University of Southern Queensland
Faculty of Engineering and Surveying

Position Control Using A Digital Servo System

A dissertation submitted by

David Scott Salomon

in fulfilment of the requirements of

Courses ENG4111 and 4112 Research Project

towards the degree of

Bachelor of Engineering (Mechatronic)

Submitted: October, 2008

Abstract

This report details an investigation into the performance of some conventional digital control strategies when applied to the problem of position control. The investigation is based on an existing DC motor servo system.

A mathematical model of the system has been developed and used for designing various forms of feedback controller. The system has been simulated with each feedback scheme in place and, in the case of the PID controller, physical experiments were performed.

State-variable feedback was found to be the most successful and most versatile. While the feedback schemes were successful in providing closed-loop control of the system with desirable response characteristics, the experiments showed that the system lacked rigidity against disturbing forces. This is a problem that must be addressed in most industrial applications of position control and further review of the literature showed that it is a problem inherent to controller designs based on analysing the system only in its linear region of operation. From observations made during this research, it seems that the controller design techniques established on linear control theory, which are presented here, are not well suited to position control, despite the fact that many authors treat them as though they are.

University of Southern Queensland

Faculty of Engineering and Surveying

ENG4111 & ENG4112 *Research Project*

Limitations of Use

The Council of the University of Southern Queensland, its Faculty of Engineering and Surveying, and the staff of the University of Southern Queensland, do not accept any responsibility for the truth, accuracy or completeness of material contained within or associated with this dissertation.

Persons using all or any part of this material do so at their own risk, and not at the risk of the Council of the University of Southern Queensland, its Faculty of Engineering and Surveying or the staff of the University of Southern Queensland.

This dissertation reports an educational exercise and has no purpose or validity beyond this exercise. The sole purpose of the course pair entitled "Research Project" is to contribute to the overall education within the student's chosen degree program. This document, the associated hardware, software, drawings, and other material set out in the associated appendices should not be used for any other purpose: if they are so used, it is entirely at the risk of the user.

Prof Frank Bullen

Dean

Faculty of Engineering and Surveying

Certification

I certify that the ideas, designs and experimental work, results, analyses and conclusions set out in this dissertation are entirely my own effort, except where otherwise indicated and acknowledged.
I further certify that the work is original and has not been previously submitted for assessment in any other course or institution, except where specifically stated.

David Scott Salomon
Student Number: 0050041363

_____

Signature

_____

Date

Acknowledgments

Thankyou to Dr Paul Wen at USQ for sharing your expertise and your guidance towards this research project.

I would also like to thank my family for their continuous support and patience during the time spent on the project.

Contents

List of Figures

1 Introduction

The core of this project is the design of a digital controller for controlling the rotational position output of an electromechanical servo system, which is a common building block in many position control systems. The focus is on developing appropriate control algorithms using classical and modern control system theory. These will then be evaluated using computer simulation and, where possible, by physical testing on the controller.

Position, in the physical sense, refers to the location of an object in space, measured by distances or a combination of distances and angles. These are fundamental physical quantities, measured in units such as meters, microns, degrees and radians. Position affects every physical activity, which most of us take for granted, from eating and walking to the work that we carry out in our occupations. The brain, equipped with the body's vast array of sensors, has an unsurpassed ability to control position, as our muscles move our limbs through all kinds of trajectories. However, object placement or motion tasks often have very high precision or speed requirements and/or very large resisting forces, which exceed the abilities of the human body. They can also involve hazardous conditions or remote environments. This is where the use of machinery becomes necessary. The use of automated machinery may have added benefits in performing repetitive work with more consistency and efficiency than can be expected from a person performing the work manually.

(a) (b)

(c) (d)

Fig. 1.1: Examples of modern systems using digital position control: (a) BMI Airbus A321-200 flight control surfaces (Pingstone 2002), (b) CNC machining (*CNC Machining Services* 2008), (c) Arm holding the read/write head of a computer hard disk (Bowey 2008) and (d) KUKA robot cutting metal parts at company Krupp in Germany (KUKA Roboter GmbH, Bachmann 2003)

Modern systems, such as aircraft flight controls, CNC machines and computer hard disk drives make use of digital systems to control position automatically. The same principles can be used to control the position of an object in each of its degrees of freedom, in order to generate motion along a three dimensional path. This is precisely what an industrial robot is designed to do.

Throughout the history spanning the invention of these and many more control systems, several design techniques have evolved, with some better suited to position control than others. The aim is to investigate the design processes involved and the performance of the resulting controllers.

1.1 Dissertation Outline

Although the engineering discipline concerning the formal analysis of control systems only began in the 19th century, the theory is extensive and largely mathematical. It is also very sequential, with a few fundamental concepts being continually built upon until analysis and design techniques are developed. Rather than present the relevant theory and literature in one section it has been broken into background sections included in each of the chapters. The design steps are then applied to the problem in a straightforward manner. A brief overview of each of the chapters is given in the list below.

- Chapter 1 – Introduction
- Chapter 2 – Modelling the System
  Contains background information on the main components of an existing position control servomechanism and their functioning. A pulse transfer function modelling the digitally controlled system is developed.
- Chapter 3 – PID Control

A PID controller algorithm is developed for the system through simulation, to give optimal response of the position output to a step input. The results are compared to those obtained experimentally.

- Chapter 4 – State Variable Feedback Control
  The system is analysed in a different manner and state variable feedback gains are determined. A state observer is formed for the case in which sensors are absent or certain signals are unattainable.

- Chapter 5 – Conclusions
  The various control strategies are compared in the context of position control.

1.2 The Problem and Research Objectives

The problem of controlling a servo system using a digital controller is by no means a new one, however, there are various forms of feedback currently being used, each with its own benefits and disadvantages and different hardware and software requirements. There are significant differences are in software, i.e. the algorithms central to controlling the system and also in the general functioning of peripheral devices used to sample data and apply driving signals.

The aims of this project are to model an existing position control system and to investigate the performance of various control strategies and corresponding algorithms. Also existing methods will be investigated, which are used to cope with non-ideal conditions which render the theory, and thus the design techniques less accurate. These conditions are namely nonlinearities in the physical behaviour of the system and sensitivity to other modelling inaccuracies or variations in component properties with time.

The techniques that will be employed build upon analysis in the *frequency domain* until Chapter 5, where the techniques come from a more recent body of control

systems theory known as *state-space* methods. State-space methods began at the same roots as frequency domain techniques but remain in the *time domain* and are regarded to give control strategies which are superior in many respects.

1.3 Equipment

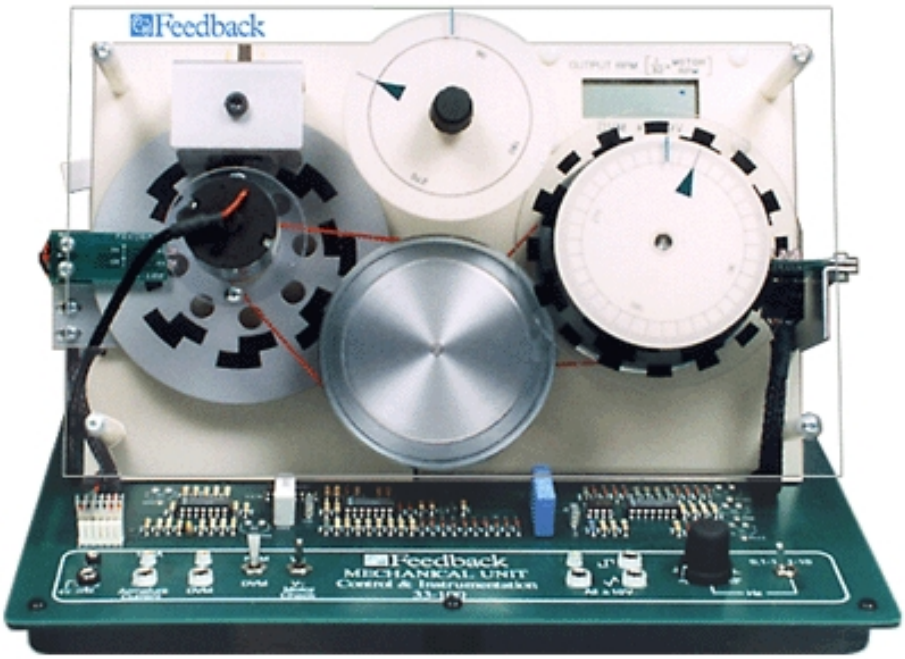
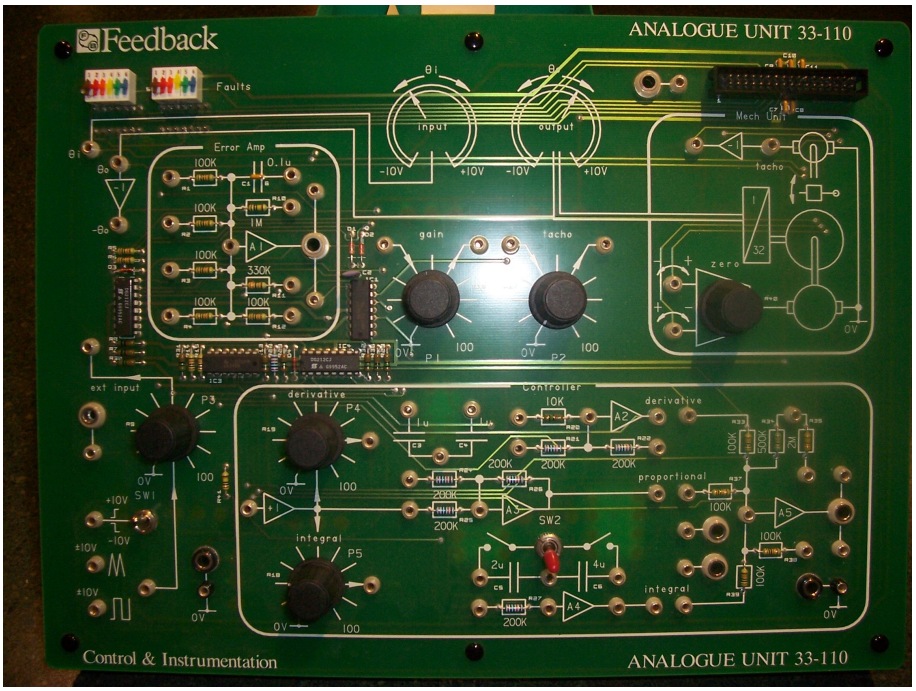


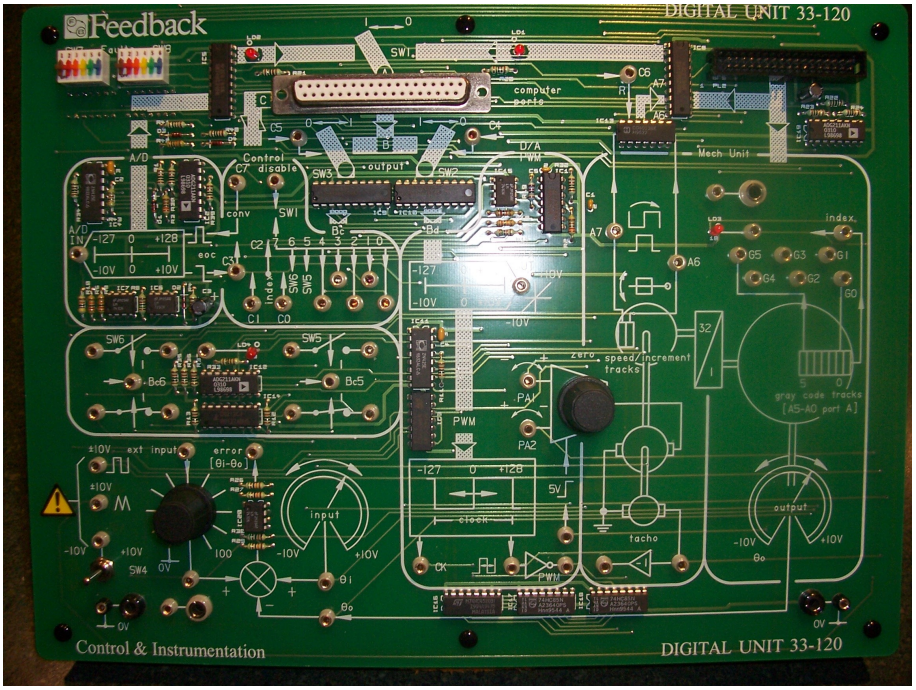
Fig. 1.2: Feedback Instruments *Servo Fundamentals Trainer* mechanical unit (Feedback Instruments Limited 1999)

The equipment used throughout the project is educational equipment made by Feedback Instruments Limited. The servomechanism, shown in Fig. 1.2, is powered by an external power supply and consists of a DC motor unit and driving circuitry and a belt driven output shaft. Feedback is via a tachogenerator mounted on the motor shaft or optical position and velocity encoders on both shafts and a reference input can be applied by turning the input potentiometer shaft. It is a rather simple system, but has many of the aspects of a practical position control system, such as

inertial loads and speed-dependent loads. Load can be increased by lowering the magnetic brake over the disc on the input shaft to induce speed-dependent Eddie currents in the disc which react with the magnetic field to oppose the motion. The analogue control board, shown in Fig. 1.3, has been used in order to become familiar with the mechanical unit and also gain a physical feel for the concepts of analogue compensation which are extended in Chapter 3 to the implementation of digital compensation.



(a)



(b)

Fig. 1.3: Feedback Instruments *Servo Fundamentals Trainer* (a) Analogue Unit and (b) Digital Unit

In discussing the choice between the use of a personal computer (PC) or single-chip microcontroller as the digital controller, Billingsley (1989, p. 42) states that " [the 'recipe book' designer] will deduce that the easiest answer is an 'intelligent peripheral'. This has not answered the problem at all; it has merely shifted the task from one of interfacing the master computer to that of interfacing the microcomputer that makes the peripheral intelligent." For simpler systems, like the one under consideration, an embedded system with a microcontroller dedicated only to the task of controlling the servo, is a simple, robust and cost-effective solution, and a PC is overkill. However, as a product which is more interactive with the user and doesn't require microcontroller programming utilities, Feedback Instruments have provided a digital control unit that is interfaced to a PC via a Universal Serial Bus (USB) interface. Feedback can be manipulated using the tutorial software, which runs in a web browser, however it is somewhat limited in this regard.

1.4 Background

In the past, devices which have aided people in positioning applications have made use of force or power amplifying technologies such as gear trains and hydraulics, with the user providing an input until the object reaches the desired position. For example, a machinist using a lathe can move the carriage and cross slide holding the tool with high precision by turning the hand wheels. The feed mechanism allows the operator to move the tool with less force and the speed of the tool is reduced, making its movement less sensitive to that of the operator's hand.

Prior to the advent of control systems theory, there had also been some inventions including the element of feedback control, such as the float-valve used by the Ancient Greeks in water clocks. (Nise 2004, p. 4) The float-valve controlled the depth of water in a vessel in order to maintain a steady flow of water from a hole near the base, which was then used to keep time. To put it another way, it controlled the

vertical position of the surface of the water. However, it wasn't until James Watt's invention of the centrifugal governor, used to regulate the speed of steam engines, that great interest was sparked in automatic control. According to Lewis (1992), "... the operation of the flyball governor was clearly visible even to the untrained eye, and its principle had an exotic flavour which seemed to many to embody the nature of the new industrial age. Therefore, the governor reached the consciousness of the engineering world and became a sensation throughout Europe."

Work began to be done on the mathematical analysis of such regulators by George Airy, followed by James Maxwell and others, with Maxwell's work on Watt's governor and its stability in 1868 and Stodola's study of the regulation of a water turbine in 1893. (Lewis 1992)

"By 1932 feedback systems were used extensively in applications such as power generation and transmission, steering of ships, autopilots for aircrafts, and process control." (Basar 2000, p. 1) Up to this point, the mathematical analysis of such systems had been carried out using differential equations in the time-domain. (Lewis 1992) Following World War I, long-distance telephoning became possible with the introduction of electronic amplifiers. (Franklin et al. 2002, p. 13) Large numbers of the repeater-amplifiers caused signal distortion. Harold Black is credited with solving the problem of reducing this distortion with his idea to use negative feedback amplifiers, while working for Bell Laboratories in 1927. The mathematics for frequency domain analysis had been developed by the mathematicians Laplace, Fourier, Cauchy and others and its use in designing stable amplifiers for telephone systems was explored by Harry Nyquist, who also worked for Bell Laboratories. (Lewis 1992) He introduced what is now called the Nyquist Stability Criterion. Another colleague, Hendrik Bode also contributed to frequency domain techniques, introducing other indicators of stability that are also based on frequency response. " [*Regeneration Theory*] by Nyquist, and the closely related papers by Black and Bode, represent a paradigm shift because they approached the problem of analysing

a feedback system in a totally different way ... Even though the work was strongly focused on feedback amplifiers, it became apparent several years later that the result could actually be applied to all control systems." (Basar 2000, p. 1)

Bode would go on to play an important engineering role in the efforts of the Allied Forces during World War II when Bell Laboratories was contracted by the National Defense Research Committee to design and build an electronic gun director for anti-aircraft guns. "Under the contract, BTL would design the machine, designated T-10, for use with the Army's new 90mm gun, which had hydraulic power controls for remote aiming." (Mindell 2000, p. 74)

Fig. 1.4: 90mm anti-aircraft gun M1 on mount M1 (Brooks 2008)

At the time, the mature technology for gun directors consisted of precision mechanical components for computation. The idea of an electronic director was proposed by D.B. Parkinson and "it used shaped wire-wound potentiometers and vacuum-tube amplifiers to perform standard arithmetic operations." (Irvine 2001, p. 24) A "ser-

vomechanism" was used to turn the potentiometer shaft. "The servo then "solved" an equation, merely by its tendency to reduce the error to zero." (Basar 2000, p. 73) The new German V-1 rockets, and these anti-aircraft guns, which were so successful against them, were often referred to as "robotic" weapons. In any such position control application, the end product is a mechanical motion. This fusion of electrical, mechanical and electromechanical subsystems is part of a field of engineering now known as "mechatronics".

Integrators were an important component in ballistics computations. Fig. 1.5 shows a mechanical integrator in the torpedo data computer of the USS Cod submarine.

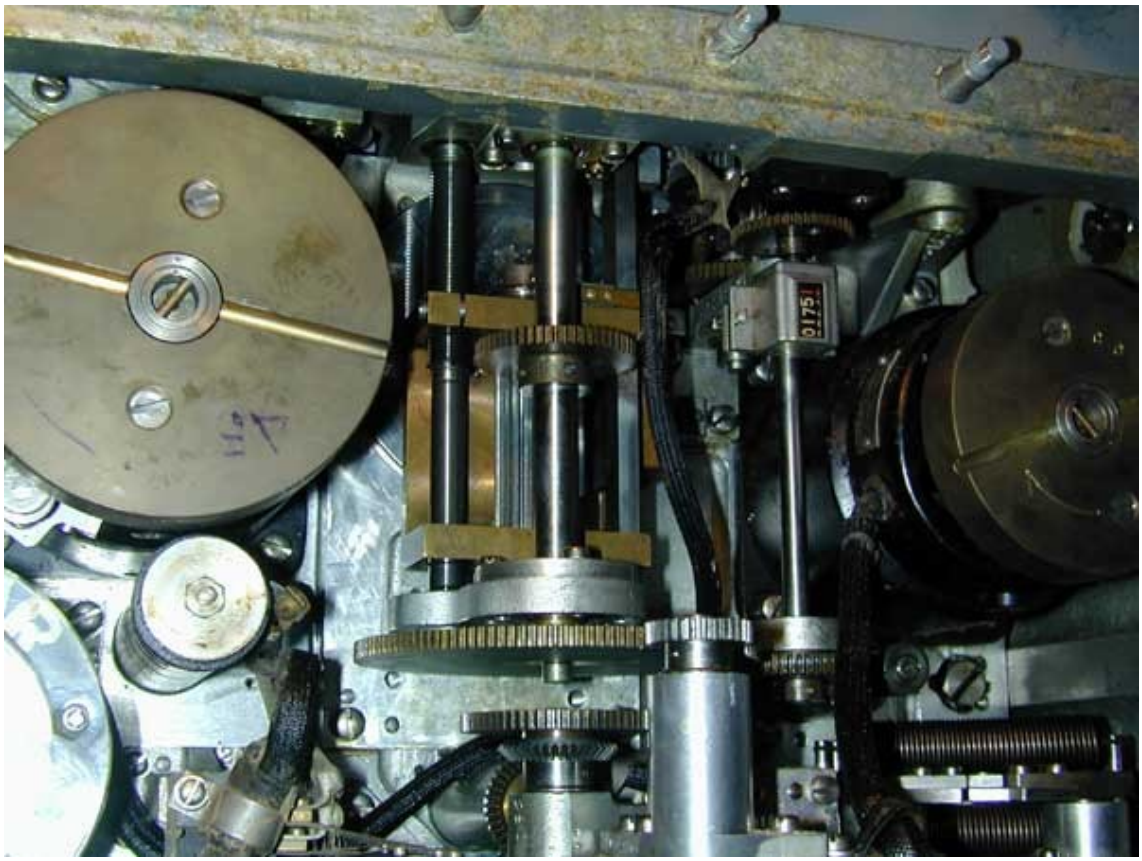


Fig. 1.5: Mechanical integrator inside the torpedo data computer of the USS Cod submarine (USS COD Home Port 2007)

Using mechanical integrators, which were later replaced by electronic integrators, "the behaviour of a system could be simulated by setting up an array of connected integrators that satisfied the same equations ... To make the simulation easy, the system equations had to be rearranged into a set of simple expressions representing the mixture of signals to be applied to each integrator." (Billingsley 1989, p. 2) This time-domain representation of a system, tied closely with simulation, led to a range of "state-space" techniques concerned with the behaviour of each variable defining the state of the system, with respect to time. These techniques were a step back towards the time-domain analysis of the 19th century, but have several advantages over frequency-domain techniques, which can only be applied to linear, time-invariant systems and are awkward in dealing with multiple-input-multiple-output (MIMO) systems. "For example, the state-space approach can be used to represent nonlinear systems that have backlash, saturation and dead zone... Time-varying systems, (for example, missiles with varying fuel levels...) can be represented in state space" (Nise 2004, p.127) Ways of dealing with these complexities became increasingly important with the arrival of the space age. In 1947, Nicolas Minorsky, who is most famous for his introduction of PID control in a 1922 paper on ship steering, published *Introduction to Non-Linear Mechanics*. "This work also made it clear that ordinary differential equations needed more consideration, that the classical development in this discipline was not sufficient for future technological development." (Flugge-Lotz 2003, p. 290)

"In the Soviet Union, there was a great deal of activity in nonlinear controls design. Following the lead of Lyapunov, attention was focused on time-domain techniques. In 1948, Ivachenko had investigated the principal of relay control... Tsypkin used the phase plane for nonlinear controls design in 1955." (Lewis 1992) In 1957, the Soviet Union launched the world's first satellite, Sputnik 1, from Kazakh SSR. "The control system of the Sputnik Rocket was tuned to provide an orbit with the following parameters: perigee height - 223 km, apogee height - 1450 km, orbital period - 101.5 min." The launch resulted in an orbit with initial parameters: perigee height

- 223 km, apogee height - 950 km, initial orbital period - 96.3 min. (*Sputnik 1* 2008)

State-space methods were sometimes found to lack robustness against variations in system properties as opposed to the earlier frequency-domain techniques, which were fairly robust (*Robust Control* 2008). Along with the digital computer, have come digital control and advances in control theory, such as adaptive control, which involves adapting the control algorithm to compensate for changes in system properties over time. Robotics is one of the forefronts where the rapid progress of control theory is being used in position control. The aim of this project is to investigate a few of the large range of techniques being used today and gain some insight regarding what works best in position control.

2 Modelling The System

2.1 Introduction

Mathematics is the formal language of automatic control theory. (Lewis 1992) In order to make use of the design techniques which have been established in the field, mathematical models of many of the devices used in the servo system, are required. To begin with, the dynamics of the system have been described with principles from electrics, electrodynamics and mechanics. The digital system using discretised control signals can then be analysed.

2.2 Background

As previously mentioned, the servo system being considered in this investigation uses a DC motor unit, together with a variety of sensors, a computer, interface components and drive circuitry in order to achieve closed-loop control of shaft position. The functioning of some of these devices will now be discussed and a brief background on some control principles provided.

2.2.1 DC Motors

This section provides a brief background on the basics of permanent magnet (PM), direct current (DC) motors. An electric motor consists of two main parts. The *stator* is that part of the motor which does not move. The *rotor* is the part that moves and is usually mounted inside the stator on bearings. The electrical winding in which voltage is induced is called the *armature winding*. (Sen 1997, p. 125) In DC machines, this winding is usually placed in slots around the rotor, so the terms rotor and armature are often used interchangeably.

Torque-Speed Characteristic

The armature and stator of a two-pole, PM, DC motor can be accurately modelled as a cylindrical core mounted between two hollowed out magnetic poles, so that a uniform air gap exists between them around the circumference of the armature, as shown in Fig. 2.1.

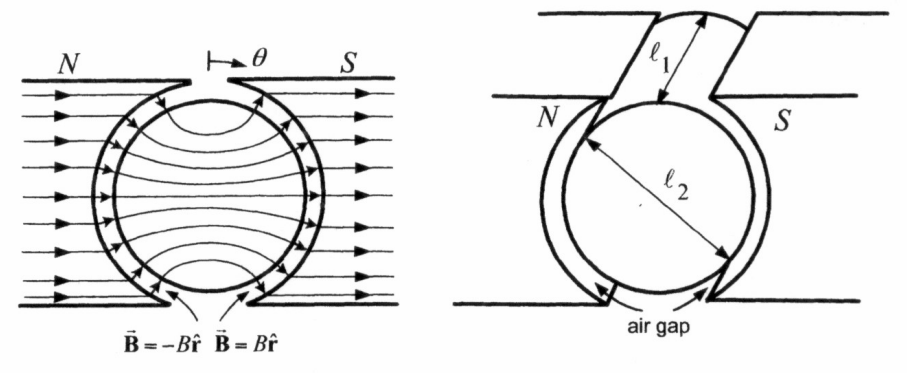


Fig. 2.1: Soft iron cylindrical core placed inside a hollowed out permanent magnet to produce a radial magnetic field in the air gap. From *Modelling and High Performance Control of Electric Machines* by Chiasson.

The angle between the centres of two adjacent poles, which in the case of Fig. 2.1 with its two poles, is 180°, is known as the *pole pitch*. The angle between the armature slots holding the two sides of a coil is known as the *coil pitch*. (Sen 1997, p. 134) Most DC motor windings use coils which are nearly full-pitch, meaning that the coil pitch is almost as large as the pole pitch. As an example, the DC motor winding shown in Fig. 2.2, was mounted in a two pole permanent magnet stator and it has five slots, so that its pole pitch is 180° or 2.5 slots. It can also be seen that the coils have been wound at a pitch of 2 slots. Therefore, they are slightly less than full-pitch.

Fig. 2.2: A permanent magnet DC motor winding with a coil pitch of 2 slots.

Chiasson (2005, pp. 15-18) develops, as follows, the relationship between the back emf of a single turn of wire and the speed of the armature.

A patch of infinitesimally small area on the outer cylindrical surface of the armature can be represented by its normal vector $\mathrm{d}\boldsymbol{S}$, with a magnitude equal to the area of the patch.

$$\mathrm{d}\boldsymbol{S} = \frac{l_2}{2}\mathrm{d}\theta\mathrm{d}z\hat{\boldsymbol{r}} \tag{2.1}$$

where: l_2 is the diameter of the armature

θ is the angle shown in Fig. 2.1 in radians

z is a distance along the depth of the armature

$\hat{\boldsymbol{r}}$ is the unit vector radial to the cylindrical surface of the armature

The magnetic flux Φ through that part of the armature surface subtending some angle θ, is given by Eqn. 2.2. An electrodynamics analysis of this structure would show that the flux density in the air is radially directed and essentially constant in

magnitude. (Chiasson 2005, pp. 6)

$$\Phi(\theta) = \iint_S \boldsymbol{B} \cdot \boldsymbol{S} \tag{2.2}$$

where $\boldsymbol{B}$ is the magnetic flux density produced by the poles

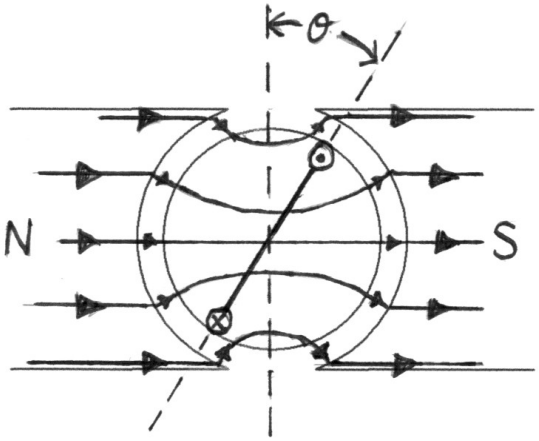


Fig. 2.3: The plane of a full-pitch turn of wire wound around the centre of the armature divides the cylindrical surface of the armature into two halves. The flux through the turn is equal to the net flux through one of these halves.

Fig. 2.3 shows a full-pitch turn of wire in a plane, which is at an angle θ with the vertical plane passing through the centre of the armature. The plane of the turn divides the cylindrical armature surface into two halves. Ignoring changes in the magnetic field within the vicinity of the ends of the armature, the flux through the full-pitch turn, is equal to the net flux through one of these halves. For $0 < \theta < \pi$, this flux is:

$$\begin{aligned}
\Phi(\theta) &= \int_0^{l_1} \int_\theta^\pi \boldsymbol{B}\hat{\boldsymbol{r}} \cdot \frac{l_2}{2} \mathrm{d}\theta \mathrm{d}z \hat{\boldsymbol{r}} + \int_0^{l_1} \int_\pi^{\pi+\theta} -\boldsymbol{B}\hat{\boldsymbol{r}} \cdot \frac{l_2}{2} \mathrm{d}\theta \mathrm{d}z \hat{\boldsymbol{r}} \\
&= \frac{Bl_1l_2}{2}(\pi - \theta) - \frac{Bl_1l_2}{2}\theta \\
&= -Bl_1l_2\theta + Bl_1l_2\frac{\pi}{2}
\end{aligned} \tag{2.3}$$

For $\pi < \theta < 2\pi$, it is:

$$\begin{aligned}\Phi(\theta) &= \int_0^{l_1}\int_\theta^{2\pi} -\boldsymbol{B}\hat{\boldsymbol{r}}\cdot\frac{l_2}{2}\mathrm{d}\theta\mathrm{d}z\hat{\boldsymbol{r}} + \int_0^{l_1}\int_{2\pi}^{2\pi+(\theta-\pi)} \boldsymbol{B}\hat{\boldsymbol{r}}\cdot\frac{l_2}{2}\mathrm{d}\theta\mathrm{d}z\hat{\boldsymbol{r}} \\ &= -\frac{Bl_1l_2}{2}(2\pi-\theta) + \frac{Bl_1l_2}{2}(\theta-\pi) \\ &= -\frac{Bl_1l_2}{2}(3\pi-2\theta) \\ &= Bl_1l_2\theta - Bl_1l_2\frac{3\pi}{2} \end{aligned} \tag{2.4}$$

This relationship between the angle of rotation of the turn and the flux through it is shown in Fig. 2.4.

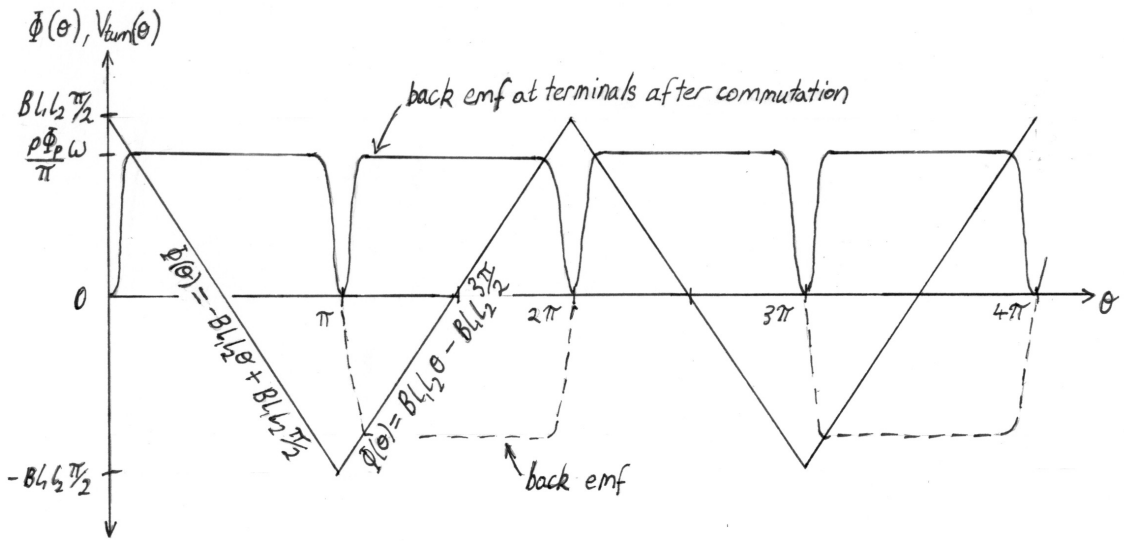


Fig. 2.4: The variation in flux through a turn of wire as the angle of rotation changes and the back emf generated according to Faraday's Law.

These results are now extended to determine the back emf generated by the whole winding as the rotor turns, for a motor with any number of poles. The flux Φ_p emanating from each pole is:

$$\begin{aligned}\Phi_\mathrm{p} &= BA_\mathrm{p} \\ &\approx \frac{B\pi l_1 l_2}{p}\end{aligned} \tag{2.5}$$

$$\frac{p\Phi_p}{\pi} = Bl_1l_2 \tag{2.6}$$

where: A_p is the area of the pole surface
p is the total number of poles

From Eqns. 2.3, 2.4 and 2.6, the flux through a full-pitch turn is:

$$\Phi(\theta) = -\frac{p\Phi_\mathrm{p}}{\pi}\theta + \frac{p\Phi_\mathrm{p}}{2} : \qquad 0 < \theta < \pi \tag{2.7}$$

$$\Phi(\theta) = \frac{p\Phi_\mathrm{p}}{\pi}\theta - \frac{3p\Phi_\mathrm{p}}{2} : \qquad \pi < \theta < 2\pi \tag{2.8}$$

By Faraday's Law, the voltage induced in the turn is the rate of change of flux through it.

$$V_\mathrm{turn} = -\frac{\partial\Phi}{\partial t} \tag{2.9}$$

$$= -\frac{\partial\Phi}{\partial\theta}\cdot\frac{\mathrm{d}\theta}{\mathrm{d}t}$$

$$= \pm\frac{p\Phi_\mathrm{p}}{\pi}\omega \tag{2.10}$$

where ω is the angular speed of the motor

In a DC motor, the current in the coils is commutated as the sides of the coil pass between the poles so that the current through it is always in one direction and the motor may develop torque. Consequently, the induced voltage is also in one direction, when measured at the motor terminals. Commutation is achieved by terminating the coils on copper segments at the end of the rotor, as shown in Fig. 2.5. Carbon brushes connected to the power supply rub against the segments so that the connection to each coil is reversed when its sides pass between poles.

Fig. 2.5: The coils are terminated on copper commutator segments at the end of the rotor.

Except during the short commutation period, the voltage is therefore constant and given by:

$$V_{\text{turn}} = \frac{p}{\pi}\Phi_{\text{p}}\omega \tag{2.11}$$

The method of winding employed dictates the way in which the ends of the coils are connected together via the commutator segments and thus, the number of parallel paths through the winding for current to flow at any one moment. Therefore, the overall back emf E_a induced in the armature winding is equal to the product of the total number of turns in the winding and the voltage induced in a single winding divided by the number of parallel paths a, as shown in Eqn. 2.12.

$$E_{\text{a}} = \frac{Np}{\pi a}\Phi_{\text{p}}\omega \tag{2.12}$$

The factor $Np/\pi a$ in Eqn. 2.12 is often called the armature constant K_a.

$$E_{\text{a}} = K_{\text{a}}\Phi_{\text{p}}\omega \tag{2.13}$$

$$\text{where } K_\text{a} = \frac{Np}{\pi a}$$

It can also be shown that the torque T generated by a DC motor is:

$$T = K_\text{a}\Phi_\text{p}I \tag{2.14}$$

where I is the current drawn by the motor

For a practical winding, in which the coils are not full-pitch, there is less flux through each coil and the back emf is reduced somewhat. Also, the magnetic field established by the poles is distorted by the current-carrying coils. Eqn. 2.13, gives great insight for the design and control of DC motors, however the values of the various parameters are best determined experimentally.

Steady State Response

When a DC motor reaches its steady-state speed, the back emf is equal the difference between the voltage V_t applied at the motor terminals and the voltage drop due to the resistance R_a of the armature winding, in accordance with Kirchhoff's voltage law.

$$E_\text{a} = V_\text{t} - I_\text{a}R_\text{a} \tag{2.15}$$

From Eqns. 2.13, 2.14 and 2.15, the relationship between speed and torque is given by:

$$\begin{aligned}\omega &= \frac{V_\text{t} - I_\text{a}R_\text{a}}{K_\text{a}\Phi_\text{p}} \\ &= \frac{V_\text{t}}{K_\text{a}\Phi_\text{p}} - \frac{R_\text{a}}{(K_\text{a}\Phi_\text{p})^2}T\end{aligned} \tag{2.16}$$

The factor $K_\text{a}\Phi_\text{p}$ is often called the speed or voltage constant k_v. A torque constant, which satisfies:

$$k_\text{T} = \frac{T}{I} \tag{2.17}$$

can also be used. Eqn. 2.14 suggests that the torque constant is also equal to $K_\text{a}\Phi_\text{p}$ which would make the torque and voltage constants equal, but this is not quite the

case, due to effects which haven't been modelled such as the nonlinear voltage drop across the commutators. Eqn. 2.18 shows the same relationship as Eqn. 2.16 with the constants determined experimentally.

$$\omega = \frac{V_\mathrm{t}}{k_\mathrm{v}} - \frac{R_\mathrm{a}}{k_\mathrm{v} k_\mathrm{T}} T \tag{2.18}$$

where k_v and k_T are determined experimentally

After performing a dynamometer test on a DC motor at the rated terminal voltage in order to obtain the no-load speed and stall torque, the voltage and torque constants, from Eqn. 2.18, are given by:

$$k_\mathrm{v} = \frac{V_\mathrm{t}}{\omega_\text{no-load}} \tag{2.19}$$

$$\frac{k_\mathrm{T}}{R_\mathrm{a}} = \frac{T_\text{stall}}{V_\mathrm{t}} \tag{2.20}$$

Transient Response

In accordance with Newton's Law, the torque required to accelerate a mass load with moment of inertia J, is:

$$T = J\dot{\omega} \tag{2.21}$$

Eqn. 2.18 is accurate in the steady state, for which the back emf due to the inductance of the armature winding has died away. However, if this inductance is negligible, it may also be used in determining the transient speed response of the motor and load. From Eqn. 2.21, the speed response of the motor is then:

$$\begin{aligned} J\dot{\omega} &= k_\mathrm{T} \frac{V_\mathrm{t} - k_\mathrm{v}\omega}{R_\mathrm{a}} \\ \omega &= \frac{V_\mathrm{t}}{k_\mathrm{v}} + c e^{-\frac{k_\mathrm{T} k_\mathrm{v}}{J R_\mathrm{a}} t} \\ &= \frac{V_\mathrm{t}}{k_\mathrm{v}} + \left(\omega(0) - \frac{V_\mathrm{t}}{k_\mathrm{v}}\right) c e^{-\frac{k_\mathrm{T} k_\mathrm{v}}{J R_\mathrm{a}} t} \end{aligned} \tag{2.22}$$

2.2 Background

Fig. 2.6 shows the mechanical unit with the square wave output connected to the trigger and one channel of the storage oscilloscope in Fig. 2.7. With the ribbon cable connected to the analogue unit in Fig. 2.8 , the square wave frequency was set very low; to a fraction of a Hz so that the transient speed response of the motor could be viewed.

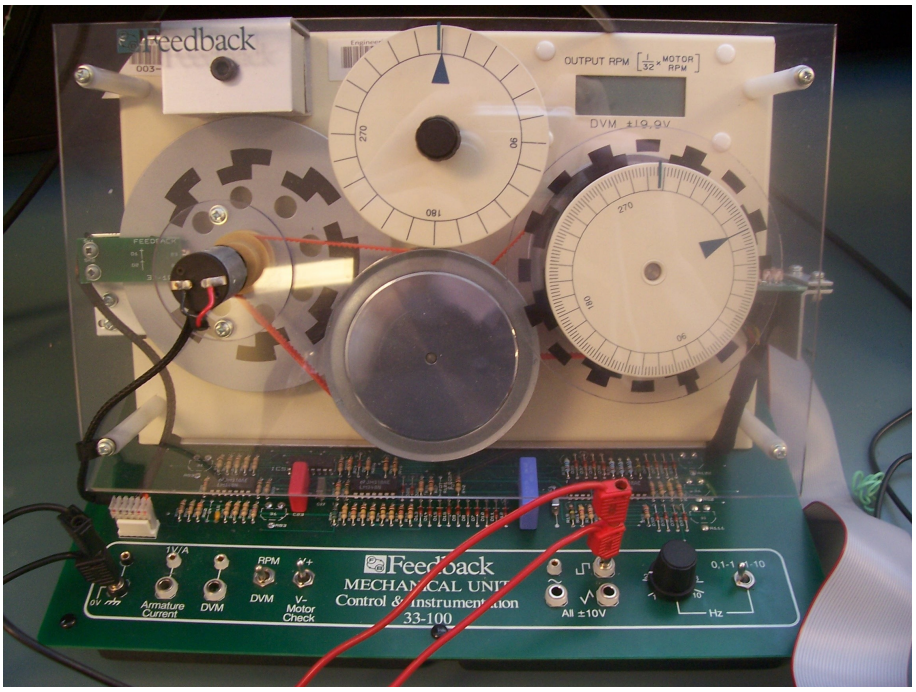


Fig. 2.6: Mechanical unit with connections to the square wave output.

Fig. 2.7: Storage oscilloscope for viewing sensor voltages.

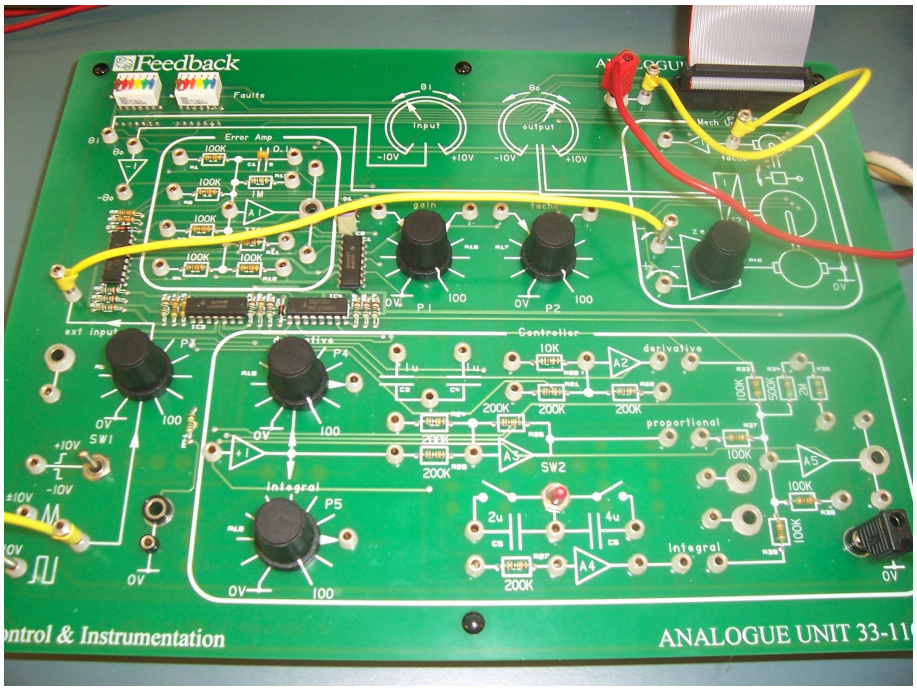


Fig. 2.8: The analogue unit with connections to the tacho output.

The second channel of the oscilloscope was connected to the tacho terminals. Figs. 2.9 and 2.10 show the tacho voltage as the motor speeds up in either direction in response to the square wave input. The transient can be seen to be of a similar form to the transient in Eqn. 2.22.

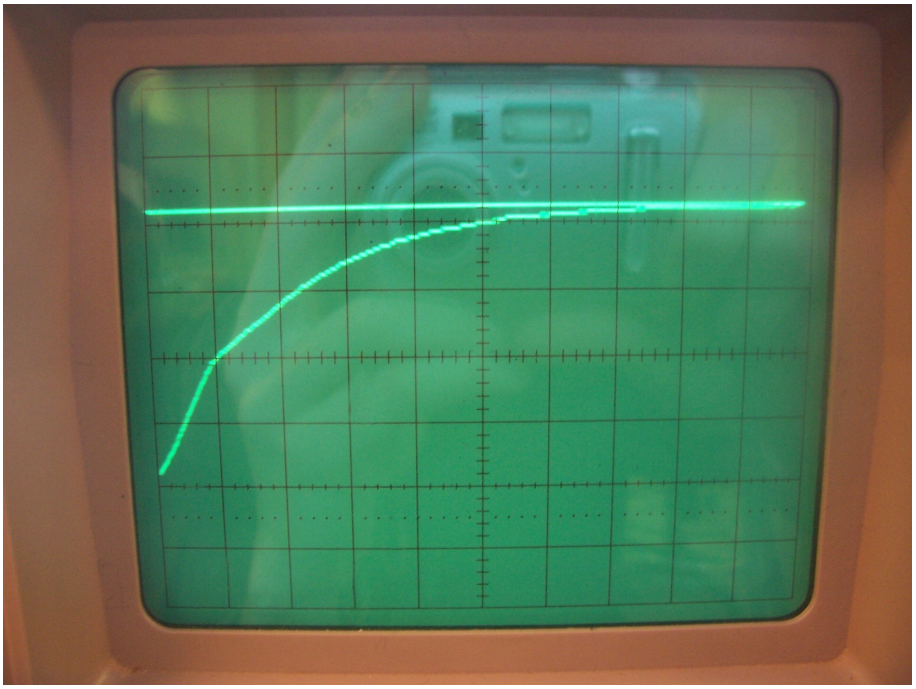


Fig. 2.9: Tacho voltage as the motor speeds up in response to square wave input.

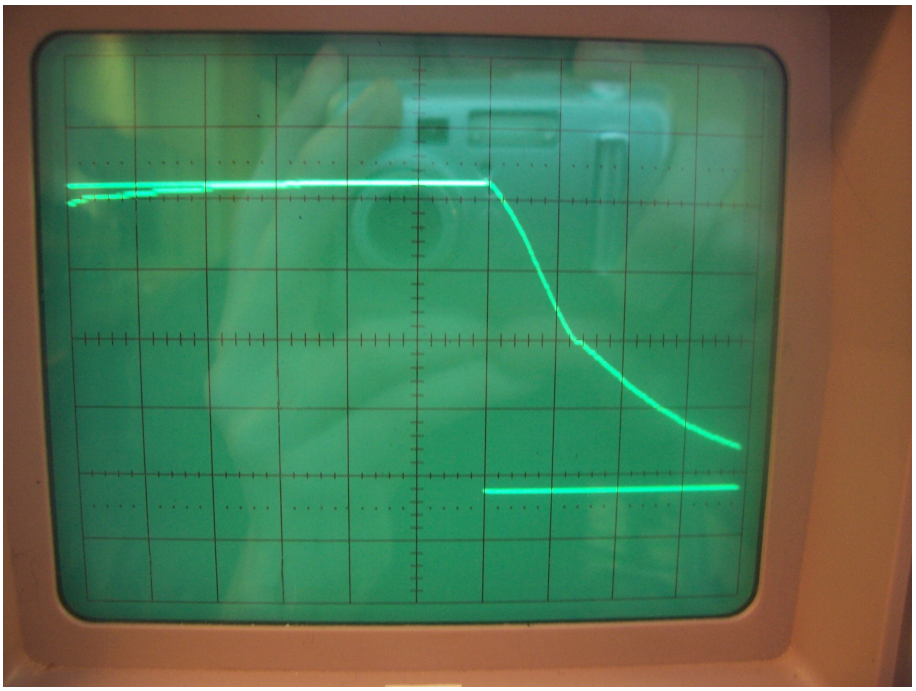


Fig. 2.10: Tacho voltage as the motor speeds up in reverse.

According to Eqn. 2.22, the time constant τ of the transient speed response of the motor and load is:

$$\tau = \frac{JR_{\mathrm{a}}}{k_{\mathrm{T}}k_{\mathrm{v}}} \tag{2.23}$$

For a load driven by a DC motor through a gear train or belt drive, the total equivalent inertia reflected back to the motor shaft, may not be known. If the voltage constant k_{v} and the ratio $k_{\mathrm{T}}/R_{\mathrm{a}}$ are known, the value of the inertia may be determined by measuring the time constant.

2.2.2 Driving Circuitry

An efficient, reversible, adjustable speed drive can be achieved using pulse width modulation (PWM) with a switch mode converter called a *H-bridge*. The H-bridge is a four quadrant, full bridge DC-to-DC converter and owes its name to the general

shape of the circuit schematic, shown in Fig. 2.11. It consists of four power switching devices, for example power MOSFETs, and associated components for applying control signals to them. The diodes, which are additional to those intrinsic to the MOSFETs may be needed to protect them from the back emf of the motor during their switch-off transitions.

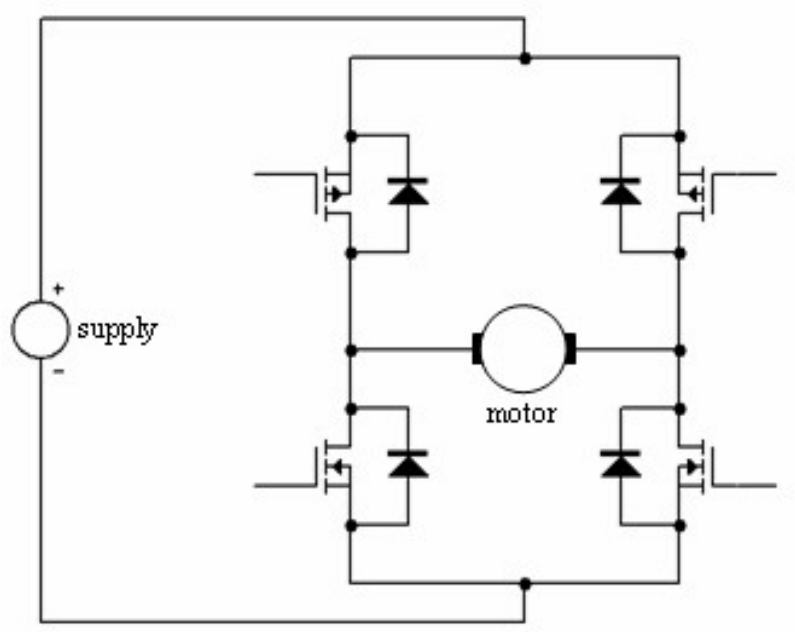


Fig. 2.11: H-bridge.

If the converter is operated in *bipolar* mode, the transistors are switched on and off such that the voltage applied to the motor is continually switched between that of the supply and the negative of the supply at a cycle frequency dependent on the control signals. The ratio of the time that it is positive, to the overall cycle period is the duty cycle. If the time constant of the motor is long in comparison to the cycle period, then the motor only responds to the average of the applied voltage. Although vibration may be induced, this can be of benefit in reducing static friction. For the motor to come to a stand-still, the duty cycle must be 50 percent, meaning that power is being used even if there is no disturbing torque. Efficiency, which is an important consideration in minimising power dissipation by the transistors,

can be increased by instead switching the applied voltage between positive of the supply and zero when the required speed is forwards, and between negative of the supply and zero when the required speed is backwards. This is the *unipolar* mode of operation.

Switching signals can be generated using analogue components, the built in module for PWM in many microcontrollers, separate PWM integrated circuits, or in software.

2.2.3 Sensors

In order to implement feedback control of a system, the feedback must be in a form compatible with the controlling devices. For an electronic controller, feedback signals provided by the sensors must consist of electrical quantities like voltage and current. Sensors used in position control include position and velocity sensors.

Position Sensing

Many systems that produce linear motion do so by converting rotational motion to translation. This might be achieved using a power screw, rack and pinion, pulley or some other mechanism. Therefore, measurement of rotation is a good way of measuring position both for systems that produce rotational motion and those that produce linear motion. A simple analogue device for measuring rotation is the *potentiometer* or *pot*. The resistance of a pot changes as its shaft rotates. Pots can be purchased with linear or logarithmic rotation-resistance characteristics and various degrees of precision. Between two of its terminals, the pot has a fixed resistance and the pot is chosen according to the current limiting and power dissipation capabilities determined by this resistance. The third terminal provides electrical paths to the other two, dividing the resistance into two. As the shaft is turned, one resistance increases and the other decreases by the same amount. By connecting two terminals

to a DC voltage supply, it forms a voltage divider, with the voltage at the third terminal varying as the shaft is rotated. This voltage can be amplified and used directly in an analogue controller or it can be sampled by an analogue to digital converter for use by a digital controller with an amplification gain determined in software.

Optical encoders provide another means of obtaining digital position measurements. Figs. 2.12 and 2.13 shows the *absolute encoder* of the Feedback Instruments unit. Dark tracks are printed on the transparent disc attached to the shaft in the pattern shown in Fig. 2.14. Infra-red readers are lined up with each of the seven rings, each one providing a low or high voltage depending on whether the light passes through or is blocked by the tracks. The binary digits assigned to each voltage are not weighted in binary but in *Gray code*. This is so that the pattern can be printed such that as each sector rotates past the readers only one bit changes at a time. If several bits were to change between each two sectors, such as in pure binary, the track which passes the reader slightly sooner than the next may have a high weighting and result in an intermediate position reading nothing like that of either sector.

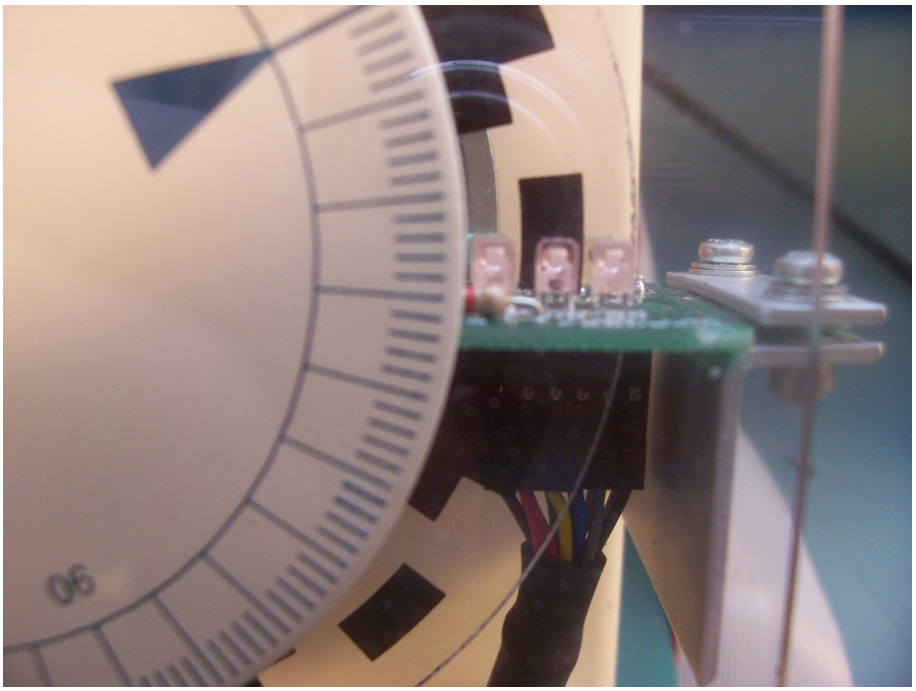


Fig. 2.12: Absolute encoder for position sensing.

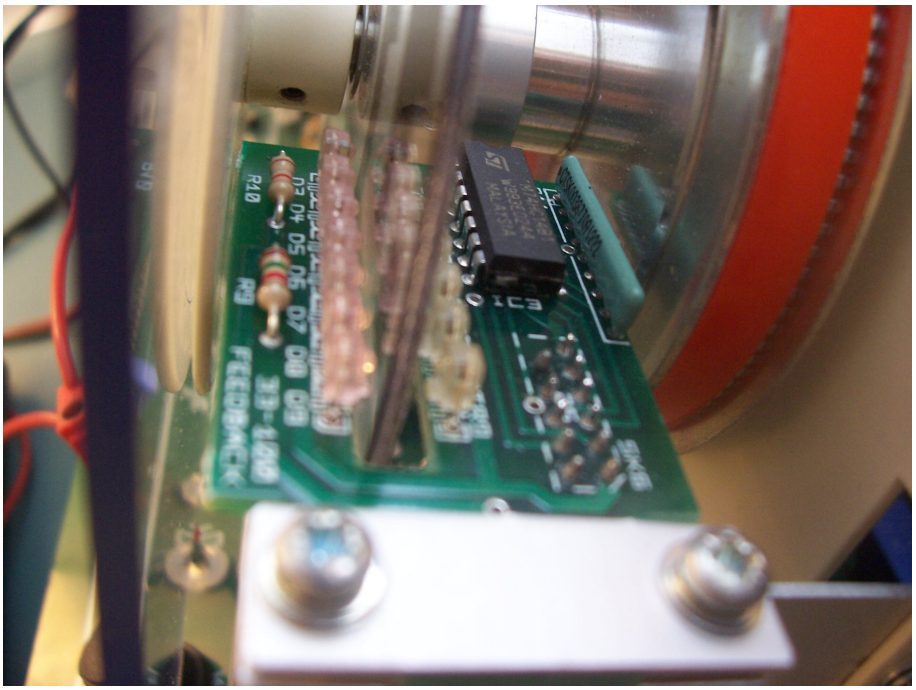


Fig. 2.13: Infra-red readers of the absolute encoder.

Fig. 2.14: Gray code pattern of dark tracks on the disc. (Feedback Instruments Limited 1999)

Fig. 2.15 shows the incremental encoder on the mechanical unit. Small windows are cut out of the steel disc and the two readers which are slightly offset from one another provide to square wave signals in quadrature whose frequency depends on the speed of the shaft. When the shaft is rotating in one direction the first wave leads the second. When the shaft is rotating in the other direction the second wave leads the first.

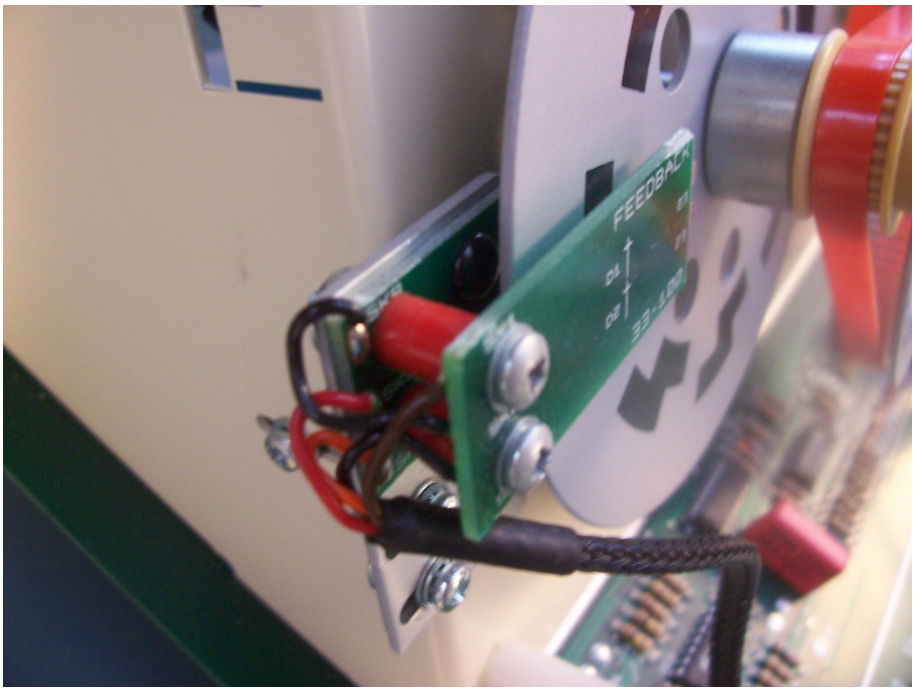


Fig. 2.15: Incremental encoder for position sensing.

Other possibilities for position sensing include the Hall-effect sensor and rotary transformers such as the resolver or synchro.

Velocity Sensing

The rate of change of position is also a valuable quantity in position control. An analogue device for measuring rotational velocity is the *tachogenerator* or *tacho*. A tacho is a low power, precision, electrical generator which generates a precise voltage proportional to the speed of its rotor. Again, this voltage can be used for analogue control or sampled for digital control. The tacho on the mechanical unit is coupled to the motor shaft, as shown in Fig. 2.16.

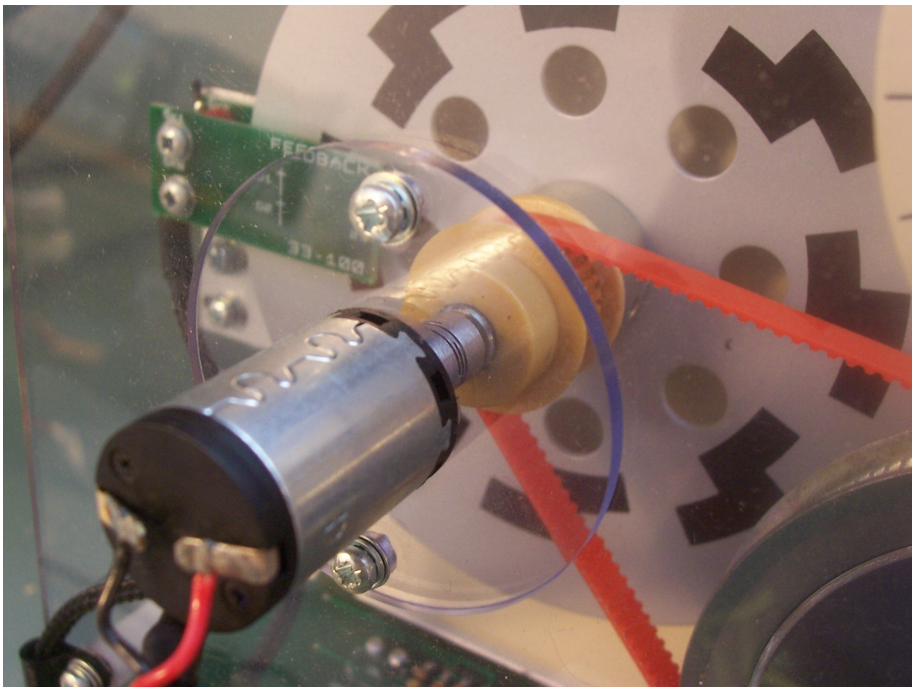


Fig. 2.16: Tacho coupled to the motor shaft.

2.2.4 The Transfer Function

The position output of the servo system, is changed by increasing or decreasing the electrical drive to the input. The position is thus controlled by an appropriate input signal. The relationship between the input and output of a single-input-single-output (SISO) analogue system may be described by a single differential equation. This relationship can also be described by a *transfer function* as shown in Fig. 2.17.

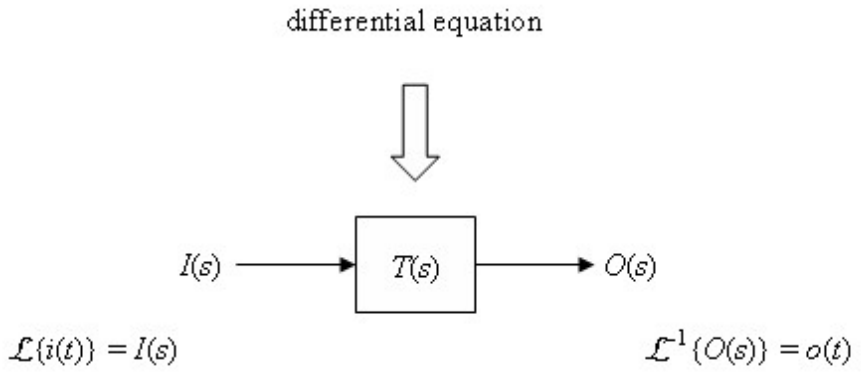


Fig. 2.17: The relationship between the input and output of a SISO system may be described by a single differential equation. This relationship can also be described by a transfer function.

The transfer function is used by first breaking the input signal into a spectrum of frequency components and their amplitudes using a mathematical transform. The variable s is used to denote complex frequency. This spectrum is then multiplied by the transfer function giving new amplitudes for the components. The components with the new amplitudes are then added together again using the inverse of the transform and the result is the output, such as position, as a function of time. The benefit of transforming signals from the *time domain* to the *frequency domain* in this way, is that it changes the problem of predicting the output response from one of solving a complex differential equation to an algebraic problem. The transform achieving this purpose for continuous-time (analogue) signals is the Laplace Transform. The Laplace transform of an analogue signal $f(t)$ which starts at time zero and its inverse are defined by:

$$F(s) = \mathcal{L}\{f(t)\} = \int_{0^-}^{\infty} f(t)e^{-st}\mathrm{d}t \tag{2.24}$$

$$f(t) = \mathcal{L}^{-1}\{F(s)\} = \frac{1}{2\pi j}\int_{\sigma-\infty j}^{\sigma+\infty j} F(s)e^{st}\mathrm{d}s \tag{2.25}$$

where σ is a real number so that the path of integration is in the region of convergence of $F(s)$

There are tables listing the transforms of commonly encountered signal forms.

The transfer function contains information about the nature of the system and its natural response to an input, which together with the forced response, forms the complete response of the system. The values of s which make the transfer function infinite are called the system's *poles*. The poles give the time constants of the response transients.

2.2.5 Closed-Loop Systems

In order to keep control of a process, some form of feedback is required. For example, in a car, it might be the driver's perception of speed and acceleration causing him/her to press or release the accelerator pedal or it might be measurements made by a cruise control system. In a position control system, if the signal from the position sensor is fed back and subtracted from the input, which is a position command signal, the result is the amount of *error* and this is used to drive the system towards correcting that error.

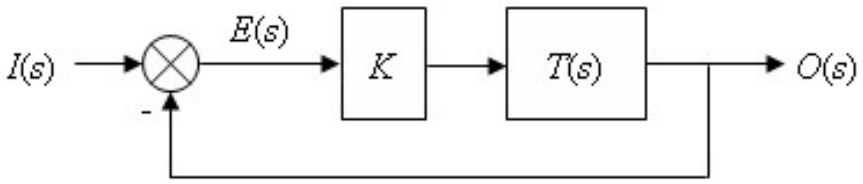


Fig. 2.18: The signal from the position sensor is fed back and subtracted from the input. The amount of error is used to drive the system.

By simply amplifying that error signal, the urgency of the action and drive to the system can be made proportional to the error. That is, if the position starts way out from the commanded position, a large amount of drive is applied and as it approaches the correct position the drive decreases to nothing. The larger the gain

of that amplifier, the quicker the response. The controller is then a basic amplifier implementing *proportional control*. This simple form of feedback is extremely important, however it introduces problems which must be overcome. A system with feedback implemented in such a way is then called the *closed-loop* system and has an associated closed-loop transfer function.

2.2.6 Compensation

In section 2.2.5, proportional control was mentioned. Although a high proportional gain is required for quick response or low *rise time*, as it is increased it starts to cause the system to overshoot its position target due to the momentum gained and it may overshoot several times, taking just as long settling to an acceptable level. Also, with very high gains, the system may become unstable. With respect to the responses achievable using proportional feedback only, low rise time and low settling time are conflicting interests. Fig. 2.19 shows the potentiometer voltage indicating the position of the output shaft in response to step inputs, with position feedback applied. Again, the step inputs were obtained from a very low frequency square wave. With the proportional gain that was used, some overshoot is observed. Fig. 2.20 shows the effect of increasing the gain. Rise time has been reduced but settling time has been lengthened with unacceptable oscillations. The positive step response only is shown in Fig. 2.21 and the corresponding error signal used to drive the motor in Fig. 2.22. In Fig. 2.23, the gain has been increased even further causing wild oscillations. Clipping of the peaks in the corresponding error signal in Fig. 2.24 shows that in practice, the influence of high gain, at the time that a large input or disturbance is applied to the system, is limited by saturation of the drive amplifier.

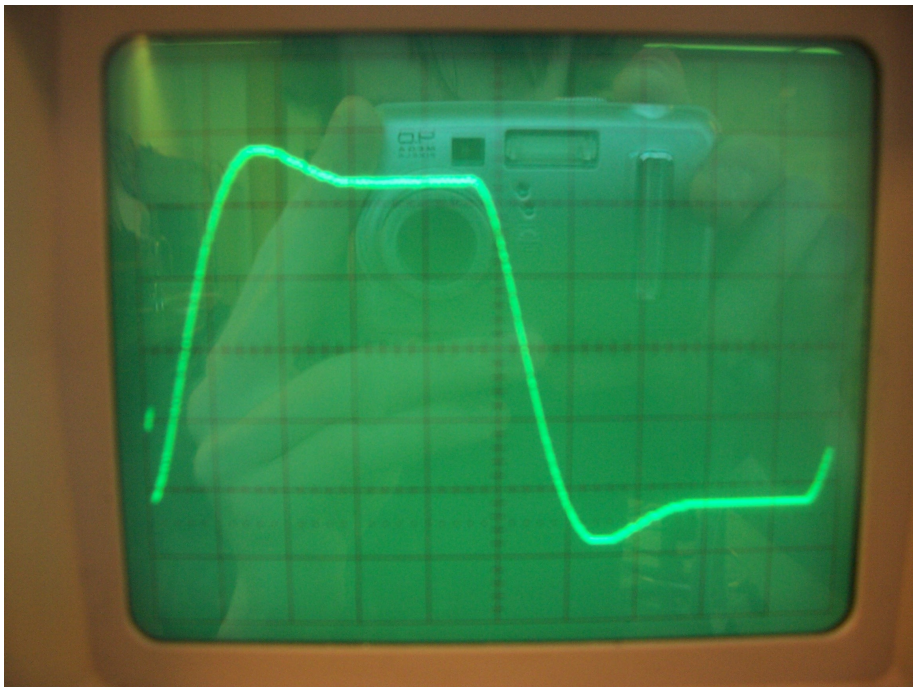


Fig. 2.19: Potentiometer voltage indicating the position of the output shaft in response to step inputs, with position feedback applied.

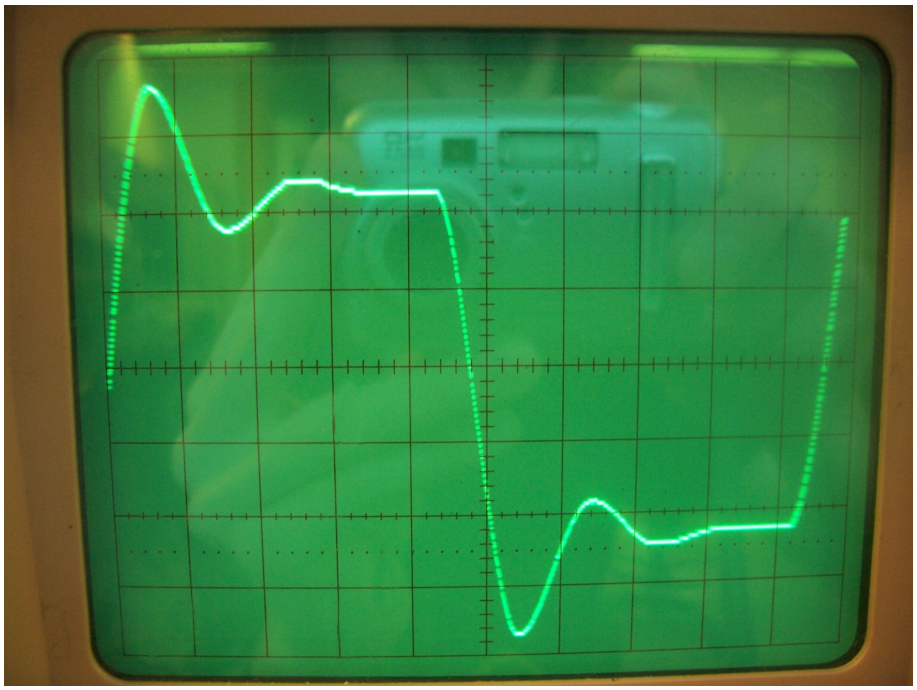


Fig. 2.20: Step response with increased proportional gain.

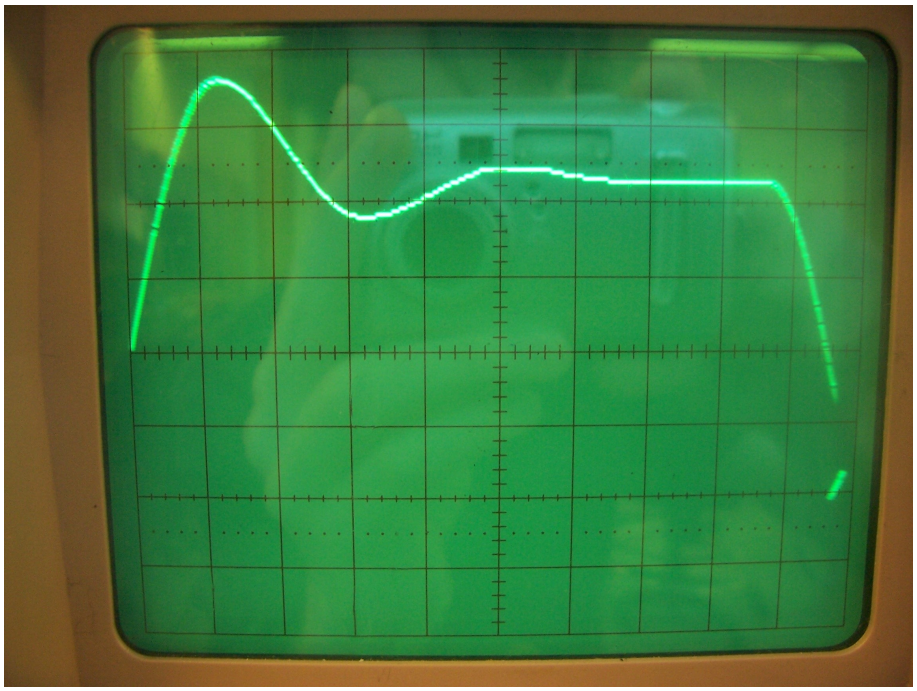


Fig. 2.21: The positive step response of Fig. 2.20 with smaller time base.

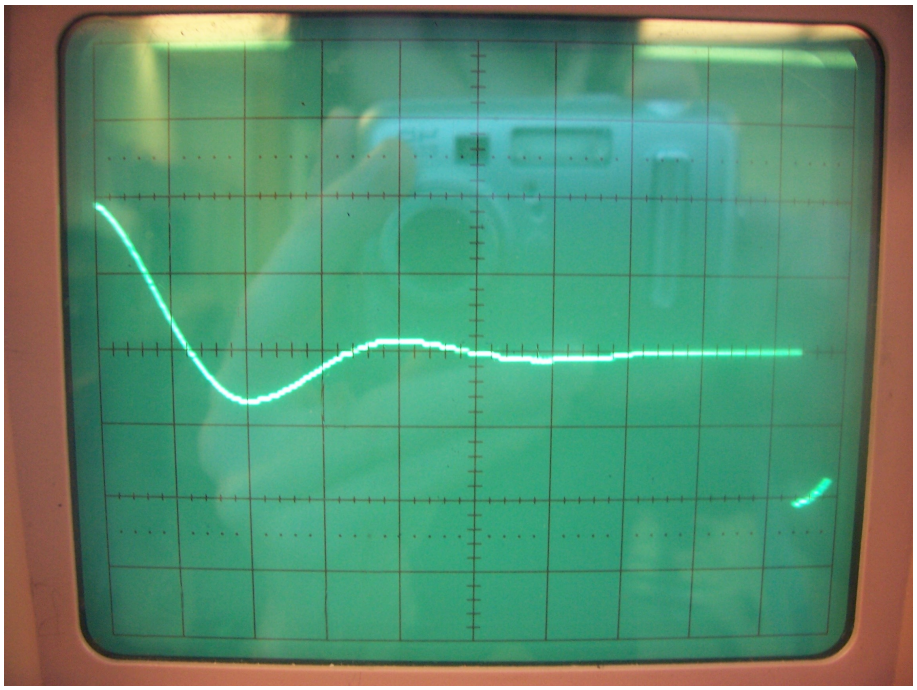


Fig. 2.22: Error signal used to drive the motor to give the response in Fig. 2.21.

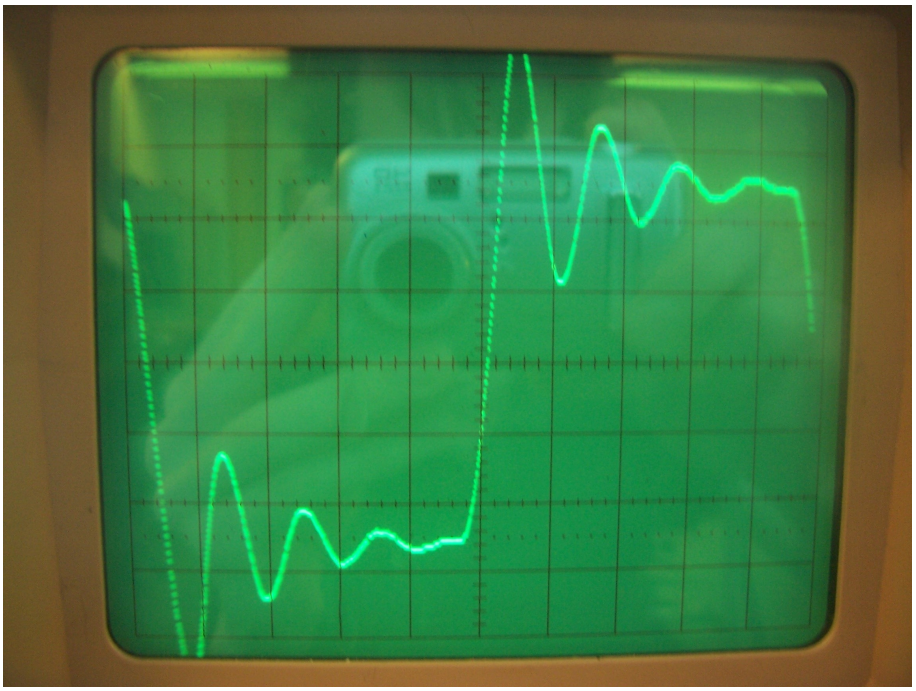


Fig. 2.23: Step response with proportional gain increased further.

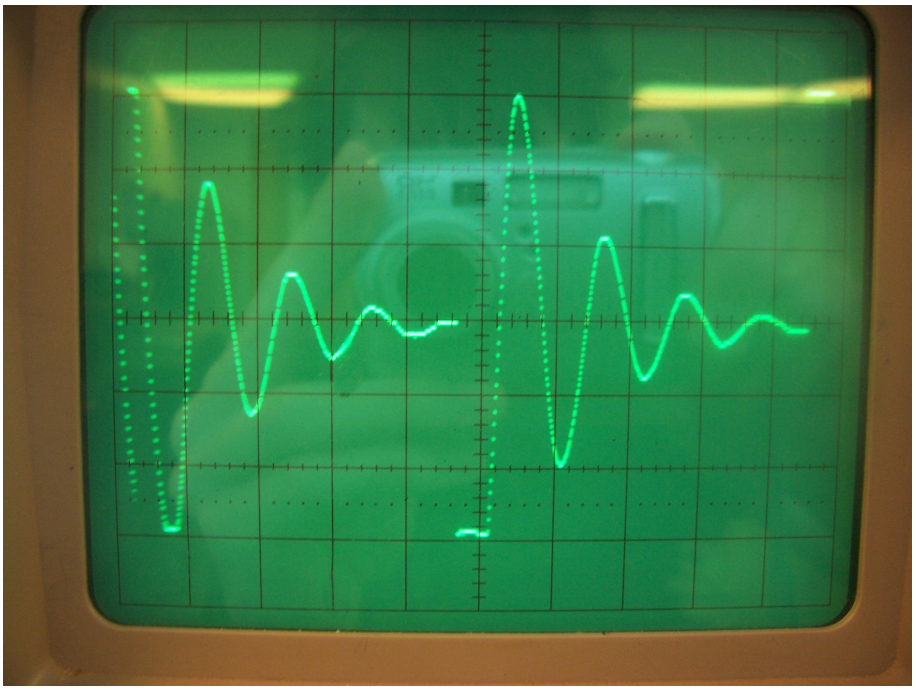


Fig. 2.24: Error signal used to drive the motor to give the response in Fig. 2.23.

Instead of changing a system or replacing in entirely, it can be *compensated* so that certain response requirements can be met simultaneously. From the point of view of the system's transfer function, compensators can be placed in cascade with the system, called *cascade compensation*, or in the feedback path, called *feedback compensation*. (Nise 2004, p. 502)

One way to reduce or stop overshoot is for the system to start to "hit the brakes" as the rate of reduction in error reaches a high level. This can be implemented by negative velocity feedback from a velocity sensor or by differentiating the error signal and adding that to the drive. Generally, an analogue signal should not be differentiated, as any high frequency electrical noise will be amplified the most, however, the digital form of derivative feedback does not suffer this effect.

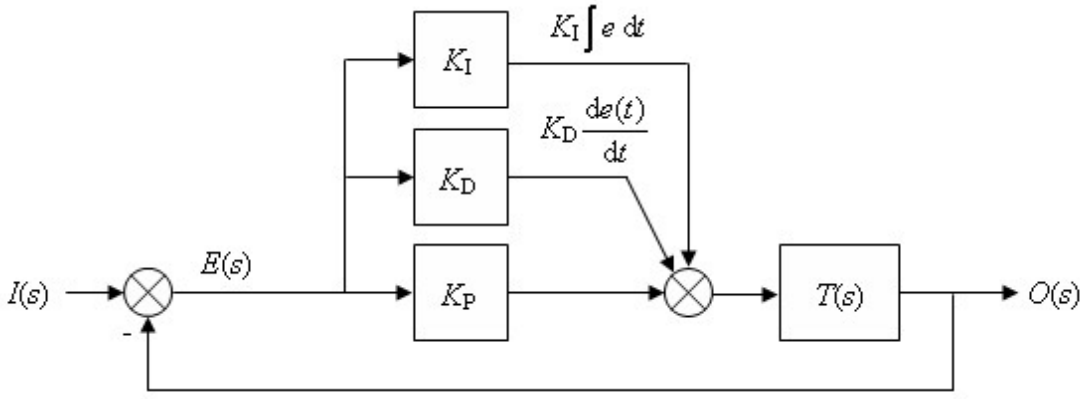


Fig. 2.25: Overshoot can be reduced by differentiating the error signal and adding that to the drive. Steady-state and following errors can be eliminated by integrating the error signal and adding that to the drive

Once the output has settled, there may be a standing or steady-state error. The system may be required to reduce this error to zero. Alternatively, the system may be required to follow a command that keeps changing, with minimal error as it lags behind, such as with a machine tool following some sought of trajectory. For these control actions to happen, a high enough drive must be applied. However, the error

providing that drive must be minimal. Therefore, this is another reason for the proportional gain to be as high as possible. But, as previously mentioned, for a linear system, high proportional gain degrades one aspect of the transient response by increasing settling time. It is possible to obtain the extra drive required to reduce steady-state and following error elsewhere. It must grow from somewhere, but stop growing once the output position follows the commanded position without error. This can be achieved using integral feedback or by integrating the error signal and adding that to drive. One problem with the application of integral feedback is that if the system is unable to completely eliminate error, due to a disturbance force for example, the integral continually grows with time until the output is released, by which time, it is too large to be a useful control signal. This is called integral wind-up and various techniques have been devised to avoid it. In the case of position control, it might also be associated with the steady-state error caused by friction. Once, integral wind-up overcomes the friction, the output can jolt suddenly, overshooting the target and probably making the situation worse.

Using three gains with effectively proportional, integral and derivative feedback, constitutes *PID compensation* and is popular in control systems in a range of industries. In this form of compensation, the integral and derivative components are called *ideal integral* and *ideal derivative* compensation because similar effects can be achieved using *lag* and *lead* networks. (Nise 2004, p. 503) The tachometer, discussed in section 2.2.3, can be used for ideal derivative compensation. Common in industry...

2.2.7 The Pulse Transfer Function

While the Laplace Transform is used on the differential equations governing analogue systems, digital systems are governed by difference equations, on which the *Z-Transform* can be used. The Z-Transform of a discrete signal f which starts at

time zero, and its inverse are defined by:

$$F(z) = \mathcal{Z}\{f\} = \sum_{n=0}^{\infty} f_n z^{-n} \tag{2.26}$$

where n is a sample number

$$f = \mathcal{Z}^{-1}\{F(z)\} = \frac{1}{2\pi j} \oint_C F(z) z^{n-1} \mathrm{d}z \tag{2.27}$$

where C is an anticlockwise contour encircling the origin

Again, there are tables listing the transforms of commonly encountered signal forms. If the values of the discrete signal f are formed from samples of a continuous signal $f(t)$ taken at regular time intervals T, then:

$$f_n = f(nT) \tag{2.28}$$

$$F(z) = \sum_{n=0}^{\infty} f(nT) z^{-n} \tag{2.29}$$

The *pulse transfer function* of a system is the Z-Transform corresponding to its normal transfer function. Nise (2004, pp. 799, 800) shows, as follows, how the Z-Transform of the output response signal of a digital system is equal to the product of the Z-Transform of the input signal and the system's pulse transfer function. Fig. 2.26 shows the notion of an "ideal sampler" which modulates the input signal with a train of impulses occurring at regular intervals. The sampler is ideal in the sense that the impulses are infinitely narrow and represented by Dirac Delta functions. This kind of impulse is a mathematical construct and is physically unachievable by a practical A/D converter, however, when the ideal sampler model is used in conjunction with a certain model for the D/A converter, the result is an accurate model of the two practical devices working together. Thus, the pulse transfer function of a computer and plant following an A/D converter is only useful if it contains a component for the D/A converter, which is likely to be included in the system anyway.

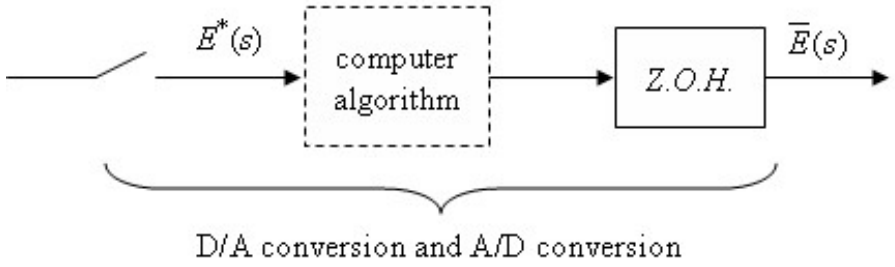


Fig. 2.26: The ideal sampler in conjunction with a Z.O.H model together accurately model the A/D converter and D/A converter working together.

An impulse-modulated input signal $r^*(t)$ is:

$$r^*(t) = \sum_{n=0}^{\infty} r(nT)\delta(t - nT) \tag{2.30}$$

where T is the period between sampling instants

Taking the Laplace Transform of Eqn. 2.30:

$$R^*(s) = \sum_{n=0}^{\infty} r(nT)e^{-nTs} \tag{2.31}$$

In the frequency domain, for a system transfer function $G(s)$, the output response $C(s)$ is then given by:

$$C(s) = R^*(s)G(s) \tag{2.32}$$

$$= \sum_{n=0}^{\infty} r(nT)e^{-nTs}G(s) \tag{2.33}$$

Using the Time-Shift Theorem for Laplace Transforms, the output signal back as a function of time is:

$$c(t) = \sum_{n=0}^{\infty} r(nT)g(t - nT) \tag{2.34}$$

The output signal at instant kT is:

$$c(kT) = \sum_{n=0}^{\infty} r(nT)g(kT - nT) \tag{2.35}$$

The Z-Transform of the output signal is:

$$C(z) = \sum_{k=0}^{\infty} c(kT)z^{-k}$$

$$= \sum_{k=0}^{\infty} \left(\sum_{n=0}^{\infty} \left(r(nT)g(kT - nT) \right) z^{-k} \right) \tag{2.36}$$

Letting $m = k - n$:

$$C(z) = \sum_{m+n=0}^{\infty} \left(\sum_{n=0}^{\infty} \left(r(nT)g(mT) \right) z^{-(m+n)} \right)$$

$$= \sum_{m+n=0}^{\infty} \left(g(mT)z^{-m} \right) \sum_{n=0}^{\infty} \left(r(nT)z^{-n} \right) \tag{2.37}$$

But $g(mT) = 0$ for all values of m less than zero. Therefore:

$$C(z) = \sum_{m=0}^{\infty} \left(g(mT)z^{-m} \right) \sum_{n=0}^{\infty} \left(r(nT)z^{-n} \right)$$

$$= R(z)G(z) \tag{2.38}$$

Also from Eqns. 2.32 and 2.38:

$$\mathcal{Z}\{R^*(s)G(s)\} = R(z)G(z) \tag{2.39}$$

Using Eqn. 2.39, the connection between the Z-Transform and digital control systems will now be shown. "In general, it is undesirable to apply a signal in sampled form, such as a train of narrow rectangular pulses, to a plant, because of the high-frequency components inherently present in that signal. Therefore a data reconstruction device, called a data hold, is inserted into the system directly following the sampler" (Phillips & Nagle 1990, p. 82) The data hold is a D/A converter. As shown in Fig. 2.26, D/A conversion takes place after the sampler has performed A/D conversion on the input voltage signal $e(t)$ and the discrete binary number representing the sampled voltage has been operated on by the computer algorithm. The input signal is labeled $e(t)$ because, as mentioned in section 2.18, it is the amount of error used to drive the system. The most common form of data hold, which is also the easiest to analyse is the *zero-order hold* . The "zero-order" refers

to the order of the polynomial which gives shape of the output between sampling instants. As shown in Fig. 4.38, the output voltage of a digital controller using a zero-order D/A conversion is constant between sampling instants and is equal to the output at the preceding sampling instant.

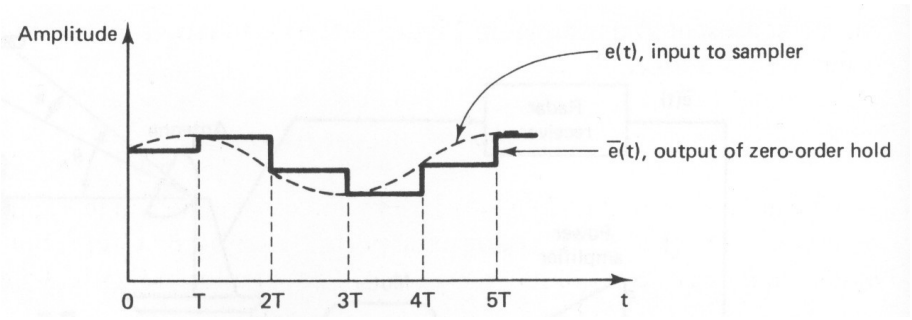


Fig. 2.27: Input and output signals of sampler/data hold. From *Digital Control System Analysis and Design* by Phillips and Nagle.

This figure shows the case where a computer is not used or where the computer algorithm performs the trivial function of passing the A/D output straight to the D/A input with unity gain. As shown by Phillips & Nagle (1990, p. 82), the staircase-shaped output signal, as a function of time, can be formed from a series of Heaviside step functions. The Heaviside step function is defined by:

$$u_c = u(t-c) = \begin{cases} 0 & t < c \\ 1 & t \geqslant c \end{cases} \tag{2.40}$$

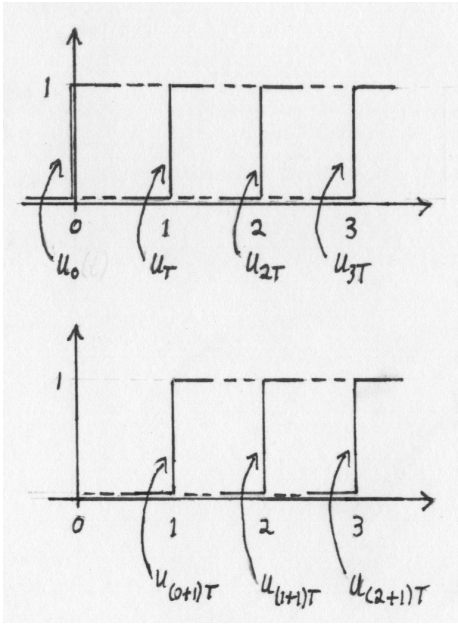


Fig. 2.28: Two series of step functions used to reconstruct the the output signal from the D/A converter.

From Figs. 4.38 and 2.28 this output signal function can be seen to be:

$$\begin{aligned}\bar{e}(t) = e(0)\big(u(t) - u(t-T)\big) + e(T)\big(u(t-T) - u(t-2T)\big) \\ + e(2T)\big(u(t-2T) - u(t-3T)\big) + \ldots\end{aligned} \tag{2.41}$$

Taking the Laplace transform of Eqn. 2.41:

$$\begin{aligned}\bar{E}(s) &= \left(\frac{1-e^{-Ts}}{s}\right)\left(e(0) + e(T)e^{-Ts} + e(2T)e^{-2Ts} + \ldots\right) \\ &= \sum_{n=0}^{\infty}\big(e(nT)e^{-nTs}\big)\left(\frac{1-e^{-Ts}}{s}\right) \qquad (2.42) \\ &= E^*(s)\left(\frac{1-e^{-Ts}}{s}\right) \qquad (2.43)\end{aligned}$$

The first factor on the RHS of Eqn. 2.42 can be seen to be the error signal if it were impulse modulated. The second factor is the model for the D/A converter mentioned earlier. If a transfer function is known for the plant it can be combined with this transfer function for the D/A converter to form a combined data hold and

plant transfer function $G_{\mathrm{HP}}(s)$.

$$G_{\mathrm{HP}}(s) = \frac{1 - e^{-Ts}}{s} G_{\mathrm{P}}(s) \tag{2.44}$$

The output response of the plant is:

$$C(s) = \bar{E}(s) G_{\mathrm{P}}(s) \tag{2.45}$$

$$= E^*(s) G_{\mathrm{HP}}(s) \tag{2.46}$$

From Eqns. 2.46 and 2.39:

$$C(z) = \mathcal{Z}\{E^*(s) G_{\mathrm{HP}}(s)\}$$

$$= E(z) G_{\mathrm{HP}}(z) \tag{2.47}$$

From Eqns. 2.31 and 2.29, it can be seen that the Z-Transform of a signal, can be found by replacing the term e^{-Ts} by z in its Laplace Transform when it is impulse modulated. Therefore, the same can be done with this term, in $G_{\mathrm{HP}}(z)$.

$$\begin{aligned} G_{\mathrm{HP}}(z) &= \mathcal{Z}\{G_{\mathrm{HP}}(s)\} \\ &= \mathcal{Z}\left\{(1 - e^{-Ts}) \frac{G_{\mathrm{P}}(s)}{s}\right\} \\ &= \mathcal{Z}\left\{\frac{G_{\mathrm{P}}(s)}{s}\right\} - \mathcal{Z}\left\{e^{-Ts} \frac{G_{\mathrm{P}}(s)}{s}\right\} \\ &= \mathcal{Z}\left\{\frac{G_{\mathrm{P}}(s)}{s}\right\} - z^{-1} \mathcal{Z}\left\{\frac{G_{\mathrm{P}}(s)}{s}\right\} \\ &= (1 - z^{-1}) \mathcal{Z}\left\{\frac{G_{\mathrm{P}}(s)}{s}\right\} \end{aligned} \tag{2.48}$$

There is one property of the Z-transform which makes it so useful in analysing digital control systems. For analogue systems, multiplying by s in the frequency domain corresponds to the operation of differentiation and dividing by s corresponds to integration. For digital systems, multiplying by z in the frequency domain corresponds to the operation of advancing to the next signal sample and dividing by z corresponds to delaying until the previous sample. This changes the problem of solving difference equations to one of solving an algebraic equation in z.

The difference equations used to guide the output response of the system are implemented in the computer algorithm. With a computer included in the system, the relationship between the error signal and the output becomes:

$$C(z) = E(z)G_{\mathrm{C}}(z)G_{\mathrm{HP}}(z) \tag{2.49}$$

2.3 The Mechanical Unit Model

There are a few ways in which to obtain a model, such as a transfer function, of a system for the purpose of designing a controller. With no access to the manufacturer's specifications for the servo system, various methods have been considered. One such method is to combine models which describe the behaviour of components of whole system. In this way, it can be shown that the transfer function between motor shaft position and voltage applied to the terminals of a DC motor and load is:

$$\frac{\theta_{\mathrm{L}}(s)}{V_{\mathrm{t}}(s)} = \frac{\frac{N_1}{N_2}k_{\mathrm{T}}}{s((J_{\mathrm{m}}s + D_{\mathrm{m}})(L_{\mathrm{a}}s + R_{\mathrm{a}}) + k_{\mathrm{T}}k_{\mathrm{v}})} \tag{2.50}$$

where: θ_{L} is the angular position of the output shaft

V_{t} is the voltage applied to the motor terminals

k_{T} is the torque constant of the motor

J_{m} is the total equivalent rotational inertia of the system reflected back to the motor shaft

D_{m} is a damping coefficient for the total equivalent drag and friction of the system reflected back to the motor shaft

L_{a} is the inductance of the armature winding of the motor

R_{a} is the resistance of the armature winding of the motor

k_{v} is the voltage constant of the motor

N_1/N_2 is the gear ratio

The DC motor and load form a third-order system. If the inductance of the motor's armature winding is negligible, the system can be modelled as a second-order system

and the transfer function reduces to:

$$\frac{\theta_{\mathrm{L}}(s)}{V_{\mathrm{t}}(s)} = \frac{\left(\frac{N_1}{N_2}\right)\left(\frac{k_{\mathrm{t}}}{R_{\mathrm{a}}J_{\mathrm{m}}}\right)}{s\left(s + \frac{1}{J_{\mathrm{m}}}\left(D_{\mathrm{m}} + \frac{k_{\mathrm{T}}k_{\mathrm{v}}}{R_{\mathrm{a}}}\right)\right)} \tag{2.51}$$

If this were the case, the quantities k_{v} and $k_{\mathrm{T}}/R_{\mathrm{a}}$ could be determined by performing a dynamometer test as discussed in section 2.2.1. These, together with the transient speed response time constant, could be used to determine the inertia J_{m}. Also, the transfer function could be obtained from experimental Bode plots of the system's frequency response more easily if the inductance is negligible.

For the servo system under consideration, dynamometer testing has not been a practical option and the inductance of the motor's armature winding does have a significant effect. According to (Kevin 2004), the transfer function specified for the servomechanism of the Feedback Instruments Servo Fundamentals Trainer together with input amplifiers is:

$$\frac{\theta_{\mathrm{L}}(s)}{V_{\mathrm{t}}(s)} = \frac{20K_1}{s(s+1.5)(s+10)} \tag{2.52}$$

This transfer function will next be used to obtain a pulse transfer function when the digital controller is interfaced to the motor.

2.4 The Digital System with Position Feedback

In the previous section, a transfer function modelling the relationship between the angular position of the output shaft and the voltage applied to the motor terminals was provided. When the digital controller is interfaced to the motor, this voltage is no longer an amplified version of a smooth analogue signal, but an amplified version of the "staircase" signal leaving the D/A converter. A diagram of the system with a digital controller and shaft position feedback is shown in Fig. 2.29.

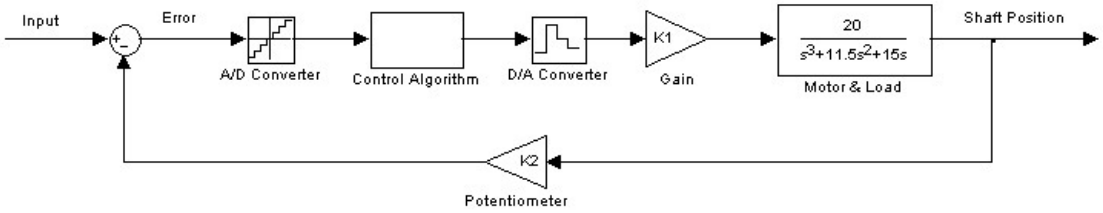


Fig. 2.29: The system with a digital controller and position feedback.

The pulse transfer function $G_{\mathrm{HP}}(z)$ of the system will now be determined.

$$G_{\mathrm{HP}}(z) = (1 - z^{-1})\mathcal{Z}\left\{\frac{G_{\mathrm{P}}(s)}{s}\right\}$$

$$\begin{aligned}\frac{G_{\mathrm{P}}(s)}{s} &= \frac{20K}{s^2(s+1.5)(s+10)} \\ &= \frac{a}{s+1.5} + \frac{b}{s+10} + \frac{c}{s^2} + \frac{d}{s} \end{aligned} \tag{2.53}$$

$$a = \left.\frac{20K_1}{s^2(s+10)}\right|_{s\to -1.5} = \frac{160}{153}K_1 \tag{2.54}$$

$$b = \left.\frac{20K_1}{s^2(s+1.5)}\right|_{s\to -10} = -\frac{2}{85}K_1 \tag{2.55}$$

$$c = \left.\frac{20K_1}{(s+1.5)(s+10)}\right|_{s\to 0} = \frac{4}{3}K_1 \tag{2.56}$$

$$c = \left.\frac{-20K_1(2s+11.5)}{(s^2+11.5s+165)^2}\right|_{s\to 0} = -\frac{46}{45}K_1 \tag{2.57}$$

$$\frac{G_{\mathrm{P}}(s)}{s} = \frac{\frac{160}{153}K_1}{s+1.5} - \frac{\frac{2}{85}K_1}{s+10} + \frac{\frac{4}{3}K_1}{s^2} - \frac{\frac{46}{45}K_1}{s} \tag{2.58}$$

Letting $e^{-1.5T} = A$ and $e^{-10T} = B$:

$$\begin{aligned}\mathcal{Z}\left\{\frac{G_{\mathrm{P}}(s)}{s}\right\} &= \frac{\frac{160}{153}K_1}{1-Az^{-1}} - \frac{\frac{2}{85}K_1}{1-Bz^{-1}} + \frac{\frac{4}{3}K_1}{(1-z^{-1})^2} - \frac{\frac{46}{45}K_1}{1-z^{-1}} \\ &= \frac{uz^{-3} + vz^{-2} + wz^{-1}}{(1-Az^{-1})(1-Bz^{-1})(1-z^{-1})^2} \end{aligned} \tag{2.59}$$

$$G_{\mathrm{HP}}(z) = \frac{uz^{-3} + vz^{-2} + wz^{-1}}{-ABz^{-3} + (AB + A + B)z^{-2} - (A + B + 1)z^{-1} + 1} \tag{2.60}$$

The closed-loop transfer function for the system with no control algorithm in place is then:

$$\begin{aligned}
\frac{\theta_{\mathrm{L}}(s)}{V_{\mathrm{t}}(s)} &= \frac{G_{\mathrm{HP}}(z)}{1 + K_2 G_{\mathrm{HP}}(z)} \\
&= \frac{\dfrac{\text{numerator of } G_{\mathrm{HP}}}{\text{denominator of } G_{\mathrm{HP}}}}{1 + K_2 \dfrac{\text{numerator of } G_{\mathrm{HP}}}{\text{denominator of } G_{\mathrm{HP}}}} \\
&= \frac{\text{numerator of } G_{\mathrm{HP}}}{\text{denomenator of } G_{\mathrm{HP}} + K_2(\text{numerator of } G_{\mathrm{HP}})} \\
&= \frac{\beta_3 z^{-3} + \beta_2 z^{-2} + \beta_1 z^{-1} + \beta_0}{\alpha_3 z^{-3} + \alpha_2 z^{-2} + \alpha_1 z^{-1} + \alpha_0 + K_2(\beta_3 z^{-3} + \beta_2 z^{-2} + \beta_1 z^{-1} + \beta_0)} \\
&= \frac{\beta_3 z^{-3} + \beta_2 z^{-2} + \beta_1 z^{-1} + \beta_0}{\phi_3 z^{-3} + \phi_2 z^{-2} + \phi_1 z^{-1} + \phi_0}
\end{aligned} \tag{2.61}$$

From this transfer function, the output at each D/A instant, as a combination of previous input and output values, can be predicted.

$$\mathcal{Z}^{-1}\left\{\theta_{\mathrm{L}}(z)\phi(z)\right\} = \mathcal{Z}^{-1}\left\{V_{\mathrm{t}}(z)\beta(z)\right\} \tag{2.62}$$

$$\phi_3\theta_{n-3} + \phi_2\theta_{n-2} + \phi_1\theta_{n-1} + \phi_0\theta_n = \beta_3 V_{n-3} + \beta_2 V_{n-2} + \beta_1 V_{n-1} + \beta_0 V_n$$

$$\theta_n = \frac{\beta_3 V_{n-3} + \beta_2 V_{n-2} + \beta_1 V_{n-1} + \beta_0 V_n - \phi_3\theta_{n-3} - \phi_2\theta_{n-2} - \phi_1\theta_{n-1}}{\phi_0} \tag{2.63}$$

This iterative equation has been used in the *MATLAB* program *simulate.m*, found in Appendix B, in order to simulate the system. The following *MATLAB* script uses this program to plot the response of the system to a step input of 5 Volts for a forward path gain of 50 and a potentiometer gain of 5 Volts / 180° rotation.

```
T=0.0005;K1=50;K2=5/180;N=10/T;input=5*ones(1,N);instants=(1:N)*T;
output=simulate(input,T,K1,K2);
plot(instants,output)
xlabel('time (s)')
ylabel('shaft position (degrees)')
```

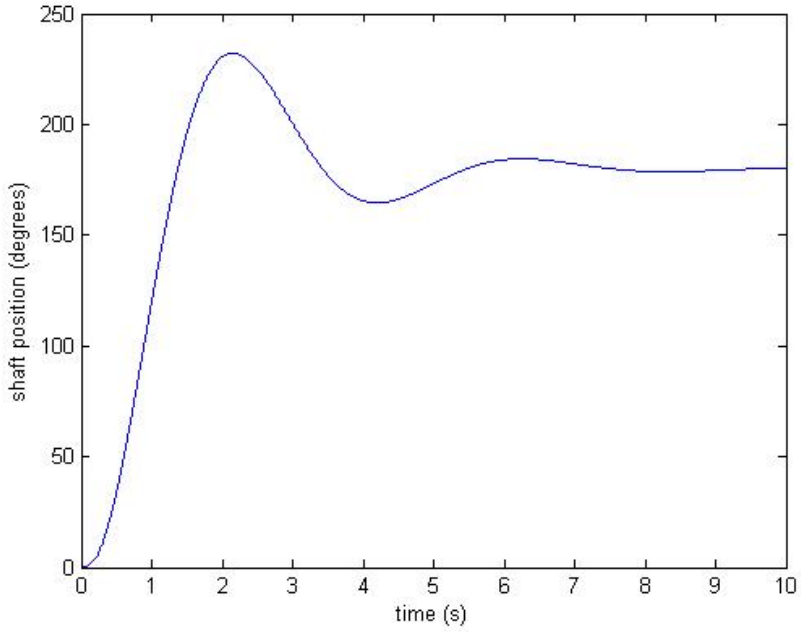


Fig. 2.30: Simulated response to a step input of 5 Volts for a forward path gain of 50 and a potentiometer gain of 5 Volts / 180^o rotation.

The response, shown in Fig. 2.30, is lot slower than any of those that were seen experimentally, indicating that the original model, provided in Eqn. 2.52 is not accurate.

2.5 Nonlinearities

A large part of control theory is based on the assumption of a linear system. This implies that if the input to the system, no matter how large, is doubled, for example, then the output will also double. In reality, the output will at some stage be limited by saturation of the drive amplifier. For the DC motor servo, this may be chosen according to rated motor voltage. Other nonlinearities, which cause the system to provide no additional output for extra input include *dead-band* and *backlash*. Dead-band in a DC motor occurs due to static friction, which stops the motor from

turning until a certain threshold voltage is reached. Backlash occurs in mechanisms with gears. It is due to the fact that, when changing direction, a small amount of rotation of an input gear must take place before the gears mesh and the output starts to rotate.

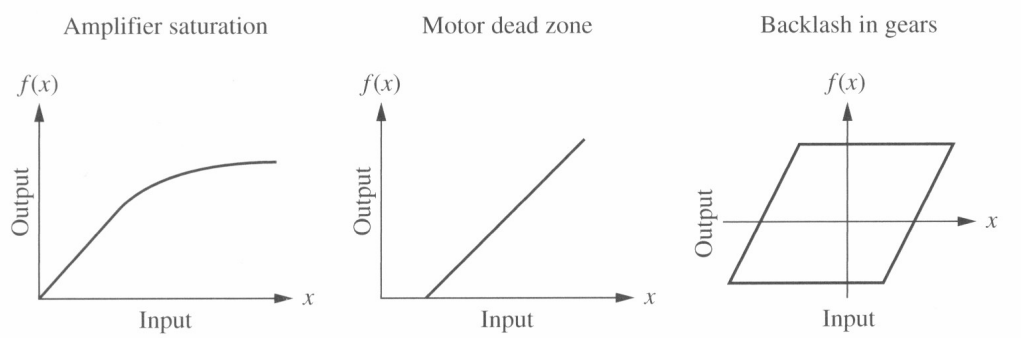


Fig. 2.31: Some nonlinearities present in physical systems. From *Control Systems Engineering* by Nise.

The nonlinearity of drive saturation need not be avoided by designing a feedback controller based on an analysis in the linear region of operation. It can be used to advantage. If, feedback gains are instead made large enough to meet speed and rigidity requirements, there may be significant overshoot when the position target is changed. By limiting the demand for drive in software, the point at which it saturates can be artificially lowered. This limits the speed at which the system approaches the target and prevents overshoot, while maintaining 'urgent' control action close to the target. A compromise must be made between this approach speed and the amount of gain close to the target.

2.6 Conclusion

A model of the system has been developed using principles from electrics, electrodynamics and mechanics. Like any model, it is not 100 percent accurate, but it

has been found to produce simulations that resemble the actual behaviour of the device. This aspect of the design, together with the concept of introducing non-linearity into the control strategy, highlights the value of digital control, which can be fine-tuned or completely changed in software rather than by the replacement of analogue components.

3 PID Control

3.1 Introduction

" . . . more than half of the industrial controllers in use today utilize PID or modified PID control schemes. Because most PID controllers are adjusted on-site, many different types of tuning rules have been proposed. . . The usefulness of PID controls lies in their general applicability to most control systems." (Ogata 2002, p. 681) Some important empirical tuning methods make PID control extremely useful when a mathematical model of the plant is not known. Defining the PID parameters in the software of a digital controller aids in this necessary process of tuning and makes possible extra capabilities such as on-line automatic tuning, bumpless switching (from manual operation to automatic operation) and gain scheduling. (Ogata 2002, p. 681)

3.2 Background

3.2.1 Digital PID Compensation

A/D converters with a reasonable resolution are readily available at low cost and there are often several of them included inside a microcontroller. However, the controller can still only work with digital signals, represented by binary numbers, so the operations performed by it, which make up the algorithm, must be basic arithmetic operations. In a digital implementation of PID compensation, there is no problem with the proportional gain because the controller can easily multiply a number by a constant, but the operations of integration and differentiation are calculus, not basic arithmetic so approximations to these are used.

Mathematically, the derivative of a function is defined by:

$$\frac{\mathrm{d}}{\mathrm{d}t}e(t) = \lim_{\Delta t \to 0} \frac{e(t+\Delta t) - e(t)}{\Delta t} \tag{3.1}$$

In a computer program, it can be approximated by:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}e(nT) &\approx \frac{e(nT) - e((n-1)T)}{T} \\ &= \frac{1}{T}(e(nT) - e((n-1)T)) \end{aligned} \tag{3.2}$$

where: T is the sampling period

n is the sample number

That is, the value of the error signal, measured at the previous A/D converter sampling instant, which has been stored in memory, is subtracted from the current value and the result is multiplied by $1/T$. This approximation consists of only a subtraction and a multiplication, both of which are even contained in the instruction sets of all modern microcontrollers. It is not exact, but with a high enough sampling frequency it is very accurate. Similarly, the integral of a function is defined by:

$$\int e(t)\mathrm{d}t = \lim_{\Delta t \to 0} \sum_{i=0}^{n-1} e_i \Delta t \tag{3.3}$$

In a digital controller, it can be approximated by a Riemann sum:

$$\begin{aligned} \int e(t)\mathrm{d}t &\approx \sum_{i=0}^{n-1} e_i T \\ &= T(e(0) + e(T) + e(2T) + e(3T) + \ldots + e((n-1)T) \end{aligned} \tag{3.4}$$

This approximation consists of addition and multiplication, however there is one problem. It would not be practical because memory locations for storing past values of the signal would quickly run out. For example, if the A/D converter was sampling at 1 kHz, after 1 second there would be 1000 numbers to have stored in memory to be retrieved and added up. This is not really a problem because the integral of the signal at the previous sampling instant used 999 of these numbers. This previous

value of the integral mixed together with proportional and derivative components is contained in the previous value of the controller output. If the output value is stored to memory at each instant, it can be used for calculation and then overwritten by the next value, and only a very small amount of memory is actually required.

These approximations to differentiation and integration yield difference equations instead of differential equations. With the three proportional, derivative and integral gains K_p, K_d and K_i, the controller output m, is given by:

$$m_n = K_\mathrm{p} e_n + \frac{K_\mathrm{d}}{T}(e_n - e_{n-1}) + K_\mathrm{i} T(e_0 + e_1 + \ldots + e_{n-1}) \tag{3.5}$$

The control algorithm using previous values of controller output is found as follows:

$$m_{n-1} = K_\mathrm{p} e_{n-1} + \frac{K_\mathrm{d}}{T}(e_{n-1} - e_{n-2}) + K_\mathrm{i} T(e_0 + e_1 + \ldots + e_{n-2})$$

$$m_n - m_{n-1} = K_\mathrm{p}(e_n - e_{n-1}) + \frac{K_\mathrm{d}}{T}(e_n - 2e_{n-1} + e_{n-2}) + K_\mathrm{i} T e_{n-1}$$

$$m_n = \left(K_\mathrm{p} + \frac{K_\mathrm{d}}{T}\right) e_n + \left(K_\mathrm{i} T - \frac{2K_\mathrm{d}}{T} - K_\mathrm{p}\right) e_{n-1} + \frac{K_\mathrm{d}}{T} e_{n-2} + m_{n-1}$$

$$m_n = q_0 e_n + q_1 e_{n-1} + q_2 e_{n-2} + m_{n-1} \tag{3.6}$$

$$\text{where:} \quad q_0 = K_\mathrm{p} + \frac{K_\mathrm{d}}{T}$$
$$q_1 = K_\mathrm{i} T - \frac{2K_\mathrm{d}}{T} - K_\mathrm{p}$$
$$q_2 = \frac{K_\mathrm{d}}{T}$$

Eqn. 3.6 provides a generic PID controller algorithm that can be used in any system. All that is required is to tune its three parameters for that system. In section 2.2.7, it was mentioned that the Z-Transform has the property that division by z in the frequency domain corresponded to a delay in the time domain. Using this property, the transfer function describing the PID control algorithm is obtained.

$$M(z) = q_0 E(z) + q_1 z^{-1} E(z) + q_2 z^{-2} E(z) + z^{-1} M(z) \tag{3.7}$$

$$\frac{M(z)}{E(z)} = \frac{q_0 + q_1 z^{-1} + q_2 z^{-2}}{1 - z^{-1}} \tag{3.8}$$

Various methods for tuning PID controllers have been established. The *Ziegler-Nichols step response* and *Ziegler-Nichols Ultimate Sensitivity* methods are based on the design experience of their inventors. *Genetic Algorithms* are an optimisation approach using the concept of evolution. *Steepest Descent Optimisation* is a numerical optimisation method.

3.2.2 Steepest Descent Optimisation

There are several performance criteria against which to optimise the response of the system. For a given sampling frequency, the performance (S) is a function only of the three PID controller gains K_p, K_i and K_d. Using a test step of size δK in the direction of any of the gain axes, three partial derivatives can be estimated:

$$\frac{\partial S}{\partial K_\mathrm{p}} \approx \frac{S(K_\mathrm{p} + \delta K, K_\mathrm{i}, K_\mathrm{d}) - S(K_\mathrm{p}, K_\mathrm{i}, K_\mathrm{d})}{\delta K} \tag{3.9}$$

$$\frac{\partial S}{\partial K_\mathrm{i}} \approx \frac{S(K_\mathrm{p}, K_\mathrm{i} + \delta K, K_\mathrm{d}) - S(K_\mathrm{p}, K_\mathrm{i}, K_\mathrm{d})}{\delta K} \tag{3.10}$$

$$\frac{\partial S}{\partial K_\mathrm{d}} \approx \frac{S(K_\mathrm{p}, K_\mathrm{i}, K_\mathrm{d} + \delta K) - S(K_\mathrm{p}, K_\mathrm{i}, K_\mathrm{d})}{\delta K} \tag{3.11}$$

The gradient vector of the performance function is then given by:

$$\nabla S(K_\mathrm{p}, K_\mathrm{i}, K_\mathrm{d}) = \left(\frac{\partial S}{\partial K_\mathrm{p}}, \frac{\partial S}{\partial K_\mathrm{i}}, \frac{\partial S}{\partial K_\mathrm{d}} \right) \tag{3.12}$$

The derivative in any direction with unit vector $\boldsymbol{u}$ is then given by:

$$D_u = \nabla S \cdot \boldsymbol{u} \tag{3.13}$$

$$= |\nabla S| \cos(\theta) \tag{3.14}$$

The derivative is greatest in the direction of the gradient vector, i.e.:

$$\boldsymbol{u} = \frac{\nabla S}{|\nabla S|}$$

$$= \left(\frac{\left(\frac{\partial S}{\partial K_\mathrm{p}}\right)}{\Delta}, \frac{\left(\frac{\partial S}{\partial K_\mathrm{i}}\right)}{\Delta}, \frac{\left(\frac{\partial S}{\partial K_\mathrm{d}}\right)}{\Delta} \right) \tag{3.15}$$

$$\text{where } \Delta = \sqrt{\left(\frac{\partial S}{\partial K_\mathrm{p}}\right)^2 + \left(\frac{\partial S}{\partial K_\mathrm{i}}\right)^2 + \left(\frac{\partial S}{\partial K_\mathrm{d}}\right)^2}$$

Small steps can then be used to track along the performance function in the direction of greatest change, i.e. in the direction of the gradient vector. So that the size of the steps does not change with the changing magnitude of the gradient vector, the steps in each of the gains is normalised to this magnitude, giving an overall step equal to the product of the unit gradient vector and an arbitrary step size constant. Using a step size constant of γ:

$$K_\mathrm{p} \text{ step} = \gamma u_{K_\mathrm{p}}$$

$$= \left(\frac{\gamma}{\Delta}\right) \frac{\partial S}{\partial K_\mathrm{p}} \tag{3.16}$$

$$K_\mathrm{i} \text{ step} = \left(\frac{\gamma}{\Delta}\right) \frac{\partial S}{\partial K_\mathrm{i}} \tag{3.17}$$

$$K_\mathrm{d} \text{ step} = \left(\frac{\gamma}{\Delta}\right) \frac{\partial S}{\partial K_\mathrm{d}} \tag{3.18}$$

In order to optimise the controller gains by tracking downwards until the performance function is minimised, the steps must be in the opposite direction to the gradient vector. The values of the gains along this path are given by:

$$K_\mathrm{p}(k+1) = K_\mathrm{p}(k) - \left(\frac{\gamma}{\Delta}\right) \frac{\partial S}{\partial K_\mathrm{p}} \tag{3.19}$$

$$K_\mathrm{i}(k+1) = K_\mathrm{i}(k) - \left(\frac{\gamma}{\Delta}\right) \frac{\partial S}{\partial K_\mathrm{i}} \tag{3.20}$$

$$K_\mathrm{d}(k+1) = K_\mathrm{d}(k) - \left(\frac{\gamma}{\Delta}\right) \frac{\partial S}{\partial K_\mathrm{d}} \tag{3.21}$$

Eqns. 3.19 to 3.21 form the steepest descent optimisation algorithm. Using this algorithm, the set of PID gains will converge on the nearest local optimum set. Unfortunately, it is likely that this is not the global optimum yielding a response that is very best response possible.

3.3 Simulation

In Section 2.4, a pulse transfer function was obtained for the system with no control algorithm in place. Now it will be combined with that for the PID controller given in Eqn. 3.8, in order to simulate the system with PID control. The open loop transfer function is now:

$$G_{\mathrm{C}}(z)G_{\mathrm{HP}}(z) = \left(\frac{q_0 + q_1 z^{-1} + q_2 z^{-2}}{1 - z^{-1}}\right)\left(\frac{uz^{-3} + vz^{-2} + wz^{-1}}{(1 - Az^{-1})(1 - Bz^{-1})(1 - z^{-1})}\right)$$

$$= \frac{\left(\begin{array}{c} q_2 u z^{-5} \\ +(q_1 u + q_2 v)z^{-4} \\ +(q_0 u + q_1 v + q_2 w)z^{-3} \\ +(q_0 v + q_1 w)z^{-2} \\ +q_0 w z^{-1} \end{array}\right)}{\left(\begin{array}{c} ABz^{-4} - (2AB + A + B)z^{-3} + (AB + 2A + 2B + 1)z^{-2} \\ -(A + B + 2)z^{-1} + 1 \end{array}\right)} \tag{3.22}$$

The closed loop transfer function is then:

$$\frac{\theta_{\mathrm{L}}(s)}{V_{\mathrm{t}}(s)} = \frac{G_{\mathrm{C}}(z)G_{\mathrm{HP}}(z)}{1 + K_2 G_{\mathrm{C}}(z)G_{\mathrm{HP}}(z)}$$

$$= \frac{\dfrac{\text{numerator of } G_{\mathrm{C}}G_{\mathrm{HP}}}{\text{denominator of } G_{\mathrm{C}}G_{\mathrm{HP}}}}{1 + K_2 \dfrac{\text{numerator of } G_{\mathrm{C}}G_{\mathrm{HP}}}{\text{denominator of } G_{\mathrm{C}}G_{\mathrm{HP}}}}$$

$$= \frac{\text{numerator of } G_{\mathrm{C}}G_{\mathrm{HP}}}{\text{denomenator of } G_{\mathrm{C}}G_{\mathrm{HP}} + K_2(\text{numerator of } G_{\mathrm{C}}G_{\mathrm{HP}})}$$

$$= \frac{\beta_5 z^{-5} + \beta_4 z^{-4} + \beta_3 z^{-3} + \beta_2 z^{-2} + \beta_1 z^{-1} + \beta_0}{\left(\begin{array}{c} \alpha_4 z^{-4} + \alpha_3 z^{-3} + \alpha_2 z^{-2} + \alpha_1 z^{-1} + \alpha_0 \\ +K_2(\beta_5 z^{-5} + \beta_4 z^{-4} + \beta_3 z^{-3} + \beta_2 z^{-2} + \beta_1 z^{-1} + \beta_0) \end{array}\right)}$$

$$= \frac{\beta_5 z^{-5} + \beta_4 z^{-4} + \beta_3 z^{-3} + \beta_2 z^{-2} + \beta_1 z^{-1} + \beta_0}{\phi_5 z^{-5} + \phi_4 z^{-4} + \phi_3 z^{-3} + \phi_2 z^{-2} + \phi_1 z^{-1} + \phi_0} \tag{3.23}$$

From this transfer function, the output at each D/A instant, as a combination of previous input and output values, can be predicted.

$$\mathcal{Z}^{-1}\{\theta_{\mathrm{L}}(z)\phi(z)\} = \mathcal{Z}^{-1}\{V_{\mathrm{t}}(z)\beta(z)\} \tag{3.24}$$

$$\phi_5\theta_{n-5} + \phi_4\theta_{n-4} + \phi_3\theta_{n-3} + \phi_2\theta_{n-2} + \phi_1\theta_{n-1} + \phi_0\theta_n$$
$$= \beta_5 V_{n-5} + \beta_4 V_{n-4} + \beta_3 V_{n-3} + \beta_2 V_{n-2} + \beta_1 V_{n-1} + \beta_0 V_n$$

$$\theta_n = \frac{\left(\begin{array}{c} \beta_5 V_{n-5} + \beta_4 V_{n-4} + \beta_3 V_{n-3} + \beta_2 V_{n-2} + \beta_1 V_{n-1} + \beta_0 V_n \\ -\phi_5\theta_{n-5} - \phi_4\theta_{n-4} - \phi_3\theta_{n-3} - \phi_2\theta_{n-2} - \phi_1\theta_{n-1} \end{array} \right)}{\phi_0} \tag{3.25}$$

Eqn. 3.25 has been implemented in the *MATLAB* program *simulatePID.m*, found in Appendix C, as an iterative equation for simulating the response of the system with a PID controller in place. The following *MATLAB* script uses the program to plot the 5 Volt step response for arbitrary proportional, integral and derivative gains all equal to 1, as shown in Fig. 3.1.

```
T=0.0005;K1=50;K2=5/180;N=10/T;input=5*ones(1,N);instants=(1:N)*T;
output=simulatePID(input,T,K1,K2,1,1,1);
plot(instants,output)
xlabel('time (s)')
ylabel('shaft position (degrees)')
```

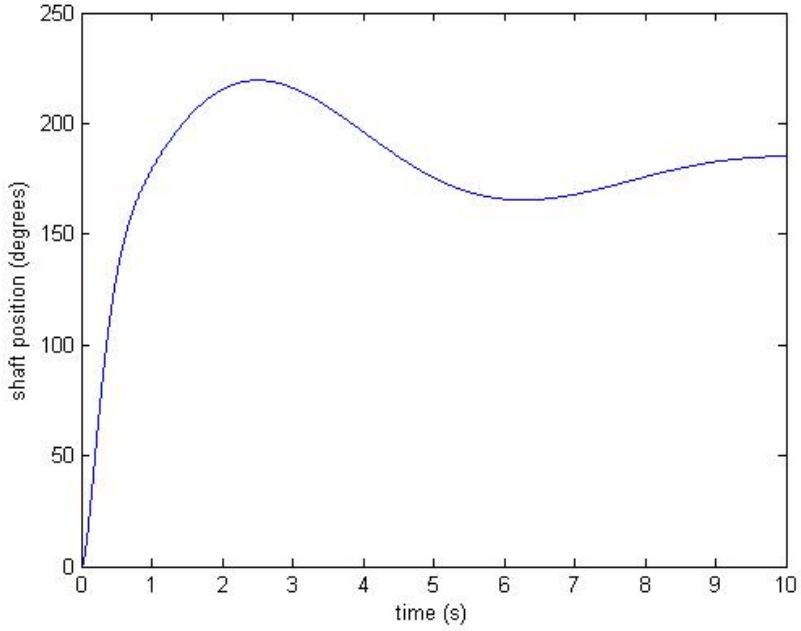


Fig. 3.1: Simulated 5 Volt step response with arbitrary controller gains $K_\mathrm{p} = 1$, $K_\mathrm{i} = 1$ and $K_\mathrm{d} = 1$.

3.4 PID Controller Tuning

In tuning a PID controller, the main objectives are usually quick response, minimal overshoot and minimal steady-state error. Alternatively, in some applications, power consumption might be a more important issue. The Steepest Descent Optimisation technique, discussed in Section 3.2.2 has been used to optimize the controller gains with respect to the *Integral of Absolute Error* (IAE) performance criterion, defined by:

$$IAE = \sum |e_n| \tag{3.26}$$

Against this criterion, the optimisation procedure attempts to minimise response time and steady-state error. The following *MATLAB* script uses the program *optimise.m*, found in Appendix D. This program uses the program *simulatePID.m* as a subroutine for simulating the response of the system for a set of controller gains. It then evaluates the performance and adjusts the gains according to the optimisation algorithm and then repeats the process until the performance values converge at a local minimum.

```
T=0.0005;K1=50;K2=5/180;N=6/T;input=5*ones(1,N);instants=(1:N)*T;
[Kp,Ki,Kd]=optimise(input,T,K1,K2,0.1,1,1,1);
```

The starting values for the gains were arbitrarily chosen to all equal 1, giving the same response as shown in Fig. 3.1. This response is the first blue curve in Fig. 3.2. The purple curve is the response of the system with no controller algorithm in place. It can be seen that the final response obtained is a significant improvement. It is interesting that the optimisation algorithm gradually increased the proportional and derivative gains while decreasing the integral gain to zero. The resulting, slightly negative integral gain will be discarded because it would lead to instability. Any negative integral is, for example, an attempt to reduce a positive position error with negative motor drive, which will not work. The final response with gains $K_\mathrm{p} = 2.75$, $K_\mathrm{i} = 0$ and $K_\mathrm{d} = 2.4$ is shown in Fig. 3.3.

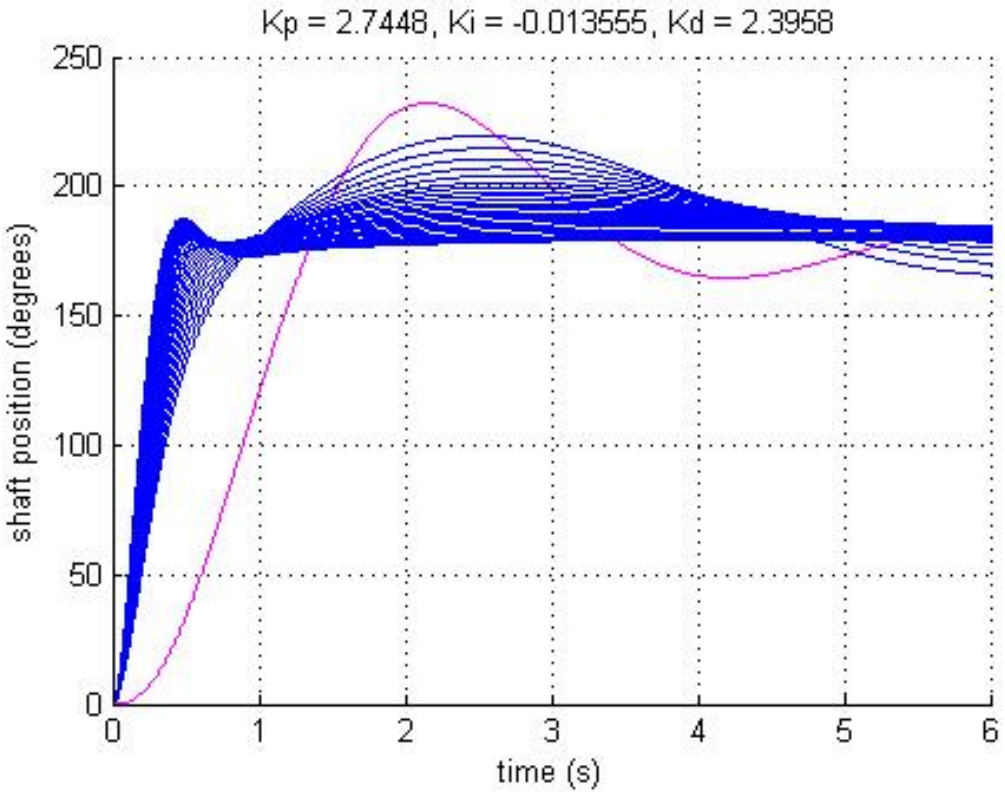


Fig. 3.2: System responses for each controller tuning iteration using the Steepest Descent Optimisation algorithm.

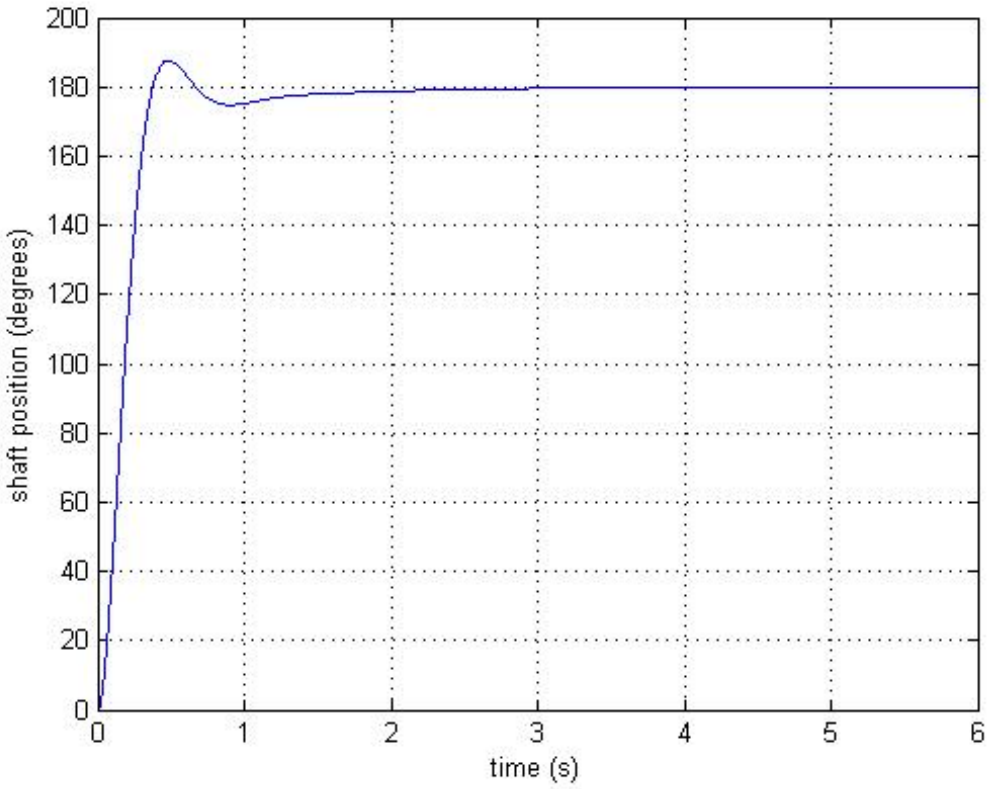


Fig. 3.3: Final response with gains $K_\mathrm{p} = 2.75$, $K_\mathrm{i} = 0$ and $K_\mathrm{d} = 2.4$.

Fig. 3.4 shows the results from another perspective. The curves are the error signals, without any amplification, resulting from each response. Fig. 3.5 shows the descent down the performance function. Fig. 3.6 shows the various gradients encountered along the particular path taken by the algorithm.

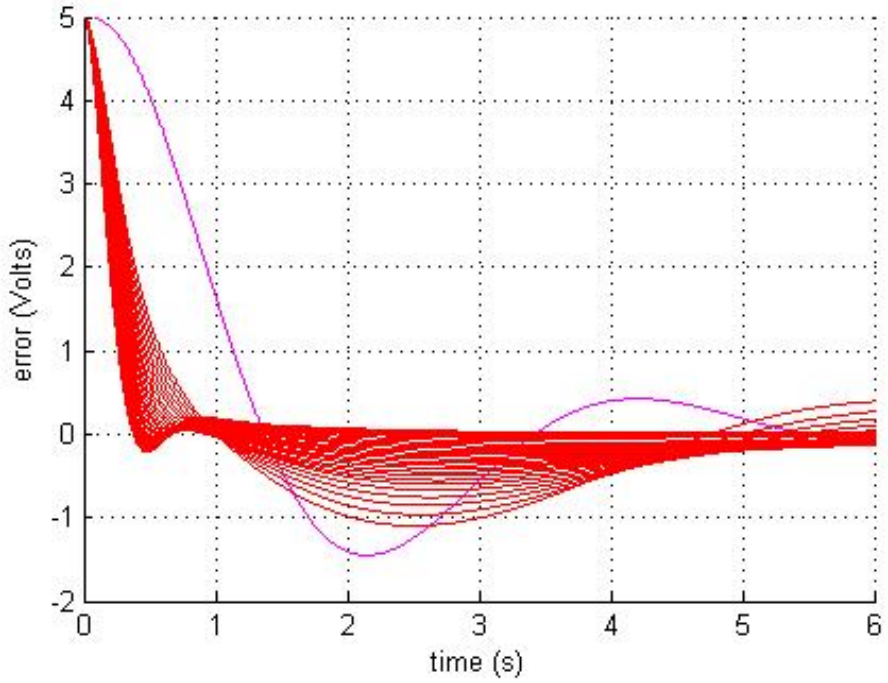


Fig. 3.4: Error signals for each controller tuning iteration.

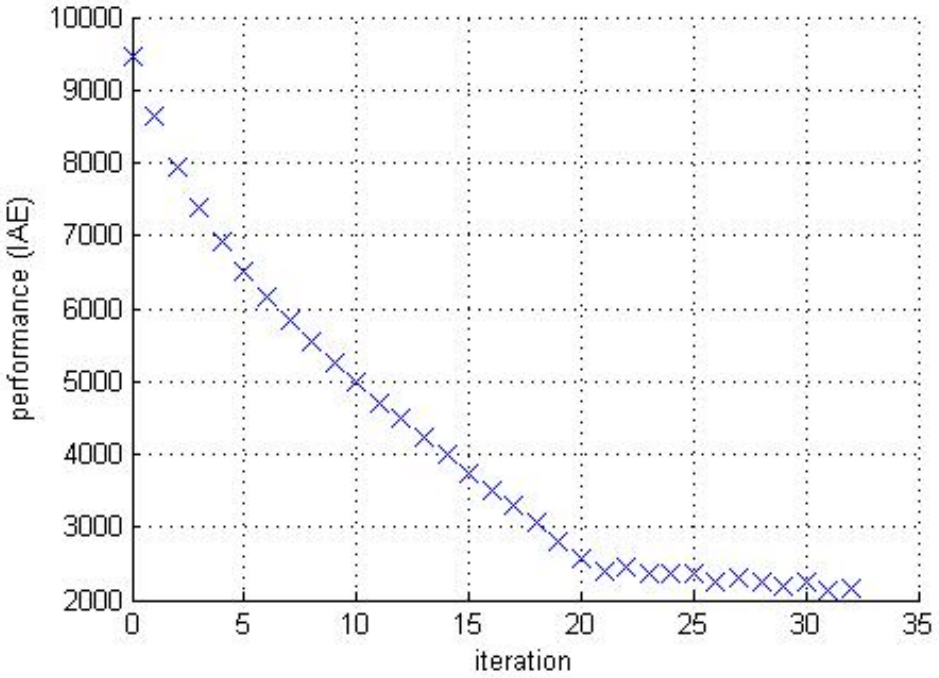


Fig. 3.5: System performance for each controller tuning iteration.

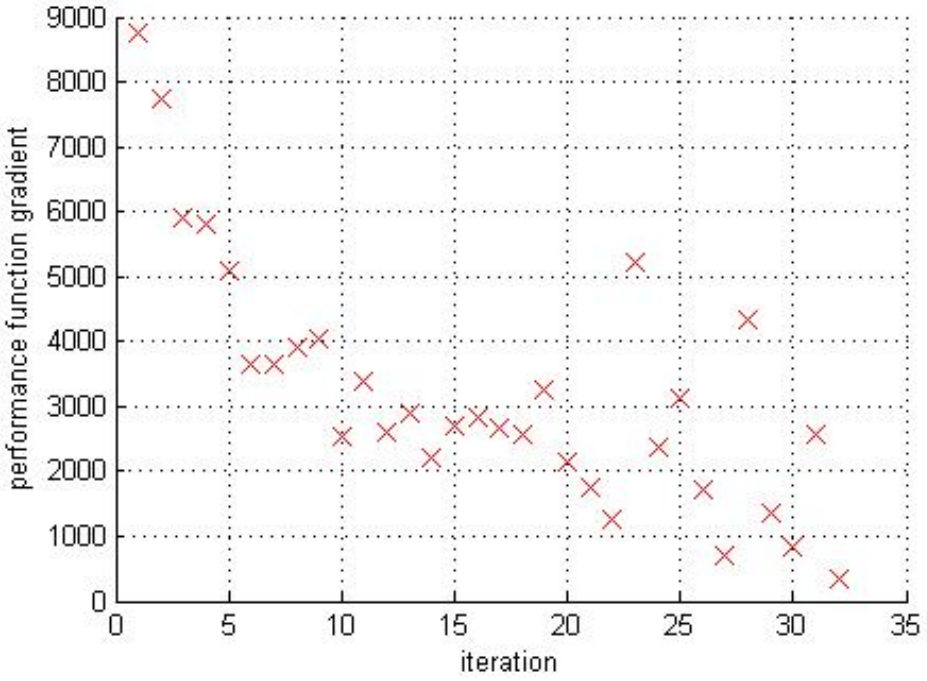


Fig. 3.6: Performance function gradient for each controller tuning iteration.

3.5 Experimental Results

The digital unit in Figs. 3.7 and 3.8 was connected to a PC via the USB interface shown in Fig 3.9. The error signal used by the controller is calculated from A/D converter measurements of the potentiometer voltage and input voltage, multiplexed through a single converter channel. The PID control tutorial in the Feedback Instruments software allows manipulation of the controller gains from the PC and gives the option of displaying shaft position, error, derivative of error or integral of error. The input can be switched between two terminals, which are supposed to be for connection to the triangular and square wave sources. Instead, the terminal for the connection to the square wave source was connected to the manual step input switch through the variable gain amplifier, so that positive and negative steps of approximately 5 volts could be applied.

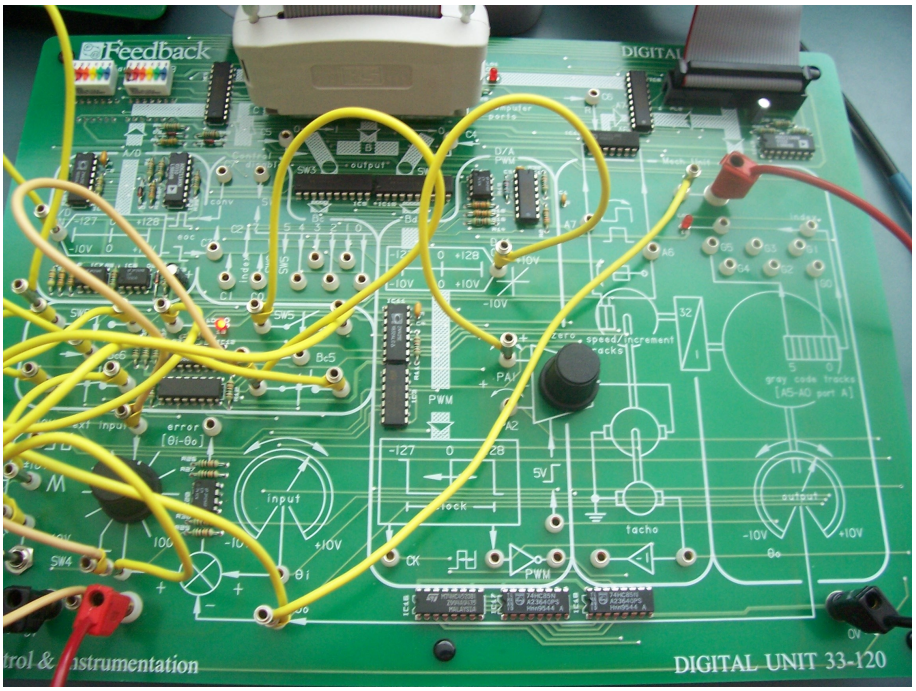


Fig. 3.7: The digital unit with jumper lead connections for multiplexed A/D conversion of the potentiometer voltage and input voltage.

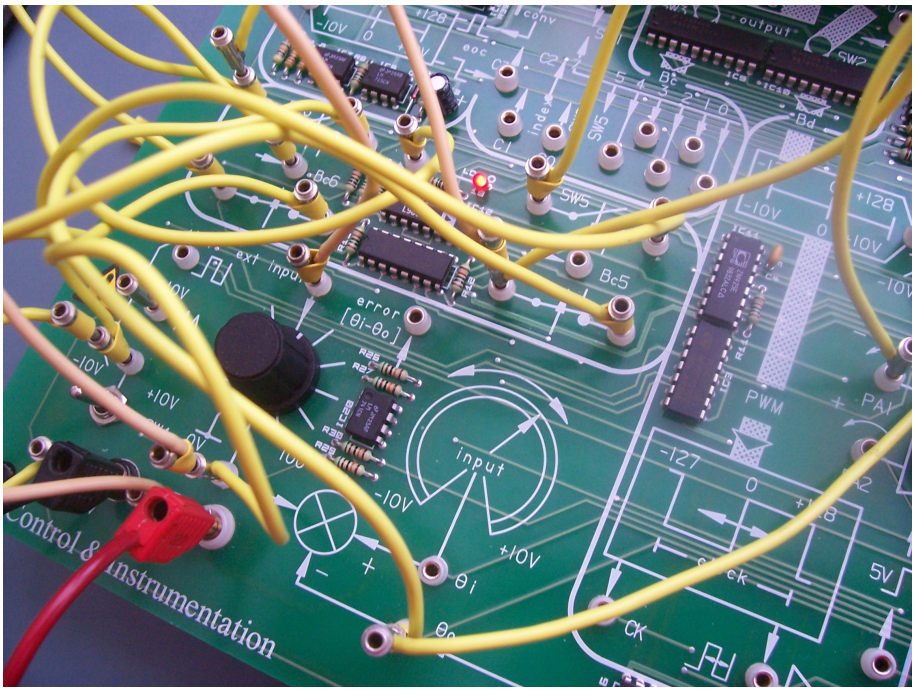


Fig. 3.8: The output shaft potentiometer terminal (bottom) and output of the input amplifier are connected to the multiplexer (center). The resulting signal is input to the A/D converter channel input (top).

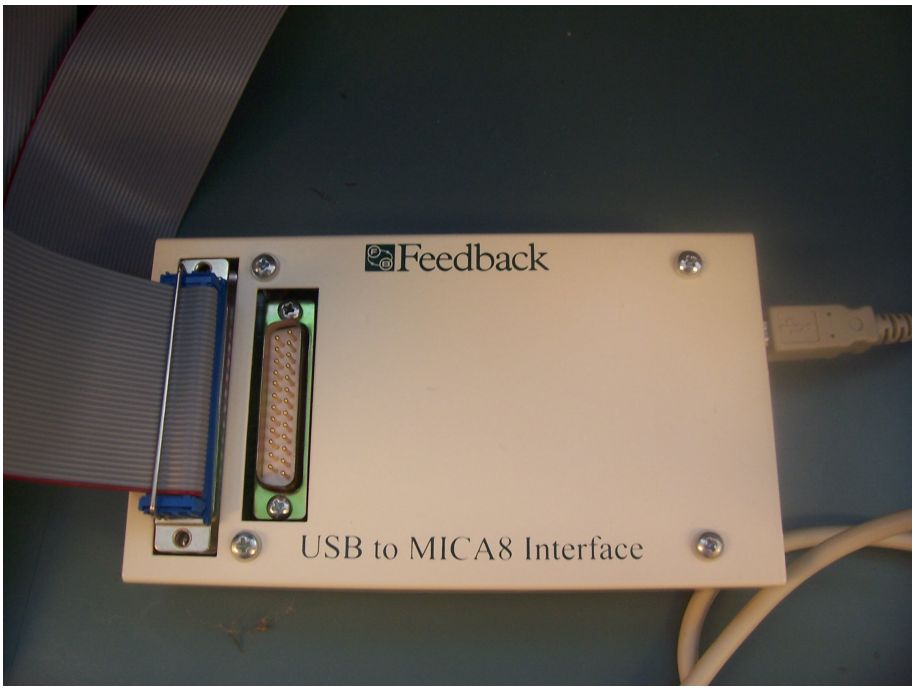


Fig. 3.9: PC USB interface.

The controller gains are altered on a scale of 0.0 to 5.0 at the graphical user interface. With the integral and derivative gains set to zero and the proportional gain set to 1.2, the response to a step input of approximately 5 Volts was as shown in Fig. 3.10.

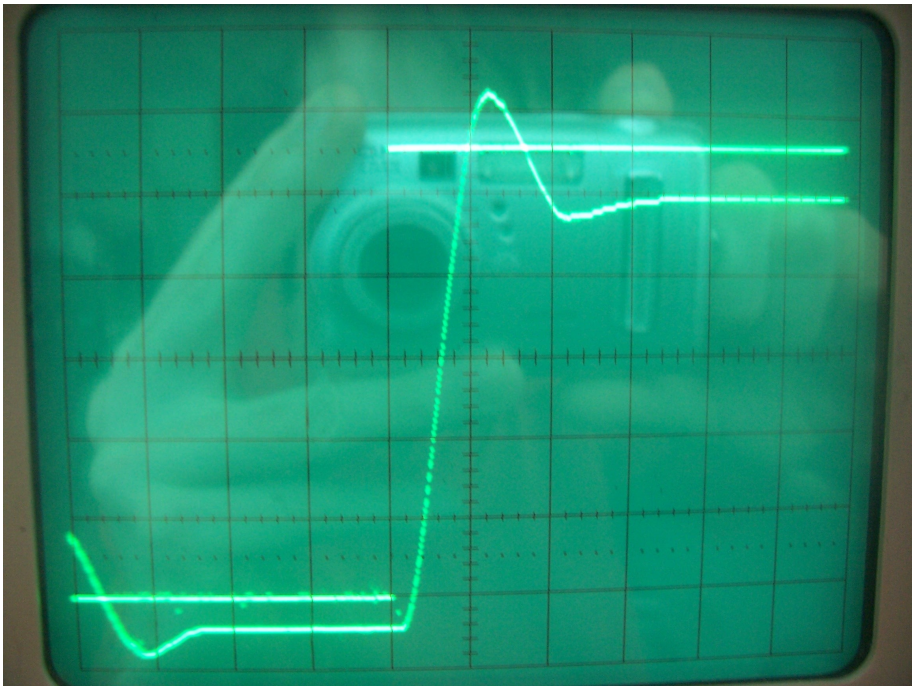


Fig. 3.10: Step response with proportional feedback only.

Unlike in the simulations in section 3.4, there was large a steady-state error. This may be due in part to un-modelled dynamics such as static friction. This error was reduced, as shown in Fig. 3.11, by increasing the integral gain to 0.1 in the software. Finally the response was damped a little, as shown in Fig. 3.12, by increasing the derivative gain to 0.3. Fig. 3.13 shows the digitally measured position displayed by the software. The positive 5 Volt step, two negative 5 Volt steps and positive 10 Volt step in input were done manually with the switch. There appears to be a slight offset in the calibration of the A/D converter, however both the shaft position and input signals were subject to this offset.

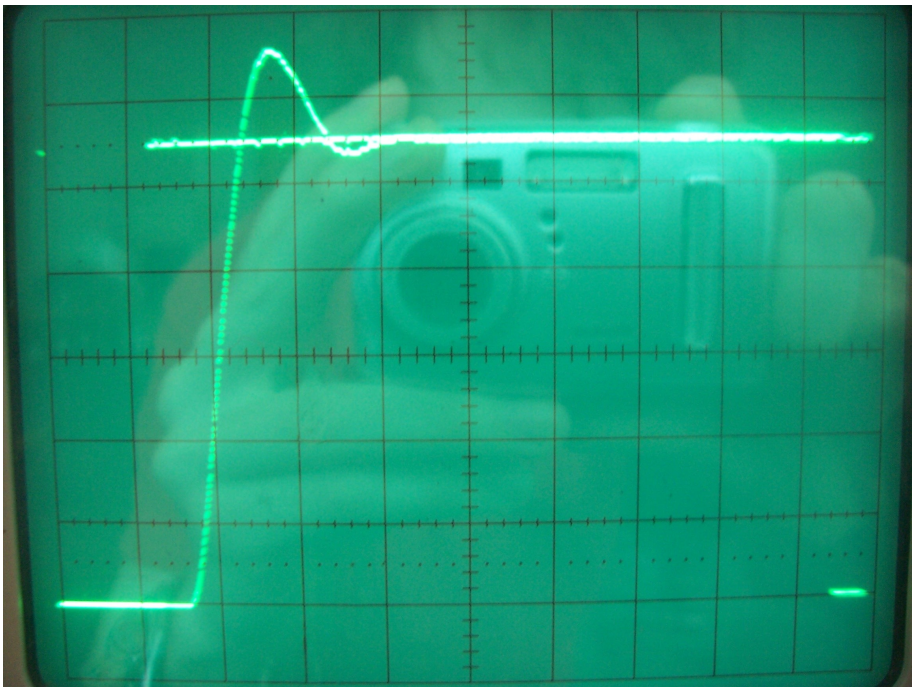


Fig. 3.11: Step response with proportional and integral feedback.

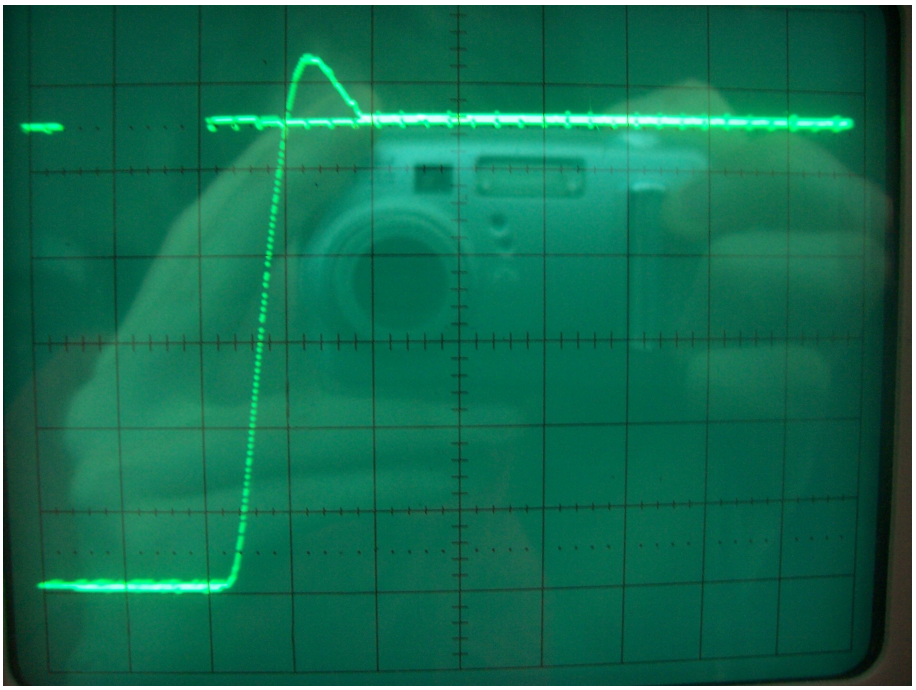


Fig. 3.12: Step response with proportional, integral and derivative feedback.

Fig. 3.13: Feedback Instruments tutorial software display showing the response of the servo to various steps in input.

3.6 Conclusion

The position output of the servo system has been successfully controlled using position feedback and the PID control algorithm in that the output shaft can be made to rotate to a commanded position given by a simple input voltage. However, the response is still not extremely fast and there is one more important problem, which, in typical industrial applications, must be addressed. It is the ability of the system to hold the output on target under disturbance forces. Fig. 3.14 shows the position and error when the disc was rotated from its target position in both directions by hand. Although the motor is not hugely powerful, this required little effort.

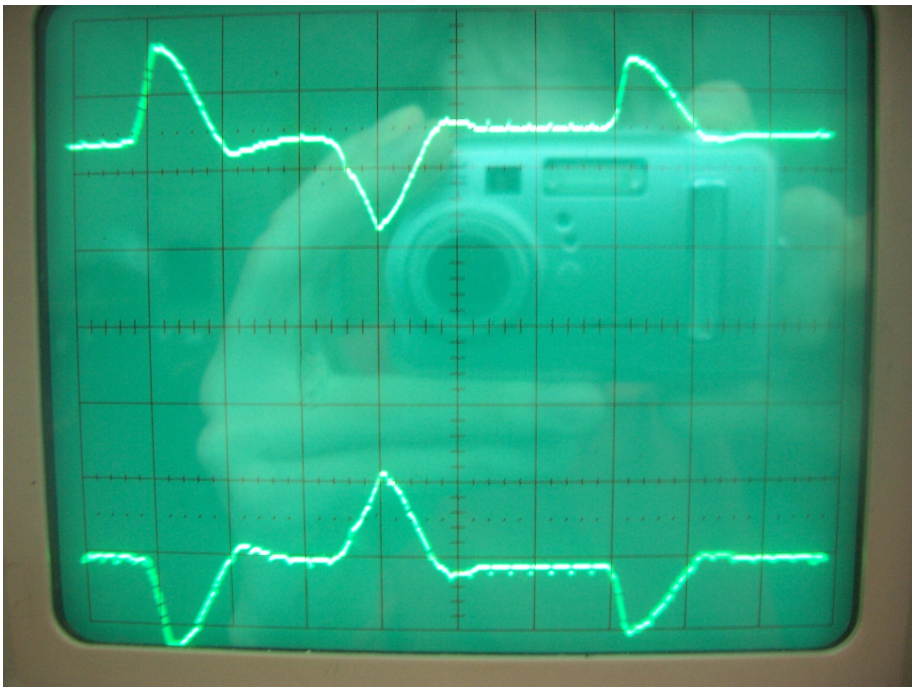


Fig. 3.14: The position and error when disturbance forces were applied by rotating the shaft from its target position by hand.

The system can only be made more rigid by using a larger position gain then what is allowed for in a linear analysis. Integral action was been found to be of little value when the controller gains were tuned using simulation. However, the physical system showed signs of friction, with non-zero steady-state errors. The use of higher position gain would also help in reducing steady-state error, eliminating the need to use integral action. When using position feedback only, such as with the PID controller, the poles of the whole system are limited in where they can be placed. By also feeding back other variables, more possibilities are opened up. State variable feedback is next topic of investigation.

4 State Variable Feedback Control

4.1 Introduction

"With the advent of the low-cost digital computer, simulation by digital means became more attractive than analogue. Meanwhile the representation of the system as a collection of first-order equations was seen in a new light. The integrator outputs could be represented as a mathematical vector, the equations fell into the shape of a 'matrix state equation' and a whole new range of techniques emerged." (Billingsley 1989, p.2)

4.2 Background

4.2.1 State Space Models

With the system represented by first-order equations, the derivative of each signal is a combination of the other signals. It is the derivatives of the signals that can applied to integrators to obtain a simulation of the signals themselves. The values of the signals or *state variables* at any instant describe the *state* of the system at that instant. For a state-space representation of a system, a number of linearly independent state variables equal to the order of the system is required. Such a representation is not unique. Nise (2004, p. 131) gives the following example. One pair of state variables for the second order RLC circuit shown in Fig. 4.1 could be the charge on the capacitor (q) and the current (i), giving the state equations 4.1 and 4.2 or, in matrix notation, Eqn. 4.3.

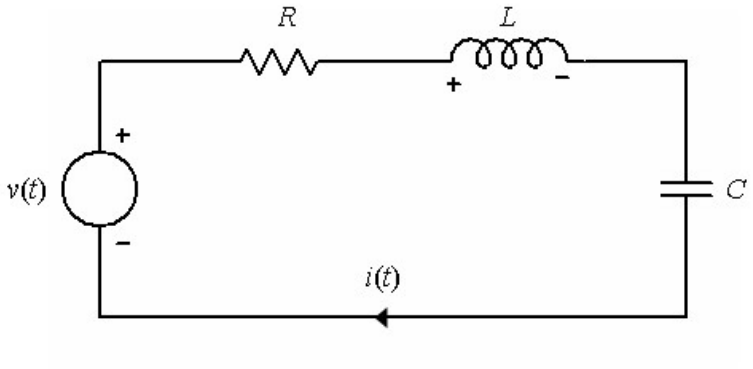


Fig. 4.1: Second order RLC circuit.

$$\frac{\mathrm{d}q}{\mathrm{d}t} = i \tag{4.1}$$

$$\frac{\mathrm{d}i}{\mathrm{d}t} = -\frac{1}{LC}q - \frac{R}{L}i + \frac{1}{L}v(t) \tag{4.2}$$

$$\begin{bmatrix} \dot{q} \\ \dot{i} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{1}{LC} & -\frac{R}{L} \end{bmatrix} \begin{bmatrix} q \\ i \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix} v(t) \tag{4.3}$$

Another pair of state variables is the resistor voltage and the capacitor voltage, giving the state equations 4.4 and 4.5 or, in matrix notation, Eqn. 4.6.

$$\frac{\mathrm{d}v_R}{\mathrm{d}t} = \frac{R}{L}(v(t) - v_R - v_C) \tag{4.4}$$

$$\frac{\mathrm{d}v_C}{\mathrm{d}t} = \frac{1}{RC}v_R \tag{4.5}$$

$$\begin{bmatrix} \dot{v_R} \\ \dot{v_C} \end{bmatrix} = \begin{bmatrix} -\frac{R}{L} & -\frac{R}{L} \\ \frac{1}{RC} & 0 \end{bmatrix} \begin{bmatrix} v_R \\ v_C \end{bmatrix} + \begin{bmatrix} \frac{R}{L} \\ 0 \end{bmatrix} v(t) \tag{4.6}$$

In general, if the system is linear, the state equations can be expressed in a matrix equation of the form:

$$\dot{\boldsymbol{X}} = \boldsymbol{A}\boldsymbol{X} + \boldsymbol{B}\boldsymbol{U} \tag{4.7}$$

where: $\boldsymbol{A}$ is the 'system' or 'plant' matrix
$\boldsymbol{B}$ is the 'input' or 'driving' matrix
$\boldsymbol{X}$ is the state vector
$\boldsymbol{U}$ is the 'input' or 'control' vector

Also the output/s of the system can be expressed in a matrix equation of the form:

$$\boldsymbol{Y} = \boldsymbol{CX} + \boldsymbol{DU} \tag{4.8}$$

where: $\boldsymbol{Y}$ is the output vector
$\boldsymbol{C}$ is the 'output' or 'connection' matrix
$\boldsymbol{D}$ is the 'feed forward' matrix

For a SISO system with no feed forward, shown in the block diagram of Fig. 4.2, the state equation and output equation become:

$$\dot{\boldsymbol{X}} = \boldsymbol{AX} + \boldsymbol{B}u \tag{4.9}$$

$$y = \boldsymbol{CX} \tag{4.10}$$

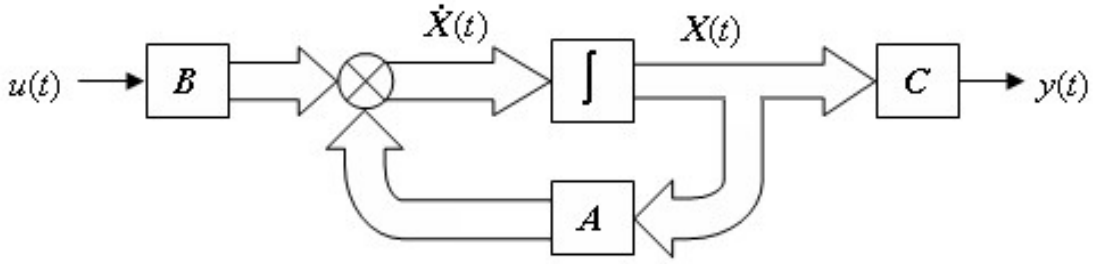


Fig. 4.2: State space representation of a SISO system.

In using state-space techniques for the purpose of designing a controller, it is important to choose state variables that can actually be conveniently measured by sensors. For the third order servo system, variables that are of interest are position, speed and acceleration. The acceleration of the output shaft in this system is a reflection of the torque generated by the motor, which is in turn, a reflection of the

current it draws and it may be more convenient to implement a current sensor than an accelerometer.

4.2.2 Discrete Time State Equations

Eqns. 4.11 and 4.12 are the digital equivalent of the matrix state and output equations for analogue systems, discussed in section 4.2.1. Just like the state equation for an analogue system is a collection of differential state equations, the state equation for a digital system is a collection of difference state equations.

$$\boldsymbol{X}(k+1) = \boldsymbol{\Phi}\boldsymbol{X}(k) + \boldsymbol{\Gamma}\boldsymbol{U}(k) \tag{4.11}$$

$$\boldsymbol{Y}(k) = \boldsymbol{C}\boldsymbol{X}(k) + \boldsymbol{D}\boldsymbol{U}(k) \tag{4.12}$$

For a SISO system with no feed forward, shown in the block diagram of Fig. 4.3, they become:

$$\boldsymbol{X}(k+1) = \boldsymbol{\Phi}\boldsymbol{X}(k) + \boldsymbol{\Gamma}u(k) \tag{4.13}$$

$$y(k) = \boldsymbol{C}\boldsymbol{X}(k) \tag{4.14}$$

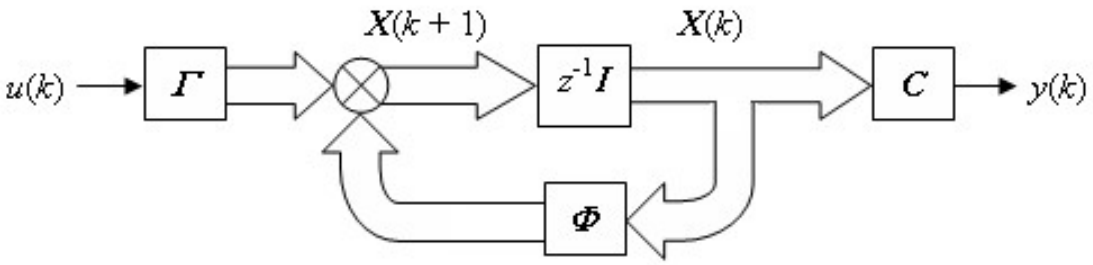


Fig. 4.3: State space representation of a digital SISO system.

The system matrix $\boldsymbol{\Phi}$ and the input matrix $\boldsymbol{\Gamma}$ can be constructed in a canonical form from the system's pulse transfer function. Alternatively, they can be constructed

from a discretisation of the analogue system using Eqns. 4.15 and 4.16.

$$\boldsymbol{\Phi} = \phi(T) \tag{4.15}$$

$$\boldsymbol{\Gamma} = \int_0^T \phi(q)\boldsymbol{B}\mathrm{d}q \tag{4.16}$$

where: ϕ is the system transition matrix

T is the A/D sampling period

The system transition matrix ϕ is the matrix exponential of the analogue system matrix $\boldsymbol{A}$ and can be evaluated using a few different methods.

$$\phi(t) = e^{\boldsymbol{A}t} \tag{4.17}$$

For a given A/D conversion frequency, the state equation can be discretised using this method with the *MATLAB* command *c2d*.

4.2.3 State Variable Feedback

A system is said to be *state controllable* at an initial state if it is possible to drive the system from that initial state to any desired state in a finite time period. If the system is state controllable at every state, then it is said to be *completely state controllable*. (Ogata 2002, p. 780) The matrix formed from the system and input matrices, as shown in Eqn. 4.18, is called the *controllability matrix*.

$$\boldsymbol{W}_{\mathrm{C}} = \begin{bmatrix} \boldsymbol{\Gamma} & \boldsymbol{\Phi\Gamma} & \boldsymbol{\Phi}^2\boldsymbol{\Gamma} & \dots & \boldsymbol{\Phi}^{k-1}\boldsymbol{\Gamma} \end{bmatrix} \tag{4.18}$$

where n is the order of the system

It can be shown that if the system's controllability matrix has a rank of n, then the system is completely state controllable. If, on the other hand, any of the state variables are independent of the input signal, then it is impossible to control these variables and the system is uncontrollable. "If the poles associated with all the uncontrollable variables are stable, then the system performance as a whole can still

be acceptable. No amount of feedback can move the positions of the poles, however." (Billingsley 1989, p. 172)

If a linear combination of the state variables is used as feedback, as shown in Fig. 4.4, the input to the system becomes:

$$r(k) = u(k) - \boldsymbol{K}^{\mathrm{T}}\boldsymbol{X}(k) \tag{4.19}$$

$$\text{where } \boldsymbol{K} = \begin{bmatrix} k_1 \\ k_2 \end{bmatrix}$$

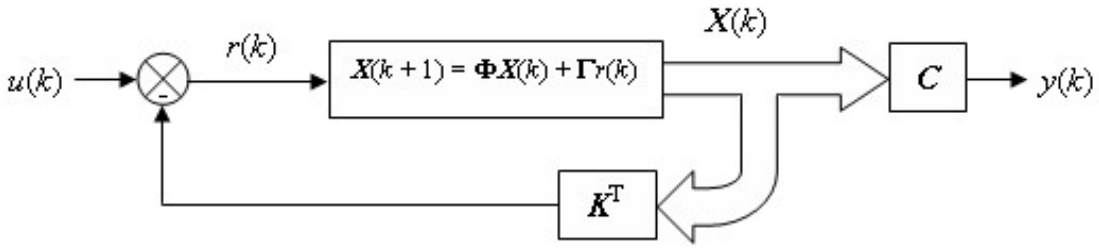


Fig. 4.4: Digital system with state variable feedback. The vector K consists of the gains applied to each state variable signal to be fed back.

u becomes a position command in software rather than the voltage applied to the motor, which is now r. $\boldsymbol{K}$ is the vector consisting of the gains applied to each of the state variable signals to be fed back. The state equation then becomes:

$$\begin{aligned} \boldsymbol{X}(k+1) &= \boldsymbol{\Phi}\boldsymbol{X}(k) + \boldsymbol{\Gamma}r(k) \\ &= \boldsymbol{\Phi}\boldsymbol{X}(k) + \boldsymbol{\Gamma}u(k) - \boldsymbol{\Gamma}\boldsymbol{K}^{\mathrm{T}}\boldsymbol{X}(k) \\ &= (\boldsymbol{\Phi} - \boldsymbol{\Gamma}\boldsymbol{K}^{\mathrm{T}})\boldsymbol{X}(k) + \boldsymbol{\Gamma}u(k) \end{aligned} \tag{4.20}$$

From Eqn. 4.20, it can be seen that the system matrix for the system with feedback is:

$$\boldsymbol{\Phi}_{\mathrm{f}} = \boldsymbol{\Phi} - \boldsymbol{\Gamma}\boldsymbol{K}^{\mathrm{T}} \tag{4.21}$$

The controller design criteria achievable with state variable feedback are a lot less limited than those achievable with PID control. When using pole placement with state variable feedback, the poles of the system can be placed anywhere to give the desired response. Eqn. 4.22 states Ackermann's formula for pole placement. P is the characteristic polynomial of the desired controlled system.

$$\boldsymbol{K}^{\mathrm{T}} = \begin{bmatrix} 0 & 0 & 0 & \dots & 1 \end{bmatrix} \boldsymbol{W}_{\mathrm{C}}^{-1} \boldsymbol{P}(\boldsymbol{\Phi}) \tag{4.22}$$

Ackermann's formula gives values for the feedback gains which will give the system the response determined by the chosen pole placement. Having said this, in a practical position controller, it is common to allow the actuator drive to saturate for much of the time, which introduces a nonlinearity into the system and alters its response from that predicted by this formula.

4.2.4 State Observers

A certain form of state space model for the servo system has been chosen due to lack of specifications to work from. The state variables in this form are not directly measurable using sensors. There are sometimes situations when signals to be used by the controller are not available from a sensor for A/D conversion. In such cases, if an approximate model of the system is known, it may be possible to control the system with an *observer*.

A system is said to be *observable* at an initial state if it is possible to determine this state from an observation of the output over a finite time period. If the system is observable at every state, then it is said to be *completely observable*. (Ogata 2002, p. 786) The matrix formed from the system and output matrices, as shown in

Eqn. 4.23, is called the *observability matrix*.

$$\boldsymbol{W}_{\mathrm{O}} = \begin{bmatrix} C \\ C\boldsymbol{\Phi} \\ C\boldsymbol{\Phi}^2 \\ \vdots \\ C\boldsymbol{\Phi}^{n-1} \end{bmatrix} \tag{4.23}$$

where n is the order of the system

It can be shown that if the system's observability matrix has a rank of n, then the system is completely observable. If, on the other hand, any of the state variables affect none of the outputs of the system, then their behaviour can not be measured and the system is unobservable. "You may say that since the variables do not affect the outputs, you do not care what they do. Unfortunately the outputs for control purposes are the transducers and sensors of the control system; the system could well be an aircraft carrying passengers who have sensors of their own, which are making them turn somewhat green." (Billingsley 1989, p. 172)

The property of observability means than any previous state of the system can be estimated from its model. The reverse is more useful; using this previous estimated state, together with the state and output equations of the model and a log of input values, the current state of the system can be estimated. The estimation of the current state can then be used for feedback and as a previous state for future state estimations. The observer is that part of the control algorithm that performs this function and is like a simulation of the system running in parallel with the physical system itself. A *full order* observer is of the same order as the system itself and can reconstruct estimations of all of the state variables, requiring only measurements of the input and output. A block diagram of an observer and its implementation is shown in Fig. 4.5.

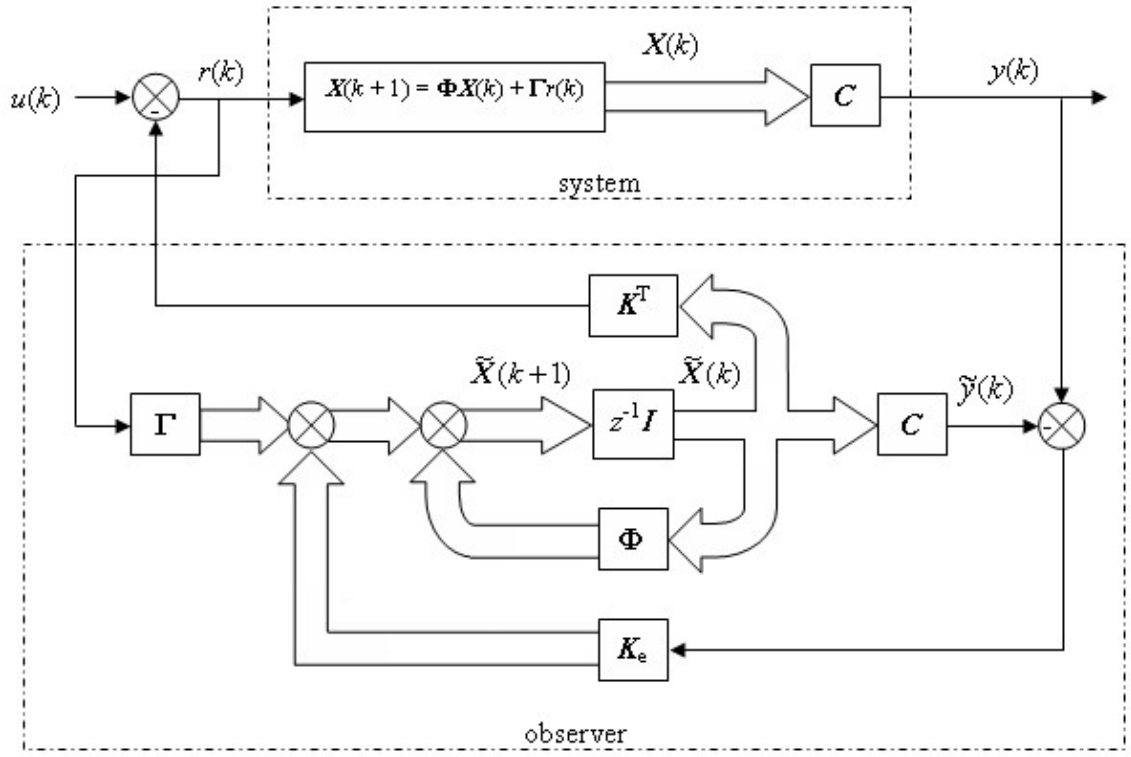


Fig. 4.5: Digital system with state variable feedback using a full order observer to estimate the state variables in contrast to them being measured by sensors.

$\boldsymbol{K}_\mathrm{e}$ is a vector of observer feedback gains, which are used reduce errors between the values estimated for the state variables by the observer and their actual values. The state equation describing the dynamics of the observer is:

$$\begin{aligned}\tilde{\boldsymbol{X}}(k+1) &= \boldsymbol{\Phi}\tilde{\boldsymbol{X}}(k) + \boldsymbol{\Gamma}r(k) + \boldsymbol{K}_\mathrm{e}(y(k) - \tilde{y}(k)) \\ &= \boldsymbol{\Phi}\tilde{\boldsymbol{X}}(k) + \boldsymbol{\Gamma}r(k) + \boldsymbol{K}_\mathrm{e}\boldsymbol{C}\boldsymbol{X}(k) - \boldsymbol{K}_\mathrm{e}\boldsymbol{C}\tilde{\boldsymbol{X}}(k) \\ &= (\boldsymbol{\Phi} - \boldsymbol{K}_\mathrm{e}\boldsymbol{C})\tilde{\boldsymbol{X}}(k) + \boldsymbol{\Gamma}r(k) + \boldsymbol{K}_\mathrm{e}\boldsymbol{C}\boldsymbol{X}(k) \qquad (4.24)\end{aligned}$$

A state equation can then be written for the error $\boldsymbol{E}(k)$ between the observer estimation of the state vector and the actual state vector.

$$\begin{aligned}\boldsymbol{E}(k+1) &= \boldsymbol{X}(k+1) - \tilde{\boldsymbol{X}}(k+1) \qquad (4.25) \\ &= \boldsymbol{\Phi}\boldsymbol{X}(k) + \boldsymbol{\Gamma}r(k) - ((\boldsymbol{\Phi} - \boldsymbol{K}_\mathrm{e}\boldsymbol{C})\tilde{\boldsymbol{X}}(k) + \boldsymbol{\Gamma}r(k) + \boldsymbol{K}_\mathrm{e}\boldsymbol{C}\boldsymbol{X}(k))\end{aligned}$$

$$= (\boldsymbol{\Phi} - \boldsymbol{K}_{\mathrm{e}}\boldsymbol{C})\boldsymbol{E}(k) \tag{4.26}$$

Feedback gains can be chosen in a way which makes the observer stable and the errors decay away. If the system is completely observable, the poles of the observer can be placed anywhere to give it a suitable response in rectifying its initial estimation errors. With this in mind, the observer should be designed for a faster response than that of the system because the system requires accurate state estimation for its control action. However, noise amplification by large feedback gains should also be considered. Eqn. 4.27 states Ackermann's formula for calculating observer gains. P is the characteristic polynomial of the desired observer.

$$\boldsymbol{K}_{\mathrm{e}} = P(\boldsymbol{\Phi})\,\boldsymbol{W}_{\mathrm{O}}^{-1} \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \tag{4.27}$$

With observer feedback gains chosen, the system with observer can be simulated. Eqns. 4.26 and 4.28 can be combined to form the overall state equation 4.29 in terms of system state variables and observer error variables.

$$\begin{aligned} \boldsymbol{X}(k+1) &= \boldsymbol{\Phi}\boldsymbol{X}(k) + \boldsymbol{\Gamma}r(k) \\ &= \boldsymbol{\Phi}\boldsymbol{X}(k) + \boldsymbol{\Gamma}u(k) - \boldsymbol{\Gamma}\boldsymbol{K}^{\mathrm{T}}\tilde{\boldsymbol{X}}(k) \\ &= \boldsymbol{\Phi}\boldsymbol{X}(k) - \boldsymbol{\Gamma}\boldsymbol{K}^{\mathrm{T}}\boldsymbol{X}(k) + \boldsymbol{\Gamma}\boldsymbol{K}^{\mathrm{T}}\boldsymbol{X}(k) - \boldsymbol{\Gamma}\boldsymbol{K}^{\mathrm{T}}\tilde{\boldsymbol{X}}(k) + \boldsymbol{\Gamma}u(k) \\ &= (\boldsymbol{\Phi} - \boldsymbol{\Gamma}\boldsymbol{K}^{\mathrm{T}})\boldsymbol{X}(k) + \boldsymbol{\Gamma}\boldsymbol{K}^{\mathrm{T}}\boldsymbol{E}(k) + \boldsymbol{\Gamma}u(k) \end{aligned} \tag{4.28}$$

$$\begin{bmatrix} \boldsymbol{X}(k+1) \\ \boldsymbol{E}(k+1) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Phi} - \boldsymbol{\Gamma}\boldsymbol{K}^{\mathrm{T}} & \boldsymbol{\Gamma}\boldsymbol{K}^{\mathrm{T}} \\ \boldsymbol{0} & \boldsymbol{\Phi} - \boldsymbol{K}_{\mathrm{e}}\boldsymbol{C} \end{bmatrix} \begin{bmatrix} \boldsymbol{X}(k) \\ \boldsymbol{E}(k) \end{bmatrix} + \begin{bmatrix} \boldsymbol{\Gamma} \\ \boldsymbol{0} \end{bmatrix} u(k) \tag{4.29}$$

4.2.5 The Control Algorithm

While the behaviour of the system using a state-variable feedback controller has been simulated using matrix equations in *MATLAB*, an implementation of the controller software is more likely to be written in a language like BASIC or C, for subsequent processing by a compiler, or even in assembly language. Except for perhaps some observer calculations, there is no simulation - simply some A/D conversion of the actual signals and application of the control action, and matrix equations are not needed. The following piece of code is the main loop of a program written by Billingsley (2006, p. 78) for use with the linear (translational) position control experiment depicted in Fig. 4.6 . The code Is written for *Qbasic* with direct access to the lower level input/output functions of the PC, which is used to control the motor through a simple but effective printer port interface. The motor drive which is output by the program is by pulse width modulation using a H-bridge circuit.

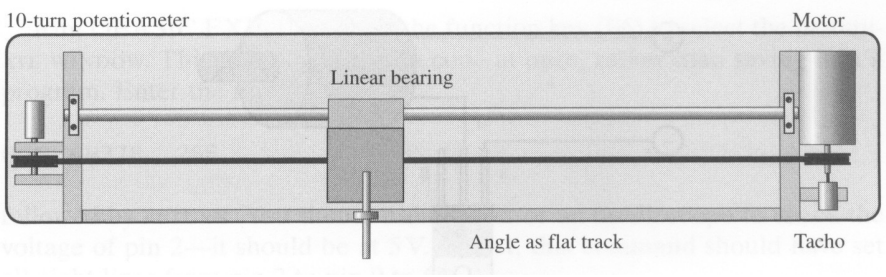


Fig. 4.6: Position control hardware. From *Essentials of Mechatronics* by Billingsley

```
ON PLAY(1) GOSUB rates
PLAY ON
PLAY "cde"
DO
    a$ = INKEY$
    if a$ = "." THEN xdemand = xdemand + 0.1
    if a$ = "," THEN xdemand = xdemand - 0.1
    if a$ = "0" THEN xdemand = 0
    g = g + dg
    if g > 1 then dg = -.1
    if g < 0 then dg = .1
```

```
    IF u > g THEN
        OUT port, 1
    ELSEIF u < g - 1 THEN
        OUT port, 2
    ELSE
        OUT port, 0
    END IF
LOOP UNTIL a$ = "q"
PLAY OFF
OUT port, 0
END

rates:
v = ADC(0)
x = ADC(1)
vobs = (x - xslow) * kt
xslow = xslow + vobs * dt
vdemand = k * (xdemand - x)
IF vdemand > vmax THEN vdemand = vmax
IF vdemand < -vmax THEN vdemand = -vmax
u = (vdemand - v) * kv   '(or vobs)
t = t + dt
IF t > tmax THEN t  = 0
PSET (t, v)
PSET (t, x), 14  'yellow
PSET (t, vobs / 20), 12 'red

PLAY "n0"
RETURN
```

The DO loop forms a triangular wave in software for comparison with a drive value "u", to get the duty cycle of the pulse width modulation. The PLAY "cde" line plays a succession of three notes taking a reliable 10 milliseconds and interrupts the DO loop upon finishing. The interrupt routine performs the A/D conversion and calculates the appropriate drive value before returning control to the main program. Billingsley has implemented state variable feedback with the lines

```
vdemand = k * (xdemand - x)
u = (vdemand - v) * kv   '(or vobs)
```

This is Equivalent to writing:

$$u = k_v k x_{\text{demand}} - k_v k x - k_v v \tag{4.30}$$

It is important to note that the drive is determined by the position *error*. The feedback gain for the position variable x is not only applied to the value measured by the sensor but also to the demanded value. Thus, in the design which follows, the input is a position demand and must be subject to a gain the same as the that applied to the position feedback. Billingsley has also implemented an observer to estimate the remaining variable, velocity, with the lines

```
vobs = (x - xslow) * kt
xslow = xslow + vobs * dt
```

Referring to Fig. 4.5, the value `x - xslow` corresponds to the difference $y(k) - \tilde{y}(k)$. The value `kt` is that element of the observer feedback gain vector which affects the observer's velocity estimation. The second line then updates the observer's position estimation.

Another point to note is the nonlinearity introduced into the system with the lines

```
IF vdemand > vmax THEN vdemand = vmax
IF vdemand < -vmax THEN vdemand = -vmax
```

This is in anticipation of higher gains values than would be chosen after a linear analysis. Limiting the drive in this way, allows the faster more rigid position control system achieved with excessive gains, whilst preventing large overshoots. Although the design procedures used in this project have been based on theory which assumes linearity is maintained, a practical position control system, which, for example, might be required to maintain high precision against machining forces, is likely to require a more 'crisp' and rigid response.

4.3 Simulation

With a range of state-space representations to choose from, some commonly used forms have been dubbed *canonical forms*. The representation of a servo system using its position, speed and acceleration, where each state variable is the derivative of the previous, is in *controller canonical* or *phase variable* form. The form arising from the choice of position, speed and current as state variables would be preferable so that a controller based on it could use the sensor signals directly. However, for lack of system specifications, the controller canonical form has been derived from the transfer function of the system and then discretised to obtain a state equation for the digital system. The general form of the transfer function for a DC motor and load was stated in section 2.3 as:

$$\frac{\theta_{\mathrm{L}}(s)}{V_{\mathrm{t}}(s)} = \frac{\frac{N_1}{N_2}k_{\mathrm{T}}}{s((J_{\mathrm{m}}s + D_{\mathrm{m}})(L_{\mathrm{a}}s + R_{\mathrm{a}}) + k_{\mathrm{T}}k_{\mathrm{v}})}$$

Three state equations for this system based on position, speed and armature current (i_{a}) can be used to describe this system.

$$\dot{\theta}_{\mathrm{L}} = \frac{N_1}{N_2}\omega_{\mathrm{m}} \tag{4.31}$$

$$\dot{\omega}_{\mathrm{m}} = \frac{k_{\mathrm{T}}i_{\mathrm{a}} - D_{\mathrm{m}}\omega_{\mathrm{m}}}{J_{\mathrm{m}}} = -\frac{D_{\mathrm{m}}}{J_{\mathrm{m}}}\omega_{\mathrm{m}} + \frac{k_{\mathrm{T}}}{J_{\mathrm{m}}}i_{\mathrm{a}} \tag{4.32}$$

$$\dot{i}_{\mathrm{a}} = \frac{V_{\mathrm{t}} - k_{\mathrm{v}}\omega_{\mathrm{m}} - R_{\mathrm{a}}i_{\mathrm{a}}}{L_{\mathrm{a}}} = -\frac{k_{\mathrm{v}}}{L_{\mathrm{a}}}\omega_{\mathrm{m}} - \frac{R_{\mathrm{a}}}{L_{\mathrm{a}}}i_{\mathrm{a}} + V_{\mathrm{t}} \tag{4.33}$$

From Eqns. 4.31, 4.32 and 4.33, the following matrix equation and output equation can be written:

$$\begin{bmatrix} \dot{\theta}_{\mathrm{L}} \\ \dot{\omega}_{\mathrm{m}} \\ \dot{i}_{\mathrm{a}} \end{bmatrix} = \begin{bmatrix} 0 & \frac{N_1}{N_2} & 0 \\ 0 & -\frac{D_{\mathrm{m}}}{J_{\mathrm{m}}} & \frac{k_{\mathrm{T}}}{J_{\mathrm{m}}} \\ 0 & -\frac{k_{\mathrm{v}}}{L_{\mathrm{a}}} & -\frac{R_{\mathrm{a}}}{L_{\mathrm{a}}} \end{bmatrix} \begin{bmatrix} \theta_{\mathrm{L}} \\ \omega_{\mathrm{m}} \\ i_{\mathrm{a}} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} V_{\mathrm{t}} \tag{4.34}$$

$$\theta_{\mathrm{L}} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \theta_{\mathrm{L}} \\ \omega_{\mathrm{m}} \\ i_{\mathrm{a}} \end{bmatrix} \tag{4.35}$$

A different representation of the system has been formed using the transfer function for the Feedback Instruments system, stated in section 2.3 as:

$$\frac{\theta_{\mathrm{L}}(s)}{V_{\mathrm{t}}(s)} = \frac{20K_1}{s(s+1.5)(s+10)}$$
$$= \frac{20K_1}{s^3+11.5s^2+15s}$$

From this transfer function, it is clear that the system can be described by the following third order differential equation:

$$\dddot{\theta}_{\mathrm{L}} + 11.5\ddot{\theta}_{\mathrm{L}} + 15\dot{\theta}_{\mathrm{L}} = 20K_1V_{\mathrm{t}} \tag{4.36}$$

Letting $c = \dfrac{\theta_{\mathrm{L}}}{20K_1}$:

$$\dddot{c} + 11.5\ddot{c} + 15\dot{c} = V_{\mathrm{t}} \tag{4.37}$$

$$\theta_{\mathrm{L}} = 20K_1c \tag{4.38}$$

Using state variables $x_1 = c$, $x_2 = \dot{c}$ and $x_3 = \ddot{c}$ the following state equations can be written and Eqn. 4.38 used as the output equation.

$$\dot{x_1} = x_2 \tag{4.39}$$

$$\dot{x_2} = x_3 \tag{4.40}$$

$$\dot{x_3} = -15x_2 - 11.5x_3 + V_{\mathrm{t}} \tag{4.41}$$

From Eqns. 4.38-4.41, the following matrix state equation and output equation can be written:

$$\begin{bmatrix}\dot{x_1} & \dot{x_2} & \dot{x_3}\end{bmatrix} = \begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ 0 & -15 & -11.5\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix} + \begin{bmatrix}0\\ 0\\ 1\end{bmatrix}V_{\mathrm{t}} \tag{4.42}$$

$$\theta_{\mathrm{L}} = \begin{bmatrix}20K_1 & 0 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix} \tag{4.43}$$

This representation of the system is in controller canonical form and the state variables are phase variables. The output matrix will remain the same for the digital system. The system and input matrices can be discretised according to Eqns. 4.15 and 4.16. Using state variable feedback, the poles of the system can be placed anywhere to give the desired response. For now, the poles have been placed so as to not push the system to hard and cause drive saturation. If it is desired that all of the system's response transients are overdamped with a time constant of $\tau = 0.1$ seconds, then with the sampling period of $T = 0.5$ milliseconds, the poles of the system should be placed at:

$$z = e^{-\frac{T}{\tau}} = e^{-\frac{0.0005}{0.1}} = 0.995 \tag{4.44}$$

This is located on the positive half of the real axis of the z-plane, close to the unit circle, which is the boundary for system stability. Any modelling inaccuracies or variations in system properties could move the poles outside the circle and lead the system to becoming unstable. In order to move the poles closer to the origin, the sampling frequency can be reduced, for example to 10 Hz. Then, the poles should be placed at:

$$z = e^{-\frac{T}{\tau}} = e^{-\frac{0.1}{0.1}} = 0.3679 \tag{4.45}$$

Using this sampling frequency, the system and input matrices have been discretised with the *MATLAB* command *c2d*, as shown in the following script.

```
A=[0 1 0; 0 0 1; 0 -15 -11.5];
B=[0; 0; 1];
C=[20 0 0];
D=0;
sys=ss(A,B,C,D);
T=0.1;
digitalsys=c2d(sys,T)
```

The following output shows the state space model for the discretised system.

```
a =
```

```
        x1  x2       x3
    x1  1   0.09809  0.003488
    x2  0   0.9477   0.05798
    x3  0   -0.8697  0.2809

b =
        u1
    x1  0.0001271
    x2  0.003488
    x3  0.05798

c =
        x1  x2  x3
    y1  20  0   0

d =
        u1
    y1  0

Sampling time: 0.1
Discrete-time model.
```

The state and output equations for the digital system are:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 0.09809 & 0.003488 \\ 0 & 0.9477 & 0.05798 \\ 0 & -0.8697 & 0.2809 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \end{bmatrix} + \begin{bmatrix} 0.0001271 \\ 0.003488 \\ 0.05798 \end{bmatrix} V_{\mathrm{t}}(k) \tag{4.46}$$

$$\theta_{\mathrm{L}} = \begin{bmatrix} 20K_1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \end{bmatrix} \tag{4.47}$$

According to Eqn. controllability, the controllability matrix and its inverse are:

$$\begin{aligned} \boldsymbol{W}_{\mathrm{C}} &= \begin{bmatrix} \boldsymbol{\Gamma} & \boldsymbol{\Phi}\boldsymbol{\Gamma} & \boldsymbol{\Phi}^2\boldsymbol{\Gamma} & \dots & \boldsymbol{\Phi}^{k-1}\boldsymbol{\Gamma} \end{bmatrix} \\ &= \begin{bmatrix} 0.0001 & 0.0007 & 0.0014 \\ 0.0035 & 0.0067 & 0.0071 \\ 0.0580 & 0.0133 & -0.0021 \end{bmatrix} \end{aligned} \tag{4.48}$$

$$\boldsymbol{W}_{\mathrm{C}}^{-1} = \begin{bmatrix} 0.5394 & -0.0980 & 0.0220 \\ -2.0930 & 0.3994 & -0.0194 \\ 1.7036 & -0.1865 & 0.0075 \end{bmatrix} \tag{4.49}$$

The system is therefore controllable. Using Eqn. 4.22, some suitable feedback gains will now be determined. As discussed earlier, the three poles of the desired controlled system are to be placed at $z = 0.3679$. This gives the system the characteristic polynomial given by Eqn. 4.50.

$$\begin{aligned} P(z) &= (z - 0.3679)^3 \\ &= z^3 - 1.1037z^2 + 0.4061z - 0.0498 \end{aligned} \tag{4.50}$$

According to Eqn. 4.22, the feedback gains required for this response are:

$$\begin{aligned} \boldsymbol{K}^{\mathrm{T}} &= \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \boldsymbol{W}_{\mathrm{C}}^{-1}(\boldsymbol{\Phi}^3 - 1.1037\boldsymbol{\Phi}^2 + 0.4061\boldsymbol{\Phi} - 0.0498\mathbf{I}) \\ &= \begin{bmatrix} 430.3 & 142.1 & 9.9 \end{bmatrix} \end{aligned} \tag{4.51}$$

The phase variables are not exactly the state variables measured by the sensors, but are proportional to them. Thus, each of the calculated gains would have to be scaled and would also have to be inclusive of the sensor gain. As, discussed in section 4.2.5 the input, which is now a position demand in software must be subject to the same gain as is applied to the position feedback. For a forward path gain of 1, the ratio between the output position and the state variable $x_1 = c$ is $20K1$ or 20. For example, for a steady-state output of 180° rotation, the corresponding value of x_1 is 9. The input position demand, therefore, should have the gain 1/20 applied as well. Then for the 180° output, the input should be $180/20 \times 430.3$ or 3872. The following script uses the program *simulate_ss.m*, found in Appendix E, to simulate the system with these feedback gains.

```
T=0.1;K1=1;K3=3872;N=3/T;input=ones(1,N);instants=(0:N)*T;
phi=[1 0.09809 0.003488; 0 0.9477 0.05798; 0 -0.8697 0.2809];
gamma=[0.0001271; 0.003488; 0.05798];
C=[20*K1 0 0];
X0=[0; 0; 0];
K=[430.3; 142.1; 9.9];
```

```
Ke=[0; 0; 0];
E0=[0; 0; 0];
[X,output,E] = simulate_ss(input,K3,phi,gamma,C,X0,'feedback',K,Ke,E0);
plot(instants,output,instants,X(1,:),'m', ...
instants,X(2,:),'r',instants,X(3,:),'g')
xlabel('time (s)')
ylabel('shaft position (degrees)')
```

The response is shown in Fig. 4.7. The blue curve is the output and the other curves show the trajectories of each of the state variables. The purple curve is the first state variable $x_1 = c$ so is a reflection of the output position. The red curve is the second state variable, which is a reflection of the speed. As expected, the motor speeds up and then begins to slow down before the target is reached, giving an overdamped response with no overshoot. The green curve is the third state variable, which is a reflection of the acceleration.

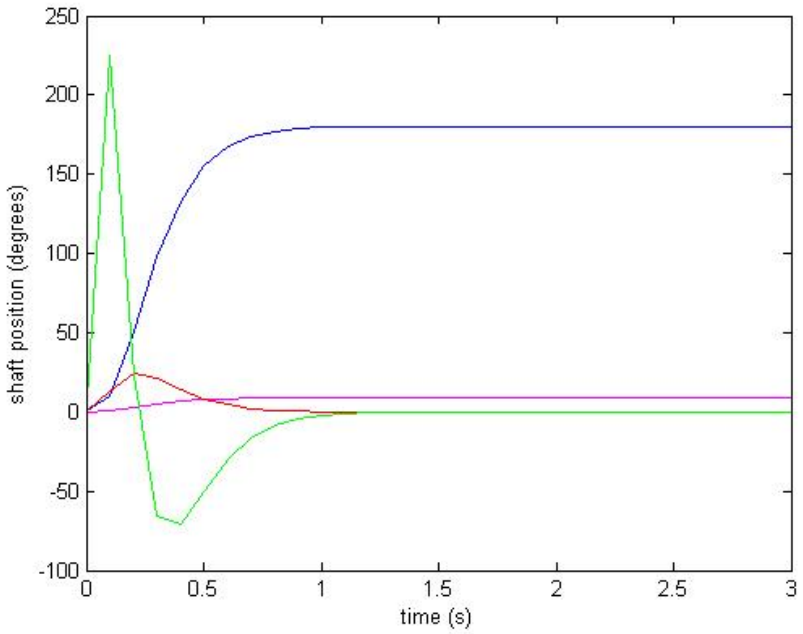


Fig. 4.7: Simulated step response of the servo system. The blue curve is the output and the other curves show the trajectories of each of the state variables.

Fig. 4.8 shows the response compared to that that was obtained using PID control.

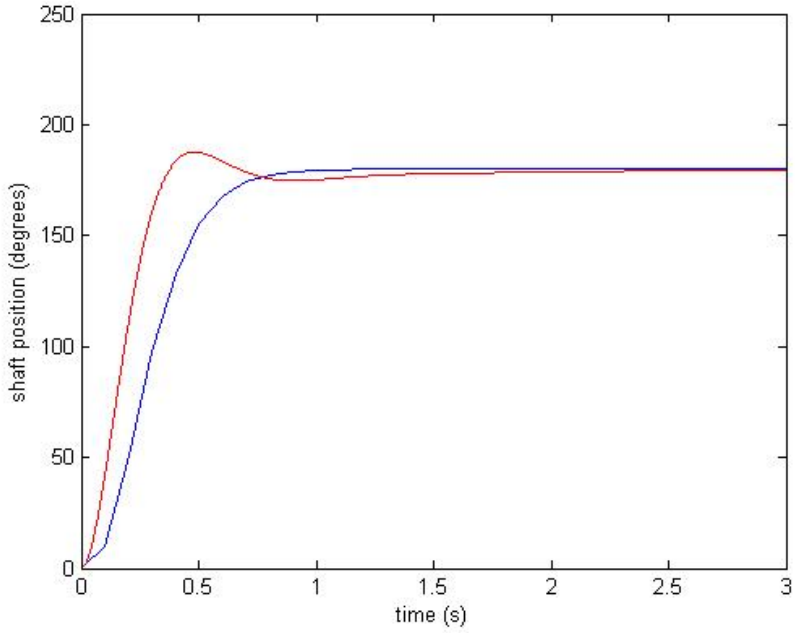


Fig. 4.8: The response obtained using the state variable feedback gains given in Eqn. compared to that obtained using PID control.

An observer will now be included in the system, eliminating the need for speed and acceleration sensors. According to Eqn. 4.23, the observability matrix and its inverse are:

$$\begin{aligned} \boldsymbol{W}_{\mathrm{O}} &= \begin{bmatrix} C \\ C\boldsymbol{\Phi} \\ C\boldsymbol{\Phi}^2 \end{bmatrix} \\ &= \begin{bmatrix} 20.0000 & 0 & 0 \\ 20.0000 & 1.9619 & 0.0698 \\ 20.0000 & 3.6704 & 0.2031 \end{bmatrix} \end{aligned} \tag{4.52}$$

$$\boldsymbol{W}_{\mathrm{O}}^{-1} = \begin{bmatrix} 0.0500 & 0 & 0 \\ -0.9795 & 1.4920 & -0.56125 \\ 13.2115 & -27.6226 & 14.4112 \end{bmatrix} \tag{4.53}$$

The system is therefore observable. Using Eqn. 4.27, some suitable feedback gains will now be determined. Assuming that noise amplification will not be excessive, a deadbeat response for the observer will be aimed for. This gives the observer the following characteristic polynomial:

$$P(z) = z^3 \tag{4.54}$$

According to Eqn. 4.27, the feedback gains required for this response are:

$$\begin{aligned} \boldsymbol{K}_{\mathrm{e}} &= \boldsymbol{\Phi}^3 \boldsymbol{W}_{\mathrm{O}}^{-1} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} 0.1114 \\ 0.6167 \\ -0.2459 \end{bmatrix} \end{aligned} \tag{4.55}$$

The following script simulates the system with some initial observer errors.

```
E0=[5; 10; -5];
Ke=[0.1114; 0.6167; -0.2459];
[X,output,E] = simulate_ss(input,K3,phi,gamma,C,X0,'observer',K,Ke,E0);
plot(instants,output,instants,X(1,:),'m', ...
instants,X(2,:),'r',instants,X(3,:),'g', ...
instants,E(1,:),'k',instants,E(2,:),'k', ...
instants,E(3,:),'k')
axis([0 3 -100 250])
xlabel('time (s)')
ylabel('shaft position (degrees)')
```

The response is shown in Fig. 4.9. The system behaves a little erratically because the system settling time is not sufficiently slower than the observer than the observer settling time. As expected, the observer's state variable estimations, which

are shown in black, all have a deadbeat response, settling within three 0.1 second sampling periods.

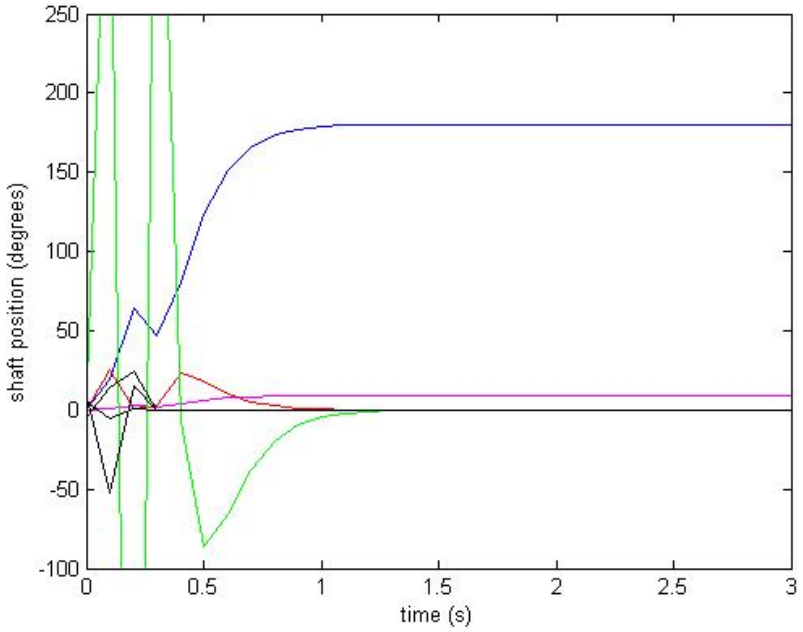


Fig. 4.9: Simulated response of the system observer.

4.4 Conclusion

The response of the system was able to be tailored more freely, within physical limits, with state variable feedback control than with PID control. Quick response with no overshoot has been obtained, in comparison to the optimised PID control, which still exhibited a small amount of overshoot and a longer settling time. Once again, however, the design that was conducted using linear control theory is unlikely to make full use of the power of the motor in delivering a fast and rigid position control system.

5 Conclusions

The aims of the project were to investigate the design processes used to implement some conventional digital control strategies in the context of a position control system and to assess their performance and suitability in position control. The design processes were carried out successfully to arrive at a number of possible control algorithms. These algorithms were expressed in a form suited to simulation but can easily be implemented in a physical system with a few simple lines of code. Whilst the analysis of their effect on the behaviour of the system can become rather involved, their implementation is much simpler than the sub-tasks of sensor interfacing, signal filtering, A/D and D/A conversion.

5.1 Further Research and Recommendations

It has been concluded from observations made during this research that, despite the apparent success of the designs implied by their step response plots, linear control strategies are not well suited to position control and superior control can be achieved with other methods. By introducing nonlinearity into the control algorithm, a very fast and rigid system can be formed. This can be achieved by limiting the motor drive to a value lower than its saturation point or by making the position feedback some other nonlinear function of position error. The behaviour of such a nonlinear system can be analysed graphically by plotting a series trajectories that the state variables of the system would follow from an initial state. For a second order system, the number of state variables is two, meaning that a two-dimensional plot of the trajectories can be drawn on a plane called the *phase plane*. By limiting drive, the phase plane is effectively divided into a range of arbitrary regions, where different control actions apply. This is in a way a form of *fuzzy logic control*. The phase plane is one of a number of methods used to analyse the stability of nonlinear systems. It seems, however, that simulation is an invaluable tool in the design a

practical position controller, which may involve some trial and error. Flexibility is an important criterion to be engineered into the design, something which gives digital control a huge advantage over analogue control. It is recommended for future research that a system be built from scratch so that a more accurate model can be determined and research undertaken into some of the nonlinear techniques just mentioned.

5.2 Summary

The state variable feedback was found to be the most successful and most versatile method. While the feedback schemes were successful in providing closed-loop control of the system with desirable response characteristics, the experiments showed that the system lacked rigidity against disturbing forces. This is a problem that must be addressed in most industrial applications of position control and further review of the literature showed that it is a problem inherent to controller designs based on analysing the system only in its linear region of operation. From observations made during this research, it seems that the controller design techniques established on linear control theory are not well suited to position control and superior control can be achieved with other methods.

References

Basar, T. (2000), *Control Theory: Twenty-Five Seminal Papers (1932-1981)*, Wiley-IEEE Press.

Billingsley, J. (1989), *Controlling With Computers*, McGraw-Hill.

Billingsley, J. (2006), *Essentials of Mechatronics*, Wiley.

Brooks, B. L. (2008), *90mm Antiaircraft Gun*, Antiaircraft Command.
`http://www.antiaircraft.org/90mm.htm`
(May 2008).

Chiasson, J. (2005), *Modelling and High Performance Control of Electric Machines*, Wiley-IEEE Press.

CNC Machining Services (2008), industry Search Australia and NZ.
`http://www.industrysearch.com.au/Products/CNC_Machining_Services-16933`
(May 2008).

Discovery II (1999), Feedback Instruments Limited. (Computer program).

Flugge-Lotz, I. (2003), 'Memorial to n. minorsky', *IEEE Transactions on Automatic Control* p. 290.

Franklin, G. F. et al. (2002), *Feedback Control of Dynamic Systems*, 4th edn, Prentice-Hall.

Irvine, M. M. (2001), 'Early digital computers at bell telephone laboratories', *IEEE Annals of the History of Computing* p. 24.

Kevin (2004), Position control using a digital servo system.

KUKA Roboter GmbH, B. (2003), *KUKA*, Wikipedia.
`http://en.wikipedia.org/wiki/KUKA`
(May 2008).

Lewis, F. L. (1992), *Applied Optimal Control and Estimation*, Prentice-Hall.

Mindell, D. A. (2000), 'Automation's finest hour: Bell labs and automatic control in world war ii', *IEEE Control Systems Magazine* pp. 72, 73.

Nise, N. S. (2004), *Control Systems Engineering*, 4th edn, Wiley.

Ogata, K. (2002), *Modern Control Engineering*, 4th edn, Prentice-Hall.

Phillips, C. & Nagle, H. (1990), *Digital Control System Analysis and Design*, 2nd edn, Prentice-Hall.

Pingstone, A. (2002), *Flight Control Surfaces*, Wikipedia.
`http://en.wikipedia.org/wiki/Flight_control_surfaces`
(May 2008).

Robust Control (2008), Wikipedia.
`http://en.wikipedia.org/wiki/Robust_control`
(May 2008).

Sen, P. C. (1997), *Principles of Electric Machines and Power Electronics*, 2nd edn, Wiley.

Sputnik 1 (2008), Wikipedia.
`http://en.wikipedia.org/wiki/Sputnik_1`
(May 2008).

Torpedo Data Computer (TDC) Restoration (2007), USS Cod Home Port.
`http://www.usscod.org/tdc-restore.html`
(May 2008).

Appendix A - Project Specification

University of Southern Queensland
FACULTY OF ENGINEERING AND SURVEYING

ENG4111/4112 Research Project
PROJECT SPECIFICATION

FOR:	**David Salomon**
TOPIC:	POSITION CONTROL USING DIGITAL SERVO SYSTEM
SUPERVISOR:	Dr. Paul Wen
ENROLMENT:	ENG4111 - S1, D, 2008 ENG4112 - S2, D, 2008
PROJECT AIM:	To develop a digital controller and algorithm suitable for controlling a servo system within certain performance requirements.
PROGRAMME:	**Issue A, 25 March, 2008**

1. Study relevant areas of control engineering, electronics and software simulation not yet encountered.
2. Gain access to the faculty's analogue servo trainer and digital servo trainer and the Feedback Instruments Limited manual. Experiment and become familiar with the equipment.
3. Research literature on digital and analog position control systems.
4. Design an analogue controller to achieve some performance characteristics.
5. Develop mathematical models for the servo system components and an overall system model.
6. Design a digital controller and control algorithm and simulate the behaviour of the servo system.
7. Implement the algorithm using a PC interface and test.

AGREED:

_____ (student) _____ (supervisor)

____/____/____ ____/____/____

Examiner/Co-examiner:_____

Appendix B - simulate.m

```
%Function:          simulate
%Syntax:            output = simulate(input,T,K1,K2)
%Purpose:           to simulate the response of the position controller
%                   with proportional feedback only
%Inputs:            input - array of input values at sampling instants
%                   T - sampling period (s)
%                   K1 - forward path gain
%                   K2 - position sensor gain
%Outputs:           position - array of output shaft position values (degrees)
%                              at sampling instants
%Author:            David Salomon
%Last Revision:     20/04/08

function output = simulate(input,T,K1,K2)

%-------------initialise parameters---------------------
N = length(input);

A = exp(-1.5*T);
B = exp(-10*T);

beta3 = K1*(-160/153*B + 2/85*A + 4/3*T*B*A + 46/45*B*A);
beta2 = K1*(46/45 + 818/765*B - 818/765*A - 4/3*T*A - 4/3*T*B - 46/45*B*A);
beta1 = K1*(-46/45 -2/85*B + 160/153*A + 4/3*T);
beta0 = 0;

alpha3 = -A*B;
alpha2 = A*B + A+B;
```

```
alpha1 = -(A+B+1);
alpha0 = 1;

phi3 = alpha3 + K2*beta3;
phi2 = alpha2 + K2*beta2;
phi1 = alpha1 + K2*beta1;
phi0 = alpha0 + K2*beta0;

input=cat(2,zeros(1,3),input,zeros(1,5));   % zero input before starting time
output=zeros(1,4);                          % zero output before starting time

%----------------main loop--------------------------
for n=5:N+4                                 % loop for length of simulation
output(n)=(beta3*input(n-3)...
    +beta2*input(n-2)+beta1*input(n-1)...
    +beta0*input(n)...
    -phi3*output(n-3)-phi2*output(n-2)...
    -phi1*output(n-1))/phi0;                % inverse Z-transform of output
end

output=output(5:end);                       % only use output from starting time onwards
```

Appendix C - simulatePID.m

```
%Function:          simulatePID
%Syntax:            output = simulatePID(input,T,K1,K2,Kp,Ki,Kd)
%Purpose:           to simulate the response of the position controller
%                   with PID control
%Inputs:            input - array of input values at sampling instants
%                   T - sampling period (s)
%                   K1 - forward path gain
%                   K2 - position sensor gain
%                   Kp - proportional gain
%                   Ki - integral gain
%                   Kd - derivative gain
%Outputs:           output - array of output shaft position values (degrees)
%                            at sampling instants
%Author:            David Salomon
%Last Revision:     20/04/08

function output = simulatePID(input,T,K1,K2,Kp,Ki,Kd)

%-------------initialise parameters---------------------
N = length(input);

q0 = Kp+Kd/T;
q1 = Ki*T - 2*Kd/T - Kp;
q2 = Kd/T;

A = exp(-1.5*T);
B = exp(-10*T);
```

```
u = K1*(-160/153*B + 46/45*A*B + 2/85*A + 4/3*T*A*B);
v = K1*(818/765*B + 46/45 - 818/765*A - 4/3*T*A - 4/3*T*B - 46/45*A*B);
w = K1*(-2/85*B - 46/45 + 160/153*A + 4/3*T);

beta5 = q2*u;
beta4 = q1*u + q2*v;
beta3 = q0*u + q1*v + q2*w;
beta2 = q0*v + q1*w;
beta1 = q0*w;
beta0 = 0;

alpha4 = A*B;
alpha3 = -2*A*B - A - B;
alpha2 = A*B + 2*A + 2*B + 1;
alpha1 = -A - B - 2;
alpha0 = 1;

phi5 = K2*beta5;
phi4 = alpha4 + K2*beta4;
phi3 = alpha3 + K2*beta3;
phi2 = alpha2 + K2*beta2;
phi1 = alpha1 + K2*beta1;
phi0 = alpha0 + K2*beta0;

input=cat(2,zeros(1,4),input,zeros(1,6));   % zero input before starting time
output=zeros(1,5);                          % zero output before starting time

%----------------main loop--------------------------
for n=6:N+5                                 % loop for length of simulation
output(n)=(beta5*input(n-5)...
    +beta4*input(n-4)+beta3*input(n-3)...
```

```
    +beta2*input(n-2)+beta1*input(n-1)...
    +beta0*input(n)...
    -phi5*output(n-5)-phi4*output(n-4)...
    -phi3*output(n-3)-phi2*output(n-2)...
    -phi1*output(n-1))/phi0;                % inverse Z-transform of output
end

output=output(6:end);                       % use only output from starting time onwards
```

Appendix D - optimise.m

```
%Function:          optimise
%Syntax:            [Kp,Ki,Kd]=optimise(input,T,K1,K2,gamma,Kpguess,Kdguess,Kiguess)
%Purpose:           to determine PID controller parameters for optimal response
%                   with respect to the performance criterion used in
%                   subfunction 'performance'
%Inputs:            input - array of input values at sampling instants
%                   T - sampling period (s)
%                   K1 - forward path gain
%                   K2 - position sensor gain
%                   gamma - constant step size
%                   Kpguess - starting value for Kp
%                   Kiguess - starting value for Ki
%                   Kdguess - starting value for Kd
%Outputs:           Kp - proportional gain for the optimal response reached
%                   Ki - integral gain for the optimal response reached
%                   Kd - derivative gain for the optimal response reached
%Author:            David Salomon
%Last Revision:     20/04/08

function [Kp,Ki,Kd] = optimise(input,T,K1,K2,gamma,Kpguess,Kdguess,Kiguess)

%-------------initialise figures---------------------
scrsz = get(0,'ScreenSize');                        % get screen size
figure('Position',[scrsz(3)/3 scrsz(4)/2 scrsz(3)/3 3*scrsz(4)/8]);
                                                    % set size and position for figure 1
fig1 = gcf;                                         % get figure handle for figure 1
figure('Position',[2*scrsz(3)/3 scrsz(4)/2 scrsz(3)/3 3*scrsz(4)/8]);
                                                    % set size and position for figure 2
```

```
fig2 = gcf;                                          % get figure handle for figure 2
figure('Position',[scrsz(3)/3 scrsz(4)/16 scrsz(3)/3 3*scrsz(4)/8]);
                                                     % set size and position for figure 3
fig3 = gcf;                                          % get figure handle for figure 3
figure('Position',[2*scrsz(3)/3 scrsz(4)/16 scrsz(3)/3 3*scrsz(4)/8]);
                                                     % set size and position for figure 4
fig4 = gcf;                                          % get figure handle for figure 4

%------------initialise parameters-------------------
N = length(input);
instants=(1:N)*T;           % array of sampling instants (s)
gradmag = 500;              % initialise gradient magnitude
k=0;                        % initialise iteration count
Kp = Kpguess;
Ki = Kiguess;
Kd = Kdguess;

output = simulate(input,T,K1,K2);                % simulate for no controller
error = input-K2*output;
figure(fig2)
hold on
grid on
plot(instants,output,'m'),xlabel('time (s)'),ylabel('shaft position (degrees)')
figure(fig3)
hold on
grid on
plot(instants,error,'m'),xlabel('time (s)'),ylabel('error (Volts)')

output = simulatePID(input,T,K1,K2,Kp,Ki,Kd);     % simulate for starting gains
error = input-K2*output;
S = performance(error);                          % evaluate controller performance
```

```
figure(fig1)
hold on
grid on
plot(k,S,'x','MarkerSize',10),xlabel('iteration'),ylabel('performance (IAE)')
figure(fig2)
plot(instants,output)
title(['Kp = ',num2str(Kp),', Ki = ',num2str(Ki),', Kd = ',num2str(Kd)])
figure(fig3)
plot(instants,error,'r')

%---------------------main loop----------------------
while abs(gradmag) > 400    % loop until a minimum is reached

    k=k+1;                  % increment iteration counter

    deltaK=0.2*gamma;               % test step size used for differentiation

    testoutKp = simulatePID(input,T,K1,K2,Kp+deltaK,Ki,Kd);    % C(t,q0+dq,q1,q2)
    testerrKp = input - K2*testoutKp;

    testoutKi = simulatePID(input,T,K1,K2,Kp,Ki+deltaK,Kd);    % C(t,q0,q1+dq,q2)
    testerrKi = input - K2*testoutKi;

    testoutKd = simulatePID(input,T,K1,K2,Kp,Ki,Kd+deltaK);    % C(t,q0,q1,q2+dq)
    testerrKd = input - K2*testoutKd;

    pderivKp = (performance(testerrKp)-performance(error))/deltaK;  % dS/dq0
    pderivKi = (performance(testerrKi)-performance(error))/deltaK;  % dS/dq1
    pderivKd = (performance(testerrKd)-performance(error))/deltaK;  % dS/dq2

    gradmag = sqrt(pderivKp^2+pderivKi^2+pderivKd^2);   % gradient magnitude
```

```
    Kp = Kp-gamma/gradmag*pderivKp;                    % new value for q0
    Ki = Ki-gamma/gradmag*pderivKi;                    % new value for q1
    Kd = Kd-gamma/gradmag*pderivKd;                    % new value for q2

    output = simulatePID(input,T,K1,K2,Kp,Ki,Kd);
    error = input-K2*output;
    S=performance(error);
    figure(fig1)
    plot(k,S,'x','MarkerSize',10)
    figure(fig4)
    hold on
    grid on
    plot(k,gradmag,'x','MarkerEdgeColor','r','MarkerSize',10)
    xlabel('iteration'),ylabel('performance function gradient')
    figure(fig2)
    plot(instants,output)
    title(['Kp = ',num2str(Kp),', Ki = ',num2str(Ki),', Kd = ',num2str(Kd)])
    figure(fig3)
    plot(instants,error,'r')

    pause(0.01)

end


function S = performance(error)

S=sum(abs(error));    % Integral of Absolute Error (IAE)
```

Appendix E - simulate_ss.m

```
%Function:          simulate_ss
%Syntax:            [X,output,E]=simulate_ss(input,K3,phi,gamma,C,X0,type,K,Ke,E0)
%Purpose:           to simulate the response of the position controller
%                   with state variable feedback control
%Inputs:            input - array of demanded position values at sampling instants
%                   K3 - gain applied to demanded position values to give
%                           correct steady state position
%                   phi - digital system matrix
%                   gamma - digital system input matrix
%                   C - system output matrix
%                   X0 - column vector of initial state variable values
%                   type - string indicating type of simulation; 'feedback' or 'observer'
%                   K - column vector of state variable feedback gains
%                   Ke - column vector of observer error feedback gains
%                   E0 - column vector of initial observer error values
%Outputs:           X - array of state variable values at sampling instants
%                   output - array of output position values at sampling instants
%                   E - array of observer errors at sampling instants
%Author:            David Salomon
%Last Revision:     01/06/08

function [X,output,E] = simulate_ss(input,K3,phi,gamma,C,X0,type,K,Ke,E0)

N = length(input);
input = K3*input;                   % apply gain to input
X(:,1) = X0;                        % initial state
phif = phi-gamma*K';                % closed loop system matrix
E(:,1) = E0;                        % initial observer errors
```

```
phio = phi-Ke*C;                    % closed loop observer error system matrix
gammao = gamma*K';                  % input matrix for system with observer

switch type
    case 'feedback'                 % case with sensor feedback
        for n=2:N+1
            X(:,n) = phif*X(:,n-1)+gamma*input(:,n-1);  % update state variables
        end
    case 'observer'                 % case with observer feedback
        for n=2:N+1
            X(:,n) = phif*X(:,n-1)+gammao*E(:,n-1)+gamma*input(:,n-1);  % update state variables
            E(:,n) = phio*E(:,n-1); % update observer errors
        end
    otherwise                       % no valid type entered
        disp('type must be "feedback" or "observer"')
end

output=C*X;                         % output position values
```