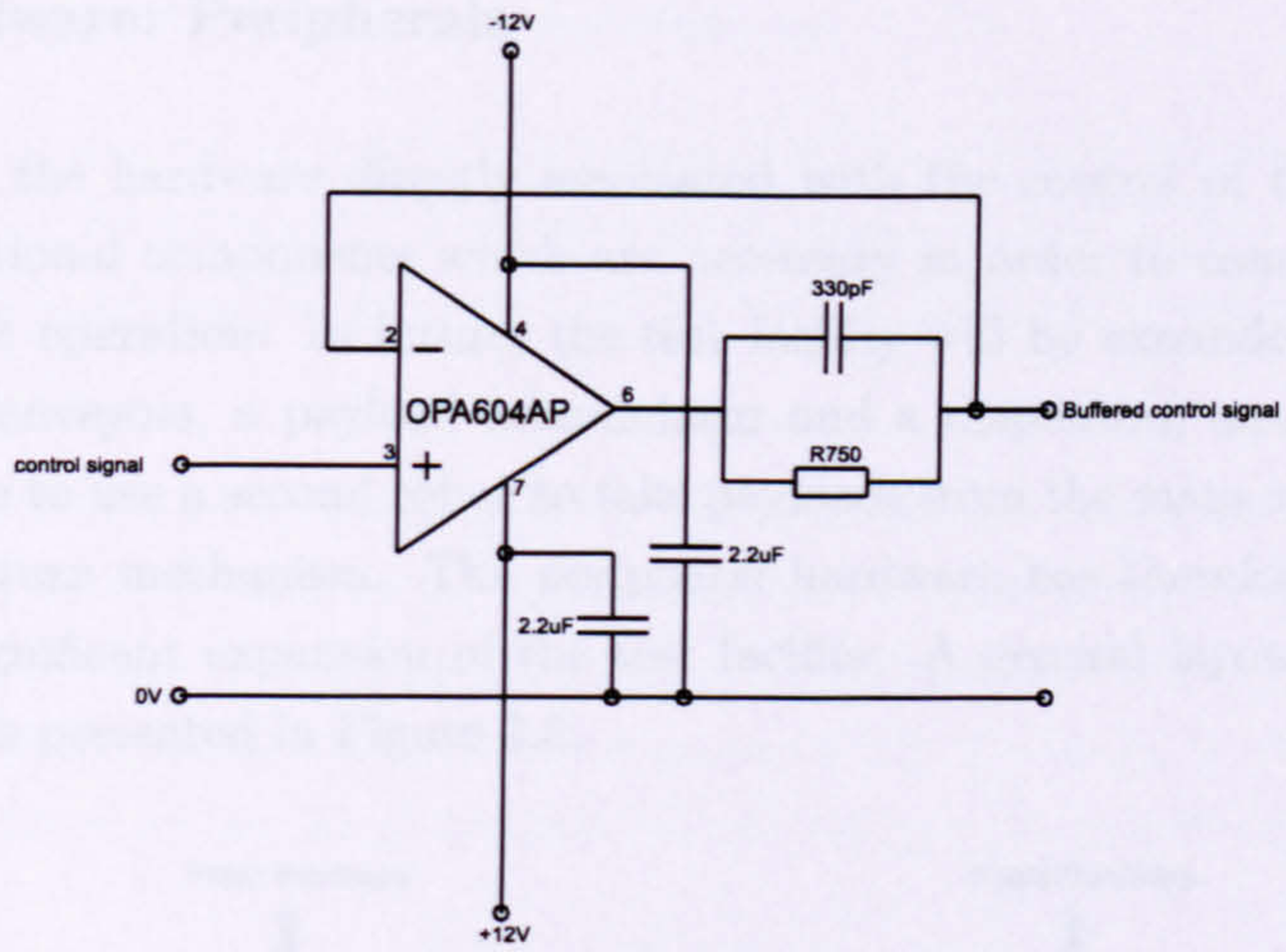


FIGURE 3.6: Control set-point amplifier circuit

amplifiers on the input (left) of the circuit convert the RS422 signal produced by the encoders into a single ended 0-5V signal. The sequence of five J-K flip-flops arranged in a line reduce the input frequency by a factor of 32, each flip-flop reduces the input frequency by a half. The remaining J-K flip-flop, NOT gate and Exclusive OR (EOR) gates adjust the relative phase between the outputs A^* and B^* to match the phase of the inputs A and B.

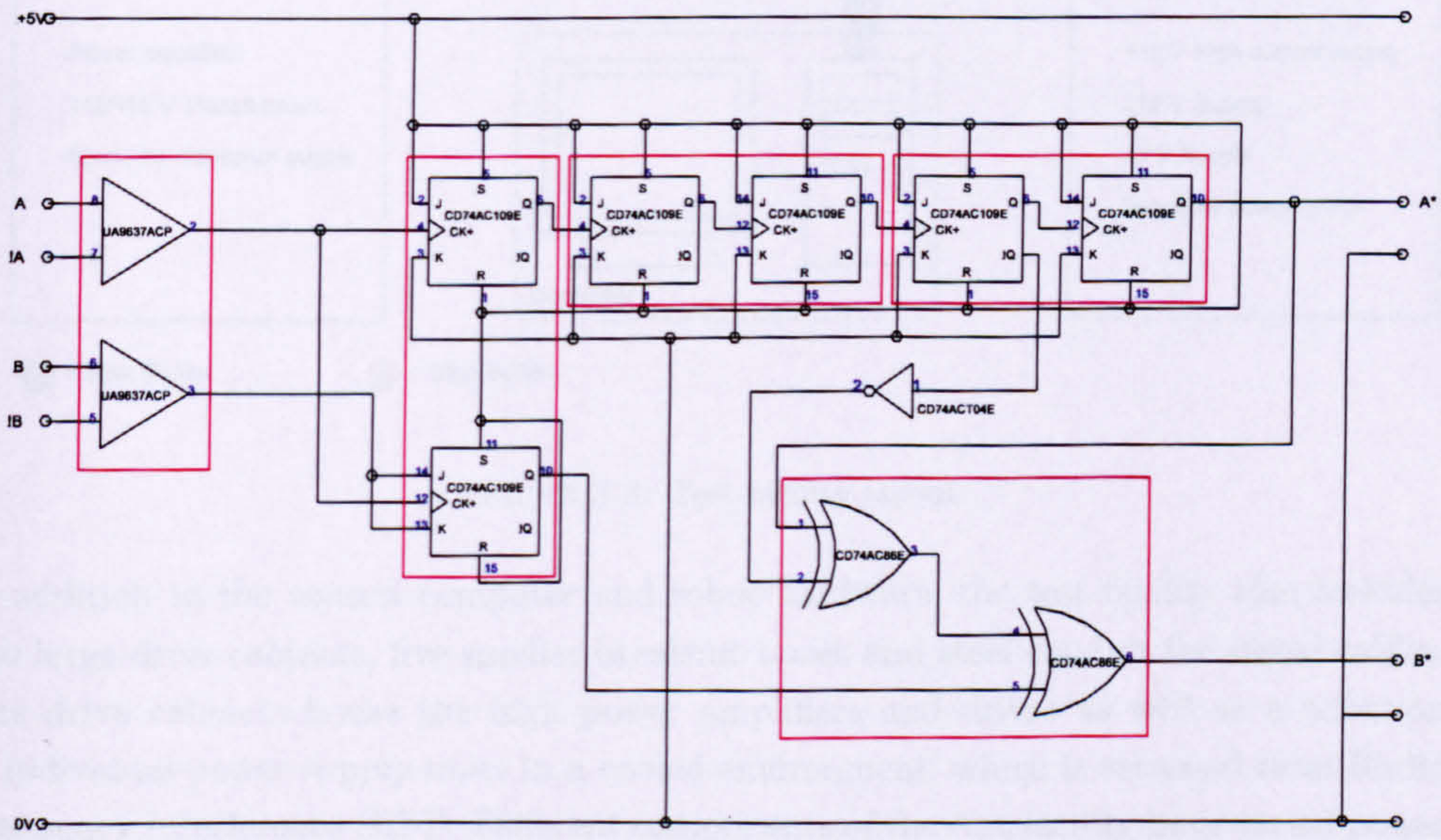


FIGURE 3.7: Encoder conditioning circuit

3.2.3 Hardware: Peripherals

In addition to the hardware directly associated with the control of the gantry robot, there are additional components which are necessary in order to complete the system and ensure safe operation. In future, the test facility will be expanded to include two plastic chain conveyors, a payload accumulator and a dispensing mechanism. It may also be possible to use a second robot to take payloads from the main conveyor and feed them into a return mechanism. The peripheral hardware has therefore been designed to allow for significant expansion of the test facility. A general layout of the existing infrastructure is presented in Figure 3.8.

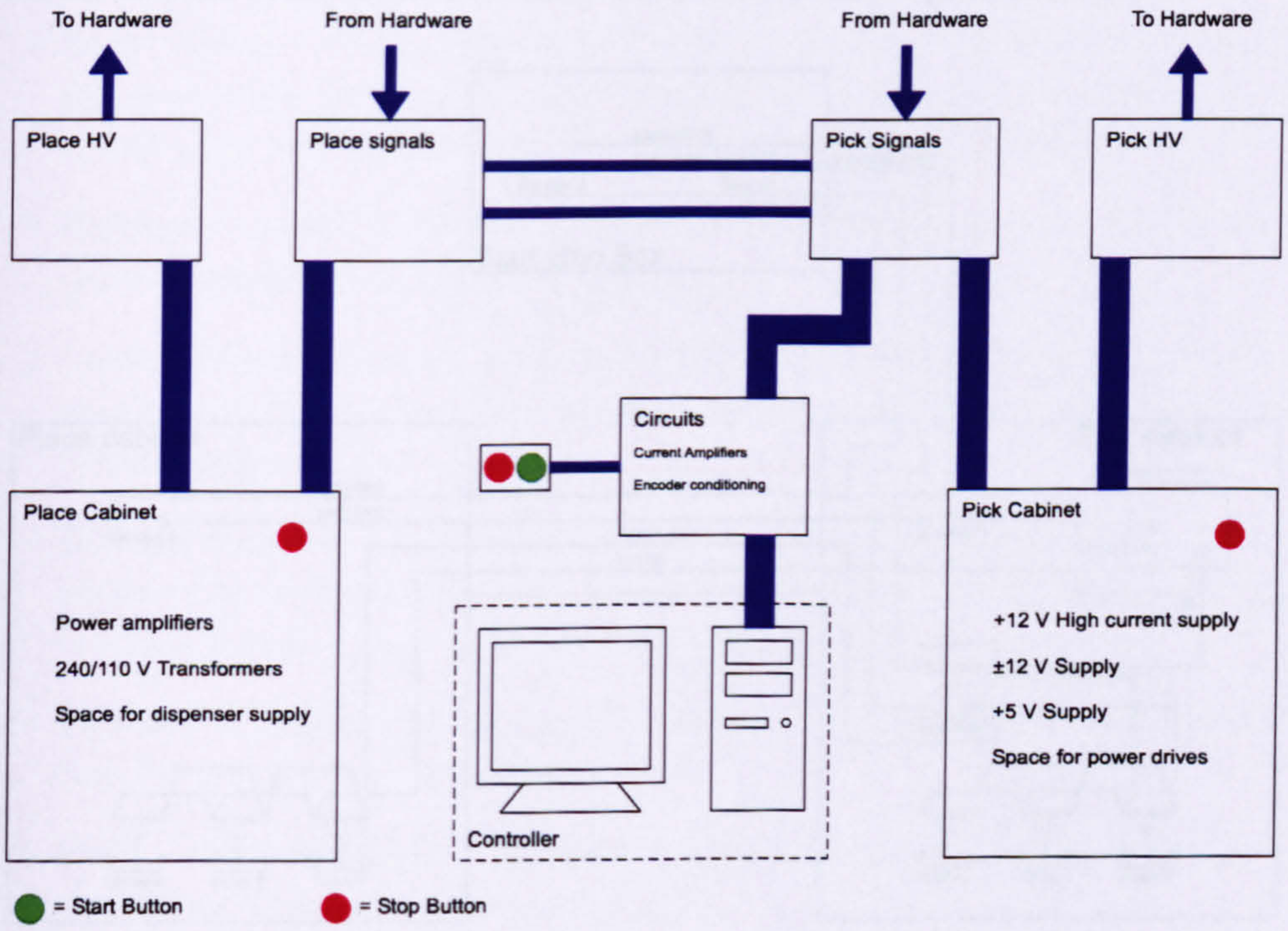


FIGURE 3.8: Test facility layout

In addition to the control computer and robot hardware, the test facility also includes two large drive cabinets, five smaller breakout boxes and steel conduit for signal cables. The drive cabinets house the high power amplifiers and drives as well as a selection of individual power supply units in a cooled environment, which is screened from Radio Frequency Interference (RFI). Different components of the test facility have varied power supply requirements, ranging from 240V alternating current (ac) for cooling fans and low voltage power supplies, 110V ac for the motor amplifiers, ±12V direct current (dc) for the control signal amplifier circuits, +12V dc for the main ignition and emergency stop circuit and +5V for the encoder circuits. The drive cabinets have been named 'Pick' and 'Place' to reflect the operation which a robot would perform at that location along the

conveyor; Picking payloads off, or placing them on.

The breakout boxes serve as connection points to which external hardware such as robots and conveyors can be connected. They also act as signal routing points, where signals and power cables from different components of the test facility can be connected to multiple other components.

Signal and power lines are run in separate, earthed conduit and each cable is screened to help minimise RFI from being transmitted from power lines to signal lines. A combined ignition and emergency stop circuit (see Figure 3.9) runs between the computer and the drive cabinets and controls the main three-phase 415V ac supply to the entire system, ensuring that the plant can be shut down completely should a problem arise.

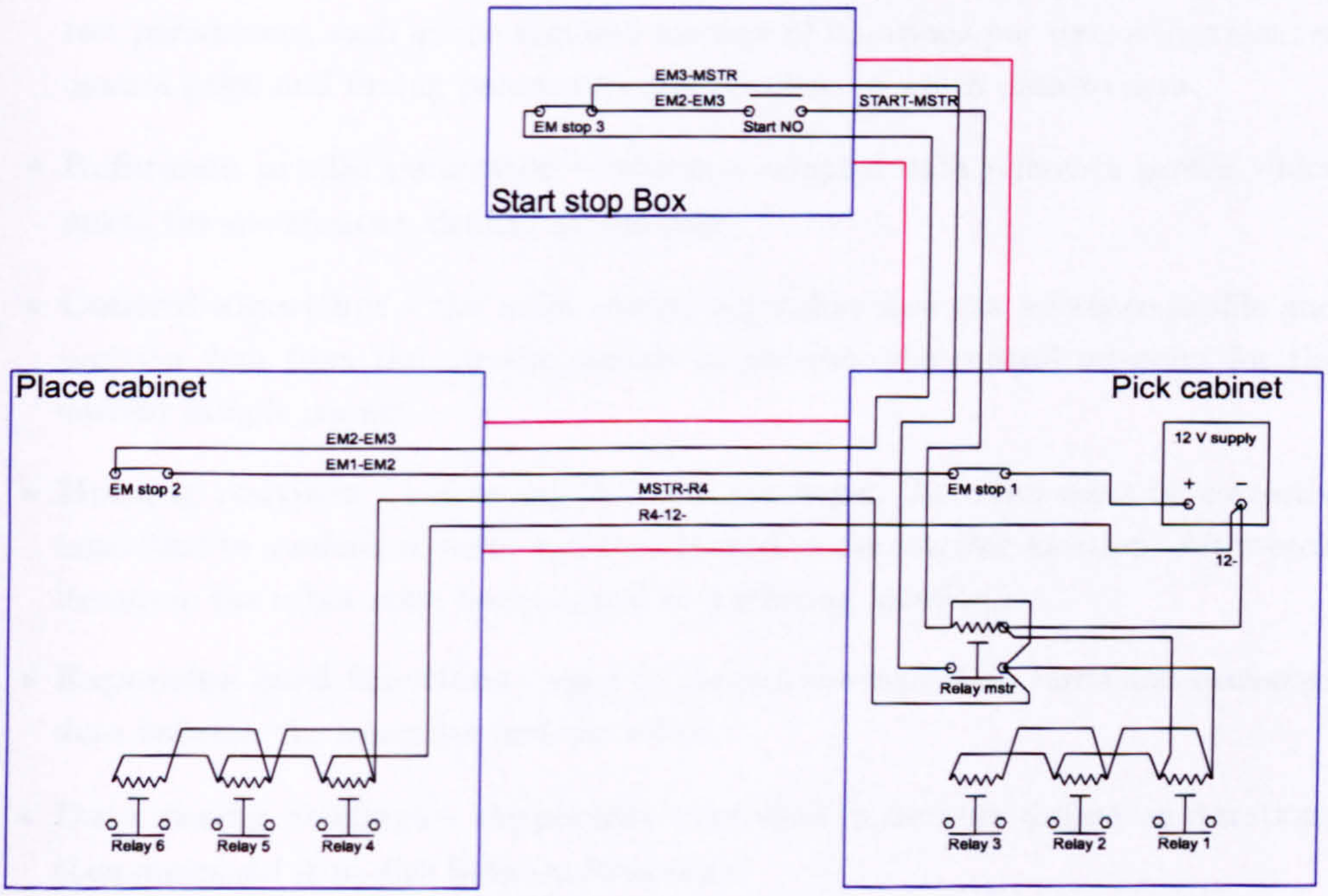


FIGURE 3.9: Ignition and emergency stop circuit

The ignition/stop circuit is powered from a 12V dc supply and uses one master relay (Relay mstr) in the Pick cabinet to control six slave relays. There are three slave relays in each drive cabinet, one for each phase of the 415V grid supply. The slave relays mimic the state of the master relay, if the master is energised, all six slave relays are also. As a standard safety precaution, the three stop buttons operate in the normally closed position. In this way, a fault in the ignition circuit immediately shuts down the plant. The start button operates in the normally open position and has a spring mechanism which maintains this state unless the button is pushed and held down. Therefore, the master relay uses a self latching circuit to remain on once the start button is released.

3.2.4 Software development

A highly modular approach has been used for developing the control software. The control task can be broken into a number of different sections, most of which are not directly associated to the control algorithm, but are required to control the expansion cards and correctly manage the data generated by the test facility. The advantage of this approach is that the implementation of a new algorithm requires changes to only a few elements of the software. This reduces development time and simultaneously increases program reliability and stability because there is less scope for creating bugs or errors. The different modules are:

- **User Menus** – simple, text-based screens which allow the operator to specify test parameters, such as the required number of iterations per test, adjustment of control gains and tuning parameters and selection of which data to save.
- **Reference profile generator** – creates a sampled data reference profile which meets the specification defined by the user.
- **Control algorithm** – the main control algorithm uses the reference profile and position data from the encoder signals to generate the control set-point for the current sample instant.
- **Homing routines** – before any ILC test can begin, the robot must be correctly initialised to a reference point and then homed to the starting location. After each iteration, the robot must be re-homed to a starting location.
- **Expansion card functions** – used to operate the expansion cards and exchange data between the computer and the robot.
- **Data saving routines** – temporarily store data in memory during an iteration, then download it to disk between iterations.
- **Matrix manipulations** – including matrix multiplication, addition, subtraction and inversion are required for model-based algorithms.
- **Miscellaneous Functions** – such as emergency stop shut-down procedures, random number generation and timed delay generation.
- **Core program** – this module manages all of the other modules, sequences them correctly and runs the program.

Figure 3.10 shows the software flowchart which details the sequence of steps performed by the program, from startup through to test completion. With reference to Figure 3.10, initially the program obtains user specified information and test constants, such as control gains and tuning parameters. Next, the expansion cards are initialised and the

reference profiles are generated, based on the user specified information. The program is now ready to prepare the robot for testing and then perform the test. At this stage, two program loops define the sequence of events which runs the test. The outer loop applies to each iteration, while the inner loop applies to all of the sample instants within an iteration. The robot is homed to a specified starting location, and the program verifies if the required number of iterations has been performed. If the answer returns 'yes', the system is shutdown safely. If not, the program enters the inner sample loop and generates a new control set-point. Once the required number of samples for one iteration is achieved, the program returns to the outer loop and verifies if the required number of iterations has been met. The process continues until all iterations are complete.

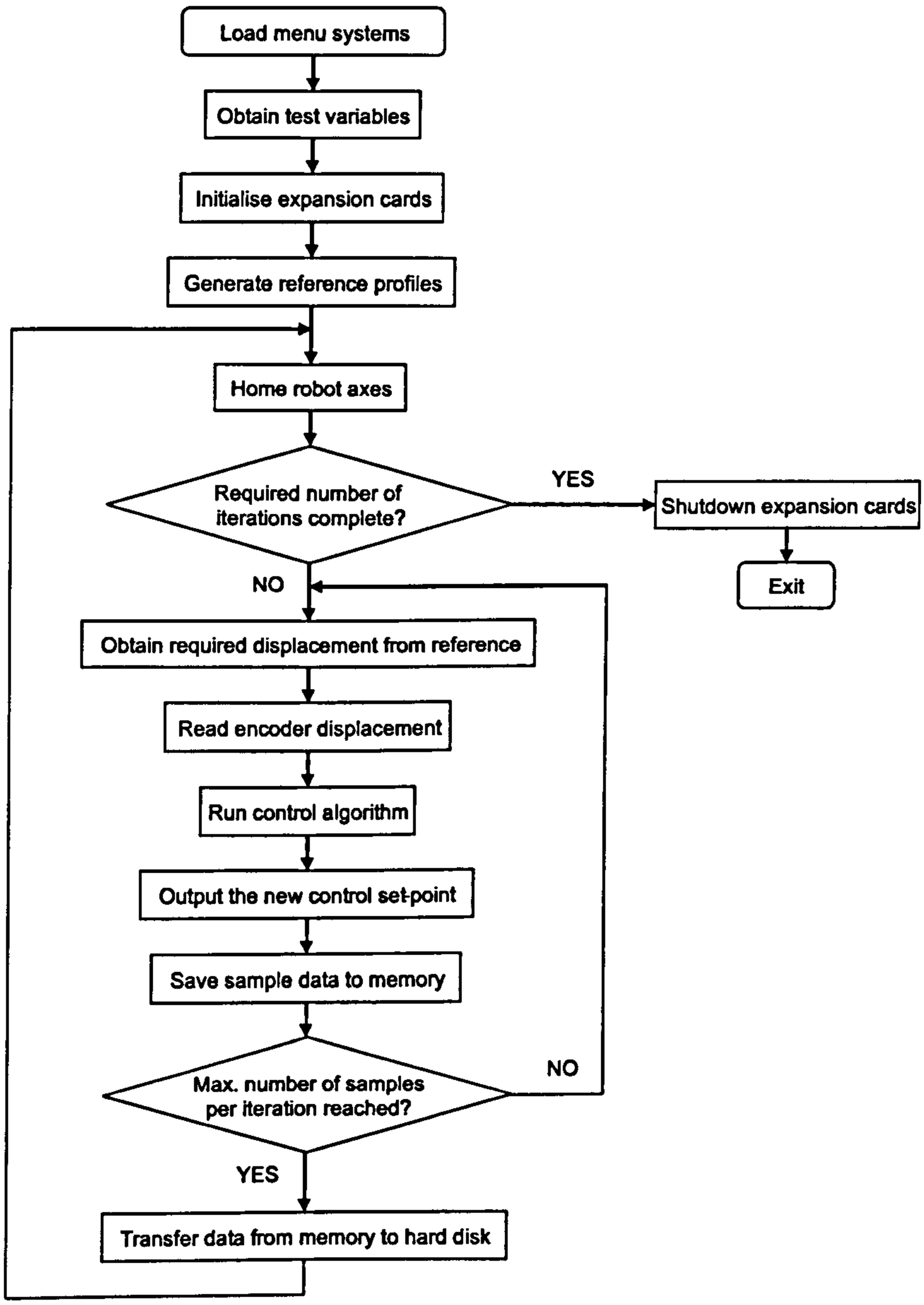


FIGURE 3.10: Software flowchart

3.3 Plant Modelling

An essential aspect of any control system implementation is the development of mathematical models which adequately describe the dynamic behaviour of the plant which is to be controlled. For more advanced algorithms, system models are an essential component of the real-time controller. But, accurate models also allow the control system to be designed, tested and refined in a simulation environment, shortening development time and reducing the risk of the controller damaging the real plant by supplying unsuitable inputs.

Control system designers generally use one of two approaches to model plant dynamics. These are the frequency-domain, transfer function and the time-domain, state-space representations. From a plant modelling perspective, the state-space approach tends to be more suitable when specific parameters pertaining to the plant such as mass, stiffness, friction and inertia are known or can be measured. State-space methods are well suited to describing non-linear plant dynamics, while transfer function models are only appropriate for linear plants. The transfer function approach tends to treat the plant itself as a 'black box' and is concerned only with the relationship between the input and output. This makes the transfer function approach more suitable for the axes of the gantry robot, for which no physical data is known.

Frequency domain modelling invariably involves supplying a known input to the plant and measuring the output. A mathematical function can then relate the input and output sequences. Commonly used input sequences are sine-waves, white noise or coloured noise. The sine-wave, spectrum approach has been used in this project. If a sine-wave of known frequency and amplitude is supplied to an open-loop, linear system, the output will also be a sine-wave, but with a different amplitude and shifted phase. The ratio of the input to output amplitudes is recorded as gain in decibels (dB) while the phase shift is recorded in degrees. If a sufficiently large range of frequencies are tested, a Bode plot representing the frequency domain response of the plant can be produced. Key features of the Bode plot can indicate plant dynamics such as poles, zeros and resonances. Using the Bode plotting rules in reverse it is possible to construct an approximate transfer function of the plant by hand.

As each axis of the gantry robot is an integrating plant, each transfer function has a pole at the origin, implying that the plant is not open loop stable. To perform open loop sine-wave spectrum tests, the axes were homed to a known starting position before each frequency was tested. For low frequencies, the amplitude of the input sine-wave had to be made sufficiently small to ensure that the axes remained within their travel limits. As the sine-wave frequency increased, the input amplitude could also be increased to compensate for the reduced open loop plant gain. 127 different frequencies ranging from 0.628rad/s (0.1Hz) to 477rad/s (76Hz) were used to generate the X-axis Bode plot, while 129 frequencies from 0.628rad/s (0.1Hz) to 879rad/s (140Hz) and 139 frequencies from

0.628rad/s (0.1Hz) to 1005rad/s (160Hz) were used for the Y and Z-axes respectively. Having developed approximate transfer functions by hand, the models were then refined using a least-mean-square optimisation technique to minimise the error between the measured gain response and the transfer function gain response.

The resulting transfer function model for the X-axis is:

$$\begin{aligned} G_{X_{21}}(s) = {} & \frac{(s+3.64\times10^5)(s+500.21)(s+35.88\pm j400.76)}{s(s+69.73\pm j459.64)(s+30.53\pm j378.98)}\times\cdots \\ \cdots\times & \frac{(s+30.46\pm j336.23)(s+27.47\pm j249.8)(s+10.78\pm j223.56)}{(s+32.59\pm j297.99)(s+21.14\pm j239.36)(s+10.64\pm j220.02)}\times\cdots \\ \cdots\times & \frac{(s+14.06\pm j195.13)(s+10.59\pm j169.44)(s+8.83\pm j124.71)}{(s+10.67\pm j192.24)(s+10.45\pm j141.63)(s+8.51\pm j119.75)}\times\cdots \\ \cdots\times & \frac{(s+5.33\pm j106.87)(s+3.36\pm j83.93)}{(s+6.05\pm j86.78)(s+12.02\pm j79.09)} \end{aligned} \tag{3.1}$$

This is a 21st order system including numerous pairs of resonant poles and zeros. These resonances are largely caused by the flexibility of the support stand and laboratory bench on which the robot is mounted. A Bode plot comparing the measured frequency response and the frequency response of the model can be seen in Figure 3.11. The model matches the plant dynamics well.

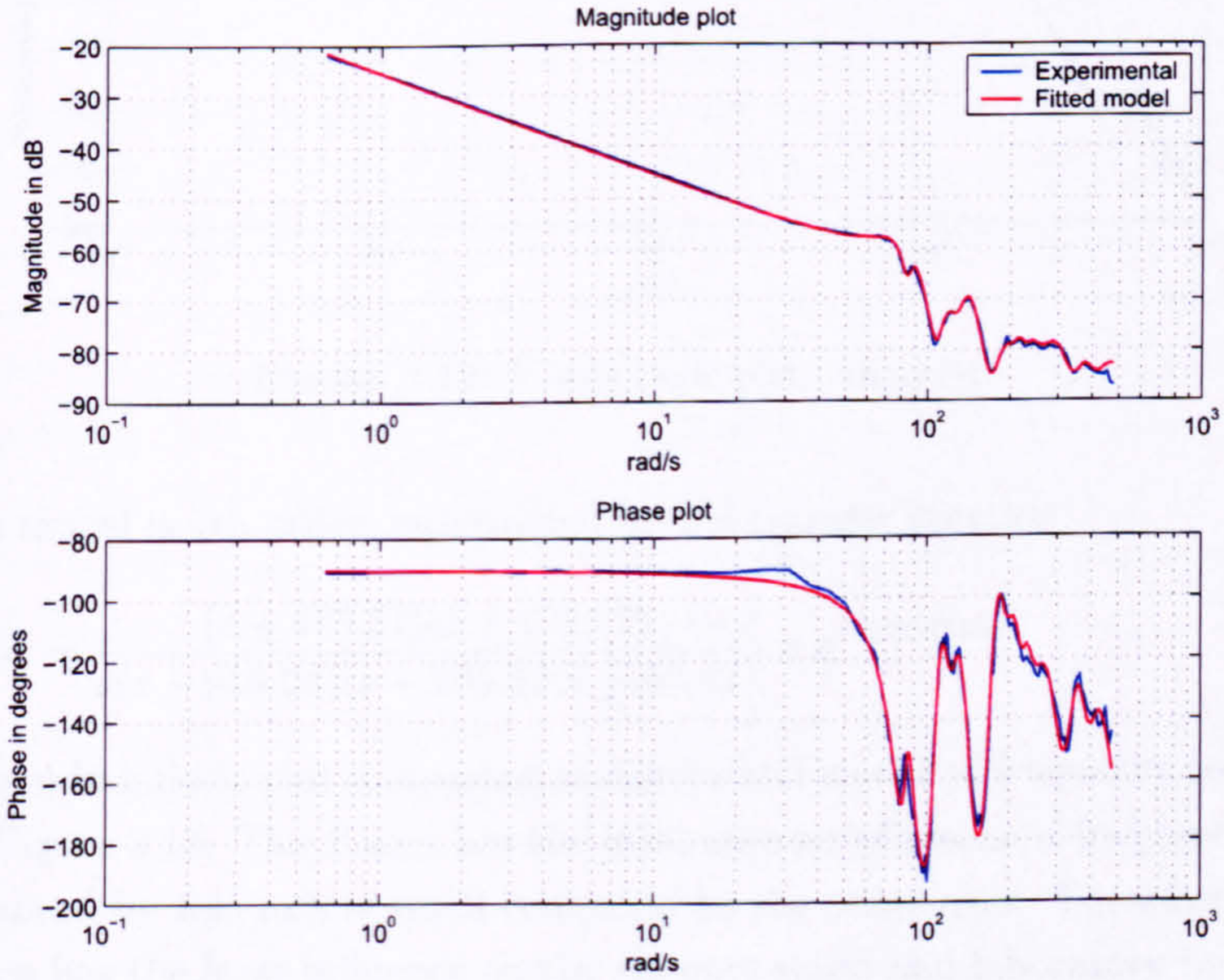


FIGURE 3.11: X-axis Bode plot, 21st order

The Y-axis model is 7th order, represented by the transfer function:

$$G_{Y_7}(s) = \frac{(s + 148.20)}{s(s + 78.54 \pm j533.34)} \times \cdots$$
$$\cdots \times \frac{(s + 49.24 \pm j526.52)(s + 42.06 \pm j87.01)}{(s + 213.42 \pm j151.47)(s + 43.31 \pm j87.19)} \times e^{-0.000726s} \quad (3.2)$$

The Bode plot in Figure 3.12 compares the measured and modelled frequency responses for the Y-axis. There are significantly fewer resonant frequencies in the dynamic response of the Y-axis than for the X-axis.

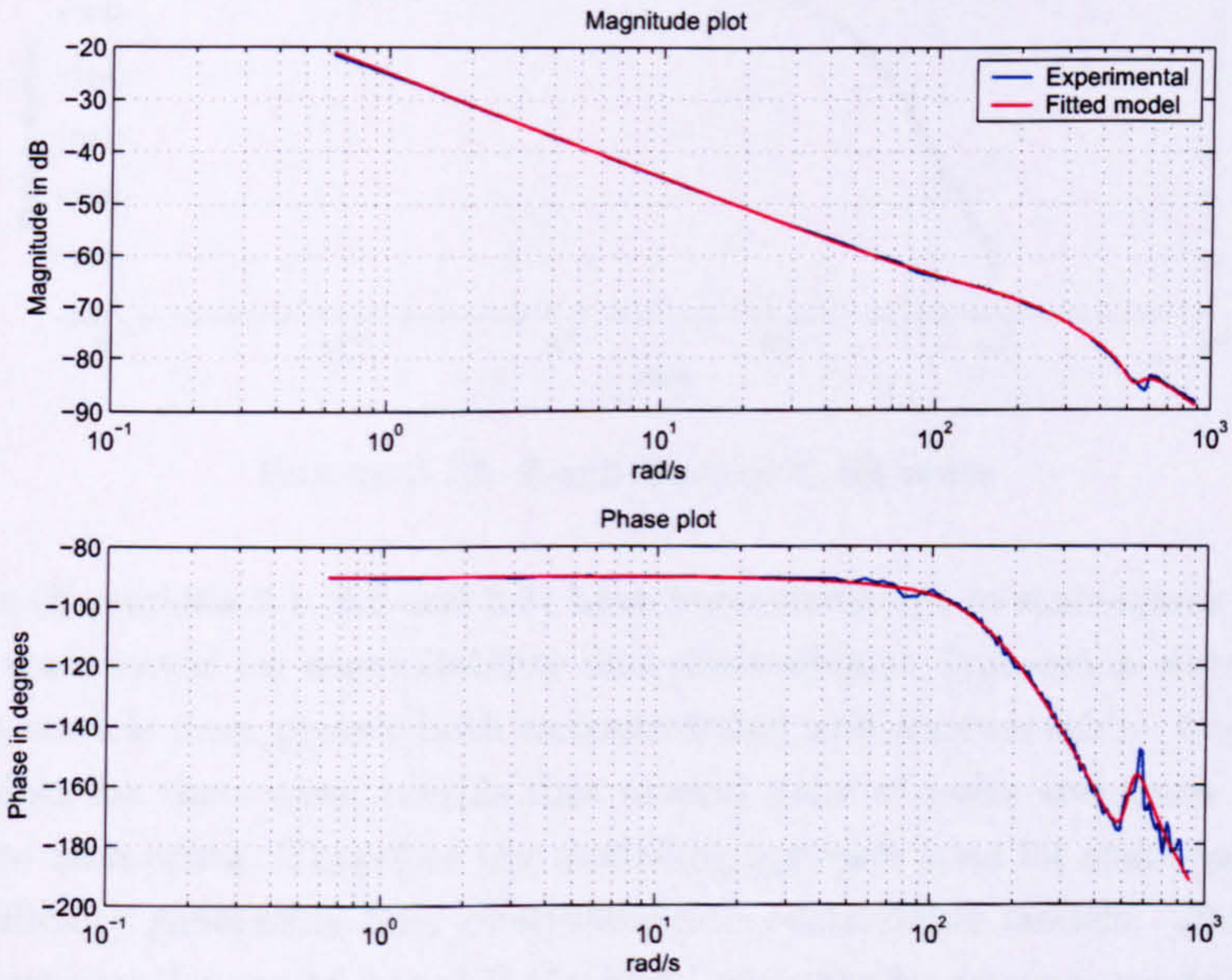


FIGURE 3.12: Y-axis Bode plot, 7th order

The Z-axis model is 4th order, represented by the transfer function:

$$G_{Z_4}(s) = \frac{(s + 473.51)(s + 199.02)}{s(s + 989.06)(s + 266.22 \pm j157.81)} \times e^{-0.000746s} \quad (3.3)$$

The corresponding Bode plot comparing measured and modelled frequency responses can be seen in Figure 3.13. The Z-axis has the least number of resonant frequencies because the mass moved by this axis is small compared to the other axes. Therefore movement of the Z-axis has the least influence on the support stand and laboratory bench.

Although transfer function models of the gantry robot are less complex to derive than state-space models, most model-based ILC algorithms are derived in the time-domain and therefore require state-space representations of the plant. However, by using suitable modelling software it is possible to convert the transfer functions into state-space. Firstly, the transfer function must be converted to discrete form, and secondly the discrete transfer function is converted to state-space.

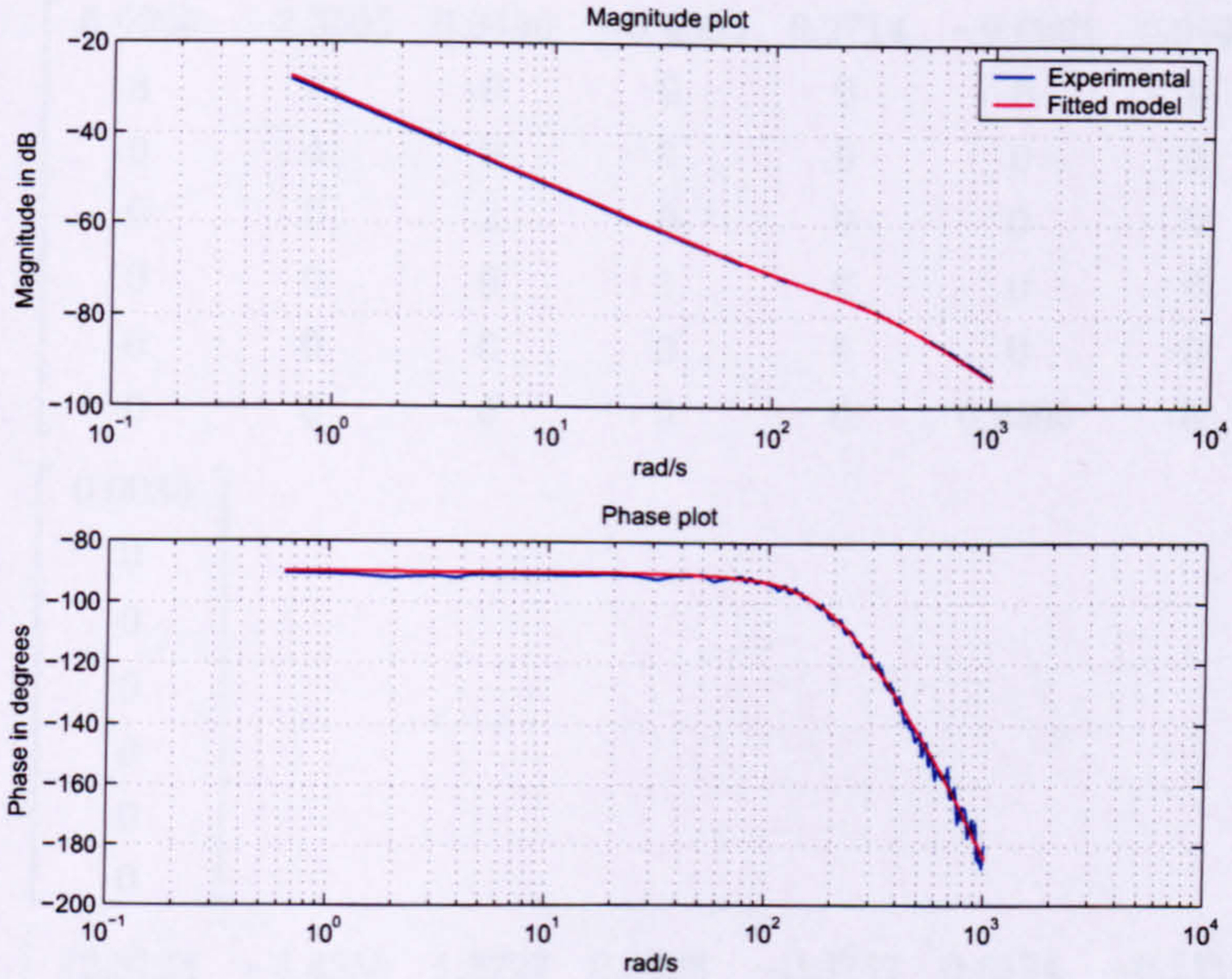


FIGURE 3.13: Z-axis Bode plot, 4th order

The models (Equations 3.1, 3.2 and 3.3) have been converted to state-space format and have been rank tested for controllability and observability. Numerous states of the X and Y-axis models have proven both uncontrollable and unobservable. Observing the pole-zero plots for these axes, reveals that several pairs of poles and zeros are located very close to each other. Therefore the modelling software used for state-space conversion has difficulty generating fully observable and controllable models. This creates a significant problem for model-based ILC which, intrinsically, requires models as part of the controller. In order to solve this problem, the order of the models has been reduced until all states are both controllable and observable. The small time delays present in the high order models have also been removed, as they are insignificant compared with the controller sample frequency. Because of this, the effect of time delays on system performance has not been considered in this thesis.

The model order reduction and removal of time delays results in the transfer function and 100Hz sample frequency state space models:

- X-axis 7th order model

$$G_{X_7}(s) = \frac{(s+500.19)(s+4.90\times10^5)}{s(s+69.74\pm j459.75)} \times \cdots$$
$$\cdots \times \frac{(s+10.99\pm j169.93)(s+5.29\pm j106.86)}{(s+10.69\pm j141.62)(s+12.00\pm j79.10)} \quad (3.4)$$

$$A_{X_7} = \begin{bmatrix} 6.6005 & -2.3565 & 0.9446 & -0.4595 & 0.2714 & -0.0901 & 0.0520 \\ 8 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.2500 & 0 \end{bmatrix} \tag{3.5}$$

$$B_{X_7} = \begin{bmatrix} 0.0039 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.6}$$

$$C_{X_7} = \begin{bmatrix} 12.8124 & -5.4858 & 1.2727 & 0.3408 & -0.9787 & 0.6174 & -0.5210 \end{bmatrix} \times 10^{-4} \tag{3.7}$$

$$D_{X_7} = \begin{bmatrix} 0 \end{bmatrix} \tag{3.8}$$

- Y-axis 5th order model

$$G_{Y_5}(s) = \frac{(s + 148.20)(s + 49.24 \pm j526.52)}{s(s + 78.54 \pm j533.34)(s + 213.42 \pm j151.47)} \tag{3.9}$$

$$A_{Y_5} = \begin{bmatrix} 4.1893 & -0.9049 & 0.4034 & -0.0926 & 0.0349 \\ 8 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0.5000 & 0 \end{bmatrix} \tag{3.10}$$

$$B_{Y_5} = \begin{bmatrix} 0.0039 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.11}$$

$$C_{Y_5} = \begin{bmatrix} 1.5218 & -6.3645 & 0.1684 & 1.2760 & -1.3748 \end{bmatrix} \times 10^{-4} \tag{3.12}$$

$$D_{Y_5} = \begin{bmatrix} 0 \end{bmatrix} \tag{3.13}$$

- Z-axis 4th order model

$$G_{Z_4}(s) = \frac{(s+473.51)(s+199.02)}{s(s+989.06)(s+266.22 \pm j157.81)} \tag{3.14}$$

$$A_{Z_4} = \begin{bmatrix} 2.8854 & -0.7589 & 0.3421 & -0.1092 \\ 4 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.5000 & 0 \end{bmatrix} \tag{3.15}$$

$$B_{Z_4} = \begin{bmatrix} 0.0039 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.16}$$

$$C_{Z_4} = \begin{bmatrix} 26.1368 & -4.4846 & -3.7900 & 5.0430 \end{bmatrix} \times 10^{-4} \tag{3.17}$$

$$D_{Z_4} = \begin{bmatrix} 0 \end{bmatrix} \tag{3.18}$$

These models are referred to as 'high order' in this document. The Bode plots comparing the dynamic response of these new models with the experimental data can be found in Figures 3.14, 3.15 and 3.16 for the X, Y and Z-axes respectively. The modelling error introduced by model order reduction is more noticeable in the phase responses than for the gain responses. However, the additional error is minimal, compared with the advantage of having full state controllability and observability.

Certain robustness tests described in Section 3.6.4 deliberately use simplified plant models to investigate the robustness of ILC controllers. These models are first order representations of the plant consisting of a gain and an integrator. In each case, the gain of the first order model has been chosen to approximate the gain of the plant at low frequency. The associated transfer function and 100Hz sample frequency state-space models are:

- X-axis 1st order model

$$G_{X_1}(s) = \frac{0.05}{s} \tag{3.19}$$

$$A = \begin{bmatrix} 1 \end{bmatrix} \tag{3.20}$$

$$B = \begin{bmatrix} 0.0078 \end{bmatrix} \tag{3.21}$$

$$C = \begin{bmatrix} 0.0064 \end{bmatrix} \tag{3.22}$$

$$D = \begin{bmatrix} 0 \end{bmatrix} \tag{3.23}$$

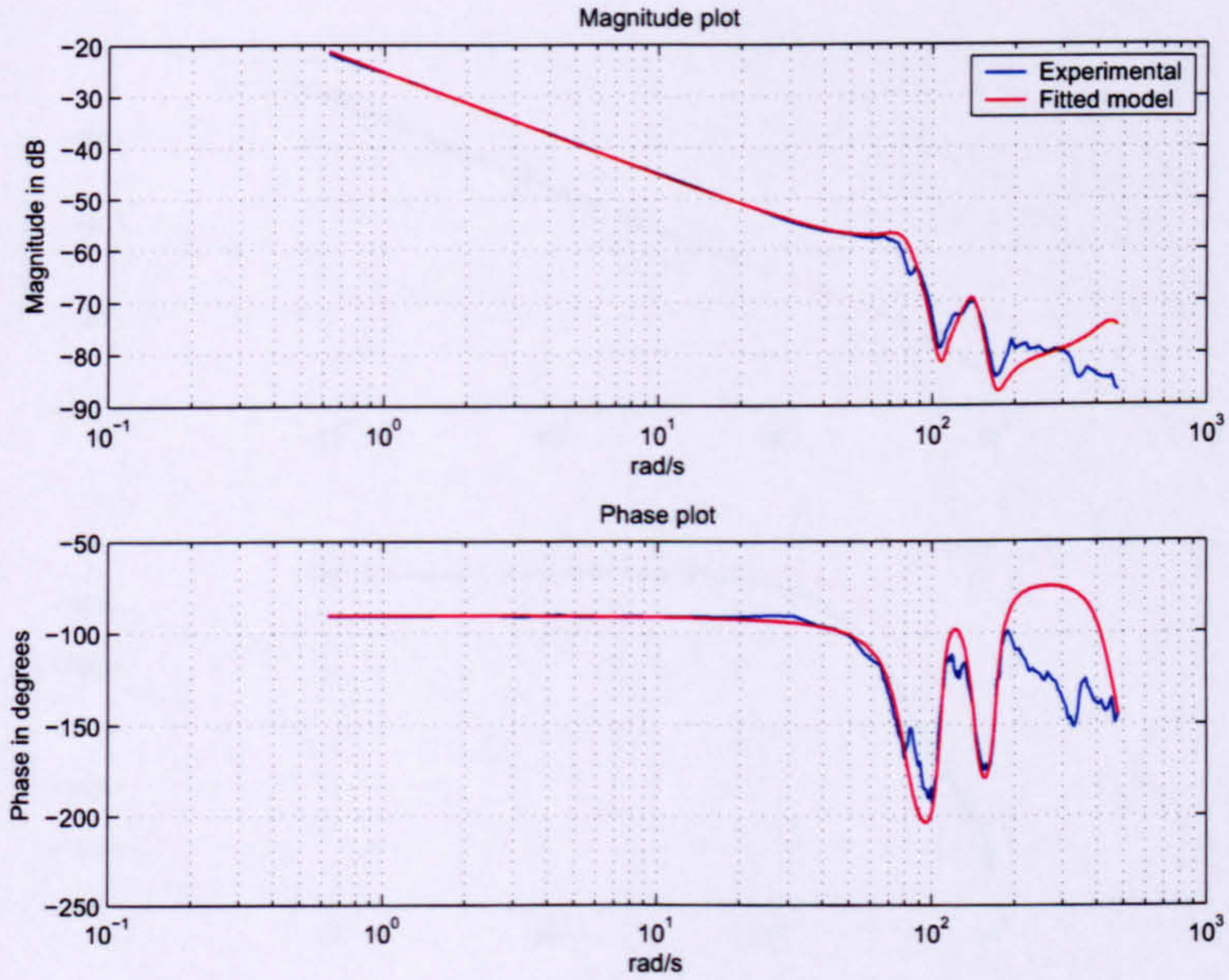


FIGURE 3.14: X-axis Bode plot, 7th order

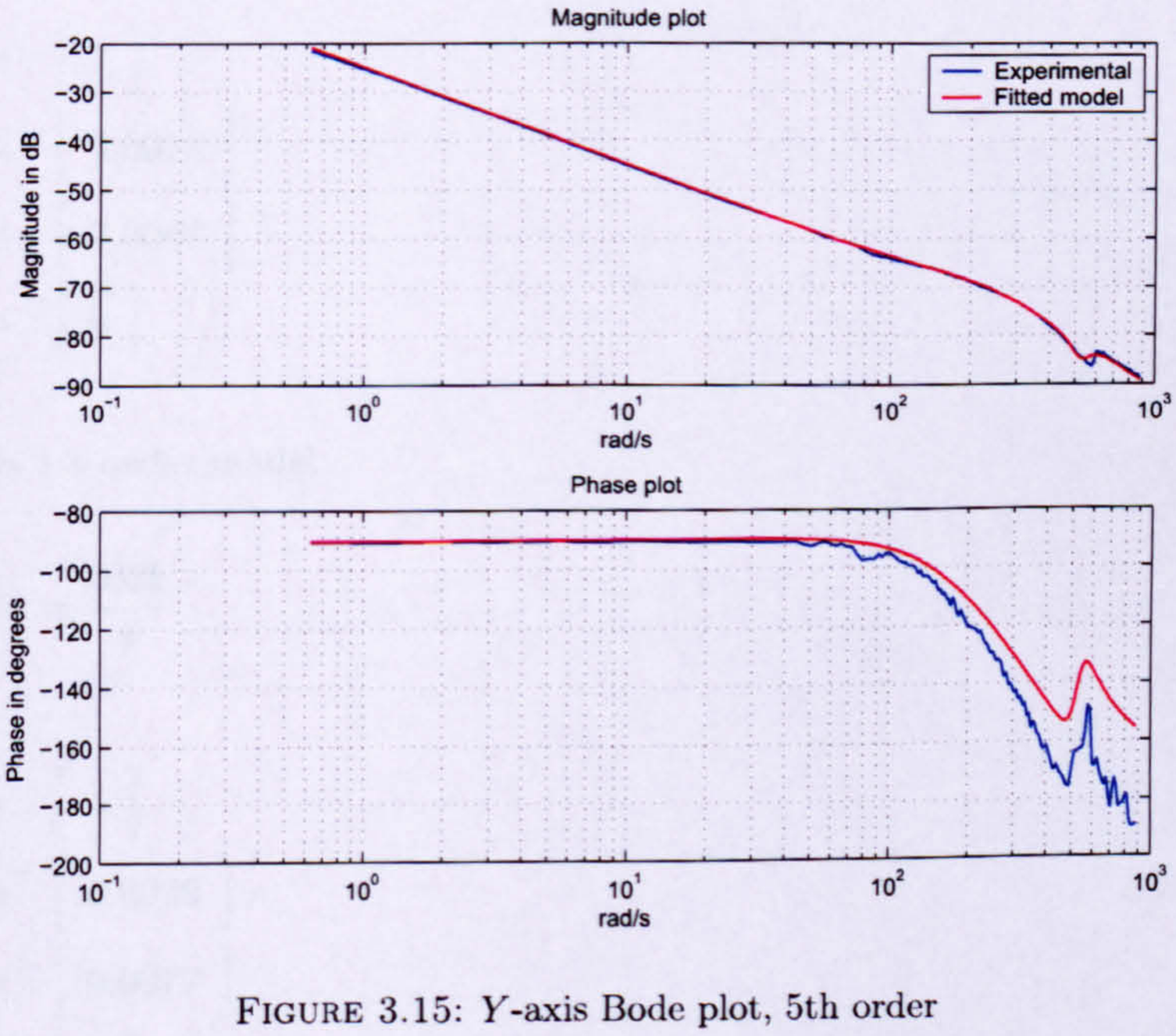


FIGURE 3.15: Y-axis Bode plot, 5th order

- Y-axis 1st order model

$$G_{Y_1}(s) = \frac{0.05}{s} \tag{3.24}$$

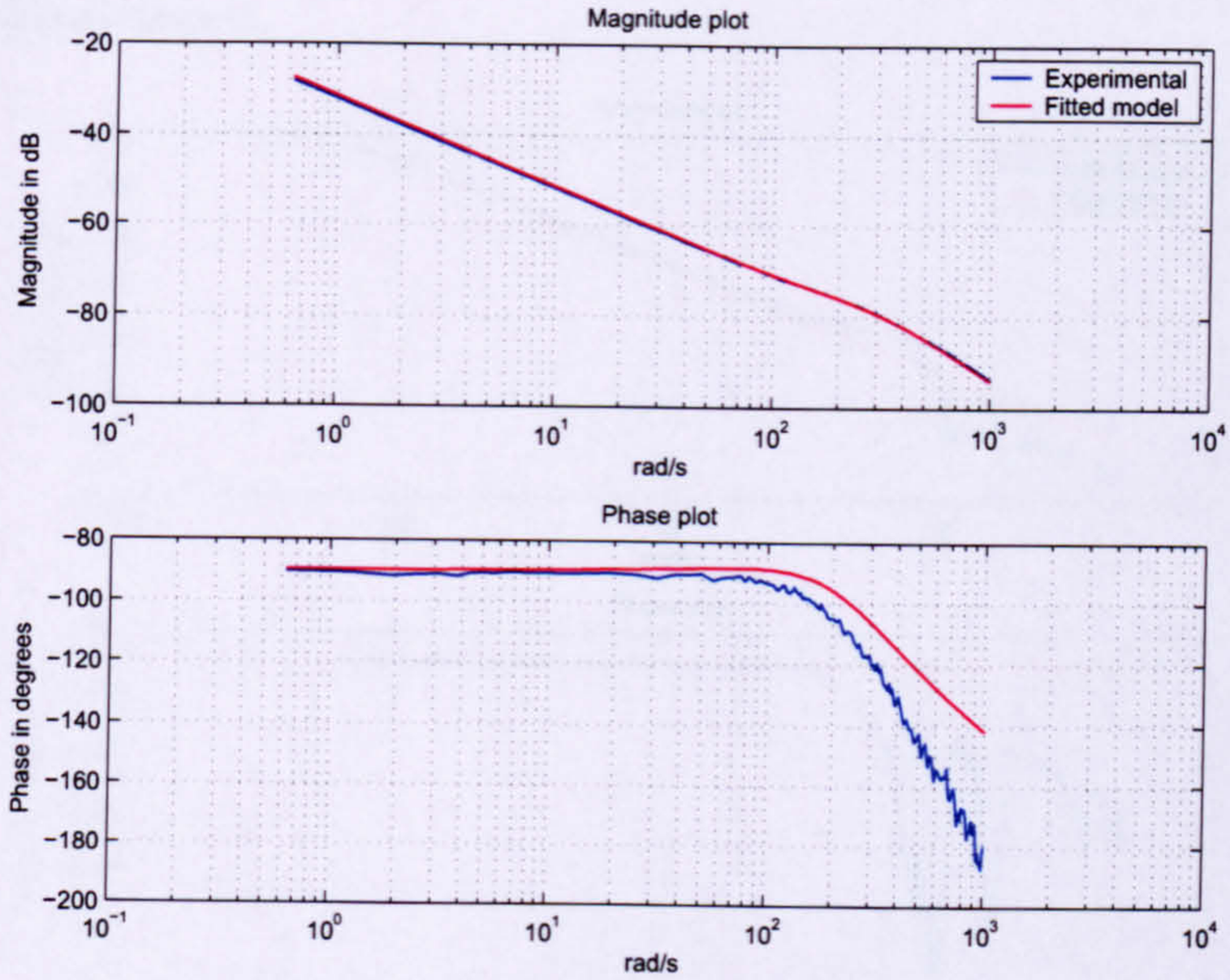


FIGURE 3.16: Z-axis Bode plot, 4th order without time delay

$$A = \begin{bmatrix} 1 \end{bmatrix} \tag{3.25}$$

$$B = \begin{bmatrix} 0.0078 \end{bmatrix} \tag{3.26}$$

$$C = \begin{bmatrix} 0.0064 \end{bmatrix} \tag{3.27}$$

$$D = \begin{bmatrix} 0 \end{bmatrix} \tag{3.28}$$

- Z-axis 1st order model

$$G_{Z_1}(s) = \frac{0.03}{s} \tag{3.29}$$

$$A = \begin{bmatrix} 1 \end{bmatrix} \tag{3.30}$$

$$B = \begin{bmatrix} 0.0039 \end{bmatrix} \tag{3.31}$$

$$C = \begin{bmatrix} 0.0077 \end{bmatrix} \tag{3.32}$$

$$D = \begin{bmatrix} 0 \end{bmatrix} \tag{3.33}$$

The Bode plots for these first order models can be found in Figures 3.17, 3.18 and 3.19. As expected, the majority of the dynamic response information has been lost when the models are reduced to first order. However, the low frequency gain and phase are consistent with the experimental data. These less accurate models are referred to as '1st

order' in this document.

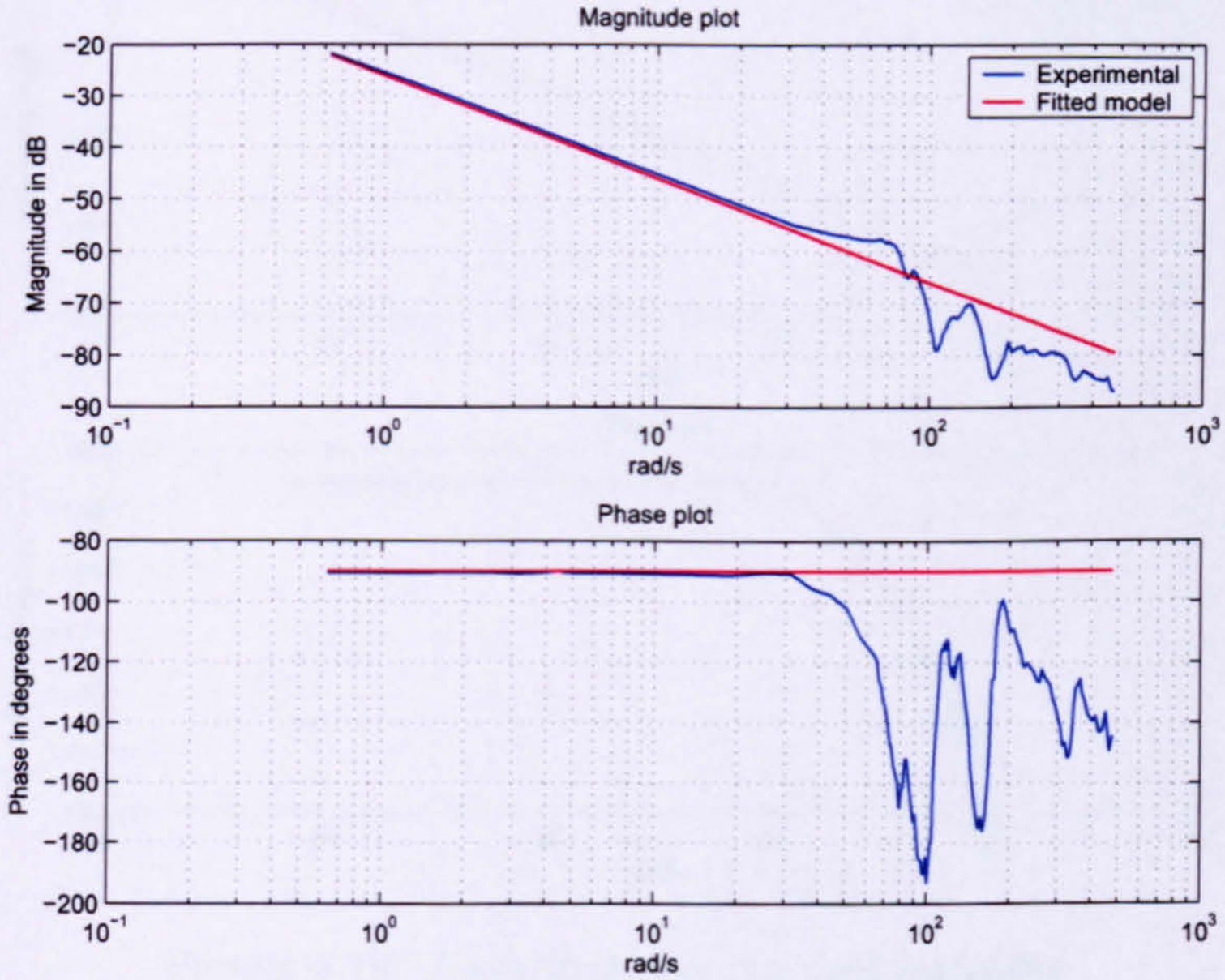


FIGURE 3.17: X-axis Bode plot, gain and integrator

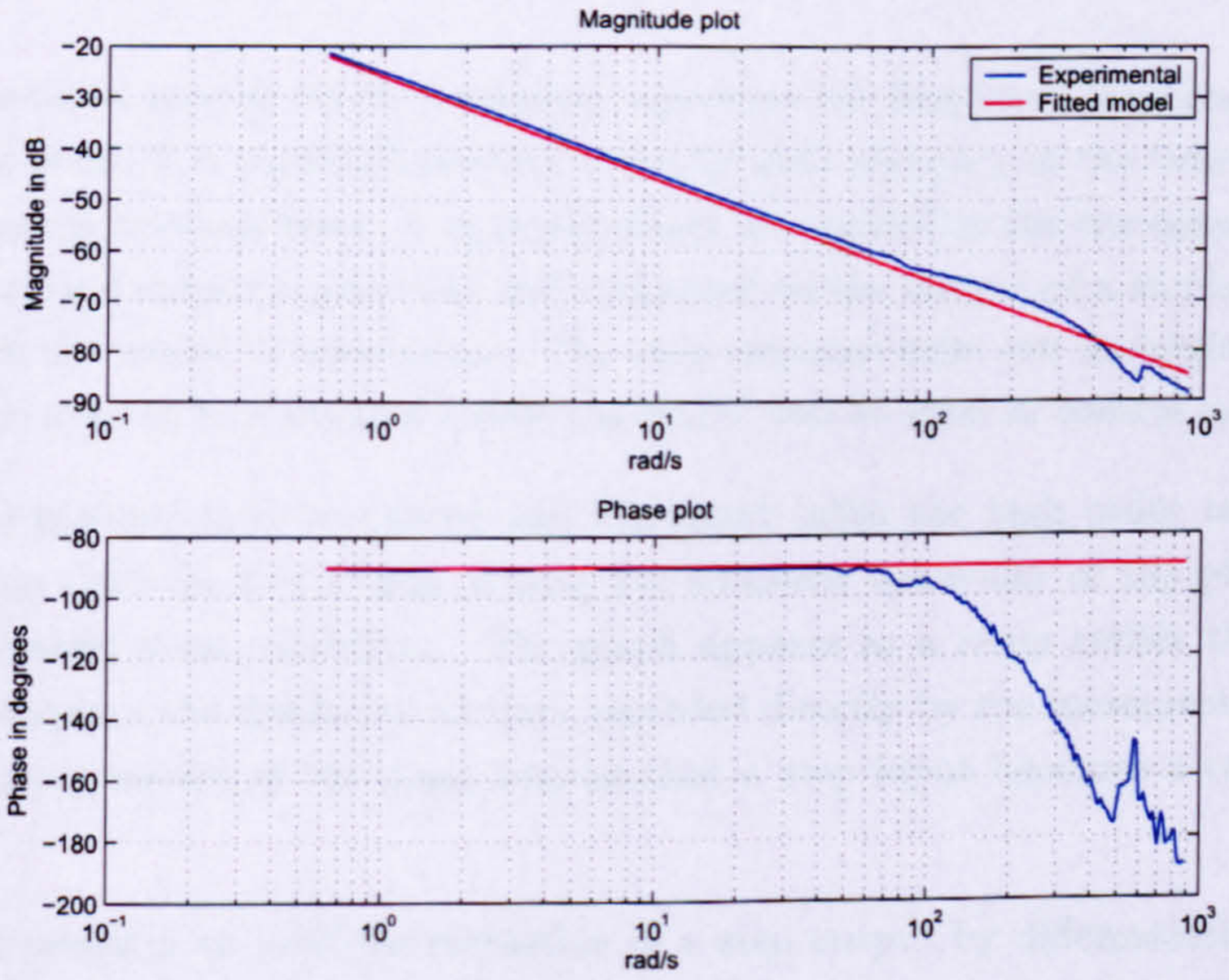


FIGURE 3.18: Y-axis Bode plot, gain and integrator

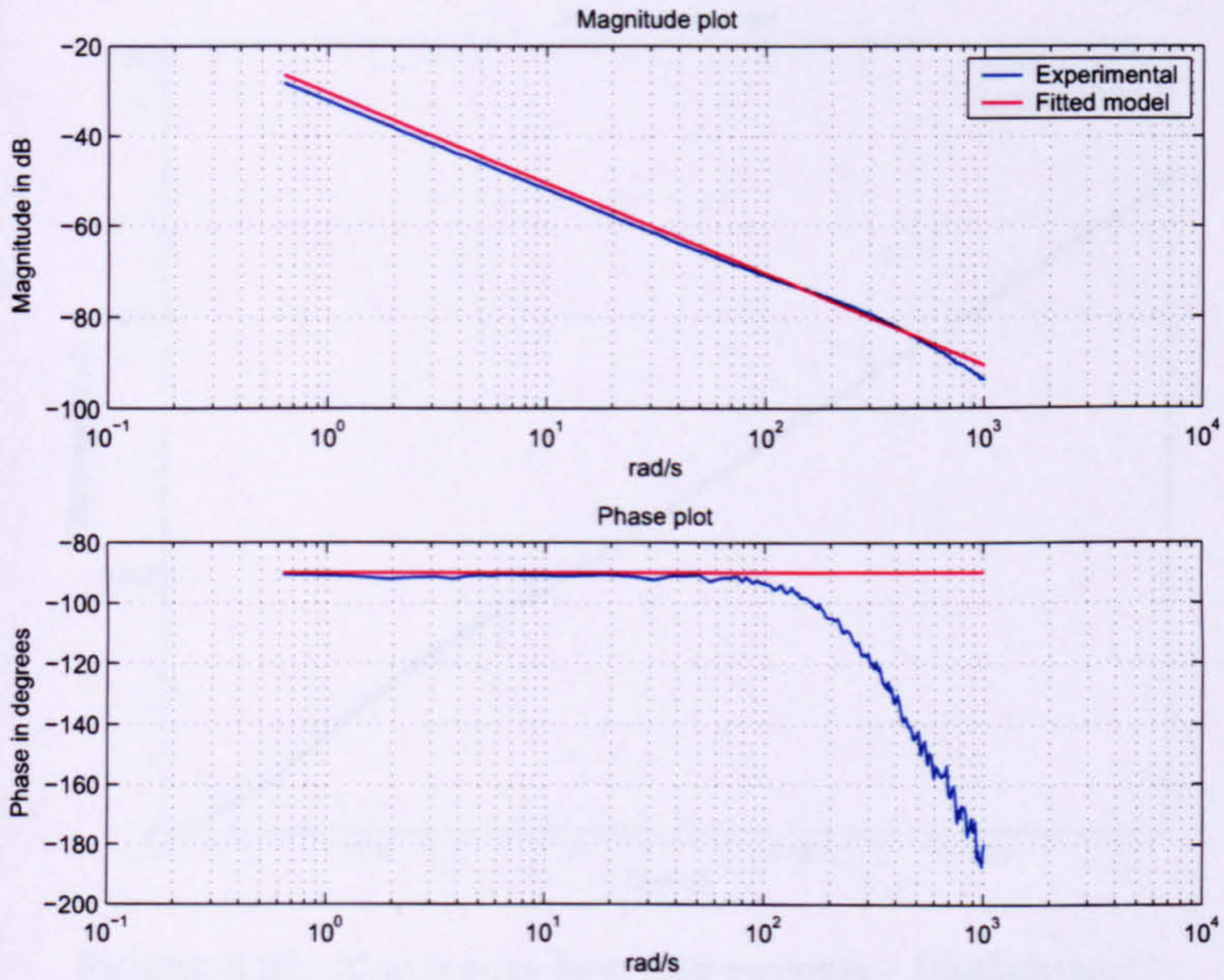


FIGURE 3.19: Z-axis Bode plot, gain and integrator

3.4 Model Verification

Having developed models which accurately represent the frequency response characteristics of the plant, it is standard practice to verify their accuracy in the time domain by performing step response tests. A unit step input is supplied to the the open loop plant and the resulting output is recorded and compared to the output of a similar test, performed with the model in simulations. The step response tests can highlight modelling errors which need to be corrected before the model can be used in control systems.

Figure 3.20 presents both measured and simulated (with the high order model), unit step response data for the X-axis, during the transient behaviour of the plant, before it reaches steady state conditions. The graph appears as a ramp rather than a step, because it displays the displacement data recorded directly by the incremental encoder. The integrating nature of the plant implies that a step input becomes a ramp at the output.

Figure 3.21 presents an artificial recreation of a step output by differentiating the displacement data presented in Figure 3.20 to obtain an approximate measure of axis velocity. The resolution of the measured data plot is poor due to the effects of measurement noise, sampling and quantisation. However, it is clear that the response of the model matches the measured data well. The amplitude and phase of the transient response oscillations are recreated accurately by the model, as is the steady-state magnitude. The simulated velocity plot emphasizes the high order dynamic response of the plant, because the transient oscillations are far from being pure sinusoids. They are a produced by a

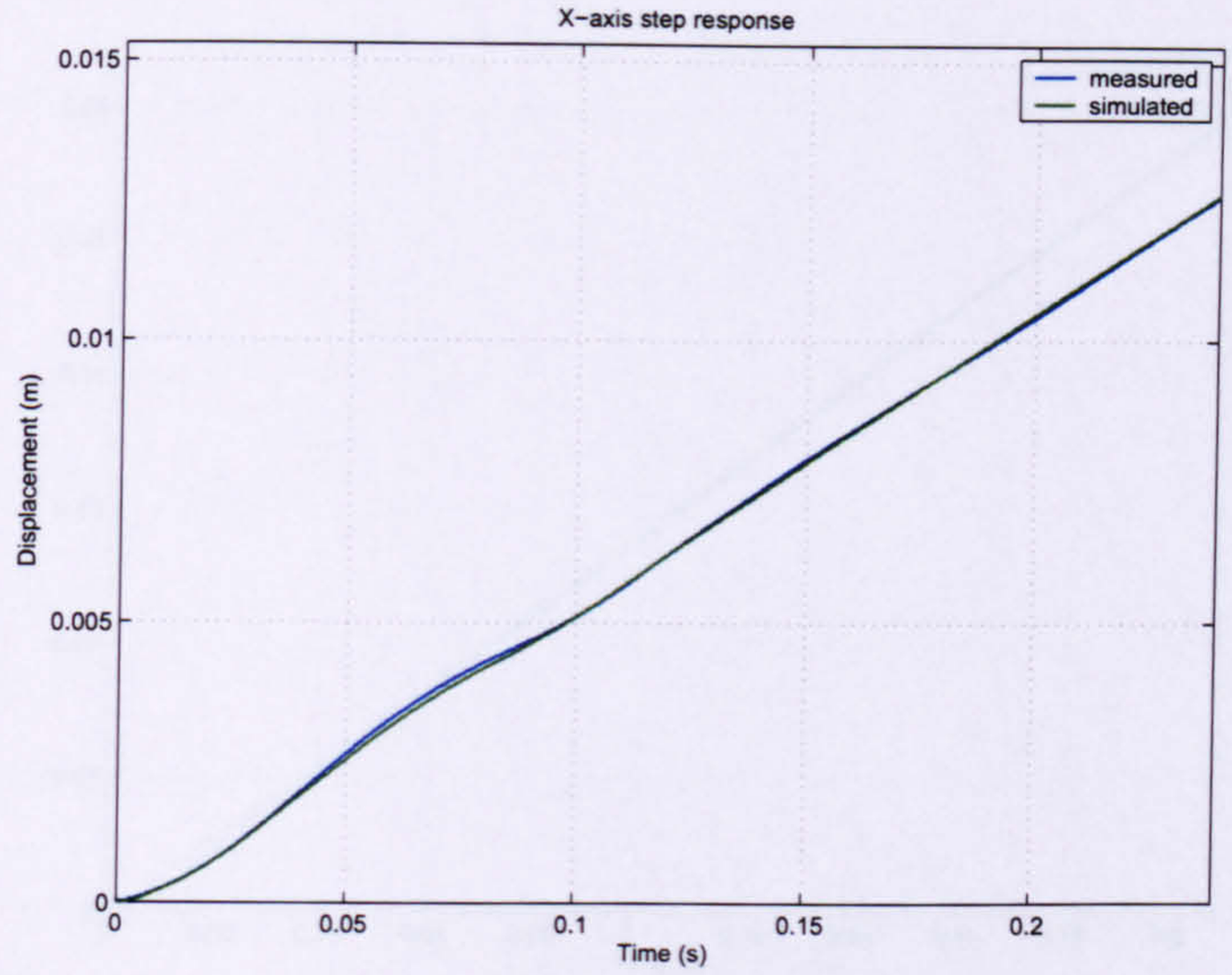


FIGURE 3.20: X-axis open loop step response - Displacement)

combination of numerous frequencies combined together.

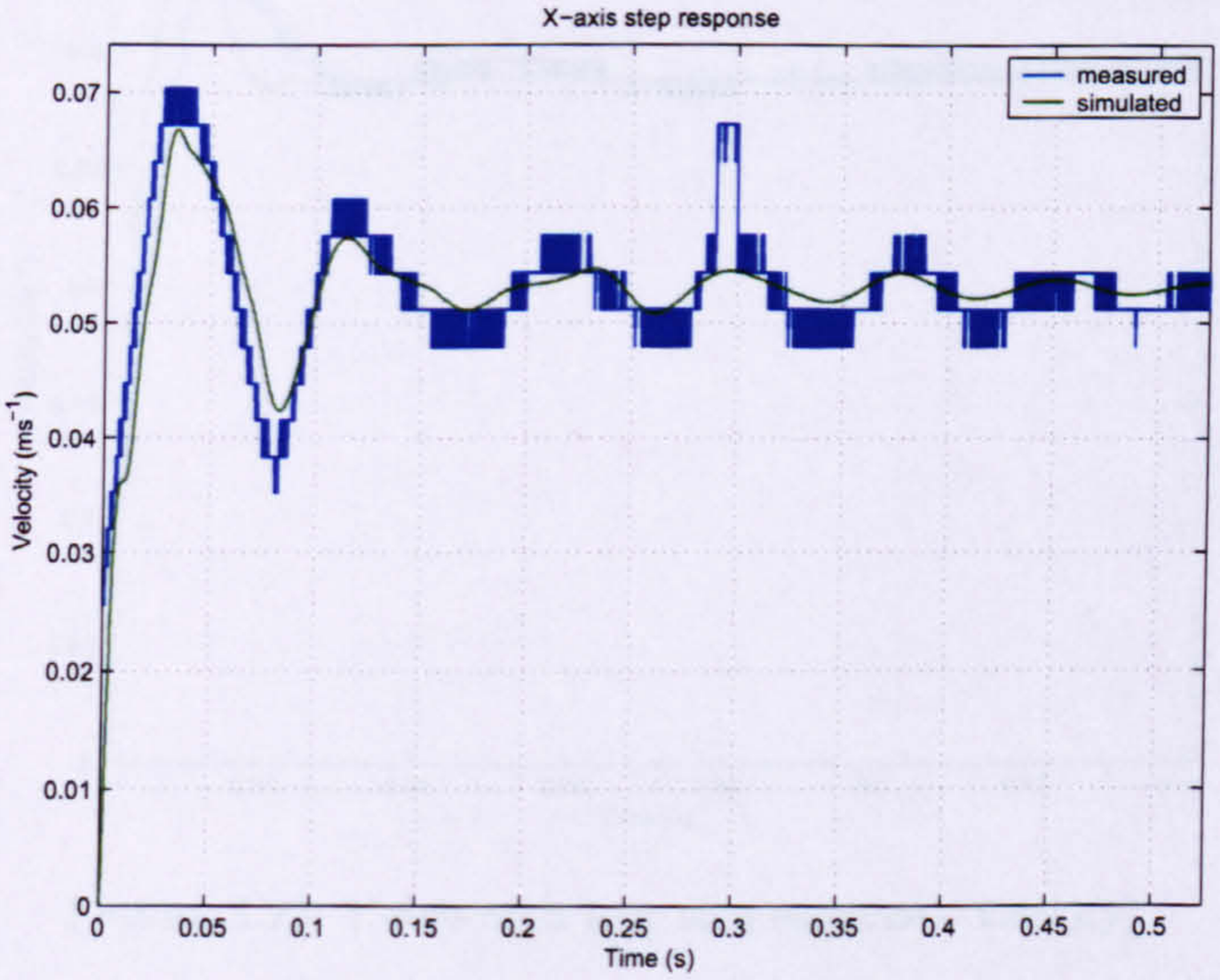


FIGURE 3.21: X-axis open loop step response - Velocity)

Figures 3.22 and 3.23 display the equivalent displacement and velocity data for the Y-axis. There is only a small component of error during the initial transient response and the steady state behaviour is accurate. The lower order of the Y-axis dynamic response, compared to the X-axis, is clearly visible, because the transient oscillations are significantly more sinusoidal. The Y-axis also has a much shorter settling time than the X-axis. This is related to the smaller component of mass which the Y-axis moves.

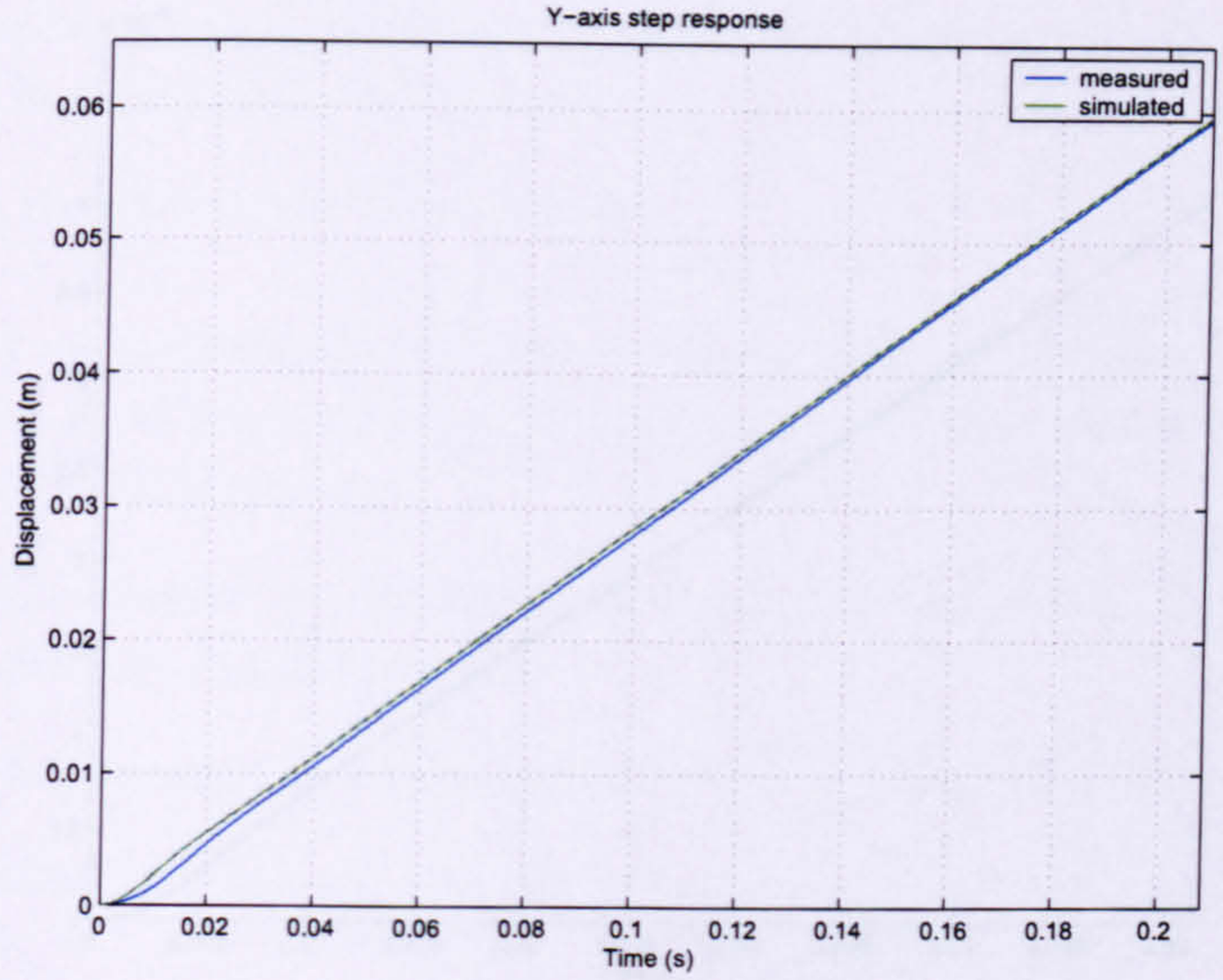


FIGURE 3.22: Y-axis open loop step response - Displacement)

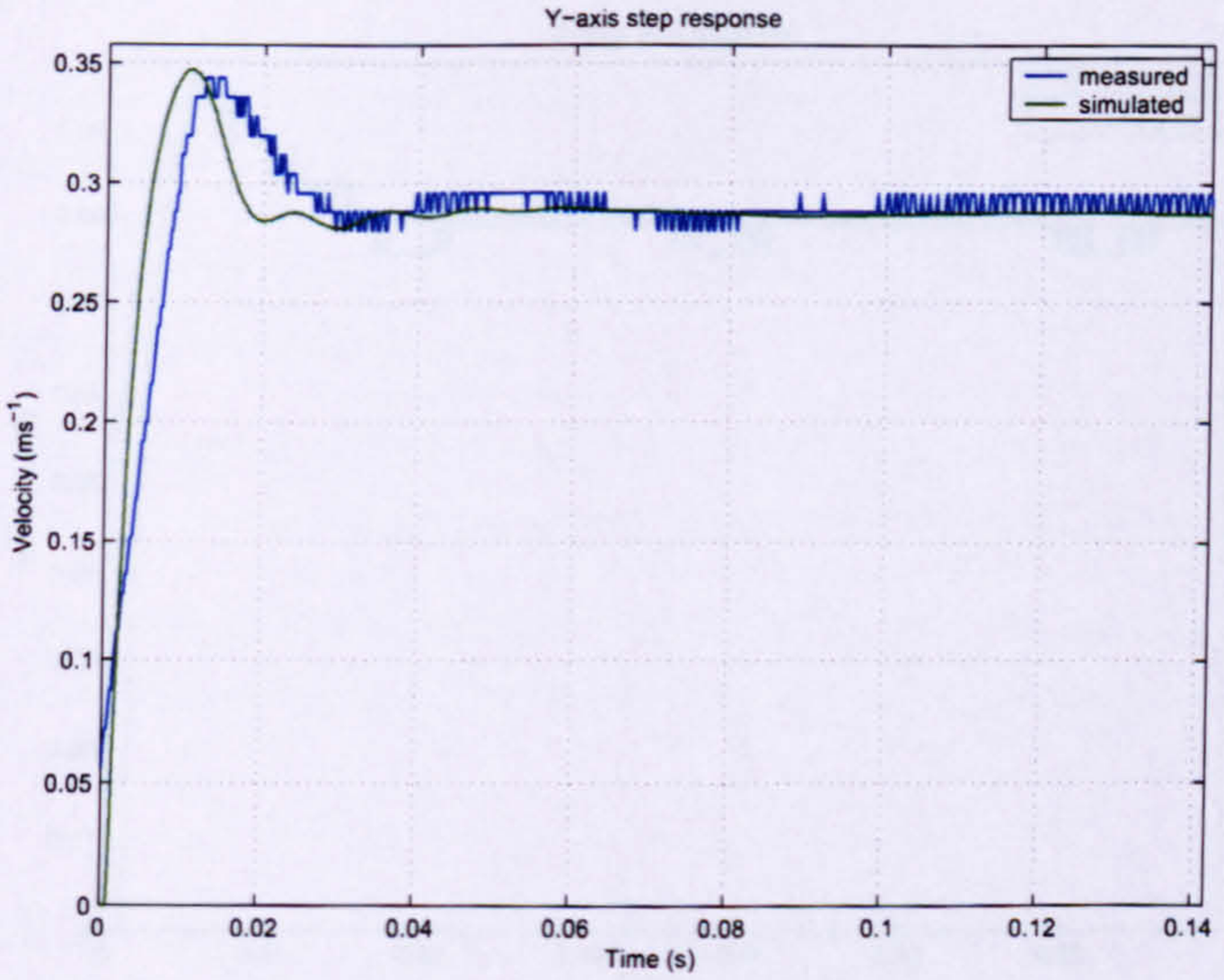


FIGURE 3.23: Y-axis open loop step response - Velocity)

Figures 3.24 and 3.25 present the data for the Z-axis step response. The measured and simulated responses are very similar in both displacement and velocity charts, confirming the accuracy of the Z-axis model. The step response is comparable to that of the Y-axis, except that there is only a single overshoot and no further oscillations.

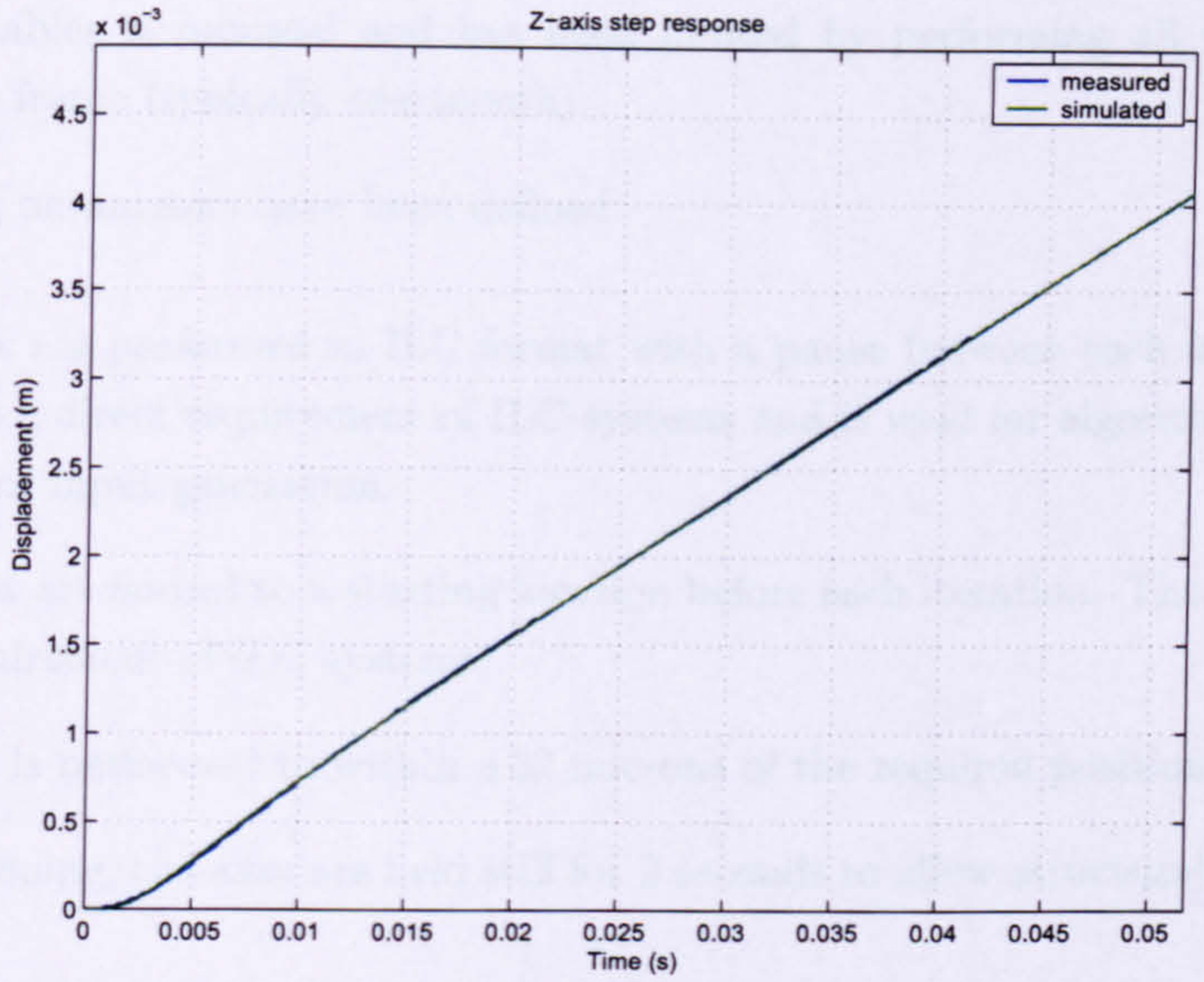


FIGURE 3.24: Z-axis open loop step response - Displacement)

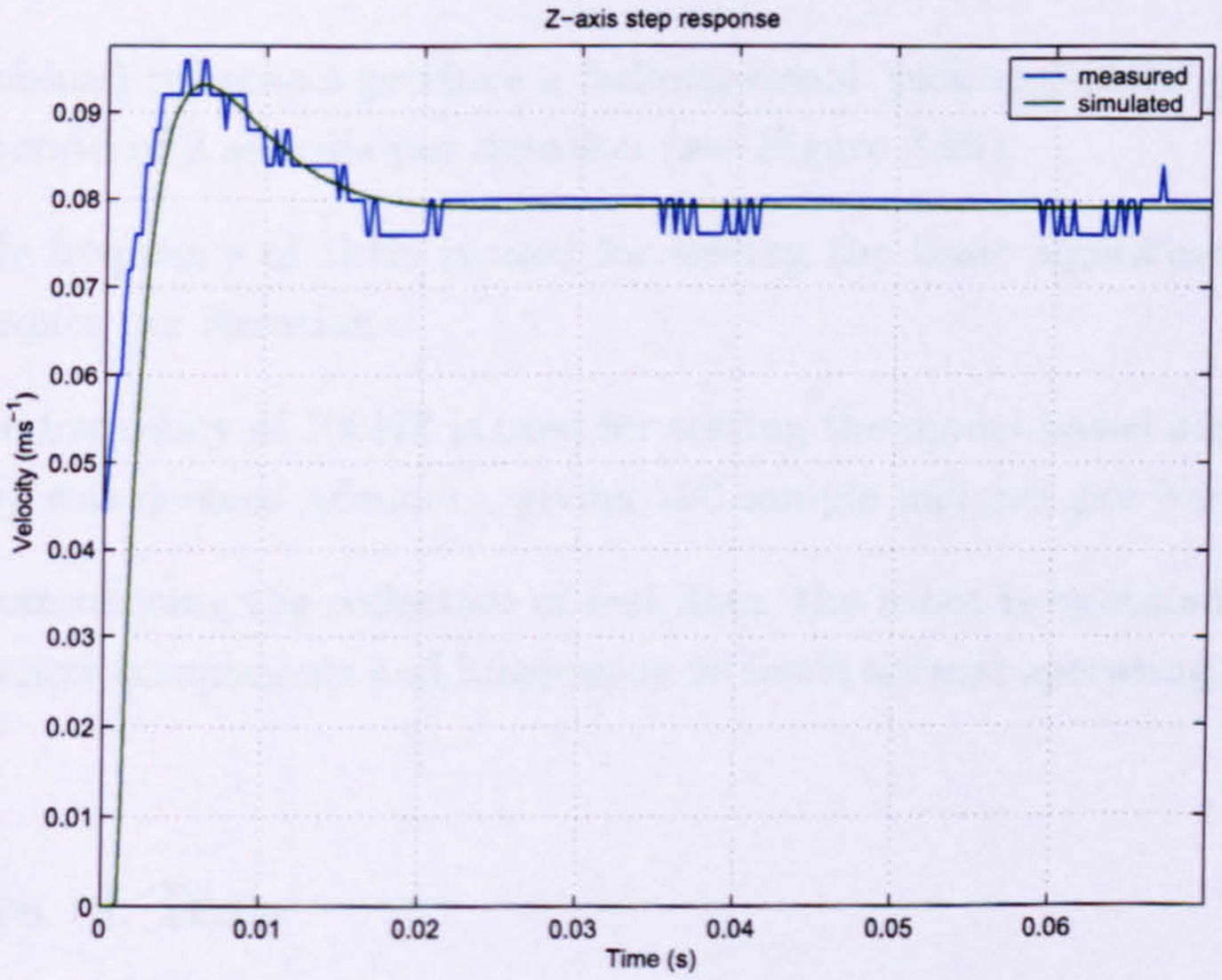


FIGURE 3.25: Z-axis open loop step response - Velocity)

3.5 Test Parameters

To compare the performance of different control algorithms, a rigid framework must be established to ensure that the comparison is meaningful. It is important to provide each algorithm with the same test conditions. A structured set of test procedures significantly reduces variation and increases test repeatability. But, it must also be recognised that certain environmental factors, for example, temperature, humidity, machine wear and machine lubrication cannot be controlled with any degree of accuracy. The effects caused

by these variables is minimal and has been limited by performing all tests within a minimal time frame (typically one month).

The following parameters have been defined:

- All tests are performed in ILC format with a pause between each iteration. This pause is a direct requirement of ILC systems and is used for algorithm calculation and plant input generation.
- The axes are homed to a starting location before each iteration. This is the second key requirement of ILC systems.
- Homing is performed to within ±32 microns of the required position.
- After homing, the axes are held still for 2 seconds to allow structural vibrations to cease.
- Each axis has a reference trajectory which is used for all tests (see Figures 3.26, 3.27 and 3.28).
- The combined references produce a 3-dimensional 'pick-and-place' operation with a time period of 2 seconds per iteration (see Figure 3.29).
- A sample frequency of 1kHz is used for testing the basic algorithms, resulting in 2000 samples per iteration.
- A sample frequency of 100HZ is used for testing the model-based algorithms. This frequency was deemed adequate, giving 200 sample instants per iteration.
- Before commencing the collection of test data, the robot is operated for 300 iterations to allow components and lubrication to reach normal operating temperatures.

3.6 Types of Test

As well as establishing general rules for operating the plant, a series of tests have been developed to investigate four key aspects of ILC performance, long-term stability, robustness to initial state error, robustness to model gain uncertainty and robustness to model gain and phase uncertainty. These tests have been implemented in the same manner for each algorithm.

3.6.1 Long-term stability

Testing long-term stability involves operating the algorithm continuously for a large number of iterations. Experience gained through the implementation of different algorithms

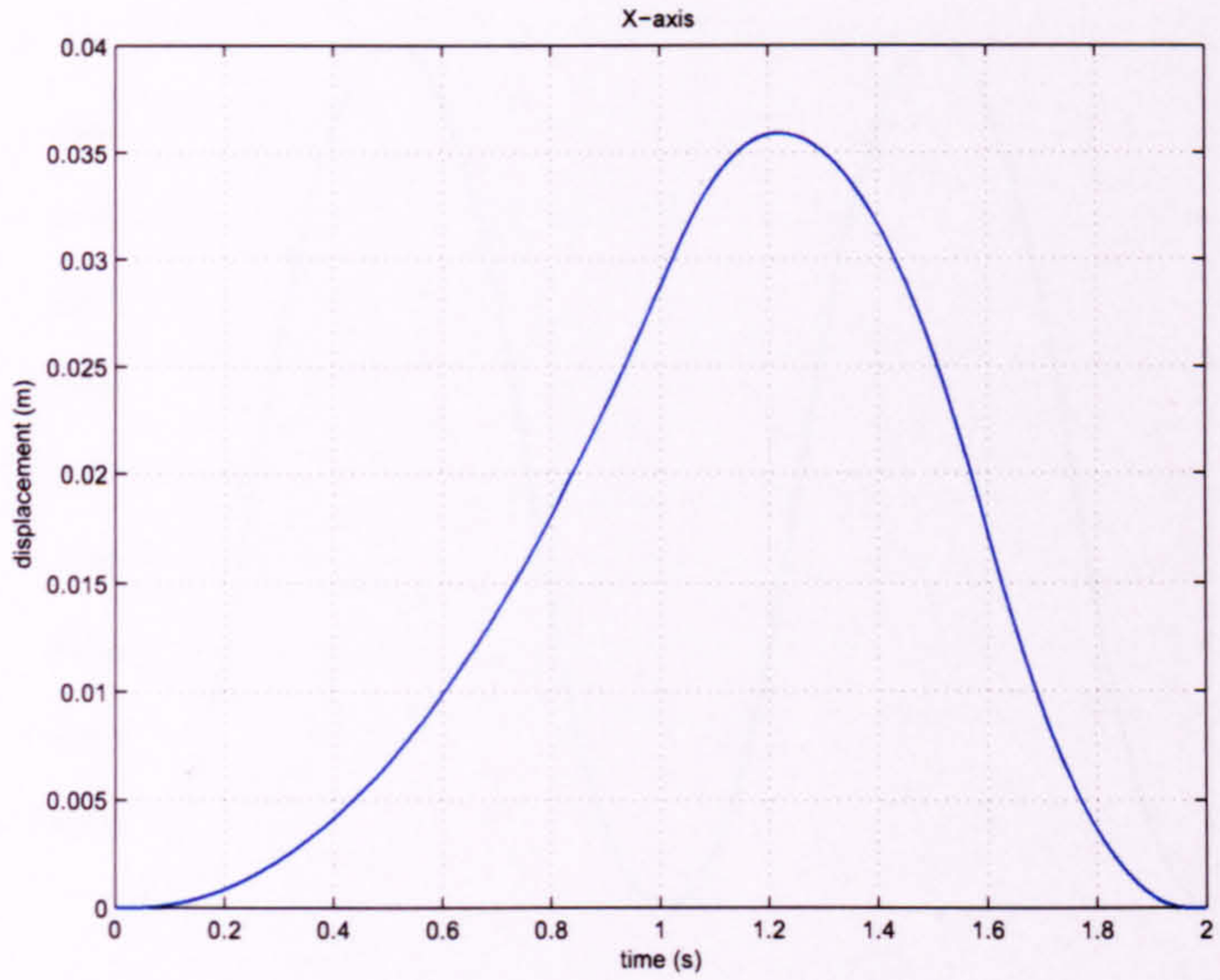


FIGURE 3.26: X-axis reference trajectory (30upm)

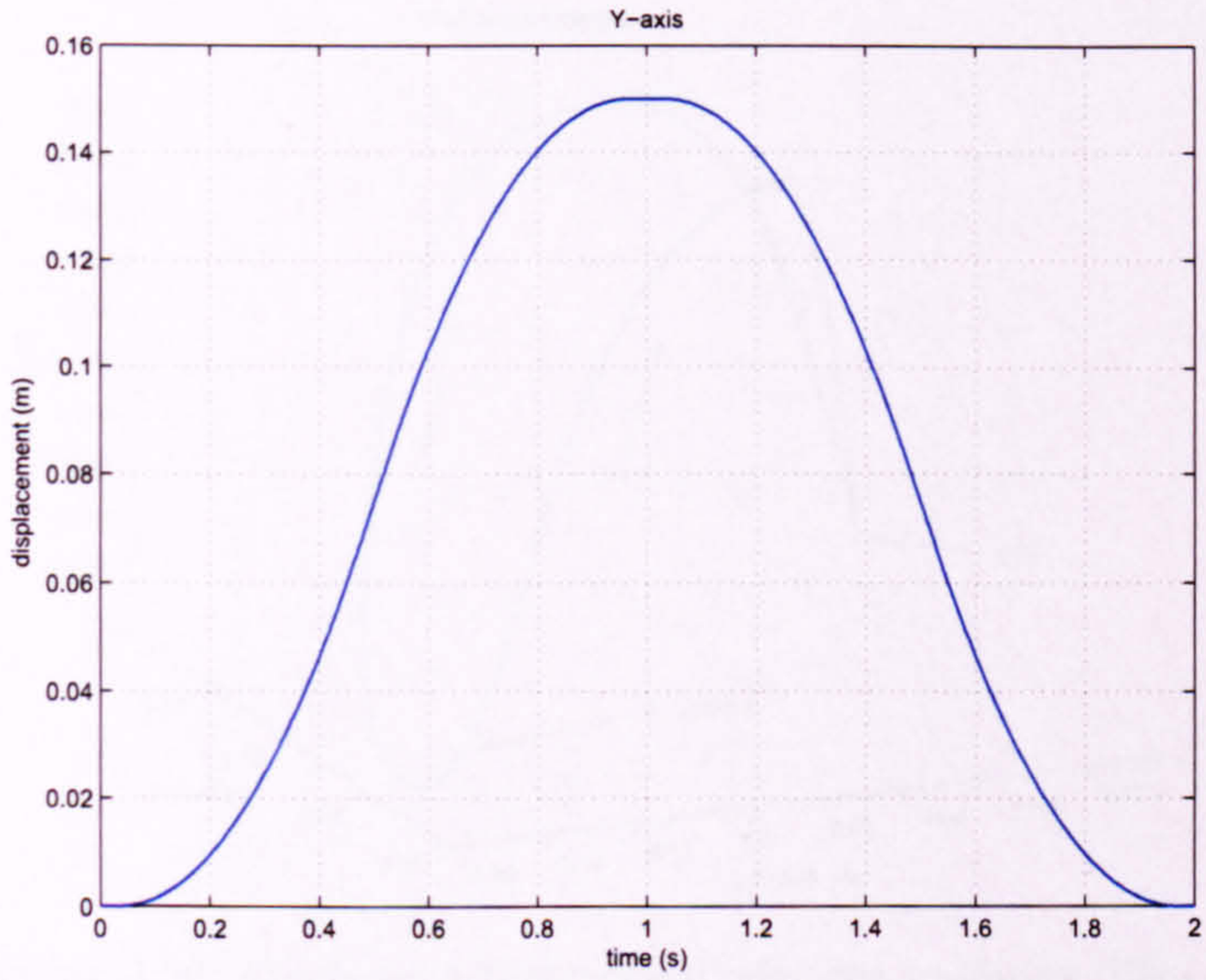


FIGURE 3.27: Y-axis reference trajectory (30upm)

on the gantry robot tends to suggest that if an algorithm is inherently unstable, the instability will be identifiable within 100 to 200 iterations, sometimes as few as 3 or 4 iterations. The long-term stability test is therefore defined as a batch of 5000 consecutive iterations. This test does not guarantee algorithm stability for an infinite number of iterations. However, in comparison to the few hundred iterations which can be achieved by inherently unstable algorithms, the 5000 iteration test is a good indicator of long term-stability. The high order models are used for this test.

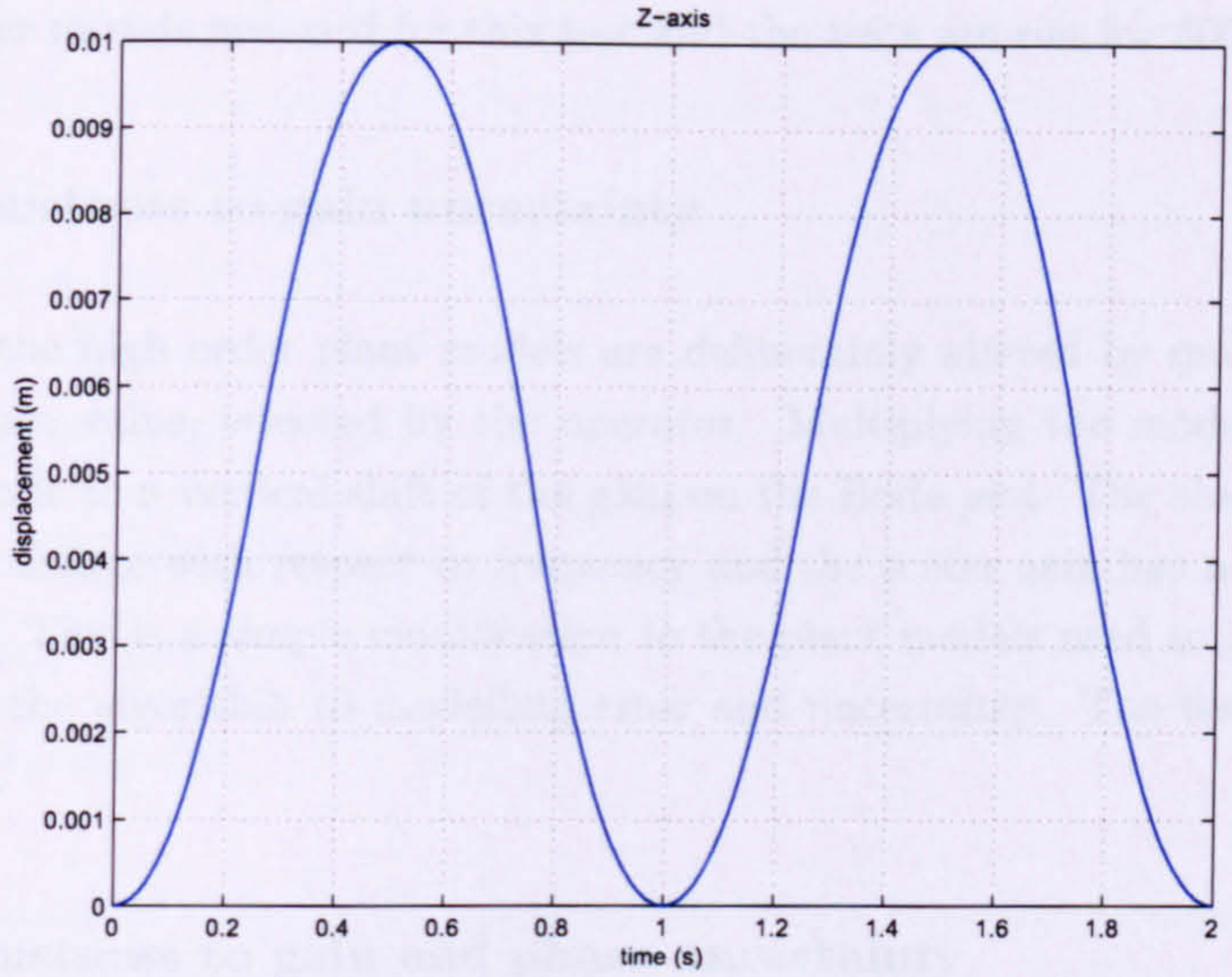


FIGURE 3.28: Z-axis reference trajectory (30upm)

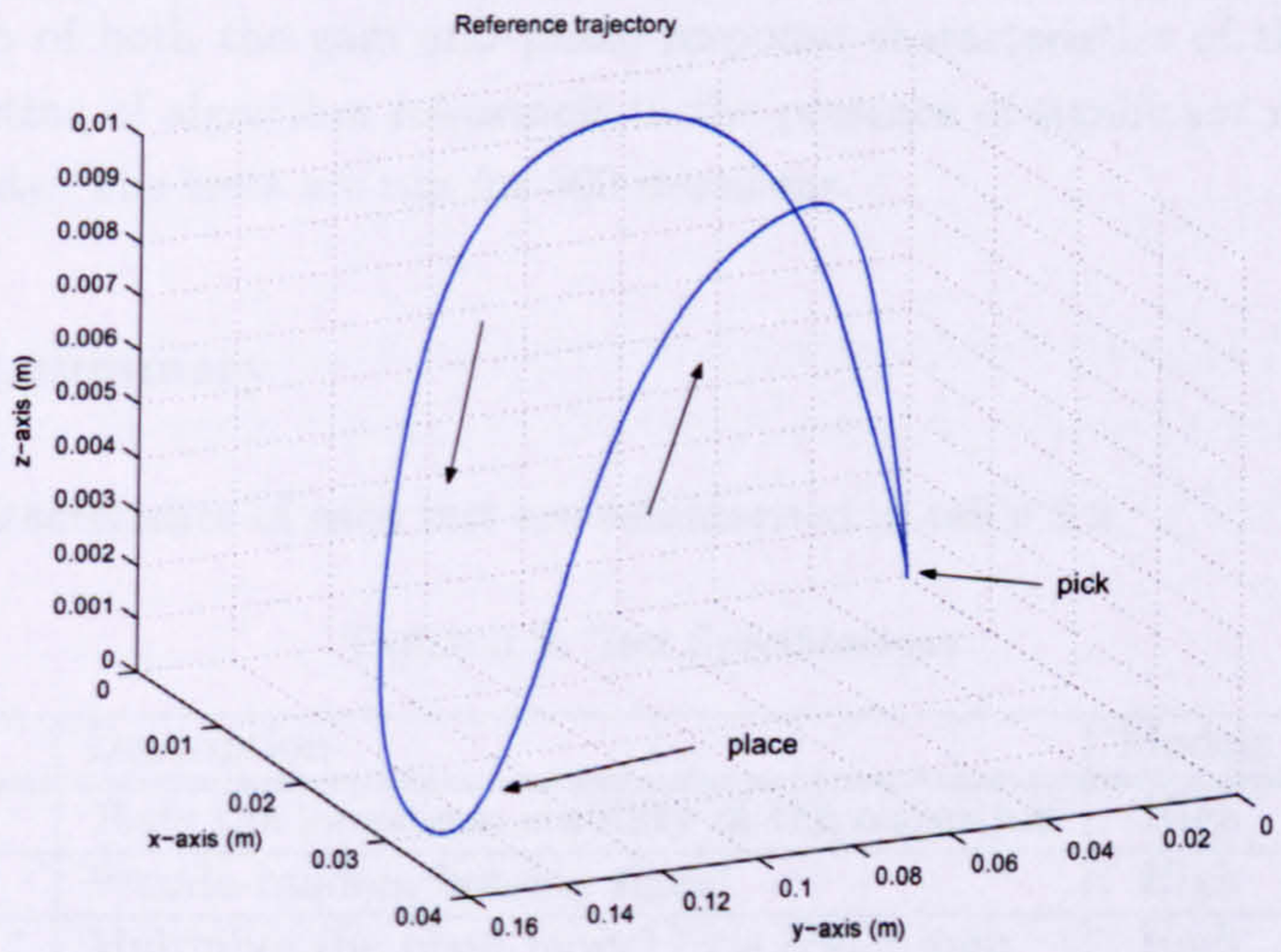


FIGURE 3.29: Combined 3-Dimensional reference trajectory (30upm)

3.6.2 Robustness to initial-state error

To measure controller robustness, initial state error is deliberately introduced at the start of each iteration by adding a bounded, pseudo-random value to the home location of each axis. This specifically adds error to the initial position of the robot. The value is pseudo-random because it is generated by a seeded random number generator, where the seed is the iteration number. Therefore, within one test, the initial displacement for each iteration appears to be random. However, within a range of tests, the same offset is generated for corresponding iterations. The initial offset is also scalable. This is achieved by selecting the maximum bound within which the pseudo-random number can occur.

The high order models are used for this test and the tests are run for 500 iterations.

3.6.3 Robustness to gain uncertainty

To test this, the high order plant models are deliberately altered by multiplying them by a scalar gain value, selected by the operator. Multiplying the models by a scalar gain corresponds to a vertical shift of the gain on the Bode plot. The shape of the gain plot does not change with respect to frequency and the scalar gain has no effect on the system phase. This is a simple modification to the plant models used to investigate the robustness of the algorithm to modelling error and uncertainty. The tests are run for 500 iterations.

3.6.4 Robustness to gain and phase uncertainty

This has been evaluated using first order plant models that provide a very approximate representation of both the gain and phase response characteristics of the plant. This allows the testing of algorithm robustness in the presence of significant modelling error and uncertainty. The tests are run for 500 iterations.

3.6.5 Test summary

The main characteristics of each test are summarised in table 3.2.

TABLE 3.2: Test Specifications

Test	Description	Models	Iterations
Long-term	Tests the long-term stability of the algorithm	High	5000
Initial error	Pseudo-random homing signal	High	500
Robustness 1	Multiplies the plant model by a scalar gain	High	500
Robustness 2	Uses first order plant models	Low	500

3.7 Measuring ILC Performance

There is currently very little published research which discusses techniques for measuring the relative and stand alone performance of ILC systems. Standard techniques for algorithm comparison used to date, generally involve plotting a measure of the tracking error for each trial and then subjectively evaluating which plot is best. In ILC systems, numerous factors can be used to describe the performance of an algorithm. Therefore this approach depends on which factors are considered more important than others. With

this in mind, a numerical performance index has been developed, which addresses and evaluates four of the main performance indicators of ILC systems.

Four variables which are of particular importance, when describing the performance of an ILC algorithm are (Xu and Tan, 2002):

- Convergence speed
- Minimum tracking error
- Transient performance
- Long-term stability

Figure 3.30 shows the typical mean squared error (mse) plot for an unstable ILC system. The mse is a representation of the tracking performance obtained during each iteration and indicates whether the tracking performance is generally improving, remaining constant or becoming worse. For a vector $\mathbf{e}$ of length N, the mse is defined mathematically by:

$$\text{mse}(\mathbf{e}) = \frac{\sum_{k=1}^{N} e_k^2}{N} \tag{3.34}$$

The mse is used in this document to represent the general level of tracking error achieved at each iteration. However, there is an alternative measure, which is also popular within the ILC community, namely the norm of the error. This usually refers to the ℓ_2 norm, defined by:

$$\|\mathbf{e}\| = \sqrt{\sum_{k=1}^{N} |e_k|^2} \tag{3.35}$$

The ℓ_2 norm is the square-root of the sum of squares, effectively a representation of the length or size of the vector to which it is applied.

Key parameters used to describe ILC performance are also indicated in Figure 3.30: e_1 is the initial mse value, i_{me} is the number of iterations required to reach minimum error, e_{min} is the minimum mse value and i_u is the number of iterations until the mse begins to increase and effectively the system becomes unstable. In the cases where the controller is unstable, often the number of iterations to minimum error i_{me} and the number of iterations to instability i_u will be the same, defining a single point of minimum error e_{min}. However, these points have been defined separately, because for some controllers, the mse does appear to remain constant at the minimum value e_{min} for a number of iterations, before the instability becomes apparent and the mse noticeably increases. The typical mse plot for a stable system which displays monotonic convergence is similar, except that the i_u point is never reached and the mse does not increase as the number of iterations

increases. Of these parameters, i_{me} and e_{min} are most commonly used to describe ILC performance because these are the most easily measured variables. The general shape of Figure 3.30 applies equally well to the typical shape of the error norm curve.

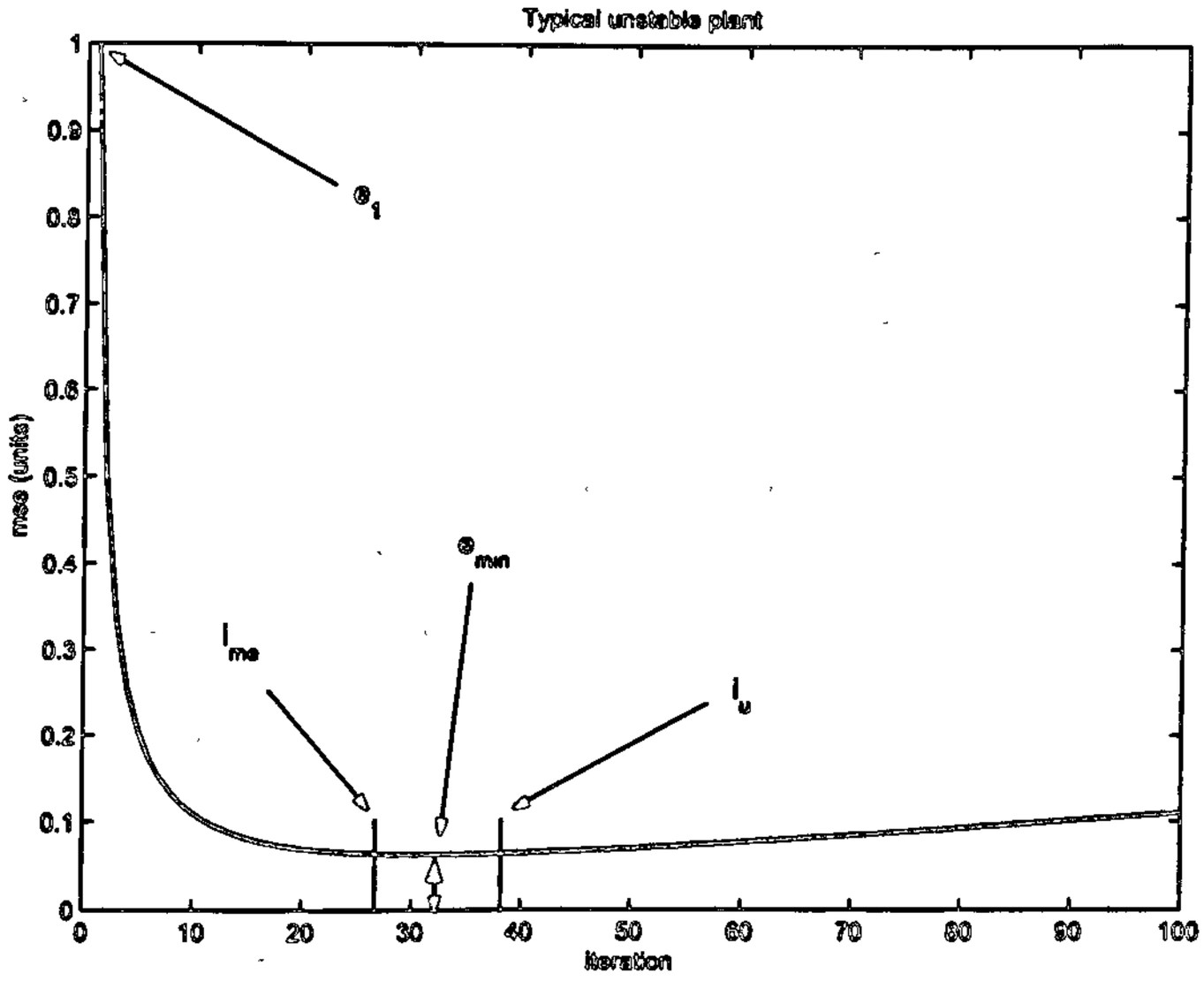


FIGURE 3.30: Typical mse curve for an unstable ILC system

When control engineers describe the performance of ILC systems, they tend to discuss qualitatively the shape and characteristics of the mse or norm chart. When directly comparing two curves plotted on the same chart, it may be a reasonable method of comparison, because it is usually clear which algorithm achieves the lowest tracking error or fastest convergence speed. However, the comparison is very subjective and the conclusions depend ultimately on which characteristics are recognized as significant by the engineer. If the two performance curves cannot be plotted on the same chart, the comparison becomes less obvious. It is therefore necessary to extract the relevant information from the performance curve in a reliable and repeatable manner. For consistency, it is assumed that the mse curve is used rather than the norm curve.

The proposed performance index involves calculating the area under the mse curve for the first N iterations, where N can be selected appropriately for the systems being analysed. This results in the Performance Index for N iterations, PI_N (Ratcliffe, Lewin, Rogers, Hätönen, Harte, and Owens, 2005a). It is suggested that most ILC systems exhibit learning behaviour during the first 100 iterations. Within this time, the majority of the learning has been achieved and the system has reached, or is near, the minimum tracking error value. Therefore it is appropriate to compare the performance of algorithms during this period. The practical implementation results in Chapters 4 and 5 have been compared using PI_{100}.

PI_N is simple to calculate. Consider the area beneath the mse plot as divided into

rectangular columns. If the width of each iteration column is taken as unity, this makes the PI_{100} a simple summation of the first 100 mse values. For the general case PI_N, this is formally defined as:

$$PI_N = \sum_{n=1}^{N} \text{mse}_n \tag{3.36}$$

where n is the iteration number.

The value of PI_N represents the performance of the algorithm with respect to both convergence rate and minimum error level. Consider Figure 3.31 which shows typical mse plots for systems with different performance characteristics:

1. Fast convergence and low final error.
2. Fast convergence but large final error.
3. Slow convergence but low final error.
4. Slow convergence and large final error.

The most desirable characteristics are fast convergence and low final error. These generate the smallest area beneath the mse curve and hence the smallest value of PI_{100}. Any other combination will produce a larger PI_{100}, unless one of the characteristics has a particularly outstanding performance, such as large final error but with particularly fast convergence, or slow convergence but with extremely small error. In either case a small PI_{100} can be generated. Which algorithm performs better is then dependent upon which characteristic is more important for a particular application.

To allow a fair comparison of algorithm performance, several test parameters must be held constant:

- The plant (or plant model in simulation).
- The reference trajectory.
- The value of N.
- The mse value for the first iteration (e_1).

Variation in any of these parameters will affect PI_N in a way which does not correspond directly to a change in performance.

Parameter e_1 is perhaps the most difficult of these values to hold constant. However, this can be achieved if the plant input is set to zero for the duration of the first iteration. The plant output should therefore remain constant, and the value of e_1 will be the mse

equivalent to the reference trajectory. With e_1 held constant, it is logical to remove the PI_N dependency on the unit of mse, by normalising the mse so that $e_1 = 1$.

It is now possible to define upper and lower bounds on the value of PI_N by considering two extreme cases of tracking performance. Firstly, if the algorithm achieves perfect tracking after only one iteration of learning, as specified previously, the mse for the first iteration is normalised to 1, but by the second iteration the mse will be 0. Irrespective of the value of N, the mse will remain equal to 0 for all $n > 1$. Therefore the sum of the mse will result in $PI_N(\text{min}) = 1$ which defines the lower bound. In the opposite case, when the algorithm learns nothing at any iteration, the mse will be equal to 1 for each iteration, therefore the upper bound can be defined as $PI_N(\text{max}) = N$. If the algorithm becomes unstable and the calculated PI_N is larger than N, then it is set to N by default. Consequently, the PI_N value will therefore lie between the bounds $1 \leq PI_N \leq N$. The closer the value of PI_N is to 1, the better the tracking performance.

3.8 Summary

The multi-axis test facility including all hardware and software has been described in detail. The frequency response method used to generate transfer function models of each axis has then been discussed and the matching Bode plots, transfer functions and state-space models have been presented. As the highest order models are both uncontrollable and unobservable, the order of the models has been reduced until all states are controllable and observable. The time delays have also been removed. Finally, first order models used for robustness testing have been defined. The high order models have been verified by comparing experimental and simulated step response tests.

A structured framework has been established to allow the fair comparison of different ILC algorithms. Test parameters have been stated and the four tests used to investigate various aspects of algorithm behaviour have been defined. Finally, a new performance index designed specifically for ILC systems has been developed. The index takes four of the most significant performance indicators into consideration and produces a repeatable numerical performance rating which ranges from 1 when perfect tracking is achieved in just one iteration, to N when the algorithm learns nothing or becomes unstable within the first N iterations of a test.

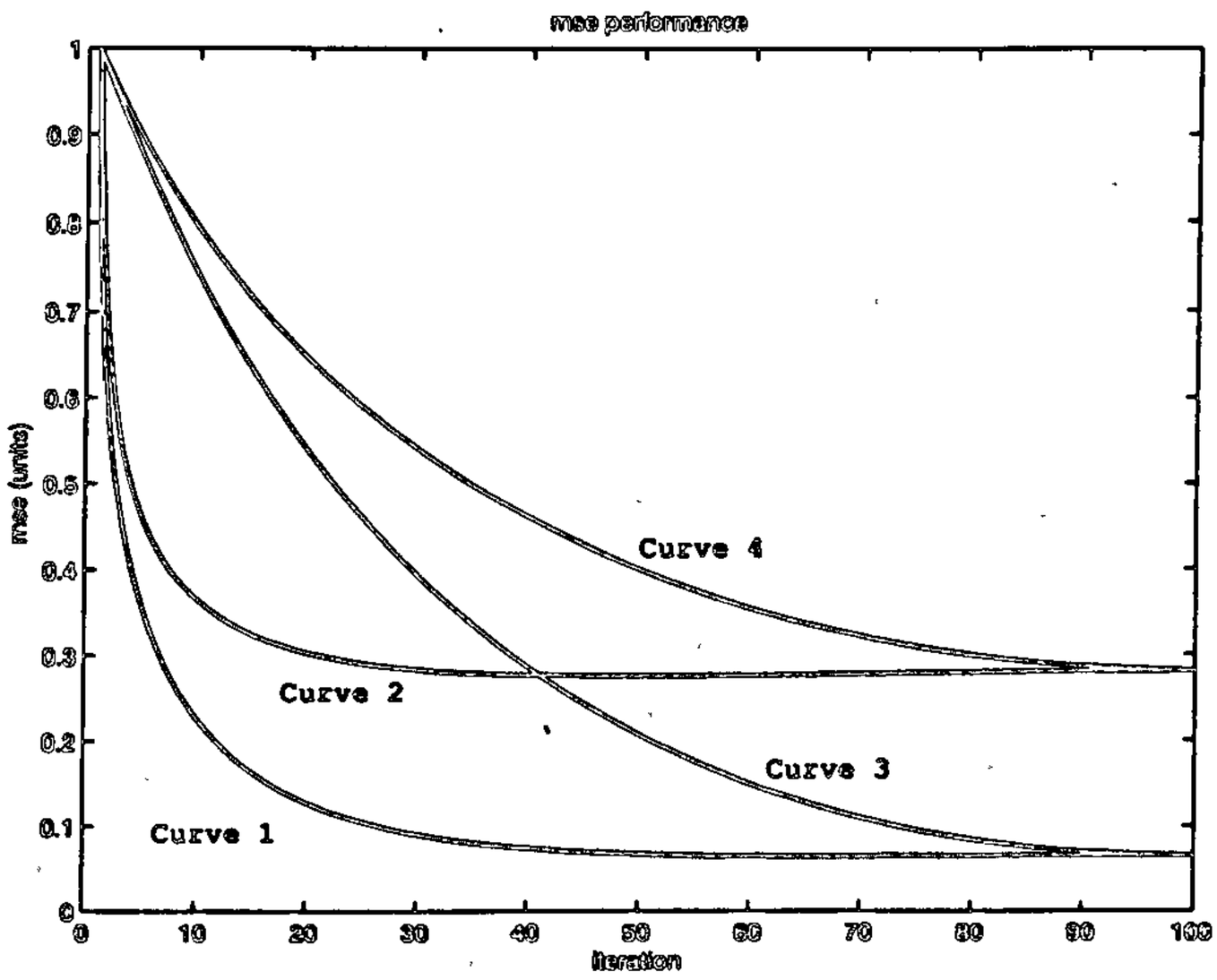


FIGURE 3.31: Different mse curves showing varied performance characteristics. 1. Fast convergence - low error, 2. Fast convergence - large error, 3. Slow convergence - low error, 4. Slow convergence - large error

Chapter 4

Implementation of Basic Controllers

4.1 Introduction

The three term or PID feedback controller, developed over 60 years ago, is still widely used in industry in a very diverse range of applications. Simplicity of design and ease of tuning are two of the qualities which make it particularly attractive to industry. PID control has been implemented on the robot and acts as a benchmark against which the ILC controllers can be compared. Basic P-type ILC, implemented alone has been found to be particularly badly suited to a naturally integrating plant. However, the hybrid combination of PID feedback with assistance from P-type ILC produces a significant increase in tracking performance. Long-term stability issues have been considered and a range of filtering techniques implemented to resolve problems of high frequency noise and resonant disturbance.

Generally, data specific to the X-axis is presented within this chapter, while the data for the Y and Z-axes are included for completeness in Appendix B. In certain cases, the data for the Y and Z-axes are more appropriate and are therefore included. Where possible, the data for all three axes is presented simultaneously using a 3-Dimensional chart. This emphasises the fact that a multi-axis plant is being utilized for experimental evaluation of ILC strategies.

4.2 PID Feedback Control - Benchmark

4.2.1 Algorithm

PID feedback control consists of three terms acting on the current sample error. In general, the proportional component provides the majority or the control effort, while the integral component seeks to remove steady state error and the derivative component

reduces the rise time under transient conditions. From a frequency response perspective, the PID controller can be considered as a simplified lead-lag compensator, which can be implemented to adjust the gain and phase margins of the open loop plant and improve the bandwidth of the control system. The controller can be represented as:

$$u(t) = K_p e(t) + K_i \int e(t)\,\mathrm{d}t + K_d \frac{\mathrm{d}e(t)}{\mathrm{d}t} \tag{4.1}$$

where K_p, K_i and K_d are the proportional, integral and derivative gains respectively. Gain tuning is frequently achieved by means of the Ziegler-Nichols method (Ziegler and Nichols, 1942). This technique relies on causing the plant to oscillate by increasing the proportional gain. Once the frequency of oscillation and the gain required are established, it is possible to calculate the necessary values of integral and derivative control which will act as a starting point for tuning. However, this technique was not suited to the gantry robot because excessively large values of proportional gain could be supplied to the plant before any form of oscillation could be detected, resulting in meaningless values of integral and derivative gain. Instead, the gains were adjusted through experimentation and simulation studies. The resulting PID gains for each axis are as shown in Table 4.1.

TABLE 4.1: PID gains for each axis

Axis	P gain	I gain	D gain
X	600	300	0.2
Y	800	300	0.2
Z	1100	300	0.2

The proportional gains are the maximum values which could be implemented, while generating only minimal vibrations. The integral and derivative gains are identical for each axis, because these values were found to produce a good response from the system.

4.2.2 Test results

PID control is a feedback driven system, which inherently requires error between the plant output and the reference, for a new control input to be generated. Because of this requirement, PID control can never achieve perfect tracking with zero error. Moreover, a PID controller has no ability to learn from previous experiments and therefore, in a control system which is required to follow a repeating trajectory, while neglecting environmental variations, the controller will produce the same level of tracking accuracy at each repetition of the task. Implementation of a PID controller on the gantry robot yields the results in Figure 4.1.

Figure 4.1 shows a plot of the mse generated for each axis during a 1000 iteration test.

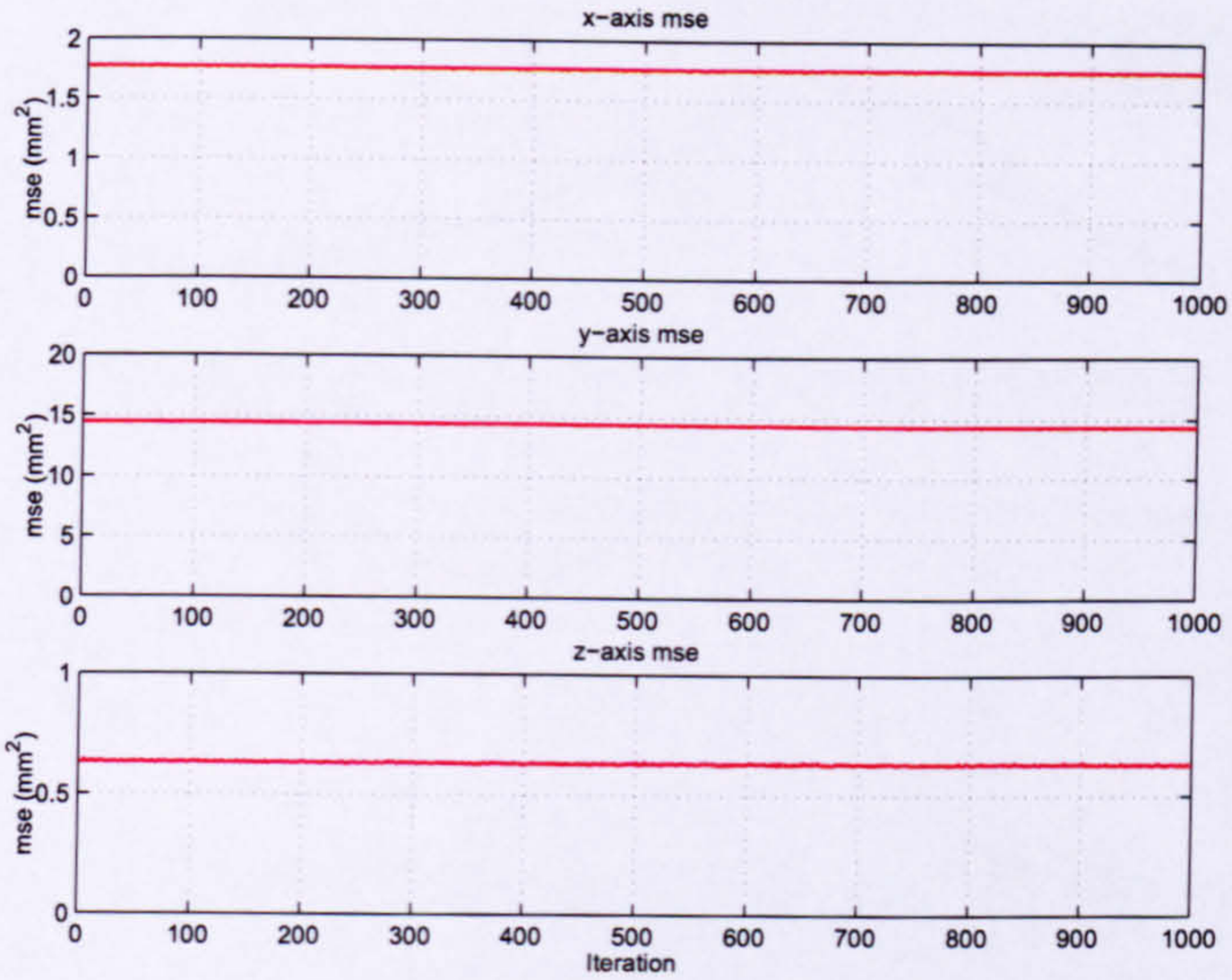


FIGURE 4.1: mse for 1000 iterations (PID controller)

Clearly the performance of the controller at iteration 1 is the same as the performance at iteration 1000. Table 4.2 displays the mean values of mse obtained over the 1000 iteration test. Calculating the mean removes the effect of tiny variations in performance which occur from iteration to iteration due to variations in non-linear factors such as friction. These are the benchmark values used for comparison with other controllers. The values for each axis differ significantly, because they correspond to the amplitude of the reference trajectories. The Y-axis reference describes a larger displacement than for the X and Z-axes, therefore the cumulative tracking error for the Y-axis is potentially larger.

TABLE 4.2: PID controller, mean mse for 1000 iterations

Axis	mse (mm^2)
X	1.76
Y	14.37
Z	0.63

In Figure 4.2, the reference trajectories, associated X-axis displacements and tracking errors achieved during iteration 1000 are shown. It is immediately obvious that the PID controller produces a significant level of residual tracking error, the majority of which can be accounted for by a time lag in the output response, which appears as a phase shift on the chart (similar results for the Y and Z-axes can be found in Appendix B).

Figure 4.3 displays the same information but in 3-Dimensional format with the data for all axes on the same chart. The resolution of the chart does not emphasise the magnitude of the residual tracking error. Figure 4.4 displays the measured tracking error, also in

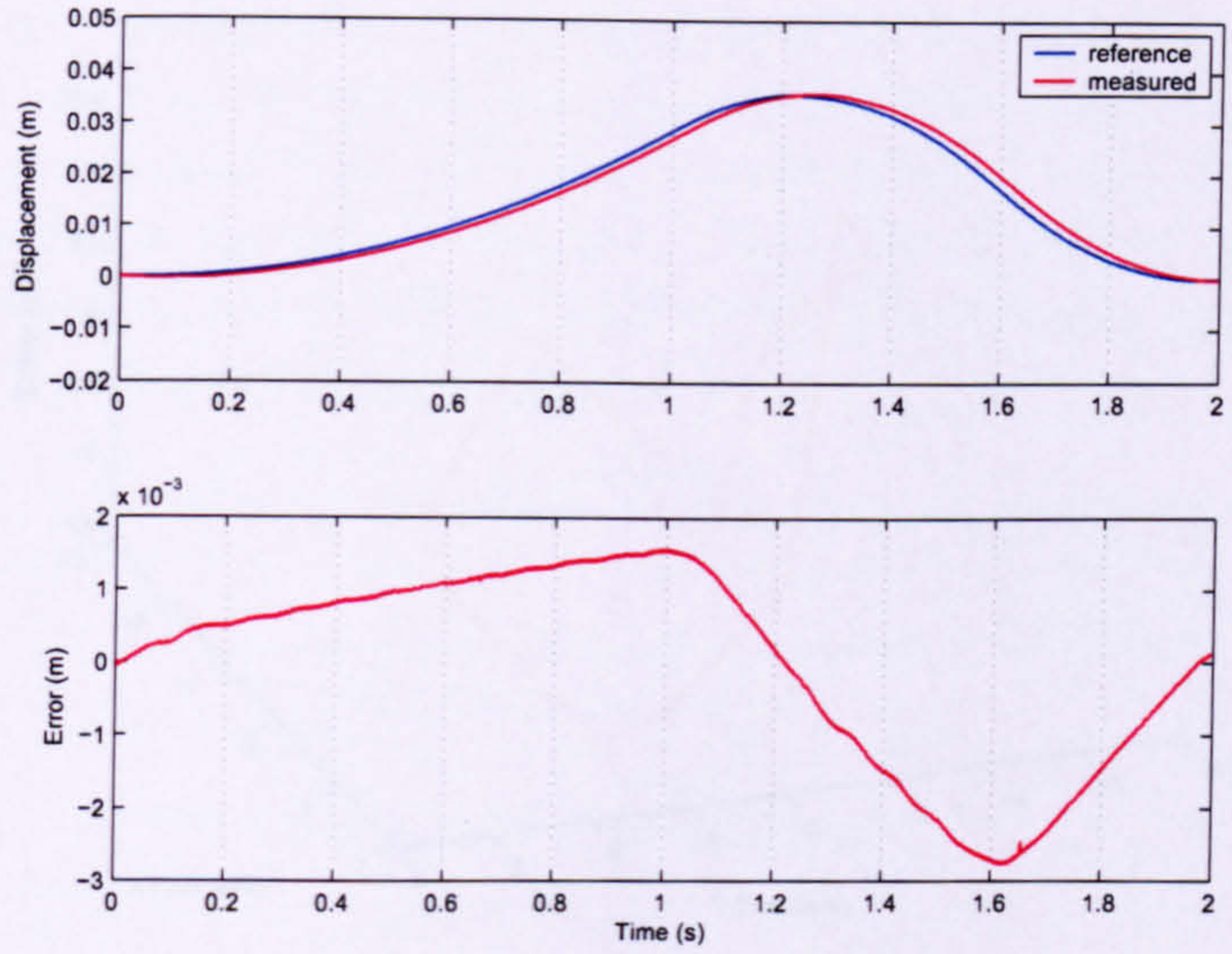


FIGURE 4.2: X-axis tracking performance and error at iteration 1000 (PID controller)

3-dimensional format. On this chart, perfect tracking would be represented by a point located at the origin. Consequently, Figure 4.4 clearly demonstrates there is potential for an improvement in tracking performance.

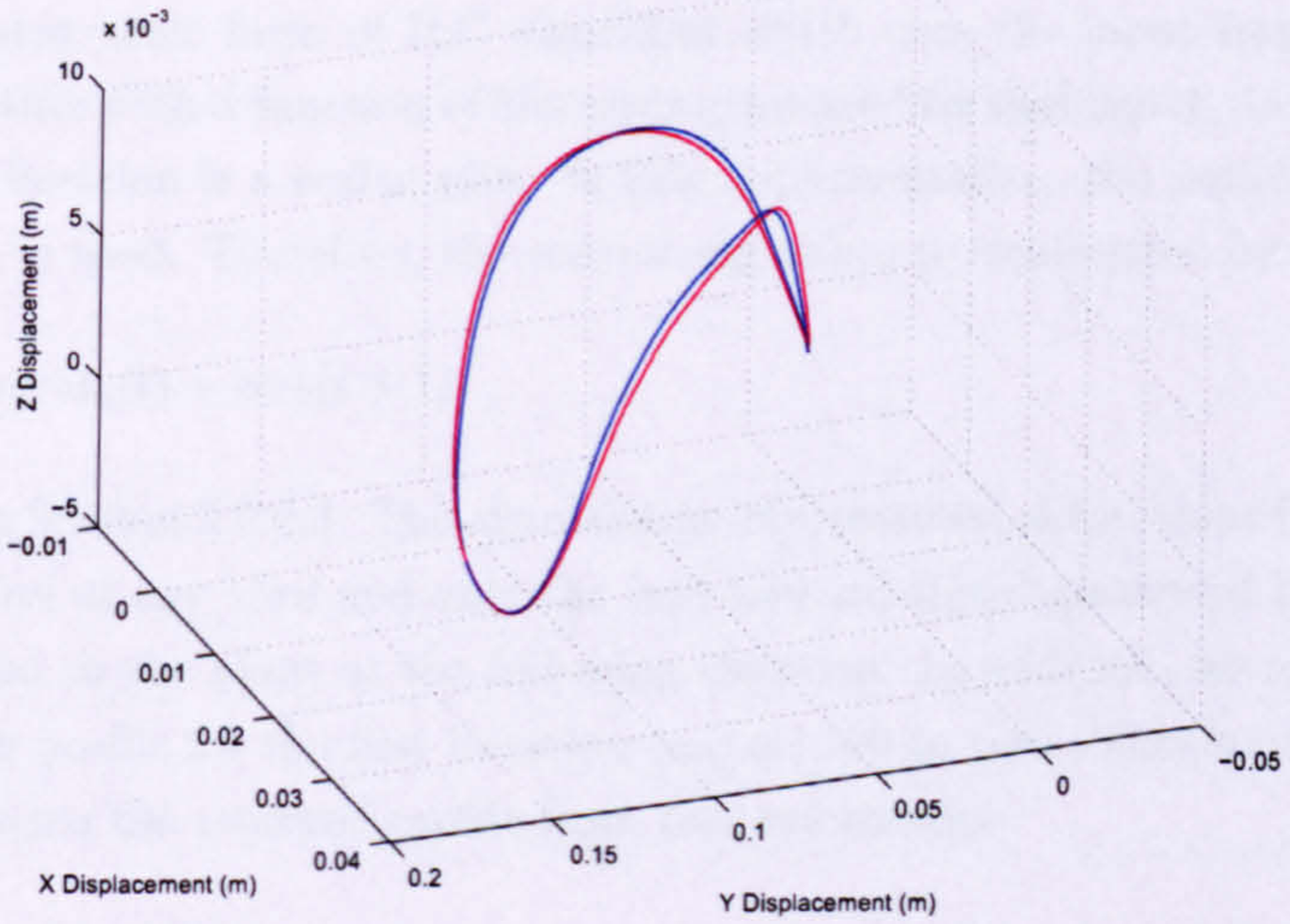


FIGURE 4.3: 3-Dimensional tracking performance at iteration 1000 (PID controller, blue = reference)

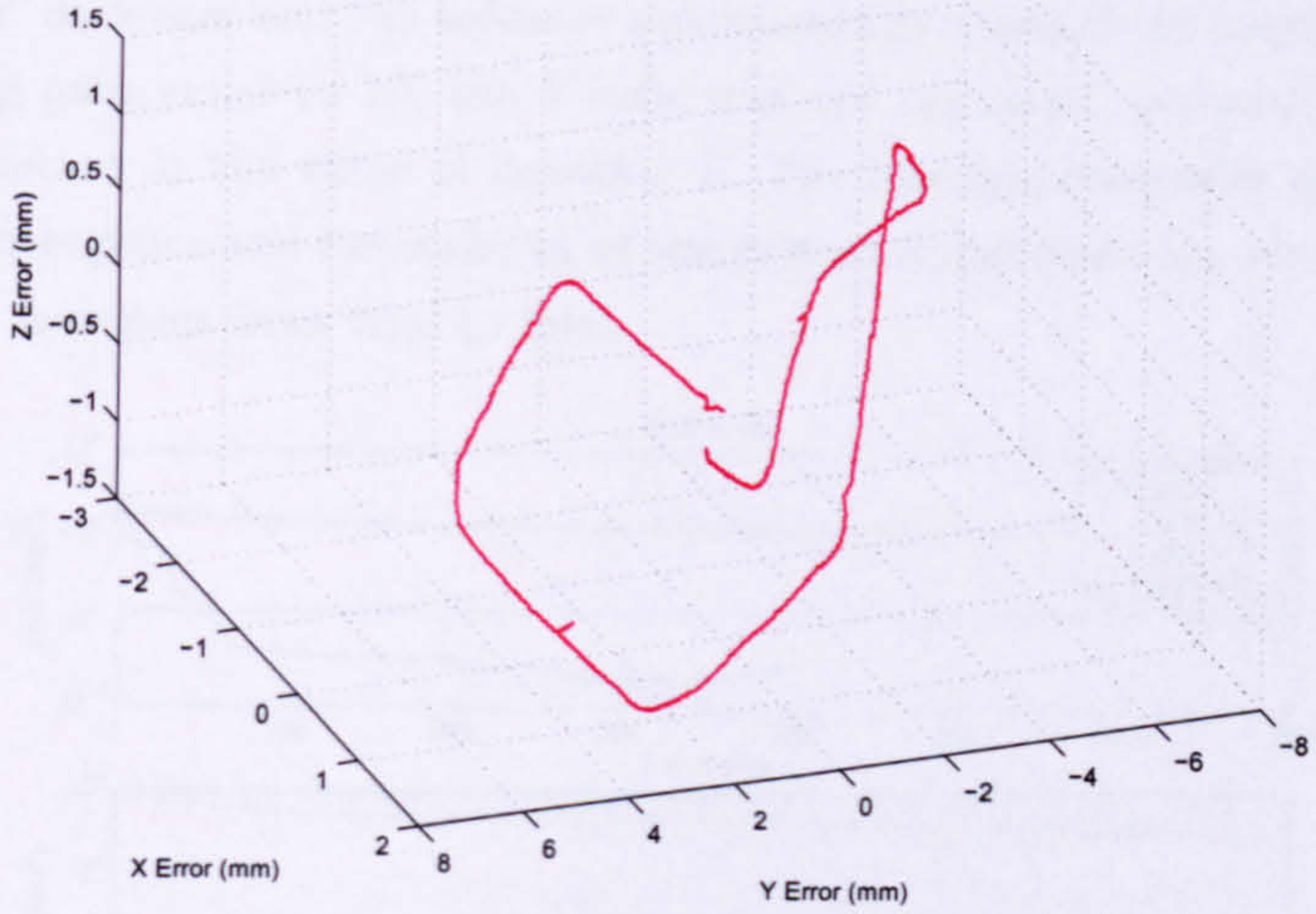


FIGURE 4.4: 3-Dimensional tracking error at iteration 1000 (PID controller)

4.3 Pure P-type ILC

4.3.1 Algorithm

This is the most basic form of ILC algorithm which uses the input from the previous iteration together with a function of the error generated by that input. In this particular example, the function is a scalar gain. In this implementation, the anticipatory form of the algorithm is used. Therefore, the control algorithm is represented by:

$$\mathbf{u}_{k+1}(t) = \mathbf{u}_k(t) + \alpha \mathbf{e}_k(t+1) \tag{4.2}$$

as discussed in Section 2.2.2.3. The algorithm is implemented alone, there is no additional feedback control of any kind and only the feed-forward signal generated by the learning controller is fed to the plant at the following iteration. In addition, all sample instants of the memory profile for the first iteration (u_0) are set to zero. This forces the learning controller to learn the required profile from zero information.

4.3.2 Test results

Figure 4.5 shows how the mse performance of the P-type algorithm varies for different values of scalar learning gain, as compared with the constant mse generated by the PID feedback controller. The learning controller has difficulty in matching the performance of the PID controller. In the case of the Y-axis, the learning controller results in an mse, over 2 orders of magnitude larger than that produced by the PID controller. For a learning gain equal to 1, the mse plots for all three axes terminate at iteration 68. This occurred

because the input demand to the Y-axis caused the robot to exceed the maximum travel limits in the Y direction and the software automatically shut-off the hardware for safety. For a learning gain equal to 10, the Y-axis was not operated, accounting for the mse remaining constant at the value of iteration 1. The learning controller did not produce monotonic convergence and the stability of the system is questionable, given that in some cases the mse increases from trial to trial.

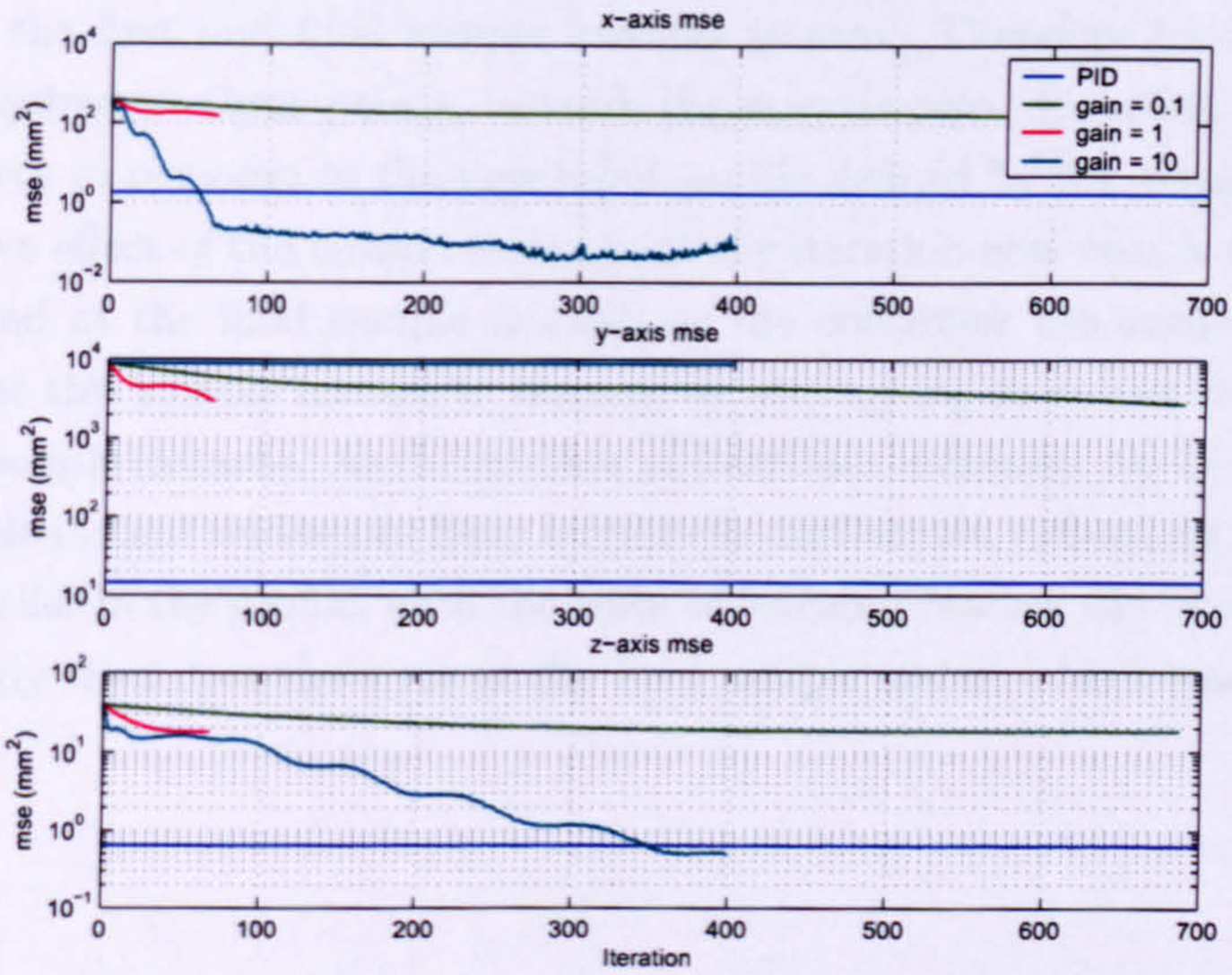


FIGURE 4.5: P-type ILC, mse performance compared to PID, (learning gains = 0.1, 1, 10)

The mse plot demonstrates that the pure feed-forward P-type learning controller is particularly badly suited to the gantry robot, due to the integrating effect of the plant, which adds -90 degrees phase shift to the frequency response and implies that the plant is open-loop unstable. The controller must also learn an input profile which resembles the derivative of the reference trajectory, rather than the reference trajectory itself. The error vector generated at the first iteration, when the robot is not yet in motion, is in fact equal to the reference trajectory. This explains why a large number of iterations are required for the controller to learn the derivative of the reference, based only on data obtained from the reference.

Though the learning process is particularly slow in these experiments, it does provide the opportunity to observe the mechanism by which the algorithm develops the required input profile in both the time and iteration domains. This process was described briefly in Section 2.3.2.2. Effectively, the controller must learn the first part of the profile before it can learn later sections, because the plant output is a continuous profile without discontinuities and therefore, the tracking performance achieved at the final sample instant depends on the performance achieved during all the previous sample instants. Conversely, the tracking performance achieved at the first sample instant does not depend

on the performance at any other time.

Figure 4.6 demonstrates this principle visually. The chart represents the motion of the X-axis with respect to time, but also with respect to iteration. Figure 4.7 represents the corresponding error variation for the same test. The first sample of every iteration lies on the vertical plane defined by time = 0. Equally, the final sample of every iteration lies on the plane defined by time = 2. The required displacement defined by the reference trajectory at the first and final sample instants is zero. Therefore for iteration 1, no learning is required at these points, because the error is zero. However, at iteration 2, the robot moves in response to the new input profile defined by the learning algorithm. The cumulative effect of the motion throughout the iteration now results in a large error being generated at the final sample instant, as the controller has actually caused the performance at this sample instant to worsen, by attempting to reduce the error during the previous sample instants. As the number of iterations increases, the error recorded at the final sample instant undergoes large amplitude oscillations, influenced by the motion of the axis, earlier in the profile, until the wave of learning reaches the later stages of the trajectory. Only then does the error at the final sample stabilise back towards zero.

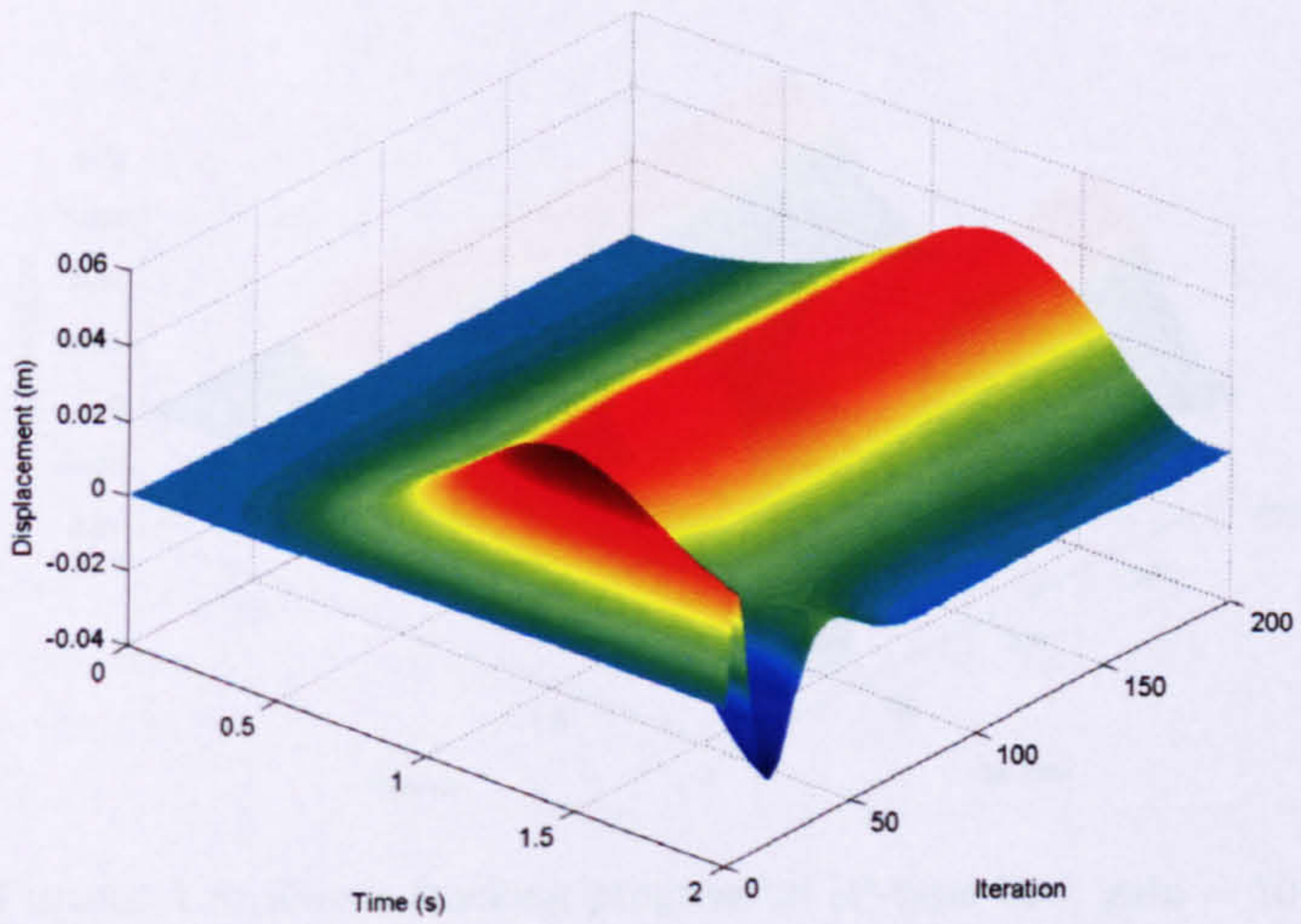


FIGURE 4.6: X-axis tracking progression (P-type ILC, gain = 10)

Figures 4.8 and 4.9 show similar plots for the Z-axis. In these examples, the progression of learning is clearly visible. Notice how the tracking progression chart should settle on the shape defined by the reference trajectory, while the error progression chart should settle on the horizontal plane defined by error = 0. For both the X and Z-axes, the P-type learning controller produces very slow error reduction. Increasing the learning gain is one possible solution to this problem. However, it produces larger input demands which can saturate the linear motor drives. This is especially true for the Y-axis where large inputs are automatically required to make this axis follow the reference trajectory, which has a larger displacement that the other axes.

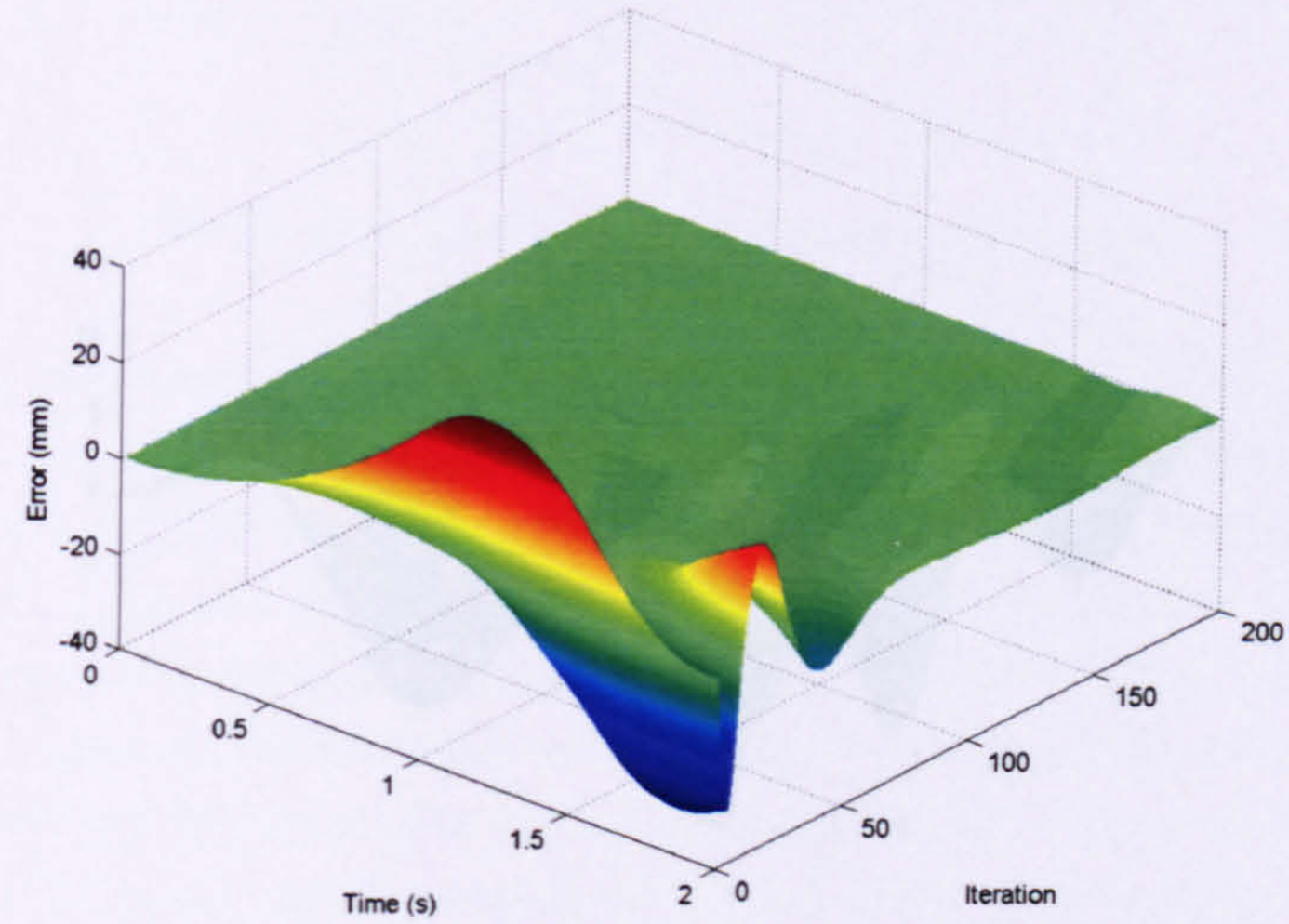


FIGURE 4.7: X-axis error progression (P-type ILC, gain = 10)

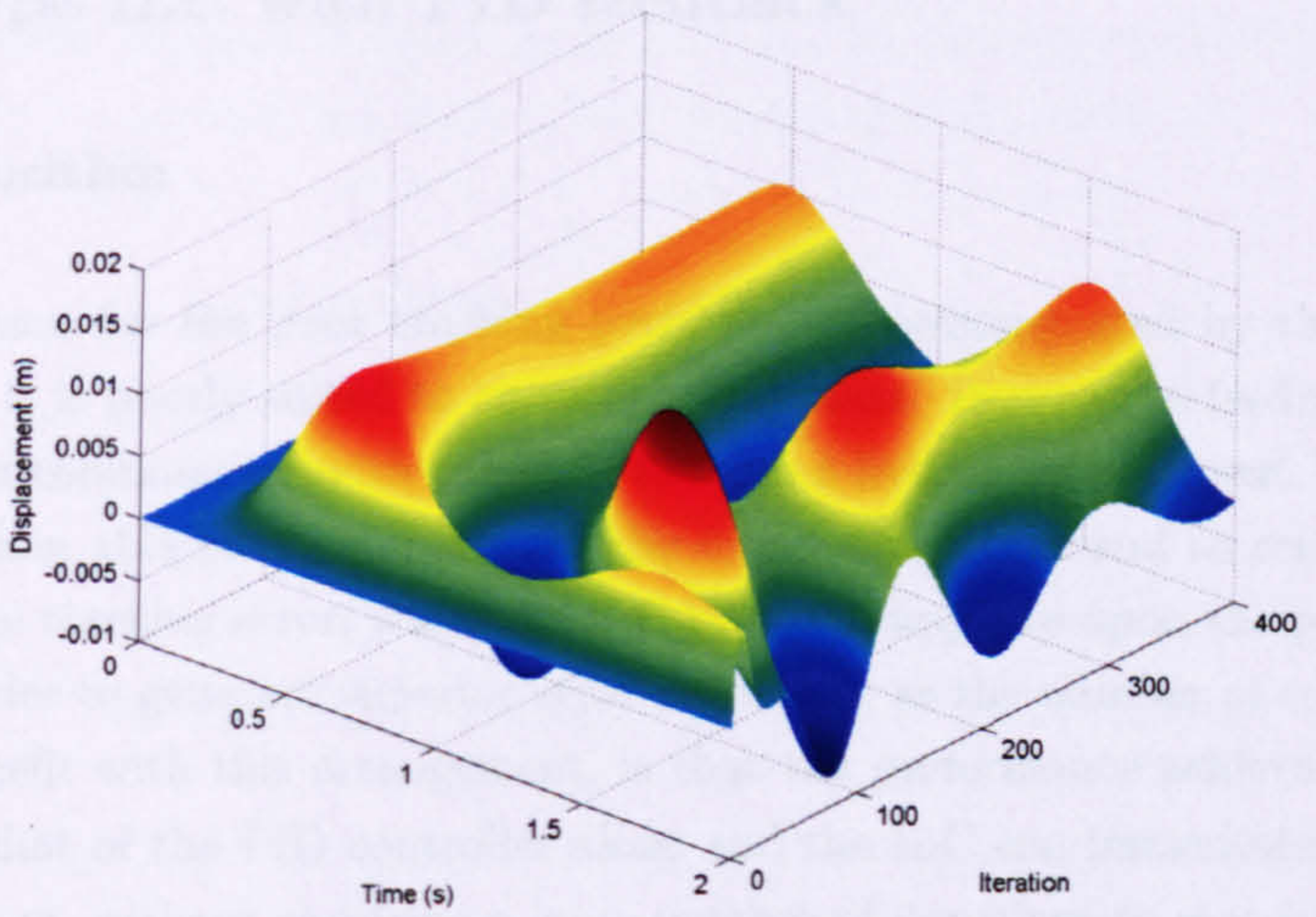


FIGURE 4.8: Z-axis tracking progression (P-type ILC, gain = 10)

Given that the P-type ILC requires many iterations to match the performance of the PID feedback controller and, for certain demands, is proven unable to match the performance, it is of little use when the overall objective is to improve the tracking performance beyond the capabilities of the PID controller. However, the ILC has clearly demonstrated the ability to learn and adapt the feed-forward input to the plant, while attempting to generate zero error. For the gantry robot, the simple P-type ILC has some potential and should not be discarded without further investigation.

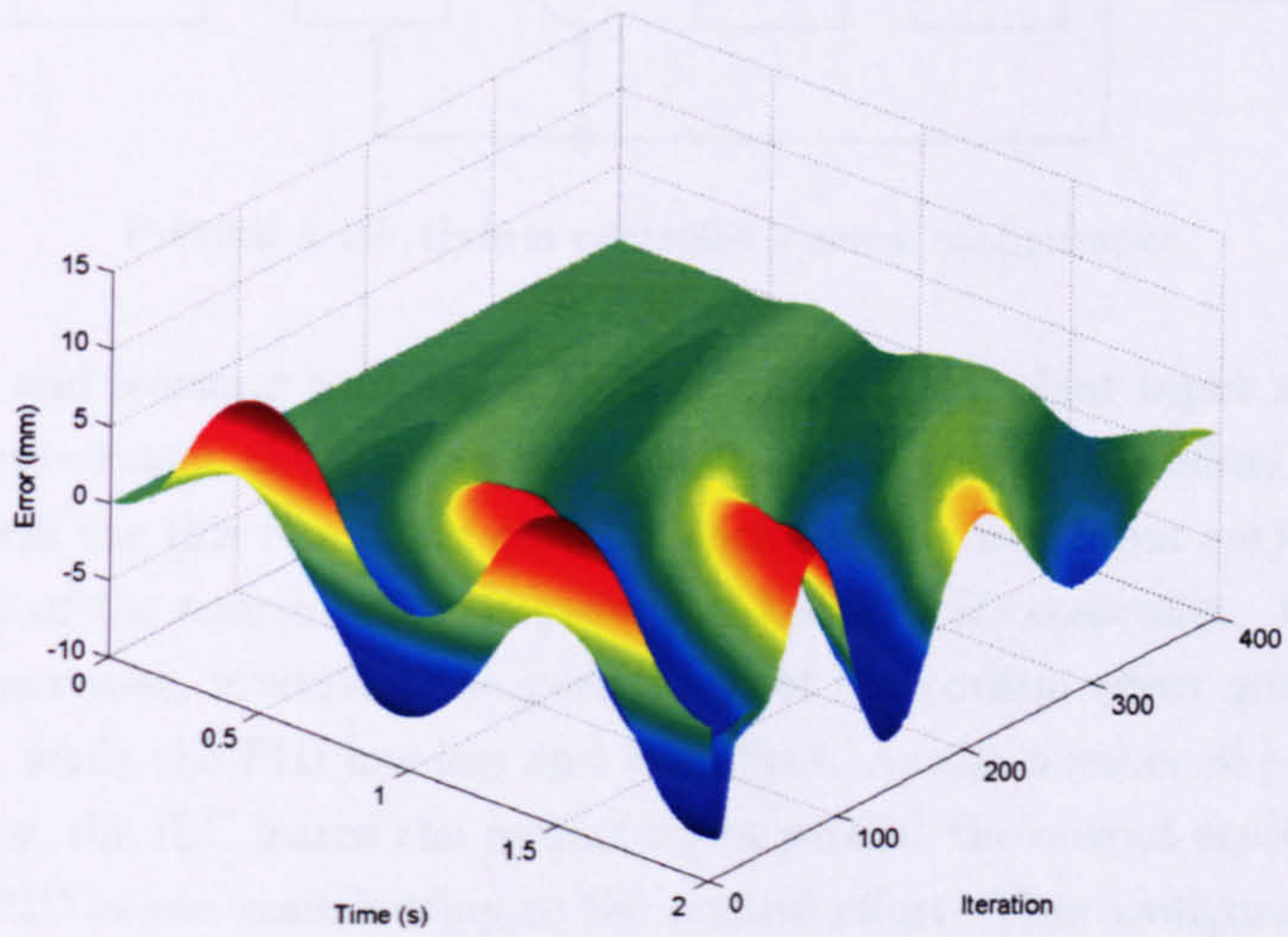


FIGURE 4.9: Z-axis error progression (P-type ILC, gain = 10)

4.4 P-type ILC with PID feedback

4.4.1 Algorithm

The main reason for the poor tracking performance demonstrated by the P-type ILC alone, is that it is poorly suited to an integrating plant. However, a feedback controller can be used in combination with the ILC, to form a hybrid arrangement. The feedback controller is then able to compensate for the integrating plant and to remove the large majority of the tracking error, while the ILC learns to improve upon the performance of the PID in order to generate superior error reduction, as the number of trials increases. An added benefit with this arrangement, is that the performance achieved at iteration 1 is equal to that of the PID controller alone and the ILC can immediately build upon this performance, without requiring a large number of iterations to match it.

Having combined two controllers into one, it is essential to select a suitable configuration. Drawing an analogy with components in an electrical circuit, there exist two main configurations for the combined PID and ILC controller: parallel and series.

Figure 4.10 shows the block diagram representation of the series arrangement. In this configuration, the two controllers operate in turn on the plant input signal. In effect, the ILC is used to learn the input demand to the PID controller, which will compensate for the error produced. Fundamentally, the PID controller still generates error with respect to the input received, but because the ILC has already adjusted this input, the error between the PID output and the original reference trajectory is reduced.

Similarly, Figure 4.11 shows the block diagram representation of the parallel arrangement.

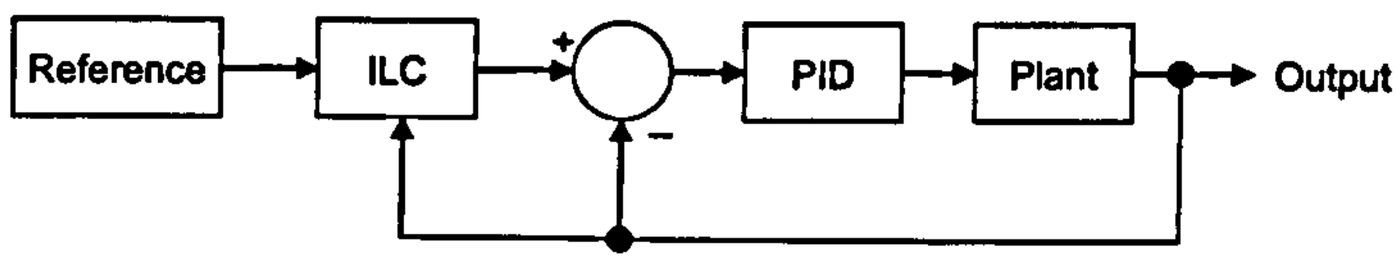


FIGURE 4.10: Hybrid controller - series configuration

The feedback and learning controllers both operate on the plant input simultaneously. As discussed previously, the PID controller will always generate residual tracking error and the ILC will use this remaining error to learn the required input trajectory. During iteration 1, all of the control effort is generated by the PID controller. As the number of iterations increases, gradually the percentage of the control effort generated by the ILC increases, while the PID has less and less effect. As the number of iterations tends towards infinity, the ILC learns the perfect input profile, the output error is reduced to zero and the PID ceases contributing to the control effort. This configuration is similar to using the ILC controller alone, except that the transition from PID to ILC results in significantly improved performance during the earlier iterations. This arrangement also has the added advantage of increased robustness to non-repeating (iteration dependent) errors and sudden changes in plant dynamics (due for example to changes in payload) because the PID controller can immediately compensate for these.

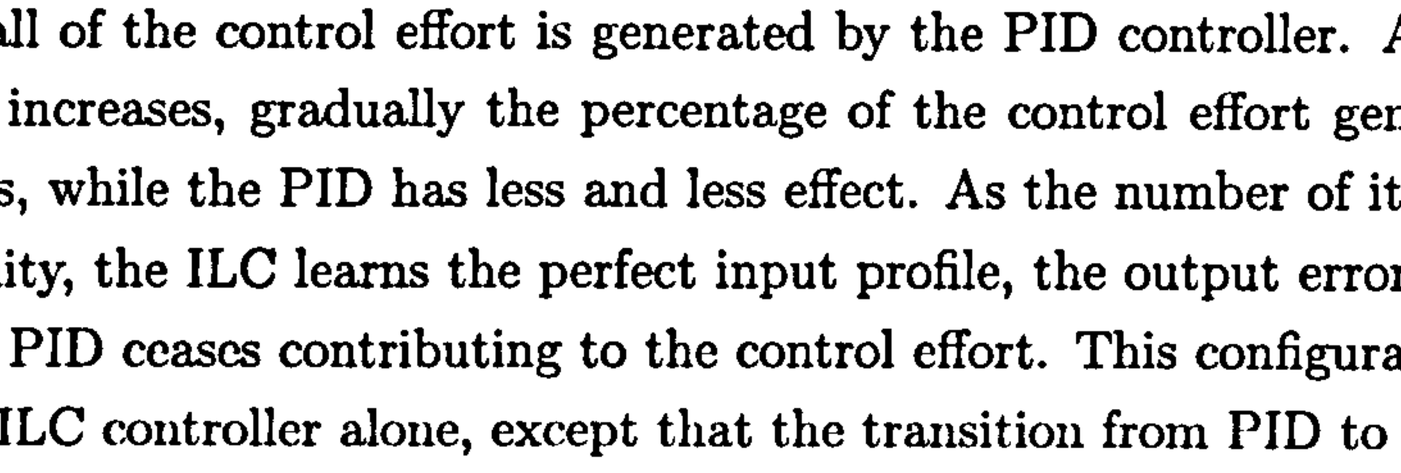


FIGURE 4.11: Hybrid controller - parallel configuration

4.4.2 Initial implementation

Both the series and parallel configurations have been implemented on the gantry robot, this allows comparison between the two methods. In both cases, the PID controller gain values were identical.

Figure 4.12 displays the mse performance for the series arrangement when the learning gain α is set to 0.1, 0.01, 0.001 and 0.0001. The benchmark PID performance is also included on the plots. In comparison to the P-type ILC alone, the hybrid arrangement is able to improve the tracking performance beyond the limitations of the PID controller. The mse can be reduced by over two orders of magnitude on all three axes, which is a significant achievement. Varying the learning gain now produces the expected linear effects on error reduction. High gain results in faster error reduction in comparison to

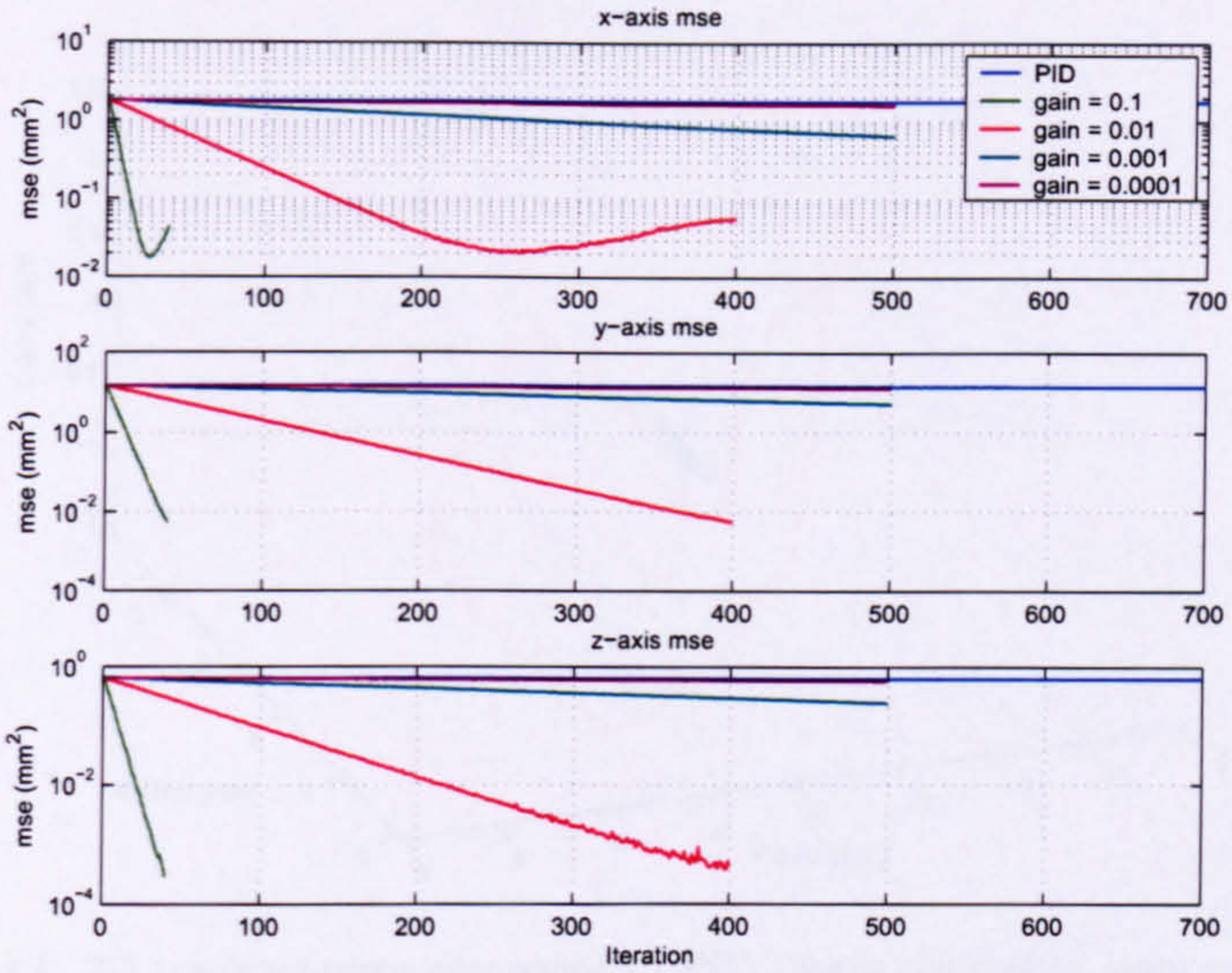


FIGURE 4.12: mse for different learning gains (series controller)

lower gain, which produces slower error reduction, because the algorithm is numerically unable to make large changes to u_{k+1}. The PI_{100} values associated with each series controller learning gain are presented in Table 4.3.

TABLE 4.3: Series PID and P-type ILC, PI_{100} for different learning gains

Gain	X-axis	Y-axis	Z-axis
0.1	100.000000	100.000000	100.000000
0.01	43.726610	43.429009	44.523145
0.001	91.931639	91.466577	90.020137
0.0001	99.045063	99.177682	98.709156

Figure 4.13 displays the 3-Dimensional tracking error plot obtained during iteration 40 with a learning gain of 0.1, compared to the error produced by the PID controller. The blue oval near the origin represents the maximum error amplitude of the hybrid controller which has much smaller bounds than the PID controller.

The overall tracking performance has been increased, but not without cost. From Figure 4.12 it can be seen that the mse plots for high gain values suddenly stop short of the 500 iterations required for the test. For example, with gain 0.1, only 40 iterations are achieved, while with gain 0.01, 400 iterations are achieved. In both cases, the tests had to be interrupted because the robot was experiencing severe mechanical vibrations of amplitude, large enough to necessitate the system to be shut down. The controller became unstable, as confirmed by the increase in mse following the initial reduction which is prominent in the X-axis plot. For gain 0.1, these vibrations are clearly visible in the tracking error progression plot for the X-axis (Figure 4.14). After the initial error

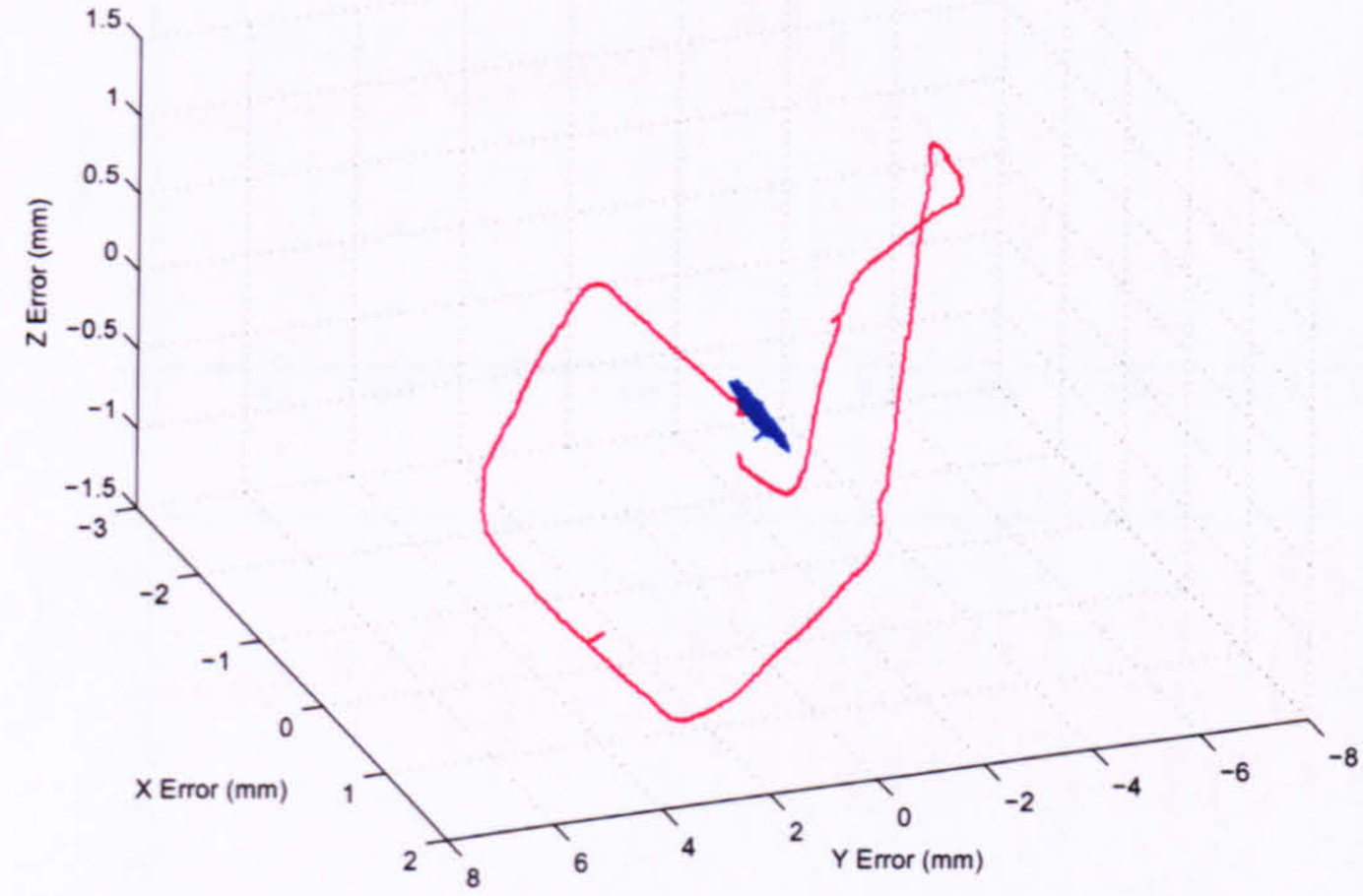


FIGURE 4.13: 3D tracking error compared to PID (series controller, gain = 0.1, iteration 40, red = PID)

reduction, ripples begin to appear as early as iteration 10 and by iteration 40, the entire system is dominated by vibrations.

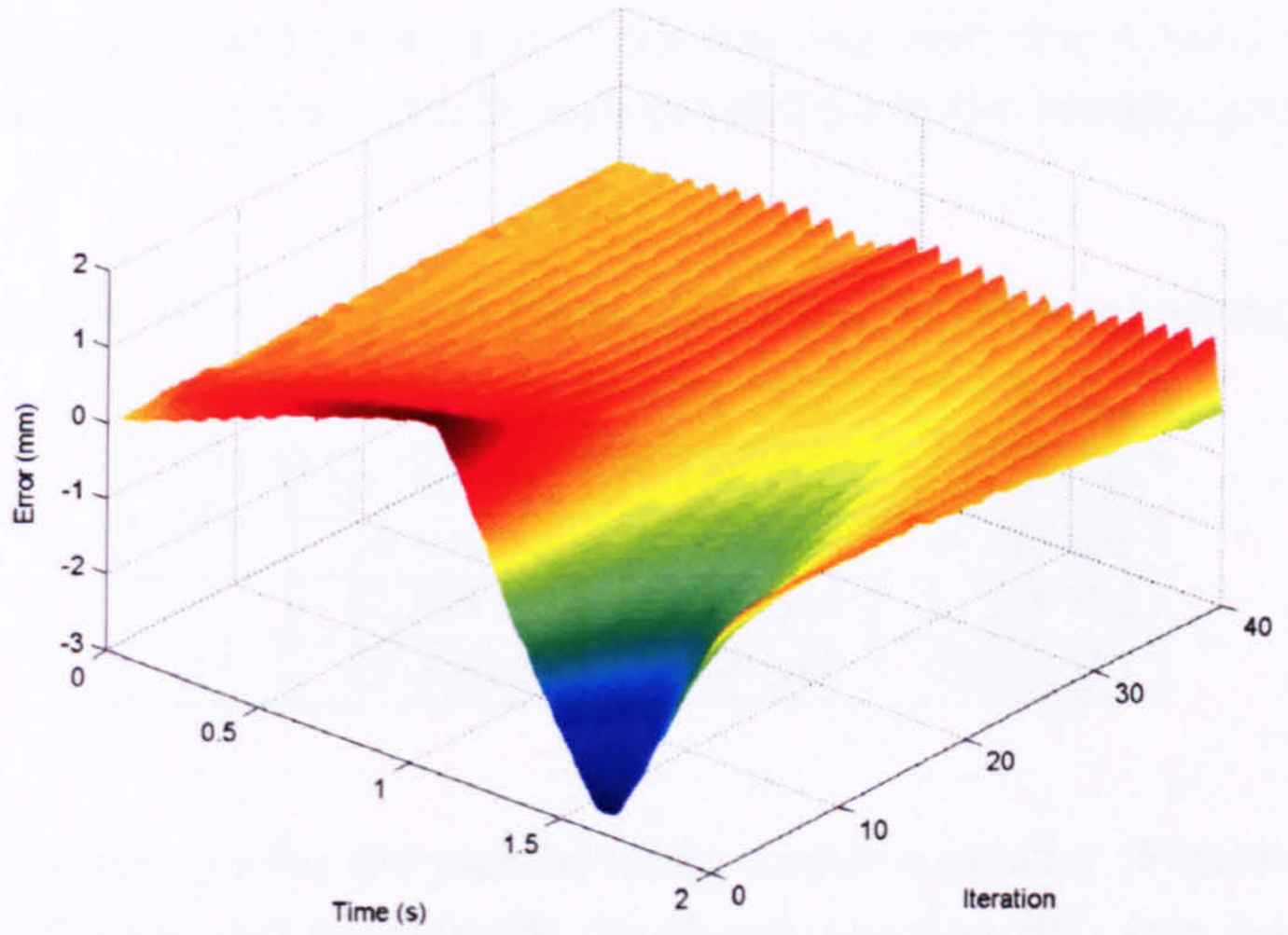


FIGURE 4.14: X-axis tracking error progression (series controller, gain = 0.1)

The lower section of Figure 4.15 confirms the severity of the error oscillations and the upper section indicates that they are a result of the input supplied to the plant (similar results can be seen in Appendix B for the Y and Z-axes).

Figure 4.16 displays the equivalent mse plots for the parallel hybrid arrangement. The characteristics of these plots are almost identical to those of the series configuration. Higher learning gain results in faster convergence, but also in a more rapid onset of instability. It is relevant to point out that the learning gain is generally much larger than

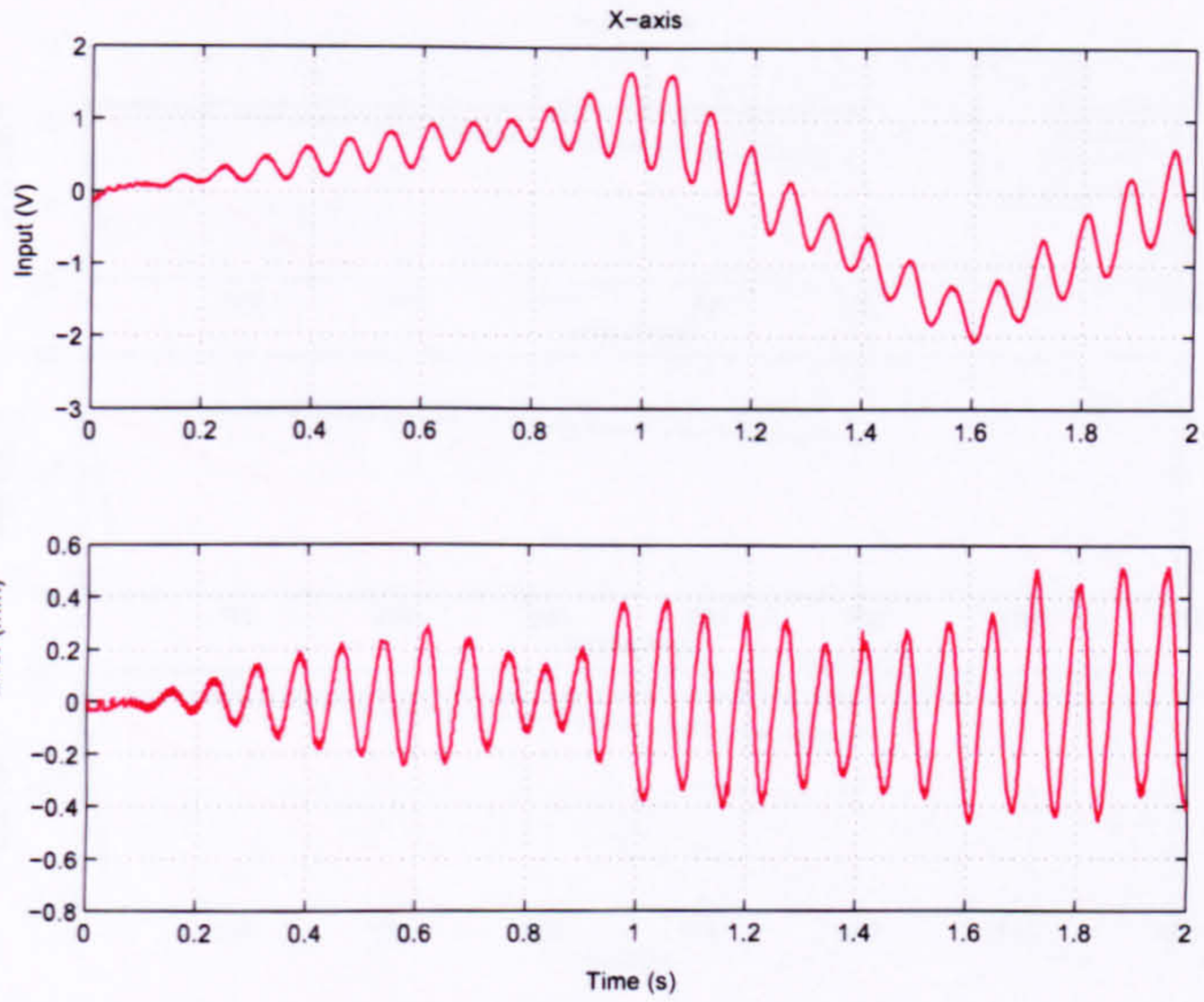


FIGURE 4.15: X-axis input demand and tracking error (series controller, gain = 0.1, iteration 40)

for the series arrangement because the learning controller needs to develop the trajectory from zero information, and the output of the learning controller is not amplified by the PID. The PI_{100} values associated with each parallel controller learning gain are presented in Table 4.4.

TABLE 4.4: Parallel PID and P-type ILC, PI_{100} for different learning gains

Gain	X-axis	Y-axis	Z-axis
100	100.000000	100.000000	100.000000
10	29.952786	37.575756	47.441492
1	85.399418	88.731863	91.938774
0.1	98.368823	98.680485	99.300515

The input and error plots for the parallel configuration controller (Figures 4.17 and 4.18 for the X and Y-axes and Appendix B the Z-axis) confirm that this system is equally susceptible to vibrations. For the parallel configuration, both the PID and ILC outputs are displayed because both have a direct effect on the plant.

The improvements in tracking performance gained by using ILC are of little importance, if the algorithm is unstable and oscillates after a small number of iterations. It is therefore necessary to modify the controller so that these oscillations do not occur. In order to achieve this objective, a thorough understanding of what causes oscillation is essential. It is clear from both Figures 4.15 and 4.17 for the X-axis that the vibrations are not random but consist of distinct frequencies. The Z-axis is not particularly affected by vibrations. Frequency domain analysis of the parallel configuration tracking error at

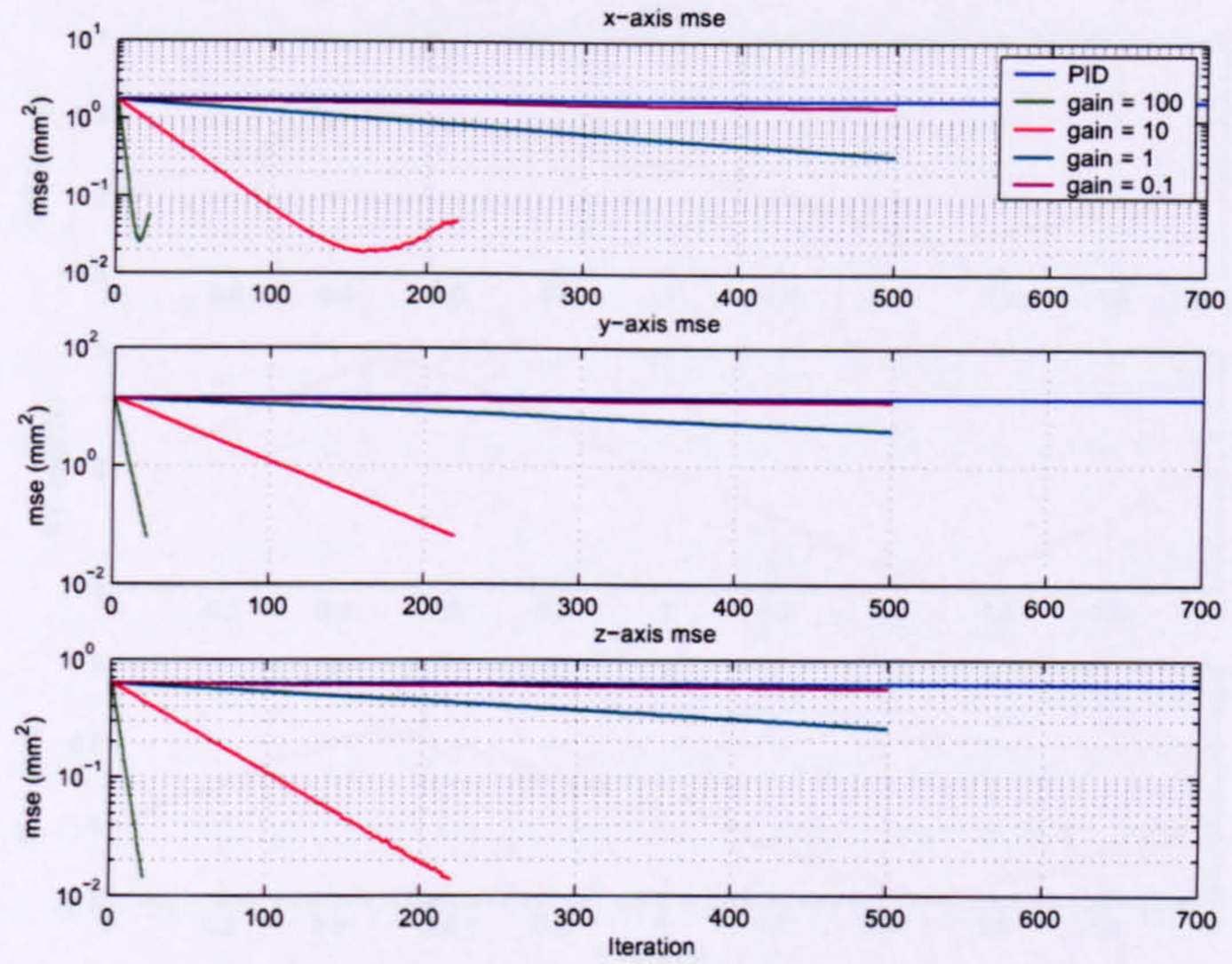


FIGURE 4.16: Parallel controller - mse for different learning gains

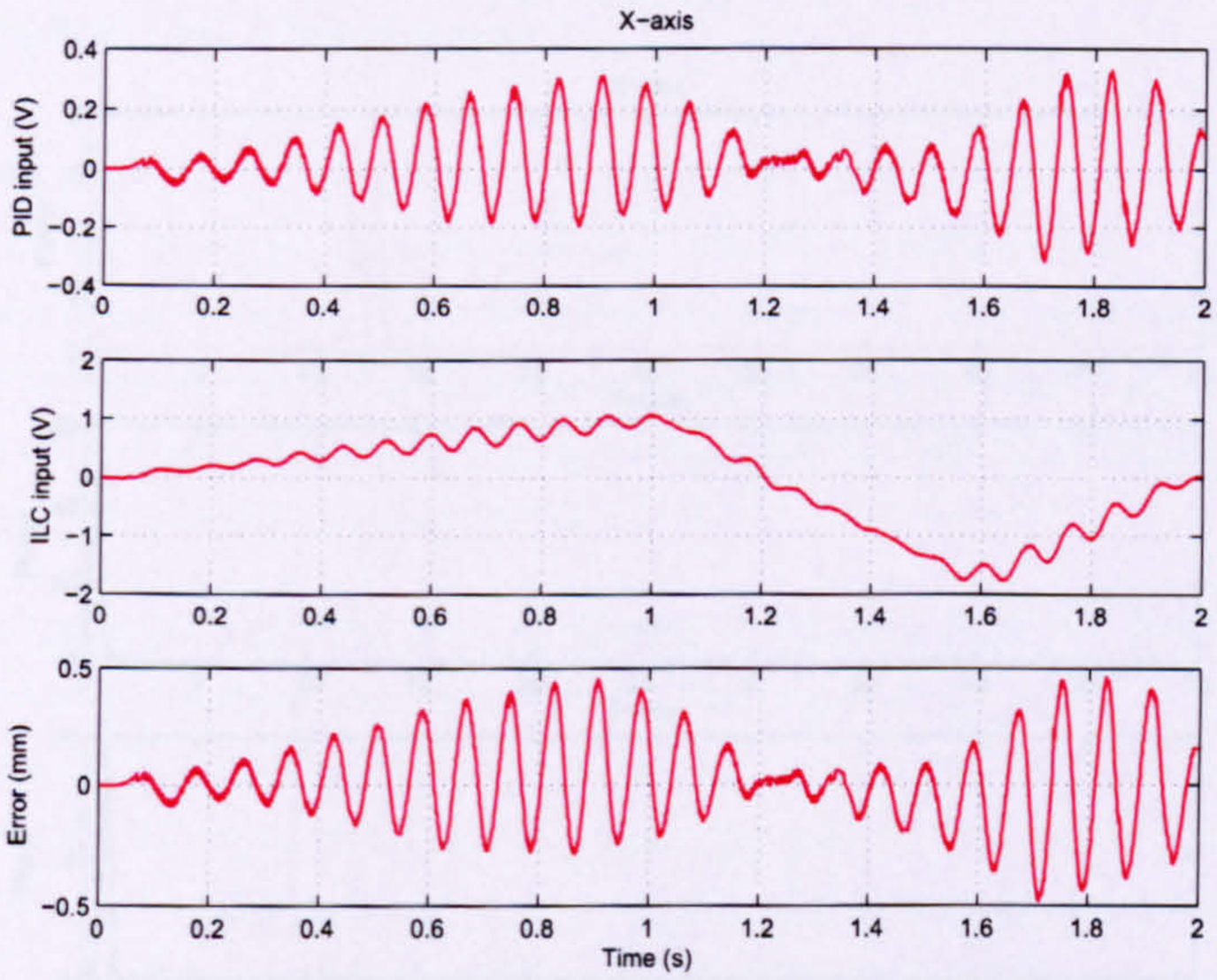


FIGURE 4.17: X-axis PID, ILC and tracking error (parallel controller, gain = 100, iteration 20)

iteration 20 with learning gain 100, produces the frequency spectra in Figure 4.19. The error present on the Y and Z-axes is the result of very low frequency components representing the fundamental frequencies of the reference trajectories. However, the X-axis error dominantly consists of frequency components at 12 and 13 Hz (75 and 81 rad/s), frequencies which are not related to the reference trajectory. Detailed analysis of the frequency spectra also reveals a small component of the Y-axis error, also at frequencies 12 and 13 Hz, which corresponds to the ripples in Figure 4.18.

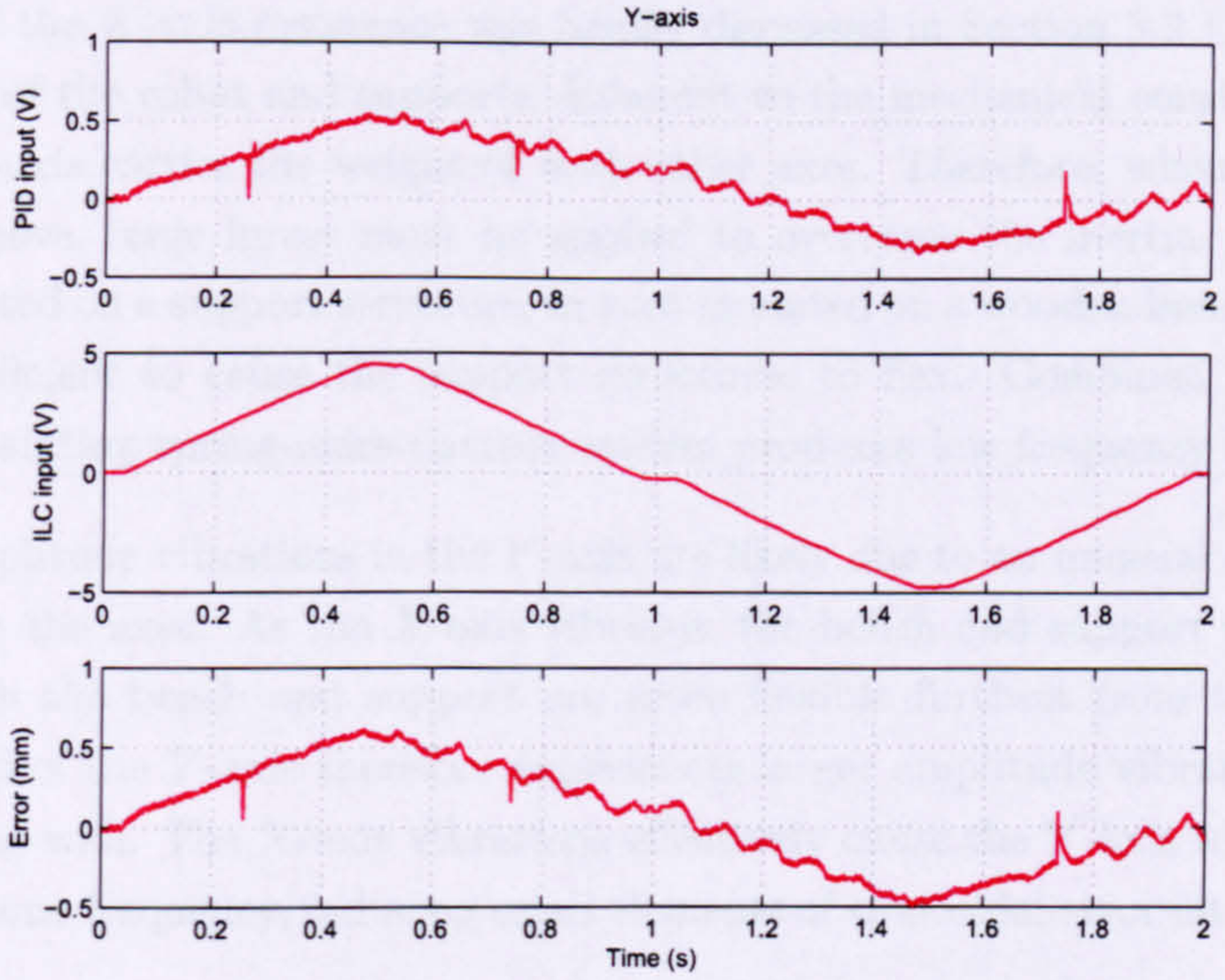


FIGURE 4.18: Y-axis PID, ILC and tracking error (parallel controller, gain = 100, iteration 20)

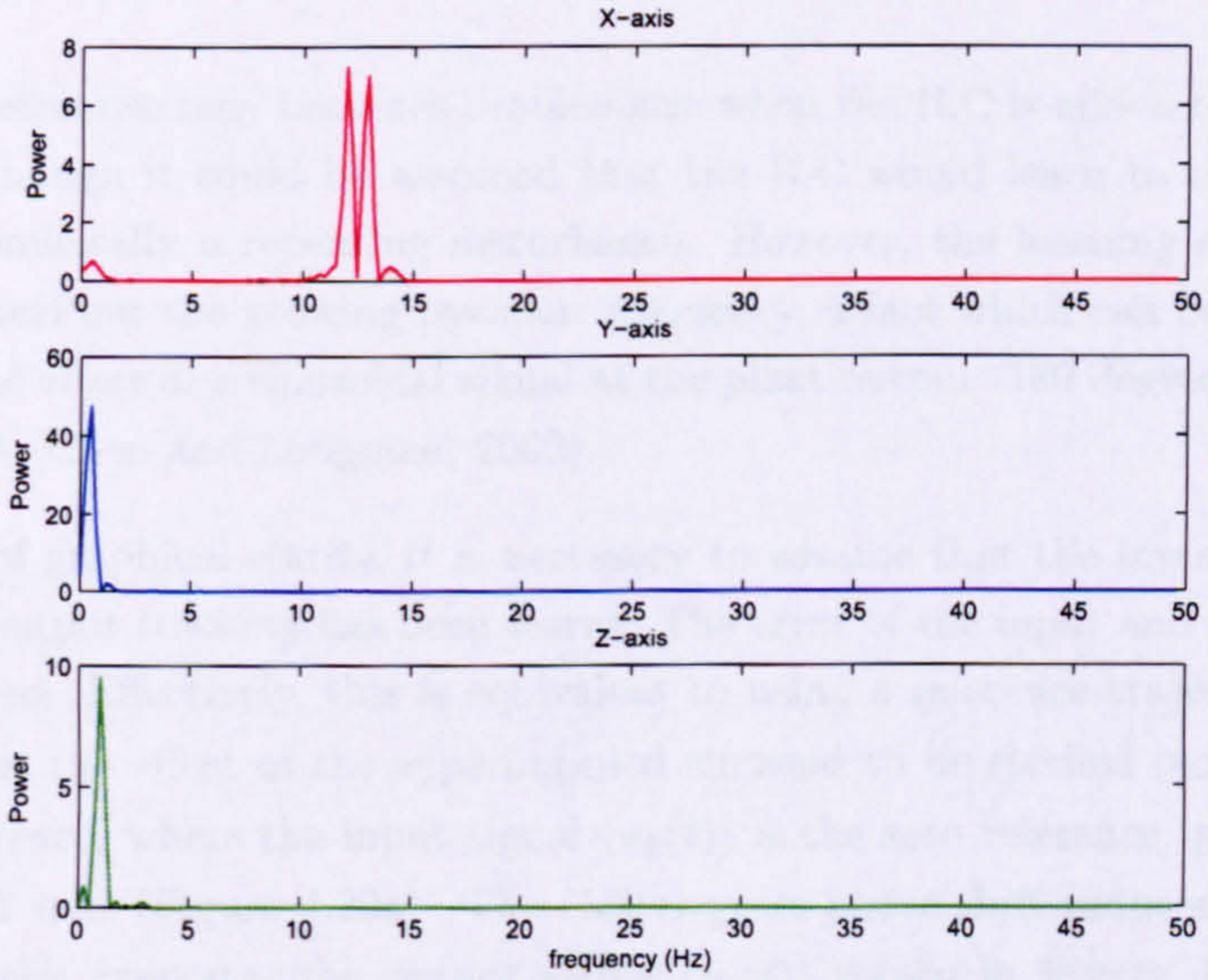


FIGURE 4.19: Error signal frequency spectra (parallel controller, gain = 100, iteration 40)

To find the source of these 12-13 Hz vibrations, it is necessary to return to the frequency response Bode plots which were used during plant modelling. The X-axis Bode plot (Section 3.3, Figure 3.11) has resonant peaks at 75 and 85 rad/s which correspond perfectly to the 12 and 13 Hz vibrations. The Y and Z-axes exhibit natural resonance at approximately 250 and 400 rad/s (40 and 63 Hz) respectively (comparatively higher frequencies, not significant to the measured vibrations). This suggests that the relatively low, natural resonance of the X-axis is the fundamental source of the ILC instability.

The source of the X-axis resonance was briefly discussed in Section 3.2.1 and is related to the design of the robot and supports. Inherent to the mechanical construction of the robot, the X-axis carries the weight of both other axes. Therefore, when the X-axis is required to move, large forces must be applied to overcome the inertia. As the entire robot is mounted on a support structure, in turn mounted on a wooden bench, the applied forces are sufficient to cause the support structures to flex. Combined with the high inertia, the resulting spring-mass-damper system produces low frequency resonances.

The small amplitude vibrations in the Y-axis are likely due to an unusual cross-coupling effect between the axes. As the X-axis vibrates, the bench and support structure flex. However, both the bench and support are more flexible furthest from the laboratory wall. This end of the Y-axis therefore experiences larger amplitude vibrations than the end nearest the wall. The X-axis vibrations effectively cause the Y-axis to pitch up and down at the same frequency, inducing small elements of sinusoidal error into the tracking performance.

4.4.3 Stability and convergence

The X-axis resonance only becomes problematic when the ILC is added to the feedback controller, although it could be assumed that the ILC would learn to compensate for what is fundamentally a repeating disturbance. However, the learning controller does not act to cancel out the growing resonant frequency, a fact which can be explained by considering the effect of a sinusoidal signal at the plant output -180 degrees out of phase with the input (Chen and Longman, 2002).

For the sake of graphical clarity, it is necessary to assume that the input signal which gives perfect output tracking has been learnt. The error of the input and output signals is therefore zero. Effectively, this is equivalent to using a reference trajectory ($r(t)$) of zero and allows the effect of the superimposed sinusoid to be studied independently of the main reference, where the input signal ($u_k(t)$) is the zero reference, plus a sinusoid of amplitude 1 unit (Figure 4.20a). The -180 degrees phase shift induced by the plant at this frequency generates the output signal ($y_k(t)$) shown in Figure 4.20b which is arbitrarily assigned an amplitude of 1 unit for simplicity. Remembering that the error signal is generated from:

$$\mathbf{e}_k(t) = \mathbf{r}(t) - \mathbf{y}_k(t) \tag{4.3}$$

the resulting error is shown in Figure 4.20c. In effect, the -180 degree phase shift has been completely removed from the error signal by subtracting the output from the reference and, when the learning controller:

$$\mathbf{u}_{k+1}(t) = \mathbf{u}_k(t) + \alpha \mathbf{e}_k(t+1) \tag{4.4}$$

calculates the input for the next iteration, the amplitude of the input oscillation is actually increased rather than decreased. Figure 4.20d shows the new input signal ($\mathbf{u}_{k+1}(t)$) if the learning gain $\alpha = 1$.

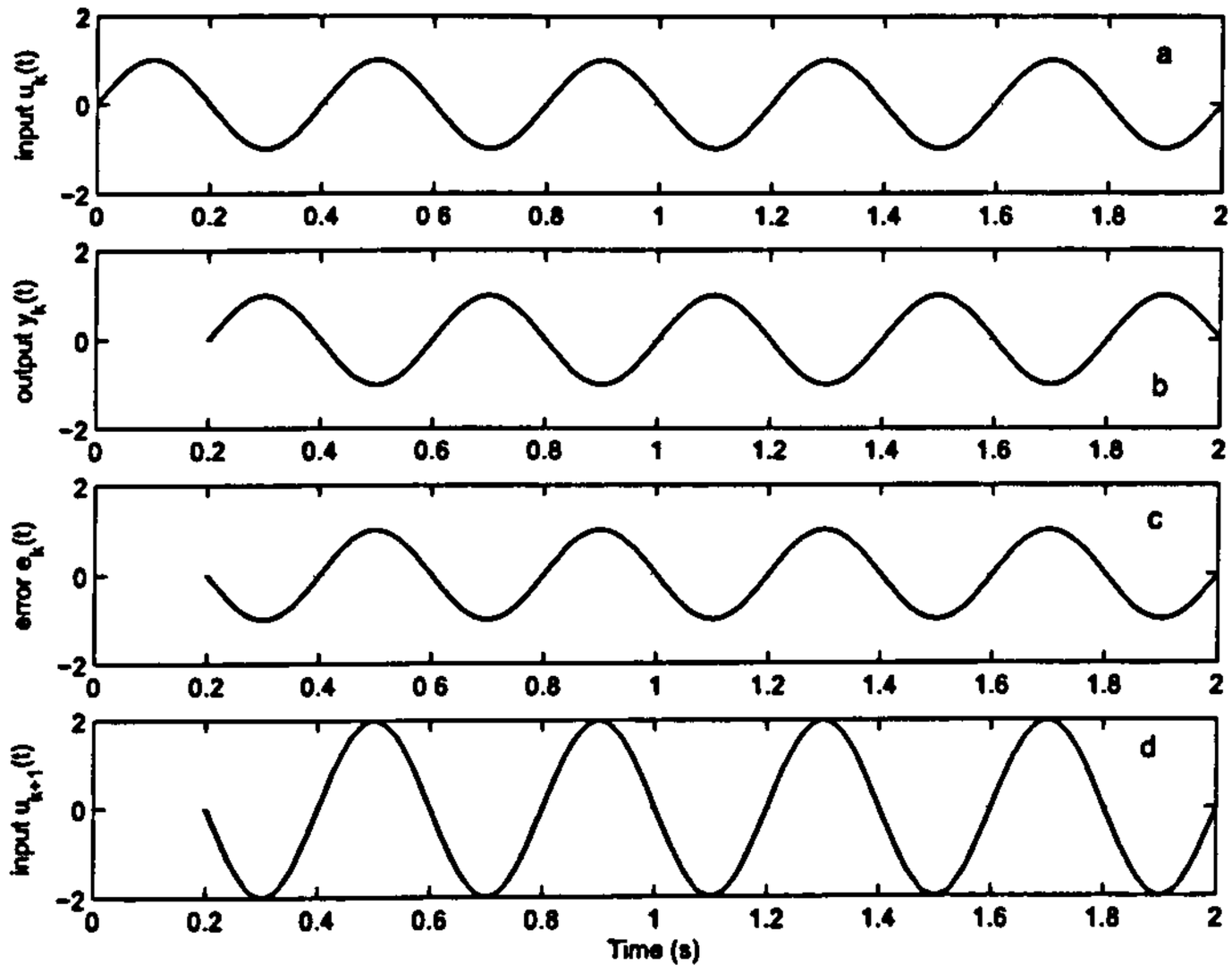


FIGURE 4.20: Mechanism of instability for resonant systems (a = sinusoid, b = shifted -180 degrees, c = error signal, d = next iteration input)

When the frequency response of the plant passes through the -180 degrees point ($\omega\pi$), the P-type learning controller incorrectly compensates for the error at frequencies above $\omega\pi$ and actively increases error. Resonant frequencies automatically add -180 degrees of phase shift and hence are particularly dangerous to the stability of the learning algorithm. In order to stabilise the P-type learning controller, it is necessary to prevent the growth of resonant frequencies. The simplest approach is to switch the learning off once the mse has been reduced to a sufficient level (Barton et al., 2000). However, the system is then no longer able to compensate for gradual changes in the plant, caused by wear or temperature for example.

It is also possible to investigate the instability of the parallel hybrid controller by means of a convergence criterion discussed by Steinbuch and van de Molengraft (2000). The system is represented by Figure 4.21 where $G(s)$ is the plant, $K(s)$ is a stabilizing feedback controller, $L(s)$ is a learning filter and $Q(s)$ is a robustness filter. If the reference trajectory is taken as zero ($r_k = 0$) then the analysis is as follows. The error at each

iteration is:

$$
\begin{aligned}
e_k &= r_k - y_k && (4.5)\\
&= 0 - (e_k K + f_k)G && (4.6)\\
&= -f_k G - e_k KG && (4.7)\\
e_k(1+KG) &= -f_k G && (4.8)\\
e_k &= \frac{-G}{1+KG} f_k && (4.9)
\end{aligned}
$$

The learning update rule is:

$$f_{k+1} = Q(f_k + Le_k) \tag{4.10}$$

then eliminate f:

$$
\begin{aligned}
e_{k+1} &= \frac{-G}{1+GK} f_{k+1} && (4.11)\\
&= \frac{-G}{1+GK} Q(f_k + Le_k) && (4.12)\\
e_{k+1} &= Q\left(1 + \frac{-G}{1+GK} L\right) e_k && (4.13)
\end{aligned}
$$

which determines how the error signal evolves from trial to trial. To achieve monotonic convergence, e_{k+1} must be smaller than e_k. Therefore monotonic convergence will occur if:

$$\left| Q\left(1 - \frac{G}{1+GK} L\right) \right| < 1 \tag{4.14}$$

Effectively, the plot of Equation 4.14 in the complex plane must lie within the unit circle to guarantee monotonic convergence. In the complex plane, a phase shift of -180 degrees implies that the plot moves to the right of the graph. As the criterion causes the plot to start at a magnitude of 1 on the real axis, a phase shift of -180 degrees immediately violates the conditions for monotonic convergence. This agrees with the theory presented earlier in this section. The criterion is used to analyse the convergence of the parallel hybrid controller in Section 4.5.

4.4.4 Band-stop filtering

Removal of specified frequencies from a given signal immediately suggests the use of digital filtering techniques. Digital filters can be designed to achieve specific levels of attenuation at given frequencies. On a Bode plot of the filter transfer function, it corresponds to a significant drop in gain at the frequencies which need to be filtered. Ideally,

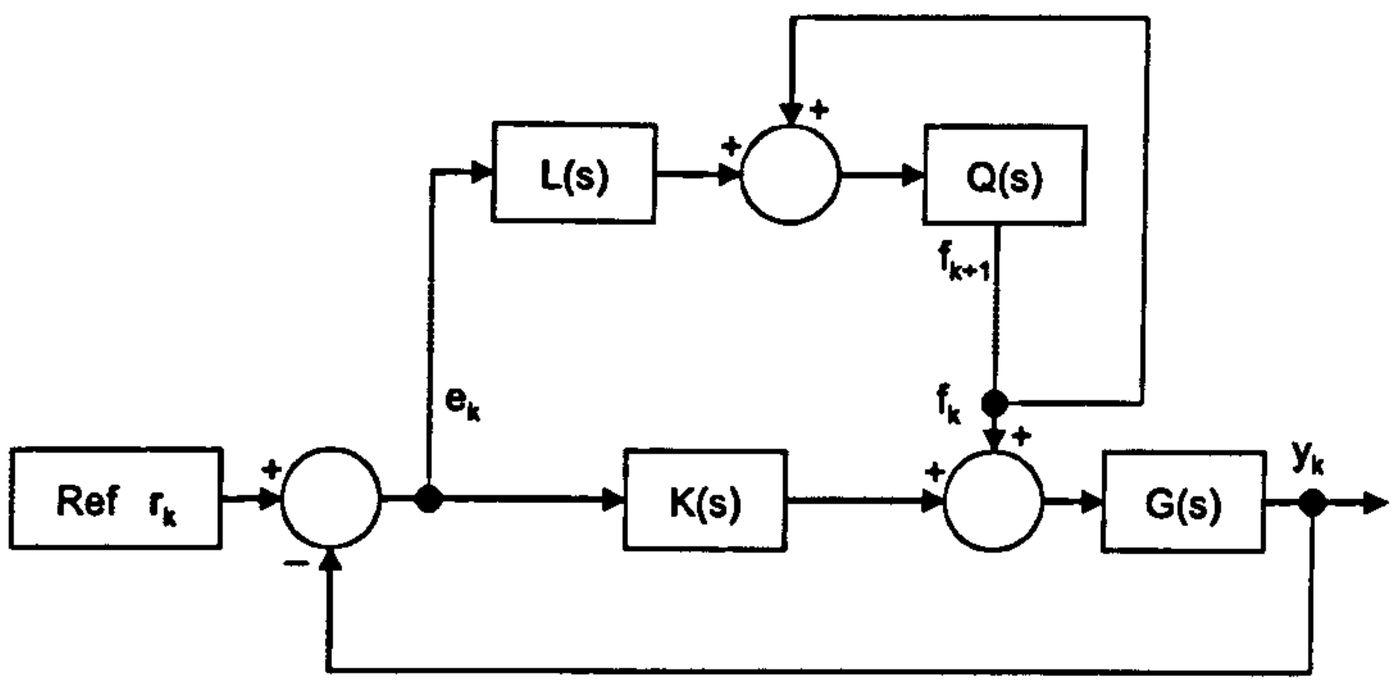


FIGURE 4.21: Parallel controller for convergence analysis

the specified frequencies are removed, while the remaining signals are unchanged. However, standard filtering techniques generally achieve a variation in gain coupled with an undesired variation in phase, which affects signals transmitted through the filter.

In an ILC context, it is possible to choose the signal which will be filtered. Fundamentally, the choice is between the plant input and the output. Which of these is chosen should not matter (Longman, 2000), because the filter will always be situated within the control loop. Figure 4.22 illustrates this with a block diagram representation of an ILC based control system, indicating the four possible locations of the filter:

1. error signal (related to the plant output).
2. ILC generated signal (before storing in memory).
3. plant input.
4. measured displacement (plant output).

It is important to remember that no filter is perfect. The signal is only attenuated, not eliminated. Therefore, signals within the stop band of the filter will pass through, but should have very small amplitude. ILC, by design, has an outer control loop, but also an inner iteration loop. The iteration loop adds new data to the previous input signal.

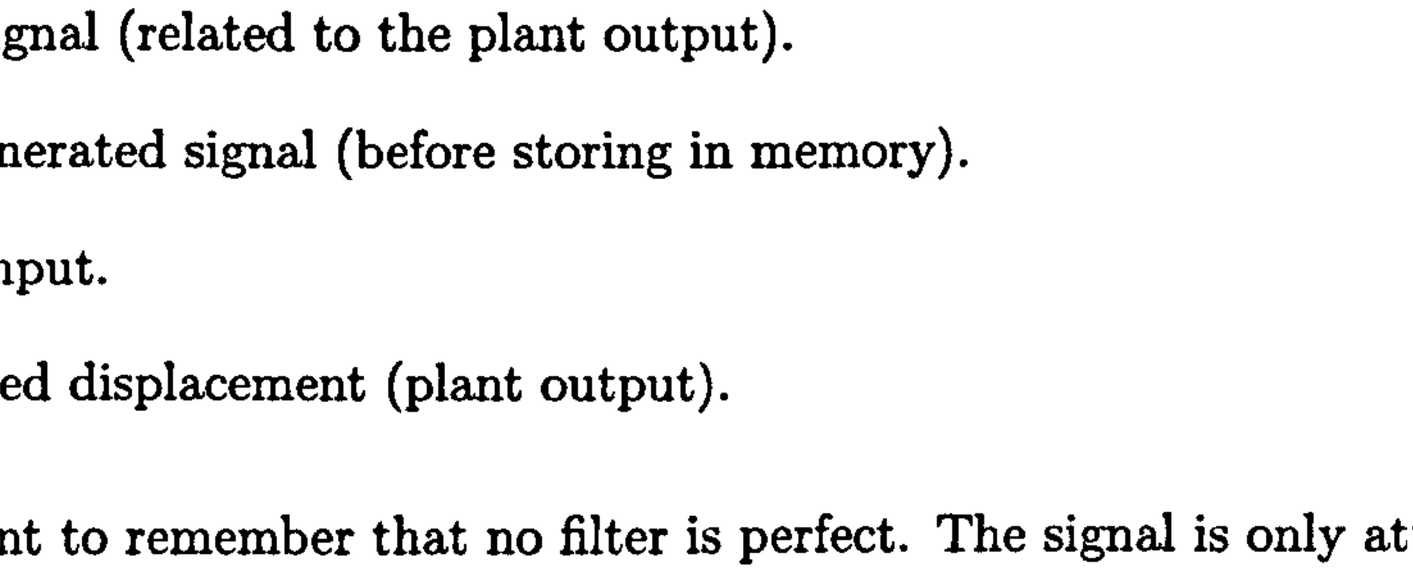


FIGURE 4.22: General ILC structure - possible locations of filter

At each trial, more data is added to this loop, and once in the loop, the data is locked

in until the system is shut down. While the majority of data is useful, unwanted signals are also stored and can build up over time, eventually resulting in instability. Filters applied in locations 1, 3 and 4 only have only one attempt at removing unwanted data before it enters the iteration loop. However, a filter designed at location 2 is within the ILC loop. Therefore, at each iteration unwanted data can actively be removed from the loop and does not build up. If the unwanted data is not fully removed on the first pass of the filter, it can be removed after successive iterations.

If a single frequency needs to be removed from a signal, a band-stop filter is most appropriate. Careful design of a digital, 2nd order, Chebychev, Band-stop filter results in the frequency response characteristics shown in the Bode plot of Figure 4.23. The design parameters provide 20 dB of attenuation between frequencies of 12 to 14 Hz (75-87 rad/s) for a 1kHz sample frequency. The choice of transfer function results in a phase plot, significantly shifted only around the region of attenuation. Above and below the attenuated frequencies, the phase shift asymptotically approaches 0. This filter is well matched to the 12 and 13 Hz resonant frequencies of the X-axis.

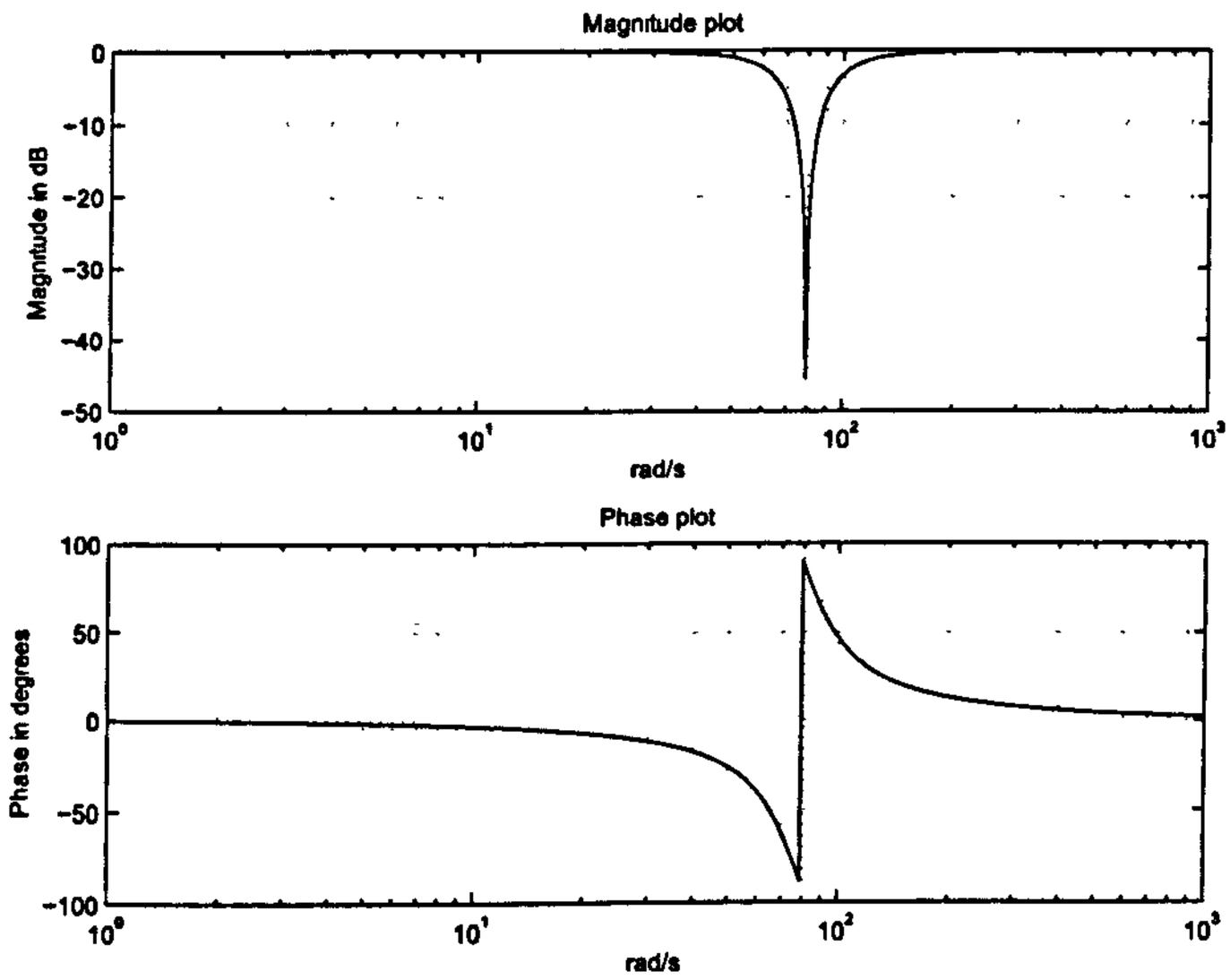


FIGURE 4.23: Band-stop filter Bode plot

Observing the mse plots, generated by implementing the band-stop filter (Figure 4.24), the onset of instability appears to have been prevented. The mse is improved beyond that achieved by the PID controller by 1 to 1.5 orders of magnitude, depending on the axis. After a small increase between iterations 20 and 40, it also appears to hold at a constant value. However, the tests were terminated at 200 iterations, as the robot started producing high levels of audible noise. The PI_{100} values for the hybrid controller with additional band-stop filtering are presented in Table 4.5.

Figures 4.25, 4.26 and 4.27 display the PID and ILC components of the plant input and the tracking error achieved at iteration 200. Clearly, the ILC input signals, particularly

TABLE 4.5: Hybrid controller with band-stop filtering, PI_{100} values

Gain	X-axis	Y-axis	Z-axis
100	5.600302	6.861328	19.487640

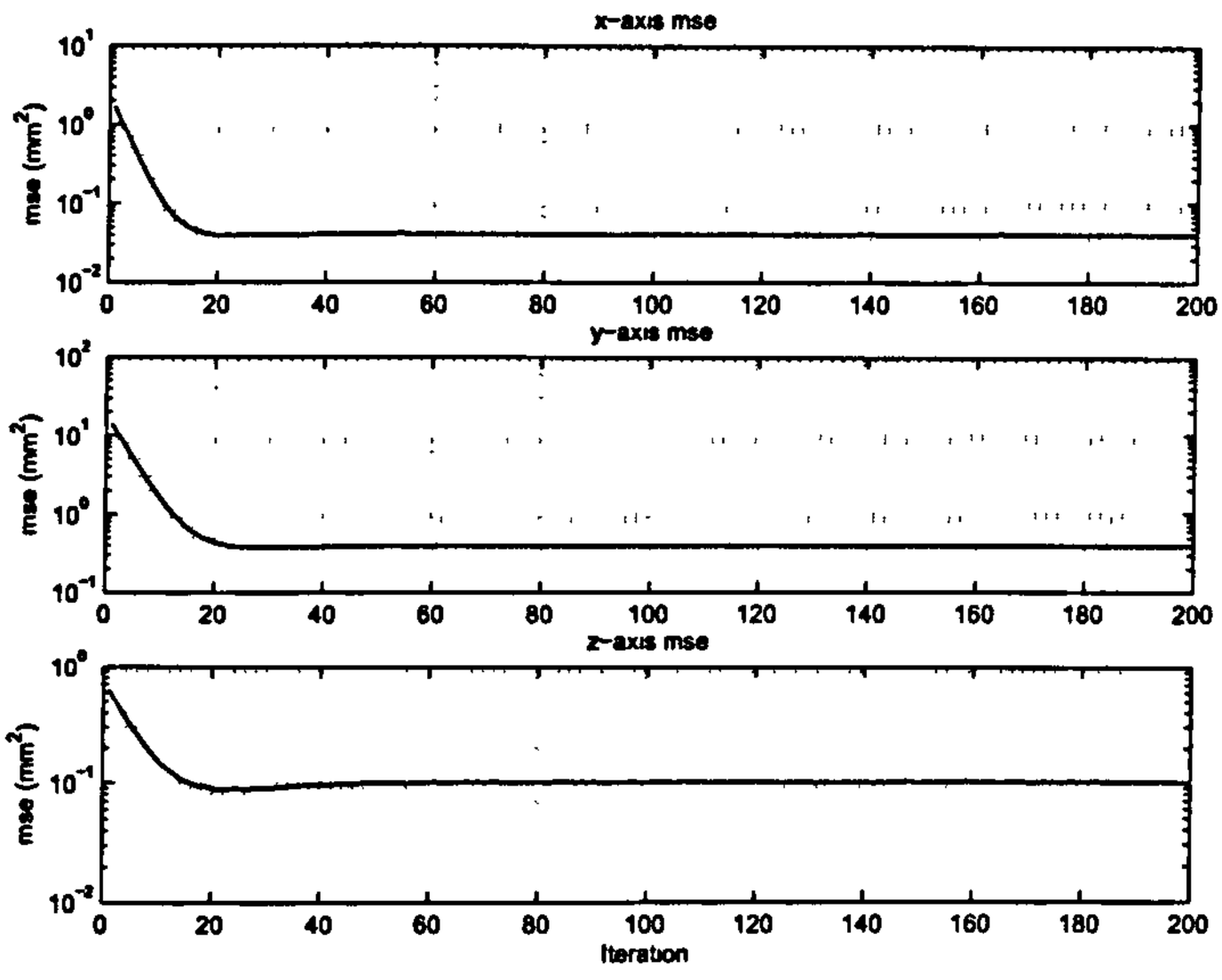


FIGURE 4.24: mse (band-stop filter, gain = 100)

for the Y-axis, still contain high frequency components.

Figure 4.28 confirms that the 12 and 13 Hz resonant frequencies have been successfully attenuated, and the main frequency component of the X-axis error is now directly related to the frequency of the reference trajectory. Truncating the spectrum below 3.5 Hz improves the resolution at higher frequencies and produces the spectrum in Figure 4.29. Frequency components in the range 35 to 45 Hz (219 to 282 rad/s) are particularly noticeable in the Y-axis error, and correspond well to the Y-axis resonant frequency. Although the addition of the band-stop filter has successfully removed the X-axis resonant frequency, the Y-axis resonance is now rendering the system unstable.

To investigate whether resonances are the sole cause of the vibration, the X-axis was operated alone with the 12-14 Hz band-stop filter. Figure 4.30 displays the X-axis inputs and error at iteration 300, when the test was terminated. By iteration 80 a grinding noise could be heard from the X-axis, which increased in intensity at every iteration. The ILC input signal in Figure 4.30 is heavily corrupted by high frequency signals, which are the likely source of this noise. Spectrum analysis of both the plant input and the tracking error, revealed a full range of frequency components up to the 500 Hz Nyquist frequency, with no single frequency particularly dominant.

Table 4.6 displays the calculated closed loop bandwidths for each axis. The plant increasingly attenuates frequencies higher than the bandwidth, therefore very high frequency

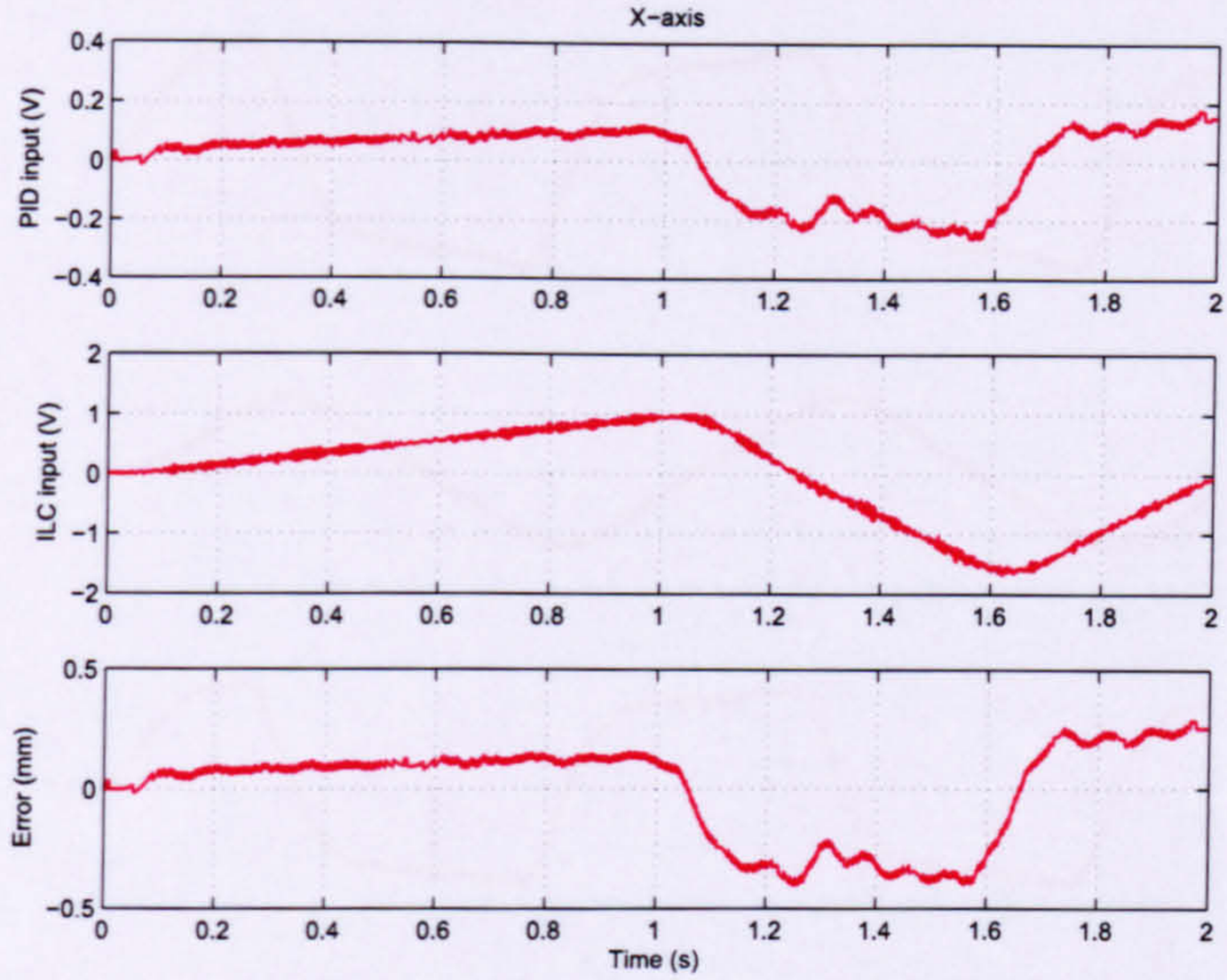


FIGURE 4.25: X-axis PID, ILC and tracking error (band-stop filter, gain = 100, iteration 200)

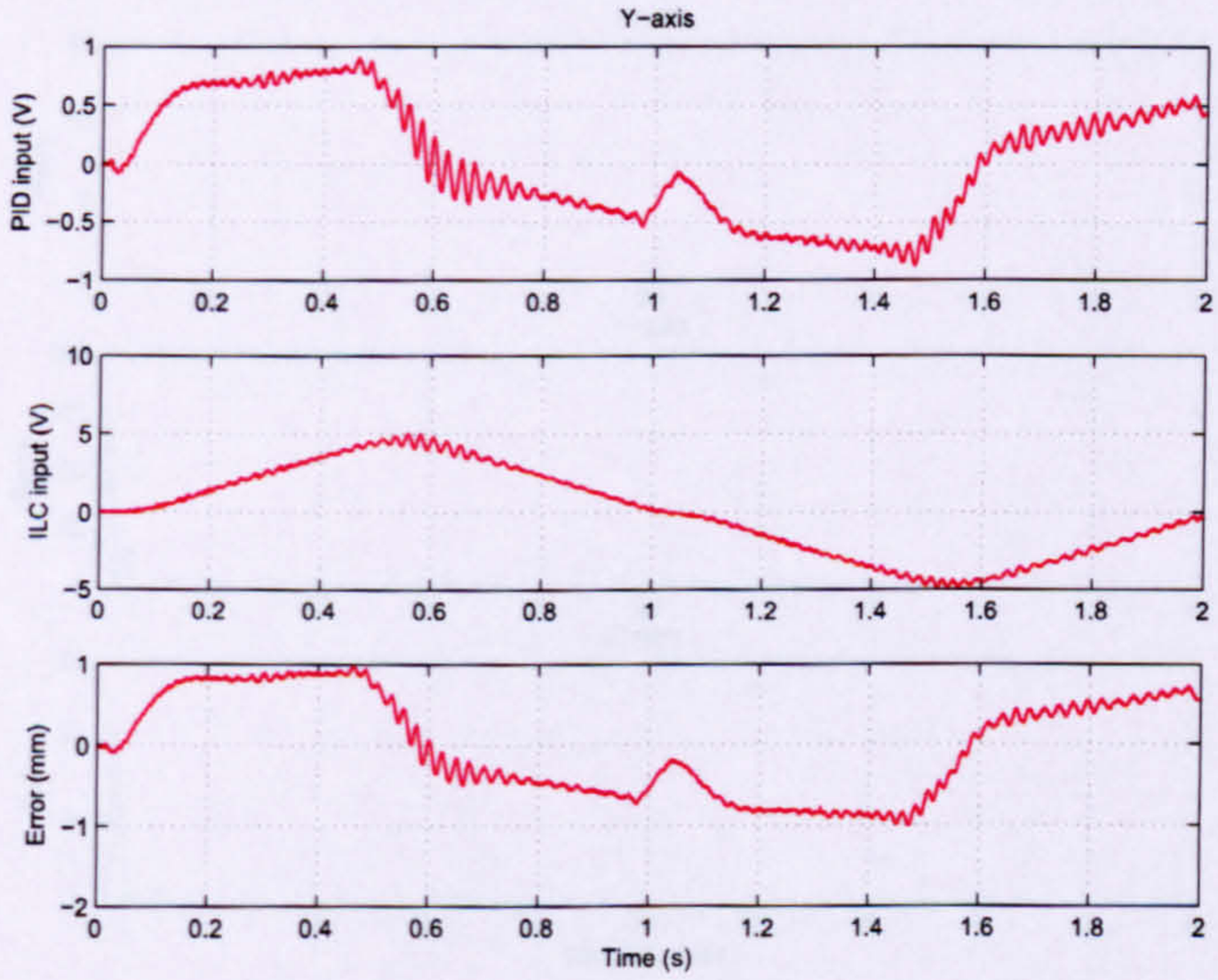


FIGURE 4.26: Y-axis PID, ILC and tracking error (band-stop filter, gain = 100, iteration 200)

components should be significantly attenuated at the plant output and should not be fed back into the controller. The likely source of these signals is measurement noise, caused by random disturbances, non-linear effects such as quantization and electromagnetic interference. Measurement noise cannot be totally eradicated from practical control systems. In ILC implementation it is a particularly important problem, because it can build up within the iteration loop. When low amplitude, high frequency noise, is added to noise already in the learning memory, some components will cancel out, while others

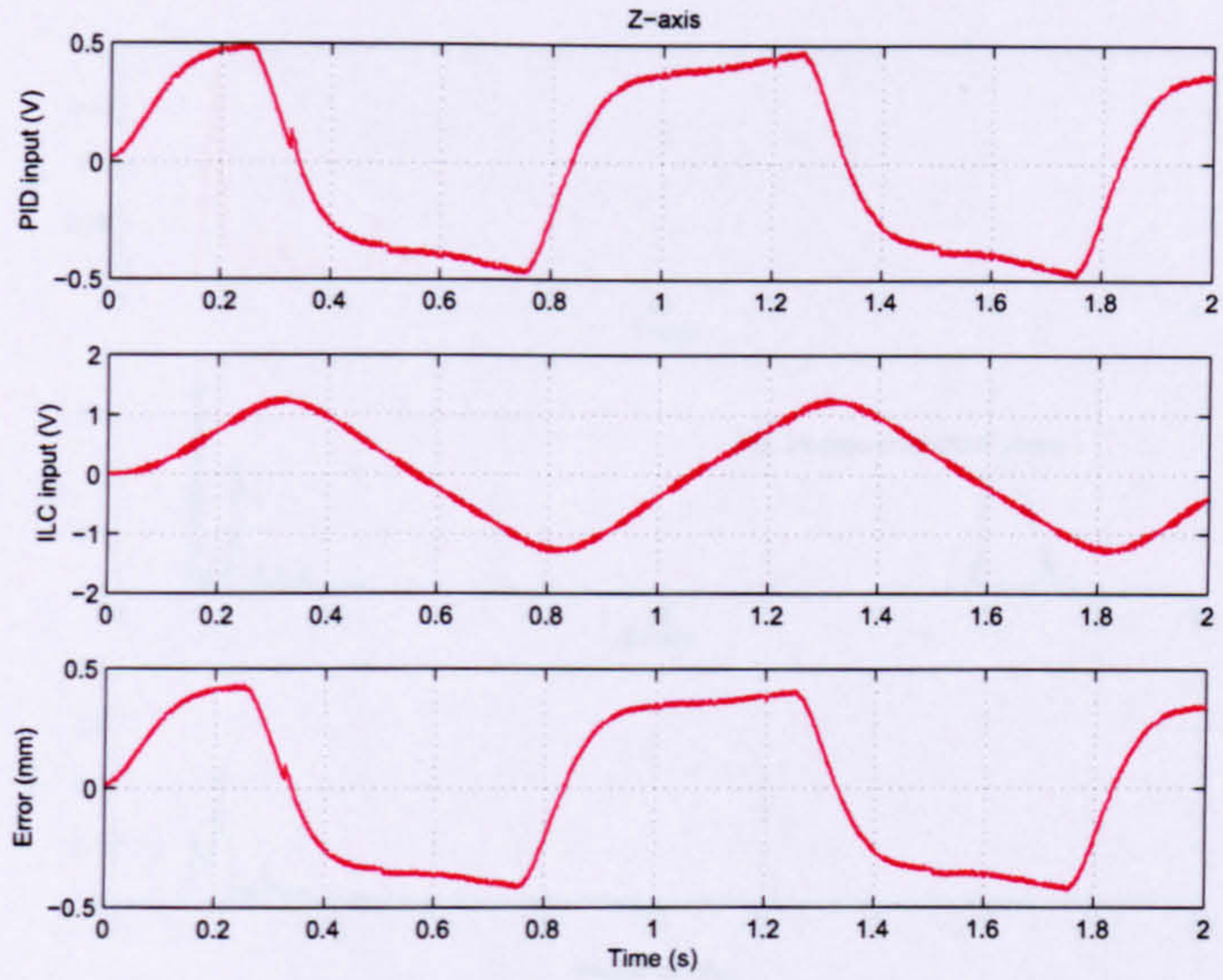


FIGURE 4.27: Z-axis PID, ILC and tracking error (band-stop filter, gain = 100, iteration 200)

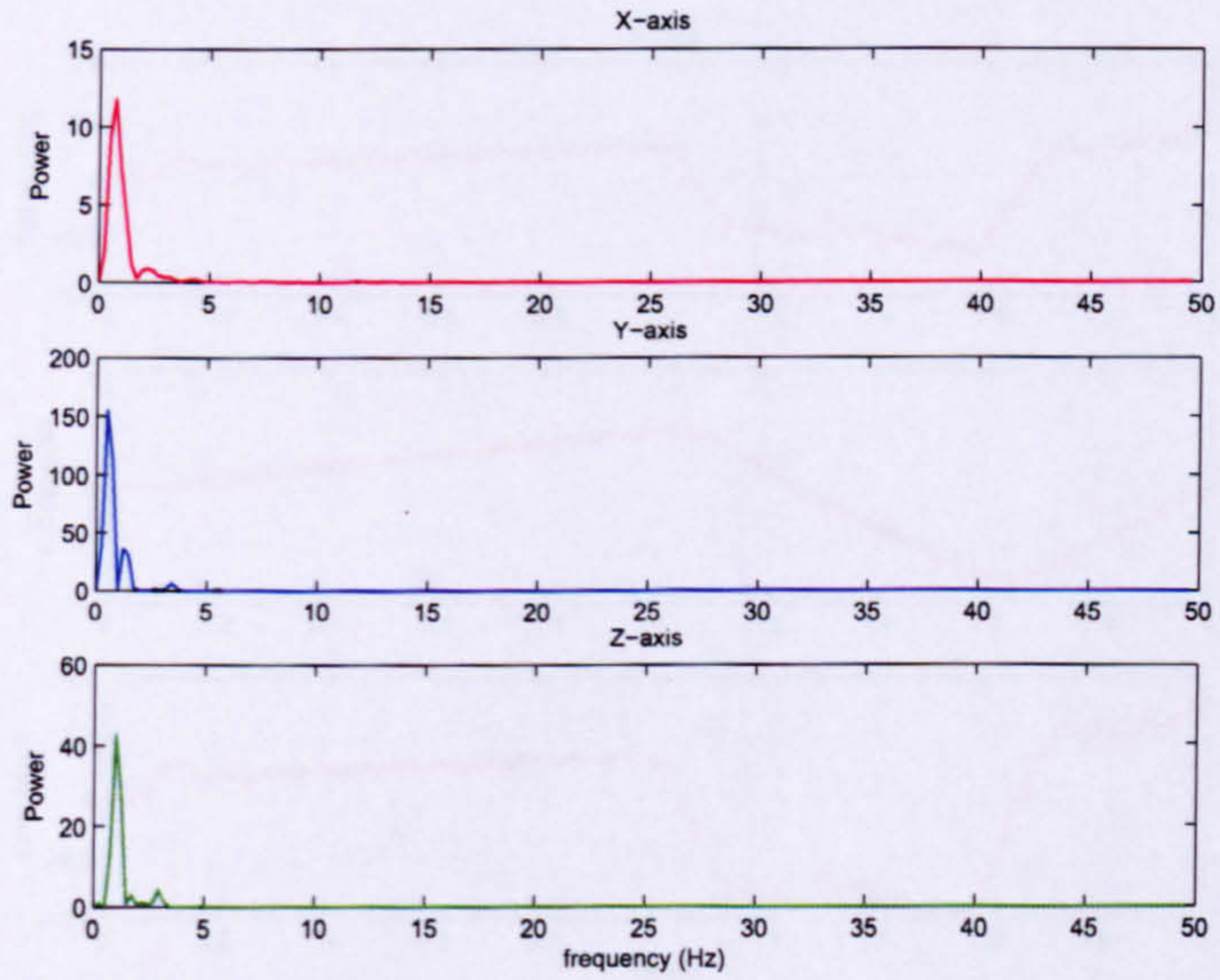


FIGURE 4.28: Error signal frequency spectra (band-stop filter, gain = 100, iteration 40)

will reinforce each other. With time, the amplitude of the noise grows and corrupts the low frequency signal which the ILC is compensating for. This characteristic is typical of the waterbed effect, proposed by Songchon and Longman (2001), which is based on Bode's integral theorem and is discussed in Section 2.3.2.3. Once the noise amplitude reaches a significant level, the plant is forced to respond to the noise signal as well as the main control input, which translates into mechanical noise as seen in the error signal.

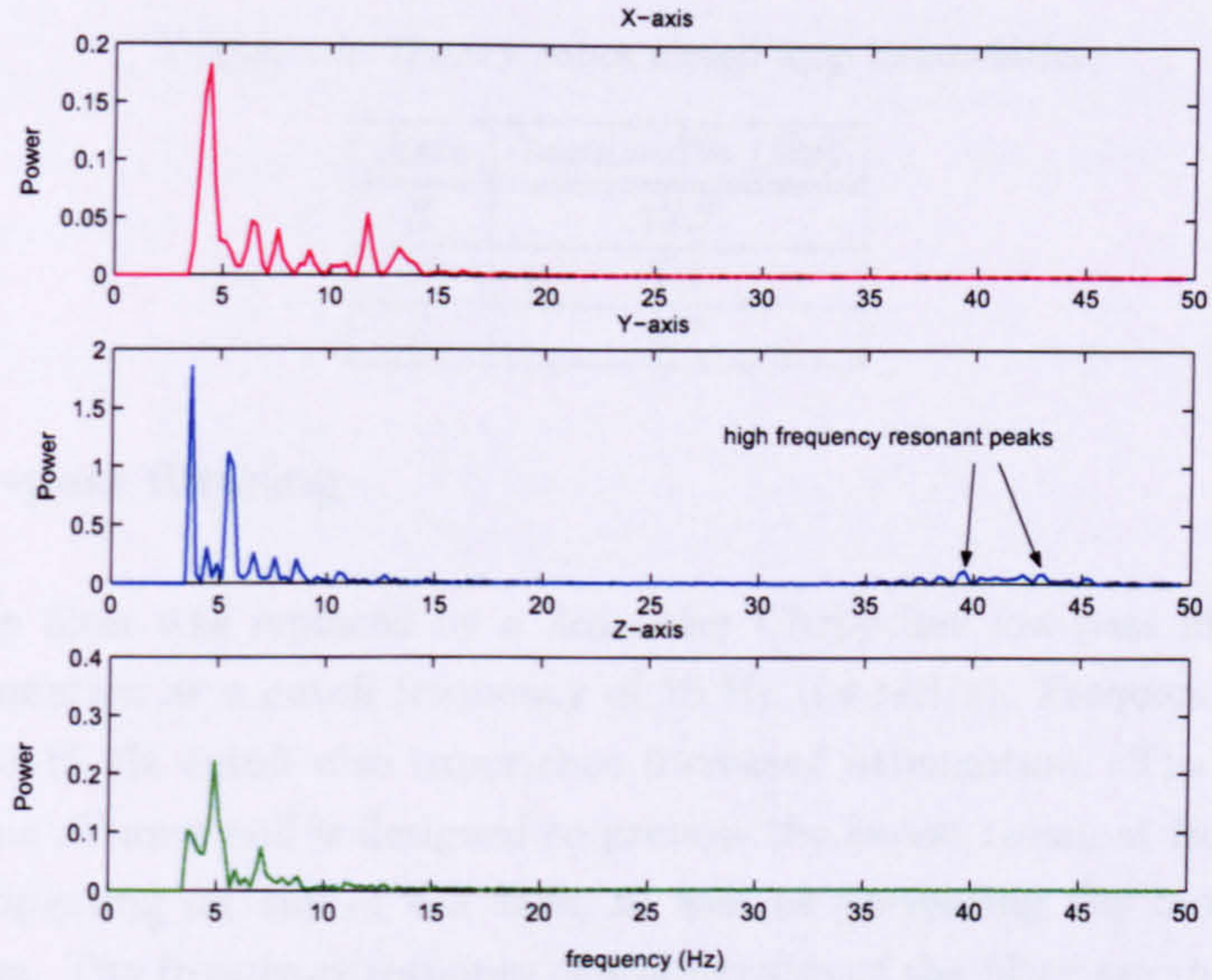


FIGURE 4.29: Error signal frequency spectra, truncated below 3.5 Hz (band-stop filter, gain = 100, iteration 40)

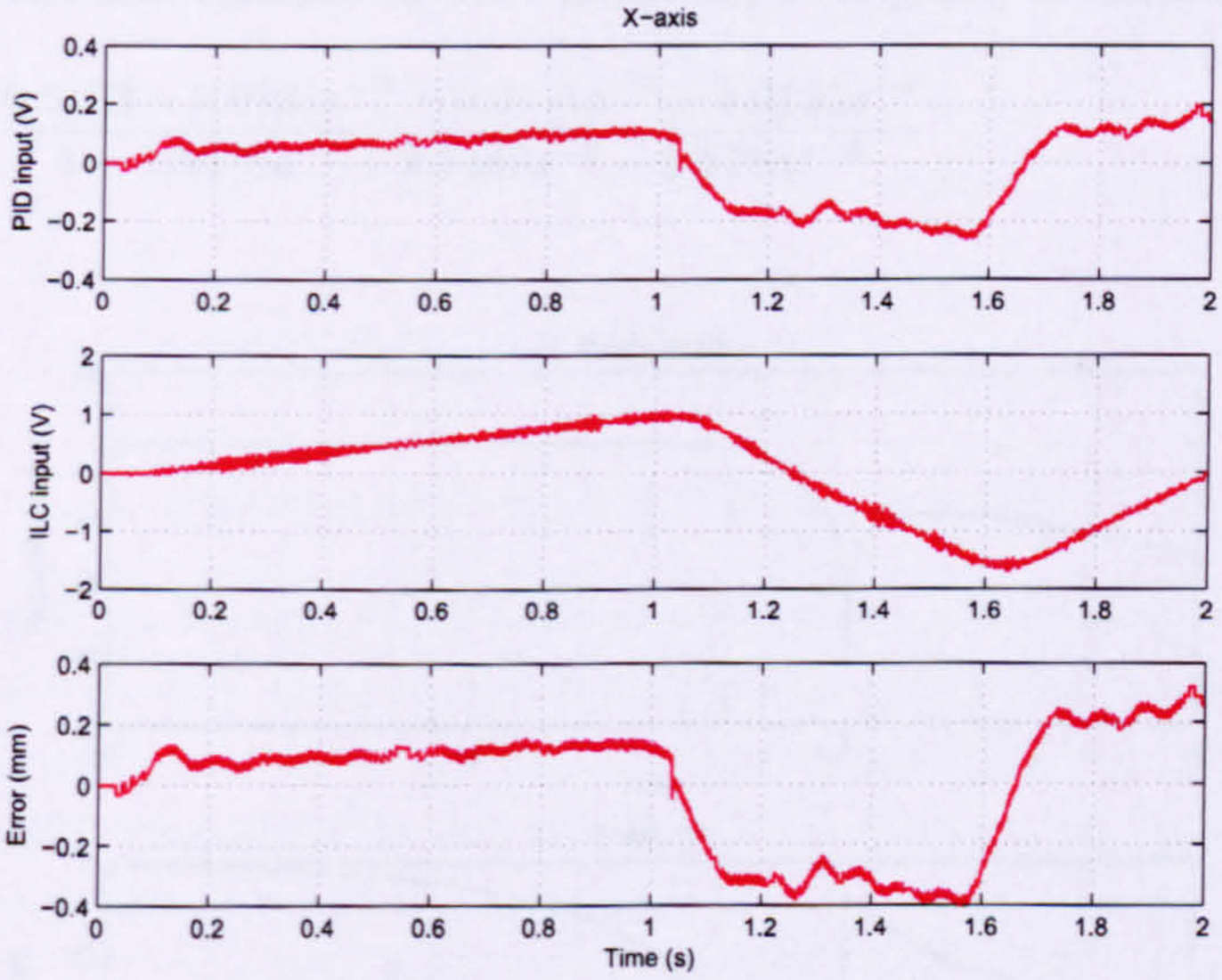


FIGURE 4.30: X-axis only, PID, ILC and tracking error (band-stop filter, gain = 100, iteration 300)

Implementing the band-stop filter has demonstrated an improvement in long term stability, by preventing the buildup of resonant frequencies. However, it has highlighted that measurement noise has an equally negative effect on controller performance. The next development therefore concerns the removal of measurement noise.

TABLE 4.6: Gantry robot closed loop bandwidths

Axis	bandwidth (Hz)
X	12.7
Y	7.1
Z	4.7

4.4.5 Low-pass filtering

The band-stop filter was replaced by a 3rd order Chebychev low-pass filter, providing 20 dB of attenuation at a cutoff frequency of 15 Hz (94 rad/s). Frequencies above and just below the 15 Hz cutoff also experience increased attenuation. The same filter is implemented on all axes and is designed to prevent the lowest resonant frequency of the robot from appearing on any of the axes, as well as preventing the build up of high frequency noise. The frequency response characteristics of the filter are shown in Figure 4.31, where the phase response characteristics are not as desirable as for the band-stop filter, because below the cut off frequency, the phase shift still is significant at 1 Hz. The filter transfer function matched to the 1 KHz sample frequency is defined by:

$$\frac{y(z)}{u(z)} = \frac{0.0133 - 0.0131z^{-1} - 0.0131z^{-2} + 0.0133z^{-3}}{1 - 2.8678z^{-1} + 2.7440z^{-2} - 0.8758z^{-3}} \tag{4.15}$$

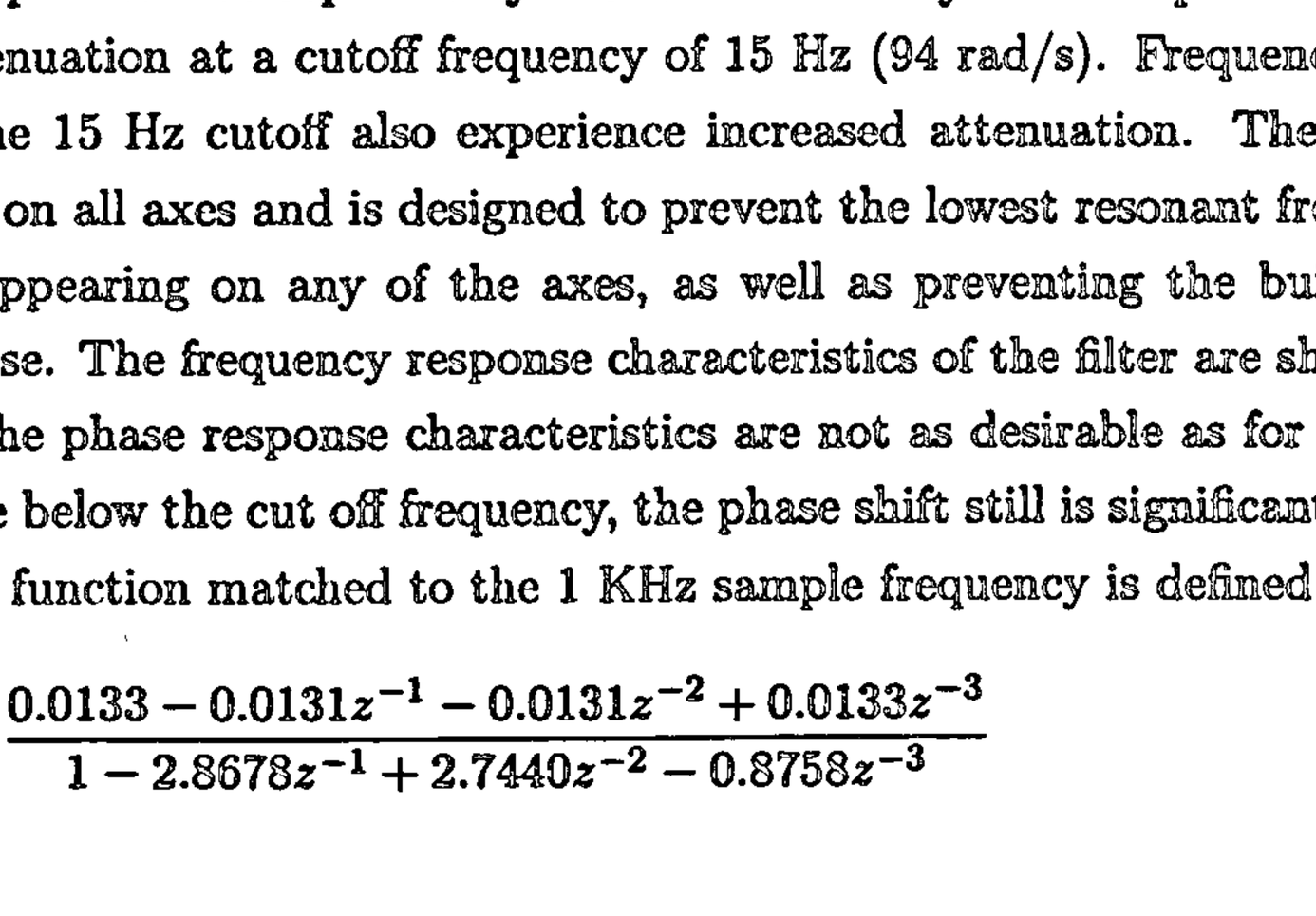


FIGURE 4.31: Low-pass filter Bode plot

Figure 4.32 displays the mse performance for the long-term stability test of 5000 iterations, using the parallel configuration PID-ILC hybrid controller and a learning gain of 100. Long-term stability has been achieved, and the 5000 iterations are successfully

completed without any need for interruption. Additionally, there is no sign of increasing mse and the ILC component of the input in Figure 4.33 shows no indication of resonance, oscillation or noise (see Appendix B for Y and Z-axis data). Table 4.7 presents the PI_{100} values for the hybrid controller with low-pass filtering.

TABLE 4.7: Hybrid controller with low-pass filtering, PI_{100} values

Gain	X-axis	Y-axis	Z-axis
100	24.270728	25.672166	78.473221

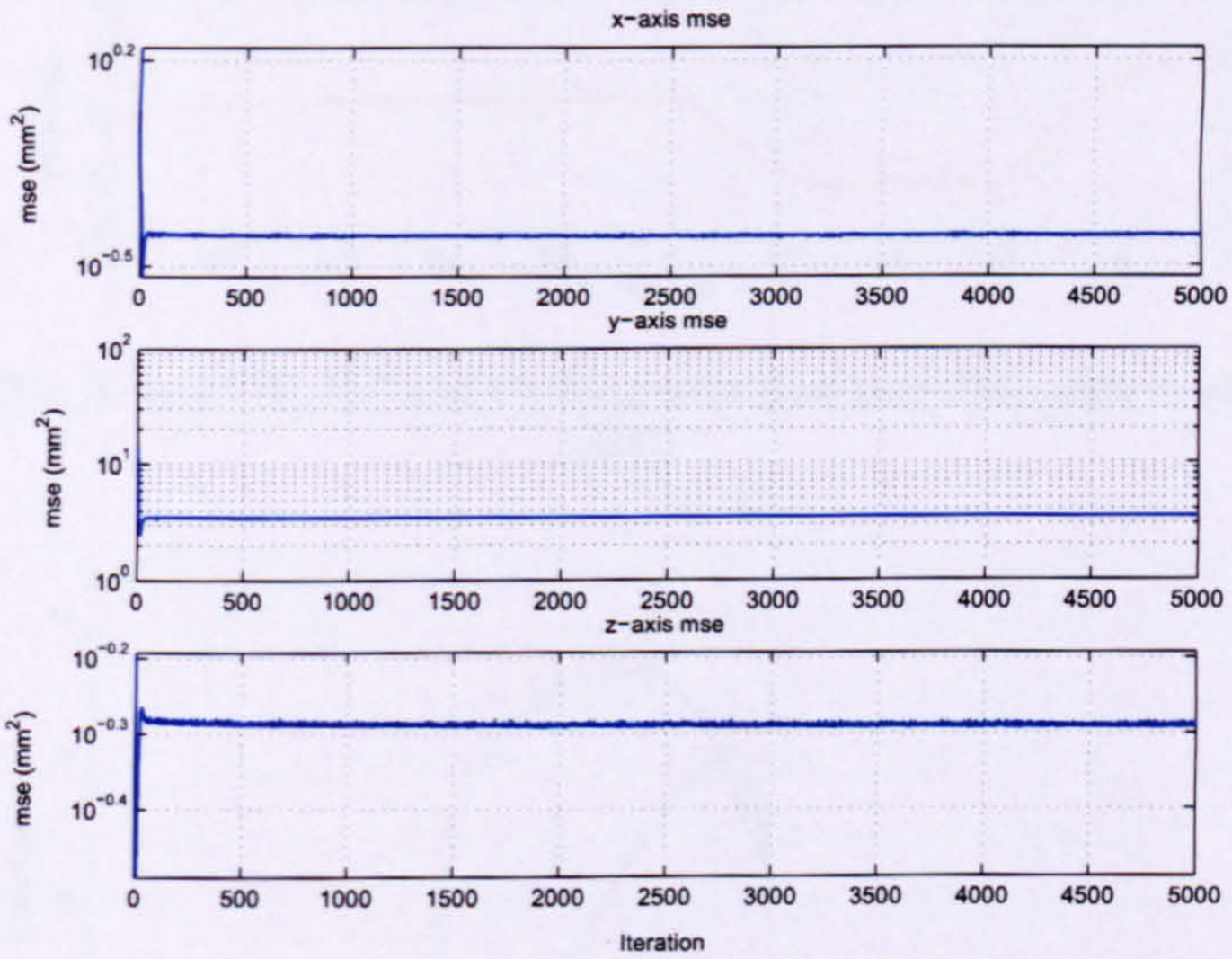


FIGURE 4.32: mse (low-pass filter, gain = 100)

Low-pass filtering is a possible viable solution for producing an industrial variant of the P-type learning controller. However, careful analysis of the vertical axis in the mse plots in Figure 4.32 reveals that all three axes have gained less than one order of magnitude mse reduction, whereas in the filter-less implementation the error was reduced by at least 2 orders of magnitude.

Figure 4.34 showing the 3-Dimensional tracking performance at iteration 5000 confirms the poor overall performance. There is significant error between the reference and the output, which resembles the error produced by the PID controller alone. The error is produced by phase lag introduced into the control loop as a result of low-pass filtering. Phase lag is added at frequencies which the ILC is required to learn, therefore the learning controller is unable to compensate accurately for the error measured at each sample instant, because the error occurs later in the iteration than expected.

This observation confirms the proposal of Elci, Longman, Phan, Juang, and Ugoletti (2002) which concludes that whenever the filter phase is non-zero below the cut-off frequency, a forcing function will produce non-zero error after convergence.

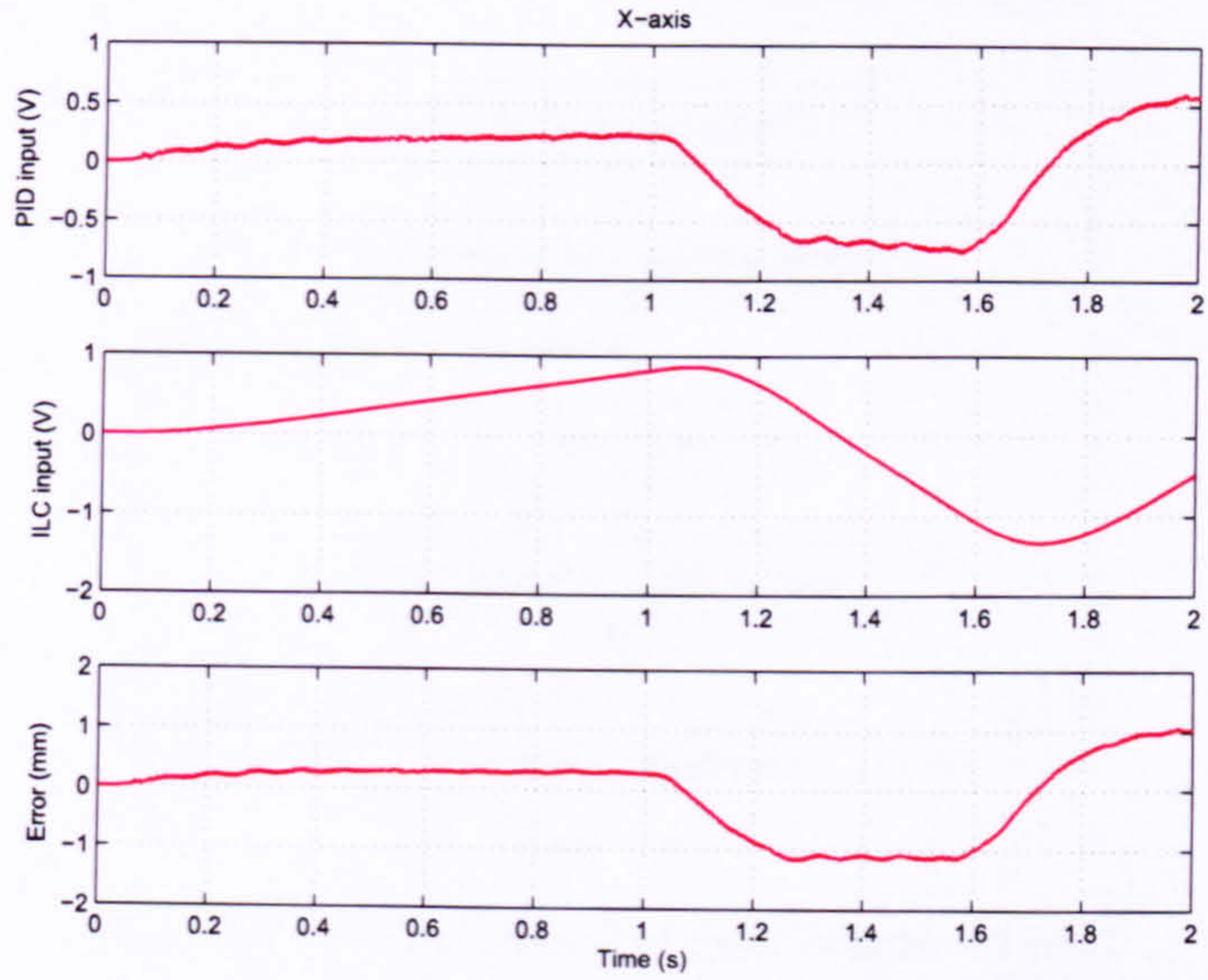


FIGURE 4.33: X-axis PID, ILC and tracking error (low-pass filter, gain = 100, iteration 5000)

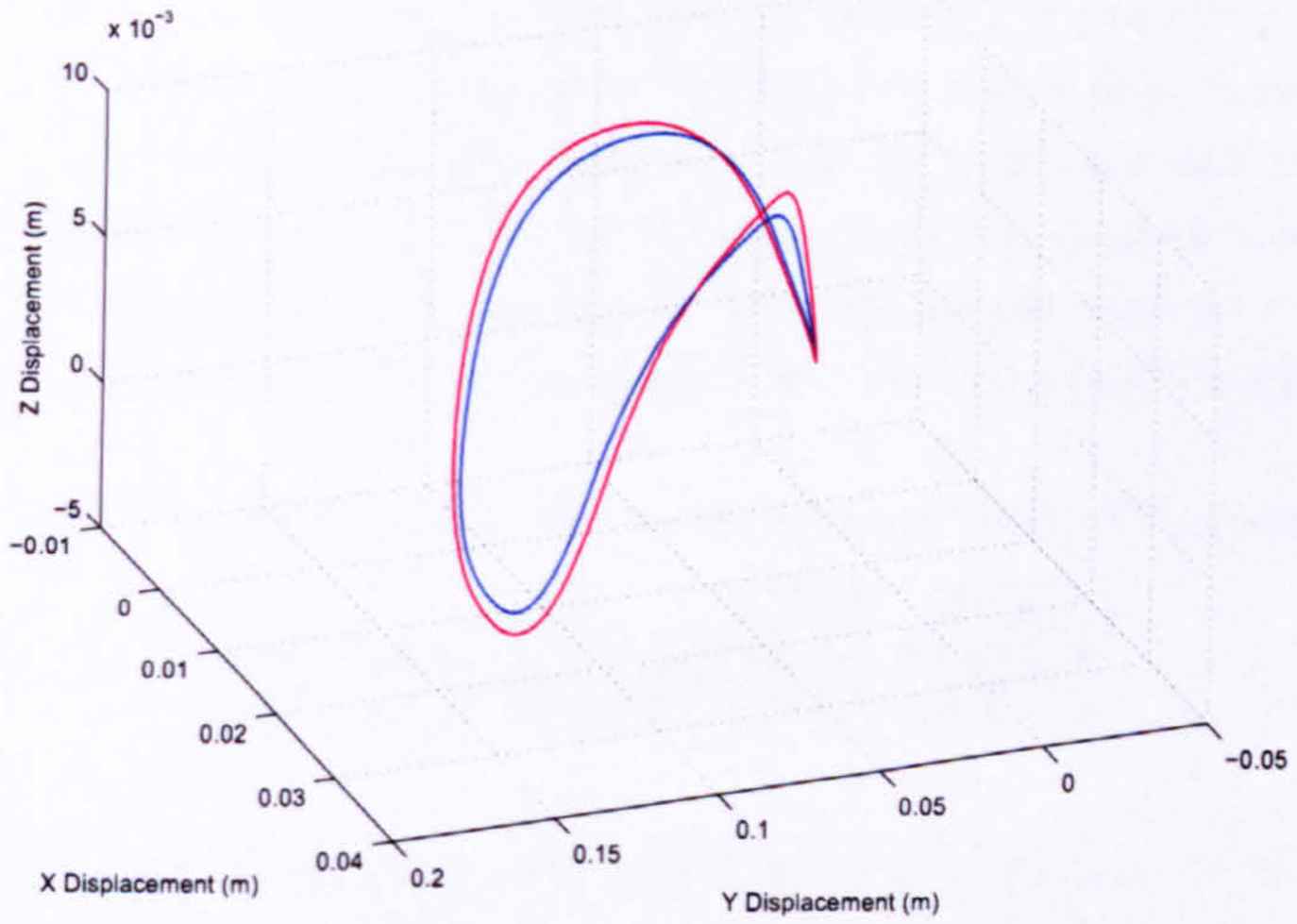


FIGURE 4.34: 3-Dimensional tracking performance at iteration 5000 (low-pass filter, gain = 100, blue = reference)

4.4.6 Zero-phase filtering

The technique required to overcome low frequency phase shift is zero-phase filtering (Markusson et al., 2001; Wang and Longman, 1996). Zero-phase filtering is difficult to achieve in continuous control environments, because it is inherently a non-causal procedure, requiring data from future, in addition to past sample instants. However, it is much simpler to implement within the ILC framework, where data is intrinsically processed in batches. The required sequence of events is presented in the flowchart,

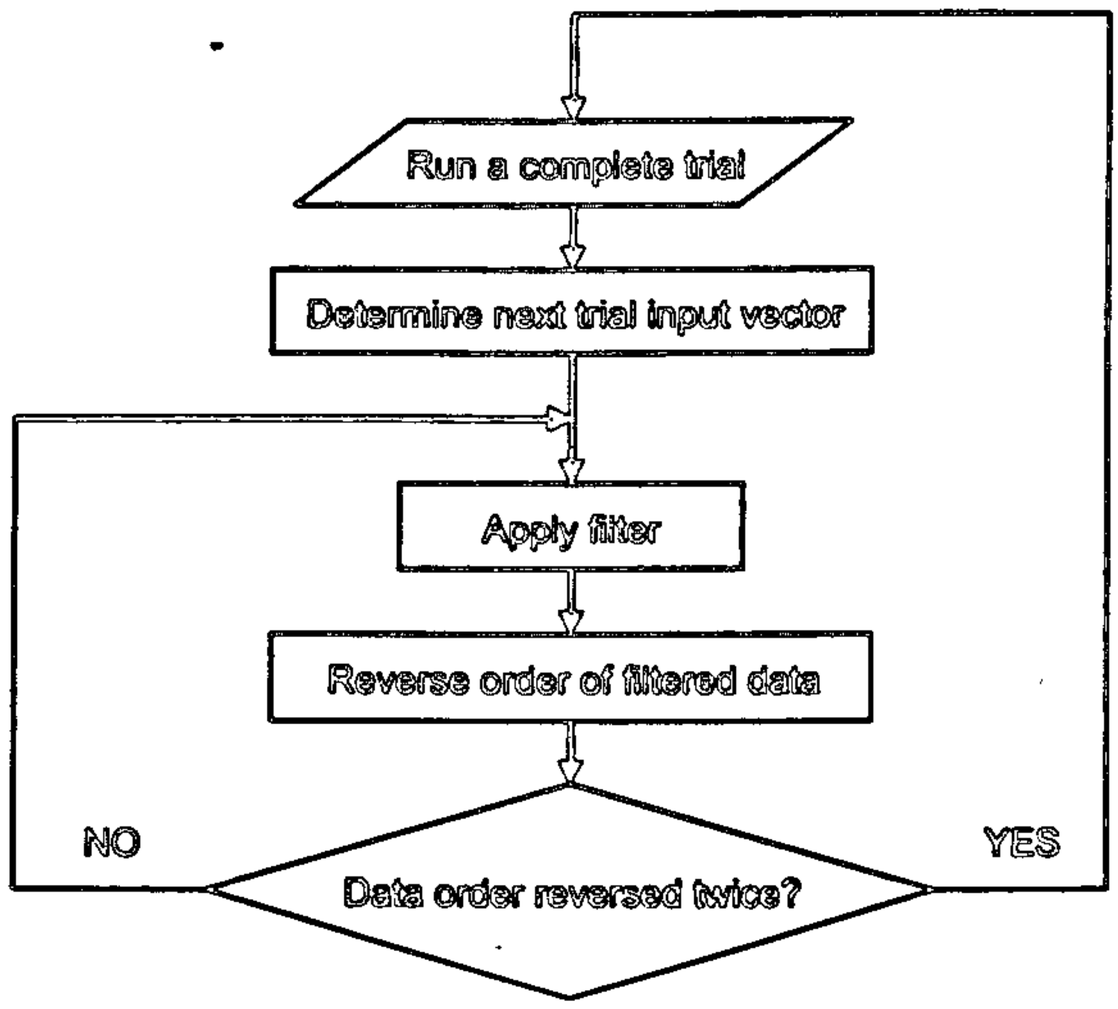


FIGURE 4.35: Zero-phase filtering technique, flowchart

Figure 4.35. On the first pass of the filter, the frequencies in the stop-band are attenuated and phase shift is added to the vector. When the filtered data is reversed and filtered for the second time, the stop-band frequencies are attenuated again and the same phase shift is added, in reverse time, thus cancelling out the phase shift from the first pass. The result is a filter with double the attenuation in the stop-band and zero phase shift. Care must be taken to ensure that the initial samples of the filtered vector are suitable for the plant. In this example, the voltage for the first sample should be zero. Table 4.8 presents the PI_{100} values for the hybrid controller with zero-phase filtering.

TABLE 4.8: Hybrid controller with zero-phase filtering, PI_{100} values

Gain	X-axis	Y-axis	Z-axis
100	4.074121	5.475265	7.926918

The low-pass filter described in Section 4.4.5 has been implemented on all three axes using the zero-phase protocol. Figure 4.36 shows the mse tracking performance for 5000 iterations with learning gain 100. The system is long-term stable as expected, and the mse has now been reduced by almost 2 orders of magnitude for each axis. The system has produced a performance which compares well with the unfiltered controller.

Figure 4.37 shows the overall improvement in reference tracking, where the error between the reference and the measured output is minimal. The graphical resolution used in the 3-dimensional plot results in an output signal, which merges into the reference signal.

Figure 4.38 displays the input and error waveforms obtained at iteration 5000 for the X-axis: the tracking error and PID input still show signs of resonance, as they contain 12 Hz frequency components (corresponding data for the Y and Z-axes can be found in Appendix B). These are small amplitude oscillations, excited by the high gain of

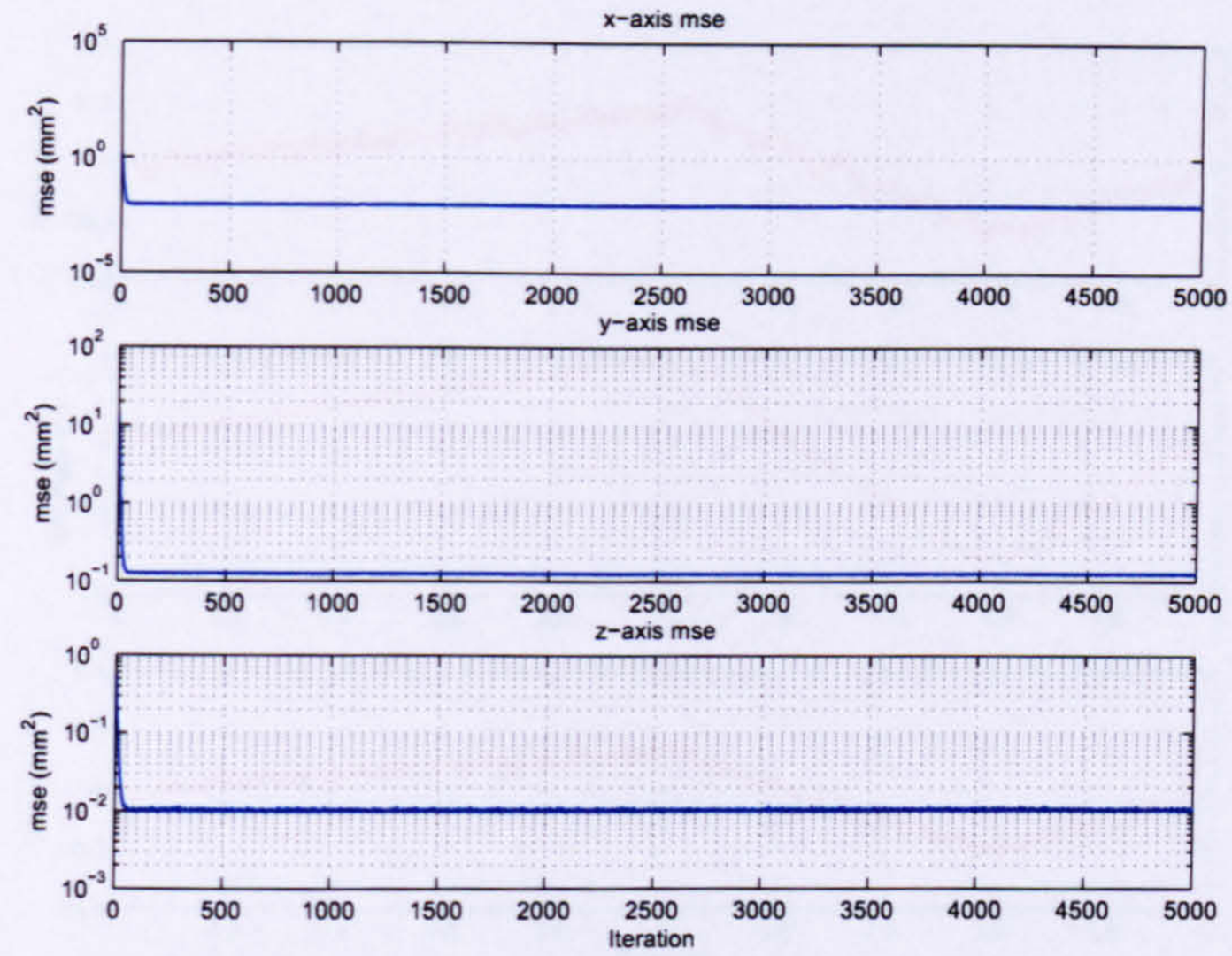


FIGURE 4.36: mse (zero-phase filter, gain = 100)

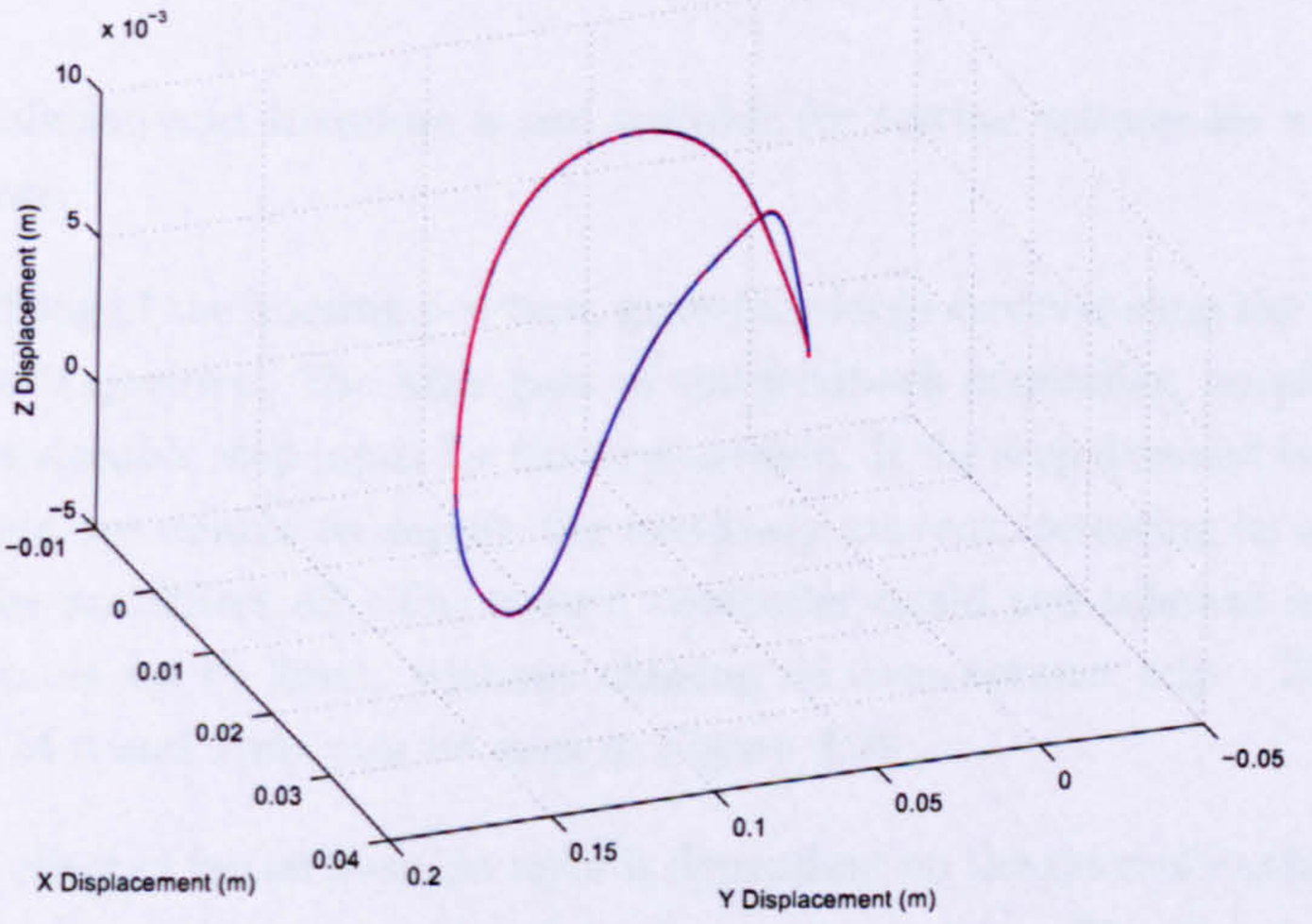


FIGURE 4.37: 3-Dimensional tracking performance at iteration 5000 (zero-phase filter, gain = 100, blue = reference)

the feedback controller, whereas the ILC input is a very smooth waveform, completely devoid of high frequency components. Therefore, while the feedback controller may excite resonance, the zero-phase filter ensures that it does not build up within the iteration loop and the amplitude does not increase.

4.4.6.1 Initial state error tolerance

Having achieved a long-term stable learning controller, it is now possible to investigate robustness to disturbance. The basic P-type algorithm does not use a model of the plant

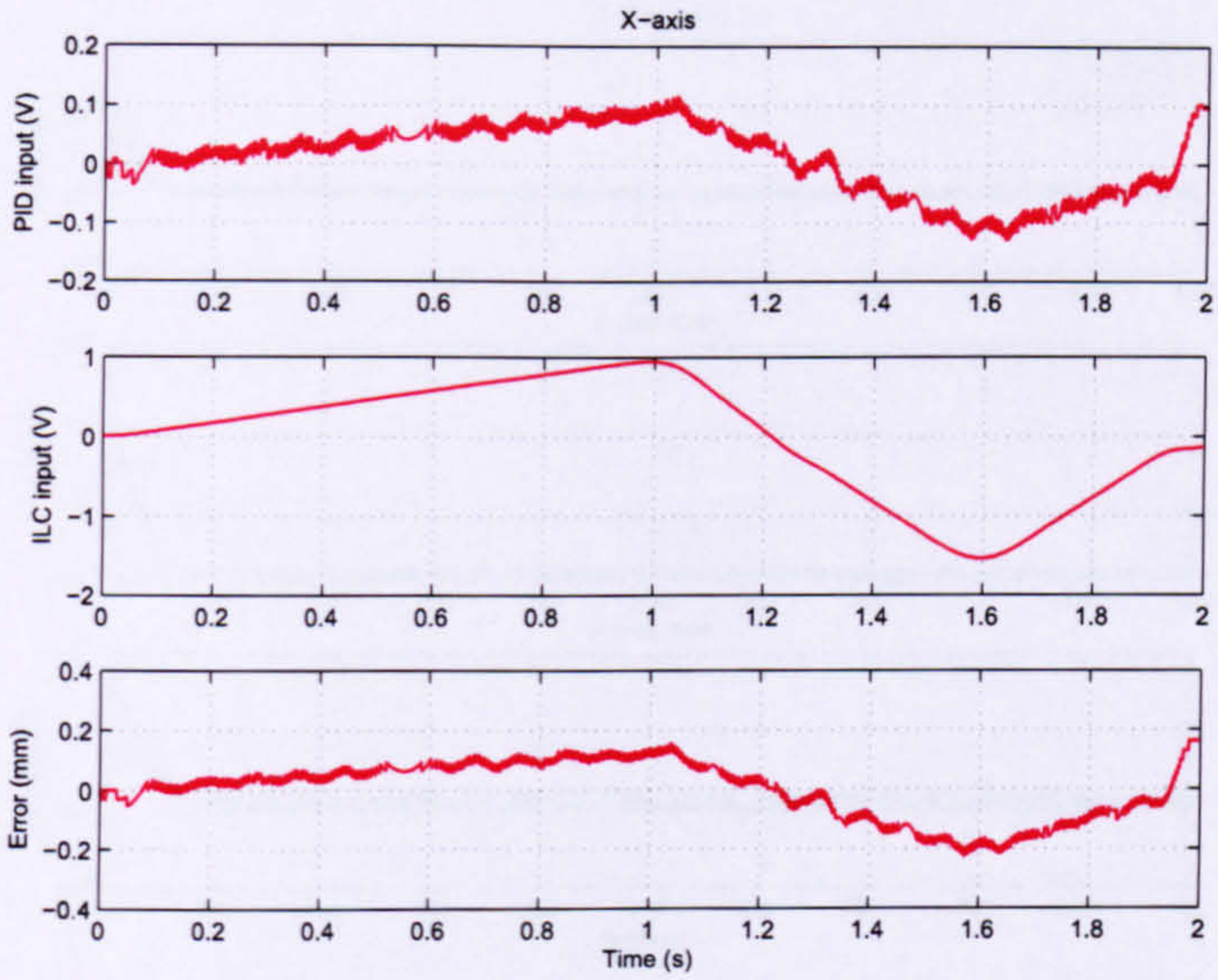


FIGURE 4.38: X-axis PID, ILC and tracking error (zero-phase filter, gain = 100, iteration 5000)

for input calculation and therefore is not suitable for testing robustness with respect to modelling errors.

Deliberate shifting of the homing position, generates large errors during the first sampling instants of the trajectory. The high gain of the feedback controller, amplifies the error and produces a sizeable step input for the first sample. If the step demand is too large, the power amplifiers are unable to supply the necessary current, resulting in a current trip, which shuts the amplifiers off. The hybrid controller could not tolerate initial position error with bounds up to 2mm, without causing an over-current trip. The results for initial bounds of 0 and 1mm can be seen in Figure 4.39.

The measured effect of initial position error is dependent on the ratio of initial error bound to the total displacement required by the reference trajectory. The Z-axis performance is most affected by small initial error, because its total travel range is 10mm. 1mm of error represents 10% of the trajectory as compared to the X and Y-axes, where it corresponds to 2.7% and 0.7% respectively. Initial error has a direct effect on the steady state tracking error level, and causes increased variation in the mse, but the stability and initial convergence rate are not compromised. Table 4.9 displays the PI_{100} values for initial error bounds 0 and 1mm.

TABLE 4.9: Zero-phase filtering with initial error, Gain = 100, PI_{100} values

Error bound (±mm)	X-axis	Y-axis	Z-axis
0	4.074121	5.475265	7.926918
1	4.240410	5.459859	8.356485

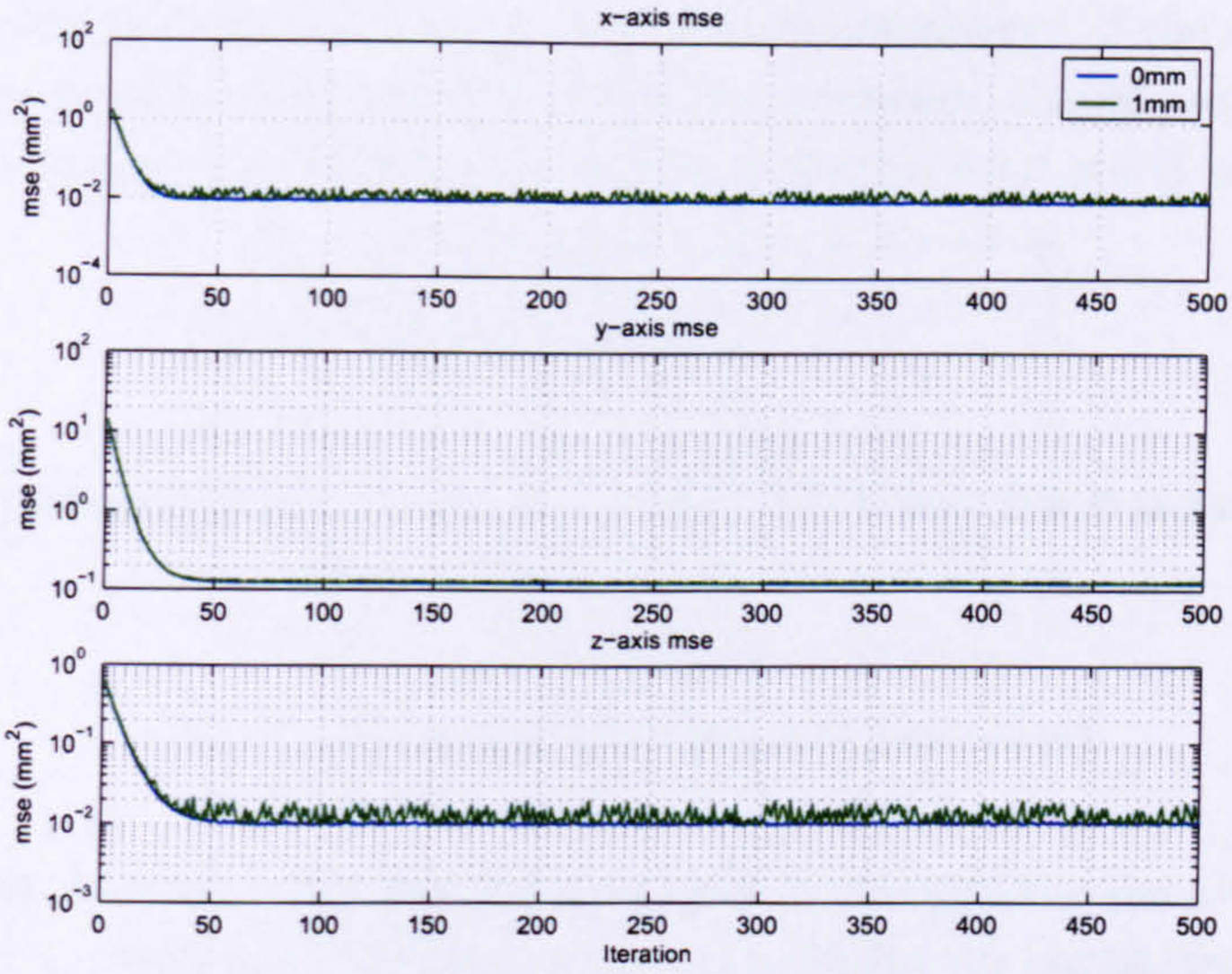


FIGURE 4.39: mse (zero-phase filter, initial error bounds 0 and 1 mm)

4.4.7 Frequency aliasing

The zero-phase filtering technique has proved very successful in achieving long-term stability and good levels of error reduction by removing high frequency noise and suppressing natural resonance. An alternative approach is proposed, based on Shannon's sampling theorem, to alias high frequency tracking error information and produce a low-pass filter with perfect cut off and no induced phase shift.

Shannon's sampling theorem states that "if a function $f(t)$ contains no frequencies higher than W Hz, it is completely determined by giving its ordinates at a series of points spaced $1/2W$ seconds apart" (Shannon, 1998). Effectively, to record a signal of a given frequency in a discrete-time system, the sample frequency must be at least twice the frequency of the signal. If this requirement is not met, then frequency aliasing occurs and the higher frequency components of the original signal are lost.

In most applications, aliasing is a drawback, usually redressed by means of anti-aliasing filters. In this application, aliasing is beneficially used to remove the effects of the undesired frequencies from the learning controller. The controller configuration has exactly the same components as the original parallel system. However, there is an additional aliasing module attached to the learning controller in place of the filters used in previous sections (Figure 4.40). The aliasing module samples the output generated by the learning controller at a frequency less than twice the resonant frequency of the plant. In this way, the resonant frequency is aliased to a lower frequency at which the plant does not resonate and the aliasing filter allows the ILC to learn frequencies below the alias cut-off. As long as the resonant frequency is greater than the frequencies which need to be learnt to satisfy the reference demand, the loss of data caused by aliasing does not

prevent the learning controller from improving the performance of the overall system. High frequency signals, which are aliased to lower frequency, should not have an effect on stability, as long as the convergence criterion in Section 4.4.3 is still satisfied.

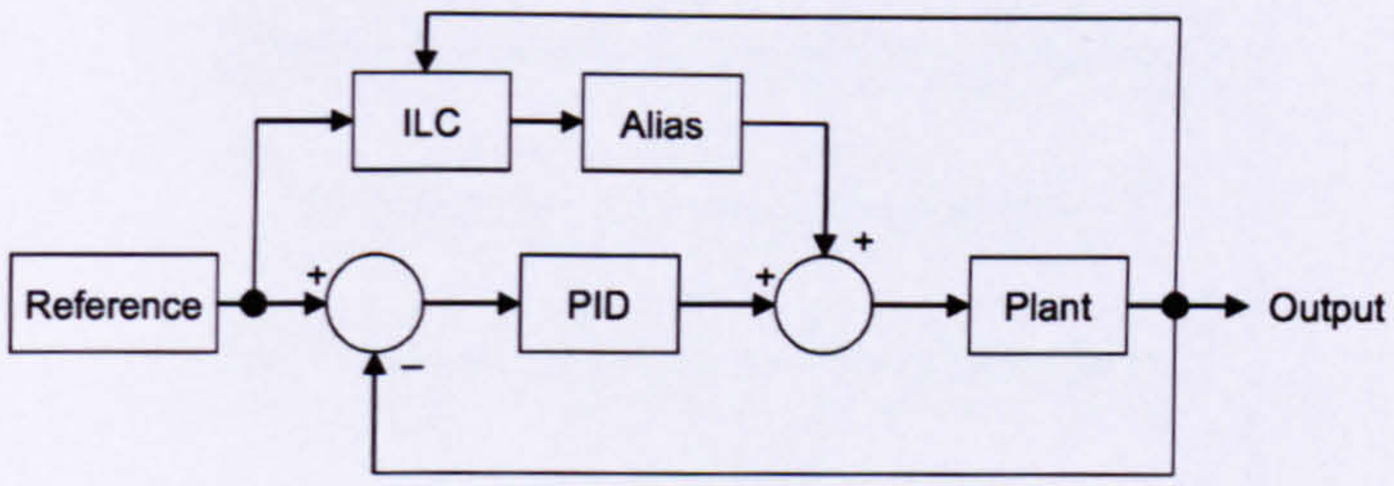


FIGURE 4.40: Parallel controller with alias module

In the following analysis, 'alias frequency' is used to describe the sample frequency of the aliasing filter. 'Feedback frequency' is used to describe the sample frequency of the remaining control system (1kHz). 'Alias gap' is used to relate the alias frequency to the feedback frequency. The alias gap is the number of feedback frequency sample instants between each alias frequency sample instant.

Using a zero-order-hold approach for the aliasing module is not suitable, as the aliasing filter generates a signal, which relates poorly to the original non-aliased signal, due to large step changes in the control voltage. However, because the ILC approach obtains all of the data from the previous trial before generating the input for the next trial, it is possible to use a non-causal approach to generate a much smoother output from the aliasing module.

In a non-causal approach, instead of holding the signal value constant between aliasing samples, it is possible to calculate the gradient between adjacent samples and join them by linear interpolation, a technique which is commonly used in practical signal processing, e.g. Unser (2000). In a parallel feedback/ILC controller arrangement the aliasing process for the ILC component is defined by the flowchart in Figure 4.41. Following the successful completion of an iteration, the ILC algorithm computes the next ILC component of the plant input vector at feedback frequency. This input vector is likely to contain unwanted frequencies. The aliasing module then re-samples the input vector at the aliasing frequency, removing all frequencies above the aliasing cut-off. The data between aliasing frequency sample instants is deliberately lost, so as to remove high frequency components. The input vector now consists of far fewer sample instants than are required for real-time operation. Linear interpolation is used to connect the aliasing frequency sample instants, so that a smooth signal is produced when the interpolated signal is re-sampled at feedback frequency. During the next iteration, each sample instant of the aliased ILC vector is summed with the input produced by the feedback controller.

The linear interpolation method reconstructs a smooth approximation of the aliased ILC signal. Figure 4.42 demonstrates the principle, when applied to a sine-wave of frequency

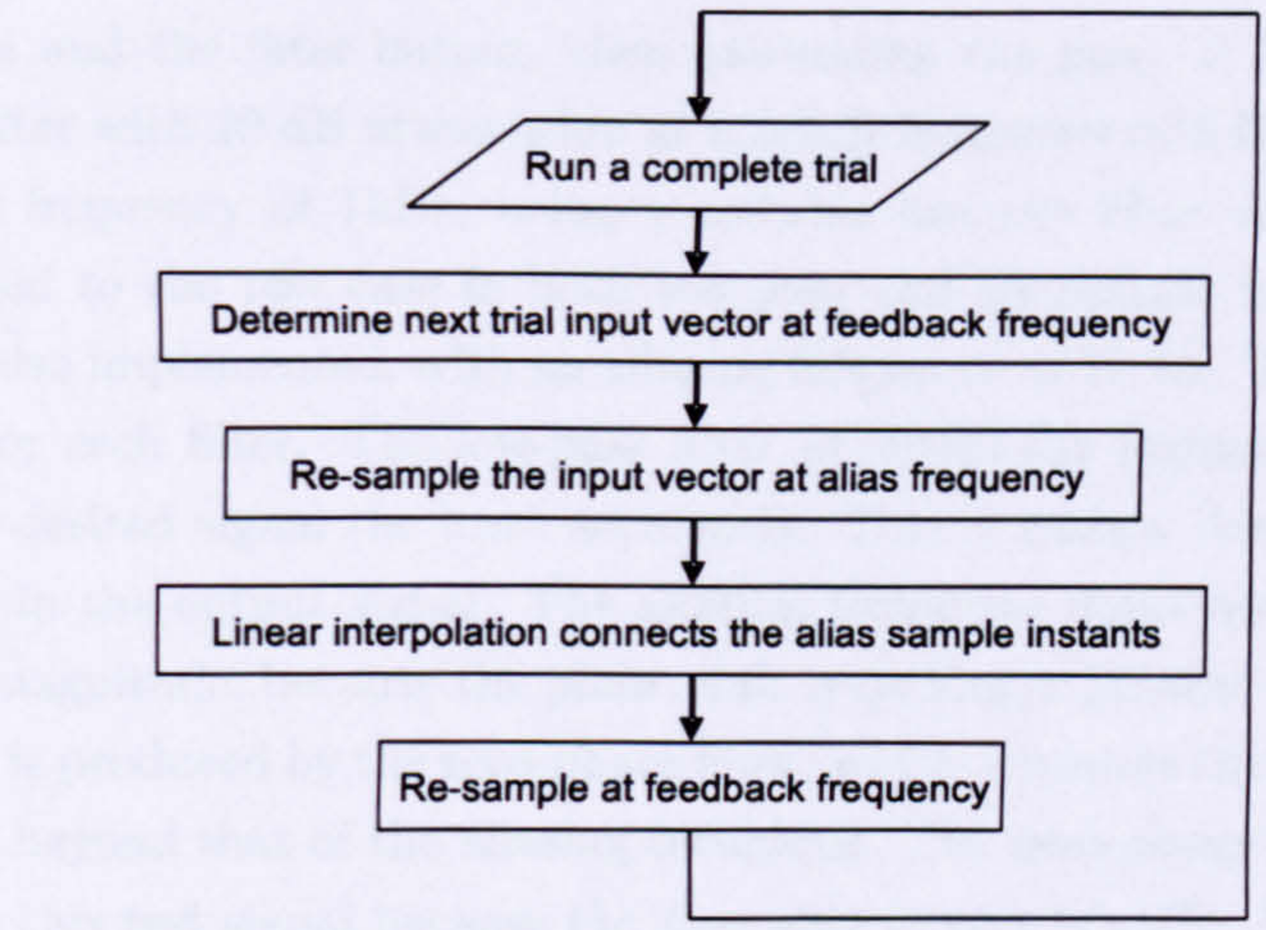


FIGURE 4.41: Aliasing technique flowchart

0.5 Hz and amplitude 1 unit. In Figure 4.42b the sine-wave is superimposed with an extra 7 Hz sine-wave of amplitude 0.1 units. Figure 4.42c shows the same signal as in b, sampled at 10 Hz with a zero-order-hold. Figure 4.42d interpolates between the sample instants of Figure 4.42c to produce a smoother signal, closely resembling the original 0.5Hz sine-wave.

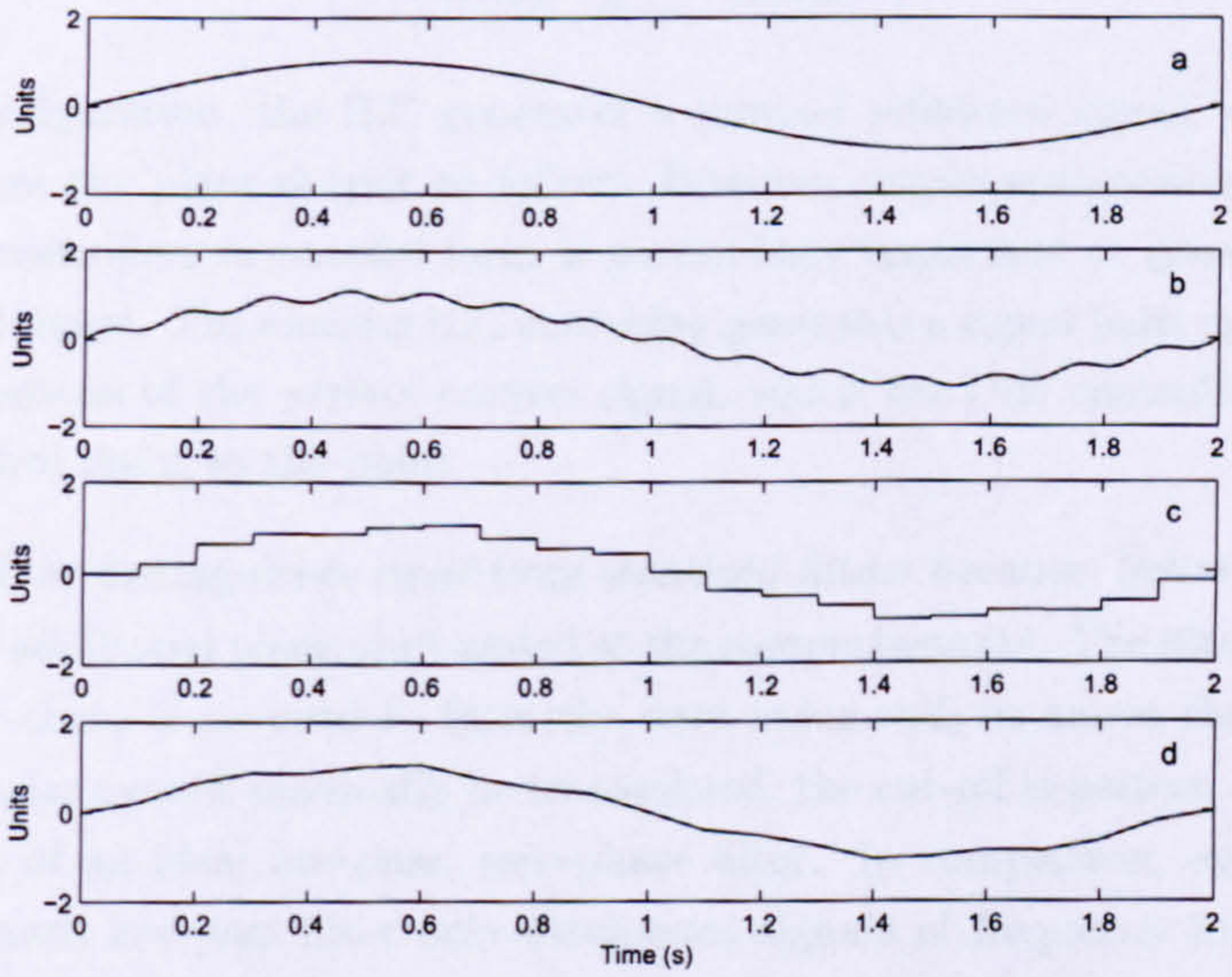


FIGURE 4.42: Comparing sampling methods (a = original sinewave, b = with additional 7Hz signal, c = zero order hold, d = with linear interpolation)

To investigate the ability of different filters, to accurately recreate a desired signal from a corrupted signal, the test case of a 0.5 Hz, amplitude = 1 unit sine-wave corrupted with a 7Hz, amplitude = 0.1 unit sine-wave has been used in simulation studies. The performance of each filtering technique is measured by finding the error between the

desired signal and the filter output, then calculating the mse. A 3rd order, low-pass, Chebychev filter with 20 dB attenuation at a cutoff frequency of 6 Hz has been designed for a sample frequency of 1kHz, using a suitable discrete filter design toolbox. The filter is applied to the test case in both low-pass and zero-phase modes. The aliasing technique is also implemented, with an aliasing frequency of 10 Hz. Table 4.10 shows the mse results for each filter. The low-pass filter produces the highest mse and therefore recreates the desired signal the least accurately. This is mainly due to the large phase shift induced in the output signal. The aliasing technique improves the mse by almost one order of magnitude, because the phase shift is no longer present. The most accurate output signal is produced by the zero-phase filter, which improves the mse a further order of magnitude beyond that of the aliasing technique. The zero-phase filter is particularly well suited to this test signal because the sine-wave is very smooth. With respect to the number of manipulations which must be performed to achieve zero-phase filtering, the simple aliasing technique performs well.

TABLE 4.10: Simulated mse values for different filters in the test case

Filter type	mse (units)
Low pass	0.01809
Zero-phase	0.00024
Aliasing	0.00287

In a series configuration, the ILC generates a ramped reference signal, which the PID controller forces the plant output to follow. However, implementation of the PID and aliasing ILC controllers in parallel form is particularly important to generate a smooth overall control signal. The aliasing ILC controller generates a signal built up from straight line approximations of the perfect control signal, which the PID controller adjusts into a smooth control input to the plant.

The aliasing filter distinguishes itself from standard filters because, below the alias cut-off, there is no additional phase shift added at the sample instants. The filter is essentially zero phase, so there is no need to filter the data twice and, as above the alias cut-off, higher frequencies cannot physically be transmitted, the cut-off is perfect. These are the characteristics of an ideal low-pass, zero-phase filter. In comparison, even a carefully designed standard low-pass filter only attenuates signals of frequency higher than the pass band and a small component of these frequencies will leak through the filter and cause the learning controller to eventually become unstable.

The aliasing filter has been implemented on the X-axis with a range of values for the alias frequency. Figure 4.43 shows the mse performance with alias gaps 50, 60 ,70 ,80, 90 and 100, corresponding to sample frequencies 20, 16.7, 14.3, 12.5, 11.1 and 10 Hz respectively. An alias gap equal to 50 (20 Hz) produces an unstable ILC controller. At 20 Hz, the sampling theorem predicts that 10 Hz signals can be transmitted across the filter and the 12 and 13 Hz resonances are therefore aliased to 10 Hz. The amplitude of

the resulting 10 Hz signal eventually builds up in the iteration loop and causes instability. The PI_{100} values associated with each alias gap are presented in Table 4.11.

TABLE 4.11: PI_{100} values for different alias gap

Alias gap	X-axis
50	3.527176
60	3.443724
70	3.422716
80	3.406701
90	3.457099
100	3.496470

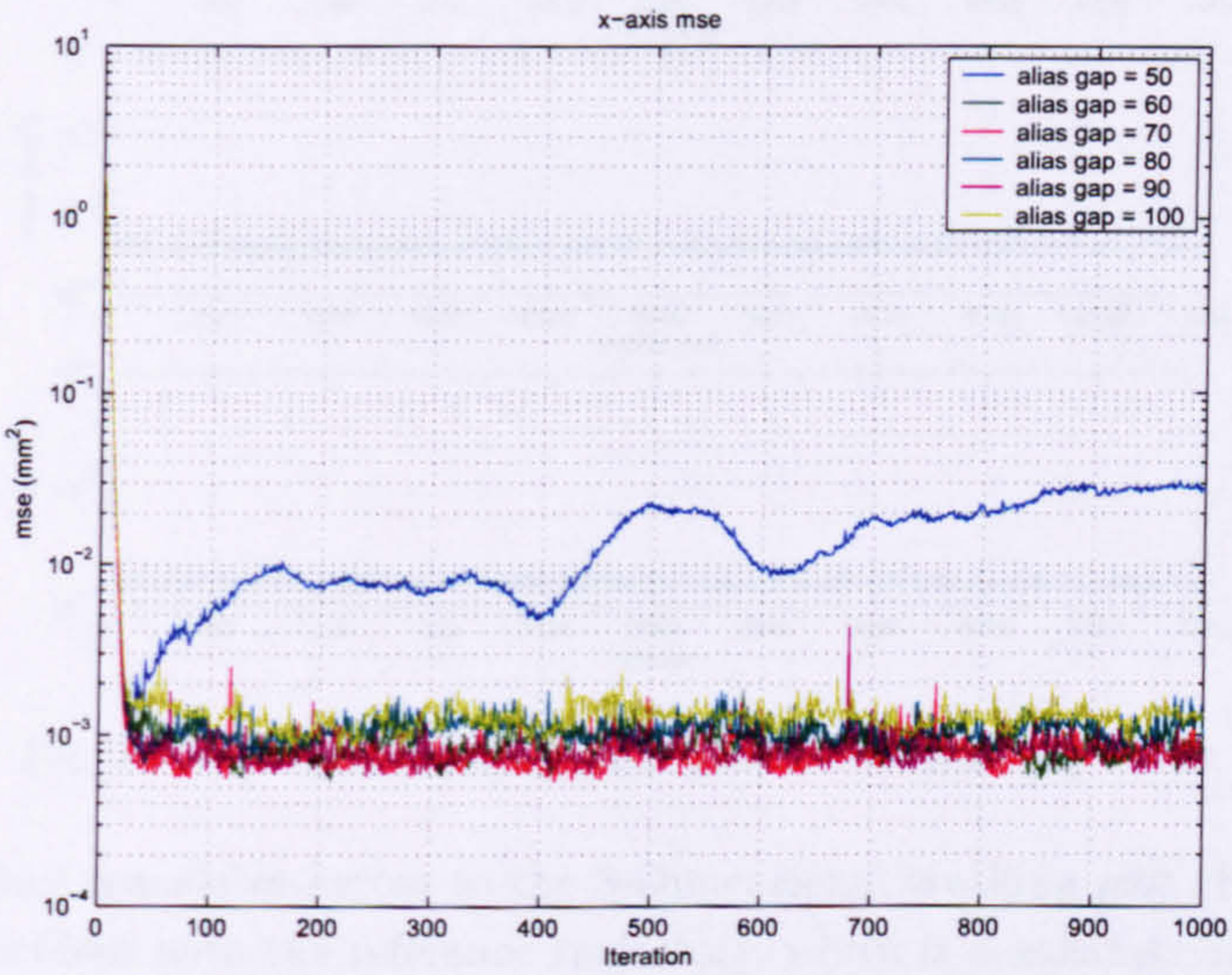


FIGURE 4.43: X-axis mse comparison for aliasing controller using different alias gap values

If the alias gap is reduced to 40 (25 Hz) the system becomes unstable more rapidly. At this frequency, 12.5 Hz signals can be transmitted across the filter, therefore the 12 Hz resonant frequency can pass un-attenuated and rapidly causes instability. If the alias gap is increased to 60 (16.66 Hz) the amplitude of the resonance when aliased to 8.3 Hz is sufficiently small for the controller to remain stable. Increasing the alias gap further has little effect on the tracking performance, until the low frequency error which needs to be learnt is also filtered and lost.

However, an alias gap of 70 provides a good safety margin between the resonant frequencies and the maximum transition frequency, without adversely affecting tracking error reduction. Figure 4.44 displays the mse for 5000 iterations, recorded during a long-term stability test. This variant of the hybrid algorithm achieves both long-term stability and excellent error reduction. The mse is reduced by at least 3 orders of magnitude,

approaching 4 orders of magnitude for the Y and Z-axes. Table 4.12 presents the PI_{100} values for the hybrid controller coupled with the aliasing filter.

TABLE 4.12: Hybrid controller with aliasing, alias gap = 70, PI_{100} values

Gain	X-axis	Y-axis	Z-axis
100	3.441941	4.392282	6.108810

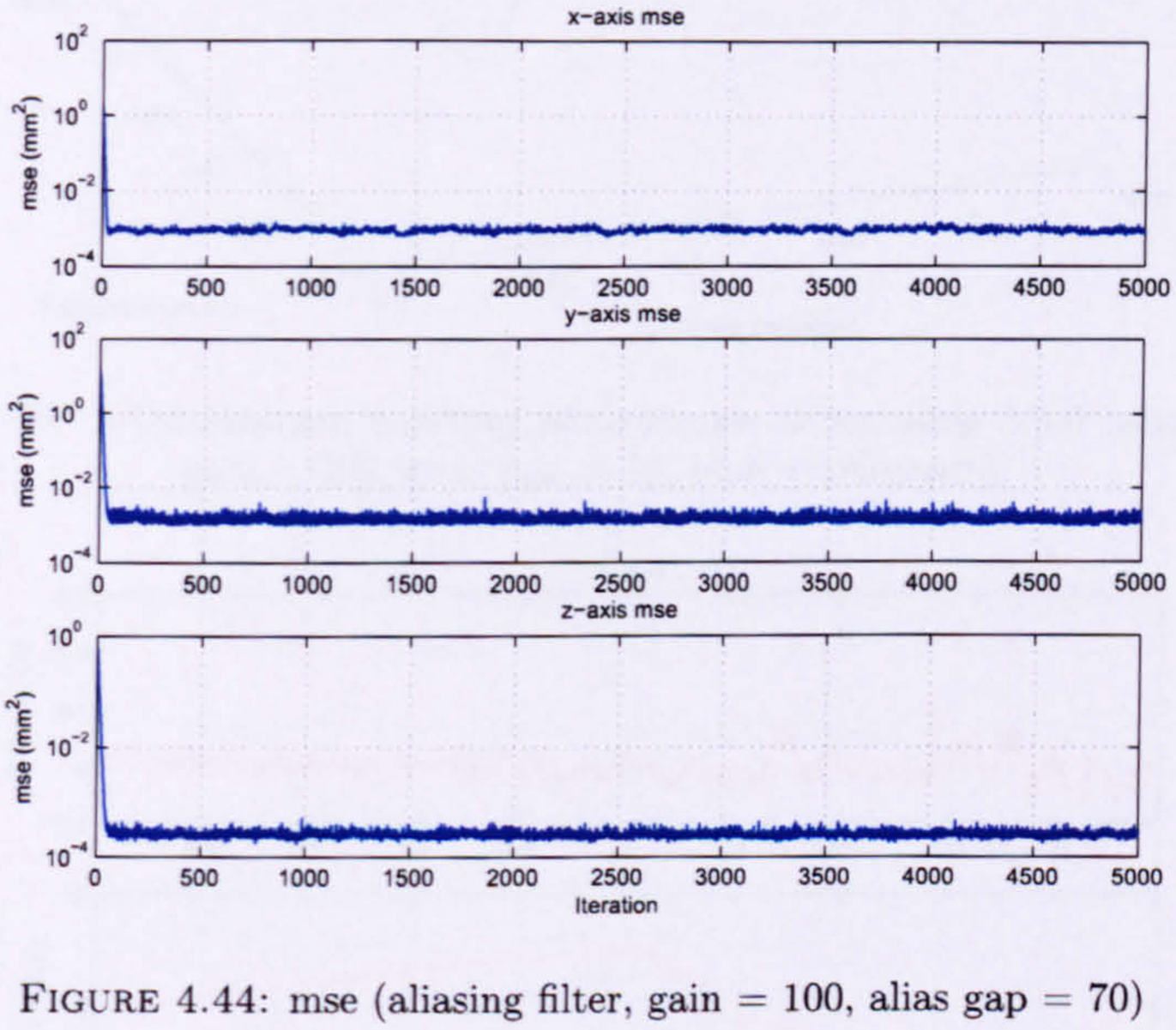


FIGURE 4.44: mse (aliasing filter, gain = 100, alias gap = 70)

This performance translates across to the 3-Dimensional tracking plot (Figure 4.45) as an output coincident with the reference trajectory, which is a substantial improvement over the feedback controller alone.

Figure 4.46 verifies that the ILC input signal remains free of noise and resonance (similar plots for the Y and Z-axes can be found in Appendix B). As with the zero-phase filter, the feedback controller excites small amplitude oscillations at the resonant frequencies. The aliasing filter is particulary well suited to the reference trajectories used in this project, because they are derived from linear variations in velocity (velocity reference profiles can be found in Appendix B). Due to the integrating nature of the plant, the learning controller is required to learn a profile which closely resembles the derivative of the reference. As the aliasing filter does not smooth sharp corners in the profile, the output from the ILC matches the reference derivative more accurately than that produced by other filtering methods.

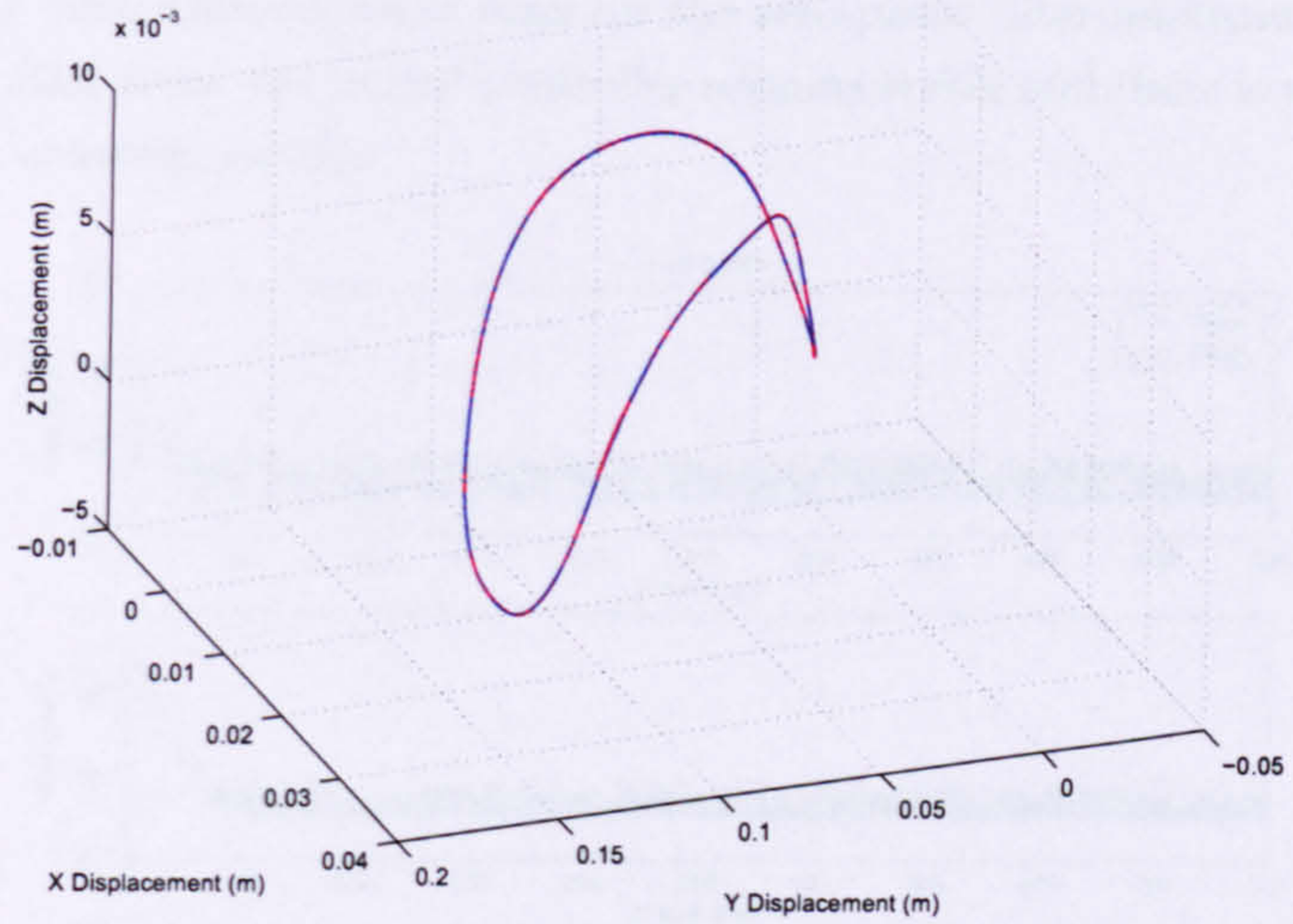


FIGURE 4.45: 3-Dimensional tracking performance at iteration 5000 (aliasing filter, gain = 100, alias gap = 70, blue = reference)

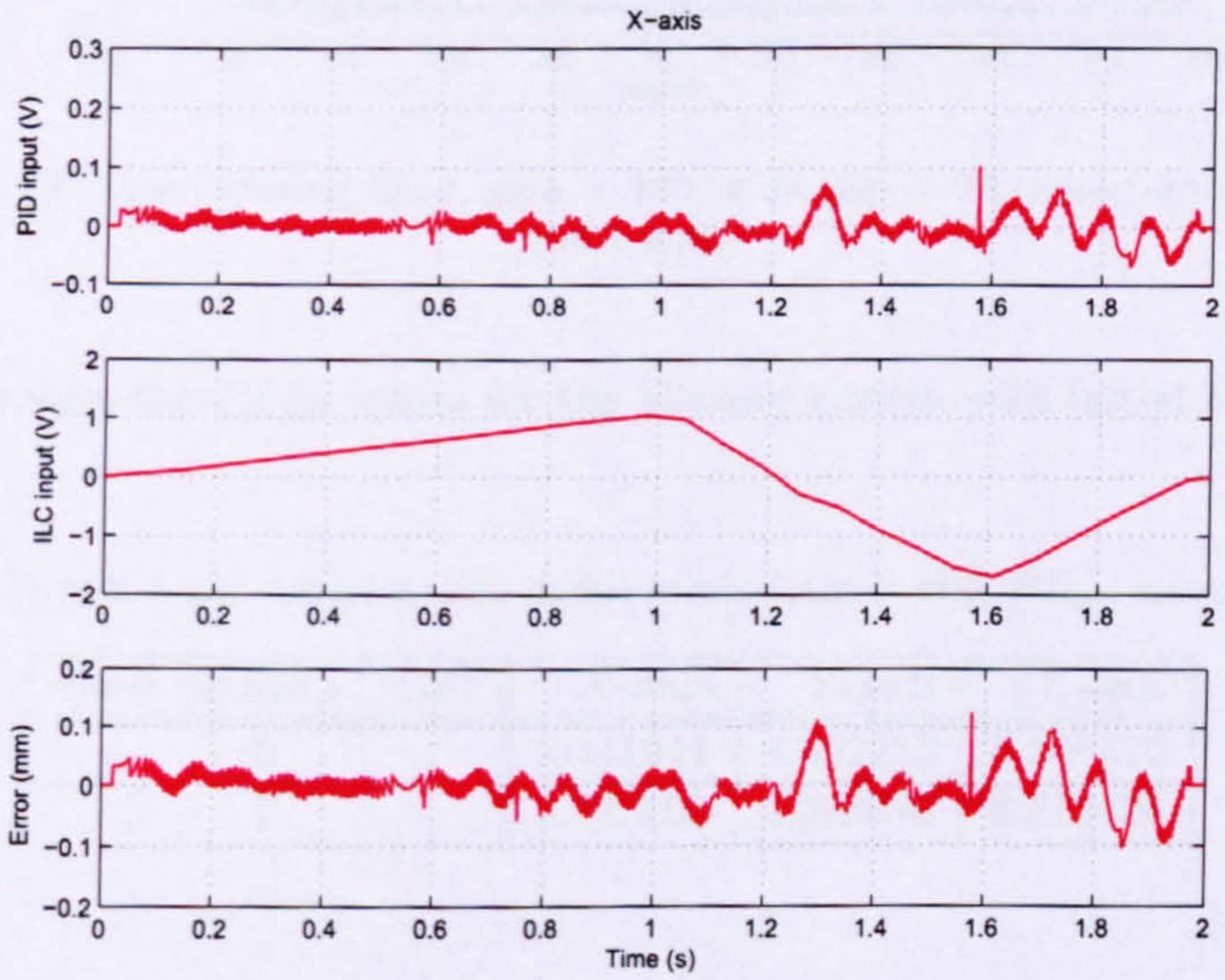


FIGURE 4.46: X-axis PID, ILC and tracking error (aliasing filter, gain = 100, alias gap = 70, iteration 5000)

4.4.7.1 Initial state error tolerance

Large amplitude control signals generated in response to initial error bounds of 2mm and greater, cause the amplifier drives to trip, therefore the initial state error test is limited to a 1mm bound. A bound of 1mm produces a more significant effect on tracking performance in the aliasing filter implementation than the zero-phase filter implementation, because the minimum tracking error level is one to two orders of magnitude smaller. The mse plot therefore shows more substantial variation, though the maximum steady

state tracking error remains lower than for the zero-phase filter implementation. In the presence of initial error, the hybrid controller remains stable and there is negligible effect on the initial convergence rate.

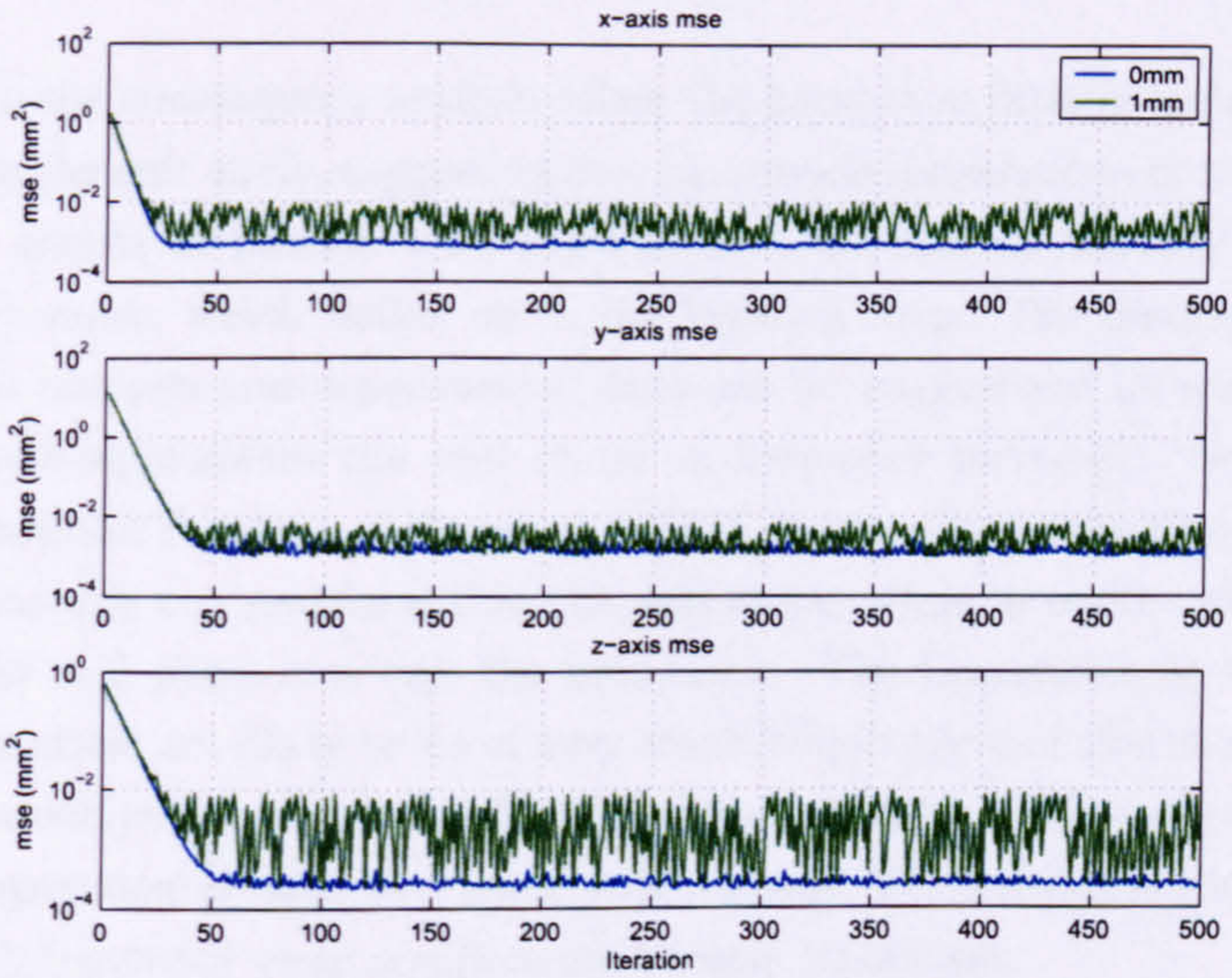


FIGURE 4.47: mse (aliasing filter, gain = 100, alias gap = 70, initial error bounds 0 and 1 mm)

Table 4.13 displays the PI_{100} values for the aliasing system with initial error bounds of 0 and 1mm.

TABLE 4.13: Aliasing with initial error, Gain = 100, PI_{100} values

Error bound (±mm)	X-axis	Y-axis	Z-axis
0	3.441941	4.392282	6.108810
1	3.625103	4.398072	6.572404

4.5 Convergence analysis

Using the monotonic convergence criterion (Equation 4.14), Nyquist plots have been generated for the parallel hybrid controller in the case when no filtering is applied, and when band-stop, low-pass and aliasing are applied. Zero-phase low-pass filtering has not been plotted, because of the difficulties involved in generating a suitable transfer function for the anti-causal filter. Figure 4.48 shows the convergence plots for the X-axis when the learning gain is 100. The plots for the Y and Z-axes produce similar results and have therefore been omitted.

Plot (a) clearly shows that the controller implementation without filtering, cannot achieve

monotonic convergence. The plot rapidly moves out of the unit circle, which defines the bounds of the convergence criterion. This corresponds well to the mse results obtained from the plant in Figure 4.16, where the mse for gain 100 increases, following the initial reduction.

Plot (b) shows the convergence analysis when the band-stop filter is included. The plot remains within the unit circle, suggesting that monotonic convergence should be achieved. However, the results in Section 4.4.4 indicate that the system stability suffers due to high frequency noise, which builds up in the learning loop. The discrepancy between the theoretical analysis and experimental data can be understood by noticing that the convergence plot approaches the unit circle as frequency increases. Small modelling errors in the transfer function of the plant at high frequencies imply that the simulated convergence analysis can remain within the unit circle, while in reality, the convergence analysis for the real plant may exit the unit circle. The frequencies at which the plot leaves the unit circle are likely to be of very small amplitude and therefore a significant amount of time will pass before their effect becomes noticeable. This analysis corresponds well to the experimental data in Figure 4.24 because 200 iterations are implemented before the high frequency error components become significant.

The convergence curve for the low pass filter implementation is presented in plot (c). The plot clearly leaves the unit circle, indicating that monotonic convergence is impossible. Referring back to Figure 4.32, this is precisely what occurs in reality. For all three axes, the mse reduces rapidly at the start of the test, reaches a minimum, but then increases to a final steady state value. Monotonic convergence is not achieved, even though the system remains stable for the 5000 iteration test. In the zero-phase implementation of the low-pass filter, a phase of 0 degrees implies that the plot travels to the left of the graph along the real axis, moving nearer the origin and further into the unit circle. The large lobes which leave the unit circle in plot (c) will therefore not occur and the conditions for monotonic convergence will be satisfied.

Plot (d) represents the convergence analysis for the aliasing technique using an alias gap of 70. Frequencies above 7.15 Hz cannot exist within the learning loop and therefore the plot does not leave the unit circle and monotonic convergence is achieved, as supported by the mse curves in Figure 4.44.

4.6 Summary

PID feedback control has been implemented on the gantry to serve as a benchmark for performance. Subsequently, P-type ILC alone was found to perform very poorly when controlling an integrating plant. The result was improved, when feedback control and ILC were combined in two hybrid variants. The feedback controller compensated for the integrating plant and provided a reasonable level of tracking accuracy during

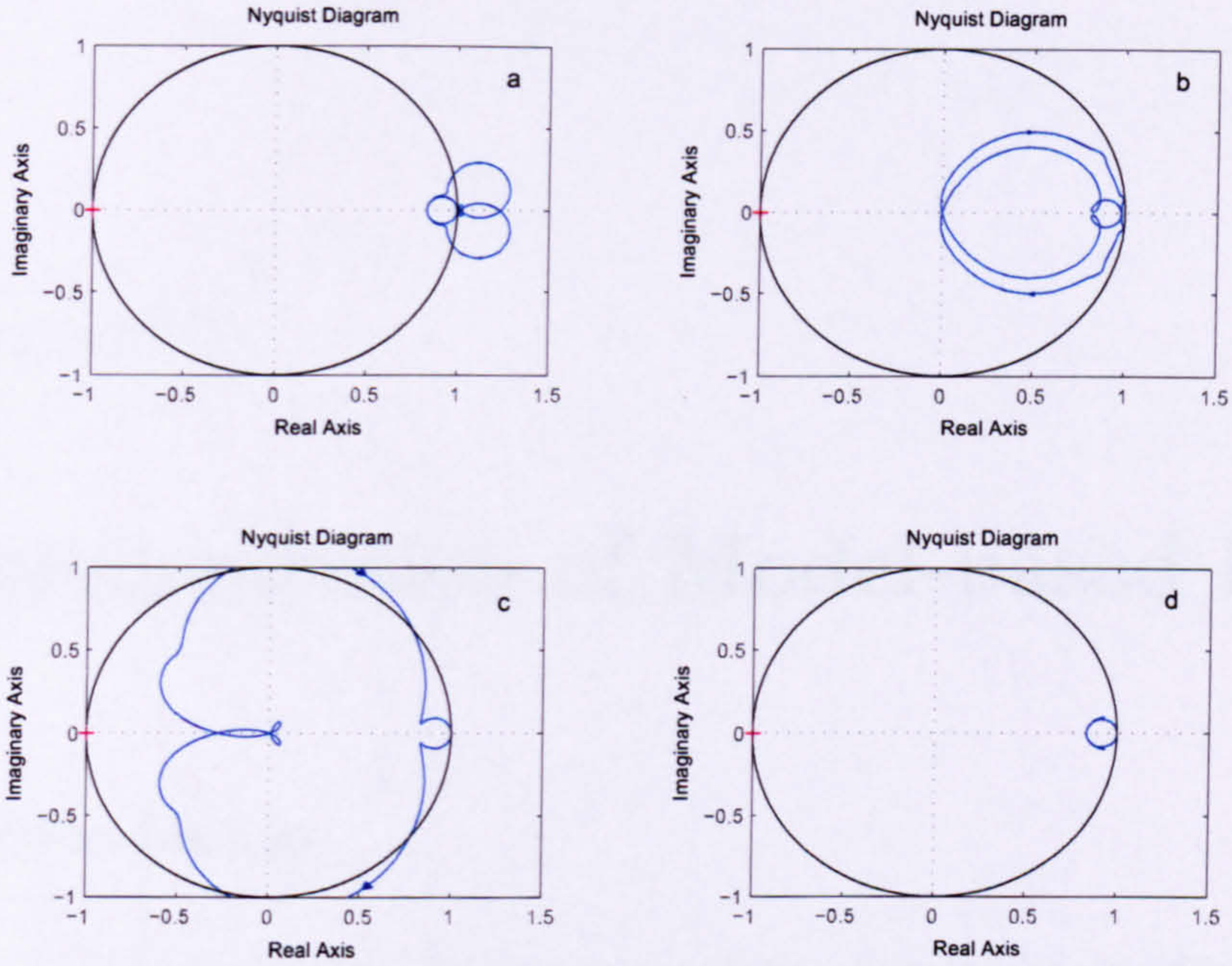


FIGURE 4.48: X-axis convergence plots, learning gain = 100 (a = no filtering, b = band-stop, c = low-pass, d = aliasing)

the first few iterations, while the ILC improved the performance of the PID. In both series and parallel configurations, the resulting controller was found to excite natural plant resonances which grew at each iteration and eventually destabilised the controller. The mechanism behind this instability has been analysed and is well understood. The build up of high frequency noise in the iteration loop has a similar effect on stability. Several filtering techniques were progressively applied to the learning controller output in an attempt to attenuate resonance and noise. Zero-phase filtering and a new aliasing technique were found to be particularly efficient for providing long-term stability and low residual tracking error.

Chapter 5

Implementation of Model-based ILC

5.1 Introduction

These ILC algorithms intrinsically use plant modelling data as a means of improving algorithm performance. While the performance of model based feedback controllers definitely depends on the accuracy of the model, ILC has the ability to compensate for modelling errors and thus further reduce tracking error. The ability to predict how the plant will behave in response to a given input, allows the use of mathematical techniques related to optimal, robust and adaptive control. This chapter describes the implementation of three previously developed ILC algorithms, formulated using optimal control methodologies: adjoint, inverse and norm-optimal ILC. Robustness of these algorithms with respect to initial state error, and plant modelling error has also been investigated.

Many model based ILC algorithms are developed in the time domain using state-space models, which consider the response of the plant for infinite positive time. Given the finite-time, infinite iteration nature of the ILC framework, it is only necessary to represent accurately the behaviour of the plant during the iteration period. Consider the discrete state space equations:

$$\left.\begin{aligned} x(t+1) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \end{aligned}\right\} \tag{5.1}$$

It is assumed that $CB \neq 0$, the system is invertible and that it is SISO. If one iteration contains m sample instants, the input and output vectors are represented by:

$$u_k = [u_k(0) \quad u_k(1) \quad \dots \quad u_k(m-1)]^T \tag{5.2}$$

$$y_k = [y_k(1) \quad y_k(2) \quad \dots \quad y_k(m)]^T \tag{5.3}$$

From these definitions, it is possible to derive the convolution sum solution of the state equations (Equations 5.1) as discussed by Longman (2000), which can be represented by:

$$y(t) = CA^t x(0) + \sum_{i=0}^{t-1} CA^{t-i-1} Bu(i) \tag{5.4}$$

Assuming that $x(0) = 0$ Hätönen, Owens, and Moore (2003a) represent this in matrix form:

$$\mathbf{y} = \mathbf{G}\mathbf{u} \tag{5.5}$$

where:

$$\mathbf{G} = \begin{bmatrix} CB & 0 & 0 & \dots & 0 \\ CAB & CB & 0 & \dots & 0 \\ CA^2B & CAB & CB & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ CA^{m-1}B & CA^{m-2}B & CA^{m-3}B & \dots & CB \end{bmatrix} \tag{5.6}$$

The elements of $\mathbf{G}$ are the Markov parameters of the plant. It is assumed that $\mathbf{G}$ is an invertible matrix. The size of the $\mathbf{G}$ matrix depends ultimately on the sample frequency and the iteration time period. If the sample frequency is high and the time period is long, the $\mathbf{G}$ matrix becomes very large, which poses a problem when the inverse matrix must be calculated. The numerical resolution of the inverse matrix solver results in a solution which is singular. The sample frequency has therefore been chosen as 100Hz and is used for all three algorithms to allow a fair comparison of performance. Higher sampling rates require significant memory, for example if the sample frequency is 1000Hz, the resulting matrices have dimensions 2000×2000, corresponding to 4,000,000 elements. Using the 'double' storage type (64 bits) this results in a matrix requiring requiring 30.52 MB of storage space. With three axes operating simultaneously, this immediately requires 91.56 MB, just to store the plant models, which is a very large demand on memory resources.

However, the chosen sample frequency of 100Hz, coupled with the 2 second time period of the reference trajectories, produces $\mathbf{G}$ matrices with dimensions 200×200, corresponding to 40,000 elements, requiring 0.31 MB of storage space. Matrices of this size require minimal memory resources and can therefore be stored in full, in the control hardware. Observing the construction of the $\mathbf{G}$ matrix (Equation 5.6), the first column contains all of the numerical values required to construct the remaining columns. Therefore, it is possible to compress the matrix into a vector representation, corresponding to the first column, and significantly reduce the amount of storage space required. In the adjoint and inverse ILC algorithms, there are a limited number of matrix operations which must be performed using the $\mathbf{G}$ matrices. Therefore it is not difficult to develop functions

which can perform these operations based on the vector representation of $\mathbf{G}$.

The model-based algorithms described in this chapter are implemented differently from the basic algorithms discussed in Chapter 4. Unlike the basic algorithms, which require a feedback controller to compensate for the integrating plant, these algorithms are implemented alone with no additional input from other controllers. Therefore, the mse recorded at iteration 1 is not a result of the feedback controller; the input during iteration 1 is held at zero and the mse corresponds directly to the reference trajectory.

5.2 Adjoint Algorithm

5.2.1 Algorithm development

Hätönen, Harte, Owens, Ratcliffe, Lewin, and Rogers (2003b) have developed a modification of the original steepest descent ILC algorithm proposed by Furuta and Yamakita (1987). Using the plant model $\mathbf{G}$, the input update algorithm is defined as:

$$\mathbf{u}_{k+1} = \mathbf{u}_k + \epsilon_{k+1}\mathbf{G}^T\mathbf{e}_k \tag{5.7}$$

where ϵ_{k+1} represents a learning gain, which is automatically selected at each iteration by minimisation of the cost function:

$$J(\epsilon_{k+1}) = ||\mathbf{e}_{k+1}||^2 + \omega\epsilon_{k+1}^2 \tag{5.8}$$

where ω is a tuning parameter selected by the design engineer and is used to adjust algorithm robustness. This cost function represents two design objectives. The first objective is to achieve small tracking error and is analogous to achieving fast convergence and low residual mse. The second objective which is to keep the magnitude of ϵ_{k+1} small, conflicts with the first, because previous experiments demonstrated that smaller learning gain results in slower convergence. The tuning parameter has been added, to improve the robustness of the algorithm, with respect to non-linear and non-repeating disturbances, as the algorithm learns more cautiously.

The solution of the cost function (Equation 5.8) yields a learning gain update defined by:

$$\epsilon_{k+1} = \frac{||\mathbf{G}^T\mathbf{e}_k||^2}{\omega + ||\mathbf{G}\mathbf{G}^T\mathbf{e}_k||^2} \tag{5.9}$$

Practical implementation of the algorithm consists in calculating Equation 5.9 followed by Equation 5.7 during the time between iterations.

5.2.2 Initial implementation

Initial implementation of the adjoint ILC algorithm is performed using the high order plant models and with ω set to zero. The algorithm therefore seeks to minimise the error norm, without limiting the magnitude of the learning gain and is able to achieve an optimal error convergence rate. The first tests consisting of 10 iterations confirm that the adjoint algorithm does converge and functions as expected. Gradually increasing the required number of iterations reveals that long-term stability and low residual tracking error can also be achieved. Table 5.1 presents the PI_{100} values for the long-term stability test, where PI_{100} is defined as the sum of mse values over 100 trials.

TABLE 5.1: Adjoint ILC, PI_{100} values

X-axis	Y-axis	Z-axis
1.786522	1.544323	2.955504

Figure 5.1 shows the mse performance obtained by all three axes during a 5000 iteration test. Initial convergence is rapid, the error is reduced by 3 to 4 orders of magnitude within 100 iterations. The algorithm then continues to learn, achieving steady state minimum tracking error by iteration 1500. The steady state mse, has upper and lower bounds, which are several orders of magnitude apart, implying that the tracking performance for consecutive iterations varies significantly. Ideally, the mse should reach a minimum value and remain constant. As it does not, the performance of the algorithm is reduced.

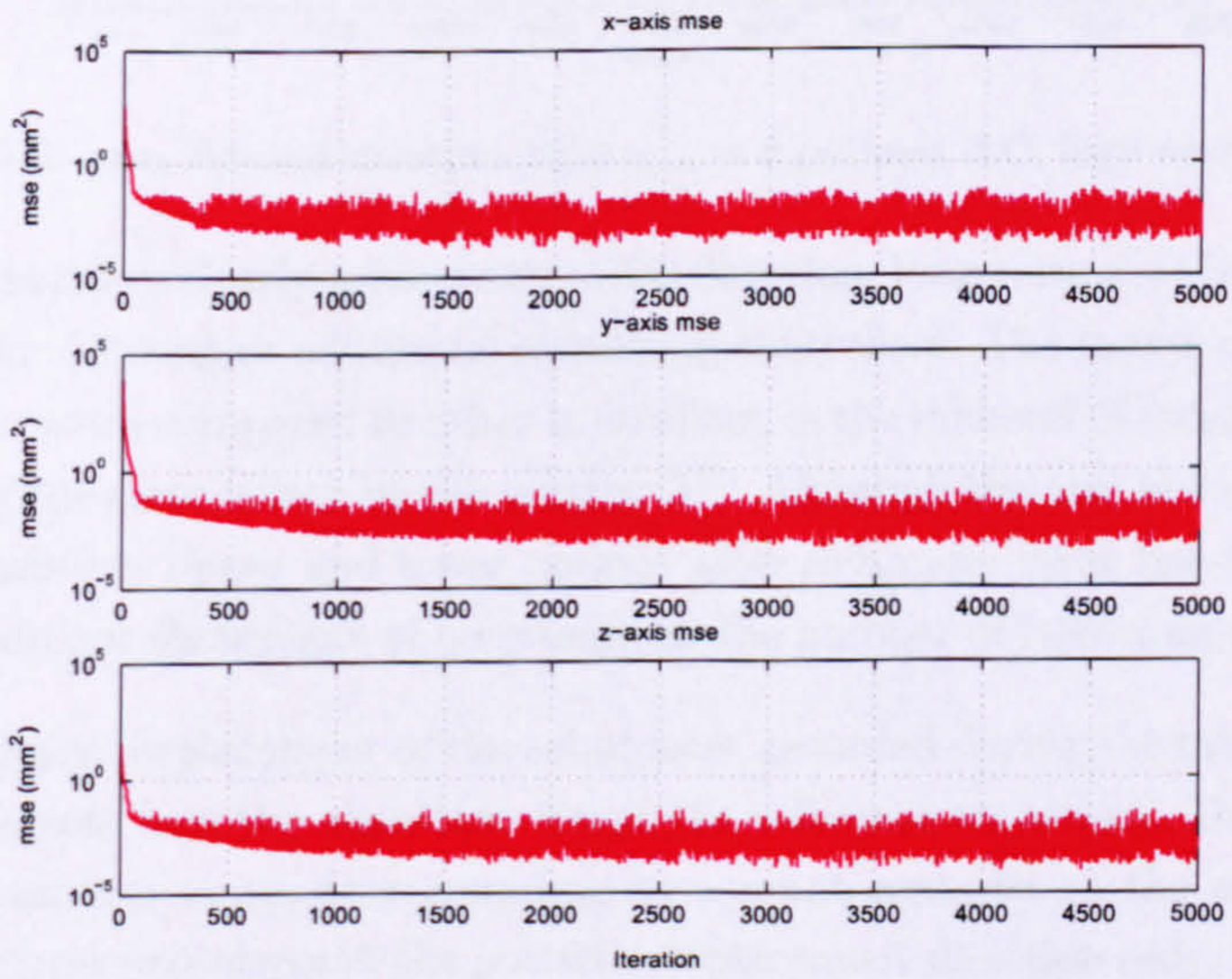


FIGURE 5.1: mse (adjoint ILC, high order models)

The variation is likely to be caused by one or more factors, which include: random and non-repeating disturbances, noise, and high learning gain when the tracking error is

small. If the learning gain is large, the algorithm may effectively try to learn too much at each iteration, the plant input is altered more than necessary and the performance varies accordingly. The influence of learning gain is relatively straightforward to investigate. The 5000 iteration test is repeated, but at iteration 2000 the learning gain ϵ_{k+1} is set to zero for all further iterations. The tracking performance should therefore be equal for iterations 2000 to 5000, because the input to the plant remains constant. Figure 5.2 shows the mse results for this test. Clearly the performance beyond iteration 2000 varies in a similar manner to the performance recorded before iteration 2000, which strongly suggests that the variation is caused by random disturbances, for which the learning controller cannot compensate.

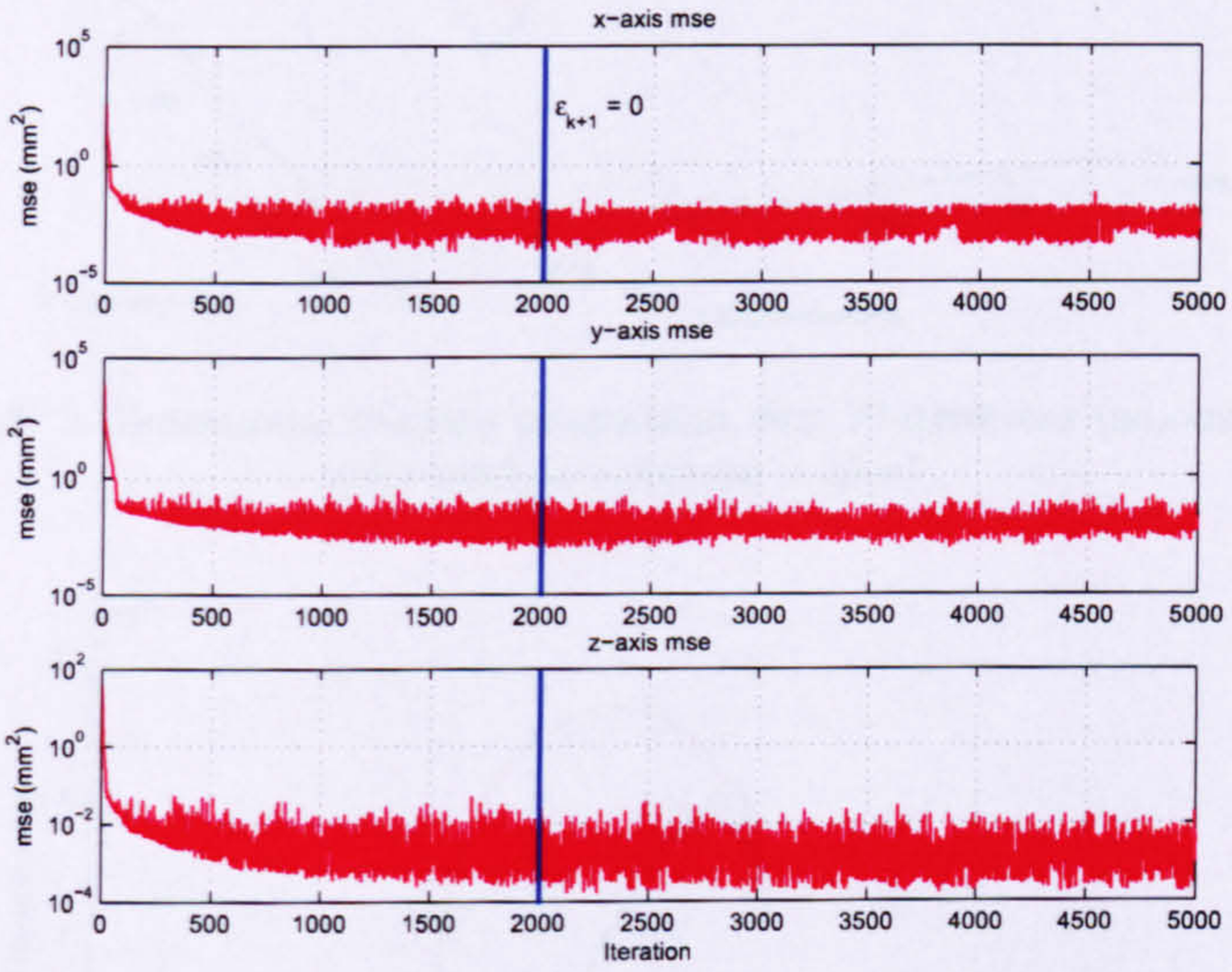


FIGURE 5.2: mse, Beyond iteration 2000 $\epsilon_{k+1} = 0$ (adjoint ILC, high order models)

The adjoint algorithm clearly achieves the 5000 iteration, long-term stability target without any need for filtering or additional stabilising controllers. The source of adjoint algorithm stability, when compared to other controllers, is the inherent filtering effect caused by multiplying the error vector by the matrix $\mathbf{G}^T$. Although the mse in Figure 5.1 varies significantly between upper and lower bounds after minimum error has been achieved, these bounds do not show signs of increasing as the number of iterations increases.

The 3-Dimensional displacement of the robot axes, recorded during the first 10 iterations (Figure 5.3) depicts how the algorithm learns the reference trajectory. During iteration 1 the robot does not move, corresponding to a point centered at the origin. During iteration 2 all three axes move in the positive displacement direction only, this is because of the integrating nature of the plant and because all points on the reference trajectory are positive with respect to the origin. By iteration 3, the shape of the trajectory is beginning to develop quite clearly, however the amplitude of motion is far too small. At subsequent iterations, the algorithm steadily increases the amplitude of the input to

match the plant output with the reference. At iteration 5000, the measured displacement and reference are almost coincident (Figure 5.4).

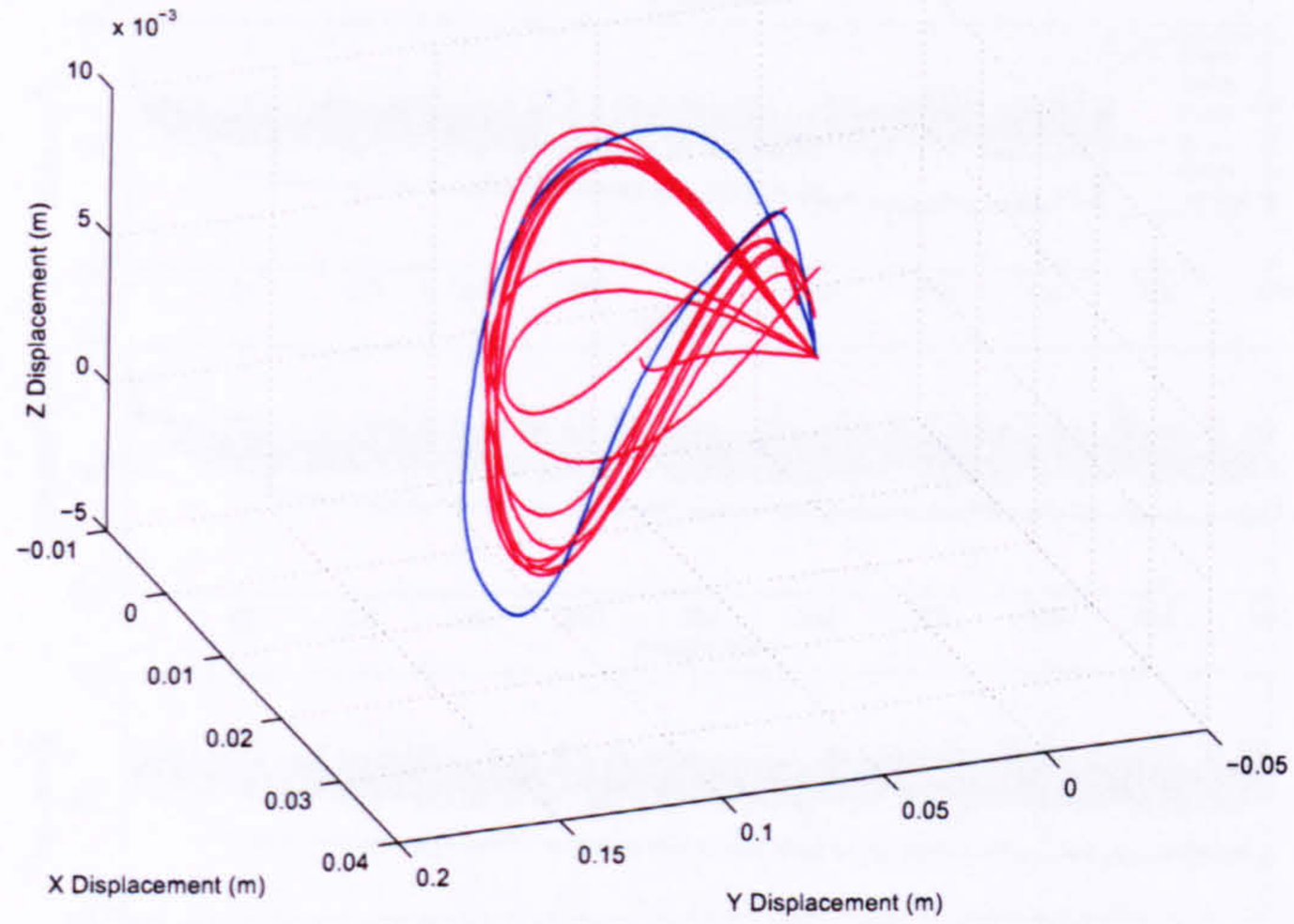


FIGURE 5.3: 3-Dimensional tracking progression, first 10 iterations (adjoint ILC, high order models, reference = blue)

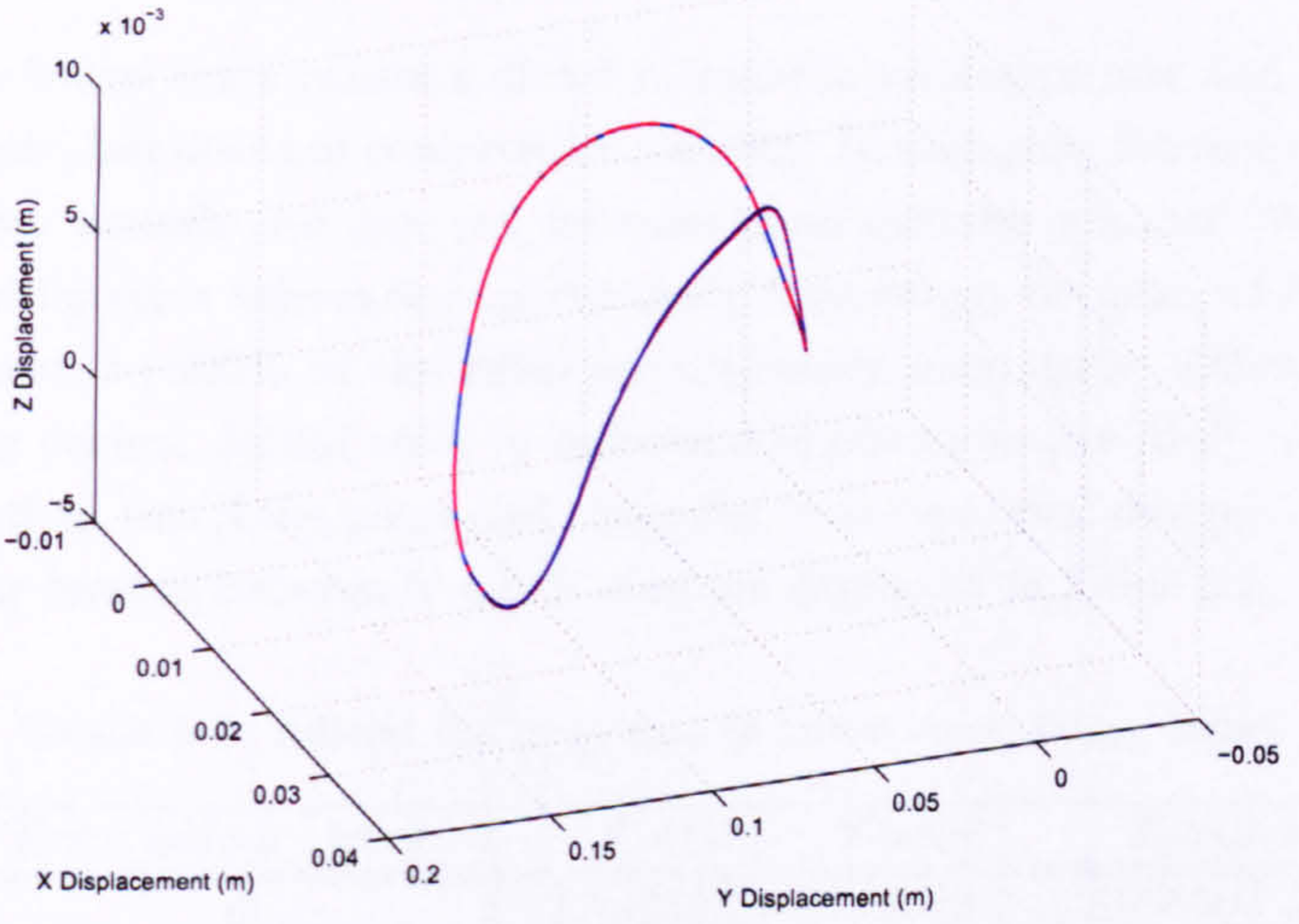


FIGURE 5.4: 3-Dimensional tracking performance (adjoint ILC, high order models, iteration 5000, reference = blue)

5.2.3 Robustness to initial state error

Due to the purely feed-forward nature of the adjoint algorithm, it is poorly suited to compensating for initial state error. The feed-forward nature of the algorithm does have an advantage over feedback based methods, because the plant input demands do not contain steps or spikes which could damage or overload the robot. Figure 5.5 displays

the mse curves corresponding to initial homing bounds of 2, 4, 6, 8 and 10 mm compared to the mse generated without initial error.

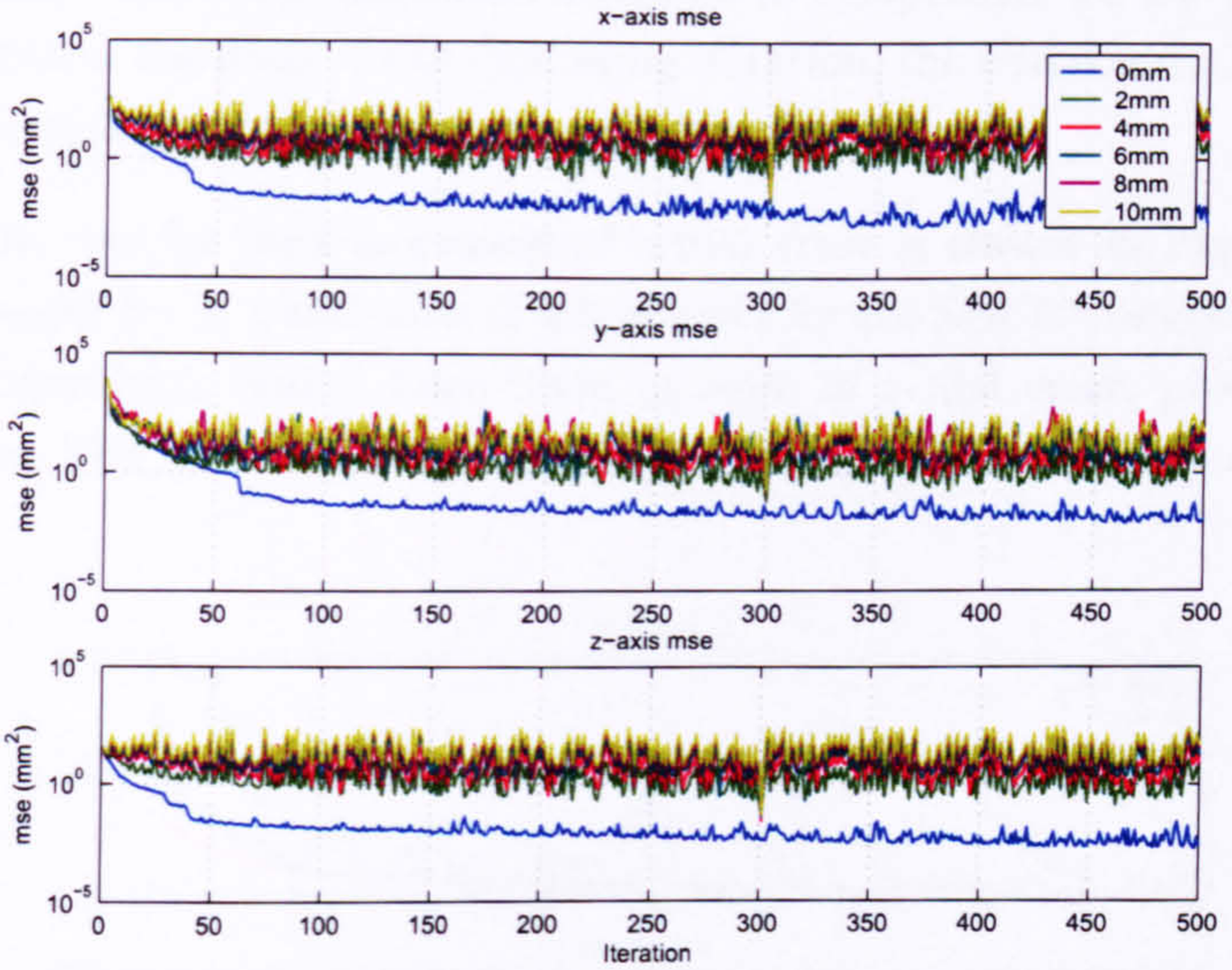


FIGURE 5.5: mse (adjoint ILC, high order models, initial error bounds 0, 2, 4, 6, 8 and 10 mm)

Increasing the initial error causes a direct increase in minimum mse and a reduction in convergence rate, but does not compromise stability. In each plot, the mse remains within upper and lower bounds and does not increase in an unstable manner. With respect to the Z-axis, initial error tolerance is particularly interesting, because ± 10 mm of initial error corresponds to 200% of the reference trajectory magnitude, although very little useful learning occurs. Initial error in industrial applications are likely to be relatively much smaller than this if the plant and controller have been well designed. PI_{100} values for initial error bounds between 0 and 10 mm are displayed in Table 5.2.

TABLE 5.2: Adjoint ILC tolerance to initial error, PI_{100} values

Error bound ($\pm$mm)	X-axis	Y-axis	Z-axis
0	1.786522	1.544323	2.955504
1	1.886287	1.421555	7.204663
2	2.639053	1.518545	11.953511
3	3.697247	1.561097	23.326395
4	5.250543	1.956606	36.321979
5	6.889515	2.499595	48.145670
6	8.964593	2.095541	64.102576
7	11.182576	2.427005	87.464704
8	13.866400	2.796461	100.000000
9	16.175517	2.334881	100.000000
10	18.854652	2.465911	100.000000

The general shape of each mse curve is very similar, because the pseudo random error is seeded by the iteration number. Larger tracking error is produced as the initial error bound increases, because the algorithm attempts to compensate for the error measured during the previous iteration. At the following iteration, the initial error is different and therefore the compensation made by the algorithm is incorrect.

The increase in mse for each increment of initial error is clearer in Figure 5.6, which displays the results for 0, 2 and 4mm of initial error for the first 50 iterations on a linear, rather than logarithmic scale. Each 2mm increase of initial error, produces a larger increase in mse. The variation in mse is also larger as the upper and lower bounds move further apart.

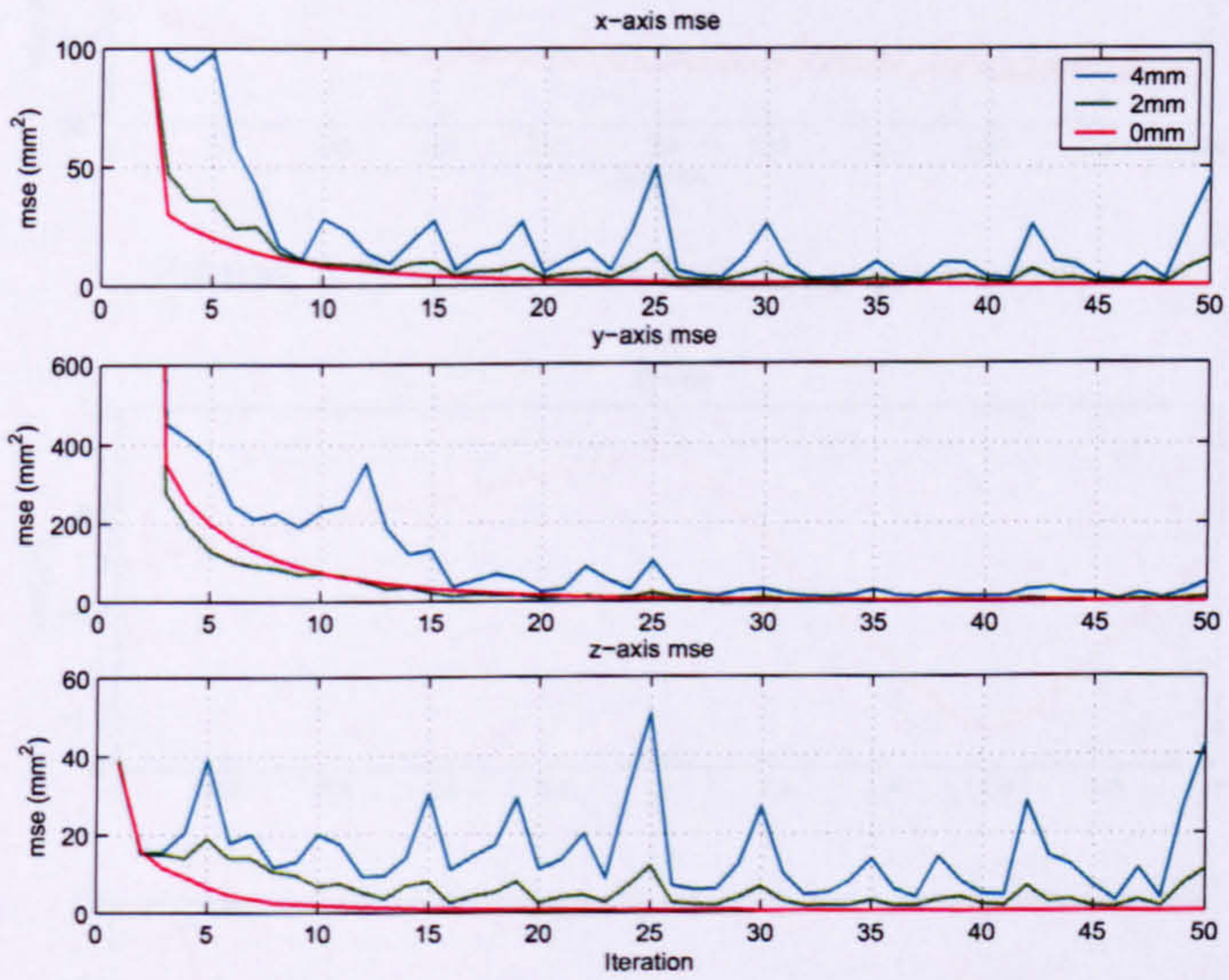


FIGURE 5.6: mse linear scale (adjoint ILC, high order models, initial error bounds 0, 2 and 4mm)

5.2.4 Robustness to plant modelling error

The adjoint algorithm has been implemented, using the first order plant models. The lack of information pertaining to phase and high frequency gain plant dynamics has a limited effect on the resulting tracking performance. The mse plots obtained for a 500 iteration test (Figure 5.7) show minimum error levels approaching one order of magnitude larger than when using the high order models. During 500 iterations, the algorithm remains stable and the mse continuously decreases. Table 5.3 shows the PI_{100} data produced by the 1st order models. The plant input signals (Figure 5.8 and similar Figures in Appendix B for the Y and Z-axes) remain smooth and the error signals do not contain resonances.

Stability and convergence rate are more noticeably affected by low frequency gain error.

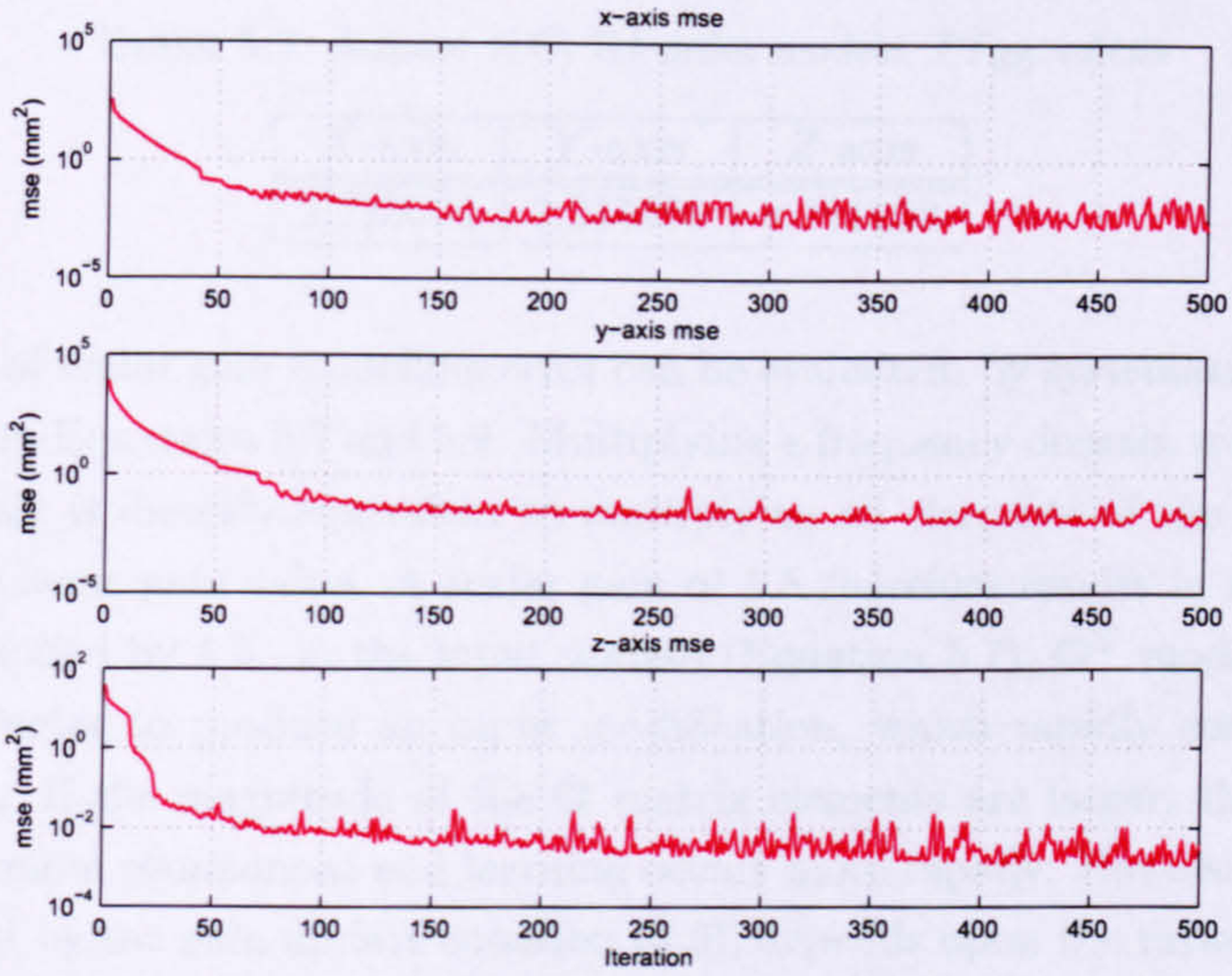


FIGURE 5.7: mse (adjoint ILC, 1st order models)

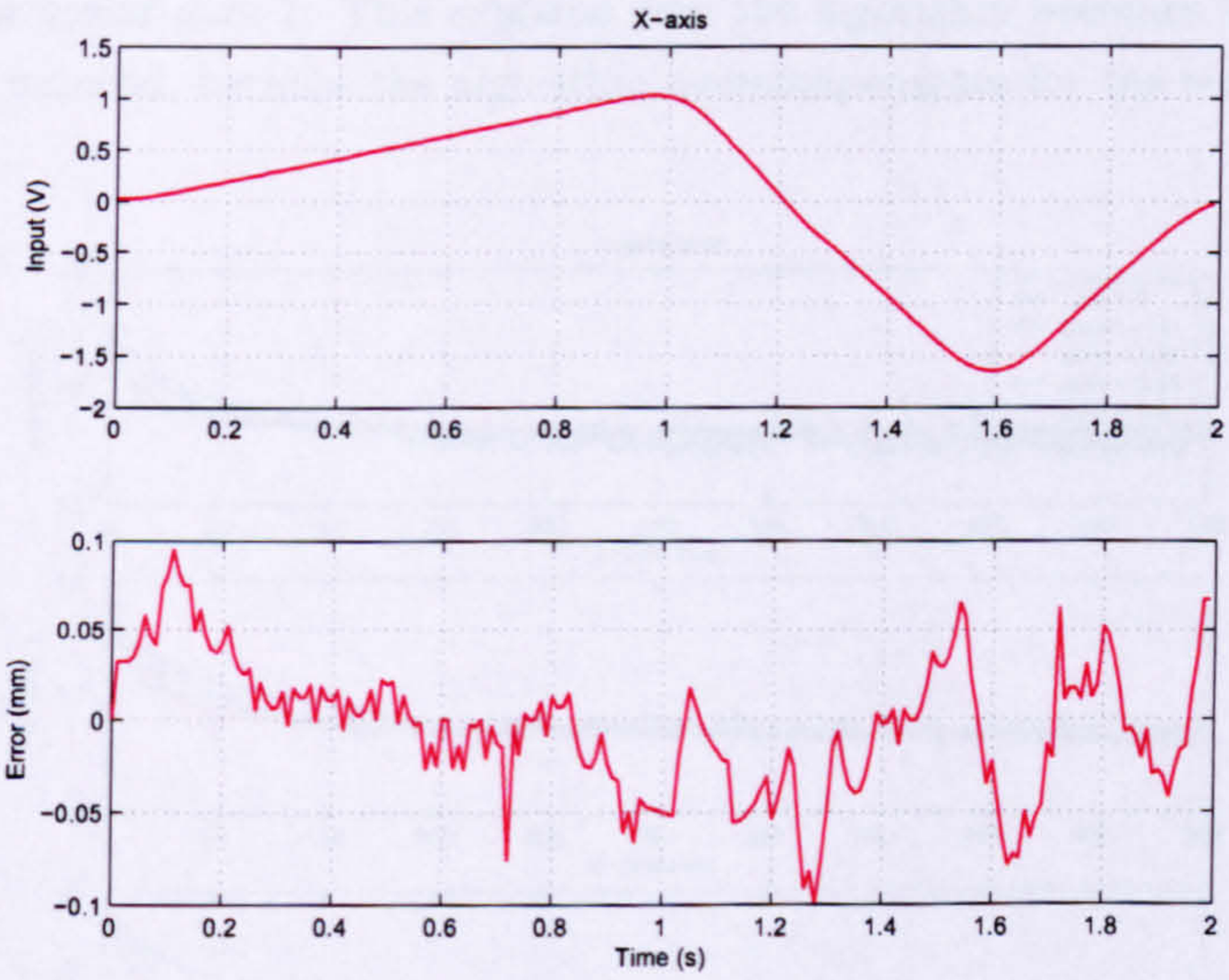


FIGURE 5.8: X-axis input demand and tracking error (adjoint ILC, 1st order models, iteration 500)

Figure 5.9 displays the mse plots generated using the high order models, when multiplied by scalar gains 1.5, 1.25, 0.75 and 0.5. Variation in the gain has minimal effect on the minimum mse level, but produces changes in the shape of the mse curve, as the error converges to minimum mse. With gain larger than 1, the convergence rate during the first iterations (approximately 10) is slower than for gain equal to 1. However, during the following 30-60 iterations, the convergence rate tends to be faster. If the gain is smaller than 1, the convergence rate is significantly slower and if it is too small (0.5), the algorithm is unstable.

TABLE 5.3: Adjoint ILC, 1st order models, PI_{100} values

X-axis	Y-axis	Z-axis
2.713064	2.235936	4.191850

The influence of scalar gain modelling error can be evaluated, by systematically considering the effect on Equations 5.7 and 5.9. Multiplying a frequency domain transfer function by a scalar gain is directly equivalent to multiplying all elements of the state-space $\mathbf{G}$ matrix by the same gain value. A scalar gain of 1.5 therefore results in all elements of $\mathbf{G}$ being multiplied by 1.5. In the input update (Equation 5.7), $\mathbf{G}^T$ modifies the shape of the error vector to produce an input modification, which rapidly converges to the optimal input. If the magnitude of the $\mathbf{G}$ matrix elements are larger, the error vector adjustment is more pronounced and learning occurs more rapidly. However, the learning gain calculated by the gain update equation (5.9), depends upon the ratio $\mathbf{G}^2/(\mathbf{G}\mathbf{G}^T)^2$, which is equivalent to $\mathbf{G}^2/\mathbf{G}^4$, and increasing the scalar gain of $\mathbf{G}$ reduces the learning gain ϵ_{k+1}. When the scalar gain is equal to 0.5, the resulting learning gain can be 4 times larger than for scalar gain 1. This explains why the algorithm becomes unstable as the scalar gain is reduced, because the algorithm overcompensates for the error recorded at each iteration.

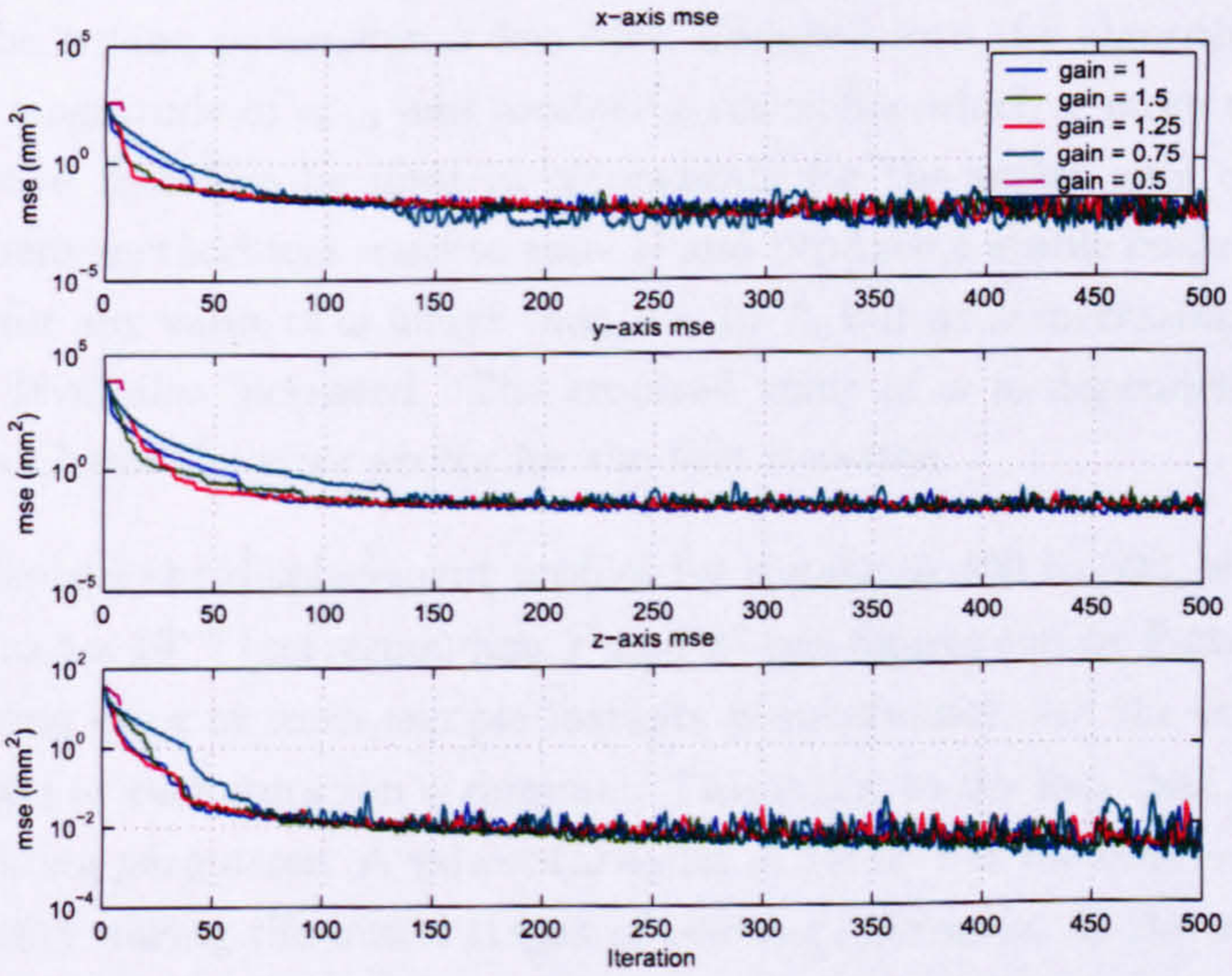


FIGURE 5.9: mse for different scalar gains (adjoint ILC, high order models, gains 1, 1.5, 1.25, 0.75 and 0.5)

Figure 5.10 shows the robot displacement for scalar gain 0.5 during 7 iterations. The modifications made by the learning algorithm to the plant input are of a magnitude large enough to cause the displacement to oscillate between two modes of operation. In the first mode, the displacement in the positive direction is too large, while in the second mode, the displacement in the negative direction is too large. As the displacement

amplitude increases at each iteration, the required displacement eventually exceeds the travel limits of the robot and the control system automatically shuts down.

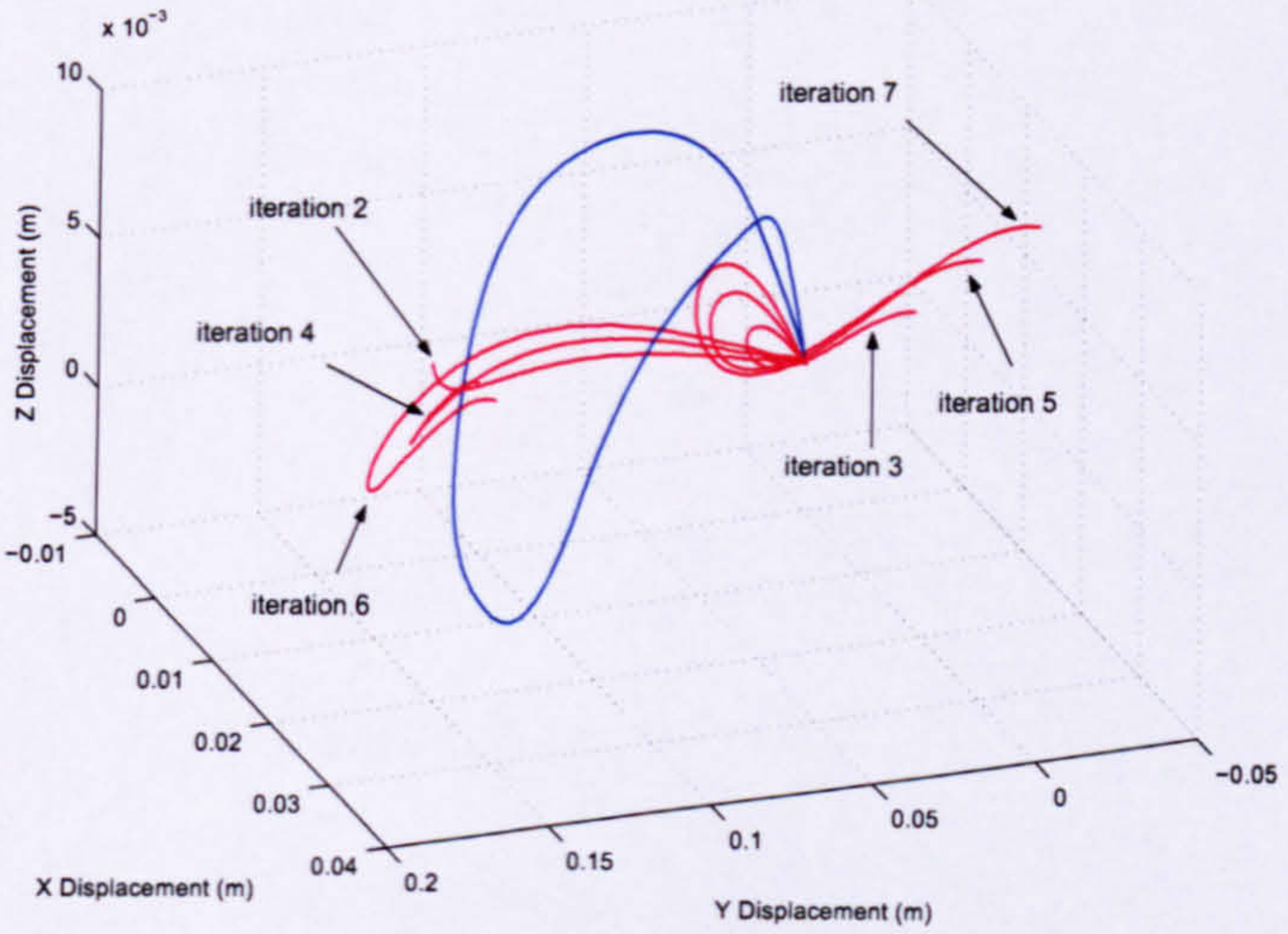


FIGURE 5.10: 3-Dimensional tracking progression (adjoint ILC, high order models, gain = 0.5)

The instability observed with scalar gain 0.5, is due to the increased magnitude of learning gain ϵ_{k+1}. The tuning parameter ω has been designed into the algorithm, specifically to reduce the magnitude of ϵ_{k+1} and produce a controller which is more robust. Tuning parameter ω can therefore be used to compensate for the scalar gain of 0.5. A trial and improvement method was used to tune ω and produce a stable controller. Stability was achieved for any value of ω larger than 5×10^{-8}, but as ω increased, the minimum recorded mse level also increased. The required value of ω is dependent on both the plant matrices G and the error vector for the first iteration.

Figure 5.11 displays the displacement profiles for iterations 490 to 500, with scalar gain 0.5 and ω set to 5×10^{-8} (corresponding Y and Z-axis figures can be found in Appendix B). The tracking error at most sample instants is substantial, yet the correction made by the algorithm at each iteration is minimal. This is due to the fact that a fixed value is used for the tuning parameter. A value of ω equal to 5×10^{-8} is the minimum required to maintain stability during the initial stages of learning. However, as the error decreases, the artificially reduced learning gain ϵ_{k+1} excessively limits the amount of information which can be learnt at each iteration and the convergence rate is slowed.

To restore a high convergence rate, while maintaining stability, the magnitude of ω must adapt to suit the current level of tracking error. As the error reduces, ω must also be made smaller. A suitable function is of the form:

$$\omega_{k+1} = \omega_0 + \omega_1 ||\mathbf{e}||^2 \tag{5.10}$$

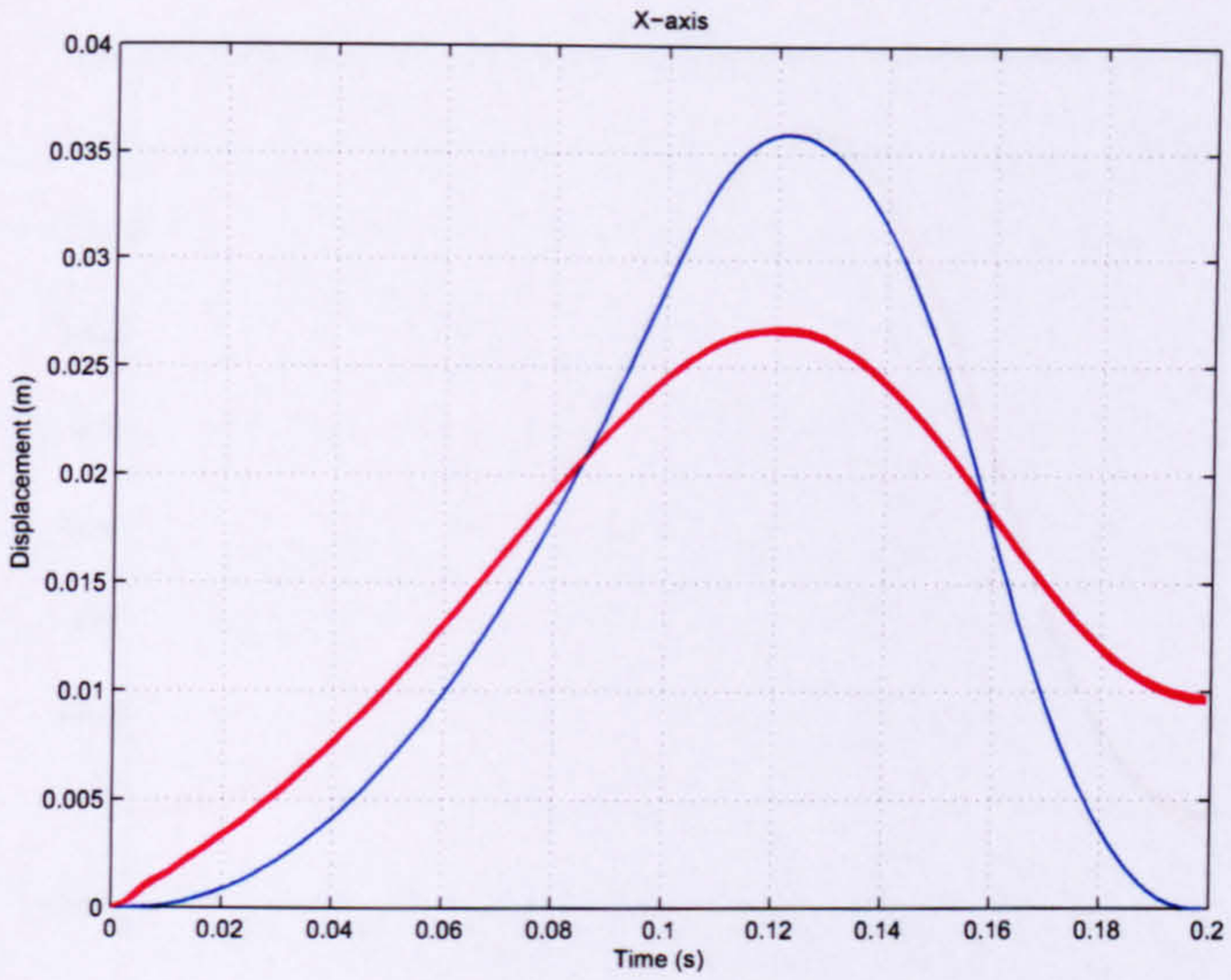


FIGURE 5.11: X-axis tracking performance iterations 490-500 (adjoint ILC, $\omega = 5 \times 10^{-8}$, high order models, gain = 0.5,)

where ω_0 and ω_1 are two new tuning parameters, which must be appropriately selected to match the operation of the control system. Parameter ω_0 can be used to guarantee stability by setting a baseline magnitude for ω_{k+1}, while $\omega_1||\mathbf{e}||^2$ adapts ω_{k+1} to match the change in tracking error.

Figure 5.12 displays the displacement profiles for iterations 490-500 when the scalar gain is 0.5 and $\omega_{k+1} = 2 \times 10^{-7}||\mathbf{e}||^2$ (corresponding Y and Z-axis figures can be found in Appendix B). The adaptive variant of the gain tuning parameter has clearly improved the convergence rate and minimum tracking error, though the Z-axis performance remains poor in comparison to the other axes. Table 5.4 presents the PI_{100} values for various scalar gains and tuning parameter settings. The poor performance of the Z-axis can be corrected by setting $\omega_{k+1} = 0$. The required Z-axis travel is sufficiently small, that the initial overshoot caused by the increased learning gain does not exceed axis travel limits. By iteration 100, the output displacement resembles the reference trajectory well.

TABLE 5.4: Adjoint algorithm, gain modelling error, PI_{100} values

Scalar gain	ω	X-axis	Y-axis	Z-axis
1.5	0	2.182499	1.982648	4.650866
1.25	0	2.088590	2.315949	3.045411
0.75	0	3.035958	2.454343	6.282407
0.5	0	100.000000	100.000000	100.000000
0.5	5×10^{-8}	21.685481	11.737366	69.209067
0.5	$2 \times 10^{-7}\|\|\mathbf{e}\|\|^2$	5.640472	3.309684	39.097583
0.5	$\omega_z = 0$	–	–	11.132384

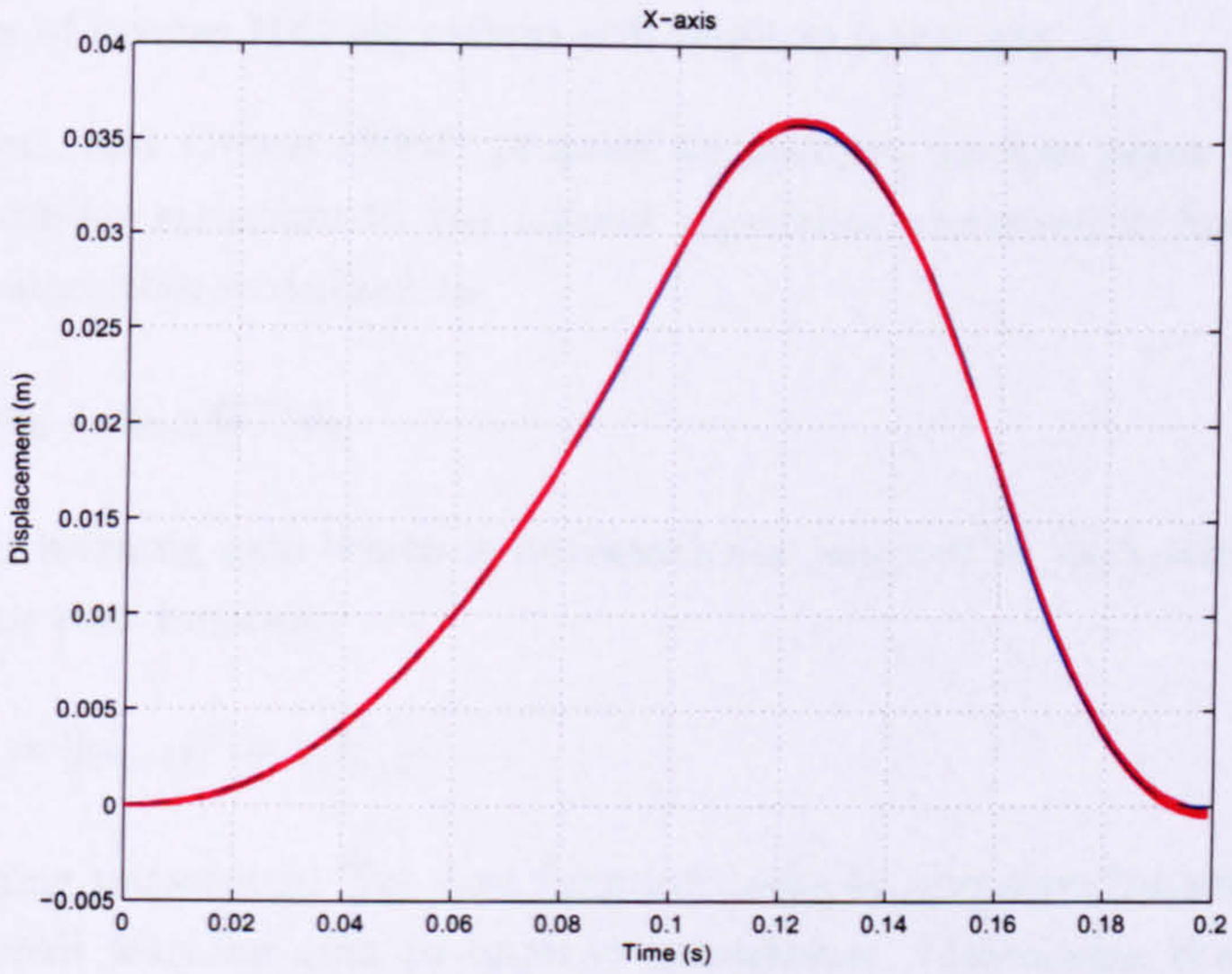


FIGURE 5.12: X-axis tracking performance iterations 490-500 (adjoint ILC, $\omega_{k+1} = 2 \times 10^{-7}||\mathbf{e}||^2$, high order models, gain = 0.5,)

5.3 Inverse Algorithm

5.3.1 Algorithm development

Inverse model based systems utilise the fundamental equation:

$$\mathbf{y} = \mathbf{G}\mathbf{u} \tag{5.11}$$

Assuming that a plant inverse can be calculated, Equation 5.11 can be rearranged in the form:

$$\mathbf{u} = \mathbf{G}^{-1}\mathbf{y} \tag{5.12}$$

If the plant dynamics and the desired output are known, Equation 5.12 theoretically generates the required input signal, which must be supplied to the plant. In effect there is no requirement for any form of controller, the perfect input signal is calculated in advance. However, in practical implementation, the use of this concept invariably leads to a system which significantly lacks robustness. The tracking performance also depends ultimately on the accuracy of the model, which frequently lacks detailed information with respect to plant dynamics and in particular, non-linear effects. These two factors are critically important to any control system and invariably prevent the practical implementation of Equation 5.12.

The ILC framework generates the opportunity to overcome these problems, by adapting the calculated input signal $\mathbf{u}$ at each iteration, to overcome modelling errors. However,

the robustness of inverse ILC algorithms still requires investigation.

Harte, Hätönen, and Owens (2004) propose an optimal, inverse plant ILC algorithm, which has a similar structure to the adjoint algorithm presented in Section 5.2. The input update algorithm is defined as:

$$\mathbf{u}_{k+1} = \mathbf{u}_k + \beta_{k+1}\mathbf{G}^{-1}\mathbf{e}_k \tag{5.13}$$

where β is the learning gain which is automatically selected at each iteration, by minimisation of the cost function:

$$J(\beta_{k+1}) = ||\mathbf{e}_{k+1}||^2 + \omega\beta_{k+1}^2 \tag{5.14}$$

and ω is a tuning parameter. The cost function seeks to minimise tracking error, while maintaining small learning gain to improve robustness. Minimising the cost function yields the gain update equation defined by:

$$\beta_{k+1} = \frac{||\mathbf{e}_k||^2}{\omega + ||\mathbf{e}_k||^2} \tag{5.15}$$

5.3.2 Initial implementation

The first implementation of the inverse algorithm was performed with ω equal to zero, therefore producing a maximum learning gain of 1. The test ran for 4 iterations, then was stopped due to excessive mechanical vibration in the structure of the robot. Analysis of the plant inputs and tracking errors (Figure 5.13 for the X-axis and associated Figures in Appendix B for the Y and Z-axes) reveals that the input voltages are contaminated with high frequency noise. Frequency spectrum analysis determines that the X-axis plant input contains a particulary distinct 25 Hz (157 rad/s) frequency component, while the X-axis error consists mainly of frequencies between 0 and 12 Hz (0-75 rad/s). Referring back to the X-axis Bode plot (in Section 3.3) this corresponds to a distinct dip in the gain plot, caused by a resonant zero pair, at which frequency the plant cannot resonate.

Observing the input update algorithm (Equation 5.13) reveals that the plant inverse model acts as a filter for the error vector. The frequency response characteristics of this filter are equivalent to the inverse of the Bode plot, therefore the anti-resonance becomes a strong resonant peak and higher frequencies are allowed to pass more easily than low frequencies. At each iteration, the X-axis filter enhances 25 Hz frequencies, which are added to the input for the next iteration. The plant cannot resonate at 25 Hz, but the high frequency components of the input provide sufficient excitation energy to stimulate the plant resonant frequencies between 10 and 12 Hz. The mechanism which drives the resonant frequency is similar to that found in basic algorithms in Chapter 4, the input becomes saturated with noise, the plant resonates and the control system is unusable.

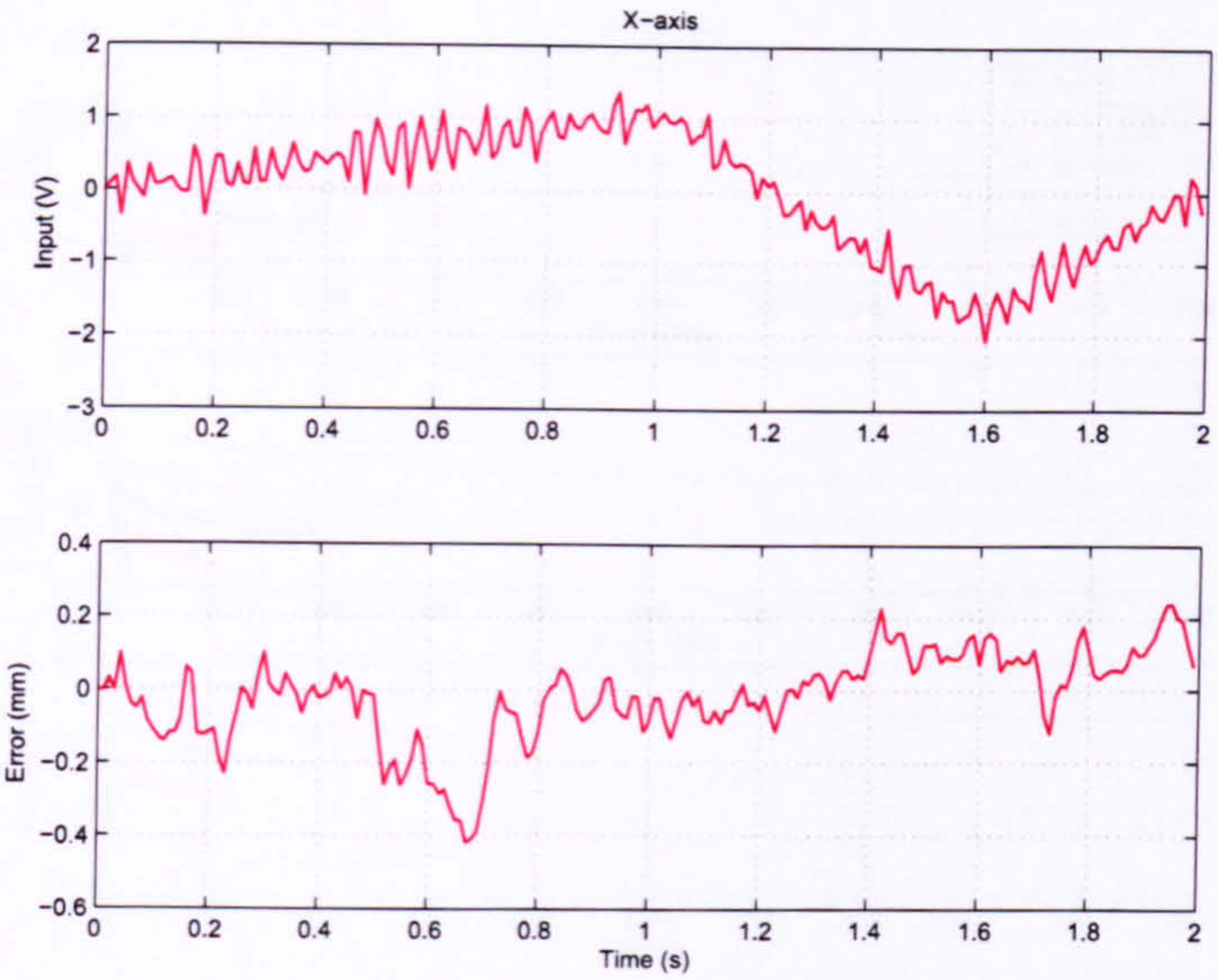


FIGURE 5.13: X-axis input and error (inverse ILC, $\omega = 0$, iteration 4)

The use of ω for stabilising the plant has been investigated. By reducing the learning gain, there is less amplification of high frequency signals and the plant input is less corrupted at each iteration (Ratcliffe, Harte, Hätönen, Lewin, Rogers, and Owens, 2004a).

The mse curves in Figure 5.14 represent the learning performance, when using ω values of 0, 0.0001, 0.001, 0.01, 0.1 and 1. Although the controller is unusable after 4 iterations with ω equal to zero, the advantage of using the inverse model is immediately noticeable. The error reduction rate is very fast and the mse is reduced by 4 orders of magnitude in just 3 iterations. Increasing the magnitude of ω extends the time before the system becomes unusable, but at the expense of convergence rate and minimum mse. Fundamentally, trying to use ω as a means of stabilisation is simply delaying the build up of high frequency signals, not preventing it.

5.3.3 Stabilisation using zero-phase filtering

Zero-phase filtering was successfully implemented in Section 4.4.6 to stabilise the basic P-type algorithm, by removing high frequency noise and resonances. This technique is equally well suited to the instability which exists in the inverse algorithm (Ratcliffe, Rogers, Harte, Hätönen, Lewin, and Owens, 2004b). The low pass transfer function (Equation 4.15) is modified to match the 100Hz sample frequency and is implemented using the zero-phase methodology, on the plant input vector generated by the inverse

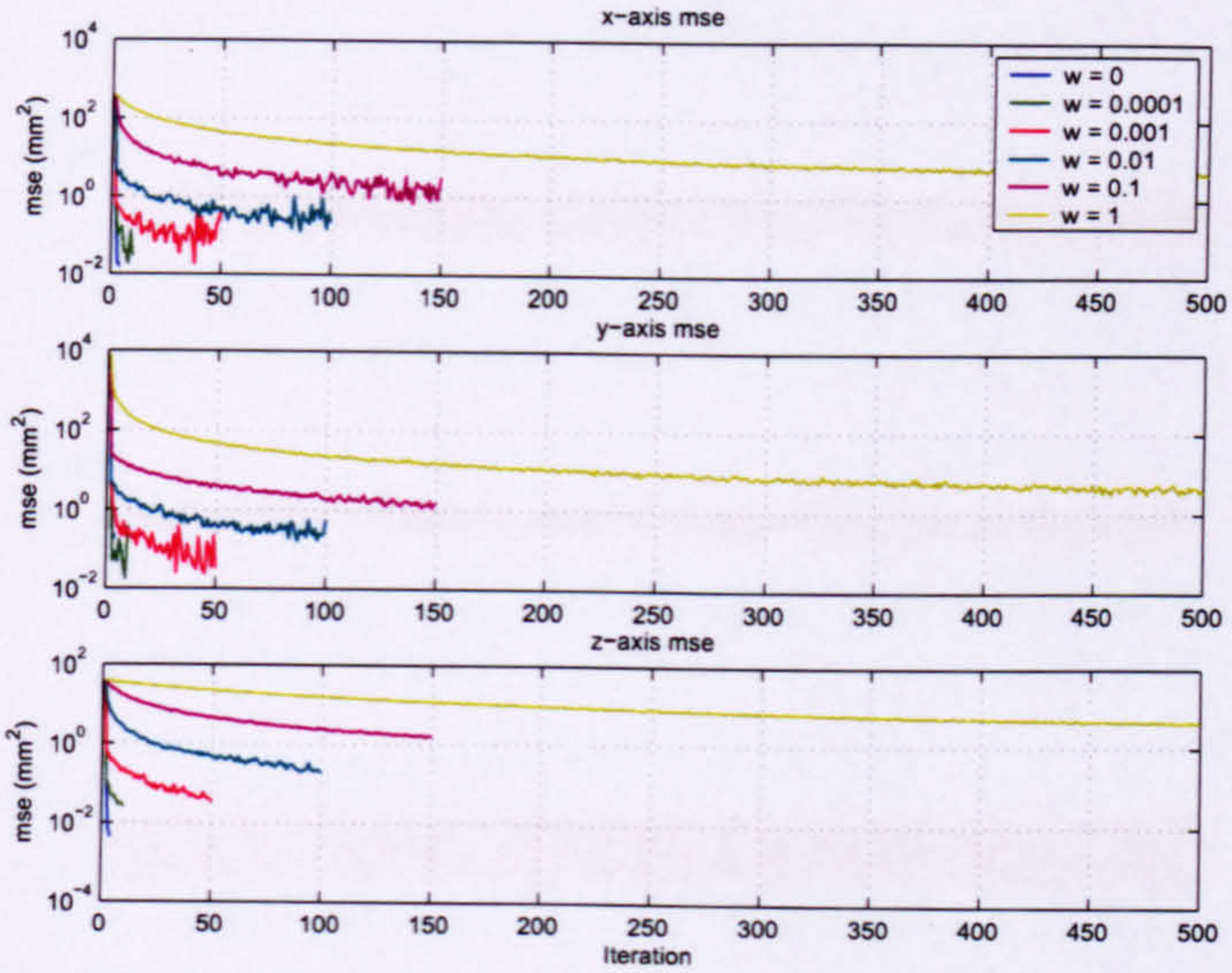


FIGURE 5.14: mse (inverse ILC, high order models)

algorithm. The filter transfer function is:

$$\frac{y(z)}{u(z)} = \frac{0.102693 + 0.002934z^{-1} + 0.002934z^{-2} + 0.102693z^{-3}}{1 - 1.644597z^{-1} + 1.091881z^{-2} - 0.236029z^{-3}} \tag{5.16}$$

Figure 5.15 shows the mse curves recorded during a 5000 iteration long-term stability test, implemented with the zero-phase filter and with ω equal to zero. The filtering technique has successfully stabilised the algorithm, which is able to reduce the mse by 5 orders of magnitude within three iterations. The absence of increasing mse suggests that the algorithm will remain stable for many further iterations. The PI_{100} values are presented in Table 5.5.

TABLE 5.5: Inverse ILC, long-term stability test PI_{100} values

X-axis	Y-axis	Z-axis
1.003734	1.000653	1.011324

In a similar manner to the adjoint algorithm, the mse for consecutive iterations varies between upper and lower bounds. This is due to the purely feed-forward nature of the algorithm and the high gain of the inverse model which amplifies the effect of non-repeating disturbances. Figure 5.16 confirms that the displacement profiles at iteration 5000 are free of resonances and match the reference trajectory well.

The rapidity of convergence can be seen in Figure 5.17, which displays the displacement profiles for iterations 1 to 3. Iteration 1 is a point centered at the origin, while iteration

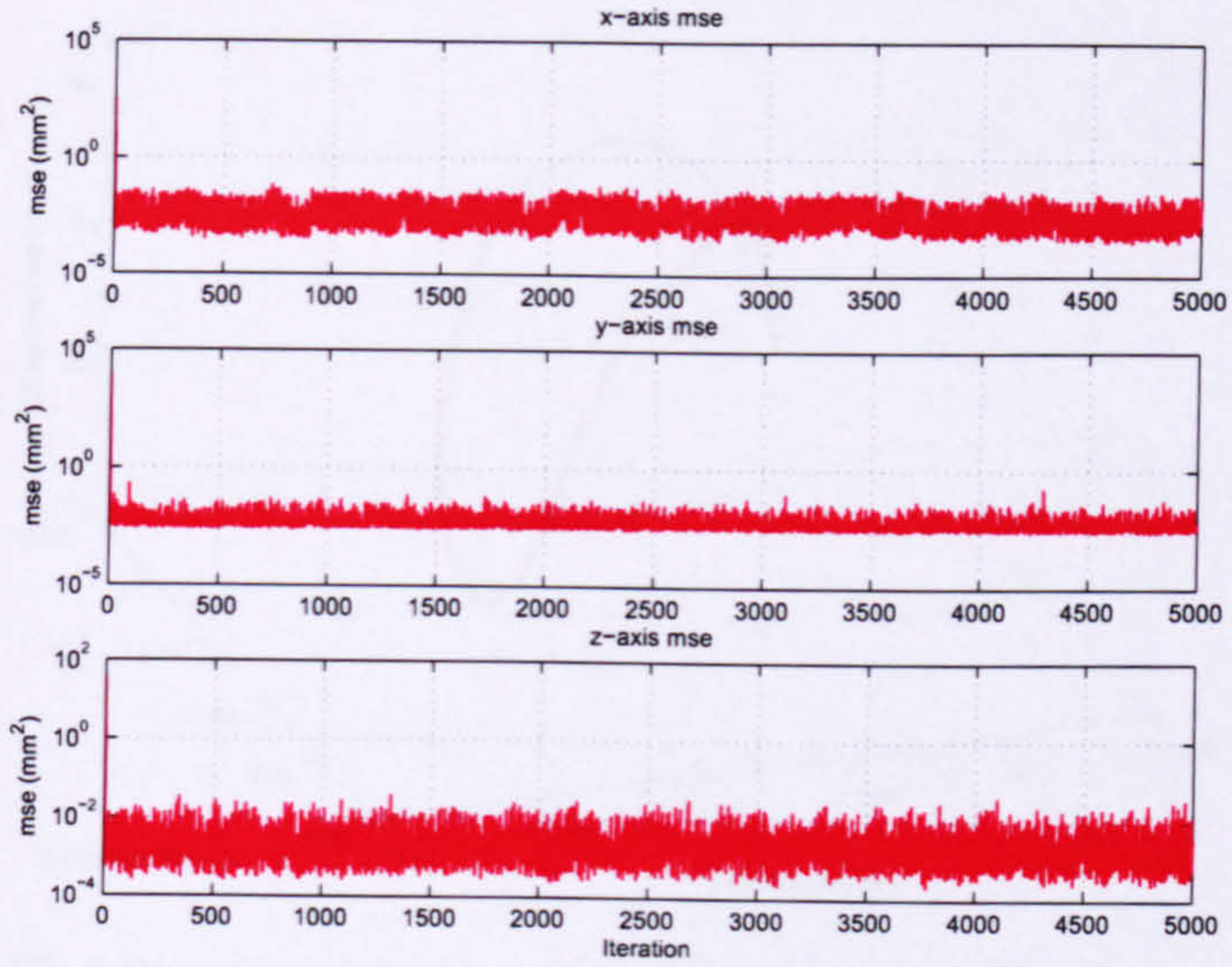


FIGURE 5.15: mse (inverse ILC, zero-phase filter, high order models)

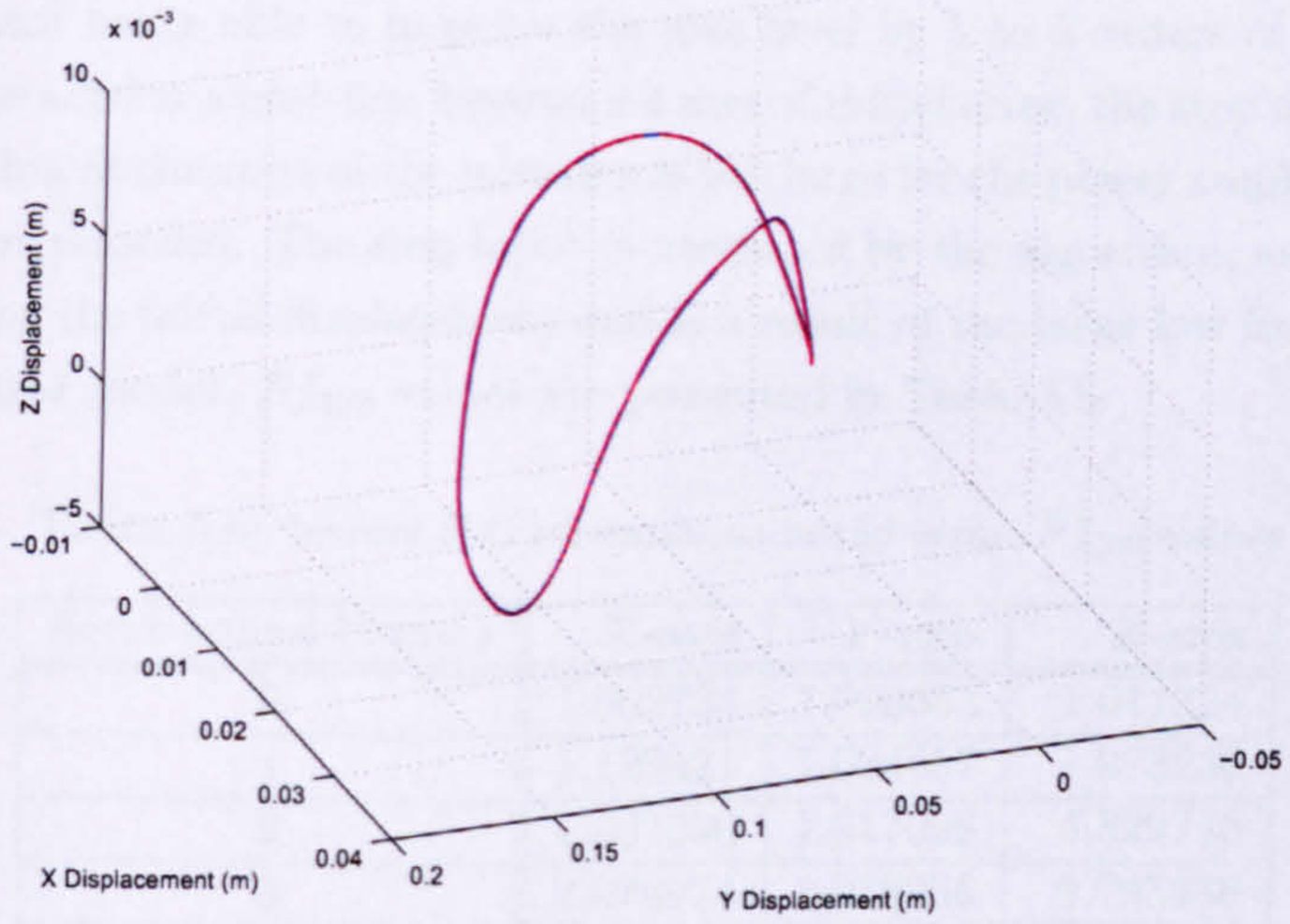


FIGURE 5.16: 3-Dimensional tracking performance (inverse ILC, zero-phase filter, high order models, iteration 5000, reference = blue)

2 resembles the reference trajectory closely and by iteration 3 the tracking is equivalent to that achieved during iteration 5000 in Figure 5.16.

5.3.4 Robustness to initial state error

Figure 5.18 displays the mse curves corresponding to initial errors of 0, 1, 2, 3, and 4mm. Within the period of 500 iterations, the stability of the inverse algorithm is not compromised by the presence of initial error. There is no indication that the mse tends to increase after the initial reduction. The algorithm can successfully tolerate ±4 mm of

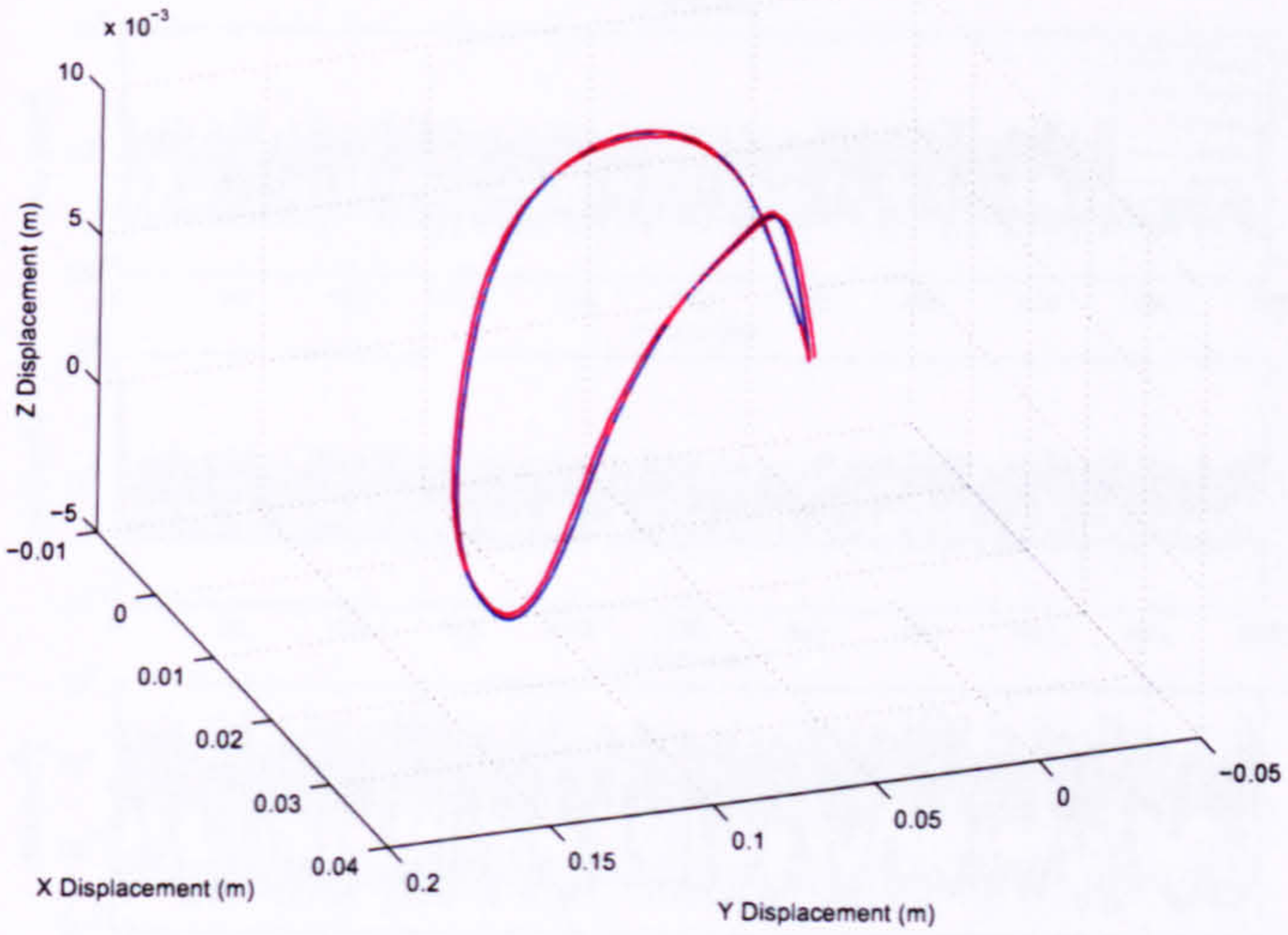


FIGURE 5.17: 3-Dimensional tracking progression, first 3 iterations (inverse ILC, high order models, reference = blue)

error, while still being able to improve the mse level by 1 to 2 orders of magnitude. In contrast to the adjoint algorithm, beyond ±4 mm of initial error, the step input calculated by the algorithm at the start of the iteration is too large for the power amplifiers, therefore no data can be recorded. The step input is produced by the algorithm, as it attempts to compensate for the initial displacement and is a result of the large low frequency gain of the inverse plant model. PI_{100} values are presented in Table 5.6.

TABLE 5.6: Inverse ILC tolerance to initial error, PI_{100} values

Error bound (±mm)	X-axis	Y-axis	Z-axis
0	1.003734	1.000653	1.011324
1	1.120421	1.004857	1.973230
2	1.497030	1.017098	4.829746
3	2.076974	1.038256	9.795388
4	3.051135	1.067486	16.522295

The linear representation of the mse for the first 50 iterations (Figure 5.19), highlights the increased variation in mse for each increment of initial error, but, also emphasizes the lack of change in convergence rate, minimum mse still being achieved in 2 to 3 iterations, irrespective of the initial error.

5.3.5 Robustness to plant modelling error

The effect of model gain error is investigated by multiplying the high order models with scalar gains of 0.5, 0.75, 1.25 and 1.5. The inverse algorithm is unusable for gains 0.5 and 0.75, because the axis displacement required during iteration 2 is significantly larger

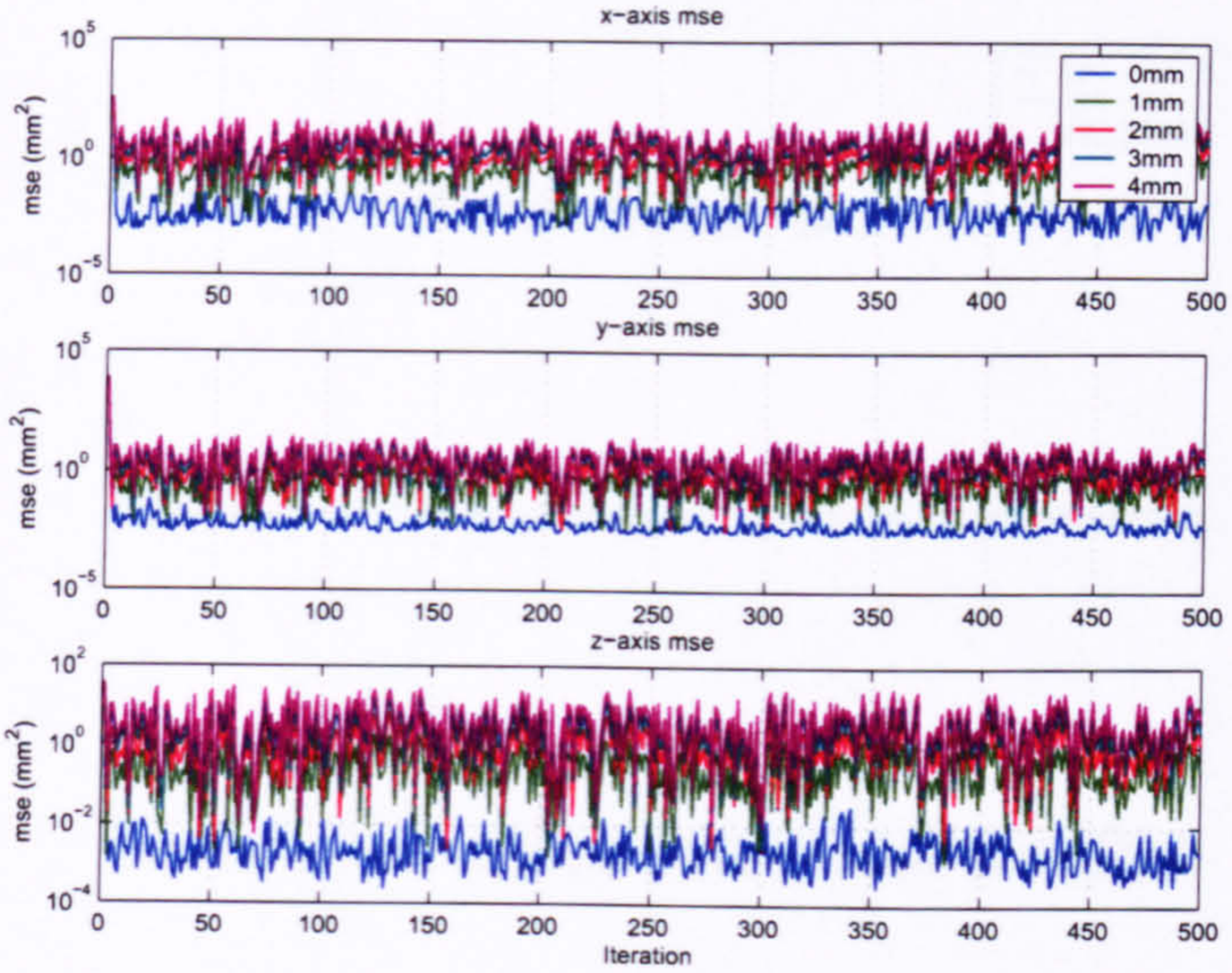


FIGURE 5.18: mse (inverse ILC, high order models, initial error bounds 0, 1, 2, 3 and 4 mm)

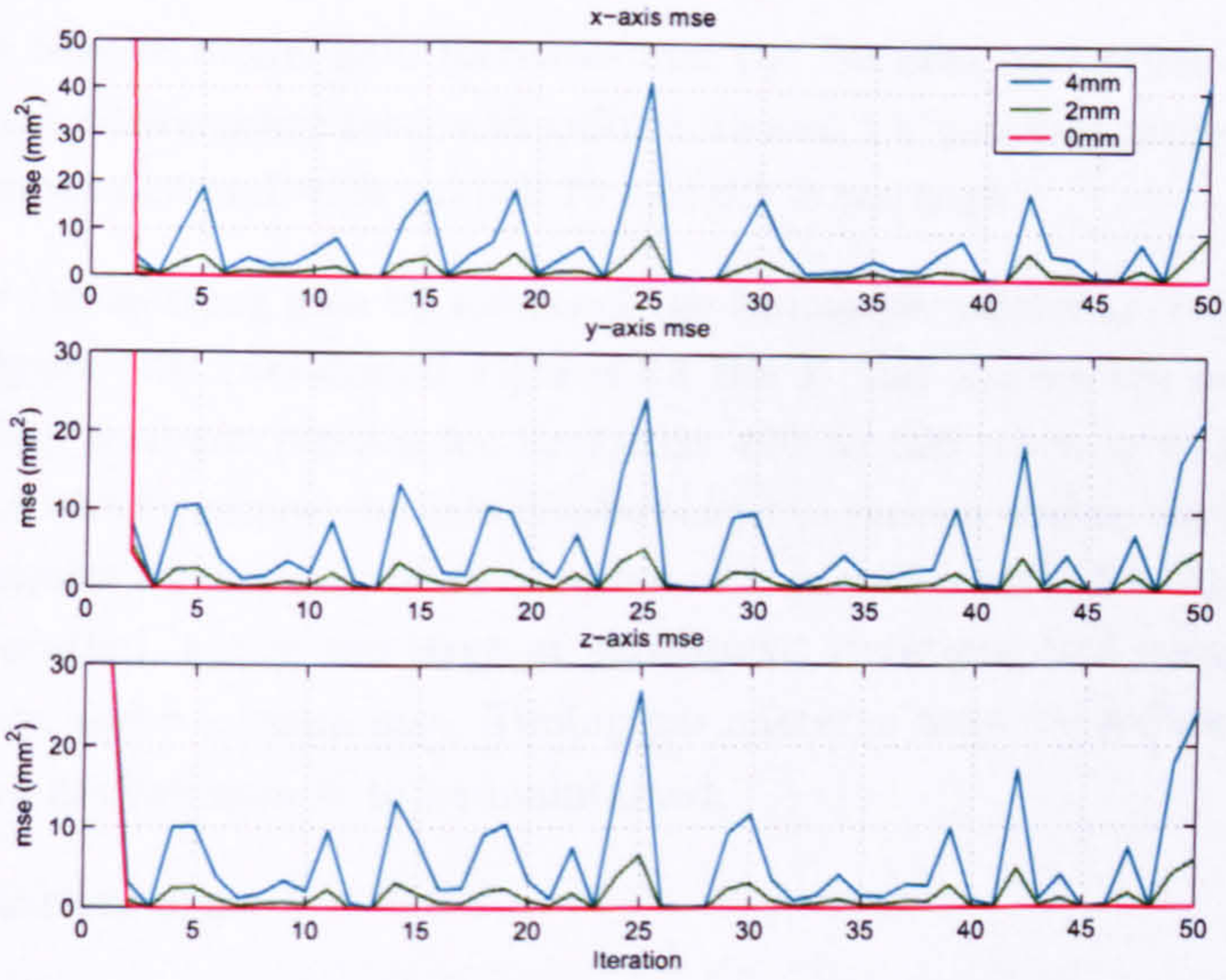


FIGURE 5.19: mse linear scale (inverse ILC, high order models, initial error bounds 0, 2 and 4 mm)

than the travel limits of the robot. Gains of 1.25 and 1.5 produce very similar results to the original gain of 1, as shown in Figure 5.20. There is no noticeable effect on stability or minimum mse. However, each increase in scalar gain tends to slow the convergence rate slightly, achieving minimum mse 3 to 4 iterations later.

With reference to the input update algorithm previously mentioned (Equation 5.13), the error vector is multiplied by $\mathbf{G}^{-1}$. If the the scalar gain is increased, the inverse model gain decreases and the learning correction is smaller. Conversely, if the scalar gain is

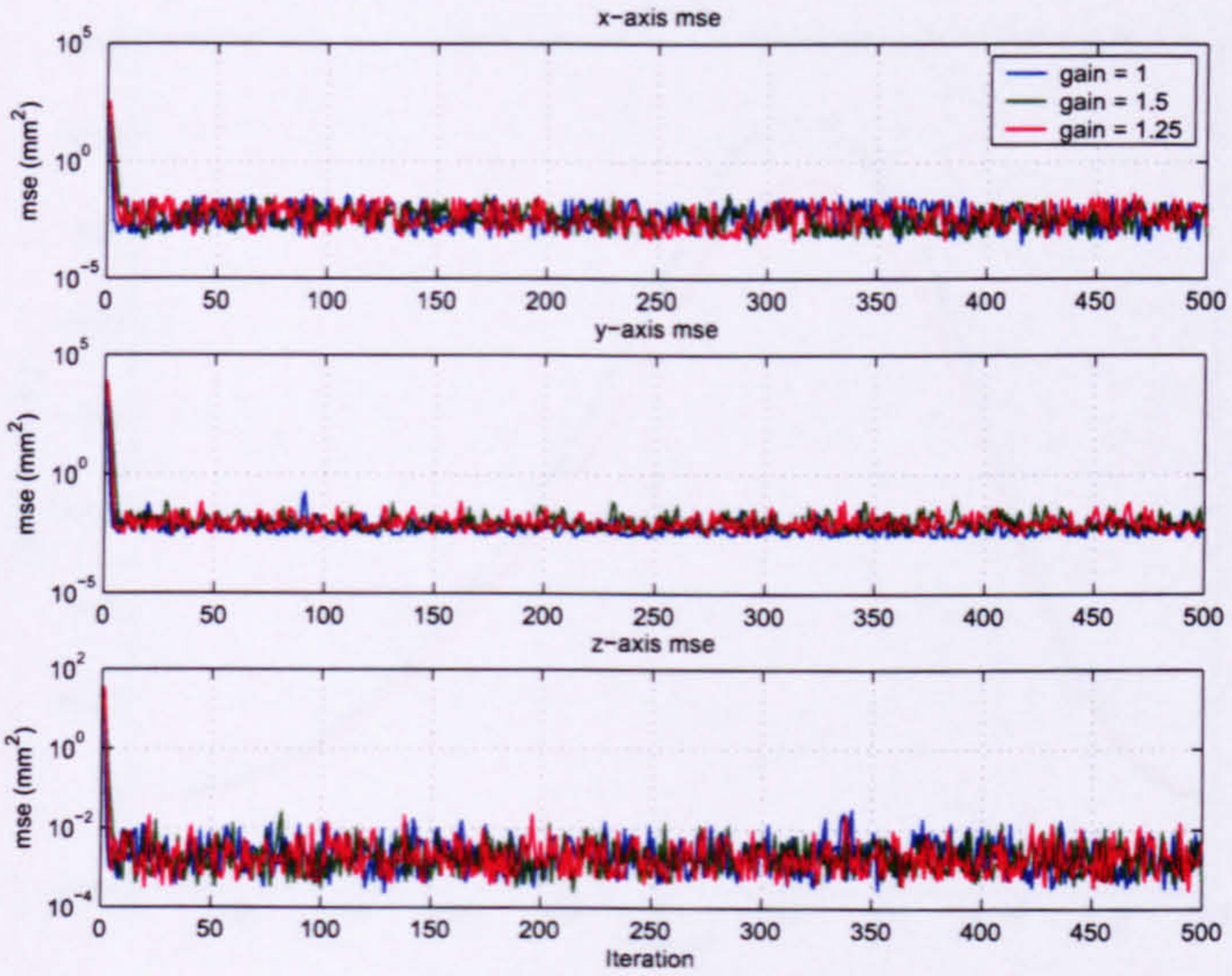


FIGURE 5.20: mse for different scalar gains (inverse ILC, high order models, gains 1, 1.5 and 1.25)

decreased, the inverse model gain increases and the learning correction is larger. This explains why the convergence rate with gains 1.25 and 1.5 becomes progressively slower and why the input demand with gains 0.75 and 0.5 is too high.

Adjustment of the learning gain by means of the tuning parameter ω corrects the scalar gain error. Figure 5.21 (associated Figures for the Y and Z-axes are in Appendix B) displays the displacement profiles for iterations 490 to 500 when ω is set to 1.4, the minimum value which permits the axis displacement to remain within travel limits. Similarly to the adjoint algorithm, the fixed value of ω is well matched to the error profile for the first iteration, but is too large at subsequent iterations and compromises both convergence rate and minimum mse. Tuning parameter ω must be reduced as the error reduces, if good performance is to be maintained.

The adaptive variant of ω:

$$\omega_{k+1} = \omega_0 + \omega_1 ||\mathbf{e}||^2 \tag{5.17}$$

is equally well suited to the inverse algorithm and when $\omega_0 = 0$ and $\omega_1 = 1$, the minimum mse and convergence rate are improved as demonstrated in Figure 5.22 (see Appendix B for Y and Z-axes), for which the model scalar gain is 0.5. $\omega_1 = 1$ is particularly well tuned for this experiment, because with reference to the gain update (Equation 5.15), the learning gain β_{k+1} is always equal to 0.5 and compensates exactly for the incorrect inverse model gain, which is 2. Table 5.7 presents the PI_{100} values for all tests involving scalar modelling error.

The effects of model order reduction on the inverse algorithm performance are minimal.

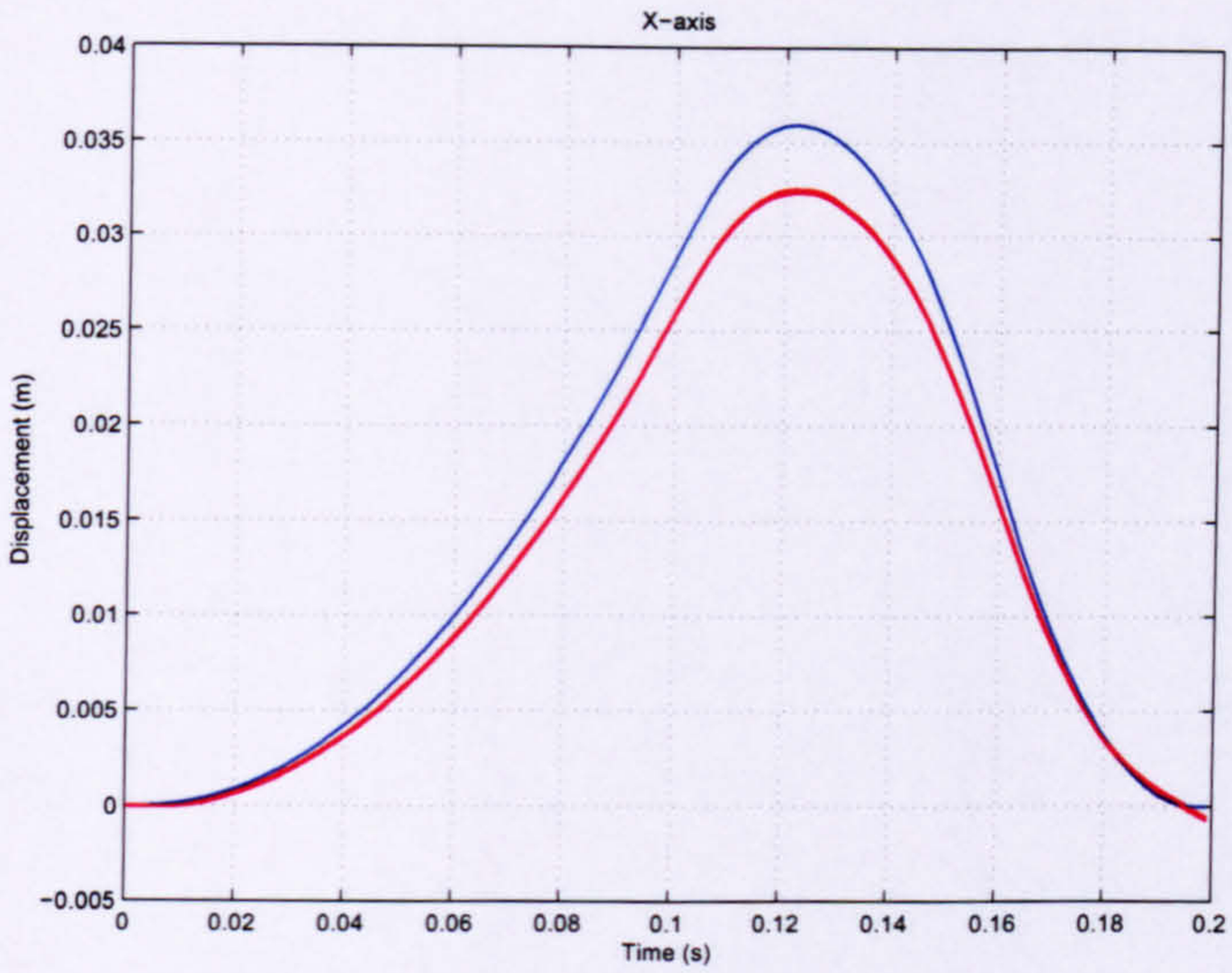


FIGURE 5.21: X-axis tracking performance iterations 490-500 (inverse ILC, zero-phase filter, $\omega = 1.4$, high order models, gain $= 0.5$)

TABLE 5.7: Inverse ILC, tolerance to scalar gain error, PI_{100} values

Gain	ω	X-axis	Y-axis	Z-axis
1.5	0	1.118844	1.128620	1.127984
1.25	0	1.040381	1.041540	1.054823
0.75	0	100.000000	100.000000	100.000000
0.75	0.4	7.825125	1.222498	36.441080
0.75	$1\|\|\mathbf{e}\|\|^2$	1.117947	1.126110	1.139603
0.5	0	100.000000	100.000000	100.000000
0.5	1.4	14.154051	1.448858	53.769616
0.5	$1\|\|\mathbf{e}\|\|^2$	1.005563	1.000994	1.011805

The mse reduction compares well with the higher order models and the error converges within 4 to 5 iterations. Convergence rate and minimum tracking error and have not been significantly affected and stability is still maintained. The inverse algorithm is implemented for 500 iterations, using the zero-phase filter and the 1st order models. Associated mse plots can be found in Figure 5.23. Note that the Y-axis tuning parameter must be set to $0.1||\mathbf{e}||^2$, to limit the Y-axis learning gain and prevent the robot from exceeding travel limits during the first iteration. This is due to a low frequency gain discrepancy between the actual plant and the first order model, which is greatly amplified by inverting the model. PI_{100} values for the first order model implementation are presented in Table 5.8.

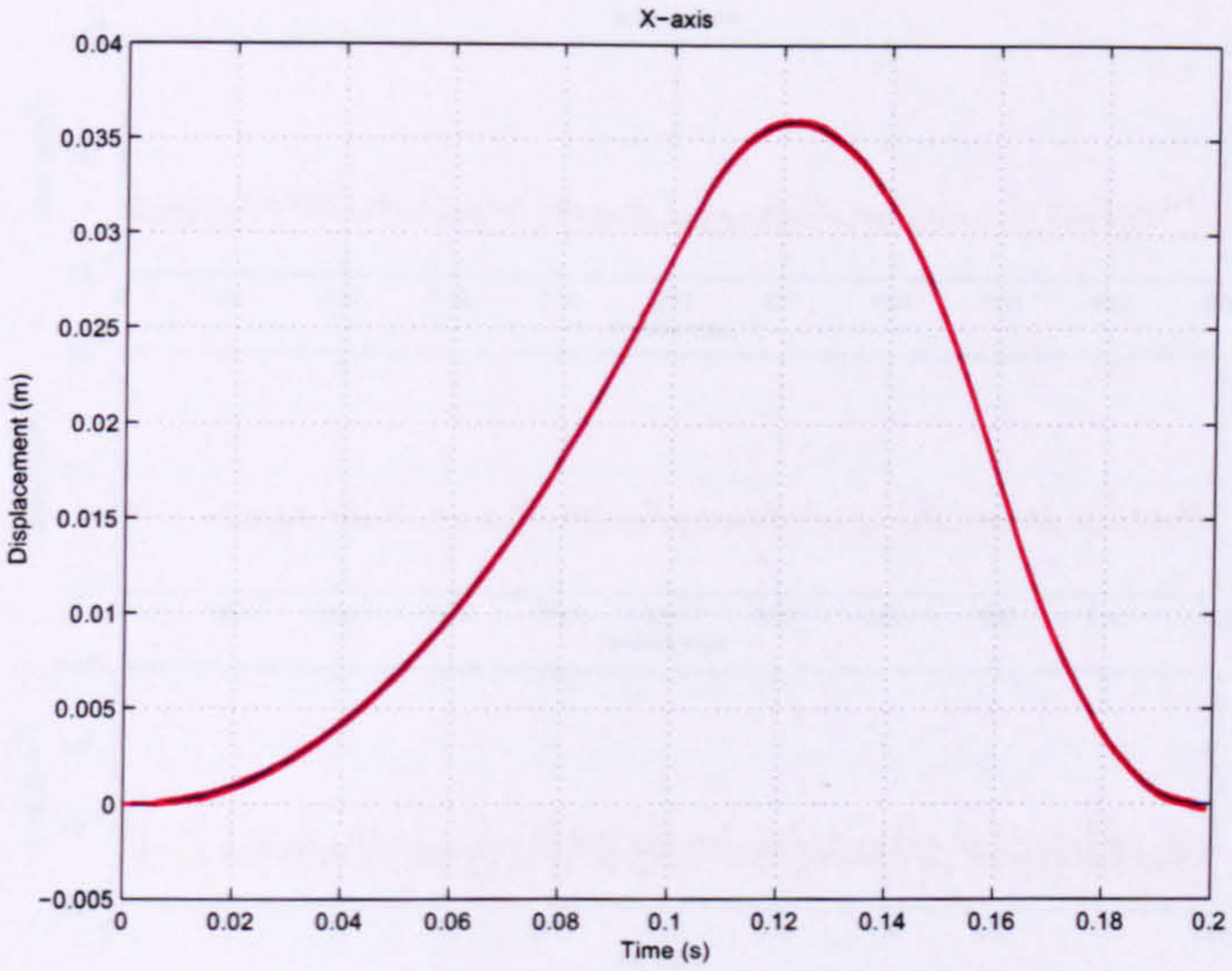


FIGURE 5.22: X-axis tracking performance iterations 490-500 (inverse ILC, zero-phase filter, $\omega = 1||\mathbf{e}||^2$, high order models, gain = 0.5)

TABLE 5.8: Inverse ILC 1st order models, PI_{100} values

X-axis	Y-axis	Z-axis
1.013886	1.002984	1.030281

5.4 Norm-Optimal Algorithm

5.4.1 Algorithm development

The fundamental concepts underlying the discrete norm-optimal (NOILC) algorithm are presented in Section 2.2.3.6. The optimised input update algorithm has a very similar structure to the adjoint algorithm:

$$\mathbf{u}_{k+1} = \mathbf{u}_k + R^{-1}\mathbf{G}^T Q\mathbf{e}_{k+1} \tag{5.18}$$

Q and R are matrices, which adjust the balance between the rate of error reduction and the rate of input signal change respectively. In general, the components of Q and R are chosen by:

$$Q = qI \tag{5.19}$$

$$R = rI \tag{5.20}$$

where q and r are scalars and I is the identity matrix of appropriate dimensions.

The distinguishing feature of the norm-optimal algorithm, is that future tracking error

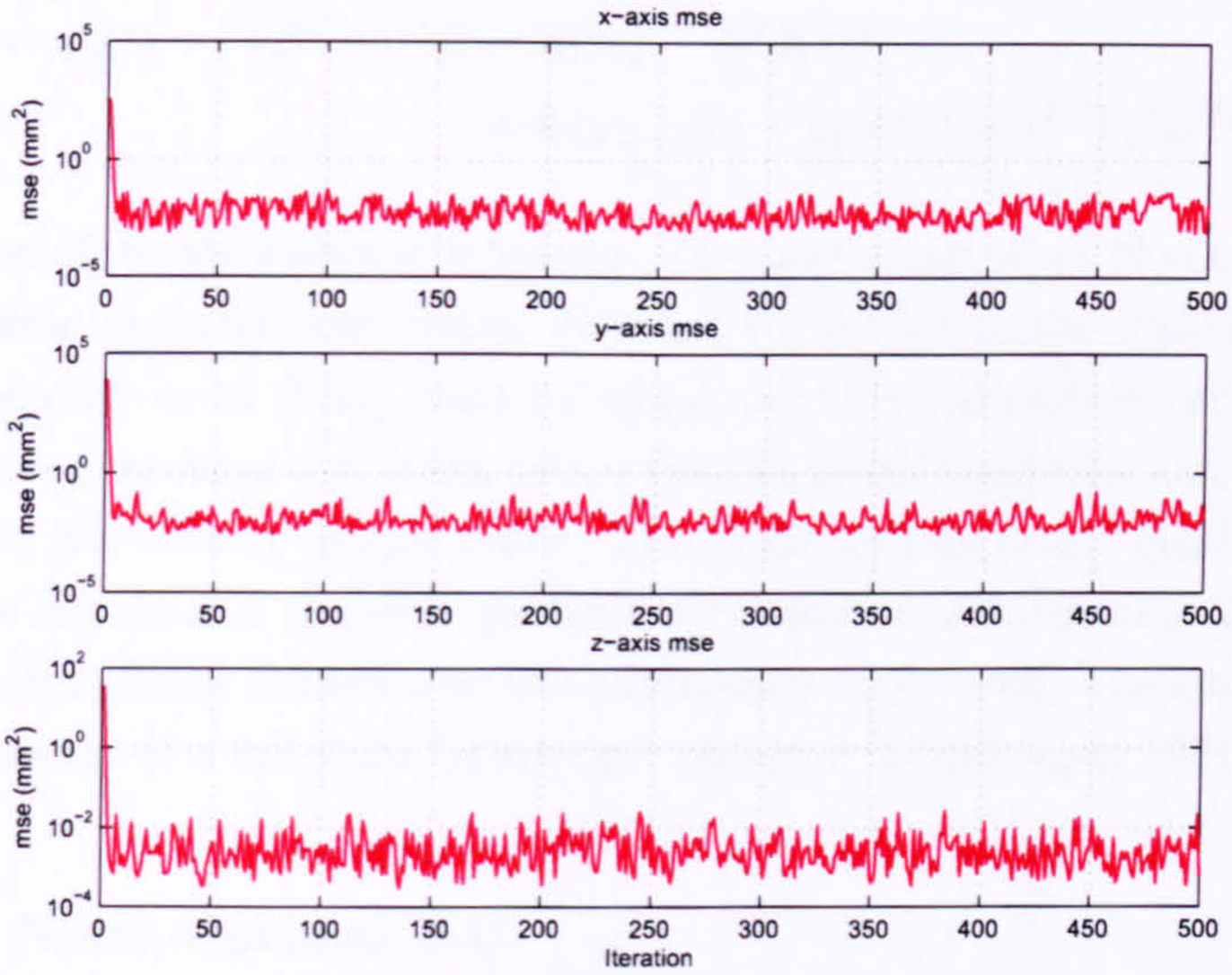


FIGURE 5.23: mse (inverse ILC, 1st order models, zero-phase filter, $\omega_y = 0.1||\mathbf{e}||^2$, $\omega_{x,z} = 0$)

$\mathbf{e}_{k+1}$ is used rather than past error $\mathbf{e}_k$. This fundamentally makes the algorithm non-causal, resulting in a system which should be impossible to implement. However, Amann et al. (1996b) overcome non-causality, by means of an additional optimal feedback control formulation of the algorithm. The resulting controller generates both feed-forward control between iterations and feedback control at each sample instant and consists of three equations:

- Matrix gain equation

$$K(t) = A^T K(t+1)A + C^T Q(t+1)C - \left[A^T K(t+1)B \times \left\{B^T K(t+1)B + R(t+1)\right\}^{-1} B^T K(t+1)A\right] \quad (5.21)$$

where $K(t)$ is a matrix gain which has the terminal condition $K(N) = 0$ and is a solution of this Riccati equation.

- Predictive component equation

$$\xi_{k+1}(t) = \left\{I + K(t)BR^{-1}(t)B^T\right\}^{-1} \times \left\{A^T \xi_{k+1}(t+1) + C^T Q(t+1)e_k(t+1)\right\} \quad (5.22)$$

where $\xi_{k+1}(N) = 0$.

- Input update equation

$$u_{k+1}(t) = u_k(t) - \left[\left\{ B^T K(t) B + R(t) \right\}^{-1} B^T K(t) \right.$$
$$\left. \times A \left\{ x_{k+1}(t) - x_k(t) \right\} \right] + R^{-1}(t) B^T \xi_{k+1}(t) \quad (5.23)$$

Implementation of the algorithm is as follows. The matrix gain K (5.21) can be calculated before the system operates, and hence, does not contribute to the real-time processing load. The predictive term (5.22) must be calculated between each iteration. Note that this equation has a terminal condition, rather than an initial condition and must therefore be computed in descending sample order. The input update (5.23) must be calculated at each sample instant and therefore particularly contributes to the real-time processing load and has a significant influence on the minimum sample time. The system states are estimated by means of a full state Luenberger observer (Luenberger, 1971).

5.4.2 Fast Norm-Optimal ILC

Fast convergence speeds and small residual tracking error should make the NOILC algorithm very attractive to industry, because they result in shorter manufacturing time and less product wastage. However, in the original format, the NOILC algorithm requires a high performance controller, if it is to be implemented with a high order model at fast sampling frequency. This is mainly due to the large numbers of multiplications, additions, subtractions, matrix transpositions and matrix inversions which need to be performed between each sample interval. To remedy this problem, a fast version of the algorithm (F-NOILC) can be used, which allows the majority of calculations to be performed during the design and commissioning of the controller (Ratcliffe, van Duinkerken, Lewin, Rogers, Hätönen, Harte, and Owens, 2005b). The remaining calculations are significantly reduced in number and consist solely of multiplications, additions and subtractions. The F-NOILC algorithm is derived by identifying numerous simplifications which can be made to the NOILC algorithm. The matrix gain equation is already performed off-line in the NOILC implementation and there is no change for the F-NOILC.

Now consider the predictive component equation (5.22). The only variables in this equation are the tracking error e_k and the predictive term itself ξ_{k+1} all of the other terms can be combined together to produce constant matrices:

$$\alpha(t) = \left\{ I + K(t) B R^{-1}(t) B^T \right\}^{-1} \quad (5.24)$$

$$\beta(t) = \alpha(t) A^T \quad (5.25)$$

$$\gamma(t) = \alpha(t) C^T Q(t+1) \quad (5.26)$$

leading to the simplified predictive component equation:

$$\xi_{k+1}(t) = \beta(t)\xi_{k+1}(t+1) + \gamma(t)e_k(t+1) \tag{5.27}$$

Exactly the same concept can be applied to the input update equation (5.23):

$$\lambda(t) = \left\{B^T K(t) B + R(t)\right\}^{-1} B^T K(t) A \tag{5.28}$$

$$\omega(t) = R^{-1}(t) B^T \tag{5.29}$$

resulting in the simplified input update equation:

$$u_{k+1}(t) = u_k(t) - \lambda(t)\left\{x_{k+1}(t) - x_k(t)\right\} + \omega(t)\xi_{k+1}(t) \tag{5.30}$$

The resulting implementation therefore requires that seven matrices in total be supplied to the real-time controller.

- state matrices, A, B and C
- F-NOILC matrices, β, γ, λ and ω

If the matrices Q and R need to be adjusted, then the F-NOILC matrices must be recalculated and downloaded again to the controller.

It must be stated that the F-NOILC algorithm does use significantly more memory than the NOILC algorithm because the memory allocation is static rather than dynamic. The NOILC can recycle memory once calculations are complete. However it is worth observing that the process of recycling the memory takes time, and therefore decreases the amount of time available for computation of the algorithm. Currently, hardware constraints favour the F-NOILC algorithm because it is relatively easier and cheaper to upgrade memory than to upgrade the central processor of the control hardware.

With respect to the improvement in computation speed, due to the reduced number of calculations, it is possible to calculate exactly the time required to perform each algebraic operation for both the NOILC and the F-NOILC, then find the total time for each variant. However, the results of this process still ultimately depend on the characteristics of the controller, the operating system and the efficiency of the program functions (Machado and Galhano, 1995). In simulation studies using an identical setup for both variations of the algorithm, the F-NOILC algorithm can be calculated almost three times faster than the NOILC algorithm. Potentially, this implies that the algorithm can be implemented at three times the sampling frequency on an existing controller, or can be used at the current sample frequency on a controller which is three times slower and therefore likely to be cheaper.

5.4.3 Initial implementation

The matrices Q and R fundamentally determine the performance of the controller. Therefore, before any meaningful experiments relating to stability and performance were undertaken, the effects of adjusting Q and R had to be investigated. Experiments were performed in which Q and R were diagonal matrices defined by $Q = qI$ and $R = rI$. All the combinations of scalars q and r given in Table 5.9 were implemented for a period of 100 iterations, following which, PI_{100} was calculated for each test.

TABLE 5.9: q and r values used in experiments

q	0.1	1	10	100	1000	10000	100000	1000000
r	100	10	1	0.1	0.01	0.001	0.0001	

Because the algorithm has two tuning parameters, the results of these experiments are best displayed in three dimensional format, relating q and r to PI_{100}. Figure 5.24 shows the data for the X-axis. Results for the Y and Z-axes are in Appendix B. It is immediately noticeable that all three plots are similar, particularly for the X and Y axes, for which the low frequency gain of the linear motors is practically identical.

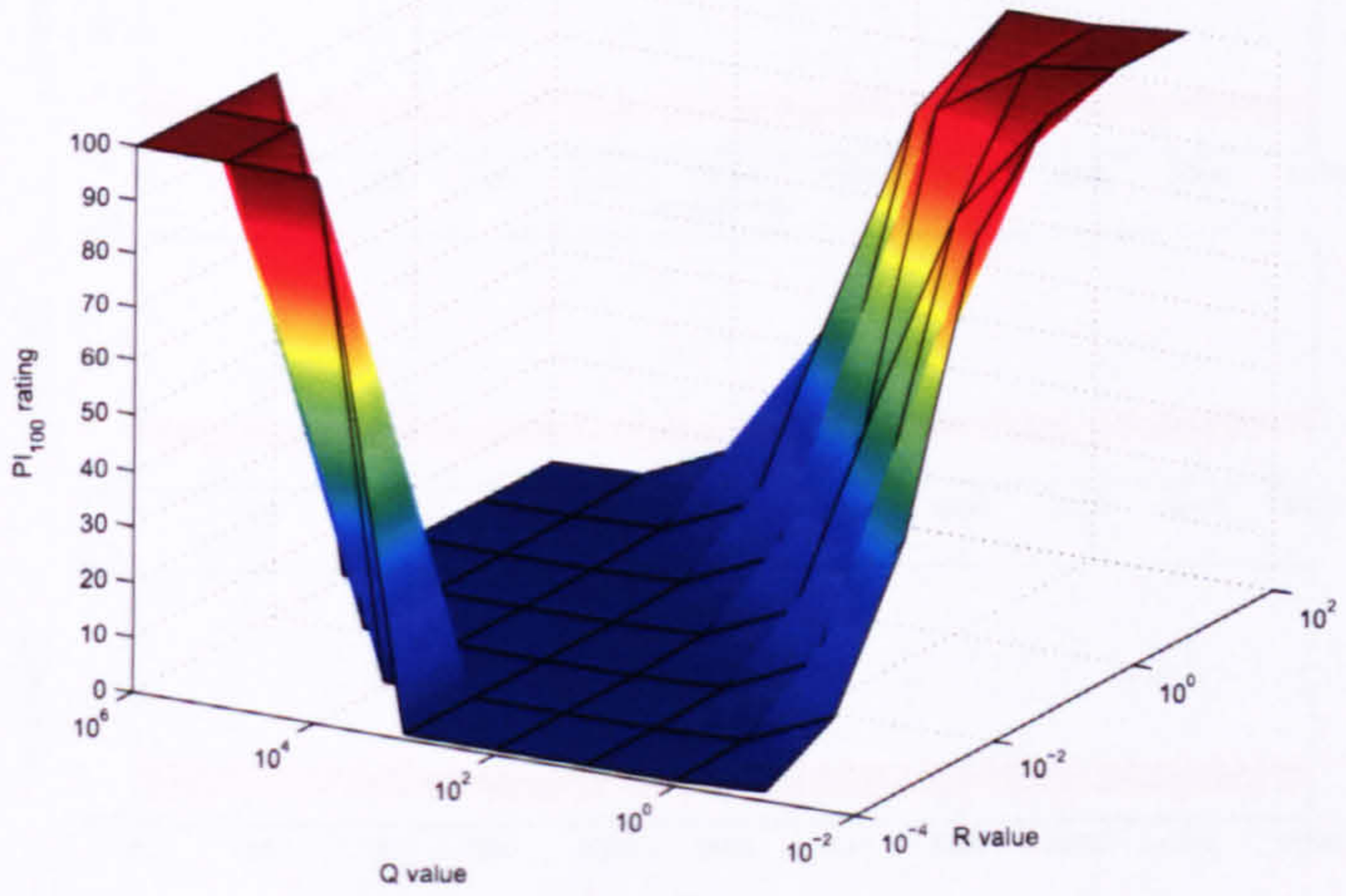


FIGURE 5.24: X-axis PI_{100} for various q and r

Because q affects the rate of error reduction and r limits the input change, interpreting the information in the plots becomes a simple task. To the right of the chart is a region of poor tracking performance where the PI_{100} value is near to or equal to 100 indicating that virtually nothing is learnt during the 100 iterations of the test. As could be expected, this corresponds to a small value for q and a large value for r. With these settings, the algorithm is far too conservative. As the ratio of q to r increases, gradually PI_{100} reduces, indicating that the performance is improving. This is represented by the slope

to the right side of the chart. As the q/r ratio continues to increase, PI_{100} is reduced to values very close to 1, indicating that the perfect trajectory is learnt in almost one iteration. The balance of error reduction to input change is now approaching optimality. Temporarily increasing q/r has little effect on the performance, until the system becomes unstable and PI_{100} reverts to 100. This is represented by the channel followed by the steep slope to the left of the chart. Therefore it is the ratio of q to r that determines controller performance, rather than the actual values for each parameter. If a larger range of q and r values were used, the chart would still present a channel cutting diagonally across it. q and r values of 100 and 0.001 respectively produce a suitable compromise between performance and stability, and were used in all further experiments.

Long term stability, low mse and fast convergence are all achieved by the norm-optimal algorithm, as demonstrated by the mse curves in Figure 5.25. As with the adjoint and inverse algorithms, the mse varies between upper and lower bounds. But, the variation is noticeably smaller, being only one order of magnitude rather than two. The likely cause for this reduction is the feedback component of the norm-optimal algorithm, which compensates for non-repeating disturbances and noise. The tracking performance at iteration 5000 (Figure 5.26) is better than the level achieved by the other model-based algorithms.

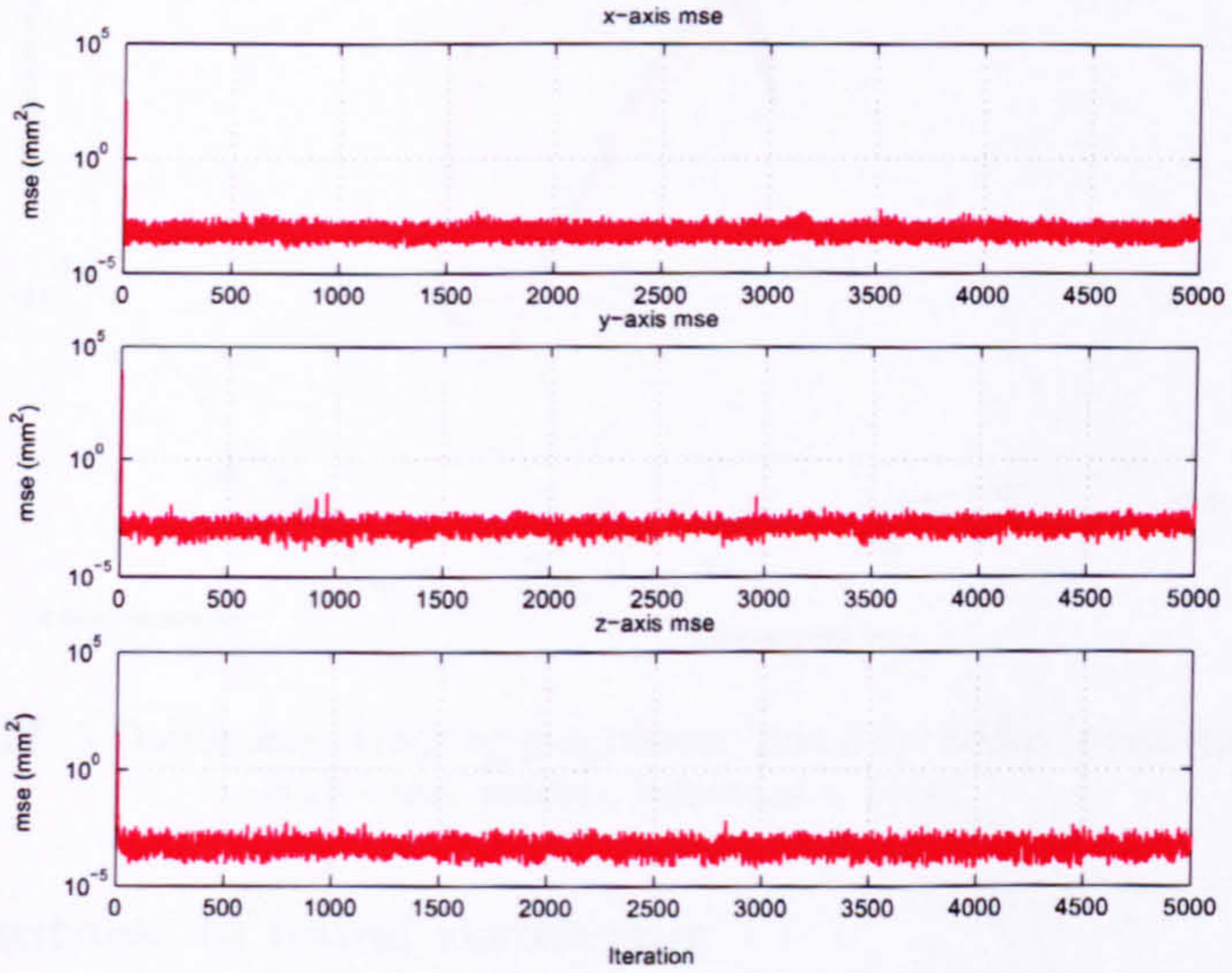


FIGURE 5.25: mse (norm-optimal ILC, high order models)

Figure 5.27 shows the displacement trajectories for iterations 1 to 5, highlighting the rapidity with which the algorithm learns. As compared to the inverse algorithm, although the X and Y-axes converge equally fast, the Z-axis converges slightly slower. PI_{100} values for the long-term stability test are presented in Table 5.10.

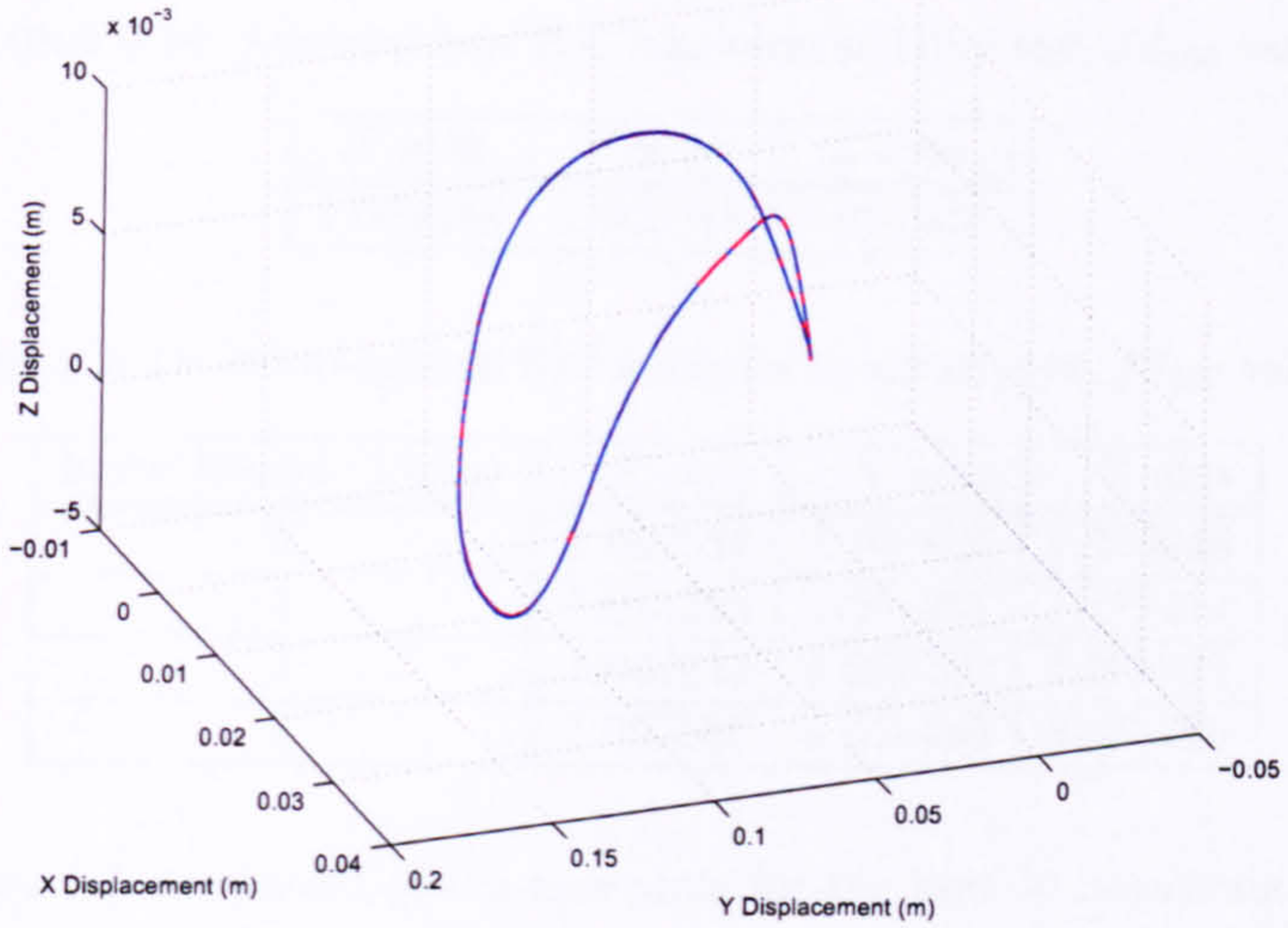


FIGURE 5.26: 3-Dimensional tracking performance (norm-optimal ILC, high order models, iteration 5000, reference = blue)

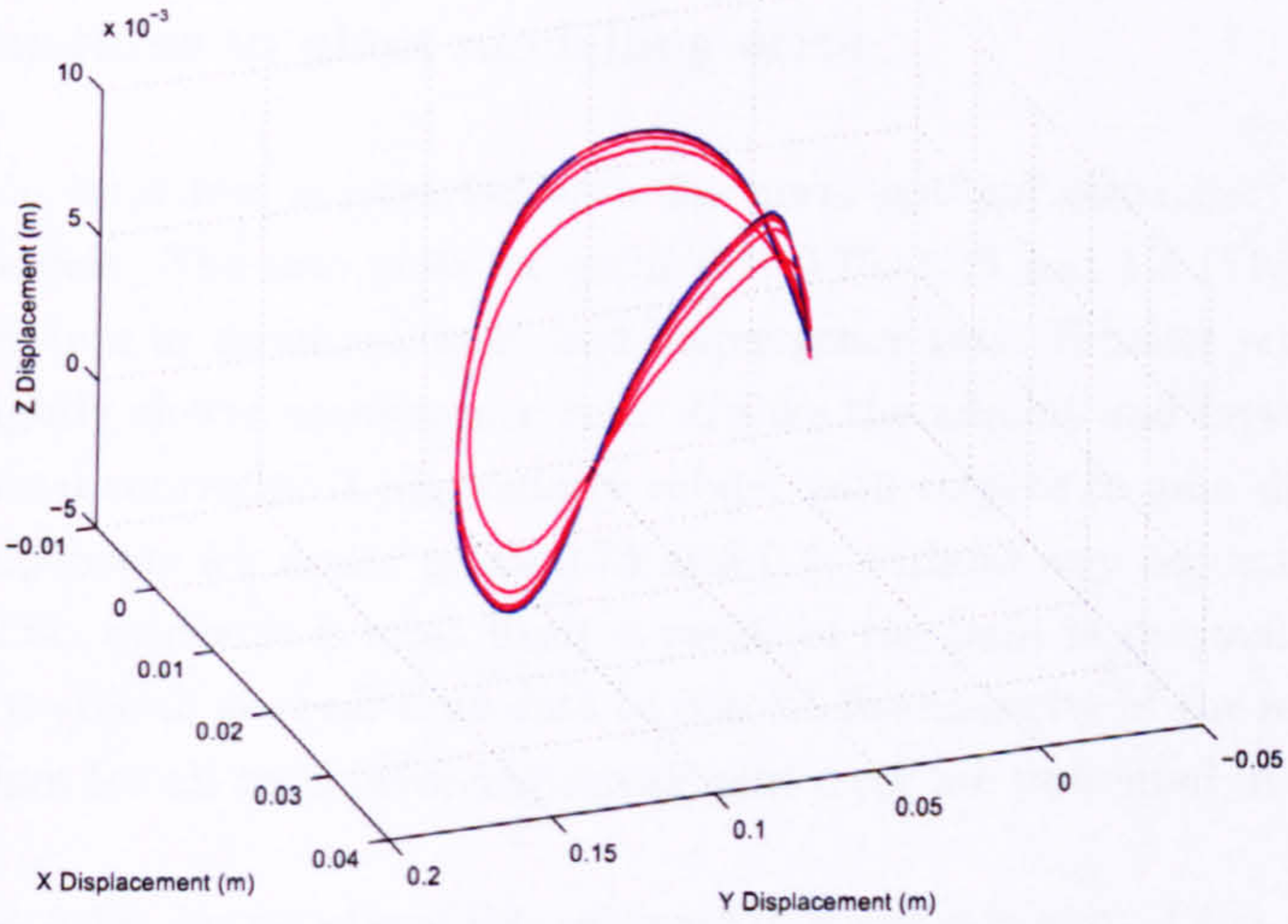


FIGURE 5.27: 3-Dimensional tracking progression, first 5 iterations (norm-optimal ILC, high order models, reference = blue)

5.4.4 Robustness to initial state error

The feedback component of the norm-optimal algorithm responds more immediately to initial error than the preceding adjoint and inverse algorithms. Therefore, larger step inputs are produced at the beginning of the iteration, and the initial error bound is limited to ±3mm, before the step inputs become too large for the plant. Figure 5.28 displays the mse curves for initial errors of 0, 1, 2 and 3mm. Within the 500 iteration test, stability is not compromised by increasing the initial error. Table 5.11 displays the PI_{100} data for different initial error bounds.

TABLE 5.10: Norm-optimal ILC long-term stability test, PI_{100} values

X-axis	Y-axis	Z-axis
1.002534	1.000888	1.065946

TABLE 5.11: Norm-optimal ILC tolerance to initial error, PI_{100} values

Error bound (±mm)	X-axis	Y-axis	Z-axis
0	1.002534	1.000888	1.065946
1	1.013238	1.001251	1.179414
2	1.044823	1.002520	1.508222
3	1.097886	1.004743	2.083286

The linear scale representation of the mse plots for the first 50 iterations (Figure 5.29), depicts the increase in both minimum mse and mse variation, as the initial error increases. The convergence rate is not affected by initial error.

5.4.5 Robustness to plant modelling error

The scalar gain error test is executed with the norm-optimal controller, using the high order plant models. The mse plots for gains 0.5, 0.75, 1.25 and 1.5 (Figure 5.30) show very little variation in minimum mse, and convergence rate. Smaller scalar gain tends to produce slightly slower convergence rate. Unlike the adjoint and inverse algorithms, the norm-optimal controller is particularly robust with respect to gain discrepancy and is able to compensate for scalar gains 0.75 and 0.5, without any adjustment of tuning parameters. This attribute is most likely a result of the built-in optimal feedback controller, which is able to use real-time data to correct the majority of the modelling error. The PI_{100} values for all tests involving scalar gain error are presented in Table 5.12.

TABLE 5.12: Norm-optimal ILC tolerance to scalar gain error, PI_{100} values

Scalar gain	X-axis	Y-axis	Z-axis
1.5	1.033856	1.025520	1.130277
1.25	1.015170	1.010434	1.092580
0.75	1.007166	1.015424	1.052615
0.5	1.080867	1.079621	1.066248

The 1st order model test produces the mse curves in Figure 5.31. Minimum mse and convergence rate are unaffected, but stability becomes a serious issue. After the initial reduction, the X-axis mse plot clearly increases from iteration 150 onwards, growing by almost one order of magnitude by iteration 500. This clearly indicates that the algorithm is unstable.

Analysis of the plant input and error profiles shown in Figure 5.32 indicates that the

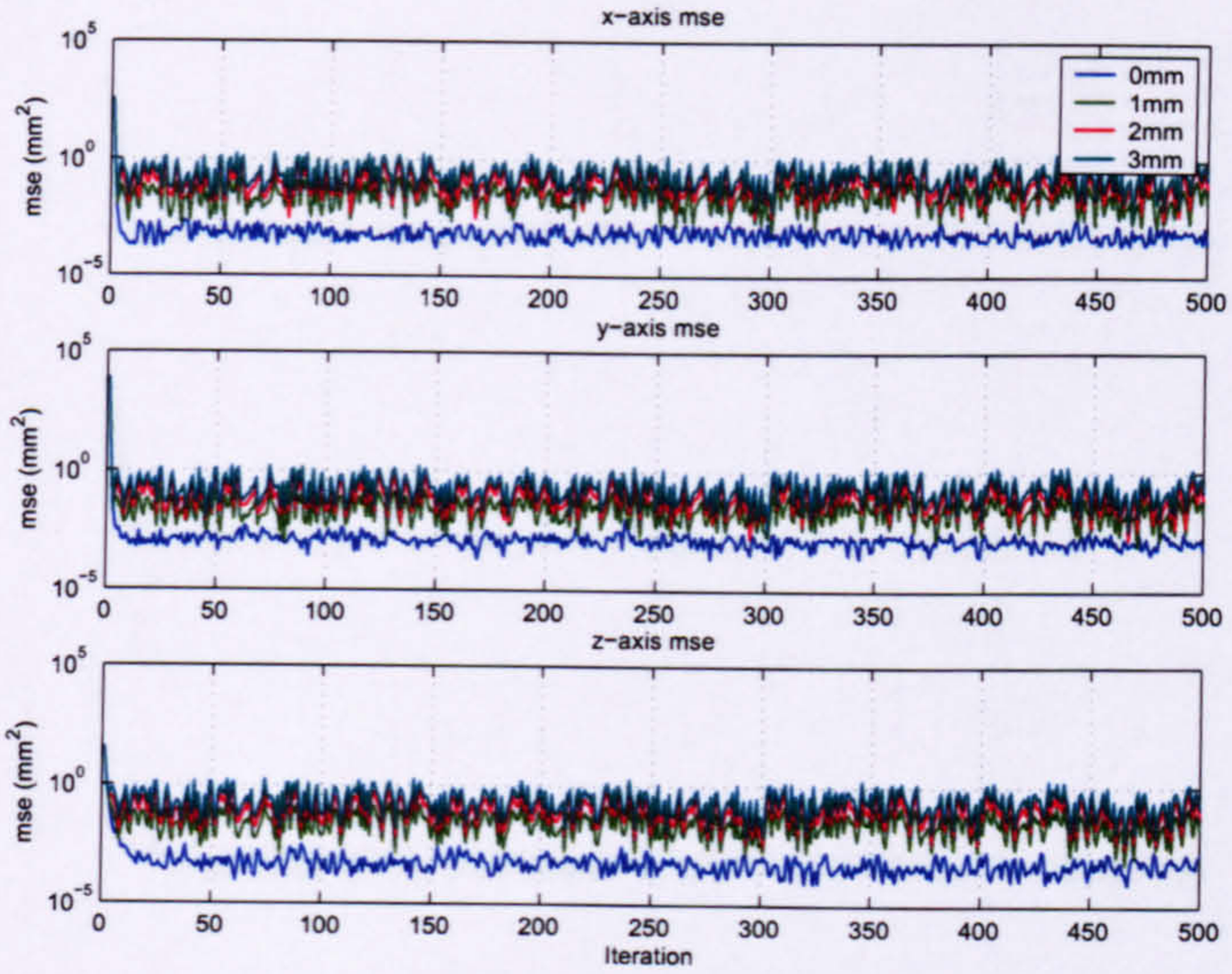


FIGURE 5.28: mse (norm-optimal ILC, high order models, initial error bounds 0, 1, 2 and 3 mm)

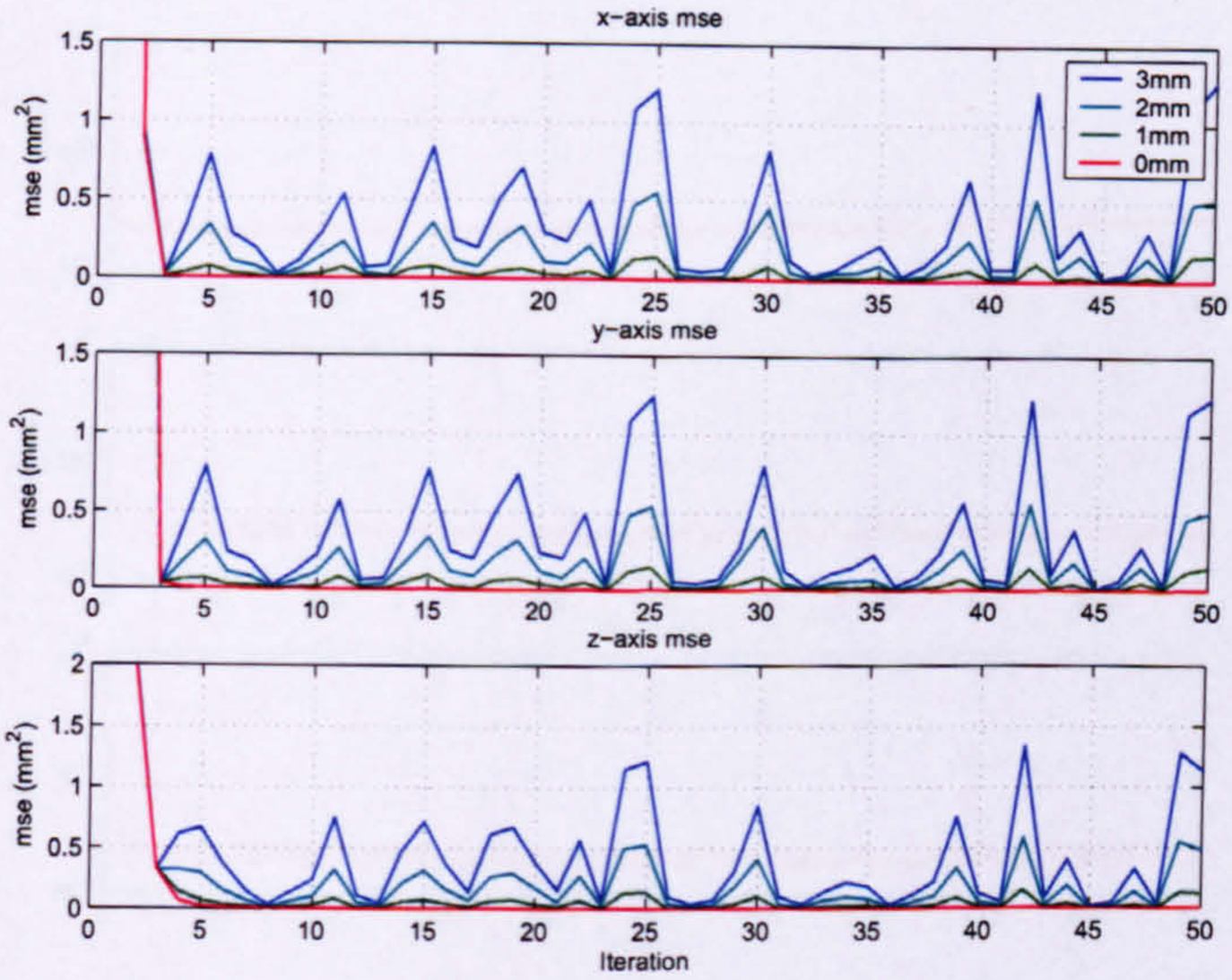


FIGURE 5.29: mse linear scale (norm-optimal ILC, high order models, initial error bounds 0, 1, 2 and 3 mm)

X-axis suffers from resonant frequency build up. Spectrum analysis confirms that a frequency of 14.5 Hz (91 rad/s) is dominant in both the input and error signals. With reference to the X-axis Bode plot (in Section 3.3), 14.5 Hz corresponds to the second highest resonant frequency. The lack of dynamic information relating to the resonant frequencies in the 1st order models is leading to algorithm instability.

Corrective measures for the resonance take the form of the zero-phase filter discussed in Section 5.3.3. The norm-optimal algorithm coupled with the zero-phase filter produces

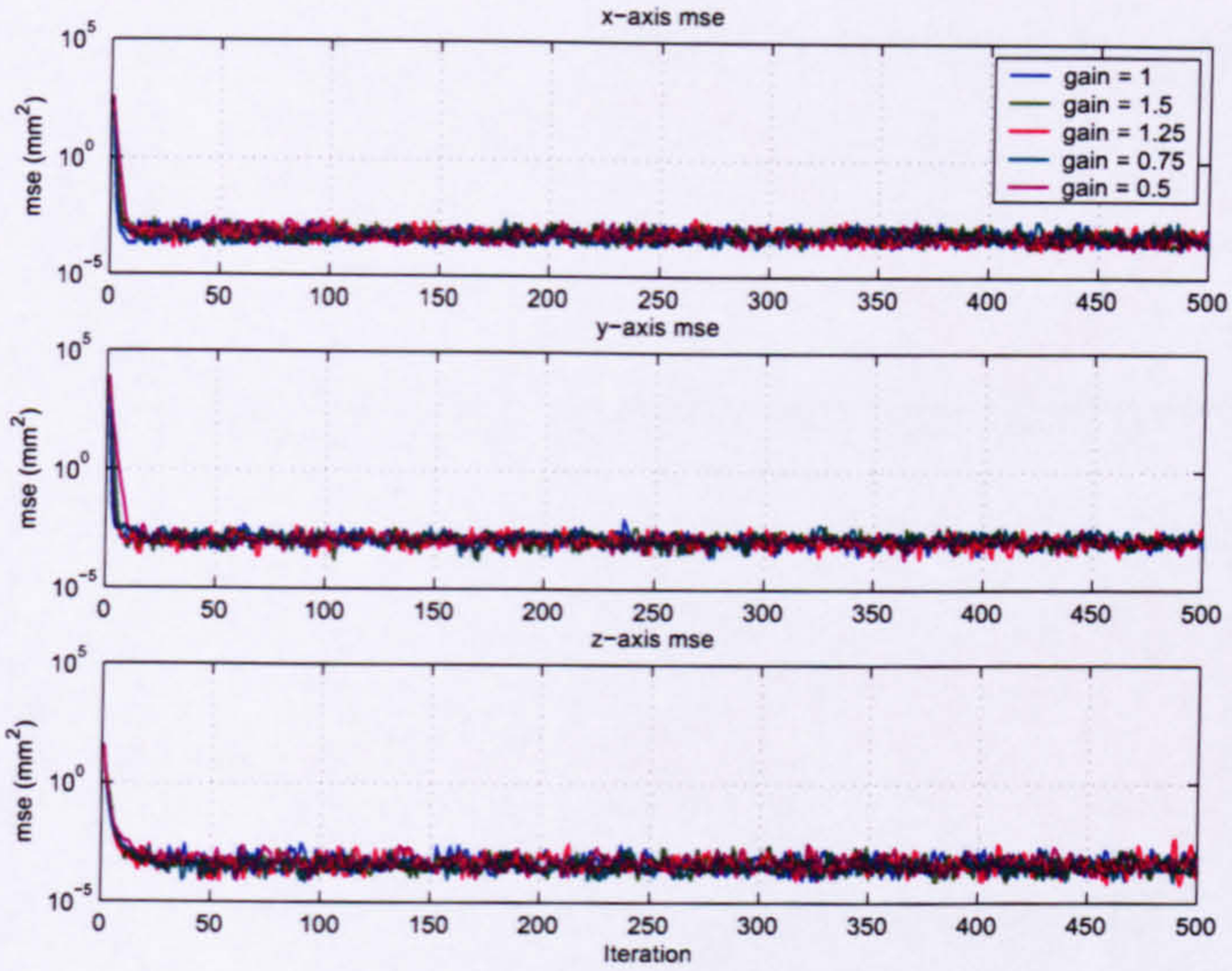


FIGURE 5.30: mse for different scalar gains (norm-optimal ILC, high order models, gains 1, 1.5, 1.25, 0.75 and 0.5)

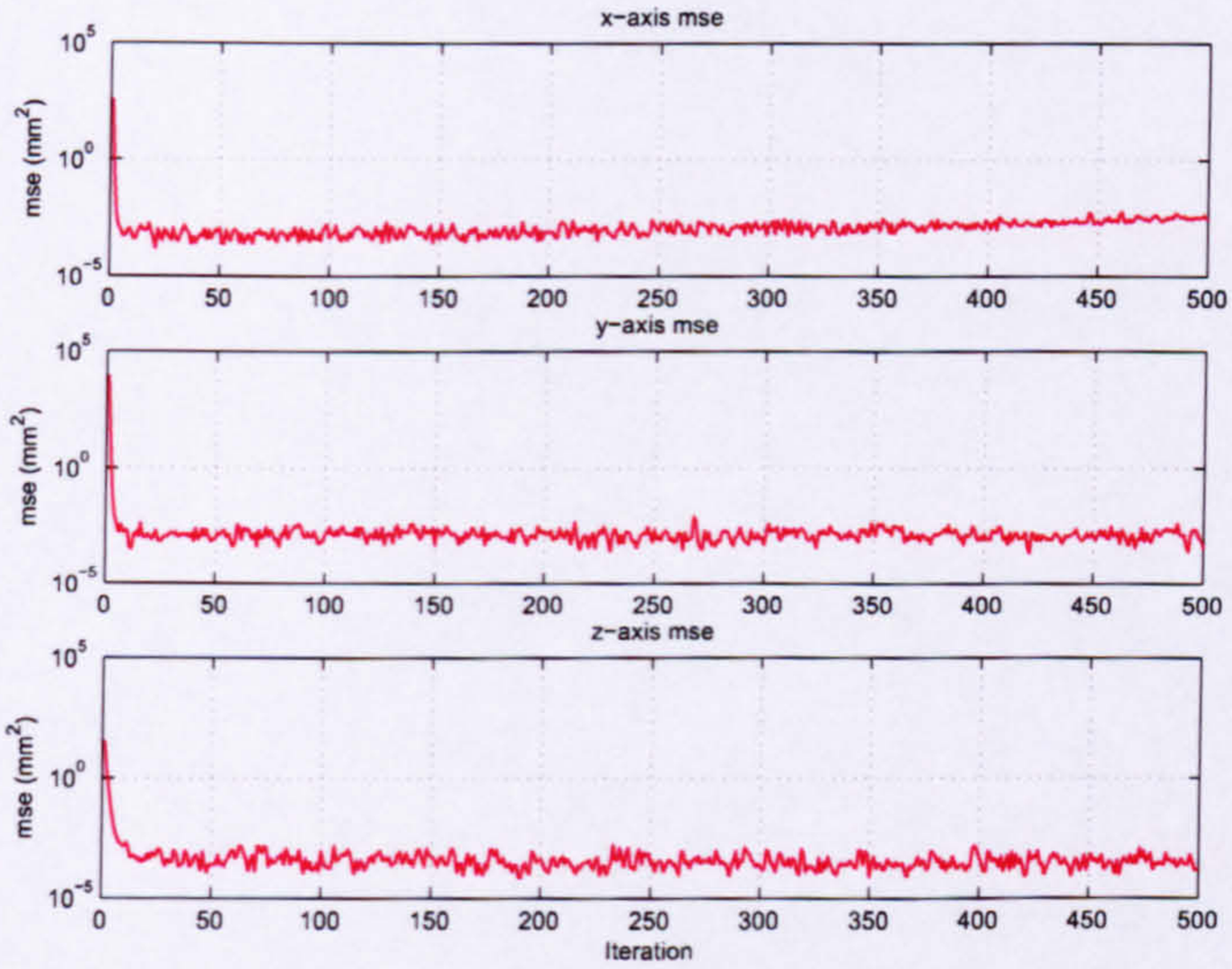


FIGURE 5.31: mse (norm-optimal ILC, 1st order models)

the mse plots in Figure 5.33. Following the initial learning, the mse remains steady and stability has been restored. Table 5.13 presents the PI_{100} data for the 1st order model implementation. Compared with the performance achieved using high order plant models, the controller achieves very similar performance, when implemented with the 1st order models, which lack detailed dynamic response data. There is a small improvement in the X-axis performance and a slight reduction for the Y and Z-axes.

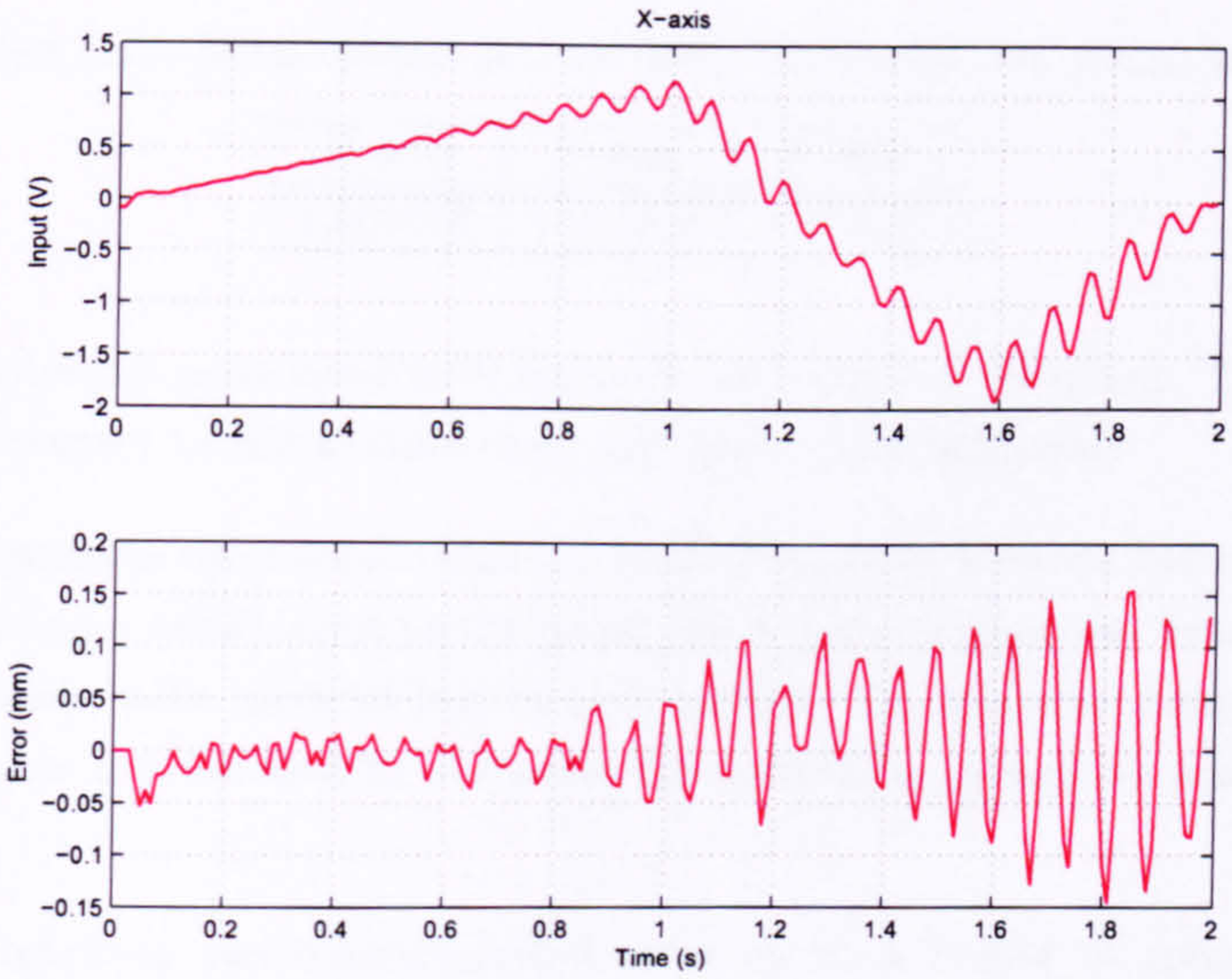


FIGURE 5.32: X-axis input demand and tracking error (norm-optimal ILC, 1st order models, iteration 500)

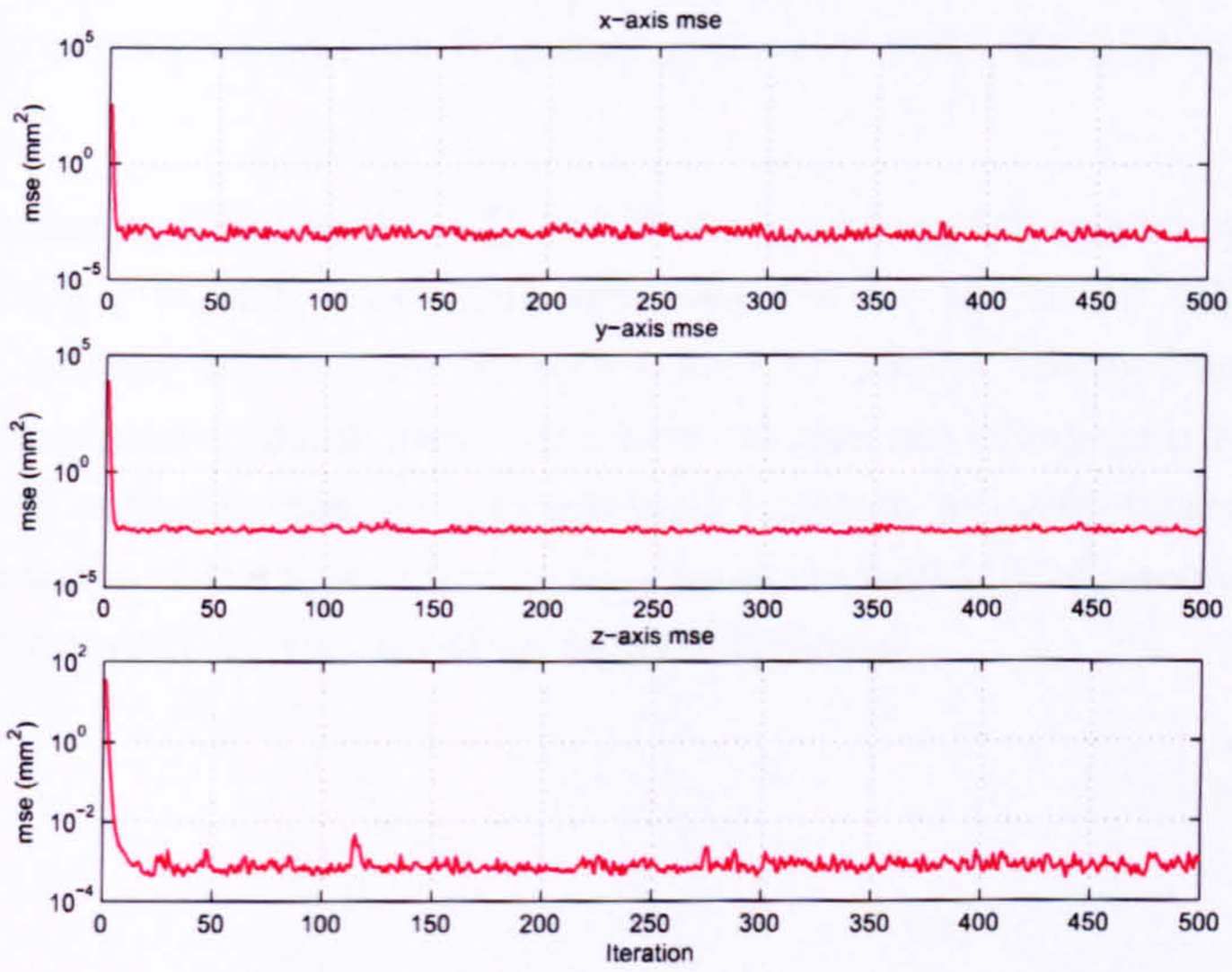


FIGURE 5.33: mse (norm-optimal ILC, zero-phase filter, 1st order models)

5.5 Summary

Three optimality based, iterative learning control algorithms have been successfully implemented on the gantry robot, producing excellent levels of mse reduction, coupled with fast convergence rates and long-term stability. The adjoint and norm-optimal algorithms are implemented without any requirement for additional filtering or external control. The inverse algorithm requires the use of a zero-phase filter to achieve stability, without which, only 4 iterations can be achieved before the input becomes heavily cor-

TABLE 5.13: Norm-optimal ILC 1st order implementation, PI_{100}, values

X-axis	Y-axis	Z-axis
1.001088	1.001818	1.076222

rupted by frequencies associated with resonant zero pairs of the plant. Each algorithm is tested with respect to initial state error and plant modelling error.

The adjoint algorithm can tolerate high levels of initial error without loss of stability and without producing a harsh input to the plant. First order models can be substituted for high order models, with minimal loss of performance. An adaptive version of the gain tuning parameter can be used to overcome the instability caused by a modelling gain error of 0.5.

The inverse algorithm can tolerate initial error up to a bound of $\pm$4mm before the plant input becomes unsuitable for the gantry robot. In a similar manner to the adjoint algorithm, an adaptive gain tuning parameter compensates for a gain modelling error of 0.5. The algorithm is more sensitive to gain discrepancy and the tuning parameter must also be used to correct a small low frequency gain error when the first order models are implemented.

The effects of norm-optimal matrices Q and R on algorithm performance are investigated and scalar values $q = 100$, $r = 0.001$ were selected for use in all other tests. The algorithm can tolerate initial error up to a bound of $\pm$3mm, without any variation in convergence speed and without loss of stability. Scalar modelling gain errors of 0.5 to 1.5 are tolerated without difficulty and minimal variation in performance. The lack of dynamic information in the first order models causes the build-up of resonant frequencies, which can be prevented by the use of zero-phase filtering.

Chapter 6

Comparative ILC Controller Performance

This chapter discusses the relative performance of each ILC algorithm implemented as part of this work. This is important, because one major reason for choosing to implement an model-based ILC algorithm, must be that the resultant system performance is superior to that which can be achieved using a simpler approach, that requires less knowledge of the plant. The performance index PI_N defined in Chapter 3, which is the sum of the first N iterations in a test, is used for this analysis. Using PI_N, it is not possible to compare the performance of basic algorithms with model-based algorithms numerically because the mse for the first trial is not equal. Therefore a discussion comparing the different mse plots is developed instead.

6.1 Basic Algorithms

Figure 6.1 displays the performance index results for the series controller arrangement with different values of learning gain. With gain equal to 0.0001, the algorithm learns very slowly and, within 100 iterations, the tracking error is barely smaller than for the feedback controller alone. Increasing the gain by a factor of 10 causes a noticeable improvement in performance, producing PI_{100} values between 90 and 92. Of the gains used in this test, 0.01 produces the greatest improvement in performance, with PI_{100} between 43 and 45. A gain of 0.1 causes rapid learning, but also accelerates the build up of noise and resonance in the iteration loop and therefore causes the plant to become unstable within the first 100 iterations.

Figure 6.2 displays the performance index results for the parallel controller arrangement. Comparison with Figure 6.1 for the series arrangement, highlights how the PI_{100} follows a similar pattern, confirming that the series and parallel implementations of a feedback

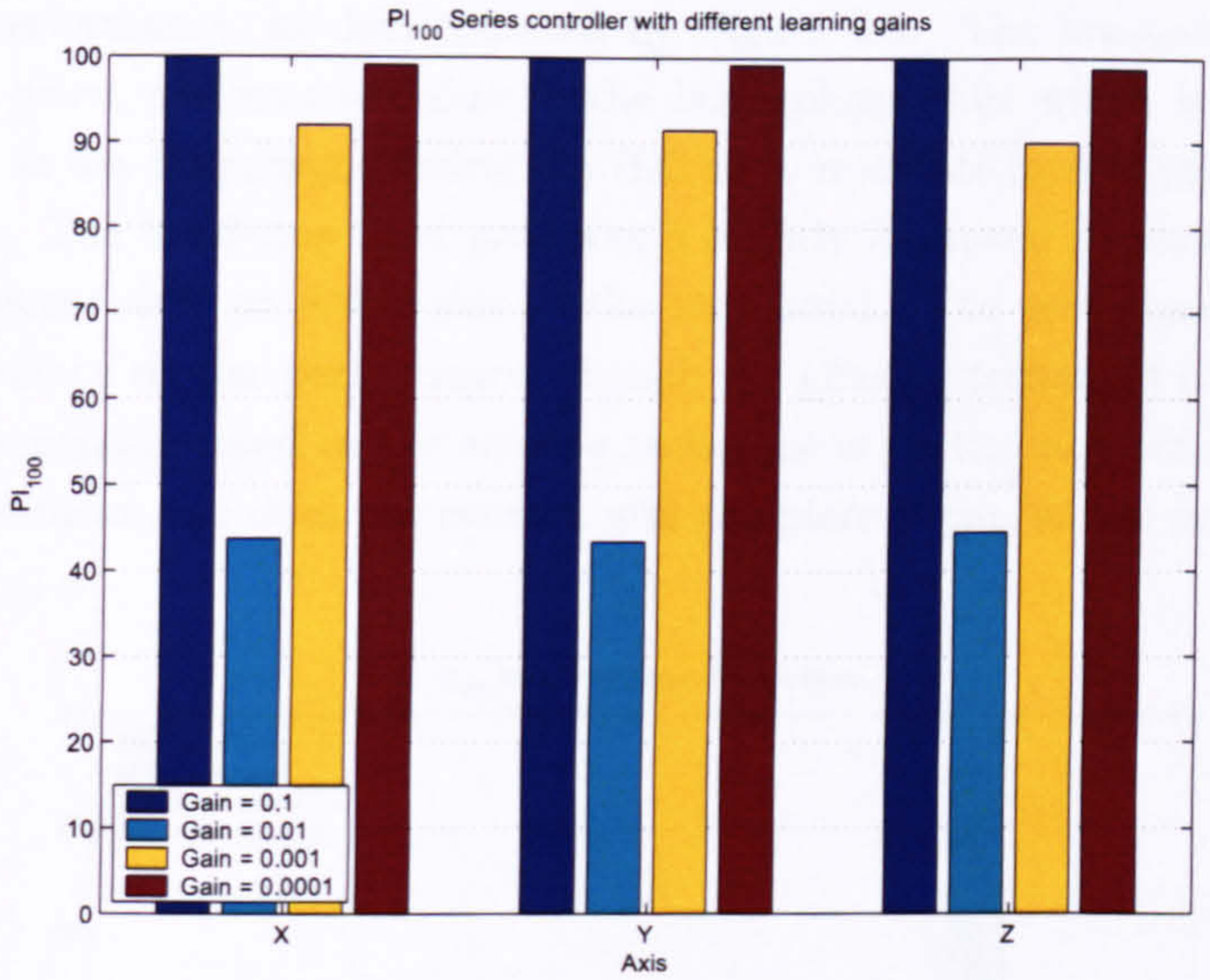


FIGURE 6.1: Comparison of PI_{100} values for the series controller with varying learning gain

controller assisted by ILC perform and behave in a similar manner.

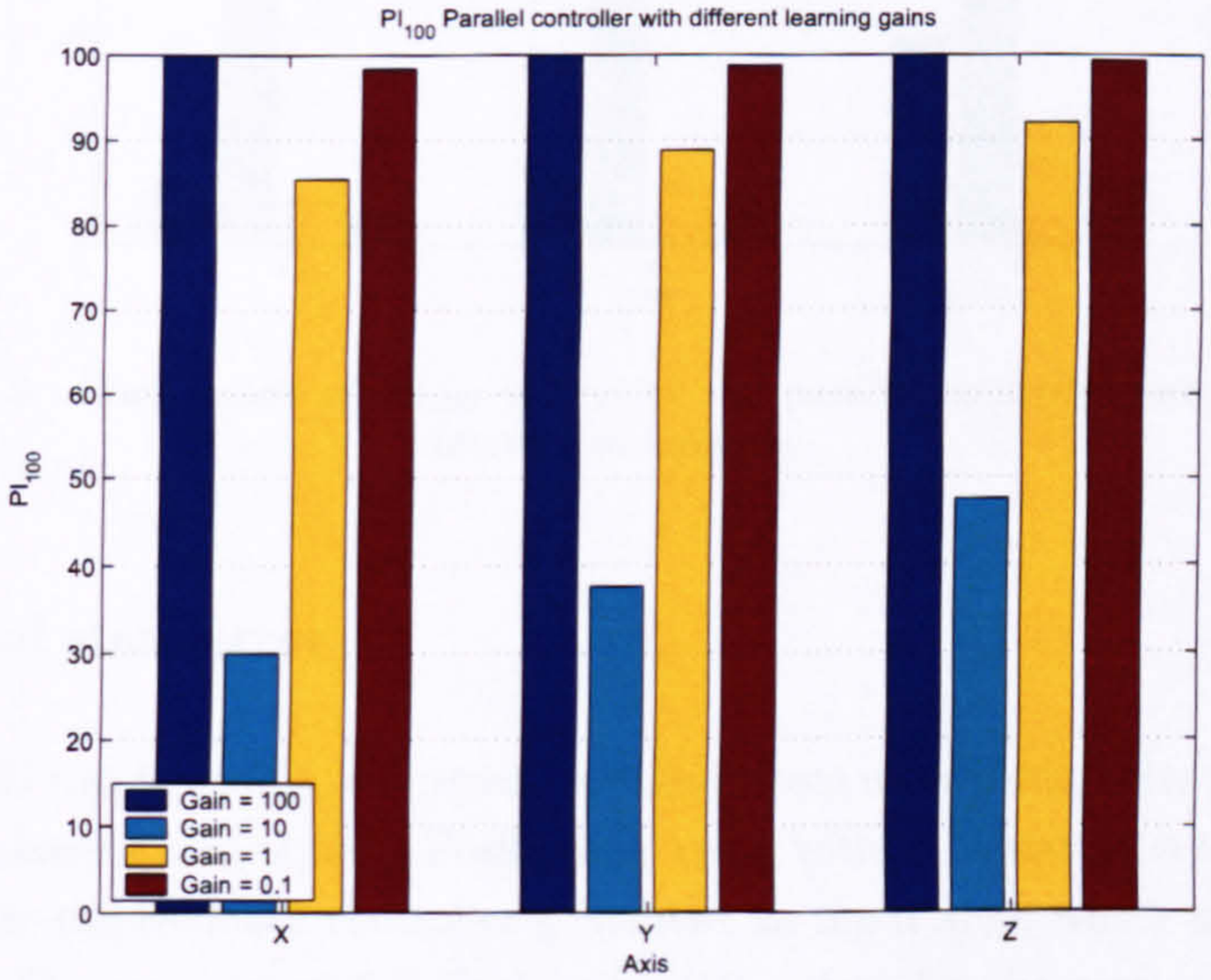


FIGURE 6.2: Comparison of PI_{100} values for the parallel controller with varying learning gain

6.1.1 Filtering comparison

Filtering techniques permit a learning gain of 100 to be used in the parallel controller implementation, without loss of stability. The choice of filter also has a significant effect

on learning performance, as demonstrated by Figure 6.3. The low-pass filter clearly produces the worst performance, due to the large phase shift which is added to the filtered signal at low frequency, causing the ILC to compensate incorrectly for measured tracking error. The band-stop filter produces a slightly improved performance but still causes large phase shift on either side of the stop band. The zero-phase and aliasing techniques produce similar performance, though the aliasing technique is best, because the linear interpolation used in the aliasing technique is particularly well suited to the reference trajectories and does not over-smooth the plant input, as the zero-phase filter does.

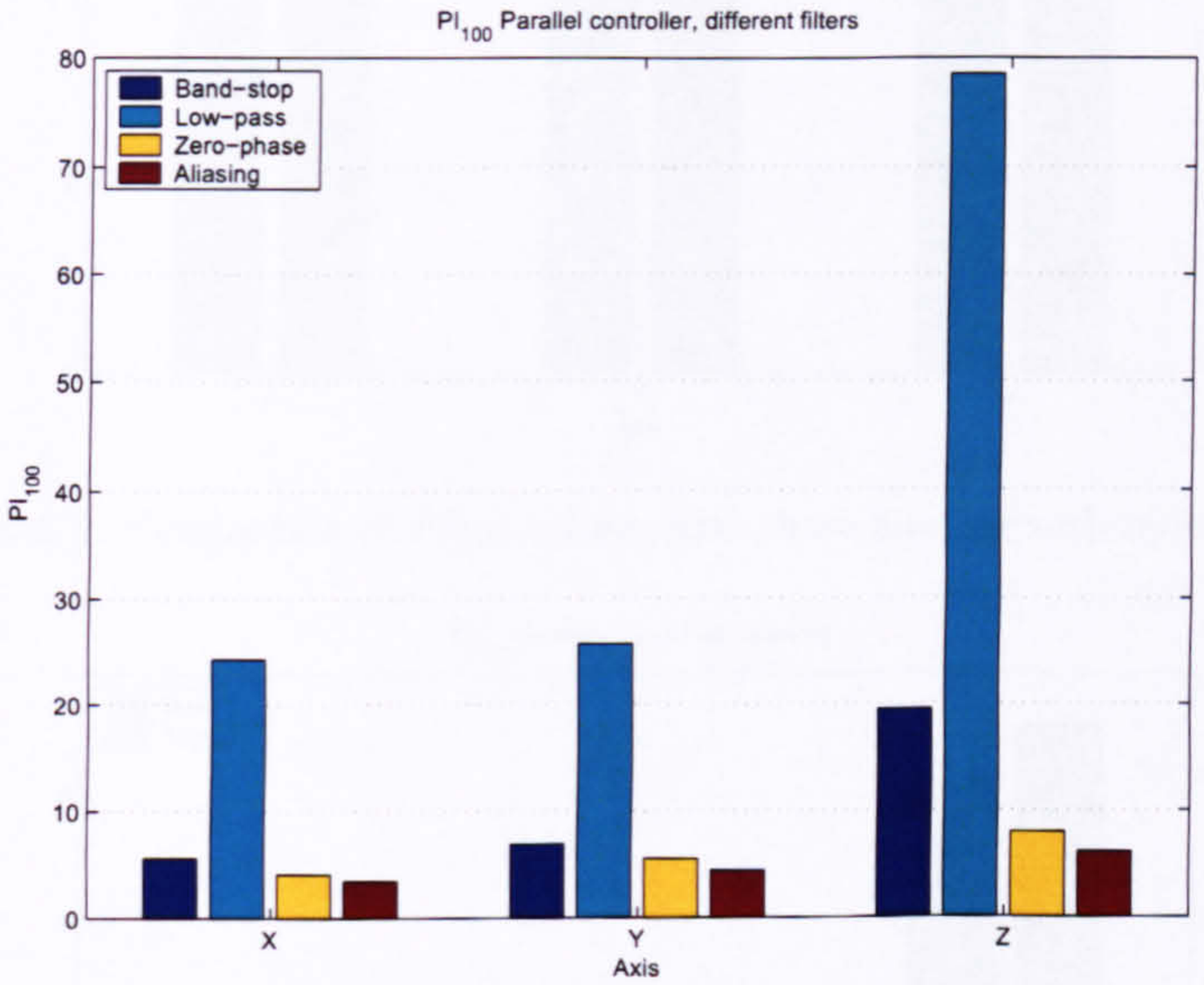


FIGURE 6.3: Comparison of PI_{100} values for the parallel controller using different filtering techniques

6.1.2 Initial state error

The hybrid ILC was tested for robustness to initial state error using both the zero-phase filtering and aliasing techniques. Positioning error with a bound of $\pm$1mm could be tolerated before the feedback controller generated an input step, which was unsuitable for the plant. Figures 6.4 and 6.5 display the PI_{100} data for the zero-phase filter and aliasing techniques respectively. The PI_{100} values for the zero-phase filter are generally higher than for the aliasing technique as discussed in the previous section.

The initial error bound has minimal effect on the Y-axis performance, while the influence on the Z-axis is greatest. This is to be expected, because the $\pm$1mm bound constitutes a greater portion of the Z-axis reference than of the Y-axis reference. The increase in PI_{100} value for each example is less than 0.5 and is not a substantial change, because the feedback controller rapidly removes the initial error at the start of the trajectory,

resulting in minimal impact on the learning controller.

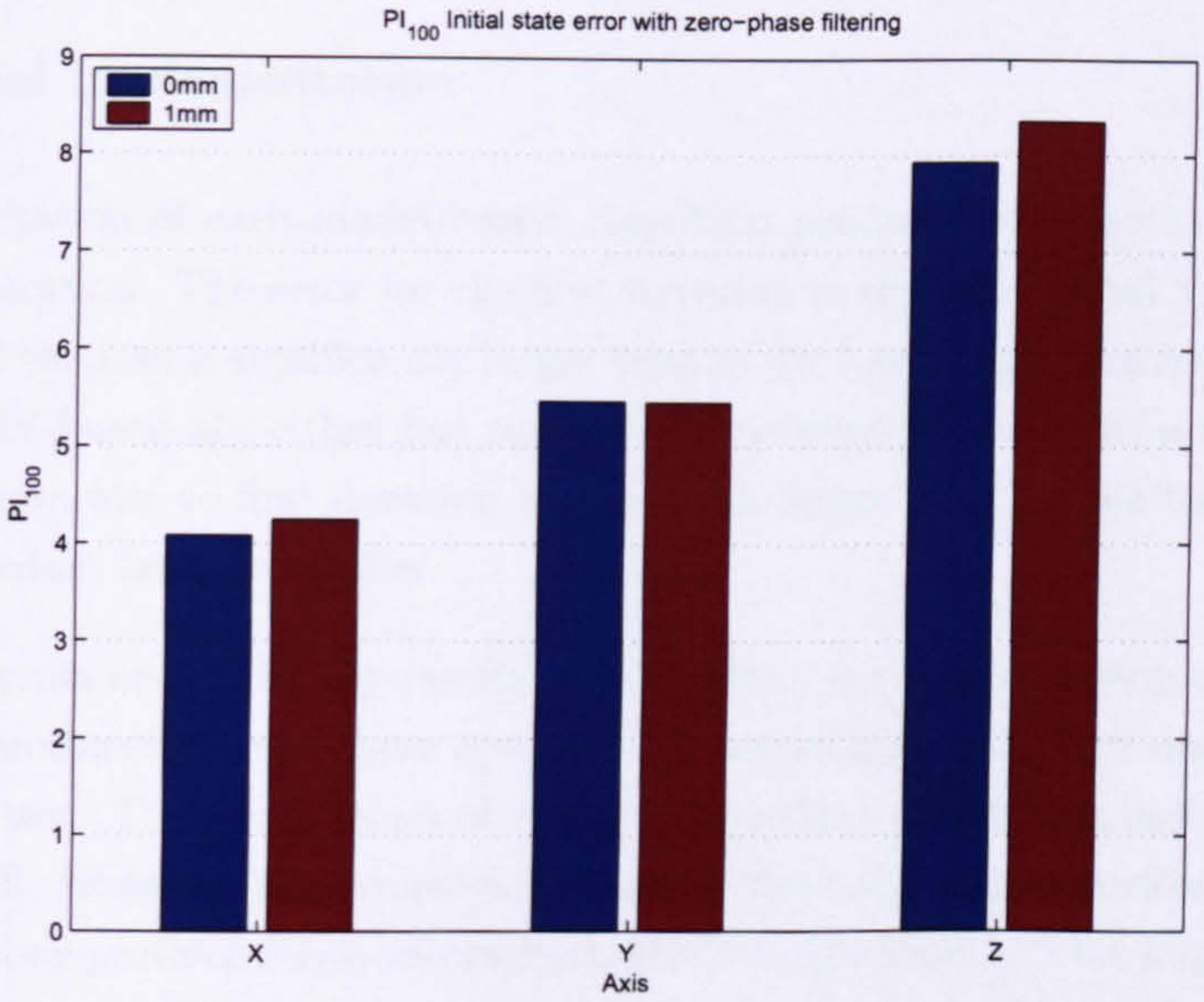


FIGURE 6.4: Comparison of PI_{100} values, zero-phase filtering with initial error

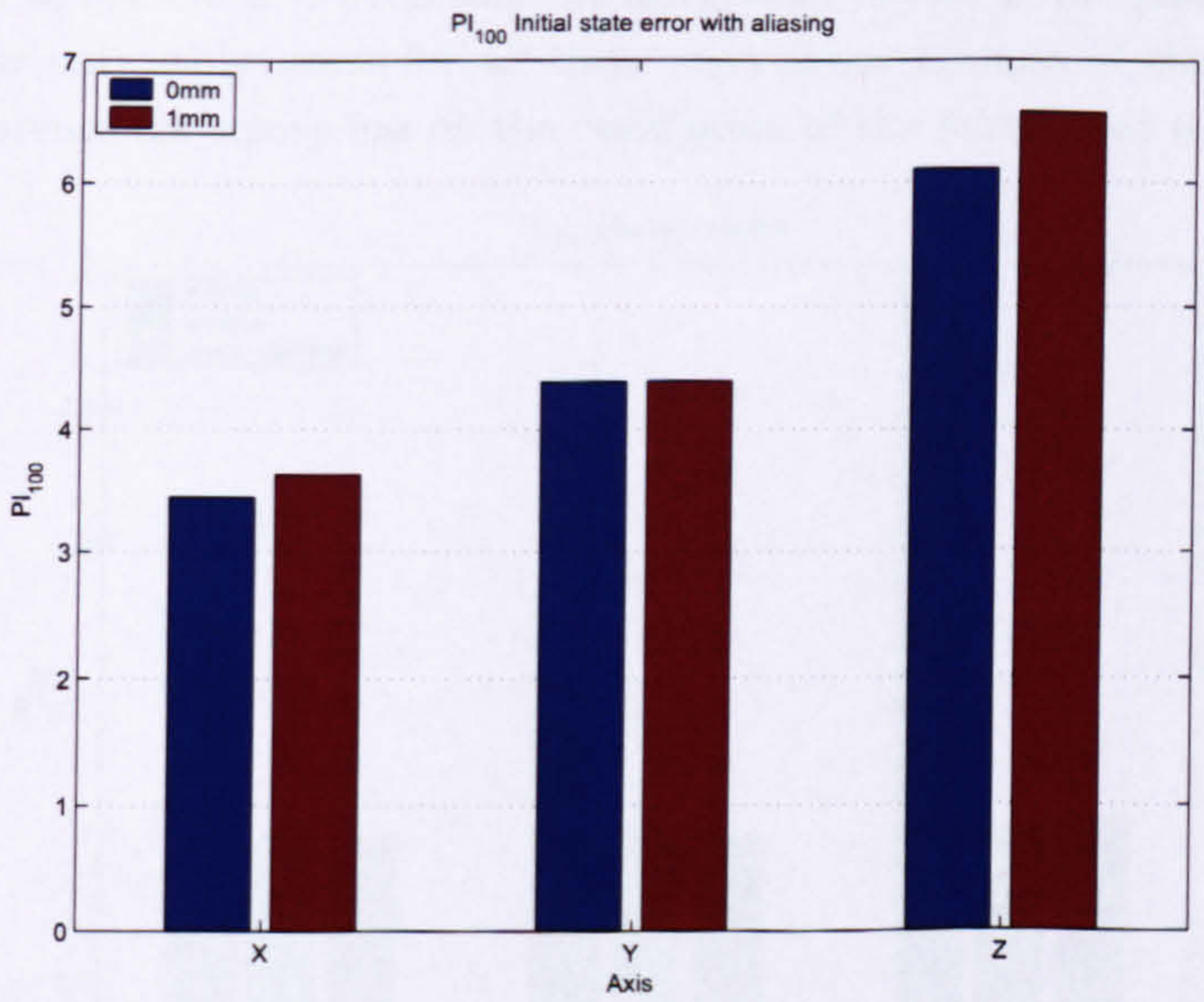


FIGURE 6.5: Comparison of PI_{100} values, aliasing with initial error

6.2 Model-based Algorithms

6.2.1 Initial implementation

The implementation of each model-based algorithm produces zero motion of the robot for the first iteration. The error for the first iteration is therefore equal to the reference trajectory and the mse is significantly larger than in the basic algorithm implementation. Once the model-based algorithm has successfully reduced the error to a minimum, the ratio of minimum mse to first iteration mse is much larger than for the basic algorithm, so the PI_{100} values become smaller.

Figure 6.6 presents one of the key results of this thesis. It is a comparison of the tracking performance produced by the three optimal ILC controllers, recorded during the long-term stability test. The small values of PI_{100} for the three algorithms indicate that they all perform well. However, the inverse and norm-optimal algorithms perform particularly well, because they produce PI_{100} values fractionally larger than 1. This implies that they learn the majority of the required plant input in one iteration. The performance of the adjoint algorithm is noticeably worse, because the input update mechanism requires a larger number of iterations to overcome the integrating nature of the plant. The Z-axis performance is noticeably worse for all three algorithms, because of the scaling effect which the reference trajectory has on the calculation of the normalised mse.

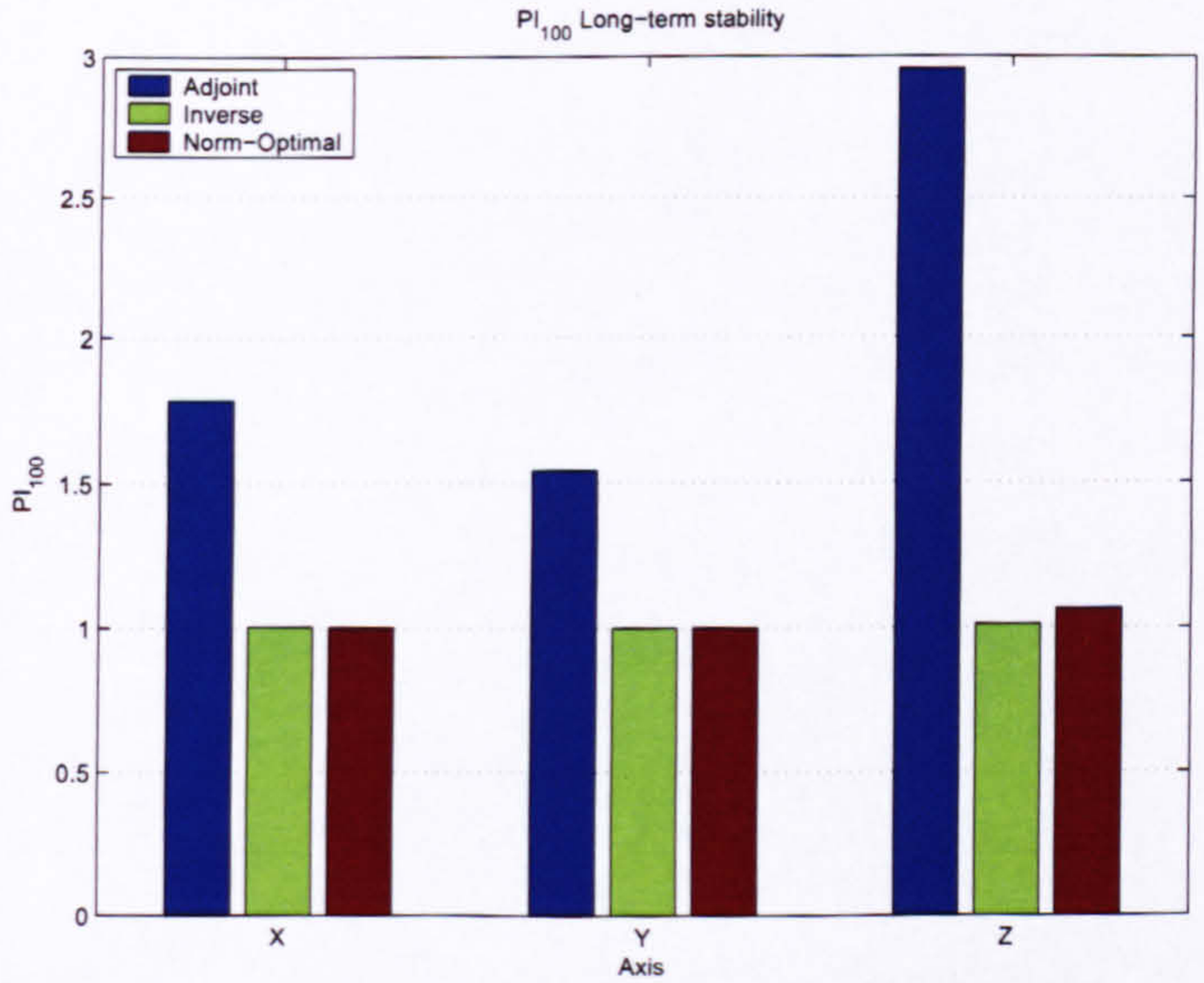


FIGURE 6.6: Comparison of PI_{100} values calculated from long-term stability test

It is possible to conclude that the algorithm which produces the best performance during the first 100 iterations, when test conditions are optimal, is the inverse ILC. This is logical, because the accuracy of the plant models has been verified and found to be high.

Therefore, the inverse algorithm produces an excellent estimate of the required plant input for iteration 2.

Direct comparison of the long-term stability mse plots (Figure 6.7) suggests that the norm-optimal algorithm consistently produces mse values, smaller than the other two algorithms. The PI_{100} comparison is more biased by initial learning rate than consistent minimum mse, therefore the inverse algorithm performs best under these criteria.

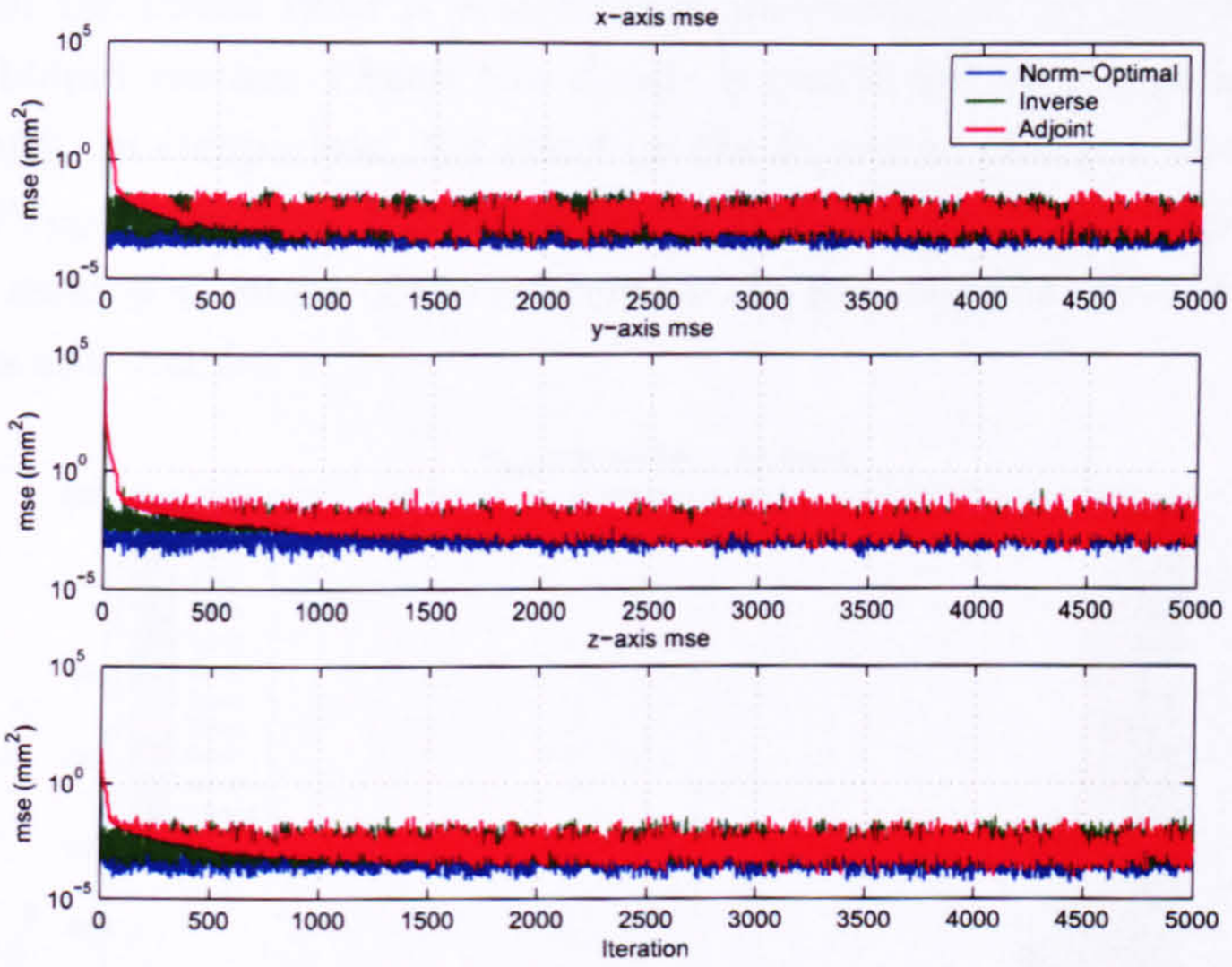


FIGURE 6.7: mse for all three model-based algorithms

Table 6.1 presents the PI_{5000} values for the long-term stability tests. Using this variant of the index, $PI_{5000}(min) = 1$, while $PI_{5000}(max) = 5000$ and the comparison is more biased to consistent mse reduction. This is a good example of how N in PI_N must be carefully chosen to influence which characteristics the performance index emphasises most.

TABLE 6.1: Model-based ILC algorithms, high order models, PI_{5000} values

Algorithm	X-axis	Y-axis	Z-axis
Adjoint	1.885475	1.551208	3.411915
Inverse	1.105030	1.005053	1.438905
Norm-Optimal	1.013335	1.001809	1.141605

The adjoint algorithm still produces the poorest performance, but the norm-optimal performance clearly surpasses that of the inverse algorithm, because the norm-optimal algorithm converges slightly slower than the inverse, but maintains lower and less variable mse.

6.2.2 Initial state error

Figures 6.8, 6.9 and 6.10 present the PI_{100} values for the adjoint, inverse and norm-optimal algorithms respectively, when initial position error is added at the start of the iteration. The adjoint algorithm is able to tolerate the largest initial error ($\pm$10mm) without loss of stability. However, the impact on tracking performance is very clear. Increasing the initial error exponentially increases the PI_{100} values for the X and Z-axes, for which the initial error is a significant percentage of the reference trajectory. As the error bound reaches $\pm$8mm the Z-axis is stable but no useful learning occurs and $PI_{100} = 100$. In comparison, the effect on the Y-axis is minimal, there is a general tendency for PI_{100} to increase, but the variation is highly non-linear, because the initial error is only a small percentage of the reference trajectory and the effect of non-repeating disturbances is still visible.

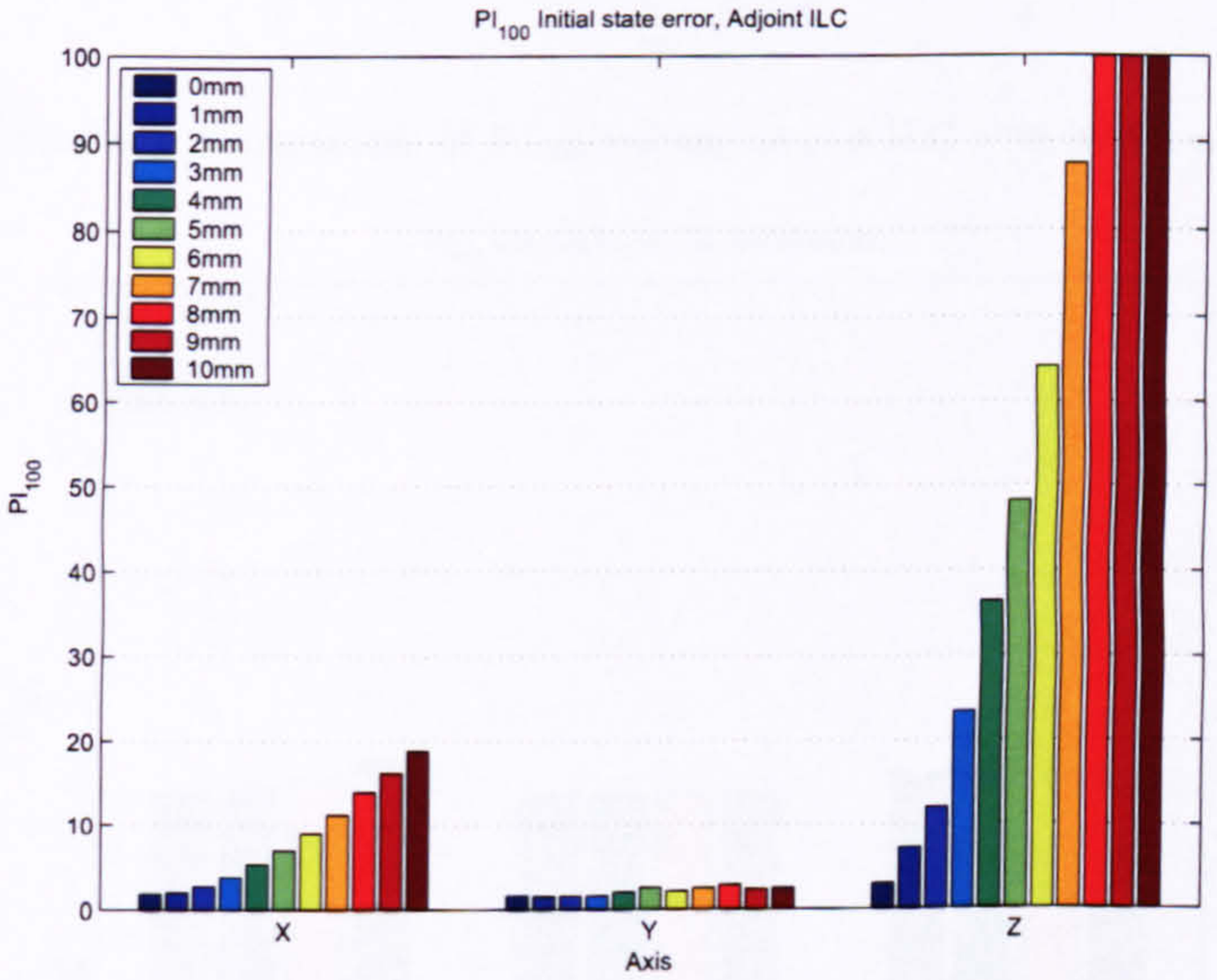


FIGURE 6.8: Comparison of PI_{100} values, adjoint ILC with initial error

The effect of initial error on the inverse and norm-optimal algorithms follows the same trends as for the adjoint algorithm. However, the magnitude of the PI_{100} values is much smaller. Therefore the initial error has a less pronounced effect on tracking performance. In particular, the norm-optimal algorithm is able to tolerate $\pm$3mm of error for the Z-axis, producing a $PI_{100} = 2.083$, while the adjoint and inverse algorithms produce $PI_{100} = 23.326$ and $PI_{100} = 9.795$ respectively.

6.2.3 Robustness to plant modelling error

The relative performance of the model-based algorithms, when implemented with 1st order models, follows the same trends as with high order models. Figure 6.11 presents

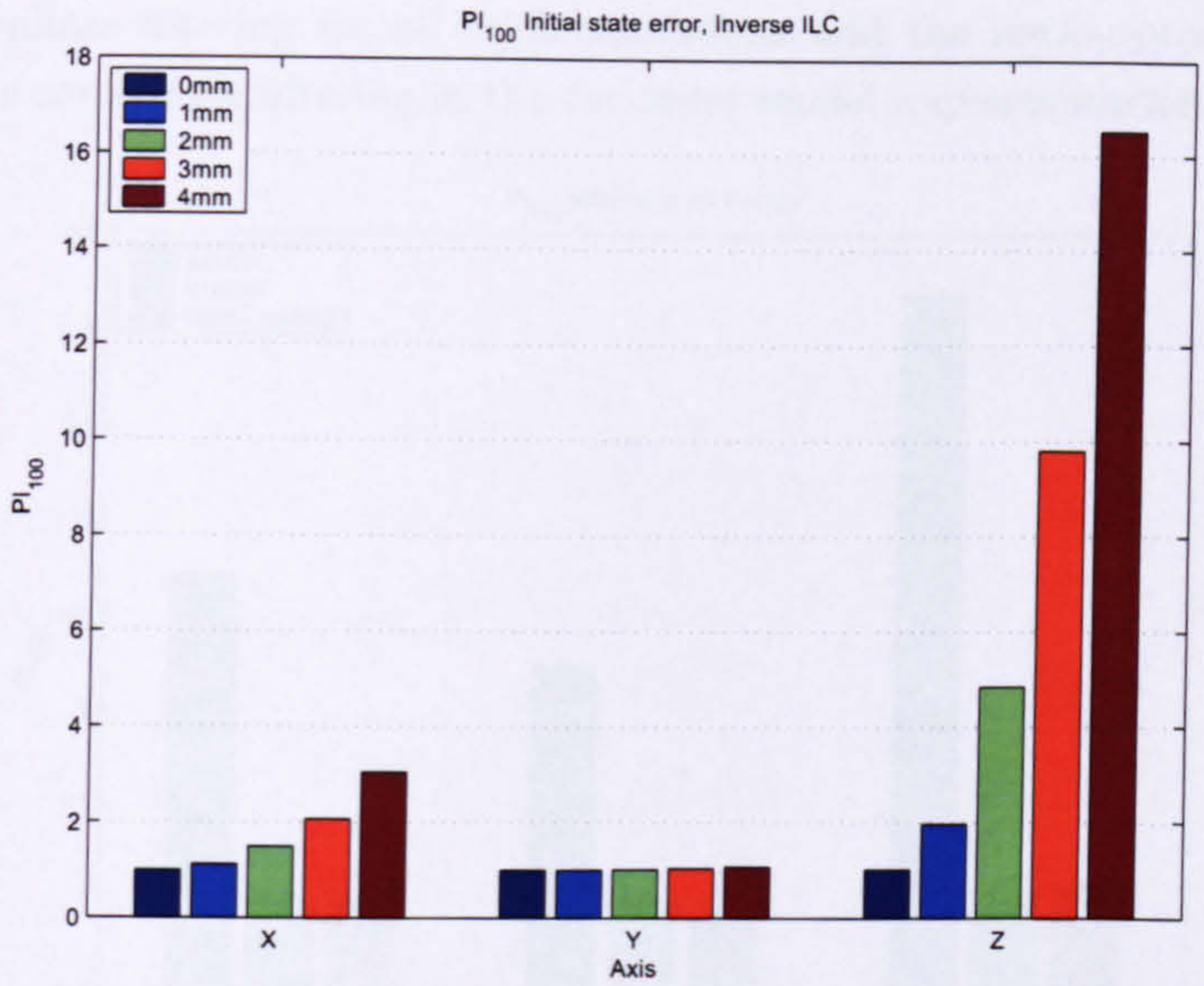


FIGURE 6.9: Comparison of PI_{100} values, inverse ILC with initial error

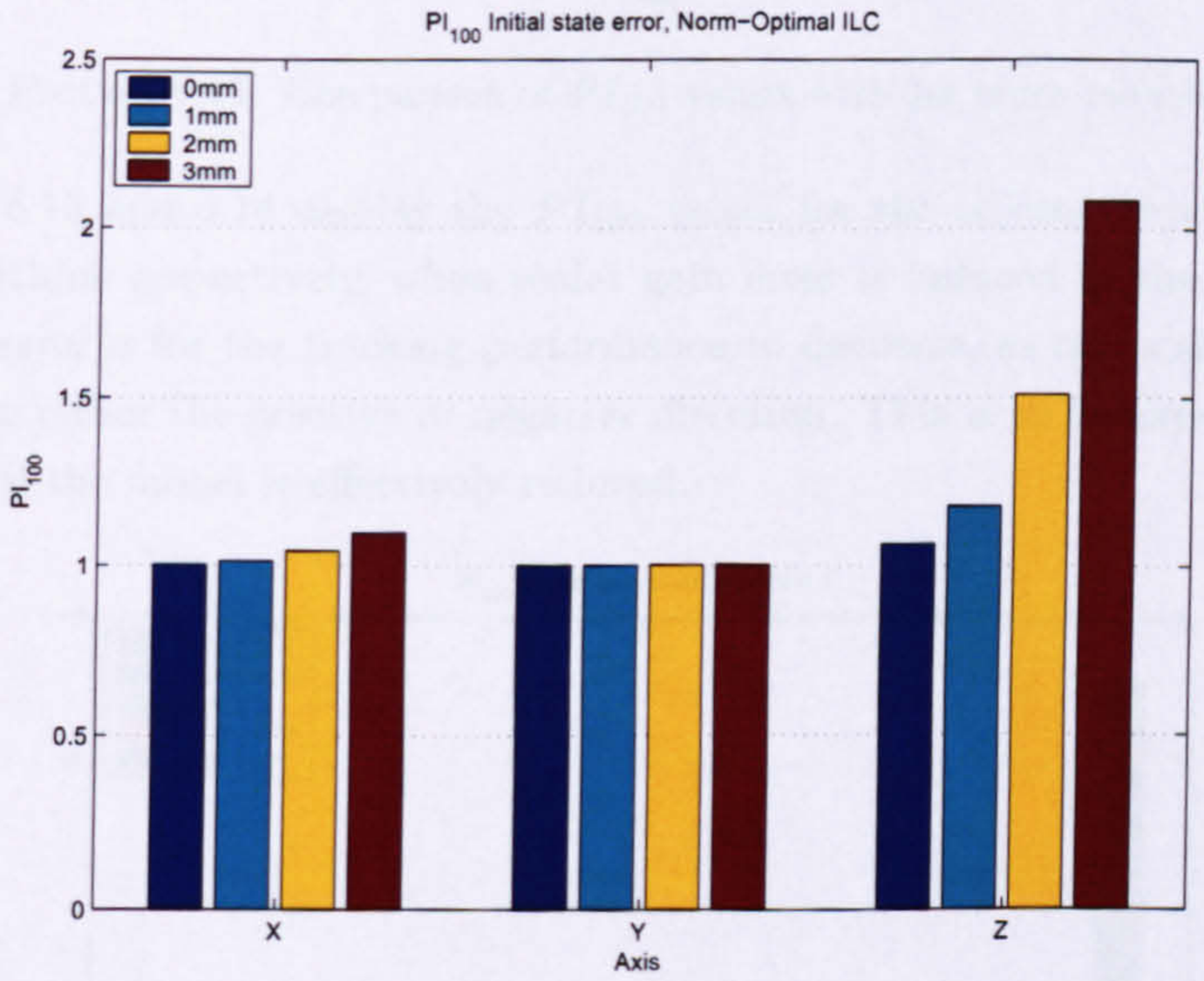


FIGURE 6.10: Comparison of PI_{100} values, norm-optimal ILC with initial error

the PI_{100} data, showing that the adjoint algorithm produces the poorest performance, while the performance of the inverse and norm-optimal algorithms is better. The adjoint algorithm is also significantly more affected by the reduction in modelling accuracy. Compared with Figure 6.6 for the high order models, the X, Y and Z-axis PI_{100} values have increased from 1.787, 1.544 and 2.956, to 2.713, 2.236 and 4.192 respectively. However, the adjoint algorithm appears to be more robust with respect to model order reduction than the other model-based algorithms, because it is implemented without any additional stabilisation techniques, unlike the inverse algorithm, which intrinsically

requires zero-phase filtering for all implementations and the norm-optimal algorithm, which requires zero-phase filtering in the 1st order model implementation.

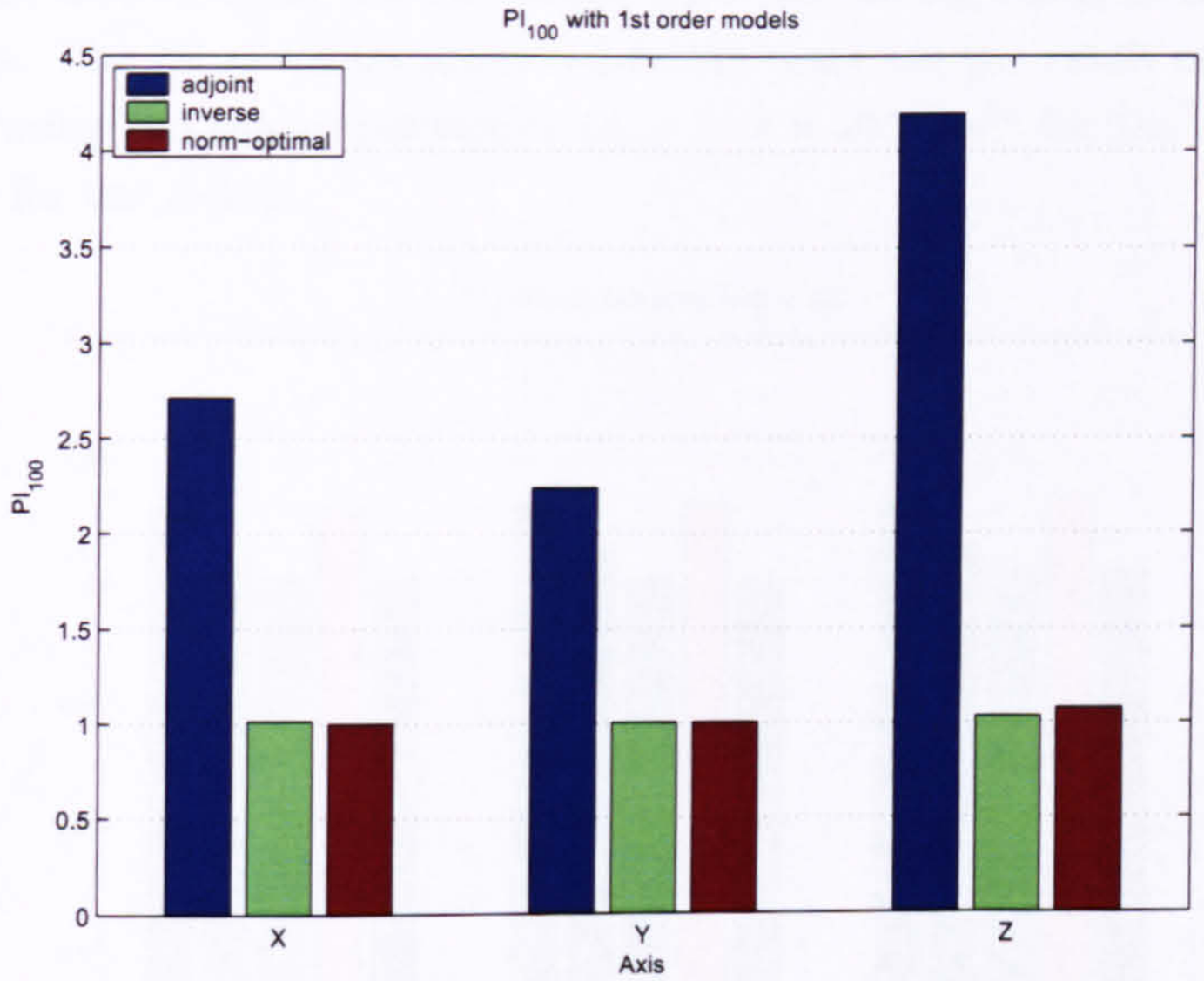


FIGURE 6.11: Comparison of PI_{100} values with 1st order models

Figures 6.12, 6.13 and 6.14 display the PI_{100} values for the adjoint, inverse and norm-optimal algorithms respectively, when scalar gain error is induced in the plant models. The general trend is for the tracking performance to decrease, as the scalar gain moves away from 1 in either the positive or negative direction. This is to be expected, because the accuracy of the model is effectively reduced.

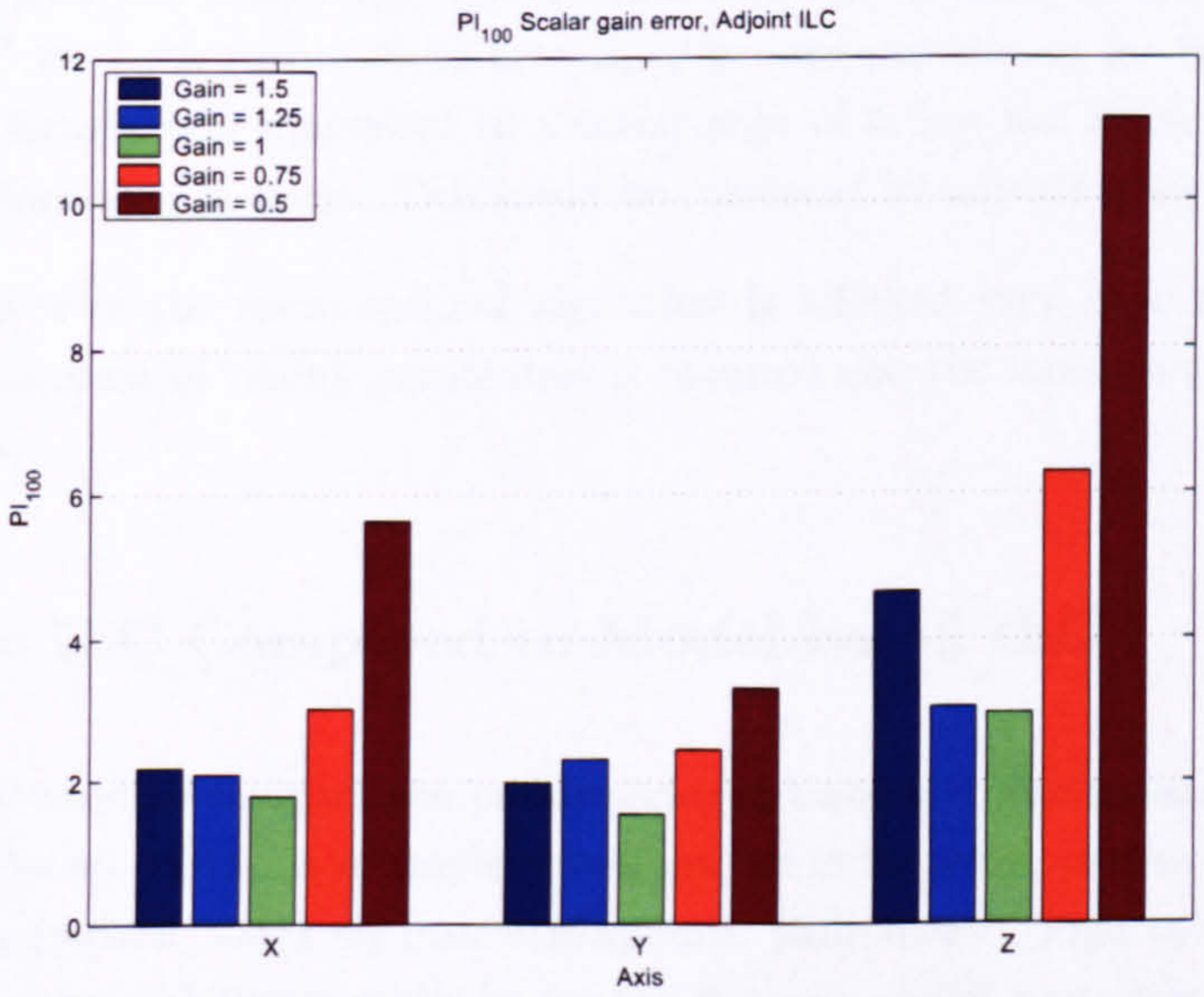


FIGURE 6.12: Comparison of PI_{100} values, adjoint ILC with scalar gain error

For the adjoint algorithm, the decrease in performance is particularly noticeable for scalar gains less than 1. The performance reduces much faster than if the scalar gain is larger than 1. It must also be noted that, for scalar gain 0.5, the algorithm is unstable for the X and Y-axes. The PI_{100} values presented in the chart are the result of learning gain reduction by using the tuning parameter $\omega_{k+1} = 2 \times 10^{-7}||\mathbf{e}||^2$ for the X and Y-axes and $\omega_{k+1} = 0$ for the Z-axis.

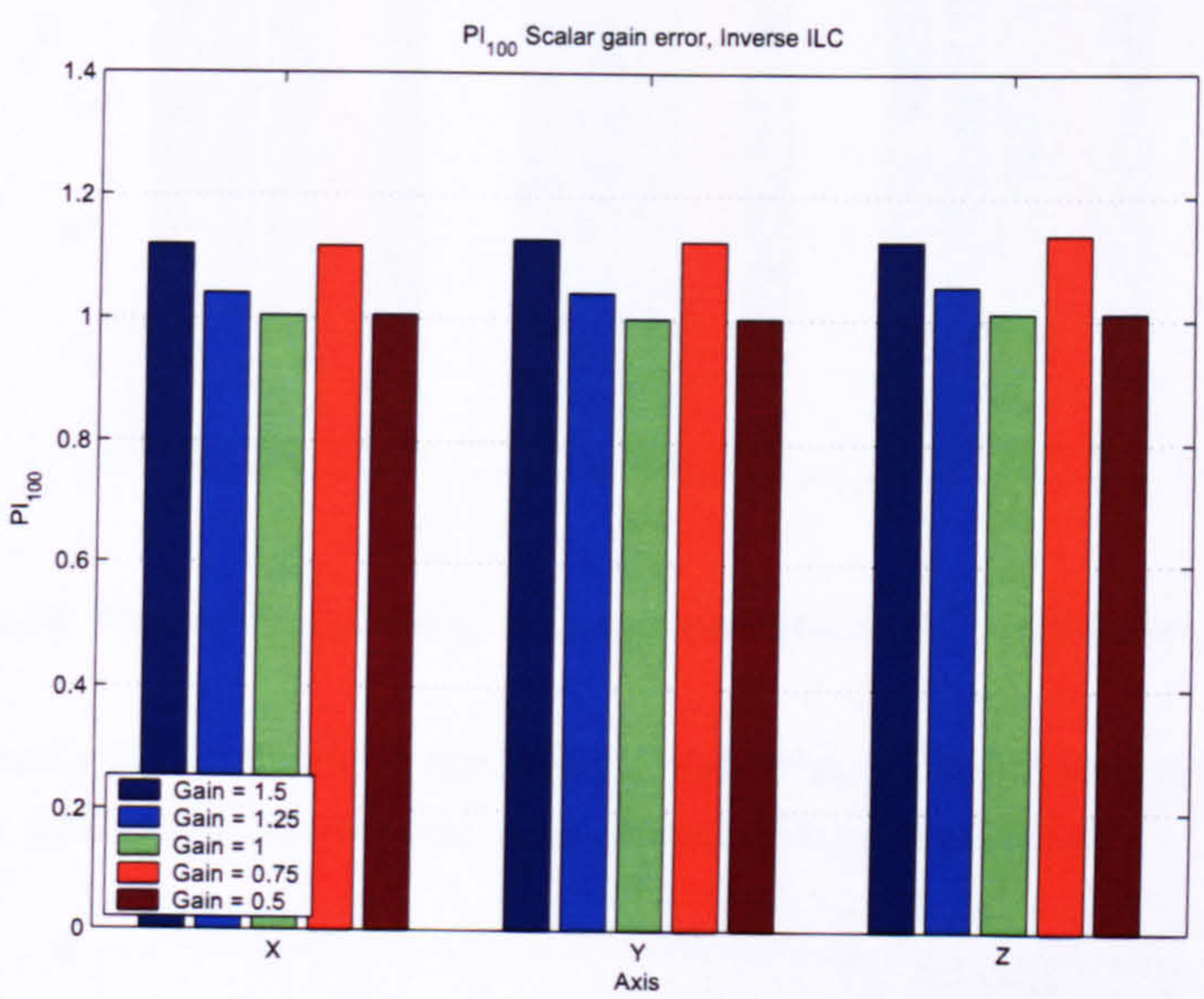


FIGURE 6.13: Comparison of PI_{100} values, inverse ILC with scalar gain error

This effect is more noticeable for the inverse algorithm, where the algorithm is unstable for gains 0.75 and 0.5. The PI_{100} values are a result of learning gain reduction using $\omega_{k+1} = 1||\mathbf{e}||^2$ for both cases. Note how ω_{k+1} is correctly chosen for scalar gain 0.5, where the performance is equivalent to a scalar gain of 1, but not for scalar gain 0.75, where the performance is worse. This could be corrected by adjusting the value of ω_1.

The performance of the norm-optimal algorithm is affected very little by scalar gain error. No adjustment of tuning parameters is required and the increase in PI_{100} values is insignificant.

6.3 Basic ILC Compared to Model-based ILC

PI_N cannot be used to compare the performance of basic and model-based algorithms, due to the different methods of implementation. It is therefore necessary to make a qualitative comparison based on mse performance plots alone. Figures 6.15 and 6.16 show the long-term stability mse results for the P-type parallel controller with aliasing, compared to the three model-based algorithms. The convergence rate of the hybrid P-type controller is noticeably slower than that of the inverse and norm-optimal algorithms,

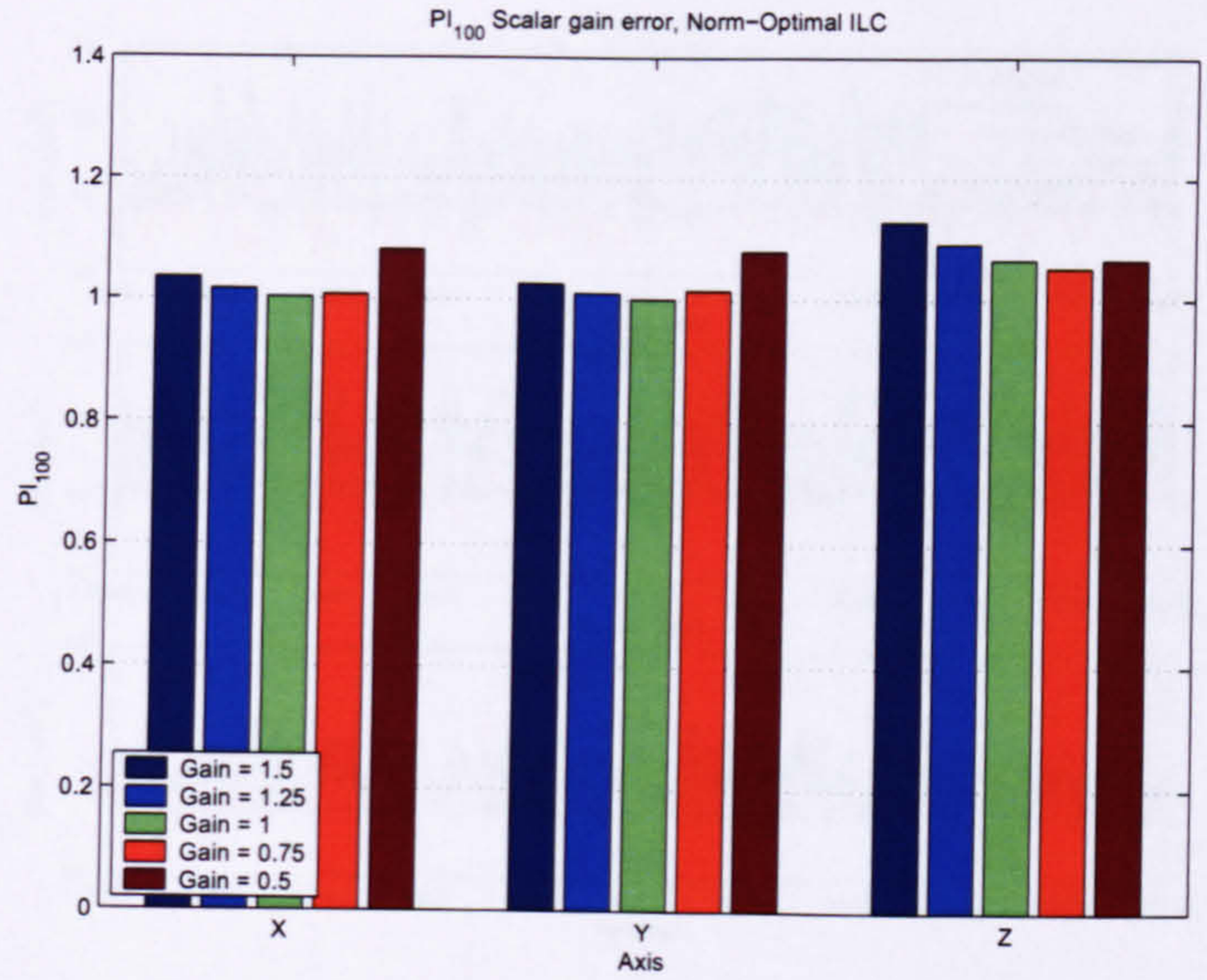


FIGURE 6.14: Comparison of PI_{100} values, norm-optimal ILC with scalar gain error

but is significantly faster than for the adjoint algorithm. The minimum mse level also compares very well to the inverse and norm-optimal implementations.

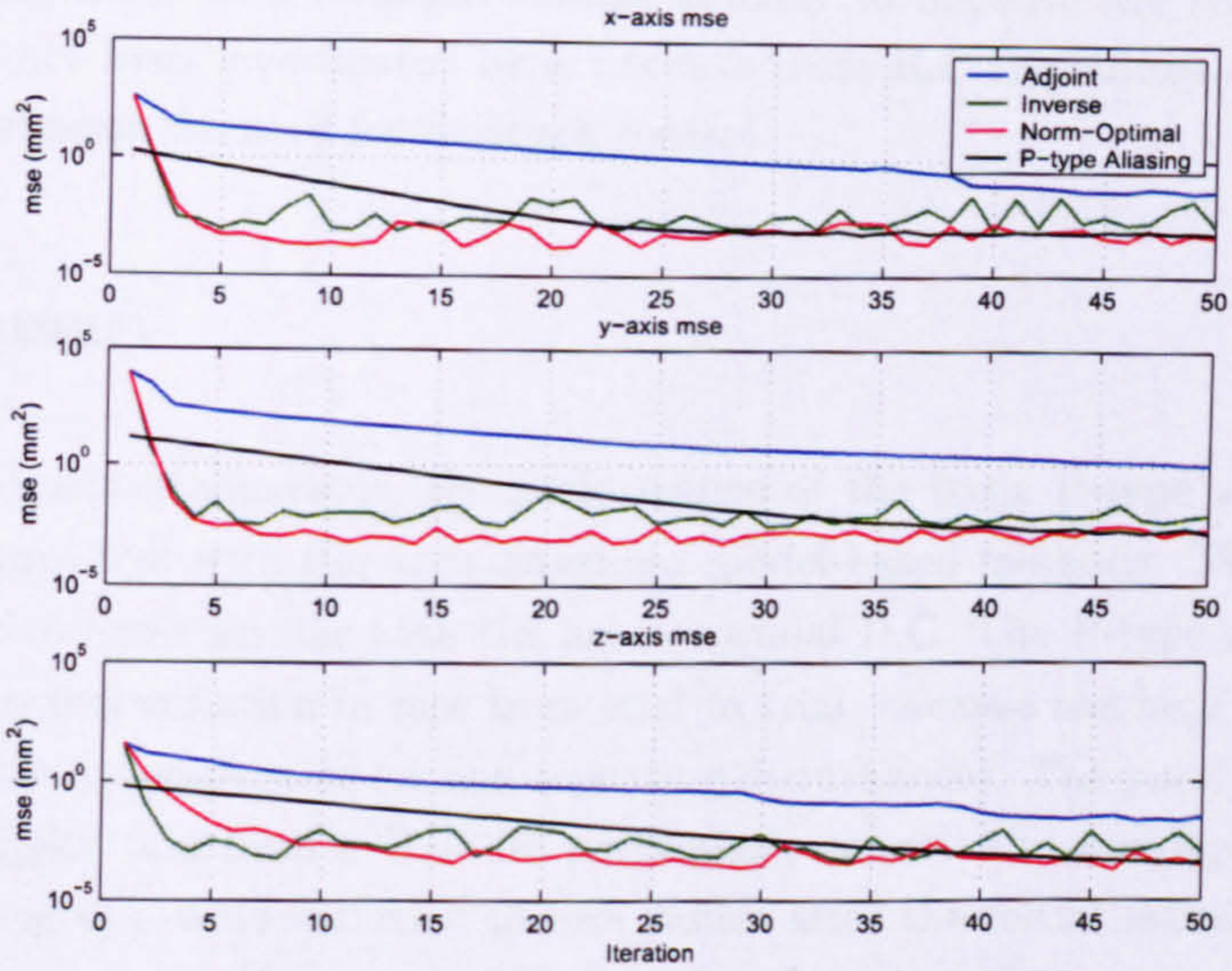


FIGURE 6.15: Performance of basic ILC compared to model-based ILC, iterations 1-50

The most noticeable difference between the basic and model-based algorithms is the significantly smaller variation in the mse of the hybrid P-type controller. The norm-optimal algorithm produces the least variation in mse of the three model-based algorithms, but the hybrid P-type remains more constant, because of the high gain feedback controller being coupled to the learning controller, which compensates for non-repeating disturbances. The adjoint and inverse controllers have no feedback control and therefore the

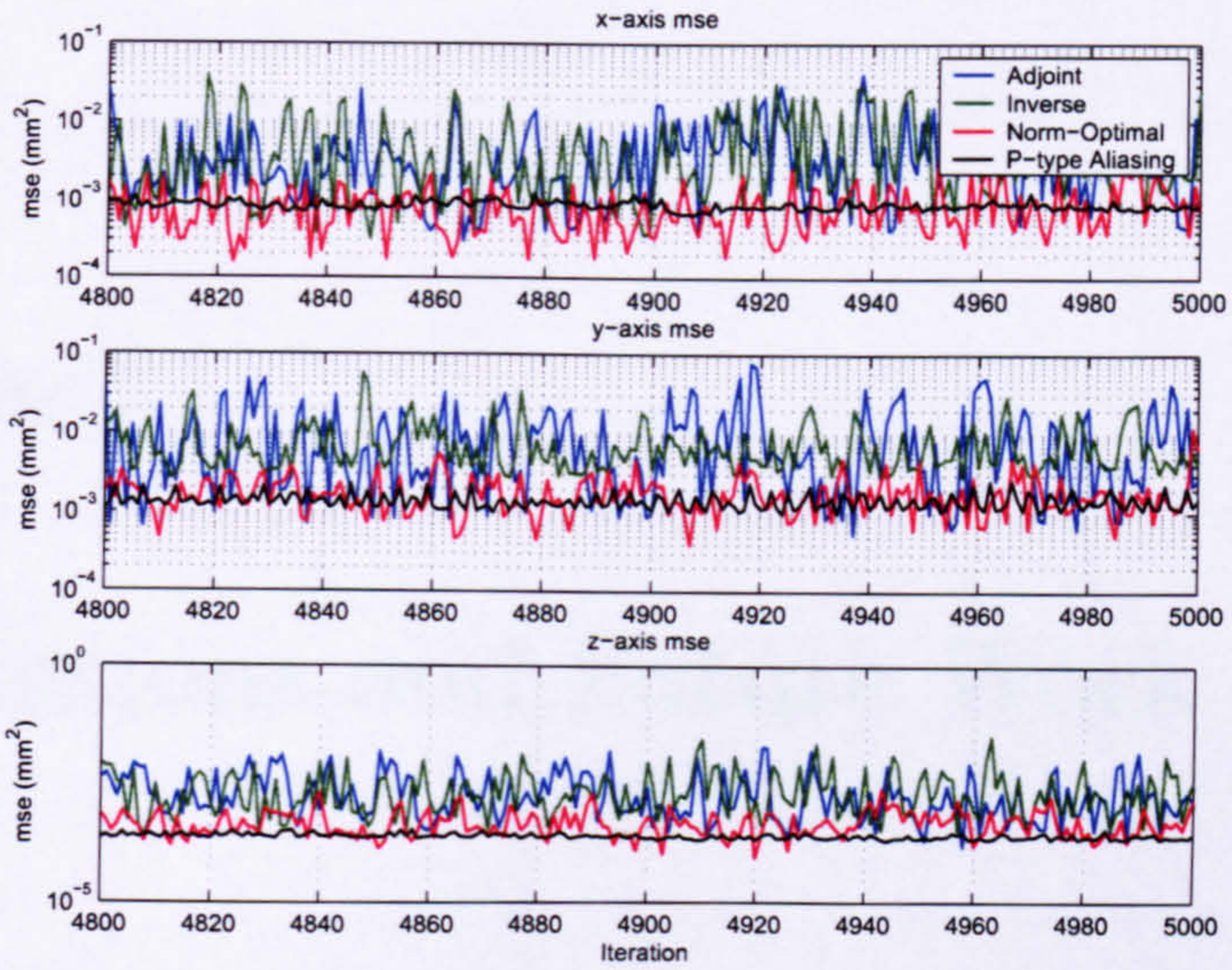


FIGURE 6.16: Performance of basic ILC compared to model-based ILC, iterations 4800-5000

mse variation is greatest, while the norm-optimal controller does have feedback control and the mse variation is smaller. Implementing the adjoint and inverse algorithms in a hybrid arrangement with feedback control is likely to improve the steady state mse level, but has not been investigated here, because these ILC algorithms are designed to be operated without the need for feedback control.

6.4 Summary

On this experimental apparatus, the performance of the basic P-type algorithm with aliasing compares well with the more advanced model-based methods. The tracking error is reduced to levels similar with the norm-optimal ILC. The P-type algorithm also results in much less variation in mse from trial to trial, because the high gain feedback controller is able to compensate for non-repeating disturbances. The purely feed-forward algorithms, adjoint and inverse ILC are particularly sensitive to non-repeating disturbances, resulting in a wide variation in mse values after the initial learning. However, these algorithms are significantly more tolerant to initial state error, because they do not produce harsh step inputs at the start of the trial, in an attempt to compensate for the initial error. All of the model based algorithms shows a good degree of robustness to plant modelling error, if suitable measures, such as the use of filtering or adjustment of tuning parameters are implemented.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

Four iterative learning control algorithms: basic P-type, adjoint, inverse and norm-optimal have been implemented on a custom built test facility, designed to represent a 'pick and place' application, which is found in many different industrial applications. Each algorithm has been tested with respect to long-term stability, robustness to initial state error and robustness to plant modelling error. A performance index has been proposed to allow a fair, quantitative evaluation of algorithm performance, to evaluate the four main factors which define performance: long-term stability, convergence rate, transient performance and minimum tracking error.

The P-type algorithm is found to be unsuitable for integrating plants when implemented alone, because the input profile which the algorithm must learn bears no relation to the shape of the reference trajectory, but is related instead to the derivative of the reference. The addition of a PID feedback controller is used to compensate for the integrating plant by providing a good estimate of the input profile, which the ILC can build upon, correct and refine. Both variants of the hybrid controller: series and parallel were tested and found to produce similar performance characteristics even though the systems are distinctly different.

The hybrid, PID with P-type ILC controllers successfully improve the tracking performance well beyond the level which can be achieved by the PID controller. Increasing the learning gain improves the convergence rate, but also accelerates the onset of instability, which manifests itself by an increase in mse following the initial reduction and the mechanical vibration of the robot.

Investigating the mechanism of instability reveals two processes which occur simultaneously and corrupt the learnt input profile. The first mechanism is related to resonant frequencies within the mechanical construction of the plant. Resonant frequencies rapidly

add 180 degrees of phase lag to the tracking error signal. The P-type algorithm is unable to tolerate phase shift larger than 180 degrees, because the simple structure of the algorithm enhances the resonance instead of reducing it. The second mechanism is concerned with the build-up of noise and non-repeating disturbances within the iteration loop. At each iteration the majority of the information supplied to the learning controller is useful for reducing tracking error. However, the learning controller is unable to compensate for random and non-repeating components of the error signal. These accumulate in the iteration loop, their magnitude increasing at each iteration, until the plant input signal becomes sufficiently corrupted and causes the plant to behave erratically.

The use of several filtering techniques was investigated in an attempt to remove resonant and non-repeating disturbances from the iteration loop after each iteration. Band-stop filtering successfully limits the magnitude of resonant frequencies and a carefully designed filter causes minimal phase shift either side of the stop band. However, the band-stop filter is unable to remove high frequency noise signals. Low-pass filtering successfully attenuates resonances as well as noise and achieves long-term stability. But the phase shift caused by adding the filter, severely limits the improvement in tracking performance which can be gained by using a learning controller. The zero-phase filtering method is particularly well suited to ILC systems and doubles the attenuation of resonant and noise frequencies without inducing any phase shift. This results in an excellent reduction in tracking error and long-term stability.

A new approach using frequency aliasing to remove unwanted frequencies was also proposed as an alternative to traditional filtering. The error signal is sampled at a lower frequency than the feedback control frequency and the reduced sampling points are connected by linear interpolation. The linearised signal is then re-sampled at the same frequency as the feedback controller and fed to the plant. This technique makes use of Shannon's sampling theorem to alias unwanted frequencies to a lower frequency, hence preventing them from building up in the iteration loop. The aliasing technique is particulary well suited to the gantry robot and improves tracking performance beyond the level of the zero-phase filter implementation. The aliasing technique also has potential for use in repetitive control systems, where zero-phase filtering is much more difficult to achieve.

Initial implementation of the adjoint algorithm revealed that it is inherently long-term stable, without any requirement for feedback control or any form of signal filtering. The majority of the tracking error is removed within 50 iterations and minimum mse is achieved in 1500 iterations. In contrast, the inverse algorithm converges rapidly to the reference trajectory within 2 iterations, but becomes unstable within 4 iterations. Use of the learning gain tuning parameter ω extends the period of stability at the expense of convergence rate. Zero-phase filtering must be used to achieve a system which is truly long-term stable. The norm-optimal algorithm is long-term stable without additional control or filtering and achieves minimum mse in 4 to 5 iterations. Investigation of

the parameters q and r revealed that their ratio, rather than individual magnitudes, determines the performance of the algorithm.

Algorithms which are assisted by feedback control, such as the hybrid P-types and the norm-optimal are less affected by initial state error than purely feed-forward algorithms, because the high gain feedback controller rapidly reduces the error within the first few sample instants, resulting in minimal impact on the feed-forward learning controller. The performance of purely feed-forward algorithms is particularly affected by initial error, because the algorithm attempts to compensate for the error in the previous iteration. But the initial error at the next iteration is different, and the compensation made between iterations is incorrect. However, the purely feed-forward algorithms, particularly the adjoint algorithm, can tolerate much larger initial error bounds, without producing large step inputs to the plant. This results from the absence of a high gain feedback controller as part of the algorithm.

The effects of model order reduction on the performance of the model-based algorithms are different. The adjoint algorithm is able to tolerate the use of 1st order models without loss of stability, but the tracking performance is noticeably reduced. In comparison, the lack of dynamic data in the 1st order models causes the norm-optimal algorithm to become unstable. Stability is restored by means of zero-phase filtering. The inverse algorithm is least affected, because the tracking performance only degrades fractionally and stability is maintained by the zero-phase filter, which is intrinsically required by the algorithm.

In contrast, scalar gain error has least effect on the norm-optimal controller, which can tolerate gain discrepancy of both a positive and negative nature with minimal loss in performance. The adjoint and inverse algorithms require the use of the tuning parameter ω_{k+1} to compensate for scalar gains smaller than 1. In general, the larger the gain discrepancy in either direction, the larger the impact on tracking performance.

The combination of ILC with a feedback controller produces a hybrid which is far better equipped to compensate for non-repeating disturbances, resulting in a system which consistently produces low tracking error with very little variation between iterations. Model-based algorithms tend to learn much more rapidly and are capable of tolerating larger initial error without loss of stability. Algorithms such as the inverse or norm-optimal should be implemented when very rapid convergence to minimum error is required in a minimum number of iterations, for example when the iteration time period is large (minutes or hours) such as chemical batch processing or hard disk drive track placement. If the plant is associated with an ongoing task where thousands or millions of iterations are performed in batches, for example food processing or robotic assembly, the simplicity of a hybrid PID-P-type ILC becomes more attractive. The slightly longer convergence to minimum error is insignificant in comparison with the improved constancy of error reduction, easier programming and ease of tuning.

7.2 Future Work

Further development of the work presented in this thesis involves the investigation of repetitive control on the gantry robot and associated peripheral devices. Repetitive control techniques are remarkably similar to iterative learning control techniques, except that the algorithm must be computed in real-time as the test proceeds: there is no stoppage time between repetitions and the data cannot be processed in batches. The terminal conditions at the end of each repetition are the initial conditions for the next repetition, there is no resetting between trials. Repetitive control is therefore well suited to an extension of the test facility, which involves the conveyor beneath the gantry robot. When the conveyor moves at a constant velocity, the robot must synchronise both velocity and position to place objects on it accurately.

The robot will be required to place payloads consisting of tins of peas onto the moving conveyor with high accuracy. The conveyor is a 6 meter long, type XK plastic chain conveyor from Flexlink. The position of the conveyor is measured by means of a rotary encoder mounted onto the conveyor by means of a mounting bracket (design drawings can be found in Appendix C Section C.1). Applications of this type occur frequently in industry and are particularly used in food processing, where containers must be filled with food or liquids. The payloads will then travel the length of the conveyor, where they will slide down a return chute (design drawings can be found in Appendix C Section C.2) onto a second conveyor, which will lift the cans into a storing and dispensing mechanism. A custom built dispenser will isolate individual payloads from the storage slide and prepare them for the gantry robot to collect with its electromagnet end effector (design drawings can be found in Appendix C Section C.3). In this way, the payloads will be continuously cycled around the plant and many repetitions will be performed with a minimal number of payloads. A design drawing of the completed test facility can be seen in Appendix C Figure C.40.

In particular, this test facility will enable the investigation of the effects of variation in payload upon tracking performance. Changes in mass and the associated inertia will be of particular importance to the evolution of the plant input, for example, the impact of removing a payload for one repetition, on the tracking performance for the following payload.

The initial stages of algorithm learning, during the first repetitions, will be of particular importance to industrial applications. If the trajectory tracking is particularly poor during the first trials, the gantry may not collect the payload, and may not place it on the conveyor at all. In these cases, payloads and learning time will be wasted, which will have an impact on throughput. When the robot does eventually collect the first payload, the change in system dynamics may have a significant impact on tracking error and stability.

Testing both repetitive and iterative learning control on the same plant will allow the investigation of the similarities and differences which exist between the two learning mechanisms.

It will also be possible to investigate the use of a dual learning loop system, where the inner learning loop is used to match the plant output to the reference trajectory, while the outer loop is used to adjust the reference trajectory itself. For example, if the robot is controlled by a learning controller, but the conveyor is controlled by standard feedback control, the improvement in robot tracking performance will be insignificant compared to the error discrepancy between the robot and the conveyor. It may be possible to use a second learning loop to adjust the reference trajectory of the robot to compensate for the conveyor error and reduce the relative error.

Finally, the zero-phase method can be implemented with success within the ILC framework although, its implementation in the repetitive control framework is hindered by the problems inherent to filtering the data forwards and backwards in batches while the plant is in operation. The aliasing technique developed in this thesis successfully stabilises the hybrid P-type ILC and allows excellent tracking performance. Its application could be adapted to continuous operation of the plant in repetitive control, requiring only a few calculations at each sample instant, implying that the aliasing technique could therefore be a very powerful tool for repetitive systems.

Appendix A

Compliant Arm Design Drawings

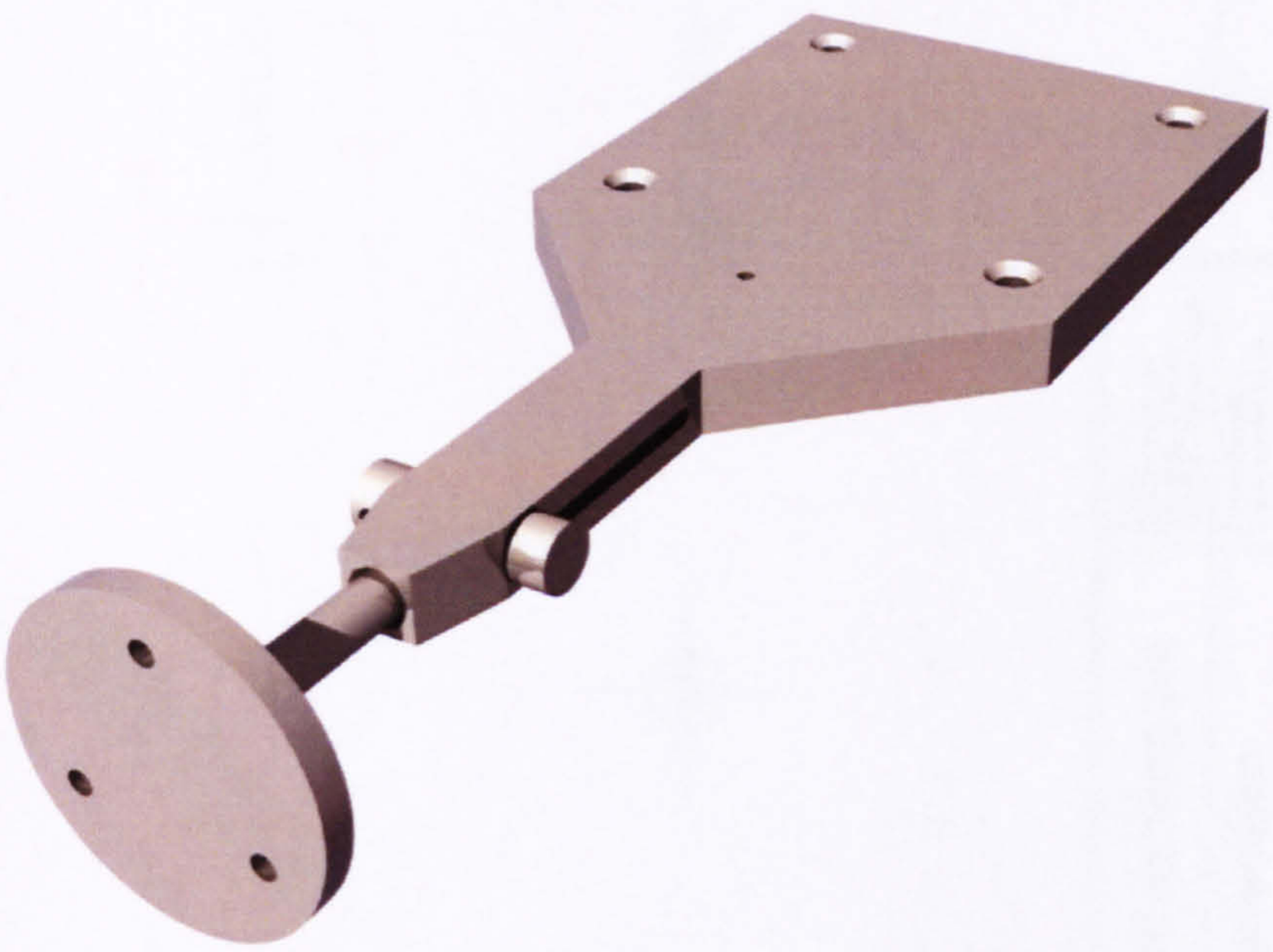


FIGURE A.1: 3D representation of the compliant arm

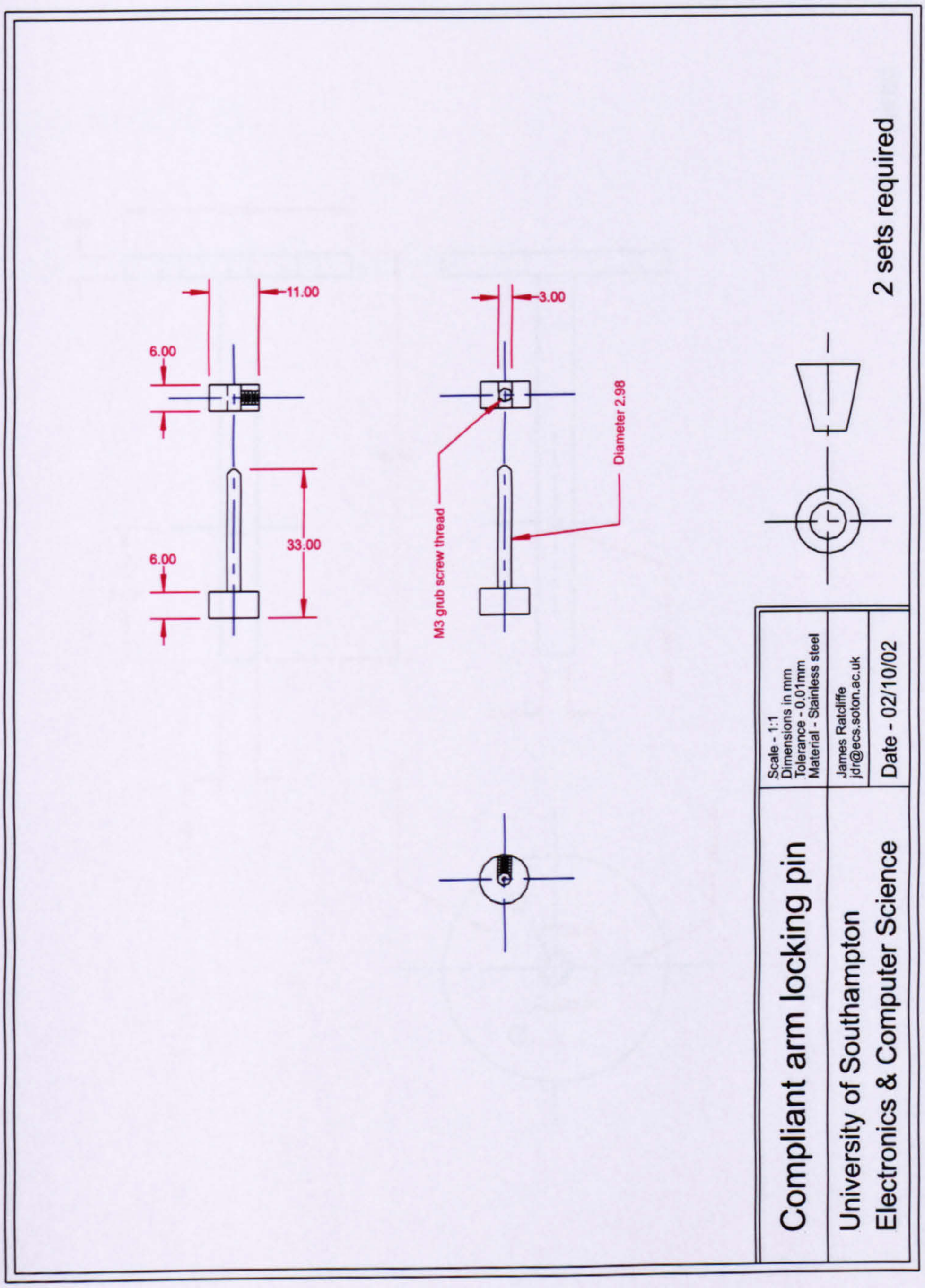


FIGURE A.2: Compliant arm locking pin

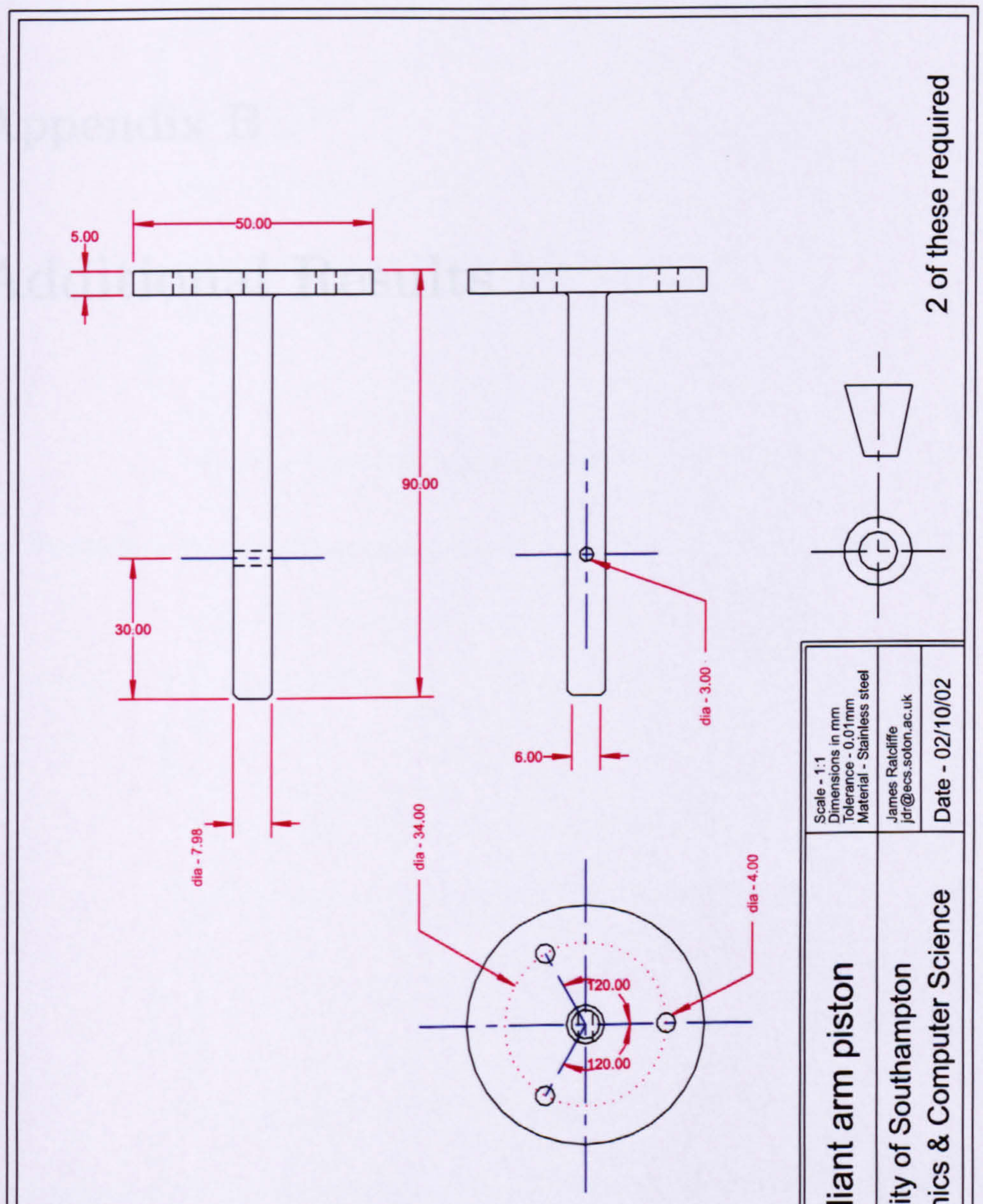


FIGURE A.3: Compliant arm piston

Appendix B

Additional Results

B.1 PID Controller

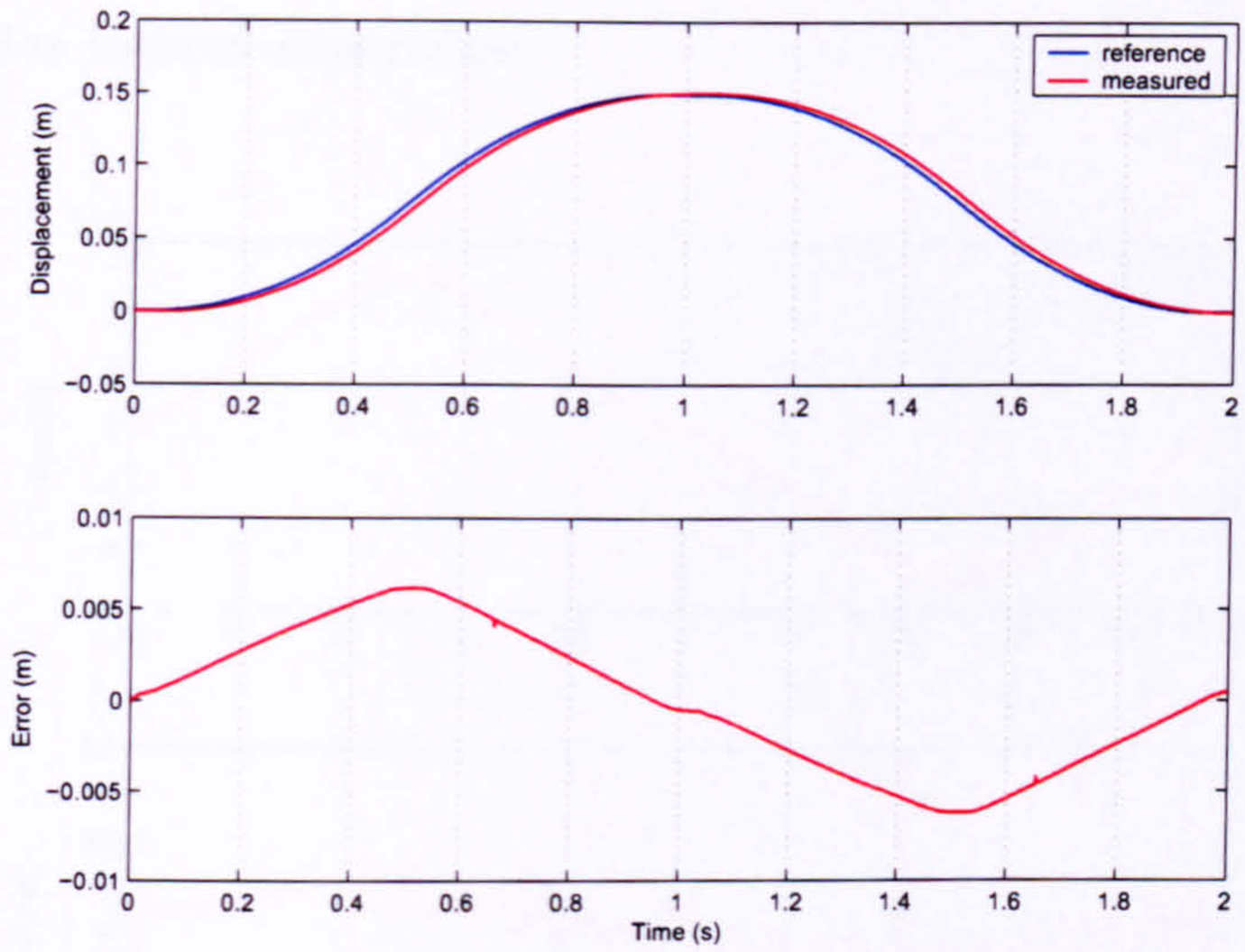


FIGURE B.1: Y-axis tracking performance and error at iteration 1000 (PID controller)

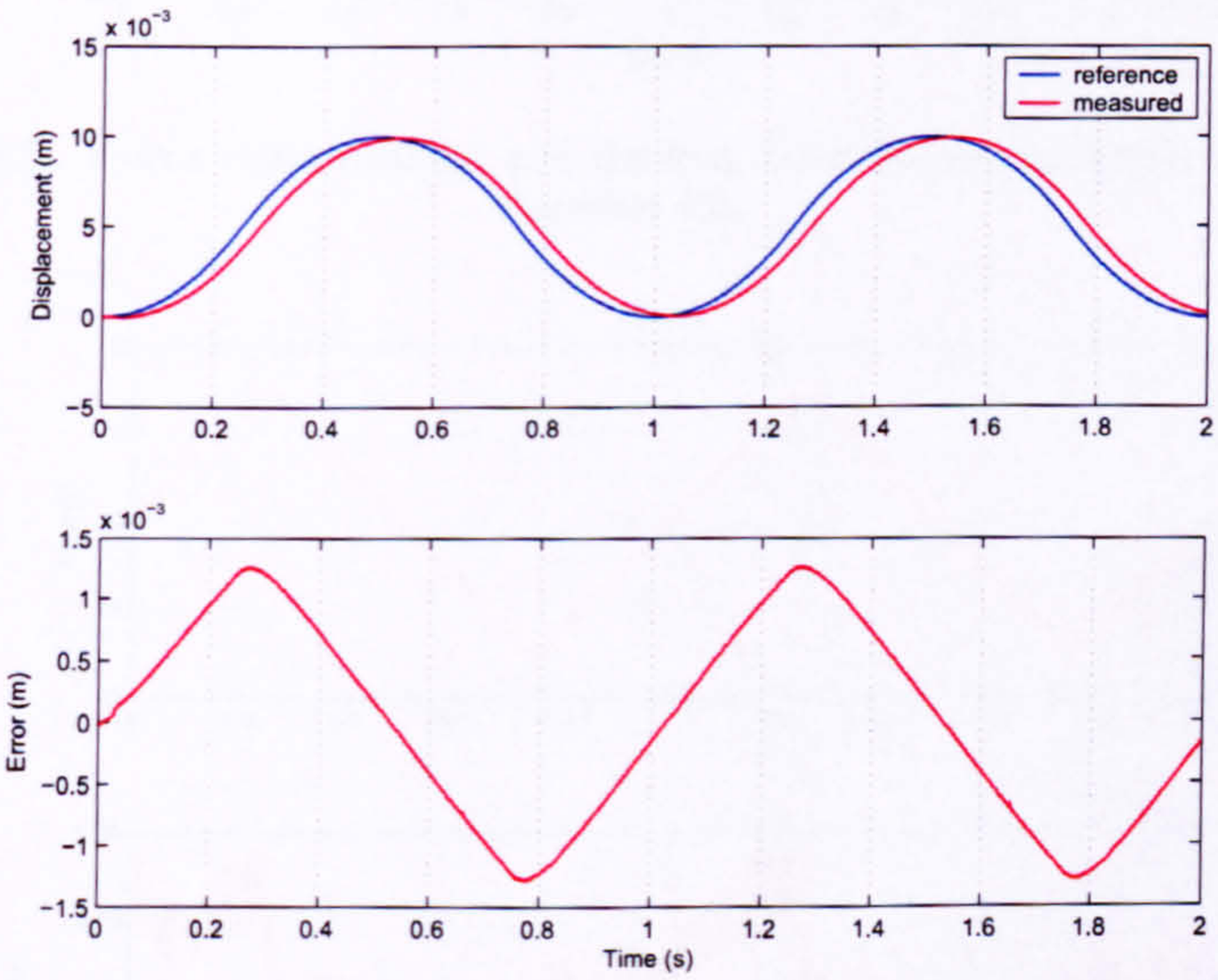


FIGURE B.2: Z-axis tracking performance and error at iteration 1000 (PID controller)

B.2 Basic P-type ILC

B.2.1 Series hybrid controller

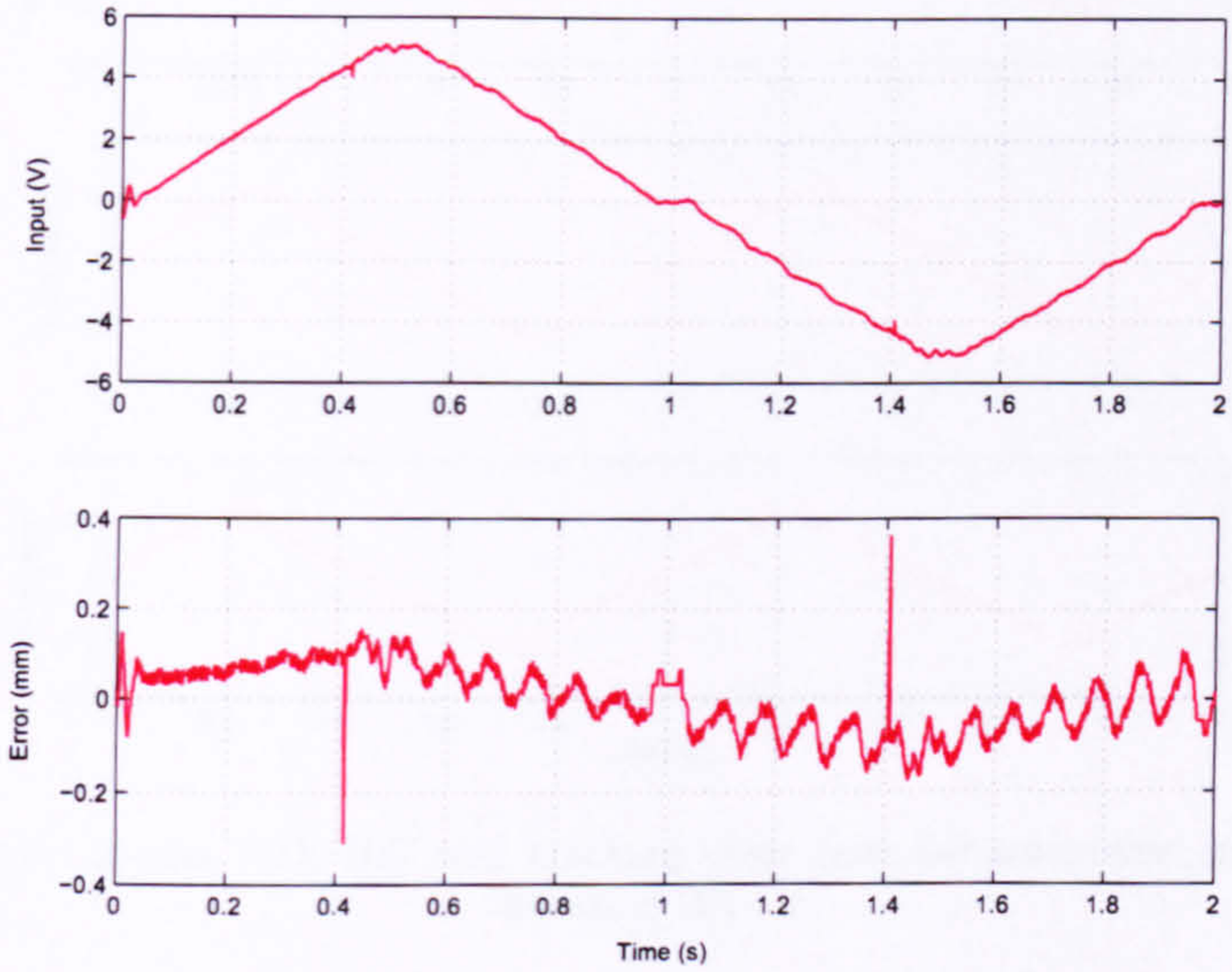


FIGURE B.3: Y-axis input demand and tracking error (series controller, gain = 0.1, iteration 40)

FIGURE B.4: Z-axis input demand and tracking error (series controller, gain = 0.1, iteration 40)

B.2.2 Parallel hybrid controller

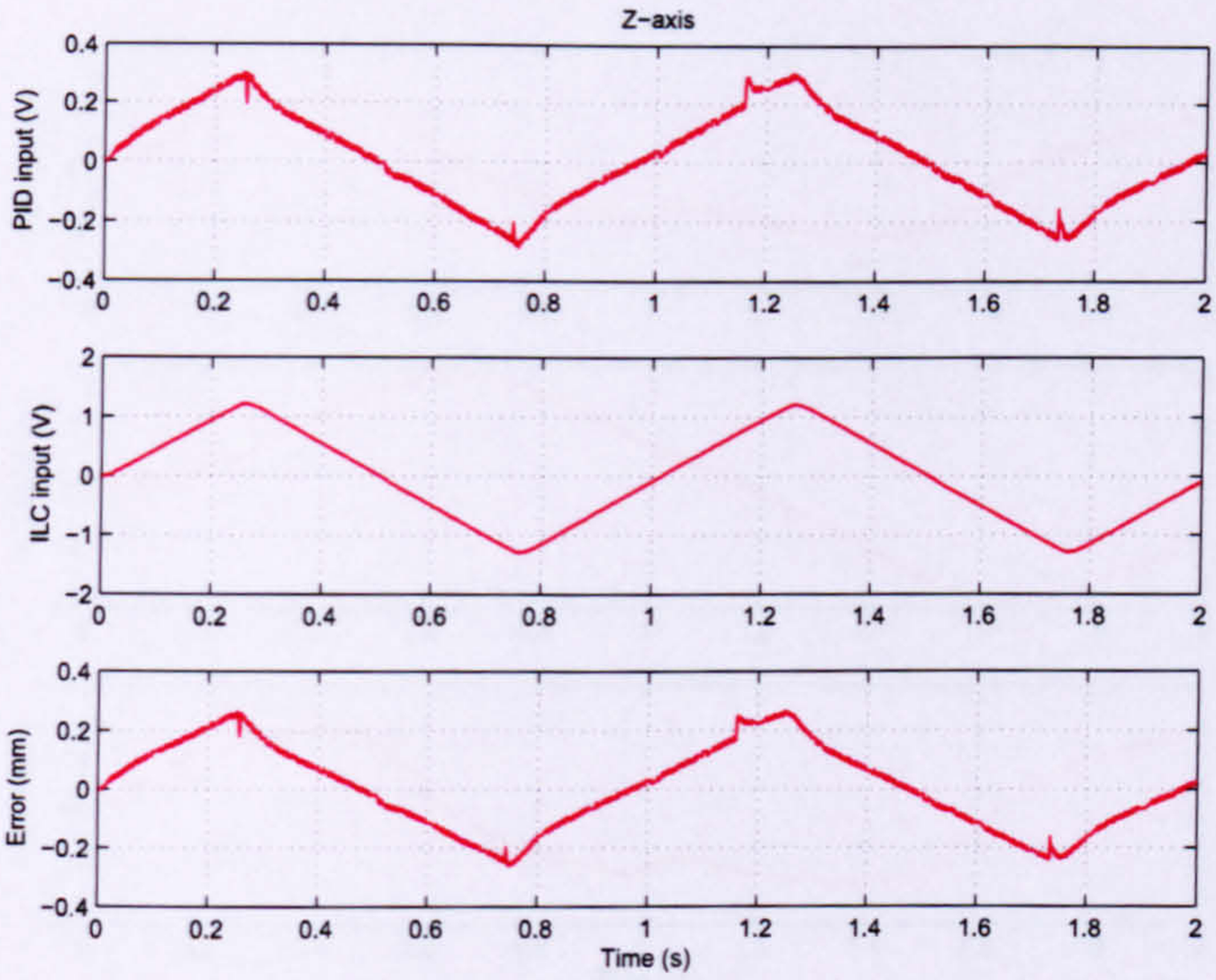


FIGURE B.5: Z-axis PID, ILC and tracking error (parallel controller, gain = 100, iteration 20)

B.2.3 Low-pass filter

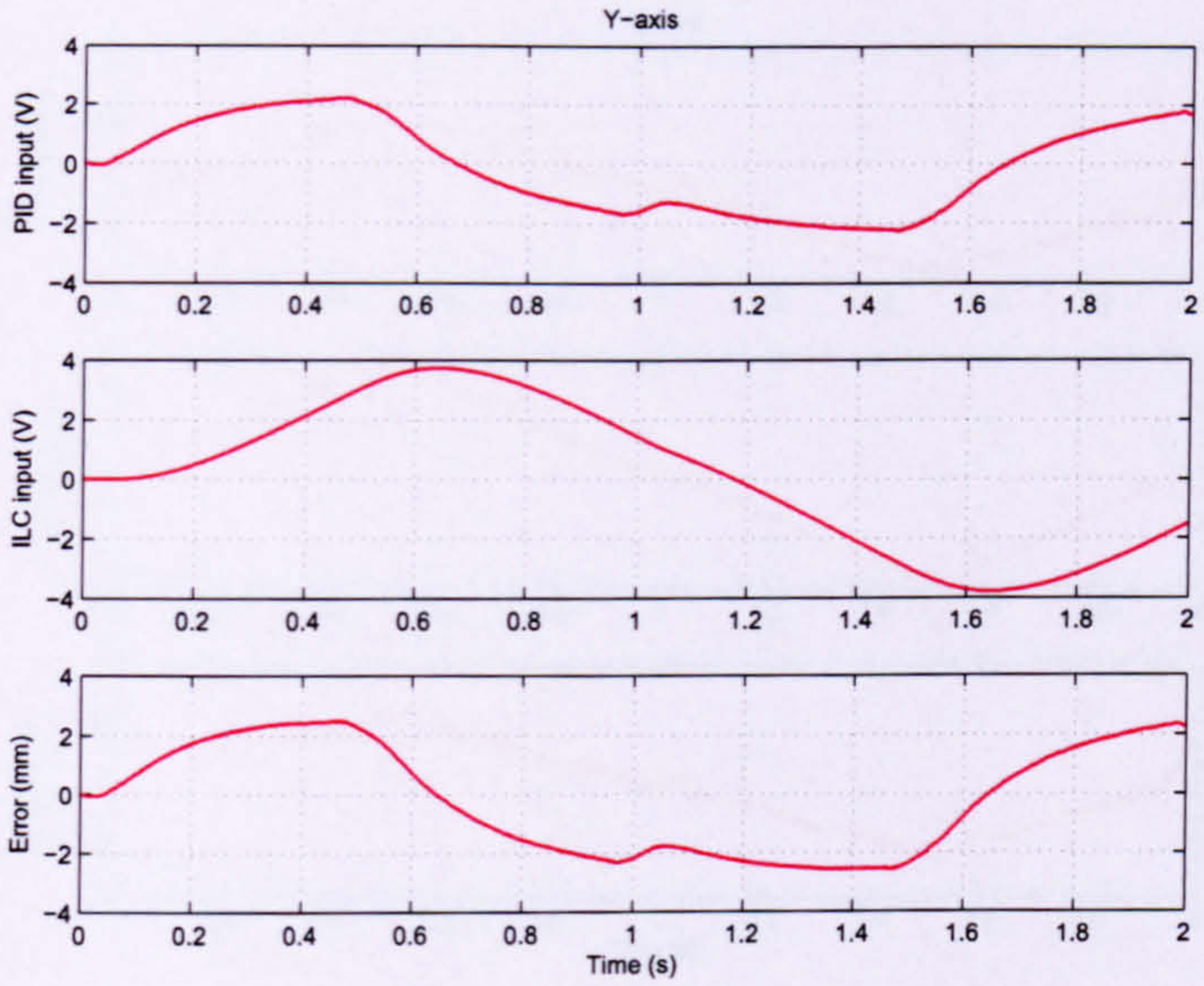


FIGURE B.6: Y-axis PID, ILC and tracking error (low-pass filter, gain = 100, iteration 5000)

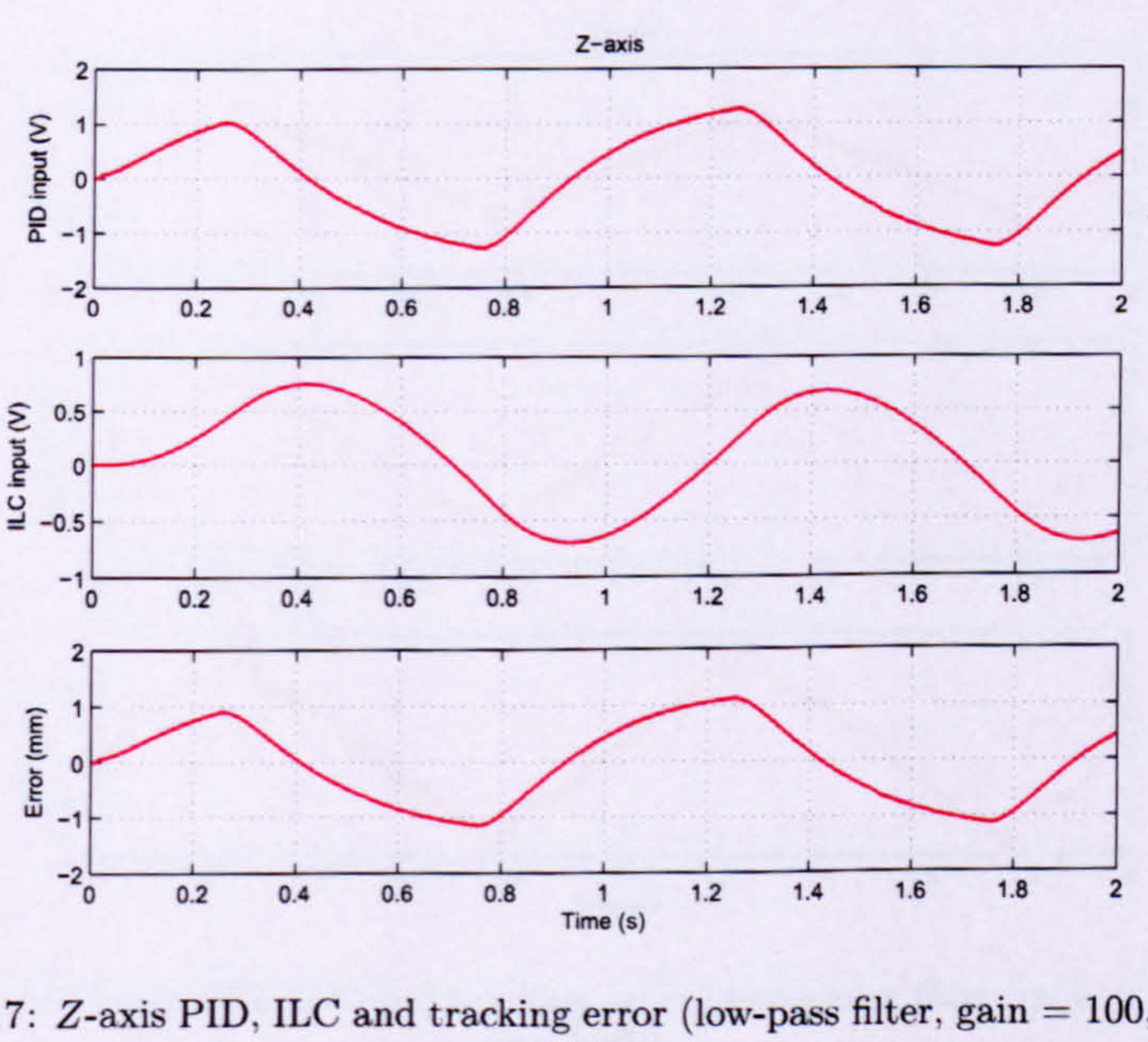


FIGURE B.7: Z-axis PID, ILC and tracking error (low-pass filter, gain = 100, iteration 5000)

B.2.4 Zero-phase filter

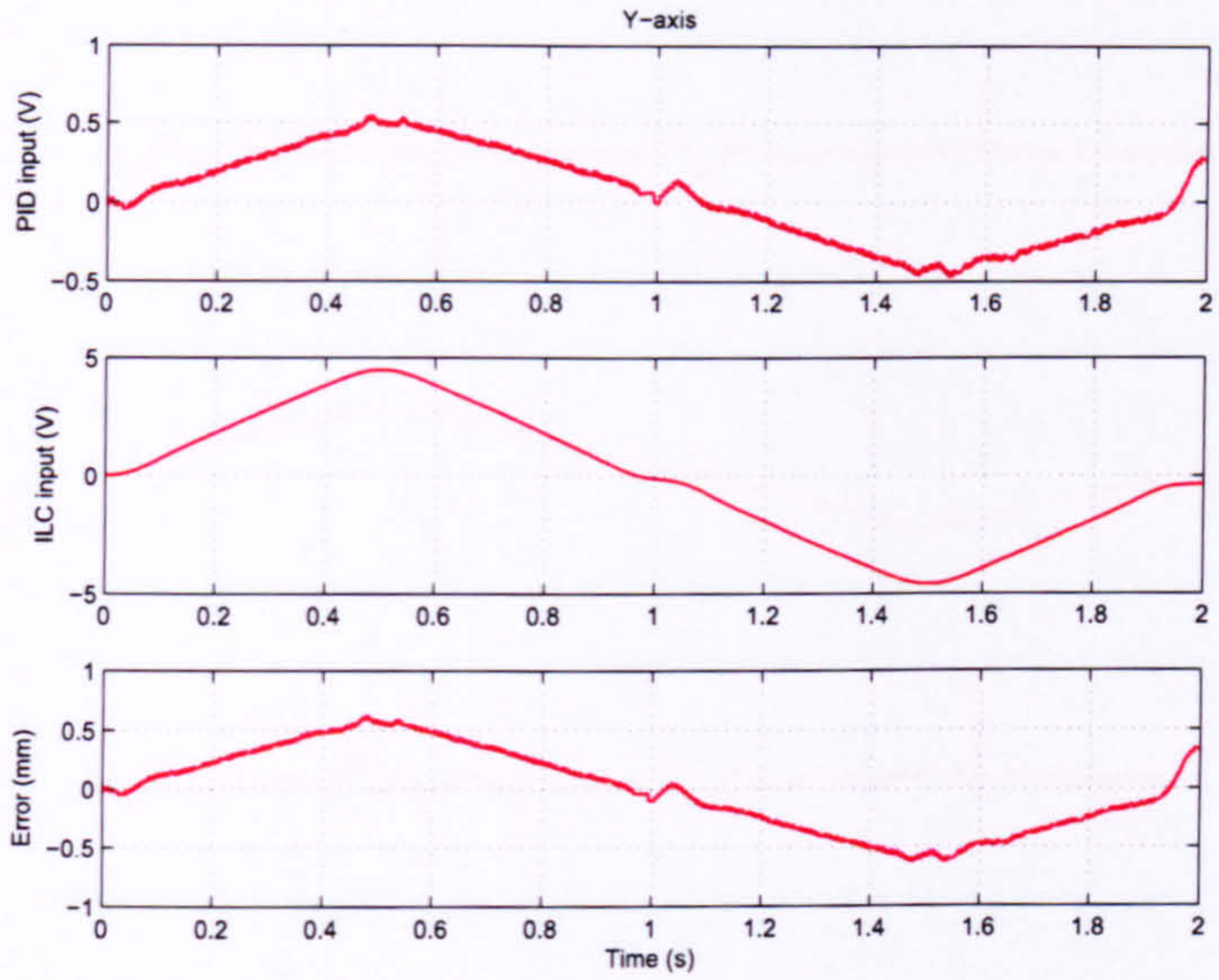


FIGURE B.8: Y-axis PID, ILC and tracking error (zero-phase filter, gain = 100, iteration 5000)

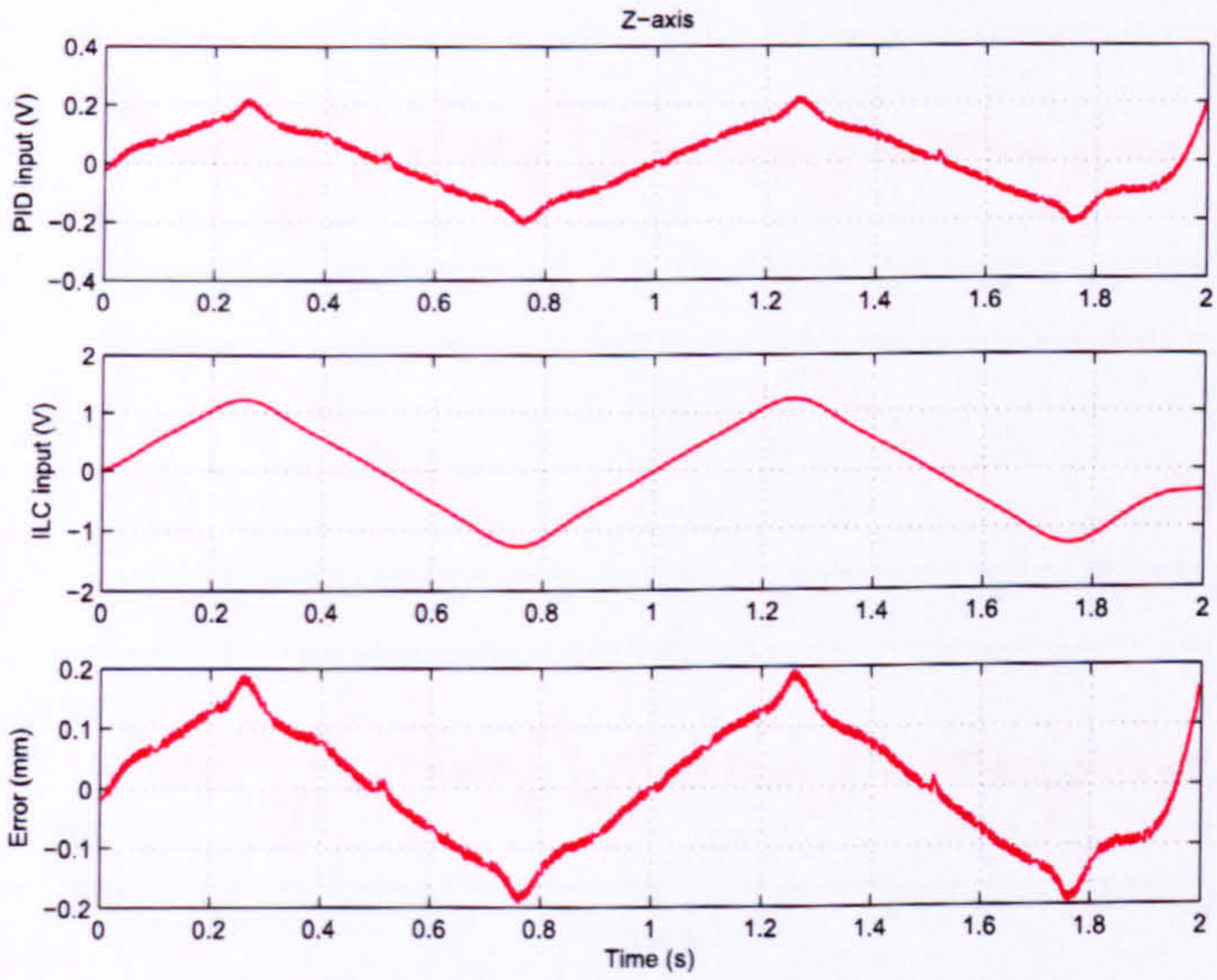


FIGURE B.9: Z-axis PID, ILC and tracking error (zero-phase filter, gain = 100, iteration 5000)

B.2.5 Aliasing filter

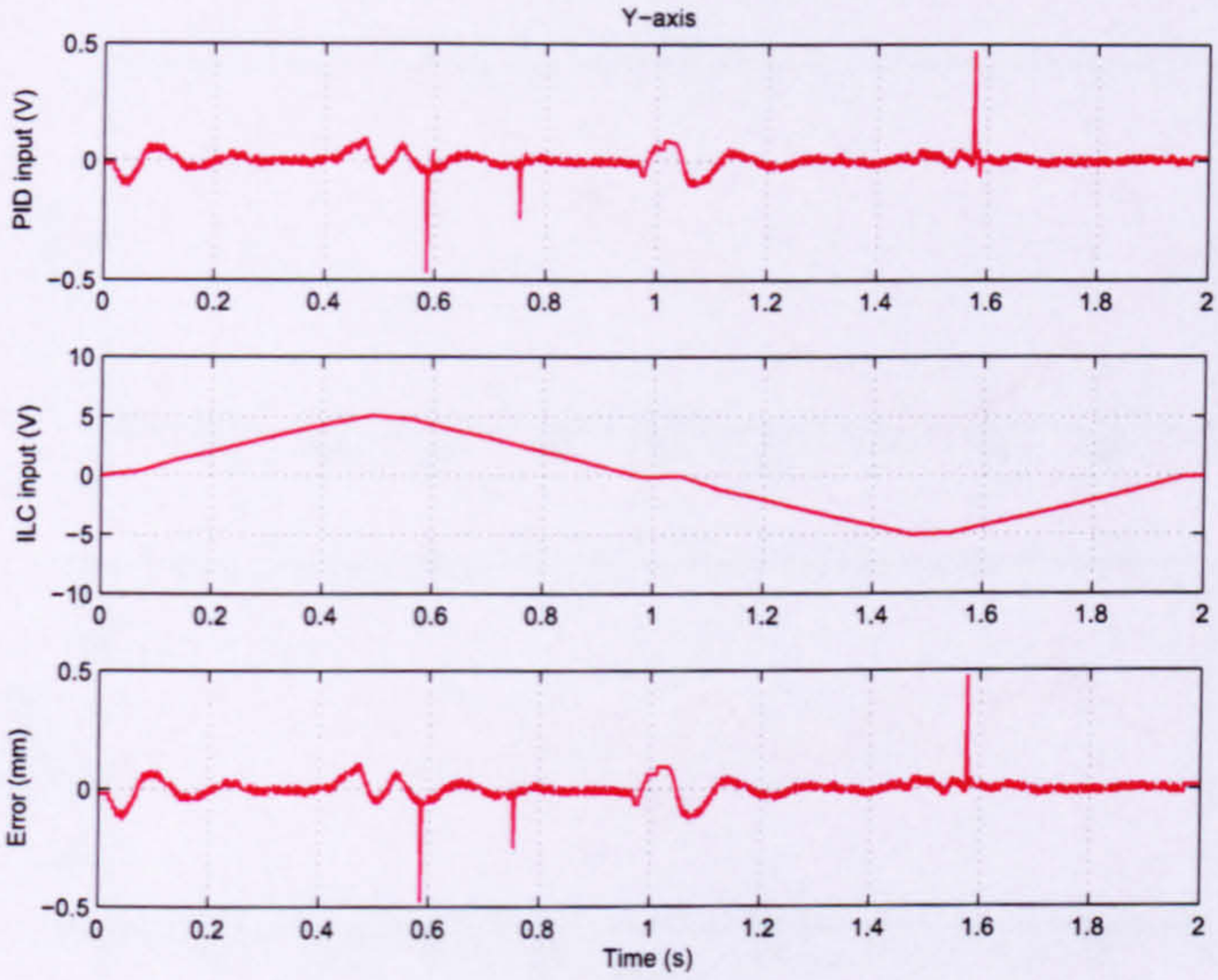


FIGURE B.10: Y-axis PID, ILC and tracking error (aliasing filter, gain = 100, iteration 5000)

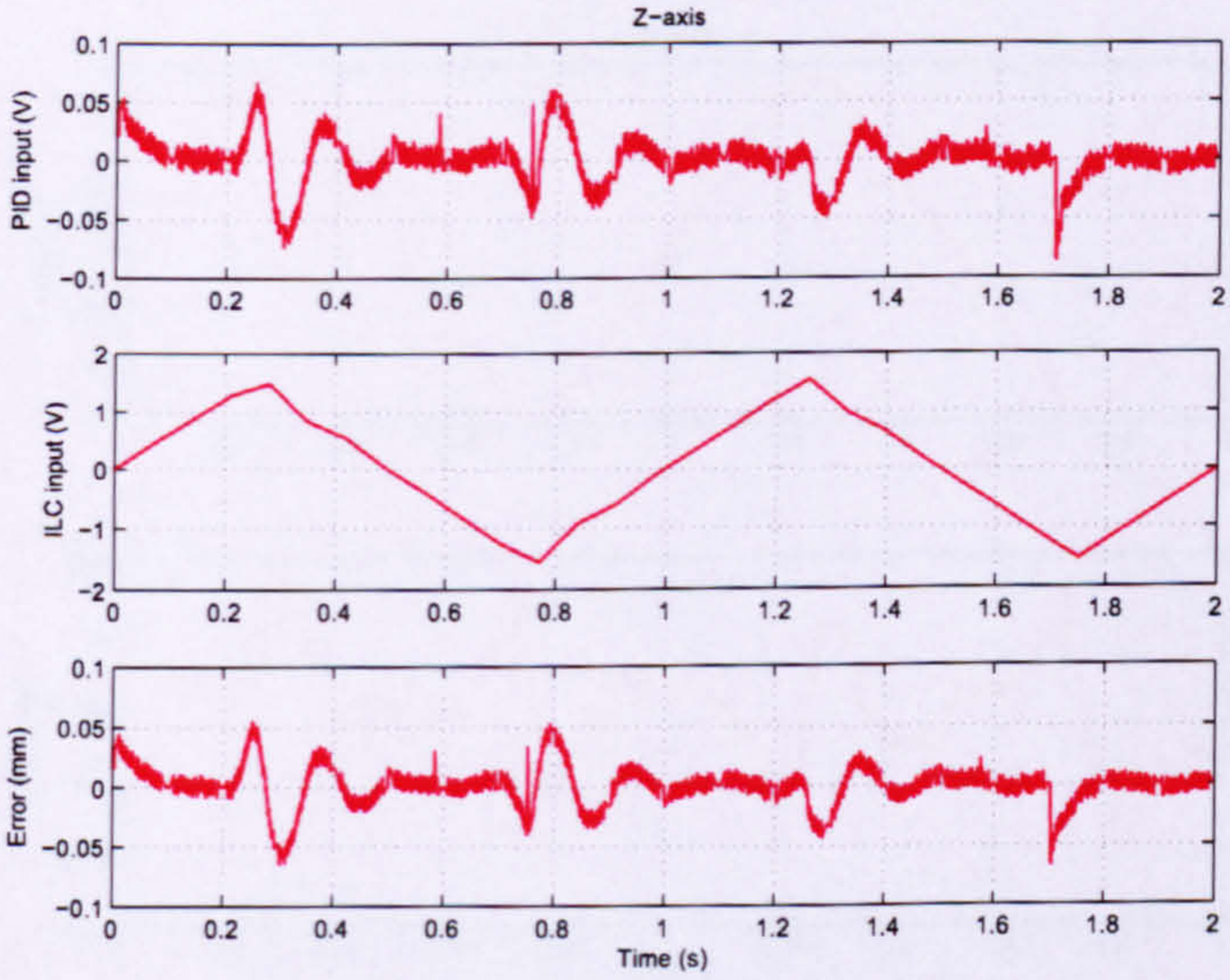


FIGURE B.11: Z-axis PID, ILC and tracking error (aliasing filter, gain = 100, iteration 5000)

B.3 Adjoint ILC

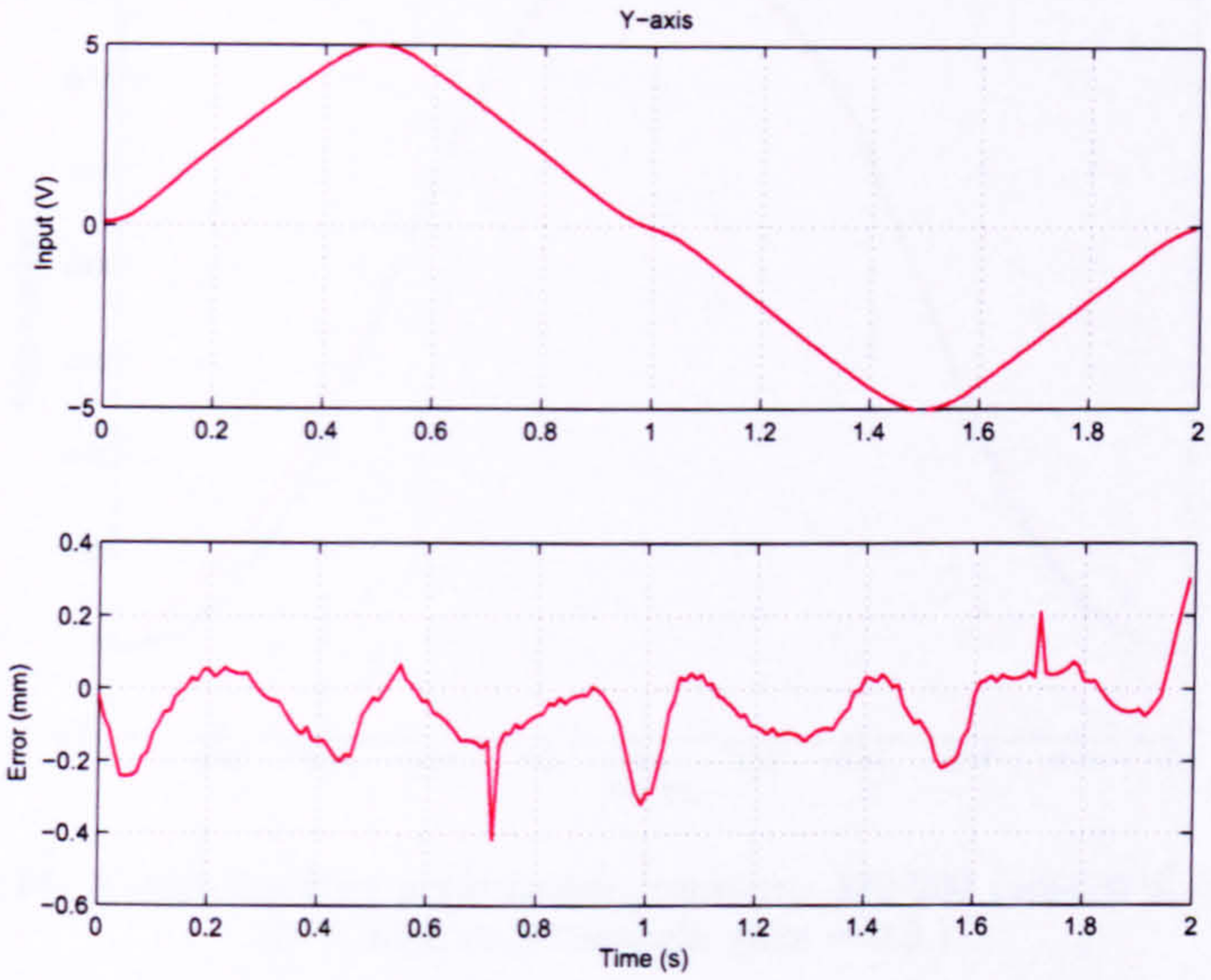


FIGURE B.12: Y-axis input demand and tracking error (adjoint ILC, 1st order models, iteration 500)

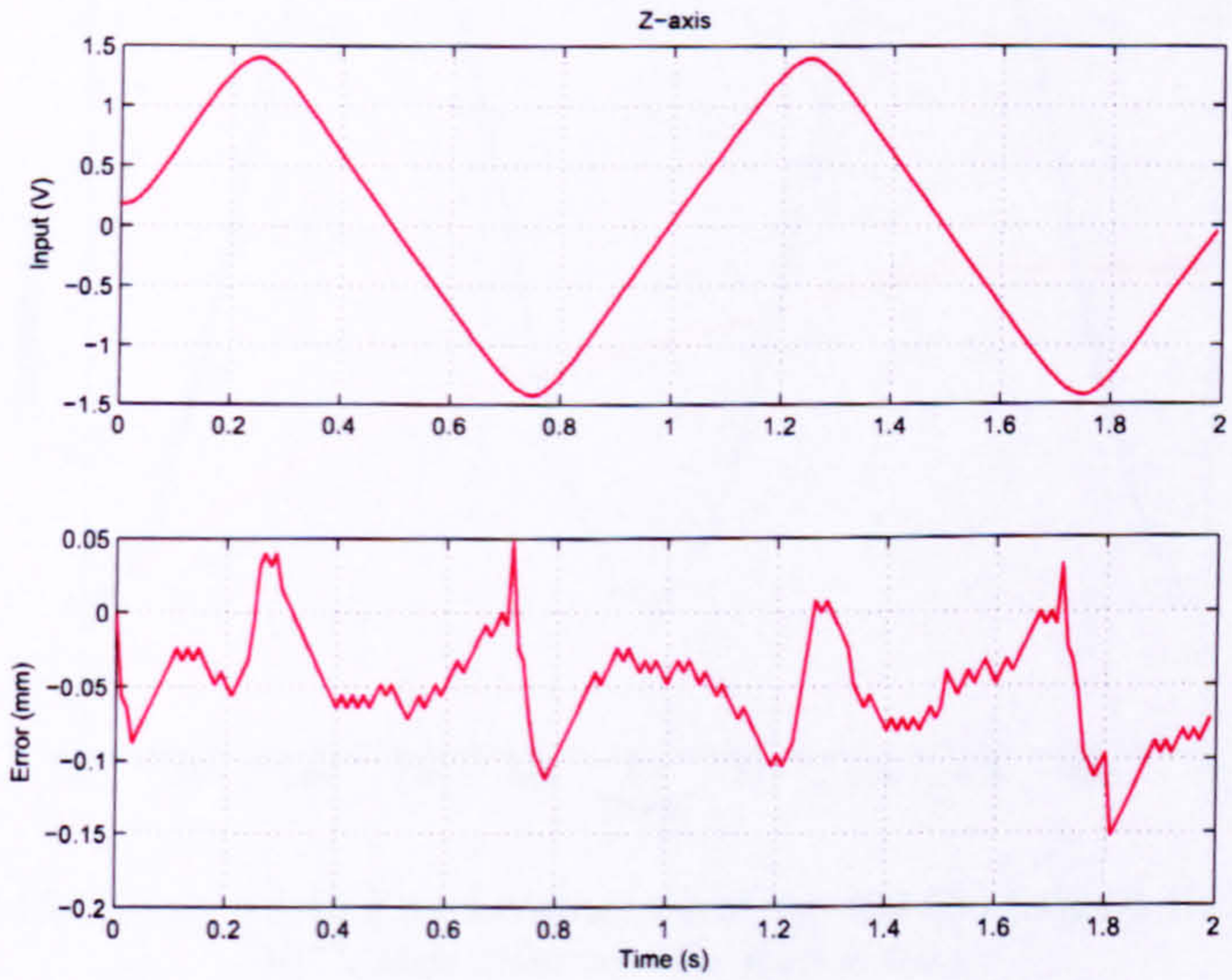


FIGURE B.13: Z-axis input demand and tracking error (adjoint ILC, 1st order models, iteration 500)

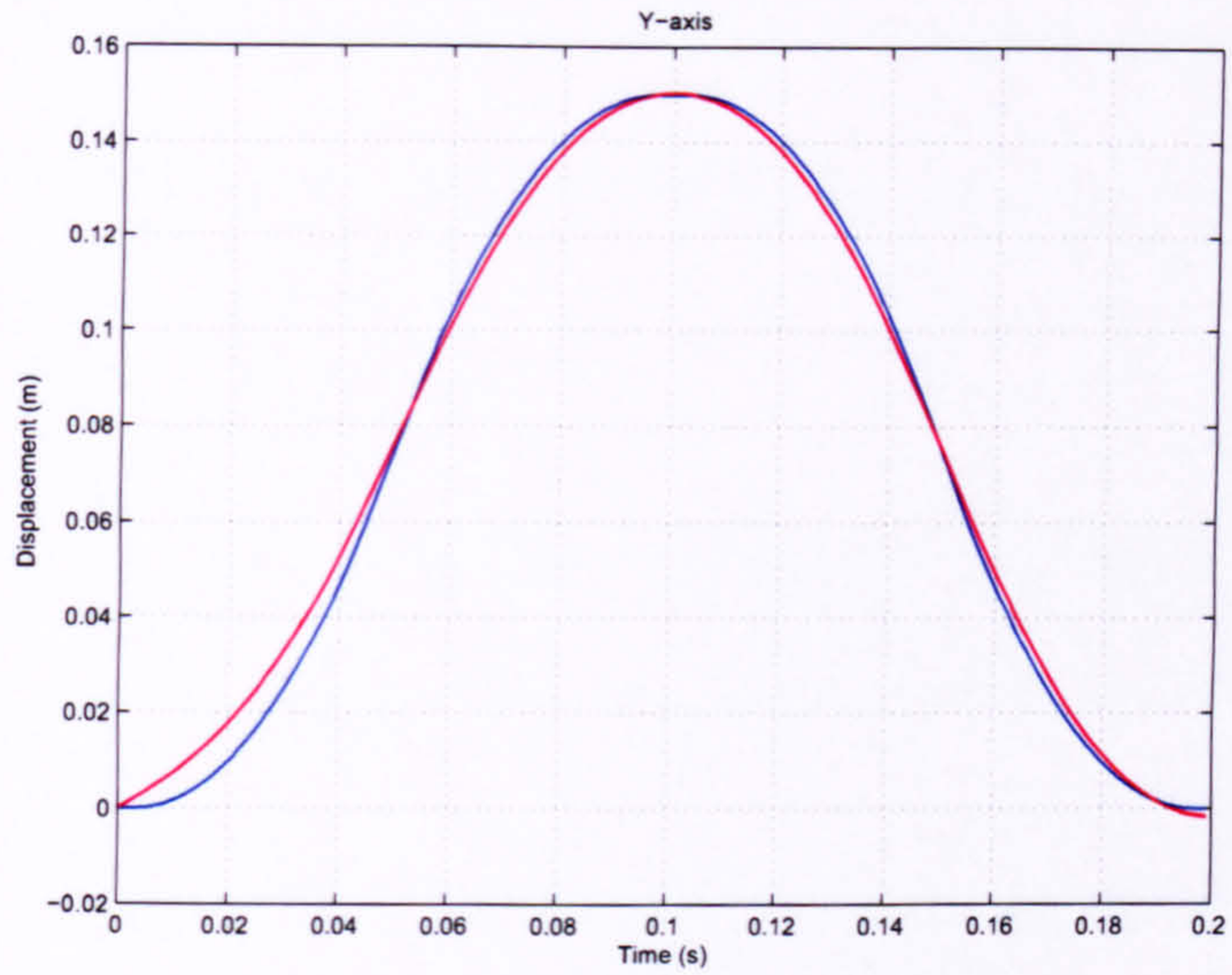


FIGURE B.14: Y-axis tracking performance iterations 490-500 (adjoint ILC, $\omega = 5 \times 10^{-8}$, high order models, gain = 0.5,)

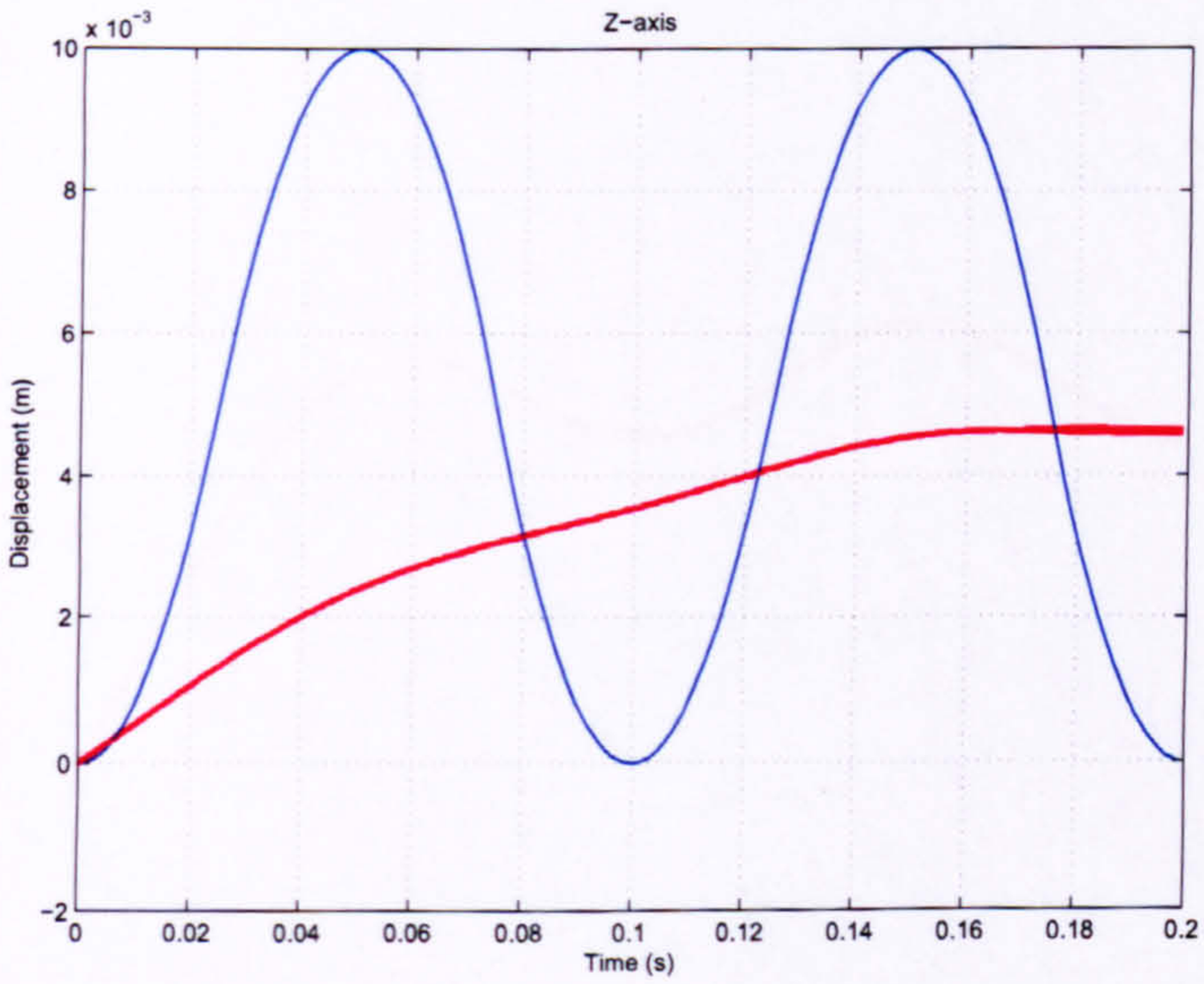


FIGURE B.15: Z-axis tracking performance iterations 490-500 (adjoint ILC, $\omega = 5 \times 10^{-8}$, high order models, gain = 0.5,)

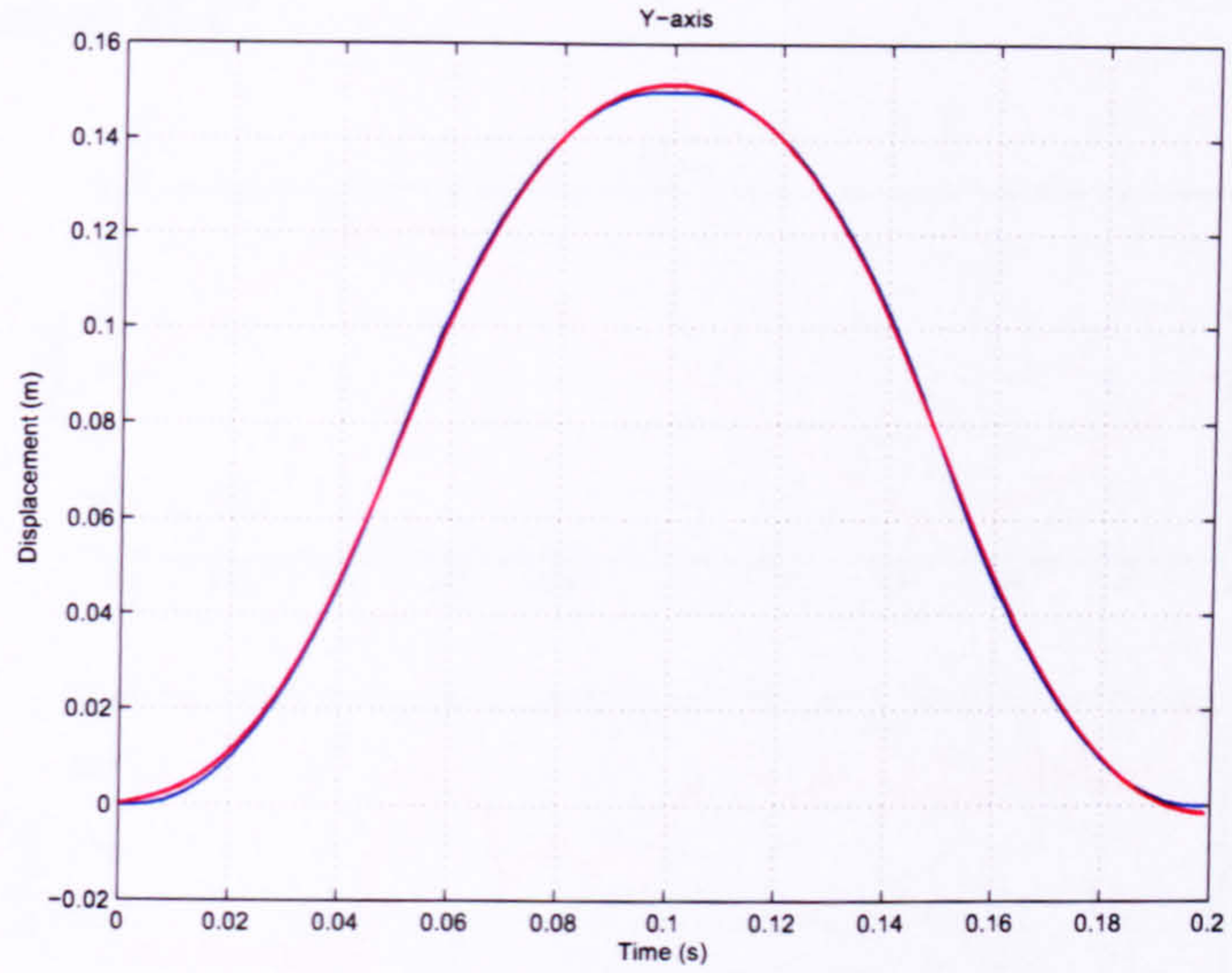


FIGURE B.16: Y-axis tracking performance iterations 490-500 (adjoint ILC, $\omega = 2 \times 10^{-7}||e||^2$, high order models, gain = 0.5,)

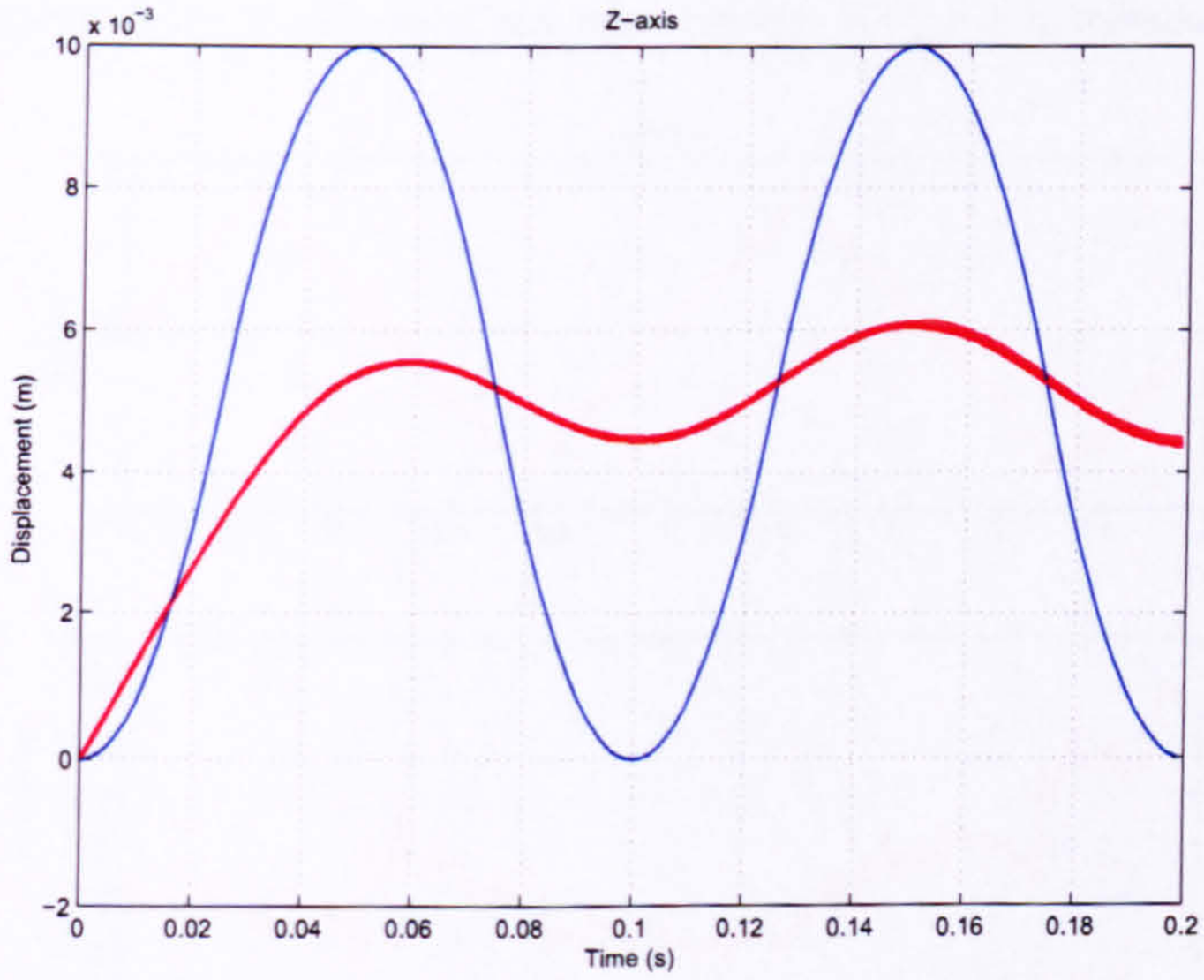


FIGURE B.17: Z-axis tracking performance iterations 490-500 (adjoint ILC, $\omega = 2 \times 10^{-7}||e||^2$, high order models, gain = 0.5,)

B.4 Inverse ILC

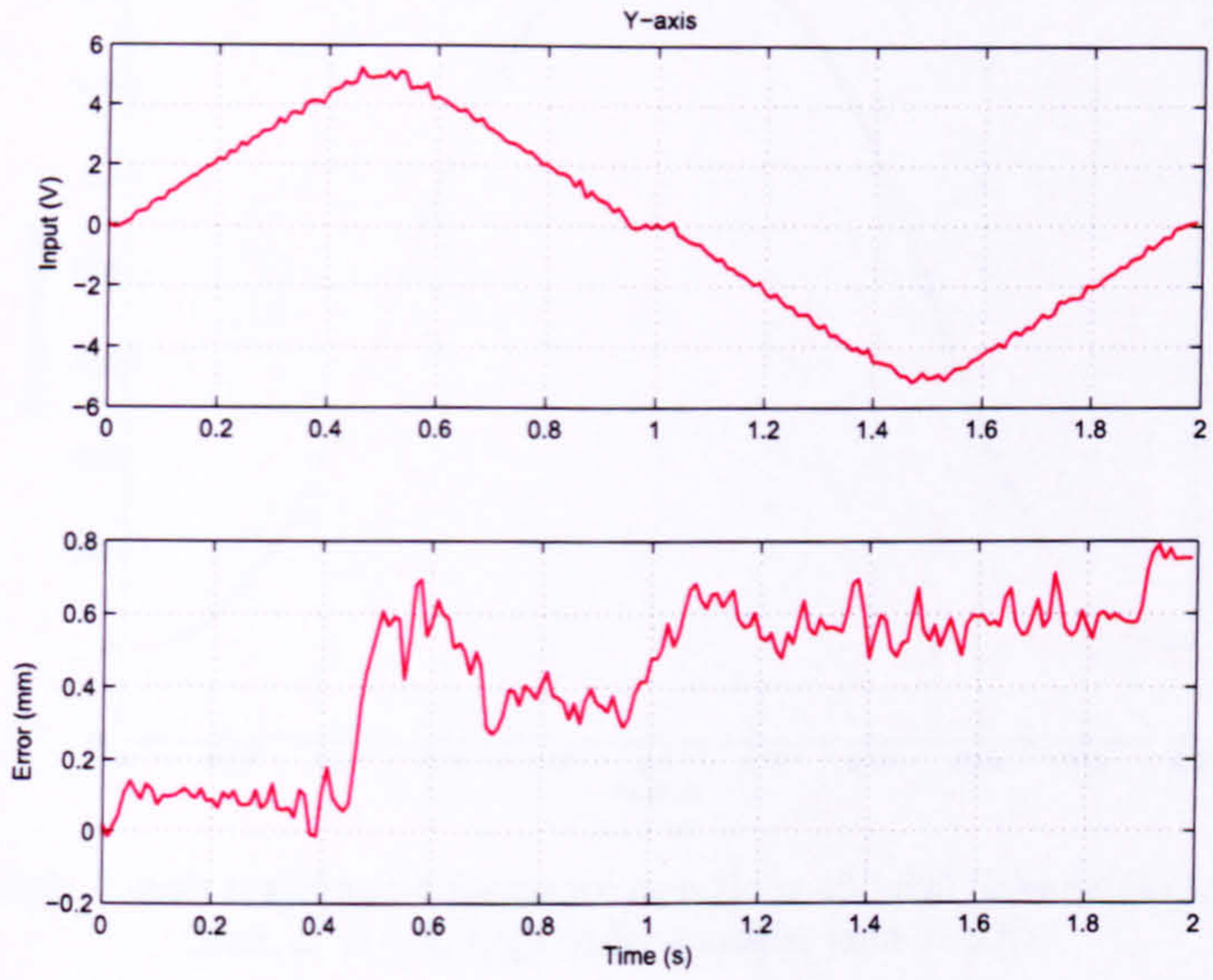


FIGURE B.18: Y-axis input and error (inverse ILC, $\omega = 0$, iteration 4)

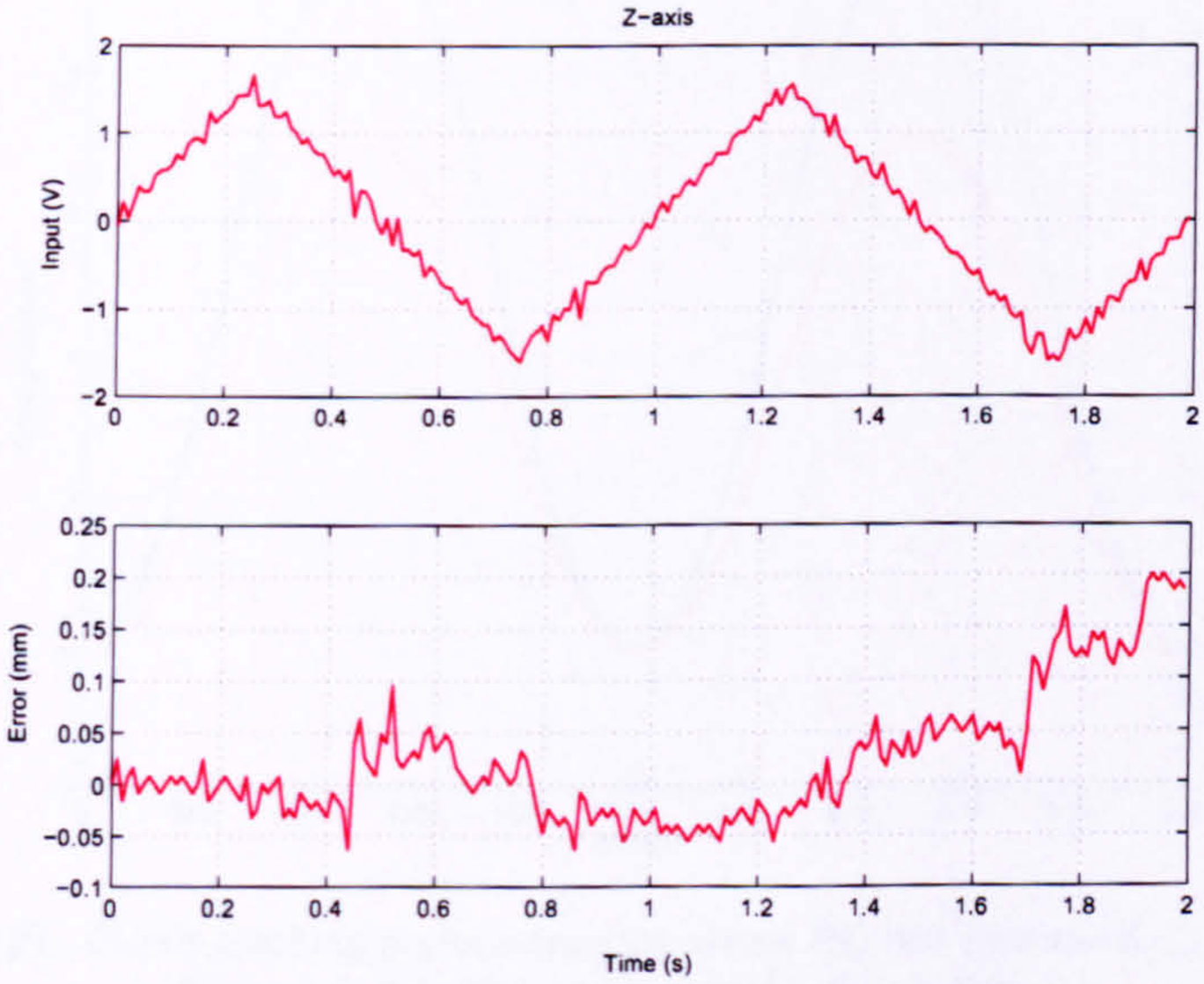


FIGURE B.19: Z-axis input and error (inverse ILC, $\omega = 0$, iteration 4)

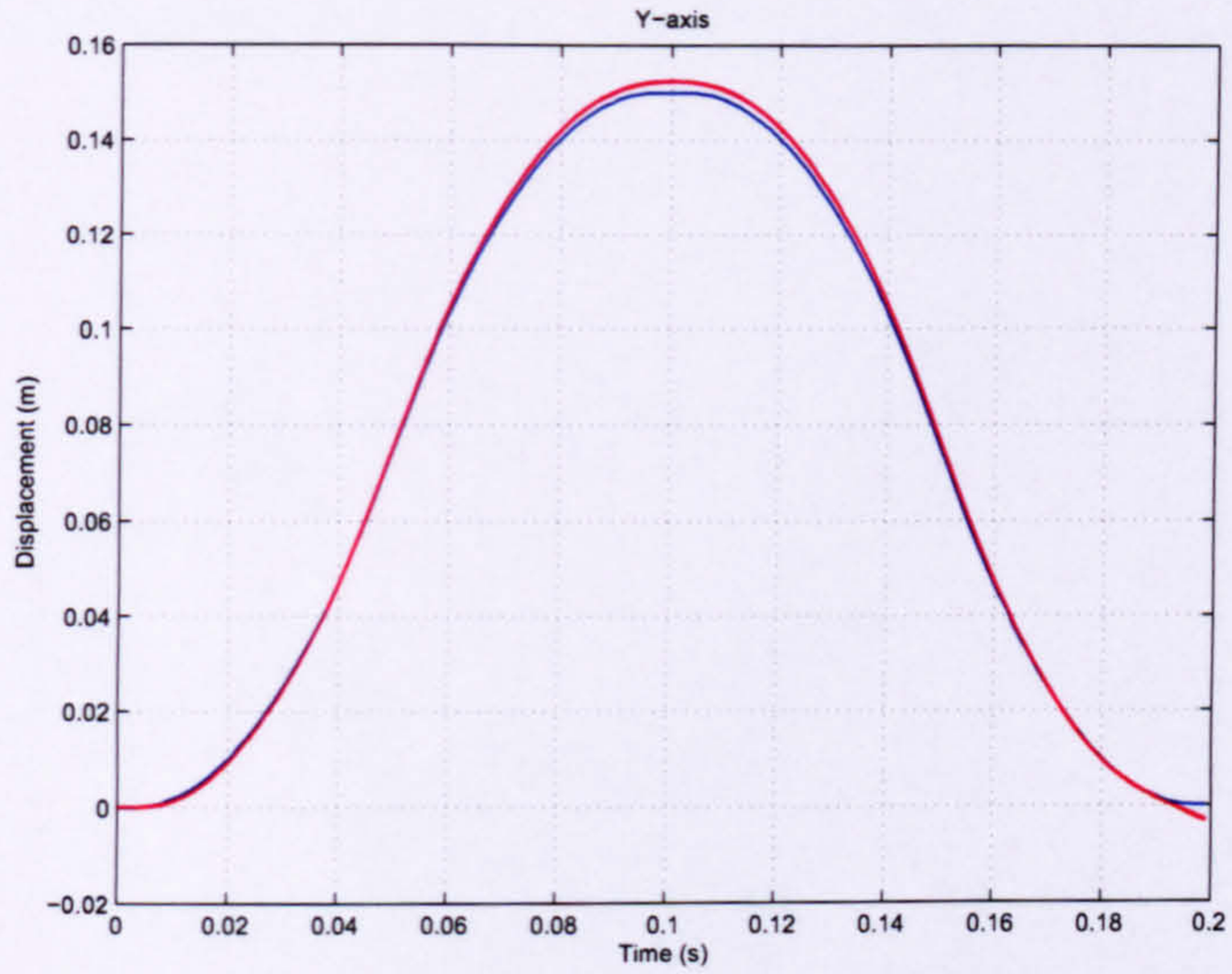


FIGURE B.20: Y-axis tracking performance iterations 490-500 (inverse ILC, zero-phase filter, $\omega = 1.4$, high order models, gain $= 0.5$)

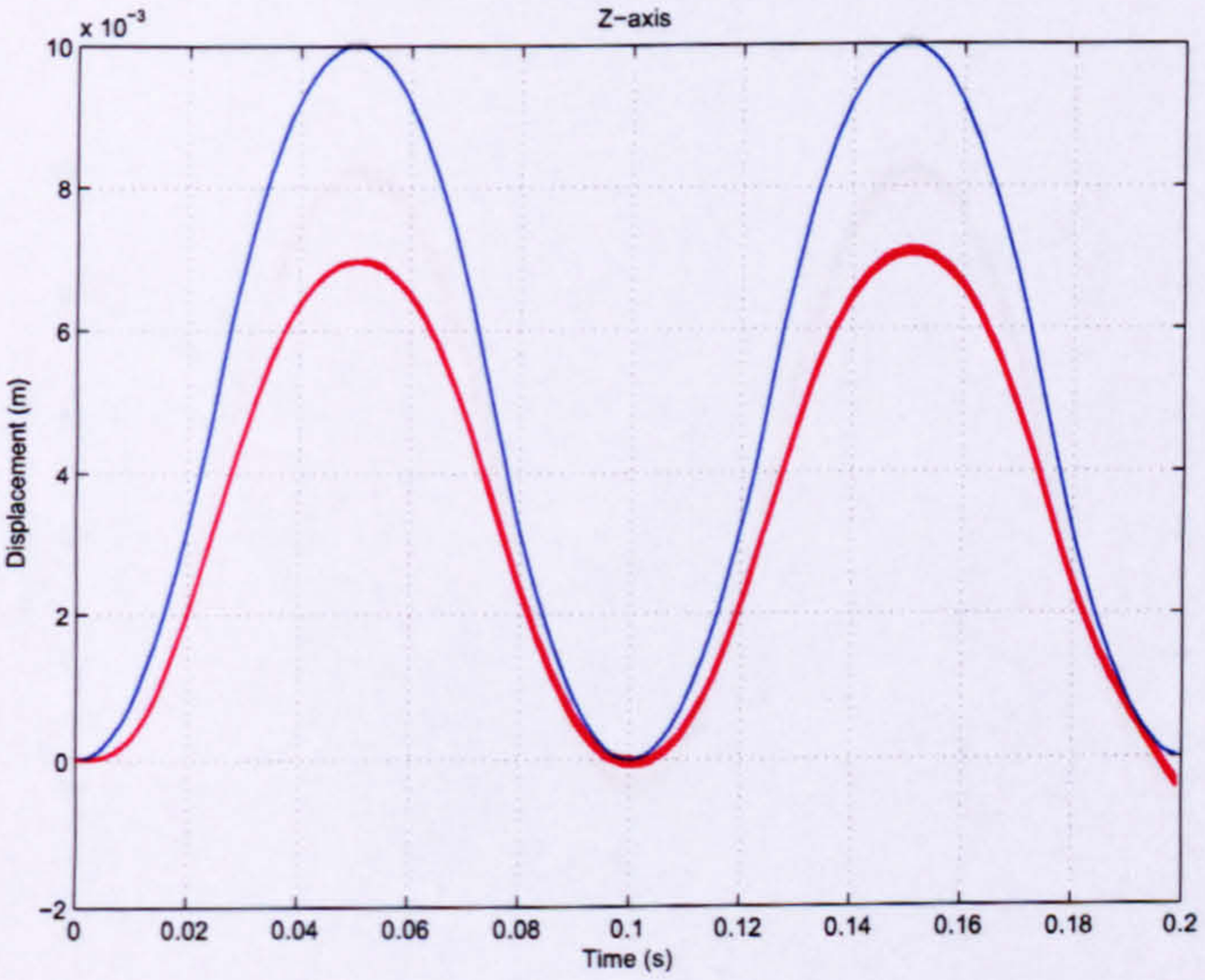


FIGURE B.21: Z-axis tracking performance iterations 490-500 (inverse ILC, zero-phase filter, $\omega = 1.4$, high order models, gain $= 0.5$)

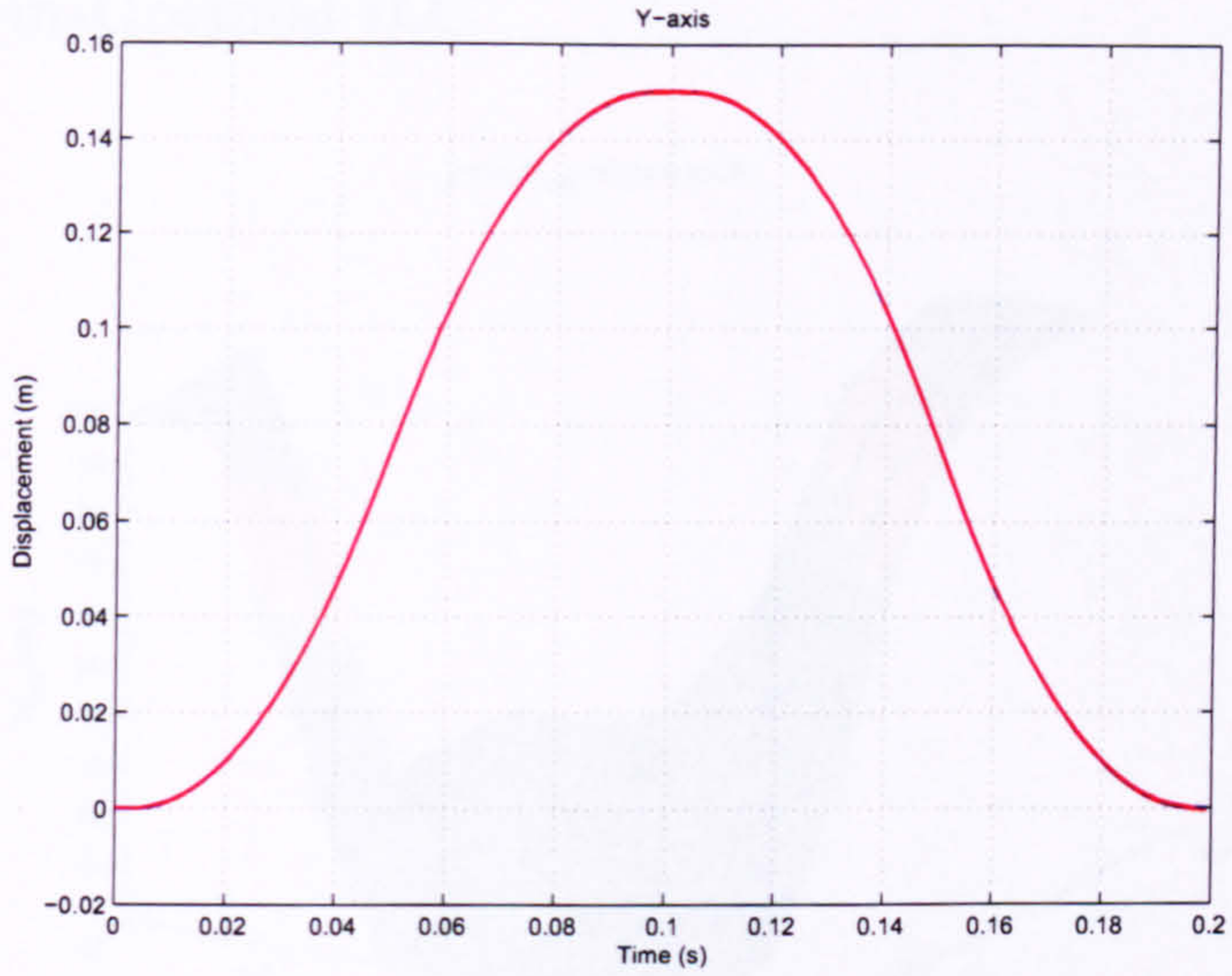


FIGURE B.22: Y-axis tracking performance iterations 490-500 (inverse ILC, zero-phase filter, $\omega = ||e||^2$, high order models, gain = 0.5)

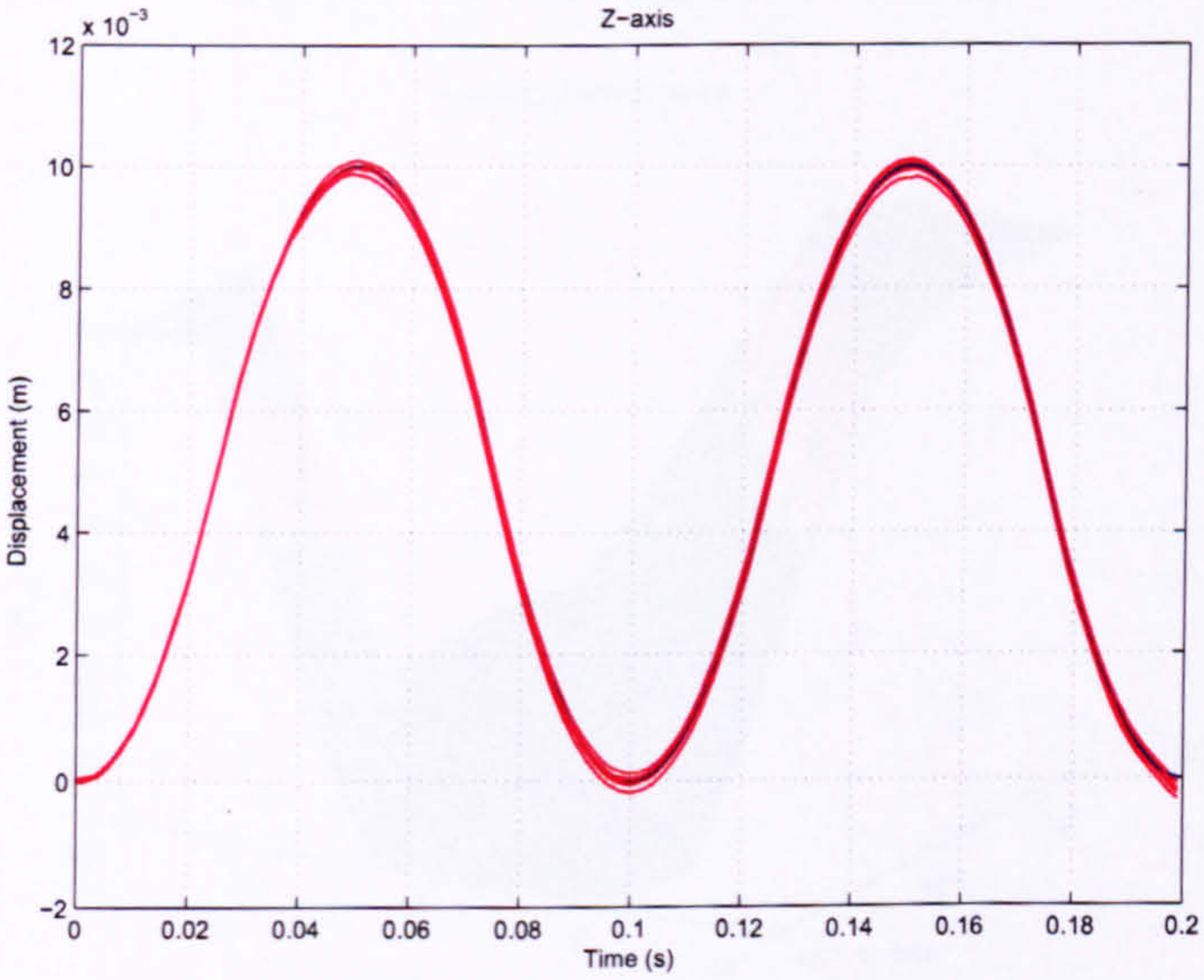


FIGURE B.23: Z-axis tracking performance iterations 490-500 (inverse ILC, zero-phase filter, $\omega = ||e||^2$, high order models, gain = 0.5)

B.5 Norm-Optimal ILC

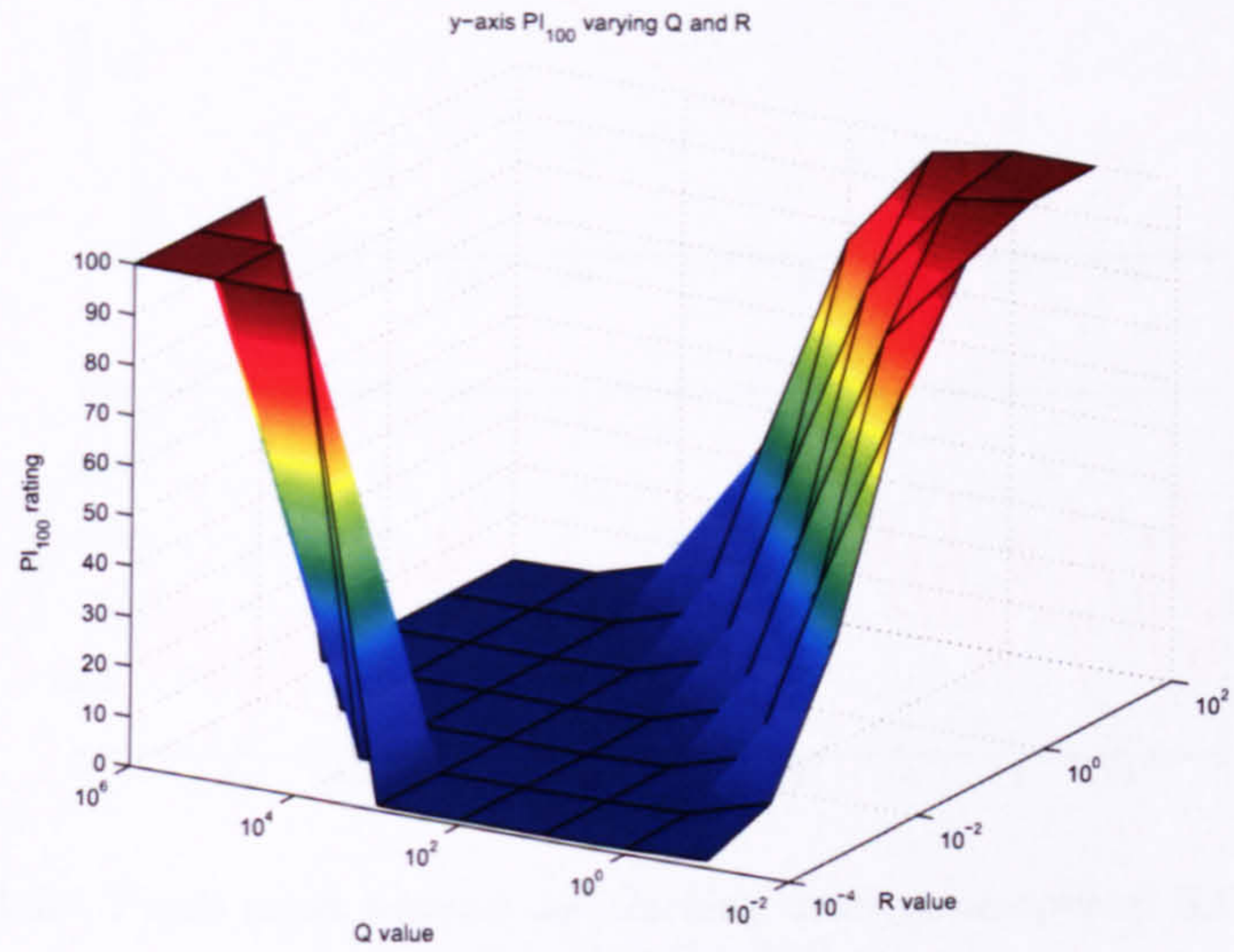


FIGURE B.24: Y-axis PI_{100} for various q and r

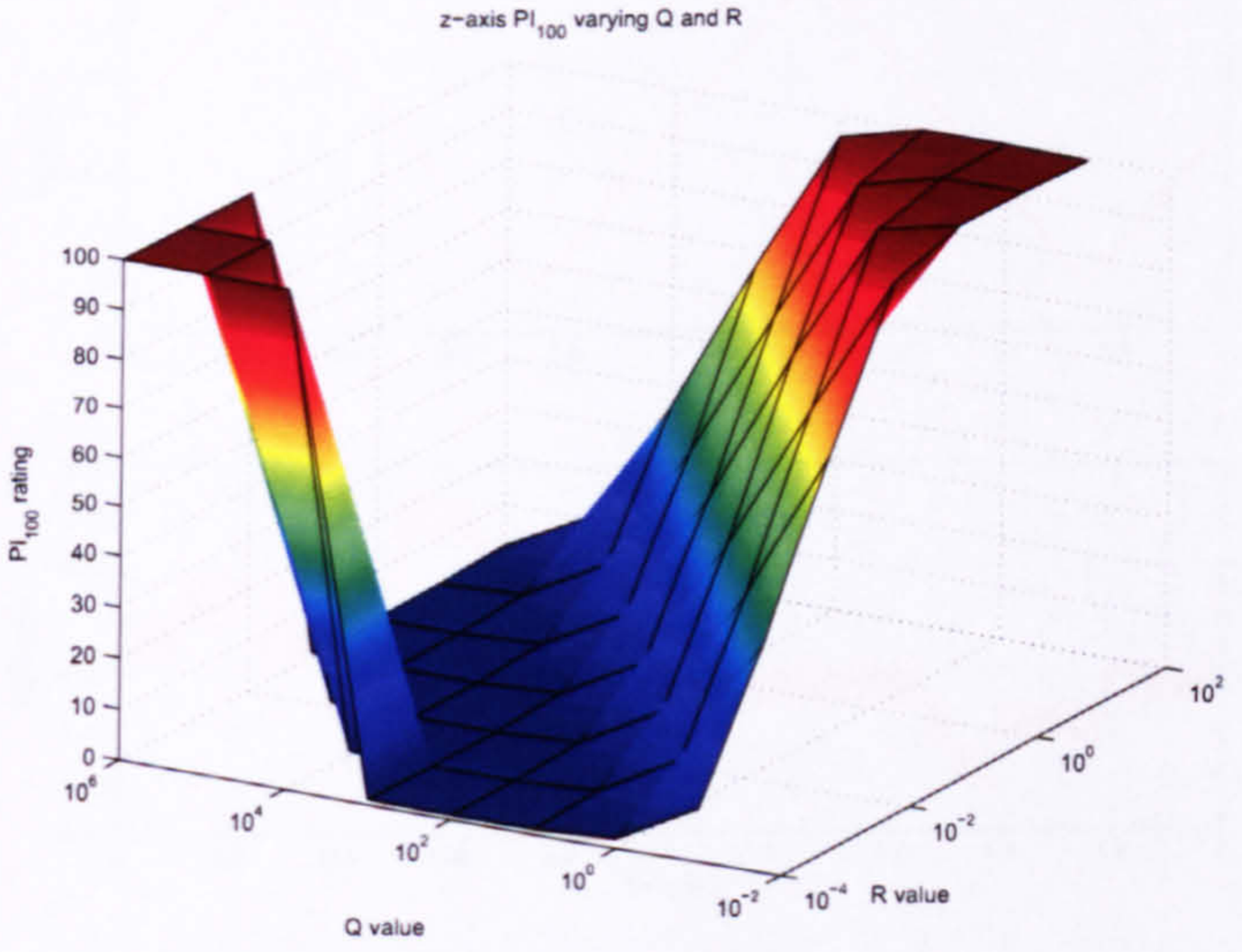


FIGURE B.25: Z-axis PI_{100} for various q and r

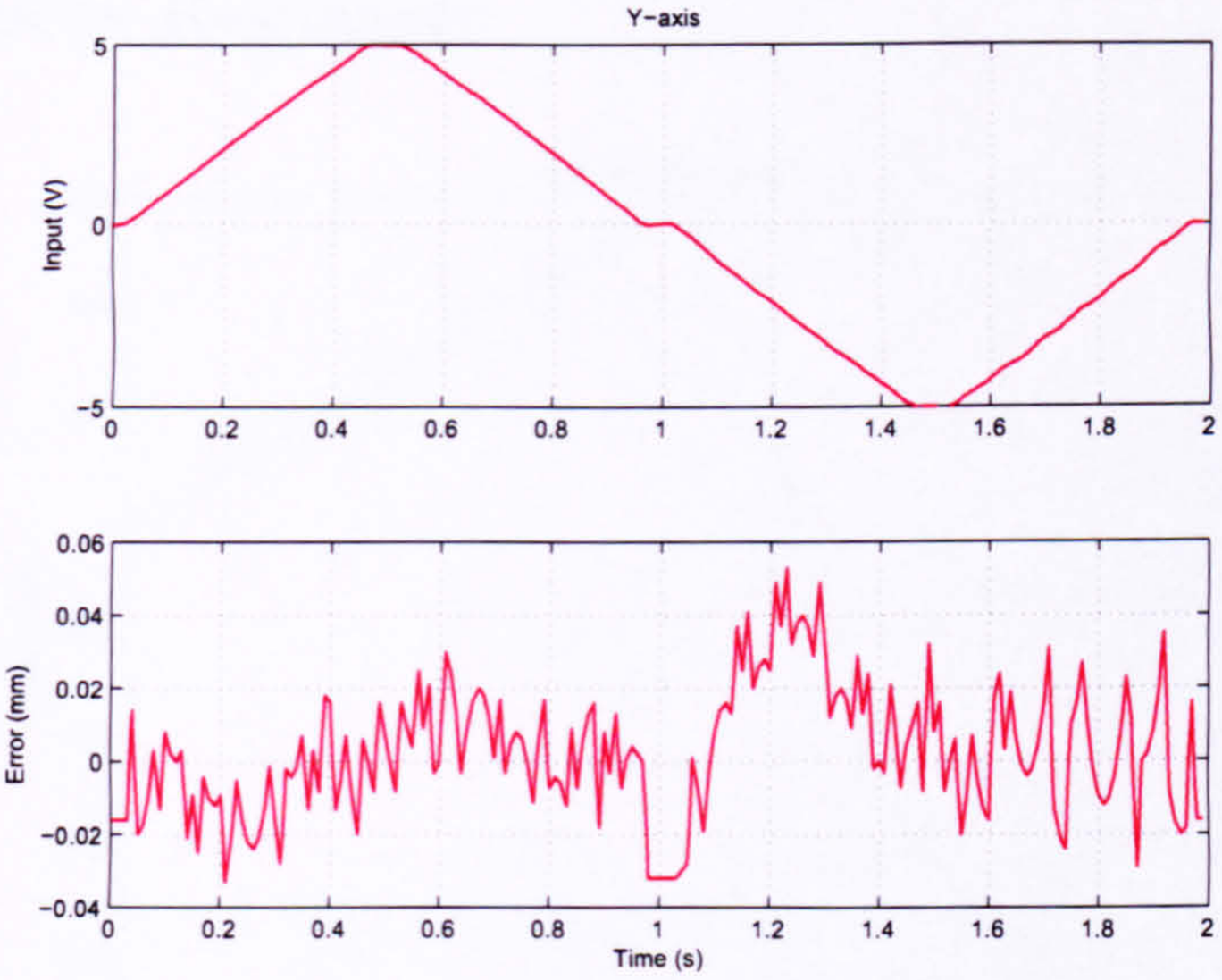


FIGURE B.26: Y-axis input demand and tracking error (norm-optimal ILC, 1st order models, iteration 500)

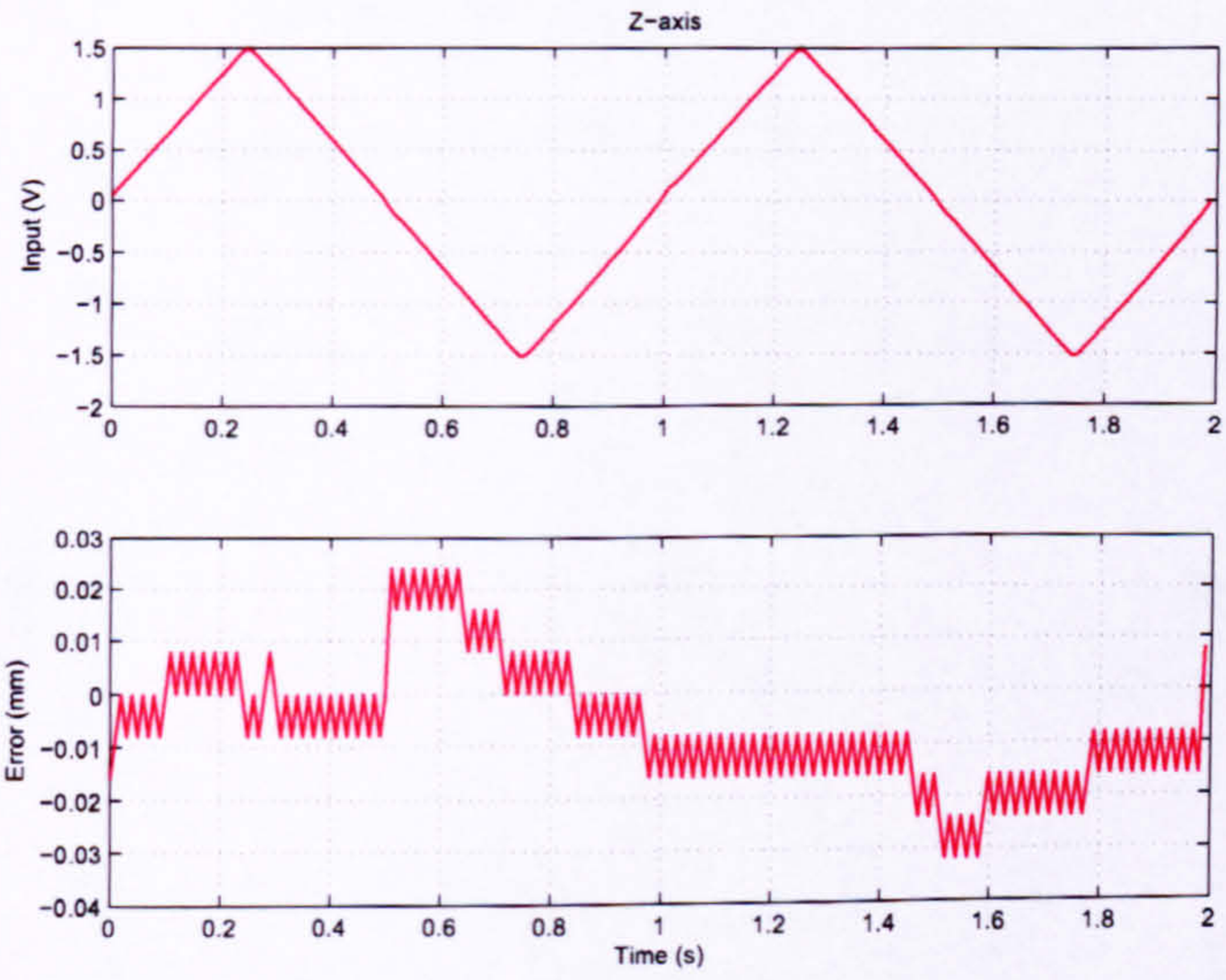


FIGURE B.27: Z-axis input demand and tracking error (norm-optimal ILC, 1st order models, iteration 500)

B.6 Velocity References

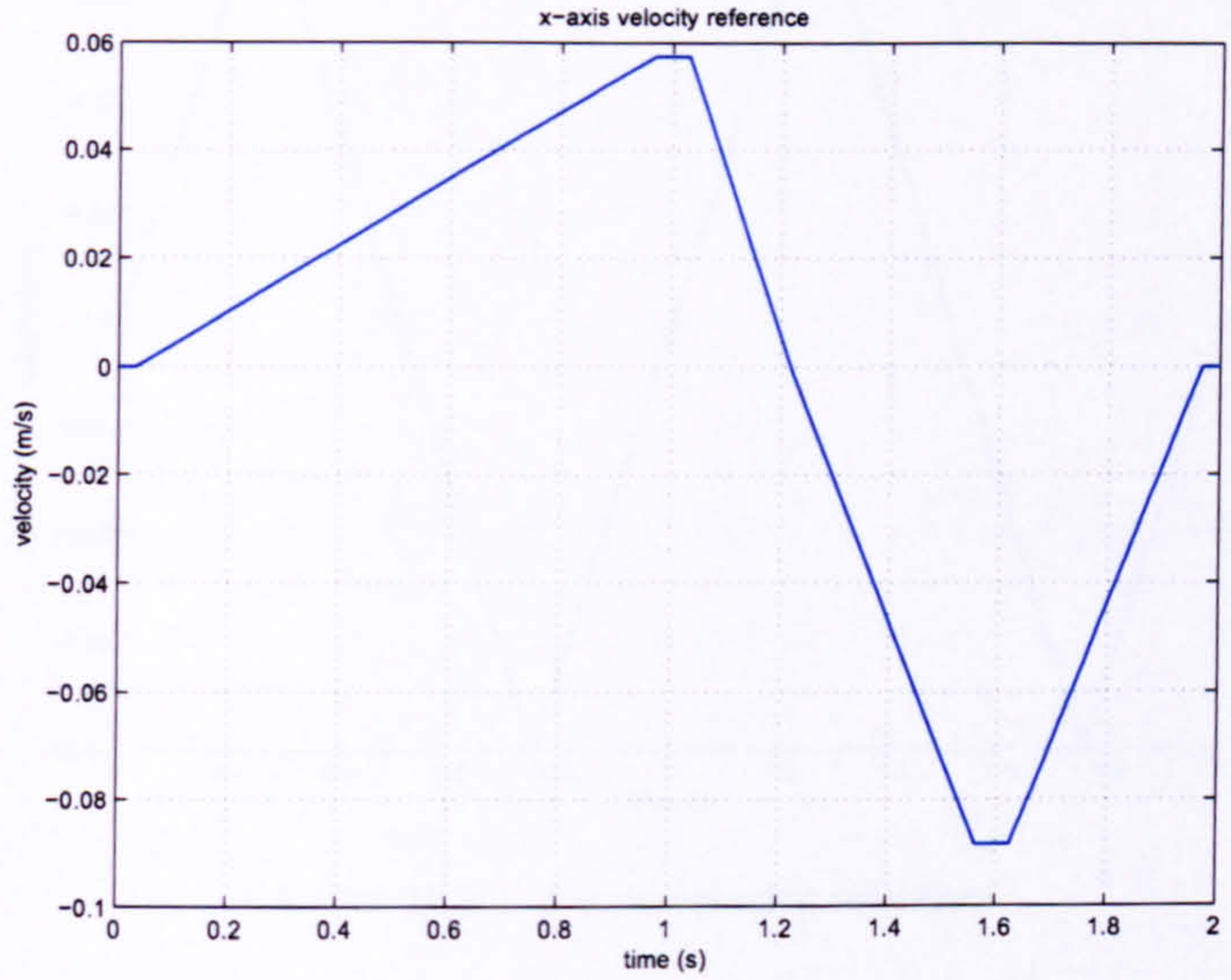


FIGURE B.28: X-axis velocity reference

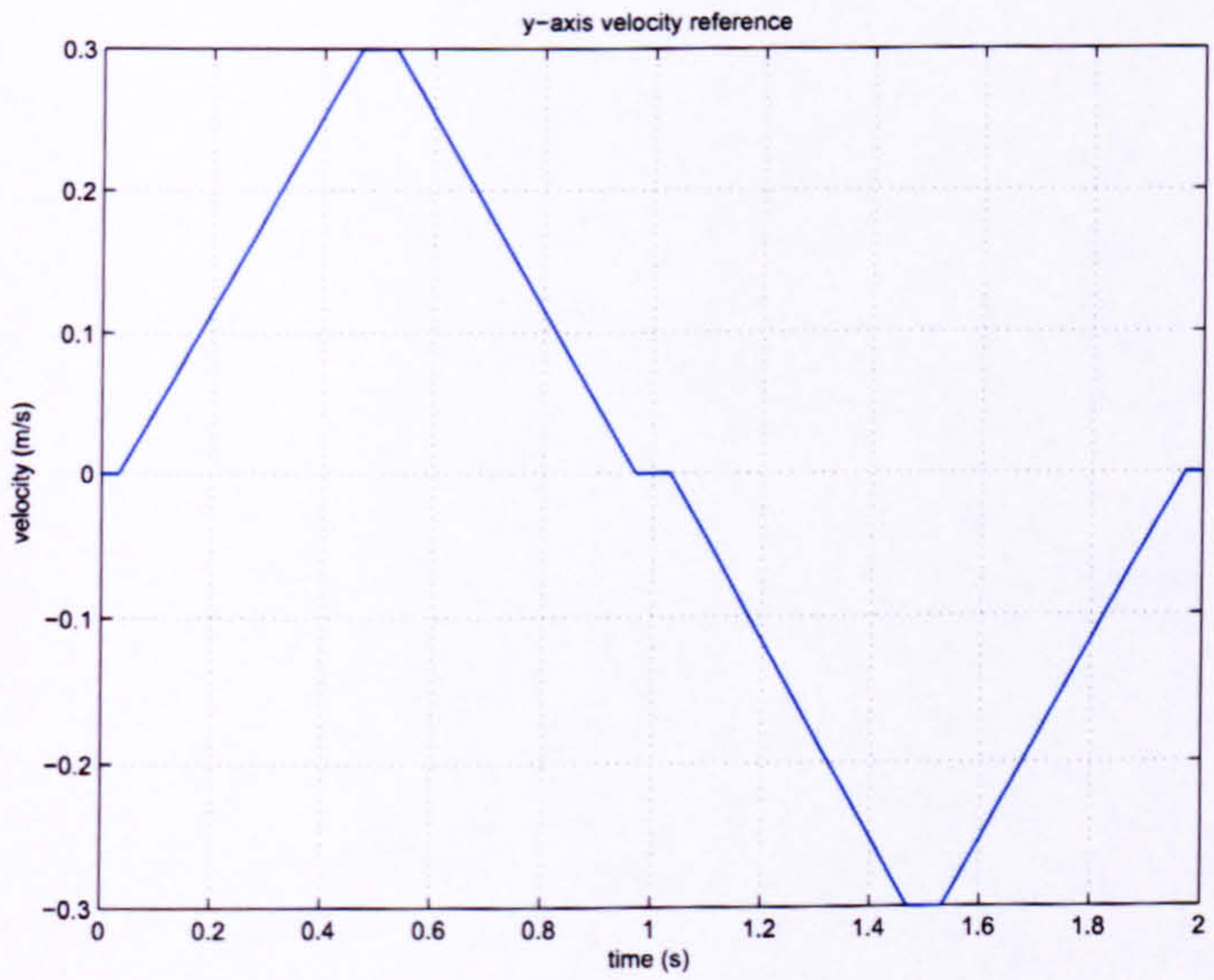


FIGURE B.29: Y-axis velocity reference

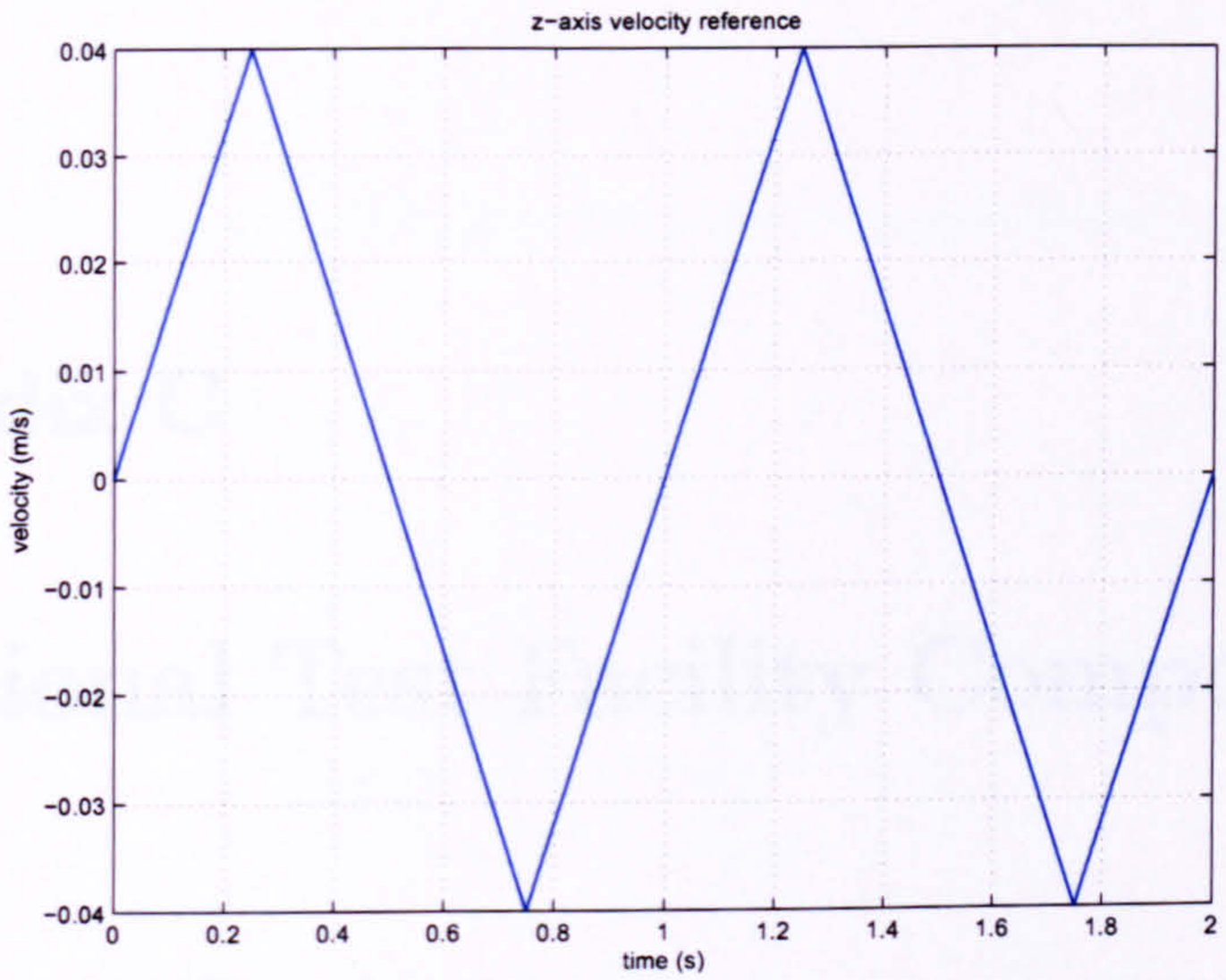


FIGURE B.30: Z-axis velocity reference

Appendix C

Additional Test Facility Components

C.1 Conveyor Encoder Adapter

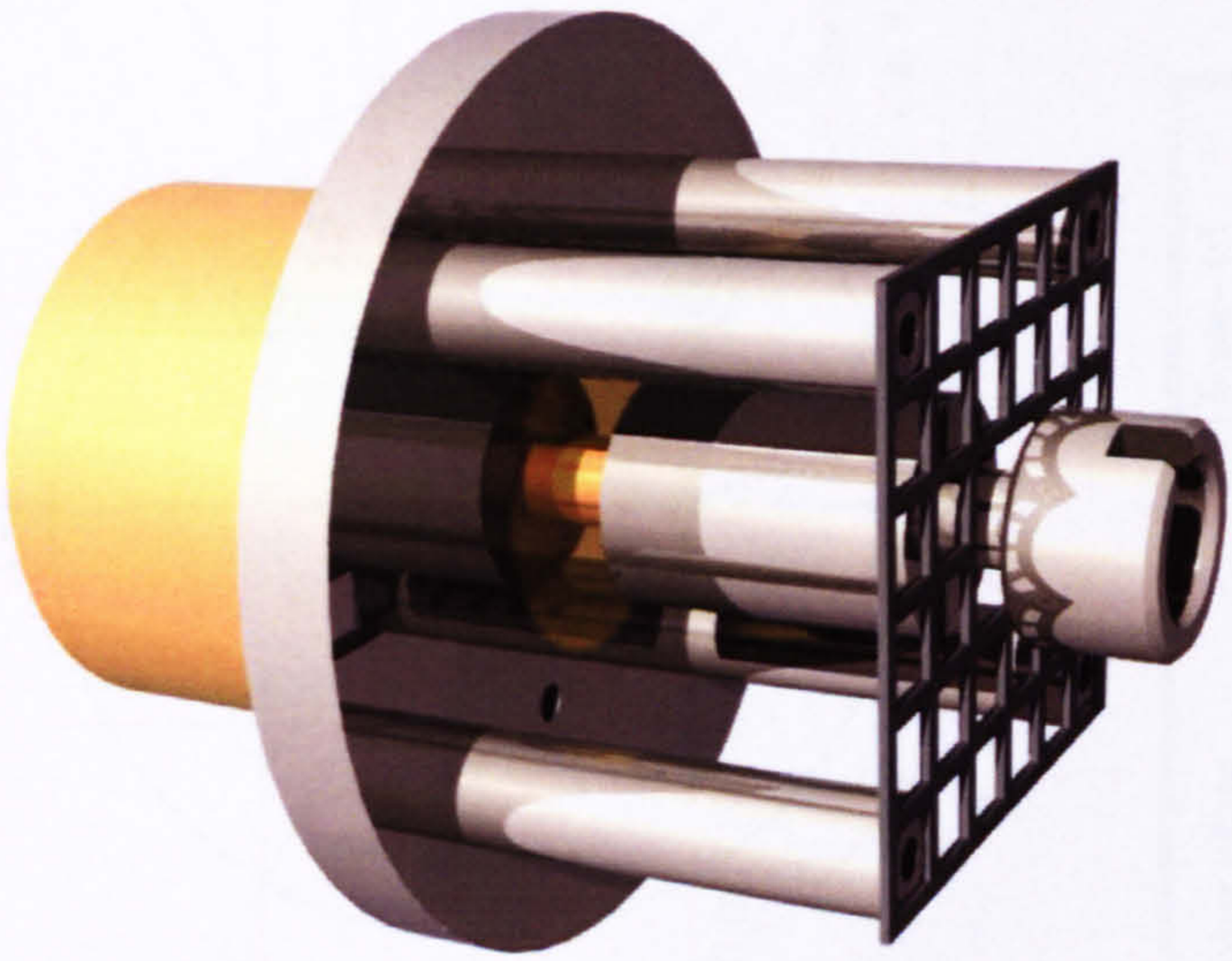


FIGURE C.1: 3D representation of the conveyor encoder bracket

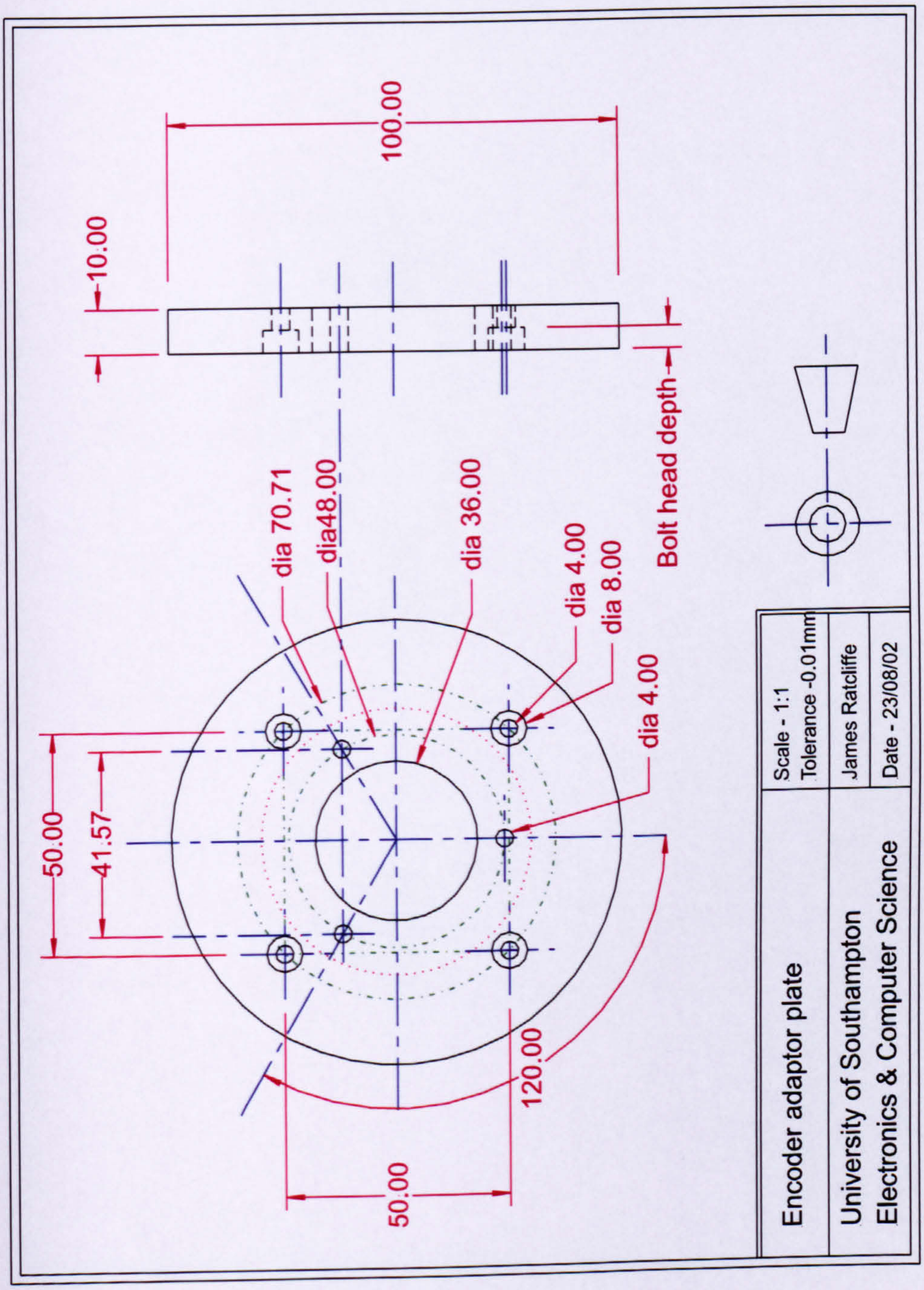


FIGURE C.2: Conveyor drive encoder adapter plate

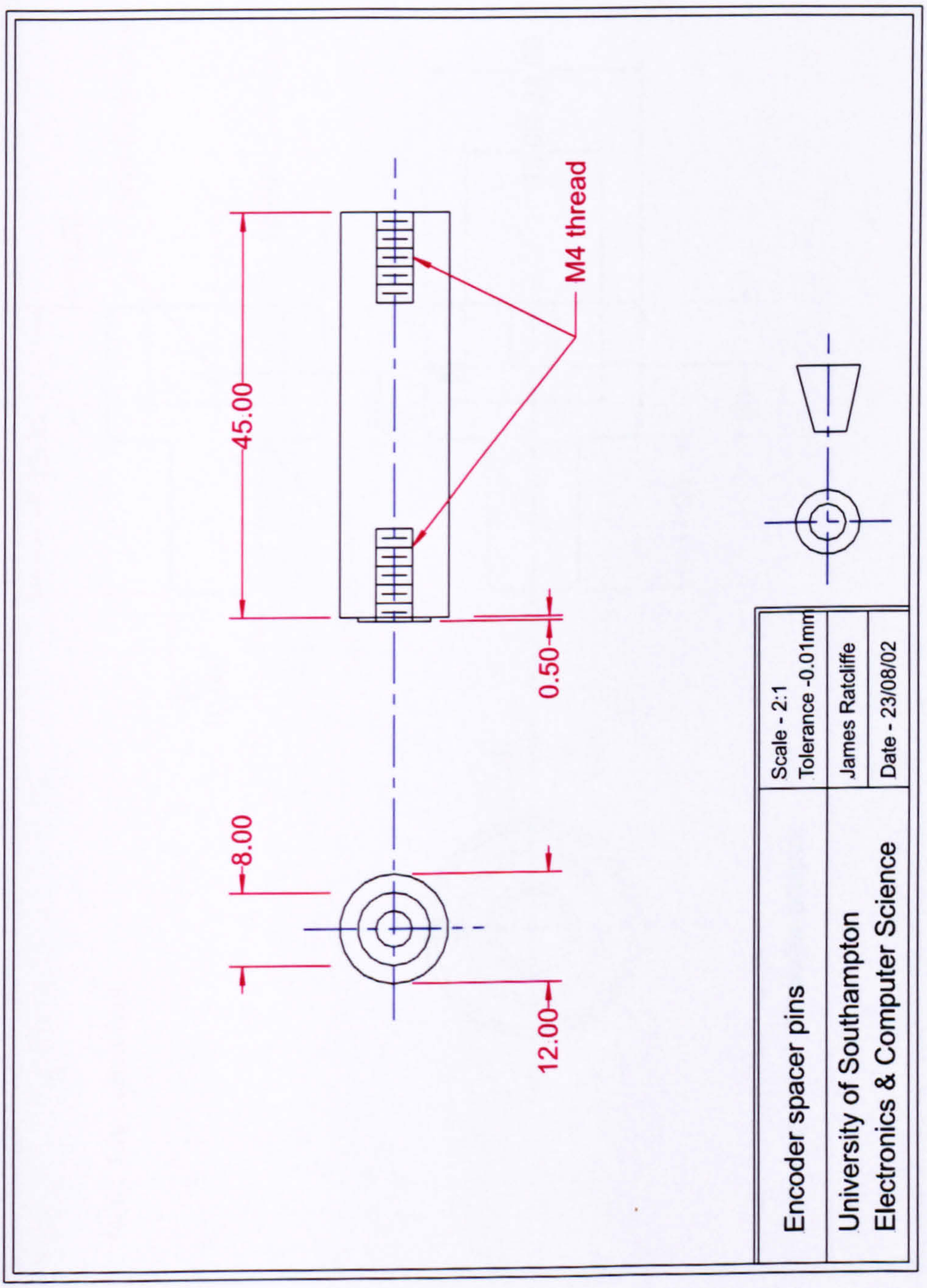


FIGURE C.3: Conveyor drive encoder adapter support struts

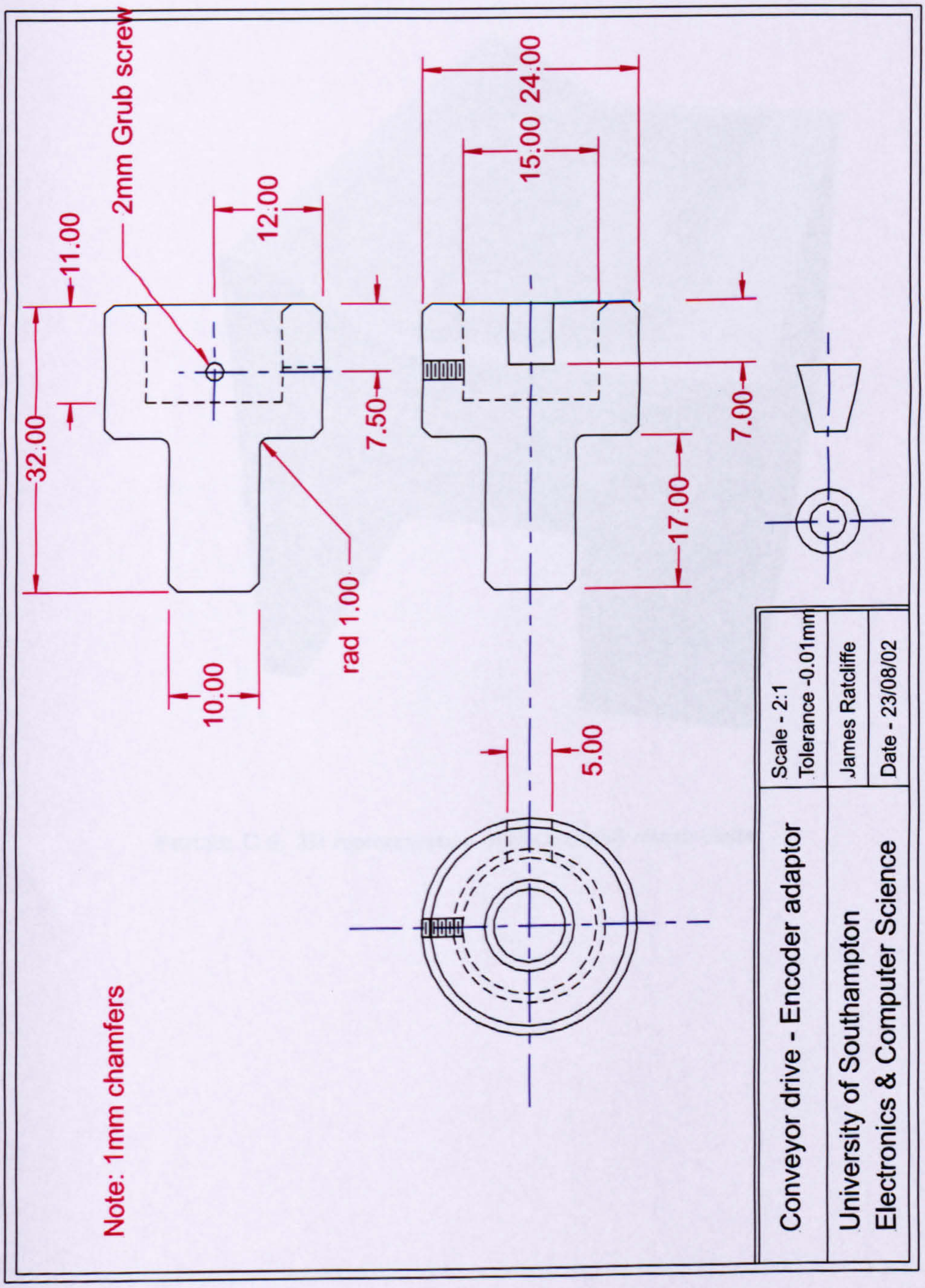


FIGURE C.4: Conveyor drive encoder shaft adapter

C.2 Payload Return Chute

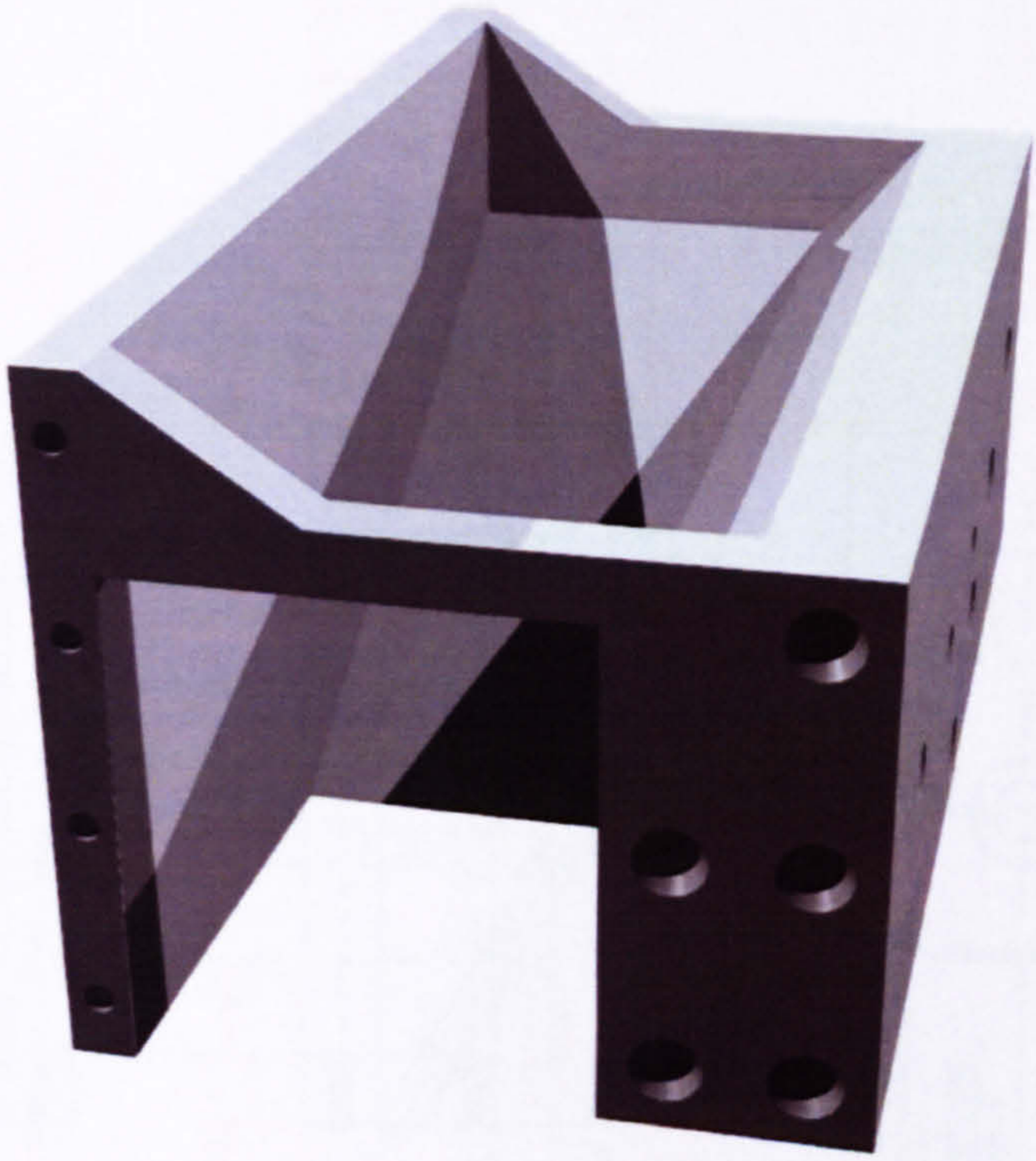


FIGURE C.5: 3D representation of the payload return chute

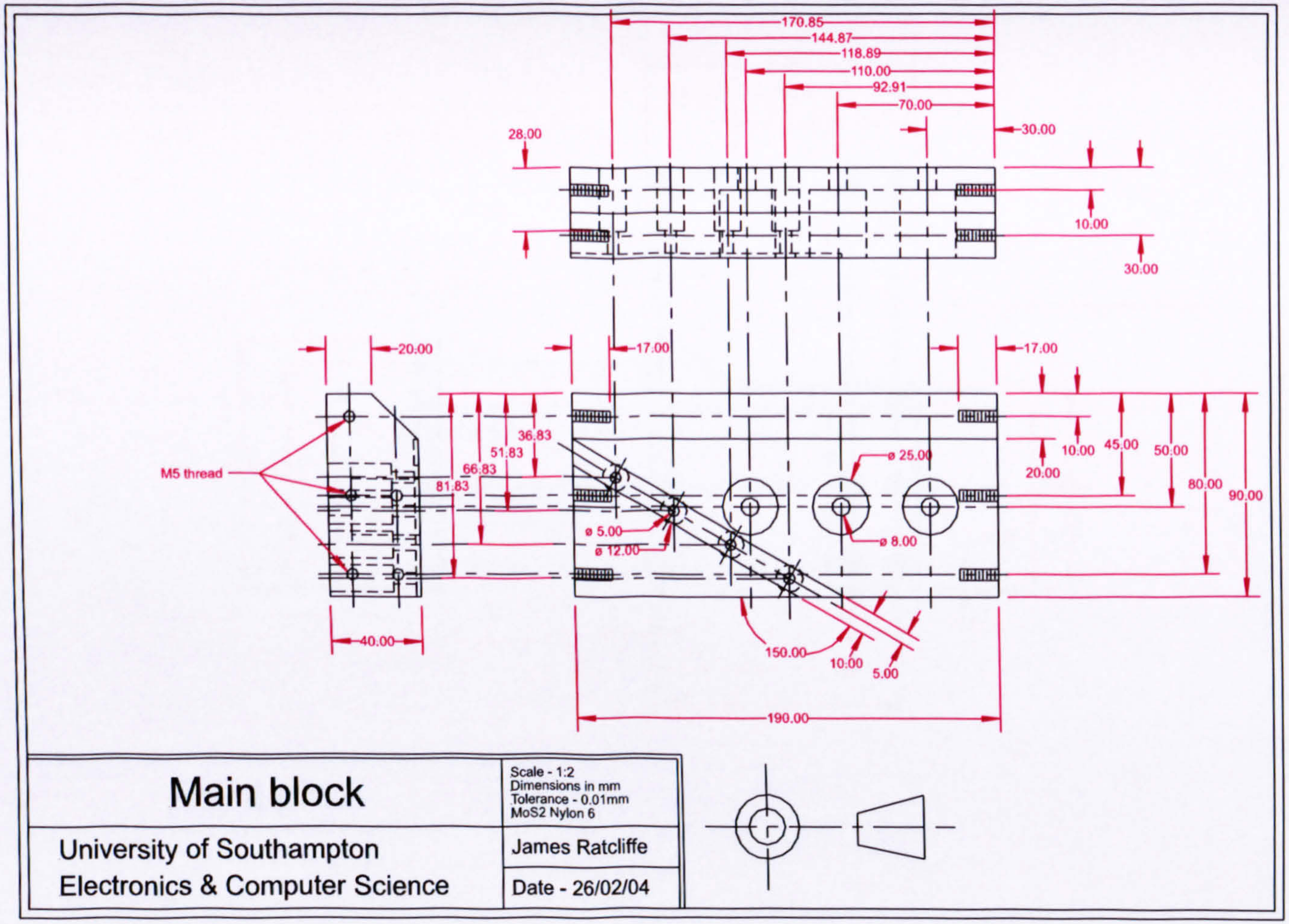


FIGURE C.6: Return chute main block

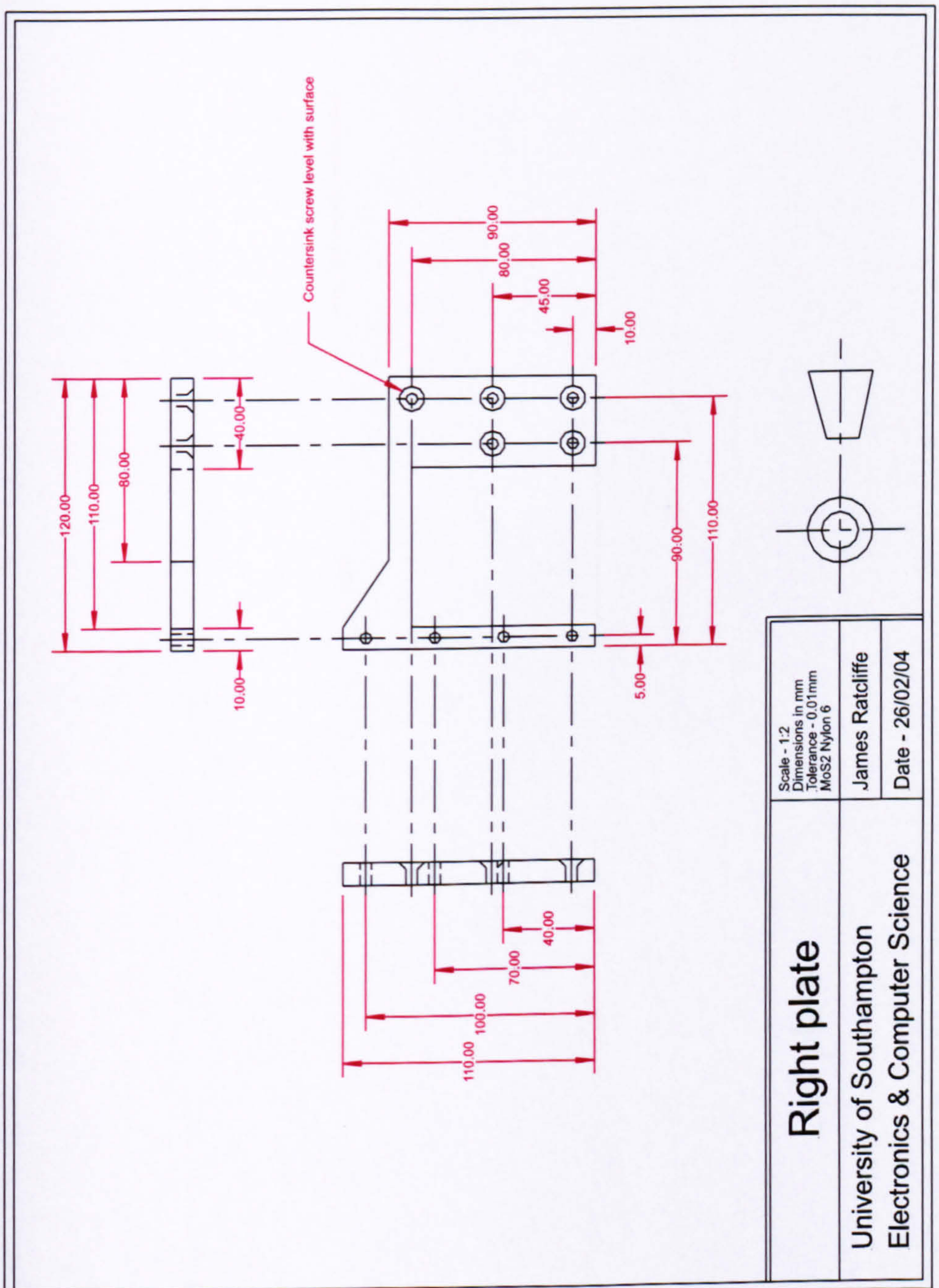


FIGURE C.7: Return chute right face

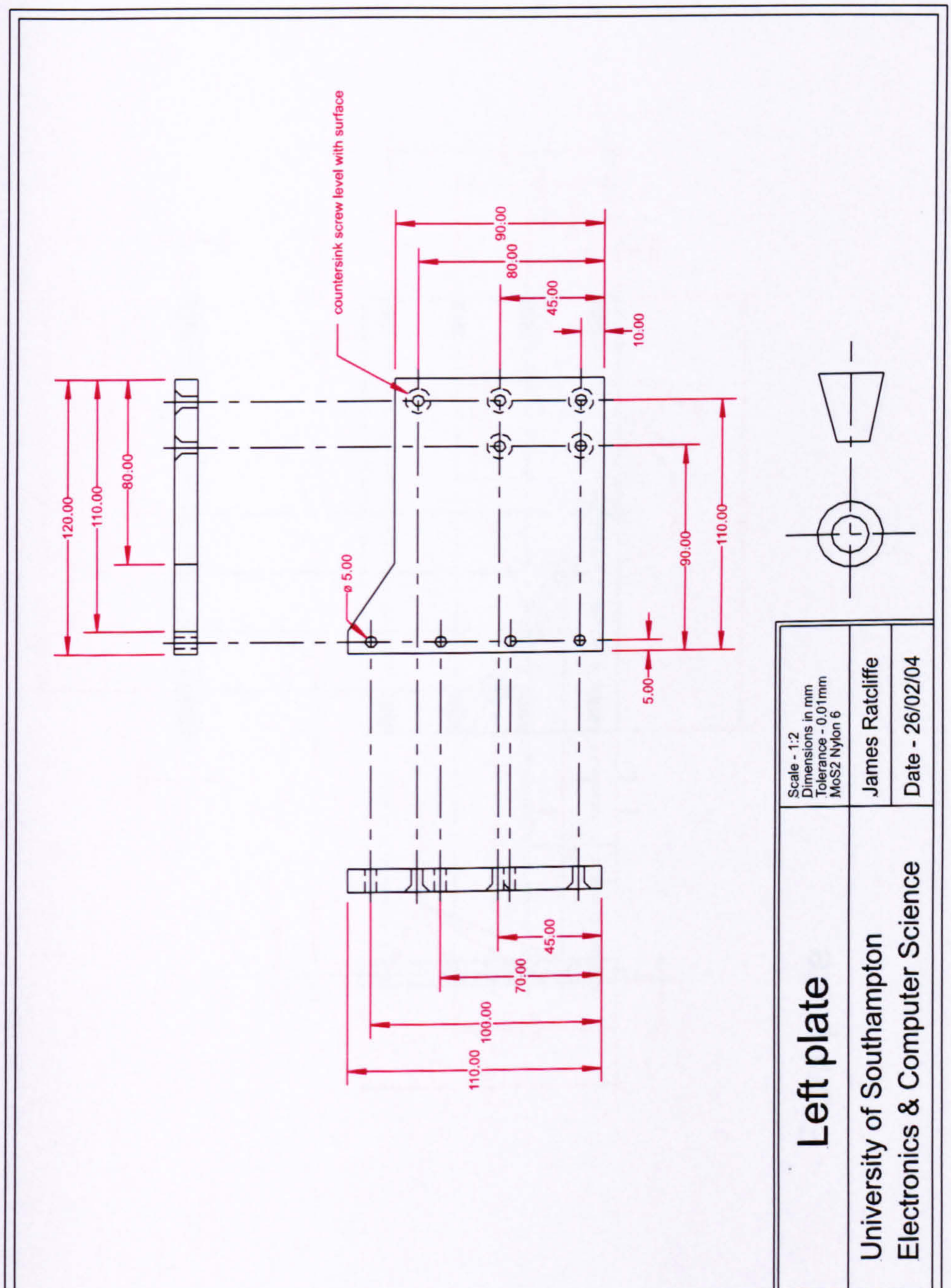


FIGURE C.8: Return chute left face

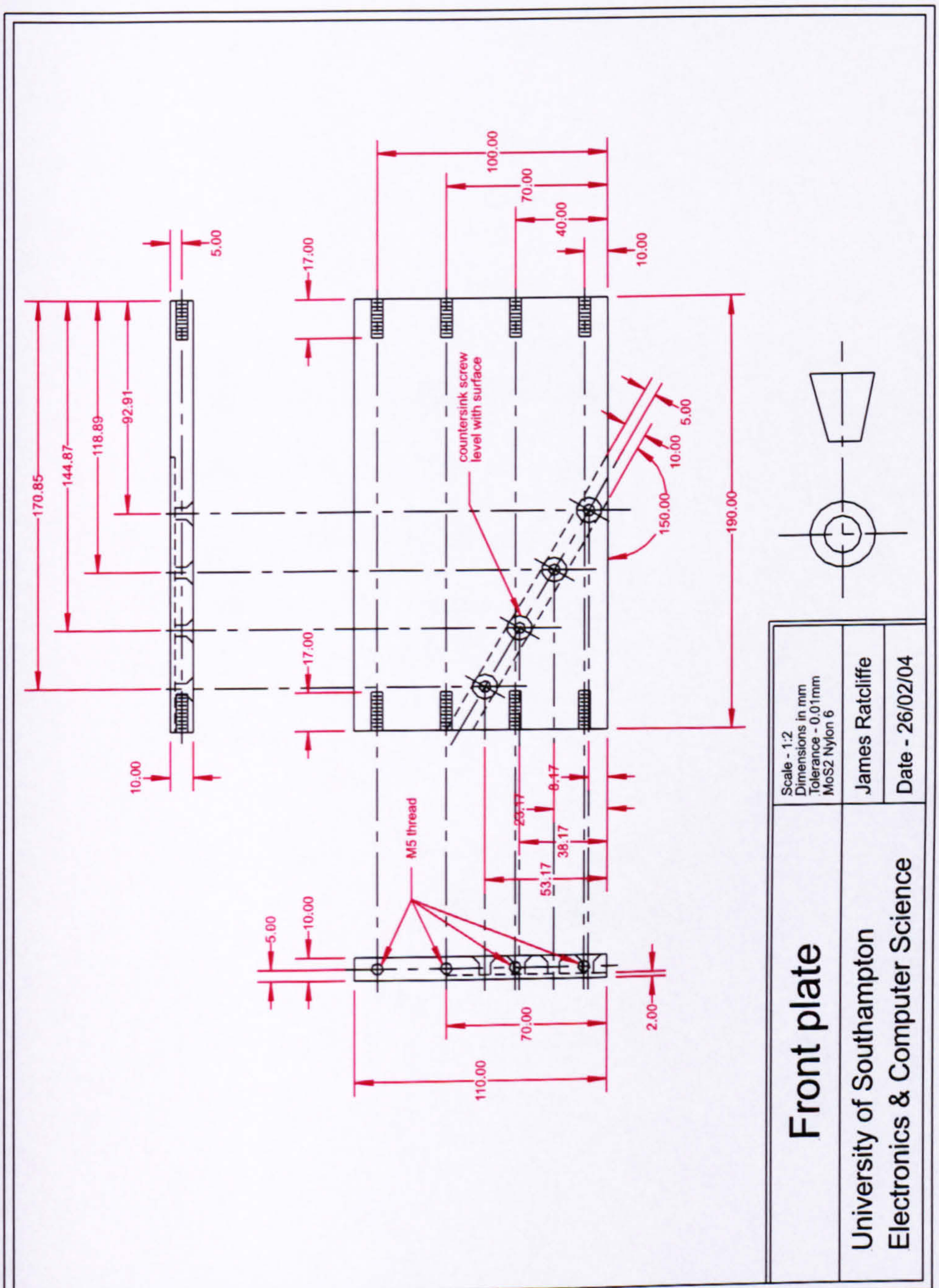


FIGURE C.9: Return chute front face

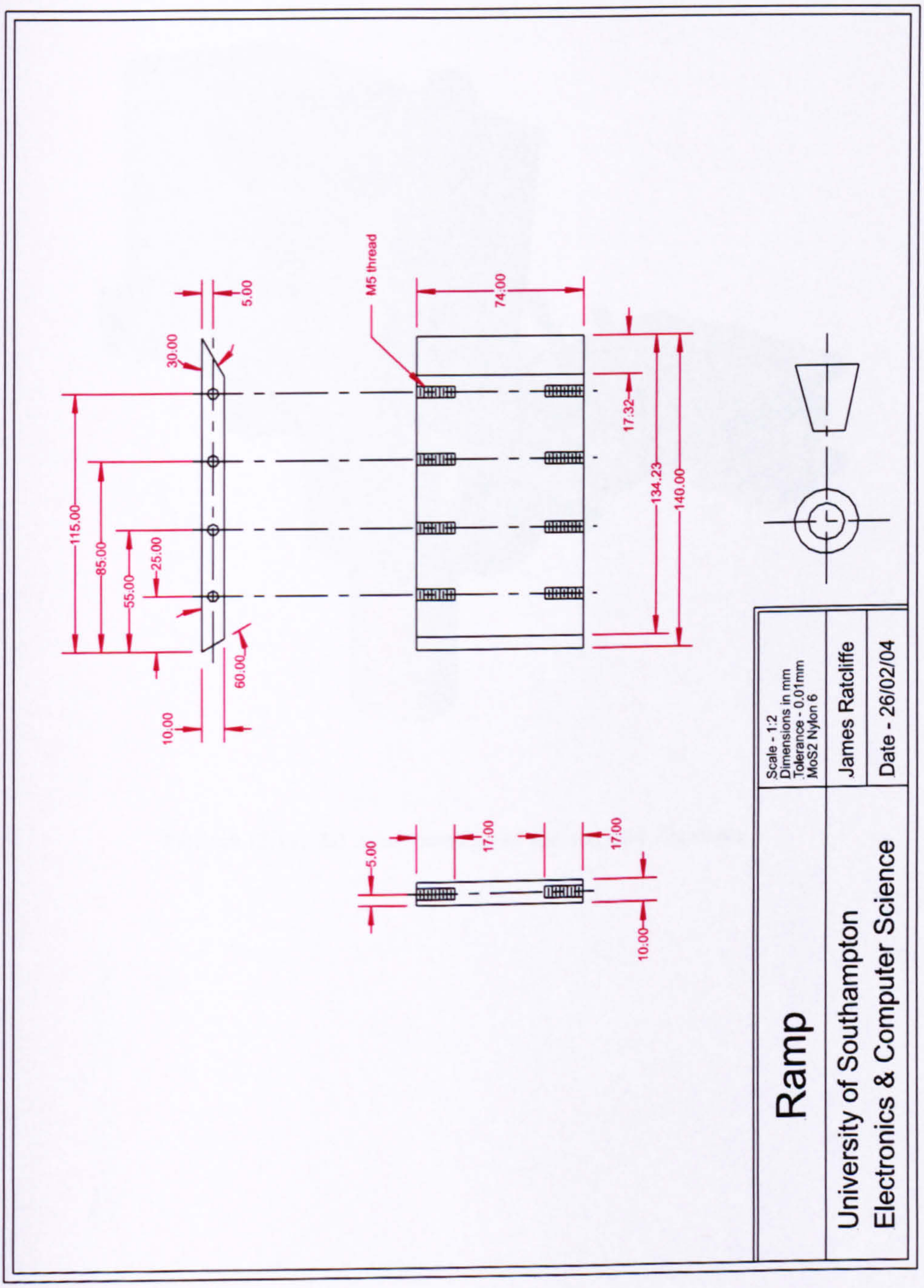


FIGURE C.10: Return chute payload ramp

C.3 Dispenser

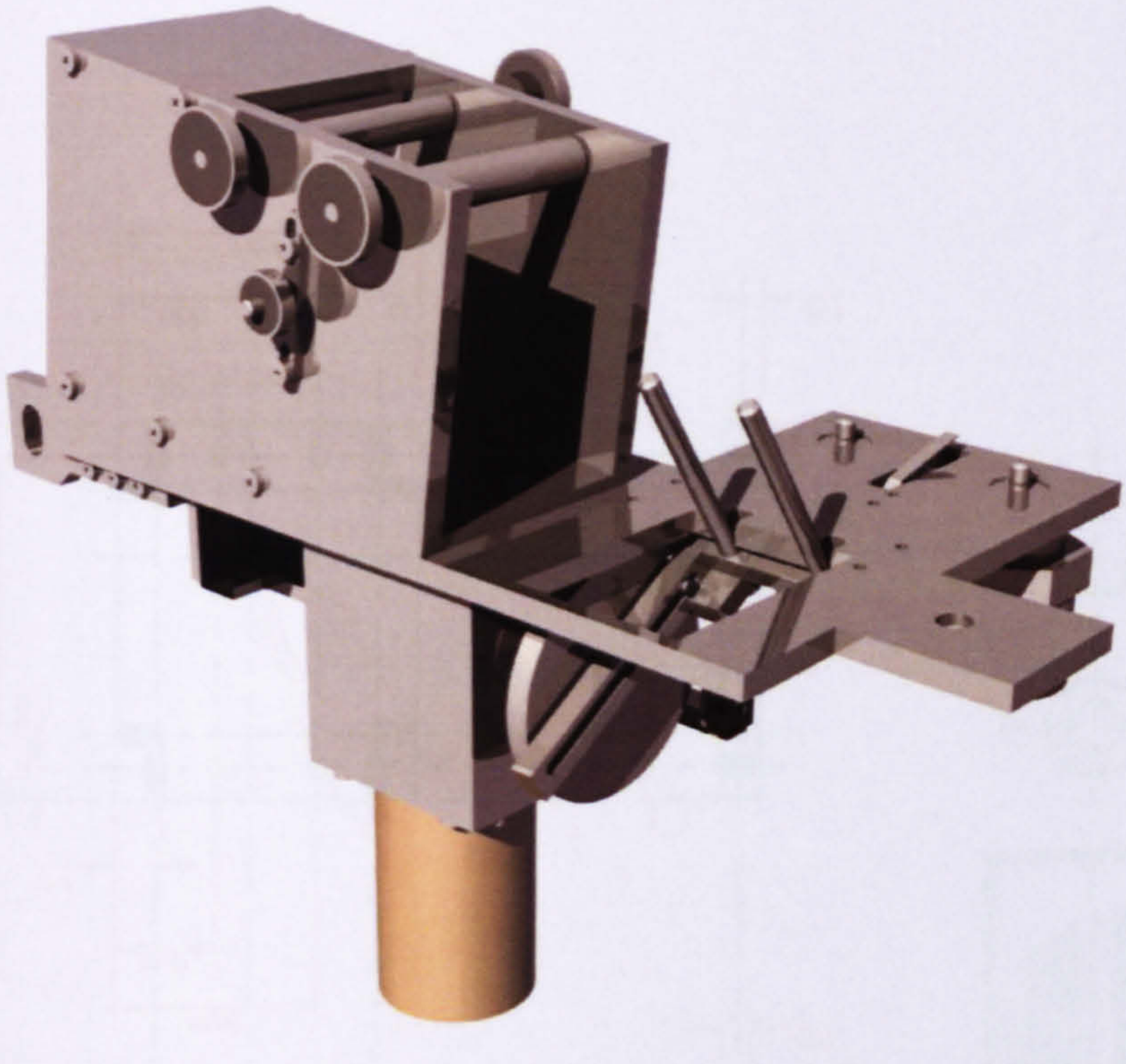


FIGURE C.11: 3D representation of the payload dispenser

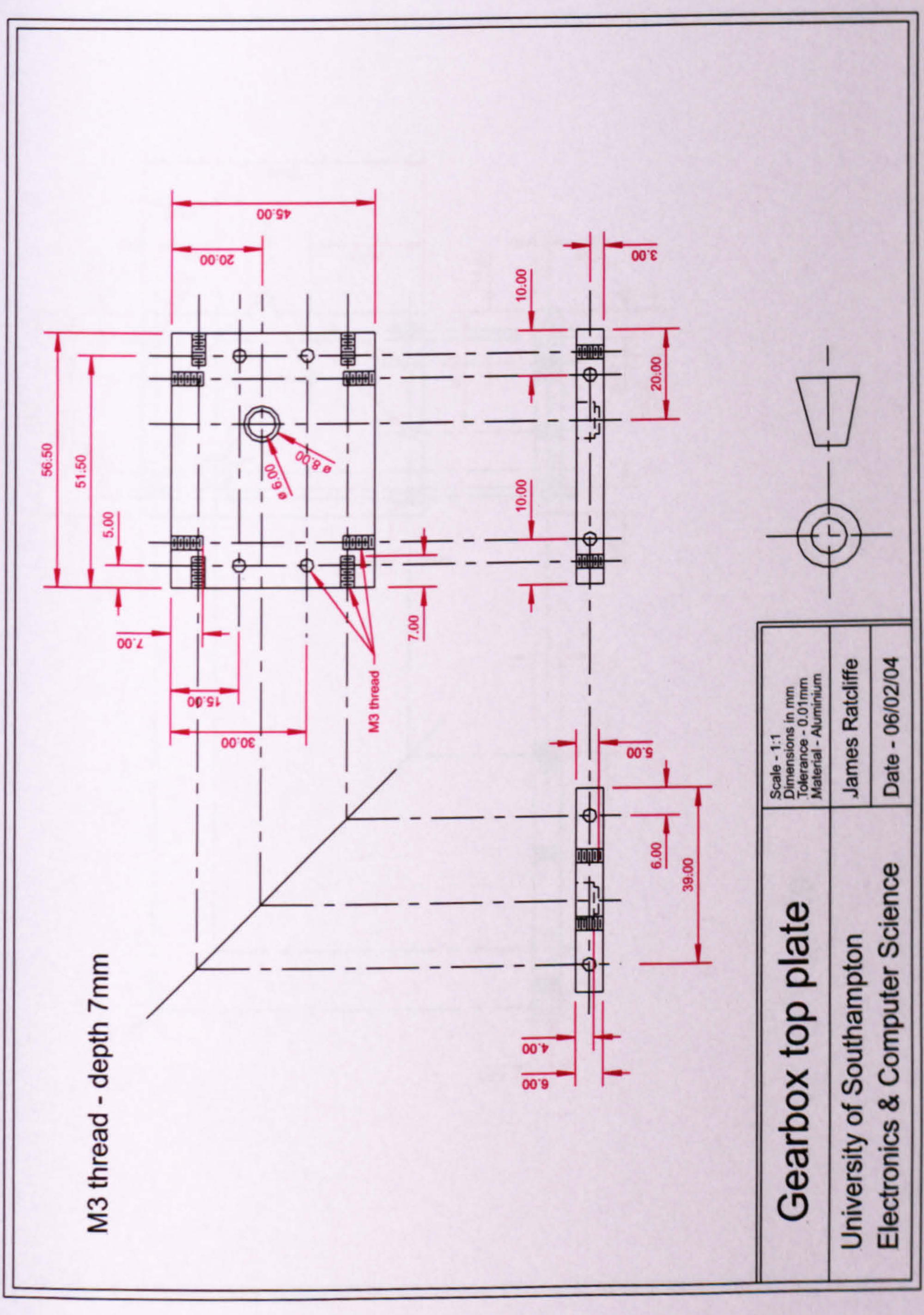


FIGURE C.12: Dispenser gearbox top plate

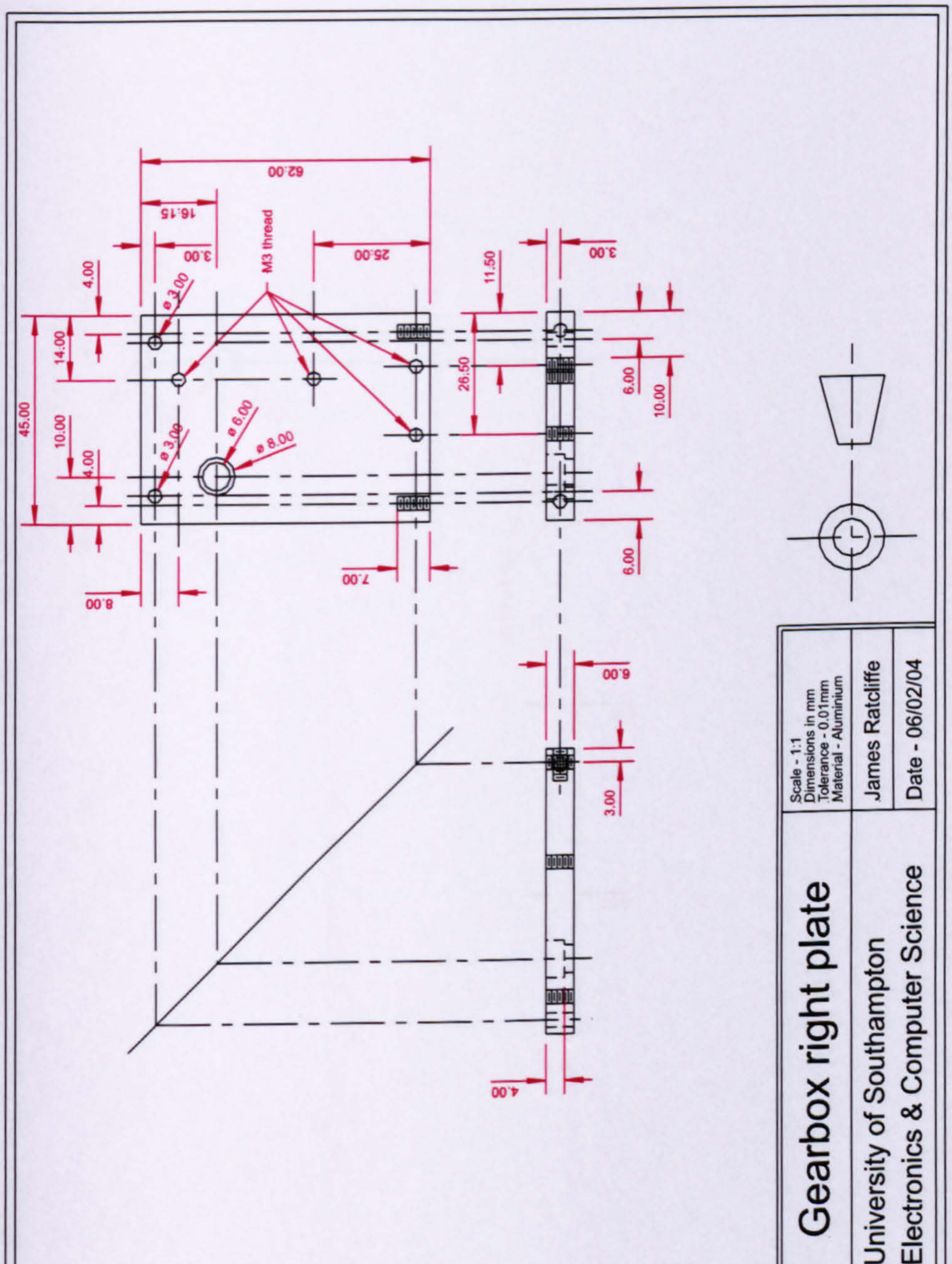


FIGURE C.13: Dispenser gearbox right plate

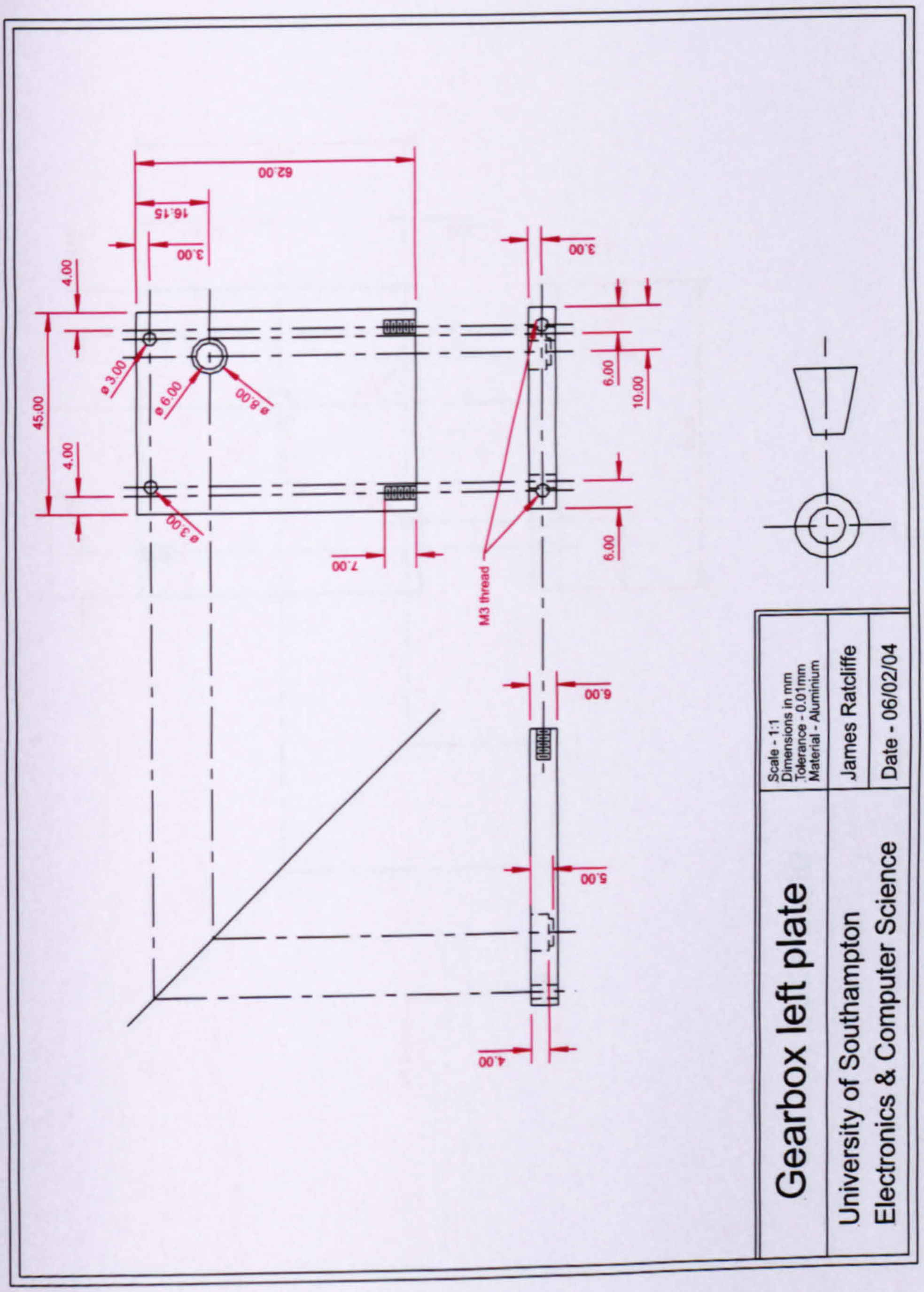


FIGURE C.14: Dispenser gearbox left plate

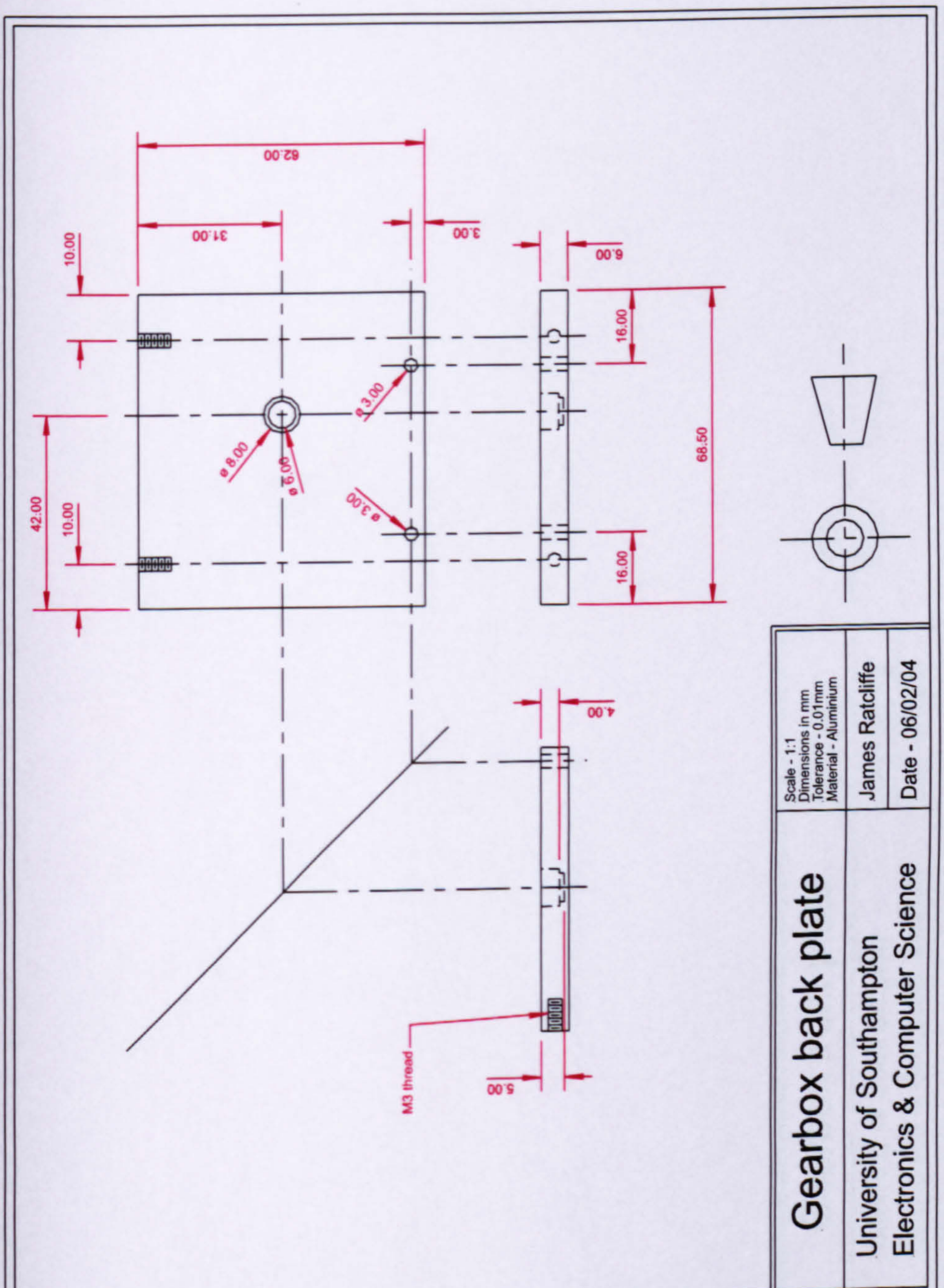


FIGURE C.15: Dispenser gearbox back plate

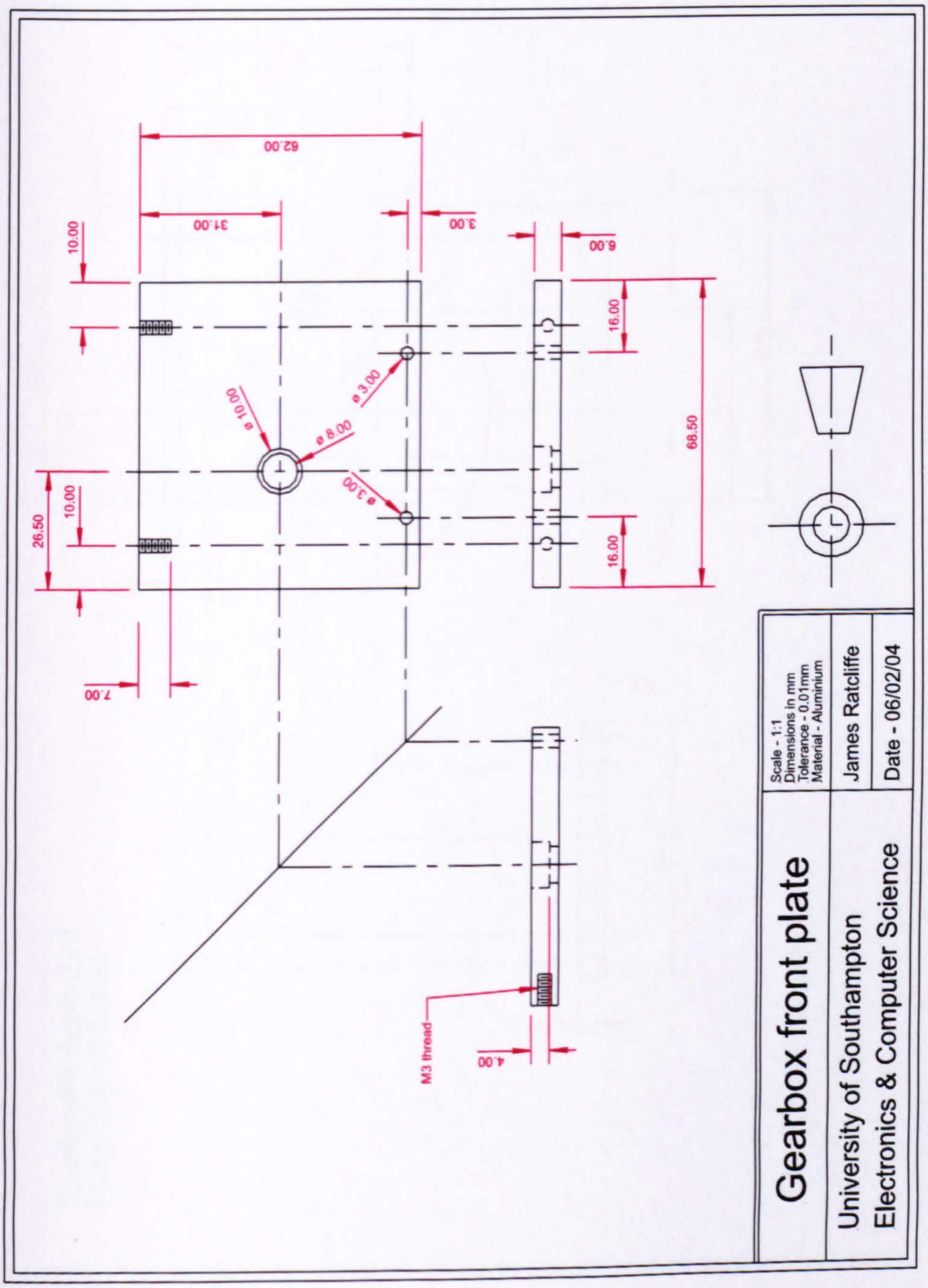


FIGURE C.16: Dispenser gearbox front plate

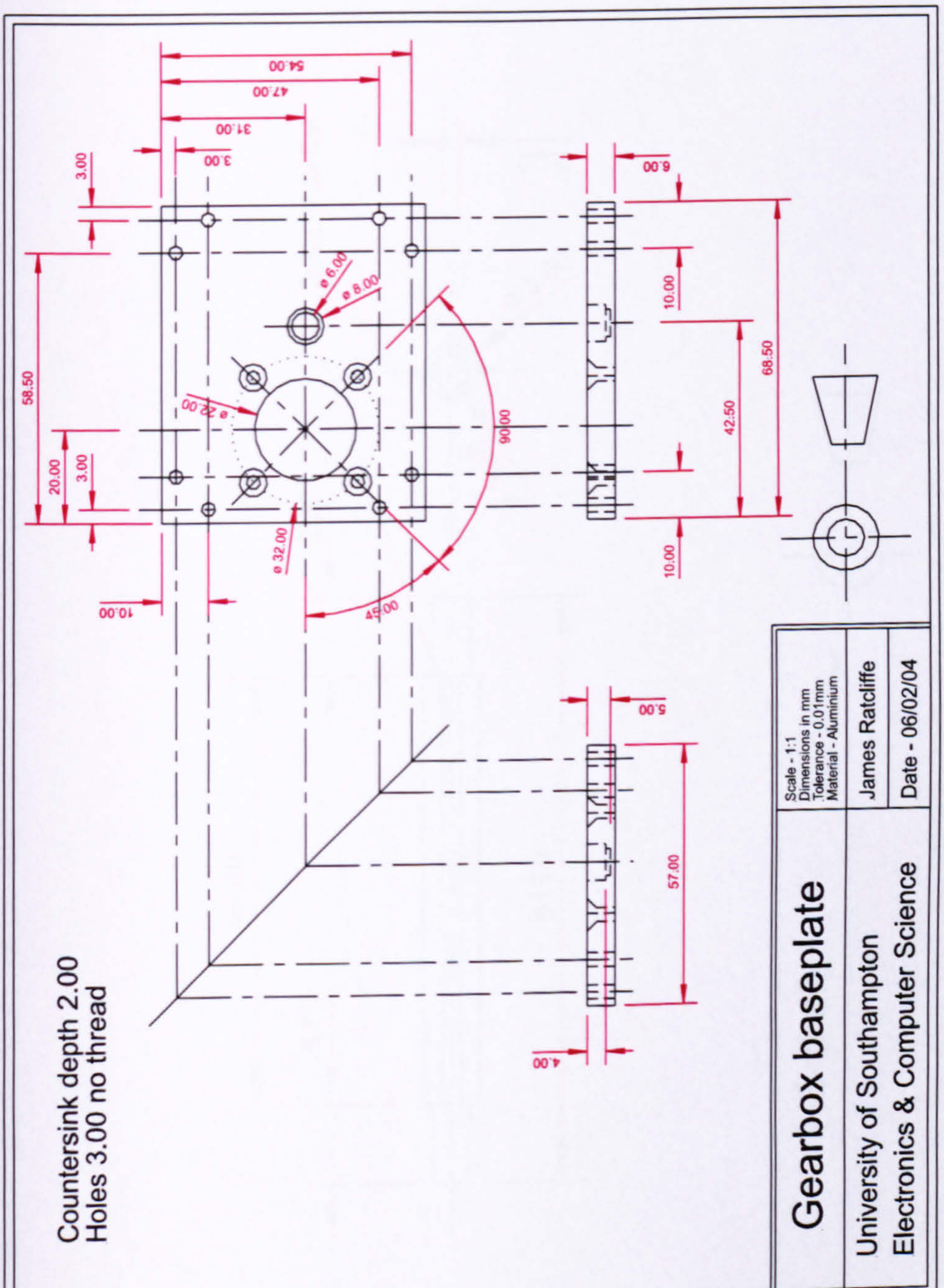


FIGURE C.17: Dispenser gearbox base plate

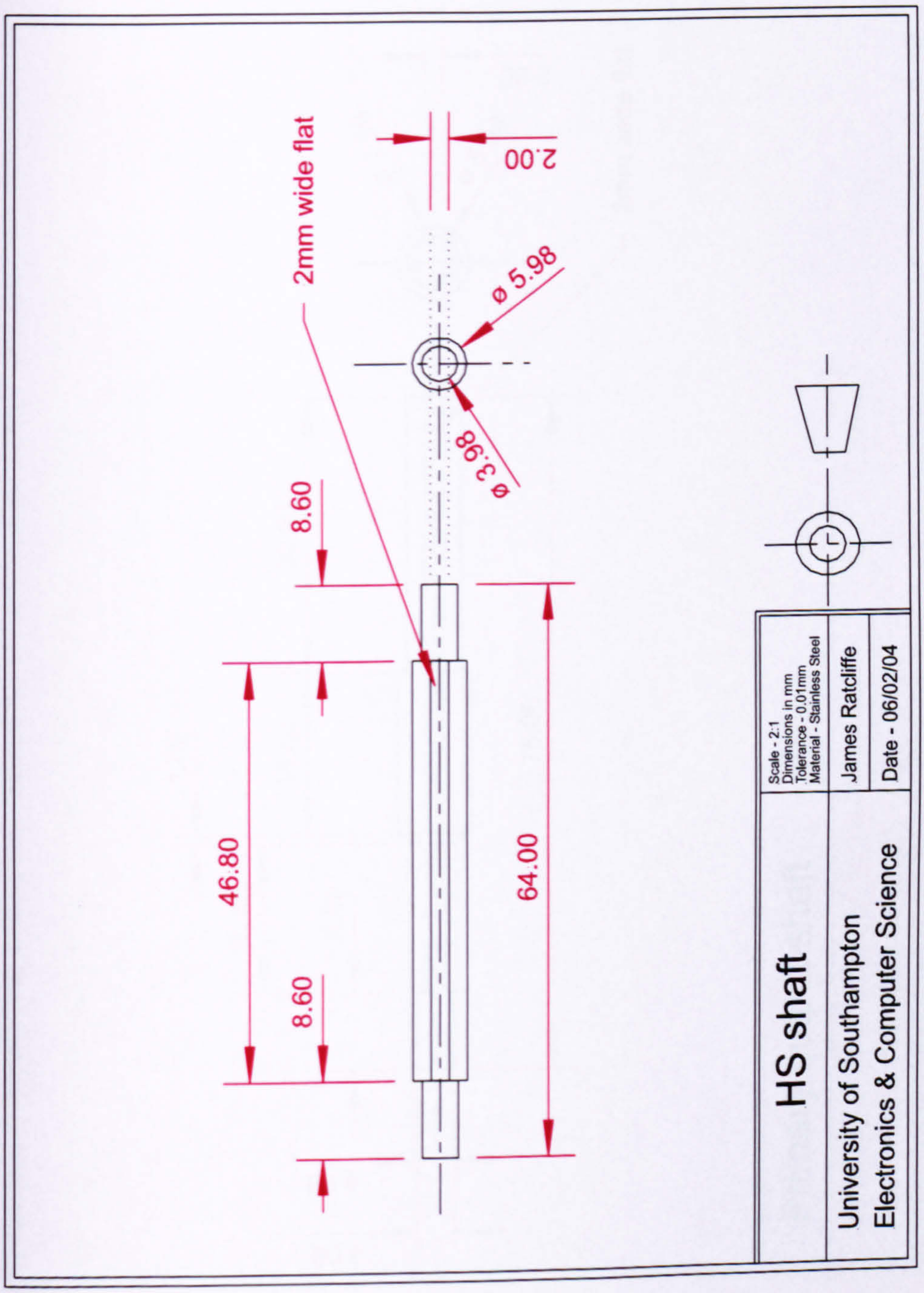


FIGURE C.18: Dispenser gearbox high speed drive shaft

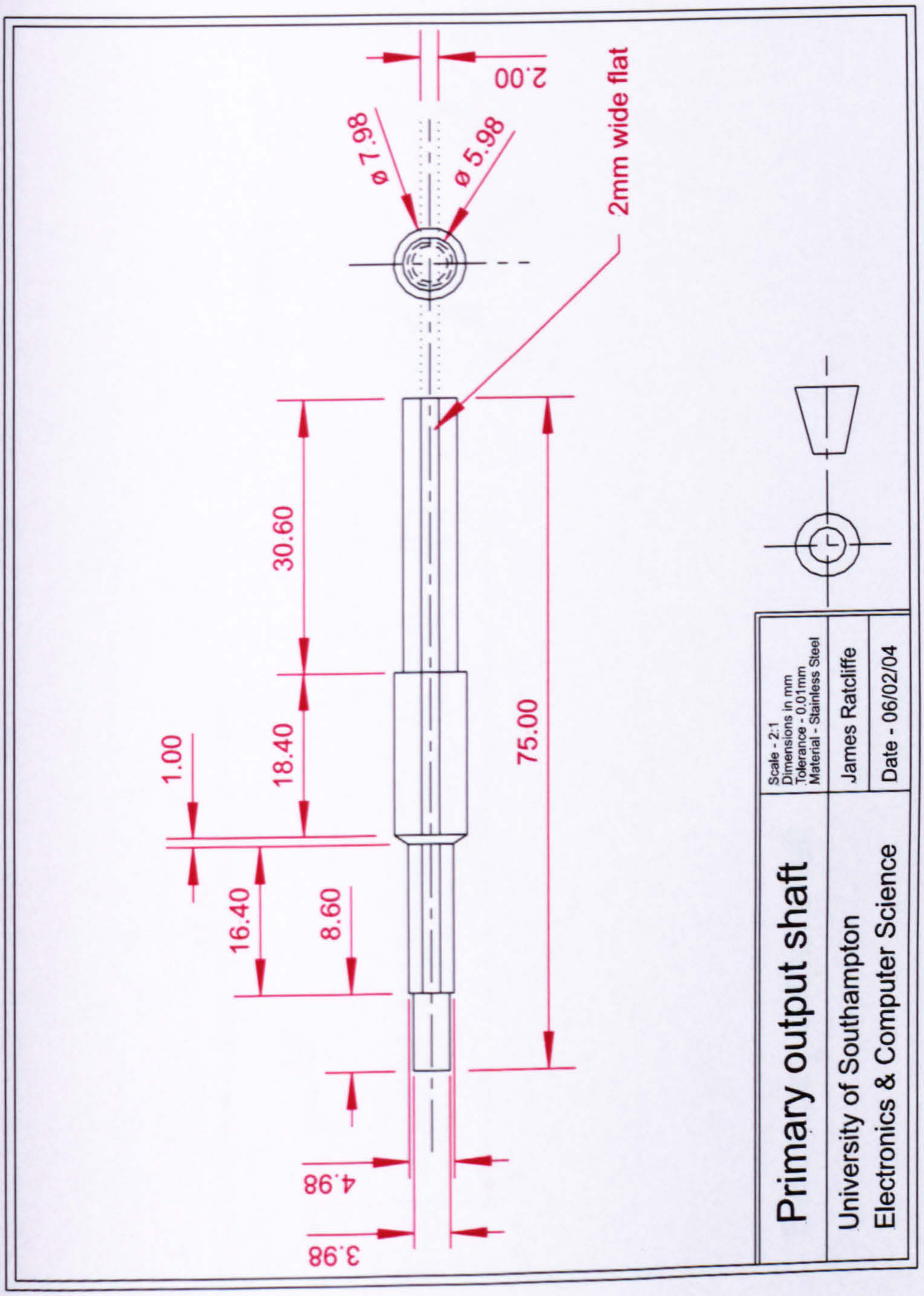


FIGURE C.19: Dispenser gearbox primary output drive shaft

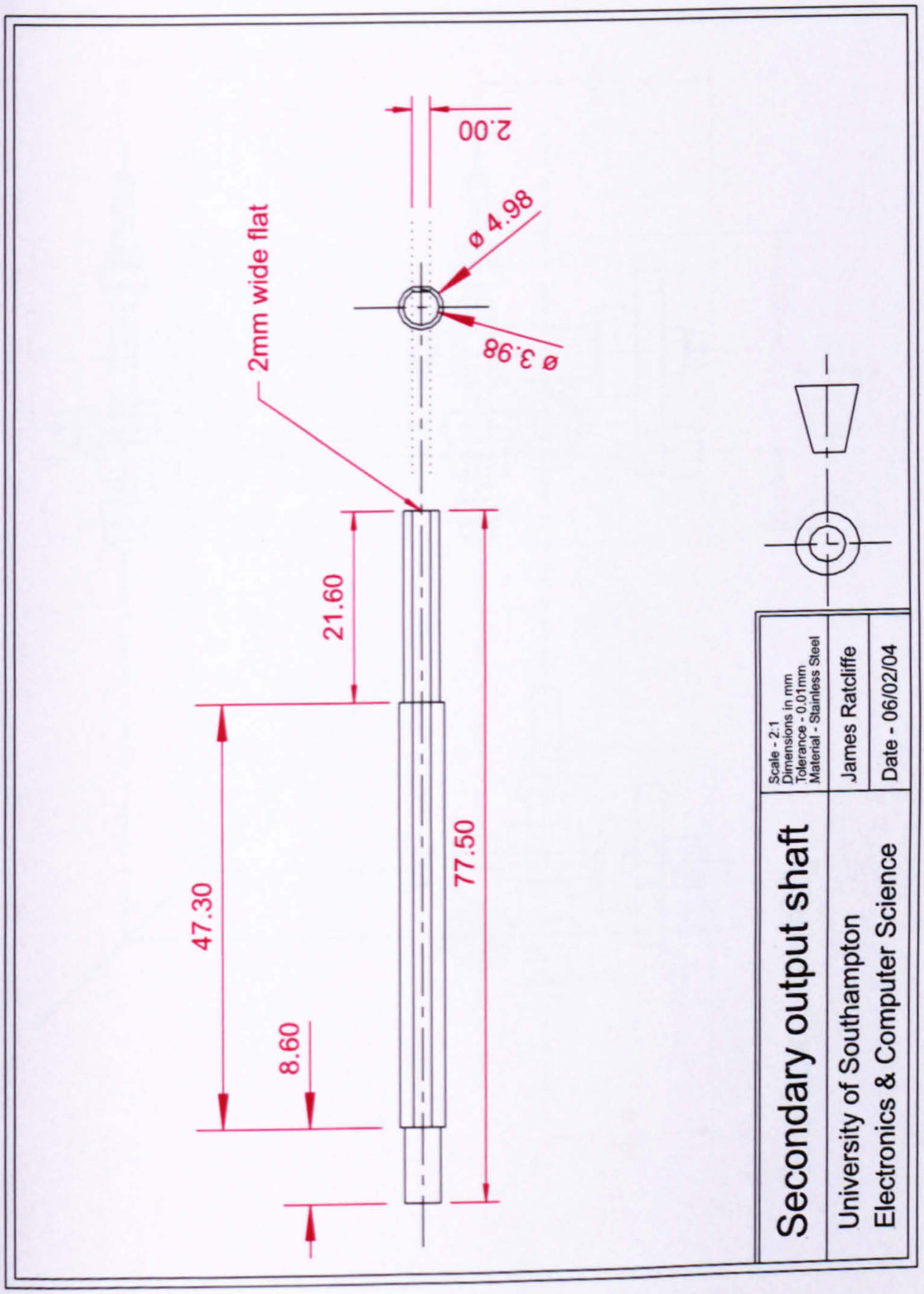


FIGURE C.20: Dispenser gearbox secondary output drive shaft

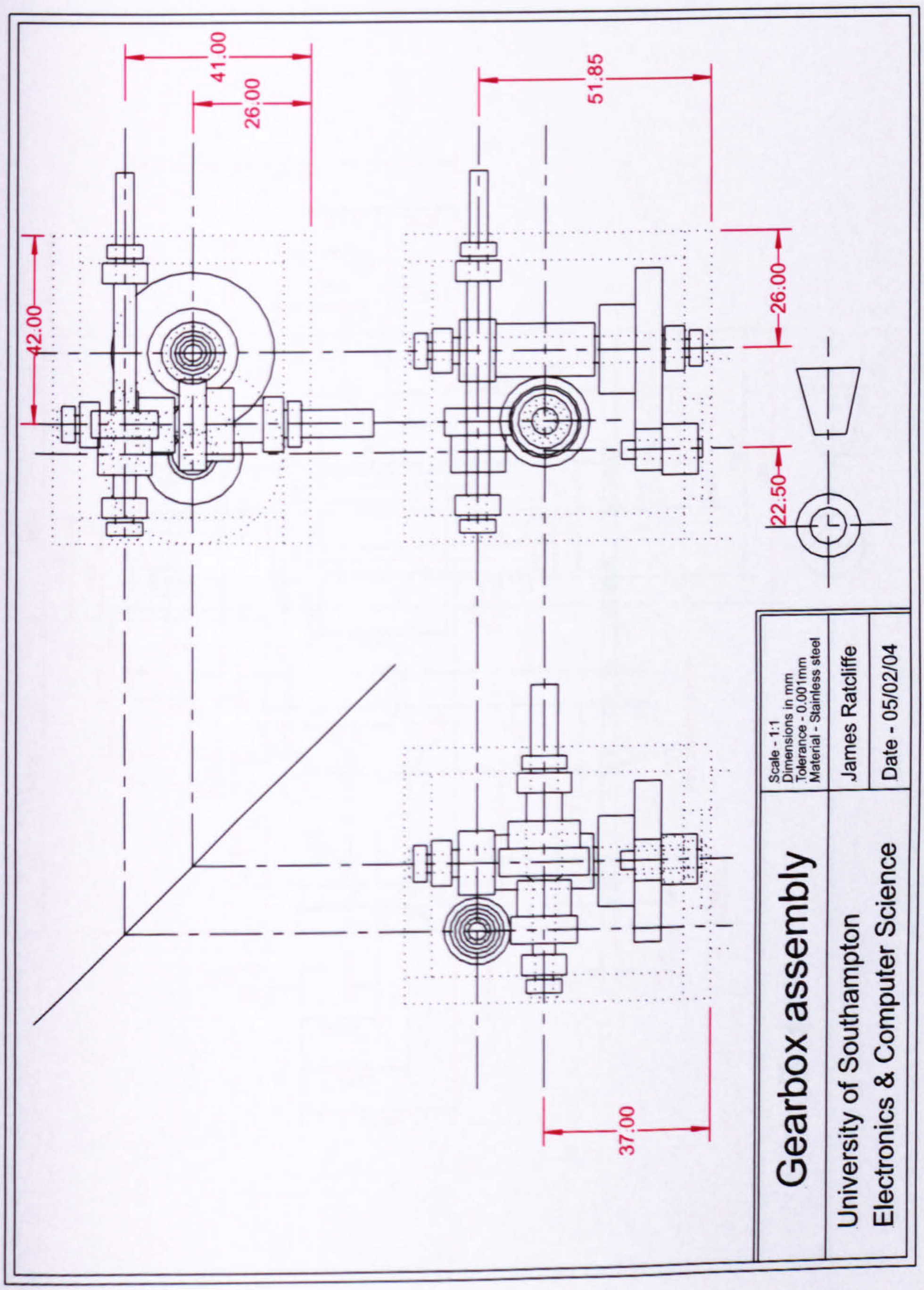


FIGURE C.21: Dispenser gearbox assembly

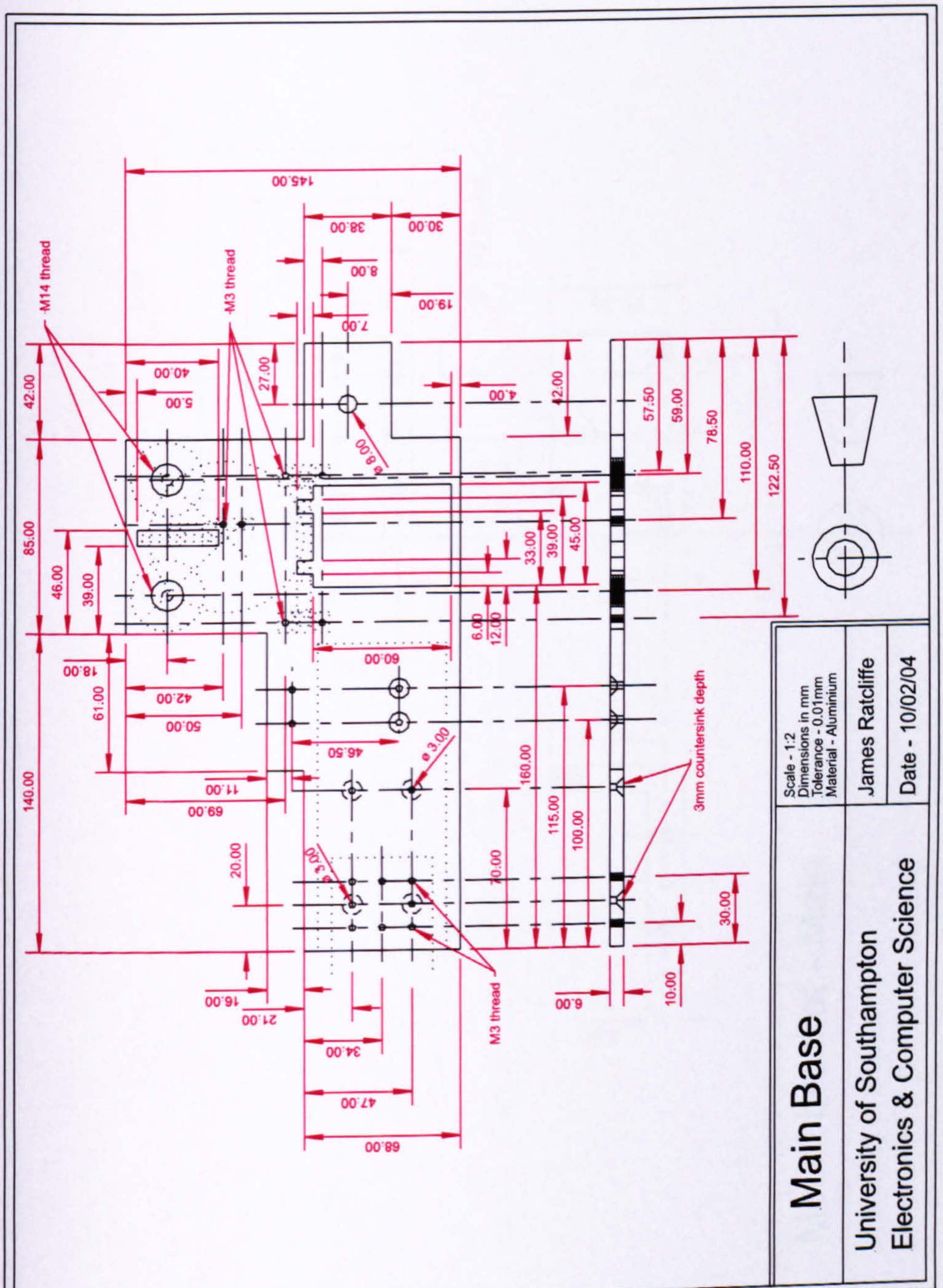


FIGURE C.22: Dispenser base plate

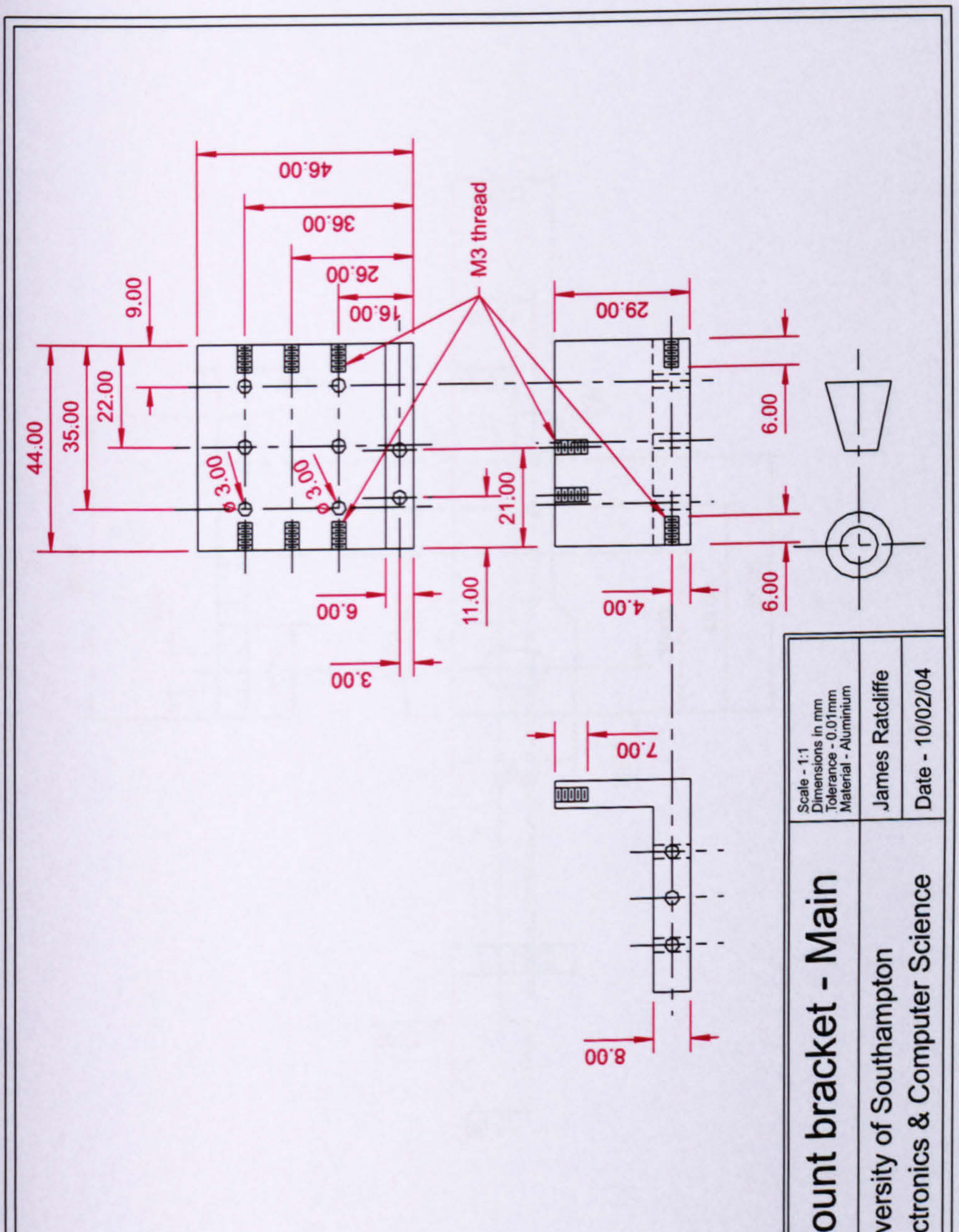


FIGURE C.23: Dispenser mount bracket

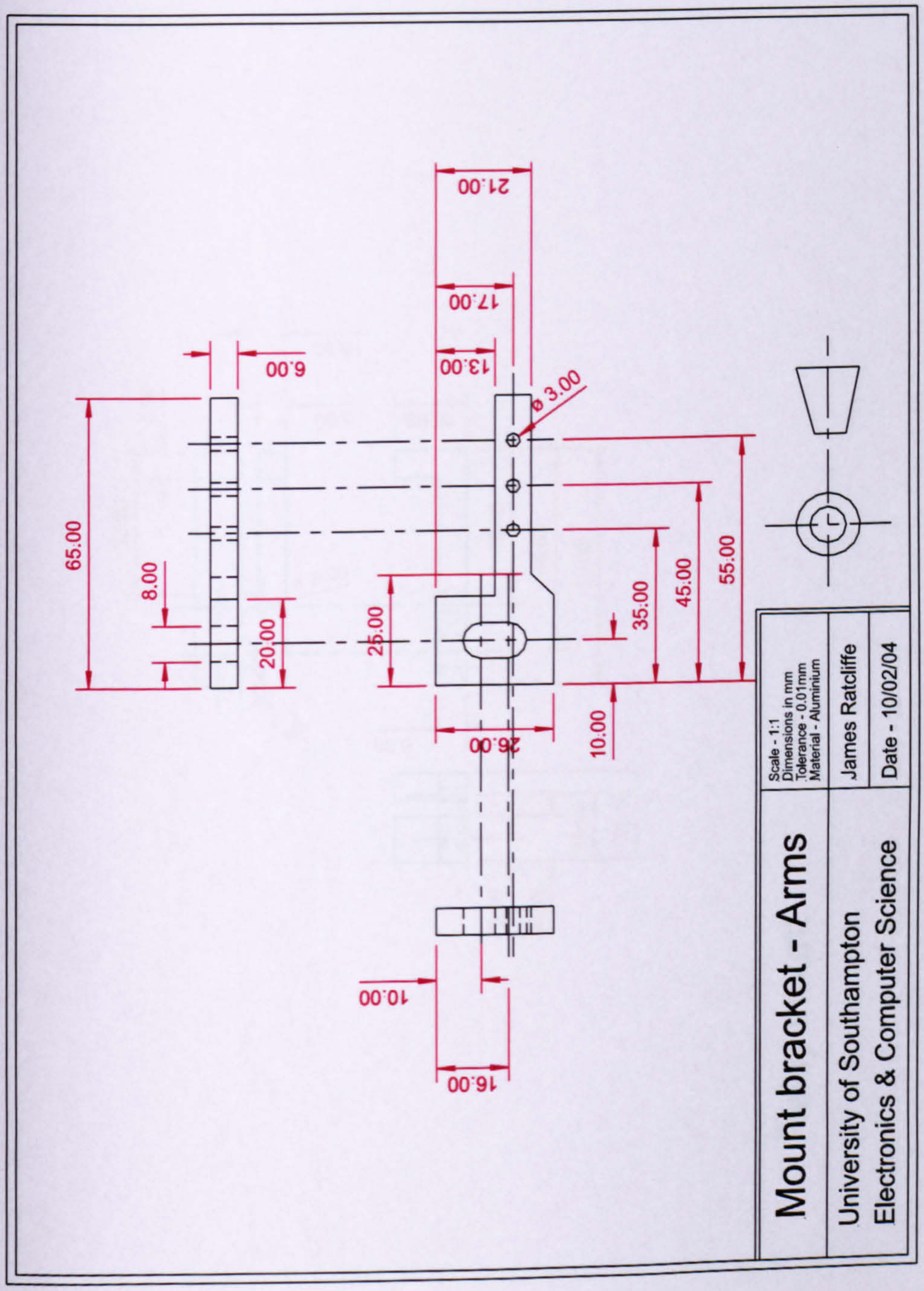
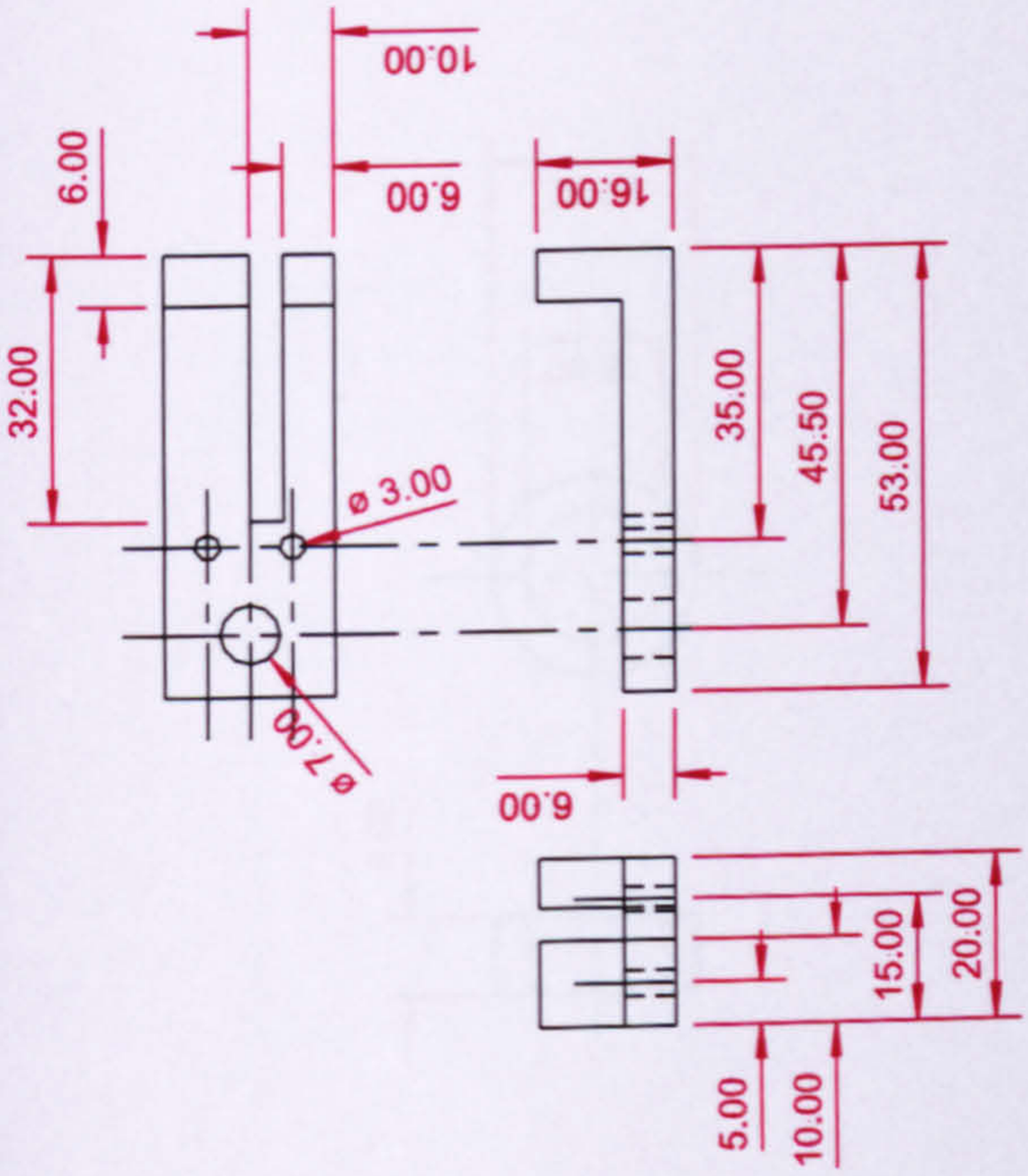
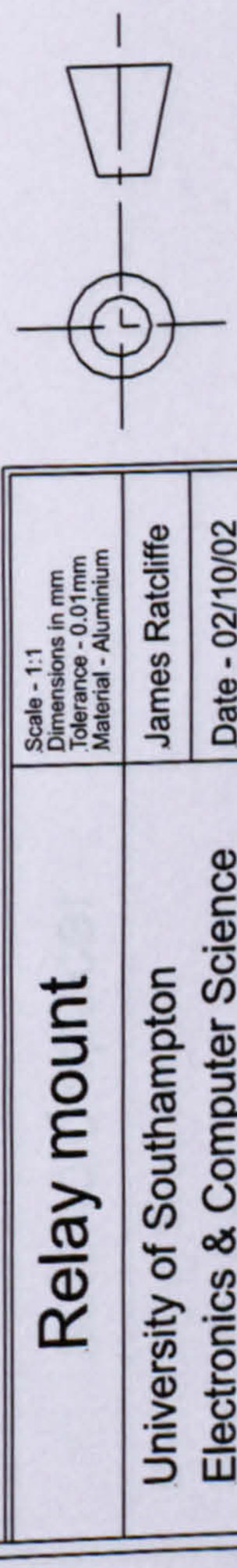


FIGURE C.24: Dispenser mount bracket arms





Relay mount	Scale - 1:1 Dimensions in mm Tolerance - 0.01mm Material - Aluminium	
University of Southampton	James Ratcliffe	
Electronics & Computer Science	Date - 02/10/02	

FIGURE C.25: Dispenser relay bracket

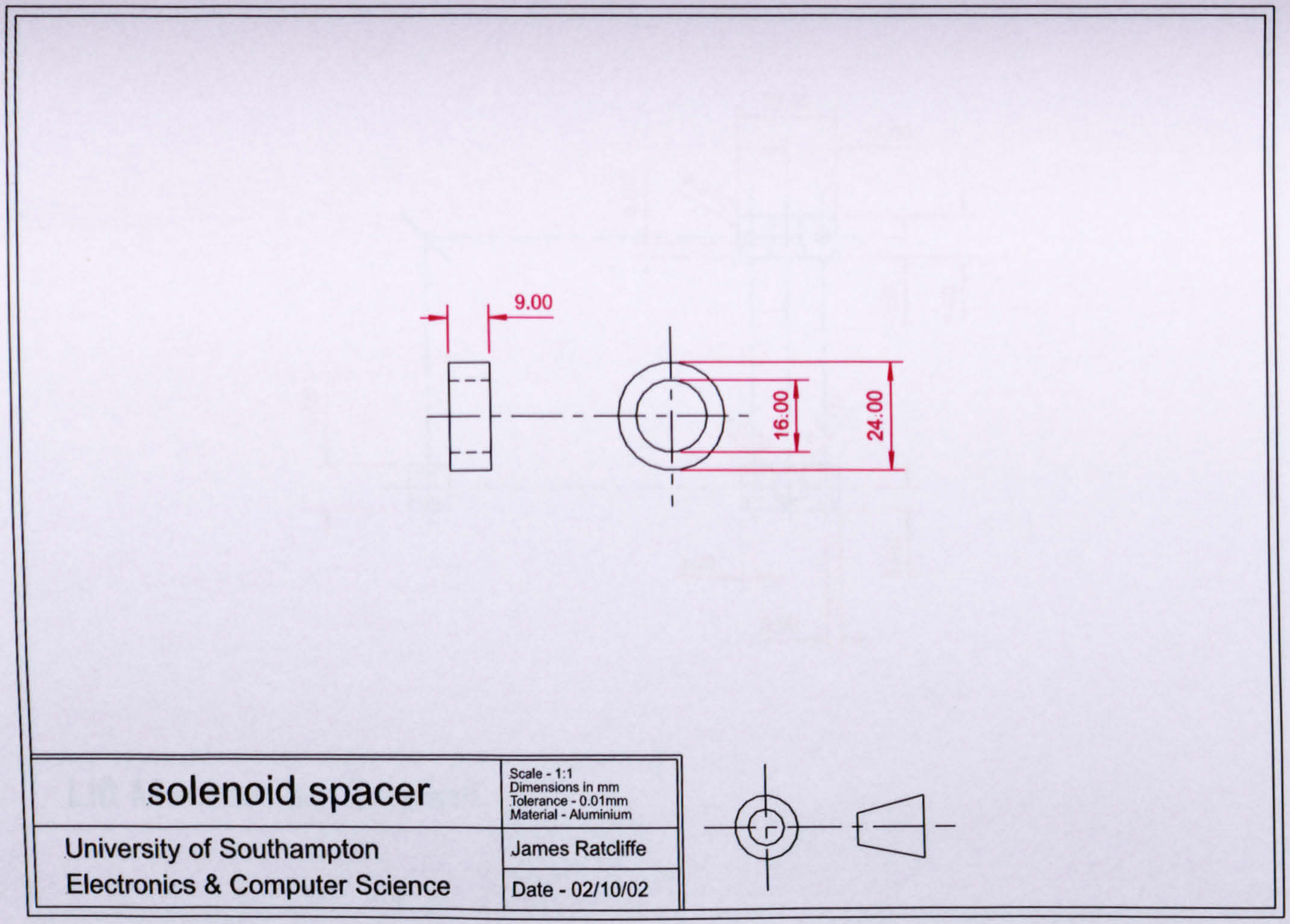
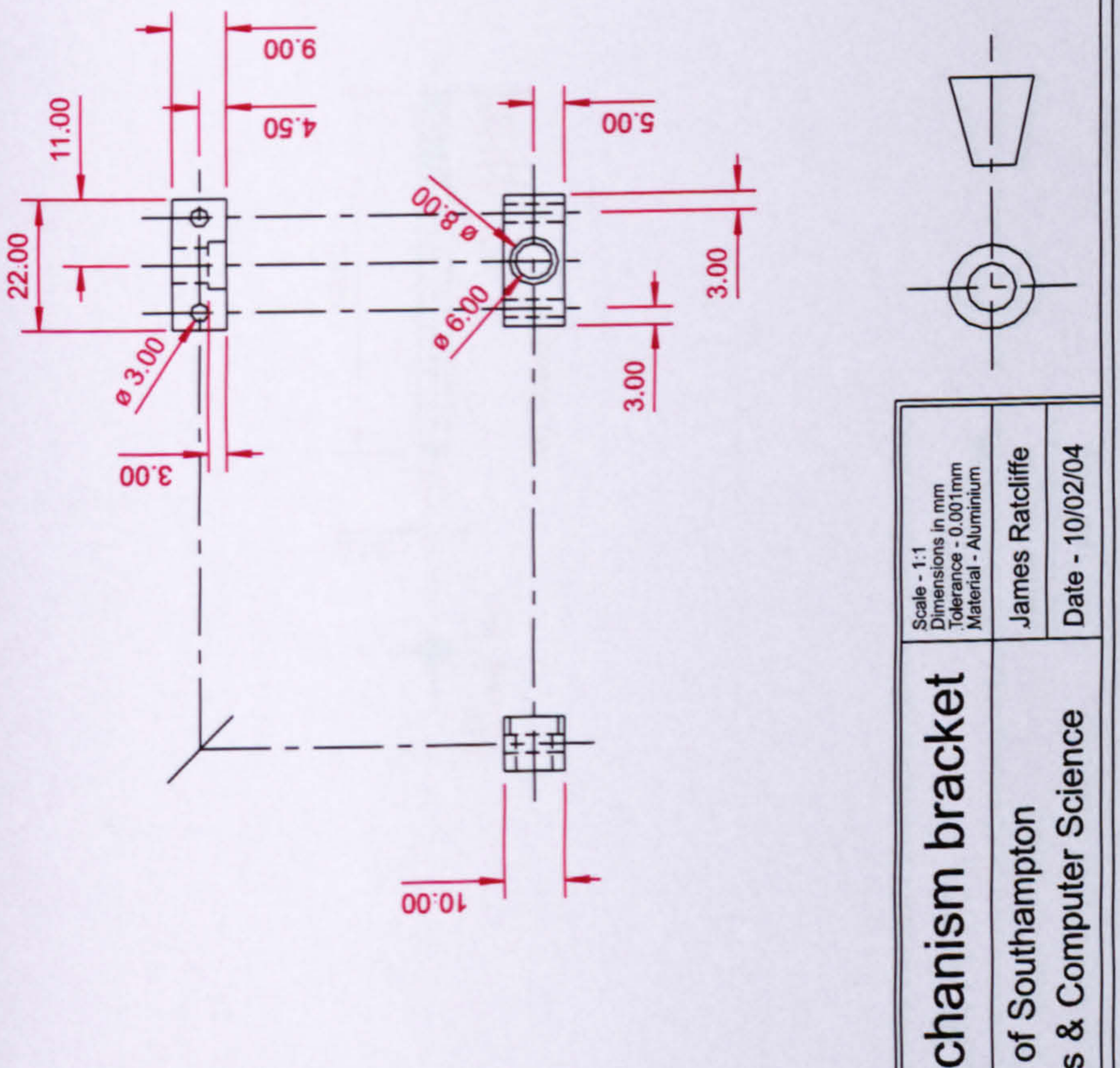


FIGURE C.26: Dispenser solenoid spacer ring



Lift Mechanism bracket

University of Southampton
Electronics & Computer Science

Scale - 1:1
Dimensions in mm
Tolerance - 0.001mm
Material - Aluminium

James Ratcliffe

Date - 10/02/04

FIGURE C.27: Dispenser bearing housing

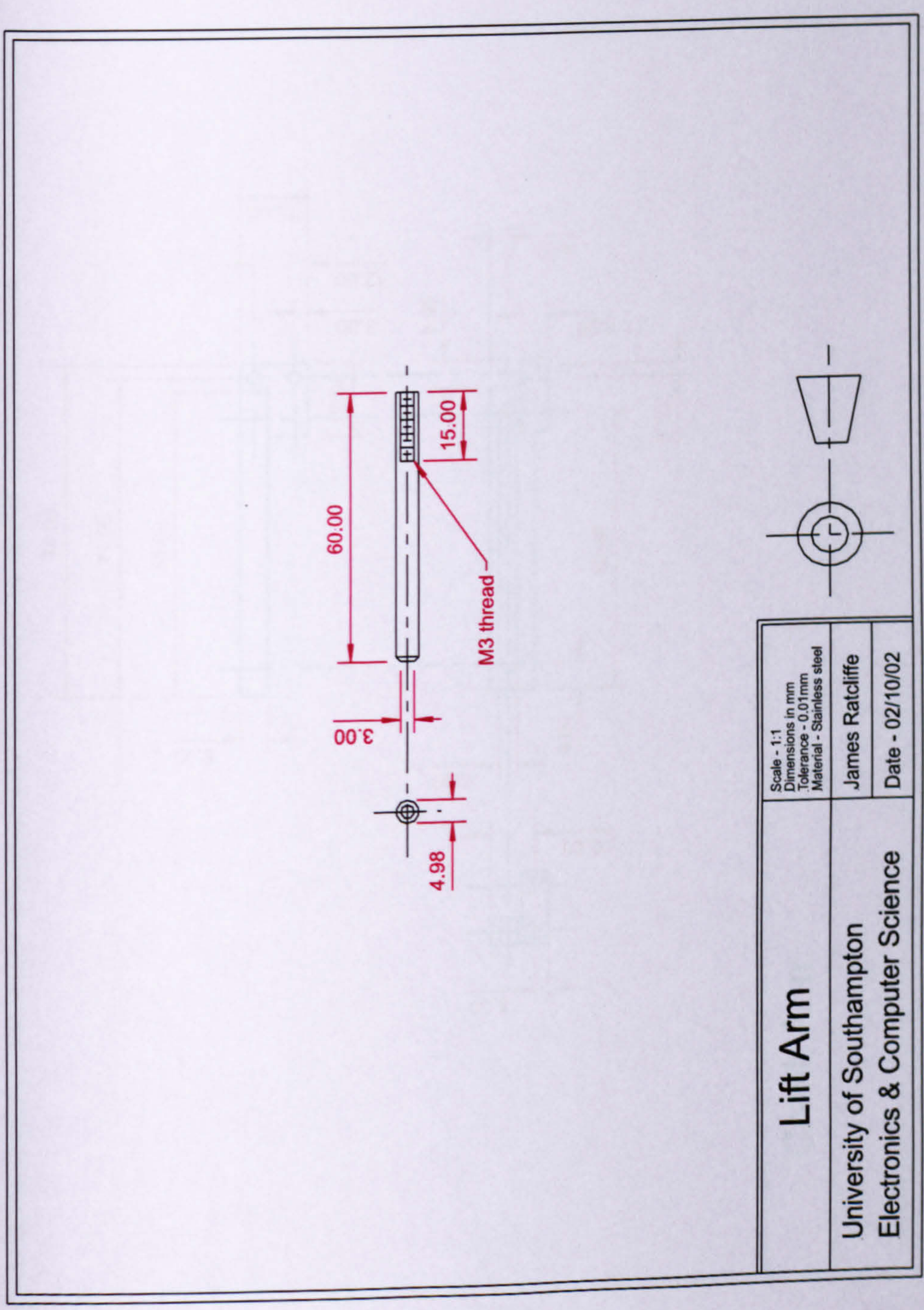


FIGURE C.28: Dispenser payload lift arms

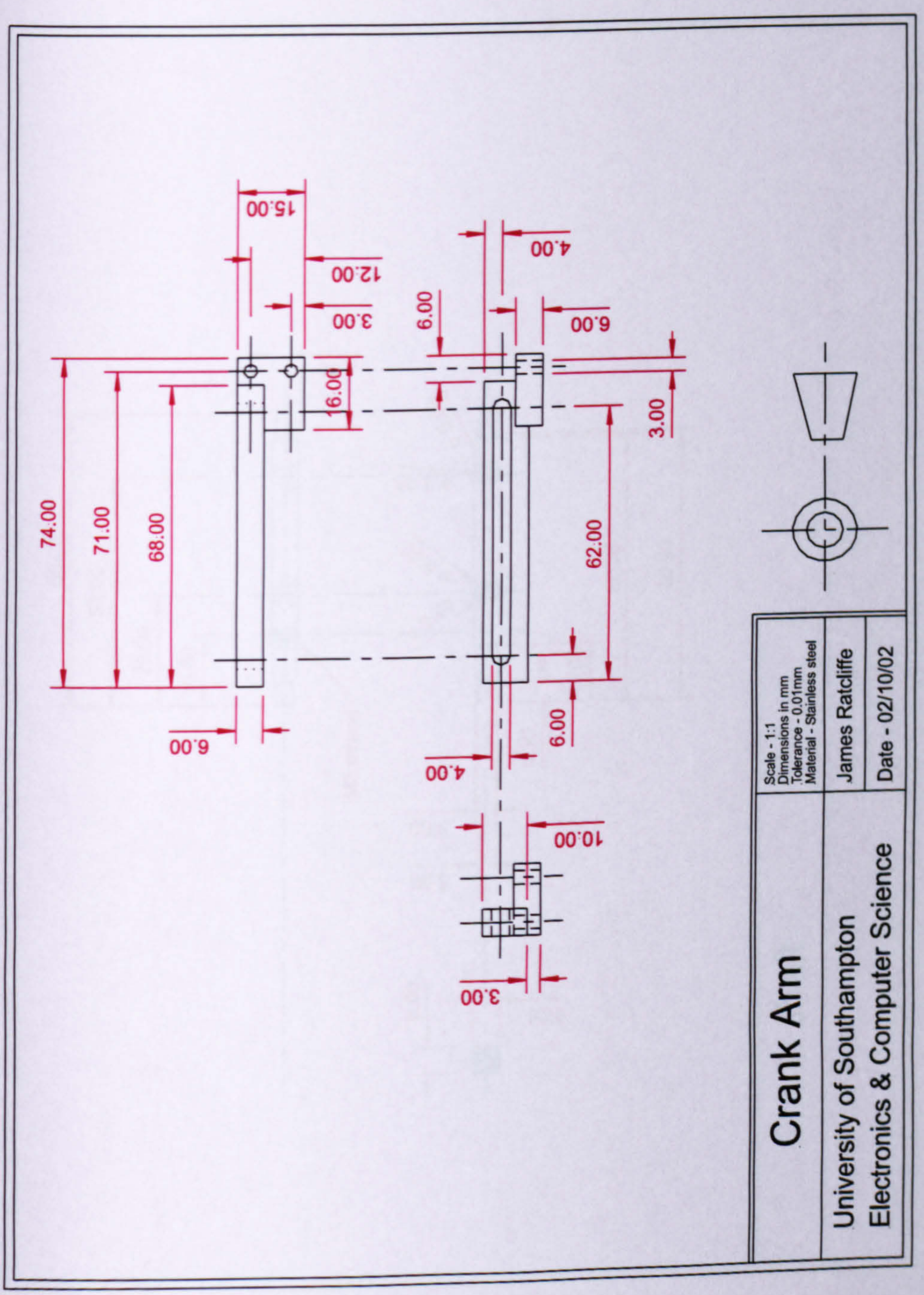


FIGURE C.29: Dispenser crank arm

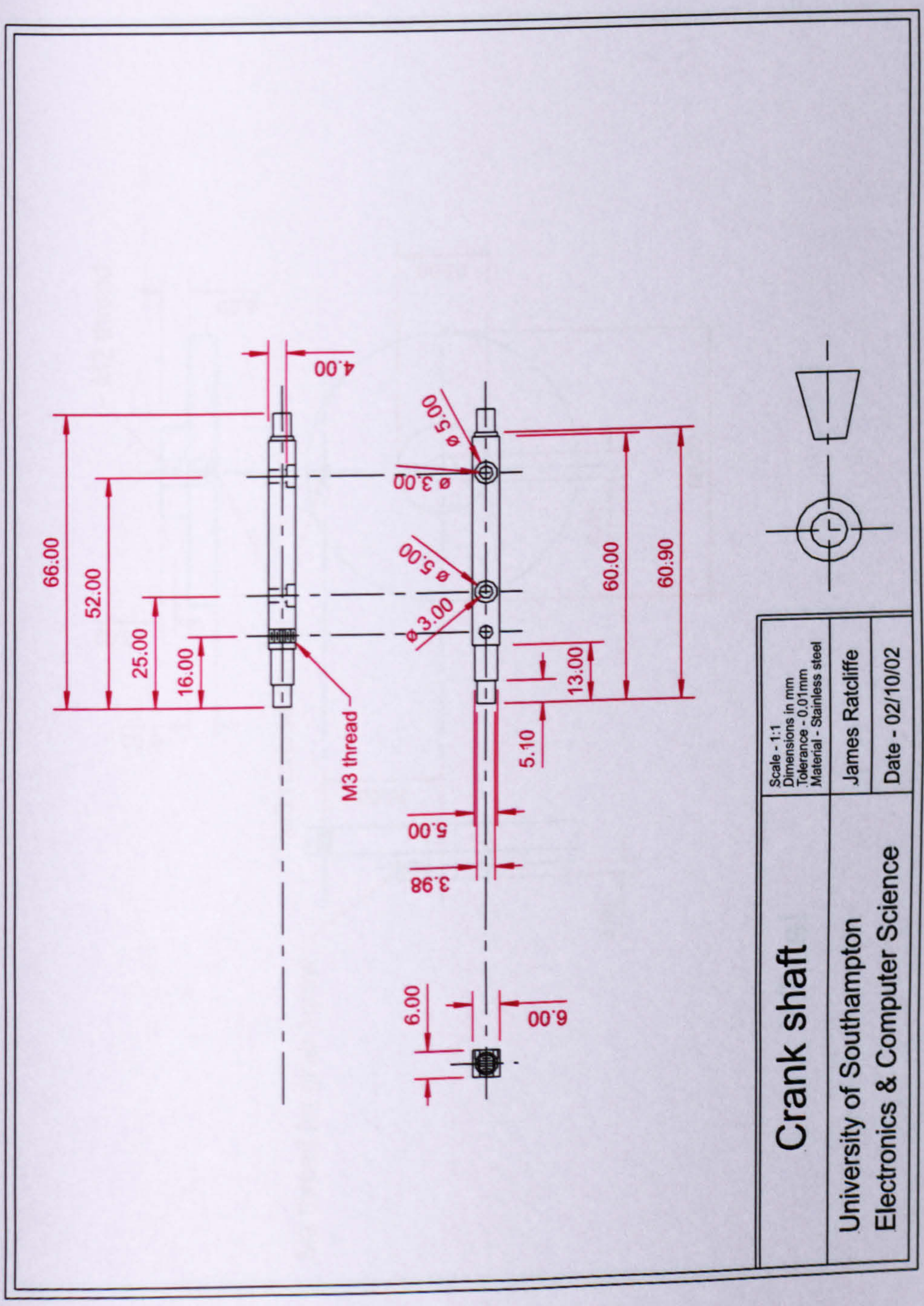


FIGURE C.30: Dispenser crank shaft

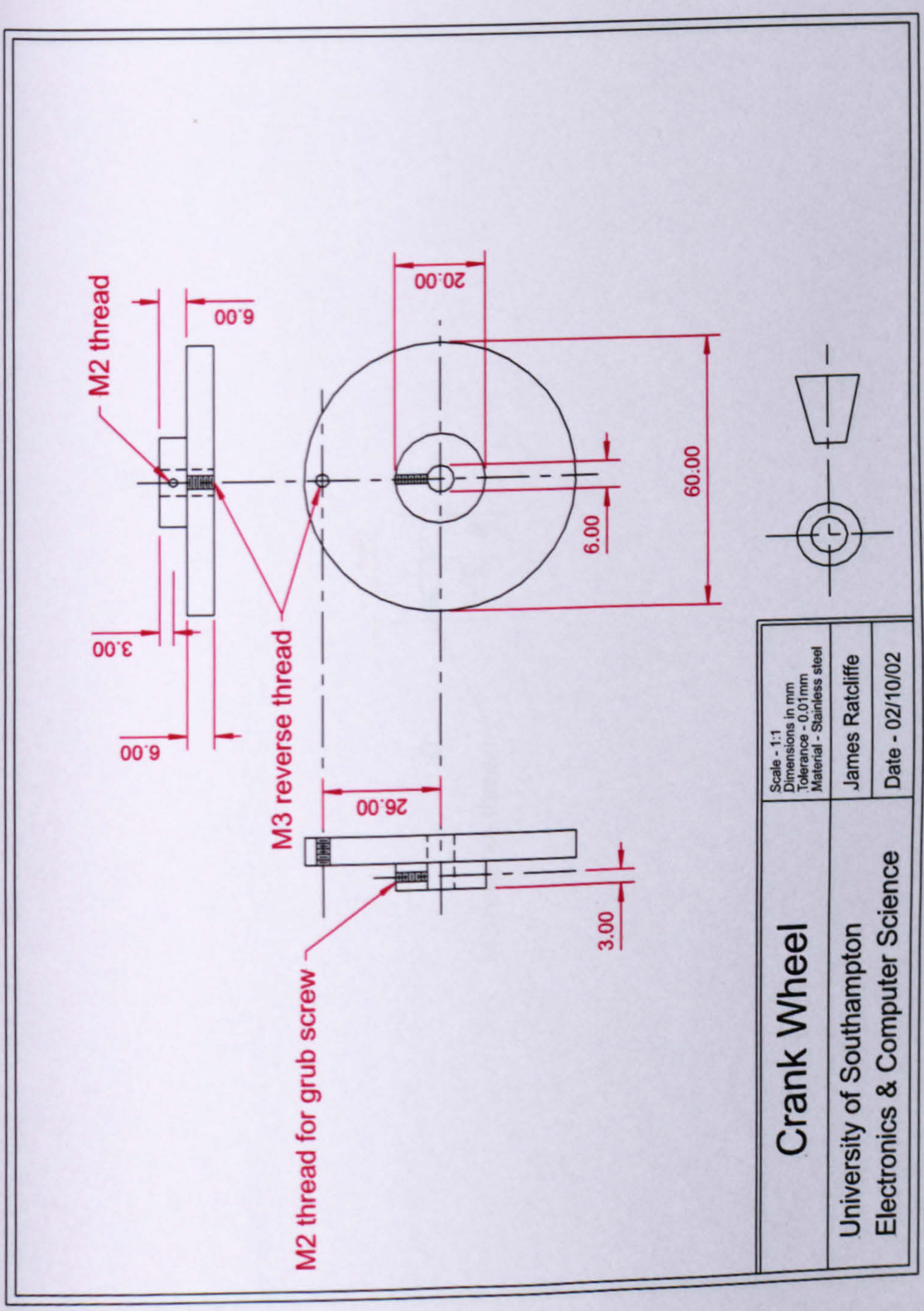


FIGURE C.31: Dispenser crank wheel

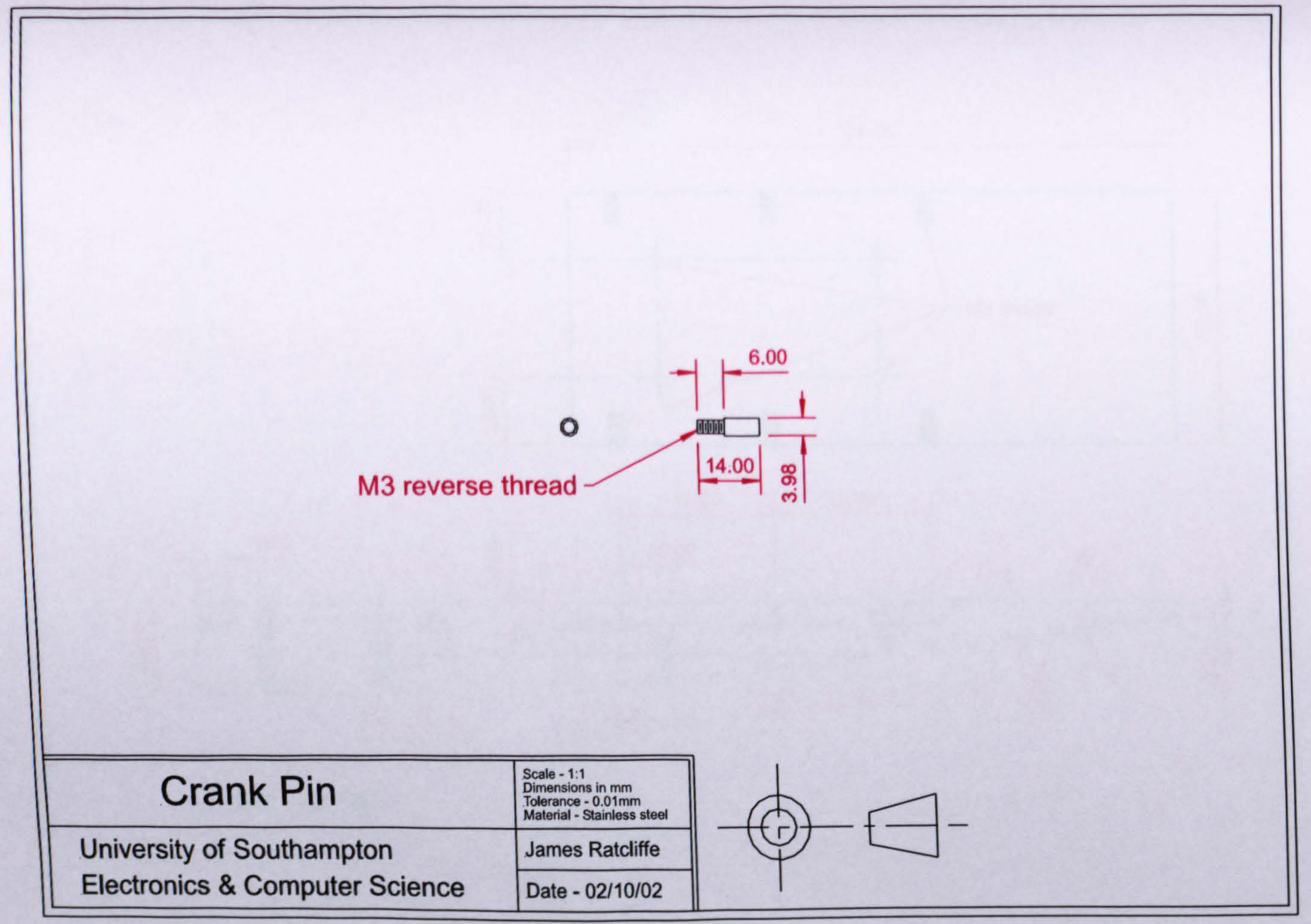


FIGURE C.32: Dispenser crank pin

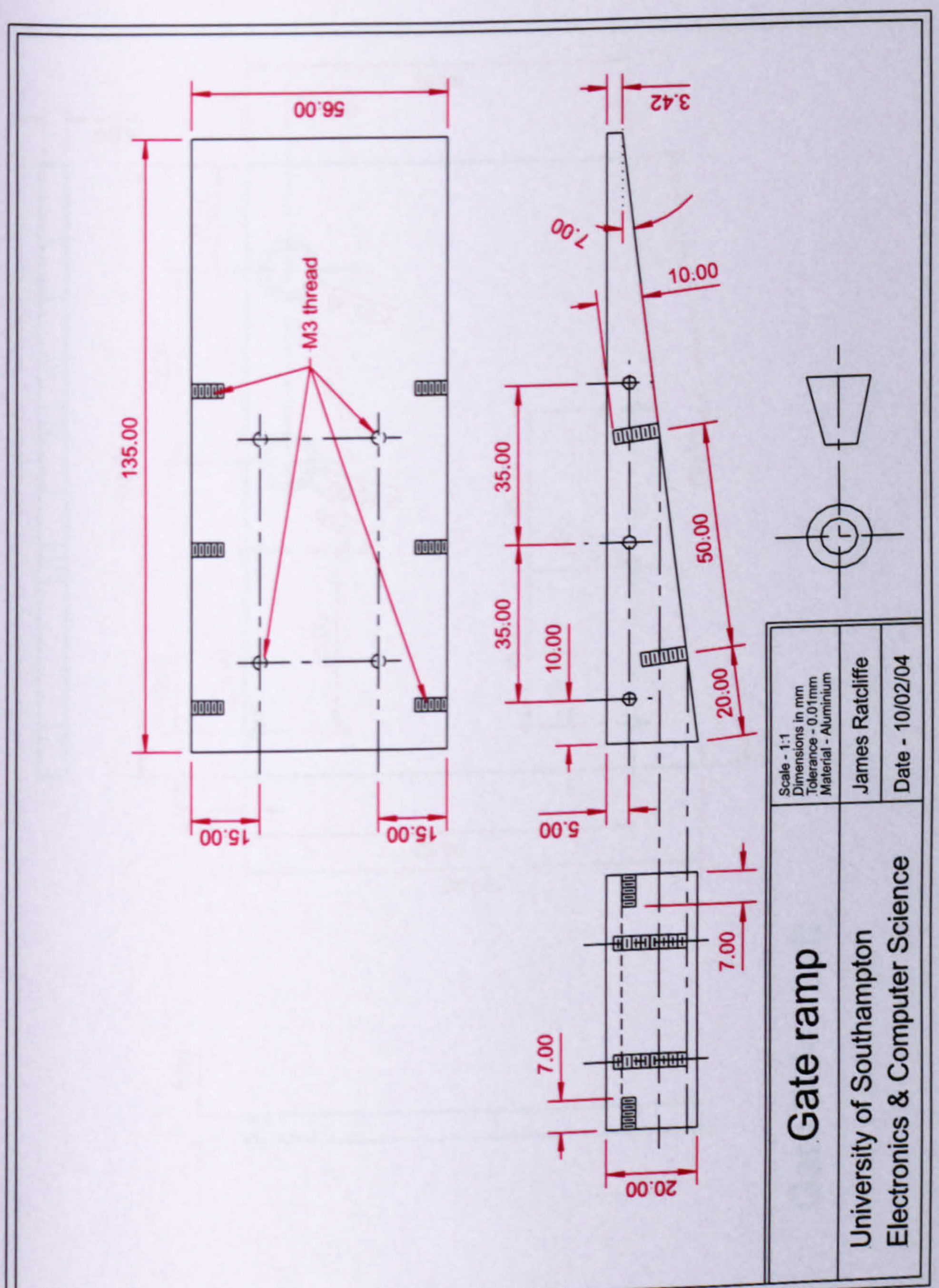


FIGURE C.33: Dispenser payload ramp

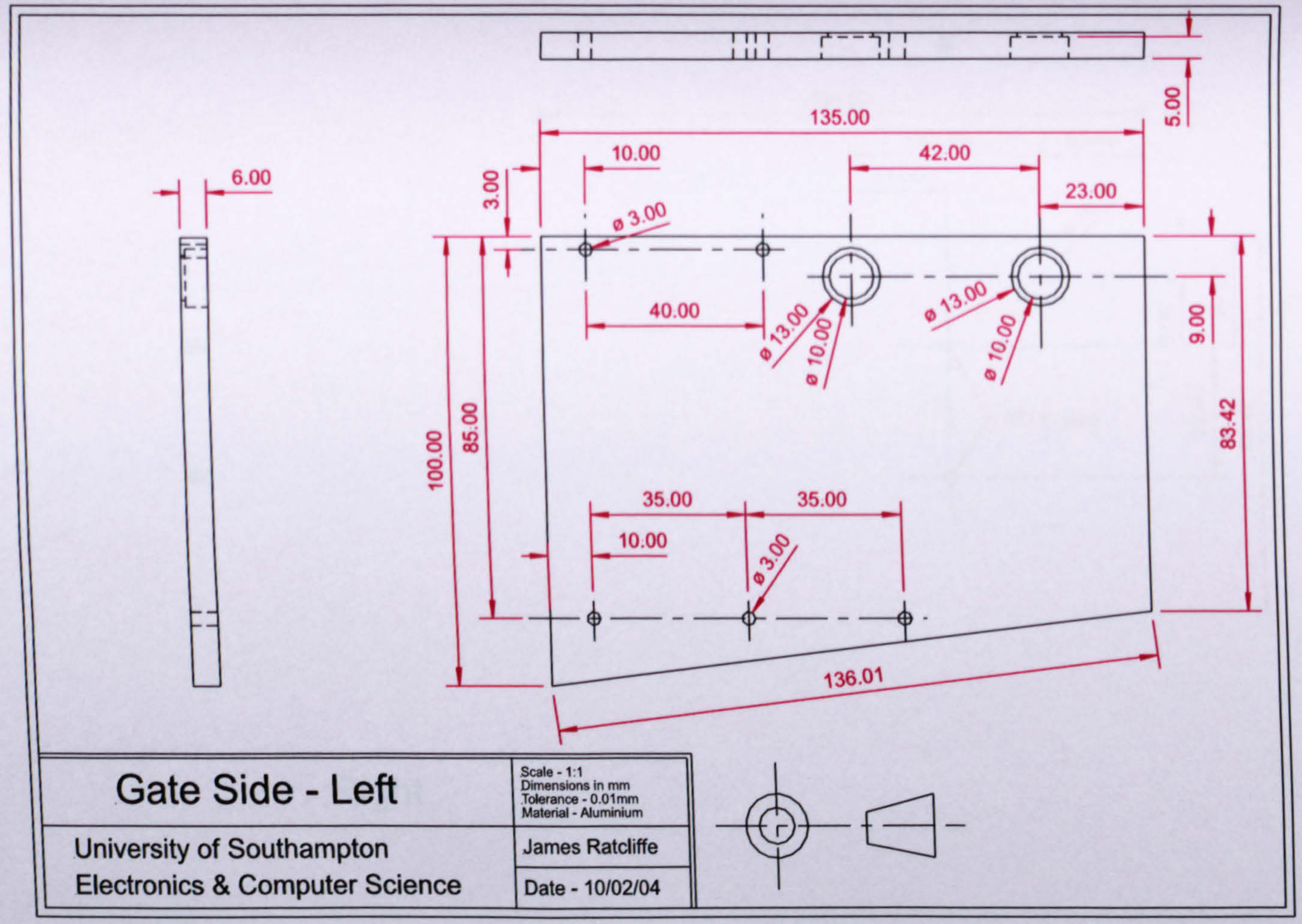


FIGURE C.34: Dispenser gate side left

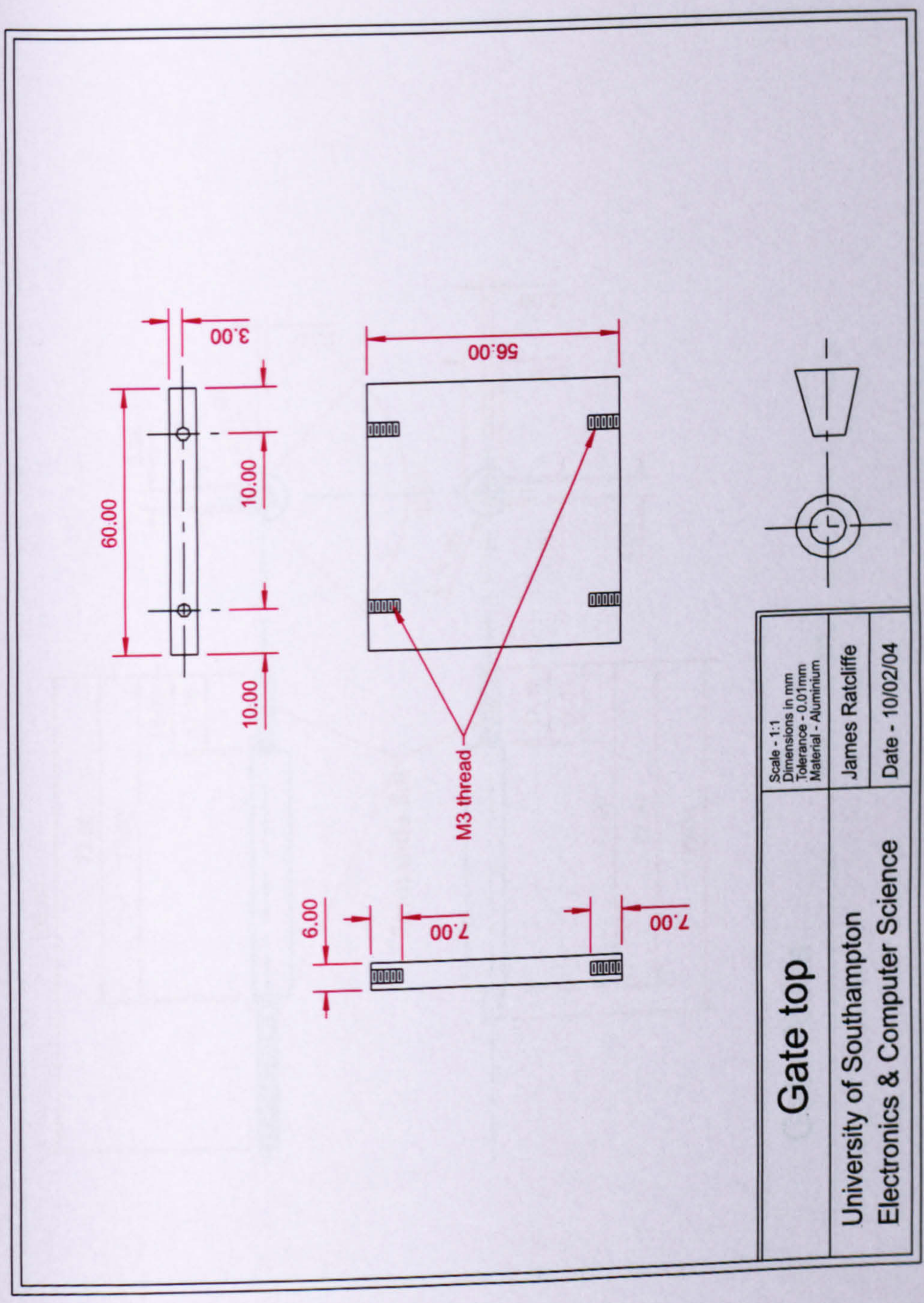


FIGURE C.36: Dispenser gate top

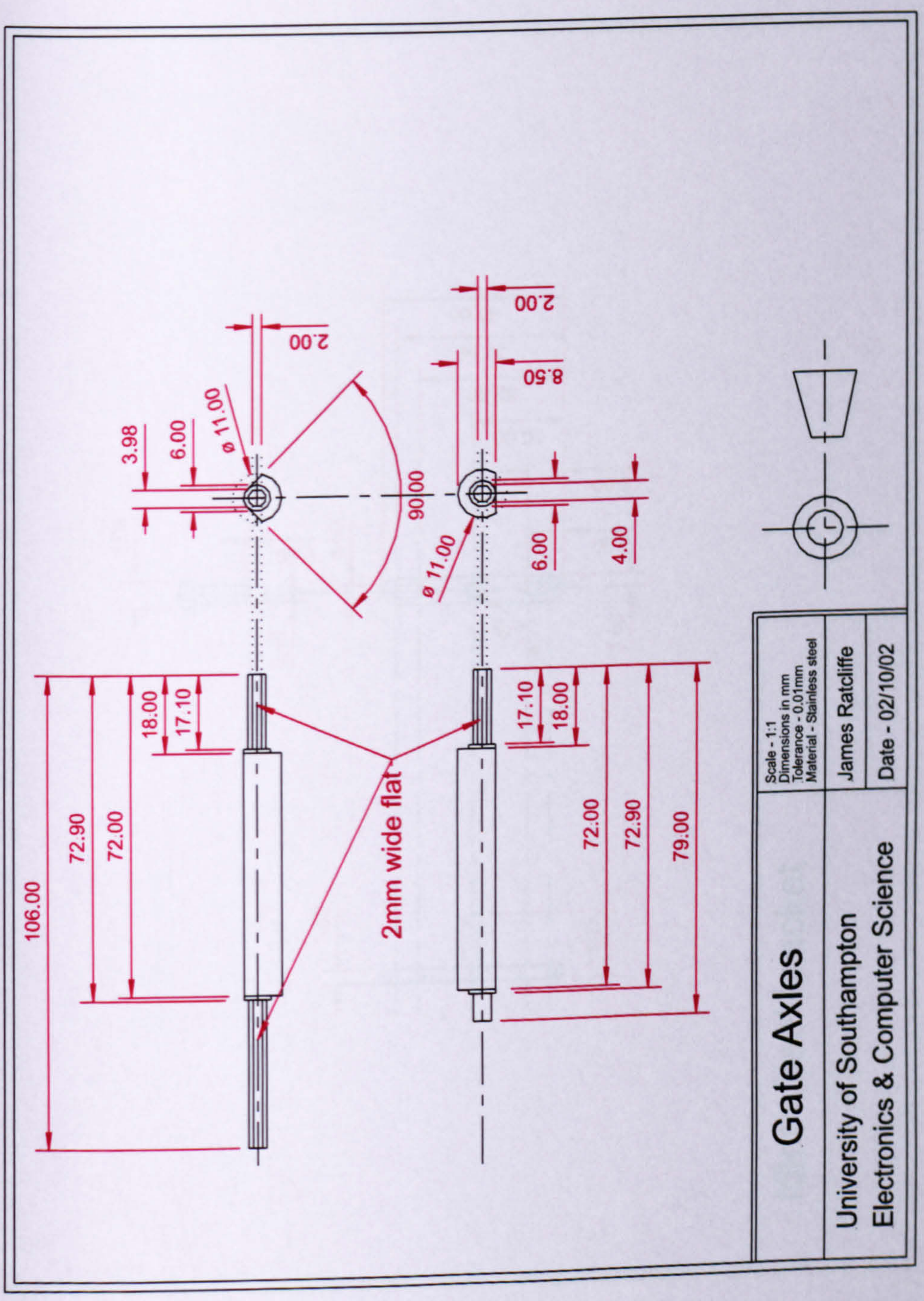


FIGURE C.37: Dispenser gate axles

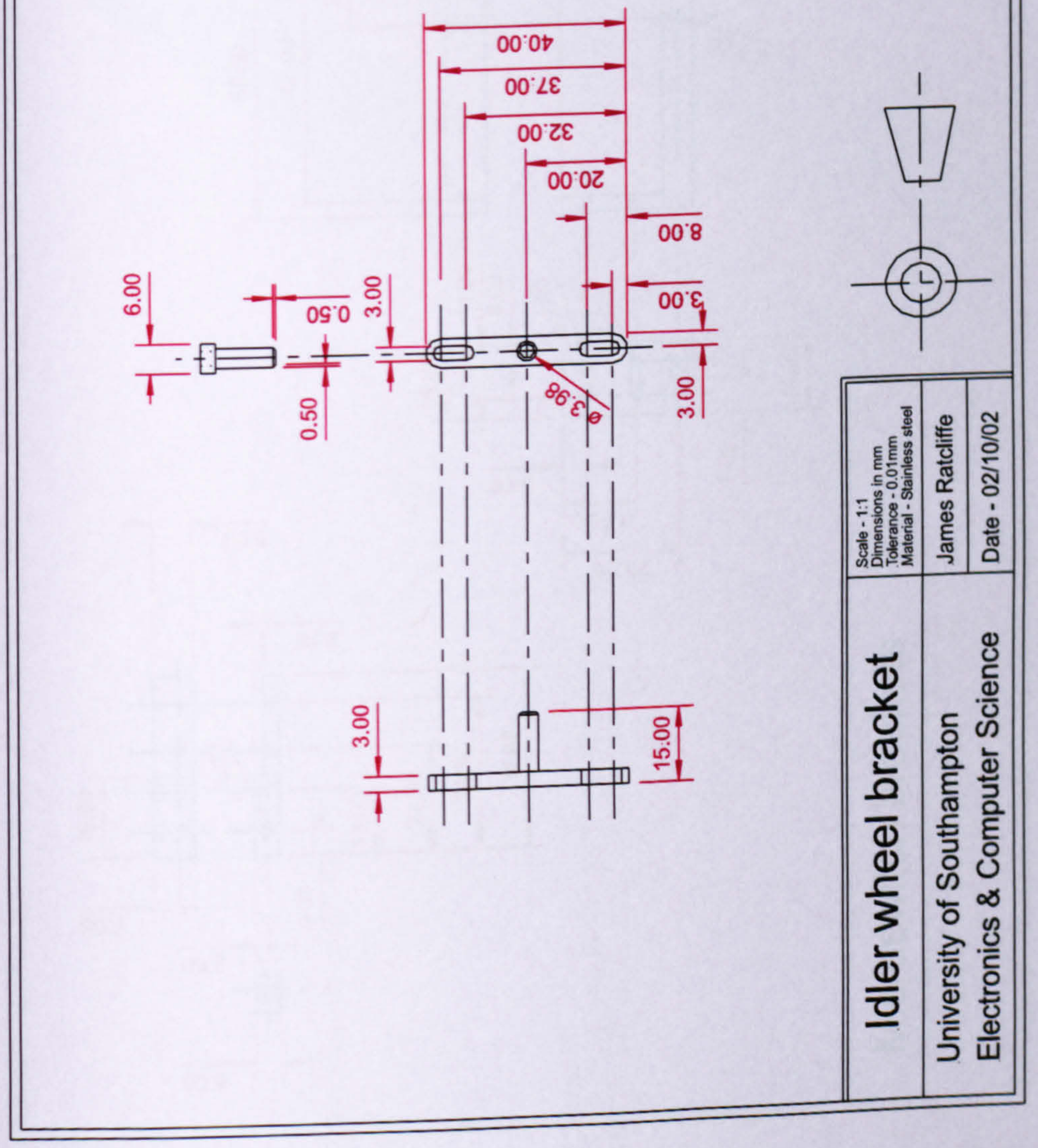


FIGURE C.38: Dispenser chain idler wheel bracket

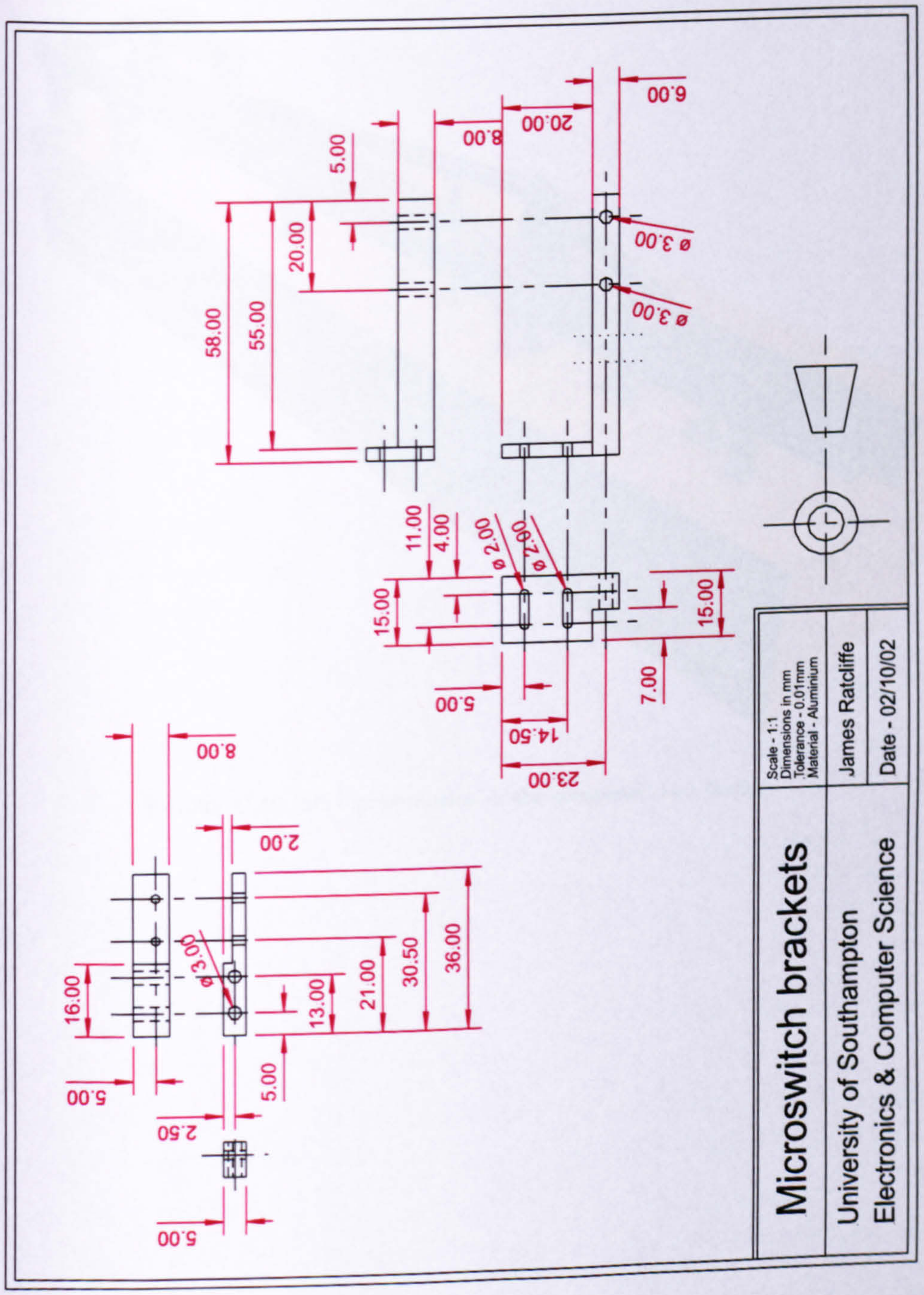


FIGURE C.39: Dispenser micro-switch brackets

C.4 Complete Test Facility

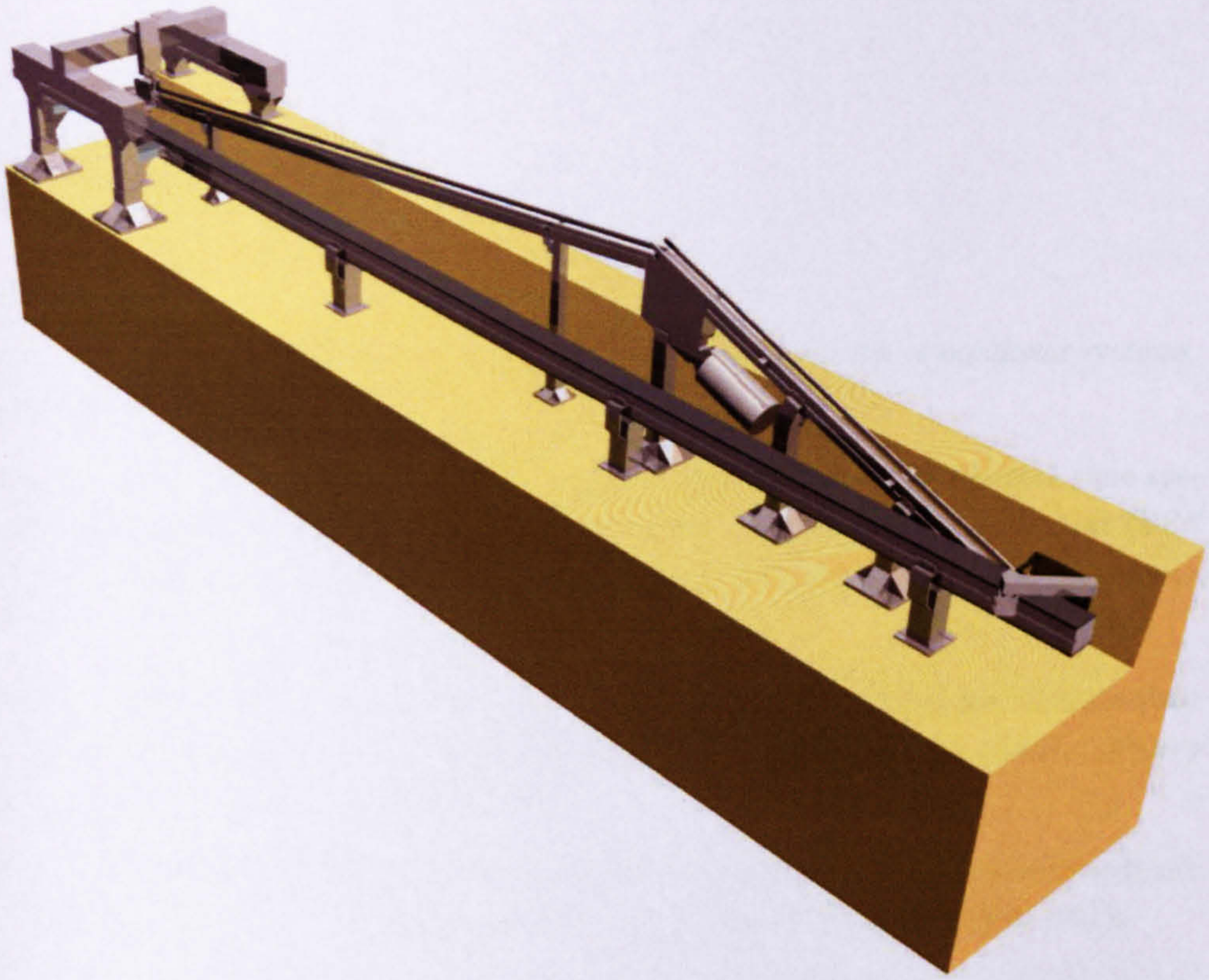


FIGURE C.40: 3D representation of the completed test facility

Bibliography

H. Ahn, C. Choi, and K. Kim. Iterative learning control for a class of nonlinear systems. *Automatica*, 29(6):1575–1578, 1993.

N. Amann, D.H. Owens, and E. Rogers. Iterative learning control for discrete time systems using optimal feedback and feedforward actions. In *Proceedings of the 34th IEEE Conference on Decision and Control, New Orleans, LA*, pages 1696–1701, December 1995.

N. Amann, D.H. Owens, and E. Rogers. Iterative learning control for discrete-time systems with exponential rate of convergence. *IEE Proceedings on Control Theory Applications*, 143(2):217–224, March 1996a.

N. Amann, D.H. Owens, and E. Rogers. Iterative learning control using optimal feedback and feedforward actions. *International Journal of Control*, 65(2):277–293, 1996b.

N. Amann, D.H. Owens, and E. Rogers. Predictive optimal iterative learning control. *International Journal of Control*, 69(2):203–226, 1998.

S. Arimoto, S. Kawamura, and F. Miyazaki. Bettering operation of dynamic systems by learning : A new control theory for servomechanism or mechatronic systems. In *Proceedings of 23rd IEEE Conference on Decision and Control, Las Vegas, NV*, pages 1064–1069, December 1984a.

S. Arimoto, S. Kawamura, and F. Miyazaki. Bettering operation of robots by learning. *Journal of Robotic Systems*, 1:123–140, 1984b.

S. Arimoto, S. Kawamura, and F. Miyazaki. Can mechanical robots learn by themselves? In *Robotics Reasearch: The Second International Symposium*, pages 127–134, 1985a.

S. Arimoto, S. Kawamura, F. Miyazaki, and S. Tamaki. Learning control theory for dynamical systems. In *Proceedings of the 24th IEEE Conference on Decision and Control, Ft. Lauderdale, FL*, pages 1375–1380, December 1985b.

S. Arimoto and T. Naniwa. Equivalence relations between learnability, output-dissipativity and strict positive realness. *International Journal of Control*, 73(10): 824–831, 2000.

A.D. Barton. *Control of high speed chain conveyor systems.* PhD thesis, Department of Electrical Engineering, University of Southampton, UK, 1999.

A.D. Barton and P.L. Lewin. Experimental comparison of the performance of different chain conveyor controllers. *Proceedings of the Institution of Mechanical Engineers*, 214 (1):361–369, 2000.

A.D. Barton, P.L. Lewin, and D.J. Brown. Practical implementation of a real-time iterative learning position controller. *International Journal of Control*, 73(10):992–999, 2000.

Z. Bien and K.M. Huh. Higher-order iterative learning control algorithm. *IEE Proceedings — Part D*, 136(3):105–112, May 1989.

P. Bondi, G. Casalino, and L. Gambardella. On the iterative learning control theory for robotic manipulators. *IEEE Journal of Robotics and Automation*, 4(1):14–22, February 1988.

G.M. Bone. A novel learning control formulation of generalized predictive control. *Automatica*, 31(10):1483–1487, 1995.

G. Casalino and G. Bartolini. A learning procedure for the control of movements of robotic manipulators. In *IASTED Symposium on Robotics and Automation, Amsterdam, The Netherlands*, pages 108–111, 1984.

H-F. Chen and H-T. Fang. Output tracking for nonlinear stochastic systems by iterative learning control. *IEEE Transactions on Automatic Control*, 49(4):583–588, 2004.

H-J. Chen and R.W. Longman. The importance of smooth updates in producing good error levels in repetitive control. In *Proceedings of the 38th IEEE Conference on Decision and Control, Phoenix, Arizona*, pages 258–263, December 1999.

K. Chen and R.W. Longman. Stability issues using FIR filtering in repetitive control. *Advances in the Astronautical Sciences*, 112(2):1321–1340, 2002.

Y. Chen, H. Dou, and K.K. Tan. Iterative learning control via weighted local-symmetrical-integration. *Asian Journal of Control*, 3(4):352–356, December 2001.

Y. Chen, Z. Gong, and C. Wen. Analysis of a high-order iterative learning control algorithm for uncertain nonlinear systems with state delay. *Automatica*, 34(3):345–353, 1998a.

Y. Chen and K.L. Moore. On D^{α}-type iterative learning control. In *Proceedings of the 40th IEEE Conference on Decision and Control, Orlando, Florida*, pages 4451–4456, December 2001.

Y. Chen, M. Sun, B. Huang, and H. Dou. Robust higher order repetitive learning control algorithm for tracking control of delayed repetitive systems. In *Proceedings of the 31st IEEE Conference on Decision and Control, Tucson, Arizona*, pages 2504–2501, December 1992.

Y. Chen, C. Wen, Z. Gong, and M. Sun. An iterative learning controller with initial state learning. *IEEE Transactions on Automatic Control*, 44(2):371–376, February 1999.

Y. Chen, C. Wen, and M. Sun. A robust high-order p-type iterative learning controller using current iteration tracking error. *International Journal of Control*, 68(2):331–342, 1997.

Y. Chen, C. Wen, J-X. Xu, and M. Sun. An initial state learning method for iterative learning control of uncertain time-varying systems. In *Proceedings of the 35th IEEE Conference on Decision and Control, Kobe, Japan*, pages 3996–4001, 1996a.

Y. Chen, C. Wen, J-X. Xu, and M. Sun. High-order iterative learning identification of projectile's aerodynamic drag coefficient curve from radar measured velocity data. *IEEE Transactions on Control Systems Technology*, 6(4):563–570, July 1998b.

Y. Chen, J-X. Xu, and T.H. Lee. Current iteration tracking error assisted iterative learning control of uncertain nonlinear discrete time systems. In *Proceedings of the 35th IEEE Conference on Decision and Control, Kobe, Japan*, pages 3038–3043, December 1996b.

Y. Chen, J-X. Xu, and T.H. Lee. An iterative learning controller using current iteration tracking error information and initial state learning. In *Proceedings of the 35th IEEE Conference on Decision and Control, Kobe, Japan*, pages 3064–3069, December 1996c.

Y. Chen, J-X. Xu, and C. Wen. Iterative learning-based extraction of aerobomb drag. *Journal of Spacecraft*, 35(2):237–240, 1998c.

K.K. Chew and M. Tomizuka. Digital control of repetitive errors in disk-drive systems. *IEEE Control Systems Magazine*, 10:16–20, January 1990.

C-J. Chien. A discrete iterative learning control of nonlinear time varying systems. In *Proceedings of the 35th IEEE Conference on Decision and Control, Kobe, Japan*, pages 3056–3061, December 1996.

C-J. Chien. A discrete iterative learning control for a class of nonlinear time-varying systems. *IEEE Transactions on Automatic Control*, 43(5):748–752, May 1998.

C-J. Chien and J-S. Liu. A p-type iterative learning controller for robust output tracking of nonlinear time-varying systems. *International Journal of Control*, 64(2):319–334, 1996.

J-W. Choi, H-G. Choi, K-S. Lee, and W-H Lee. Control of ethanol concentration in a fed-batch cultivation of acinctobactoer calcoacceticus RAG-1 using a feedback-assisted iterative learning algorithm. *Journal of Biotechnology*, 46:29–43, 1996.

J.Y. Choi and J.S. Lee. Adaptive iterative learning control of uncertain robotic systems. *IEE Proceedings on Control Theory Applications*, 147(2):217–223, March 2000.

J.J. Craig. Adaptive control of manipulators through repeated trials. In *Proceedings of the American Control Conference, San Diego, USA*, pages 1566–1573, 1984.

A. De Luca and S. Panzieri. An iterative scheme for learning gravity compensation in flexible robot arms. *Automatica*, 30(6):993–1002, 1994.

D. De Roover. Sysnthesis of a robust iterative learning controller using an H_∞ approach. In *Proceedings of the 35th IEEE Conference on Decision and Control, Kobe, Japan*, pages 3044–3049, December 1996.

D. De Roover and O.H. Bosgra. Dualization of the internal model principle in compensator and observer theory with application to repetitive and learning control. In *Proceedings of the American Control Conference, Albuquerque, New Mexico, USA*, pages 3902–3906, June 1997.

D. De Roover and O.H. Bosgra. Synthesis of robust multivariable iterative learning controllers with application to a wafer stage motion system. *International Journal of Control*, 73(10):968–979, 2000.

D. De Roover, O.H. Bosgra, and M. Steinbuch. Internal-model-based design of repetitive and iterative learning controllers for linear multivariable systems. *International Journal of Control*, 73(10):914–929, 2000.

T-Y. Doh, J-H. Moon, and M.J. Chung. An iterative learning control for uncertain systems using structured singular value. *Transactions of the ASME*, 121:660–666, December 1999.

H. Dou, K.K. Tan, T.H. Lee, and Z. Zhou. Iterative learning feedback control of human limbs via functional electrical stimulation. *Control Engineering Practice*, 7:315–325, 1999.

K. Dutton, S. Thompson, and B. Barraclough. *The art of control engineering.* Addison-Wesley, 1998. ISBN 0-201-17545-2.

H. Elci, R.W. Longman, M. Phan, J-N. Juang, and R. Ugoletti. Discrete frequency based learning control for precision motion control. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, San Antonio, TX*, volume 3, pages 2767–2773, 1994.

H. Elci, R.W. Longman, M.Q. Phan, J-N. Juang, and R. Ugoletti. Simple learning control made practical by zero-phase filtering: applications to robotics. *IEEE Transactions on Circuits and System—I:Fundamental Theory and Applications*, 49(6):753–767, June 2002.

G.F. Franklin, J.D. Powell, and A. Emani-Naeini. *Feedback Control of Dynamic Systems.* Addison-Wesley, 3 edition, 1994. ISBN 0-201-52747-2.

C.T. Freeman, J.J. Hätönen, P.L. Lewin, E. Rogers, and D.H. Owens. Experimental evaluation of a new repetitive control algorithm on a non-minimum phase spring-mass-damper system. In *Proceedings of the IFAC Workshop on Adaptation and Learning in Control and Signal Processing and the IFAC Workshop on Periodic Control Systems, Yokohama, Japan*, pages 681–686, 2004a.

C.T. Freeman, P.L. Lewin, and E. Rogers. Experimental evaluation of simple structure ilc algorithms for non-minimum phase plants. In *Proceedings of the 3rd IFAC Symposium on Mechatronic Systems, Sydney, Australia*, pages 205–210, 2004b.

C.T. Freeman, P.L. Lewin, and E. Rogers. Phase-lead based iterative learning control implemented experimentally on a non-minimum phase plant. In *Proceedings of the 3rd IFAC Symposium on Mechatronic Systems, Sydney, Australia*, pages 193–198, 2004c.

M. French, G. Munde, E. Rogers, and D.H. Owens. Recent developments in adaptive iterative learning control. In *Proceedings of the 38th IEEE Conference on Decision and Control, Phoenix, Arizona*, pages 264–269, December 1999.

J.A. Frueh and M.Q. Phan. Linear quadratic optimal learning control (LQL). In *Proceedings of the 37th IEEE Conference on Decision and Control, Tampa, Florida*, pages 678–683, December 1998.

J.A. Frueh and M.Q. Phan. Linear quadratic optimal learning control (LQL). *International Journal of Control*, 73(10):832–839, 2000.

K. Furuta and M. Yamakita. The design of learning control systems for multivariable systems. In *Proceedings of the IEEE International Symposium on Intelligent Control, Philadelphia, Pennsylvania*, pages 371–376, 1987.

K. Galkowski, E. Rogers, and D.H. Owens. New 2D models and a transition matrix for discrete linear repetitive processes. *International Journal of Control*, 72(15):1365–1380, 1999.

Z. Geng, R. Carroll, and J. Xie. Two-dimensional model and algorithm analysis for a class of iterative learning control systems. *International Journal of Control*, 52(4): 833–862, 1990.

J. Ghosh and B. Paden. Pseudo-inverse based iterative learning control for nonlinear plants with disturbances. In *Proceedings of the 38th IEEE Conference on Decision and Control, Phoenix, Arizona*, pages 5206–5212, December 1999.

J. Ghosh and B. Paden. Iterative learning control for nonlinear nonminimum phase plants. *Journal of Dynamic Systems, Measurement, and Control*, 123:21–30, 2001.

J. Ghosh and B. Paden. A pseudoinverse-based iterative learning control. *IEEE Transactions on Automatic Control*, 47(5):831–836, May 2002.

J.S. Glower. Adaption rates for repetitive control schemes. *International Journal of Adaptive Control and Signal Processing*, 11:533–547, 1997.

D. Gorinevsky. Distributed system loopshaping design of iterative control for batch processing. In *Proceedings of the 38th IEEE Conference on Decision and Control, Phoenix, Arizona*, pages 245–250, December 1999.

M. Grundelius. Iterative optimal control of liquid slosh in an industrial packaging machine. In *Proceedings of the 39th IEEE Conference on Decision and Control, Sydney, Australia*, pages 3427–3432, December 2000.

M. Grundelius and B. Bernhardsson. Constrained iterative learning control of liquid slosh in an industrial packaging machine. In *Proceedings of the 39th IEEE Conference on Decision and Control, Sydney, Australia*, pages 4544–4549, December 2000.

S. Gunnarsson and M. Norrlöf. Some aspects of an optimization approach to iterative learning control. In *Proceedings of the 38th IEEE Conference on Decision and Control, Phoenix, Arizona*, pages 1581–1586, December 1999.

S. Gunnarsson and M. Norrlöf. On the design of ILC algorithms using optimization. *Automatica*, 27:2011–2016, 2001.

S. Hara, T. Omata, and M. Nakano. Synthesis of repetitive control systems and its application. In *Proceedings of the 24th IEEE Conference on Decision and Control, Ft. Lauderdale, FL*, pages 1387–1392, December 1985.

T.J. Harte, J.J. Hätönen, and D.H. Owens. Robust monotone inverse type iterative learning control for discrete-time systems. *Proceedings of the IFAC Workshop on Adaptation and Learning in Control and Signal Processing and the IFAC Workshop on Periodic Control Systems, Yokohama, Japan*, 2004.

V. Hatzikos, J. Hätönen, and D.H. Owens. Genetic algorithms in norm-optimal linear and non-linear iterative learning control. *International Journal of Control*, 77(2):188–197, January 2004.

H. Havlicsek and A. Alleyne. Nonlinear control of an electrohydraulic injection molding machine via iterative adaptive learning. *IEEE/ASME Transactions on Mechatronics*, 4(3):312–323, September 1999.

L. Hideg. Design of a static neural element in an iterative learning control scheme. In *Proceedings of the 37th IEEE Conference on Decision and Control, Tampa, Florida*, pages 690–694, December 1998.

S. Hillenbrand and M. Pandit. A discrete time iterative learning control law with exponential rate of convergence. In *Proceedings of the 38th IEEE Conference on Decision and Control, Phoenix, Arizona*, pages 1575–1580, December 1999.

S. Hillenbrand and M. Pandit. An iterative learning controller with reduced sampling rate for plants with variations of initial states. *International Journal of Control*, 73 (10):882–889, 2000.

W. Hoffmann, K. Peterson, and A.G. Stefanopoulou. Iterative learning control for soft landing of electromechanical valve actuator in camless engines. *IEEE Transactions on Control Systems Technology*, 11(2):174–184, March 2003.

J. Hätönen, D. H. Owens, and K. L. Moore. An algebraic approach to iterative learning control. *International Journal of Control*, 77(1):45–54, 2004.

J. Hätönen and D.H. Owens. Convex modifications to an iterative learning control law. *Automatica*, 40:1213–1220, 2004.

J. Hätönen, D.H. Owens, and K.L. Moore. An algebraic approach to iterative learning control. *International Journal of Control*, 77(1):45–54, 2003a.

J.J. Hätönen, T.J. Harte, D.H. Owens, J.D. Ratcliffe, P.L. Lewin, and E. Rogers. A new robust iterative learning control algorithm for application on a gantry robot. In *Proceedings of the 6th IEEE International Conference on Emerging Technologies and Factory Automation, Lisbon, Portugal*, pages 305–312, 2003b.

Q. Hu, J-X. Xu, and T.H. Lee. Iterative learning control design for smith predictor. *Systems and Control Letters*, 44:201–210, 2001.

Y-C. Huang and R.W. Longman. The source of the often observed property of initial convergence followed by divergence in learning and repetitive control. *Advances in the Astronautical Sciences*, 90:555–572, 1996.

D.H. Hwang, Z. Bien, and S.R. Oh. Iterative learning control method for discrete-time dynamic systems. *IEE Proceedings — D*, 138(2):139–144, March 1991.

D.H. Hwang, B.K. Kim, and Z. Bien. Decentralized iterative learning control methods for large scale linear dynamic systems. *International Journal of Systems Science*, 24 (12):2239–2254, 1993.

T-J. Jang, H-S. Ahn, and C-H. Choi. Iterative learning control for discrete-time nonlinear systems. *International Journal of Systems Science*, 25(7):1179–1189, 1994.

T-J. Jang, C-H. Choi, and H-S. Ahn. Iterative learning control in feedback systems. *Automatica*, 31(2):243–248, 1995.

P. Jiang and R. Unbehauen. An iterative learning control scheme with deadzone. In *Proceedings of the 38th IEEE Conference on Decision and Control*, pages 3816–3817, December 1999.

P. Jiang and R. Unbehauen. Robot visual servoing with iterative learning control. *IEEE Transactions on Systems, Man, and Cybernetics — Part A: Systems and Humans*, 32 (2):281–286, March 2002.

R.P. Judd, L. Hideg, and R.P. Van Til. Coefficient test for stability analysis of iterative learning control systems. In *Proceedings of the 30th IEEE Conference on Decision and Control, Brighton, England*, pages 2942–2943, 1991.

S. Kawamura, F. Miyazaki, and S. Arimoto. Applications of learning method for dynamic control of robot manipulators. In *Proceedings of the 24th IEEE Conference on Decision and Control, Ft. Lauderdale, FL*, pages 1381–1386, December 1985.

S. Kawamura, F. Miyazaki, and S. Arimoto. Realization of robot motion based on a learning method. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(1):126–133, 1988.

D-I. Kim and S. Kim. An iterative learning control method with application for cnc machine tools. *IEEE Transactions on Industry Applications*, 32(1):66–72, January/February 1996.

W.C. Kim, I.S. Chin, K.S. Lee, and J. Choi. Analysis and reduced-order design of quadratic criterion-based iterative learning control using singular value decomposition. *Computers and Chemical Engineering*, 24:1815–1819, 2000.

T. Kuc, J.S. Lee, and K. Nam. An iterative learning control theory for a class of nonlinear dynamic systems. *Automatica*, 28(6):1215–1221, 1992.

T. Kuc, K. Nam, and J.S. Lee. An iterative learning control for robot manipulators. *IEEE Transactions on Robotics and Automation*, 7(6):835–841, December 1991.

J.E. Kurek and M.B. Zaremba. Iterative learning control synthesis based on 2-D system theory. *IEEE Transactions on Automatic Control*, 38(1):121–125, January 1993.

H-S. Lee and Z. Bien. Study on robustness of iterative learning control with non-zero initial error. *International Journal of Control*, 64(3):345–359, 1996.

J-J. Lee and J-W. Lee. Design of iterative learning controller with vcr servo system. *IEEE Transactions on Consumer Electronics*, 39(1):13–24, 1993.

J.H. Lee, K.S. Lee, and W.C. Kim. Model-based iterative learning control with a quadratic criterion for time-varying linear systems. *Automatica*, 36:641–657, 2000.

K.S. Lee, S.H. Bang, S. Yi, J.S. Son, and S.C. Yoon. Iterative learning control of heat-up phase for a batch polymerization reactor. *Journal of Process Control*, 6(4):255–262, 1996.

P.L.L. Lewin. Iterative learning control of repetitive processes. In *Proceedings of the 2nd International Conference on Climbing and Walking Robots, Portsmouth, UK*, pages 295–303, 1999.

C.J. Li and S.Y. Li. To improve workpiece roundness in precision diamond turning by in situ measurement and repetitive control. *Mechatronics*, 6(5):523–535, 1996.

Y-J. Liang and D.P. Looze. Performance and robustness issues in iterative learning control. In *Proceedings of the 32nd IEEE Conference on Decision and Control, San Antonio, Texas*, pages 1990–1995, December 1993.

R.W. Longman. Iterative learning control and repetitive control for engineering practice. *International Journal of Control*, 73(10):930–954, 2000.

R.W. Longman, R. Akogyeram, A. Hutton, and J-N. Juang. Trade-offs between feedback, feedforward, and repetitive control for systems subject to periodic disturbances. *Advances in the Astronautical Sciences*, 108:1281–1300, 2001.

D.G. Luenberger. An introduction to observers. *IEEE Transaction on Automatic Control*, 16(6):596–602, December 1971.

L.Y.X. Ma, T.S. Low, and S.K. Tso. Discrete iterative learning controller. *Electronics Letters*, 29(12):1046–1048, June 1993.

J.A.T. Machado and A.M.S.F. Galhano. Benchmarking computer systems for robot control. *IEEE Transactions on Education*, 38(3):205–210, August 1995.

O. Markusson, H. Hjalmarsson, and M. Norrlöf. Iterative learning control of nonlinear non-minimum phase systems and its application to system and model inversion. In *Proceedings on the 40th IEEE Conference on Decision and Control, Orlando, Florida*, pages 4481–4482, December 2001.

M. Mezghani, G. Roux, M. Cabassud, M.V. Le Lann, B. Dahhou, and G. Casamatta. Application of iterative learning control to an exothermic semibatch chemical reactor. *IEEE Transactions on Control Systems Technology*, 10(6):822–834, November 2002.

T. Mita and E. Kato. Iterative control and its application to motion control of robot arm — a direct approach to servo-problems. In *Proceedings of the 24th IEEE Conference on Decision and Control, Ft. Lauderdale, FL*, pages 1393–1398, December 1985.

J. Moon, M. Lee, and M. Chung. Track-following control for optical disk drives using an iterative learning scheme. *IEEE Transactions on Consumer Electronics*, 42(2): 192–198, May 1996.

J-H. Moon and M.J. Chung. A robust approach to iterative learning control design for uncertain systems. *Automatica*, 34(8):1001–1004, 1998.

K.L. Moore. Multi-loop control approach to designing iterative learning controllers. In *Proceedings of the 37th IEEE Conference on Decision and Control, Tampa, Florida*, pages 666–671, December 1998.

K.L. Moore. An iterative learning control algorithm for systems with measurement noise. In *Proceedings of the 38th IEEE Conference on Decision and Control, Phoenix, Arizona*, pages 270–275, December 1999.

K.L. Moore. A non-standard iterative learning control approach to tracking periodic signals in discrete-time non-linear systems. *International Journal of Control*, 73(10): 955–967, 2000.

K.L. Moore and Y. Chen. A separative high-order framework for monotonic convergent iterative learning controller design. In *Proceedings of the American Control Conference, Denver, Colorado, USA*, pages 3644–3649, June 2003.

K.L. Moore and J-X. Xu. Speccial issue on iterative learning control. *International Journal of Control*, 73(10):819–823, 2000.

M. Norrlöf. Comparative study on first and second order ilc — frequency domain analysis and experiments. In *Proceedings of the 39th IEEE Conference on Decision and Control, Sydney, Australia*, pages 3415–3420, December 2000.

M. Norrlöf. An adaptive iterative learning control algorithm with experiments on an industrial robot. *IEEE Transactions on Robotics and Automation*, 18(2):245–251, April 2002.

M. Norrlöf and S. Gunnarsson. A frequency domain analysis of a second order iterative learning control algorithm. In *Proceedings of the 38th IEEE Conference on Decision and Control, Phoenix, Arizona*, pages 1578–1592, December 1999.

M. Norrlöf and S. Gunnarsson. Disturbance aspects of iterative learning control. *Engineering Applications of Artificial Intelligence*, 14:87–94, 2001.

M. Norrlöf and S. Gunnarsson. Experimental comparison of some classical iterative learning control algorithms. *IEEE Transactions on Robotics and Automation*, 18(4): 636–641, August 2002a.

M. Norrlöf and S. Gunnarsson. Time and frequency domain convergence properties in iterative learning control. *International Journal of Control*, 75(14):1114–1126, 2002b.

S. Oh, Z. Bien, and I.H. Suh. An iterative learning control method with application for the robot manipulator. *IEEE Journal of Robotics and Automation*, 4(5):508–514, October 1988.

Y. Onuki and H. Ishioka. Compensation for repeatable tracking errors in hard drives using discrete-time repetitive controllers. *IEEE/ASME Transactions on Mechatronics*, 6(2):132–136, 2001.

D.H. Owens. Iterative learning control - convergence using high gain feedback. In *Proceedings of the 31st IEEE Conference on Decision and Control, Tucson, Arizona*, pages 2545–2546, December 1992.

D.H. Owens, N. Amann, E. Rogers, and M. French. Analysis of linear iterative learning control schemes - a 2D systems/repetitive processes approach. *Multidimensional Systems and Signal Processing*, 11(1):125–177, 2000.

D.H. Owens and G. Munde. Error convergence in an adaptive learning controller. *International Journal of Control*, 73(10):851–875, 2000.

M. Pandit and K-H. Buchheit. Optimizing iterative learning control of cyclic production processes with application to extruders. *IEEE Transactions on Control Systems Technology*, 7(3):382–390, May 1999.

K-H. Park and Z. Bien. A generalized iterative learning controller against initial state error. *International Journal of Control*, 73(10):871–881, 2000.

K.H. Park, Z. Bien, and D.H. Hwang. Design of an iterative learning controller for a class of linear dynamic systems with time delay. *IEE Proceedings - Control Theory Applications*, 145(6):507–512, November 1998.

M.Q. Phan and J.A. Frueh. Learning control for trajectory tracking using basis functions. In *Proceedings of the 35th IEEE Conference on Decision and Control, Kobe, Japan*, pages 2490–2492, December 1996.

M.Q. Phan and J.A. Frueh. Model reference adaptive learning control with basis functions. In *Proceedings of the 38th IEEE Conference on Decision and Control, Phoenix, Arizona*, pages 251–257, December 1999.

A.N. Poo and Y.X. Ma. Application of discrete learning control to a robotic manipulator. *Robotics and Computer Integrated Manufacturing*, 12(1):55–64, 1995.

W. Qian, S.K. Panda, and J-X. Xu. Torque ripple minimisation in pm synchronous motors using iterative learning control. *IEEE Transactions on Power Electronics*, 19 (2):272–279, 2004.

J.D. Ratcliffe, T.J. Harte, J.J. Hätönen, P.L. Lewin, E. Rogers, and D.H. Owens. Practical implementation of a model inverse iterative learning controller. In *Proceedings of the IFAC Workshop on Adaptation and Learning in Control and Signal Processing and the IFAC Workshop on Periodic Control Systems, Yokohama, Japan*, pages 687–692, 2004a.

J.D. Ratcliffe, P.L. Lewin, E. Rogers, J.J. Hätönen, T.J. Harte, and D.H. Owens. Measuring the performance of iterative learning control systems. In *Accepted for the joint 2005 International Symposium on Intelligent Control and 13th Mediterranean Conference on Control and Automation*, 2005a.

J.D. Ratcliffe, E. Rogers, T.J. Harte, J.J. Hätönen, P.L. Lewin, and D.H. Owens. Experimental testing of a new iterative learning control algorithm. In *Proceedings of UKACC Control 2004, Bath, UK*, September 2004b.

J.D. Ratcliffe, L. van Duinkerken, P.L. Lewin, E. Rogers, J.J. Hätönen, T.J. Harte, and D.H. Owens. Fast norm-optimal iterative learning control for industrial applications. In *Accepted for the 24th American Control Conference, Portland, USA*, 2005b.

I. Rotariu, R. Ellenbroek, and M. Steinbuch. Time-frequency analysis of a motion system with learning control. In *Proceedings of the American Control Conference, Denver, Colorado, USA*, pages 3650–3654, 2003.

S.S. Saab. Stochastic p-type/d-type iterative learning control algorithms. *International Journal of Control*, 76(2):139–148, 2003.

W.G. Seo, B.H. Park, and J.S. Lee. Intelligent learning control for a class of nonlinear dynamic systems. *IEE Proceedings on Control Theory Applications*, 146(2):165–170, March 1999.

C.E. Shannon. Communication in the presence of noise. *Proceedings of the IEEE*, 86(2): 447–457, February 1998. Reprinted from the Proceedings of the IRE, vol. 37, no. 1, pp 10–21, Jan. 1949.

T. Sogo and N. Adachi. Convergence rates and robustness of iterative learning control. In *Proceedings of the 35th IEEE Conference on Decision and Control, Kobe, Japan*, pages 3050–3055, December 1996.

T. Sogo, K. Kinoshita, and N. Adachi. Iterative learning control using adjoint systems for nonlinear non-minimum phase systems. In *Proceedings of the 39th IEEE Conference on Decision and Control, Sydney, Australia*, pages 3445–3446, December 2000.

T. Songchon and R.W. Longman. On the waterbed effect in repetitive control using zero-phase filtering. *Advances in the Astronautical Sciences*, 108:1321–1340, 2001.

M. Steinbuch and R. van de Molengraft. Iterative learning control of industrial motion systems. In *Proceedings of the 1st IFAC Conference on Mechatronic Systems, Darmstadt, Germany*, pages 967–972, 2000.

T. Sugie and T. Ono. On an iterative learning control law for dynamical systems. In *IFAC 10th Triennial World Congress, Munich*, pages 339–344, 1987.

T. Sugie and T. Ono. An iterative learning control law for dynamical systems. *Automatica*, 27(4):729–732, 1991.

M. Sun and D. Wang. Anticipatory iterative learning control for nonlinear systems with arbitrary relative degree. *IEEE Transactions on Automatic Control*, 46(5):783–788, May 2001a.

M. Sun and D. Wang. Robust discrete-time iterative learning control: initial shift problem. In *Proceedings of the 40th IEEE Conference on Decision and Control, Orlando, Florida*, pages 1211–1216, December 2001b.

M. Sun and D. Wang. Iterative learning control with initial rectifying action. *Automatica*, 38:1177–1182, 2002.

M. Sun and D. Wang. Initial shift issues on discrete-time iterative learning control with system relative degree. *IEEE Transactions on Automatic Control*, 48(1):144–148, January 2003.

A. Tayebi. Adaptive iterative learning control for robot manipulators. *Automatica*, 40: 1195–1203, 2004.

A. Tayebi and M.B. Zaremba. Robust iterative learning control design is straightforward for uncertain lti systems satisfying the robust performance condition. *IEEE Transactions on Automatic Control*, 48(1):101–106, January 2003.

M. Togai and O. Yamano. Analysis and design of an optimal learning control scheme for industrial robots: a discrete system approach. In *Proceedings of the 24th IEEE Conference on Decision and Control, Ft. Lauderdale, FL*, pages 1399–1404, December 1985.

R.L. Tousain, J.-C. Boissy, M.L. Norg, M. Steinbuch, and O.H. Bosgra. Suppressing non-periodically repeating disturbances in mechanical servo systems. In *Proceedings of the 37th IEEE Conference on Decision and Control, Tampa, Florida, USA*, pages 2541–2542, 1998.

M. Uchyama. Formulation of high-speed motion pattern of a mechanical arm by trial (in japanese). *Transactions of the Society for Instrumentation and Control Engineers*, 14: 706–712, 1978.

M. Unser. Sampling—50 years after Shannon. *Proceedings of the IEEE*, 88(4):569–587, April 2000.

D. Wang. A simple iterative learning controller for manipulators with flexible joints. *Automatica*, 31(9):1341–1344, 1995.

D. Wang. Convergence and robustness of discrete time nonlinear systems with iterative learning control. *Automatica*, 34(11):1445–1448, 1998.

D. Wang. On anticipatory iterative learning control designs for continuous time nonlinear dynamic systems. In *Proceedings of the 38th IEEE Conference on Decision and Control, Phoenix, Arizona*, pages 1605–1610, December 1999.

D. Wang. On d-type and p-type ILC designs and anticipatory approach. *International Journal of Control*, 73(10):890–901, 2000.

D. Wang and C.C. Cheah. An iterative learning scheme for impedance control of robot manipulators. *International Journal of Robotics Research*, 17(10):1091–1104, October 1998.

Y. Wang and R.W. Longman. Use of non-causal digital signal processing in learning and repetitive control. *Advances in the astronautical sciences*, 90(1):649–668, 1996.

J. Xu and Y. Tan. On the convergence speed of a class of higher-order ilc schemes. In *Proceedings of the 40th IEEE Conference on Decision and Control, Orlando, Florida*, pages 4932–4937, December 2001.

J-X. Xu, Y. Chen, T.H. Lee, and S. Yamamoto. Terminal iterative learning control with an application to RTPCVD thickness control. *Automatica*, 35:1535–1542, 1999.

J-X. Xu, Q. Hu, T.H. Lee, and S. Yamamoto. Iterative learning control with smith time delay compensator for batch processes. *Journal of Process Control*, 11:321–328, 2001.

J-X. Xu and Q. Ji. New ilc algorithms with improved convergence for a class of non-affine functions. In *Proceedings of the 37th IEEE Conference on Decision and Control, Tampa, Florida*, pages 660–665, December 1998.

J-X. Xu and Y. Tan. Robust optimal design and convergence properties analysis of iterative learning control approaches. *Automatica*, 38:1867–1880, 2002.

J-X. Xu and B. Viswanathan. Adaptive robust iterative learning control with dead zone scheme. *Automatica*, 36:91–99, 2000.

J-X. Xu, B. Viswanathan, and Z. Qu. Robust learning control for robotic manipulators with an extension to a class of non-linear systems. *International Journal of Control*, 73(10):858–870, 2000.

D.R. Yang, K.S. Lee, H.J. Ahn, and J.H. Lee. Experimental application of a quadratic optimal iterative learning control method for control of wafer temperature uniformity in rapid thermal processing. *IEEE Transactions on Semiconductor Manufacturing*, 16 (1):36–44, February 2003.

Y. Yongqiang and D. Wang. Better robot tracking accuracy with phase lead compensated ilc. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4380–4385, 2003.

D. Zheng and A. Alleyne. Stability of a novel iterative learning control scheme with adaptive filtering. In *Proceedings of the American Control Conference, Denver, Colorado, USA*, pages 4512–4517, June 2003.

Y. Zhou, M. Steinbuch, and G. Leenknegt. A cost-effective scheme to improve radial tracking performance for high speed optical disk drives. *Journal of Vibration and Control*, 10:795–810, 2004.

J.G. Ziegler and N.B. Nichols. Optimum settings for automatic controllers. *Transactions of the ASME*, 64:759–768, 1942.

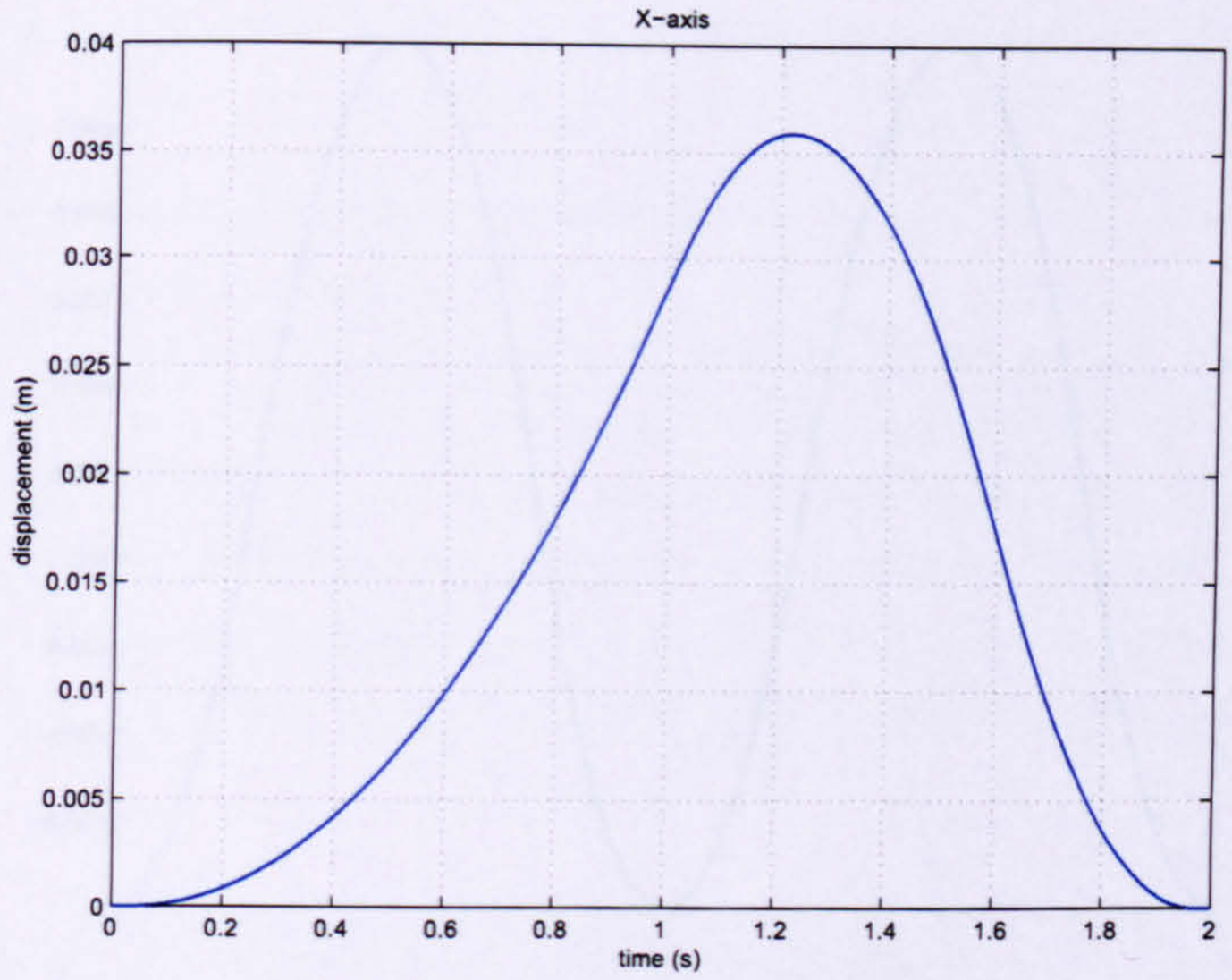


FIGURE 3.26: X-axis reference trajectory (30upm)

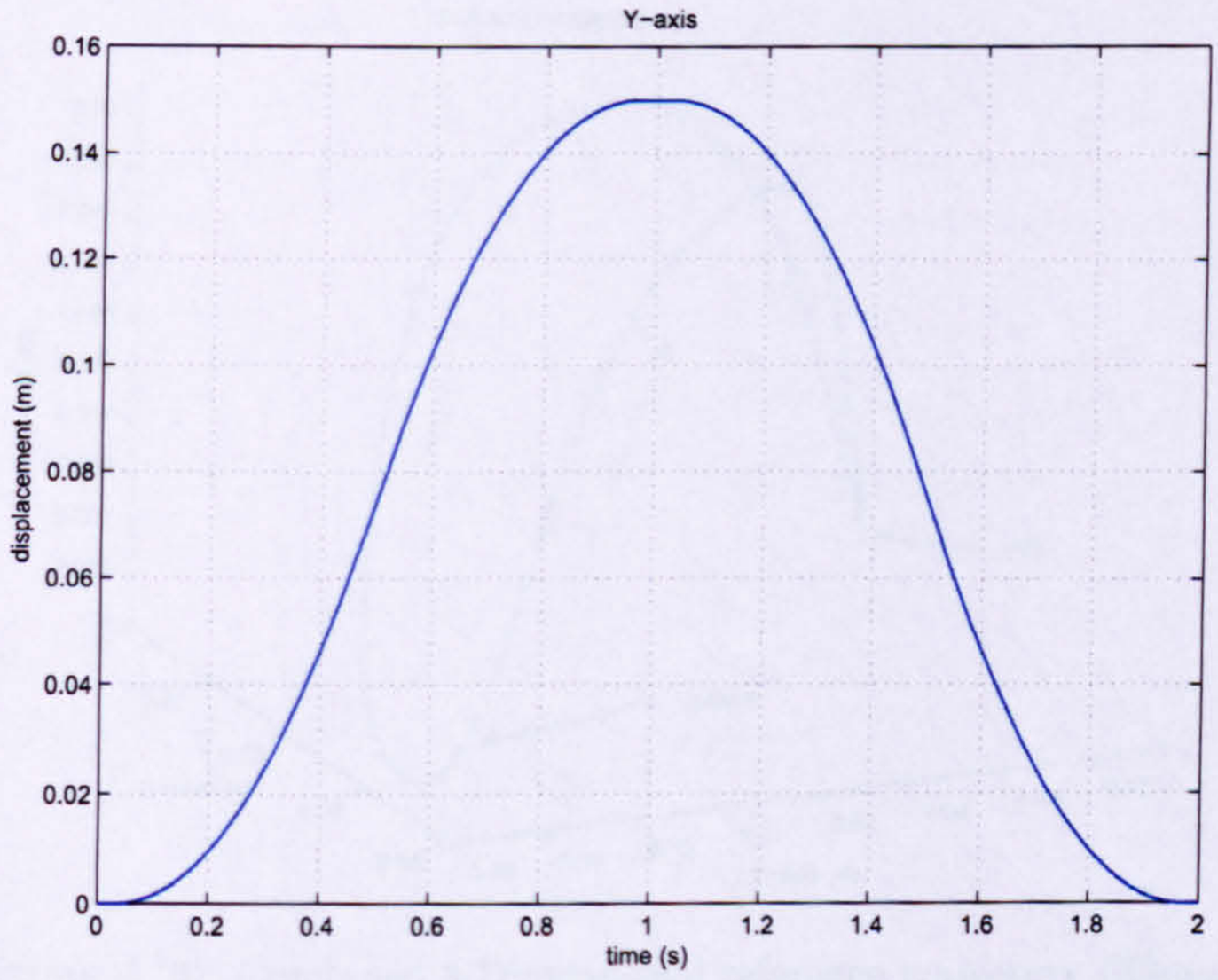


FIGURE 3.27: Y-axis reference trajectory (30upm)

on the gantry robot tends to suggest that if an algorithm is inherently unstable, the instability will be identifiable within 100 to 200 iterations, sometimes as few as 3 or 4 iterations. The long-term stability test is therefore defined as a batch of 5000 consecutive iterations. This test does not guarantee algorithm stability for an infinite number of iterations. However, in comparison to the few hundred iterations which can be achieved by inherently unstable algorithms, the 5000 iteration test is a good indicator of long term-stability. The high order models are used for this test.