The Characterisation and Emulation of Rotary Mechanical Machines

David Wells

Master of Philosophy

ASTON UNIVERSITY

April 2000

The University of Aston in Birmingham

The Characterisation and Emulation of Rotary Mechanical Machines

by David Wells

Master of Philosophy, 2000.

Summary

Typically machine control software is tested using software blocks to replace the physical machines. It is often beneficial to further test the controllers and drive hardware before using physical systems since unavailability or potential damage cost due to failure may make this impractical. The design of a physical machine emulator was investigated that 'pretends' to be a (single-shaft) machine or machine component. In addition to this the 'test-rig' was designed to characterise a machine connected to it. A set of models could then be compiled and used in isolation or in combination to form machines of arbitrary complexity that could be emulated.

Models suitable to represent machines and components of the test-rig are examined, in particular system classification, continuous and discrete-time systems, linear and non-linear systems and their behaviour. Control strategies are proposed and theoretical and practical performance tests conducted. PID control is employed and implemented using a sampled data system incorporating a DSP. Further control strategies such as velocity feedback and feedforward are combined where applicable to enhance the test-rig control.

The subject of system identification and parameter estimation is summarised, and the relevant methods are applied to identify and parameterise machine structures. Practical tests are conducted on physical machines constructed specifically for the purpose of characterisation. The design and construction of these machines and the test-rig is explained and practical performance tests conducted.

Indexing terms: rotary machines, characterise, emulate, system identification, parameter estimation.

Acknowledgements

I would like to thank Dr. S.D. Garvey for his supervision of this project, and for his help and support throughout.

I would also like to thank Mr. P. Pizer and Mr. K. Davies for their excellent work constructing the test-rig mechanical components.

I must also thank the DTI and EPSRC for funding the research.

Contents

List of Figures

Abbreviations:

ADC	- Analogue to Digital converter
AU	- Aston University
BJ	- Box Jenkins
DAC	- Digital to Analogue converter
DHSM	- Design of High Speed Machinery (a DTI / EPSRC programme)
DSP	- Digital Signal Processor / Digital Signal Processing
DTI	- Department of Trade and Industry
EPSRC	- Engineering and Particle Science Research Council
ETFE	- Empirical Transfer Function Estimate
FFT	- Fast Fourier transform
FIR	- Finite Impulse Response
IIR	- Infinite Impulse Response
IV	- Instrumental Variable
lsb	- least significant bit
LTF	- Laplace transfer function
LU	- Loughborough University
LS	- Least Squares
msb	- most significant bit
LSE	- Least Squares Estimate
ML	- Maximum Likelihood
MLE	- Maximum Likelihood Estimate
OE	- Output Error
PDF	- Probability Density function
PE	- Parameter Estimation
PEM	- Prediction Error Methods
SHS	- Simon Handling Systems ?
SI	- System Identification
SISO	- Single Input, Single Output

Operators and Notational Conventions:

Operators

E is the expectation operator

Notation

$e(t)$ = 1) controller error, or 2) disturbance at time t usually white noise (sequence)

$G(q)$ = "true" transfer function from u to y

$G(q,\theta)$ = transfer function from u to y corresponding to a set of parameters θ

$\hat{G}(\cdot)$ = estimated transfer function (with parameters)

$\hat{G}_N(q)$ = estimate of transfer function $G(q)$ based on Z^N

$\hat{\hat{G}}_N$ = empirical estimate of transfer function $G(q)$ based on Z^N

$u(t)$ = input variable at time t

$v(t)$ = disturbance variable at time t

V = loss function

$x(t)$ = state vector at time t

$\overline{x}$ = 1) arithmetic mean, or 2) not x (Boolean algebra)

$y(t)$ = output variable at time t

$\hat{y}(t)$ = predicted output variable at time t

Z^N = set of input-output data $\{u(0), y(0), \ldots, u(N), y(N)\}$

$\varepsilon(t,\theta)$ = prediction error $y(t) - \hat{y}(t \mid \theta)$

λ = variance

θ = 1) angle, where $\dot{\theta}$ and $\ddot{\theta}$ are velocity and acceleration respectively, or
2) vector used to parameterise models

$\hat{\theta}$ = estimated parameters

σ = variance or standard deviation

μ = probability distribution

ω = frequency (rads.sec^{-1})

CHAPTER 1

1 Introduction

1.1 Background to the project

This area of research forms part of a joint project between the Department of Mechanical and Electrical Engineering at Aston University (AU) and the Manufacturing Systems Integration Research Institute at Loughborough University (LU). The project is titled "An Integrated Approach to the design of Control Systems for High-Speed Machines", and is funded by the DTI, EPSRC, SHS Ltd., and Cirrus Technologies Ltd., as part of the DHSM LINK programme with industry for the Design of High Speed Machinery. The project seeks to unify existing work on control system design for high performance machinery from the highest level where genericity and reusability are key issues to the lowest level where real time performance is paramount. The conjoined techniques are to be tested and proven in two specially assembled environments; a system emulation in software and system implementation in hardware. The project combines work in three areas - software engineering for control systems, simulation and emulation of complex dynamic systems and practical dynamic measurements on systems. The simulation and emulation aspects will involve characterising time-dependant (single-shaft) systems in some (generic) way and then constructing an emulator unit which is capable of "pretending" to be, for example four-bar mechanism or a slider crank etc. It is this part of the project which the author and this report is concerned with.

1.2 Justification

What does the "Machine Characterisation / Emulation" hope to achieve?

1.2.1 In the context of the project

Tools already exist for testing high level machine control code which replaces "real" outputs to drive units and "real" inputs from sensors with software blocks which emulate "ideal" manufacturer-supplied units. Moreover the high level machine control code can be executed on the target computer, eliminating the problems associated with porting software. The purpose of the software emulation would be to carry out initial trial concepts inexpensively. If certain control ideas cannot be made to work in the emulation, then it is almost certain that they could not be made to work in physical

reality. However, if they look workable in the emulation, then there is at least a strong possibility that the machine is realisable. This full software emulation allows testing and further development of this code to be conducted prior to hardware implementation, but there is no guarantee the "ideal" software blocks will behave exactly the same as the actual inputs and outputs of the machine. By using a physical machine emulator, these actual inputs and outputs will be employed, which may not have "ideal" characteristics (see figure 1 below).

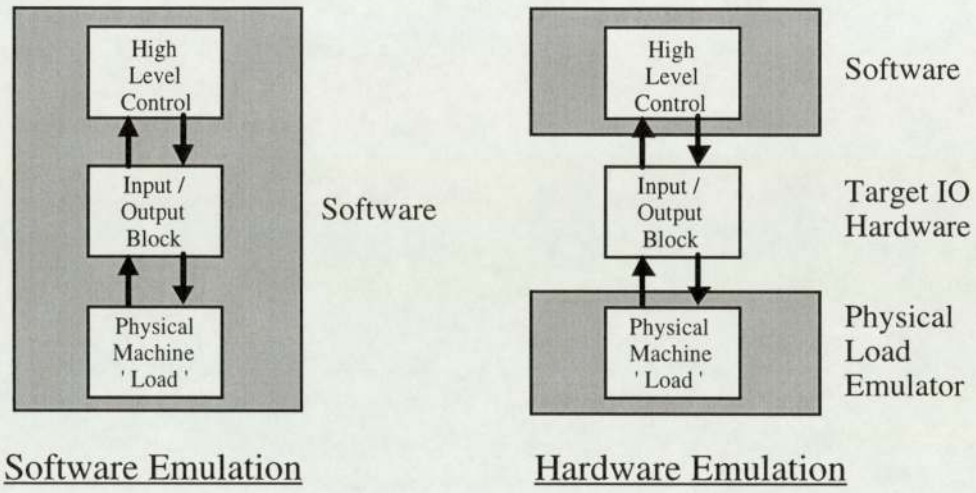


Figure 1.1 Diagrammatical representation of a) Software and b) Hardware Emulation.

Software emulation is clearly the cheapest test that can be carried out on a proposed machine design. Potentially, it is limited in accuracy only by the in-exactness of models for the various machine "components". Many machine components can be modelled very accurately within the majority of contexts, but some components have not been modelled in any detail and occasions arise when familiar components are used in an unconventional context that might be outside the normal range for which the models are accurate. Under these conditions it may be desirable to 'characterise' the machine, which will result in a much more accurate representation. The fact that no software product is ever either completely finished or completely bug-free also raises the possibility that a machine design might appear to work in the emulation which could not work in reality. Thus, there is inevitably a need to "prototype" the proposed machine as closely as possible. The "hardware" prototyping would be used as a proof of concept stage.

1.2.2 Other application areas

The Machine Emulator is expected to have uses outside the area of system development. It is predicted to have the ability to emulate a wide range of machines with a high bandwidth capability, including machines of an arbitrary description. This is a potentially useful tool for testing motor / drive pairs. During motor and drive development, various input demands are applied, and the corresponding mechanical response measured into a steady or varying machine, typically a dynamometer or induction brake providing an opposing torque. Another test is to apply a constant demand, and observe the motor/drive's ability to respond to changes in the machine. The machine emulator will be able to source as well as sink mechanical energy, which is not possible from a dynamometer or induction brake, and will also have a higher bandwidth.

1.3 The representation of Machines

The largest outcome of a previous project at Aston entitled *Design and Application Methodologies for Multi-Axis High-Speed Machines with Independent Drives* was a bundle of software called the *Design Methodology Suite*. This software allows a model of a dynamic system to be incrementally assembled and contains tools for synthesis and analysis of the system. It is claimed a whole-system simulation using this software is the most harsh test which can be carried out without prototyping in hardware. This software is now under further development (S.D Garvey 1996) in the context of this project, where dynamic models are to be used real-time in conjunction with LU's control software package. A generic simulation structure has been adopted where instances of machine (and motor / drive) classes can be used to build up an accurate system model.

The type of machines which will be of greatest interest to this project will be heavily cyclic, time dependant linear / nearly linear. The machines will also be processed off-line, and this will be discussed in more detail later.

CHAPTER 2

2 Literature Survey

This chapter is split into two sections. The first is an overview of similar work in this area, and the second is an overview of System Identification and Parameter Estimation.

2.1 Machine Emulation and Characterisation

There seems to be little published literature on characterisation and emulation per se [29], but a significant amount on their constituent parts i.e. dynamic models, system identification, discrete real-time systems and mechanical construction issues. Some work has been published regarding the use of dynamometers to mimic mechanical "loads", in particular for the use of testing motors, but this has been limited to energy dissipation only. The application of dynamic models was investigated in the first instance, and then system identification.

J. Pu *et al.* (1989) considered modelling and control methods of pneumatic and electric drives, with the aim of achieving improved motion control. It was found that in general a model based control strategy can enable optimisation of selected performance characteristics, provided that an accurate model of the machine is known (or can be identified) and can be implemented within the motion controller in a way that can account for changing operating conditions.

Several papers have been published which employ the use of dynamic models (and use other methods) to control robot manipulators in the presence of uncertainties such as payloads with unknown mass / inertia properties and external disturbances, and also take into account mode uncertainty. (J.K. Mills *et al.* 1989, C.Y. Kuo *et al.* 1990). C. Canudas de Wit *et al.* (1995) proposed a new dynamic model for control of systems with friction which captures most friction phenomena that are of interest for feedback control.

G. Dodds & N. Glover (1995) created a rudimentary motor-load system which is used to derive complex models of robotic systems for use in feed forward control. The system models the electrical hardware and then the actuator and mechanical transmission effects can be determined. Real time filtering and estimation of differential parameters

is also discussed for off-line as well as for on-line implementation, but not in much detail.

2.2 System Identification and Parameter Estimation

System Identification is the evaluation of a system model representing the essential aspects of a system and presenting knowledge of that system in a usable form (Eykhoff, 1974). Parameter Estimation is the derivation of the model parameters.

The mathematical approaches used in identification are of either the deterministic or stochastic type. In the first case the noise is either not acting on the system or it is negligible. Stochastic models are models which take into account the noise. A deterministic model can be obtained by simply omitting the term corresponding to the random input.

The following sub-sections are derived from current literature and provide an overview only.

2.2.1 Methods of Data Processing

Since the mid 1960's System Identification and Parameter Estimation has received serious interest, mainly due to advances in computers. The three main areas of research have been:

1) Analogue Methods for Continuous-time models,
2) Digital Methods for Continuous-time models,
3) Digital Methods for Discrete-time Models.

It is Discrete-time models which are of most interest to the author, as data will be sampled and produced in this way, even though much conceptual analysis will be in terms of continuous-time equations.

There are two approaches to data processing; 'off-line' or batch processing approach, and 'on-line' or recursive approach. In the batch processing case the computational operations are carried out on the complete set of data as a whole, in contrast to the recursive approach where the parameter estimate is updated continuously while working

serially through the data. In general recursive algorithms yield less efficient parameter estimates for a given set of time-series data, but have the advantage of inherent on-line operation. The method more applicable to this project is batch processing, as the data will be processed off-line.

2.2.2 Linear System Identification

Mathematical Models

The main representations of system models, in discrete-time form are:

State Space Representation

$$x(t+1) = Ax(t) + Bu(t) + w(t) \tag{2.1}$$

$$y(t) = Cx(t) + Du(t) + v(t) \tag{2.2}$$

Stochastic Difference Equation

$$\sum_{i=0}^{v} a_i y(t-i) = \sum_{i=0}^{v} b_i u(t-i) + n(t) \tag{2.3}$$

Generalised Regression Model

$$y(t) = \sum_{i=1}^{N_a} a_i y(t-i) + \sum_{i=0}^{N_b} b_i u(t-i) + n(t) \tag{2.4}$$

Where:

u	- input variable
y	- output variable
x	- state vector
v, w, n	- noise
A, B, C, D	- matrices of parameters
a_i, b_i	- parameters
v	- z of the dynamic system
N_a, N_b	- upper bound of the past history considered

Basic Identification Procedures

The simplest of these methods are deterministic, admit zero mean noise, but cannot express the uncertainty of the estimates caused by the noise. Some examples are:

- Approximation of monotonous step responses by tangent method (Strejc, 1958),
- Repeated integration of differential equations (Strejc 1958, 1961),
- Numerical deconvolution (Cuenod & Sage 1967; Sage & Melsa 1971)

Some identification procedures apply an error cost function but do not assume the existence of noise. These tend to have a number of equations set up for identification equal to the number of model parameters being sought. It is sufficient then to set the partial derivatives with respect to the unknown parameters to zero. Within this category are:

- The use of orthogonal filters (Lampard 1955; Kitamori 1960),
- Model adjustment technique (Marsík 1966,1967; Brunner 1961; Balchen & Høsøien 1966),
- Search methods and gradient methods (Eykhoff 1974; Sage & Melsa 1971).

Stochastic methods are based on the evaluation of a large number of data measured on the system, so a computer is necessary. It is assumed that noise is acting on the system to be identified, is mostly unknown, but satisfies some statistical properties. An increasing amount of information should successively increase the quality of estimates. Relations, or iterative formulas used for estimation are called estimators. Because an infinite number of samples is not available, the estimates can never reach the true values. This means that no solution exists satisfying exactly the selected system model for all input / output data sets, but only in the sense of the chosen error cost function. The stochastic approaches of identification are categorised according to the error cost function chosen for the estimation quality:

- Least squares (Strejc 1980),
    - Ordinary least squares (Anderson 1958; Levin 1960; Deutsch 1965),
    - Weighted least squares (Deutsch 1965),
    - Markov estimate (Deutsch 1965),
    - Stochastic approximation (Robbins & Monro 1951; Keifer & Wolfowitz 1952; Blum 1954; Dvoretzky 1965),
    - Kalman-Bucy filtering (Kalman 1960; Kalman & Bucy 1961)
    - Instrumental variable method (Kendal & Stuart 1961; Young 1970),

- Generalised least squares (Eykhoff 1967; Clarke 1967; Hastings-James & Sage 1969),
- Extended least squares (Panuska, 1968; Young 1972),
- Square-root filtering (Kaminski 1971; Peterka 1975; Kárny 1976),

- Maximum liklihood estimation (Anderson 1958; Deutsch 1965; Sage & Melsa 1971; Åström 1979),
- Baye's estimation (Ho & Lee 1964; Peterka 1976, 1978).

2.2.3 Non-linear System Identification

Mathematical Models

Non-linear models can be thought of as linear models, but with unknown parameters in terms of difference equation coefficients. Simplifications can be made through various assumptions, and the state-space representation can be used. In this case the state vector x(t) is extended by the parameter θ(t) so that the new state vector is:

$$\xi(t) = \begin{bmatrix} x(t) \\ \theta(t) \end{bmatrix} \tag{2.5}$$

Basic Identification Procedures

The most important methods are:

- Gradient Techniques (Sage 1968; Bryson & Ho 1969; Bekey & Karplus 1968),
- Stochastic Approximation (Robbins & Monro 1951; Keifer & Wolfowitz 1952; Blum 1954; Dvoretzky 1956),
- Quazilinearization ((Henrici 1962; Kumar & Sridhar 1964; Bellman & Kalaba 1965; Detchmendy & Sridhar 1965; Sage & Burt 1965; Sage & Smith 1966),
- Difference Approximation,
- Non-linear Filtering (Sage & Melsa 1971; Jazwinski 1970),
- Invariant imbedding (Sage & Melsa 1971).

2.2.4 Classification of Estimation Methods

The following section has mainly been extracted from "Parameter Estimation for Continuous-Time Models - A Survey" (Young 1981).

Output Error (OE) Methods

Here, the parameters are chosen so that they minimise the error between the model output, denoted by $\hat{y}$, and the observed output y, i.e. e(t) = y(t) - $\hat{y}$ (t). A system model can be represented as a state-space polynomial matrix description of the form:

$$\begin{aligned} A(s)x(t) &= B(s)u(t) \quad \text{(i)} \quad \} \\ y(t) &= x(t) + \xi(t) \quad \text{(ii)} \quad \} \end{aligned} \tag{2.6}$$

or substituting (ii) into (i):

$$y(t) = G(s)u(t) + \xi(t) \tag{2.7}$$

where $G(s) = A^{-1}(s)B(s)$ is the transfer function, $A(s)$ and $B(s)$ are appropriately dimensioned coefficient matrices ($s \equiv d/dt$), $\xi(t)$ is the combined effect of the input and output disturbances at the output of the system, $x(t)$ is the hypothetical 'noise-free' input, and $y(t)$ is the output.

In a SISO case of equation (2.6), the error can be defined as:

$$e(t) = y(t) - \frac{\hat{B}}{\hat{A}}u(t) \tag{2.8}$$

where $\hat{B}$ and $\hat{A}$ are the estimates of $B(s)$ and $A(s)$ respectively. See figure 2.1a.

Equation Error (EE) Methods

The EE approach derives from an analogy with static regression analysis and linear least squares estimation. Here the error function is generated directly from the input-output equations of the model. Referring to equation (2.6), $e(t)$ is defined as:

$$e(t) = \hat{A}y(t) - \hat{B}u(t) \tag{2.9}$$

as illustrated in figure 2.1b.

An alternative 'generalised equation error' (GEE) can be defined to avoid problems that arise from the differentiation of a possible noisy signal. Here the input and output signals are passed through a state variable filter, denoted by $F(s)$ in figure 2.1c.

Prediction Error (PE) Methods

As in the OE case, the error is defined as $e(t) = y(t) - \hat{y}(t)$, but $\hat{y}(t)$ is defined as some 'best prediction' of $y(t)$ given the current estimates of the parameters which characterise the system and the noise models. $\hat{y}(t)$ is the conditional mean of $y(t)$ given all current and past information on the system. In a SISO case of equation (2.6), the error can be defined as:

$$e(t) = \frac{\hat{C}}{\hat{D}}\left[y(t) - \frac{B}{A}u(t) \right] \tag{2.10}$$

as illustrated in figure 2.1d.

Other arrangements are possible, for example, the PE approach within an EE context. The SISO case of this would be formulated:

$$e(t) = \frac{\hat{C}}{\hat{D}\hat{A}}\left[\hat{A}y(t) - \hat{B}u(t) \right] \tag{2.11}$$

which is shown in figure 2.1e.

In general, the PE method is more complex than the OE and EE equivalents since the concurrent estimation of the noise model parameters is required.

Other Methods

The Maximum Likelihood (ML) method is a special case of PE, where the formulation of the error function is restricted by the additional assumption that the stochastic disturbances to the system have specified amplitude probability distribution functions.

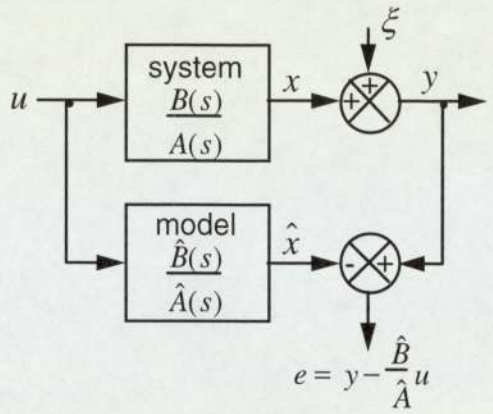


Fig 2a. Output Error (OE)

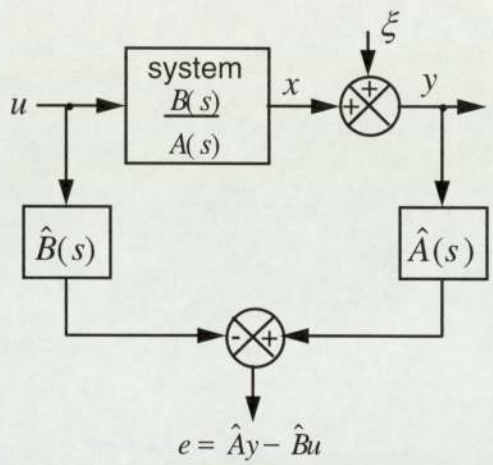


Fig 2b. Equation Error (EE)

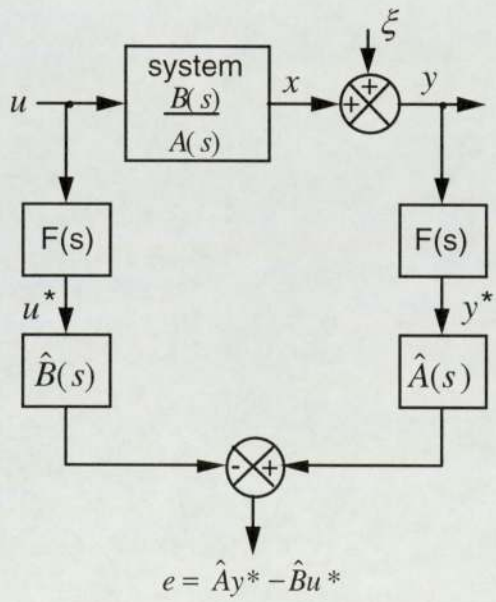


Fig 2c. Generalized Equation Error (GEE)

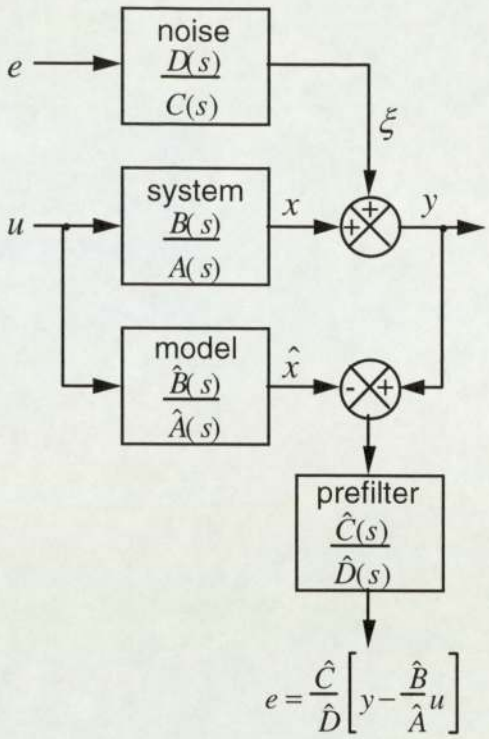


Fig 2d. Prediction Error (PE) in OE context

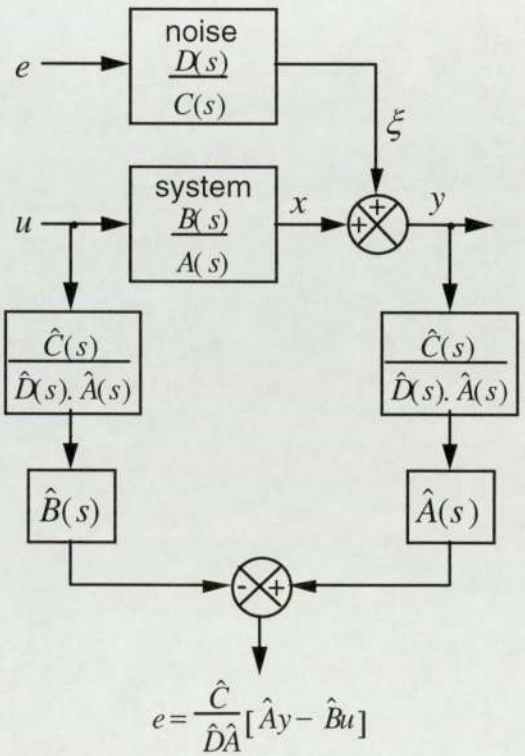


Fig 2e. Prediction Error (PE) in EE context

Figure 2.1 Classification of Estimation Methods.

The Bayesian (B) method is an extension of the ML method, in that deduced information on the probability distributions is included in the formulation of the problem.

2.2.5 The use of Neural Networks in System Identification

The mid to late 1980's saw wide spread interest in Artificial Neural Network (ANN) research due to renewed funding. Neural networks are particularly good at pattern recognition, and this can be extended to system identification. The features which make it suitable are learning, high speed processing of massive amounts of data, and the ability to handle signals with degrees of uncertainty.

Various methods and applications of neural networks have been published. S. Chen and S.A. Billings (1992) presented three network architectures related to the identification of nonlinear discrete-time systems; multi-layer perceptron, radial basis function network and functional-link network. Advantages and disadvantages are discussed and illustrated using simulated and real data. S. Reynold and R. Shoureshi (1992) present a time-domain approach using 'Hopfield' networks (Hopfield, Tank 1985, 1986), and its application to the identification problem of linear time varying or time invariant systems. The model is described which is then referred to parametric identification in state space form, and simulation results show the feasibility of this identification scheme.

D.T. Pham and X. Liu (1993) describe the use of Elman-type (1990) and modified Elman-type recurrent neural networks to identify dynamic systems, and it is shown the behaviour of high order linear and non-linear systems were able to be modelled.
More recently, back propagation techniques have been developed. C. Pal (1994) describes a modified back propagation technique which is claimed to have a faster convergence rate and better accuracy than previous techniques, which is applicable to dynamic system identification.

Other comparable methods have also emerged. C.S. Berger (1994) presented a method of identifying nonlinear dynamic models which exhibits fast convergence and adjusts its memory requirements to cope with the complexity of the problem. W.A. Porter (1995)

and E.B. Kosmatopoulos (1995) describe methods of identification where no prior knowledge of the system is required (black-box). J.H. Chen introduces a new neural network architecture which is claimed to overcome the shortcomings of traditional neural networks, that is slow convergence and long training time amongst others. The most recent contribution found was from G. Lera [1996]. Here a new type of recurrent network for modelling the input-output behaviour of a general class of discrete nonlinear systems is presented. This uses Elman [1990] and Jordan [1986] networks and is based on a state-space description of a nonlinear system.

Perhaps one of the most useful papers, at least to begin with, will be 'Neural Networks and Applications Tutorial' (I. Guyon 1991), because it assumes no prior knowledge on neural networks. This tutorial starts with an introduction to neural networks, goes on to describe and compare different architectures and then gives applications examples and a case study.

CHAPTER 3

3 System Models

A system can be thought of as a collection of objects interlacing with each other and can be quite large and complex. It can be useful to simplify the problem at the expense of completeness and accuracy. For example a system model may be limited to be accurate enough only over a particular range of operation.

The most useful and frequently employed type of system design / analysis is where a system is characterised in terms of subsystems and components and their interactions with each other. Only sufficient detail of component parts is required for the system to operate over a particular range and accuracy. Frequently a combined analytical and experimental approach is necessary where accurate and complete models do not exist, for example during the prototyping of systems.

System analysis is finding the response of a particular system to a specified input or range of inputs. This is important when the system does not exist (in the case of a feasibility study), or when experimental evaluation is impractical or too dangerous for experimentation.

The system design problem is determining the system characteristics to produce a response to a specified input. This is often accomplished by using a parametric model, and calculating the parameters to give the desired response. Most systems are represented by means of specified relationships between the system variables. These can take the form of graphs, tables, differential equations, difference equations or a combination of these. Perhaps the most common system representation is in terms of ordinary differential equations with constant coefficients. This can encompass a large variety of systems and can often be used as approximate representations for systems that fall outside of this category.

3.1 System Classification

A very general mathematical model that encompasses almost all linear systems is shown below.

$$a_0(t)y + a_1(t)\frac{dy}{dt} + ... + a_{n-1}(t)\frac{d^{n-1}y}{dt^{n-1}} + a_n(t)\frac{d^n y}{dt^n} \qquad y(t) = 0 \text{ for all } t<t_0$$

$$= b_0(t)x + b_1(t)\frac{dx}{dt} + ... + b_{m-1}(t)\frac{d^{m-1}x}{dt^{m-1}} + b_m(t)\frac{d^m x}{dt^m} \qquad x(t) = 0 \text{ for all } t<t_0$$

(3.1)

There is one input *x(t)* and one output *y(t)*. The conditions of *x(t)* and *y(t)* are necessary because otherwise the physical system could anticipate an excitation which is impossible. The equation is therefore unique and any specific input signal results in a corresponding unique output signal.

3.2 System Nomenclature

3.2.1 System Order

The order of a system is the highest derivative of the response to appear in equations describing that system. Equation 3.1 is therefore said to be of *n*th order.

3.2.2 Causal / Non-Causal (or physical or non-anticipatory)

A causal system is one in which the present response does not depend upon future values of the input. Non-causal systems do not exist in the real world.

3.2.3 Deterministic / Stochastic

The outputs of a deterministic system can be determined from knowledge of the systems inputs up to that time. A stochastic system has an element of random behaviour, and its outputs are not always a specific function of the input.

3.2.4 Linear / Non-Linear

If all the derivatives of the excitation and response are raised to the first power and there are no products of these derivatives then the system is said to be linear. A linear system is usually so, partly because none of its components parameters change as a function of the excitation applied to it. Any system component however will change its characteristics if the forces applied are large enough. Linearity is therefore an approximation and will be defined within a particular range of normal input magnitudes. If a system contains a non-linear component then the whole system is

treated as non-linear. (An example of this is backlash in a gearbox which would otherwise be a linear system).

3.2.4.1 Superposition

If a system (or system element) is linear, then it will obey the superposition principle. The superposition principle is based upon what happens to the output of a system when two different signals are applied to its input, separately and then summed together (superimposed). If two separate inputs u_1 and u_2 are applied to a system giving rise to outputs y_1 and y_2 respectively, then the output which arises when the sum of these inputs $(u_1 + u_2)$ will be the sum of the individual outputs $(y_1 + y_2)$ only if the system is linear. A simple example of a linear system is a linear electronic amplifier with input u, output y and an amplification factor k. It is apparent that this obeys the superposition theorem as figure 3.1a demonstrates. If the amplifier adds a constant offset c to the output so that $y = Ku + c$, then it does not obey the superposition theorem and is non-linear, as $y_1 + y_2 \neq K(u_1 + u_2)$. It is also worth considering the case that if the amplifier is over-driven the output will reach its supply rails and 'clip'. This is clearly non-linear behaviour.

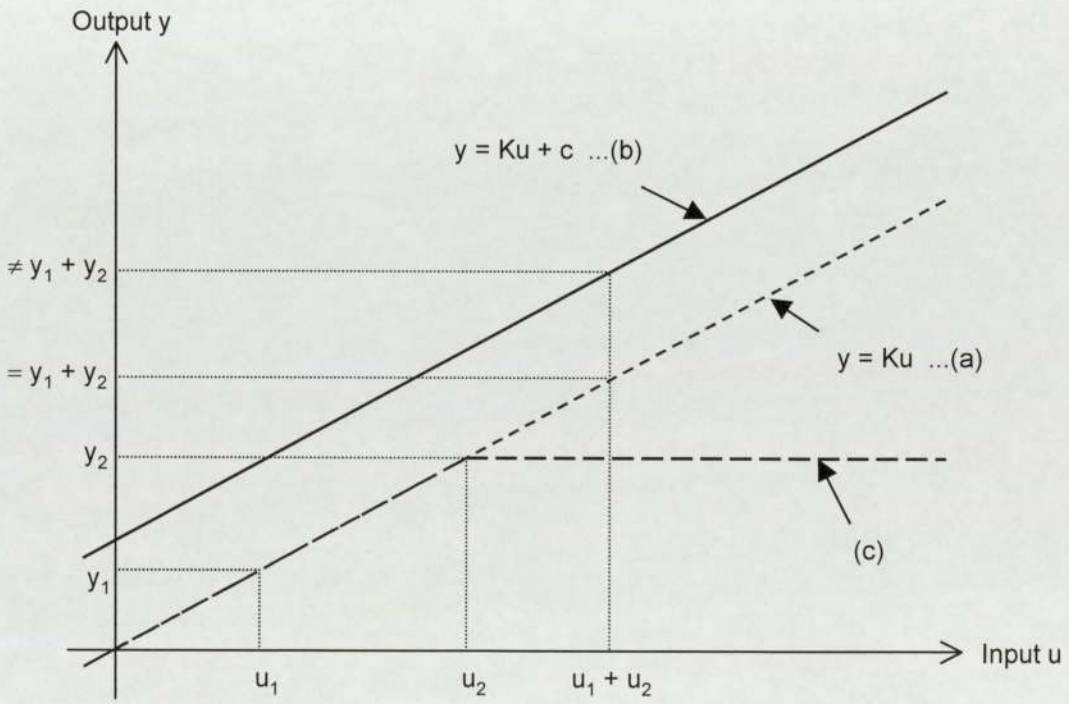


Figure 3.1 Superposition and Linearity (a) Linear System Model, (b) Non-Linear System Model (linear model with offset), (c) Non-Linear Model.

There are therefore two general requirements for obtaining linear models (which obey the principle of superposition), which are:

1) offsets must be removed, and
2) signals must be within a range of normal input magnitudes.

It is worth noting that system (c) in figure 3.1 is linear between the axis intercept and y_2, and is therefore a linear system in this range of operation.

3.2.5 Time-Variant / Time-Invariant

Equation 3.1 represents a time-variant (or time-varying) system since the coefficients of the derivatives are functions of time. Such systems are difficult to analyse because differential equations with non-constant coefficients are difficult to solve. Systems with constant coefficients are known as fixed, time-invariant or stationary.

3.2.6 Lumped-Parameter / Distributed Parameter

To simplify system analysis it is usual to consider each element as a single property or function. For example, an electrical inductor is assumed to have pure inductance with no resistance or capacitance. If the resistance is significant then a separate model of a pure resistor is made to represent it. Such a system is known as a lumped parameter system, and each element has one independent variable, time. If more than one independent variable is considered, partial differential equations arise making analysis harder. A lumped-parameter system is usually only valid if the physical size of the system is of no concern, since excitations propagate through it instantaneously. A transmission line is an example of a distributed-parameter system.

When lumping parameters it is important to make valid assumptions. For example, a model that includes every minor detail would take a long time to develop and may be difficult to achieve for limited returns over a simplified model. An over-simplified model however may bear no dynamic resemblance to the original system.

3.2.7 Continuous Time / Discrete Time

Continuous-time systems are usually represented by equations where inputs and outputs are represented for all values of time. Discrete-time systems are represented by

equations where inputs and outputs are represented for discrete values of time. All physical systems are continuous-time, but it is sometimes convenient to consider a systems behaviour at discrete instants in time, for example when using a digital computer. Continuous-time systems are usually represented by differential equations, and discrete-time systems by difference equations. Discrete-time systems are often more convenient to use and construct a model of a system with non-linearity or time-variation.

3.3 Modelling the Plant

The process of mathematical modelling (for control system design) can be thought of as an iterative process roughly comprising 6 stages:

1) Identifying the various inputs, outputs and disturbances,
2) Produce an idealised mathematical representation of the plant,
3) Develop this representation (model) for the required accuracy,
4) If necessary obtain data (possibly by performing experiments) to find unknown parameters,
5) Re-do steps 2-4 until the model sufficiently represents the plant,
6) Simplify the model for the operating limits of the control problem. This may include linearising non-linear equations for a specific operating range and removing redundant detail.

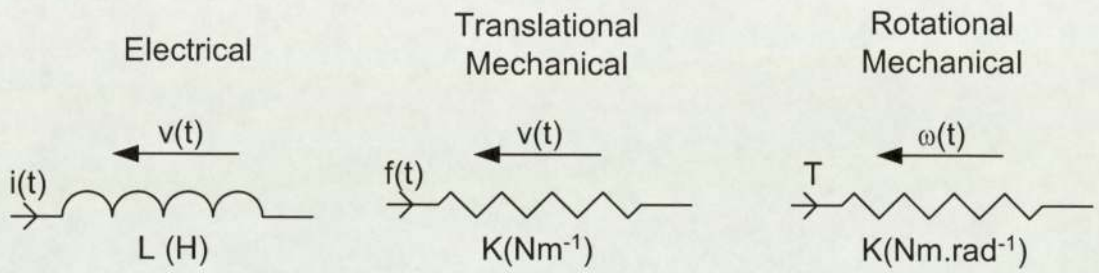
The extent to which the above stages may be carried out depends upon the type of plant under examination. Mathematical modelling is not an exact science and there may well be more than one model which represent the plant sufficiently. This chapter will investigate steps 1, 2, 3 and 6, and chapter 5 will investigate steps 4, 5 and 6.

3.3.1 Lumped Parameter Models

A lumped-parameter model is one in which certain aspects of the system being modelled are imagined to be lumped at a single location, for example a pendulum may be considered to be a rod of no mass with a mass concentrated at one end. This type of

model makes assumptions some of which were mentioned in section 3.2.6. Furthermore the system must be linear, time-invariant and deterministic.

When making lumped parameter models it is useful to use an analogy so that the models are of consistent form and different physical elements may be described by the same form of equation. There are many analogies and one will briefly be considered here, the *force-current* analogy. Examples of analogous physical elements of this kind are:

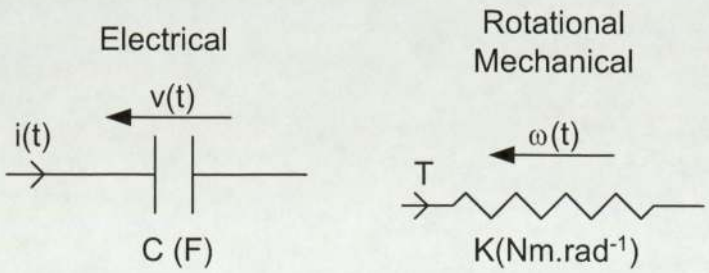


where…

$$i(t) = \frac{1}{L}\int v(t).dt \qquad f(t) = K\int v(t).dt = K.x(t).dt \qquad T(t) = K\int \omega(t).dt = K.\theta$$

Here the *force* and *current* and corresponding equations can be thought to be analogous. The *through* variables are current, force and torque, and the *across* variables are voltage, velocity and angular velocity respectively. Many physical analogues are possible, but of most interest to this project are the rotational mechanical elements, and occasional reference to their electrical counterparts.

It is also possible to use voltage as an analogue to torque. The electrical equivalent of the above rotational mechanical spring becomes:



where…

$$v(t) = \frac{1}{C}\int i(t).dt \qquad\qquad T(t) = K\int \omega(t).dt$$

3.3.2 Modelling a Lumped-Parameter Rotational System

The mechanical system shown in figure 3.2 has two degrees of freedom. This means that two measurements are required to specify the position (not the state) of all elements in the system. These measurements are θ_1 and θ_2.

The shaft has a torsional stiffness k ($N\ m\ rad^{-1}$), the inertias have values of J_1 and J_2, and the rotational dampers have friction coefficients B_1 and B_2. The first step in the modelling process is to identify the inputs and outputs. The system is forced by one input torque (T) and has two outputs θ_1 and θ_2. The next step is to produce an idealised mathematical representation of the plant.

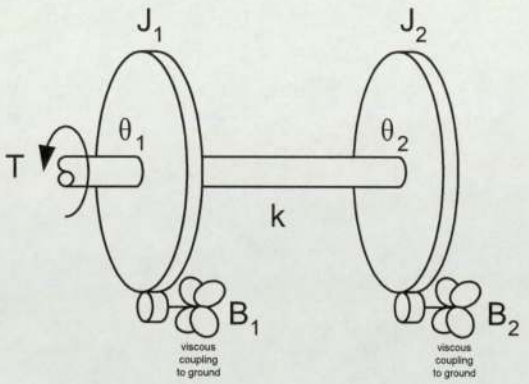


Figure 3.2 Rotational Mechanical System

The system first needs to be broken into parts which are easier to work with than the system as a whole, and these are shown below in figure 3.3. It is important when forming the equations that displacement and forces are all measured in the same direction, so that any acting in the opposite direction are made negative. For example, the opposing torque of the shaft acting on J_1, and the inertia of J_1 are both acting in the opposite direction to input torque and hence will have the opposite sign in that equation of motion.

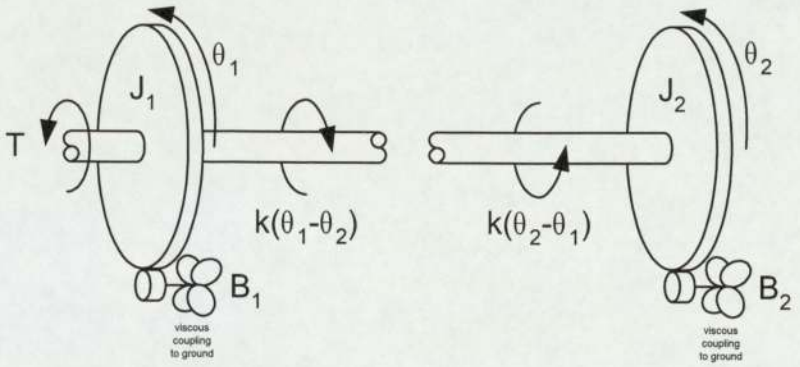


Figure 3.3 Rotational Mechanical System Components

In this example there are two equations of motion derived and these are:

$$T - J_1\frac{d^2\theta_1}{dt^2} - B_1\frac{d\theta_1}{dt} - k(\theta_1 - \theta_2) = 0$$

$$\text{or} \quad J_1\frac{d^2\theta_1}{dt^2} + B_1\frac{d\theta_1}{dt} + k(\theta_1 - \theta_2) = T \tag{3.2}$$

and

$$J_2\frac{d^2\theta_2}{dt^2} + B_2\frac{d\theta_2}{dt} - k(\theta_1 - \theta_2) = 0 \tag{3.3}$$

3.3.2.1 Electrical Equivalent of the Rotational System

It is sometimes convenient to translate systems between different physical types of plant, for example mechanical to electrical or vice-versa. For the system shown in figure 3.2 the through variable is torque and the across variable is angular velocity. Although the system is shown in free space, it is assumed that the measurements are made to some stationary reference point. The system circuit diagram for the rotational mechanical system is shown in figure 3.4. It can be seen by inspection that the rotational mechanical components have electrical "equivalents". The electrical equivalent of the inertias are capacitors, and the electrical equivalent of the shaft (or torsion spring) is an inductor. An equivalent electrical circuit is shown in figure 3.5 where the through variable is current and the across variable is voltage.

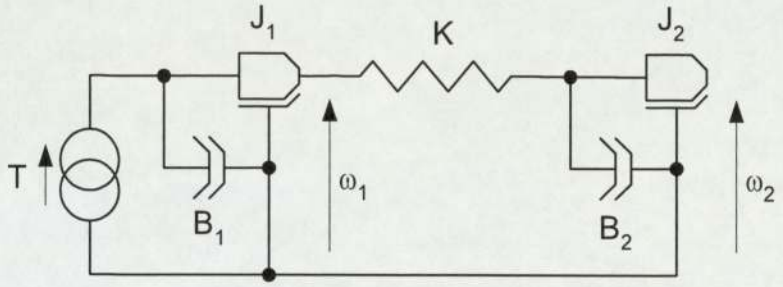


Figure 3.4 System Circuit Diagram for the Rotational Mechanical System

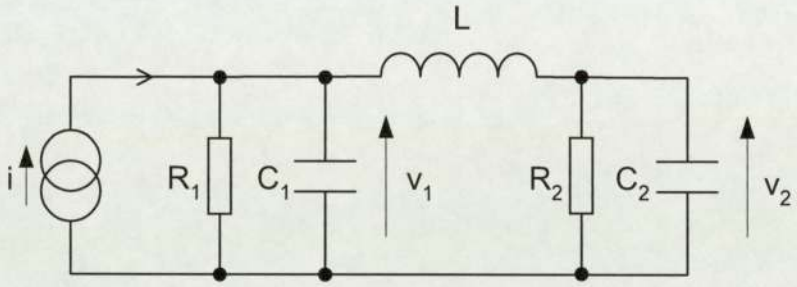


Figure 3.5 Electrical Equivalent of the Rotational Mechanical System

The equations for the two equivalent system components are:

Mechanical:

$$J_1 \frac{d\omega_1}{dt} + B_1.\omega_1 + k\int(\omega_1 - \omega_2).dt = T \quad (3.4)$$

Electrical:

$$C_1 \frac{dv_1}{dt} + \frac{v_1}{R_1} + \frac{1}{L}\int(v_1 - v_2).dt = i \quad (3.5)$$

and

$$J_2 \frac{d\omega_2}{dt} + B_2.\omega_2 - k\int(\omega_1 - \omega_2).dt = 0 \quad (3.6)$$

$$C_2 \frac{dv_2}{dt} + \frac{v_2}{R_2} - \frac{1}{L}\int(v_1 - v_2).dt = 0 \quad (3.7)$$

where $\omega_n = \frac{d\theta_n}{dt}$

3.4 State-space Models

The equations of motion shown in the example in section 3.3.2 are both *second order* equations because the highest derivative involved is a second derivative. It is possible to solve ordinary linear first and second order differential equations directly, but complex systems may contain equations of many orders, and a systematic approach is required. The state-space method is such an approach.

The state-space method works by replacing high-order differential equations with a set of first-order differential equations. Equation 3.1 is an n^{th} order differential equation that can also be represented by a set of first-order differential equations, known as the normal-form of system equations. For an n^{th} order system there will be n simultaneous first-order differential equations and n unknown time functions to be solved for. These time functions are called state-variables, and their values at any instant in time specify the state of that system. These variables may represent signals in the system, or they may be a set of abstract quantities, as there are many ways in which the state variables may be selected, but must obey the following general rules:

- They must be linearly independent – one cannot simply be a combination of others
- There must be enough to completely specify the dynamic behaviour of the system
- They cannot be inputs to the system

For a general n^{th} - order linear system with state variables $q_1(t)$, $q_2(t)$ ... $q_n(t)$ and one input $x(t)$ a set of simultaneous 1^{st} order, linear differential equations can be written in the form:

$$\frac{dq_1(t)}{dt} = a_{11}(t)q_1(t) + a_{12}(t)q_2(t) + ... + a_{1n}(t)q_n(t) + b_1(t)x(t)$$

$$\frac{dq_2(t)}{dt} = a_{21}(t)q_1(t) + a_{22}(t)q_2(t) + ... + a_{2n}(t)q_n(t) + b_2(t)x(t)$$

.

.

.

$$\frac{dq_n(t)}{dt} = a_{n1}(t)q_1(t) + a_{n2}(t)q_2(t) + ... + a_{nn}(t)q_n(t) + b_n(t)x(t) \qquad (3.8)$$

A single output y(t) can be represented as:

$$y(t) = c_1(t)q_1(t) + c_2(t)q_2(t) + ... + c_n(t)q_n(t) \qquad (3.9)$$

The advantages of this representation is that it can be written in matrix form which can be applied to all systems of all orders with little additional complexity. It is less

notationally complex than the general model of equation 3.1 and easier to implement models of this kind on a computer. If the system is fixed (time-invariant or stationary) the coefficients $a_{ij}(t)$, $b_i(t)$ and $c_i(t)$ are constants instead of being functions of time.

The states are written as the column vector $x = [x_1\ x_2\ x_3\ \ldots\ x_n]^T$, and the input and output vectors are u and y respectively. The state-space model is made up of the following pair of equations:

$$\dot{x} = Ax + Bu \qquad \text{- state equation} \qquad (3.10)$$

$$y = Cx + Du \qquad \text{- output equation} \qquad (3.11)$$

where, if the system is of n-th order with m inputs and p outputs,

x = $n \times 1$ state vector (column vector)
u = m × 1 input vector (column vector)
y = p × 1 output vector (column vector)
A = n × n system (or plant) matrix
B = n × m input (or distribution) matrix
C = the p × n output (or measurement) matrix
D = the p × m feed forward (or output distribution) matrix

The vector $\dot{x}$ is the time derivative of the vector x, comprising the derivative of each individual element. The quantities A, B, C and D are purely numerical matrices (except in the case of time-variant or time-varying models). A block diagram of a state space system [1] is shown in figure 3.6.

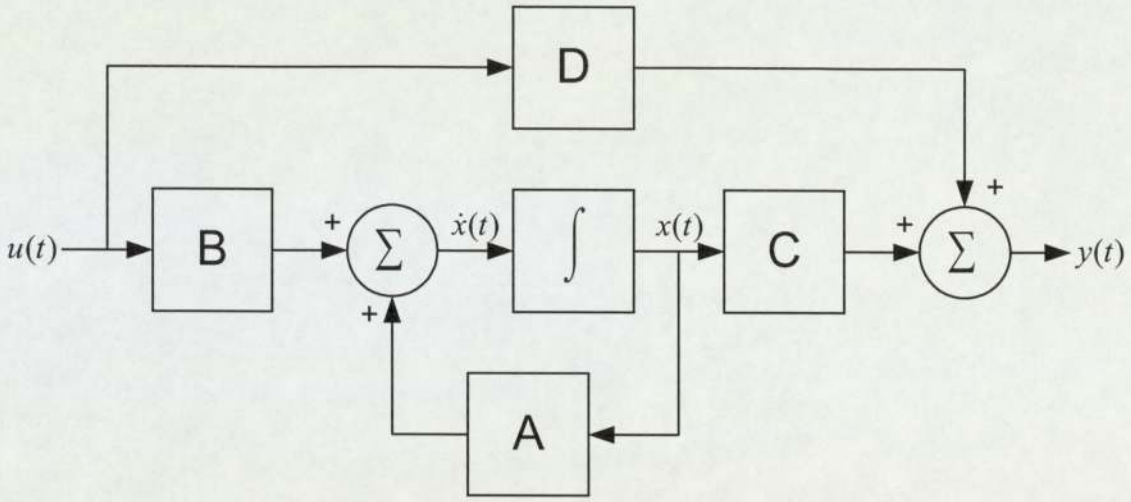


Figure 3.6 Block Diagram for State Space System

3.4.1 State-Space Model of the Rotational System

The rotational system described in section 3.3.2 can be easily converted from the ordinary linear differential equations into a state-space model. Equations 3.2 and 3.3 can be written as:

$$J_1\ddot{\theta}_1 + B_1\dot{\theta}_1 + k\theta_1 - k\theta_2 = T \qquad \text{and} \qquad (3.12)$$

$$J_2\ddot{\theta}_2 + B_2\dot{\theta}_2 - k\theta_1 + k\theta_2 = 0 \qquad (3.13)$$

The system is a second-order two degree-of-freedom system, so four state-variables will be required. These can be selected as follows:

$$x_1 = \theta_1 \qquad x_2 = \dot{\theta}_1 \qquad x_3 = \theta_2 \qquad x_4 = \dot{\theta}_2$$

in which case:

$$x = \begin{bmatrix} \theta_1 \\ \dot{\theta}_1 \\ \theta_2 \\ \dot{\theta}_2 \end{bmatrix} \qquad \text{and} \qquad \dot{x} = \begin{bmatrix} \dot{\theta}_1 \\ \ddot{\theta}_1 \\ \dot{\theta}_2 \\ \ddot{\theta}_2 \end{bmatrix}$$

So, rearranging 3.12 and 3.13 so that $\ddot{\theta}_1$ and $\ddot{\theta}_2$ are on the LHS:

$$\ddot{\theta}_1 = -\frac{k}{J_1}\theta_1 + \frac{k}{J_1}\theta_2 - \frac{B_1}{J_1}\dot{\theta}_1 + \frac{1}{J_1}u \quad \text{where } u \text{ is torque T(t)} \quad \text{and}$$

$$\ddot{\theta}_2 = \frac{k}{J_2}\theta_1 - \frac{k}{J_2}\theta_2 - \frac{B_2}{J_2}\dot{\theta}_2$$

the matrices A, B, C and D can be defined as:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{k}{J_1} & -\frac{B_1}{J_1} & \frac{k}{J_1} & 0 \\ 0 & 0 & 0 & 1 \\ \frac{k}{J_2} & 0 & -\frac{k}{J_2} & -\frac{B_2}{J_2} \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ -\frac{1}{J_1} \\ 0 \\ 0 \end{bmatrix} \qquad C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad D = 0$$

The state equation ($\dot{x} = Ax + Bu$) and output equation ($y = Cx + Du$) for this system will look like:

$$\begin{bmatrix} \dot{\theta}_1 \\ \ddot{\theta}_1 \\ \dot{\theta}_2 \\ \ddot{\theta}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{k}{J_1} & -\frac{B_1}{J_1} & \frac{k}{J_1} & 0 \\ 0 & 0 & 0 & 1 \\ \frac{k}{J_2} & 0 & -\frac{k}{J_2} & -\frac{B_2}{J_2} \end{bmatrix} \cdot \begin{bmatrix} \theta_1 \\ \dot{\theta}_1 \\ \theta_2 \\ \dot{\theta}_2 \end{bmatrix} + \begin{bmatrix} 0 \\ -\frac{1}{J_1} \\ 0 \\ 0 \end{bmatrix} \cdot u \qquad (3.14)$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \theta_1 \\ \dot{\theta}_1 \\ \theta_2 \\ \dot{\theta}_2 \end{bmatrix} \qquad (3.15)$$

Note that C is a unity matrix, and D is zero. The output y is therefore equal to x which comprises $\theta_1, \dot{\theta}_1, \theta_2$ and $\dot{\theta}_2$. This will be used later for a simulation of the system.

3.4.2 State-Space Models from Ordinary Differential Equations

Equation 3.1 is an *n*th order ordinary differential equation representing a system with a single input u and a single output y and is a stationary system if the coefficients are constant. Different choices of state variables produce different but equivalent state models of the same system. It can be shown that a special set of choices leads to what is know as *canonical state models* [1] which reflect a more ordered and structured approach to state model formulation. (Generally speaking canonical means irreducible, and usually refers to making an equation look simpler. In the case of a system of equations which are coupled, the canonical form comprises an equivalent set of equations which are not coupled). There are four common canonical state models which are known as the *observable*, *controllable* (also known as *standard* canonical form or *phase variable* form), *observability* and *controllability* canonical forms.

Equation 3.16 is a normalised ($a_n = 1$) ordinary differential equation representing an *n*th order system with a single input u and a single output y (as equation 3.1 but with constant coefficients).

$$a_0 y + a_1 \frac{dy}{dt} + ... + a_{n-1} \frac{d^{n-1}y}{dt^{n-1}} + \frac{d^n y}{dt^n} = b_0 u + b_1 \frac{du}{dt} + ... + b_{m-1} \frac{d^{m-1}u}{dt^{m-1}} + b_m \frac{d^m u}{dt^m} \qquad (3.16)$$

If $m = n - 1$ it can be represented by the four different but equivalent canonical state models, 3.17, 3.18, 3.19 and 3.20 below.

Observable canonical form:

$$\begin{bmatrix} \dot{x}_1 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} 0 & \cdots & \cdots & \cdots & \cdots & 0 & -a_0 \\ 1 & \ddots & & & & \vdots & \vdots \\ 0 & \ddots & \ddots & & & \vdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & & \vdots & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \vdots & \vdots \\ \vdots & & & \ddots & \ddots & 0 & \vdots \\ 0 & \cdots & \cdots & \cdots & 0 & 1 & -a_{n-1} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} b_0 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ b_m \end{bmatrix} u \qquad (3.17a)$$

$$y = \begin{bmatrix} 0 & \cdots & \cdots & \cdots & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ x_n \end{bmatrix} \qquad (3.17b)$$

Controllable canonical form:

$$\begin{bmatrix} \dot{x}_1 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \cdots & \cdots & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & 1 \\ -a_0 & \cdots & \cdots & \cdots & \cdots & \cdots & -a_{n-1} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} 0 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ 0 \\ 1 \end{bmatrix} u \qquad (3.18a)$$

$$y = \begin{bmatrix} b_0 & \cdots & \cdots & \cdots & \cdots & \cdots & b_m \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ x_n \end{bmatrix} \qquad (3.18b)$$

Observability canonical form:

$$\begin{bmatrix} \dot{x}_1 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \cdots & \cdots & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & 1 \\ -a_0 & \cdots & \cdots & \cdots & \cdots & \cdots & -a_{n-1} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} \beta_m \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \beta_0 \end{bmatrix} u \qquad (3.19a)$$

$$y = \begin{bmatrix} 1 & 0 & \cdots & \cdots & \cdots & \cdots & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ x_n \end{bmatrix} \qquad (3.19b)$$

β_1, β_2 and β_3 are called the *Markov parameters* of the system and are given by

$\beta_m = b_m$, $\beta_{m-1} = b_{m-1} - a_{n-1}b_m$, $\beta_{m-2} = a_n{}^2 b_m - a_n b_{m-1} - a_{n-1}b_m + b_{m-2}$ etc…

Controllability canonical form:

$$\begin{bmatrix} \dot{x}_1 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} 0 & \cdots & \cdots & \cdots & \cdots & 0 & -a_0 \\ 1 & \ddots & & & & \vdots & \vdots \\ 0 & \ddots & \ddots & & & \vdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & & \vdots & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \vdots & \vdots \\ \vdots & & & \ddots & \ddots & 0 & \vdots \\ 0 & \cdots & \cdots & \cdots & 0 & 1 & -a_{n-1} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ \vdots \\ \vdots \\ \vdots \\ 0 \end{bmatrix} u \qquad (3.20a)$$

$$y = \begin{bmatrix} \beta_m & \cdots & \cdots & \cdots & \cdots & \cdots & \beta_0 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ x_n \end{bmatrix} \qquad (3.20b)$$

where the *Markov parameters* of the system are the same as for the observability form.

Canonical state models provide an organised way of representing a high order scalar differential equation as opposed to an arbitrary representation, and they are beneficial for controller design, identification theory and system analysis.

3.4.3 Alternative State-space Model of the Rotational System

It is possible to combine equations 3.12 and 3.13 into one fourth-order differential equation by eliminating either θ_1 from 3.12 or θ_2 from 3.13 and replacing with differential terms consisting of either θ_2 or θ_1 and derivatives respectively.

Equation 3.12 can be rearranged

$$\theta_2 = \frac{J_1}{k}\ddot{\theta}_1 + \frac{B_1}{k}\dot{\theta}_1 + \theta_1 + \frac{1}{k}T \tag{3.21}$$

and differentiated twice to give

$$\dot{\theta}_2 = \frac{J_1}{k}\dddot{\theta}_1 + \frac{B_1}{k}\ddot{\theta}_1 + \dot{\theta}_1 + \frac{1}{k}\dot{T} \tag{3.22}$$

and

$$\ddot{\theta}_2 = \frac{J_1}{k}\ddddot{\theta}_1 + \frac{B_1}{k}\dddot{\theta}_1 + \ddot{\theta}_1 + \frac{1}{k}\ddot{T} \tag{3.23}$$

substituting 3.21 and 3.13 into equation 3.12

$$J_2\left(\tfrac{J_1}{k}\ddddot{\theta}_1 + \tfrac{B_1}{k}\dddot{\theta}_1 + \ddot{\theta}_1 + \tfrac{1}{k}\ddot{T}\right) + B_2\left(\tfrac{J_1}{k}\dddot{\theta}_1 + \tfrac{B_1}{k}\ddot{\theta}_1 + \dot{\theta}_1 + \tfrac{1}{k}\dot{T}\right) + k\theta_1 + k\left(\tfrac{J_1}{k}\ddot{\theta}_1 + \tfrac{B_1}{k}\dot{\theta}_1 + \theta_1 + \tfrac{1}{k}T\right) = 0 \tag{3.24}$$

expanded and like terms grouped gives

$$(B_1 + B_2)\cdot\dot{\theta}_1 + \left(J_1 + J_2 + \tfrac{B_1B_2}{k}\right)\cdot\ddot{\theta}_1 + \tfrac{B_1J_2 + J_1B_2}{k}\cdot\dddot{\theta}_1 + \tfrac{J_1J_2}{k}\cdot\ddddot{\theta}_1 = T + \tfrac{B_2}{k}\dot{T} + \tfrac{J_2}{k}\ddot{T} \tag{3.25}$$

which is of the form of equation 3.16 where $u = T$, $y = \theta_1$, $n = 4$ and the coefficients are:

$$a_0 = 0,\ a_1 = (B_1 + B_2),\ a_2 = \left(J_1 + J_2 + \frac{B_1 B_2}{k}\right),\ a_3 = \frac{B_1 J_2 + J_1 B_2}{k},\ a_4 = \frac{J_1 J_2}{k},$$

$$b_0 = 1,\ b_1 = \frac{B_2}{k},\ b_2 = \frac{J_2}{k}\ \ b_3 = 0.$$

To put this in one of the canonical state-space forms both sides of the equation need to be divided by a_0. Any one of the canonical state-space forms may then be used to create the required model. For example, the observable state-space model of equation 3.17:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -\frac{kB_1 + kB_2}{J_1 J_2} \\ 0 & 1 & 0 & -\left(\frac{kJ_1 + kJ_2 + B_1 B_2}{J_1 J_2}\right) \\ 0 & 0 & 1 & -\frac{B_1 J_2 + B_2 J_1}{J_1 J_2} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} \frac{k}{J_1 J_2} \\ \frac{B_2}{J_1 J_2} \\ \frac{1}{J_1} \\ 0 \end{bmatrix} \cdot u \qquad (3.26a)$$

$$y = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \qquad (3.26b)$$

3.4.4 Other forms

For certain system analysis it is sometimes useful to expand the coefficient matrices to separate the physical elements. For example, the rotational system equation shown in equation 3.14 could be re-written:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & J_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & J_2 \end{bmatrix} \cdot \begin{bmatrix} \dot{\theta}_1 \\ \ddot{\theta}_1 \\ \dot{\theta}_2 \\ \ddot{\theta}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -k & 0 & k & 0 \\ 0 & 0 & 0 & 1 \\ k & 0 & -k & 0 \end{bmatrix} \cdot \begin{bmatrix} \theta_1 \\ \dot{\theta}_1 \\ \theta_2 \\ \dot{\theta}_2 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -B_1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -B_2 \end{bmatrix} \cdot \begin{bmatrix} \theta_1 \\ \dot{\theta}_1 \\ \theta_2 \\ \dot{\theta}_2 \end{bmatrix} + \begin{bmatrix} 0 \\ -1 \\ 0 \\ 0 \end{bmatrix} \cdot u \qquad (3.27)$$

Although strictly speaking this is no longer state-space form, the grouping of the Inertias, Spring Constants and Friction Coefficients can be helpful in system analysis.

3.5 Initial Conditions

Sections 3.3 and 3.4 have been concerned with creating differential-equation models of systems. Used in system analysis it is necessary to specify the initial conditions of energy storage in the system, that is the state of the system. The values of the initial conditions are dictated by the nature of the system analysis. In this text the behaviour of a systems response to a forcing input is of most interest, rather than the systems transient response due to non-zero initial conditions. If a linear system is stable, the effect of the initial conditions on the output decreases with time. It is therefore common to set the initial conditions to zero. The effect of the initial conditions on non-linear systems however does not necessarily decrease with time. Initial conditions will be discussed in more detail later in the text in the context of specific systems.

3.6 Fourier Series and Transforms

Mathematical models described so far have been *time-domain* models for linear systems, in the form of ordinary linear differential equations arranged in a convenient form. There are many systems which are periodic in nature, for example the current and voltage in an alternating-current electrical circuit, or the displacement, velocity and acceleration of a slider-crank mechanism. The Fourier Series is a mode of analysing a periodic function in terms of its constituent sine and cosine components. This can be further extended to the Fourier-transform, which provides a method for creating and analysing *frequency-domain* models. Their use in this text is limited and for that reason they are not studied here in much detail.

Fourier series can be applied to many more functions than Taylor's and Maclaurin's series, and while the theory of analysis is complicated, the application of these series is quite simple.

3.6.1 Fourier Series Representation of Time Functions

If a finite, one-valued function $f(x)$ recurs periodically over successive intervals of 2π it is possible to represent it as a series of the form:

$$f(x) = a_0 + a_1 \cos(x) + a_2 \cos(2x) + \ldots + a_n \cos(nx)$$

$$+ b_1 \sin(x) + b_2 \sin(2x) + ... + b_n \sin(nx) \qquad (3.22)$$

which can be written:

$$f(x) = a_0 + \sum_{n=1}^{\infty} (a_n \cos nx + b_n \sin nx)$$

(3.23)

where a_0, a_1, …, a_n, and b_0, b_1, …, b_n, are real constants.

It can be shown that over a range $-\pi$ to $+\pi$ the coefficients are given by:

$$a_0 = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x).dx$$

$$a_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x).\cos(nx).dx \qquad \text{(where } n = 1, 2, 3, \ldots\ldots\text{)} \qquad (3.24)$$

$$b_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x).\sin(nx).dx \qquad \text{(where } n = 1, 2, 3, \ldots\ldots\text{)} \qquad (3.25)$$

3.6.1.1 Convergence of Fourier Series

The rate at which the partial sum of a Fourier series approaches the value of the function is important as it dictates the number of terms necessary to obtain the desired accuracy. Although it is beyond the scope of this text to discuss convergence and compare to other series, it is worth noting that a set of conditions exist known as the Dirichlet conditions [4] which assure the desired convergence of a Fourier series.

Dirichlet Conditions

These essentially restrict the function to be finite (convergent), have a finite number of maxima and minima in any finite time period, and have a finite number of discontinuities in any finite time period.

3.6.2 Fourier Transforms

For signals that exist for only a finite period of time the Fourier series expansion becomes difficult and confusing to use, mainly because the finite signal is not easily represented by a set of continuous signals. A way around this problem exists and is called the *Fourier transform* [4]. By increasing the period of the function $f(t)$ so that the limit $T\rightarrow\infty$, all of the time function will be included in the series representation. The validity of the resulting expression depends on the nature of $f(t)$ and the mathematical operations conducted in the development. Using the complex exponential form of the Fourier transform (substituting $e^{\pm jn\omega_0 t} = \cos n\omega_0 t \pm j \sin n\omega_0 t$ into equation 3.23) the *Fourier transform* of *f(t)* is given by:

$$\mathcal{F}\{f(t)\} = \mathrm{F}(\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}dt \tag{3.26}$$

and the *inverse Fourier transform* written as:

$$\mathcal{F}^{-1}\{\mathrm{F}(\omega)\} = f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega)e^{j\omega t}d\omega \tag{3.27}$$

and the functions $f(t)$ and $\mathrm{F}(\omega)$ are called Fourier transform pairs. For a function $f(t)$ to have a Fourier transform it must adhere to the Dirichlet conditions (i.e. be convergent). Fourier transform pairs can be created using the above integral, and tables are available containing common Fourier transform pairs.

3.7 Laplace Transform Models

The Fourier transform is a useful tool for the analysis of signals and systems, but its use is limited to functions which can have such a transform, functions which converge (meet the Dirichlet conditions). Unfortunately this excludes many useful functions but by introducing a convergence factor (σ) these functions then become integrable. This is done by integrating under a new variable s where $s = \sigma + j\omega$ and is assumed to be positive and large enough to ensure the product $f(t)e^{-st}$ converges to zero as $t \rightarrow \infty$. Known as the Laplace-transform, it provides the mathematical foundation for most

frequency-domain models, and is of particular importance to control techniques. The Laplace transform method is a substitutional one in which linear differential equations are transformed into the complex-frequency or Laplace domain, manipulated and then inverse-transformed back. The mathematical manipulation is simplified as the integration operation in the time domain is replaced by algebraic manipulation of the transformed equations.

The basic Laplace transformation of a time signal $f(t)$ is defined as:

$$\mathcal{L}\{f(t)\} = F(s) = \int_0^\infty f(t)e^{-st}dt \qquad (3.28)$$

and the following notation is used

$\mathcal{L}\{x(t)\} = X(s) = \bar{x}$ and conversely $\mathcal{L}^{-1}\{\{X(s)\} = \mathcal{L}^{-1}\{\bar{x}\} = x(t)$.

In equation 3.22 the exponent st must be dimensionless and therefore s has the dimension of $time^{-1}$. Since s is a complex quantity it is referred to as the *complex frequency*. Laplace transform tables are use to transform equations to and from the complex frequency domain.

3.7.1 Time response of an unforced system

Consider the system shown if figure 3.7 below.

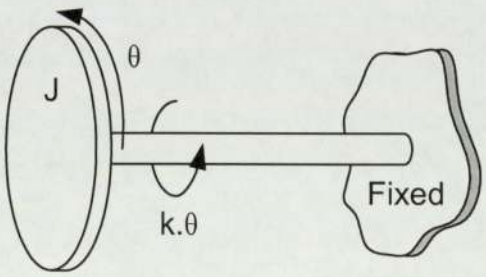


Figure 3.7 Simple unforced lumped-parameter system

This can be described by the differential equation 3.29

$$J\ddot{\theta} + k.\theta = 0 \tag{3.29}$$

The Laplace transform of this is

$$J\left(s^2\overline{\theta} - s\theta(0) - \dot{\theta}(0)\right) + k\overline{\theta} = 0 \tag{3.30}$$

If, for the sake of argument let $J = 1\times10^3 kg.m^2$, $k = 49.28Nm^{-1}$ and the initial conditions $\theta(0) = 0$, $\ddot{\theta} = 25$. Substituting these into equation 3.30 and rearranging for $\overline{\theta}$ gives

$$\overline{\theta} = \frac{25s}{s^2 + 49.28\times10^3}$$

$$\overline{\theta} = \frac{25s}{s^2 + 222^2} \tag{3.31}$$

Using tables to find the inverse Laplace transform of this provides the time-domain solution for θ:

$$\theta = 25\cos(\omega.t) \qquad \text{where} \qquad \omega = 222 \tag{3.32}$$

This is the unforced response of the system given the above initial conditions known as a transient response. It is, as expected an undamped oscillation (with frequency 35.33 Hz). It is a cosine function as the initial condition $\ddot{\theta} = 25$ is a maximum at $t = 0$.

3.7.2 Time response of a system forced with a Unit Impulse

The same system shown if figure 3.7 can be analysed in terms of its unit impulse. This is done by making all initial conditions zero and the input a unit impulse (1 is a Laplace unit impulse) instead of zero (or fixed). Equation 3.30 becomes

$$J\left(s^2\overline{\theta}\right) + k\overline{\theta} = 1 \tag{3.33}$$

which, after substituting in the parameters and rearranging gives

$$\overline{\theta} = \frac{10^3}{s^2 + 222^2} \tag{3.34}$$

The inverse Laplace transform of this is

$$\theta = 4.5\sin(\omega.t) \qquad \text{where} \qquad \omega = 222 \tag{3.35}$$

which is an undamped oscillation (with frequency 35.33 Hz), and is a sine function as the initial conditions θ, $\dot{\theta}$, and $\ddot{\theta}$ are all zero at $t = 0$.

3.7.3 Initial and Final value theorems

If *f(t)* and *F(s)* are a Laplace transform pair, then the initial value of a time function is given by

$$f(0) = \lim_{t\to 0}[f(t)] = \lim_{s\to\infty}[sF(s)] \tag{3.36}$$

and the final value of a time function is given by

$$f(\infty) = \lim_{t\to\infty}[f(t)] = \lim_{s\to 0}[sF(s)] \tag{3.37}$$

As an example, consider equation 3.31. Substituting this into equation 3.36 gives

$$f(0) = \lim_{s\to\infty}\left[s \times \frac{25s}{s^2 + 222^2} \right]$$

Dividing numerator and denominator by s^2 gives

$$f(0) = \lim_{s\to\infty}\left[\frac{25}{1 + \frac{222^2}{s}} \right] = 25$$

which agrees with equation 3.32 when t=0.

3.7.4 Transfer Function Models

For a system having one input *u(t)* and one output *y(t)* the Laplace transform solution often takes the form

$$F(s) = \frac{Y(s)}{U(s)} \tag{3.38}$$

F(s) is called the Laplace transfer function (usually referred to as LTF) where *Y(s)* and *U(s)* are usually polynomials such that

$$F(s) = \frac{b_m s^m + b_{m-1} s^{m-1} + \cdots + b_0}{s^n + a_{n-1} s^{n-1} + \cdots + a_0} \tag{3.39}$$

If $m=n$ the system is said to be *proper*.
If $m<n$ the system is said to be *strictly proper*.

For real physical systems the Laplace solution is almost always a strictly proper rational polynomial, and the inverse Laplace transform is found using partial fractions to decompose *F(s)* into simpler terms.

3.7.5 State-space model of a transfer function

In the same way ordinary differential equations were put into their state-space canonical form in section 3.4.2, a rational polynomial transfer function in the form of equation 3.39 may also be put into canonical state-space form. The transfer function must take this form, and the coefficient of the highest order denominator must be unity. The transform may then be conducted by inspection. For example, equation 3.39 can be put into the form of equation 3.18, the controllable canonical form.

To transfer a state space equation into a transfer function, the equations of the form 3.10 and 3.11 are first Laplace transformed into functions of *s*.

$$\mathcal{L}\{\dot{x}(t) = Ax(t) + Bu(t)\} \quad \text{becomes} \quad sx(s) = Ax(s) + Bu(s) \tag{3.40}$$

$$\mathcal{L}\{y(t) = Cx(t) + Du(t)\} \quad \text{becomes} \quad y(s) = Cx(s) + Du(s) \tag{3.41}$$

Zero initial conditions are assumed, and equation 3.40 is rearranged for *x(s)* and substituted into equation 3.41 to yield that of equation 3.42.

$$y(s) = \left\{C[sI - A]^{-1} B + D\right\} \cdot u(s) \tag{3.42}$$

where the transfer function is given by

$$\frac{y(s)}{u(s)} = C[sI - A]^{-1} B + D \tag{3.43}$$

Matlab can be used to perform this conversion by using the function **ss2tf**, and also from a state-space form to LTF form using the function **tf2ss**.

3.7.6 Poles and Zeros

The *poles* of a Laplace function are the values of s that make the function evaluate to infinity and are the roots of the denominator polynomial of the function.

The *zeros* of a Laplace function are the values of s that make the function evaluate to zero and are the roots of the numerator polynomial of the function and, if the order of the numerator is lower than that of the denominator, when $s = \infty$ because the denominator becomes infinite and the function tends to zero.

To use an example, the following equation

$$F(s) = \frac{s - 5}{(s + 2)(s - 3)} \tag{3.44}$$

has two poles, one at 3 and the other at –2, one zero at 5 and two zeros at infinity, one for each denominator s. In general the number of such zeros is equal to the number of poles minus the number of numerator zeros, in the case of equation 3.44 this is one. The poles and zeros can be complex values of s and will have real and imaginary parts. A complex pole of zero will have a complex conjugate, so in general any root will be of the form $s = \sigma + j\omega$ and may be plotted on an Argand diagram. Such a plot is known as

an *s-plane plot* or a *Pole-Zero* plot. Poles are usually identified by crosses, and zeros by a circle, so for equation 3.44 which has two poles and one zero, the Pole-Zero plot will be as shown in figure 3.8.

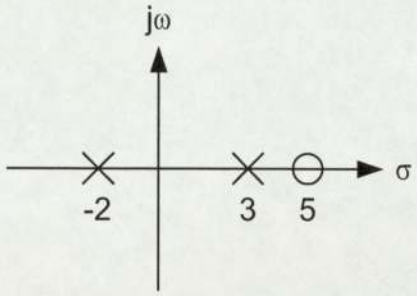


Figure 3.8 Pole-Zero plot of equation 3.44

Poles and Zeros of system functions will be used in chapter 4 to determine stability.

3.8 Block Diagrams

Block diagrams provide a pictorial representation of systems and their associated control structure. The application of block reduction techniques (or *block diagram algebra*) condenses the Laplace transform system and other equations into a form suitable for either design or inverse transformation to investigate responses in the time domain.

The issue of initial conditions was discussed in section 3.5 and it was noted that in this text the behaviour of a systems response to a forcing input is of most interest, rather than the systems transient response due to non-zero initial conditions. If a linear system is stable then the influence of the initial conditions on the output becomes negligible as time progresses, so initial conditions are omitted (they are assumed to be zero) from a transfer function block as shown in figure 3.9.

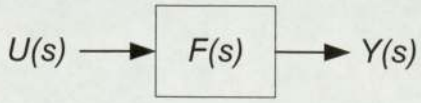


Figure 3.9 Typical block diagram of a transfer function

It is important to note that the response of a non-linear system does not necessarily become independent of initial conditions as time progresses, so they need to be included in a block diagram, usually as separate inputs or separate blocks, as is also the case for linear systems with non-zero initial conditions.

3.8.1 Block diagram of the rotational mechanical system

The rotational mechanical system of section 3.3.2 is broken down into two parts described by two differential equations 3.2 and 3.3. Their Laplace transforms are equations 3.45 and 3.46 respectively, shown below.

$$J_1 s^2 \overline{\theta}_1 + B_1 s \overline{\theta}_1 + k\overline{\theta}_1 - k\overline{\theta}_2 = \overline{T} \tag{3.45}$$

$$J_2 s^2 \overline{\theta}_2 + B_2 s \overline{\theta}_2 - k\overline{\theta}_1 + k\overline{\theta}_2 = 0 \tag{3.46}$$

These can be rearranged to

$$\overline{\theta}_1 = (\overline{T} + k\overline{\theta}_2) \cdot \frac{1}{J_1 s^2 + B_1 s + k} \tag{3.47}$$

and

$$\overline{\theta}_2 = \frac{k}{J_2 s^2 + B_2 s + k} \cdot \overline{\theta}_1 \tag{3.48}$$

which describes a system with an input $T(s)$, an output $\theta_2(s)$, and a feedback signal $\theta_2(s)$ multiplied by a constant k. The block diagram representation of this system is shown in figure 3.10.

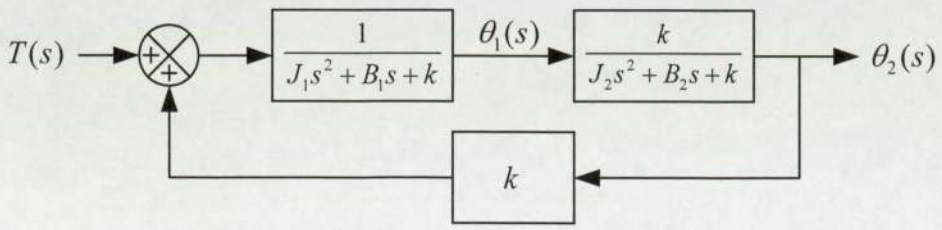


Figure 3.10 Block diagram of the rotational mechanical system

3.8.2 Block diagram reduction

Equation 3.47 could have been written as equation 3.49 which would have made the block diagram slightly more complicated shown in figure 3.11.

$$\bar{\theta}_1 = \frac{1}{J_1 s^2 + B_1 s + k} \cdot \bar{T} + \frac{k}{J_1 s^2 + B_1 s + k} \cdot \bar{\theta}_2 \qquad (3.49)$$

The best configuration really depends on the system in question, its application, and relation to the physical system it corresponds to.

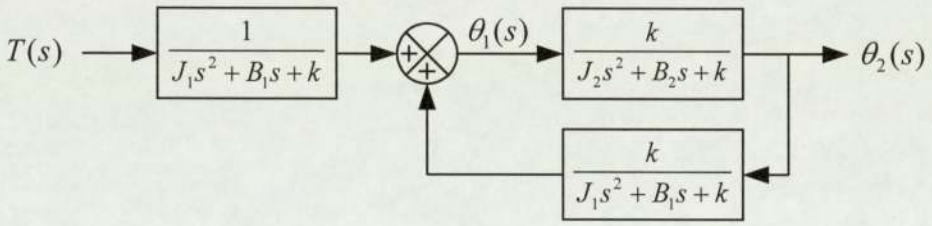


Figure 3.11 More complicated block diagram of the same system

Block reduction techniques are based on three rules. The first deals with blocks in series, the second deals with blocks in parallel and the third deals with blocks in a feedback loop.

If two blocks with transfer functions G_1 and G_2 are connected in series, they can be reduced to a single block with a transfer function G_1G_2.

If two blocks with transfer functions G_1 and G_2 are connected in parallel and their outputs summed, they can be reduced to a single block with a transfer function (G_1+G_2).

If two blocks with transfer functions G_1 and G_2 are connected in a feedback loop (for example the right-hand two blocks in figure 3.11), they can be reduced to a single block with a transfer function $G_1/(1-G_1G_2)$.

As an example figure 3.11 can be reduced using rule (3) to that of figure 3.12.

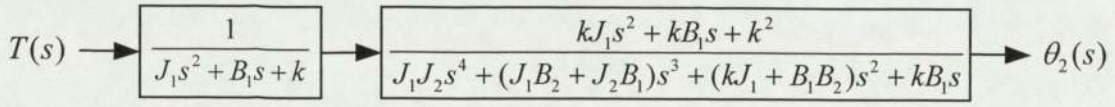


Figure 3.12 Reduced block diagram of the same system

This could be further reduce to one block using rule (1). It is worth noting however that all the information which the block diagram contained about the internal structure of the system is lost by combining these blocks.

3.9 Discrete-Time models

All the time-domain systems discussed so far have been continuous-time models (see section 3.2.7). Many systems are modelled and analysed using digital computers and the algorithms use samples of continuous data taken at discrete instants in time. A discrete-time model is usually obtained by converting differential equations of the continuous-time model into difference equations. Difference equations are discrete-time approximations of continuous-time differential equations. A derivative can be approximated by finite differences of the form

$$\frac{dy}{dt} \approx \frac{y(t+\Delta t) - y(t)}{\Delta t} \tag{3.50}$$

where Δt is a small time interval (assumed to be constant), and integrals can be approximated by the form

$$\int y.dt = \frac{1}{n\Delta t} \sum_{i=1}^{n} y_i e_i \Delta t \tag{3.51}$$

Generally speaking the smaller the time between samples the more accurate the derivative or integral will be, although this can lead to increased susceptibility to noise. It is important that the period between data samples is kept small compared with that of the data so that accurate reconstruction of the data is possible and aliasing does not occur. Frequently anti-aliasing and re-construction filters are place before and after the digital compensator respectively to prevent aliasing on unwanted high frequencies (e.g.

noise) and to remove the high frequency component after re-construction, but at a time-delay penalty to the system.

3.9.1 Difference Equation Models

A difference equation model can be expressed in the form

$$y_n = a_1 y_{t-1} + a_2 y_{t-2} + \cdots + a_n y_{t-n} + b_0 u_t + b_1 u_{t-1} + b_2 u_{t-2} + \cdots + b_m u_{t-m} \tag{3.52}$$

where *an* and *bn* values are constants, y is the output and u is the input. This is called an ARMA model because the output y_n consists of two parts, an Auto-Regressive part which is a weighted dependence on previous outputs (the $a_i y_{t-n}$ terms) and a Moving-Average part which is a weighted average of the present and past inputs (the $b_i u_{t-n}$ terms).

3.9.2 State-Space Models

The general form of the discrete-time state-space model is

$$x_{t+1} = \Phi x_t + \Delta u_t \tag{3.53}$$

$$y_{t+1} = C x_{t+1} + D u_{t+1} \tag{3.54}$$

The output equation (3.54) is identical to its continuous-time counterpart, but the matrices Φ and Δ are functions of the continuous-time **A** and **B** matrices and the sampling time (τ), and can be found by applying the equations [2]:

$$\Phi(\tau) = e^{A\tau} = I + A.\tau + \frac{A^2\tau^2}{2!} + \frac{A^3\tau^3}{3!} + \cdots \tag{3.55}$$

$$\Delta(\tau) = A^{-1}\left[\Phi(\tau) - I\right]B = \left[I.\tau + \frac{A\tau^2}{2!} + \frac{A^2\tau^3}{3!} + \cdots \right] \cdot B \tag{3.56}$$

or alternatively by using the Matlab function **c2d** which takes the A and B matrices and the sample-time as parameters and returns the Φ and Δ matrices (denoted as A_d and B_d).

3.9.3 The z-transform

The Laplace transform changes continuous-time quantities into complex-time quantities allowing functions to be written in terms of the variable *s*. In much the same way the z-transform maps Laplace transform functions into discrete-time functions allowing it to be written in terms of a variable *z*.

In its simplest form, multiplication by z^{-1} is used as a notation to represent a delay of one time step in a discrete-time signal. Equation 3.52 can thus be rewritten

$$y(z) = a_1 z^{-1} y(z) + a_2 z^{-2} y(z) + \cdots + a_n z^{-n} y(z) + b_0 u(z) + b_1 z^{-1} u(z) + b_2 z^{-2} u(z) + \cdots + b_m z^{-m} u(z) \qquad (3.57)$$

which can be rearranged into the z-domain transfer function

$$\frac{Y(z)}{U(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_m z^{-m}}{a_1 z^{-1} + a_2 z^{-2} + \cdots + a_n z^{-n}} \qquad (3.58)$$

In the complex-frequency domain, multiplying by *s* implies differentiation in the time-domain, and in the z-domain, multiplying by z^{-1} implies a time delay of one sampling interval. Equation 3.50 shows that for a sample *n* where the sampling interval is T_s, the gradient of a variable *y* can be given by

$$\left.\frac{dy}{dt}\right|_n \approx \frac{y_n - y_{n-1}}{T_s} \qquad (3.59)$$

In the z-domain y_n becomes *Y(z)* so equation 3.59 becomes

$$\frac{Y(z)(1 - z^{-1})}{T_s} \qquad (3.60)$$

It is apparent that multiplying *Y(z)* by $(1-z^{-1})/T_s$ is equivalent to differentiation, and *s* can be substituted in a Laplace function by

$$s = \frac{(1-z^{-1})}{T_s} \tag{3.61}$$

This is a first approximation and there are more accurate formulas which will not be discussed here. One such formula is the *Tustin transformation*, given by

$$s = \frac{2(1-z^{-1})}{T_s(1+z^{-1})} \tag{3.62}$$

which offers a closer approximation in most cases, but is often not worth the extra computation for the improved accuracy obtained.

3.9.4 The Shift (q) Operator

An operator analogous to the differential operator in linear differential equations with constant coefficients, is the shift ('q') operator in linear difference equations with constant coefficients. In operator calculus, all signals are considered *doubly-infinite time sequences* such as

f(k) where $k = \ldots, -1, 0, 1, \ldots$

The *forward-shift operator* is denoted by *q* and has the property

$$qf(k) = f(k+1) \tag{3.63}$$

The inverse of the forward-shift operator is the *backward-shift operator* (or *delay-operator*) and is denoted by q^{-1}, such that

$$q^{-1}u(t) = u(t-1) \tag{3.64}$$

Operator calculus gives compact descriptions of systems and makes the manipulation of difference equations much simpler, for example

$$y(k+na)+a_1y(k+na-1)+\cdots+a_{na}y(k)=b_0u(k+nb)+\cdots+b_{nb}u(k)$$

$$\text{where } na \geq nb \qquad (3.65)$$

Using the shift operator gives

$$q^{na}y(k)+a_1q^{na-1}y(k)+\cdots+a_{na}y(k)=b_0q^{nb}u(k)+\cdots+b_{nb}u(k)$$

which after factorising becomes

$$(q^{na}+a_1q^{na-1}+\cdots+a_{na})y(k)=(b_0q^{nb}+\cdots+b_{nb})u(k) \qquad (3.66)$$

The z-transform maps a *semi-infinite time sequence* into a function of a complex variable, and takes the initial values into consideration. In operator calculus, double-infinite time sequences are taken into consideration. The variable z is a complex variable and should be distinguished from the operator q.

3.9.5 Including Noise in the Model

Section 3.2.3 mentioned that a stochastic system has an element of random behaviour, and its outputs are not always a specific function of the input. Real physical systems will have noise on the measured signals and therefore have an amount of stochastic behaviour. Chapter 5 investigates methods for determining a mathematical model for an unknown system which describes the relationship between the input and output. These quantities are measured and consequently contain noise. Noise can be described as an undesirable addition to the useful signal or disturbance. There are various ways of including noise in system models and these are discussed below in the context of the individual models.

3.9.5.1 Noise Characteristics

One common method to include noise in the model is to lump the noise into an additive term $v(t)$ as shown in figure 3.13, represented by

$$y(t) = \sum_{k=1}^{\infty} g(k)u(t-k) + v(t) \tag{3.67}$$

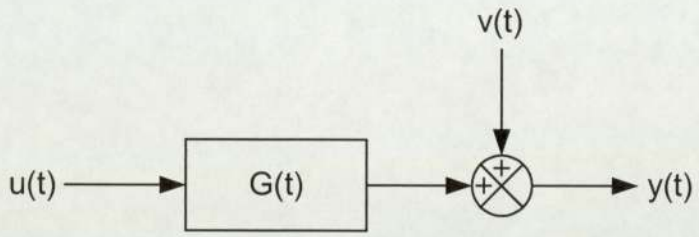


Figure 3.13 Disturbance added to the output of the system

The most characteristic feature of noise is that its value is not predictable. Information about past disturbances can be important about making qualified guesses about future values, and this is usually made within a certain degree of probability.

v(t) can be given as

$$v(t) = \sum_{k=0}^{\infty} h(k)e(t-k) \tag{3.68}$$

where $e(t)$ is discrete-time *white noise*; a sequence of independent, equally distributed random variables with a certain *probability density function* (PDF). It is usually assumed that $h(0)$ is 1 so that these transfer coefficients are normalised.

3.10 Discrete-Time Models of Linear Time-Invariant Systems with Noise

This section of the chapter discusses classes of models for linear time-invariant systems. They all include a noise aspect, and are particularly suited to the characterisation function of the test-rig where models are created from measured variables containing noise. Chapter 5 discusses the choice of models and parameter estimation methods in detail.

3.10.1 Transfer-Function Models

A discrete-time system $G(t)$ with input $u(t)$, output $y(t)$ and disturbance $e(t)$ can be represented by the equation

$$y(t) = G(q)u(t) + H(q)e(t) \tag{3.69}$$

where q is the shift operator and G(q) and H(q) are

$$G(q) = \sum_{k=1}^{\infty} g(k)q^{-k}, \qquad H(q) = 1 + \sum_{k=1}^{\infty} h(k)q^{-k} \tag{3.70}$$

The model corresponds to three functions, G, H and the PDF of e, $f_e(.)$. G and H are usually expressed in terms of a finite number of numerical values, such as rational transfer functions or state-space representations. It is common to assume that $e(t)$ is Guassian.

Quite often it is not possible to determine the coefficients from knowledge of the physical system, and the *determination of parameters* is made using estimation procedures (see chapter 5). Such parameters are denoted by the parameter vector θ, so the model is now described by

$$y(t) = G(q,\theta)u(t) + H(q,\theta)e(t) \tag{3.71a}$$

and

$$f_e(x,\theta) = \text{the PDF of } e(t); \text{ white noise} \tag{3.71b}$$

Equation 3.71 is no longer a model but a *set* of models.

G and H are often represented as rational functions where the numerator and denominator coefficients are the parameters. Such models are known as *black-box* models.

The following three sections will outline different ways of describing equation 3.71 in terms of θ, i.e. different ways of parameterising the model set.

3.10.1.1 Equation-Error (EE) Model Structure

A simple discrete-time input-output model was described in section 3.9.1. Adding a noise term $e(t)$ yields the linear difference equation

$$y(t)+a_1y(t-1)+\cdots+a_{n_a}y(t-n_a)=b_1u(t-1)+\cdots+b_{n_b}u(t-n_b)+e(t) \tag{3.72}$$

Since the white-noise term enters as a direct error in the difference equation, this model is called an *equation error* model. The adjustable parameters are

$$\theta=\begin{bmatrix}a_1 & a_2\ldots a_{n_a} & b_1\ldots b_{n_b}\end{bmatrix}^T \tag{3.73}$$

If two polynomials are introduced:

$$A(q)=1+a_1q^{-1}+\cdots+a_{n_a}q^{-n_a} \tag{3.74}$$

$$B(q)=b_1q^{-1}+\cdots+b_{n_b}q^{-n_b} \tag{3.75}$$

equation 3.72 corresponds to equation 3.71 with

$$G(q,\theta)=\frac{B(q)}{A(q)} \qquad \text{and} \qquad H(q,\theta)=\frac{1}{A(q)} \tag{3.76}$$

Equation 3.72 is referred to as an ARX model because the AR refers to the autoregressive part $A(q)y(t)$ and the X to the extra input, the exogenous (externally generated) input $B(q)u(t)$. When $n_a=0$, equation 3.72 is a special case and is called a *finite impulse response* (FIR) model.

The signal flow is shown in figure 3.14.

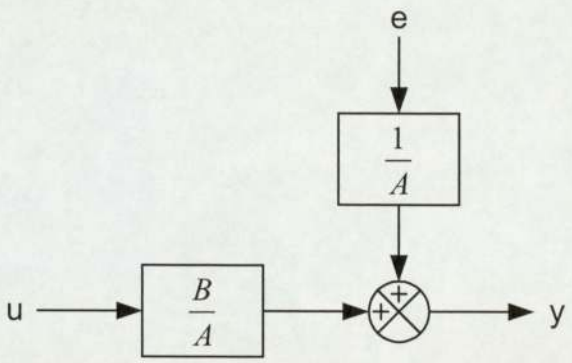


Figure 3.14 ARX model structure

3.10.1.2 ARMAX Model Structure

The description of the properties of the noise term in equation 3.72 can be made more flexible by making the equation error as a moving average of white noise, giving the model:

$$y(t)+a_1y(t-1)+\cdots+a_{n_a}y(t-n_a)=b_1u(t-1)+\cdots+b_{n_b}u(t-n_b)+ e(t)+c_1e(t-1)+\cdots+c_{n_c}e(t-n_c) \quad (3.77)$$

The noise can be represented by the polynomial

$$C(q)=1+c_1q^{-1}+\cdots+c_{n_c}q^{-n_c} \quad (3.78)$$

Equation 3.77 can be rewritten

$$A(q)y(t)=B(q)u(t)+C(q)e(t) \quad (3.79)$$

which corresponds to equation 3.71 with

$$G(q,\theta)=\frac{B(q)}{A(q)} \quad \text{and} \quad H(q,\theta)=\frac{C(q)}{A(q)} \quad (3.80)$$

and

$$\theta = \left[a_1 \ldots a_{n_a} \quad b_1 \ldots b_{n_b} \quad c_1 \ldots c_{n_c}\right]^T \tag{3.81}$$

The error term now incorporates a moving average (MA) part, so the model is called ARMAX, and is a widely used model.

3.10.1.3 Other EE Model Structures

Instead of modelling the equation error as a moving average as in the ARMAX case, it can be modelled as an autoregression, giving the ARARX model set

$$A(q)y(t) = B(q)u(t) + \frac{1}{D(q)}e(t) \tag{3.82}$$

and

$$D(q) = 1 + d_1 q^{-1} + \cdots + d_{n_d} q^{-n_d} \tag{3.83}$$

This model corresponds to equation 3.71 with

$$G(q,\theta) = \frac{B(q)}{A(q)} \qquad \text{and} \qquad H(q,\theta) = \frac{1}{A(q)D(q)} \tag{3.84}$$

and

$$\theta = \left[a_1 \ldots a_{n_a} \quad b_1 \ldots b_{n_b} \quad c_1 \ldots c_{n_c}\right]^T \tag{3.85}$$

It is also possible to use an ARMA description of the equation error giving the ARARMAX structure

$$A(q)y(t) = B(q)u(t) + \frac{C(q)}{D(q)}e(t) \tag{3.86}$$

which corresponds to equation 3.71 with

$$G(q,\theta) = \frac{B(q)}{A(q)} \qquad \text{and} \qquad H(q,\theta) = \frac{C(q)}{A(q)D(q)} \tag{3.87}$$

and

$$\theta = \left[a_1 \ldots a_{n_a} \quad b_1 \ldots b_{n_b} \quad c_1 \ldots c_{n_c} \quad d_1 \ldots d_{n_d}\right]^T \tag{3.88}$$

3.10.1.4 Output Error (OE) Model Structures

The equation error model structures all have the polynomial A in the denominator of the transfer functions G and H. The *output error* (OE) models parameterise the transfer functions independently by having an intermediate variable $w(t)$ such that

$$y(t) = w(t) + e(t) \tag{3.89}$$

where

$$F(q)w(t) = B(q)u(t) \tag{3.90}$$

The model can be rewritten

$$y(t) = \frac{B(q)}{F(q)}u(t) + e(t) \tag{3.91}$$

where the transfer functions G and H of equation 3.71 are

$$G(q,\theta) = \frac{B(q)}{F(q)} \qquad \text{and} \qquad H(q,\theta) = 1 \tag{3.92}$$

with parameters

$$\theta = \left[a_1 \ldots a_{n_a} \quad f_1 \ldots f_{n_f} \right]^T \tag{3.93}$$

3.10.1.5 Box-Jenkins (BJ) Model Structures

The OE model above can be extended to further model the error term. The Box-Jenkins model describes the error term as an ARMA model which gives

$$y(t) = \frac{B(q)}{F(q)} u(t) + \frac{C(q)}{D(q)} e(t) \tag{3.94}$$

The transfer functions G and H are independently parameterised as rational functions, and was treated by Box and Jenkins [14]. Figure 3.15 shows the signal flow for this type of model.

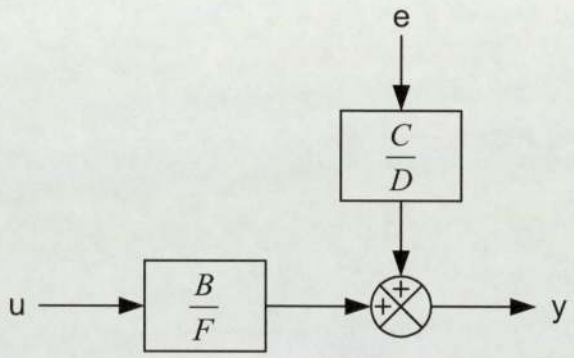


Figure 3.15 Box-Jenkins model structure

3.10.1.6 Summary of EE and OE Model Structures

The EE and OE models discussed in this section can all be represented by the generalised model structure of equation 3.95, depending on which of the five polynomials (A, B, C, D, or F) are used.

$$A(q)y(t) = \frac{B(q)}{F(q)} u(t) + \frac{C(q)}{D(q)} e(t) \tag{3.95}$$

Figure 3.16 shows the signal flow for the EE and OE model sets

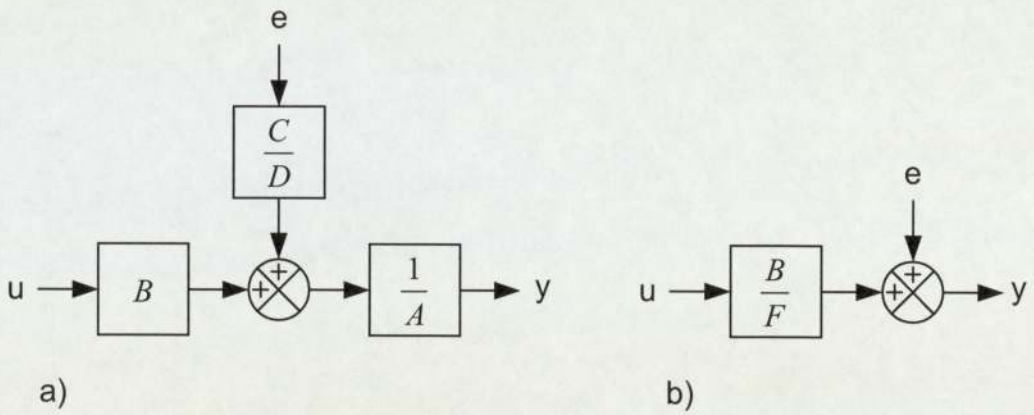


Figure 3.16 a) EE and b) OE model set structure

Table 3.1 lists the commonly used model structures with the corresponding polynomials used and equations.

Model Structure	Polynomials used	Model Equation
ARX	A B	$A(q)y(t) = B(q)u(t) + e(t)$
ARMAX	A B C	$A(q)y(t) = B(q)u(t) + C(q)e(t)$
ARMA (time series)	A C	$A(q)y(t) = B(q)u(t)$
FIR	B	$y(t) = B(q)u(t)$
OE	B F	$y(t) = \frac{B(q)}{F(q)}u(t) + e(t)$
BJ	B F C D	$y(t) = \frac{B(q)}{F(q)}u(t) + \frac{C(q)}{D(q)}e(t)$

Table 3.1 Some common black-box SISO models

3.10.2 State-Space Models

The state-space model discussed in section 3.4 can be extended to include disturbance, and the stochastic state-space model is

$$x(t+1) = A.x(t) + B.u(t) + w(t) \tag{3.96}$$

$$y(t) = C.x(t) + D.u(t) + e(t) \tag{3.97}$$

where $w(t)$ and $e(t)$ are stochastic processes with certain covariance properties. Another way to represent this is the *innovations form* [11], the discrete-time case of which is

$$x(t+1) = A.x(t) + B.u(t) + K.e(t) \tag{3.98}$$

$$y(t) = C.x(t) + D.u(t) + e(t) \tag{3.99}$$

where the matrix K is known as the steady-state Kalman gain. This form is known as the innovations form because $e(t)$ appears explicitly. Using the shift operator q, the equations can be rearranged

$$y(t) = G(q,\theta)u(t) + H(q,\theta)e(t) \tag{3.100}$$

which corresponds to equation 3.71 with

$$G(q,\theta) = C(\theta)\left[qI - A(\theta)\right]^{-1} B(\theta) \tag{3.101}$$

$$H(q,\theta) = C(\theta)\left[qI - A(\theta)\right]^{-1} K(\theta) + I \tag{3.102}$$

3.11 Discrete-Time Models of Time-Varying Systems with Noise

Equation 3.69 describes a discrete-time, linear, time-invariant model. A time-varying transfer function can be introduced by

$$G_t(q) = \sum_{k=1}^{\infty} g_t(k) q^{-k} \tag{3.103}$$

Using this, equation 3.69 becomes a time-variant model,

$$y(t) = G_t(q)u(t) + H(q)e(t) \tag{3.69}$$

It is usual and more convenient however to use the state-space form. This is simply obtained by letting the matrices be time-varying

$$x(t+1) = A(t).x(t) + B(t).u(t) + K(t).e(t) \tag{3.104}$$

$$y(t) = C(t).x(t) + D(t).u(t) + e(t) \tag{3.105}$$

Time-invariant systems are often used where there are non-equal sampling instants (multi-rate systems), in systems with time-varying parameters, and with linearisation.

3.12 Table Look-Up

For certain systems it is desirable to describe their properties using numerical tables or plots, often called graphical models. There are three reasons applicable here for the use of look-up tables:

1) the equations are too computationally intensive,
2) the equations are non-linear or contain non-linearity, and
3) the necessary models are not available, or a look-up table may just be simpler.

3.12.1 Simplification of Computationally Intensive Equations

As the number of parametric variables increases in an equation, so does the computational overhead. Equations of motion can often be simplified if all or a subset of the independent variables are treated as parameters. The simplified forms of these parameterised equations must be made for large sets of parametric values. The choice of variables often determines the balance between computational costs and storage costs. In many cases, making one or more of the variables is an equation a parameter will greatly simplify the functional relationship. Consider the following power series of $\sin(x)$ for example

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots \tag{3.106}$$

depending on the required accuracy, the computation required could be very high, and it may be beneficial therefore to use a one-dimensional look-up table with x as the index. Generally, a function with n parameters can be replaced by a lookup table of n

dimensions, so it is likely to be more efficient to use a number of smaller parameter look-up tables, where necessary.

3.12.2 Simplification of Equations Containing Non-Linearity

Chapter 5 discusses system identification methods for the characterisation of machines. The identification methods employed assume a linear model, so models containing non-linearities, or non-linear models may be difficult or impossible to obtain. Assuming the model is time-invariant, it is feasible to use a look-up table to store the dynamic properties of the non-linear part of the machine instead of a mathematical model.

3.13 Non-Linear Models

A non-linear model is one for which the principle of superposition is invalid. Most dynamical systems are non-linear to some extent, and most linear systems are only (nearly) linear within a particular operating region. A common approach therefore is to restrict signal levels and ignore any slight non-linearities. If a system contains significant non-linear behaviour, a linear model will not be an accurate representation of its behaviour. The model must be modified by adding non-linear effects until it is sufficiently accurate. There are no analysis methods which work well with all non-linear systems as there are with linear systems, so each non-linear system needs to be treated separately. If the non-linearity is small then it may be permissible to ignore it, or alternatively the system may be linearised, or another approach is to treat the non-linear part of the system separately. If one of these approaches is taken, the system may not function well if an unexpected non-linear action occurs. This problem can be avoided to some extent by simulation studies, where the system is simulated and its reactions to non-linear actions analysed.

3.13.1 Non-linear System Elements

Non-linear system elements can be either *continuous* or *discontinuous*. Continuous non-linearities can be defined as elements whose input-output characteristics are continually

differentiable. A function such as $y = x^3$ is a continuous non-linear component, and examples are transducer characteristics such as thermistor or rubber springs.

Discontinuous non-linearities cannot be modelled by analytic functions. *Single-valued non-linearities* are components which for each input there is only one possible output (except at switching points). Examples are saturation, deadzone and quantisation. A *double-valued non-linearity* has one of two possible outputs for a particular input, e.g. hysterisis. A *multi-valued non-linearity* has many possible outputs for a particular input, e.g. backlash.

3.13.2 Linearisation

The linearisation of non-linear models are convenient because much of the design and analysis techniques applicable to linear systems can be used. Such models only remain valid over a suitable range of operation, and outside of this further linear models will be required. One technique of linearisation is described by [2], which essentially uses a Taylor's expansion to find the tangent at a chosen operating point, and assumes small derivations about this point.

3.13.3 Representation of Models with Non-Linearities

A non-linear model gives wide scope to describe systems, but analysis of such systems using established control system design techniques (chapter 4), or performing system identification (chapter 5) is difficult. It is often useful to use physical insight into the character of non-linearities, and to isolate them from the linear part of the system. It is quite common for system dynamics to be linear, but with static non-linearities at the input or output, such as saturation of actuators or non-linearity of measurement transducers. A model with a static non-linearity at the input is called a *Hammerstein model*, and a model with a static non-linearity at the output is called a *Weiner model*. A combination of the two is called a *Weiner-Hammerstein model*.

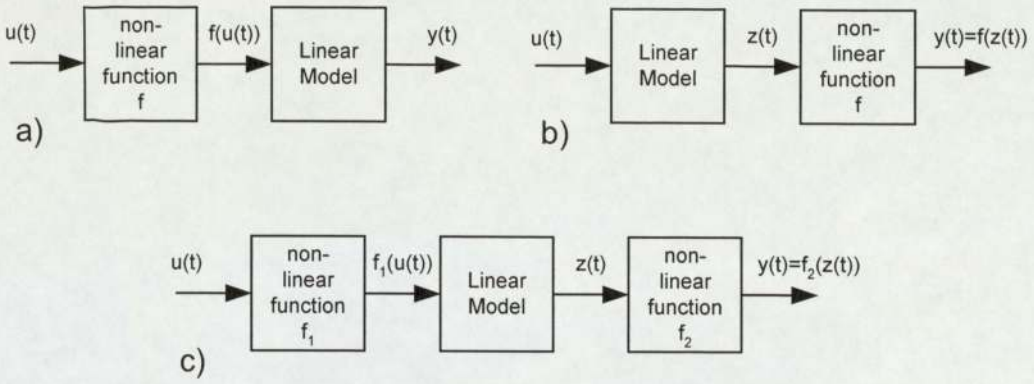


Figure 3.17 a) Hammerstein b) Weiner and c) Weiner-Hammerstein models

3.14 Dynamic Behaviour of Systems

Previous sections in this chapter have shown how dynamic system components and systems can be represented by mathematical models. This section looks at the dynamic behaviour of these system components.

A system component can be represented by a block such as that of figure 3.9 (but not necessarily in the *s*-domain). Typically the system will have a steady-state or transient response and a dynamic or forced response. The steady state output is easily measured and methods (such as the initial and final value theorems of section 3.7.3) exist to calculate this. To determine the dynamic behaviour it needs to be exited by some means, that is to apply an input signal of some kind (called a forcing function) and observe the systems output response to this input. An infinite number of forcing functions could be applied, but to simplify analysis certain specific forcing functions are chosen. Some of these are not possible to reproduce in a real-world physical system, but close approximations are possible and in any case they are still useful for theoretical analysis.

3.14.1 Forcing Functions

Unit Step

The unit-step is where the input suddenly changes from zero to one and remains there, i.e. $x(t) = 0$ for $t < 0$, and $x(t) = 1$ for $t > 0$. This has the Laplace transform $X(s) = 1/s$. This is a widely used signal in practice and is the most commonly used disturbance applied to systems.

Unit Ramp

A unit ramp has a slope of 1 and is described by $x(t) = 0$ for $t < 0$, and $x(t) = t$ for $t > 0$. The Laplace transform of this is $X(s) = 1/s^2$. This is a step change of the derivative of the input and is useful where a step change is not possible.

Unit Impulse

A unit impulse is an impulse which is infinitely narrow in time and infinite in magnitude. The area under the impulse is equal to 1, and the Laplace transform is $X(s) = 1$. This is less frequently used in practice because to be useful the amplitude has to be so high that it is either unobtainable, so high that system is damaged or driven into a region of non-linear operation, or both of these.

The above three forcing functions are called **transient disturbances** because the system normally starts from a steady-state before application of the signal and then settles to a steady state after some time.

Sinusoidal Input

When a sine wave is applied to the input of a linear system, the output will build up to a steady level which is also sinusoidal and of the same frequency but different amplitude and phase. This kind of analysis is called frequency or harmonic response analysis. The Laplace transform of a sinusoidal signal is $X(s) = \mathcal{L} \sin \omega t = \omega/(s^2 + \omega^2)$.

3.14.2 Time-Domain Response Analysis

The most frequently used forcing input for system analysis is the unit step, and a set of criteria have been developed to quantitatively specify a (bounded) system's response to this input:

Steady-state Error

The difference between the required output and the steady-state output, normally found using the final value theorem.

Rise-time

The time required for a system to achieve 90% (sometimes other ratios) of its final value.

Overshoot

The amplitude of the first peak of a response of a system. Usually expressed as a percentage of the steady-state value.

Subsidence Ratio

In a decaying oscillation this is the ratio of successive cycles of the response.

Settling Time

The time taken for the response to reach and stay within a range of its steady-state value, usually 5%.

Various time responses can be plotted using Matlab functions such as **impulse**, **step**, and their discrete-time counterparts **dimpulse** and **dstep**. By choosing appropriate parameters the impulse response of the state-space rotational mechanical system of figure 3.2 can be plotted using Matlab script (see appendix A.1):

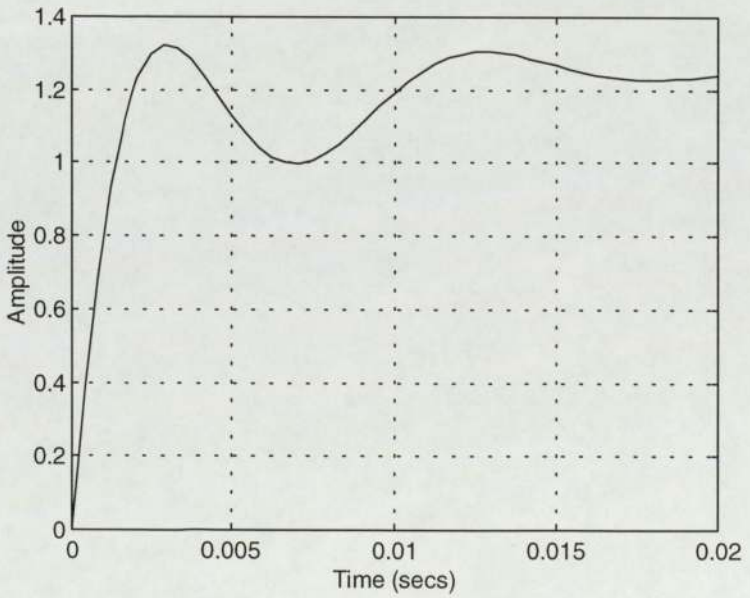


Figure 3.18 Impulse response of the rotational mechanical system.

3.14.3 Frequency-Domain Response Analysis

For a sinusoidal input to a system, the response can be determined in terms of a magnitude and phase relative to this input. The magnitude and phase of a system represented by the transfer function $G(s)$ can be found by replacing s by $j\omega$ to give $G(j\omega)$, since $s = \sigma + j\omega$ and $G(j\omega)$ is a special case of $G(s)$ [2,5]. The magnitude and phase can be found by applying the equations below [2]:

$$M(\omega) = \frac{\prod_{i=1}^{m} \sqrt{(a_i^{\,2} + (c_i\omega)^2)}}{\prod_{i=1}^{n} \sqrt{(g_i^{\,2} + (h_i\omega)^2)}} \tag{3.107}$$

$$\phi(\omega) = \sum_{i=1}^{m} \tan^{-1}\left(\frac{c_i\omega}{a_i}\right) - \sum_{i=1}^{n} \tan^{-1}\left(\frac{h_i\omega}{g_i}\right) \tag{3.108}$$

where a_i and g_i are the ω parts and c_i and h_i are the real parts of the rational polynomial transfer function $G(j\omega)$, and m and n are the numbers of terms in the numerator and denominator respectively.

The response of the system to a sinusoidal input of differing frequencies is termed its harmonic response, and this information is typically plotted onto one of three types of plot shown below. The same harmonic response information is plotted on all three graphs but each emphasises different aspects which may be used for different purposes.

Bode Plot

A Bode plot is a pair of plots in which the magnitude is expressed in dB's and the phase in degrees, both plotted against the logarithm of frequency (in rad/sec). These are convenient for estimating systems consisting of more than one component because when they are both plotted together, the combined magnitude or phase at a particularly frequency is found by simply adding the two responses. The Bode plot of the state-space rotational mechanical system of figure 3.2 also used in section 3.14.2 can be found using the Matlab command `bode(A,B,C,D);` and is shown in figure 3.19.

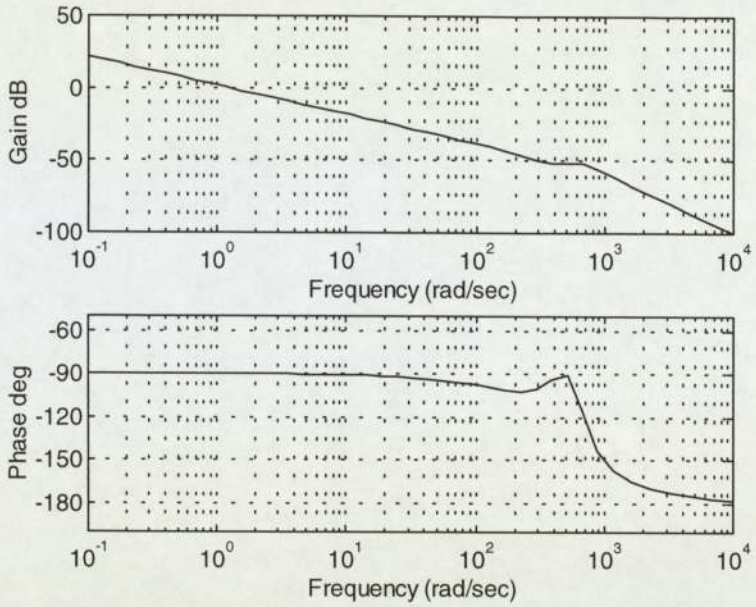


Figure 3.19 Bode plot of the rotational mechanical system.

Nyquist (or Polar) Plot

The Nyquist plot is essentially an Argand diagram where the input is assumed to lie along the positive real (horizontal) axis, and for each frequency value there is a corresponding output vector of length 'magnitude' angled from the input vector 'phase' degrees (in a clockwise direction). The plot consists of the locus of the ends of the output vectors for frequency values ranging from 0 to ∞. The Nyquist plot of same mechanical system can be found using the Matlab command `nyquist(A,B,C,D);` and is shown in figure 3.20.

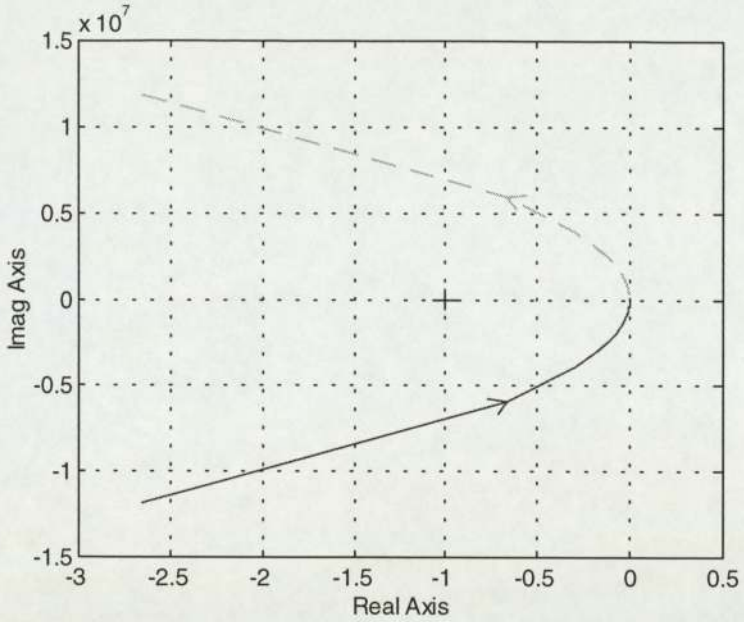


Figure 3.20 Nyquist plot of the rotational mechanical system.

3.14.4 Characteristics of a first-order system

First-order system components occur frequently such as the mechanical system shown in figure 3.21 which (assuming zero initial conditions) can be described by equation 3.109 below.

$$\bar{\theta}_2 = \frac{\bar{\theta}_1}{1+\frac{B}{k}s} \qquad \text{or} \qquad Y(s)\frac{X(s)}{1+\frac{B}{k}s} \tag{3.109}$$

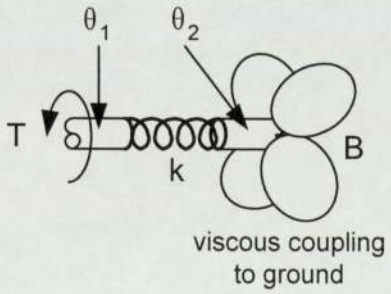


Figure 3.21 First order rotational mechanical system.

If the input $X(s)$ is a unit step ($X(s)=1/s$) or unit impulse ($X(s)=1$) and $\frac{B}{k}=1$ then using Matlab it can be shown that the time response of the system is:

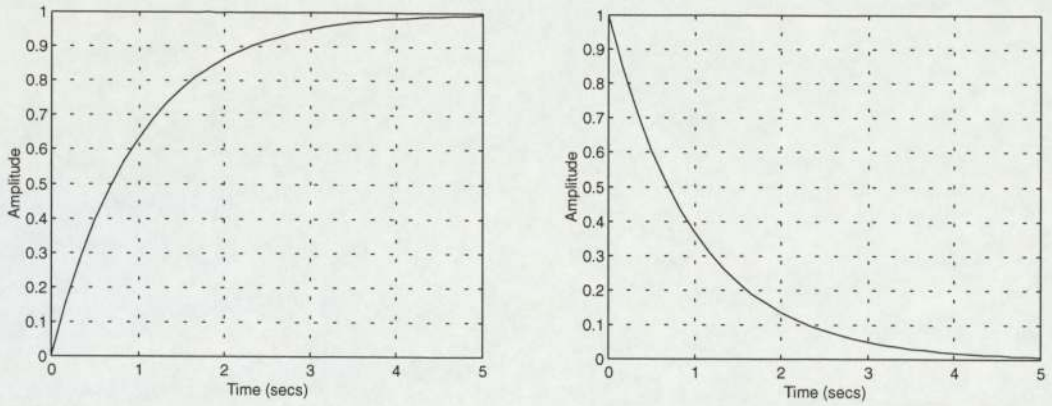


Figure 3.22 Time response of a first order system to a) unit step and b) unit impulse.

The important characteristics [5] are:

1) the response is exponential with asymptotes to the steady state (input) value,
2) the output reaches value $0.63x(t)$ when $t = \frac{B}{k}$,
3) the output has values $0.95x(t)$ and $0.98x(t)$ when $t = 3\frac{B}{k}$ and $4\frac{B}{k}$ respectively,

The system's response to a sinusoidal input of magnitude 1 ($X(s) = \frac{\omega}{(s^2+\omega^2)}$) can also be found using Matlab. Appropriate plots are the Bode plot and Nyquist plot:

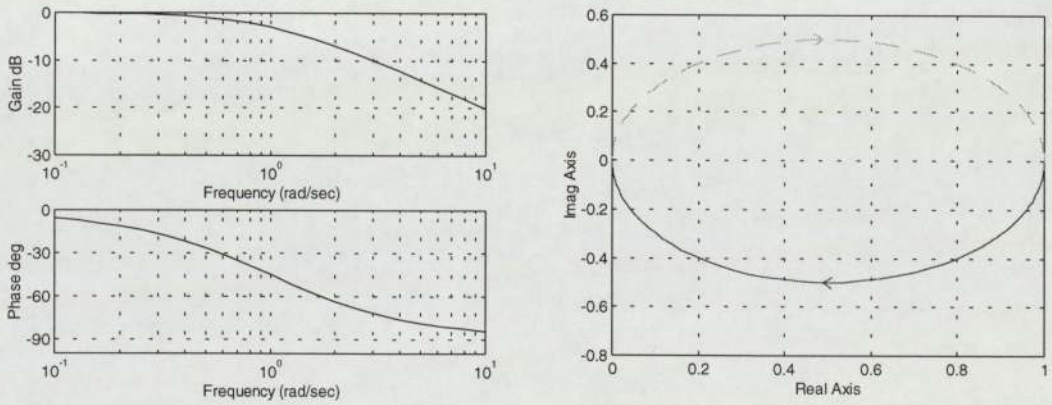


Figure 3.23 a) Bode plot and b) Nyquist plot showing the response of a first-order system to harmonic input.

The important characteristics of the Nyquist plot [5] are:

1) the Nyquist plot is a semicircle with centre (0.5, j0) and radius 0.5,
2) for a very small ω the magnitude is unity and the phase is zero,
3) as $\omega \to \infty$ the magnitude tends to zero and the phase tends to 90° lag,

4) the lag is 45° when $\omega = 1/\frac{B}{k}$,

The important characteristic of the Bode plot is that the magnitude plot has a 'corner frequency' (see second-order system below) of $\omega = 1/\frac{B}{k}$, after which the magnitude drops off at approximately 20dB per decade.

3.14.5 Characteristics of a second-order system

A general second-order system can be represented by the normalised differential equation:

$$\ddot{y} + 2\zeta\omega_n \dot{y} + \omega_n{}^2 y = K\omega_n{}^2 x \tag{3.110}$$

if the steady-state gain K is unity, the Laplace Transform of this is:

$$Y(s) = \frac{\omega_n{}^2 \cdot X(s)}{s^2 + 2\zeta\omega_n s + \omega^2} \tag{3.111}$$

ω_n is called the *undamped natural frequency* (rad s^{-1}) and ζ is called the *damping ratio*. Figure 3.24 shows the response to step and impulse inputs for a range of ζ, plotted using a Matlab script (see appendix A.2).

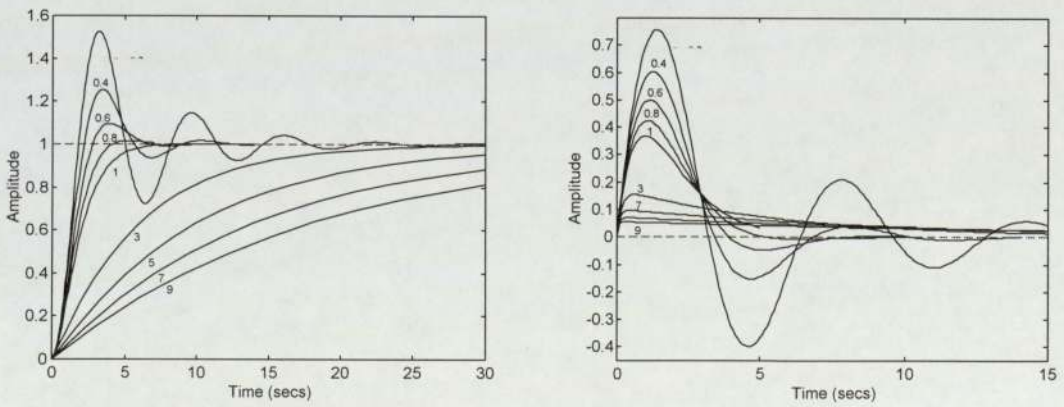


Figure 3.24 a) Step and b) Impulse response of a second order system

The smaller the value of ζ the more oscillatory the response. The frequency oscillation is called the *damped frequency* and is given by

$$\omega_d = \omega_n \sqrt{1-\zeta^2} \tag{3.112}$$

When $\zeta \geq 1$ there is no overshoot, and a value of $\zeta \approx 0.7$ is considered in control terms to give a 'good' response, i.e. small overshoot, quick settling. If $\zeta < 1$ the inverse Laplace transform of equation 3.111 with a unit-step input is [5]:

$$y(t) = 1 - \frac{e^{1\zeta\omega_n t}}{\sqrt{(1-\zeta^2)}} \sin\left(\omega_n \sqrt{(1-\zeta^2)}t + \cos^{-1}\zeta\right) \tag{3.113}$$

The steady-state gain of the system is unity so the transient part is the second term of this equation which is an oscillation which decays to zero as $t \to \infty$. If $\zeta > 1$ equation 3.111 can be factorised and inverse Laplace transformed to give a pair of exponential terms which also decay to zero as $t \to \infty$.

The frequency response of equation 3.111 is also be plotted using the same Matlab script as above:

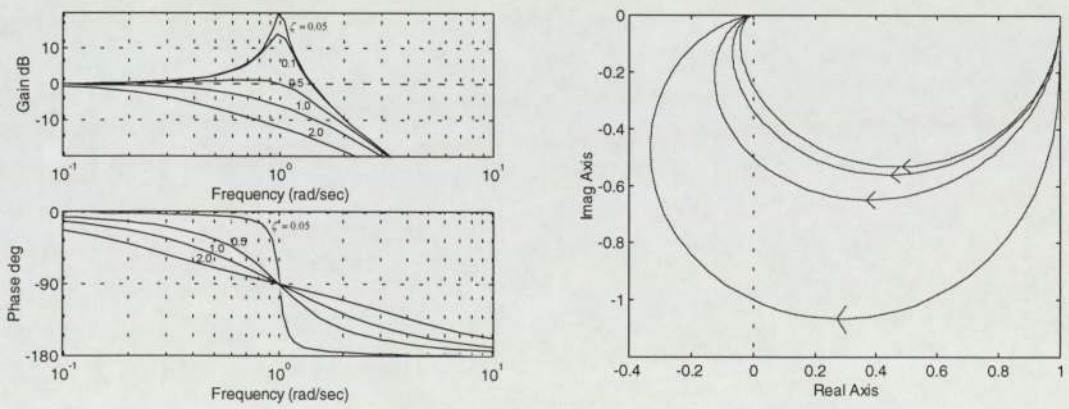


Figure 3.25 a) Bode plot and b) Nyquist plot showing the response of a second-order system to harmonic input.

The magnitude plot shows the curves tend to zero for very low frequencies and a slope of -40 dB per decade of frequency for increasing frequency after the 'corner frequency',

where the response intersects the zero dB line. There is a peak at the systems resonant frequency which tends to infinity as $\zeta \to 0$. The phase plot shows a rapid increase in phase lag for small increases in frequency around ω_n for low values of ζ, between a value close to 0° to a value close to 180°.

3.14.6 Characteristics of higher-order systems

Section 3.7 discussed the Laplace transfer model of a system, and equation 3.39 represents such a model in terms of a rational Laplace polynomials. It was also shown that the roots of the denominator are the poles of the function. If the denominator is factorised to give $(s\text{-}p_1)(s\text{-}p_2)(s\text{-}p_3)\ldots(s\text{-}p_n)$ the transfer function can be written as:

$$\frac{Y(s)}{X(s)} = \frac{N_m(s)}{(s-p_1)(s-p_2)(s-p_3)...(s-p_n)} \tag{3.114}$$

which can be separated by partial fraction expansion to give:

$$\frac{Y(s)}{X(s)} = \frac{A_1}{(s-p_1)} + \frac{A_2}{(s-p_2)} + \frac{A_3}{(s-p_3)} + \cdots + \frac{A_n}{(s-p_n)} \tag{3.115}$$

Laplace inversion gives the time response:

$$y(t) = A_1 e^{p_1 t} + A_2 e^{p_2 t} + A_3 e^{p_3 t} + \cdots + A_n e^{p_n t} \tag{3.116}$$

It can be shown [5] that applying various transient forcing functions, the response is made up of a steady-state component, and a transient component $\sum_{i=1}^{n} A_i e^{p_i t}$. The contribution of each term depends on the value of the pole p_i (which determines its position in the s plane) and the magnitude A_i. If p_i is real and negative $A_i e^{p_i t}$ will decay to zero at a rate determined by p_i, and if p_i is real and positive $A_i e^{p_i t}$ will increase exponentially. If p_i is complex, there will be another root forming a conjugate pair which together result in a cosine wave (at a frequency that depends on their distance from the origin), which increases exponentially, decreases exponentially, or remains the

same amplitude depending on whether the real part of the poles is positive, negative or zero respectively.

A stable linear system will have poles in the left-half of the s-plane. Poles with large negative real parts produce a rapid exponential decay, and consequently these poles have less effect on a system than poles with smaller negative real parts. It is therefore the poles with most positive real parts that have most effect on a system, and these are called *dominant poles*. *Dominant pole analysis* requires that the non-dominant poles are well to the left of the dominant poles, and that any pole close to a dominant pole is close to a zero (which cancels its effect). Pole positions is important to system stability, and will be discussed further in the chapter 4.

CHAPTER 4

4 Test Rig Control

This chapter is concerned with the control of the machine emulation and characterisation test rig with various machines connected to it. When the test-rig is characterising a machine, the test-rig will be driving itself and the machine being characterised. Similarly when the test-rig is emulating a machine, the test-rig will be driving itself and the motion source connected to it (typically a motor / drive pair). The dynamic behaviour of these configurations may vary considerably and consequently impact the control strategy of each. The model of the 'plant' under control is very important in the design of the control system, and is why system models were discussed prior to this chapter.

This chapter will first identify the control problem, and then examine various control strategies that may be employed to control the test-rig. Much of the mathematical procedures involved will be performed using Matlab and its associated 'Control' toolbox.

4.1 The General Structure of the Test-rig

The general construction of the test-rig was developed early in the project, and a diagram of this is shown in figure 4.1. It consists essentially of a motion source connected to a torque transducer, the other end of which is connected to an external machine. The variables directly measured are the output shaft torque and angle, and the motors current. The characterising / emulating 'controlled-variable' (discussed below) calculation and the outer control loop (discussed below) are performed using a DSP (digital signal processor) board. This consists of four pairs of discrete-time sampled analogue input and output signals and a fast processor designed to perform mathematical operations quickly. Chapters 6 and 7 explore the physical design of the test-rig in detail.

The drives are PWM (pulse-width modulation) current-controlled generators that produce a current proportional to the input torque-reference voltage (chapter 7 describes PWM drive operation). The current passing through an inductance produces a proportional magnetic flux, which in conjunction with the stator field in a motor produces rotary motion. "Air-gap torque" is the torque between the stator and the rotor

of a motor, and is proportional to the motor current. It is different to the output torque because it is also driving the rotor's inertia. Permanent-magnet DC motors have what is known as a 'mechanical time constant'. If a stationary DC motor has a 1V step input applied to its terminals, then ignoring inductance (as the electrical time constant is negligible by comparison) and friction, a current of $1/R$ amps will flow. This will give rise to a torque of Kt/R Nm, and an angular acceleration of $K_t/(R.J)$ rad/s^2. As the motor speeds up, it will generate a back emf and the current flowing in the windings will therefore reduce. After an infinite amount of time has passed, the motor will have reached speed ω rads/s^2 where $K_t.\omega = 1$ (i.e. back-emf = applied voltage). The motor mechanical time-constant is (for any given voltage) terminal_speed / initial_acceleration = $(R.J)/(K_t^2)$, and is a classical first-order system, where the time-constant is the time taken for the motor to reach 63.21% of its final velocity. Larger motors tend to have a disproportionately higher inertia for their torque rating than smaller ones. Therefore, to achieve the required torque without forgoing a low mechanical time constant the motion source was chosen to consist of four smaller motors coupled mechanically. This of course assumes that the coupling method employed does not impact the mechanical time constant significantly, or introduce any other unacceptable characteristics.

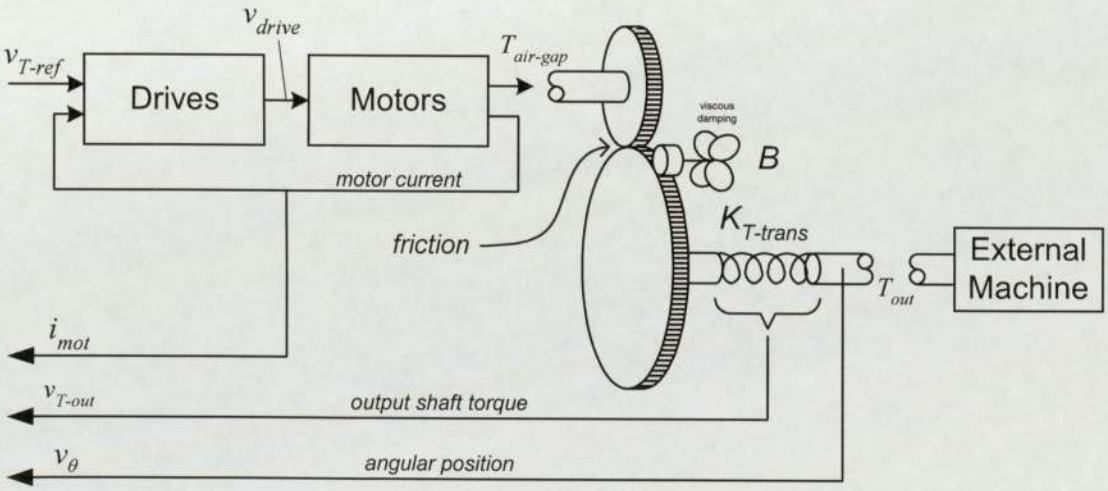


Figure 4.1 Outline Block Diagram of the test-rig

Two gearboxes were designed, both minimising backlash (an undesirable non-linear property of gear trains), friction and inertia. Considerable effort has been put into minimising the backlash, as this non-linear component would significantly impact the performance of the test-rig - chapter 6 discusses the gearbox designs. The backlash has

been reduced to a level where it can be considered negligible, but unfortunately the gears do have significant inertia, friction and, in the case of the Mk-II gearbox, viscous damping. These quantities have different values depending on which side of the gearbox they are observed. The gearbox ratios are 2.388:1 and 2.5:1 for the Mk-I and Mk-II respectively. Figure 4.2a shows a simple one-step gear train consisting of two gears, and figure 4.2b shows the conceptual layout of the test-rig gearbox.

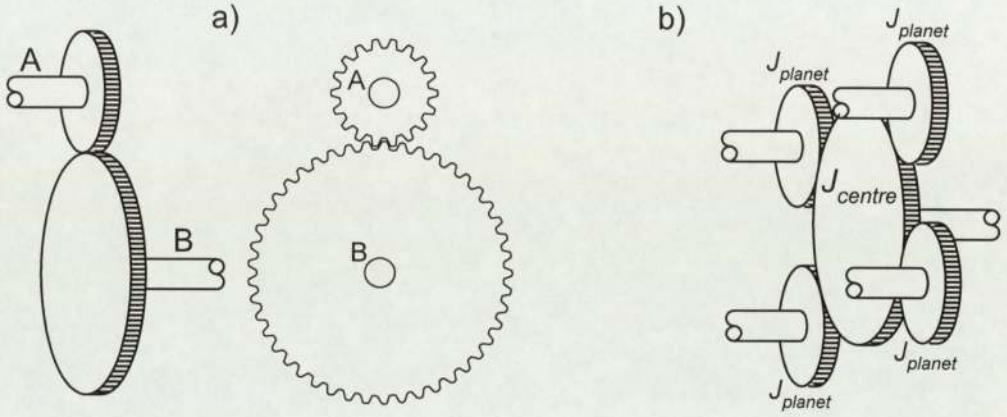


Figure 4.2 a) Simple one-step gear train, and b) Conceptual diagram of the test-rig gearbox

A torque transducer with a very high shaft stiffness (38.2 KNm/rad) was chosen so that any resonant frequencies set up as a result of this flexibility and the inertias connected to each end would be very high and small in amplitude. Unfortunately, stiffer torque transducers tend to have a higher inertia (426×10^{-6} kg.m^2), and high torque ratings (200 Nm) leading to lower resolution ($\pm 10V \equiv \pm 200\%$ of rated torque). The inertia is still acceptable compared with the rest of the test-rig but the low resolution means that the torque signal needs to be amplified, also amplifying any noise present. This can be filtered to some extent but only by incurring a slight time delay penalty. This noise problem is unfortunate, but is a physical limitation imposed by the choice of torque transducer. The shaft stiffness is extremely high, and for this application will be considered rigid. Figure 4.3 is a photograph of the dynamic mechanical parts of the test-rig, with the gearbox Mk-I connected.

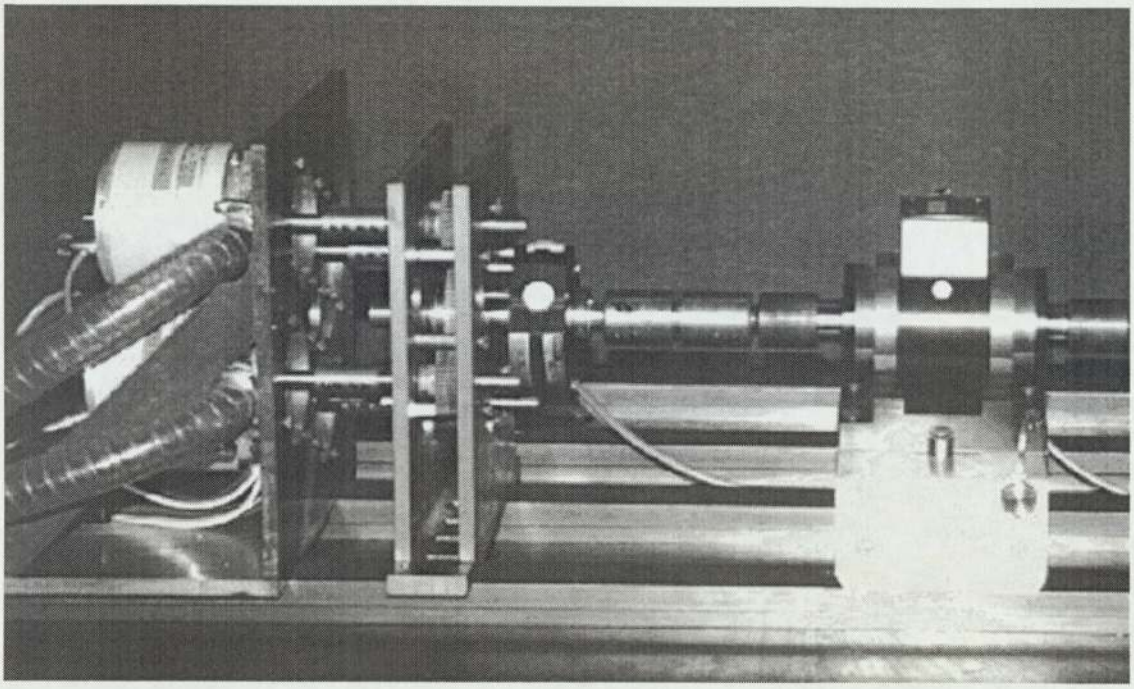


Figure 4.3 The dynamic mechanical parts of the test-rig, with the gearbox Mk-I connected

4.1.1 Referred Inertia

If gears *A* and *B* in figure 4.2a have inertia's J_A and J_B, the inertia seen at shaft *A* and shaft *B* will be given by

$$J_{shaft-A} = J_A + J_B / n^2 \tag{4.1}$$

$$J_{shaft-B} = J_B + J_A \times n^2 \tag{4.2}$$

respectively, where n is the gear ratio of gear *B* to gear *A*,

$$n = \frac{num\_teeth_B}{num\_teeth_A} \tag{4.3}$$

Equation 4.1 shows that the inertia at shaft *A* will be the inertia of gear *A* plus the *referred inertia* of gear *B* (J_B / n^2). This means that for the test-rig configuration shown in figure 4.2b the referred inertia of the gearbox centre gear and everything connected

directly to it will be $4n^2$ times less as seen by each motor. It is important therefore to keep the inertia of the gears low, particularly the planet gears.

4.1.2 Friction and Damping

Friction occurs between the meshing teeth of gears in the gearbox. Viscous damping is also present in the gearbox Mk-II, as it is oil lubricated, and occurs to a much lesser extent in the bearings of both gearboxes. Frictional and viscous damping forces at the point of the meshing gear teeth are measured at the shaft by the product of the force and the perpendicular distance from the centre of the gear. That is, the opposing torque developed by these forces is a function of the magnitude of the force and the gears radius. The gear's radius is dictated by its circumference, which in turn is dictated by the gear geometry and ratio. Since the relationship between radius and circumference is linear, the relationship between the gear ratio and relative torque due to these losses at the ends of the gear train is proportional to the gear ratio. Referring to figure 4.2a, if the torque due to losses on each shaft are T_{loss_A} and T_{loss_B} then the relationship between them is $T_{loss\_B} / T_{loss\_A} = n$, where n is the gear ratio (2.5 in this diagram and for the gearbox Mk-II also).

The torque due to friction can be assumed constant in magnitude, and opposite in sign to velocity. The torque due to viscous damping is the product of angular velocity $\dot{\theta}$ and the damping factor B. Although B is dependent on oil viscosity, it will be assumed to be constant as tests are conducted quickly and the temperature will not rise significantly.

4.1.3 Measurement of variables

The only variables measured directly are the output shaft torque (T_{shaft}), the output shaft angle (θ), and the motor current (i_{mot}). The angular velocity and angular acceleration of the output shaft are also required if the effects of the test-rig dynamics are to be estimated. Chapter 6 and 7 discuss the angle monitoring method, and electronic calculation of angular velocity.

Differentiation of signals is inherently difficult, as any noise is amplified. The electronic estimation of the velocity is reasonably accurate and noise-free because it uses a

voltage-to-frequency converter, whose input is the edges of the orthogonal pulses produced by the shaft-encoder, which have a very fine resolution (4096 pulses per revolution). A tacho-generator was initially employed to monitor the output-shaft velocity, but this was found to produce a signal with excessive ripple. Velocity observation is discussed in detail in chapter7.

A voltage proportional to the motor current is available for measurement and this can be used to calculate the acceleration of the output shaft (assuming the shaft torque is known). The shaft acceleration (rads/s^2) can be approximated by:

$$\ddot{\theta} = k_1.I + k_2.T \tag{4.4}$$

The air-gap torque is determined by the motor's *torque constant*, which is measured in Nm/A. The contribution of this torque is related to the output shaft acceleration by the torque constant and the test-rig dynamics, approximated by k_1 (rads/s^2.A). k_2 represents the contribution to output shaft acceleration by the output shaft torque, and has units rads/s^2.Nm.

Figure 4.4 shows the system variables of interest, which are presented to the DSP board containing the controller.

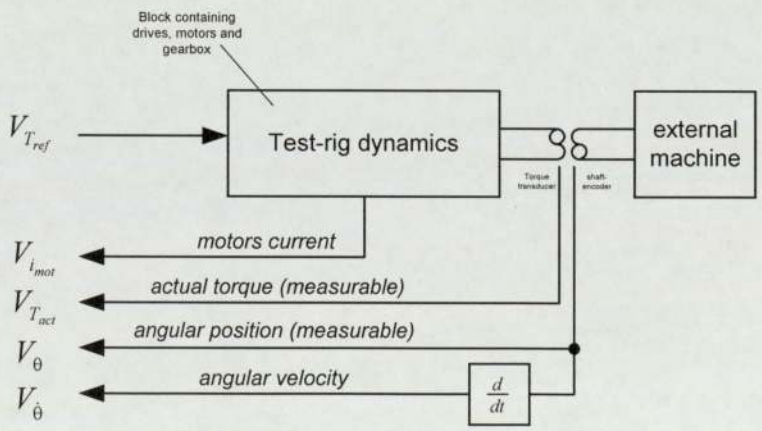


Figure 4.4 System variables of interest

4.2 The Control Problem

The two general functions of the test-rig and their control requirements are as follows:

1) Machine Characterisation. When performing a characterising function the DSP board is used to derive the torque reference from a set of tests that are used to characterise a machine. The resulting measured variables are used to create a model of the machine (chapters 5 and 10). When characterising a machine it is the nature of the stimuli and the relationship between the resulting variables that is of importance, not the precise control of the external machine. Besides, without an accurate model of the (unknown) machine being characterised it is difficult to control it accurately (except using advanced control techniques, which are not investigated here). The types of machines to be characterised are most likely to be heavily cyclic, which means that ultimately the angle also needs to be controlled.

2) Machine Emulation. When performing an emulating function, the DSP board is used to derive the angle reference from past and present measured variables (particularly torque) and a model of the machine being emulated (chapter 9). To emulate a machine an accurate representation of this model and the accurate control of the resulting output shaft angle is of importance. It is reasonable to assume that the external machine is most likely to be a drive / motor pair, being mainly inertial.

Since accurate control is more important to the machine emulation function, and machine characterisation may even be conducted with open loop torque control, the controller design will initially be for the machine emulation function.

Summary:

It is apparent that the test-rig has dynamics that complicate the control problem, and closing the control loop around the controlled variable alone is insufficient. The inherent inertia and damping of the test-rig are minimised as far as practicable, and the (non-linear) backlash has been reduced to a level where it can be ignored. The justification for the particular construction of the test-rig has been left to chapter 6, but it can be assumed the set-up is optimal within the constraints of the project.

It is the purpose of this chapter to devise a control scheme allowing accurate control of the output shaft angle and torque. The test-rig's friction, damping and inertia should be transparent to any external machine connected to it. That is, the test-rig should be capable of emulating a machine with parameters greater or less than the test-rig's own parameters.

4.2.1 Simplified Model of the Test-rig

The drives have a very fast and linear current control loop which is many times faster than the response of the motors, and is much faster than other dynamic parts of the system. The dynamic behaviour of this part of the system will therefore be treated as a constant gain *assuming it will be used within its operating region* (the system is non-linear outside this region). The motors are electrically connected in series and rigidly mechanically connected in parallel. Since they are driven by a constant current (torque control), the motor's and gearbox's lumped parameters of inertia, damping and friction can be regarded as a single block. The only other inertia and damping are in the torque transducer, and since this has an extremely high shaft stiffness (higher than other parts of the system) which can be ignored, these parameters may be lumped with the others. All the inertia, damping and friction within the test-rig can therefore be referred to one point in the system - the output shaft (gearbox referred inertia calculations are made in chapter 6). Figure 4.5 shows a block diagram of this simplified test-rig model.

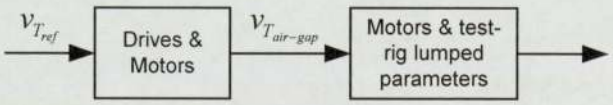


Figure 4.5 Simplified Test-rig model

4.3 Continuous-Time Control

4.3.1 Controller Structure

Figure 4.6 shows a standard arrangement for control systems. The pre-filter modifies the setpoint *R(s)*, and is often not required. The comparator generates the error signal *E(s)* from the setpoint and the plant output *Y(s)* that is fed back. The forward path

compensator *C(s)* generates the plant-actuating signal *U(s)* from the error signal. Any external disturbance *D(s)* is typically summed to the output, but can also be summed to *U(s)* or regarded as a separate input to the plant.

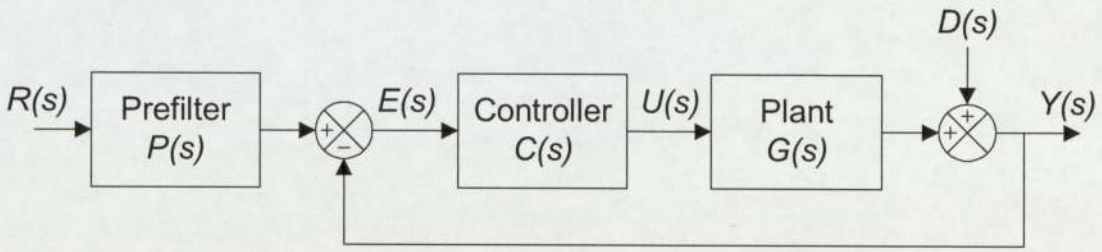


Figure 4.6 Block Diagram of a Typical Control System

The model of the plant usually dictates the controller design. The plant block typically contains the actuators and transducers whose speed of response is significantly faster than that of the plant, in which case their characteristics can be ignored. The system block diagram(s) of the test-rig are similar to that of figure 4.6, and it is the aim of this chapter to develop practical test-rig control strategies.

4.3.2 System Stability

There are a number of ways of examining system stability. The requirement is to be able to quantify the output resulting from some input stimulus applied to the system. Some standard stimuli are used for this purpose, which are easily manipulated and for which the resulting system responses can be easily quantified. A method applicable to linear systems is to apply a bounded input signal and observe the response. If the response is bounded then the system is stable. Examples of bounded signals are the *step input*, a pulse input (particularly the *unit impulse*), and a steady-state *sinusoid*. (An example of an unbounded signal is a positive exponential $A.e^{xt}$). If the system settles to its original steady-state level then it is said to be *asymptotically* stable. If it produces any other steady state signal then it is said to be *marginally* stable. Any other response to an impulse means the system is unstable. The classification of non-linear systems is more complicated and various techniques have been developed to analyse the behaviour of these.

4.3.3 Stability of Laplace transfer models

A unit impulse is frequently used to determine a system's stability. If after the impulse has been applied the system settles to its original steady state then it is said to be asymptotically stable. If the system produces a bounded response other than the original steady state, it is said to be marginally stable. Any other response is said to be unstable. A unit impulse is mathematically convenient to use as its Laplace transform is 1. Assuming the transfer function is a rational polynomial with denominator order n, the output can be expressed in terms of partial fractions

$$y(s) = \frac{A_1}{s+p_1} + \frac{A_2}{s+p_2} + \cdots + \frac{A_n}{s+p_n} \tag{4.5}$$

If the denominator is set to zero, the resulting equation is called the system's *characteristic equation*. The roots are the poles of the transfer function $F(s)$. Equation 4.5 will have n poles $-p_1, -p_2, \ldots, -p_n$ which may be real or occur in complex conjugate pairs. The time response of each pole can be found from the inverse Laplace transform

$$\mathcal{L}^{-1}\left[\frac{A_i}{s+p_i}\right] = A_i e^{-p_i t} \tag{4.6}$$

Real Poles

- If the root of $(s+p_i)$ is negative the exponent in the time response is also negative and this element decays exponentially,
- If the root is zero the time response will reduce to the constant A_i, and
- If the root is positive the exponent increases exponentially.

Complex Poles

With a pair of complex conjugate poles

$$\frac{A_i}{s+p_i} \quad \text{and} \quad \frac{A_{i+1}}{s+p_{i+1}}$$

where $A_i = g + jh$, $A_{i+1} = g - jh$, $p_i = a - jb$, and $p_i = a + jb$ the inverse Laplace transform becomes

$$\mathcal{L}^{-1}\left[\frac{g+jh}{s+a-jb}+\frac{g-jh}{s+a+jb}\right]=(g+jh)e^{-(a-jb)t}+(g-jh)e^{-(a+jb)t} \qquad (4.7)$$

which can be simplified using the laws of indices and $re^{-j\theta}=r(\cos\theta-j\sin\theta)$ to be

$$\mathcal{L}^{-1}\left[\frac{g+jh}{s+a-jb}+\frac{g-jh}{s+a+jb}\right]=2e^{-at}\left[g\cos bt-h\sin bt\right] \qquad (4.8)$$

From equation 4.8 it can be seen that the exponential term is the product of the real part of the complex conjugate poles and time, and it therefore this part which determines the system stability and not the imaginary part.

To Summarise

If the real parts of all system poles are negative then the system will be asymptotically stable,

If the real parts of any system poles are positive then the system will be unstable,

If the real parts of any system poles are zero and non-repeated then the system will be stable,

If the real parts of any system poles are zero and repeated then the system will be unstable,

Figure 4.7 summarises a system in terms of its dominant pole positions, and indicates the stable and unstable regions of the *s*-plane.

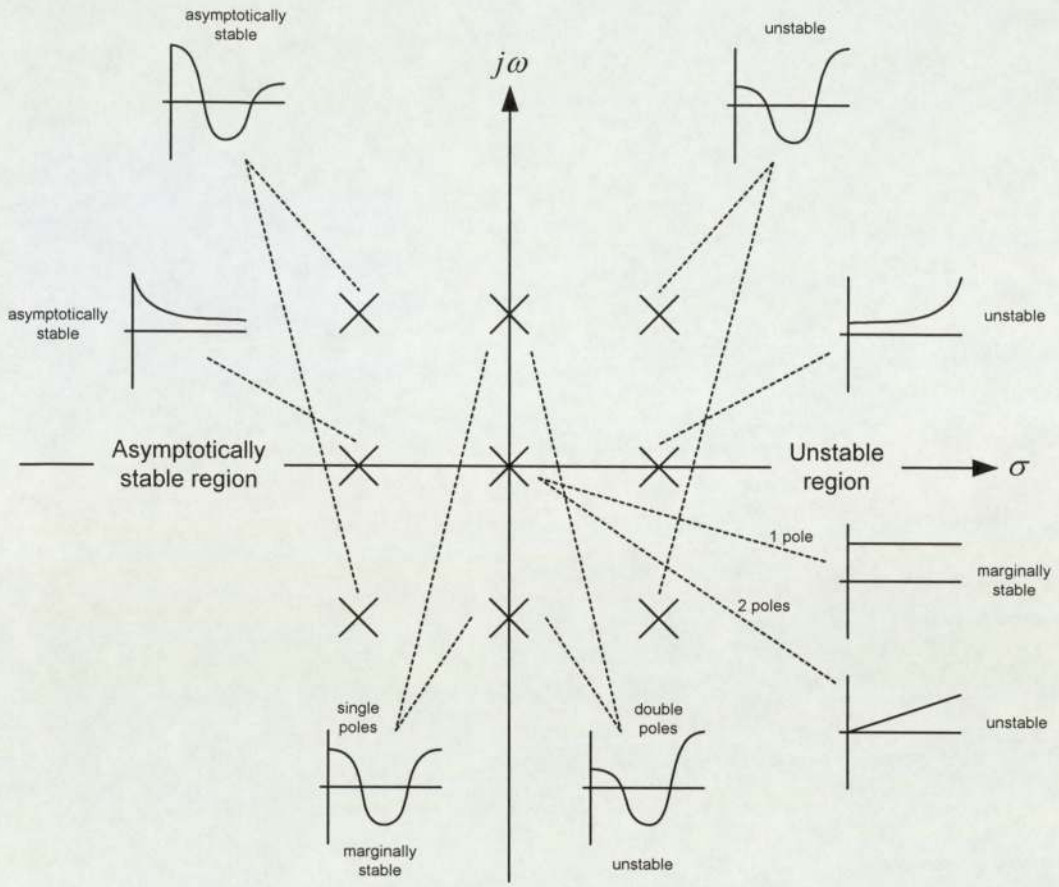


Figure 4.7 Stability regions, pole positions and impulse responses in the s-plane

A system's poles are often called its *characteristic values*.

It can be seen by inspection that the simple spring-inertia system described by equation 3.31 (section 3.7.1) has two poles at *j222* and *–j222*. Both these poles are unrepeated and have zero real parts so system is therefore marginally stable as expected, since it is an undamped oscillation.

4.3.4 Stability of State-Space models

Section 3.7.5 discussed a method for transforming a state-space model into a Laplace transfer function given by equation 3.43 shown below

$$\frac{y(s)}{u(s)} = C[sI - A]^{-1} B + D$$

This can be rearranged to give the rational equation

$$\frac{y(s)}{u(s)} = \frac{B.C + D.[sI - A]}{[sI - A]} \tag{4.9}$$

If the input is a unit impulse, the poles are the roots of the denominator, $[sI - A]$. The *eigenvalues* of the **A** matrix are the values of the scalar quantity λ that satisfy the equation

$$|\lambda I - A| = |A - \lambda I| = 0 \tag{4.10}$$

which is the system's *characteristic equation*. The eigenvalues of the **A** matrix are the systems *characteristic values* and are identical to the roots of the determinant $|sI - A|$. The characteristic values of a given system will be the same regardless of the chosen state-space representation, as the system dynamics will be identical.

4.3.5 Closed-loop poles and zeros

Figure 4.8 shows a typical feedback configuration with an additional element *H(s)* in the feedback path.

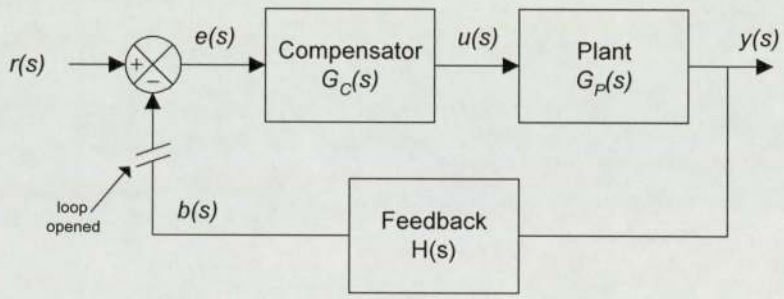


Figure 4.8 Typical feedback controller

The open-loop transfer function of this system is:

$$G_{OL} = \frac{b(s)}{e(s)} = H(s)G_P(s)G_C(s) \tag{4.11}$$

and the closed-loop transfer function is:

$$G_{CL} = \frac{y(s)}{r(s)} = \frac{G_P(s)G_C(s)}{1 + H(s)G_P(s)G_C(s)} \tag{4.12}$$

The characteristic equation of equation 4.12 is the same as the open loop equation 4.11, and the closed-loop poles are therefore a function of the open-loop poles and zeros. *H(s)* is the transducer dynamics or feedback compensation. The plant transfer function is usually fixed and the controller design problem is usually to select $G_C(s)$ (and possibly *H(s)*) to obtain the required closed-loop stability and performance. This is achieved by shaping the system's frequency response curve or by placing the dominant closed-loop poles in preferred *s*-plane positions.

If the transfer functions of equation 4.12 are expanded, the numerator of the resulting equation is made up of the numerators of $G_C(s)$ and $G_P(s)$ and the denominator of *H(s)*. The closed-loop zeros are therefore the zeros of the plant $G_P(s)$ and forward path controller $G_C(s)$ and the poles of the feedback element *H(s)*.

Zero positions

Adding a zero to a system reduces the system's rank (the difference between the number of poles and zeros) by one. The ultimate phase shift of a system is given by $-R90°$ where *R* is the rank of the system. The added zero reduces rise time, reduces the peak time, increases the overshoot and rotates the polar plot anticlockwise. The closer the zero is to the right-hand half of the *s*-plane the greater these effects. Zeros in the right half of the *s* plane can produce some peculiar responses and should be avoided.

Pole positions

Adding a pole to a system increases the system's rank and consequently increases the phase shift at high frequencies and twists the polar plot clockwise. An added pole tends to reduce the oscillations in a system and as it becomes more dominant, make it more sluggish.

Having zeros or poles close to an open-loop system's dominant poles can extensively alter its response, and these are reflected in the closed-loop response. Adding poles and

zeros in the compensator and feedback path are therefore very important in control system design.

4.3.6 PID control

The PID controller is a closed-loop controller that consists of three feedback terms, the proportional term, the integral term, and the derivative term.

Proportional Control (P)

With Proportional Control, the output of the controller is directly proportional to the error signal, and the action is given by:

$$y(t) = K_p e(t) \tag{4.13}$$

To achieve both high accuracy and rapid response, the value of K_p must be made large. Unfortunately, most systems contain lag (e.g. the inertia of a rotating load), so increasing the gain causes instability. This lag may alter the feedback from negative to positive causing the output to drive past the desired value (overshoots) and as it is corrected it swings either side of the value until it finally settles. In the worst case oscillations in the output will be set up.

Derivative Control (D)

A standard method of stabilising a close loop control system is to provide an additional feedback signal that is proportional to the rate at which the error changes. Derivative control action can be represented by:

$$y(t) = K_d \frac{de(t)}{dt} \tag{4.14}$$

where K_d is the derivative control gain.

A combination of proportional and derivative control will result in accurate, fast and stable response, but under certain conditions can cause the output to be offset from the input. Offset occurs because the derivative action will act to retard the output. The addition of another correction signal will eliminate this effect.

Integral Control (I)

With integral control, an additional correcting signal is applied to the controller input which is proportional to the integral of the error. This can be expressed as:

$$y(t) = K_i \int e(t).dt \tag{4.15}$$

where K_i is the integral gain constant. This control mode forces the controller to output a drive signal as long as any error exists, thus offset errors cannot occur.

PID Control

A closed loop system using a combination of proportional, integral and derivative control has the following control action:

$$y(t) = K_p e(t) + K_i \int e(t).dt + K_d \frac{de(t)}{dt} \tag{4.16}$$

$$= K_p \left[e(t) + \frac{K_i}{K_p} \int e(t).dt + \frac{K_d}{K_p} \cdot \frac{de(t)}{dt} \right] \tag{4.17}$$

Usually $\frac{K_p}{K_i}$ is called the integral time constant (T_I), and $\frac{K_d}{K_p}$ is called the derivative time constant (T_d), so the above equation can be written as:

$$y(t) = K_p \left[e(t) + \frac{1}{T_I} \int e(t).dt + T_d \frac{de(t)}{dt} \right] \tag{4.18}$$

This is the equation for the control action of analogue servo systems.

4.4 Control of the Test-rig for Machine Emulation – Method 1

Section 4.2 outlined the general control problem for the test-rig. The emulation function will be developed in the first instance, and then attention will be given to the characterisation function, which requires a slightly different control approach.

Two methods are investigated in this chapter to emulation a machine. Perhaps the most intuitive method is discussed in this section, and an alternative approach is discussed in the next section.

4.4.1 Machine Emulation Control Requirement

When performing an emulating function, a model of the emulated machine is used to derive an angle reference from past and present measured and internal variables (chapters 8 and 9 discuss the choice of machines and models in detail). It is a requirement of the test-rig controller to accurately control this output shaft angle so that the external machine (typically a drive motor pair) sees an accurate representation of the machine being emulated. Figure 4.9 shows the conceptual way by which the *external motion source* is connected to the test-rig during *machine emulation*, and also how the *external machine* is connected to the test-rig during *machine characterisation*.

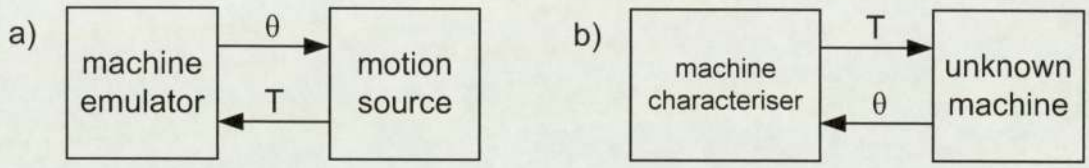


Figure 4.9 Conceptual connection of test-rig during a) emulation, and b) characterisation

4.4.2 Model of the Plant (test-rig) under control

Figure 4.5 shows a very simplified representation of the test-rig, and figure 4.4 shows the system variables of interest. This can be expanded by incorporating other aspects of the test-rig, such as its dynamics. The dynamics of the test-rig involve inertia, viscous damping and friction.

The application of torque to the test-rig inertia and viscous damping can represented by the equation

$$T = J\ddot{\theta} + B\dot{\theta}$$

Taking the Laplace transform of this becomes

$$\overline{T} = Js^2\overline{\theta} + Bs\overline{\theta}$$

which rearranges to

$$\frac{\overline{\theta}}{\overline{T}} = \frac{1}{Js^2 + Bs}$$

Both sides of the equation can be differentiated (multiplied by s)

$$\frac{s\overline{\theta}}{\overline{T}} = \frac{s}{Js^2 + Bs}$$

which is a transfer function relating torque to velocity

$$\frac{\overline{\dot{\theta}}}{\overline{T}} = \frac{1}{Js + B} \tag{4.19}$$

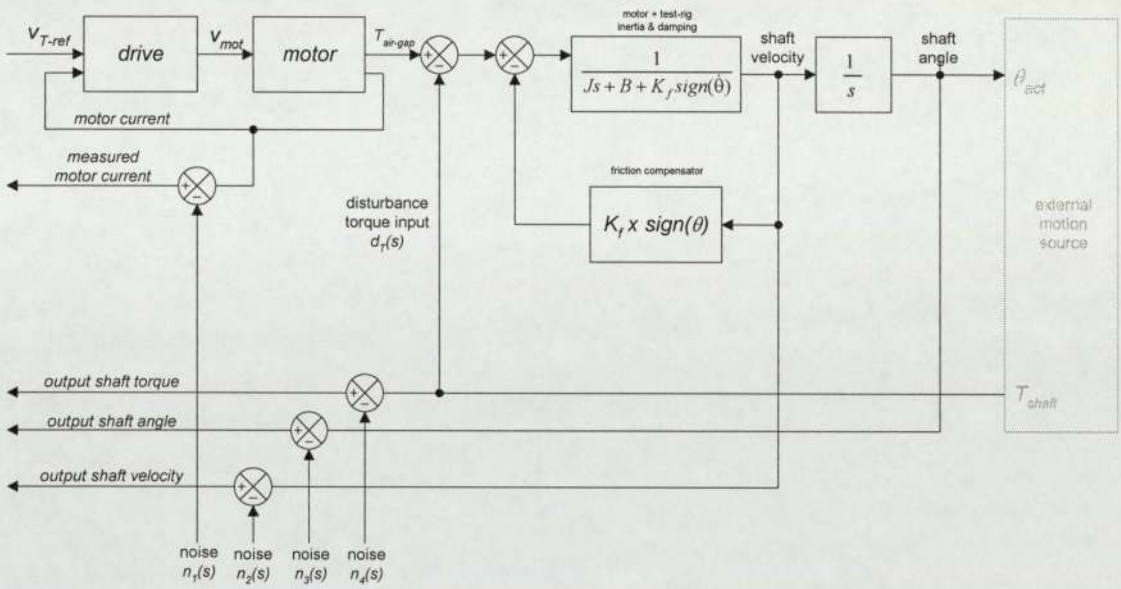


Figure 4.10 Detailed block diagram of test-rig

Friction is a non-linear function of velocity with units Nm such that

$$T_{friction} = K_{friction} \times sign(\dot{\theta}) \tag{4.20}$$

Since the test-rig is being modelled using Laplace Transfer Functions (LTF's) which cannot be used with any non-linearities, it is normal practice to place a non-linear control loop around any non-linearities such as friction. The friction and control loop

cancel, and the non-linear control part can be added to the linear controller at the time of implementation. The two blocks labelled *test-rig friction* and *friction compensator* can therefore be ignored for now because they cancel to zero.

As discussed in section 4.2.1, the response of the drive's current control loop is very fast and linear. This combined with the electrical time constant of the motors is many times faster than the response of the other dynamic parts of the system, and can therefore be considered ideal (within their operating range). The drives and motors will therefore be treated as a constant k_{dm}. The two drives were adjusted to give 8.155 A per V_{in}, and the motors each produce 0.064 Nm/A. Each drive has two motors connected in series to it, so total torque produced is therefore 8.155 x 0.064 x 4 = 2.088 Nm/V_{in}. Multiplying this by the gearbox ratio gives 2.088 x 2.395 = 5 Nm/V_{in}.

The external motion source from a test-rig perspective has one output, torque, and a corresponding input, angle. Since the torque from the external motion source is not predictable, it is shown as a disturbance input to the test-rig. The four measured variables also include signals $n_n(s)$, which represent measurement errors or noise on the feedback signals.

To design a controller for the test-rig, the lumped parameters inertia and damping are required. The most significant inertias are in the gearbox and torque transducers – the motors have a very low inertia (7.1×10^{-6} $kg.m^2$). The referred inertia calculations for the test-rig are in appendix C2.5; using the gearbox Mk-I it is 2.15×10^{-3} $kg.m^2$, and using the gearbox Mk-II it is 0.63×10^{-3} $kg.m^2$, as referred to the output shaft. The controller design will be based on using the gearbox Mk-I as this was used for most of the test-rig development. Viscous damping is small in the Mk-I gearbox and motors, but is still significant. Since it will only effect the controller parameters and not the dynamics of the test-rig, it can be over-estimated for now as 215×10^{-6} $Nm/rad.s^{-1}$ which is also mathematically convenient for the test-rig LTF. Friction is a significant factor, much more so than viscous damping, but this will be dealt with later as described above. The three parameters of importance now are therefore:

$J = 2.15 \times 10^{-3}$ $kg.m^2$, $B = 215 \times 10^{-6}$ $Nm/rad.s^{-1}$, and $K_{dm} = 5$ Nm/V_{in}.

The overall LTF of the test-rig is

$$\frac{K_{dm}}{Js^2+Bs} = \frac{5}{(2.15\times10^{-3})s^2+215\times10^{-6}s} \approx \frac{2326}{s^2+0.1s} \tag{4.21}$$

4.4.3 Equivalent state-space form

Section 3.4 in chapter 3 described state-space models and how to convert a LTF to state-space form. The LTF of the test-rig can be converted to state-space form, which is convenient for simulation and implementation.

The linear part of the test-rig equation can be written in the time-domain as:

$$T = \frac{J}{K_{dm}}\ddot{\theta} + \frac{B}{K_{dm}}\dot{\theta} \tag{4.22}$$

This can be used to derive the A, B, C and D matrices, or an alternative (and far easier method for more complex systems) is to use the Matlab Control-toolbox function `tf2ss`. Using the following command lines

```
» num=2326;
» den=[1 0.1 0];
» [A,B,C,D] = tf2ss(num,den);
```

gives A, B, C and D matrices:

$$A = \begin{bmatrix} -0.1 & 0 \\ 1 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad C = \begin{bmatrix} 0 & 2326 \end{bmatrix} \qquad D = 0$$

4.4.4 Open-loop Poles and Eigenvalues of the test-rig

Section 4.3.2 and 4.3.3 discussed the stability of Laplace-transfer and State-space models respectively. The eigenvalues of the state-space form are the same as roots of the Laplace transfer function, and can be found using the Matlab function `pzmap` (or by inspection).

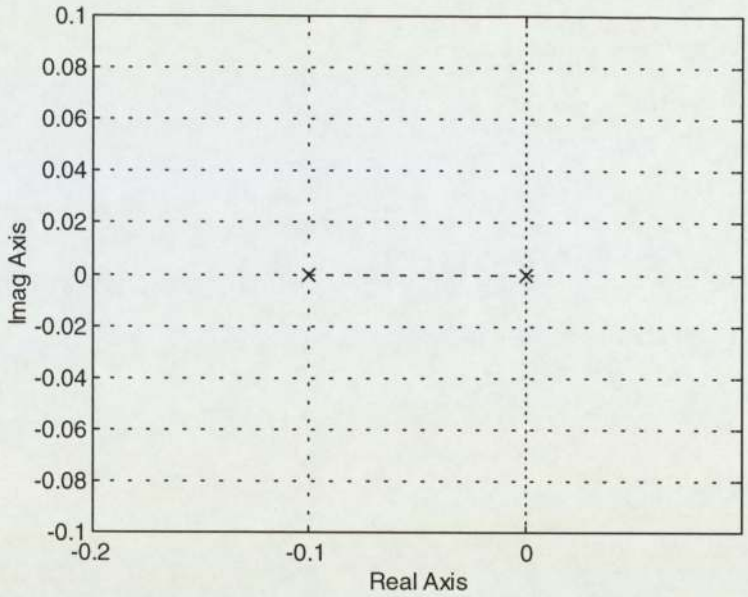


Figure 4.11 Pole-zero map of the test-rig using `pzmap`

The system has no zeros except two at infinity (in general the number of zeros at infinity is equal to the number of poles minus the number of numerator zeros). The poles are at –0.1 and 0, so the system is a type 1 system. Both roots are real and the dominant pole is at the origin, which means that the system is predominantly first order and marginally stable. Its open-loop response to an impulse and a step input shown in figure 4.12a and b respectively.

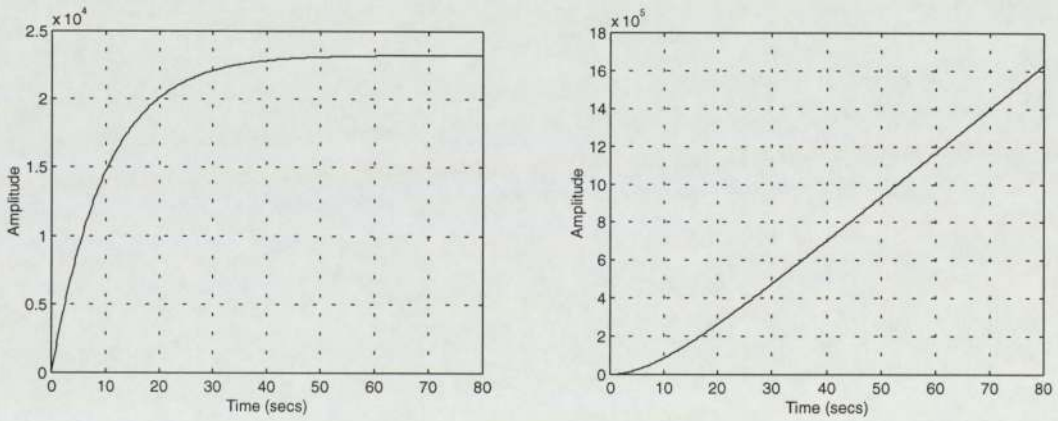


Figure 4.12 a) impulse and b) step responses of the open-loop test-rig

It is clear that the open-loop test-rig is unacceptable for use as a shaft positioning system, and this is the control problem.

4.4.5 Closing the test-rig control loop

Figure 4.10 shows the test-rig in detail. This can now be reduced slightly to show only the blocks of interest, and to incorporate a closed loop controller, shown in Figure 4.13.

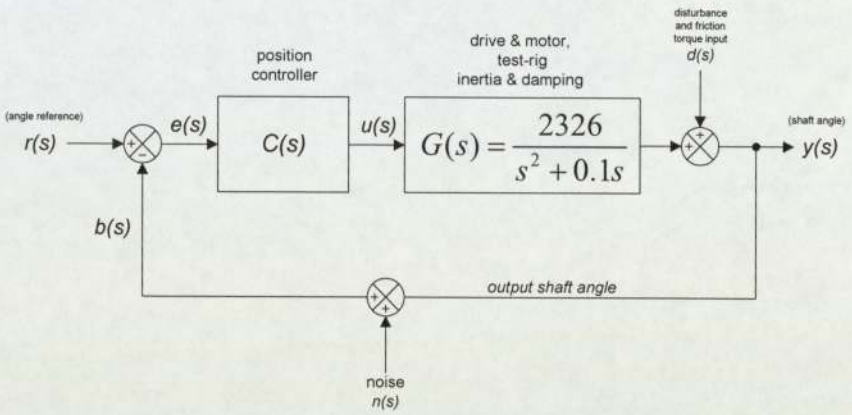


Figure 4.13 Block diagram of test-rig with a position control loop

The external disturbance *d(s)* is shown here as a disturbance to output-shaft angle. The input $\theta_{ref}(s)$ and output $\theta_{shaft}(s)$ have been replaced by *r(s)* and *y(s)* for clarity. [2] Block diagram reduction on figure 4.13 will show that the closed loop output for this system is given by

$$y(s) = \frac{G(s)C(s)}{1+G(s)C(s)}\left[r(s)-n(s)\right]+\frac{1}{1+G(s)C(s)}d(s) \tag{4.23}$$

and the controller output by

$$u(s) = \frac{C(s)}{1+G(s)C(s)}\left[r(s)-n(s)-d(s)\right] \tag{4.24}$$

Both equations 4.23.and 4.24 contain the common denominator term or *characteristic equation* $1+G(s)C(s)$ and the system's dynamics are therefore independent of external disturbance and noise inputs. It is also worth noting that the closed-loop characteristic equation is given by *1 + the system's open-loop transfer function.*

The disturbance input and non-linear friction can therefore be ignored for now, which reduces the block diagram even further to figure 4.14.

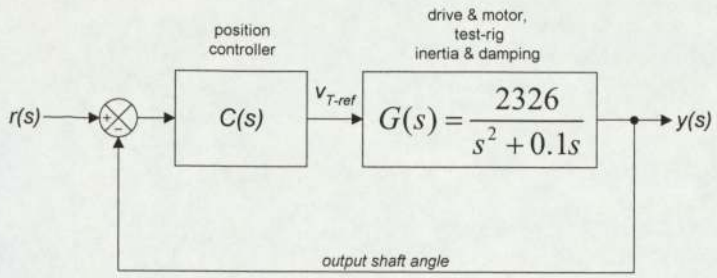


Figure 4.14 Simplified block diagram of test-rig with a position control loop

The closed-loop response can now be plotted, and as a starting point, the controller was set to a gain of 1. The response of the system to a step input is shown in figure 4.15.

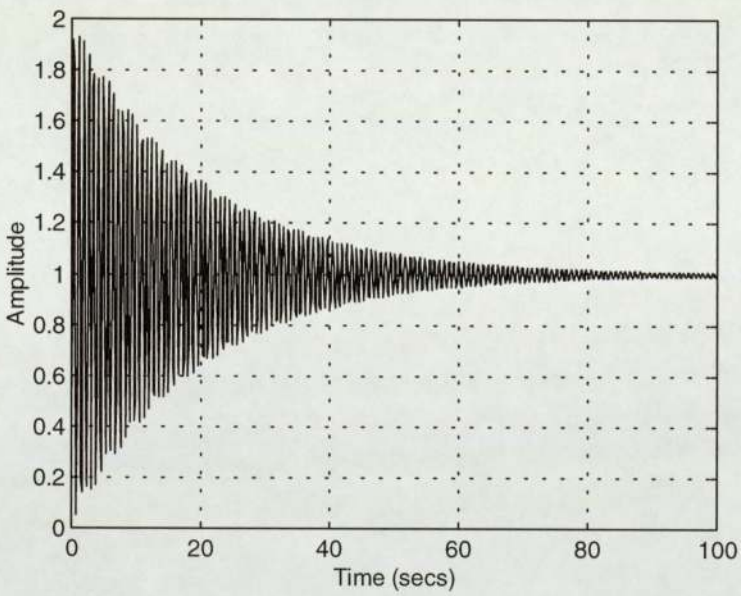


Figure 4.15 Step responses of the test-rig with controller gain=1

With a controller gain of 1, the system has a closed-loop gain of 2326, the poles are -0.05 +48.2286j and -0.05 -48.2286j, and the system is massively under-damped.

4.4.6 Selection of test-rig gain using the Root Locus Plot

The root locus diagram is based on a system's open-loop transfer function such as equation 4.11, which will give the closed-loop pole positions. It is a plot of the locus of the positions in the *s* plane of the roots of the characteristic equation as the gain (or any other variable) varies from zero to infinity [6]. It shows which roots are dominant, how

close the other roots are, and how the positions of the dominant roots vary as K varies. Matlab has a function `rlocus(num,den)` which calculates and plots the locus of the roots of the closed-loop characteristic equation

$$H(s) = 1 + K\frac{num(s)}{den(s)} = 0$$

for a set of gains K, where $\frac{num(s)}{den(s)}$ is the system's open-loop transfer function (this can be easily assimilated to equations 4.11 and 4.12).

Figure 4.16 shows a root locus plot of the test-rig transfer function, equation 4.21.

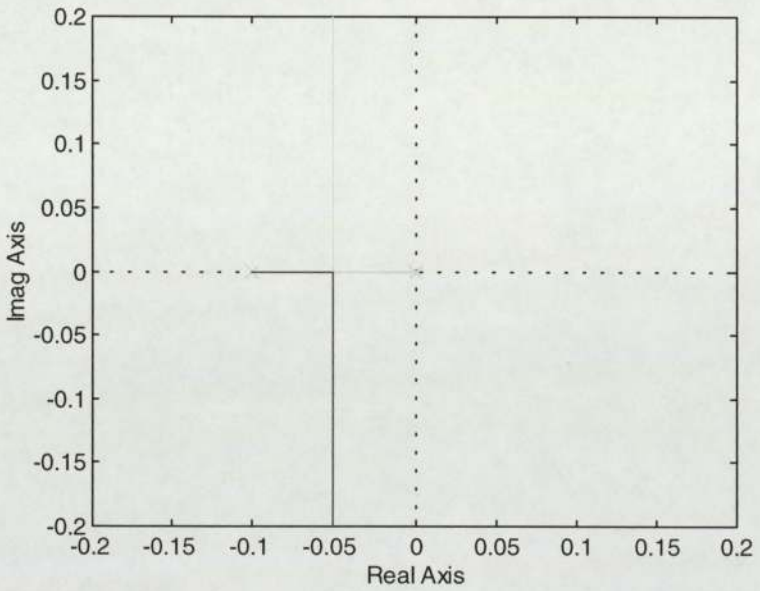


Figure 4.16 Root locus plot of the test-rig transfer function with $C(s)=1$

With a closed-loop gain of 2326 the system is under-damped as shown in figure 4.15. Using the Matlab function `rlocfind`, the gain and pole positions can be found at various points on the plot using a graphical pointer. The gain at the point where the two loci meet is 1.0748×10^{-6}, and there are two poles at this point at -0.05. Setting a controller gain to 1.0748×10^{-6} makes the system gain 2.5×10^{-3} and the step response of this is shown in figure 4.17a.

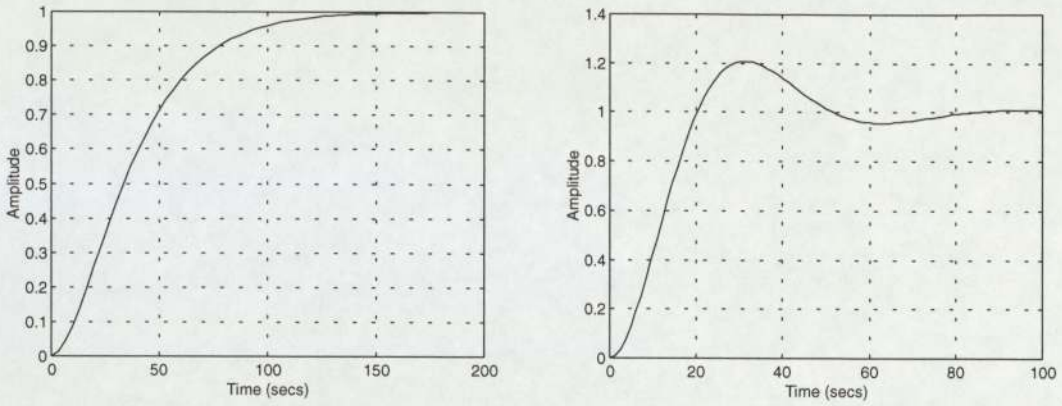


Figure 4.17 Step responses of test-rig with gain of a) 2.5 x 10^{-3} and b) 12.5 x 10^{-3}

It can be seen that by reducing the gain the system becomes less oscillatory, but the settling time is increased. By increasing the gain by a factor of 10 the system has two poles at -0.05 ± 0.1j settling time is marginally improved but the overshoot is now unacceptable – see figure 4.17b.

It is therefore evident that a proportional controller alone is insufficient to control the test-rig.

4.4.7 Frequency Response of the test-rig

This section considers the response of the test-rig to a sinusoidal forcing input. The test-rig is a minimum-phase system (all poles are in the left half of the *s*-plane) and is a system of rank 2. [2] As frequency increases therefore, the phase-shift will ultimately be $R \times -90 = -180°$.

Figure 4.18 shows a bode plot of the simulated test-rig, with a gain of 12.5 x 10^{-3}.

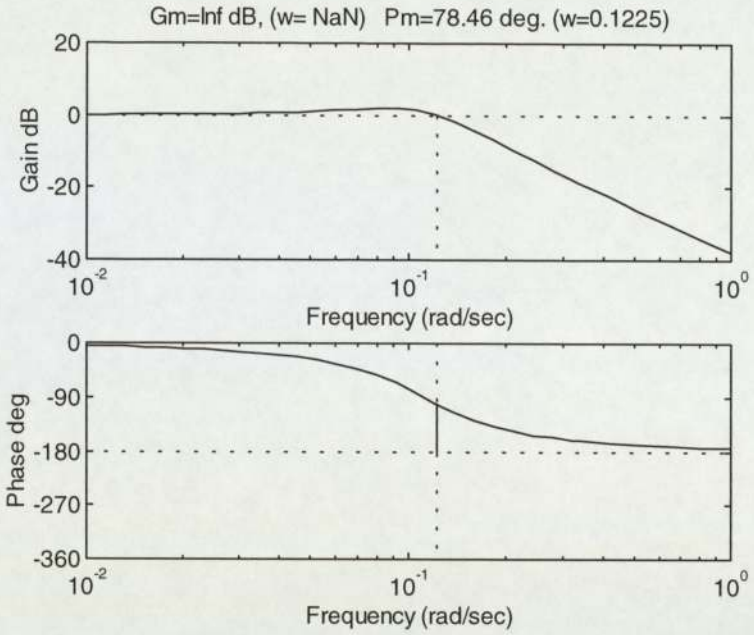


Figure 4.18 Bode-plot of the test-rig with gain set to 12.5 x 10^{-3}

The gain margin is found to be ∞ and the phase margin 78.46°. The system is therefore stable at all frequencies. A general rule [2] is that the gain margin should be between 2 and 2.5 and the phase margin between 45° and 65°. The gain margin is meaningless here because the system never reaches -180° phase shift. The phase margin is slightly higher than recommended, which indicates a higher damping ratio than recommended.

4.4.8 Steady-state Error

The final value theorem states that the final value of a system's output in response to, for example, a unit step input is given by

$$f(\infty) = \lim_{t\to\infty}[f(t)] = \lim_{s\to 0}[sF(s)]$$

For the test-rig with a gain of 12.5 x 10^{-3}, the closed loop transfer function is

$$\frac{0.0025}{s^2 + 0.1s + 0.0025} \qquad (4.25)$$

the final error to a step input is the difference between the input and steady-state response:

$$1-\lim_{s\to 0}\left[s\times\frac{1}{s}\times\frac{0.0025}{s^2+0.1s+0.0025}\right]$$

which is zero. The final error to a unit ramp input is infinity.

4.4.9 Position Controller

Section 4.3.6 described the PID controller from a time-domain perspective. This section discusses its application to the control of the test-rig from a Laplace domain perspective.

The Laplace transfer function of the PID controller of equation 4.16 is

$$G_C = K_P + \frac{K_I}{s} + sK_D \qquad (4.26)$$

where K_P = proportional gain, selected for adequate rise time
K_I = integral gain (units: gain / second), selected for steady state accuracy
K_D = derivative gain (units: gain x seconds), selected to reduce oscillations and improve settling time

This can also be written in terms of integral and derivative time constants, equivalent to the time domain equation 4.18

$$G_C = K_P\left(1+\frac{1}{sT_I}+sT_D\right) \qquad (4.27)$$

where T_I = integral action time (units: seconds) = K_P / K_I
T_D = derivative action time (units: seconds) = K_D / K_P

The proportional term is simply a gain. Low values of K_P tend to give rise to stable responses but high steady state errors, while higher values tend to improve the steady state errors but worsen the transient response, and if too high, make the system unstable.

The integral term gives a ramp output to any error input, and is therefore used to remove steady-state error. Integral action increases the system's type number by one, and the introduced pole at the origin of the *s*-plane tends to make the system more sluggish. Integral action can also make the system less stable though because it introduces a 90° phase lag which increases oscillation and reduces the phase margin. In PI controllers, this can usually be offset by reducing the gain of the P term.

If the rate of change of error is large then a large overshoot is likely, and derivative action will correct for this. Derivative action decreases the system's type number by one (introducing a zero at the origin), causing a 90° phase lead, and an increase in the bandwidth of the system making it more sensitive to noise. The increase in damping allows higher gains of P and I terms to be used. Derivative action has no effect on the steady state response. It should be used with care as any noise is amplified, which is usually dissipated in the system, most probably in the motors as heat.

Applying this to the test-rig

The general requirement for the test-rig controller is that the test-rig must respond in as short a time as possible with minimum overshoot and offset, but it will inevitably be a trade off between these. The system is critically damped when the controller gain is 1.0748×10^{-6}. At this gain, the system has two poles at -0.05 (from figure 4.16 using Matlab), but the settling time is too long.

Figure 4.17a shows the response of this system to a step input. What is required is a PD controller to damp out the oscillations and reduce the settling time.

4.4.10 Lead Compensation

In practice, ideal derivative action cannot be achieved, so derivative action is approximated using a *lead-compensator*. The equation generally describing this is

$$T_D s \approx \frac{T_D s}{\alpha T_D s + 1} \qquad (4.28)$$

where α is less than 1 and normally greater than 0.1. A pole is introduced at $-1/\alpha T$ and a zero at $-1/T$. The transfer function of a practical PD controller is therefore

$$C(s) = \frac{K_D s + K_P}{\alpha K_D s + 1} \tag{4.29}$$

and the transfer function of a practical PID controller is

$$C(s) = \frac{K_D s^2 + K_P s + K_I}{s(\alpha K_D s + 1)} \tag{4.30}$$

Since there is no steady state error, an 'I' term is not required, so a PD controller will be used. The values of KP and KD were found by trial and error and using KP = 0.5x10-2, KD = 10, and α = 0.05, the following response was obtained

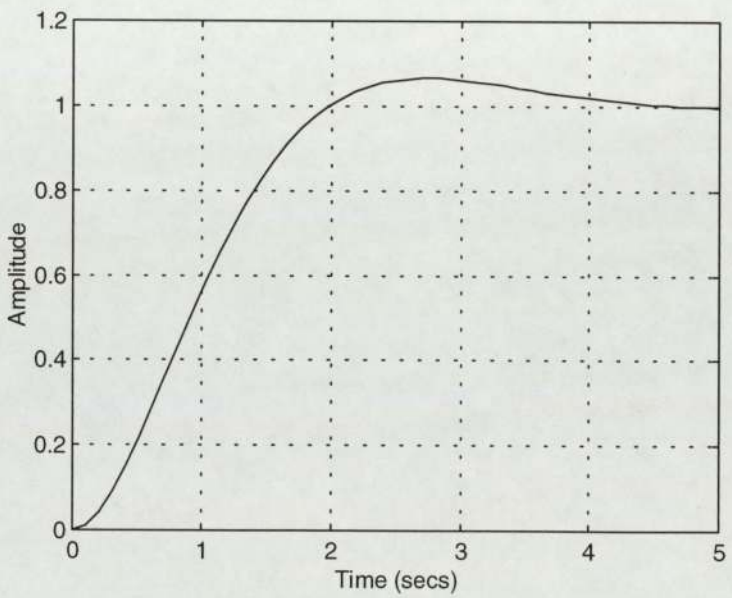


Figure 4.19 Compensated response of the test-rig

The Matlab script to obtain this response is in appendix A.3. The behaviour of the lead compensator is determined by α and K_d, and α is difficult to select. With an ideal differentiator ($\alpha = 0$) a near perfect response could be obtained, but this is of course unobtainable in practice. The settling time is many times better than before, but still unacceptable to this project.

4.4.11 Velocity Feedback

In motion control, it is conventional to close control loops as shown in figure 4.20.

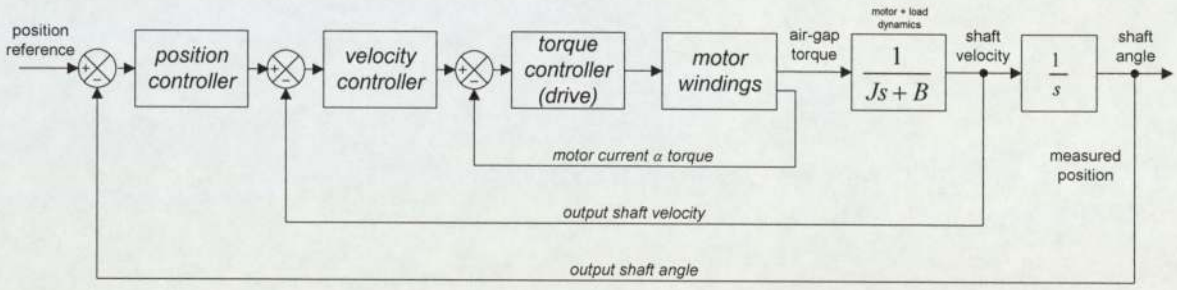


Figure 4.20 Position-Velocity-Torque control loops

The torque control loop's dynamics depends mainly on the internal delays, set up by the electrical time constant (L/R≈0.16ms), and its behaviour is first-order with a very short time constant. For frequencies of velocity control, the torque loop will be virtually zeroth-order.

Velocity feedback is a control loop outside the torque control loop, but within the position control loop. The velocity of the shaft is monitored using a velocity observer discussed in section 4.1.3. Its constructional details are in chapter 7. Using this signal avoids the problems of amplification of any high frequency noise component of the signal inherent in the process of differentiation of the position signal. The velocity feedback acts as damping to the position control system allowing higher position loop gains than before. The characteristic equation is the same for P + D control, but the numerator of the overall transfer function is different, resulting in a similar but different dynamic response, better than that of position and torque loops alone.

4.4.12 Velocity Feedforward

[9] The main disadvantage with the above scheme is that the position control loop has to drive the velocity loop, which can add delay for a fast changing signal. This problem can be overcome by feeding forward a signal representing the velocity demand, as in figure 4.21.

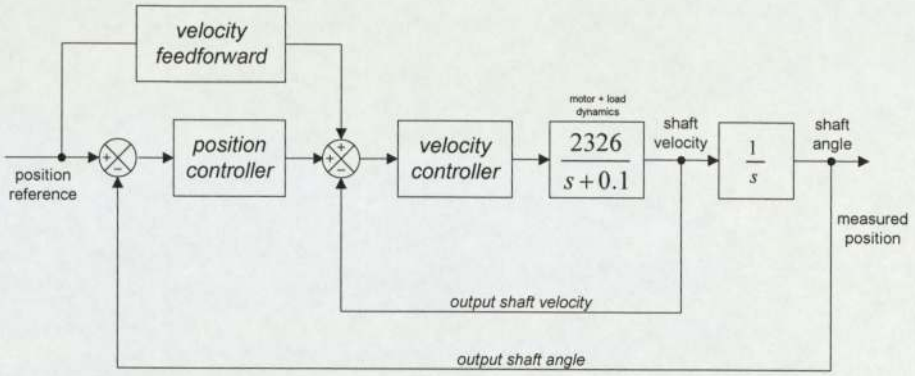


Figure 4.21 Control loops incorporating velocity feedforward

A typical gain for this transfer function (ideally sG_f) is 0.75. The problem with this scheme is that the velocity feedforward transfer function incorporates some kind of differentiation approximation, which tends to amplify errors. The advantage is that this addition does not affect loop stability and behaves like an external disturbance which enhances performance.

4.4.13 Practical Test-rig Controller

The controller designs above have all been based on a noise free system with ideal characteristics and zero delays in many parts of the system. In practice, there will be many delays in the system, particularly in the drives and motor electronics, and position and velocity observers/processing electronics. The DSP board which will contain the controllers will have delays in the anti-aliasing and reconstruction filters, and also in the processing of data. The system will also contain some non-linear behaviour, for example backlash in the gearbox (although this has been minimised). The controller designs above are useful as a starting point, and armed with this simplified behaviour it is possible to design controllers which can be adjusted in the field to match the system taking into account other previously ignored factors. Methods have been developed to set up controllers once in place, perhaps one of the best known is the Zeiger-Nichols method (J.G. Zeiger and N.B. Nichols, 1942), and other more recent methods [10].

Because the system can operate quite differently in practice than in simulation, a general controller will be designed and tuned when in place. The performance of this will be reviewed using real data.

PID is the most common form of controller and will be adopted here. Integral action can make a system sluggish, and in a velocity loop acts identically as a position loop and increases open-loop phase shift which will limit the maximum gain before instability. It is therefore highly undesirable in the velocity loop as it can cause stability problems. Ideal derivative action increases the bandwidth of the system but can cause problems by amplifying noise. A lead compensator is closer to a practical derivative controller, but its implementation can incur a computational overhead.

A control system shown in figure 4.22 was initially considered since this would be the most flexible. It was uncertain that the velocity feedforward or I-term of the position controller would be needed, but these could always be removed by setting their coefficients to zero. Ultimately this system was not implemented because the system described in section 4.5 was applied instead.

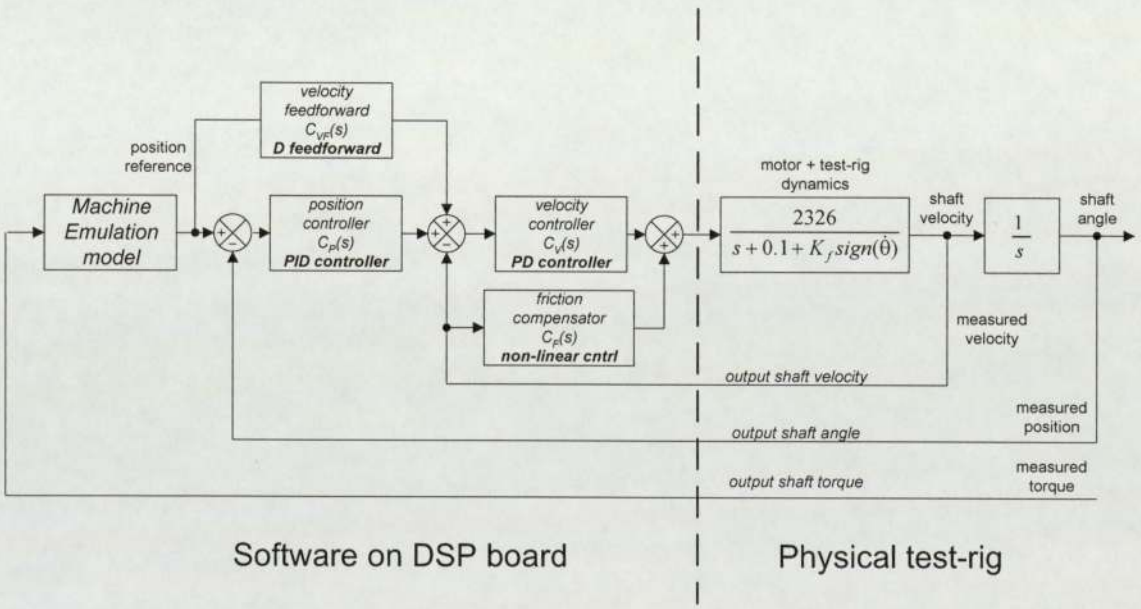


Figure 4.22 Implementation of test-rig emulation control – Method 1

Section 4.7 describes the software implementation of the controller components and of the overall controller.

4.5 Control of the Test-rig for Machine Emulation – Method 2

Section 4.4 offered one approach to machine emulation. This assumed that the model of the load took shaft-torque as the input, and produced an angular-displacement output. It would be convenient instead to have a model which produced a torque output given an angular displacement input. This would allow torque to be controlled instead which is much easier as it is not integrated by the test-rig as velocity and angle are. The problem is that the models would need to be 'inverted' which is mathematically complex. However, there is a way to invert a model approximately without incurring all of the problems associated with model inversion.

The way this is done is to put a stiff controller* around the model that is being emulated such that the model is driven to always track the angle and angular velocity being measured. The input to the (open-loop) model is the shaft-torque. This is described by figure 4.23.

*Stiff refers to a system which has a mixture of relatively fast and relatively slow dynamics, neither of which can be ignored.

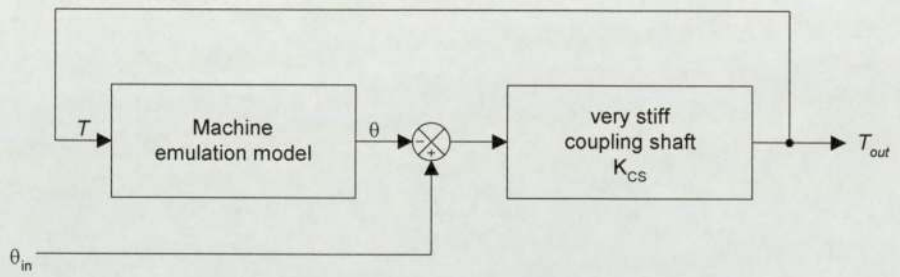


Figure 4.23 'Model Inversion' Approximation

There is a big advantage to using a very stiff controller to control a numerical model compared with using a very stiff controller to control a real model (the test-rig) to get angle and angular velocity. In one case, there is noise, delay and uncertain dynamics. In the other case, there is not. In practice however high values of K_{CS} in digital control will lead to instability, and only approximate models are achievable using lower values of K_{CS}.

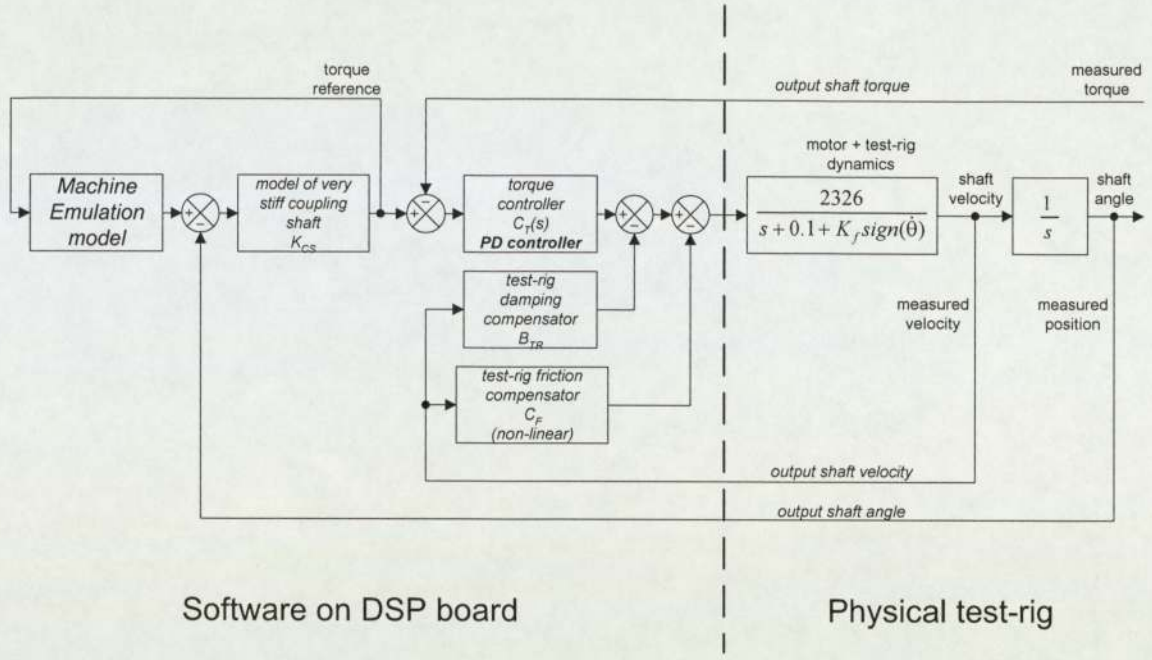


Figure 4.24 Implementation of test-rig emulation control – Method 2

Figure 4.24 shows the controller structure chosen for the torque control method of machine emulation. The 'model inversion approximation' method shown in figure 4.23 is used to derive the torque reference which is fed into the shaft-torque controller. The D term is incorporated to compensate for increased rise times due to the test-rig inertia, but this also amplifies noise on an already noisy signal, and so is unlikely to be used. The losses due to damping and friction effect the torque directly, and the errors due to these will appear directly in the torque error term. The damping and friction compensators are therefore unlikely to be beneficial, but their effect will be observed during practical testing of the controller.

It can be seen that the only difference between figures 4.22 and 4.24 is the control scheme. Both incorporate an identical model of the machine being emulated, physical test-rig, and measured variables. The scheme shown in figure 4.24 was chosen since torque control is required, and tested using test profiles on a test set-up (section 4.8), and then using machine-emulation-models performing an emulation function (chapter 9).

4.6 Control of the Test-rig for Machine Characterisation

As discussed in section 4.2, during machine characterisation the nature of the torque stimuli and the relationship between the resulting torque and angle is of importance. The precise control of the external machine is not important, but the exercising of all of the dynamics of the machine being characterised is. Since the types of machines to be characterised are most likely to be heavily cyclic, this involves applying torque stimuli over a range of angles. This means that ultimately the angle also needs to be controlled, but only very loosely. The angle will be varying due to the torque so only the mean angle needs to be adjusted by a slight torque offset, essentially providing a static offset. A structure similar to figure 4.24 is therefore appropriate, but incorporating extra control over the angle as shown in figure 4.25.

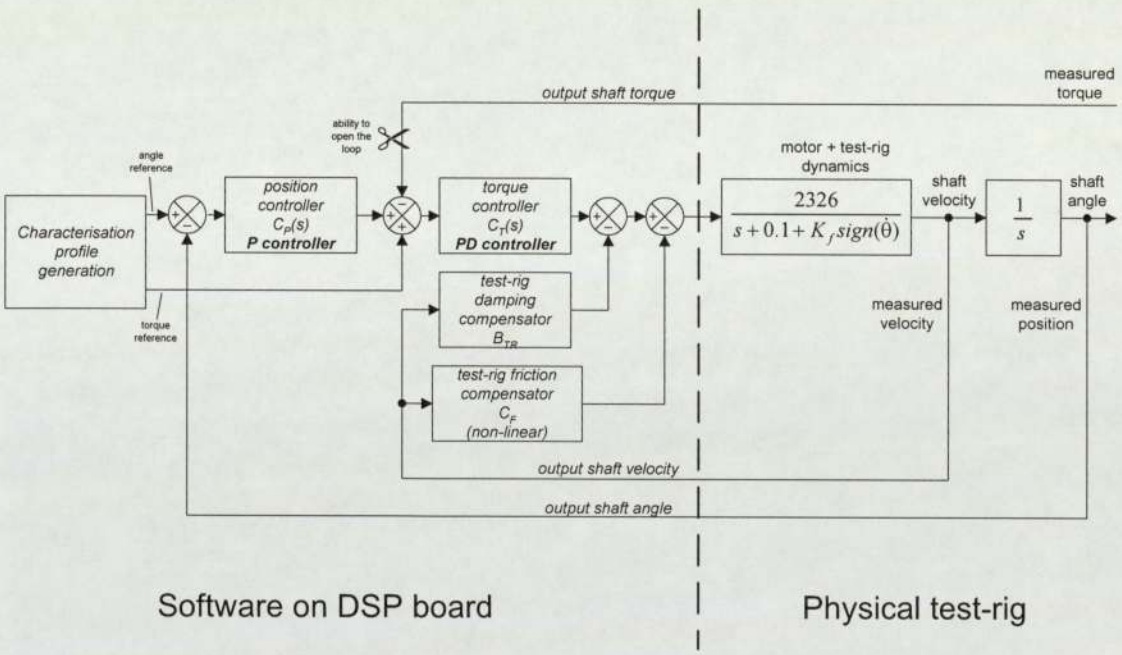


Figure 4.25 Implementation of test-rig characterisation control

Since tight control of the angle is not required, a proportional only controller is used for shaft position. The same torque controller as described in section 4.5 is used to control the torque, but since the machine being characterised has unknown dynamics the D term will be omitted for stability. For some system identification methods (chapter 5) it may be necessary for the system to be driven open-loop. Torque is the most significant loop to cut, as this constitutes the system identification 'input'. Provision has been made for operating the test-rig open-loop by allowing the torque feedback to be cut, and for the

friction and damping coefficients to be made equal zero. This may be beneficial to some system identification methods discussed in chapter 5. It is expected that the way the angle is controlled means that it will not affect the dynamic properties of the system significantly.

4.7 Software Implementation of Controller Components

The torque control blocks of figures 4.24 and 4.25 are identical in that they both generate a torque reference using the shaft angle. The only difference between these control strategies is that the emulation control generates a torque reference from a machine model, and the characterisation control generates a torque from a perturbation strategy. Control code was therefore written which is common to both, and is flexible such that the torque reference and control parameters are easily changed. The general code written to control the test-rig is in appendix B.1. Minor modifications are required for the application to different tests, and are discussed in the context of the test descriptions.

The DSP board has four analogue inputs and four analogue outputs. Three of the inputs are used for angle, velocity, and torque, and the fourth is spare. One of the outputs is connected to the test-rig drives, and one is used to feed a perturbation signal to a drive-motor pair used to apply torque to the test-rig when it is emulating a machine. A sampling clock is applied to the DSP board that is fixed at 10 kHz, and anti-aliasing filters (described in chapter 7) are used to filter the velocity and torque signals prior to sampling.

Function calls are kept to a minimum because they incur an overhead due to stack usage and program jumps. The control program shown in appendix B.1 has 15 distinct sections.

1) Variable and macro definitions

The variables used throughout the program are defined here, and include sufficient storage for past and present values of angle, velocity and torque. Macros are also defined that specify the DC offsets of the analogue inputs, and the scaling factors for the signals.

2) Read in control parameters

Frequently changed parameters are stored in a separate text file ("ctrlpars"), and are read in before the control loop executes. This allows parameters to be changed quickly without the need for program re-compilation.

3) Initialise memory and DSP

Enough memory is dynamically allocated for data logging. The timer for measuring sample rates (used for program performance analysis and debugging) is initialised, and the ADC's are stopped and their FIFO's emptied. A user response is then required to continue, and sampling is then started.

4) Input signals

Function calls to `read_next_voltaged()` return the most recent inputs from the ADC's.

5) Scale and filter input signals

The input signals are scaled into meaningful quantities and offsets are removed. Provision has also been made for the digital filtering of signals subsequent to sampling. Only the torque and velocity are filtered to remove noise, as the angle is required to have sharp transitions between $-\pi$ and $+\pi$.

6) Generate torque requirement

The torque requirement is calculated using either:

a) a machine emulation model and stiff shaft coupling model (figure 4.24), or
b) a torque profile and angle profile (and associated control) for characterisation (figure 4.25)

7) Angle control algorithm

This algorithm (see 6b above) is used for generating a torque for angle control during characterisation. The control is PI and loosely controlled so as to not correlate the input and output signals excessively (chapter 5 discusses this in detail).

8) Damping and friction compensation

When the damping (B_{TR}) and friction (K_{FRIC}) are known they can be used here no negate the non-linear friction and the damping of the test-rig.

9) Torque control algorithm

The torque controller is a PID control algorithm and the parameters are defined in the "ctrlpars" file. The I and D terms are trapezoidal and finite-divided-difference methods respectively, and although crude are sufficient for this application. The next two sections discuss the differentiation and integration methods in more detail.

10) Output signal(s)

The output signals are scaled to translate from meaningful units to a voltage that is applied to test-rig motor's drives.

11) Timing issues

The timer measuring the period between samples is reset, and the sample counter is reset. This allows the data logging to store the data in the correct position of the storage arrays.

12) Logging data

The data is stored in arrays for retrieval later (when the test is complete).

13) Maintaining a sufficient history of variables

A number of past and present values of data are stored, which are required for the control loops and emulation models.

14) Shutting down the test-rig

Once the test is complete, the test-rig is shut down by applying a zero torque reference to the drives, and a message is printed to notify the user.

15) Saving recorded data to file

The data is saved to a file in a text format, which although not particularly compact is suitable for use with Matlab.

4.7.1 Differentiation in Software

A basic form of differentiation can be found by

$$f'(x_i) = \frac{f(x_t) - f(x_{t-1})}{x_t - x_{t-1}} \tag{4.31}$$

This is sometimes referred to as a *rectangular-slope* method [28], and calculates the slope between the current and previous sample, where the period is of time $x_t - x_{t-1}$ ($\equiv T$). It is also known as a *first backward finite difference* equation [24]. This and other divided difference equations are developed from the Taylor series to approximate derivatives numerically, for example *backward* and *centered difference* approximations. The errors of this differentiation method is proportional to the square of the step size, and more accurate approximations of the first derivative can be developed by including higher-order terms of the Taylor series. Approximations can also be developed for higher-order derivatives, but are not used here. Incorporating the second-derivative term for example, gives the equation

$$f'(x_t) = \frac{3f(x_t) - 4f(x_{t-1}) + f(x_{i-2})}{2h} \tag{4.32}$$

and is more accurate. This equation is known as a *second-order backward finite-divided-difference* formula.

Matlab script is shown in appendix A.4 ("diffrtn4.m" and "taylex1.m") which differentiates integrated torque data (f_{TR}) from a simulation by using two methods. The first is the rectangular-slope method, and the second is the backward finite-divided-difference method (using 5 previous values and a highest derivative of first-order). A graph plotting the results is shown in figure 4.26.

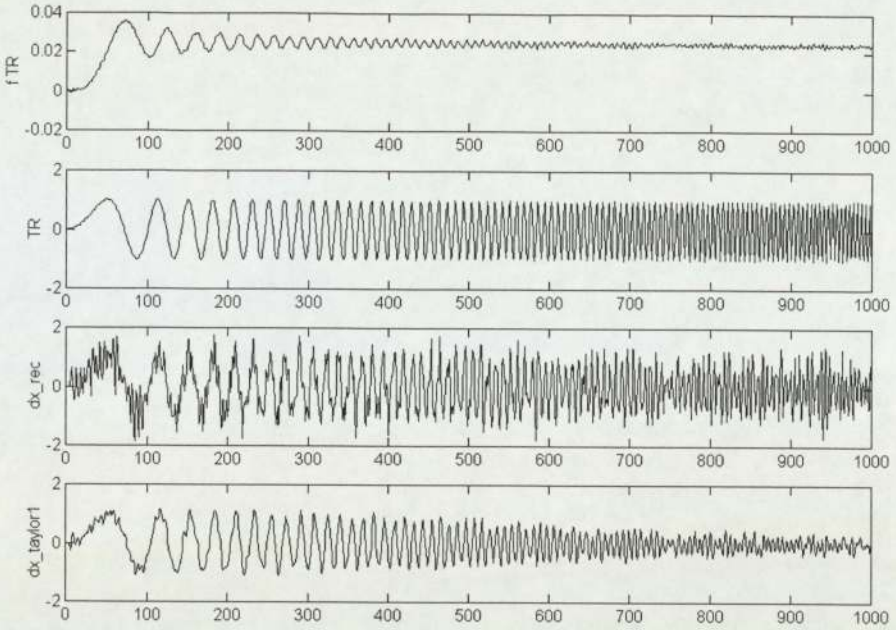


Figure 4.26 Graph comparing two data differentiation methods

The Taylor's method has better noise immunity but filters the signal, since at higher frequencies the signal has significantly less amplitude than the true value of torque. Both these were tested using the test-rig but the backward finite-divided-difference method was actually employed because of its better noise immunity.

4.7.2 Integration in Software

Two integration methods were tried, both with similar results. The *rectangular* (or strip) method calculates the area under the signal by assuming the next sample will be the same as the current one, so if I is the integral of a signal x,

$$I_{rec}(t) \approx I_{rec}(t-1) + x(t) \times T \tag{4.33}$$

The *trapezoidal* method calculates each 'strip' under the curve by calculating the height as the previous sample plus half the difference between that and the previous sample. I.e. it is approximating the area of the trapezoid under the straight line connecting the data points $x(t\text{-}1)$ and $x(t)$,

$$I_{trap}(t) \approx I_{trap}(t-1) + \frac{T \times \left(x(t) + x(t+1)\right)}{2} \tag{4.34}$$

4.8 PID tuning of the test-rig

The torque control I and D parameters are coefficients of the error integral and derivative respectively. This approach (as opposed to using integral and derivative action time) was taken because it allows these terms to be disabled easily by simply setting them to zero. The Zeigler-Nichols rules for controller tuning [2,10] is widely used but tends to produce an underdamped response, which can usually be reduced by lowering the P-gain. Increasing the D-gain increases response speed, but because of the noise amplification was found to be of limited use. The basic approach adopted was to increase the P-gain (with I-gain and D-gain set to zero) until the system became oscillatory, and then back it off slightly. To avoid the problems of amplification of any high frequency noise only a small amount of D-gain could be applied. When this was increased the motors became noisy and consumed excessive power.

During experimentation, the test-rig became very oscillatory and broke the centre gear of the spur-gear gearbox, as figure 4.27 illustrates. The replacement gear was made from steel, and substantially lightened by removing much of the material from within the gear.

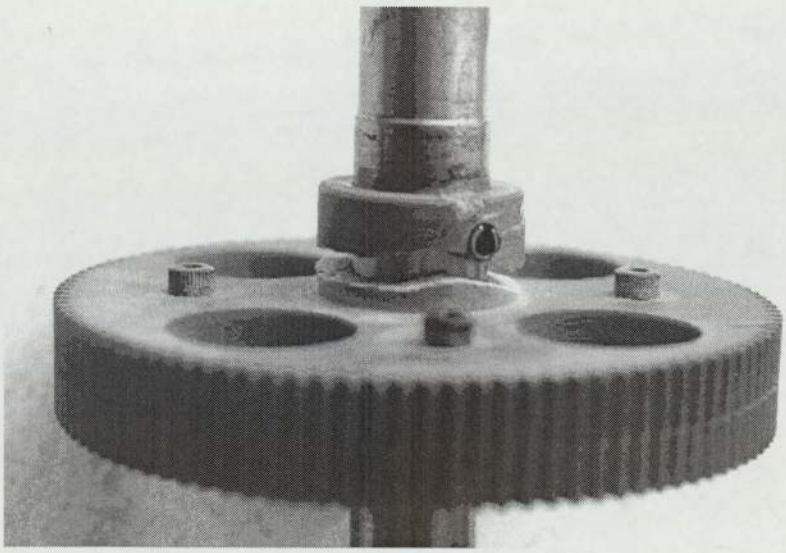


Figure 4.27 Gear broken due to unstable control

CHAPTER 5

5 System Identification and Parameter Estimation

Chapter 3 discussed various system models and their output responses to various inputs. Given sufficient inputs and corresponding outputs of a system, it is possible to construct a model of that system. This process is known as *system identification*. If the model has parameters, they will need to be found. This process is known as *parameter estimation*.

There are five main aspects to the system identification problem:

1) the choice of input stimuli within a permissible range which will exercise all dynamic aspects of the system,
2) the manipulation of data to remove irrelevant components,
3) the choice of system model,
4) the fitting of parameters, and
5) the testing of the model.

The above process is iterative and if after step 5 the model is not good enough, then it will be necessary to go back to step 3 or 4, or even 1 or 2. The amount of user interaction required means that identification cannot be brought into a fully automated procedure.

Matlab has a 'System Identification' toolbox, which has several functions to perform each of the above steps. Since most of the numerical computation required will be performed using this software, this chapter will not discuss the methods in fine detail, but rather provide an overview, highlighting the application of the various methods and the merits and drawbacks of each. Chapter 10 will make use of this background in selecting appropriate methods to characterise rotary mechanical machines.

Two classes of identification methods will be explored; non-parametric and parametric, with reference to their corresponding models (the general form of which were discussed in chapter 3), and methods for selecting and validating models will also be considered. Perturbation signals will also be discussed, but since these can be specific to a particular experiment, they will also appear to some extent throughout the chapter. Finally the manipulation of the test data to extract the useful parts will be investigated.

5.1 The System Identification Objective

The aim of this chapter is to discuss methods that are applicable to the machine characterisation function of the test-rig, specifically concerning the creation of mathematical representations of any external machine connected to it. Chapter 8 describes a small selection of machines that were built specifically for the purpose of machine characterisation in the context of this project. All of these machines, and any others that are likely to be characterised on this test-rig have common properties; they are mostly linear (only a small amount of non-linearity if any), time-invariant, and heavily cyclic. Time-variant means that the machine's behaviour changes with time, and cyclic means that these behavioural changes occur cyclically, which in this application means that the behavioural changes are related to the shaft angle. The models of these machines are SISO models, having one input, torque, and one output, angle. This chapter therefore has a heavy bias towards this kind of machine and corresponding models and identification methods.

5.2 Background and Notation

Different kinds of system models and their properties are discussed in Chapter 3. Since the test-rig collects discrete-time data, the discrete-time models of sections 3.9 to 3.12 are particularly relevant to this chapter.

The method adopted by Matlab to represent SISO data is to use column vectors y and u to represent output and input signals, where the row number corresponds to the sample number and their dimensions are (number of samples) by (number of inputs or outputs). This output-input data is collectively represented by a matrix whose first column(s) is the output data, and next column(s) is the input data, $z = [y \quad u]$. The Matlab function `idplot` is used to graph the data, `idplot(z)`. The same method of representing the data will also be adopted in this text.

In section 3.10.1, a general transfer function model (equation 3.71) was described containing two polynomials (G and H) which are functions of a set of parameters, θ. Different ways of describing this equation in terms of θ were outlined, that is, different ways of parameterising the model set. This set of models will be used extensively in this chapter.

System identification uses statistical methods and probability theory extensively. It is beyond the scope of this text to derive or discuss these in detail, but some aspects are discussed as they arise and when the author believes it beneficial to do so.

5.3 Non-Parametric System Identification Methods

Non-parametric identification methods for system identification are used to obtain models which are curves or functions, and not necessarily parameterised by a finite number of parameters. This section discusses such methods.

5.3.1 Transient Analysis

With transient analysis, the input is taken as a step or an impulse, and the recorded output constitutes the model. Chapter 3 discusses the response of first and second-order systems to step and impulse inputs, and this section relates this to the case where the model is unknown. Consider a real system $G_0(q)$, with input $u(t)$, output $y(t)$, and disturbance $v(t)$ described by the function

$$y(t) = G(q)u(t) + v(t) \tag{5.1}$$

Impulse Response

An impulse input of amplitude α described by

$$u(t) = \begin{cases} \alpha, & t = 0 \\ 0, & t \neq 0 \end{cases} \tag{5.2}$$

applied to equation 5.1 will give an output

$$y(t) = \alpha g(t) + v(t) \tag{5.3}$$

If the noise level $v(t)$ is low, the coefficients $g_0(t)$ and errors $v(t)$ can be estimated by

$$\hat{g}(t) = \frac{y(t)}{\alpha} \qquad \text{and} \qquad error = \frac{v(t)}{\alpha} \tag{5.4 a, b}$$

Step Response

Similarly, a step input of amplitude α described by

$$u(t) = \begin{cases} \alpha, & t \geq 0 \\ 0, & t < 0 \end{cases} \tag{5.5}$$

applied to equation 5.1 will produce the output

$$y(t) = \alpha \sum_{k=1}^{t} g(k) + v(t) \tag{5.6}$$

Likewise the coefficients $g_0(t)$ and errors $v(t)$ can be estimated by

$$\hat{g}(t) = \frac{y(t) - y(t-1)}{\alpha} \qquad \text{and} \qquad error = \frac{v(t) - v(t-1)}{\alpha} \tag{5.7 a, b}$$

Equation 5.7 is likely to have large errors if the noise is significant since it is a differentiation approximation. It is suitable however to find characteristics such as delay time, static gain, or dominating time constants, and these methods are used to determine parameters for control purposes.

Transient analysis provides insight into cause and effect relationships, and time constants, damping factors, natural frequency and static gains are easily estimated.

Both of the above methods suffer from the fact that many physical systems do not allow inputs of this type, or of sufficient amplitude where the error is insignificant.

5.3.2 Correlation Analysis

The discrete-time form of model used in correlation analysis is

$$y(t) = \sum_{k=0}^{\infty} g(k)u(t-k) + v(t) \tag{5.11}$$

$g(k)$ is called the *weighting function*, since it describes the weight that the input at time $(t - k)$ has in the output at time t. The input is a stationary stochastic process independent of disturbances, such as white noise of zero mean and covariance σ^2. The following relation (called the Weiner-Hopf equation) [15] holds for the covariance functions

$$r_{yu}(\tau) = \sum_{k=0}^{\infty} g(k) r_u(\tau - k) \tag{5.12}$$

where $r_{yu}(\tau) = Ey(t+\tau)u(t)$ and $r_u(\tau) = Eu(t+\tau)u(t)$ (5.13 a, b)

(E is termed the *expectation operator*) and the covariance functions can be estimated from the data as

$$r_{yu}(\tau) = \frac{1}{N} \sum_{t=1-\min(\tau,0)}^{N-\max(\tau,0)} y(t+\tau)u(t) \qquad \tau = 0, \pm 1, \pm 2, \ldots \tag{5.14}$$

$$r_u(\tau) = \frac{1}{N} \sum_{t=1}^{N-\tau} u(t+\tau)u(t) \qquad \hat{r}_u(-\tau) = \hat{r}_u(\tau) \qquad \tau = 0, 1, 2, \ldots \tag{5.15}$$

An estimate $\hat{g}_0(k)$ of the weighting function $g_0(k)$ can be determined by solving

$$\hat{r}_{yu}(\tau) = \sum_{k=0}^{\infty} \hat{g}(k) \hat{r}_u(\tau - k) \tag{5.16}$$

The ease of which this can be solved depends on the input ([13] discusses this in detail). A simple way to estimate $\hat{g}_0(k)$ is (when the input is not "exactly white") to truncate 5.11 at n and treat it as an nth-order FIR model with the parametric least-squares method (section 5.4.2). Correlation analysis assumes that the input is uncorrelated with the disturbances, meaning that this method will not work properly when the data is collected from a system under output feedback.

5.3.3 Frequency Response Analysis

Frequency analysis uses a range of sinusoidal input frequencies, and the change in amplitude and phase of the sinusoidal output determines the frequency response of a system. The same model as equation 5.1 is applicable, and if the input is

$$u(t) = \alpha \cos(\omega t), \qquad t = 0, 1, 2, \ldots \tag{5.8}$$

and the system is stable, the output can be shown to be

$$y(t) = b \cos(\omega t + \varphi) + v(t) \tag{5.9}$$

where $b = \alpha \left| \hat{G}(j\omega) \right|$ and $\varphi = \arg[G(j\omega)]$ (5.10 a, b)

assuming that the system is initially at rest, so ignoring any transient effects. The estimate $\hat{G}_N(j\omega)$ is determined by finding the amplitude and phase for a number of frequencies in the range of interest.

Often there is noise and irregularities that make φ difficult to determine directly. It is then necessary to correlate the output with $\cos(\omega t)$ and $\sin(\omega t)$. This procedure is called *frequency analysis with the correlation method.*

5.3.4 Fourier Analysis

Section 3.6.2 briefly discussed Fourier transforms, some of which will be applied here. A system can be represented by $Y(s) = G(s)U(s)$. Transforming this to the time-domain (convolution integral) gives

$$y(t) = \int_0^t h(\tau) u(t - \tau) \quad d\tau \tag{5.11}$$

where h is called the impulse response. $G(s)$ and $h(t)$ are related by G being the Laplace transform of h

$$G(s) = \int_0^{\infty} h(t)e^{-st} \quad d\tau \tag{5.12}$$

If the input is chosen to be $u = \sin(\omega t)$ and the system is stable, and all transients have died away, the output can be given by

$$y(t) = |G(j\omega)| \sin(\omega t + \arg[G(j\omega)]) \tag{5.13}$$

The function $G(j\omega)$ is therefore the response the systems angular frequency ω. The following relationship then holds true

$$Y(\omega) = G(j\omega)U(\omega) \qquad \text{and} \qquad G(j\omega) = \frac{Y(\omega)}{U(\omega)} \tag{5.14 a, b}$$

If $y(t)$ and $u(t)$ are known over a finite interval $0 \le t \le S$, the two equations

$$Y_S(\omega) = \int_0^S y(t)e^{-j\omega t} \quad dt \qquad \text{and} \quad U_S(\omega) = \int_0^S u(t)e^{-j\omega t} \quad dt \tag{5.15}$$

can be used to form the *estimate*

$$\hat{\hat{G}}_S(j\omega) = \frac{Y_S(\omega)}{U_S(\omega)} \tag{5.16}$$

where $\hat{\hat{G}}_S$ is known as the *empirical transfer function estimate* (ETFE) of the G, since it is formed directly from data without any other model assumptions other that linearity. If the input is $u(t) = u_0 \cos(\omega t)$ the estimate $\hat{\hat{G}}_S(j\omega)$ can be shown to be the Fourier transform

$$\hat{\hat{G}}_S(j\omega) = \frac{2}{u_0 S}\left(\int_0^S y(t)\cos(\omega t).dt - j\int_0^S y(t)\sin(\omega t).dt \right) \tag{5.17}$$

For discrete time samples of $u(kT)$ and $y(kT)$ where (k = 1, …, N), the following approximations are made

$$Y_S(\omega) = T\sum_{k=1}^{N} y(kT).e^{-j\omega kT} \qquad \text{and} \qquad U_S(\omega) = T\sum_{k=1}^{N} u(kT).e^{-j\omega kT} \tag{5.18 a, b}$$

T is the sampling interval, and $S = N \times T$. If N is a power of 2 and $\omega = k \times 2\pi$, then the above process is known as the fast Fourier transform (FFT).

5.3.5 Spectral Analysis

Spectral analysis is a common method for analysis of signals and linear systems, and is an estimate of the frequency response of a system. It does not require any special input signals, but does not work for systems operating under feedback (i.e. so that the input and noise disturbances are uncorrelated). The mathematics behind this method is difficult, so a rather simplistic approach is adopted here.

The spectrum of a signal is its frequency content, and the notation for the spectrum of a signal $v(t)$ is $\Phi_v(\omega)$ found by

$$\Phi_v(\omega) = |V(\omega)|^2 \tag{5.19}$$

Spectral density is the measure of the signal's energy (or power), and between the frequencies ω_1 and ω_2 is found by

$$\int_{\omega_1}^{\omega_2} \Phi_v(\omega) \quad d\omega \tag{5.20}$$

There are a number of ways of defining spectrum definitions, but for energy and power spectra it is the sum of the square of the absolute value of its Fourier transform, for both continuous and discrete time signals.

The *cross spectra* between two signals u and y ($\Phi_{yu}(\omega)$) is defined as the product between the Fourier transform of y and the conjugate Fourier transform of u. Two

signals are said to be uncorrelated if their cross spectrum is zero. Cross spectra are mainly used for signals which belong to stochastic processes, and $\Phi_{yu}(\omega)$ is a complex number equal to the covariance between $Y(\omega)$ and $\overline{U(\omega)}$. Essentially this means that $y(t)$ will on average have the component of $u(t)$, but $\left|\Phi_{yu}(\omega)\right|$ times larger and $\arg\Phi_{yu}(\omega)$ radians phase delayed. For a system

$$y(t) = G(qt)u(t) + v(t) \tag{5.21}$$

it can be shown [25] that

$$\Phi_y(\omega) = \left|G(j\omega)\right|^2 \Phi_u(\omega) + \Phi_v(\omega) \tag{5.22}$$

and the cross spectra is given by

$$\Phi_{yu}(\omega) = G(j\omega)\Phi_u(\omega) \tag{5.23}$$

From this the frequency function can be estimated

$$\hat{G}_N(j\omega) = \frac{\Phi_{yu}^N(\omega)}{\Phi_u^N(\omega)} \tag{5.24}$$

The transfer function is obtained in the form of a bode plot (or other equivalent form).

5.3.6 Summary of Non-Parametric SI Methods

This section has outlined some simple techniques of transient and frequency response and how they can give some insight into the properties of linear systems. They are relatively easy to apply but give only moderately accurate models. The methods discussed are summarised below.

Transient Analysis

Transient analysis is easy to apply, and gives a step or impulse response as a model. It is very sensitive to noise and can only give a crude model.

Correlation Analysis

This is based on a white noise input and gives a weighting function as the resulting model. It is less sensitive to noise on the output.

Frequency Analysis

This method is based on sinusoidal inputs and the resulting model is a frequency response. It is usually represented as a bode plot or equivalent transfer function. The drawback with this method is that it tends to take longer.

Spectral Analysis

Spectral Analysis does not require any special input signals, but does not work for systems operating under feedback. The transfer function is usually represented as a bode plot or equivalent transfer function form.

Fourier Analysis

Fourier analysis is closely linked with spectral analysis but only works for periodic signals and can be rather crude in practice.

5.4 Parametric System Identification Methods

This section reviews the various models and identification methods that are applicable to the characterisation function of the test-rig. Predictor models for each type of model and the methods employed to reduce the prediction errors are described.

5.4.1 Parametric Models and Predictors

There are two general kinds of parameterised models, *white-box models* and *black-box models*.

5.4.1.1 White-box models

White-box (or custom) models are models that have been constructed to accurately represent a physical system and the parameters represent actual physical quantities, even though their values are not known. An example of a white-box model is the pair of equations 3.2 and 3.3, which describe the rotational mechanical system example. These equations fit the general form of equation 3.69, and in this case the one-step-ahead predicted output of $y(t)$ can be represented by

$$\hat{y}(t \mid \theta) = G(q, \theta)u(t) \tag{5.25}$$

Equation 5.25 is the predicted value of the output at time t according to the model, and is referred to as the *predictor* form. It is written as $\hat{y}(t \mid \theta)$ to emphasise that the output of the model will depend on the vector θ.

5.4.1.2 Black-box models

Often systems cannot be modelled based on physical insights, but it is possible to use standard models, which can handle a wide range of different system dynamics. Black-box models are families of models, such as those described in section 3.10. The parameters of these types of models are unlikely to have any physical interpretation, but can accurately describe the relationship between the inputs and outputs of the system.

5.4.1.3 Grey-box models

Often insufficient information is available to specify the complete structure of a white-box model, but assuming a black-box model would be a waste of information and result in a less accurate model. Cases between these two structures are called *grey-box* models since it is analogous to a box where only some of the structure can be seen.

5.4.1.4 Prediction

Taking the OE model described in section 3.10.1.4 as an example, it is possible to predict what the output $y(t)$ will be based on measurements $y(s)$ and $u(s)$, $s \leq t - 1$. The

noise term cannot be predicted since it is independent of previous values. The OE model for this (equation 3.91 fitted to equation 3.71) is

$$y(t) = G(q,\theta)u(t) + e(t) \tag{5.26}$$

This has the prediction

$$\hat{y}(t) = G(q,\theta)u(t) \tag{5.27}$$

The ARX model (equation 3.72) can be rearranged for $y(t)$

$$y(t) = -a_1 y(t-1) - \cdots - a_{n_a} y(t-n_a) + b_1 u(t-1) + \cdots + b_{n_b} u(t-n_b) + e(t) \tag{5.28}$$

and the prediction is simply this with the error term removed

$$y(t) = -a_1 y(t-1) - \cdots - a_{n_a} y(t-n_a) + b_1 u(t-1) + \cdots + b_{n_b} u(t-n_b) \tag{5.29}$$

The difference between 5.20 and 5.23 is that the OE model predictor is based entirely on the input, whereas the ARX model also uses past values of the output.

The general description of discrete-time transfer functions is given by equation 3.71a

$$y(t) = G(q,\theta)u(t) + H(q,\theta)e(t)$$

Dividing this by $H(q,\theta)$ gives

$$H^{-1}(q,\theta)y(t) = H^{-1}(q,\theta)G(q,\theta)u(t) + e(t) \tag{5.30}$$

Since the noise term is unknown, the prediction of $y(t)$ is simply obtained by deleting $e(t)$ and rearranging for $y(t)$ thus

$$\hat{y}(t \mid \theta) = [1 - H^{-1}(q,\theta)]y(t) + H^{-1}(q,\theta)G(q,\theta)u(t) \tag{5.31}$$

This is a therefore a general predictor expression for the next value of output based on past inputs and outputs for models of the type described by equation 3.71.

5.4.1.5 Minimising the Prediction Errors

$\hat{y}(t|\theta)$ is a prediction of $y(t)$ at time $(t\text{-}1)$ irrespective of the type of model it corresponds to. How good this prediction is can be measured by calculating the prediction error

$$\varepsilon(t,\theta) = y(t) - \hat{y}(t|\theta) \tag{5.32}$$

Over a time period containing N samples (t = 1, …, N), it is possible to evaluate how well the model with the parameter θ represents the system by evaluating the sum of the squares of the errors over this period, called the quadratic *criterion function* or *loss function* (whose parameters are the parameter value θ, and the set of input-output data $Z^N = \{u(0), y(0), \ldots, u(N), y(N)\}$)

$$V_N(\theta, Z^N) = \frac{1}{N}\sum_{t=1}^{N}\varepsilon^2(t,\theta) \tag{5.33}$$

The problem is to choose the value of θ that minimises

$$\hat{\theta}_N = \arg\min_{\theta} V_N(\theta, Z^N) \tag{5.34}$$

(where 'arg min' denotes the minimising argument). Several variants of equation 5.33 can be used, and in general any arbitrary positive, scalar valued function $\ell(e)$ can be used as a measure of error, and the problem is then to minimise

$$V_N(\theta, Z^N) = \frac{1}{N}\sum_{t=1}^{N}\ell(\varepsilon(t,\theta)) \tag{5.35}$$

Methods that minimise this loss function are called *prediction error* (PE) methods. There are many established methods that satisfy the criteria of equation 5.34 as a good

choice for parameter estimation. For example, equations 5.34 and 5.35 give the *maximum likelihood* (ML) estimate of θ, if $\ell(\cdot)$ is chosen as

$$\ell(\varepsilon) = \log f_e(\varepsilon) \tag{5.36}$$

where $f_e(\varepsilon)$ is the probability density function (PDF) of the noise $e(t)$. This is discussed in more detail in section 5.4.3.

The prediction error sequence is frequently passed through a stable linear filter $L(q)$

$$\varepsilon_F(t,\theta) = L(q)\varepsilon(t,\theta) \qquad 1 \le t \le N \tag{5.37}$$

which gives extra flexibility in dealing with effects such as high frequency disturbances not essential to the modelling problem, or slow drift terms and offsets. L thus acts as a *frequency weighting*. For now it will be assumed that $L(q) \equiv 1$ as the pre-processing of the data is discussed in section 5.7.

5.4.2 Linear-Regression and the Linear Least-Squares Method

The general predictor described in the previous section could be written as the *linear regression*

$$\hat{y}(t \mid \theta) = \varphi^T(t)\theta + \mu(t) \tag{5.38}$$

where θ is the *parameter vector* (described in section 3.10.1) and φ is the *regression vector*, where for the ARX structure is

$$\varphi(t) = \left[-y(t-1) \quad -y(t-2) \quad \ldots \quad -y(t-n_a) \quad u(t-1) \quad \ldots \quad u(t-n_b)\right]^T \tag{5.39}$$

$\mu(t)$ is a data dependant vector, and can be ignored for this section. From equation 5.38 the prediction error can be given by

$$\varepsilon(t \mid \theta) = y(t) - \varphi^T(t)\theta \tag{5.40}$$

and the criterion function resulting from equations 5.35 and 5.37with L(q) = 1 and $\ell(\varepsilon) = \frac{1}{2}\varepsilon^2$, is the *least-squares (LS) criterion* for the linear regression (equation 5.38):

$$V_N(\theta, Z^N) = \frac{1}{N}\sum_{t=1}^{N} \tfrac{1}{2}\left[y(t) - \varphi^T(t)\theta\right]^2 \tag{5.41}$$

Since this is a linear parameterisation and a quadratic function of θ, it can be minimised analytically, providing the inverse exists, which gives the *least squares estimate* (LSE)

$$\theta_N^{LS} = \arg\min V_N(\theta, Z^N) = \left[\frac{1}{N}\sum_{t=1}^{N}\varphi(t)\varphi^T(t)\right]^{-1}\frac{1}{N}\sum_{t=1}^{N}\varphi(t)y(t) \tag{5.42}$$

5.4.2.1 Weighted Least-Squares

It may be that measurements at different time instants are considered to be of varying reliability. This may because noise corruption changes or that some measurements are less representative of a system's characteristics. In such cases the choice of $\ell(\cdot)$ can be time varying

$$V_N(\theta, Z^N) = \frac{1}{N}\sum_{t=1}^{N}\ell(\varepsilon(t,\theta),\theta,t) \tag{5.43}$$

so that less reliable measurements can be given less weight. An explicit weighting function $\beta(N,t)$ can be applied, so the criterion function is

$$V_N(\theta, Z^N) = \frac{1}{N}\sum_{t=1}^{N}\beta(N,t)\ell(\varepsilon(t,\theta),\theta) \tag{5.44}$$

so equation 5.41 becomes

$$V_N(\theta, Z^N) = \frac{1}{N}\sum_{t=1}^{N}\beta(N,t)\left[y(t) - \varphi^T(t)\theta\right] \tag{5.45}$$

Introducing this weighting function into equation 5.42 yields

$$\theta_N^{LS} = \arg\min V_N(\theta, Z^N) = \left[\frac{1}{N}\sum_{t=1}^{N}\beta(N,t)\varphi(t)\varphi^T(t)\right]^{-1}\frac{1}{N}\sum_{t=1}^{N}\beta(N,t)\varphi(t)y(t) \tag{5.46}$$

An advantage of the least squares method is that the global minimum of equation 5.41 can be found efficiently and unambiguously (only the one global minima exists). Its main disadvantage is that if the equation error is not white noise then the parameters will not converge to the true values of the parameters. Further modelling may be incorporated into the equation error, but this typically leaves the LS environment.

5.4.3 Maximum-Likelihood Method

The system identification problem is one of extracting information from unreliable or stochastic data, and representing this information with parameters obtained from an estimator. Many estimator functions are possible, and a particular one that maximises the probability of an observed event is the *maximum likelihood estimator* (MLE). This estimator assumes that the noise in the model is Gaussian and was briefly discussed in section 5.4.1.5. The estimator can be written

$$\hat{\theta}_{ML} = (y^N) = \arg\max_{\theta} f_y(\theta; y^N) \tag{5.47}$$

The definition of the *probability density function* (PDF) $f_e(x) = f_e(x_1, \ldots, x_n)$ of a random vector e is said to have *Gaussian* or *normal* distribution if

$$f_e(x) = \frac{1}{(2\pi)^{n/2}}\frac{1}{(\det P)^{1/2}}\exp\left[-\tfrac{1}{2}(x-m)^T P^{-1}(x-m)\right] \tag{5.48}$$

where m is the mean and the covariance matrix is P containing λ_i, and det P is the determinant of the matrix P.

To calculate the maximum likelihood estimator, the joint PDF for the observations is firstly calculated. The PDF for $y(i)$ is

$$\frac{1}{\sqrt{2\pi\lambda_i}}\exp\left[-\frac{(x_i-\theta)^2}{2\lambda_i}\right] \tag{5.49}$$

and since all the $y(i)$ are independent, the PDF of y^N (y^N = y(1), y(2), …, y(N)), the *likelihood function* can be given by

$$f_y(\theta;x^N)=\prod_{i=1}^{N}\frac{1}{\sqrt{2\pi\lambda_i}}\exp\left[-\frac{(x_i-\theta)^2}{2\lambda_i}\right] \tag{5.50}$$

Maximising the likelihood function is the same as maximising its logarithm, thus

$$\hat{\theta}_N^{ML}(y^N)=\arg\max_{\theta}\log f_y(\theta;x^N)$$

$$\hat{\theta}_N^{ML}(y^N)=\arg\max_{\theta}\left[-\frac{n}{2}\log 2\pi-\sum_{i=1}^{N}\frac{1}{2}\log\lambda_i-\sum_{i=1}^{N}\frac{(y(i)-\theta)^2}{\lambda_i}\right] \tag{5.51}$$

from which it can be found

$$\hat{\theta}_N^{ML}(y^N)=\frac{\sum_{i=1}^{N}\frac{y(i)}{\lambda_i}}{\sum_{i=1}^{N}\frac{1}{\lambda_i}} \tag{5.52}$$

5.4.4 Instrumental Variables Method

Ideally the prediction error for a good model is independent of past data, i.e.

$$\varepsilon(t,\theta)=y(t)-\hat{y}(t\,|\,\theta) \tag{5.53}$$

is independent of the data set Z^{t-1} (Z^N = $\{u(0), y(0), \ldots, u(N), y(N)\}$). If this is not the case and $\varepsilon(t,\theta)$ is correlated with Z^{t-1} then the predictor is not ideal. This means that a

good model produces prediction errors that are independent of past data. To check this would amount to testing if all non-linear transformations of $\varepsilon(t,\theta)$ are uncorrelated with all possible functions of Z^{t-1}, which is not feasible in practice. Instead a finite-dimensional vector sequence $\zeta(t)$ is derived from Z^{t-1}, and a transformation of $\varepsilon(t,\theta)$ is defined which must be uncorrelated with this sequence,

$$\frac{1}{N}\sum_{t=1}^{N}\zeta(t)\alpha(\varepsilon(t,\theta))=0 \tag{5.54}$$

$\alpha(\varepsilon)$ is the chosen transformation of $\varepsilon(t,\theta)$, and the value of θ which satisfies this equation would be the best estimate $\hat{\theta}^N$.

To allow extra freedom in dealing with non-momentary properties of the prediction errors, the data is filtered to remove unwanted properties (equation 5.37).

$$\varepsilon_F(t,\theta)=L(q)\varepsilon(t,\theta)$$

A sequence of correlation vectors constructed from past data (and possibly θ) is chosen

$$\zeta(t,\theta)=\zeta(t,Z^{t-1},\theta) \tag{5.55}$$

and also a function $\alpha(\varepsilon)$ which is discussed below. An estimator can be calculated

$$\hat{\theta}_N=\mathrm{sol}\left[f_N(\theta,Z^N)=0\right] \tag{5.56}$$

which means 'the solution to the equation $f_N(\theta,Z^N)=0$', where

$$f_N(\theta,Z^N)=\frac{1}{N}\sum_{t=1}^{N}\zeta(t,\theta)\alpha(\varepsilon_F(t,\theta)) \tag{5.57}$$

The estimate is taken to be the value that minimises this function

$$\hat{\theta}_N = \arg\min \left| f_N(\theta, Z^N) \right| \tag{5.58}$$

which can be written

$$\hat{\theta}_N = \arg\min \left| \frac{1}{N} \sum_{t=1}^{N} \zeta(t,\theta)\alpha(\varepsilon_F(t,\theta)) \right| \tag{5.59}$$

The above method is a conceptual method, the implementation of which depends on the model structures and the choice of ζ, The best known is perhaps the application of this method to a linear regression, which is called the *instrumental-variable* method (IV).

It is worth noting, that if $L(q) = 1$ and $\zeta(t,\theta) = \varphi(t)$ then equation 5.59 describes the LSE that corresponds to the LSE of equation 5.42. The linear regression model is described by

$$y(t \mid \theta) = \varphi^T(t)\theta \tag{5.60}$$

It was pointed out in section 5.4.2.1 that the LSE $\hat{\theta}_N$ will typically not tend to the actual parameters θ_0 because the equation error is not perfectly 'white' due to correlation between the noise $v(t)$ and $\varphi(t)$. A general correlation vector $\zeta(t)$ is chosen, the elements of which are called the *instruments*, or *instrumental variables*. This gives

$$\hat{\theta}_N^{IV} = \text{sol}\left\{ \frac{1}{N} \sum_{t=1}^{N} \zeta(t)\left[y(t) - \varphi^T(t)\theta \right] = 0 \right\} \tag{5.61}$$

(which incorporates the regression model 5.60) and can also be written

$$\hat{\theta}_N^{IV} = \left[\frac{1}{N} \sum_{t=1}^{N} \zeta(t)\varphi^T(t) \right]^{-1} \frac{1}{N} \sum_{t=1}^{N} \zeta(t) y(t) \tag{5.62}$$

provided that the inverse exists. It can be seen from equation 5.61 that for $\hat{\theta}_N$ to tend to θ_0 for large N,

$$\frac{1}{N}\sum_{t=1}^{N}\zeta(t)v(t) \tag{5.63}$$

should tend to zero. The instrumental variable $\zeta(t)$ must therefore be correlated with the regression variables but uncorrelated with the noise.

5.4.4.1 The Choices of Instruments

One possibility for choosing the instruments for a SISO system is to assume the model (equation 5.60) is an ARX model

$$A(q)y(t) = B(q)u(t) + v(t) \tag{5.64}$$

represented by equation 3.72

$$y(t) + a_1 y(t-1) + \cdots + a_{n_a} y(t-n_a) = b_1 u(t-1) + \cdots + b_{n_b} u(t-n_b) + v(t)$$

and choose the instruments similarly to this model such that the they are correlated with the regression variables but uncorrelated with the noise:

$$\zeta(t) = K(q)\left[-x(t-1)\ldots -x(t-n_a) \quad u(t-1)\ldots u(t-n_b)\right]^T \tag{5.65}$$

where K is a linear filter and $x(t)$ is generated from the input through a linear system

$$N(q)x(t) = M(q)u(t) \tag{5.66}$$

Most instruments are generated in this way, and because they are a function of past inputs by linear filtering can be written conceptually as

$$\zeta(t) = \zeta(t, u^{t-1}) \tag{5.67}$$

The input must be generated open-loop so that it is uncorrelated with the noise $v(t)$ in the system, and $\zeta(t)$ will therefore be correlated with the regression variables and not the noise since it is generated from the input sequence.

One choice of instruments is to apply the LS method to the ARX equation (3.72) and then use this LS model to determine N and M, letting $K = 1$. This method is permissible for open-loop systems but a different approach is required for closed-loop systems.

5.4.5 Recursive Identification Methods

All the identification methods discussed in this chapter so far have been *off-line* or *batch* methods, in which the recorded data is used simultaneously to find the parameter estimates. Identification methods where the parameter estimates are computed recursively in time while the system is in operation are called *recursive* or *on-line identification methods*, since the measured input-output data is processed recursively (sequentially) as it becomes available. The model is based on observations up to the current time and the need for this typically arises when a model is required to make a decision about the system 'on-the-fly', often termed *adaptive*, as in adaptive control, adaptive filtering etc…

When the test-rig is performing tests on an external machine, the test-rig will apply various stimuli and collect data, which will subsequently be processed off-line. Decisions may then be made to require further tests, but the data will not be used recursively. These types of identification methods will therefore not be considered further.

5.4.6 Summary of Parametric SI Methods

The least squares method is easy to use and can be applied to the identification of dynamic models. The estimates obtained are consistent but only under restrictive conditions. I.e. if the equation error is not white noise then the parameters will not converge to the true values of the parameters. Two different ways of modifying the LS method were given where consistent estimates can be obtained under less restrictive conditions. Firstly minimisation of the prediction error leads to the prediction error

methods, and secondly modification of the normal equations associated with the least squares method which leads to the class of instrumental variables methods.

The prediction error methods (PEM) are a generalisation of the least squares method, where the parameter estimate is determined as the minimising vector of a suitable function of the sample covariance matrix of the prediction errors. A special case of the PEM for Gaussian distributed disturbances is the maximum likelihood method. Under certain assumptions the PEM estimates are consistent, and under a Gaussian assumption the estimates are statistically efficient (i.e. have minimum possible variance).

The instrumental variables methods assume that the system is causal and asymptotically stable, and the input and disturbance are independent (the system is open loop), and belongs to the set of models considered (ARX). Under these (mild) assumptions the parameter estimates are consistent and Gaussian distributed, and the covariance matrix of the parameter estimates can be optimised by appropriate choices of pre-filter and the IV matrix.

5.5 Experiment Design

The goal of the identification procedure is to obtain a good model with a reasonable amount of work. The practical experiment has already been defined by virtue of the project goals and is discussed in detail in chapter 9. The remaining issues to be addressed are:

1) the choice of perturbation signals,
2) the choice of sampling interval and pre-sampling filters,
3) applying these signals and practical considerations,
4) pre-processing the collected data,
5) choice of model and identification method to apply,
6) model validation.

As the beginning of the chapter outlined, the above processes of 3 - 6 are iterative and a final model will only be obtained after some trial and error. The remainder of this chapter discusses these issues from a general viewpoint.

5.5.1 Experimental Variables

The general model of the external machine in this project is a dynamic SISO model with torque input and an angular output. Two other variables are measurable, motor current and angular velocity (although this is estimated), but these are not intended to be used for machine characterisation.

5.6 Perturbation Signals and Sampling Interval

The choice of input signal has a substantial influence on the observed data. The input signal determines the operating point of the system and which parts and modes of the system are exited during the experiment. Certain identification methods require a special type of input, particularly for non-parametric identification methods. For frequency analysis the input must be a sinusoid, for transient analysis a step or impulse, and for correlation analysis a white noise or pseudo-random sequence. For other types of identification methods it is only required that the input is persistently exciting (pe) of a certain order, i.e. that it contains sufficiently many distinct frequencies. To identify a system of nth-order the input typically needs to be of order $2n$. The amplitude of the input is also of importance.

The machines being characterised will have operational limits, and if these are exceeded they will exhibit non-linear behaviour – that is, the machines can only be assumed (nearly) linear within their operating region. With this in mind it is usually beneficial to use large inputs as this increases the signal-to-noise ratio and the disturbances will play a smaller role. A simple rule is that the experimental condition should resemble the conditions under which the models will be used in the future.

Section 3.14 briefly discussed some forcing functions and the dynamic behaviour of various types of systems to these inputs. This section will extend this theory for the use in the context of system identification and investigate the use of more complex forcing functions.

5.6.1 White (Gaussian) Noise

A true white noise has a mean value of zero, and has normally distributed amplitude with probability symmetrical about zero. It is a good test signal particularly for correlation testing, but in practice only white noise of a limited bandwidth is possible.

5.6.2 Pseudo-random Binary Sequences

An important type of periodic signal for identification is the pseudo-random binary sequence (PRBS) which has two levels, and can only switch from one level to the other at discrete points, $t = 0, \Delta t, 2\Delta t, \ldots$. When the signal changes state is pre-determined so that the PRBS is deterministic and experiments are repeatable (in contrast to the discrete-interval random binary signal). The PRBS is periodic with period $T = N\Delta t$, where N is the length of the sequence and an odd integer, and in any one period there are $\frac{1}{2}(N+1)$ intervals when the signal is at one level and $\frac{1}{2}(N-1)$ intervals when the signal is at the other. The most commonly used signals are based on maximum-length sequences (m-sequences) and are easily generated using feedback shift register circuits.

The maximum length of binary m-sequences is $N = 2^n - 1$, where n is a positive integer greater than zero. They can be generated using an n-stage feedback shift register with a feedback to the first register consisting of the modulo-2 sum (exclusive-OR) of the logic value of the last stage and one or more of the other stages. The reason that the upper bound of 2^n cannot be obtained is that the occurrence of an all zero state must be prevented. If this state did occur the state vector would remain zero for all future iterations. The register must therefore be started with any number other than all zeros. For all binary m-sequences each binary number (except for all zeros) occurs exactly once. Figure 5.1 shows a shift register circuit for generating a PRBS based on an m-sequence of length $2^n - 1$.

It is important to note that not all combinations of bits for the feedback connection work, and tables of ones that do work are available [13]. The autocorrelation properties of a PRBS resemble those of white noise, and it is also easy to delay.

There are three levels to select for use as an input signal, the period, the clock period and the amplitude.

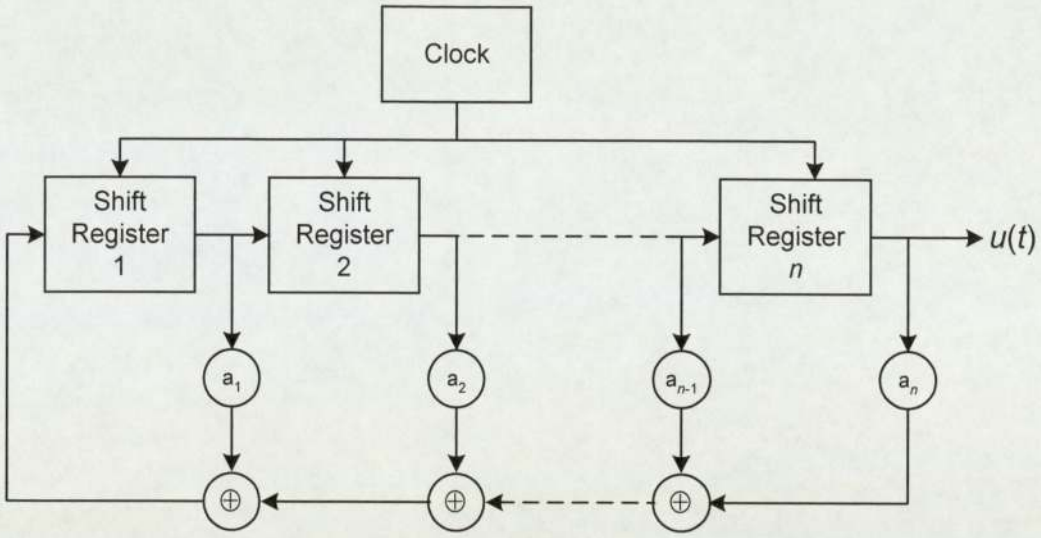


Figure 5.1 Shift register circuit for generating a PRBS based on a $(2^n - 1)$-digit m-sequence

It can be shown [15] that the PRBS has similar properties to a white noise but the spectral densities are different since the PRBS is periodic. A C function to generate a PRBS signal is shown in appendix B.2.

5.6.3 Sum of Sinusoids

This class of input signal is given by

$$u(t) = \sum_{j=1}^{m} a_j \sin(\omega_j t + \varphi_j) \tag{5.68}$$

where ωn are distinct angular frequencies of amplitudes a_n and φn are their corresponding phases.

5.6.4 Chirp Signals and Swept Sinusoids

A chirp signal is a sinusoid with a frequency that changes over a frequency band (Ω) $\omega_1 \leq \omega \leq \omega_2$ and time period $0 \leq t \leq M$ such that

$$u(t) = A\cos\left(\frac{\omega_1 t + (\omega_2 - \omega_1)t^2}{2M}\right) \tag{5.69}$$

The instantaneous frequency can be found by differentiating the cosine argument w.r.t. time

$$\omega_i = \omega_1 + \frac{t}{M}(\omega_2 - \omega_1) \tag{5.70}$$

Due to the sliding frequency there will also be some power contributions outside the band Ω. Functions to generate swept sine perturbation signals are shown in appendices A.5 and B.3.

5.6.5 Periodic Signals

The PRBS and Sum of Sinusoids are inherently periodic, and the Swept Sinusoid can be made periodic by simple repetition. When creating periodic signals the following must be ensured:

- The PRBS signal must be generated over one full period ($2^n - 1$) and then repeated.
- To create a sum of sinusoids of period M, the frequencies must be chosen by $\omega_\ell = 2\pi\ell / M$ where $\ell = 0, 1, \ldots, (M - 1)$.
- To make the chirp signal periodic, ω_1 and ω_2 must chosen by $2\pi k_n / M$ for two integers k_1 and k_2. The signal generated from equation 5.69 is then repeated.

5.6.5.1 Properties of Periodic Signals

A signal of period M can have at most a maximum of M discrete frequencies in its spectrum and is persistently exciting of at most order M. It is sometimes beneficial to use only one period of data for building non-parametric models as this increases the signal to noise ratio. Some methods have a performance threshold for finite samples and poor signal-to-noise ratios, and there may be an accuracy benefit from averaging the measurement over periods.

Using a periodic signal allows estimates of the noise in a system to be made. Once the transients have died away, the differences in the output response for identical input periods must be due to the noise. This can be used in the model validation process to distinguish between model errors and noise. The output can be represented by

$$y(t) = y_u(t) + v(t) \tag{5.71}$$

where $y_u(t)$ is the noise free part of the output. It is periodic and the estimate of this over K periods for each period $1 \leq t \leq M$

$$\hat{y}_u(t) = \frac{1}{k}\sum_{k=0}^{K-1} y(t + kM) \tag{5.72}$$

The noise estimate is therefore $\hat{v}(t) = y(t) - \hat{y}_u(t)$ from which noise levels and colours can be estimated. The noise variance is estimated by

$$\hat{\lambda}_v = \frac{1}{(K-1)M}\sum_{t=1}^{KM} \hat{v}^2(t) \tag{5.73}$$

5.6.6 Choice of Sampling Rate and Pre-sampling Filters

In sampled data system there will be an inevitable loss of information and it is important to select the sampling instances to minimise these losses. It is far easier to work with data that is obtained through equidistant sampling instants of sampling interval T, and this is assumed in this text.

5.6.6.1 Aliasing

Suppose a signal is sampled with a sampling interval T, so that $s_k = s(kT)$ ($k = 1, 2, \ldots$), and the *sampling frequency* is written $\omega_s = 2\pi/T$. Shannon's sampling theorem states that for a continuous signal to be reconstructed from a set of equally spaced samples, the signal must be sampled at a frequency which is greater than twice the highest frequency component present in the signal. This means that a sampled sinusoid with frequency higher than the *Nyquist frequency*, $\omega_N = \omega_s/2$, cannot be distinguished from

one in the interval $-\omega_N$ to ω_N, and part of the signal spectrum that corresponds to frequencies higher than ω_N will be interpreted as contributions from lower frequencies. This is called *aliasing* because the frequencies appear under assumed names. It also means that the spectrum of the sampled signal will be a superposition of different parts of the original spectrum, which is called *frequency folding*.

5.6.6.2 Anti-aliasing / Pre-sampling Filters

The information about the frequencies higher than the Nyquist frequency will be lost due to sampling, but it is important to ensure the folding effect does distort the frequencies below the Nyquist frequency. This is achieved by pre-sampling the signal with a filter known as a *pre-sampling* or *anti-aliasing* filter, and is placed before the sampling.

Signals frequently consist of a useful part and a noise contribution. The noise contribution is usually broadband, so by choosing a sampling frequency such that the useful part of the signal is below ω_N, the effect of filtering the signal is to remove the high-frequency noise contributions. This of course changes the properties of the noise, but is necessary to prevent noise effects from the higher frequency region folding into the region $-\omega_N$ to ω_N.

5.6.6.3 Sampling Rate

For system identification data-acquisition, it is usual to sample sufficiently quickly so that the process is well damped above the Nyquist frequency, and high-frequency components that originate from the input are insignificant. From an information theoretic point of view it is beneficial to sample as fast as possible since slower sampling leads to data sets that are subsets of the maximal one, and hence less informative. There are two aspects that prevent sampling as fast as technically possible:

1) Building models with very small sampling interval compared to the natural time constants is numerically sensitive due to the effects of round-off errors, and
2) The model fit may be concentrated to the high-frequency band (this is discussed in the next section).

A sampling rate much slower than the interesting time constants of the system would provide little information about the system's dynamics, and a fast sampling rate would not allow for much noise reduction. A good choice of the sampling rate is therefore a trade-off between noise reduction and relevance for the dynamics. In practice, where computer speed is not a problem, the sampled frequency is usually made five to ten times higher than the highest frequency component (or the fastest system pole) in the sampling input. It is often advantageous to sample at a higher rate than this, as the choice of T can be made later by digitally pre-filtering and reducing the original data record.

5.6.7 Summary

To summarise, the following principles are important:

- The experimental condition should resemble the conditions under which the models will be used
- The system is made identifiable by using a persistently exiting input and not allowing too simple feedback mechanisms
- Periodic inputs can be particularly advantageous for single-input systems (where an integer number of periods should be applied)
- A sampling rate of ten times the guessed bandwidth (or fastest system pole) in the system is usually a good choice

5.7 Pre-processing the Data

When the data has been collected it is unlikely to be adequate for immediate use in identification algorithms. There are several areas where the data is deficient and will cause problems unless proper action is taken.

5.7.1 Offsets, Drifts and De-trending

Offsets and low-frequency disturbances such as drift are not uncommon in data. They can often occur as external disturbances, or may be characteristics of the system which are not required for the identification problem (e.g. non-linear friction). This will force

the model to waste some parameters correcting the levels, so it is desirable to remove these unwanted artefacts either by data pre-treatment or by including them in the noise model.

The easiest method of removing offsets is to subtract the mean levels from the input and output sequences before the estimation:

$$y(t) = y^m(t) - \bar{y} \qquad \text{where} \qquad \bar{y} = \frac{1}{N}\sum_{t=1}^{N} y^m(t) \tag{5.74a, b}$$

$$u(t) = u^m(t) - \bar{u} \qquad \text{where} \qquad \bar{u} = \frac{1}{N}\sum_{t=1}^{N} u^m(t) \tag{5.75a, b}$$

Extending the noise model is rather more complicated and is more suited to on-line identification where the mean cannot be explicitly calculated and the use of a high-pass filter is employed. This has the same effect as removing offsets, and slow drifts also.

It is particularly important to remove offsets when OE models are used because the discrepancy in levels will become the dominating factor and the system dynamics will be less influential. High pass filtering of the data

5.7.2 Outliners

Real data is prone to bad disturbances, for example sensor malfunction, conversion failure, or large disturbances such as spurious electrical noise. It is important that these outliners do not affect the models too any significant extent. Bad values such as these are usually easy to detect in a residual plot (residuals are the 'leftovers' from the modelling process, i.e. the part of the data that the model could not reproduce). There are a few ways of dealing with outliners (or 'repairing' the damaged data set). They can either be manually removed, filtered out (smoothed) or treated as missing data. Treating them as missing data involves estimating the missing values using PE methods and will not be discussed further in this text.

5.8 SI Method Selection, Model Selection and Validation

There are no definite rules that can be applied to determine the 'best model structure' or 'best identification method'. For real data there are no 'best' structures or methods. Different models can be obtained from the same data, and they will represent the actual system either more or less accurately depending on the criteria used to validate the model and its application. Experiment performance and data acquisition will invariably take considerable effort and time, and it is worth spending time estimating different models and using different methods.

5.8.1 Choice of SI Method

It is clear that for the purpose of this text the process will be identified off-line. When choosing an SI method the purpose of the identification is important since it specifies the type of model and its accuracy. The following list shows SI methods arranged in order of general ascending accuracy and computational complexity:

- Transient analysis
- Frequency analysis
- Least squares method
- Instrumental variables method
- Prediction error method (including ML)

Of course the data and expected model govern the actual application of these methods, and in practice other factors will influence the choice such as previous experience and available software.

Applicability of Methods

Whether the system was operating in closed-loop when the data was collected determines which methods can be used. All of the prediction methods generally work equally well for data from closed-loop systems. The OE and BJ models normally give a correct description of the dynamics G even if the noise dynamics H are inaccurate. This is not the case for closed-loop systems. The spectral analysis method and instrumental variable techniques give unreliable results when used on closed-loop data and their use should be avoided under such conditions.

The prediction-error approach (including LS & ML methods) is applicable to all model structures including white-box models and black-box-parameterised, and can be applied to open and closed-loop data. The minimisation method is the same for all variations on the theme, and only the predictor is calculated differently. The correlation method is also generally applicable but is only used for linear black-box models. The IV method is a variant of this and is for the ARX model.

5.8.2 Model Structure Selection

The choice of model structure is perhaps the most important aspect for SI, and is based on knowledge of the identification procedures and physical insight into the system being identified. Chapter 3 described model structures applicable to the SI problem, and this chapter has described some methods to identify the parameters in those models. Once a model has been chosen it needs to be validated to determine its fitness, and this is discussed in section 5.8.3.

Selecting the type of model involves the choice of linear or non-linear, white-box or black-box etc... and selecting the size of the model set involves selecting the order of the model or the degrees of the polynomials involved. A priori knowledge of the system is invaluable to the identification problem and it is this knowledge which forms the starting point of the exercise. The models appropriate to the test-rig characterisation function will generally be assumed linear and black-box-parameterised. A rule of thumb is to try simple models first and try more sophisticated models if the simpler ones do not pass the validation tests.

Applicability of Various Black-Box-Parameterised Models

Referring to table 3.1 the following model structures are discussed:

- The ARX model $A(q)y(t) = B(q)u(t) + e(t)$ is the easiest to estimate since the corresponding estimation problem is of a linear regression type. The ARX model is a good starting point for identification. The main disadvantage is that the disturbance model $H(q,\theta) = 1/A(q)$ involves the system poles and the system dynamics may be incorrectly estimated because the A polynomial also has to

describe the disturbance properties. Higher orders of model may therefore be required. If the signal-to-noise is good, this problem is less pronounced.

- The ARMAX model $A(q)y(t) = B(q)u(t) + C(q)e(t)$ has more flexibility in handling the disturbance modelling since an extra polynomial C.

- The OE model $y(t) = \frac{B(q)}{F(q)}u(t) + e(t)$ has the advantage that no parameters are used to describe the noise model and the system dynamics are described separately. The system needs to operate in open-loop but a description of the transfer function can be obtained regardless of the nature of the disturbance. Minimisation of the criterion function is more difficult than in the ARMAX case.

- The BJ model $y(t) = \frac{B(q)}{F(q)}u(t) + \frac{C(q)}{D(q)}e(t)$ incorporates polynomials to describe all parts of the system and the disturbance properties are modelled separately from the system dynamics.

The ARX and ARMAX models have common dynamics for the noise and the input and therefore are suitable when dominating disturbances enter early in the system, for example the input. In contrast the BJ model is convenient when the disturbance occur late in the system, for example as measurement noise in the output.

Model Order Estimation

The following procedure is useful to estimate the required order of a black-box-parameterised model:

1) Sometimes the dynamics from u to y contains a delay of n_k samples, and some leading B coefficients are therefore zero. The delay of a linear system can be estimated using one of the non-parametric SI methods, tested using an ARX model, and chosen for the model that gives the best performance.

2) Test many ARX models of different orders with this delay and pick the model that gives the best performance.

3) The model may be of the wrong order, and is likely to be too high an order since the poles of an ARX model also describe the noise properties. A residual analysis test will show this; a rule of thumb is that a slowly varying cross correlation function outside of the confidence region is an indication of too few poles, while sharper peaks indicate too few zeros or wrong delays. If there is pole-zero cancellation on a pole-zero plot, the extra poles are likely to be there to describe the noise. The remaining poles and zeros give a good indication of the necessary order of the dynamic model. Once this has been ascertained, ARMAX, OE or BJ models with the same order of G can be fitted, with first or second order models for the noise characteristics H.

Bias and Variance

The model errors that arise as a result of noise influence on the measurements is called *variance errors*. Using longer measurement sequences can typically reduce variance errors. Errors that prevent the model from adequately describing the system, even from noise-free data, because the model is simply not capable of describing the system, are called *bias errors*. Bias errors are apparent when the model is used with data collected under different conditions. A good model therefore is one that has both small variance and bias error.

5.8.3 Model Comparison

When a number of models of various structures have been made, they need to be compared. Prediction-error variances are best evaluated when they are given *new* data, i.e. data other than that used in the model estimation. This is often called *cross-validation*, and it is a better test than using 'old' data. A larger model will always give a lower value of the criteria function for a particular set of data since it has been minimised over more parameters. The reason for this is that the extra and unnecessary parameters are used to fit the disturbances specific to that data. This is called *over-fit* and is undesirable because it is a poorer model of the system. The usual procedure is to use one set of data for the estimation, and the other set for its evaluation. A number of methods are used to locate the transition from relevant model fit to over-fit. They all follow this basic form

$$\min_{d,\theta} f(d,N)\sum_{t=1}^{N}\varepsilon^2(t,\theta) \tag{5.76}$$

where N is the number of data and d is θ's dimension (the number of estimated parameters). The function $f(d,N)$ increases with d and decreases with N, so the model selected will represent a balance between model fit and the number of parameters. Some common choices of $f(d,N)$ are:

Akaike's information criterion (AIC):

$$\min_{d,\theta} f\left(1+\frac{2d}{N}\right)\sum_{t=1}^{N}\varepsilon^2(t,\theta) \tag{5.77}$$

Final Prediction Error (FPE):

$$\min_{d,\theta} f\left(\frac{1+d/N}{1-d/N}\frac{1}{N}\right)\sum_{t=1}^{N}\varepsilon^2(t,\theta) \tag{5.78}$$

Rissanen's minimal description length (MDL):

$$\min_{d,\theta} f\left(1+\frac{2d}{N}\cdot\log N\right)\sum_{t=1}^{N}\varepsilon^2(t,\theta) \tag{5.79}$$

The FPE is a statistical estimate of the PE variance from using a new set of data, and the MDL aims at minimising the size of parameter storage.

Besides comparing the model prediction error variances, the models can be simulated with the second data set and their response studied. For linear models this involves examining their bode-plots, pole-zero diagrams, and comparison of different models as discussed in the previous section.

5.8.4 Model Validation

The validation of a model is to decide whether it is acceptable for its intended use, and this is closely related to its quality.

Model Quality

Model quality can have different meanings since it can be judged on different grounds. A model's quality can depend on its application, its stability when used with different data sets, and its ability to reproduce the behaviour of the system. Comparing different models created from different data-sets is useful to gain confidence in a model. Simulations and bode diagrams are useful for this purpose). It is important to remember that some analysis methods are unreliable under feedback. A good test of a model is to simulate it using fresh data and then compare the output to the measured output.

Residual Analysis

The *residuals* are the parts of the data that the model could not reproduce, and are given by

$$\varepsilon(t) = \varepsilon(t, \hat{\theta}_N) = y(t) - \hat{y}(t \mid \hat{\theta}_N) \tag{5.80}$$

The residuals should be independent from the input (i.e. uncorrelated). If this is not the case then the model is likely to be under-fitted. The covariance between residuals and past inputs is given by

$$\hat{R}^N_{\varepsilon u}(\tau) = \frac{1}{N}\sum_{t=1}^{N} \varepsilon(t+\tau)u(t) \qquad \text{where} \quad |\tau| \leq M \tag{5.81}$$

- If these numbers are small then the model is likely to be a good fit, and for large N, will be approximately normally distributed with zero mean if $\varepsilon(t)$ and $u(t)$ are independent.

- If there is correlation for negative values of τ, i.e. values of $\varepsilon(t)$ affect later values of $u(t)$, and this suggests that the data was collected during feedback, and not that the model is necessarily incomplete.

- When the ARX model is used, the LS procedure makes the correlation between $\varepsilon(t)$ and $u(t)$ zero for the data used for the estimation.

If a model of the disturbance signal is required, the residuals should be mutually independent, and can be found by plotting

$$\hat{R}_{\varepsilon}(\tau) = \frac{1}{N}\sum_{t=1}^{N}\varepsilon(t)\varepsilon(t+\tau) \tag{5.82}$$

If the numbers are not small for $\tau \neq 0$, part of $\varepsilon(t)$ could have been predicted from past data, and is also a sign of model deficiency.

5.9 Cyclic Machines

Cyclically varying machines are more difficult to deal with than the machines considered so far since they are non-stationary, or time-variant. This means that parameters will change as a function of angle, and therefore as a function of time as the machine rotates. As discussed previously, there are two ways of viewing the problem; 1) to represent the varying parameters as functions of angle, where angle is an input, leading to non-linear models which are not desirable. 2) to represent the varying parameters as functions of time, leading to time-variant but still linear models. The latter is the preferred choice due to the comparative mathematical simplicity and will be used wherever possible.

5.9.1 Inertia variation $G(\theta)$

The kinetic energy in a machine can be written

$$K.E. = \tfrac{1}{2} J_P \Omega^2 \tag{5.83}$$

(since it would be $\frac{1}{2}mv^2$ for a translational mechanical system). J_P is termed the *polar-inertia* and if it depends on angle (θ), the variation in K.E. can be written (using the product rule)

$$\frac{dK.E.}{dt} = \tfrac{1}{2} J_P \frac{d(\Omega^2)}{dt} + \tfrac{1}{2}\Omega^2 \frac{d(J_P)}{dt} \tag{5.84}$$

since both Ω^2 and J_P are time varying. Applying the chain rule yields

$$\frac{dK.E.}{dt} = \tfrac{1}{2} J_P 2\Omega \frac{d\Omega}{dt} + \tfrac{1}{2}\Omega^2 \frac{dJ_P}{d\theta} \cdot \frac{d\theta}{dt} \tag{5.85}$$

which simplifies to either

$$\frac{dK.E.}{dt} = \tfrac{1}{2} 2.\Omega.J_P . \frac{d\Omega}{dt} + \tfrac{1}{2}\Omega^2 \frac{dJ_P}{dt} \tag{5.86}$$

or

$$\frac{dK.E.}{dt} = \tfrac{1}{2} J_P 2\Omega \frac{d\Omega}{dt} + \tfrac{1}{2}\Omega^3 \frac{dJ_P}{d\theta} \tag{5.87}$$

which is linear. Hypothetically if the machine is loss-less, then if it is rotating with no external interaction it will have a specific K.E. The change in K.E. will therefore be zero $\left(\frac{dK.E.}{dt} = 0\right)$ and substituting into equation 5.87 and rearranging gives

$$\frac{d\Omega}{dt} = \frac{-\tfrac{1}{2}\Omega^3 \frac{dJ_P}{d\theta}}{\Omega.J_P} \tag{5.88}$$

This represents the natural acceleration and deceleration of the machine due to varying inertia, and to make it rotate with constant velocity would require the application of a torque to cancel it out. The torque required is

$$T_G = J_P \left(\frac{-\tfrac{1}{2}\Omega^3 \frac{dJ_P}{d\theta}}{\Omega.J_P} \right) \tag{5.89}$$

which simplifies to

$$T_G = \tfrac{1}{2}\Omega^2 \frac{dJ_P}{d\theta} \tag{5.90}$$

This torque is the inertia variation (as a function of angle) multiplied by the acceleration, i.e. *Torque required* $= G(\theta)\Omega^2$, where $G(\theta) = \frac{1}{2} \cdot \frac{dJ_P}{d\theta}$.

There are two issues that have to be resolved, and they are what tests cover the behaviour of the machine (i.e. tests that exercise and extract all important dynamics of the machine), and what model structure to use to represent the machine.

5.9.2 Tests to Charaterise Cyclic Machines

There are three main tests required to characterise a time-varying machine:

1) A "quasi-stationary" test to determine friction in both the forward and backward directions. This test requires the machine to be rotated as slowly as possible through at least one cycle in both directions. The quasi-static torque can then be measured and is likely to be different in both directions since energy can be stored and released in cyclically varying machines. For example in the forward direction the viscous drag corresponding to a given angle can be written $T_{QSF}(\theta(t))$.

2) A "constant-velocity" test for a range of different velocities to determine the viscous drag and variability of the inertia. The idea behind these tests is to attempt to drive the machine at constant velocity and observe the change in torque. This torque variation is made up of a viscous-drag component (B), and an inertia variation component (G), and for a given constant velocity Ω_1 ($\Omega \equiv \dot{\theta}$) can be written

$$T_{\Omega 1}(\theta) = \Omega_1^{\,0}.T_{QSF}(\theta) + \Omega_1^{\,1}.B(\theta) + \Omega_1^{\,2}.G(\theta) \tag{5.91}$$

A set of these equations can be created for different constant velocities for Ω_1, $\Omega_2 \ldots$, and placed in a matrix equation for convenience

$$\begin{bmatrix} 1 & \Omega_1 & \Omega_1^{\,2} \\ 1 & \Omega_2 & \Omega_2^{\,2} \\ \vdots & \vdots & \vdots \\ 1 & \Omega_n & \Omega_n^{\,2} \end{bmatrix} \cdot \begin{bmatrix} T_{QSF}(\theta) \\ B(\theta) \\ G(\theta) \end{bmatrix} = \begin{bmatrix} T_{\Omega_1}(\theta) \\ T_{\Omega_2}(\theta) \\ \vdots \\ T_{\Omega_n}(\theta) \end{bmatrix} \tag{5.92}$$

which can be rearranged to give the parameters

$$\begin{bmatrix} T_{QSF}(\theta) \\ B(\theta) \\ G(\theta) \end{bmatrix} = \begin{bmatrix} 1 & \Omega_1 & \Omega_1^{\,2} \\ 1 & \Omega_2 & \Omega_2^{\,2} \\ \vdots & \vdots & \vdots \\ 1 & \Omega_n & \Omega_n^{\,2} \end{bmatrix}^{+} \begin{bmatrix} T_{\Omega_1}(\theta) \\ T_{\Omega_2}(\theta) \\ \vdots \\ T_{\Omega_n}(\theta) \end{bmatrix} \tag{5.93}$$

where $\begin{bmatrix} 1 & \Omega_1 & \Omega_1^{\,2} \\ 1 & \Omega_2 & \Omega_2^{\,2} \\ \vdots & \vdots & \vdots \\ 1 & \Omega_n & \Omega_n^{\,2} \end{bmatrix}^{+}$ is the pseudo-inverse of $\begin{bmatrix} 1 & \Omega_1 & \Omega_1^{\,2} \\ 1 & \Omega_2 & \Omega_2^{\,2} \\ \vdots & \vdots & \vdots \\ 1 & \Omega_n & \Omega_n^{\,2} \end{bmatrix}$

3) A sinusoidal excitation of the machine at frequency ω with the machine turning at a much lower frequency Ω_{mean} ($<<\omega$). A set of distinct excitation frequencies can be applied starting at a frequency lower than the expected minimum resonant frequency of the machine. In this case the torque at the machine comprises

$$\begin{aligned} T(t) = {} & T_{QSF}(\Omega_{mean}.t) \\ & + \Omega_1.B(\Omega_{mean}.t).\Omega_{mean} \\ & + \Omega_1^{\,2}.G(\Omega_{mean}.t).\Omega_{mean}^{\,2} \\ & + T_{OSC}\cos(\omega t + \phi) \end{aligned} \tag{5.94}$$

and the shaft angle can be found by

$$\theta(t) = \Omega_{mean}.t + A.\sin(\omega t + \phi) + B.\cos(\omega t + \phi) \tag{5.95}$$

Ω_{mean} is a constant velocity, so differentiating this equation twice yields

$$\dot{\Omega}(t) = 0 - \omega^2 A.\sin(\omega t + \phi) - \omega^2 B.\cos(\omega t + \phi) \tag{5.96}$$

The excitation signal is $T_{OSC}\cos(\omega t + \phi)$, and the phase-shifted contribution of the response A is the contribution from any damping and also if the machine excitation has not reached a steady state. If the machine excitation has reached steady state and the choice of ω is appropriate, A should be insignificant compared to B, and the A term can be ignored. If this is the case, multiplying equation 5.96 by J_P gives the torque equation

$$J_P\dot{\Omega}(t) = -J_P\omega^2 B.\cos(\omega t + \phi) \tag{5.97}$$

and $J_P\dot{\Omega}(t)$ is the excitation torque $T_{OSC}\cos(\omega t + \phi)$, so

$$T_{OSC} = -B\omega^2 J_P(\Omega_{mean}.t) \tag{5.98}$$

ω should be chosen to minimise A and this is done by ensuring that it is sufficiently high compared to rotating frequency Ω_{mean} (so that $T_{OSC}\cos(\omega t + \phi)$ is superimposed on the slower constant velocity rotation), and lower than any resonant frequencies of the machine. If ω is too small $J_P(\theta)$ will tend to be averaged ('smeared') out and will not correctly represent the inertias at the corresponding angles. Figure 5.2 illustrates this.

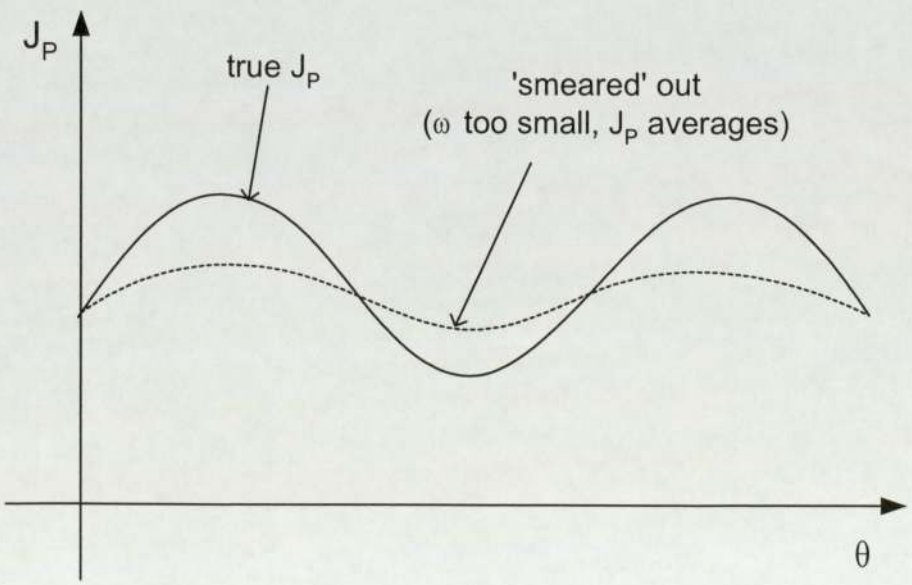


Figure 5.2 Illustration of averaged J_P due to ω too low

The values of $J_p(\theta)$ can be (and possibly should be) checked numerically by testing the condition

$$\frac{dJ_p(\theta)}{d\theta} \approx 2G(\theta) \tag{5.99}$$

for values of θ covering the range 0 to 2π.

5.9.3 Tests to Identify Internal Resonances

The combined mass and flexibility of linkages within a machine gives rise to internal resonances, some of which effect the machine shaft angle. Hypothetically, if a mechanism were constructed entirely of rigid parts then its state could be specified by the state variables $\{\theta, \dot{\theta}\}$ alone. In practice, machines will contain components where the combined mass and flexibility give rise to resonant frequencies, some of which are within the frequency spectrum of interest (and therefore detectable), and some of which are not. If the mechanism has resonances within the frequency range of interest, then the state vector has additional vectors $\{\boldsymbol{q}, \dot{\boldsymbol{q}}\}$ which describe a displacement and velocityof the shaft from a reference position. Figure 5.3 shows a four bar mechanism, an example of a machine with internal resonances.

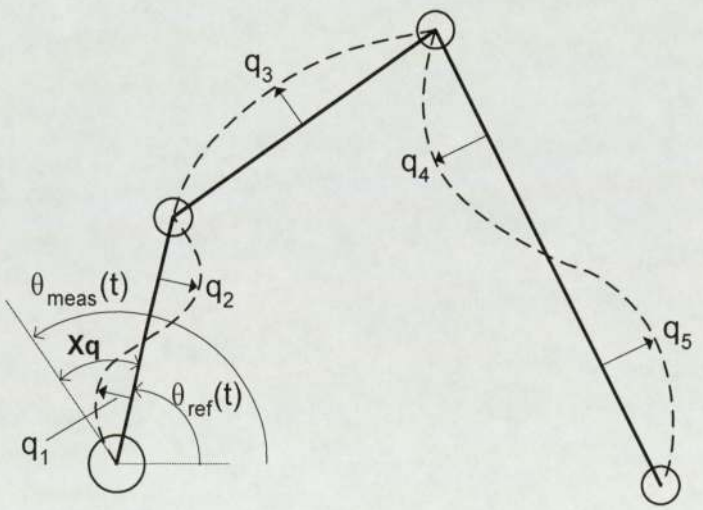


Figure 5.3 Example of machine with internal resonances

The machine has additional state vectors consisting of q_1, q_2, ..., q_n, and their first derivatives that describe the state of the internal modes of oscillation. Equation 5.100 describes the components that make up the shaft torque for a non-constant velocity.

$$T_{\Omega}(\theta_{ref}(t)) = T_{QSF}(\theta_{ref}(t)) + \Omega.B(\theta_{ref}(t)) + \Omega^2.G(\theta_{ref}(t)) + J_p\ddot{\theta}_{ref}(t) \qquad (5.100)$$

The $J_p\ddot{\theta}(t)$ term is zero for a constant velocity. The internal dynamics of the machine can be represented by the equation

$$\mathbf{M}(\theta_{ref}(t)).\ddot{\boldsymbol{q}}(t) + \mathbf{C}(\theta_{ref}(t)).\dot{\boldsymbol{q}}(t) + \mathbf{K}(\theta_{ref}(t)).\boldsymbol{q}(t) = \Omega^2\mathbf{Q}_{ACC}(\theta_{ref}(t)) + \mathbf{Y}.T(t) \qquad (5.101)$$

The **M**, **C** and **K** matrices are mass, damping and stiffness coefficients respectively. $\mathbf{Q}_{ACC}$ is a vector of internal forces and the term $\Omega^2\mathbf{Q}_{ACC}(\theta_{ref}(t))$ represents the internal imbalance excitation. $\mathbf{Y}.T(t)$ is a force produced by the torque $T(t)$.

The internal oscillations will cause the shaft to oscillate around the measured angle θ_{meas}, and an output equation relating equations 5.100 and 5.101 is

$$\theta_{meas} = \theta_{ref} + \mathbf{X}\boldsymbol{q}(t) \qquad (5.102)$$

It is important that the speed of rotation is sufficient such that the shaft will not stop rotating or reverse, since this eliminates the non-linear frictional behaviour of the machine.

The identification of cyclic machines is discussed in the context of example machines in chapter 9.

CHAPTER 6

6 Test Rig Design

Some findings of this chapter have already been used in previous chapters, but this is unavoidable given the structure of this text. In practice much of the work was performed concurrently, and this chapter aims to consolidate the mechanical design aspects of the project.

As previously discussed, the test-rig has two main purposes. Firstly it is required to *characterise* a rotary mechanical machine. The 'external machine' is likely to have the ability to store energy (i.e. converting kinetic energy to potential energy), and vice-versa, for example due to an inertial machine, or perhaps even a periodically changing inertia such as in a slider-crank mechanism. Secondly it is required to *emulate* a rotary mechanical machine, or mechanical component. The test-rig output shaft is likely to be connected to a drive / motor pair that will have the ability to apply torque to the shaft in either direction in an unpredictable manner.

It was decided that the general form of the test-rig should take the form shown in figure 6.1. Essentially it is required to produce a torque on the output shaft which is either a perturbation torque (in the case of machine characterisation), or a torque which is a consequence of the measured variables (in the case of machine emulation).

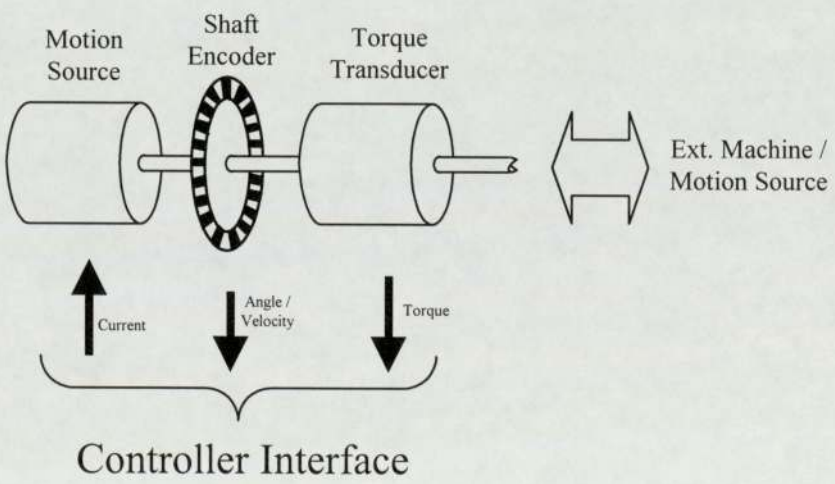


Figure 6.1 General Construction of the Test-Rig.

6.1 Mechanical Specification

The requirements of the test-rig that need to be specified are maximum torque, maximum rotational velocity, and torque-bandwidth (discussed below). The power is also important, and is the product of torque and angular velocity. The torque-bandwidth needs to be sufficient to excite all the modes of interest of an external machine during a characterisation experiment, and to represent the dynamics of a machine during an emulation experiment. When emulating a machine the test-rig needs to match or exceed the torque and angular velocity rating of the motion source. These of course depend on the operating range of the machine being emulated and the specification of the motion source driving it. It is assumed that the motion-source component of the test-rig is an electric motor, since this is the most feasible option.

6.1.1 Test Rig Requirements

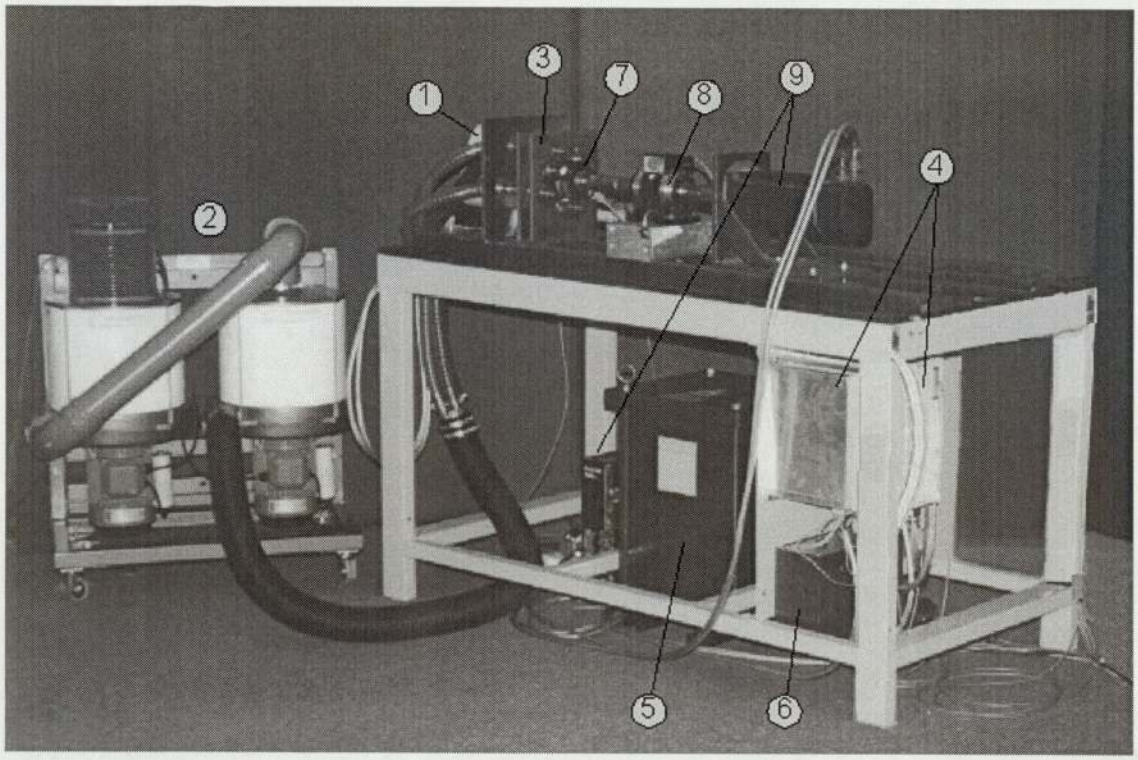
The specification of the machines that may be emulated can vary considerably, and is consequently not a basis for specifying the test-rig performance. Since high performance motion sources were available in the laboratory it seemed natural to use these to drive the test-rig in machine emulation mode, and the specification of these was used as a basis for the test-rig specification, which was decided to be **approximately**:

Maximum Torque:	50 Nm peak
Maximum Velocity:	3000 rpm
Torque Bandwidth:	>300 Hz

6.2 Mechanical Arrangement

Various mechanical configurations to provide a controlled to the output shaft were investigated but only one seemed feasible which was chosen, and is discussed here. Figure 6.2 shows the power electrical and mechanical parts of the test-rig with an external drive / motor pair (Electrocraft / BRU200) connected, used to drive the test-rig during machine emulation. The test-rig is built on a ½" thick steel tabletop mounted on a rigid steel framework. The electrically noisy high power equipment and cables were kept apart from the small signal cables to reduce noise in the measured variables. The laboratory's slide rail system was used to clamp the moving equipment to the tabletop,

which is sufficiently rigid to prevent unwanted vibrations. The other parts of the test-rig shown in the key are discussed in their corresponding sections below.



KEY:

1 = Motors (×4)
2 = Double Blower
3 = Gearbox Mk I
4 = Motor Drives (×2)
5 = Drives Power Supply Transformer
6 = Motor Series Inductors
7 = Shaft Encoder
8 = Torque Transducer
9 = External Drive / Motor Pair

Figure 6.2 Test-rig with external drive / motor pair connected

6.2.1 Motor Selection Criteria

The most straightforward construction of the test-rig would be to use a directly coupled high performance motor. A market survey was performed to compare various motors to make the best choice for this test-rig. The criteria on which to base this choice are maximum speed, torque, power output, inertia and mechanical time constant.

6.2.1.1 Mechanical Time Constant

The mechanical time constant (assuming a first-order system approximation) is the time required for the motor's speed to attain 63.2% of its final value for a fixed voltage level, and is influenced largely by the motor's inertia, shown below

$$T_m = \frac{R.J}{K_T^{\ 2}} \tag{6.1}$$

Tm = mech. time const., R= motor resistance, J = rotor inertia, K_T = torque constant (Nm/A)

It is a measure of the goodness of a motor but it particularly relates to the operation of the motor in open loop. It is generally desirable to have a motor with a short time-constant.

6.2.1.2 Torque Bandwidth

Bandwidth is normally defined as the frequency at which the magnitude of a quantity drops to $1/\sqrt{2}$ (its half power point) of its zero-frequency level. The rotor inertia and the stiffness of the output shaft of a motor form a low-pass filter, which limits (and determines) the upper frequency limit of a motor, and this can be found by shaft_stiffness / rotor_inertia. Torque bandwidth can therefore be found by $f = \frac{1}{2\pi}\sqrt{\frac{k_S}{2J_R}}$.

6.2.1.3 Power Rate

This torque is required to rotate the shaft, and since the product of torque and velocity is power, the power-bandwidth, known as the *power-rate*, is an important factor. Most motors however specify their mechanical time constant with other parameters, from which power-rate can be derived. Power-rate into an inertial load is given by

$$Power\ Rate = T_L \times \ddot{\theta}_L \tag{6.2}$$

units ω / sec.

6.2.1.4 Inertia Matching

If an external machine (referred to in this discussion as a load) is predominantly inertial, then this inertia will need to progress between states of high and low kinetic energy quickly. Some of the air-gap torque T_{AG} (see section 4.1) is required to accelerate the rotor's inertia, so given the motors inertia J_M and the 'load' inertia J_L, the shaft torque can be found by

$$T_{SH} = T_{AG} \times \frac{J_L}{J_M + J_L} \tag{6.3}$$

The acceleration of an inertial load is given by T_{SH} / J_L, and the power rate into it is therefore given by ${T_{SH}}^2 / J_L$. Substituting equation 6.3 into this gives the power rate into the load,

$${T_{AG}}^2 \times \frac{J_L}{(J_M + J_L)^2} \tag{6.4}$$

The maximum 'power-rate' from the motor is found to be achieved when $J_L = J_M$, and this is called *inertia matching*. Ideally the inertia of the test-rig will match the inertia of any external machine attached to it, but since this is not predictable an exact specification of the required test-rig inertia is impossible. Typically during machine design, source and load inertias are connected together using a gear train, the ratio of which is determined by the two known inertias. No inertia is ever increased to improve matching, and the required angular velocities and torque also determine the ratio of the gear train. Since the load inertia is unknown, these two factors and the motor geometry alone were used to determine the gear ratio (section 6.3 discusses gearbox design). The motors used for the test rig have inertia of 7.1×10^{-6} kg.m^2 per motor which is very low, and the inertia of the gearbox alone referred to each motor is 1.83×10^{-5} kg.m^2, for the first gearbox construction (referred inertia calculations are in appendix C2.5).

6.2.2 Choice of Motors

The motors found to have the lowest mechanical time constants were brushed DC motors with iron-less rotors (of a 'basket wound' construction), but these tend to have a

relatively low torque rating. Motors were available with a mechanical time constant of 1.4 ms producing 1.1 Nm continuous, 2.56 Nm peak (max. speed 7000 rpm), and it was decided that coupling eight motors of this type using a gearbox would be the preferred option. To provide the required torque / speed they would be geared down approximately 2.5:1 and this would also provide the mechanical coupling of their shafts. Figure 6.3 shows the conceptual test-rig construction.

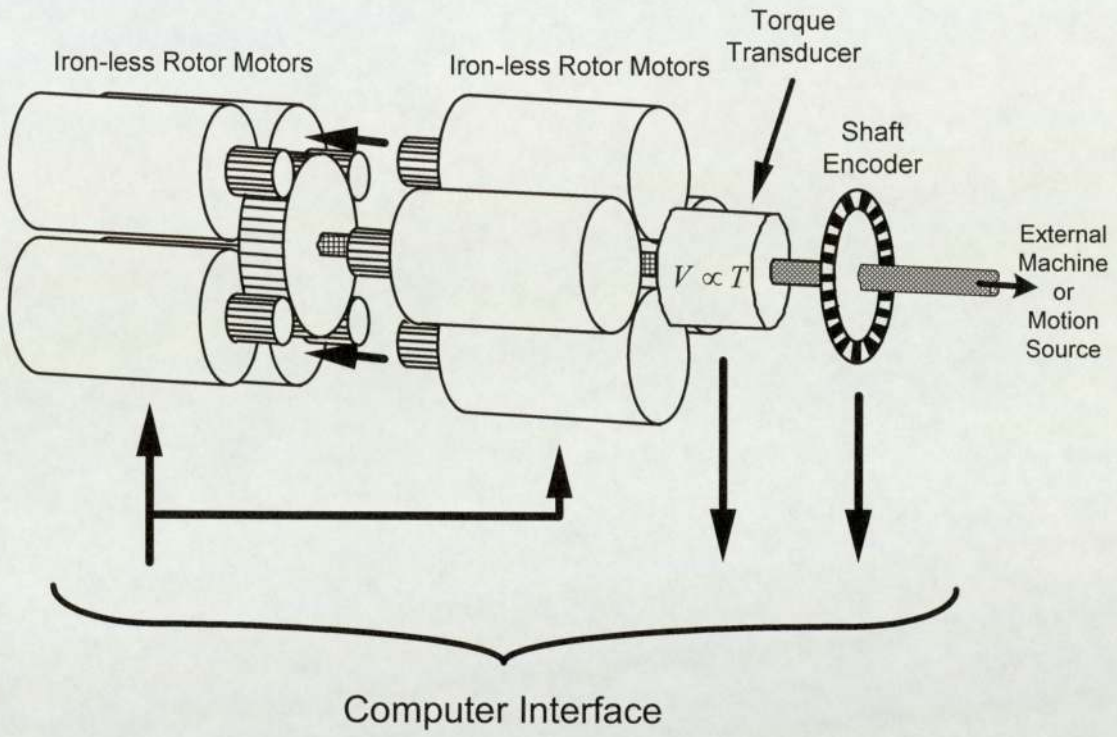


Figure 6.3 Approach 2, using geared iron-less rotor motors.

It was mentioned in section 6 that low inertia is important for system performance, and it is due the high T:J ratio of these motors that they were chosen. With an ideal gearbox, the test-rig was now potentially capable of delivering 8 x (1.1 x 2.5) = 22 Nm continuous (51.2 Nm peak), speeds of up to 2800 rpm. Due to the high cost of the motors and lack of a price incentive for buying eight at a time, four were initially bought to evaluate the design with the option of extending this to eight at a later stage. This would limit the torque to 11 Nm (25.6 Nm peak). Ultimately only four motors were purchased but this is sufficient to prove the concept of the test-rig. The important remaining issue was to design a gearbox with minimal inertia and backlash.

6.2.3 Motor Cooling

The operating range of the motors is shown in figure 6.4. For the motors to run at their operating limits they each need to be force cooled with 11 litres/min of air at 9250 Pa (≅2.73 in Hg, or 92.5 mbar) pressure. A 2.2 kW double blower was purchased which provides 44 litres/min at 9250 Pa (but 75 dBA of noise!). A manifold was constructed from soldered brass that splits the blower's 3" outlet pipe into four 1" pipes to feed the motors forced-air cooling input. These can be seen in figure 6.2. The forced-air cooling is not essential, but allows the motors to be operated at full power thus allowing the test-rig to achieve a higher torque and power rate.

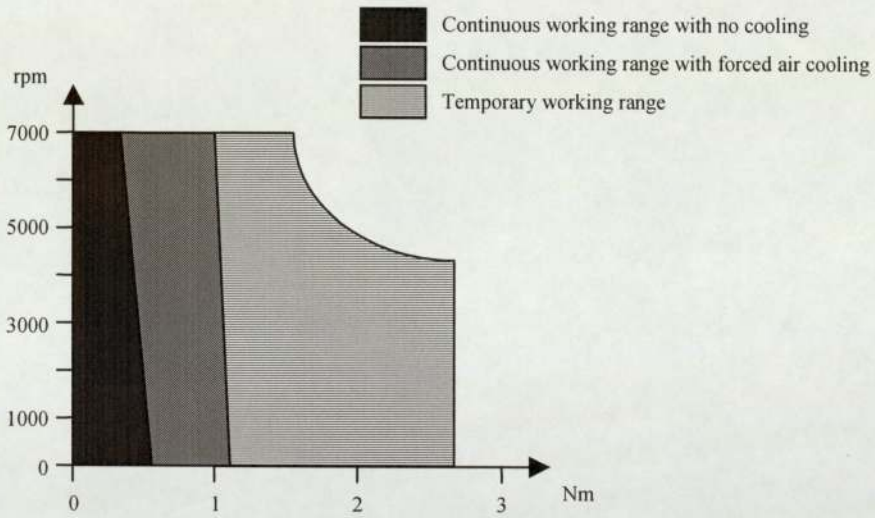


Figure 6.4 Motors Operating Range.

6.2.4 Motors Brackets

The motors are mounted on a bracket that has four mounting holes orthogonal to one another. The mounting holes are skewed by 45° so that two of these brackets holding motors may be offered up to each side of the gearbox, which is capable of connecting 8 motors, 4 each side. Only one bracket was therefore used (see appendix C2.3). Figure 6.3 shows the motor orientation. A different method was required for the gearbox Mk II, and this is discussed in the context of the gearbox.

6.3 Gearbox Design

The design of gearboxes is an engineering discipline in itself, and it is beyond the scope of this project to provide an optimum design. Basic design fundamentals are observed however and the designs described here attempt to make the best use the resources available.

The main issue is the mounting geometry of the motors. This effects the size and type of gear and the method of eliminating backlash. Two gearboxes were designed. The first design uses four planetary spur gears around a centre gear and uses a split-gear system to pre-load the gears to remove backlash. The second design uses four spiral bevel gears connected to the motors sandwiched between two larger spiral bevel gears, and presses the gear meshing together to pre-load the gears to remove backlash. In addition to providing a coupling method, the gear ratio also reduces the velocity and increases the torque. The motors velocity is 7000 rpm maximum, so to provide a close match to the specification described in section 6.1.1 the required gear ratio is 2.33.

6.3.1 Spur Gear Gearbox

The construction of the first gearbox is simple and is shown in figure 6.5. Only four planet gears were actually used, but provision was made on the gearbox casing for eight. The teeth are parallel with the axis so the gears produce only radial and tangential forces. The shaft bearings are single row radial ball bearings, and radial movement is sufficiently reduced by axially pre-loading each axis bearings using beryllium-copper crinkle washers. Normal meshing of the gears occurs when they are mounted at standard centre distances where backlash is quoted at 0.08 - 0.15mm. This equates to a maximum initial angular backlash of 0.4°. Each planet gear is constructed of two gears, one fixed to the shaft and one free to rotate about the shaft. The free gear is bolted to the fixed one with limited rotational adjustment available to allow their relative positions to take up any radial backlash (this adjustment appears to be necessary initially whilst the gears 'bed-in' and also periodically as the gears wear). The transmission force is therefore only acting on one of these gears at a time (per motor), depending on the direction of rotation.

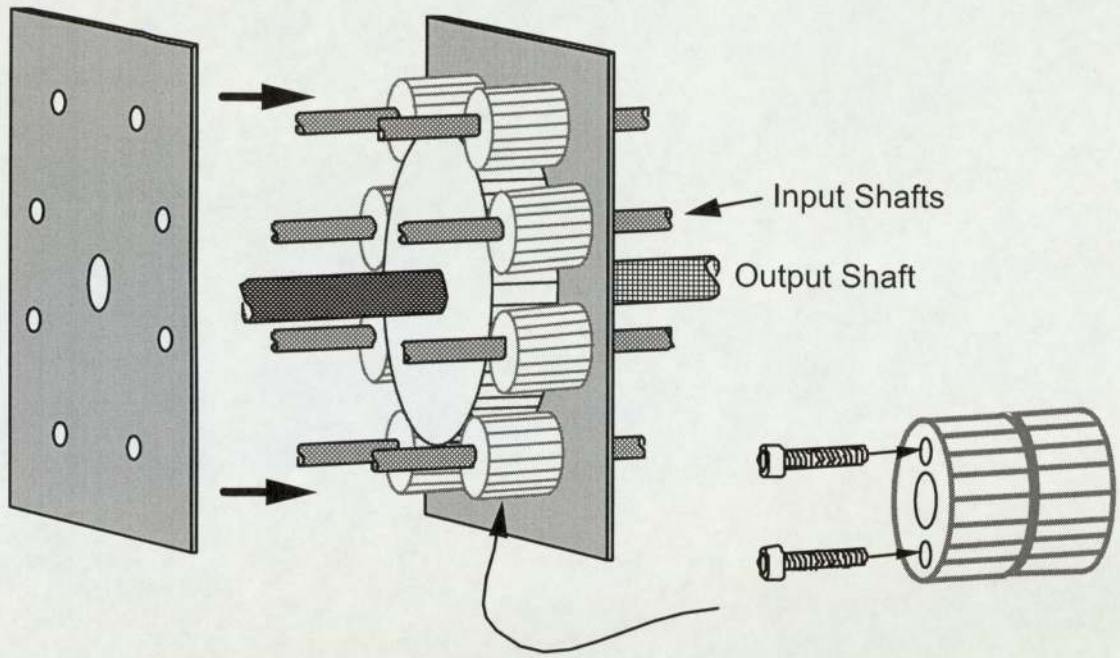


Figure 6.5 Gearbox MK I, using spur gears

The gears are rated for a particular torque, speed, lubrication and lifetime. The gears are not run continuously for long periods of time and in any case are easily replaced. By reducing the lifetime the power (torque x speed) rating of the gears can be increased, and a thinner lighter material (Tufnol) used. The advantage of this is the obviation of lubrication and a large reduction in inertia compared to the other available materials.

The main problem with this design is that when the motors are placed next to each another the distance between the motor centres is quite high. As a result, large diameter gears are required (particularly the centre gear) which have a greater than desirable inertia. Large holes were drilled through the sides of the centre gear to reduce mass and thus inertia. The inertia of the gearbox referred to the output shaft was calculated.

The inertia of a solid disc (which a gear wheel essentially is) is given by:

$$J = \frac{m \times d^2}{8} \qquad \text{where } m = \text{mass}, d = \text{diameter} \qquad (6.5)$$

The number of teeth on the centre gear is odd so the planet gear teeth are meshing at different places and the *cogging torque* in minimised. Due to the geometry of the motors and availability of the gears, the gear ratio is 103:43 or (2.388:1).

Using the density of Tufnol = 1.36×10^3 Kg/m^3 the mass and the inertia of the solid gears can be calculated. The referred inertia of the gearbox at the output shaft can be found by:

$$J_{referred\_to\_O/P\_shaft} = \left(J_{motor} + J_{planet}\right) \times num\_motors \times n^2 + J_{centre} \qquad (6.6)$$

where n is the gear ratio.

For the Tufnol gears used, the gearbox inertia referred to the output shaft is 418.4×10^{-6} Kg.m^2. Including the motor inertias, the referred inertia at the output shaft is 580.4×10^{-6} Kg.m^2. The calculations and mechanical drawings for these can be found in appendix C2.2.

6.3.2 Spiral Bevel Gear Gearbox

The design outlined above has the disadvantage that the large spur gear wheels introduce considerable inertia into the system and the straight cut gears generate some cogging torque. An alternative to this design is shown below in figure 6.7, which uses spiral bevel gears, has much smaller gears and consequently much less inertia.

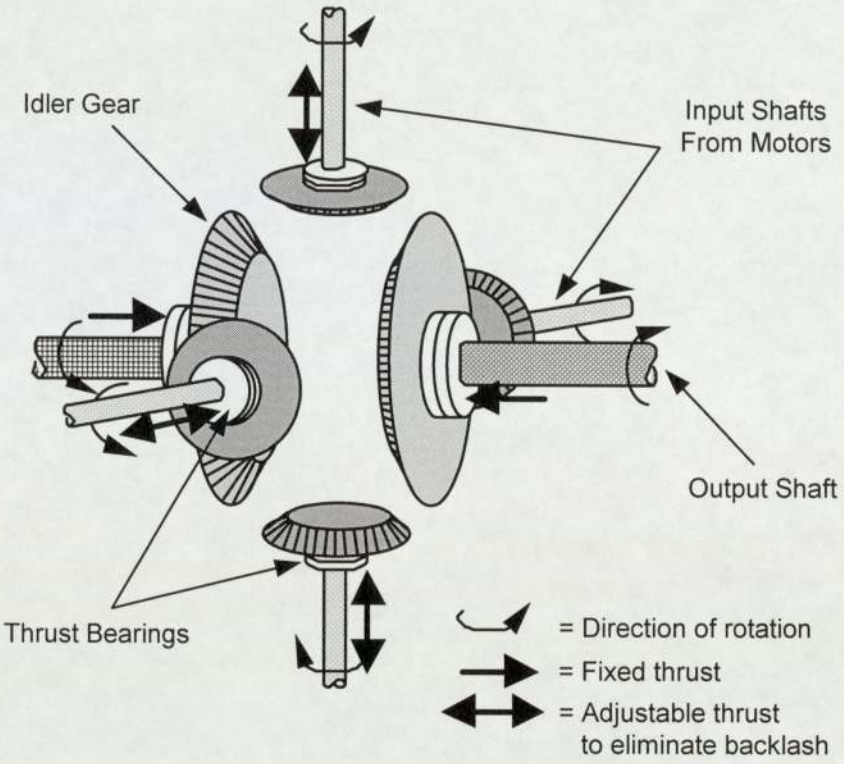


Figure 6.7 Gearbox MK II, using bevel gears

Spiral cut gears also gears generate much less cogging torque than straight cut gears. Depending on the direction of rotation, the spiral bevel gears either thrust inward or outward (see figure 6.6). Backlash can therefore be eliminated by preventing the gears from thrusting outward by using thrust bearings which thrust the input gears inwards towards the output gears to take up any slack. The adjustment is provided using bicycle 'bottom bracket cups' (LHS). This reduces the gears centre distance thus pre-loading them and removing backlash. Because this does not allow any tolerance for gear run-out or thermal expansion, the input gears are forced inward via beryllium-copper crinkle washers that take up this movement and hence provide a near constant thrust force.

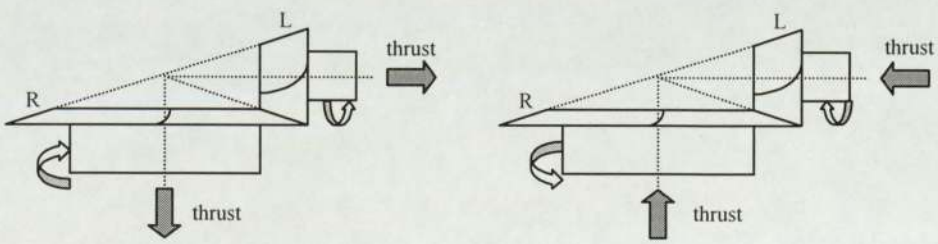


Figure 6.6 Rotation / Thrust Directions for Spiral Bevel Gear.

The gears are bought in sets, have a gear ration of 2.5:1 and are case hardened. Due to the material, and potential high speed of the gears, a high quality lubricant is required. A special lubricant is available but only in quantities of 10 gallons or more, and since the properties of this are similar to synthetic engine oil, the latter was used.

Figure 6.8 shows a close-up of the physical construction of the Gearbox MK II with one side removed, and figures 6.9 a and b show the gearbox with one side removed and fitted.

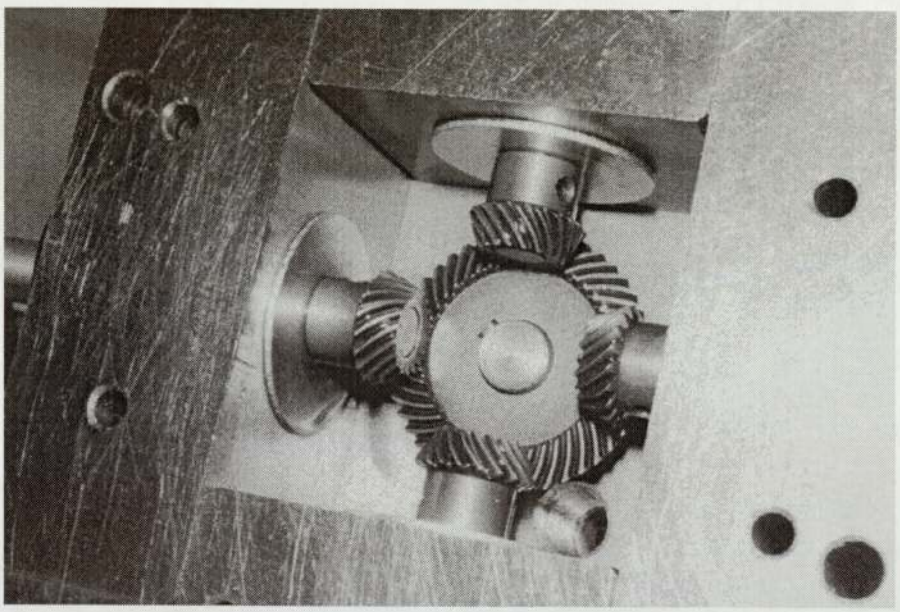


Figure 6.8 Internal Gearing of the Gearbox MK II with one side removed

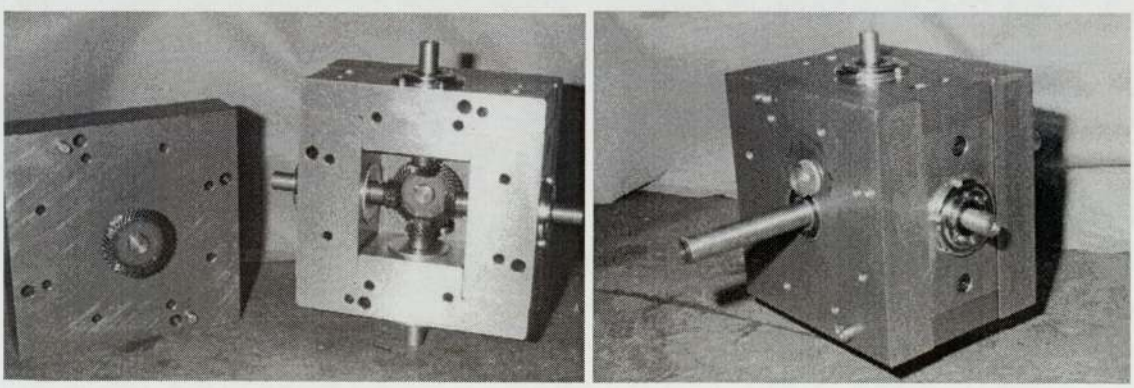


Figure 6.9 Gearbox MK II a) with one side removed, and b) fully assembled

Drawings in Appendix C2.4 show the construction of the spiral bevel gearbox, and also for the positioning of the motors. The motor positioning is awkward in that they have to be mounted radially to the gearbox, occupying considerable space. This is not a problem however provided the motors / gearbox assembly is at one end of the bench so that two of the motors can be positioned below the level of the bench top.

The spiral bevel gear gearbox was designed and constructed late in the project, so considerable emulation and characterisation testing was performed using both gearboxes. Using the same calculations as in section 6.3.1, the referred inertia of the complete test-rig can be calculated, and was found to be 204×10^{-6} Kg.m^2 using the spur-gear gearbox. The calculations for this are in appendix C2.5.

6.4 Measurement of Shaft Variables

The variables of interest to this chapter are the variables that describe the state of the output-shaft, i.e. torque, angle and velocity. Their measurement is discussed but the electronic interfacing is left to the next chapter.

6.4.1 Shaft Torque

Chapter 4 discusses why the dynamic shaft torque cannot be measured using the motor's current. For most servo-machines, the air-gap torque is proportional to current. The air-gap torque is the torque between the rotor and stator. Because the rotor has a significant inertia, the air-gap torque is not an accurate measure of output torque (except for steady state conditions). Hence a torque transducer is required to be placed in between the test-rig and the output-shaft.

A very simplified view of the system is to look at the test-rig connected to external machine as two inertias connected by a flexible coupling. The resonant frequency of this system can be found by:

$$f_n = \frac{1}{2\pi}\sqrt{\frac{K_s(J_1+J_2)}{J_1J_2}} \qquad (6.7)$$

(where K_s is coupling stiffness, J_1 and J_2 are the inertias)

The natural frequency of this system needs to be as high as possible so that it does not limit the bandwidth of the test-rig or cause instability in a closed-loop control situation. To achieve this it is important that the inertias are as low as possible and the coupling stiffness as high as possible. It is not possible to change the inertia of the external machine, but it is a design consideration to minimise the inertia of the test-rig components. The most flexible part of the coupling is the torque transducer. Unfortunately torque transducers rely on the shaft flexibility to measure the torque, and consequently the more rigid transducers are the higher torque-rated ones, which have a bigger inertia due to a larger diameter internal shaft. This inertia can be considered as consisting of two separate inertias (added to J_1 and J_2), either side of an inertia-less shaft of a particular stiffness. Loss of measurement resolution is not a problem as fine accuracy is likely to be swamped with noise in any case, so a 200Nm transducer (TM212) was purchased, which is thought to be a good trade-off between torsional-stiffness and inertia for this purpose. The TM212 has a torsional stiffness of 38.2 KNm/rad and inertia of 425×10^{-6} Kg.m^2. If a mainly inertial machine is connected to the test rig, the resonant frequency of this second-order system can be calculated using equation 6.7. This should be much higher than the dynamics of interest.

6.4.2 Shaft Angle

There are two main types of shaft encoder, absolute and incremental. An incremental shaft encoder is used here since they generally offer better resolution, and a custom made resolution (1024 lines) was chosen for reasons explained in chapter 7. It contains a disc with a number of equally spaced radial lines which produce two output signals, 'A' and 'B'. There is also a third output 'I'. A and B are both symmetrical square waves which are phase shifted by 90° (quadrature) so that the direction of rotation can be determined. I provides one pulse per revolution for position referencing. The ∅20mm hollow shaft allows it to be connected around the output shaft using appropriate packing. The inertia of the encoder is 750 g.cm^2, which is very low compared with the inertias of the other test-rig components.

6.4.3 Shaft Velocity

Two methods of monitoring shaft velocity were explored.

The first uses a tacho-generator which is similar to a small motor, but has a much more linear response that produces an emf proportional to angular velocity (1.4 mV/rpm). It has an extremely low inertia (0.64 $g.cm^2$) which can be considered negligible. The interface to this is discussed in chapter 7.

The second method employs a frequency-to-voltage converter circuit that generates a voltage proportional to the frequency of one of the shaft encoder pulses (A or B). This circuit is also discussed in chapter 7.

CHAPTER 7

7 Electrical / Electronic Interfacing Issues

This chapter discusses the power electrical issues such as the choice of drives and the meeting their power requirement, and the smaller electronic signal issues such as the generation and conditioning of measurement signals. The circuit diagrams for all the circuits discussed in this chapter are in appendix E.

7.1 The Motor Drives

The motors operating range is shown in figure 6.4. The peak torque is 2.56 Nm at 40 A and this is possible with a terminal voltage of 75V. The length of time this can be sustained for is defined by the thermal time constant of the motors, and is discussed in the context of motor protection later in this chapter. Motor drives can generally be configured to operate in two modes, torque and velocity. Torque mode is used in this application, which controls the motor current (proportional to air-gap torque) to be proportional to an input *reference* voltage signal. If more than one motor is connected to any drive then they will need to be connected in series to ensure the current is shared evenly, and that all motors produce the same torque. This may not be the case if they were connected in parallel because any uneven motor resistances (which are effected by temperature) would determine the current flowing in each drive. It would then be difficult to control the electrical power consumed by each motor and the rating of one may be exceeded. Operating the motors in series therefore leads to three possible configurations of drives and motors for this application, shown in figure 7.1.

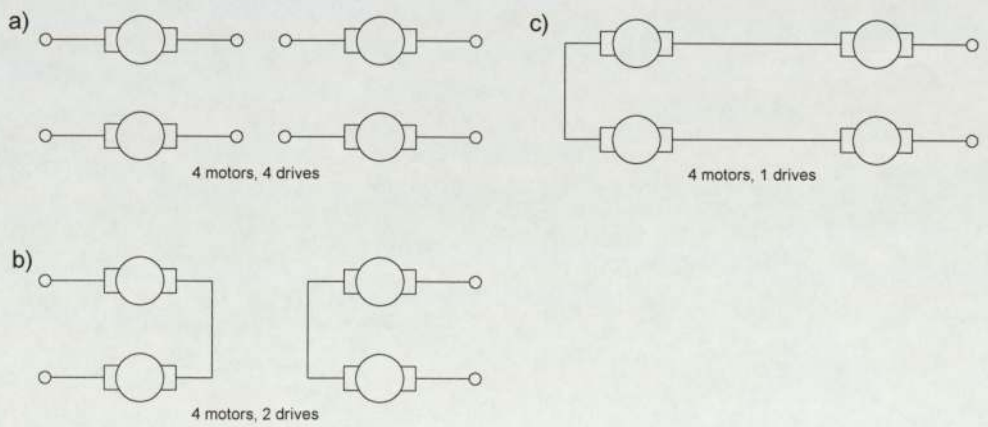


Figure 7.1 Possible Electrical Configurations of the Motors

The choice of configuration depends mainly on the drives available, so a search for suitable drives was performed.

7.1.1 Choice of Motor Drives

The choice of high performance linear DC drives was small because the current trend is to use brush-less DC and AC motors which require special drives. Most of the drives available are designed to be used with standard motors in standard configurations, which the test-rig is not. The motor manufacturers offer (expensive) drives specifically for the chosen motors, but these would require one drive per motor ('a' in figure 7.1) which is an expensive solution. The ideal solution would have been configuration 'c' in figure 7.1, but this would mean a peak voltage of 300V and a peak current of 40A, and stock drives with this specification do not appear to be available, except for thyristor models which switch on the supply frequency of 50Hz.

The drives most suited to this application (after the expensive linear drives) are transistorised PWM servo controllers. These change the analogue input signal to a constant frequency, varying duty cycle (i.e. pulse-width modulation - PWM) signal that is applied to high current H-bridges. The frequency of operation tends to decrease with size, and is usually less than 10 kHz for drives suited to this project. The switching frequency however determines the sampling speed of the torque control loop, and the minimum load inductance required (section 7.1.2). A fast switching speed is therefore desirable. A drive was found having a switching frequency of 17kHz, so two of these were purchased to drive the motors in configuration 'b' (figure 7.1). Each drive is capable of supplying 35A continuous (70A peak) at 70-205V DC (supply voltage 50-145V AC 3-phase), and can be configured as torque (current) controllers.

It is preferable to carefully balance the drives so that equal current flows down each pair of motors. This was accomplished by setting the two drives to operate in torque mode, adjusting them so that their gains were identical, and setting their current limits to ±40A.

7.1.2 Transformers and Series Inductors

The maximum power which can be delivered to the motors with forced-air cooling is 4 × 1 kW = 4 kW continuous, and 4 × 3 kW = 12 kW peak. A 6 kVA transformer capable of providing this was therefore chosen, which has a 415 V primary winding and 60, 80, 100, and 140 V secondary tappings. The drives operate by switching a DC bus, which is obtained from rectifying and smoothing the AC supply. From the motors operating graph it can be calculated that the motors maximum voltage is 75 V. To prevent them suffering insulation damage the voltage should be kept below the motors limit × 2, so the 100 V tapping was used, making the DC bus $100 \times \sqrt{2} = 141$ V.

The drives have a switching frequency of 17 kHz which means that the PWM (pulse width modulation) output contains this fundamental frequency and its harmonics. The low frequency content of this output created by adjusting the pulse width is used to drive the motors, and so a minimum load inductance is required to filter out the high frequency components. A low inductance is beneficial since this reduces the electrical time-constant and thereby minimises the time-delay in producing torque. The minimum inductance specified by the drives is 0.7 mH, and since the inductance of two motors in series is 2 × 0.125 = 0.25 mH, a further 0.45 mH is required. Two 0.45 mH were therefore used, built to operate at 18 kHz and rated at 40 A.

7.1.3 Motor Regeneration

Regeneration is the action of motor braking, where the motor acts as a generator and in taking kinetic energy from the load converts it into electrical energy and returns it to the drive. This has the affect of increasing the bus voltage which could damage the drive if excessive. The drives have provision for connecting 'power dump' resistors so two high power 40 Ω resistors are used to dissipate any regenerative power. It is worth noting that some more sophisticated drives allow this energy to be fed back into the supply, but this can be expensive and is not popular.

7.1.4 Electrical Noise Issues

The orientation of the test-rig is such that all the signal lines (i.e. small signal lines from transducers etc…) are kept well away from power cables and sources of high electrical

noise. The test-rig bench is grounded at one point using a very low impedance connection, and panels are electrically bonded together to provide a low impedance at high frequencies. The drives are particularly electrically noisy since they switch high currents at high frequencies and this noise will have a tendency to emanate from connections to the drive. The motor supply cables are high power, screened (and armoured), are earthed at both ends and are run from the drives to the motors inside one of the metal tabletop sections. Ferrite rings were placed on both ends of the motor supply cables, and to the input of the transformer to reduce high frequency emissions.

7.1.5 Interface to DSP

The output from the DSP board is –3V to +3V and the output impedance is 2 KΩ. The voltage input to each drive is –10V to +10V with input impedance of 20 KΩ, so an interfacing circuit is required to amplify the signal by at least 3.33 and buffer the resulting signal with a low output impedance to reduce induced noise in the signal cables. An operational amplifier circuit was constructed with two stages, the first having a gain of 3.4, and the second consisting of two voltage followers to drive each motor drive. A third voltage follower was also used to drive an analogue meter for development purposes.

7.2 Motor Protection

The motor protection circuit uses Hall-effect current transducers to detect the amount of current flowing in the motors, and an opto-isolated open-collector arrangement to detect a motor voltage imbalance (appendix E.2.1). A 4.7 KΩ pre-set resistor is used to set the amount of current that flows in the event of a motor failure (either open or short circuit).

The purpose of measuring the current is so that the heating effect in the motors can be estimated. The heating within the motors can be approximated to a first order system, where the temperature is proportional to the square of the current times time (I^2t), since the power dissipated by the motors is I^2R (R is the resistance of the motors and is assumed to be constant). A circuit was designed (appendix E.2.2) which uses analogue multipliers to square the current. This is fed into an RC first-order circuit which has the same thermal time constant as the motors. A relay switched by the blower changes the

value of this time constant to correspond to different thermal time constants with and without forced-air cooling (3.2 seconds and 10 seconds respectively). Comparators are used to detect when the RC voltage (estimate of temperature) exceeds a maximum (set by VRx3 and VRx4). Voltage-followers are also included to buffer the current and estimated temperature should they be required for use by the DSP board. The I^2t trips and `BLOWER_ON` signals are then fed to the logical control part of the motor protection (appendix E.2.3).

The control part of the motor protection simply latches any fault signal and switches off a relay which disconnects the drives. An emergency-stop switch is also connected allowing the user to shut off the drives. LED's are used to signal the source of the fault.

7.3 Torque Transducer Interface

The torque transducer is powered from the signal-processing unit by a 24 volt supply. It produces a voltage proportional to the output shaft torque of 1V per 40 Nm (output impedance 500Ω). The output is single-sided but is routed to the signal-processing unit through separately shielded wires in the supplied lead. The manufacturers suggest using an instrumentation amplifier to receive this signal as this will significantly reduce the effect of common-mode noise in the cable. A high quality precision instrumentation amplifier was chosen (INA 118), and the gain set to 1. This signal is then amplified by an operational amplifier with a gain set to 4, and both signals are output, so that the DSP can be connected to either. Another op-amp is also used to drive an analogue meter, which displays the torque for development purposes.

7.4 Shaft Encoder Interface

The shaft encoder used on the test-rig is a relative *incremental* type, which produces two quadrature signals, A and B, on the rotation of the shaft. An additional signal is produced, I, which is the index signal, occurring once per revolution. Many shaft encoder interface designs have been published, and they generally use some kind of latch circuit between the encoder signals and an array of binary up-down counters. The circuit designed here uses a similar technique but counts on the *edges* of the two quadrature signals from the shaft encoder. The top two traces of figure 7.2 shows the

two shaft encoder signals, A and B. The third trace is the I signal and the fourth trace is discussed later.

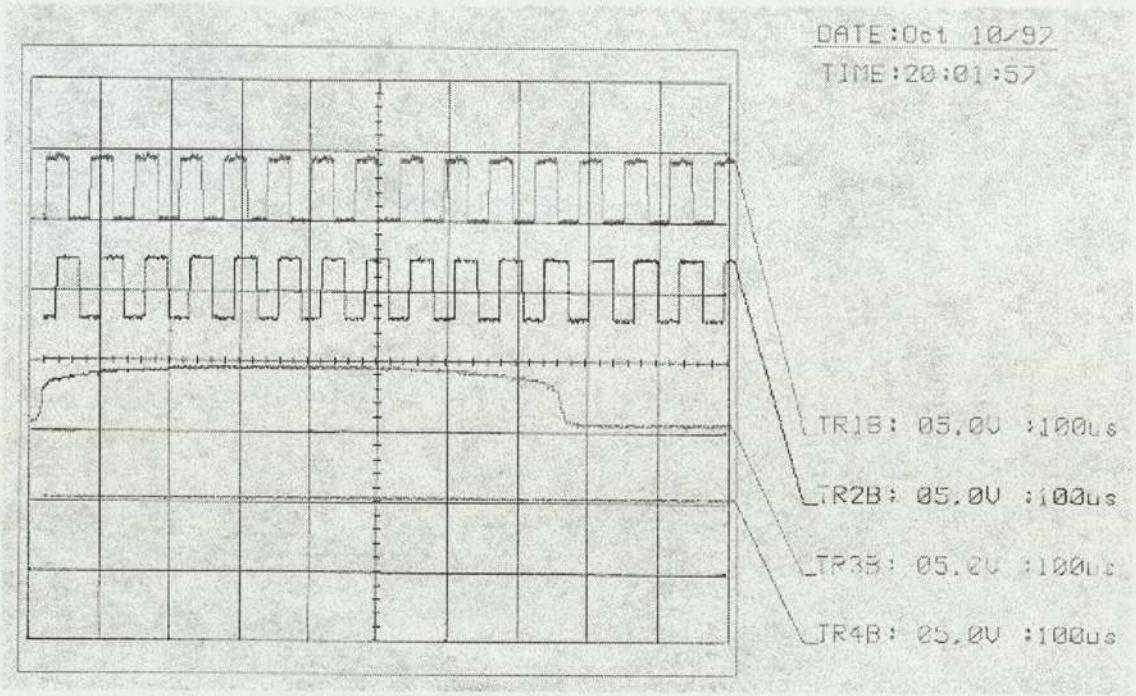


Figure 7.2 Signals from the shaft encoder

The shaft encoder was specified to have 2048 pulses per revolution, which means that by counting on the edges of the A and B signals, a resolution of 8129 is achievable. Programmable logic devices were used to implement the design because they are easily re-configurable and can be designed to have a convenient pin-out, making construction easier. The PLD designs are shown in appendix E.4.1 and E.4.2.

A synchronous state-machine was designed which produces a pulse on the detection of each edge of the A and B signals. A direction signal is also generated which corresponds to the direction of the shaft and dictates whether the counters are to count up or down. Figure 7.3 is a state graph of this circuit. Since the output to the counters is a function of the present state only, it is referred to as a *Moore machine*. The state diagram therefore has the output associated with the state. In this machine the least significant bit (lsb) of the state variables is used as the counter increment / decrement pulse, and the most significant bit (msb) as the direction signal. It can be seen therefore that when the counters are counting in one direction the states are rotating clockwise around the outside of the state graph, and when the counters are counting in the other

direction the states are rotating anti-clockwise around the inside of the state graph. Transitions between the two concentric sets of states occur when the order of the A and B pulses signifies a change in shaft direction.

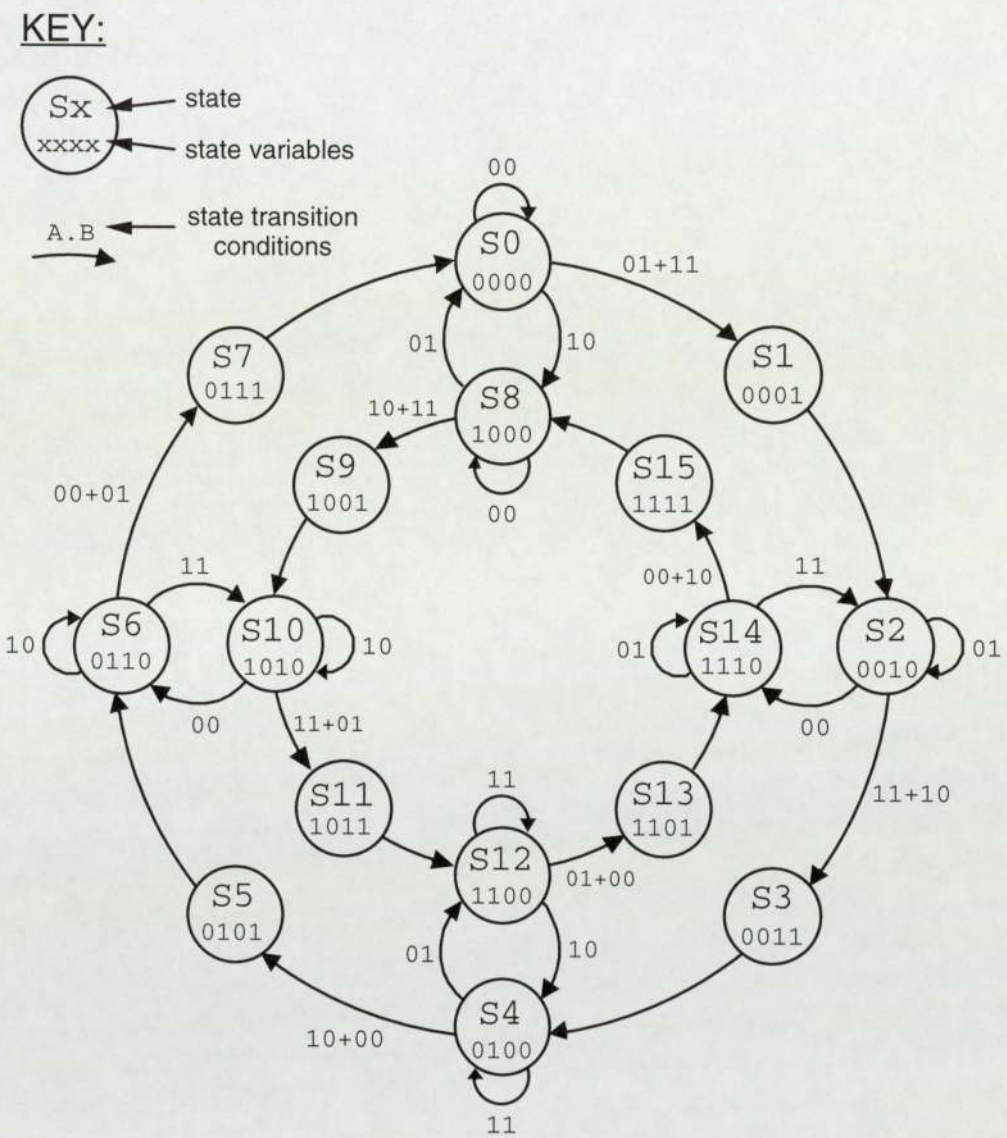


Figure 7.3 State graph of the counter input pulse generation circuit

Figures 7.4a and 7.4b show the simulation of the design, used to verify the functionality of the device prior to programming. The figures are a continuation of the same plot.

A modulo-8192 counter is required to count a complete revolution, and a synchronous counter is required since ripple counters produce transient states and suffer from

accumulated propagation delays. The largest PLD available was a 22V10, having 10 macro-cells, so the largest counter possible in one device is a modulo $2^{10} = 1024$, assuming it would fit into the device. It was decided therefore to use two of these devices and use two separate counters, with the overflow / underflow of the first incrementing / decrementing the second. The first counter is a modulo-64, and the second is a modulo-128, which combined makes a modulo-8192 counter as required.

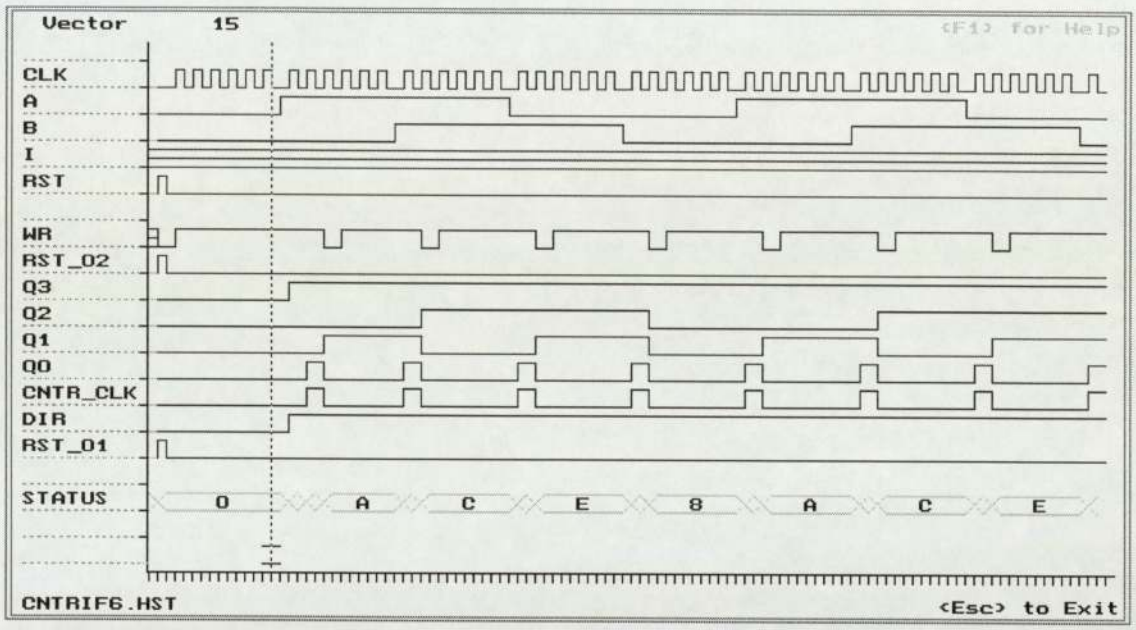


Figure 7.4a Simulation of the state machine's operation (clockwise)

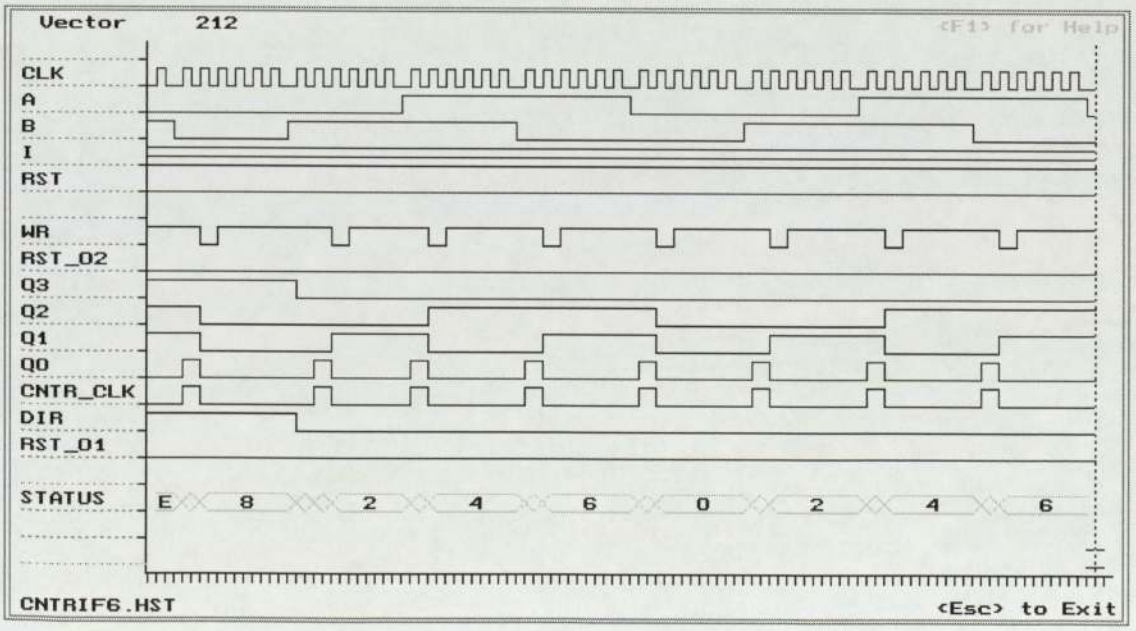


Figure 7.4b Simulation of the state machine's operation (anti-clockwise)

Ideally the DSP board would read this binary value directly, but the board procured for this project has no provision for this, so the only other option is to convert the signal into an analogue form to feed into the DSP board, which then converts it back! The binary count is fed into a digital-to-analogue converter (DAC) to produce a ramp waveform representing the angular count. The most suitable DAC for this purpose in terms of output voltage was a DAC712. This is a 16-bit DAC so to get the required voltage range out, ±10V, the least-significant two bits and the most-significant bit are grounded. A low pulse starts the DAC conversion on the $\overline{WR}$ pin, which is generated in the PLD containing the state-machine (CNTRIF6). This signal is the same as the first counter increment / decrement pulse, but delayed by one master clock cycle to allow the data lines on the DAC input to settle.

Figure 7.5 shows a plot of (from top to bottom) the A and B signals, the I signal, and the DAC converter output, for the test-rig output shaft rotating at a constant speed. Figure 7.2 shows the same signals from the same test but in much finer resolution such that it is impossible to see the DAC output as a ramp.

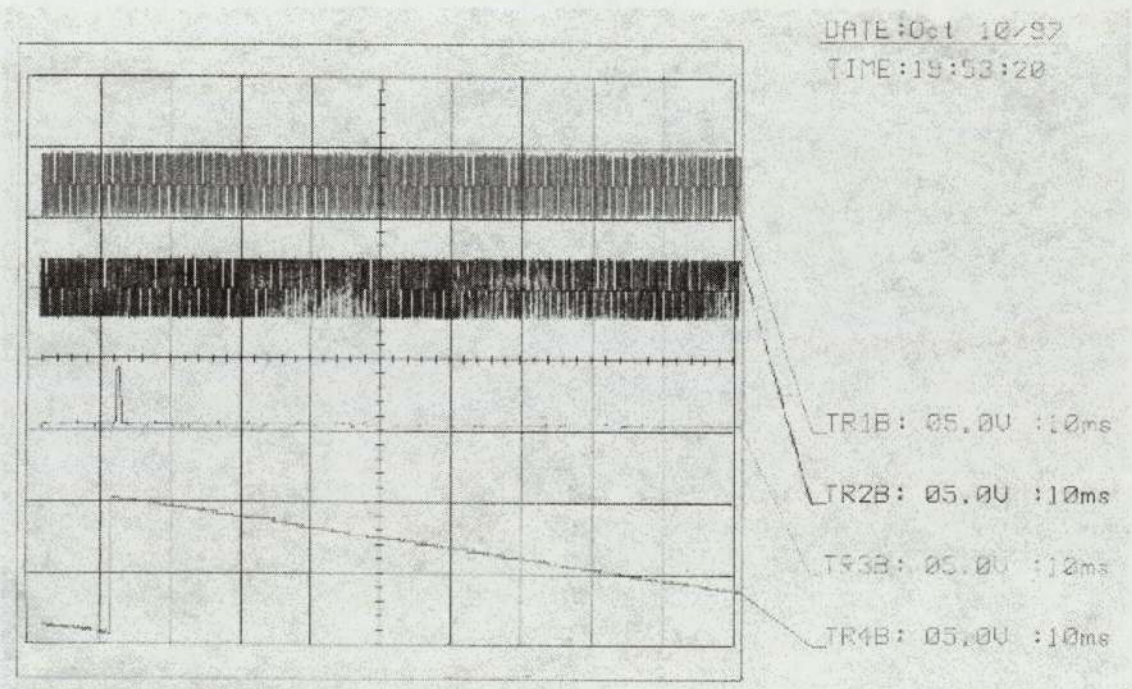


Figure 7.5 Signals from the shaft encoder and output from the DAC

The DAC output is fed directly to the DSP board, and no anti aliasing filter is used since the ramp will not be changing any faster than 50 Hz (the test-rig max speed = 3000

rpm). Putting a filter in line with the signal will also have the adverse effect of rounding the corners of the ramp waveform, which would have more effect at higher speeds.

7.5 Velocity observation

A tachometer was initially used to measure velocity. It produces a voltage proportional to velocity but also tends to produce significant ripple due to the low number of rotor segments. As an alternative, a velocity observation circuit was investigated, which uses a frequency to voltage converter (referred to here as an F→V converter).

The output from the angle counter provides 8192 pulses per revolution. At a shaft speed of 60rpm (1 Hz) the counter frequency is 8 kHz, and at the test-rigs maximum velocity of approximately 3000rpm (50 Hz) the counter frequency is 400 kHz. Even though the test-rig is unlikely to be used at this speed, a F→V converter capable of operating at this frequency is desirable. The Analogue Devices ADVFC32 IC was chosen since the maximum input frequency is 500 kHz. The circuit is not shown here since it is well documented in the data sheet, and requires a minimal set of external components.

The basic operation of the IC is reasonably easy to understand. Every time the input signal crosses a comparator threshold in a negative direction, a monostable is activated that switches a known current into a capacitor for a known period of time. As the frequency increases, the amount of charge injected into this integration capacitor increases proportionally. The voltage across the capacitor stabilises when the leakage current equals the current being switched into it. An external resistance determines this leakage. The net result is an average output voltage, which is proportional to the input frequency.

Before the test-rig construction was complete, it was thought that perhaps the shaft might rest at zero velocity on a shaft encoder pulse boundary, thus producing a train of spurious, perhaps high frequency pulses, that would register as a velocity. To overcome this potential problem a circuit was designed to inhibit the pulse input to the F→V converter.

Referring to figure 7.3, an angular oscillation of a 'stationary' shaft between directions would involve the toggling between inner and outer circles of the state-graph. This would correspond to a change in the msb of the state variables, and it is possible to filter this signal to remove any changes of less than a particular width. For example, if the shaft were oscillating between S0 and S8 in figure 7.3, a square wave of one state-change width would be produced, and if the shaft were oscillating around S8, S9, S10, S6, S7, S0, then a square wave of six state-changes width would be produced. A circuit to inhibit the F→V converter during a transition of less than three state-changes, using cascaded D-type flip-flops is shown in figure 7.6.

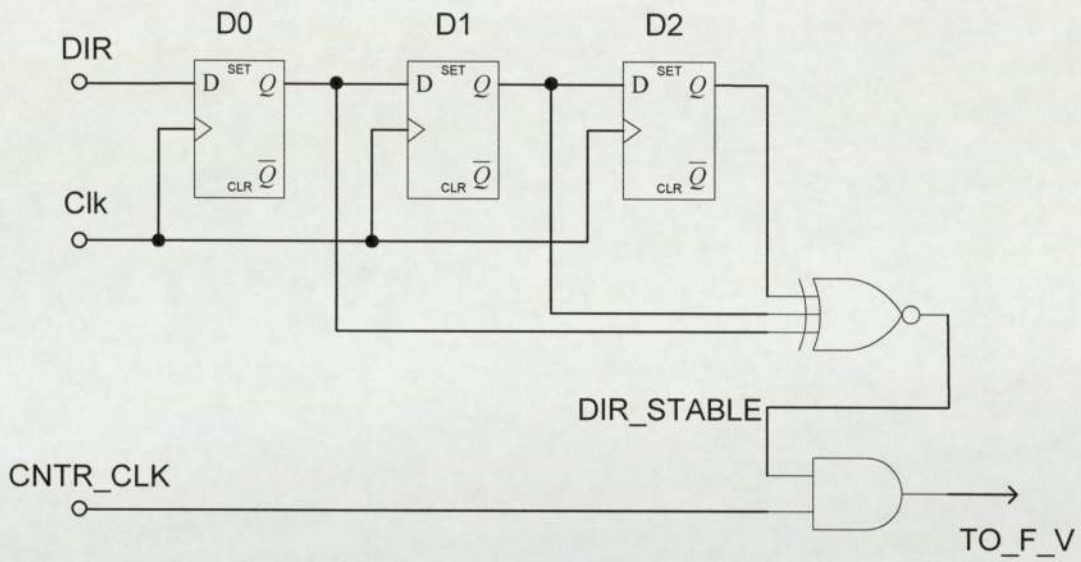


Figure 7.6 Circuit to inhibit F→V converter during direction change

During testing it transpired that the shaft actually oscillated about several shaft-encoder positions, and the circuit of figure 7.6 did not perform adequately. This circuit was implemented in a PLD for flexibility, and ultimately a design using six D-type flip-flops was implemented. The PLD design code for this is in appendix E.4.3, and the simulation is shown in figure 7.7.

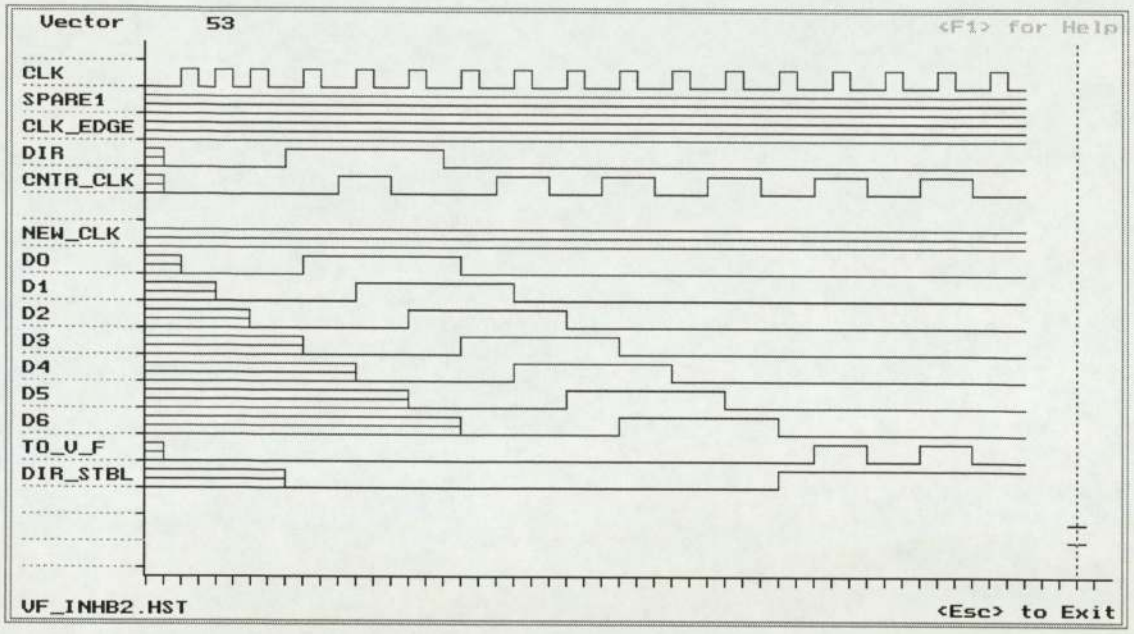


Figure 7.7 Simulation of the F→V converter inhibitor

Under certain conditions it was found that the shaft could oscillate even more than six positions, as figure 7.8 illustrates, and because forward and backward direction pulses are the same to the F→V converter it would appear to be a rotating shaft. This is likely to have a serious impact on the control of the test-rig, and so it was decided to use the tachometer instead, with suitable filtering.

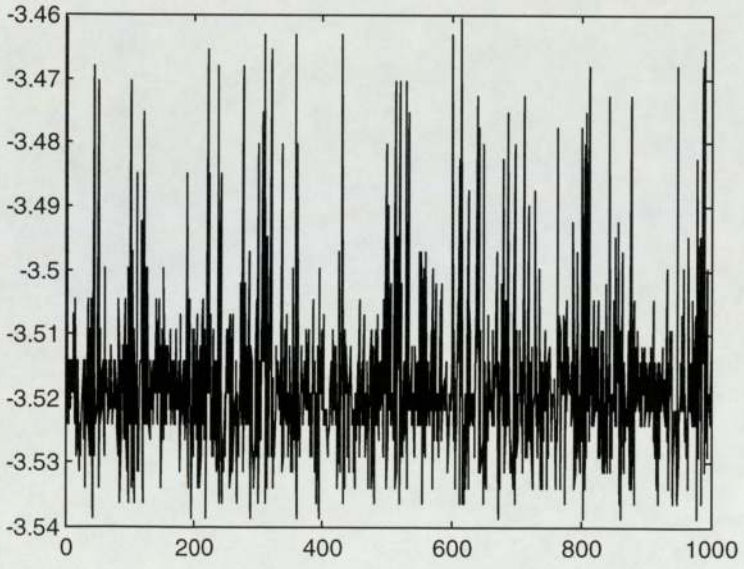


Figure 7.8 Plot of angle on a stationary shaft (motor drives on)

7.6 Anti-Aliasing Filters

Section 5.6.6 discusses the choice of sampling rate and the use of pre-sampling filters to prevent aliasing. Shannon's sampling theorem states that the signal must be sampled at twice the highest frequency content of the signal. The signal must therefore be filtered before sampling to remove frequencies above half the sampling frequency (i.e. the Nyquist frequency, $\omega_N = \omega_s/2$). A sampling frequency of 10 kHz was chosen, so low-pass filters with a cut-off frequency of 5 kHz were required.

Three types of filters were considered (which are the most common), and all involve a trade-off between pass-band to stop-band sharpness, flatness of the pass-band and stop-band, and phase shift. Chebyshev filters give a fast roll-off from pass-band to stop-band but incur some frequency ripple in the pass-band. Butterworth filters have a very flat pass-band, but Bessel filters have a very flat pass-band and provide excellent phase characteristics (at the expense of the roll-off sharpness). Since the shape of the sampled signals is important this linear phase-shift filter was chosen. A four-pole low-pass Bessel filter was designed with a cut-off frequency of 1500 Hz. This frequency was chosen because the dynamics of any machine to characterise / emulate will fall within this range. The roll-off is slower than the other filters, so it should be sufficiently low at and above 5 kHz to exclude any frequencies above the Nyquist frequency. Figure 7.9 shows the circuit for one of these filters of which three were built. Two of the filters were used for velocity and torque inputs, and the third is a spare. The component values were calculated using standard tables [26].

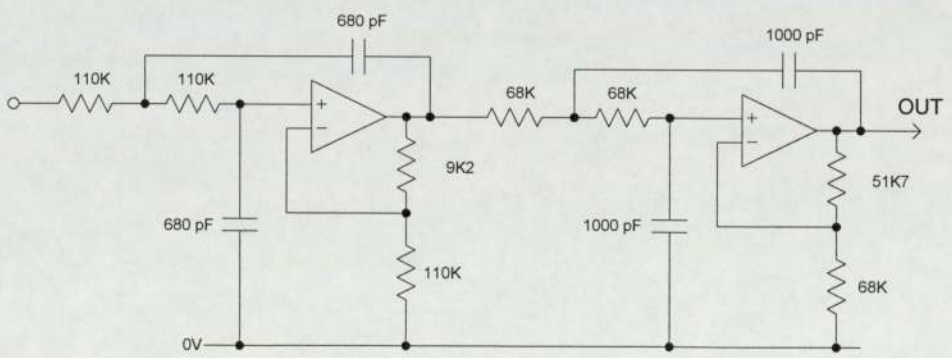


Figure 7.9 Four-pole Bessel filter circuit

CHAPTER 8

8 Design of example Machines to Characterise / Emulate

Sample machines with known characteristics are necessary to demonstrate the test-rigs operation. This enables a machine representation from a characterisation experiment to be used in an emulation experiment, and the accuracy of this determined by comparing the emulation to the physical machine. Many operations of the test-rig were tested individually and various control software and physical machines were used for this purpose. The characterisations were initially attempted using models of the machines in Matlab and applying identification methods, and these non-tangible machines are also described in this chapter.

8.1 Choice of Machines

A number of machines were chosen to test the functionality of the test-rig with varying complexity. As discussed in chapter 5, the choice of input stimuli should exercise all dynamic aspects of the system. The machines discussed here have been designed such that they have a variety of dynamic properties that require different identification methods to be employed.

8.2 Cam / Sprung Follower

A cam / sprung follower such as that shown in figure 8.1 was one of the first emulated machines due to its mathematical simplicity.

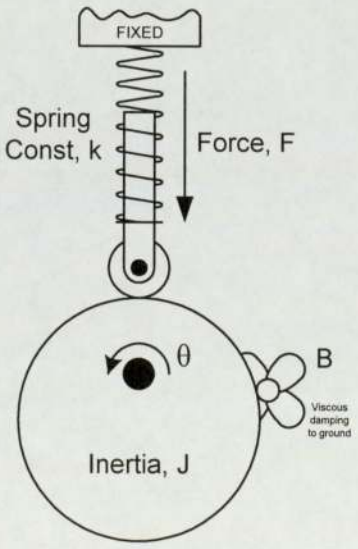


Figure 8.1 Diagram of the cam / sprung follower machine

The force exerted on the cam due to the spring alone (the mass of the follower is considered negligible) is $k.\sin(\theta)$, so including the inertia and damping the torque can be found by

$$T = k.\sin\theta + B\dot{\theta} + J\ddot{\theta} \tag{8.1}$$

DSP code used to implement this emulation is in appendix B.4. Data was not collected but this emulation was used as a demonstration, where the motor's current was limited (for safety) and the test-rigs shaft could be turned by hand. The action of the machine could be clearly detected, and increasing the inertia gave rise to an increased settling time of the machine, also clearly visible.

8.3 Various Torsional Spring / Inertia Systems

The torsional spring / inertia systems discussed here are simple linear time-invariant machines which are relatively easy to estimate the parameters for using techniques discussed in chapter 5. Matlab models of some of these machines are described and simulations are used in initial identification experiments discussed in chapter 9.

8.3.1 Inertia - Torsional Spring - Ground System

This system is shown in figure 8.2 and a similar machine was discussed in section 3.7.1. It is a second order, single degree of freedom system and can be described by equation 8.2. It is the rotary equivalent of the mass – spring – damper system described in many textbooks.

$$J\ddot{\theta} + B\dot{\theta} + k.\theta = T \tag{8.2}$$

The state-space equivalent of this equation can be determined by inspection and is as follows:

$$\begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{J} & -\frac{B}{J} \end{bmatrix} \cdot \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ -\frac{1}{J} \end{bmatrix} \cdot T \tag{8.3}$$

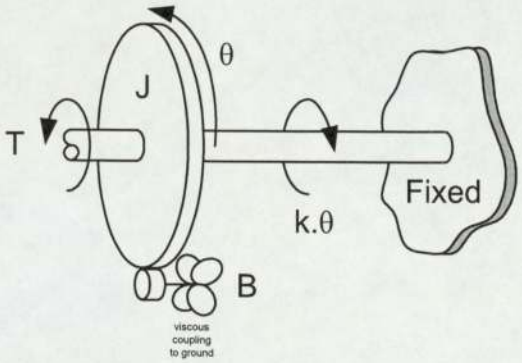


Figure 8.2 Diagram of the "Inertia - Torsional Spring – Ground" machine

A simulation of this machine was performed using equation 8.3 in Matlab, and the script for this is in appendix A.6 ($J=1\times10^{-3}$ Kg.m^2, K_S=197 Nm/rad and B=0.21 Nm/rad/s). Figure 8.3 shows this machine's response to a swept-sine torque input. The machine is a second order system, and it can be seen from figure 8.3 that is has a resonance at approximately t = 380 ms, where the frequency is 38 Hz (frequency = rate × time).

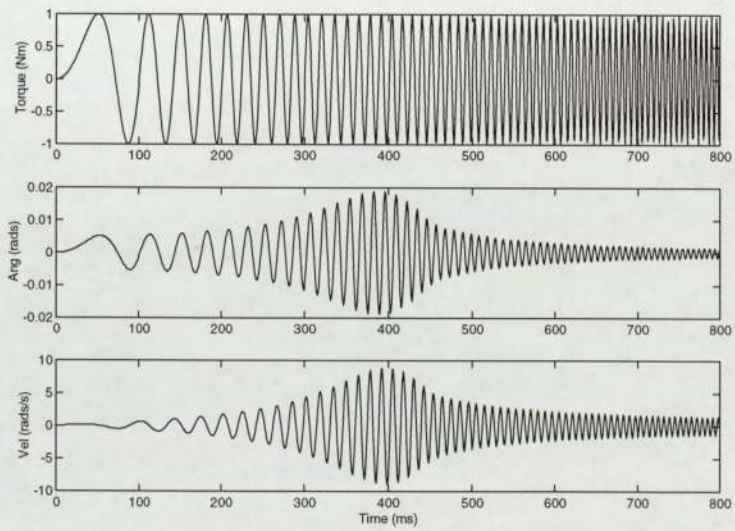


Figure 8.3 Simulation of the "Inertia - Torsional Spring – Ground" machine

8.3.1.1 Resonant Frequency of a Spring / Inertia System

The restoring torque of the torsional spring in figure 8.2 is directly proportional to the angular displacement of the inertia. If the system is given an impulse torque input then the system will oscillate, and this oscillation will be simple harmonic motion (s.h.m.). The general equation for s.h.m. is of the form of a second-order differential equation

$\frac{d^2x}{dt^2} = const.x$, which equation 8.2 is if the external torque is zero and there is no damping. It is worth noting that there is 90° phase difference between the displacement (angle) and velocity, and also between the velocity and acceleration. The equations for these respectively is

$$x = A.\sin(\omega t), \qquad \dot{x} = A.\omega.\cos(\omega t), \qquad \ddot{x} = -A.\omega^2.\sin(\omega t) \qquad (8.4\ a,b,c)$$

Ignoring damping and assuming the external torque is zero after an initial perturbation (e.g. an impulse), the system in figure 8.2 can be described by the equation

$$k.\theta + J.\ddot{\theta} = 0 \qquad (8.5)$$

Substituting equations 8.4a and 8.4c into this and factorising gives

$$(k - \omega^2 J).A\sin(\omega t) = 0 \qquad (8.6)$$

$(k - \omega^2 J)$ must equal zero, so $\omega^2 J = k$. The resonant frequency can therefore be found by rearranging this for f_0

$$f_0 = \frac{1}{2\pi}\sqrt{\frac{k}{J}} \qquad (8.7)$$

If there is damping present, then the oscillations will be damped and die away and the resonant frequency will be slightly lower (see section 3.14.5). Increasing the damping of the system above will reduce the amplitude of the response in figure 8.3, and that of the resonant frequency will be less pronounced.

8.3.2 Torsional Spring - Inertia System

The spring inertia system first described in chapter 3 has been referred to throughout this text, but a simpler physical model was actually constructed. This was used for initial characterisation tests, which could be easily compared to simulations of the same machine. The machine consists of a torsional spring driven at one end and with an adjustable inertia connected to the other. Additional (balanced) masses can be added or removed to the fixed one to change the overall mass, and thus inertia. Figure 8.4 outlines the construction of the machine, and figure 8.5 shows the actual machine used in the physical experiments.

The equations of motion that describe this machines behaviour are

$$J\frac{d^2\theta_2}{dt^2}+B\frac{d\theta_2}{dt}-T=0 \qquad \text{and} \qquad T=k(\theta_1-\theta_2) \tag{8.8a, b}$$

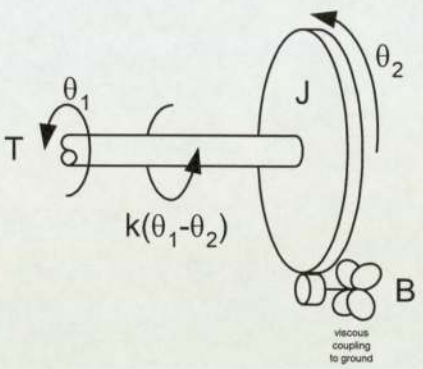


Figure 8.4 Diagram of the Torsional-Spring / Inertia machine

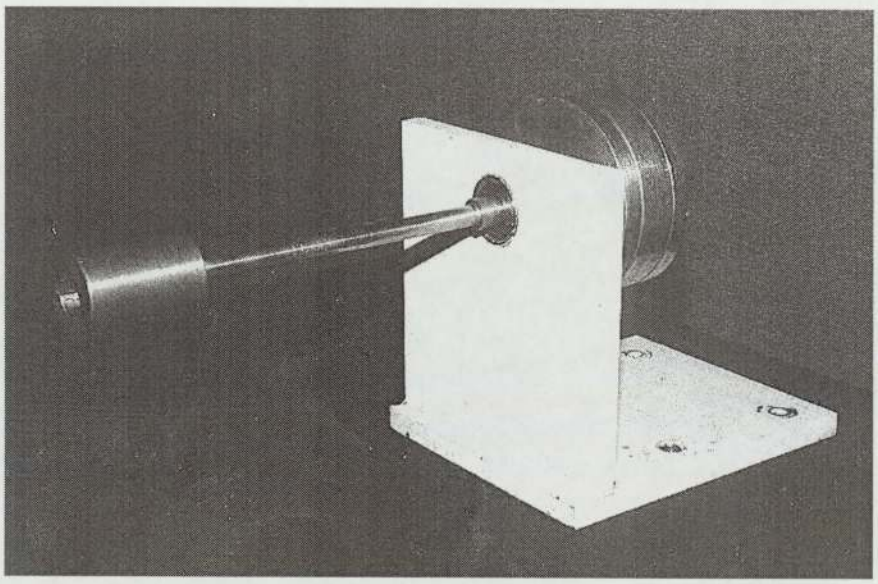


Figure 8.5 Physical Torsional Spring - Inertia machine

Equations 8.8a and b can be put into state-space form through inspection and used for simulation.

$$\begin{bmatrix}\dot{\theta}_2\\ \ddot{\theta}_2\end{bmatrix}=\begin{bmatrix}0 & 1\\ 0 & -\frac{B}{J}\end{bmatrix}\cdot\begin{bmatrix}\theta_2\\ \dot{\theta}_2\end{bmatrix}+\begin{bmatrix}0\\ -\frac{1}{J}\end{bmatrix}\cdot T \qquad \theta_1=\begin{bmatrix}1 & 0\end{bmatrix}\cdot\begin{bmatrix}\theta_2\\ \dot{\theta}_2\end{bmatrix}+\frac{1}{k}\cdot T \tag{8.9a, b}$$

A simulation of this machine ($J=1\times10^{-3}$ Kg.m^2, $K_S=197$ Nm/rad and $B=0.21$ Nm/rad/s) was performed using Matlab, and the script for this is in appendix A.7. Figure 8.6 shows this machine's response to a swept-sine torque input.

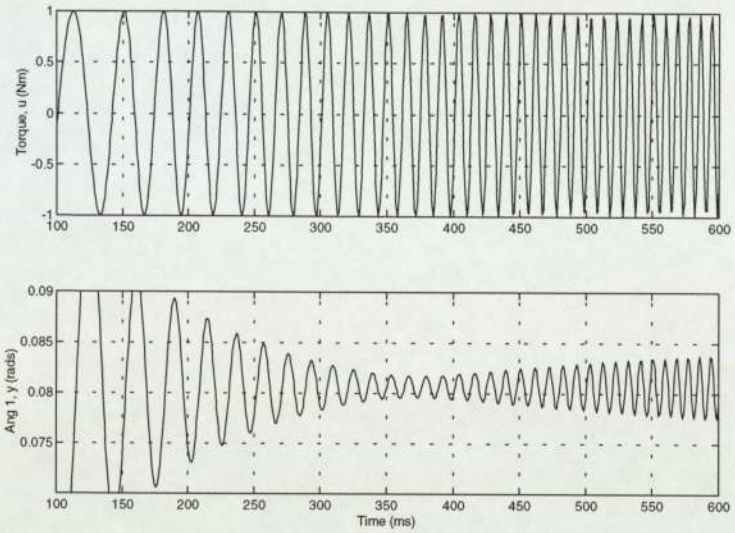


Figure 8.6 Simulation of the "Torsional Spring - Inertia" machine

The system output (angle 1) decreases with increasing input (torque) frequency and this mechanical system can be likened to an electrical LCR filter circuit.

8.3.3 Inertia - Torsional Spring - Inertia System

Section 3.3.2 describes a two degree-of-freedom mechanical system consisting of two damped inertias connected by a torsional spring, shown in figures 3.2 and 3.3. The input and output in this context are the torque and angle at one end of this system (for the sake of argument, the left-hand side). The system is described by equations 3.2 and 3.3, and put in state-space form in equations 3.14 and 3.15. The test-rig has the ability to measure θ_1 and $\dot{\theta}_1$ only, so for simulation the output equation

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \theta_1 \\ \dot{\theta}_1 \\ \theta_2 \\ \dot{\theta}_2 \end{bmatrix} \tag{8.10}$$

will be used, providing the necessary data for characterisation in chapter 9. A simulation of this system using a Matlab script (see A8) gives the machine's response to a swept-sine torque input, shown in figure 8.7 where $J_1=1.2\times10^{-3}$ Kg.m^2, $J_2=1.2\times10^{-3}$ Kg.m^2, $K_S=237$ Nm/rad, $B_1=0.5$ Nm/rad/s and $B_2=0.5$ Nm/rad/s).

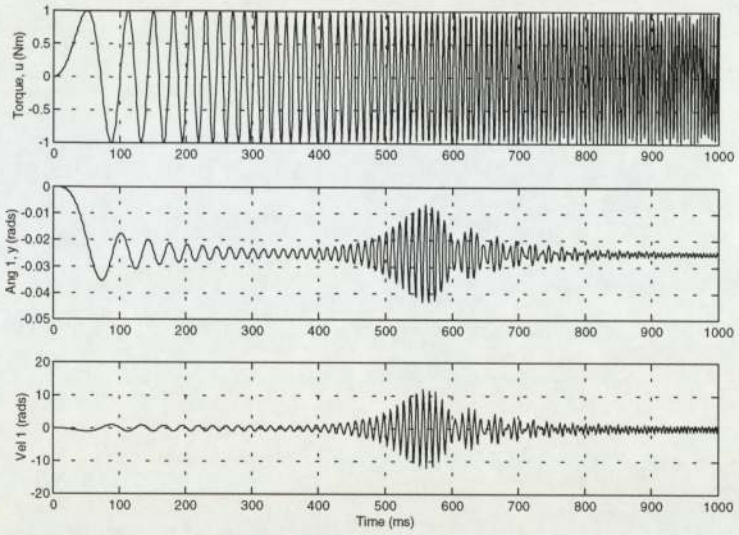


Figure 8.7 Simulation of the "Inertia - Torsional Spring - Inertia" machine

The response of this system to an impulse input is shown in figure 3.18. The resonant frequency using equation 6.7 is calculated to be 100Hz, and can also be seen from the graph to be approximately 100 Hz at t = 570 ms.

8.4 Electrical Analogues of Torsional Spring / Inertia Systems

Chapter 3 discussed the criteria for lumped parameter models and presented some analogous physical elements that suit the force-current analogy. An electrical equivalent of the example rotational system (discussed above in 8.3.3) is given in section 3.3.2.1. The electrical equivalent of the inertia is capacitance, and of the torsional spring is inductance. The through variable is current and the across variable is voltage (analogous to torque and angular velocity respectively). The integral of voltage is therefore analogous to the shaft angle, and is simple to perform numerically.

A linear voltage-controlled current-source was constructed analogously similar to the test-rig, which produces a current proportional to the applied input voltage. The current was not monitored because of the additional circuit complexity required, and in any case the current tracks the input voltage accurately. The current-source circuit is shown in figure 8.8. The current-source interface and variable-monitoring circuit is shown in figure 8.9.

The circuit was originally intended to be driven directly from the DSP board's DAC so that:

0V would correspond to –0.5A (0.5A sink),

2.5V would correspond to 0A, and

5V would correspond to +0.5A (0.5A source).

This was ultimately changed to –5V corresponding to –0.5A and +5V corresponding to +0.5A, and since buffering was required the input op-amp configuration of figure 8.9 was employed.

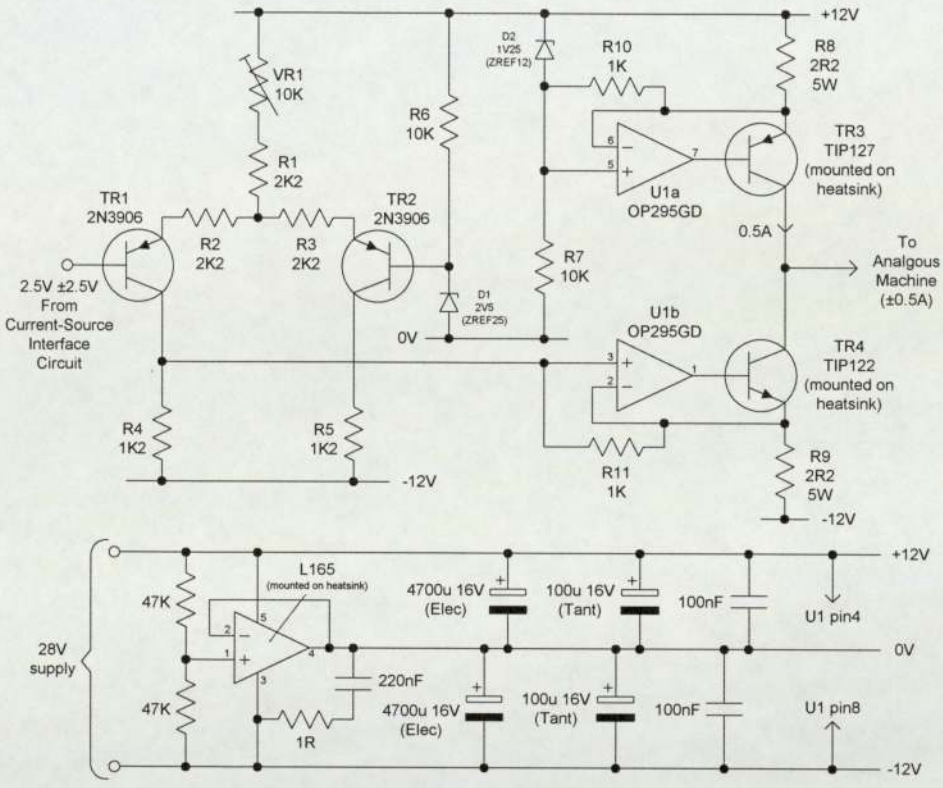


Figure 8.8 Current-source driver circuit

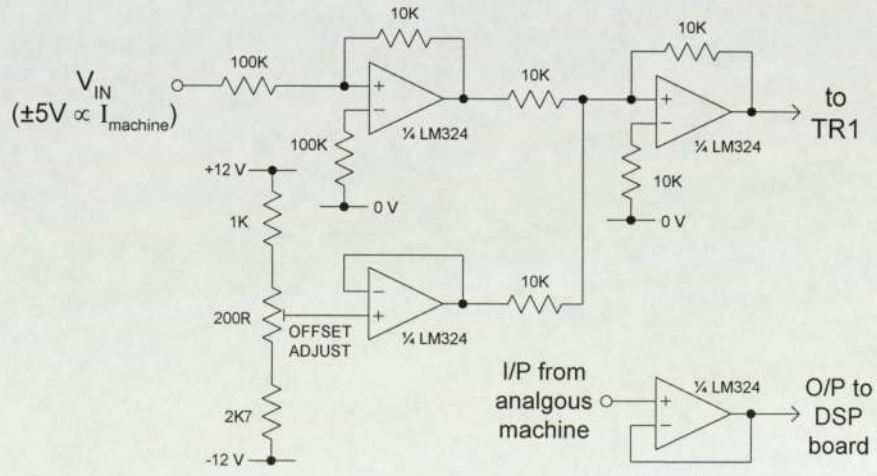


Figure 8.9 Current-source interface and voltage monitoring circuit

Current Driver Circuit Description

TR1 and TR2 are a long tail pair differential amplifier and the collector of TR1 is 1.25V ±1.25V above the –12V rail. U1b and TR4 form a voltage follower producing 1.25V ±1.25V across R9 which results in TR4 sinking 0.5A ±0.5A from the output node.

Constant Current Source

D2 produces 1.25V below the +12V rail. U1a and TR3 form a voltage follower producing 1.25V across R8 which results in TR3 sourcing 0.5A to the output node.

The net current in the load is therefore 0A ±0.5A.

TR3 and TR4 dissipate 6W with 0A O/P, 12W with ±0.5A O/P, and 24W under transient conditions. A regulated uni-polar supply (28V) was used, and U2 (L165) is responsible for providing a stable 0V between the two supply rails. The power dissipated by this device is 0W with 0A O/P and 6W with ±0.5A O/P. All three of these devices were therefore mounted on a heatsink with suitable cooling.

Figure 8.10 shows the actual construction of the current source (top half of prototyping board) with an analogous machine connected (lower half of prototyping board). The analogous machine is discussed in 8.4.1 below.

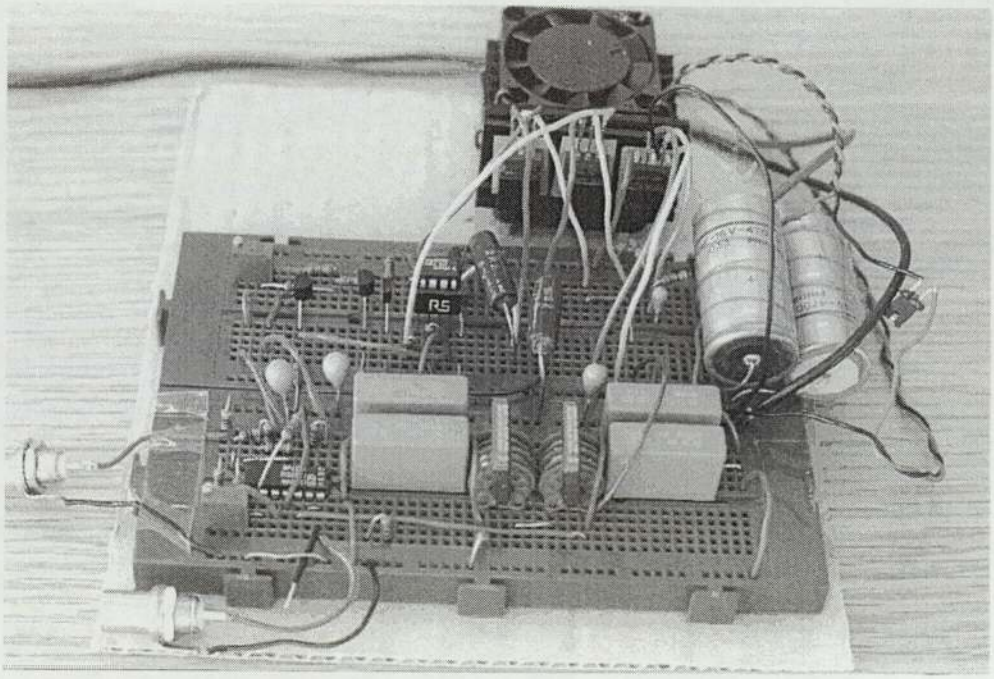


Figure 8.10 Physical construction of current source and analogous machine

8.4.1 Electrical Analogue of Inertia - Torsional Spring - Inertia System

The analogous circuit of figure 3.5 was constructed. Ideally components would have been chosen to accurately represent the mechanical machine of section 8.3.3, but due to "preferred values" and component availability the actual values used gave it a slightly higher theoretical resonant frequency of 202 Hz.

$R_1 = N/C$, $R_2 = 1\ K\Omega$, $C_1 = 6.6\ \mu F$, $C_2 = 6.6\ \mu F$, $L = 94\ mH$.

Mechanical components that are analogous to this electrical machine are
($B_1 = N/C$, $B_2 = 1.0$, $J_1 = 0.5\times10^{-3}\ Kg.m^2$, $J_2 = 0.5\times10^{-3}\ Kg.m^2$, $K_s = 403$).

For clarity the circuit diagram of this machine is shown in figure 8.11.

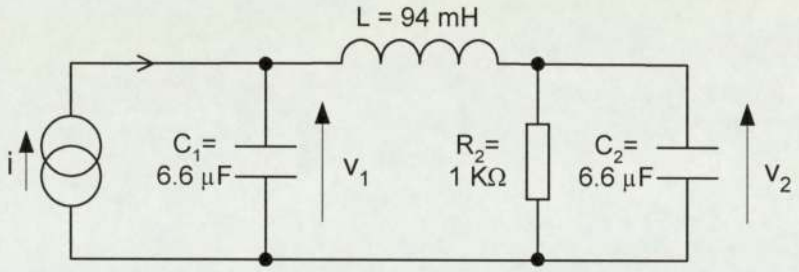


Figure 8.11 Electrical analogue of Inertia - Torsional Spring - Inertia system

DSP code was written to perturb the machine using an impulse, step, PRBS and swept sine signal, shown in appendix B.5. Figure 8.12 shows the response of the analogous machine to impulse and swept sine perturbation signals respectively. Data was sampled at 10 kHz, so the time period between data samples is 100 µs. The test parameters are defined in a file that is read by the DSP code, and are for the impulse and swept sine respectively is:

```
501             num_samples
2               num_channels
1               test_number
10              impulse_length
2.0             decades_per_second
10.0            start_frequency
1000.0          max_frequency
imptest         filename
```

```
10001           num_samples
2               num_channels
3               test_number
10              impulse_length
2.0             decades_per_second
10.0            start_frequency
1000.0          max_frequency
swpsn10K        filename
```

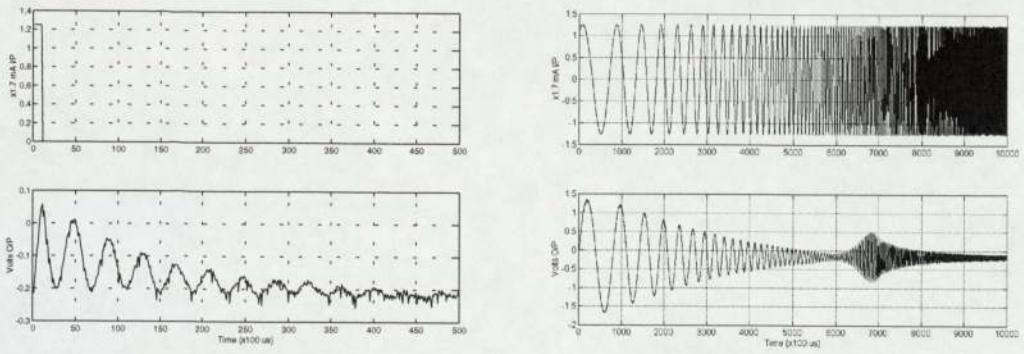


Figure 8.12 Response of the analogous machine to impulse and swept sine signals

The resonant frequency was found graphically to be 205 Hz which was higher than expected, but this may be due to variation in actual component values within component tolerances.

8.5 Slider-Crank Mechanism

Figure 8.13 is a diagram of the first example machine that has cyclically changing parameters, a slider-crank mechanism. This type of machine has a sliding mass connected to a rotating link. There are three turning pairs (1-2, 2-3, 1-4) and one sliding pair (3-4).

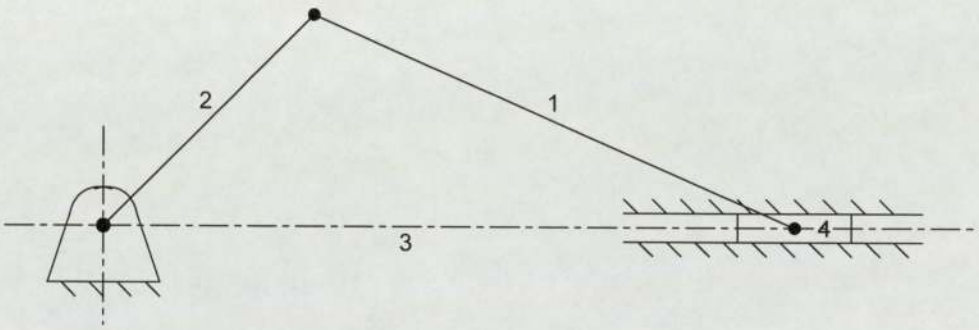


Figure 8.13 Slider crank mechanism conceptual diagram

Figure 8.14 shows an 'exploded view' of the same machine constructed at Aston. Detailed design drawings of this machine are in appendix D.1, and figure 8.15 shows the constructed machine.

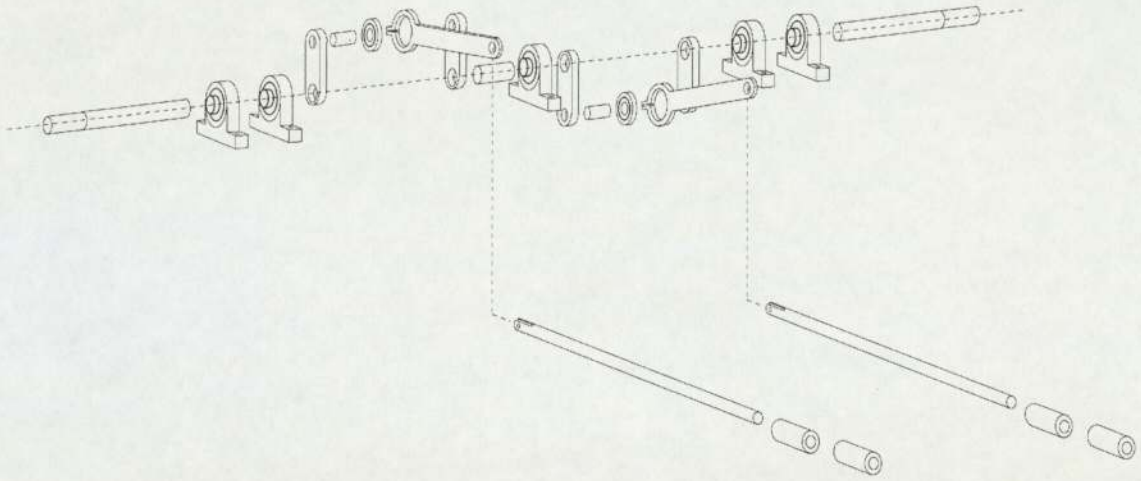


Figure 8.14 Slider crank mechanism constructional exploded view

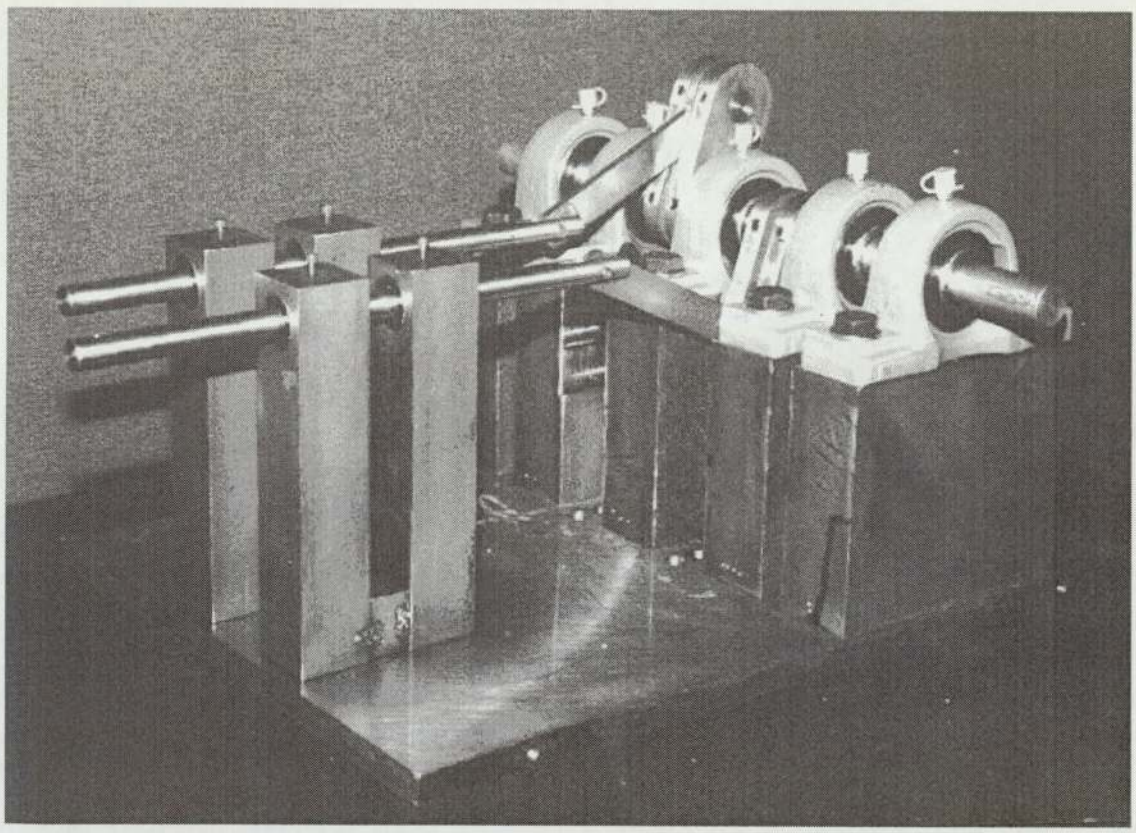


Figure 8.15 Slider crank mechanism constructed at Aston

This machine can be considered to have two very significant parameters, a constant inertia and a cyclically varying inertia, as well as some less significant parameters such as torsional stiffness, lumped viscous damping and possibly some friction.

The most intuitive way of representing the cyclically varying inertia is as a function of angle. In its simplest form the machine can then be represented by an equation of the form:

$$J_{var\_max} \cdot f(\theta)\frac{d^2\theta_2}{dt^2} + J_{const}\frac{d^2\theta_2}{dt^2} + B\frac{d^2\theta_2}{dt^2} - k(\theta_1 - \theta_2) = 0 \qquad (8.11)$$

However, if coefficients of a differential equation are made functions of state variables, inputs, or outputs, then the model becomes non-linear. This is not a problem for emulation where a model is given, but characterisation becomes much more difficult than for linear systems. This problem is discussed in more detail in chapter 9.

8.6 Four-bar Mechanism

Figure 8.16 is a conceptual diagram of a four bar mechanism built at Aston. Detailed design drawings are in appendix D.2. This type of machine has a varying inertia, and the linkages (bars) have independent modes of oscillation which are internal to the machine and detectable from the input shaft.

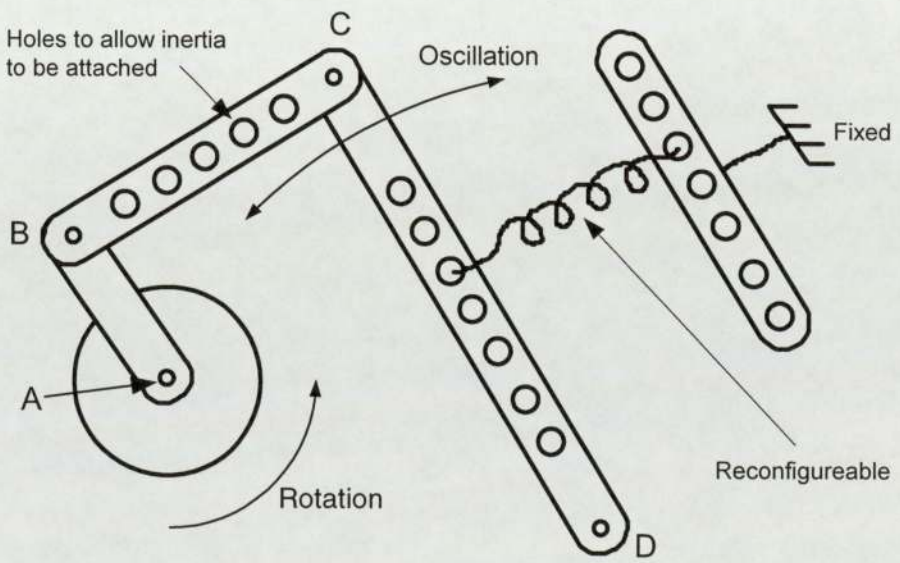


Figure 8.16 Four bar mechanism - conceptual sketch

Provision has been made for springs to be attached so that energy may be stored and released in a cyclic manner. Each bar has distinct resonance, and this may be calculated using the dimensions of the bar and the materials properties.

Strength is the force a material can withstand before breaking. Stiffness is a materials opposition to being distorted (flexibility $^{-1}$). Ductility is the ability of a material to being distorted (toughness $^{-1}$).

Stress is the force acting per unit cross-sectional area $= \frac{F}{A}$

Units = Pa (1 Pa=1 Nm^{-2}) (8.12)

where F = force and A = cross-sectional area.

Strain is the extension of unit length $= \frac{e}{l}$ (no units) (8.13)

where e = extension and l = original length.

When bending the bars there will initially be a linear region called elastic deformation, where the bar will resume to its original shape. After this, bending the bar further will result in non-linear deformation, called plastic deformation, where the bar will not resume to its original shape. The breaking stress (or ultimate tensile strength) is when the bar breaks. During elastic deformation the tensile strain is directly proportional to tensile stress. This is known as Hooke's law and can be written:

$$E = \frac{tensile\ stress}{tensile\ strain} = \frac{F/A}{e/l} = \frac{Fl}{Ae} \qquad \text{Units = Pa} \qquad (8.14)$$

E is a constant known as the Young's modulus and depends on the nature of the material; not it's dimensions. A material with large E resists elastic deformation strongly, for steel E = 21. Since the bars in this machine are of fixed dimensions and material, equation 8.14 can be rearranged for F:

$$F = \frac{A.E}{l} e \qquad (8.15)$$

where A, E, and l are constants. The resonance of each bar can then be calculated using the s.h.m. equation (similar to that in section 8.3.1.1):

$$f_0 = \frac{1}{2\pi} \sqrt{\frac{mass\ of\ oscillating\ system}{force\ per\ unit\ displacement}} \qquad (8.16)$$

The dimensions were chosen to use standard sized steel flats and to give natural frequencies that would be detectable. These frequencies worked out to be AB = 10 kHz, BC = 775 Hz, and CD= 258 Hz. Dimensional details are given in appendix D.2, and a script to calculate this frequency is in appendix A.9 (rsntng_bar.m).

The geometrical positioning of the bars was selected such that the machine would rotate, and the bar CD would be given a 'kick' to start it oscillating. Matlab script was written to simulate the movement of the machine with different arrangements of the bars, and the movement of the bars could be observed using animation. A plot of one such animation is shown below in figure 8.17, and the Matlab script for this is in appendix A.10.

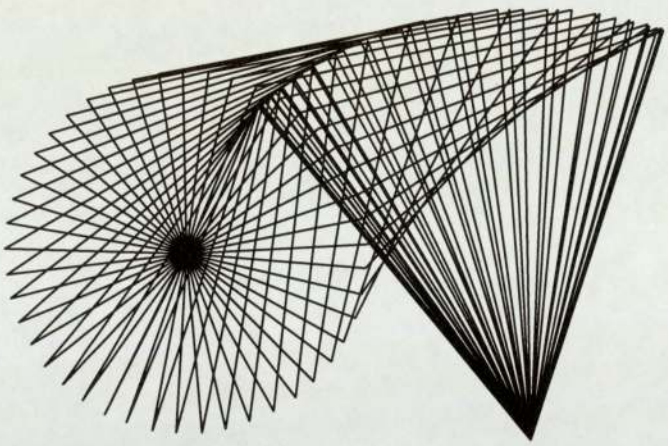


Figure 8.17 Plot of a 4-bar mechanism animation

This script also plots the angular movement, velocity and acceleration of bar CD around point D (refer to figure 8.16), and these plots are shown in figure 8.18.

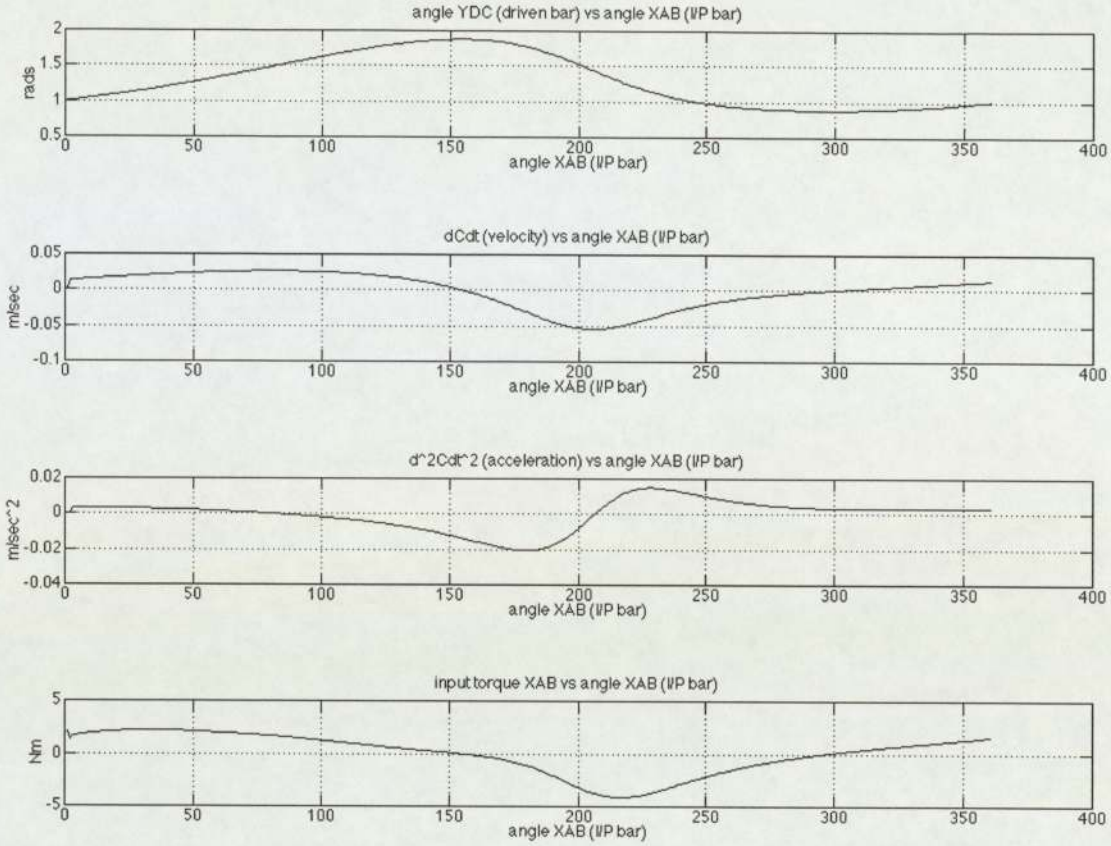


Figure 8.18 Plots showing simulated angular behaviour of bar CD and input torque, against a uniform input shaft angle.

The input torque is calculated ignoring gravitational forces acting on the machine. In practice there will be opposing and assisting torque every cycle due to gravitational pull on the 3 moving bars, which are more significant at lower velocities. The constructed machine is shown in figure 8.19, and constructional details are in appendix D.2.

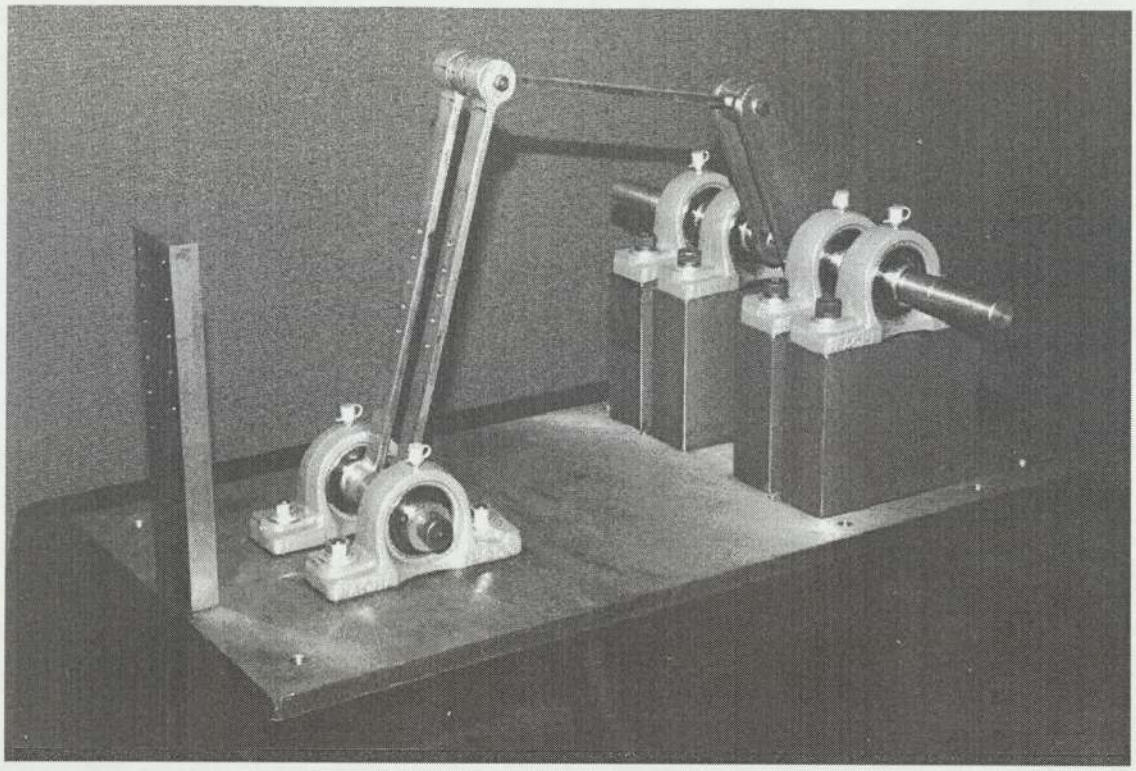


Figure 8.19 4-bar mechanism constructed at Aston

CHAPTER 9

9 Machine Characterisation & Emulation

This chapter discusses machine characterisation and emulation in the context of the test-rig, and gives examples using both simulations and experimental data. The example machines employed are those described in the previous chapter.

All characterisation experiments will be conducted off line i.e. the data will be analysed after the experiment has been performed. To collect and analyse data on-line (real-time) would require more sophisticated methods and is beyond the scope of this text. The characterisation problem can be thought of consisting of two parts. Firstly, sufficient data needs to be collected for an accurate enough model of the machine to be obtained. This requires all dynamic properties of interest in the machine need to be exercised, and the data collected to be of sufficient frequency and resolution. The frequency range of interest to the test-rig has been previously defined as 0 to 1kHz, so a sampling frequency of 10kHz for practical tests has been chosen to cover this range with sufficient accuracy. Perturbation signals were discussed in chapter 5, and the applicability of these is discussed in the context of each experiment. Secondly the information has to be extracted and represented. Chapter 5 outlines various methods of identifying a system, and again the applicability of these is discussed in the context of each experiment.

9.1 Adding noise to simulations

Real physical systems have noise on the measured signals, as discussed in section 3.9.5. This chapter uses a number of simulated models, and consequently the signals tend to be 'clean'. To make the data more realistic, noise is added subsequent to the machine simulation. Measured noise on real physical systems is generally random in nature and will often have frequency components much higher than those associated with the signal it corrupts.

If $a(t)$ is the pure signal, and $n(t)$ is the noise, then if $b(t)$ is the actual signal

$$b(t) = a(t) + n(t) \tag{9.1}$$

if T is the sampling period, then the mean-square of the actual signal is found by

$$\frac{\int_0^T b^2(t)\,dt}{T} \qquad \left(\equiv \frac{\sum_0^T b^2(t)}{T} \right) \tag{9.2}$$

$$= \frac{\int_0^T (a(t)+n(t))^2 \, dt}{T} \tag{9.3}$$

$$= \frac{\int_0^T a^2(t)\, dt + 2\int_0^T a(t)n(t)\, dt + \int_0^T n^2(t)\, dt}{T} \tag{9.4}$$

The $2\int_0^T a(t)n(t)\, dt$ term will tend to zero if $a(t)$ and $n(t)$ are uncorrelated in terms of phase, which is the case with random noise.

A Matlab function `addnoise()` was written that adds noise to a signal vector matrix. It takes two arguments, the signal and the noise factor. Given the signal vector, a noise vector is created of the same size containing uniformly distributed random values (noise). The noise vector is then scaled so that the mean-square of the noise is the correct proportion of the mean-square of the signal. The noise vector is then added to the signal vector and the function exits. The Matlab script for this function is in appendix A.11.

9.2 PE of Linear Time-Invariant Machines Using Least-Squares Methods

Section 5.4.2 outlined the LS and weighted LS methods. The **least-squares** method of parameter identification is simpler and easier to understand than many other techniques, and exhibits equally good statistical properties for most practical situations. This subsection works through this PE method longhand before using the Matlab *System Identification toolbox* to perform the same tasks automatically.

A system may be represented by:

$$\hat{y} = \theta_1\varphi_1(x) + \theta_2\varphi_2(x) + \ldots + \theta_n\varphi_n(x) \tag{9.5}$$

where $\varphi_1, \varphi_2, \ldots, \varphi_n$ are known functions, and
$\theta_1, \theta_2, \ldots, \theta_n$ are known parameters

If pairs of observations are made $\{(x_i, y_i), i = 1, 2, \ldots, N\}$, the parameters are required to be determined so that $\hat{y}$ computed from equation 9.5 and experimental values x_i agree as closely as possible. The LS method calculates values to minimise the loss function:

$$V(\theta) = \frac{1}{N}\sum_{i=1}^{N} \varepsilon_i^{\,2} \tag{9.6}$$

where $\varepsilon_i = y_i - \hat{y}_i$

$$= y_i - \theta_1\varphi_1(x_i) - \ldots - \theta_n\varphi_n(x_i) \qquad I = 1, 2, \ldots, N \qquad (9.7)$$

which is simply another way of expressing equation 5.41.

If the following matrices are defined:

$$\varphi = \begin{bmatrix} \varphi_1 & \varphi_2 & \ldots & \varphi_n \end{bmatrix}^T \qquad \Phi = \begin{bmatrix} \varphi^T(x_1) \\ \vdots \\ \varphi^T(x_N) \end{bmatrix}$$

$$\theta = \begin{bmatrix} \theta_1 & \theta_2 & \ldots & \theta_n \end{bmatrix}^T$$

$$y = \begin{bmatrix} y_1 & y_2 & \cdots & y_N \end{bmatrix}^T$$

$$\varepsilon = \begin{bmatrix} \varepsilon_1 & \varepsilon_2 & \cdots & \varepsilon_N \end{bmatrix}^T$$

then equations 9.6, 9.7 and 9.5 may be rewritten:

$$V(\theta) = \frac{1}{N}\varepsilon^T\varepsilon = \frac{1}{N}\|\varepsilon\|^2 \qquad (9.8)$$

$$\varepsilon = y - \hat{y} \qquad (9.9)$$

$$\hat{y} = \Phi\theta \qquad (9.10)$$

and similarly

$$y = \Phi\hat{\theta} \qquad (9.11)$$

If both sides are multiplied by Φ^T, then the following equation holds true:

$$\Phi^T\Phi\hat{\theta} = \Phi^T y \qquad (9.12)$$

where $\hat{\theta}$ are the estimated parameters and Φ are functions of x. If $\Phi^T\Phi$ is non-singular, the minimum is unique and is given by

$$\hat{\theta} = \left(\Phi^T\Phi\right)^{-1}\Phi^T y \qquad (9.13)$$

$\left(\Phi^T\Phi\right)^{-1}\Phi^T$ is said to be the pseudo-inverse of Φ if $\Phi^T\Phi$ is non-singular.

If data weighting is required, then equation 9.13 can be rewritten

$$\hat{\theta} = \left(\Phi^T W \Phi\right)^{-1}\Phi^T W y \tag{9.14}$$

Matlab has a built-in function `pinv()` that calculates the pseudo-inverse of a matrix. In the section that follows the above method is used to estimate the parameters for a linear time-invariant machine.

9.3 LS Estimate of an Inertia - Torsional Spring - Ground system

The Matlab script described in section 8.3.1 simulates the Inertia - Torsional Spring - Ground system, and saves the simulation data in a file. Only the perturbation torque, response angle and velocity, and period are stored, which is all that would be available from measurements on the test-rig.

The Matlab script that performs this parameter estimation is in appendix A.12. The machine simulation was modified to make the period 1×10^{-4} s (simulated sampling rate T = 10 kHz), and add more simulation steps. This makes it similar to the test-rig, and the larger data set allows a more accurate parameter estimation to be performed. 2% noise is added to the perturbation torque, response angle and velocity signals, to simulate noise which may normally be present in the measurements of a practical experiment. Equation 8.2 describes the machine. Rearranging and labelling in the context of the Matlab script gives

$$T_R = J\ddot{\theta} + B\dot{\theta} + k_s\theta \tag{9.15}$$

Since $\ddot{\theta}$ is not available, this equation needs to be integrated:

$$\int T_R = J\dot{\theta} + B\theta + k_s\int\theta \tag{9.16}$$

The data is rearranged and integrated which leaves data of the form

$$\int T_R = \begin{bmatrix} \int T_{R1} \\ \vdots \\ \int T_{Rn} \end{bmatrix} \qquad x = \begin{bmatrix} \int \theta_1 & \cdots & \int \theta_n \\ \theta_1 & \cdots & \theta_n \\ \dot{\theta}_1 & \cdots & \dot{\theta}_n \end{bmatrix} \qquad \text{so} \qquad x^T = \begin{bmatrix} \int \theta_1 & \theta_1 & \dot{\theta}_1 \\ \vdots & \vdots & \vdots \\ \int \theta_n & \theta_n & \dot{\theta}_n \end{bmatrix}$$

where n is the sample number. A plot of this data is shown in figure 9.1, where the x-axis is time (ms).

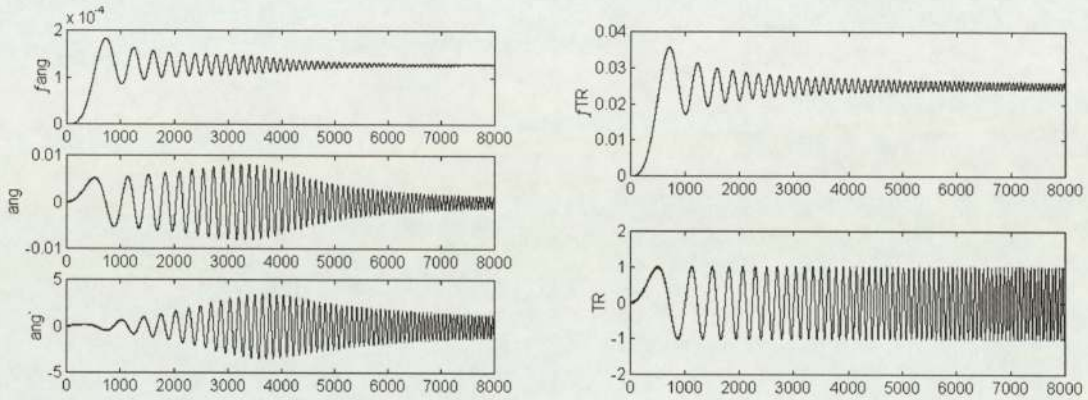


Figure 9.1 Manipulated data from the "Inertia - Torsional Spring – Ground" machine

The remaining problem is to find a solution to the equation

$$\int T_R = C_0\left(\int \theta_1\right) + C_1\left(\theta_1\right) + C_2\left(\dot{\theta}_1\right) \tag{9.17}$$

where C_0, C_1 and C_2 are the parameters and correspond to k_s, B and J respectively.

Since the numerical size of the data values effects its contribution in the estimation of the parameters, it is often beneficial for the data values to be weighted so that they all carry the same weight, i.e. they need to be normalised. This is performed by making a diagonal matrix containing the sum of the squares of the coefficients in each equation in x

$$d = \begin{bmatrix} \sqrt{\int \theta_1^{\,2} + \theta_1^{\,2} + \dot{\theta}_1^{\,2}} & 0 & \cdots & 0 \\ 0 & \sqrt{\int \theta_2^{\,2} + \theta_2^{\,2} + \dot{\theta}_2^{\,2}} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \sqrt{\int \theta_n^{\,2} + \theta_n^{\,2} + \dot{\theta}_n^{\,2}} \end{bmatrix} \tag{9.18}$$

and then dividing x and T_R by d. Equation 9.17 can be rewritten to combine the parameters (C_0, C_1, C_2) into one matrix,

$$\int T_R = C \times x^T \tag{9.19}$$

where C comprises the parameters. The roots of this equation 9.17 are found by

$$C = pinv(x^T) \times \int T_R \tag{9.20}$$

Running the parameter estimation script five times produces:

	K_s	B	J
Actual parameters	197.3921	0.31	1×10^{-3}
1st run	199.14035	0.301427	0.000992
2nd run	197.59914	0.300749	0.000992
3rd run	198.81667	0.297941	0.001030
4th run	196.56652	0.307889	0.001003
5th run	197.36450	0.301863	0.001038

This demonstrates that the parameters are estimated within reasonable accuracy about the true values, and that the least-squares method produces good results for this type of problem.

9.4 LS Estimate of Inertia - Torsional Spring - Ground Using Matlab Toolbox

Section 9.3 demonstrated how to estimate parameters of a simple system using a 'long-hand' method. This section performs the same task using the Matlab System Identification Toolbox. The toolbox contains functions to estimate parameters to system models where the model structure is both known and unknown, view the data, and validate the models subsequent to identification.

The Matlab System Identification toolbox uses a matrix *theta*, which is common to most of its functions, and contains information about model structure, estimated

parameters, their estimated accuracy and other information. Most of the toolbox functions operate on this matrix, and it can be converted to more familiar forms and between continuous time and discrete time using model conversion functions. The internal representation of the theta format is specific to Matlab, the details of which are unimportant.

Two data sets were created using the same simulation script as the last section and adding noise to the data in the same way. The data sets were created at different times and are therefore unique since the noise is of random nature. One of these data sets (`est_dat.mat`) will be used for parameter estimation and the other (`val_dat.mat`) for model validation.

The Matlab System Identification toolbox is designed to operate on SISO systems and systems with many input and/or many output signals (called multi-variable), but these variables are treated as being separate entities. It is therefore not possible to use these toolbox methods using an input signal and its derivative. The system models being created are SISO systems where the input in torque and the output is angle. Angular velocity is therefore not strictly required for identification, and is only used for control purposes (discussed in chapter 4).

The Matlab script for this section is in appendix A.13 (ident.m). Firstly the data is loaded, and the input-output data is merged into a column matrix. This can be viewed using `idplot()`, and is shown in figure 9.2.

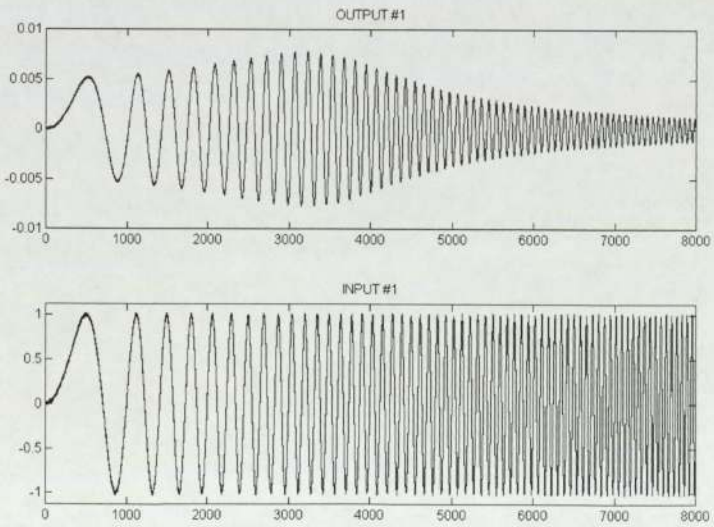


Figure 9.2 Data used for parameter estimation prior to processing

The data is then filtered to remove frequencies higher than that of interest, and the constant levels are removed to make the data zero mean. The model structure is known, and is continuous-time of the following form

$$a_0 y + a_1 \frac{dy}{dt} + a_2 \frac{d^2 y}{dt^2} = b_0 y + b_1 \frac{du}{dt} + b_2 \frac{d^2 u}{dt^2} \qquad (9.21)$$

where b_0 and b_1 are both equal to zero. The data was produced using a (simulated) continuous-time model but the Matlab identification functions estimate parameters for discrete-time models only. The data was initially fitted to a discrete-time model of the form

$$a_0 y(t) + a_1 y(t-T) + a_2 y(t-2T) = b_0 u(t) + b_1 u(t-T) + b_2 u(t-2T) \qquad (9.22)$$

where b_0 and b_1 are both equal to zero. The function `arx()` was used to perform this operation and the discrete-time model parameters a_0, a_1, a_2 and b_1 were estimated to be 1.0, -1.9694, 0.9714 and 0.9898×10^{-5} respectively. The equivalent continuous-time parameters for equation 9.21 are $b_0 = 1$, $b_1 = 0$, $b_2 = 0$, $a_0 = 195.5426$, $a_1 = 0.2888$, and $a_2 = 0.001$, where a_0, a_1 and a_2 are approximately the values of ks, B and J from the original simulation.

Both the parameter estimations described above have estimated the damping coefficient lower than it actually is, and this is because the original machine simulation is inaccurate. The calculation of the state vector (x) uses the Euler method, which essentially calculates the state vector using the old value, the slope and the step size. This is inaccurate since the magnitude of the predicted value is always greater than the true value, and a simulation using this method will have inherent negative damping. More accurate methods are available, and Matlab had a function called `lsim()` which is accurate. The for loop that simulates the machine within smpl_ld can therefore be replaced with the following line for a much more accurate simulation;

```
[y, x] = lsim( A, B, C, D, u, times, x0);
```

Using this more accurate simulation to produce a new data set and then performing the same identification as before now yields the parameters 195.2176, 0.3191, and 0.0010. It can be seen that the damping is now much closer to its true value.

Using new data for model validation (`val_dat.mat`), the estimated model can be simulated using this input data, and the output of this compared to the original.

It is desirable to evaluate how well the model fits the data. A simple test is to run a simulation whereby real input data is fed into the model, and the output of this simulation compared to the actual output. For this comparison new data is used (`val_dat.mat`) which was not used to build the model, and the Matlab function `idsim()` used to simulate the estimated system. Plotting the two system outputs on the same graph produces the plot shown in figure 9.3. Since the two are almost identical it is difficult to distinguish between the two.

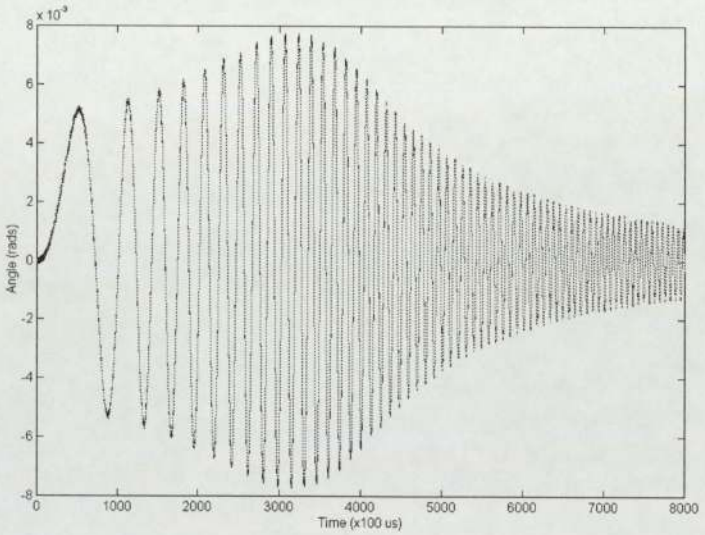


Figure 9.3 Comparison between actual (solid) and simulated (dotted) outputs

9.5 Characterisation of the Electrical Analogous Machine

The electrical Capacitor – Inductor – Capacitor system, analogous to the Inertia – Spring – Inertia system, is a real physical system and is more appropriate for identification in this text since the data collected is real and contains real noise. The circuit of this system is described in section 8.4.1 and the method of data collection also.

Identification was initially performed using a sweep-sine perturbation input to the system, the same input as shown in figure 9.2. Two identical tests were made and the data was saved as "swpsn_e" and "swpsn_v" for estimation and validation respectively. Figure 9.4 shows the actual output from the electrical system to this input signal.

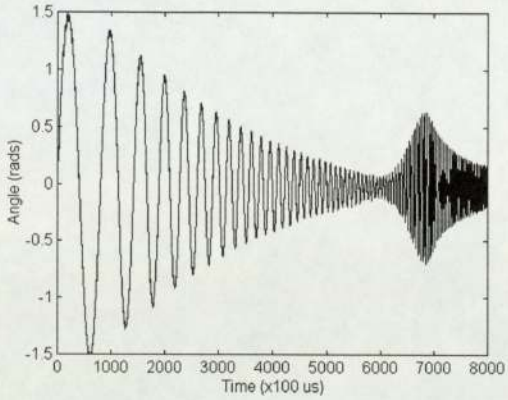


Figure 9.4 Output from physical electrical system

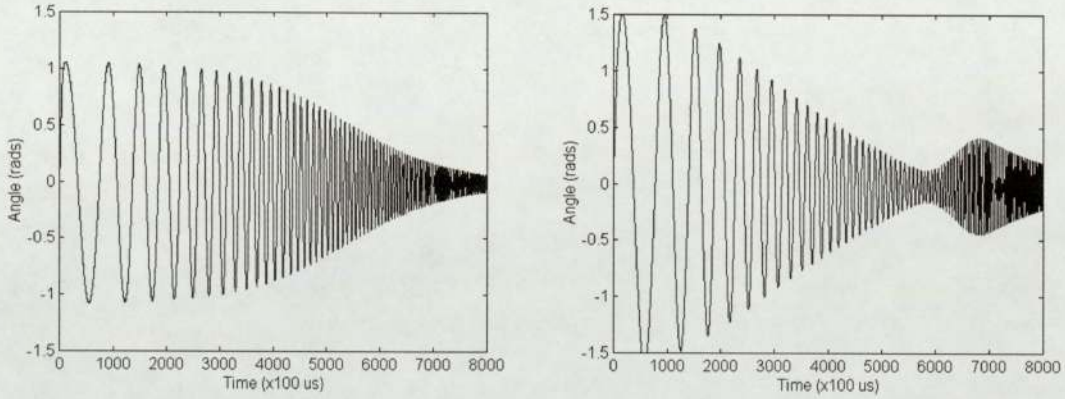


Figure 9.5 a, b; Outputs from identified system for different sized ARX models

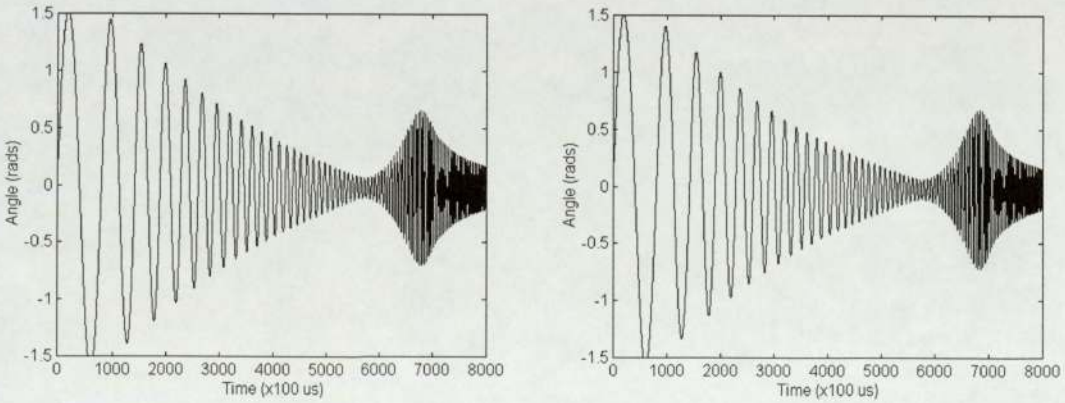


Figure 9.5 c, d; Outputs from identified system for different sized ARX models

The `arx()` function was used to fit the data to a number of different ARX models using the LS estimate. The models were then simulated using the sweep-sine perturbation signal and the simulated output compared with the output from the physical system. Model structures of different orders were investigated and some of the more interesting ones are shown in figure 9.5.

Figure 9.5a shows the simulation of an ARX model with two output coefficients, a_1 and a_2, ($a_0 = 1$ because the equation is normalised), and two input coefficients, b_1 and b_2, ($b_0 = 0$). This model is clearly of insufficient order to capture the dynamics of the system being identified. Figure 9.5b shows the simulation of an ARX model with four output coefficients, and four input coefficients. This model captures the input - output relationship in more detail, but a better model is specified using a model with six output coefficients and five input coefficients, shown in figure 9.5c. Figure 9.5d is a plot of an estimated model using eight output coefficients and seven input coefficients. Using the sweep-sine input signal it is apparent that this higher-order model contains no additional useful information about the machine. Using a higher-order model than necessary can often allow undesirable characteristics to be included in the model such as noise contributions, and the model is then described as *over-fitted*.

The model using six output coefficients and five input coefficients is of higher-order than necessary to describe the electrical machine, and this can be due to a number of reasons. Firstly the system may be more complex than the circuit shows due to parasitic L, C and R in the circuit, and non-ideal behaviour of the current drive. Secondly the LS identification process is not ideal, and may over-fit a model where noise is present, particularly if the noise is non-white. This is demonstrated by using the LS method to identify a simulated machine with a) no noise present, and then b) with noise present. The simulated machine of section 8.3.3 is used to generate data.

If no noise is added to the data the `arx()` function will fit a good model using 4-input and 4-output coefficients – the number of parameters required to correctly describe the system. Figure 9.6a shows the output from the simulated system (solid lines) and a simulation of the estimated system using a different data set with added noise (dotted lines). Since the two are identical it is difficult to distinguish between the two. If noise is added then the estimation is poor (figure 9.6b), and to regain a model that describes

the system to the accuracy of figure 9.6a requires six output coefficients and five input coefficients. The same number as the real electrical above. This suggests that it is the limitation of the LS identification procedure that is responsible for the necessary over-fitting of the ARX model, and not the imperfections in the physical machine.

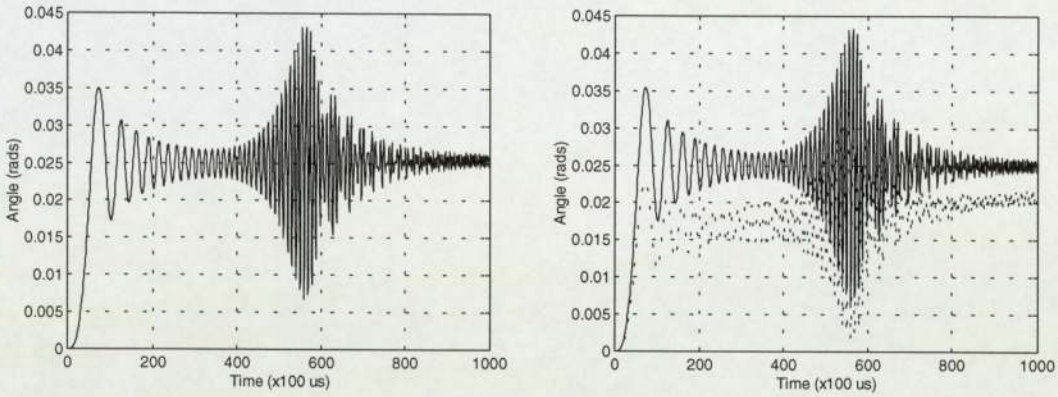


Figure 9.6 a, b: Comparison between simulation (solid) and estimated (dotted) model outputs: 4 input & 4 output parameters identified from data with a) no noise, b) noise

The ARX models created are discrete-time models of continuous-time systems. To represent a continuous-time model exactly in discrete-time would require an infinite number of input and output terms in the series. This is both impractical and undesirable (for reasons stated above), and by using a finite number of terms sufficient detail can be captured. The characteristics of interest to this experiment are the dynamics of the machine. Since the time constants of the machine are relatively low the signals were pre-processed to filter out components above 1 kHz, prior to experimenting with model structures of different orders. This was effective for the swept-sine perturbation and output signals because any frequency content above this is known to be noise.

9.5.1 Characterisation of the Electrical Analogous Machine using PRBS

The same electrical system was also characterised using a PRBS perturbation signal as discussed in section 8.4.1. Figure 9.7a shows the first 200 steps (20 ms) of the unfiltered input-output data. A disadvantage of using the electrical system described in section 8.4 is that the current is assumed to follow the reference voltage accurately in the absence of measured feedback. The swept-sine signal is likely to be followed accurately because the frequency content is limited. The PRBS signal theoretically has an infinite frequency

content, but in practice the circuit will filter this signal and 'round off' the edges. The input data is therefore not strictly true, and only by filtering this signal in the same manner as the driver circuit is an accurate input-output data set obtained. Filtering the data through guesswork is likely to remove important information relating the output to the input, and is shown in figure 9.7b. Identification using this input-output data is therefore less likely to be as accurate as swept-sine test, or will involve a model of a higher order to include noise dynamics.

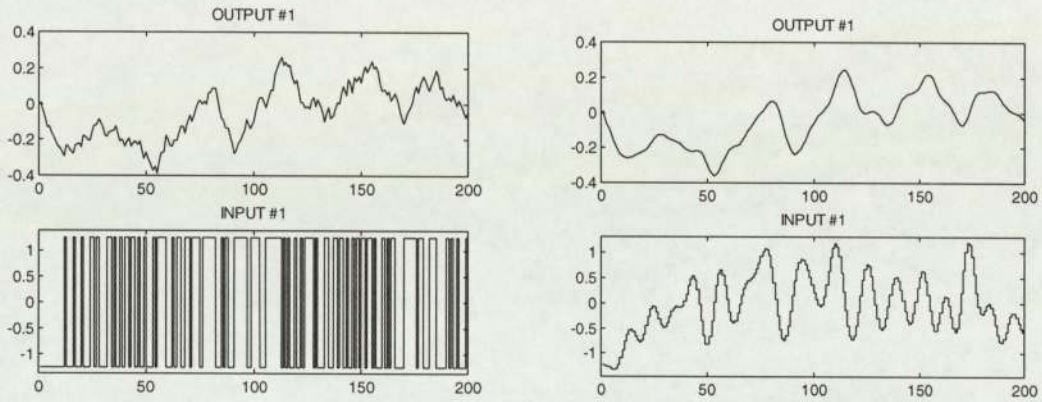


Figure 9.7 a, b; Unfiltered and filtered PRBS input-output Signals

Typically analysing data gives rise to a large collection of models. Choosing the best model, called model validation is discussed in chapter 5, and for this experiment is conducted in the next section.

9.5.2 Electrical Analogous Machine Model Validation and Selection

There is no absolute procedure for validating models, but usually they are tested using a different set of data then that used for estimation, and evaluated using some kind of criterion. The Matlab Identification toolbox has several functions for comparing different structures which are used here. Appendix A.14 contains the script "idnt_arx.m" referred to in this section, the important part of which is shown below:

```
V = arxstruc(ze, zv, struc(2, 2, 1:5));      % create arx's of different delays
nn = selstruc(V,0);                           % establish suitable delay value
nk = nn(3);                                   % extract delays
V = arxstruc(ze, zv, struc(1:20, 1:20, nk-1:nk+1));
                                              % test all combinations of arx
                                              % models with up to 20 a & b params
                                              % with delays around selected value
nn = selstruc(V)     % plot fit vs number of selected parameters
```

```
th = arx(ze, nn);    % fit data to selected model to calculate parameters
th = sett(th, T);    % set the sampling interval
```

The identification is performed on data collected using a PRBS perturbation signal, and validated using the swept-sine perturbation signal used in the previous section. This is called cross-validation and is discussed in section 5.8.3. The `arxstruc()` function fits a range of ARX models specified in its third argument, to the estimation and validation data in the first and second arguments respectively. The sum of squared prediction errors is computed as applied to the validation data, and the resulting loss functions are stored with their corresponding structures in the matrix *V*. The function `selstruc()` selects the most appropriate structure according to a criterion specified in the second argument. These are discussed in section 5.8.3. Akaike's Final Prediction Error (FPE) and Information Theoretic Criterion (AIC). Both simulate a cross-validation situation where only one set of data is available. Rissanen's Minimum Description length (MDL) selects the structure that allows the shortest overall description of the observed data. When substantial noise is present, ARX models need to be of high order to describe the system dynamics and the noise dynamics, since they are directly coupled in this type of model (refer to figure 3.14). Alternatively the dynamic model only may be computed by using the IV method to fit the same data to an ARX model of lower order.

The two scripts "idnt_arx.m" and "idnt_iv4.m" find the model which fits the data to an ARX model, and are identical, except "idnt_iv4.m" uses the IV counterpart of the `arxstruc()` function, `ivstruc()`. Figure 9.8 shows the output data from the electrical system using the PRBS and swept-sine tests.

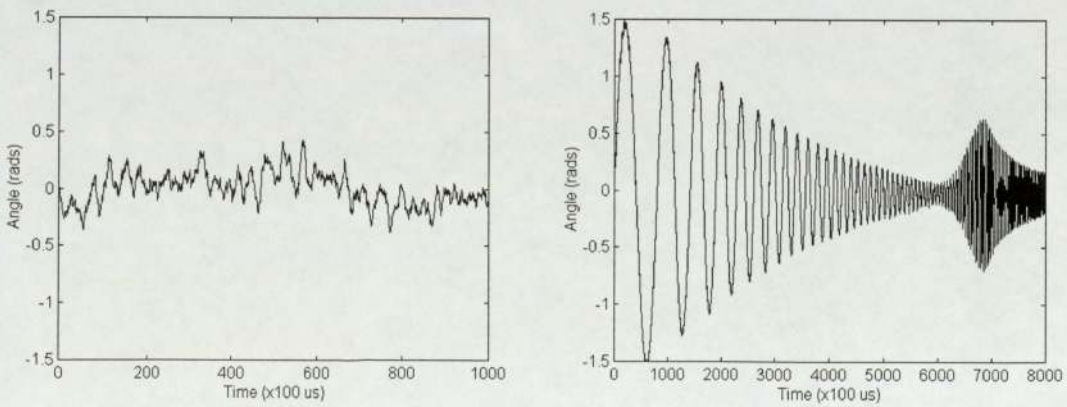


Figure 9.8 a, b; Output data from physical PRBS and swept-sine tests

Five second-order ARX structures are created with different delays, and the best fitting of these determines the centre delay for the fitting of ARX models with up to twenty *a* and *b* parameters. This equates to 1200 models, which are plotted on a graph showing loss function vs the number of parameters. The default models returned have the lowest loss function, and from using LS and IV methods are models of 16 *a* and 18 *b* parameters, and 3 *a* and 1 *b* parameters respectively. Figure 9.9 a and b shows the swept-sine output from the LS and IV derived simulated models respectively.

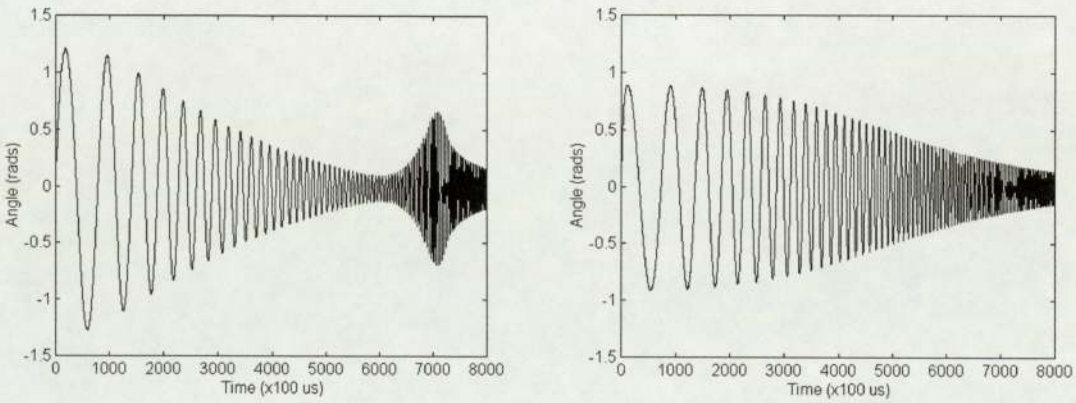


Figure 9.9 a, b; Output data from simulated PRBS and swept-sine tests using LS

The high-order model estimated by the LS function fits the data well, but a better model was estimated at the beginning of section 9.5 using only six output coefficients and five input coefficients (shown in figure 9.6c). It would appear therefore that the model is over-fitted. Noise is being represented in the model and although the validation data is reproduced accurately, this would not necessarily be the case for validation data with different characteristics.

The model estimated using the IV method is of much lower order, but does not represent the machine very well at all. This is probably because the IV function is attempting to differentiate between the machine dynamics and the noise dynamics and there is either insufficient machine dynamics information, or perhaps the two signals are correlated in some way.

The PRBS data set is likely to contain much less machine information than the swept-sine data set of the previous section leading to less concise models. Using the PRBS perturbation signal has probably not exercised the dynamics of the machine sufficiently,

or has exercised the dynamics of the machine sufficiently but generated excessive noise in doing so.

9.5.3 Non-Parametric Characterisation of the Electrical Analogous Machine

Section 5.3 discusses the theoretical background to non-parametric system identification methods. The Matlab SI toolbox contains several functions to perform this kind of analysis. The Matlab script shown in appendix A.15 (nonpar.m) performs several non-parametric analyses. Analysis is conducted on the swept-sine data collected from the electrical system test (swpsn_e).

The function `cra()` performs correlation analysis and plots the results on a graph. The input and output are filtered so that they are as uncorrelated as possible and their covariance functions plotted. The correlation function between these filtered signals is then plotted, which can be used to check the correlation between the filtered output and input. Finally this same correlation function is scaled so that it is an estimate of the systems impulse response, and plotted with 99% confidence level lines (dashed).

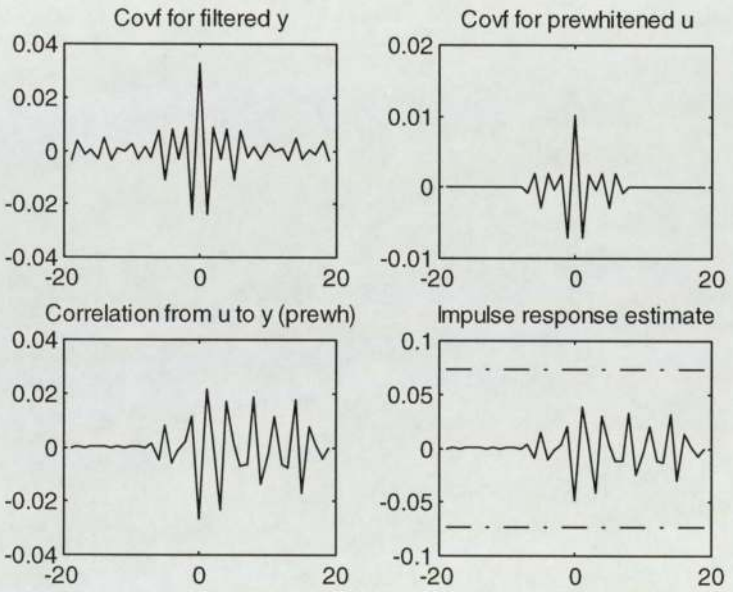


Figure 9.10 Correlation analysis of the electrical system

The function `spa()` performs spectral analysis on the output-input data. Two matrices are returned, the estimated frequency function and the estimated disturbance spectrum. Figure 9.11 is a plot of the estimated frequency function plotted with linear frequency scales, and figure 9.12 is a plot of the estimated disturbance spectrum plotted with linear frequency scales.

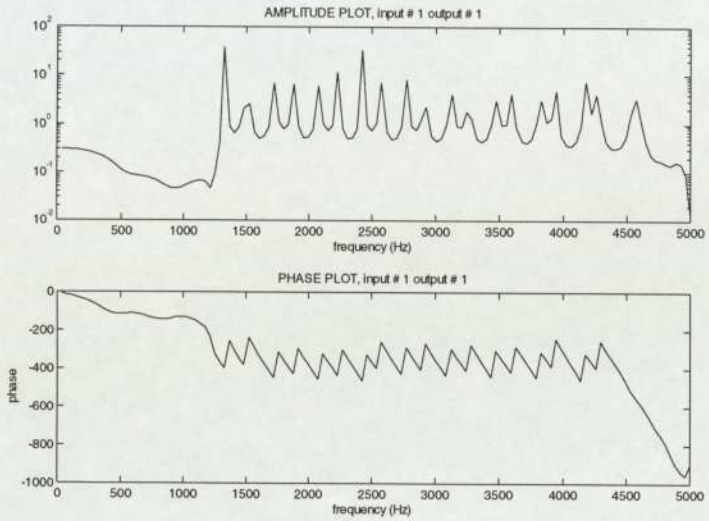


Figure 9.11 Estimated frequency function (linear frequency scales)

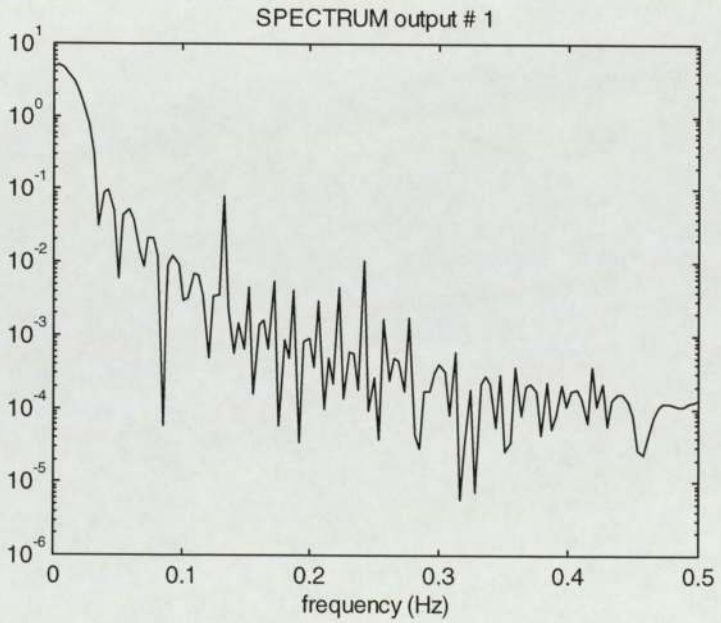


Figure 9.12 Estimated disturbance spectrum (linear frequency scales)

9.6 Emulation of the Electrical Analogous Machine

The physical parameters could be extracted in a similar manner to section 5.4, but this in not necessary since the model will be used directly for emulation. The optimal model order was obtained in the previous section using the script 'idnt_arx.m'. Creating a model to fit the data is performed in Matlab using the command `th = arx(ze,nn);` where ze is the data and nn specifies the model order. The polynomial coefficients are calculated using `[A B] = th2arx(th).`

The control software used to perform the emulation is described in section 4.5. The model was written in C code, and is a polynomial containing the coefficients returned:

```
/* calculate model torque dem */
#define am0   1.00000000000000
#define am1  -5.25556352321326
#define am2  11.74504443782740
#define am3 -14.27965314019990
#define am4   9.93854432348819
#define am5  -3.73848816027420
#define am6   0.59013408221406
#define bm0   0.00000000000000
#define bm1   0.01276541800102
#define bm2  -0.04658864455239
#define bm3   0.06708665896887
#define bm4  -0.04532714871371
#define bm5   0.01209339513882
mod_dem = bm0*torq[t] + bm1*torq[t-1] + bm2*torq[t-2] + bm3*torq[t-3] + bm4*torq[t-4] +
                                                                        bm5*torq[t-5]
        - am1*ang[t-1] - am2*ang[t-2] - am3*ang[t-3]  - am4*ang[t-4]  - am5*ang[t-5] -
                                                                        am6*ang[t-6];
```

A swept-sine signal was generated and fed via a spare DSP output to a drive-motor pair mechanically connected to the test-rig. Figure 9.13 shows the mechanical configuration.

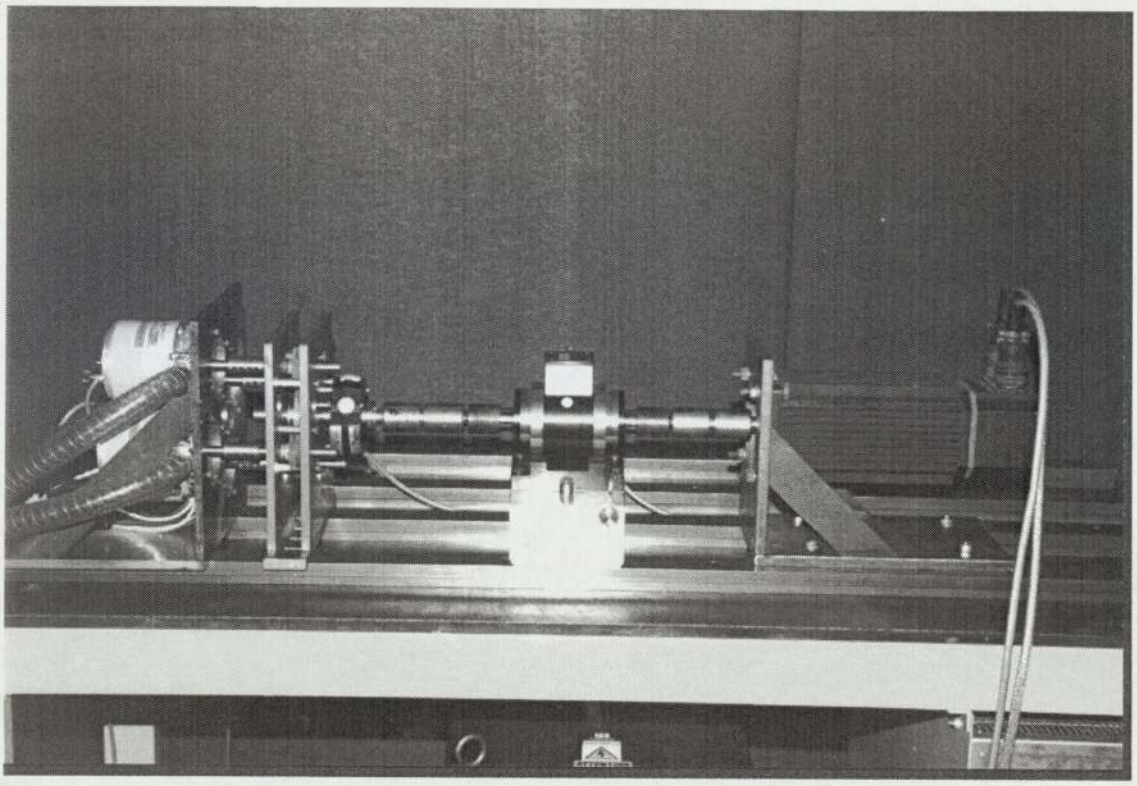


Figure 9.13 Drive-motor pair connected to test-rig to drive emulated machine

The test was run for 50,000 samples, at a sampling frequency of 10kHz, for a total time of 5 seconds. The data was then filtered to remove unwanted noise above 500Hz, and a plot of this data is shown in figure 9.14.

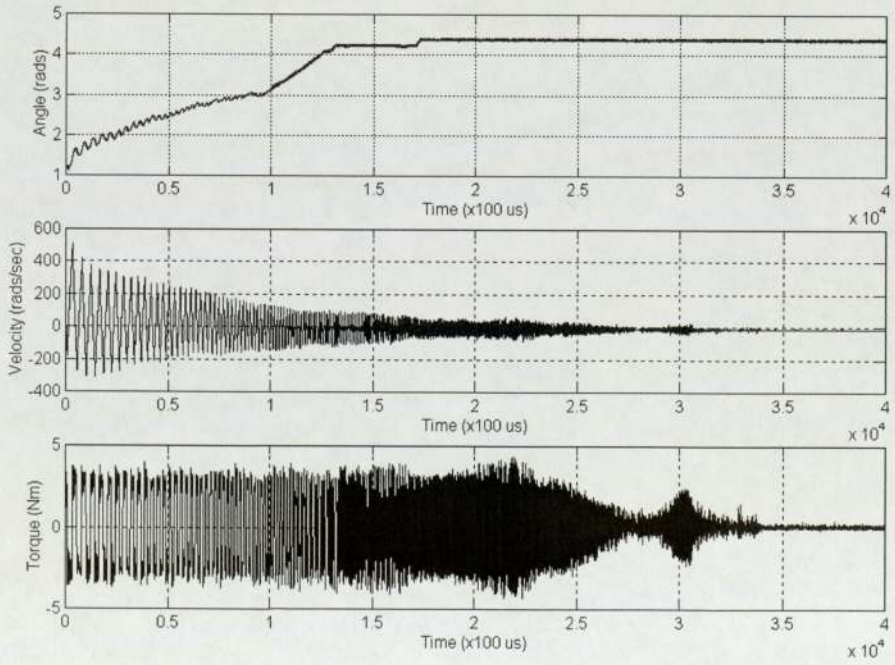


Figure 9.14 Plot of the measured data obtained from emulation test

The angle data is disappointing as it reveals very little about the oscillation of the shaft. The torque however shows the dynamic response of the test-rig emulation, and is comparable to the electrical analogous tests. Figure 9.15 is a close-up of the torque plot showing the model dynamics, and is comparable to the plot of the electrical-analogue model response to the same test. It is important to note however that the angle plot of figure 9.14 does not show much change at resonance. It is therefore possible that what is actually seen is the resonance of a system like that shown in figure 3.2 where one inertia is the test-rig, and the other is the external motor, and the non-rigid shaft in-between is the couplings and gear-train between the two.

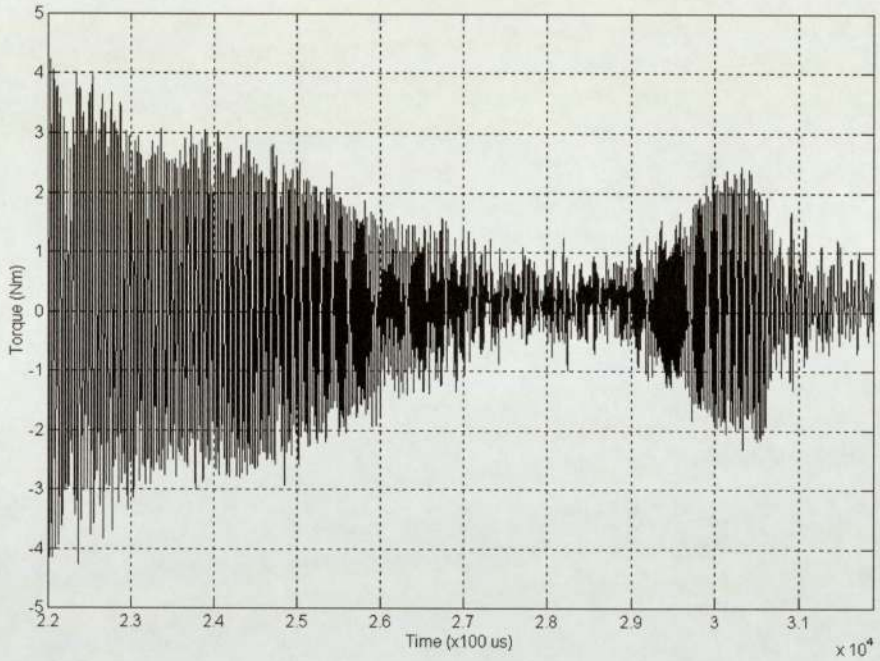


Figure 9.15 Close-up of measured torque showing dynamics of interest

Using the close-up (figure 9.16) of the resonant part of the emulated model (figure 9.15), the resonant frequency can be calculated to be 345 Hz ($1/((8245-8216) \times 100\times10^{-6})$) since this is the reciprocal of the period.

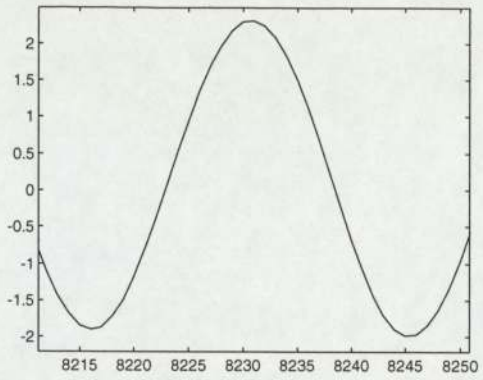


Figure 9.16 Close-up revealing resonant frequency of the emulated machine

9.7 Characterisation of Cyclic Machines

Chapter 5 discusses the structure of cyclic machines and some of the tests required to identify certain aspects of these machines. It is preferable to represent the varying parameters as functions of time, leading to time-variant and linear models, but it is also convenient to represent the varying parameters as functions of angle, leading to non-linear models.

9.7.1 Characterisation Tests for Cyclic Machines

Chapter 5 outlined four tests to characterise a time-varying machine:

1) A "quasi-stationary" (or static) test to determine friction in both the forward and backward directions, $T_{QSF}(\theta(t))$ and $T_{QSR}(\theta(t))$.

2) A "constant-velocity" test for a range of different velocities to determine the viscous drag component (B), and inertia variation component (G).

3) A sinusoidal excitation of the machine at frequency ω with the machine turning at a much lower frequency Ω_{mean} ($<<\omega$), to determine the actual value of the inertia.

4) An additional test was described to characterise internal resonance's that can occur in machines incorporating some flexibility. Additional state vectors are required $\{\boldsymbol{q}, \dot{\boldsymbol{q}}\}$, which describe a displacement from a reference position. Equation 5.100 describes the

shaft torque for a non-constant velocity. It is made up of four terms and is repeated below for convenience

$$T_{\Omega}(\theta_{ref}(t)) = T_{QSF}(\theta_{ref}(t)) + \Omega.B(\theta_{ref}(t)) + \Omega^2.G(\theta_{ref}(t)) + J_P\ddot{\theta}_{ref}(t)$$

The internal dynamics of the machine can be represented by equation 5.101;

$$\mathbf{M}(\theta_{ref}(t)).\ddot{\boldsymbol{q}}(t) + \mathbf{C}(\theta_{ref}(t)).\dot{\boldsymbol{q}}(t) + \mathbf{K}(\theta_{ref}(t)).\boldsymbol{q}(t) = \Omega^2\mathbf{Q}_{ACC}(\theta_{ref}(t)) + \mathbf{Y}.T(t)$$

The **M**, **C**, and **K** terms represent the forces due to the action of the mass, damping and stiffness respectively. The $\Omega^2\mathbf{Q}_{ACC}(\theta_{ref}(t))$ term represents the internal imbalance excitation of the machine, and $\mathbf{Y}.T(t)$ is a force produced by the torque $T(t)$.

The state vector contains a number of co-ordinates, each of which corresponds to a degree of freedom in the machine. It is likely that one of the co-ordinates is dominant over the others, and for modelling purposes it may be sufficiently accurate to consider only one or two of these.

If the machine is rotating slowly at a constant velocity so that $\boldsymbol{q} = 0$, and an impulse of unit area (ideally 1 Nm.s) at $\theta = 0$ is applied, a transient signal $\mathbf{X}\boldsymbol{q}(t)$ will occur which is detectable on the shaft (θ_{meas}). The vector **Y** converts this impulse on $T(t)$ into a force to which the system of equation 5.101 responds, and the shaft will oscillate about the reference angle. The measured angle can be found by equation 5.102 ($\theta_{meas} = \theta_{ref} + \mathbf{X}\boldsymbol{q}(t)$) which relates equations 5.100 and 5.101.

If the row vector **X** and column vector **Y** have a one in their first positions and zeros elsewhere, and assuming **q** and its first derivative are known, **M**, **C** and **K** are easily identifiable.

9.7.2 Characterisation of the Four-bar Mechanism

The four-bar mechanism is a cyclic machine and the characterisation approach of the previous section will be applied. All the scripts used to perform this characterisation are in appendix A16.

Quasi-stationary test

The mechanism was rotated in a forward direction at a very slow speed (approximately 5rpm) and the angle, velocity and torque data recorded. A plot of the data is shown in figure 9.17 showing angle, velocity and torque from top to bottom.

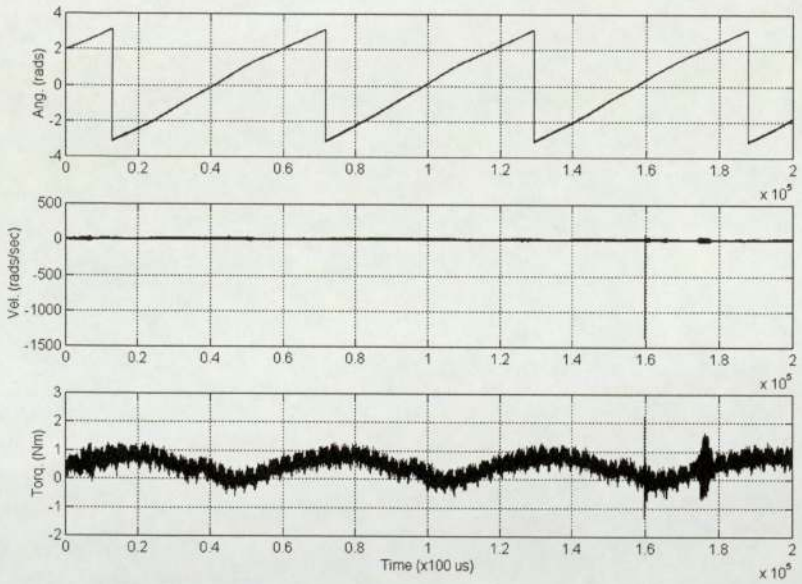


Figure 9.17 Data recorded from quasi-stationary forward test

The data was then split up into separate cycles of the machine (delimited by the angle transition), and reconstructed into shorter data of equal lengths (512 steps) using their Fourier transforms. This also has the effect of filtering the signal to remove the noise and unwanted dynamics of the machine and test-rig. A plot showing the result of this for the forward and reverse directions are shown in figure 9.18 a and b respectively (appendix A.16.1). The new time between samples is 39.0625 ms (100μs × 200000/512).

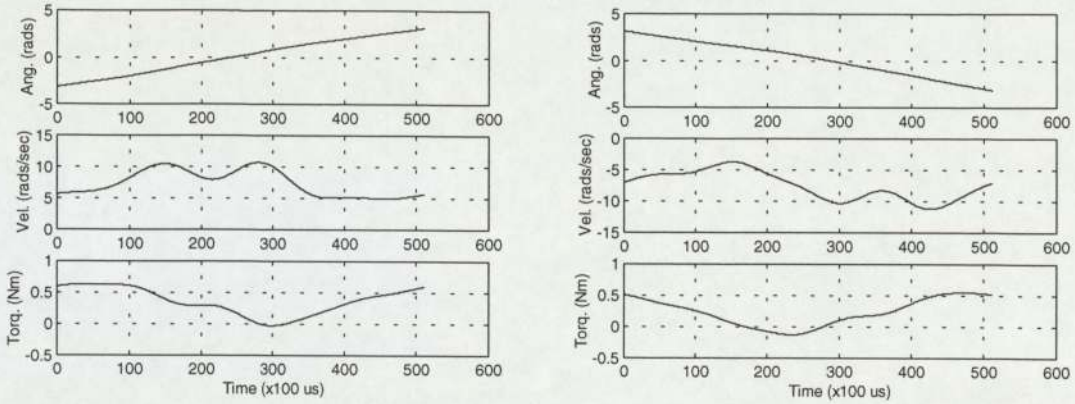


Figure 9.18a, b; One cycle of data for a)forward and b) reverse directions

Constant velocity test

The mechanism was rotated at three different speeds at as near constant velocity as could be obtained from the test-rig controller. Increasing the gain of the controller tended to give rise to noise and oscillations that acted as additional perturbation signals. The three different speeds were 180 rpm (3 Hz), 360 rpm (6 Hz) and 540 rpm (9 Hz). The tests were run for 20 seconds as in the previous test and the data was split into revolutions and averaged as in the quasi-stationary test.

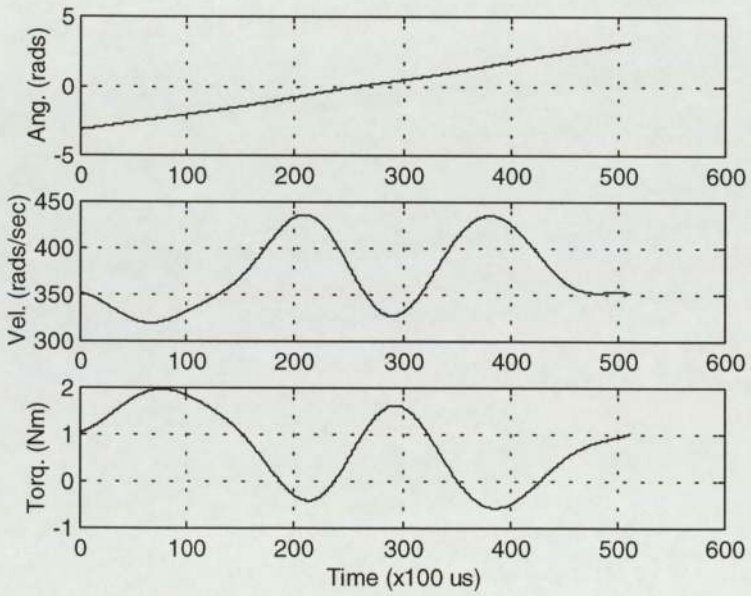


Figure 9.19 Average of one cycle of data for constant velocity at 180rpm

The torque data was then arranged as in equation 5.93 so that the equation could be solved for the parameters $T_{QSF}(\theta)$, $B(\theta)$ and $G(\theta)$;

$$\begin{bmatrix} T_{QSF}(\theta) \\ B(\theta) \\ G(\theta) \end{bmatrix} = \begin{bmatrix} 1 & 3 & 9 \\ 1 & 6 & 36 \\ 1 & 9 & 81 \end{bmatrix}^{-1} \begin{bmatrix} T_3(\theta) \\ T_6(\theta) \\ T_9(\theta) \end{bmatrix} \tag{9.23}$$

A Matlab script was written to perform this task (appendix A16.6), and the parameters were found as a function of angle.

The above equation makes the assumption that the data was collected with a constant velocity. A small variation from this would have caused less accurate results, but the results plotted in figure 9.19 would suggest that the variation is quite large. A check to determine whether the data is likely to be useful is to differentiate the velocity signal to give acceleration and multiply by this an estimated inertia to give the torque (appendix A.16.7).

The length of the rotating arm is 105mm, and a rough estimation of the average inertia can be found if the lumped mass at the outside of the rotating arm is guessed to be about 50g. This estimated inertia is $0.105^2 \times 0.05 \approx 0.5$ kg.m^2. Multiplying by the differentiated velocity gives the estimated torque variation due to the velocity variation, shown in figure 9.20.

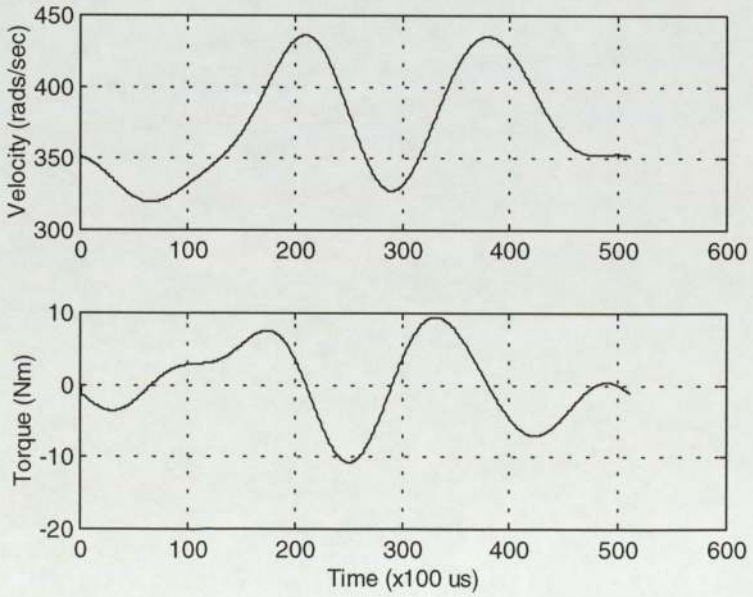


Figure 9.20 Estimation of torque (Nm) due to velocity fluctuation (rad/s)

The purpose of this test is to determine the viscous drag and inertia variation. Equation 5.91 is based on the velocity being constant. The velocity however is not constant and an additional term is included which is the product of inertia and acceleration, as in equation 5.100. This product would appear to be overestimated, but because the contribution to the measured torque is so significant, the calculated values of B and G are almost definitely wrong. The analysis of the data from the next two tests relies on these parameters, so to perform the calculations is purposeless. The data was collected however and is shown in subsequent figures.

Ideally the test would be re-performed at this point to ensure the velocity is near constant. One method is to make changes to the controller to ensure tighter control of the velocity loop. Another method is to use the velocity data plotted in figure 9.20 to drive the machine harder when the velocity reduces. This would make the velocity variation less, and if this procedure were performed iteratively then in theory the velocity could be made constant. In practice this may not be achievable, but it may be close enough for the calculation of the B and G parameters to be near their true values.

Sinusoidal excitation test

This test also requires the machine to be rotated at a constant velocity, but with a sinusoidal excitation superimposed. The purpose of this is to calculate the actual value of the inertia knowing the viscous-drag (B) and the excitation torque (T_{OSC}) assuming otherwise constant velocity, described by equation 5.98. The data recorded from this test was filtered and averaged in the same manner previously discussed in this section, and is shown in figure 9.21.

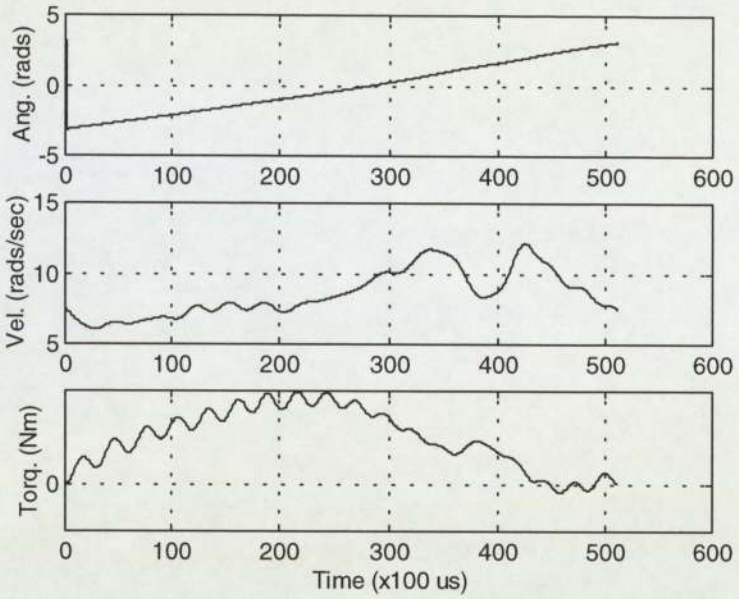


Figure 9.21 Average of one cycle of data for sinusoidal excitation test

If the data were gathered at a near-constant velocity the actual value of the inertia could have been obtained by applying equation 5.98. Unfortunately the data collected is not as valuable as was hoped, and time limitations prevent these tests from being repeated.

CHAPTER 10

10 Conclusion

Models in the context of rotating machinery have been discussed and the selection and parameterisation of these models investigated. A test-rig to perform characterisation and emulation tasks was designed and constructed and practical tests using specially built machines were performed. This work has encompassed a range of engineering disciplines combined to accomplish the rotary machine characterisation and emulation task.

10.1 Achievements and Limitations

System models were discussed and a number of these chosen to represent different machines. State-space models were used extensively in simulation, but in practical tests the ARX structure was found to be the most useful, primarily because of the identification methods employed. Electrical analogues of rotational mechanical components were discussed and an electrical analogue of a rotational machine comprising two inertias and a torsional spring was given. A current source was used to drive this electrical machine and preliminary tests using the DSP board were conducted prior to performing similar tests on the test-rig.

Various test-rig control strategies were investigated and a versatile controller suitable for characterisation and emulation was implemented. It performed well for most tests, but the characterisation of the four-bar mechanism required tighter control of the velocity. Velocity feedforward control was not implemented, so only machines with small cyclically varying inertias would be able to be identified. Further development of the controller would be required to enhance this test, and either velocity feedforward or adaptive control techniques would be necessary.

Four low-inertia motion-sources were used, and two complex methods of mechanical coupling were constructed. The first, a spur-gear gearbox performed sufficiently well for all tests conducted, but the spiral bevel-gear gearbox with smoother meshing and lower inertia would have enhanced the torque bandwidth of the test-rig, enabling machines to be characterised and emulated with more precision. The electrical interfacing performed well, and electrical noise was kept to a minimum. The angle measurement system was complicated in that the digital measurement was converted

into an analogue signal and then back to a digital quantity. The inevitable addition of noise and loss of accuracy was unfortunate but appeared not to impact the operation on the test-rig adversely.

A small number of machines were constructed specifically to characterise and emulate. The electrical system was relatively noise free, required no software control and for these reasons was ideal for preliminary tests. The inertia-torsional spring system tests gave useful data, but were not discussed in the text because the machine was used extensively in simulation examples. Two cyclically varying machines were designed, but ultimately only the four-bar mechanism was used in practical tests.

An overview was given of the most common system identification and parameter estimation methods, for both parametric and non-parametric models. Only the least-squares and instrumental-variables methods were used to characterise machines, which is all that was possible in the time constraints of the project. These methods performed sufficient well to identify the machines tested though, and an emulation of the electrical machine produced credible results.

Figure 10.1 The author demonstrating the test-rig to visitors

This project forms part of a larger project titled "An Integrated Approach to the design of Control Systems for High-Speed Machines" as discussed in chapter 1. The test-rig was constructed in the "Machine Control and Drives Laboratory" in the department of Mechanical Engineering at Aston university, and was used in association with other projects in the laboratory for demonstrational purposes for research at Aston. Figure 10.1 shows the author (right) demonstrating the test-rig to visitors at a presidential visit by the IMechE. The president of the IMechE, Pam Liversidge is pictured on the left.

10.2 Future Work

A number of interesting machines were made, and all but one was used in characterisation experiments. The experimental characterisation of cyclically varying machines was partly performed, but inadequate experimental results and time constraints prevented it from being thoroughly investigated. A more sophisticated controller would be required to further the research, perhaps using advanced techniques such as robust control. Tests for machine characteristics could be partly automated, and series of tests could be performed for particular types of machines. Greater modelling accuracy and higher bandwidth could be obtained through test-rig enhancements, but the existing construction offers much scope for further research.

REFERENCES

References

[1] A.D. Decarlo, "Linear Systems, A State Variable Approach With Numerical Implementation", *Prentice-Hall International, 1989.*

[2] K. Dutton, S. Thompson, B. Barraclough, "The Art of Control Engineering", *Addison-Wesley, 1997.*

[3] J.O. Bird, A.J.C. May, "Mathematics for Electrical Technicians – Levels 4&5", *Longman, 1981.*

[4] C.D. McGillem, G.R. Cooper, "Continuous and Discreet Signal and System Analysis", *Holt, Rinehart & Winston, inc., 1974.*

[5] J. Schwarzenbach, K.F. Gill, "System Modelling and Control", *Edward Arnold, 1979.*

[6] J. Schwarzenbach, "Essentials of Control", *Addison Wesley Longman, 1996.*

[7] G. Loveday, B. Brighouse, "Microprocessors in Engineering Systems", *Pitman, 1987.*

[8] K.J. Åström, B. Wittenmark, "Computer-Controlled Systems", *(3^{rd} edn.) Prentice-Hall, 1997.*

[9] D.R. Seawood "Continuous Phase Synchronised Drives (For a Rod Making Machine)", *Thesis, Aston University 1989.*

[10] C.E. Hinton, "The Maximum-Gain, Minimum-Integral Principle Applied to Materials Testing", IEE PG16 Colloquium on "Getting the Most Out of PID in Machine Control", *IEE, 24 Oct. 1996.*

[11] L. Ljung, "System Identification – Theory for the User", *(1^{st} & 2^{nd} Edn.) Prentice-Hall 1987 & 1999.*

[12] J. Juang, "Applied System Identification", Prentice-Hall, 1994.

[13] K. Godfrey, Perturbation Signals for System Identification", *Prentice-Hall, 1993.*

[14] G.E.P. Box, G. M. Jenkins, "Time Series Analysis, Forcasting and Control", *Holden-Day, 1976.*

[15] T. Söderström, P. Stoica, "System Identification", *Prentice-Hall, 1989.*

[16] A.P. Sage, J.L. Melsa, "System Identification", *Academic Press, 1971.*

[17] P. Eykhoff, "System Identification, Parameter and State Estimation", *John Wiley & Sons, 1974.*

[18] W.D.T Davies, "System Identification for Self-Adaptive Control", Wiley-Interscience, 1970.

[19] P. Eykhoff (editor), "Trends and Progress in System Identification", Permagon Press, 1981.

[20] P.Eykhoff (editor), "Identification and System Parameter Identification", Parts 1 & 2, *North-Holland, 1973.*

[21] J.P. Norton, "An Introduction to Identification", *Academic Press, 1986.*

[22] T.C. Hsia, "System Identification – Least Squares Methods", *Lexicon Books, 1977.*

[23] M.H. Raibert, "Analytical Equations vs. Look-Up for Manipulation: A Unifying Concept", *Proceedings of IEEE Conference on Decision and Control, New Orleans, Dec 1977.*

[24] S.C. Chapra, R.P. Canale, "Numerical Methods for Engineers", *McGraw-Hill, 1990.*

[25] L. Ljung, T. Glad, "Modelling of Dynamic Systems", *Prentice Hall, 1994.*

[26] P. Horowitz, W. Hill, "The Art of Electronics", (2nd Edn.) Cambridge University Press, 1989.

[27] T. Bohlin, "Interactive System Identification: Prospects and Pitfalls", Springer-Verlag, 1991.

[28] J. Golten, A. Verwer, "Control System Design & Simulation", McGraw Hill, 1991.

[29] C.R. Hewson, M.Sumner, G.M. Asher, P.W. Wheeler, "Dynamic mechanical load emulation test facility to evaluate the performance of AC inverters", Power Engineering Journal, IEE, Feb. 2000.

APPENDIX A

Appendix A Matlab Scripts

A.1 Impulse Response of 2-Inertia + Spring System

```
%       J1,A1                   J2,A2
%        ---                     ---
%       |   |                   |   |
%  T ---|   |---/\/\/\/\/\---|   |
%       |   |                   |   |
%        ---                     ---
%         |                       |
%        |*| B1                  |*| B2
%         -                       -
%         |                       |
%        ///                     ///

J1   = 1.0E-03;                        % Kgm^2
J2   = 1.0E-03;                        % Kgm^2
Ks   = 0.5 * J1 * (2*pi*100)^2;        % System has 100Hz res. freq.
B1   = 0.4;                            % 0.444 = sqrt(J1*Ks)
B2   = 0.4;

A = [       0,        1,        0,        0        ;
       -Ks/J1,   -B1/J1,    Ks/J1,        0        ;
            0,        0,        0,        1        ;
        Ks/J2,        0,   -Ks/J2,   -B2/J2        ];

B = [        0        ;
             1/J1     ;
             0        ;
             0        ];

C = [1 0 0 0];
D = 0;

figure;
impulse(A,B,C,D);
```

A.2 Time Response of a Second-Order System

```
% time response of a second-order system to a unit step and
% unit impulse and frequency response plots

wn = 1;             % natural frequency

num = wn^2;


% ----- step responses to different damping ratios -----
figure;

for damp=0.2: 0.2: 1.0,
den = [1 (2*damp*wn) (wn^2)];
step(num,den);
hold on
end

for damp=1: 2: 10,
den = [1 (2*damp*wn) (wn^2)];
step(num,den);
hold on
end


% ----- impulse responses to different damping ratios -----
figure;

for damp=0.2: 0.2: 1.0,
den = [1 (2*damp*wn) (wn^2)];
impulse(num,den);
hold on
end

for damp=1: 2: 10,
den = [1 (2*damp*wn) (wn^2)];
impulse(num,den);
hold on
end


% ----- bode responses to different damping ratios -----
figure;
damp=0.05; den = [1 (2*damp*wn) (wn^2)]; bode(num,den); hold on;
damp=0.1; den = [1 (2*damp*wn) (wn^2)]; bode(num,den); hold on;
damp=0.5; den = [1 (2*damp*wn) (wn^2)]; bode(num,den); hold on;
damp=1.0; den = [1 (2*damp*wn) (wn^2)]; bode(num,den); hold on;
damp=2.0; den = [1 (2*damp*wn) (wn^2)]; bode(num,den); hold on;



% ----- nyquist responses to different damping ratios -----
figure;
damp=0.5; den = [1 (2*damp*wn) (wn^2)]; nyquist(num,den); hold on;
damp=1.0; den = [1 (2*damp*wn) (wn^2)]; nyquist(num,den); hold on;
damp=1.5; den = [1 (2*damp*wn) (wn^2)]; nyquist(num,den); hold on;
damp=2.0; den = [1 (2*damp*wn) (wn^2)]; nyquist(num,den); hold on;


% now use 'zoom' to frame the interesting part of the plots
```

A.3 Compensated Response of Test-rig

```
% PDcntrl2.m
% uses lead compensation for D part of controller
% pole-zero map and step response of a PID controller in series
% with the test-rig (D implemented as a Lead-compensator)

clear

% controller parameters and transfer function
Kp = 0.5E-4;    %was 1.0748E-6 for P only critically damped
Kd = 10;    alpha=0.05;

Pnum = Kp;
Pden = 1;

Dnum = [Kd 1];
Dden = [alpha*Kd 1];
```

```
[Cnum,Cden] = series(Pnum,Pden,Dnum,Dden);

% plant transfer function
Gnum = 2326;
Gden = [1 0.1 0];

% 'add' the two systems (can't use 'series' because C(s) is not proper)
[num,den] = series(Cnum,Cden,Gnum,Gden);
%'open loop transfer function'   printsys(num,den,'s')

% close the loop, -ve feedback
[numc,denc] = cloop(num,den,-1);
%'closed loop transfer function' printsys(numc,denc,'s')

% plot pole-zero map & step response
figure
pzmap(numc,denc);
%[p,z] = pzmap(num,den)        % display the poles & zeros
figure
step(numc,denc);
```

A.4 Differentiation comparisons

```
% diffrtn4.m - DW 24/7/97
%
% loads 'torque' which is data from ident_1.2 and contains
% T the time step (1ms), TR torque, and fTR the integral of TR.
% compares 2 methods of differentiating fTR; rectangular
% and (Taylor) finite-difference formula.
% Similar to diffrtn3.m.

clear
load 'torque'
data_length = size(TR,1);

% differentiation variables
dx_rec = zeros(data_length,1);
dx_taylor = zeros(data_length,1);

% add noise
fTR = addnoise(fTR,0.025);

% filter signal
[b,a] = butter(3,2500/5000);
fTR = filter(b,a,fTR);


% This script computes a finite-difference formula for the 1stm derivative of a
% discrete signal based on present value of that signal and N previous values.
iok = 0;
while iok==0;
    iok = 1;
    N = input(' Enter number of previous values to use    :  ');
    M = input(' Enter highest order of derivative to use  :  ');
    if (M>N)
       disp(' Must have N > N !! ');   iok=0;
    end
end
R = 0.0 - (0:N).';
TE = taylex1(R, M);
TI = pinv(TE);
format long
pinvte = (TI(2,:)).'
format short


num_prev = size(pinvte,1);    % number of previous values required

% perform differentiation
for t=num_prev:data_length
      dx_rec(t) = (fTR(t)-fTR(t-1))/T;
%     dx_taylor(t) = -1000*sum(fTR(t-num_prev+1:t) .* pinvte);
      for index=0:5
            dx_taylor(t) = dx_taylor(t) + (fTR(t-index) * pinvte(index+1));
      end
      dx_taylor(t) = dx_taylor(t)*1000;
end


figure;
subplot(4,1,1), plot(fTR, 'w'), ylabel('f TR');
```

```
subplot(4,1,2), plot(TR, 'w'), ylabel('TR');
axis([0 1000 -2 2]);
subplot(4,1,3), plot(dx_rec, 'w'), ylabel('dx_rec');
axis([0 1000 -2 2]);
subplot(4,1,4), plot(dx_taylor, 'w'), ylabel('dx_taylor1');
axis([0 1000 -2 2]);
```

```
% taylex1.m
% Computes a 1-dimensional Taylor expansion matrix (SDG 1997).
% R is a list of "x" coordinates
% M is the highest order of derivative of interest

function TAYLOR = taylex1(R, M);


N = max(size(R));

L=M+1;
%disp([' Representing ' int2str(L) '  derivatives'])
TAYLOR=zeros(N,L);
DERIV=zeros(L,2);

DERIV(1,1)=0;
DERIV(1,2)=1;

for i=2:L;
   DERIV(i,1)=i-1;
   DERIV(i,2)=DERIV(i-1,2)*(i-1);
end

% Fill the array TAYLOR
for i=1:N
   tx = R(i);
   for j = 1:L
       TAYLOR(i,j)=(tx^DERIV(j,1))/DERIV(j,2);
   end
end
```

A.5 Swept-Sine Generation

```
% SWPSIN version 4
% Returns the value of a swept sine wave after t seconds.
%    x = swpsin4(rate,f0,t)
%    f0 is start frequency,
%    f increases exponentially at 'rate' decades/sec.
%                              (15/8/97 - DW)

function [x] = swpsin4(rate,f0,t)

k = rate * log(10);
% freq = f0 * 10^(k10 * t)
angl = (f0/k) * exp(k * t);
x = sin(angl);
```

A.6 Simulation of 1-Inertia + Spring System

```
% simple_load.m - 17/1/97 - DW
% Simulation of a 1-inertia system + spring, for parameter estimation.
% Try parameter ident. on an easier model first.
%
%       J, A
%        ---
%       |   |             |\
% TR ---|   |---/\/\/\/\/\|\
%       |   |             |\
%        ---
%         |
%        |*| C
%
% State Variables :    xx = [    velocity        ;
%                              acceleration    ]
%
%                     x = [    angle        ;
%                              velocity    ]
%
% Inputs             TI -- External Torque on Inertia
```

```
%
% Parameters          J  -- Value of Inertia
%                     Ks -- Spring stiffness
%                     C  -- Damping on J
%
%                     T  -- time step (seconds)
% -------------------------------------------------------

clear
% set up variables

No_of_steps = 800;
T           = 0.001;              % second(s)
TR = zeros(No_of_steps,1);        % input torque profile

t=0;
for i=1:No_of_steps
    TR(i) = sweepsin(100,t);      % torque I/P is sweep sine
    t=t+T;
end

J    = 1.0E-03;                    % Kgm^2 ?
Ks   = 0.5 * J * (2*pi*100)^2;     % System has 100Hz res. freq. ?
C    = 0.21;                       % 0.444 = sqrt(J*Ks) why?

A = [        0,         1   ;
        -Ks/J,    -C/J    ];

B = [          0        ;
             1/J        ];

xx = [0; 0];                      % initial rates
x = zeros(2,No_of_steps);

for t=2:No_of_steps;
    u = TR(t);                                          % input, u = TI(t)
    x(1:2,t) = x(1:2,t-1) + xx*T;                       % x(t) = x(t-1) + xx*T
    xx = A*x(:,t) + B*u;                                % xx = Ax + Bu
end;

figure;
subplot(3,1,1), plot(TR(:)), ylabel('Torque (Nm)');
subplot(3,1,2), plot(x(1,:)), ylabel('Ang (rads)');
subplot(3,1,3), plot(x(2,:)), ylabel('Vel (rads/s)'), xlabel('Frequency (Hz)');


% save the variables the parameter estimation part needs (T, TR, & x)
save 'data' T TR x
```

A.7 Simulation of Spring + 1-Inertia System

```
% machine8_3_2.m - 28/11/97 - DW
% Simulation of a spring + 1-inertia system.
%
%                 A2
%                 ---
% A1,            |   |
% TR -/\/\/\/\/\-| J |
%                 |   |
%                 ---
%                  |
%                 |*| Dmpng
%
% State Variables :    xx = [    velocity        ;
%                                acceleration    ]
%
%                       x = [    angle       ;
%                                velocity    ]
%
% Inputs              TI -- External Torque on Inertia
%
% Parameters          J      -- Value of Inertia
%                     Ks     -- Spring stiffness
%                     Dmpng  -- Damping on J
%
%                     T      -- time step (seconds)
% -------------------------------------------------------

clear
% set up variables
```

```
plot_xmin = 100;

No_of_steps = 600;
T           = 0.001;                    % second(s)
TR = zeros(No_of_steps,1);              % input torque profile

t=0;
for i=1:No_of_steps
    TR(i) = sweepsin(100,t);            % torque I/P is sweep sine
    t=t+T;
end

J     = 1.0E-03;                        % Kgm^2 ?
Ks    = 0.5 * J * (2*pi*100)^2;         % System has 100Hz res. freq. ?
Dmpng = 0.31;                           % Nm/rad/s ?

A = [   0,    1         ;
        0,    -Dmpng/J  ];

B = [    0      ;
         1/J    ];

C = [    1,    0    ];

D = 1 / Ks;

xx = [0; 0];                            % initial rates
x = zeros(2,No_of_steps);

for t=2:No_of_steps;
    u = TR(t);                          % input, u = TI(t)
    x(1:2,t) = x(1:2,t-1) + xx*T;       % x(t) = x(t-1) + xx*T
    xx = A*x(:,t) + B*u;                % xx = Ax + Bu
    y(t) = C*x(:,t) + D*u;              % y(t) = Cx + Du
end;

figure;
subplot(2,1,1), plot(TR(:)), ylabel('Torque, u (Nm)'), grid,
                axis([plot_xmin, No_of_steps, -1, 1]);
subplot(2,1,2), plot(y(:)), ylabel('Ang 1, y (rads)'), grid,
                axis([plot_xmin, No_of_steps, 0.07, 0.09]);
                xlabel('Time (ms)');


u = TR;     % input (Sys. Ident. toolbox requires row vectors)
y = y';     % output (Sys. Ident. toolbox requires row vectors)

% save the variables the parameter estimation part needs (T, TR, & y)
save 'data' T u y
```

A.8 Simulation of 2-Inertia + Spring System

```
% machine8_3_3.m - 11/97 - DW
% Simulation of a spring + 2-inertia system.
%
%       A1                A2
%       ---               ---
%      |   |      Ks      |   |
% TR--|J1 |-/\/\/\/\/\-|J2 |
%      |   |              |   |
%       ---               ---
%        |                 |
%       |*| C1             |*| C2
%
% State Variables :     xx = [    velocity_1     ;
%                               acceleration_1 ;
%                               velocity_2     ;
%                               acceleration_2 ]
%
%                       x = [    angle_1    ;
%                               velocity_1 ;
%                               angle_2    ;
%                               velocity_2 ]
%
% Inputs              TI      -- External Torque on Inertia
%
% Parameters          J1,J2   -- Value of Inertia
%                     Ks      -- Spring stiffness
%                     B1,B2   -- Damping on J1,J2
%
%                     T       -- time step (seconds)
```

```
% ------------------------------------------------------

clear
% set up variables

plot_xmin = 100;

No_of_steps = 1000;
T           = 0.001;                % second(s)
TR = zeros(No_of_steps,1);          % input torque profile

t=0;
for i=1:No_of_steps
    TR(i) = sweepsin(100,t);        % torque I/P is sweep sine
    t=t+T;
end

J1    = 1.2E-03;                                % Kgm^2
J2    = 1.2E-03;                                % Kgm^2
Ks    = 0.5 * ((J1+J2)/2) * (2*pi*100)^2;       % System has 100Hz res. freq. ?
B1    = 0.5;                                    % Nm/rad/s
B2    = 0.5;                                    % Nm/rad/s

A = [   0,      1,      0,      0;
        -Ks/J1, -B1/J1, Ks/J1,  0;
        0,      0,      0,      1;
        Ks/J2,  0,      -Ks/J2, -B2/J2];


B = [    0      ;
         -1/J1  ;
         0      ;
         0      ];


C = [   1,  0,  0,  0];


xx = [0; 0; 0; 0];                  % initial rates
x = zeros(4,No_of_steps);

for t=2:No_of_steps;
    u = TR(t);                      % input, u = TI(t)
    x(1:4,t) = x(1:4,t-1) + xx*T;   % x(t) = x(t-1) + xx*T
    xx = A*x(:,t) + B*u;            % xx = Ax + Bu
    y(t) = C*x(:,t);                % y(t) = Cx + Du (D=0)
end;

figure;
subplot(3,1,1), plot(TR(:)), ylabel('Torque, u (Nm)'), grid;
subplot(3,1,2), plot(y(:)), ylabel('Ang 1, y (rads)'), grid;
subplot(3,1,3), plot(x(2,:)), ylabel('Vel 1 (rads)'), grid,
                xlabel('Time (ms)');

u = TR;     % input (Sys. Ident. toolbox requires row vectors)
y = y';     % output (Sys. Ident. toolbox requires row vectors)

% save the variables the parameter estimation part needs (T, TR, & y)
save 'data' T u y
```

A.9 Resonating Bar Calculations

```
% rsntng_bar.m - 16/8/96 - DW
% Matlab script to calculate the natural frequency
% of a bar given its dimentions, density & youngs modulus.
% The bar is to be used in the second example load; a four-
% bar mechanism which will store strain energy, and
% have a detectable internal resonance - originating mainly
% from this bar.
%
%
%
%       <--------- L --------->    'm' is additional mass
%       ___________m___________    <--- end view:
%       ^                     ^                 _______
%    support                support             |       | h
%                                                ~~~~~~~
%                                               <-- b -->
%
% ------------------------------------------------------
```

```
clear
% set up variables
b = 0.010;          % dimension b (m) - actually this cancels out
                        %                                 and isn't relevant! (in theory)
h = 0.012;          % dimension h (m)
L = 0.430;          % dimension l (m)
E = 210e9;          % Young's modulus for steel (Pa = Nm^-2)
d = 7000;           % density of steel = 7800 Kg/m^3

m = 0.100;          % mass of the additional weight (kg)

I = (b*h^3)/12;                     % 2nd moment of area

k = 2*( 3*E*I/(L/2)^3 );        % stiffness of centre point


am = b * h * L * d;                 % actual mass of the 'spring'

em = am/3;                          % mass is Ö 3 because the effective
                                        % mass of a spring is about 1/3 of
                                        % it's actual mass. (p.285 'Advanced
                                        % physics' - T. Duncan)

em = em + m;                        % add the mass of the weight

fn = (1/(2*pi))*sqrt(k/em)      % natural frequency of a spring
                                        % fixed at both ends:
                                        %                        _____
                                        %          1            |  k
                                        %     fn = -- *         | ---
                                        %          2¹          \|  m
```

A.10 Four-bar Mechanism Movement Simulation

```
% four_bar_mech.m - 16/8/96 - DW
% Matlab script to analyse the movement of a four bar mechanism.
%
%         x2,y2                    x3,y3
%              B_________________C
%               \        b       \
%                \                \
%              a  \                \
%                  \                \
%                   \                \         ks       |\
%   X___             A               c  \--\/\/\/\/\/\--|\
%              x1,y1                    \                |\
%                                        \
%                                         \
%                                          \
%                                           \
%   Y___                                     D
%                                              x4,y4
% ------------------------------------------------------------

%clear              % clear variable space
%close all  % close all current figure windows

% set up variables
a = 0.105;          % length of bars (m)
b = 0.200;          % ---- " ---- " ----
c = 0.250;          % ---- " ---- " ----
x1 = 0.150;         % coordinates of A (m)
y1 = 0.150;         % ----- " ---- " -----
x4 = x1 + b;        % coordinates of D (m)
y4 = 0.042;         % ----- " ---- " -----
mass1 = 0.1;        % mass in centre of driven bar CD (kg)
mass2 = 0.1;        % mass of driven bar CD (kg)
mass3 = 0.07;       % mass of link BC (kg)
mass = 0.5*mass1 + 0.3*mass2 + 0.5*mass3;
                        % the effective mass at point C
ks = 200.0;         % spring-constant of spring (N/m)
rpm = 1500;         % speed of rotation (revs / min)
rps = rpm/60;       % ---- " --- " ---- (revs / sec)
T = rps/360;        % time for 1 loop iteration i.e. 1¡

XAB = zeros(360,1);                 % angle XAB record
YDC = zeros(360,1);                 % angle YDC record
dYDC_dt = zeros(360,1);         % dYDC_dt record
ddYDC_dt = zeros(360,1);        % ddYDC_dt record
TR = zeros(360,1);                  % I/P torque required record
```

```
xx = zeros(360,1);                  % xx, coord. record for animation
yy = zeros(360,1);                  % yy, coord. record for animation

% initial values
last_C_dist = 0;  % initial angle
last_dCdt = 0;          % initial angular velocity
m = 1;                        % m is bar coordinates index

% get animation requirements
revs = input('how many rotations to animate?');

% calculate remaining coordinates for values of XAB:
for n=1:360
      XAB(n) = (2*pi*n)/360;        % convert deg to rad
      x2 = x1 - a*cos(XAB(n));      % calculate coords of B
      y2 = y1 + a*sin(XAB(n));      % --- " --- " --- " ---

      BD = sqrt((x4-x2)^2+(y4-y2)^2);     % calculate length BD

      YDB = atan((y2-y4)/(x4-x2));  % calculate angle YDB

      BDC = acos((c^2+BD^2-b^2)/(2*c*BD));      % calc. angle BDC

      DCB = asin(BD*(sin(BDC)/b));  % calculate angle DCB

      DBC = pi - (BDC + DCB);             % calculate angle DBC

      CBA = DBC + (pi-YDB)-(pi-XAB(n));   % calculate angle CBA
      if (CBA>(2*pi))
            CBA = CBA - 2*pi;             % keep within 0<CBA<2¹
      end

      YDC(n) = YDB + BDC;                       % calculate angle YDC

      x3 = x4 - c*cos(YDC(n));            % calculate coords of C
      y3 = y4 + c*sin(YDC(n));            % --- " --- " --- " ---

      C_dist(n) = YDC(n) * c;             % act. dist. of C from horz.
      dCdt(n) = (C_dist(n)-last_C_dist)/T;      % vel. of C
      ddCdt(n) = (dCdt(n)-last_dCdt)/T;         % accn. of C
      last_C_dist = C_dist(n);            % for next dYDC_dt
      last_dCdt = dCdt(n);                % for next ddYDC_dt

      FC = mass*ddCdt(n) + 0.5*ks*(0.5323-C_dist(n));
                                          % f = m.a + 0.5*ks*(C_dist_max-C_dist)
      FCB = FC / sin(DCB);                % force into link
      FBA = FCB * sin(CBA);               % force into driving bar
      TR(n) = FBA * a;                    % input torque

      xx(m) = x1;       % coordinates of A
      yy(m) = y1;       % --- " ---- " ---
      xx(m+1) = x2;     % coordinates of B
      yy(m+1) = y2;     % --- " ---- " ---
      xx(m+2) = x3;     % coordinates of C
      yy(m+2) = y3;     % --- " ---- " ---
      xx(m+3) = x4;     % coordinates of D
      yy(m+3) = y4;     % --- " ---- " ---
      m = m + 4;        % increment by the number of coordinates
end

% zero the first point in dCdt and the first 2 in dCdt,
% because they are invalid
dCdt(1) = 0;
ddCdt(1:2) = [0;0];

% plot driven bar angle, angular velocity and acceleration.
figure(1);
subplot(4,1,1), plot (YDC),
      title('angle YDC (driven bar) vs angle XAB (I/P bar)'),
      xlabel('angle XAB (I/P bar)'),
      ylabel('rads');
      grid;
subplot(4,1,2), plot (dCdt),
      title('dCdt (velocity) vs angle XAB (I/P bar)'),
      xlabel('angle XAB (I/P bar)'),
      ylabel('m/sec');
      grid;
subplot(4,1,3), plot (ddCdt),
      title('d^2Cdt^2 (acceleration) vs angle XAB (I/P bar)'),
      xlabel('angle XAB (I/P bar)'),
      ylabel('m/sec^2');
      grid;
```

```
subplot(4,1,4), plot (TR),
      title('input torque XAB vs angle XAB (I/P bar)'),
      xlabel('angle XAB (I/P bar)'),
      ylabel('Nm');
grid;

% animate the four bar mechanism
if (revs~=0)
      figure(2);
      hold;                          % allows for multiple plots
      axis([0, 0.5, 0, 0.3]); % set scaling for plots
      axis('equal');                 % square aspect ratio
      axis(axis);                    % freeze scaling & current limits
      axis('off');                   % turn off labeling & tick marks
      for n = 1:revs
            for m = 1: 4: 360*4
                  plot(xx(m:m+3),yy(m:m+3),'w');       % plot each increment
                  plot(xx(m:m+3),yy(m:m+3),'k');       % erase -- " --- " --
            end
      end
                  plot(xx(1:4),yy(1:4),'w');     % plot last increment
      hold;
end
```

A.11 Adding Noise to a Signal

```
% addnoise(x,noise_factor) returns matrix 'x' with 'noise_factor'
% rms uniformly distributed noise added.
% (14/2/97 - DW)

function [y] = addnoise(x,noise_factor)

noise = rand(size(x))-0.5;                          % create noise array same size as x
x_rms = sqrt(mean(mean(x.^2)));                     % mean square of x
an_rms = sqrt(mean(mean(noise.^2)));       % actual mean square of noise

rn_rms = (noise_factor*x_rms);

correction_factor = rn_rms/an_rms;
                                                          % to make noise_ms = x_ms *
noise_factor
noise = noise * correction_factor;  % noise_ms now equals x_ms * noise_factor
y = x + noise;
```

A.12 Inertia - Torsional Spring - Ground system LS PE

```
% par_est.m (V1.3) - 21/1/96 - DW
% Parameter identification of a simple load - a 1 inertia, 1 spring
% see 'simple_load'.
%
% Uses:     T     -     time step (seconds)
%           TR    -     I/P torque profile
%           x     -     x = [ angle_1           ;           (x = state vector)
%                             velocity_1  ]
%
% ------------------------------------------------------------------

clear
load 'data'

%Add noise to recorded signals x and TR...
x(1,:) = addnoise(x(1,:),0.02);
x(2,:) = addnoise(x(2,:),0.02);
TR = addnoise(TR,0.02);
%figure;
%subplot(2,1,1), plot(x(1,:)), ylabel('Ang.');
%subplot(2,1,2), plot(x(2,:)), ylabel('Vel.');

% move thetal and thetal. from rows 1&2 to rows 2&3 (create row 3)
% fill row 1 with zeros ready for the 1st integral
% and create fTR with zeros for the 1st integral
x(3,:) = x(2,:);
x(2,:) = x(1,:);
x(1,:) = zeros(1,size(x,2));
fTR = zeros(size(TR));

% generate the 1st integral of theta and put in row 1
% also generate 1st integral of TR
for t=2:(size(x,2))
      x(1,t) = x(1,t-1)+(T/2)*(x(2,t)+x(2,t-1));      %trapez integ
```

```
      fTR(t) = fTR(t-1)+(T/2)*(TR(t)+TR(t-1));  %trapez integ
end

%figure;
%subplot(3,1,1), plot(x(1,:)), ylabel('fang');
%subplot(3,1,2), plot(x(2,:)), ylabel('ang');
%subplot(3,1,3), plot(x(3,:)), ylabel('ang`');
%figure;
%subplot(2,1,1), plot(fTR), ylabel('fTR');
%subplot(2,1,2), plot(TR), ylabel('TR');

% The equation which to find the parameters for is:
% fTR = c0*f[theta1] + c1*[theta1] + c2*[theta1.]
% where 'f' represents an integration & the params. are
% c0 = -Ks, c1 = C, c2 = J

% transpose x to correspond to equations
x = x';

% normalise the equations by dividing the 'coefficients' by the
% sum of the squares of the coefficients in each equation...
m = size(x,1);
for i=1:m
      xtemp = x(i,:);                          % each row of data
      d = sqrt(xtemp * xtemp');                % d = sum of squares of coeffs
      if d == 0          % check and trap any zero's to prevent / by 0 later
            d = 1.0;
      end;
      x(i,:) = x(i,:)./d;                      % normalise 'coefficients'
      fTR(i) = fTR(i)./d;                      % normalise 'coefficients'
end

% 'nr' is the normalised set of roots
nr = pinv(x)*fTR;

fprintf(1,'Ks = %f\nC = %f\nJ = %f\n', nr(1), nr(2), nr(3));
```

A.13 Inertia - Torsional Spring - Ground system LS PE

```
%ident.m - to use the identification toolbox for LS PE

% add noise to the clean simulated signals, twice, and save for PE & verification
clear
load 'data'

% Define the quantity of noise to be added
pc = 2;          % percent noise to add to signal
pc = pc/100;

% Add noise to recorded signals and save for identification
ye = addnoise(y,pc);
ue = addnoise(u,pc);
save 'est_dat' T ye ue

% Add noise to recorded signals and save for verification
yv = addnoise(y,pc);
uv = addnoise(u,pc);
save 'val_dat' T yv uv


% load the data to be used for parameter estimation
clear
load est_dat             % data with noise
ze = [ye ue];            % [output input]

figure
idplot(ze)                    % plot the input / output data

% filter data to remove frequencies higher than that of interest
[Bf,Af] = butter(4,1000/5000);  % 4th order, 1kHz cut-off, 5kHz sampling
ze(:,1) = filter(Bf,Af,ze(:,1));
ze(:,2) = filter(Bf,Af,ze(:,2));

zn1 = dtrend(ze);             % remove const. levels to make data zero mean

th = arx(ze, [2 1 1]);        % perform the discrete time parameter estimation

th = sett(th, T);             % set the sampling frequency to the model

[A,B]=th2arx(th)              % remove the ARX coefficients
```

```
A_row = A(1,:);              % a0, a1 and a2
B_row = B(1,2);              % b1 (is the only non-zero input parameter)

delt = 1E-4;                 % specify delta-t (i.e. time step)

T = [ 1.0   delt  delt^2/2; % Taylor-Expansion matrix
      1.0    0.0     0.0  ;
      1.0  -delt  delt^2/2];

model = A_row*T / B_row      % calculate the continuous time parameters


% using new data, compare the actual O/P, and simulated O/P of new model

load val_dat             % different data with noise for model validation
ysim = idsim(uv, th);    % put different data through new model
figure
plot(yv, 'w');           % plot validation O/P data
hold;
plot(ysim, 'k:');        % plot new model simulated data on same graph
ylabel('Angle (rads)'), xlabel('Time (x100 us)');
```

A.14 ARX Model Selection and Validation

```
%idnt_arx.m - uses the identification toolbox for elec model SI
%             using the functions arxstruc() and selstruc().
%             prbstest is used for SI and swpsn_v for validation

clear
load prbstest                              % data from elec model
ze = [prbstest(:,2), prbstest(:,1)];       % [output input]
clear prbstest

load swpsn_v                               % different data for model validation
zv = [swpsn_v(1:8000,2), swpsn_v(1:8000,1)];  % [output input]
clear swpsn_v

T = 1e-4;

ze = dtrend(ze);             % remove DC offsets
zv = dtrend(zv);             % remove DC offsets

figure

V = arxstruc(ze, zv, struc(2, 2, 1:5));      % create arx's of different delays
nn = selstruc(V,0);                          % establish suitable delay value
nk = nn(3);                                  % extract delays
V = arxstruc(ze, zv, struc(1:20, 1:20, nk-1:nk+1));
                                             % test all combinations of arx
                                             % models with up to 20 a & b params
                                             % with delays around selected value
nn = selstruc(V)     % the selected structure

th = arx(ze, nn);    % fit data to selected model to calculate parameters
th = sett(th, T);    % set the sampling interval

% using new data, compare the actual O/P, and simulated O/P of new model

ysim = idsim(zv(:,2), th);  % simulate new model using actual I/P signal
                            % to compare simulated O/P to the actual O/P signal
figure
plot(zv(:,1), 'w');         % plot validation O/P data
axis([0 8000 -1.5 1.5]);
ylabel('Angle (rads)'), xlabel('Time (x100 us)');

figure
plot(ysim, 'w');            % plot new model simulated data
axis([0 8000 -1.5 1.5]);
ylabel('Angle (rads)'), xlabel('Time (x100 us)');
```

A.15 Non-Parametric Characterisation

```
load swpsn_e                          % data from elec model
ze = [swpsn_e(:,2), swpsn_e(:,1)];    % [output input]
clear swpsn_e
T = 1e-4;

ze = dtrend(ze);
```

```
% perform correlation analysis

figure
ir = cra(ze);           % ir = estimated impulse response vector

%sr = cumsum(ir);        % sr = estimated step response vector
%figure
%plot(sr)


% perform spectral analysis

[G,PHIV] = spa(ze);     % G = estimated freq func
                        % PHIV = estimated disturbance spectrum
GN = sett(G, T);        % set sampling interval

figure                  % plot estimated freq func
ffplot(GN)              % plots freq fncn with linear freq scales (uses Hz)
%bodeplot(GN)           % plots freq fncn with log freq scales (uses rad/sec)

figure                  % plot estimated disturbance spectrum
ffplot(PHIV)
%bodeplot(PHIV)

Ge = etfe(ze);
```

A.16 Four-bar Mechanism Characterisation

A.16.1 Quasi-stationary test

```
% qsf_fprc.m

clear
load  qsff2

qsff2(:,1) = qsff2(:,1) - (( abs(max(qsff2(:,1))) - abs(min(qsff2(:,1))) )/2 );
                                        % max = -min
qsff2(:,1) = qsff2(:,1) * ( (2*pi) / (max(qsff2(:,1))-min(qsff2(:,1))) );
                                        % convert angle to radians (range -pi to pi)

%figure
%subplot(3,1,1), plot(qsff2(:,1), 'w'), grid, ylabel('Ang.');
%subplot(3,1,2), plot(qsff2(:,2), 'w'), grid, ylabel('Vel.');
%subplot(3,1,3), plot(qsff2(:,3), 'w'), grid, ylabel('Torq.');
%xlabel('x100 us');

% find the transitions of the angle signal
count = 1;
last_ang = qsff2(1,1);
for n=2:200000,
    new_ang = qsff2(n,1);
    if ( abs(last_ang-new_ang) > 3 )
        ang_trans(count) = n;
        count = count + 1;
    end
    last_ang = qsff2(n,1);
end

% change the length of the data for each revolution
Np = 512;   % New number of points
Nh = 5;     % Use the first 5 harmonics above 0
n=1;
for count=1:3:(size(ang_trans,2)-1),
    if ( (ang_trans(n)+1) < ((ang_trans(n+1)-2)-500) )
        temp = len_chng( qsff2( (ang_trans(n)+1):(ang_trans(n+1)-2), :), Nh, Np );
                                        % change data length of nth rev
        split_data(:,n) = temp(:,1);
        split_data(:,n+1) = temp(:,2);
        split_data(:,n+2) = temp(:,3);
        n = n + 3;
    end
end

% take the average of each revolution data
ang_mean(:) = split_data(:,1);
vel_mean(:) = split_data(:,2);
tor_mean(:) = split_data(:,3);
for n=4:3:((n-1)/3),
    ang_mean(:) = ang_mean(:) + split_data(:,n);
    vel_mean(:) = vel_mean(:) + split_data(:,n+1);
    tor_mean(:) = tor_mean(:) + split_data(:,n+2);
```

```
end
one_rev_fwd(:,1) = ang_mean(:) / (n/3);
one_rev_fwd(:,2) = vel_mean(:) / (n/3);
one_rev_fwd(:,3) = tor_mean(:) / (n/3);

% ENSURE -pi < ANGLE < pi AND VEL IS +VE
one_rev_fwd(:,1) = one_rev_fwd(:,1) - (( abs(max(one_rev_fwd(:,1))) -
abs(min(one_rev_fwd(:,1))) )/2 );
                                        % max = -min
one_rev_fwd(:,1) = one_rev_fwd(:,1) * ( (2*pi) / (max(one_rev_fwd(:,1))-
min(one_rev_fwd(:,1))) );
                                        % convert angle to radians (range -pi to pi)

if ( min(one_rev_fwd(:,2)) < 0 )         % vel shouldn't go -ve (bit of a bodge)
    one_rev_fwd(:,2) = one_rev_fwd(:,2) - min(one_rev_fwd(:,2));
end

figure
subplot(3,1,1), plot(one_rev_fwd(:,1), 'w'), grid, ylabel('Ang.');
subplot(3,1,2), plot(one_rev_fwd(:,2), 'w'), grid, ylabel('Vel.');
subplot(3,1,3), plot(one_rev_fwd(:,3), 'w'), grid, ylabel('Torq.');
xlabel('x100 us');

qsf_fwd = one_rev_fwd;

save qsf_fwd qsf_fwd        % save data for analysis
```

A.16.2 len_chng.m

```
% len_chng.m
%
% re-create (and plot?) a data matrix using a different number of points
% where (:,1) is the angle, (:,2) velocity and (:.3) torque.
% new vector = rcnstrct( signal, nHarm, nPoints )
%


function new_data = len_chng( data_matrix, Nh, Np )


%figure
%subplot(3,1,1), plot(data_matrix(:,1), 'w'), grid;
%subplot(3,1,2), plot(data_matrix(:,2), 'w'), grid;
%subplot(3,1,3), plot(data_matrix(:,3), 'w'), grid;

% RECONSTRUCT THE SIGNALS WITH DIFFERENT DATA LENGTHS USING FOURIER TRANSFORM
%new_sig(:,1) = rcnstrct( data_matrix(:,1), Nh, Np );
                                        % won't work because of gibbs phenomenon
x = cos(data_matrix(:,1));
y = sin(data_matrix(:,1));
new_x = rcnstrct( x, Nh, Np );
new_y = rcnstrct( y, Nh, Np );
new_sig(:,1) = atan2(new_y, new_x);
new_sig(:,2) = rcnstrct( data_matrix(:,2), Nh, Np );
new_sig(:,3) = rcnstrct( data_matrix(:,3), Nh, Np );

%figure
%subplot(3,1,1), plot(new_sig(:,1), 'w'), grid;
%subplot(3,1,2), plot(new_sig(:,2), 'w'), grid;
%subplot(3,1,3), plot(new_sig(:,3), 'w'), grid;

new_data = new_sig;
```

A.16.3 rcnstrct.m

```
% rcnstrct.m
%
% re-create (and filter) a signal using a different number of points
% (this, fouri_s.m & gen_fbas.m written by SDG)
% new vector = rcnstrct( signal, nHarm, nPoints )
%


function new_vector = rcnstrct( signal, Nh, Np )

% --- Now re-create (and filter) that signal
% --- using a different number of points (Np)

coeffs = fouri_s( signal, Nh );   % Now we have found the coeffs.
basis  = gen_fbas( Np, Nh );      % Construct a diff. basis.
```

```
new_vector = basis * coeffs;       % This is new signal.
```

A.16.4 fouri_s.m

```
% fouri_s.m

   function coeffs = fouri_s( signal, nharm )
% --- fouri_s ---
% This function finds a "small" Fourier transform of a
% signal represented by a large number of discrete points.
% It produces a vector of coefficients of the basis
% functions ...
% cos(0*theta)
% cos(1*theta)    sin(1*theta)
% cos(2*theta)    sin(2*theta)
% cos(3*theta)    sin(3*theta)
% ... up to
% cos(nharm*theta)    sin(nharm*theta)
% This can be used as a part of a filtering operation.
% NOTE : signal is assumed to be a set of COLUMN vectors.

ssig = size(signal,1);

% --- Generate the basis of Fourier functions.
basis = gen_fbas( ssig, nharm );

ibass = basis/(basis.'*basis);

coeffs = ibass.'*signal;
```

A.16.5 gen_fbas.m

```
% gen_fbas.m

   function basis = gen_fbas( length, nharm )
% --- gen_fbas ---
% This function generates a basis of "Fourier" functions
% up to a certain order of harmonics.
% cos(0*theta)
% cos(1*theta)    sin(1*theta)
% cos(2*theta)    sin(2*theta)
% cos(3*theta)    sin(3*theta)
% ... up to
% cos(nharm*theta)    sin(nharm*theta)

% Create a vector of angles.
theta = 2*pi*(0:length-1).'/length;

basis = zeros(length, 2*nharm+1 );

basis(:,1) = ones(length,1);
kt=2;

for iharm=1:nharm
   basis(:,kt) = cos(iharm*theta);  kt=kt+1;
   basis(:,kt) = sin(iharm*theta);  kt=kt+1;
end
```

A.16.6 cyclic.m

```
% cyclic.m

clear
load cvf3a
load cvf6a
load cvf9a

torques(1,:) = cvf3a(:,3)';
torques(2,:) = cvf6a(:,3)';
torques(3,:) = cvf9a(:,3)';

vel_pwrs = [1 3 9; 1 6 36; 1 9 81];

params = inv(vel_pwrs) * torques;

save params params
```

A.16.7 torq_est.m

```
% torq_est.m

clear
load cvf3a

T= 0.0001;              % T is the time step 0.1ms
inertia = 0.5E-3;       % kg.m^2

vel = cvf3a(:,2);

for t=2:size(vel,1),
      accn(t) = (vel(t)-vel(t-1))/T;  % adequate for estimation
end

torque = accn * inertia;

figure
subplot(2,1,1), plot(vel, 'w'), grid, ylabel('Velocity');
subplot(2,1,2), plot(torque, 'w'), grid, ylabel('Torque');
xlabel('x100 us');
```

APPENDIX B

Appendix B DSP and PC Programs

B.1 Test-Rig Control Code

```
/* control.c
 *
 * This module contains the code to control the test rig, and by changing
 * the parameters in the file "ctl_pars", can be used for both emulation
 * and characterisation since the method for calculating the torque (and
 * angle) demand produces a torque reference only.
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include <link.h>
#include <math.h>
#include "..\ipopext\ipopext.h"
#include "..\timer\timer.h"
#include "..\misc\misc.h"
#include "..\prtrbtn\swpsin.h"
#include "..\prtrbtn\prbs.h"


int main(int argc, char *argv[]) {

    #define offset4_1   -0.056  /* (volts) to correct for cp4/ch1 offset */
    #define offset4_2   -0.056  /* (volts) to correct for cp4/ch2 offset */
    #define offset5_1   -0.139  /* (volts) to correct for cp5/ch1 offset */
    #define offset5_2   -0.140  /* (volts) to correct for cp5/ch2 offset */

    #define scale4_1    -1.0    /* rads/V */
    #define scale4_2    -297.5; /* rpm/V */
    #define scale5_1    5.0     /* Nm/V (200Nm/10V /4) */
    #define scale5_2    1.0

    #define scale0_3    1.0     /* spare output */
    #define scale0_4    1.0     /* spare output */
    #define scale2_3    1.0     /* 1.0V corresponds to 1Nm ??? */
    #define scale2_4    1.0     /* spare output */

    #define Btr         0.0     /* test-rig damping */
    #define Kfric       0.0     /* test-rig friction (assumed const) */

    /* these next variables are primarily for characterisation */
    unsigned int num_samples;   /* 10000 = 1 second at 10kHz sampling */
    unsigned int num_channels;  /* number of data channels to record */
    unsigned int test_number;   /* test to perform */
    unsigned long impulse_length;   /* for impulse test */
    double dec_per_sec;         /* for sweepsine test - decades per second */
    double start_freq;          /* for sweepsine test - start frequency */
    double stop_freq;           /* for sweepsine test - stop frequency */

    int cont_mode;              /* 1=continuous mode, overrides num_samples */

    char filename[20];          /* name of the file to save results */

    double Pang, Iang;          /* angle PI controller parameters */
    double torq_ang_ctrl;       /* for angle controller */
    double Ptorq, Itorq, Dtorq; /* torque PI controller parameters */

    double torq_damp_correction; /* for test-rig damping correction */
    double torq_fric_correction; /* for test-rig friction correction */

    double T;                   /* measured sample rate */
    unsigned long time=0;       /* elapsed time (units 100us) & array indexer */
    unsigned long index=0, n;   /* array indexer and array index offset */

    two_ch_data  adc_data_4, adc_data_5;

    double torq_req, ang_req;   /* generated by either:
                                   machine emulation model, or
                                   characterisation profile generation */

    /* allocate storage for history of signals */
    #define t 6                 /* present + t past values */
    double ang[t+1]={0};
    double vel[t+1]={0};
```

```
double torq[t+1]={0};
double torq_error[t+1]={0};
double torq_i_error[t+1]={0};
double torq_d_error;

double torq_out;

/* for data capturing */
double *data;                /* pointer to data array */
int data_size;               /* length of array */

FILE *fd;                    /* input and output file descriptor */

/* taylors expansion constants for diffrtn */
#define te_length 6          /* present + 10 past values */
const double pinvte[te_length] = {
                                        0.14285714285714,
                                        0.08571428571429,
                                        0.02857142857143,
                                       -0.02857142857143,
                                       -0.08571428571429,
                                       -0.14285714285714
                                                          };

/* GET CONTROL PARAMETERS FROM FILE "ctrlpars" */
fd = fopen("ctrlpars", "r");
if (fd == NULL){
    printf("file 'ctrlpars' does not exist\n");
    exit(1);
}

printf("\n");
fscanf(fd, "%d %*s ", &num_samples);
fscanf(fd, "%d %*s ", &cont_mode);
fscanf(fd, "%d %*s ", &num_channels);
fscanf(fd, "%s %*s ", filename);
printf("num_samples = %d\n", num_samples);
printf("cont_mode = %d\n", cont_mode);
printf("num_channels = %d\n", num_channels);
printf("filename = %s\n", filename);
printf("\n");

fscanf(fd, "%lf %*s ", &Pang);
fscanf(fd, "%lf %*s ", &Iang);
printf("P_angle = %lf\n", Pang);
printf("I_angle = %lf\n", Iang);
printf("\n");

fscanf(fd, "%lf %*s ", &Ptorq);
fscanf(fd, "%lf %*s ", &Itorq);
fscanf(fd, "%lf %*s ", &Dtorq);
printf("P_torque = %lf\n", Ptorq);
printf("I_torque = %lf\n", Itorq);
printf("D_torque = %lf\n", Dtorq);

printf("\n");
fclose(fd);

/* allocate memory for data logging */
data_size = num_channels * num_samples; /* no_channels * no_lines */
data = (double*)calloc((data_size + 6), sizeof(double));
if (data == NULL) {
    printf("not enough DSP memory available for recording data\n\n");
    exit(0);
}

/* initialize timer for sample rate measurement */
initialize_timer1();

/* setup the sampling etc... */
stop_sampling(4);               /* halt sampling via ADC comports */
stop_sampling(5);
timer_wait(200);                /* wait for DAC FIFO's to empty */
set_extern_trig(4);             /* use the external trigger I/P */
set_extern_trig(5);
adc_data_4 = empty_FIFO(4);     /* clear ADC FIFO's */
adc_data_5 = empty_FIFO(5);
reset_prbs();

return_to_continue();

/* start sampling & reset timer */
```

```
start_sampling(5);
start_sampling(4);
reset_timer1();


/* MAIN CONTROL LOOP */

while( (time<num_samples) || (cont_mode) ) {

    /* INPUT SIGNALS */
    adc_data_4 = read_next_voltages(4);     /* ch1 is angle     */
                                            /* ch2 is velocity  */
    adc_data_5 = read_next_voltages(5);     /* ch1 is torque    */
                                            /* ch2 is spare     */

    /* SCALE, FILTER AND ALLOCATE SIGNALS */
    ang[t] = (adc_data_4.ch1 + offset4_1) * scale4_1;
    vel[t] = (adc_data_4.ch2 + offset4_2) * scale4_2;
    torq[t] = (adc_data_5.ch1 + offset5_1) * scale5_1;
    /*spare[t] = (adc_data_5.ch2 + offset5_2) * scale5_2;*/

    /* filter torque signal further as necessary [b,a] = butter (3,1500/5000) */
    /*
    #define bt1  0.04953299635725
    #define bt2  0.14859898907176
    #define bt3  0.14859898907176
    #define bt4  0.04953299635725
    #define at2 -1.16191748367173
    #define at3  0.69594275578965
    #define at4 -0.13776130125989
    torq[t] = bt1*torq[t]+bt2*torq[t-1]+bt3*torq[t-2]+bt4*torq[t-3]
                        -at2*torq[t-1]-at3*torq[t-2]-at4*torq[t-3];
    */

    /* don't filter angle signal */

    /* filter velocity signal further as necessary [b,a] = butter (3,1500/5000) */
    /*
    #define bv1  0.04953299635725
    #define bv2  0.14859898907176
    #define bv3  0.14859898907176
    #define bv4  0.04953299635725
    #define av2 -1.16191748367173
    #define av3  0.69594275578965
    #define av4 -0.13776130125989
    vel[t] = bv1*vel[t]+bv2*vel[t-1]+bv3*vel[t-2]+bv4*vel[t-3]
                      -av2*vel[t-1]-av3*vel[t-2]-av4*vel[t-3];
    */


    /* CALCULATE REQUIRED TORQUE AND ANGLE; torq_req AND ang_req */
    /* i.e. for charac - put perturbation profile generator here,
       and for emulation - call model() from here                  */
    torq_req = 0.0;
    ang_req = 0.0;


    /* ANGLE 'PI' CONTROLLER
       - calculate torque demand due to angel error (for charac only) */
    /* torq_ang_dem = (ang_req - ang[t]) * Pang; */
    /* need to put some thought into this - I need angle difference, and
       ang[t] is in the range 0 to 2*pi */
    /*torq_ang_ctrl = ang_req;*/
    torq_ang_ctrl = 0.0; /* for now */


    /* CALCULATE DAMPING AND FRICTION COMPESATION */
    torq_damp_correction = vel[t] * Btr;
    torq_fric_correction = ((vel[t]>0) ? 1 : -1) * Kfric;


    /* TORQUE 'PID' CONTROLLER - calculate torque demand */
    /* calculate error */
    torq_error[t] = torq_req + torq_ang_ctrl - torq[t];

    /* integrate error (trapezoidal) for I term */
    torq_i_error[t] = torq_i_error[t-1] + (T/2)*(torq_error[t]+torq_error[t-1]);

    /* differentiate error (backward-finite-divided-difference) for D term */
    torq_d_error = 0;
    for(n=0; n<te_length; n++){
```

```
        torq_d_error += torq_error[t-n] * pinvte[n];
    }
    torq_d_error /= T;

    /* control algorithm */
    torq_out = Ptorq*torq_error[t] + Itorq*torq_i_error[t] + Dtorq*torq_d_error
            - torq_damp_correction - torq_fric_correction;


    /* SCALE AND OUTPUT THE OUTPUT SIGNALS */
    output_voltage(2, 3, (torq_out * scale2_3) );
    /* output_voltage(2, 4, 0.0); */
    /* output_voltage(0, 3, 0.0); */
    /* output_voltage(0, 4, 0.0); */


    /* TIME SAMPLING RATE */
    T = timer1_time();  /* T = sample rate */
    reset_timer1();
    time++;             /* increment elapsed time, x100us if fs = 10kHz */


    /* RECORD DATA FOR RETRIEVAL LATER (if not in continuous mode) */
    if (!cont_mode) {
        data[index] = ang[t];
        data[index+1] = vel[t];
        data[index+2] = torq[t];
        data[index+3] = T;
        index += num_channels;
    }


    /* RIPPLE OLD VARIABLES DOWN THE LINE AND STORE CURRENT ONES */
    vel[t-3] = vel[t-2];
    vel[t-2] = vel[t-1];
    vel[t-1] = vel[t];

    torq[t-3] = torq[t-2];
    torq[t-2] = torq[t-1];
    torq[t-1] = torq[t];

    torq_error[t-6] = torq_error[t-5];
    torq_error[t-5] = torq_error[t-4];
    torq_error[t-4] = torq_error[t-3];
    torq_error[t-3] = torq_error[t-2];
    torq_error[t-2] = torq_error[t-1];
    torq_error[t-1] = torq_error[t];

    torq_i_error[t-1] = torq_i_error[t];

    index += num_channels;

} /* end while (END OF EXPERIMENT) */

/* output nothing */
output_voltage(2, 3, 0.0);
output_voltage(2, 4, 0.0);
output_voltage(0, 3, 0.0);
output_voltage(0, 4, 0.0);

printf("test complete");

/* WRITE DATA TO A FILE ON PC */
if (!cont_mode) {               /* if not in continuous mode */
    printf(", writing to file...\n");
    fd = fopen(filename, "w");
    if (fd == NULL) {
        printf("cannot open '%s' for writing.\n", filename);
        exit(1);
    }
    fprintf(fd, "\n");
    for (index=0; index<data_size; index+=num_channels){
        for (n=0; n<num_channels; n++){
            /* store data in file in text format */
            fprintf(fd, "%10f ", data[index+n]);
        }
        fprintf(fd, "\n");
    }
    fclose(fd);
} /* enf if(!cont_mode) */

printf("\n");
```

```
    return(0);
}
```

B.2 PRBS Generation

```
/*
 * prbs.c - DW 19/9/97
 *
 * Returns the next binary output in a 32-bit pseudo random
 * binary sequence. The internal state is static so that
 * successive calls form the sequence.
 *
 * other variables are static to speed-up function calling
 */

#include "prbs.h"

#include <stdlib.h>
#include <link.h>

static unsigned long shift_regs = 0x10000000;


double prbs8() {
      static int fb, s4, s5, s6, s8;

      if (shift_regs & 0x10000000)  s4 = 1;
      else                          s4 = 0;

      if (shift_regs & 0x08000000)  s5 = 1;
      else                          s5 = 0;

      if (shift_regs & 0x04000000)  s6 = 1;
      else                          s6 = 0;

      if (shift_regs & 0x01000000)  s8 = 1;
      else                          s8 = 0;

      fb = s4 ^ (s5 ^ (s6 ^ s8));

      if ((!s6 && s8) || (s6 && !s8))     fb = 1;
      else                                fb = 0;

      if ((!fb && s5) || (fb && !s5))     fb = 1;
      else                                fb = 0;

      if ((!fb && s4) || (fb && !s4))     fb = 1;
      else                                fb = 0;

      shift_regs >>= 1;

      if (fb==1)
            shift_regs |= 0x80000000;     /* set msb */
      else
            shift_regs &= 0x7FFFFFFF;     /* clear msb */

      if (shift_regs & 0x01000000)        /* if lsb == 1 */
            return(1.0);
      else
            return(-1.0);
}


double prbs16() {
      static int fb, s4, s13, s15, s16;

      if (shift_regs & 0x10000000)  s4 = 1;
      else                          s4 = 0;

      if (shift_regs & 0x00080000)  s13 = 1;
      else                          s13 = 0;

      if (shift_regs & 0x00020000)  s15 = 1;
      else                          s15 = 0;

      if (shift_regs & 0x00010000)  s16 = 1;
      else                          s16 = 0;

      fb = s4 ^ (s13 ^ (s15 ^ s16));

      if ((!s15 && s16) || (s15 && !s16)) fb = 1;
```

```
      else                                  fb = 0;

      if ((!fb && s13) || (fb && !s13))     fb = 1;
      else                                  fb = 0;

      if ((!fb && s4) || (fb && !s4))       fb = 1;
      else                                  fb = 0;

      shift_regs >>= 1;

      if (fb==1)
            shift_regs |= 0x80000000;       /* set msb */
      else
            shift_regs &= 0x7FFFFFFF;       /* clear msb */

      if (shift_regs & 0x00010000)          /* if lsb == 1 */
            return(1.0);
      else
            return(-1.0);
}


double prbs32() {
      static int fb, s10, s30, s31, s32;

      if (shift_regs & 0x00400000)  s10 = 1;
      else                          s10 = 0;

      if (shift_regs & 0x00000004)  s30 = 1;
      else                          s30 = 0;

      if (shift_regs & 0x00000002)  s31 = 1;
      else                          s31 = 0;

      if (shift_regs & 0x00000001)  s32 = 1;
      else                          s32 = 0;

      fb = s10 ^ (s30 ^ (s31 ^ s32));

      if ((!s31 && s32) || (s31 && !s32)) fb = 1;
      else                                fb = 0;

      if ((!fb && s30) || (fb && !s30))   fb = 1;
      else                                fb = 0;

      if ((!fb && s10) || (fb && !s10))   fb = 1;
      else                                fb = 0;

      shift_regs >>= 1;

      if (fb==1)
            shift_regs |= 0x80000000;       /* set msb */
      else
            shift_regs &= 0x7FFFFFFF;       /* clear msb */

      if (shift_regs & 0x00000001)          /* if lsb == 1 */
            return(1.0);
      else
            return(-1.0);
}


void reset_prbs() {
      shift_regs = 0x10000000;
}
```

B.3 Swept-Sine Generation

```
/*
 * swpsin.c - DW 19/9/97
 *
 * Returns the value of a swept sine wave after t seconds.
 * x = sweepsin(rate, f0, f_max, t);
 *       f0   = start frequency,
 *       rate = decades/sec,
 *       f increases exponentially at 'rate' decades/sec.
 *       0 is returned if f_max is exceeded.
 *
 * to compile use : cl30 -v40 -c -s -g -al -mr swpsin.c
 */
```

```
#include "swpsin.h"

#include <stdlib.h>
#include <link.h>
#include <math.h>

#define pi              3.1415926536


double sweepsin(double rate, double f0, double f_max, double t) {
      double k, angl, freq;

      k = rate * 2.30258509299405;              /* k = rate * log(10.0); */
      freq = f0 * pow(10.0, (rate * t));
      if (freq>f_max)
            return(0.0);                              /* return 0.0 if fmax exceeded */
      angl = 2.0 * pi * (f0/k) * exp(k * t);    /* why 2* pi * ??? */
      return(sin(angl));
}
```

B.4 Cam Emulation

```
/* cam.c - DW 8/10/97
 * 1st load emulation on this rig config
 * - a cam + sprung cam follower.
*/

#include <stdio.h>
#include <stdlib.h>
#include <link.h>
#include <math.h>
#include "..\ipopext1\ipopext.h"
#include "..\timerint\timer.h"
#include "functns.h"

#define TRUE            1
#define FALSE     0
#define pi        3.1415926536


int main(int argc, char *argv[]){
      /* define macros & variables etc... */
      two_ch_data  adc_data_4, adc_data_5;
      #define offset4_1  0.0732     /* (volts) to correct for cp4/ch1 offset */
      #define offset4_2 -0.04       /* (volts) to correct for cp4/ch2 offset */
      #define offset5_1 -0.2300     /* (volts) to correct for cp5/ch1 offset */
      #define offset5_2 -0.001            /* (volts) to correct for cp5/ch2 offset */
      #define scale4_1   0.6377     /* -0.6471 rads/V */
      #define scale4_2  -625.64     /* rpm/V */
      #define scale5_1   5.33       /* Nm/V (200Nm/5V /4) */
      #define scale5_2   7.7257     /* Amps / Volt */
      #define A          0.2926     /* (rad/s^2)/amp */
      #define B         -0.5093     /* (rad/s^2)/Nm */
      #define ks         1.0        /* spring constant */
      #define pi         3.1415926536
      #define t 6                   /* present + t past values +1 */
      double ang[t]={0};
      double vel[t]={0};
      double torq[t]={0};
      double current[t]={0};
      double accn[t]={0};
      double error[t]={0};
      double i_error[t]={0};
      double d_error;
      double dem_torq=0.0, peak_torq=0.0, torq_out=0.0, voltage_out=0.0;
      double freq=0.0;              /* swept sin frequency */
      double P, I, D;               /* PID controller parameters */
      double T;                     /* measured sample rate */
      char C;                       /* to weed out text from PID file */
      FILE *in_file;
      int n;
      unsigned long int no_channels, data_size, index=0;
      long double cumulative_time=0.0;
      double *data;
      char filename[30];
      FILE *out_file;

      /* taylors expansion constants for diffrtn */
      #define te_length 6           /* present + 10 past values */
      const double pinvte[te_length] = {
                                    0.14285714285714,
```

```
                                        0.08571428571429,
                                        0.02857142857143,
                                       -0.02857142857143,
                                       -0.08571428571429,
                                       -0.14285714285714
                                        };

      /* setup the sampling etc... */
      set_extern_trig(4);
      set_extern_trig(5);
      adc_data_4 = empty_FIFO(4);
      adc_data_5 = empty_FIFO(5);

      /* get peak_torq, P, I and D parameters from file "PID" */
      in_file = fopen("PID", "r");
      if (in_file == NULL){
            printf("file 'PID' does not exist\n");
            exit(1);
      }
      fscanf(in_file, "%c %f %c %f %c %f %c %f", &C, &peak_torq, &C, &P, &C, &I, &C,
&D);
      printf("peak_torq = %f, P = %f, I = %f, D = %f\n", peak_torq, P, I, D);
      fclose(in_file);

      /* initialize timer for sample rate measurement */
      initialize_timer1();

      /* allocate memory for data logging */
      no_channels = 4;
      data_size = no_channels * 100001;   /* no_channels * no_lines        */
                                          /* 2 * 10000 => 1 second @ 10KHz */
      data = (double*)calloc(data_size, sizeof(double));
      if (data == NULL){
            printf("not enough DSP memory available for that many lines\n\n");
            exit(0);
      }

      /* initialize data to 0 'cos system doesn't !? */
      for (index=0; index<data_size; index++){
            data[index]=0.0;
      }
      index=0;

      return_to_continue();

      /* start sampling & reset timer */
      start_sampling(4);
      start_sampling(5);
      reset_timer1();

      /* control loop bit */
      while(TRUE){
            /* input signals */
            adc_data_4 = read_next_voltages(4); /* ch1 is angle */
                                                /* ch2 is velocity */
            adc_data_5 = read_next_voltages(5); /* ch1 is torque */
                                                /* ch2 is spare */
            /* time sampling rate */
            T = timer1_time();      /* T = sample rate (secs) */
            reset_timer1();
            cumulative_time += T;

            /* scale and allocate signals */
            ang[t] = (adc_data_4.ch1 + offset4_1) * scale4_1;
            vel[t] = (adc_data_4.ch2 + offset4_2) * scale4_2;
            torq[t] = (adc_data_5.ch1 + offset5_1) * scale5_1;
            current[t] = (adc_data_5.ch2 + offset5_2) * scale5_2;

            /* don't filter angle signal */

            /* filter velocity signal as necessary */
            #define bv1  0.29146494465726E-4    /* [b,a] = butter (3,100/5000) */
            #define bv2  0.87439483397844E-4
            #define bv3  0.87439483396068E-4
            #define bv4  0.29146494466281E-4
            #define av2 -2.87435689267748
            #define av3  2.75648319522570
            #define av4 -0.88189313059249
            vel[t] = bv1*vel[t]+bv2*vel[t-1]+bv3*vel[t-2]+bv4*vel[t-3]
                                    -av2*vel[t-1]-av3*vel[t-2]-av4*vel[t-3];

            /* filter torque signal as necessary */
```

```
        #define bt1  0.29146494465726E-4    /* [b,a] = butter (3,100/5000) */
        #define bt2  0.87439483397844E-4
        #define bt3  0.87439483396068E-4
        #define bt4  0.29146494466281E-4
        #define at2 -2.87435689267748
        #define at3  2.75648319522570
        #define at4 -0.88189313059249
        torq[t] = bt1*torq[t]+bt2*torq[t-1]+bt3*torq[t-2]+bt4*torq[t-3]
                                    -at2*torq[t-1]-at3*torq[t-2]-at4*torq[t-3];

        /* don't filter current signal */

        /* calculate shaft acceleration */
        accn[t] = A*current[t] + B*torq[t];

        /* calculate dem_torq for 'cam + sprung cam follower' */
        dem_torq = peak_torq * ks * sin(ang[t]);

        /* calculate error */
        error[t] = dem_torq - torq[t];

        /* integrate error (trapezoidal) for I term */
        i_error[t] = i_error[t-1] + (T/2)*(error[t]+error[t-1]);

        /* differentiate error (finite-divided-difference) for D term */
        d_error = 0;
        for(n=0; n<=(te_length-1); n++){
              d_error += error[t-n] * pinvte[n];
        }
        d_error *= 1.0/T;

        /* control algorithm */
        torq_out = dem_torq;
        /*torq_out = P*error[t] + I*i_error[t] + D*d_error;*/
        voltage_out = torq_out * 1.0; /* ??? O/P is actually torq demand */

        /* output the motors torque demand signal */
        output_voltage(2, 3, voltage_out);

        /* record data for retrieval later */
        /*data[index] = ang[t];          */
        /*data[index+1] = vel[t];        */
        /*data[index+2] = torq[t];       */
        /*data[index+3] = current[t]; */
        /*index += no_channels;          */

        /* ripple old variables down the line as required */
        vel[t-3] = vel[t-2];
        vel[t-2] = vel[t-1];
        vel[t-1] = vel[t];

        torq[t-3] = torq[t-2];
        torq[t-2] = torq[t-1];
        torq[t-1] = torq[t];

        /* current[t-3] = current[t-2]; */
        /* current[t-2] = current[t-1]; */
        /* current[t-1] = current[t]; */

        /* accn[t-3] = accn[t-2]; */
        /* accn[t-2] = accn[t-1]; */
        /* accn[t-1] = accn[t];*/

        error[t-6] = error[t-5];
        error[t-5] = error[t-4];
        error[t-4] = error[t-3];
        error[t-3] = error[t-2];
        error[t-2] = error[t-1];
        error[t-1] = error[t];

        i_error[t-1] = i_error[t];
    } /* end while (end of control loop) */

    /* stop motors */
    output_voltage(2, 3, 0.0);

    printf("\a");     /* beep when done */
    return(0);
}
```

B.5 Electrical Machine Perturbation

```
/* ele_mod1.c - DW 5/9/99
 *
 * This program perturbs electrical machines similar to the following
 * using the current drive, with a step, impulse or chirp signal.
 *
 *                     C1      L1       C2
 *  O--->--------------------/\/\/\/\-----
 *    i I/P   ^        |                 |
 *            |        |                 |
 *            |       ---               ---
 *         v O/P      ---               ---
 *            |        |                 |
 *            |        |                 |
 *  O-----------------------------------
 *
 *  DAC O/P is proportional to plant current I/P and
 *  ADC I/P is proportional to plant voltage output.
 *  An external ADC/DAC trigger input is used (set to 10kHz).
 *  The input and output signals are logged for off-line data analysis.
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include <link.h>
#include <math.h>
#include "..\ipopext\ipopext.h"
#include "..\timer\timer.h"
#include "..\misc\misc.h"
#include "..\prtrbtn\swpsin.h"
#include "..\prtrbtn\prbs.h"


int main(int argc, char *argv[]) {

      #define offset5_1 -0.139      /* (volts) to correct for cp5/ch1 offset */
      #define offset5_2 -0.140      /* (volts) to correct for cp5/ch2 offset */
      #define offset4_1 -0.056      /* (volts) to correct for cp4/ch1 offset */
      #define offset4_2 -0.056      /* (volts) to correct for cp4/ch2 offset */

      #define scale5_1   1.0        /* volts input */
      #define scale5_2   1.0        /* spare input */
      #define scale4_1   1.0        /* spare input */
      #define scale4_2   1.0        /* spare input */

      #define scale2_3   1.25       /* 2.5 V p/p  (1.7 mA/V output ~ 4.25mA p/p) */
      #define scale2_4   1.0        /* spare output */
      #define scale0_3   1.0        /* spare output */
      #define scale0_4   1.0        /* spare output */

      unsigned int num_samples;     /* 10000 = 1 second at 10kHz sampling */
      unsigned int num_channels;    /* number of data channels to record */
      unsigned int test_number;     /* test to perform */

      two_ch_data  adc_data_4, adc_data_5;

      unsigned long impulse_length; /* for impulse test */

      double dec_per_sec;           /* for sweepsine test - decades per second */
      double start_freq;                  /* for sweepsine test - start frequency */
      double stop_freq;             /* for sweepsine test - stop frequency */

      double T;                     /* measured sample rate */
      unsigned long t=0;                  /* elapsed time (units 100us) & array indexer
*/
      unsigned long index=0, n;     /* array indexer and array index offset */

      double i_ref=0.0;             /* current reference to circuit */
      double *data;                 /* pointer to data array */
      int data_size;                /* length of array */

      char filename[30];                  /* name of the file to save results */
      char tmpstr[30];              /* for reading parameters from file */
      FILE *fd;                     /* input and output file descriptor */


      /* get parameters from file "params" */
      fd = fopen("params", "r");
      if (fd == NULL){
            printf("file 'params' does not exist\n");
```

```
      exit(1);
}
printf("\n");
fscanf(fd, "%d %s ", &num_samples, tmpstr);
printf("num_samples = %d\n", num_samples);
fscanf(fd, "%d %s ", &num_channels, tmpstr);
printf("num_channels = %d\n", num_channels);
fscanf(fd, "%d %s ", &test_number, tmpstr);
printf("test_number = %d\n", test_number);
fscanf(fd, "%d %s ", &impulse_length, tmpstr);
printf("impulse_length = %d\n", impulse_length);
fscanf(fd, "%f %s ", &dec_per_sec, tmpstr);
printf("dec_per_sec = %.4f\n", dec_per_sec);
fscanf(fd, "%f %s ", &start_freq, tmpstr);
printf("start_freq = %.4f\n", start_freq);
fscanf(fd, "%f %s ", &stop_freq, tmpstr);
printf("stop_freq = %.4f\n", stop_freq);
fscanf(fd, "%s %s ", filename, tmpstr);
printf("filename = %s\n", filename);
printf("\n");
fclose(fd);

/* display selected test (perturbation signal) */
if ((test_number < 1)||(test_number > 6)) {
      printf("invalid choice of perturbation signal\n\n");
      return(1);
}
printf("Selected perturbation signal: ");
if (test_number==1) printf(">"); else printf(" ");
printf(" 1 = impulse\n                                    ");
if (test_number==2) printf(">"); else printf(" ");
printf(" 2 = step\n                                       ");
if (test_number==3) printf(">"); else printf(" ");
printf(" 3 = swept-sine\n                                       ");
if (test_number==4) printf(">"); else printf(" ");
printf(" 4 = PRBS\n                                      ");
printf("\n");

/* allocate memory for data logging */
data_size = num_channels * (num_samples+1);     /* no_channels * no_lines */
data = (double*)calloc(data_size, sizeof(double));
if (data == NULL) {
      printf("not enough DSP memory available for recording data\n\n");
      exit(0);
}

/* initialize timer for sample rate measurement */
initialize_timer1();

/* setup the sampling etc... */
stop_sampling(4);             /* halt sampling via ADC comports */
stop_sampling(5);
timer_wait(200);              /* wait for DAC FIFO's to empty */
set_extern_trig(4);           /* use the external trigger I/P */
set_extern_trig(5);
adc_data_4 = empty_FIFO(4);   /* clear ADC FIFO's */
adc_data_5 = empty_FIFO(5);
reset_prbs();

/* initial conditions (of input signals) */
adc_data_4.ch1 = 0.0;         /* ch1 is spare */
adc_data_4.ch2 = 0.0;         /* ch2 is spare */
adc_data_5.ch1 = 0.0;         /* ch1 is electrical cct output volts */
adc_data_5.ch2 = 0.0;         /* ch2 is spare */

/*return_to_continue();*/

/* start sampling & reset timer */
start_sampling(5);
start_sampling(4);
reset_timer1();


/* calculate and output the perturbation signal (plant input),
   measure the corresponding input (plant output) & store both    */
while(t<num_samples) {

      /* calculate current reference for signal-type and time */
      switch (test_number) {
            case 1:                 /* impulse - max O/P for 10 step, then 0 */
            if (t<impulse_length)
                        i_ref = 1.0;
```

```
                else
                        i_ref = 0.0;
                break;
        case 2:                 /* step - O/P was zero before 1st iteration */
        i_ref = 1.0;
                break;
        case 3:                 /* swept-sine, 2 decs/s, start f=10 Hz */
        i_ref = sweepsin(dec_per_sec, start_freq, stop_freq, t*0.0001);
                break;
        case 4:                 /* swept-sine, 2 decs/s, start f=10 Hz */
        i_ref = prbs16();
                break;
        /* the next two tests are for software development only */
        case 5:                 /* test to determine the TPS board latency */
                if (t>10)
                        i_ref = 1.0;
                if (t>15)
                        i_ref = 0.0;
                if (t>20) {
                        if ( (t%2) != 0)
                                i_ref = 1.0;
                        else
                                i_ref = 0.0;
                }
                if (t>30)
                        i_ref = 0.0;
                break;
        case 6:                 /* continuous square wave, f = sample / 2 */
                if ((t%2) != 0)
                        i_ref = 1.0;
                else
                        i_ref = 0.0;
                if (t==98)
                        t = 0;          /* this makes the sqr wave O/P continuous */
                break;
    }

    /* scale and output the 'current' demand signal, and spares */
    output_voltage(2, 3, (-i_ref * scale2_3) );     /* fed into inverting amp */
    output_voltage(2, 4, 0.0);
    output_voltage(0, 3, 0.0);
    output_voltage(0, 4, 0.0);

    /* input signals and remove offsets */
    adc_data_5 = read_next_voltages(5); /* ch1 is voltage */
                                        /* ch2 is spare */
    adc_data_4 = read_next_voltages(4); /* ch1 is spare */
                                        /* ch2 is spare */

    /* time sampling rate */
    T = timer1_time();          /* T = sample rate */
    reset_timer1();

    /* scale and store signals */
    data[index+0] = i_ref * scale2_3;
    data[index+1] = (adc_data_5.ch1 + offset5_1) * scale5_1;

    index += num_channels;

    t++;  /* increment elapsed time, assuming fs = 10kHz */

} /* end while (end of experiment) */

/* output nothing */
output_voltage(2, 3, 0.0);
output_voltage(2, 4, 0.0);
output_voltage(0, 3, 0.0);
output_voltage(0, 4, 0.0);

/* advance ADC values one sample since the inputs are deleyed
   by one sample due to a TPS DSP board 'peculiarity' */
for (index=0; index<data_size; index+=num_channels){
        data[index+1] = data[index+3];
}
data_size -= num_channels;

printf("test complete, writing to file...\n");

/* write data to a file on PC */
/*get_file_name(filename);*/
fd = fopen(filename, "w");
```

```
        for (index=0; index<data_size; index+=num_channels){
              for (n=0; n<num_channels; n++){
                    fprintf(fd, "%10f ", data[index+n]); /* store data on file */
              }
              fprintf(fd, "\n");
        }

        fclose(fd);
        printf("\n\a");          /* beep when done */
        return(0);
}
```

APPENDIX C

Appendix C Test-Rig Mechanical Hardware

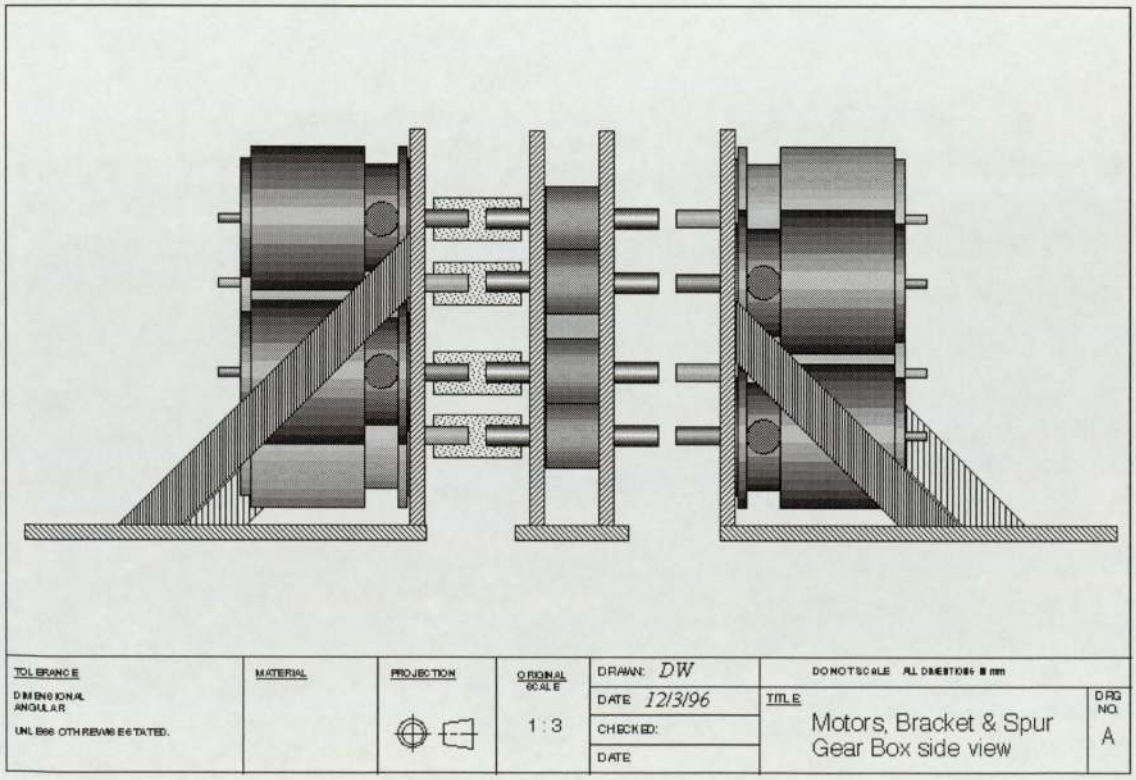
This appendix provides practical design details of the test-rig.

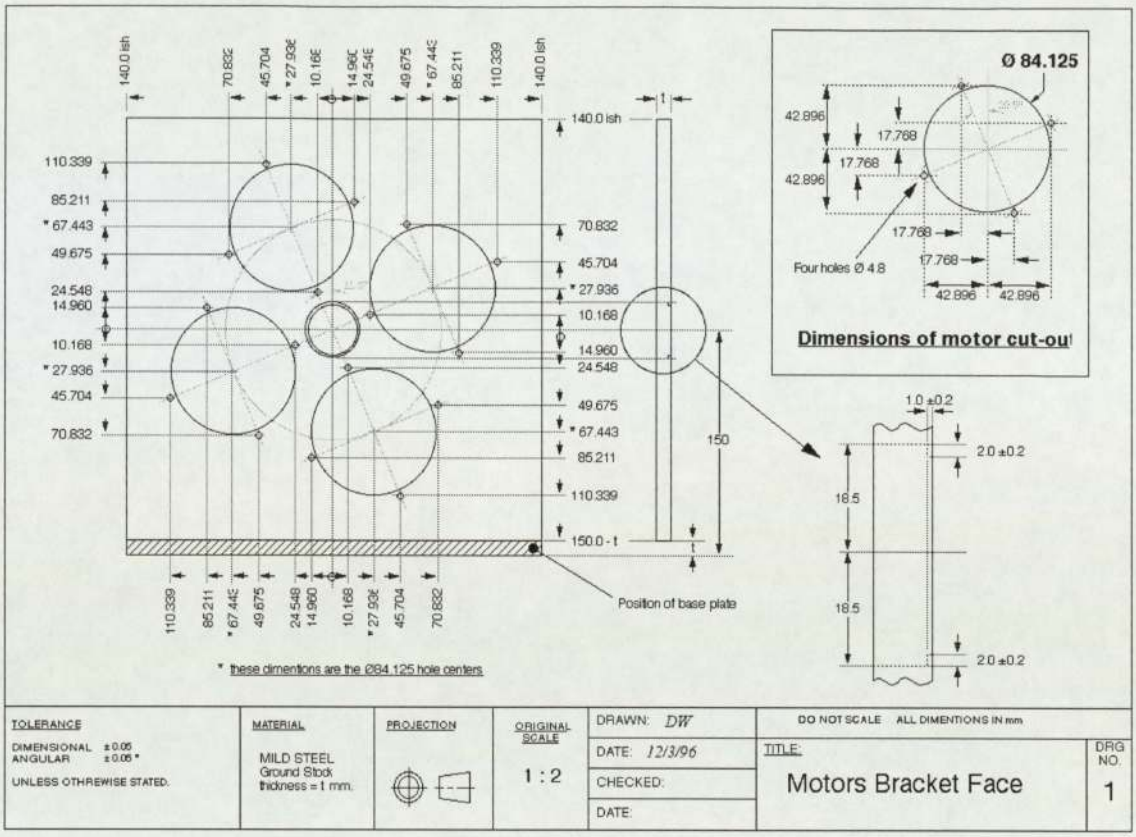
C.1 Hardware Procured

Equipment	Manufacturer	Model No.	Spec
Shaft Encoder	Honher	88 20362D	2048 line hollow
Torque-Transducer	Vibro-meter	TM212	200Nm
Tacho-Generator	Minimotor	1624 T 1,4 G9	1.4mV/rpm
Servo Drives	Norwin	1770	20 / 40 A max / peak

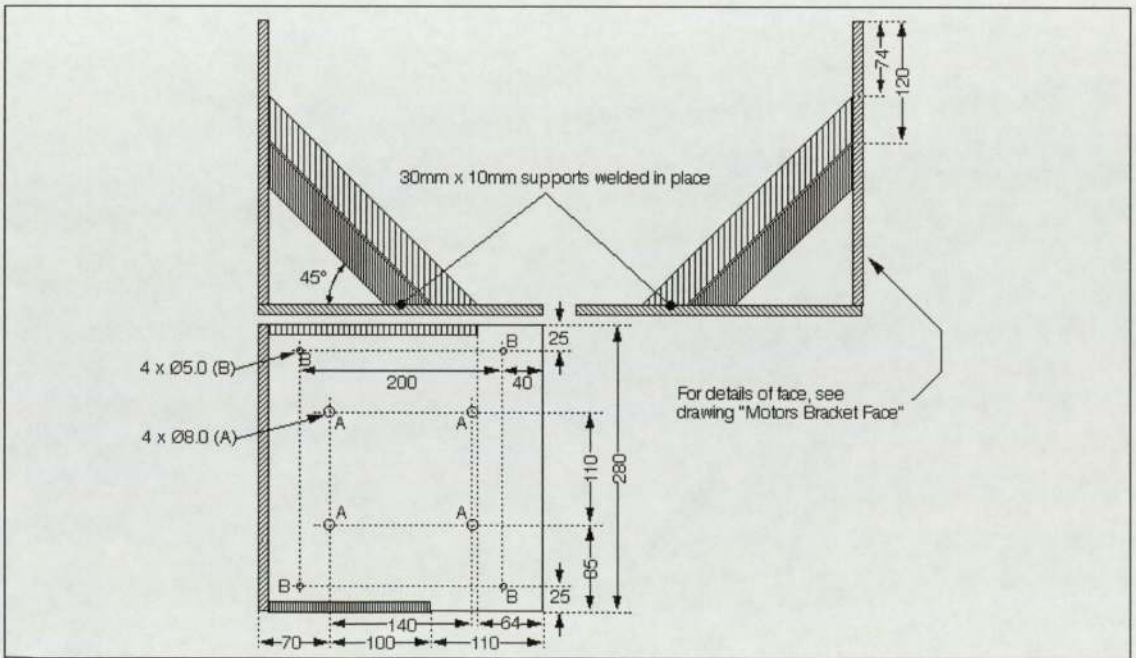
C.2 Test-Rig Mechanical Design Drawings

C.2.1 Motors Bracket for Spur Gear Gearbox





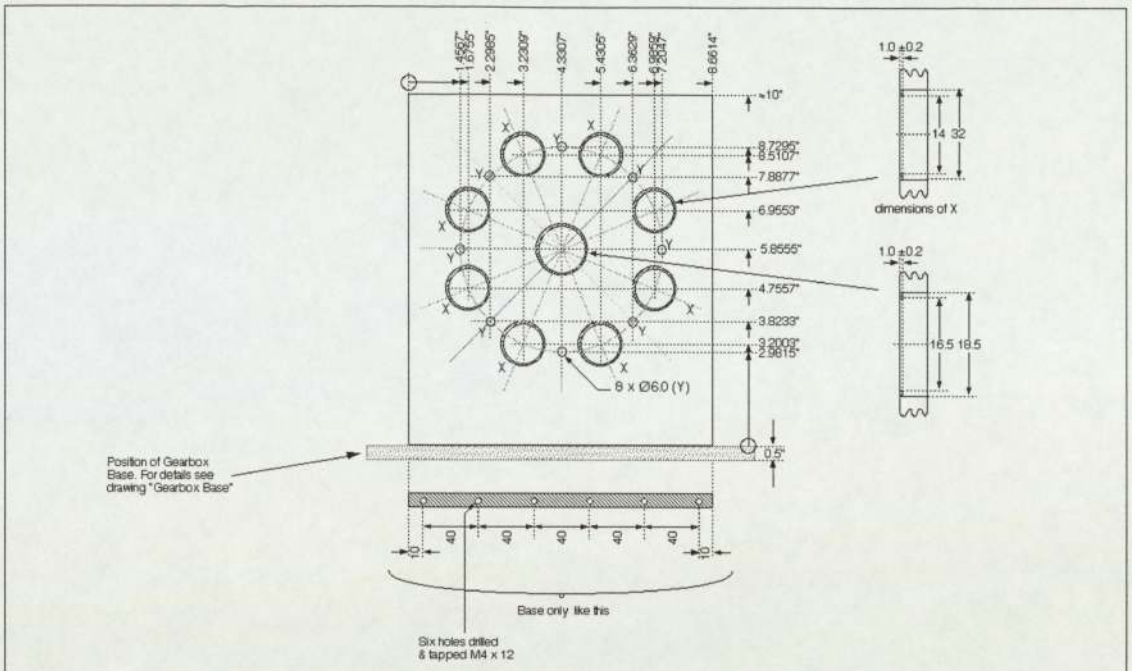
TOLERANCE DIMENSIONAL ±0.05 ANGULAR ±0.05 ° UNLESS OTHREWISE STATED.	MATERIAL MILD STEEL Ground Stock thickness = t mm.	PROJECTION	ORIGINAL SCALE 1 : 2	DRAWN: DW DATE: 12/3/96 CHECKED: DATE:	DO NOT SCALE ALL DIMENTIONS IN mm TITLE: Motors Bracket Face DRG NO. 1



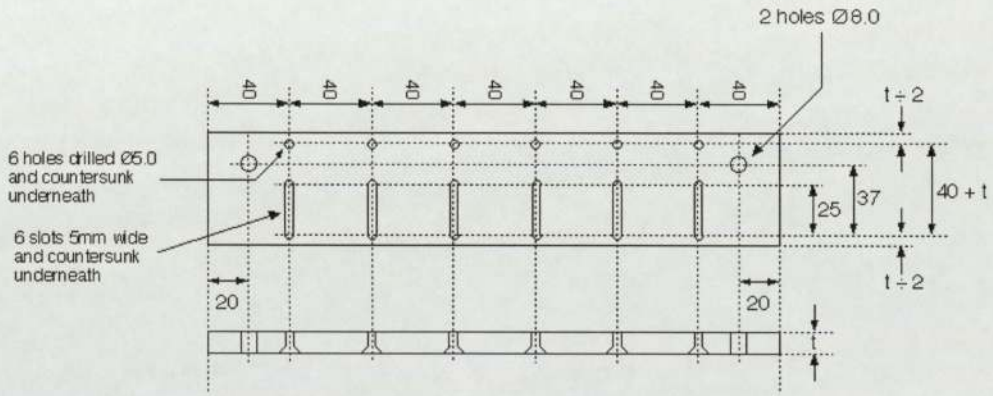
TOLERANCE DIMENSIONAL ±0.05 ANGULAR ±0.05 ° UNLESS OTHREWISE STATED.	MATERIAL MILD STEEL 't' mm thick	PROJECTION	ORIGINAL SCALE 1 : 4	DRAWN: DW DATE: 12/3/96 CHECKED: DATE:	DO NOT SCALE ALL DIMENTIONS IN mm TITLE: Motor Bracket Base & Sides DRG NO. 2

C.2.2 Spur Gear Gearbox

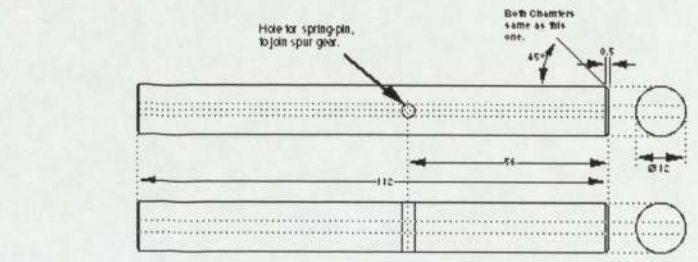
Item No.	Part Description	Drwg No	Qty Reqd	Source
Items built at Aston:				
1	Gearbox Faces	1	2	Aston
2	Gearbox Base	2	1	Aston
3	Gearbox Input Shaft	3	4	Aston
4	Gearbox Output Shaft	3	1	Aston
Standard parts:				
5	(Input Shaft) Bearings	-	8	NACHI (P/N 6201ZZ)
6	(Output Shaft) Bearings	-	2	NACHI (P/N 6904ZZ)
7	Centre Gear (with boss)	4	1	HPC (P/N G1-103/T)
8	Centre Gear	4	1	HPC (P/N PG1-103/T)
9	Planet Gears (with boss)	4	8	HPC (P/N G1-43/T)
10	Planet Gears	4	8	HPC (P/N PG1-43/T)
11	Spacers	-	10	HPC (P/N HSE 6-35)
12	Hard Steel Shaft Ø15 (I/P)	-	1 m +?	
13	Hard Steel Shaft Ø20 (I/P)	-	200mm+?	



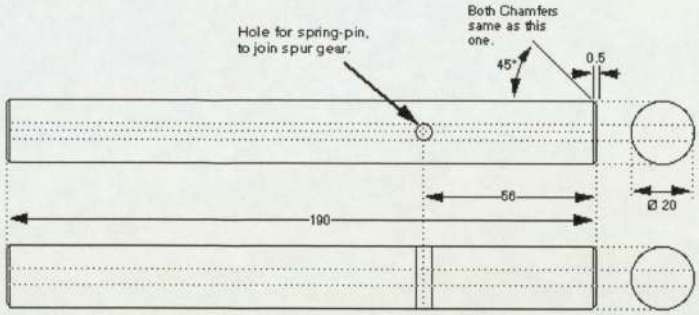
TOLERANCE	MATERIAL	PROJECTION	ORIGINAL SCALE	REVISION: 1.3	DO NOT SCALE ALL DIMENTIONS IN mmUNLESS OTHERWISE STATED	
DIMENSIONAL ±0.05 ANGULAR ±0.05° UNLESS OTHREWISE STATED.	MILD STEEL Ground Stock 10 mm thick.		1 : 2	DRAWN: DW DATE: 14/5/96 CHECKED: DATE:	TITLE: Gearbox Faces	DRG NO. 1



TOLERANCE	MATERIAL	PROJECTION	ORIGINAL SCALE	REVISION: V1.2	DO NOT SCALE ALL DIMENTIONS IN mm	
DIMENSIONAL ±0.2 ANGULAR ±1° UNLESS OTHREWISE STATED.	MILD STEEL Ground Stock 't' mm thick		1 : 2	DRAWN: DW DATE: 13/5/96 CHECKED: DATE:	TITLE: Gearbox Base	DRG NO. 2



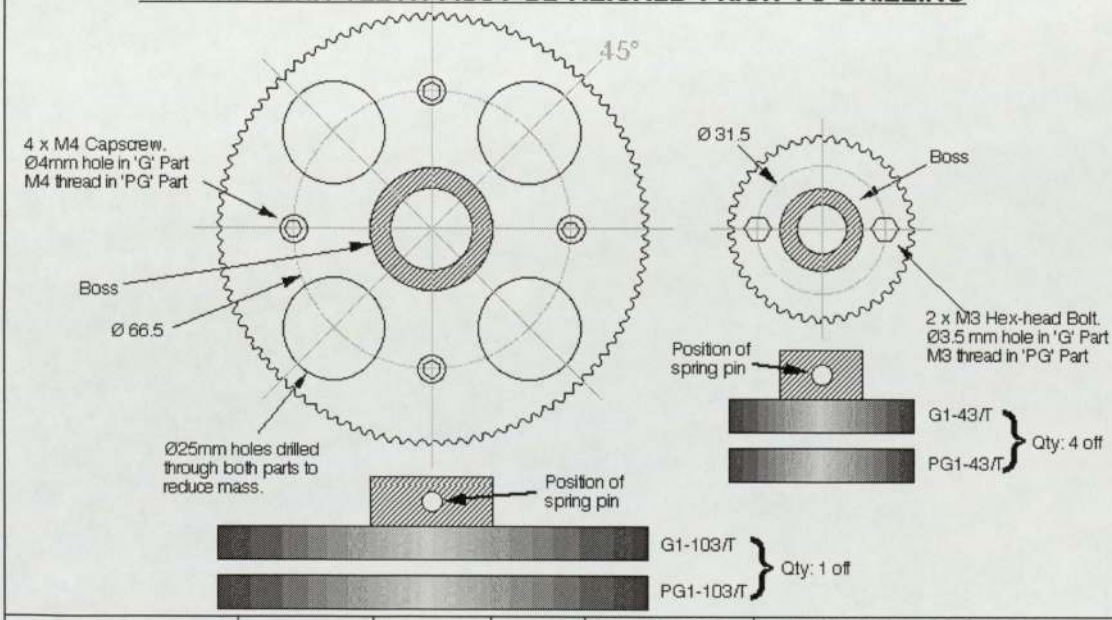
Input Shaft
(original scale 1:1)
Qty: 4 off



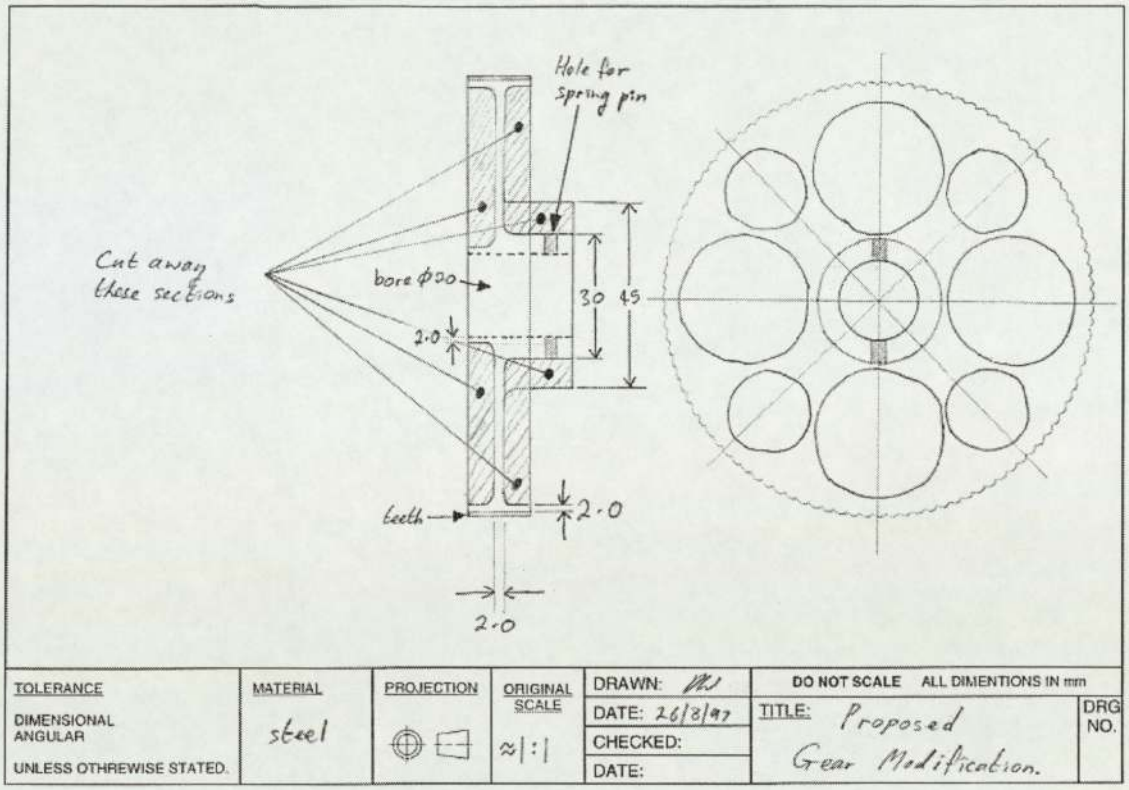
Output Shaft
(original scale 1:1.5)
Qty: 1 off

TOLERANCE	MATERIAL	PROJECTION	ORIGINAL SCALE	DRAWN: DW	DO NOT SCALE ALL DIMENTIONS IN mr	
DIMENSIONAL ±0.1 ANGULAR ±5° UNLESS OTHREWISE STATED.	STEEL		see above	DATE: 14/5/96 CHECKED: DATE:	TITLE: Gearbox Shafts	DRG NO. 3

NB: THE GEAR TEETH MUST BE ALIGNED PRIOR TO DRILLING

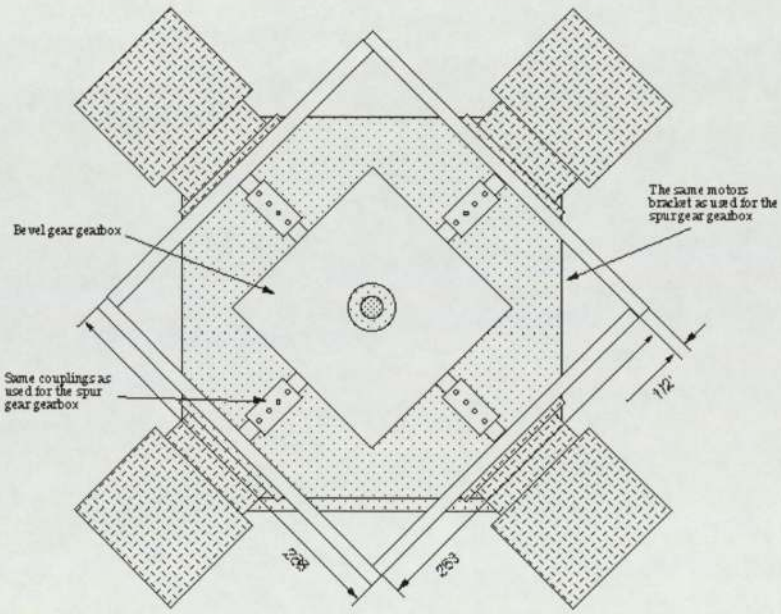


TOLERANCE	MATERIAL	PROJECTION	ORIGINAL SCALE	DRAWN: DW	DO NOT SCALE ALL DIMENTIONS IN mr	
DIMENSIONAL ±0.1 ANGULAR ±5° UNLESS OTHREWISE STATED.	TUFNOL		1 : 1	DATE: 15/5/96 CHECKED: DATE:	TITLE: Modifications to spur gears	DRG NO. 9

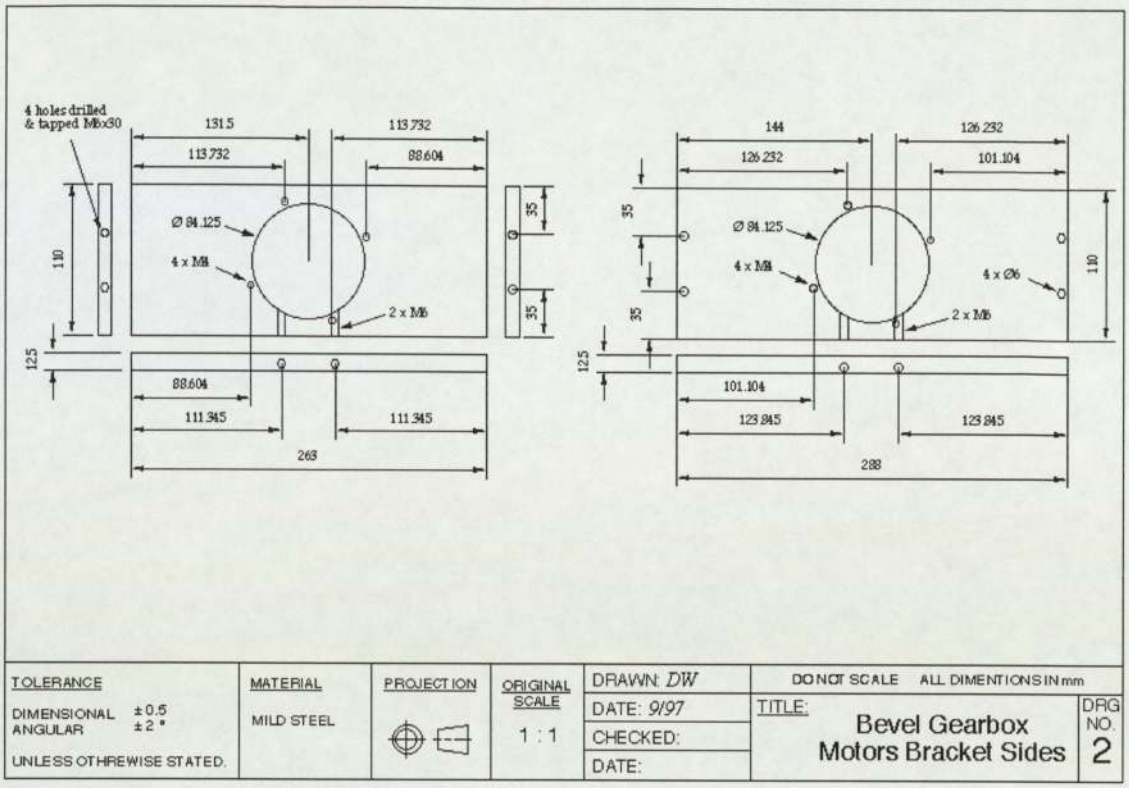


TOLERANCE	MATERIAL	PROJECTION	ORIGINAL SCALE	DRAWN:	DO NOT SCALE ALL DIMENTIONS IN mm	DRG NO.
DIMENSIONAL ANGULAR UNLESS OTHREWISE STATED.	steel		≈1:1	DATE: 26/3/97 CHECKED: DATE:	TITLE: Proposed Gear Modification.	

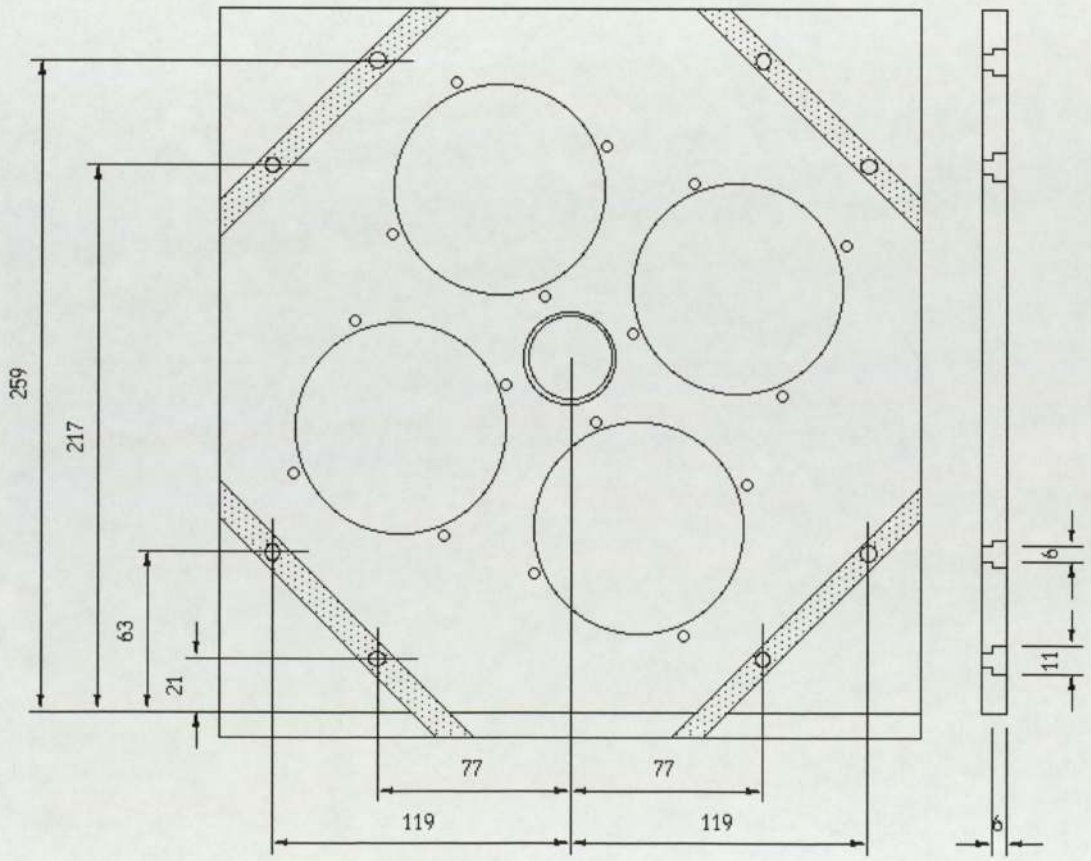
C.2.3 Motors Bracket for Bevel Gear Gearbox



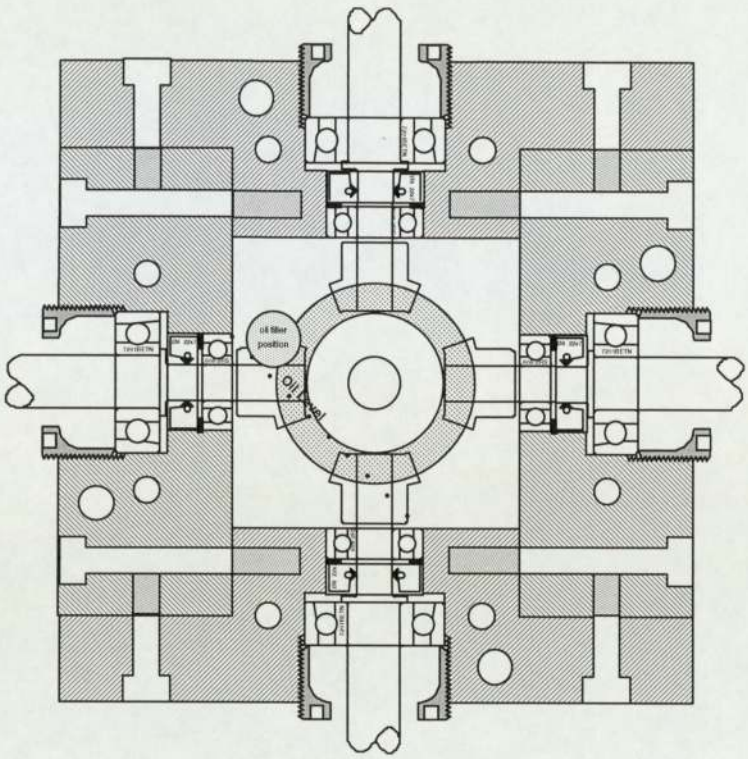
TOLERANCE	MATERIAL	PROJECTION	ORIGINAL SCALE	DRAWN: DW	DO NOT SCALE ALL DIMENTIONS IN mm	DRG NO.
DIMENSIONAL ±0.5 ANGULAR ±2° UNLESS OTHREWISE STATED.	MILD STEEL		1 : 1	DATE: 9/97 CHECKED: DATE:	TITLE: Motors, Bracket & Bevel Gear Box Layout	1

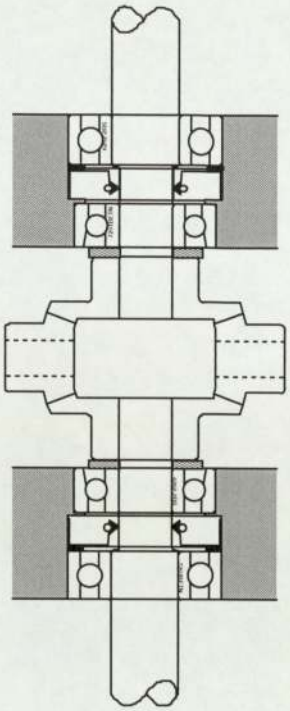


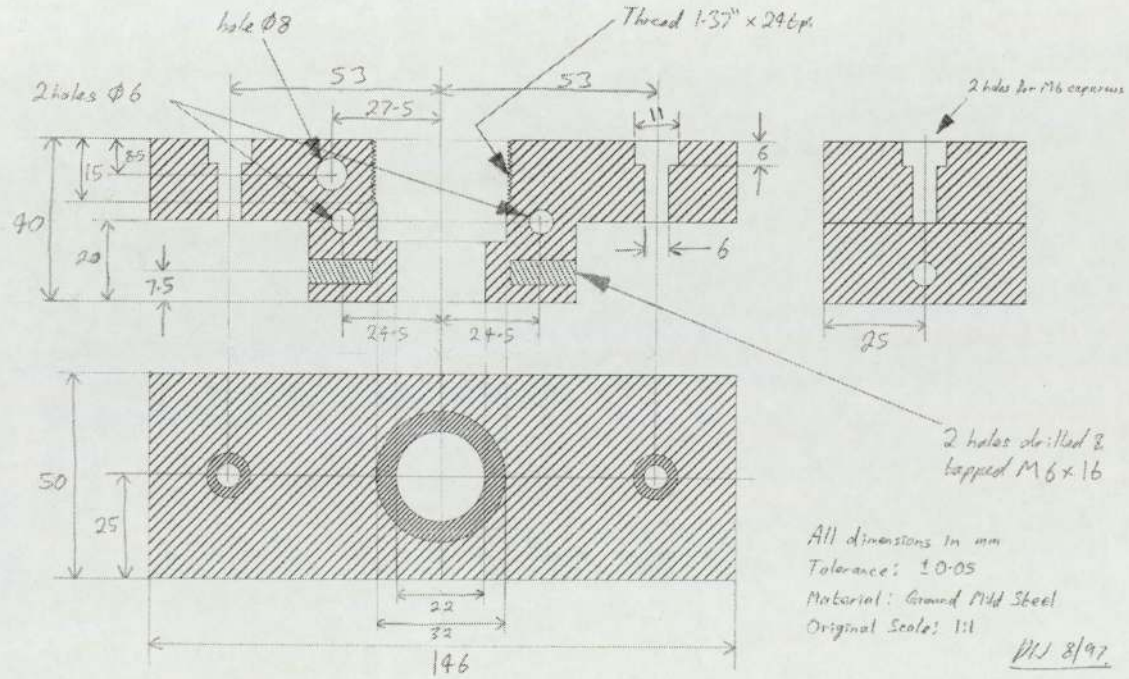
TOLERANCE	MATERIAL	PROJECTION	ORIGINAL SCALE	DRAWN: DW	DO NOT SCALE ALL DIMENTIONS IN mm	
DIMENSIONAL ±0.5 ANGULAR ±2° UNLESS OTHREWISE STATED.	MILD STEEL		1 : 1	DATE: 9/97 CHECKED: DATE:	TITLE: Bevel Gearbox Motors Bracket Sides	DRG NO. 2

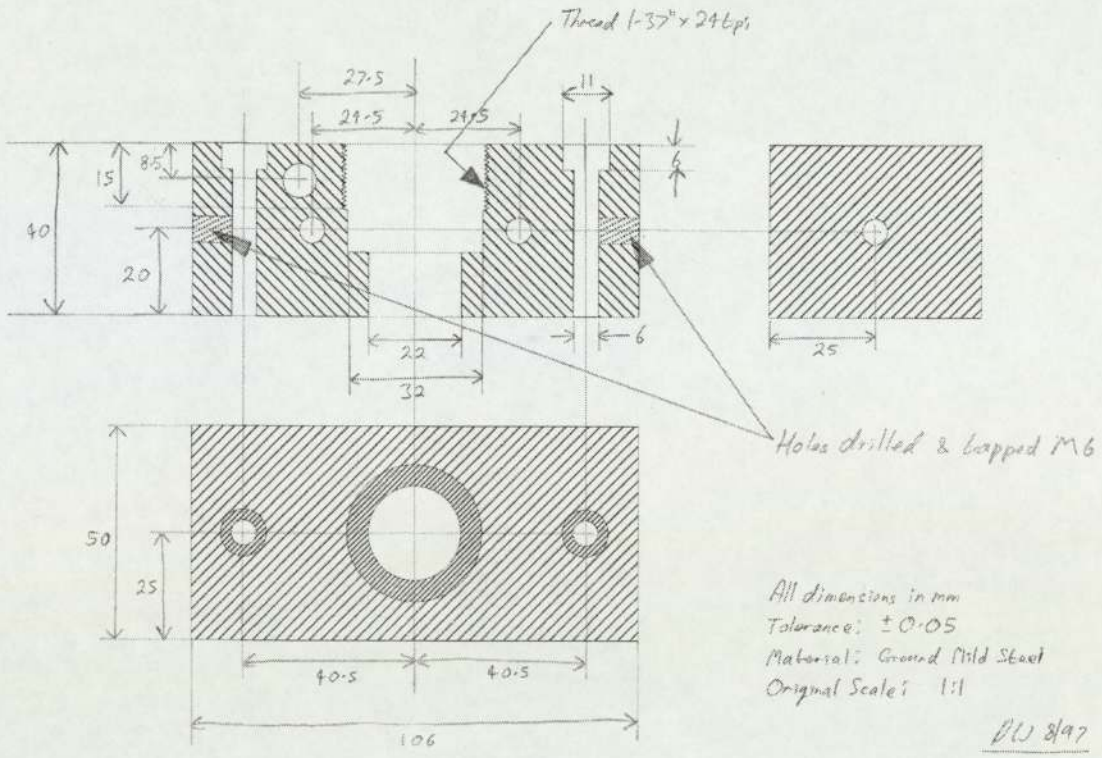


C.2.4 Bevel Gear Gearbox





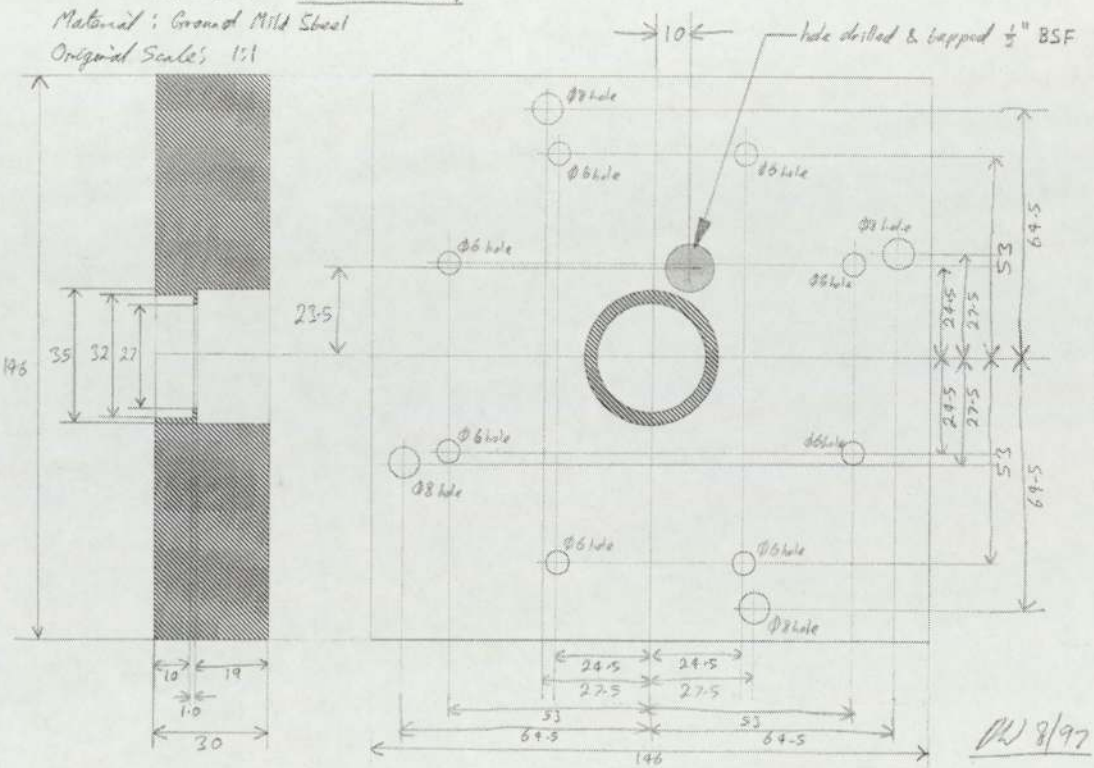




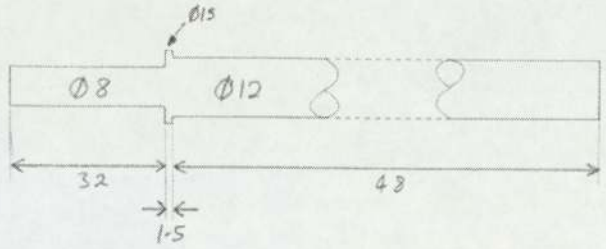
All dimensions in mm
Tolerance: ±0·05
Material: Ground Mild Steel
Original Scale: 1:1

DW 8/97

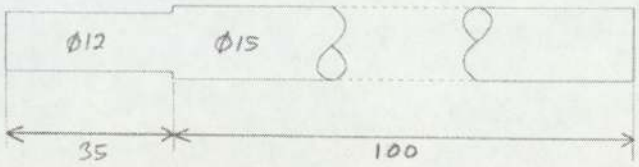
Tolerance: ±0·05 All dims in mm
Material: Ground Mild Steel
Original Scale: 1:1



DW 8/97



× 4 off



× 2 off

All dimensions in mm
Tolerance: ±0·05
Material: Silver Steel
Original Scale: 1:1 ATJ 8/97

C.2.5 Referred Inertia Calculations

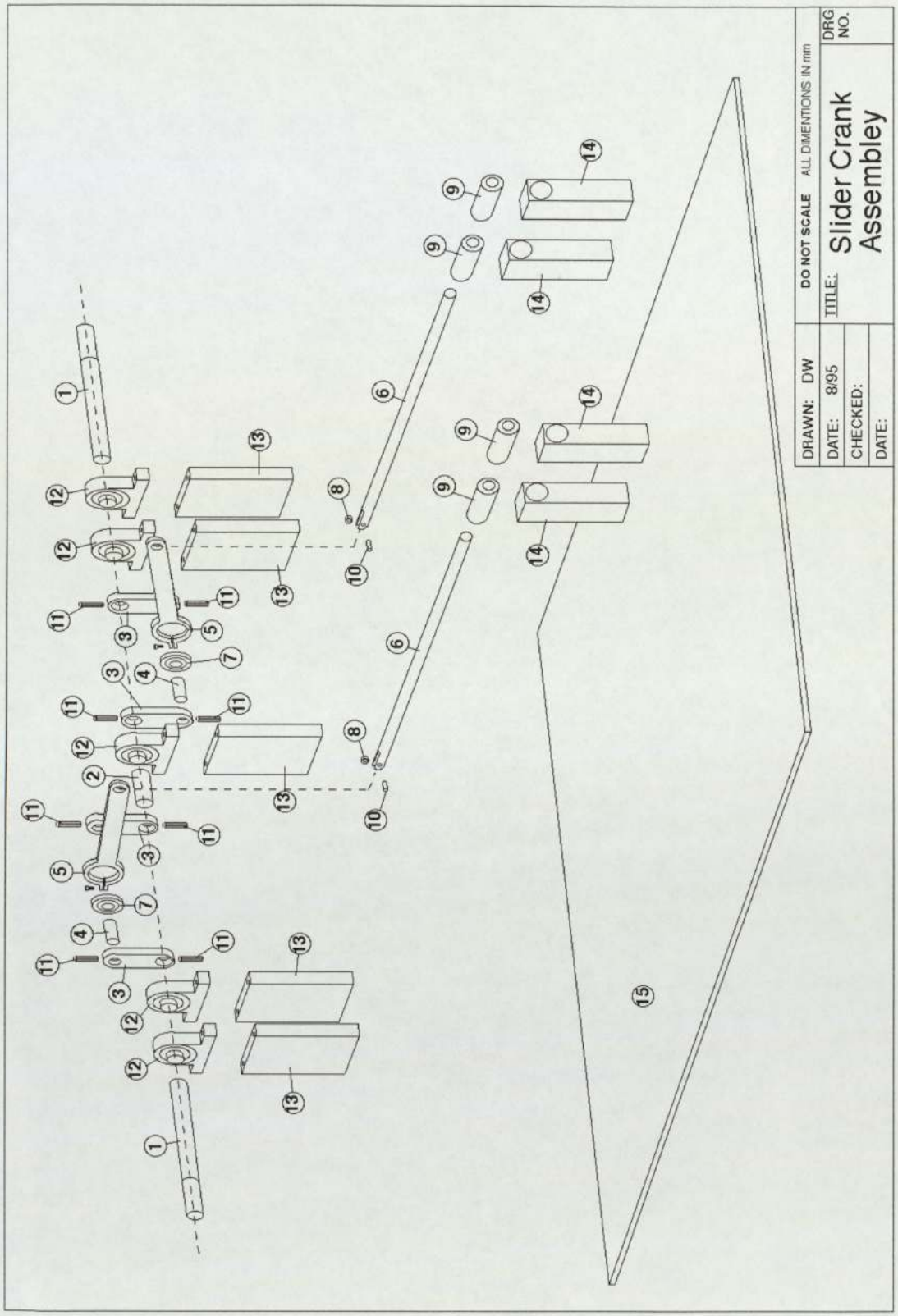
Gearbox:		Spur (Mkl)	Spur (Mkl)	Spur (Mkl-b)	Bevel (Mk II)
Material		Steel 1.0 MOD	Tufnol 1.0 MOD		Steel 1.0 MOD
density of gear material (kgm^-3)	D	7.70E+03	1.36E+03		7.70E+03
					approximatly:
planet gears - Ø of gear (pcd - m)	d	4.30E-02	4.30E-02		1.80E-02
planet gears - width of gear (m)	h	1.60E-02	1.60E-02		7.00E-03
planet gears - Ø of boss (m)	d	2.00E-02	2.00E-02		1.50E-02
planet gears - width of boss (m)	h	1.20E-02	1.20E-02		9.00E-03
volume of planet gear (m^3) =	V = p * (d/2)^2 * h	2.32E-05	2.32E-05		1.78E-06
volume of planet boss (m^3) =	V = p * (d/2)^2 * h	3.77E-06	3.77E-06		1.59E-06
mass of planet gear (kg) =	m = V * D	1.79E-01	3.16E-02		1.37E-02
mass of planet boss (kg) =	m = V * D	2.90E-02	5.13E-03		1.22E-02
inertia of planet gear (kgm^2) =	J = (m * d^2) / 8	4.14E-05	7.30E-06		5.56E-07
inertia of planet boss (kgm^2) =	J = (m * d^2) / 8	1.45E-06	2.56E-07		3.44E-07
total inertia of planet gear (kgm^2) =	J_planet = J_gear + J_boss	4.28E-05	7.56E-06	7.56E-06	9.00E-07
centre gear - Ø of gear (pcd - m)	d	1.03E-01	1.03E-01		4.50E-02
centre gear - width (m)	h	1.60E-02	1.60E-02		6.00E-03
centre gear - Ø of boss (m)	d	4.50E-02	4.50E-02		2.50E-02
centre gear - width of boss (m)	h	1.00E-02	1.00E-02		1.00E-02
volume of centre gear (m^3) =	V = p * (d/2)^2 * h	1.33E-04	1.33E-04		9.54E-06
volume of centre boss (m^3) =	V = p * (d/2)^2 * h	1.59E-05	1.59E-05		4.91E-06
mass of centre gear (kg) =	m = V * D	1.03E+00	1.81E-01		7.35E-02
mass of centre boss (kg) =	m = V * D	1.22E-01	2.16E-02		3.78E-02
inertia of centre gear (kgm^2) =	J = (m * d^2) / 8	1.36E-03	2.40E-04		1.86E-05
inertia of centre boss (kgm^2) =	J = (m * d^2) / 8	3.10E-05	5.48E-06		2.95E-06
total inertia of centre gear (kgm^2) =	J_center = J_gear + J_boss	1.39E-03	2.46E-04	1.39E-03	2.16E-05
gear ratio (n:1)		2.39E+00	2.39E+00	2.39E+00	2.39E+00
number of motors		4.00E+00	4.00E+00	4.00E+00	4.00E+00
motor inertia (kgm^2)		7.10E-06	7.10E-06	7.10E-06	7.10E-06
referred ineria of g/box per motor (kgm	J_planet + J_centre/(n^2*num_motors)	1.04E-04	1.83E-05	6.86E-05	1.85E-06
inertia of the torque transducer (TM212)		4.26E-04	4.26E-04	4.26E-04	4.26E-04
referred inertia of rig at I/P shaft (kgm^2	(J_motor+J_planet)*num_motors*n^2 +				
	J_centre + J_torque_trans	2.96E-03	1.01E-03	2.15E-03	6.30E-04

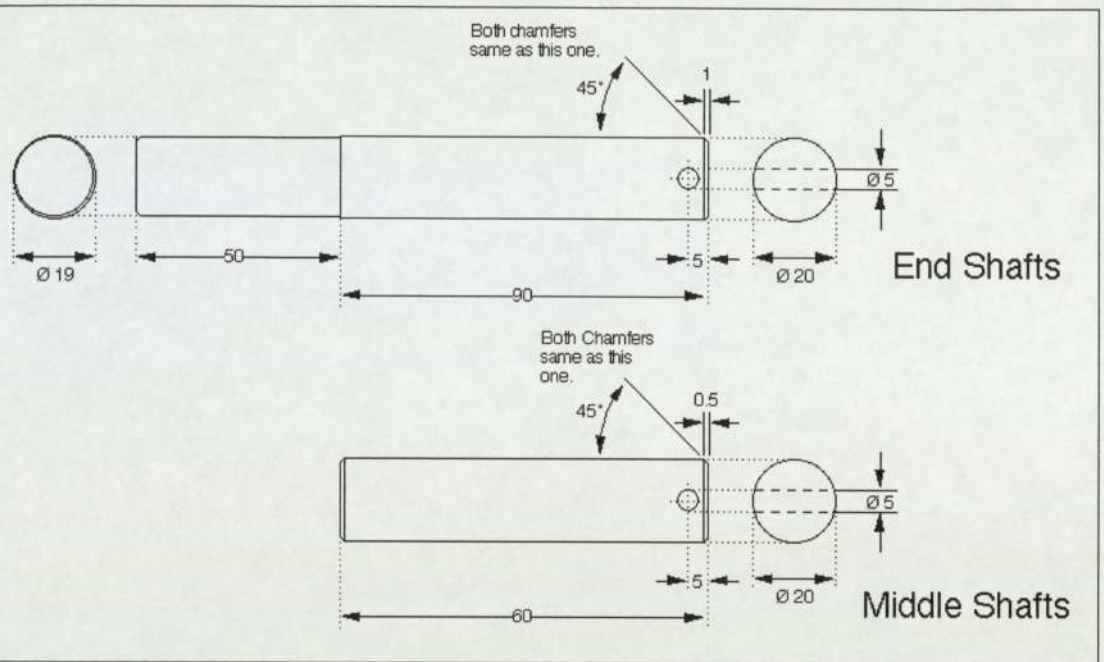
APPENDIX D

Appendix D Example Machine Design Drawings

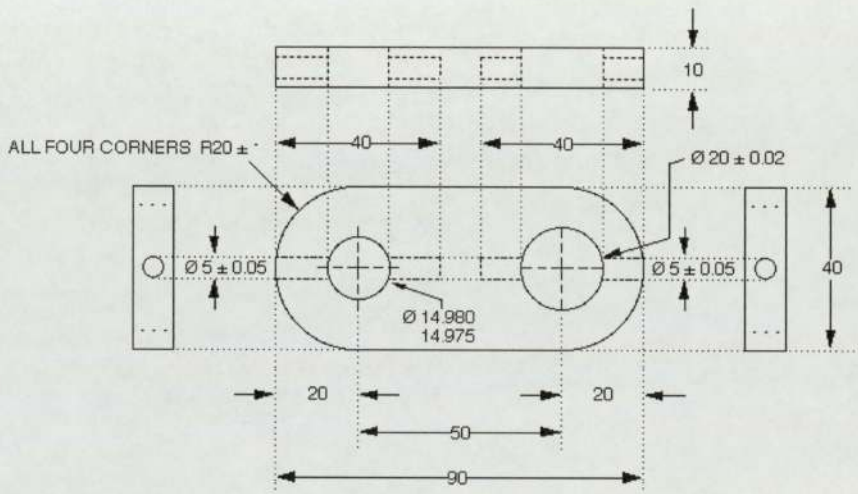
D.1 Slider Crank Mechanism

Part No.	Drwg No.	Part Description	Qty Reqd	Source
1	1	End Shaft	2	Aston
2	1	Middle Shaft	1	Aston
3	2	Crankshaft Arm	4	Aston
4	3	Crankshaft Journal	2	Aston
5	4	Connecting Rod	2	Aston
6	5	Slider	2	Aston
7	-	Big End Bearing	2	NACHI (P/N 6802 ZZ)
8	-	Little End Bearing	2	HPC (P/N QM 9-5)
9	-	Slider Bearing	4	HPC (P/N KB 12)
10	-	Gudgeon Pin	2	HPC (P/N DH 5-12 -Box of 10)
11	-	Spring Pin	8	HPC (P/N SAP 5-45 -Box of 10)
12	-	Main Bearing	5	NACHI (P/N UCP 204)
13	6	Main Bearing Support	5	Aston
14	6	Linear Bearing Support	4	Aston
15	7	Base Plate	1	Aston

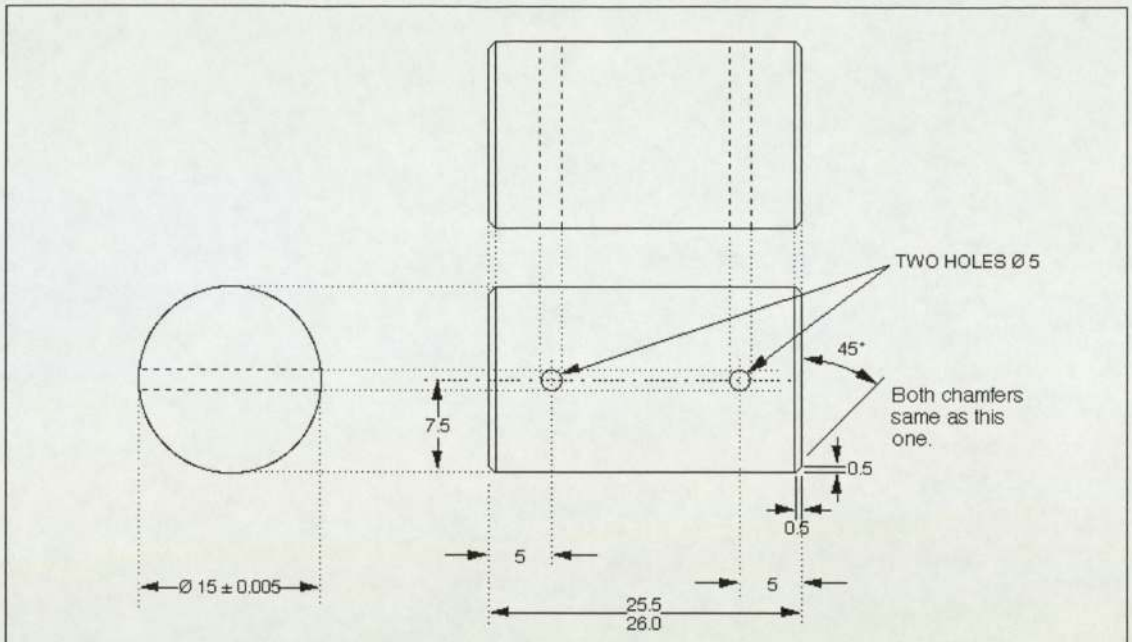

DRAWN: DW
DO NOT SCALE ALL DIMENTIONS IN mm
DRG NO.
DATE: 8/95
TITLE:
Slider Crank Assembley
CHECKED:
DATE:



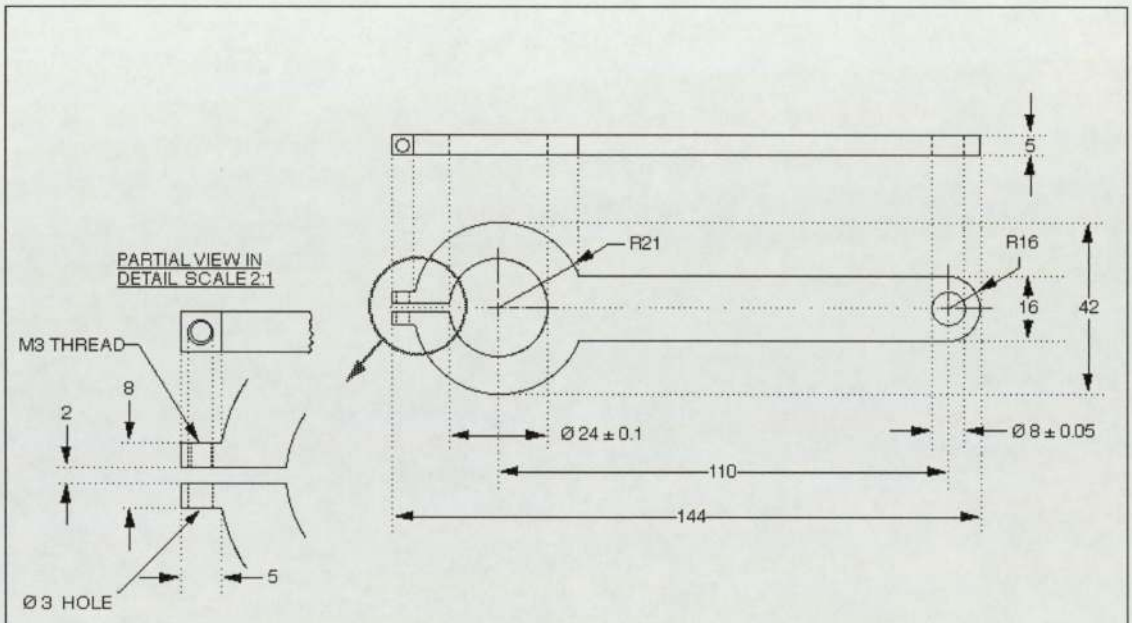
TOLERANCE	MATERIAL	PROJECTION	ORIGINAL SCALE	DRAWN: DW	DO NOT SCALE ALL DIMENTIONS IN mr	
DIMENSIONAL ±0.1 ANGULAR ±5° UNLESS OTHREWISE STATED.	SILVER STEEL		1 : 1	DATE: 2/8/95 CHECKED: DATE:	TITLE: Shafts	DRG NO. 1



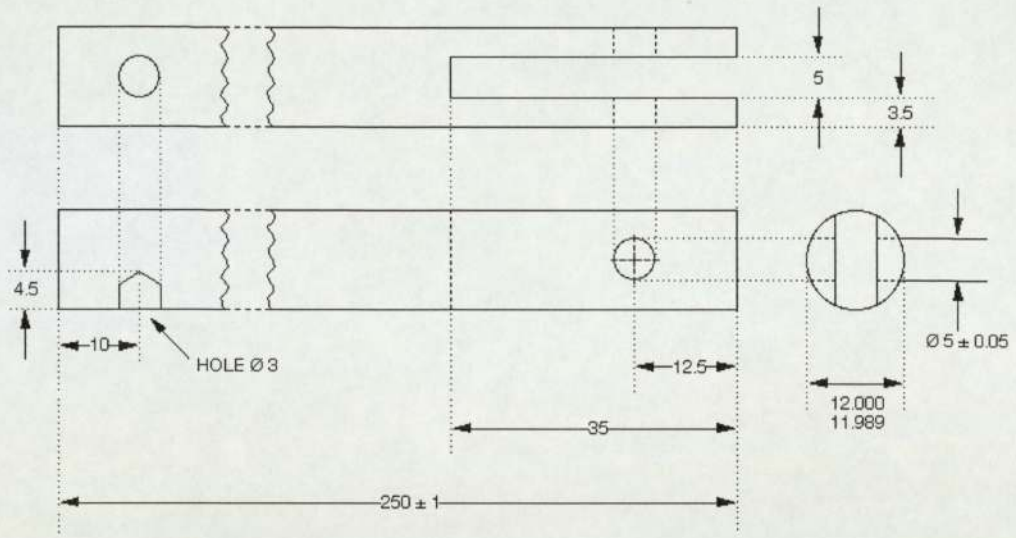
TOLERANCE	MATERIAL	PROJECTION	ORIGINAL SCALE	DRAWN: DW	DO NOT SCALE ALL DIMENTIONS IN mr	
DIMENSIONAL ±0.1 ANGULAR ±2° UNLESS OTHREWISE STATED.	MILD STEEL		1 : 1	DATE: 1/8/95 CHECKED: DATE:	TITLE: Crankshaft Arms	DRG NO. 2



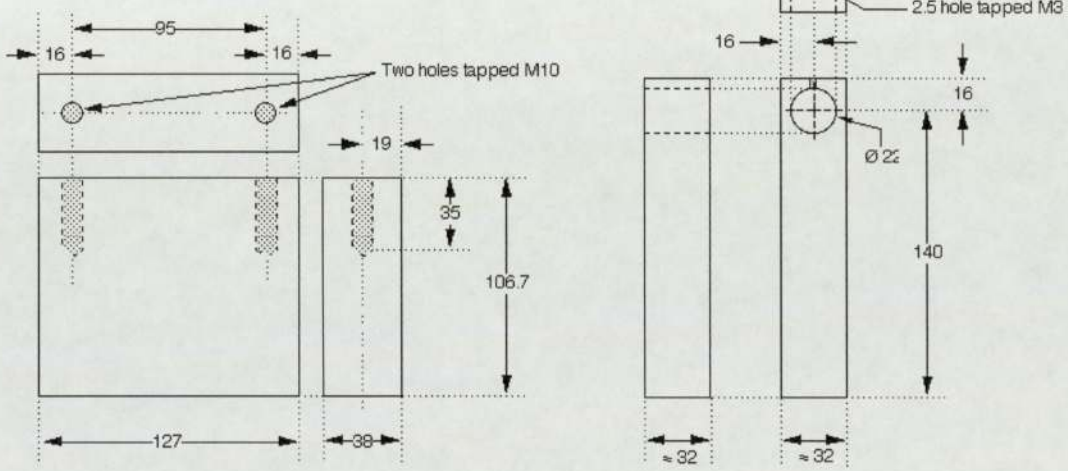
TOLERANCE	MATERIAL	PROJECTION	ORIGINAL SCALE	DRAWN: DW	DO NOT SCALE ALL DIMENTIONS IN mr	
DIMENSIONAL ±0.1 ANGULAR ±2° UNLESS OTHREWISE STATED.	SILVER STEEL		3 : 1	DATE: 3/8/95 CHECKED: DATE:	TITLE: Crankshaft Journals	DRG NO. 3



TOLERANCE	MATERIAL	PROJECTION	ORIGINAL SCALE	DRAWN: DW	DO NOT SCALE ALL DIMENTIONS IN mr	
DIMENSIONAL ±0.2 ANGULAR ±2° UNLESS OTHREWISE STATED.	MILD STEEL		1 : 1	DATE: 1/8/95 CHECKED: DATE:	TITLE: Connecting Rods	DRG NO. 4



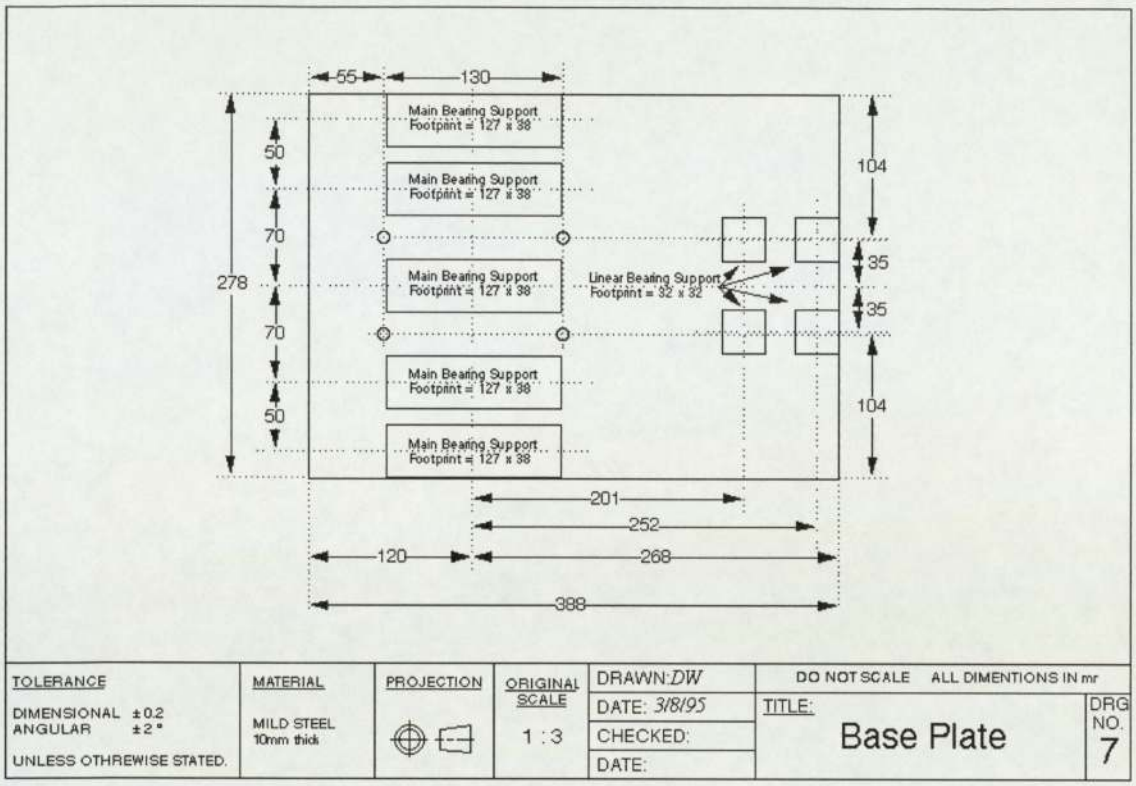
TOLERANCE	MATERIAL	PROJECTION	ORIGINAL SCALE	DRAWN: DW	DO NOT SCALE ALL DIMENTIONS IN mr	
DIMENSIONAL ±0.05 ANGULAR ±2° UNLESS OTHREWISE STATED.	STAINLESS 44(SHAFTING		2 : 1	DATE: 1/8/95 CHECKED: DATE:	TITLE: Slider	DRG NO. 5



Main Bearing Supports

Linear Bearing Supports

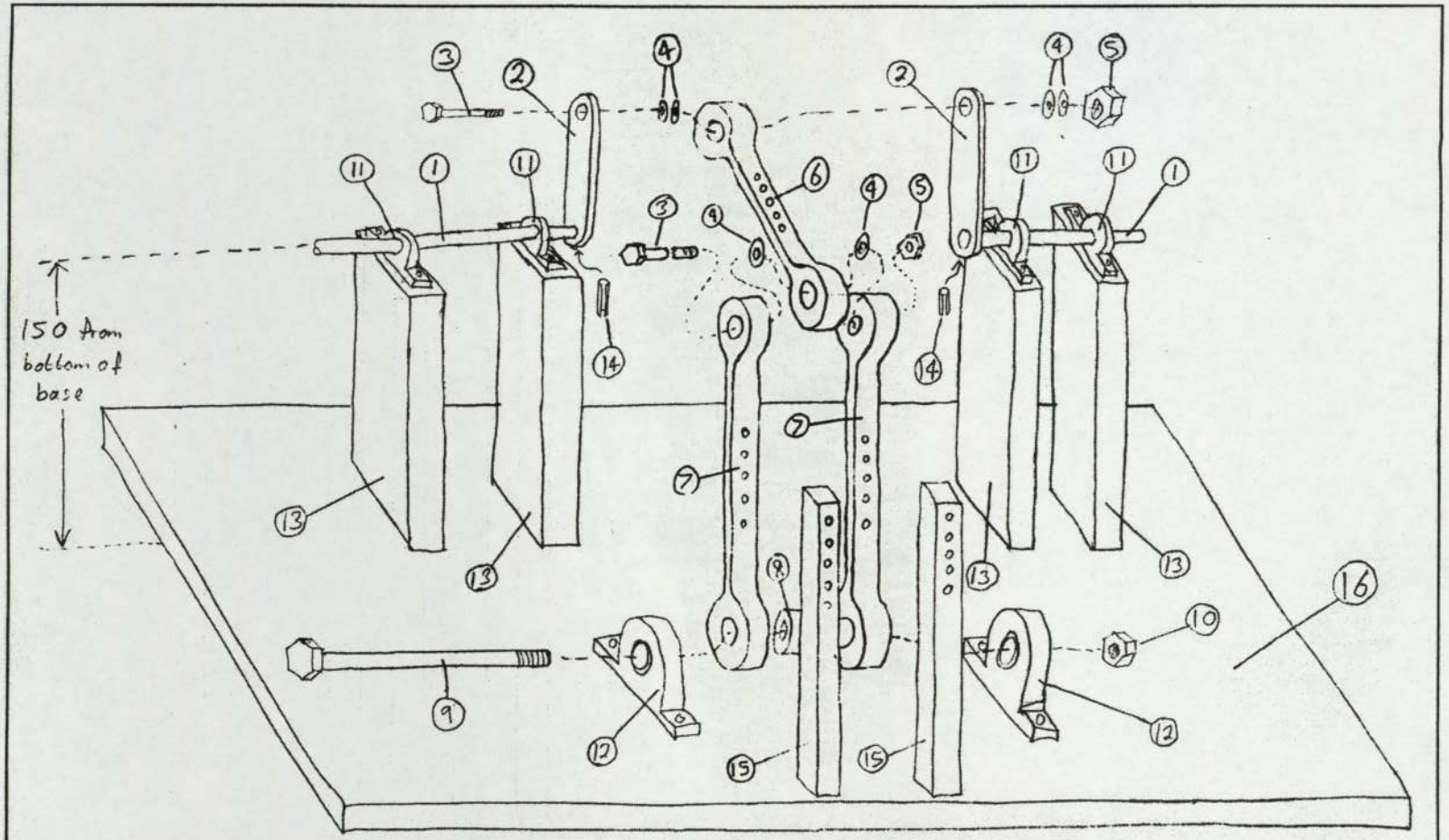
TOLERANCE	MATERIAL	PROJECTION	ORIGINAL SCALE	DRAWN: DW	DO NOT SCALE ALL DIMENTIONS IN mr	
DIMENSIONAL ±0.2 ANGULAR ±2° UNLESS OTHREWISE STATED.	MILD STEEL		1 : 1	DATE: 3/8/95 CHECKED: DATE:	TITLE: Bearing Supports	DRG NO. 6



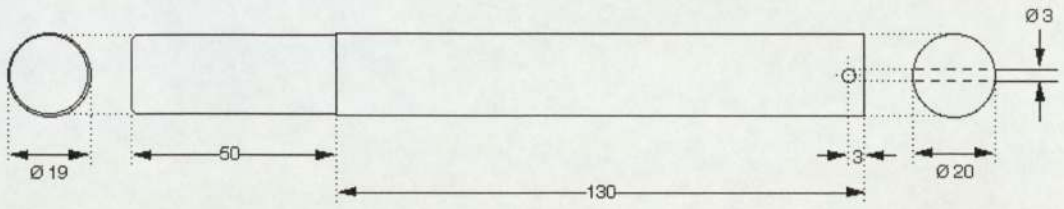
D.2 Four Bar Mechanism

Part No.	Drwg No.	Part Description	Qty Reqd	Source
1	1	Shaft	2	Aston
2	2	Rotating Arms	2	Aston
3	-	Bolt M5	2	Perriam
4	-	Washer M5	4	Perriam
5	-	Nyloc Nut M5	2	Perriam
6	6	Middle Link	1	Aston
7	7	Driven Bars	2	Aston
8	8	Spacer	1	Aston
9	-	Bolt M12	1	Perriam
10	-	Nyloc Nut M12	1	Perriam
11	-	Main Bearing	4	NACHI (P/N UCP 204)
12	-	Pivot Bearing	2	NACHI (P/N UCP 201)
13	13	Main Bearing Support	4	Aston
14	-	Spring Pin Ø2mm	2	Perriam

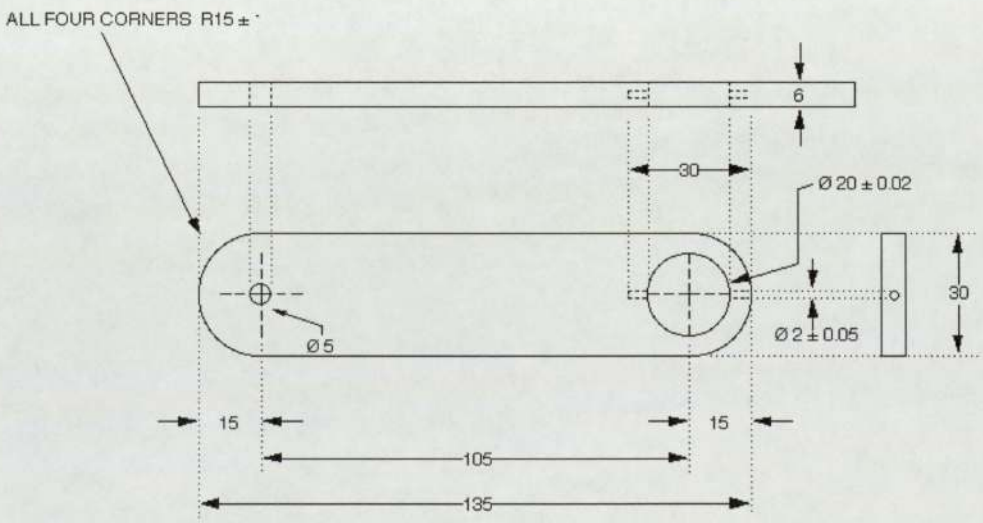
15	15	Spring Supports	1	Aston
16	16	Base Plate	1	Aston
		Machine Screws M? x ?	12	Perriam
		Machine Screws M10 x 40	8	Perriam
		Machine Screws M10 x 25	4	Perriam
		Cap Screws M8 x 30	14	Perriam
		Bearings for links	4	RS (540-334)



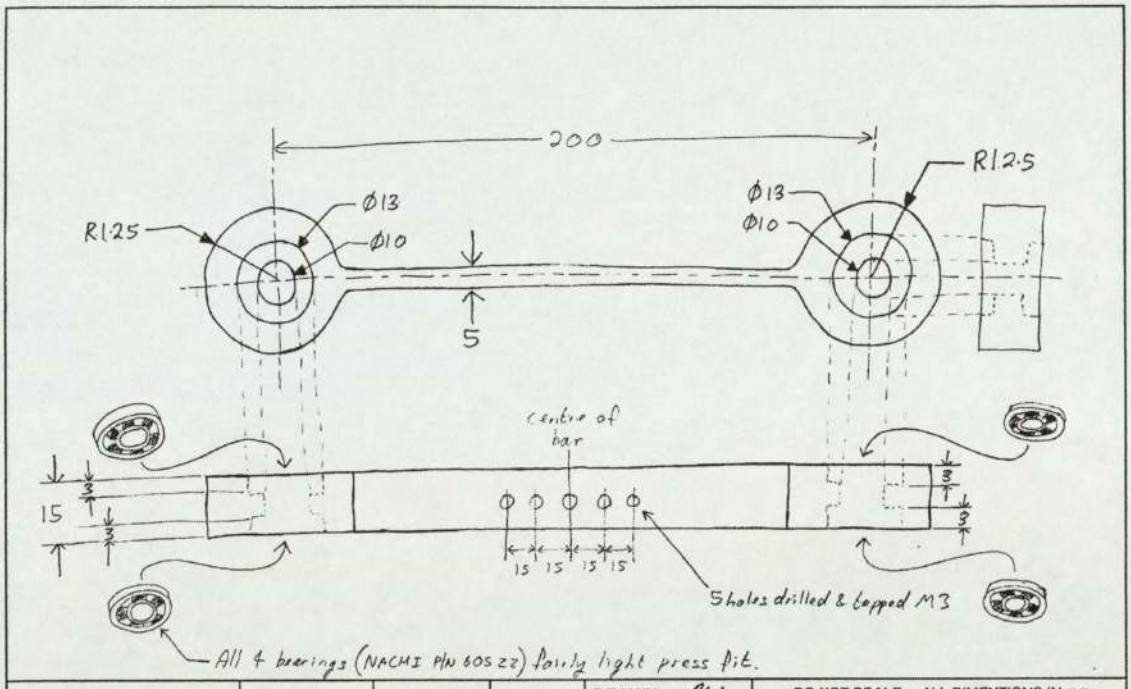
TOLERANCE DIMENSIONAL ±0.2 ANGULAR ±2° UNLESS OTHREWISE STATED.	MATERIAL Individually Specified	PROJECTION	ORIGINAL SCALE NOT TO SCALE	DRAWN: DW DATE: 29/8/96 CHECKED: DATE:	DO NOT SCALE ALL DIMENTIONS IN mm TITLE: Four Bar Mechanism Assembley Diagram DRG NO.



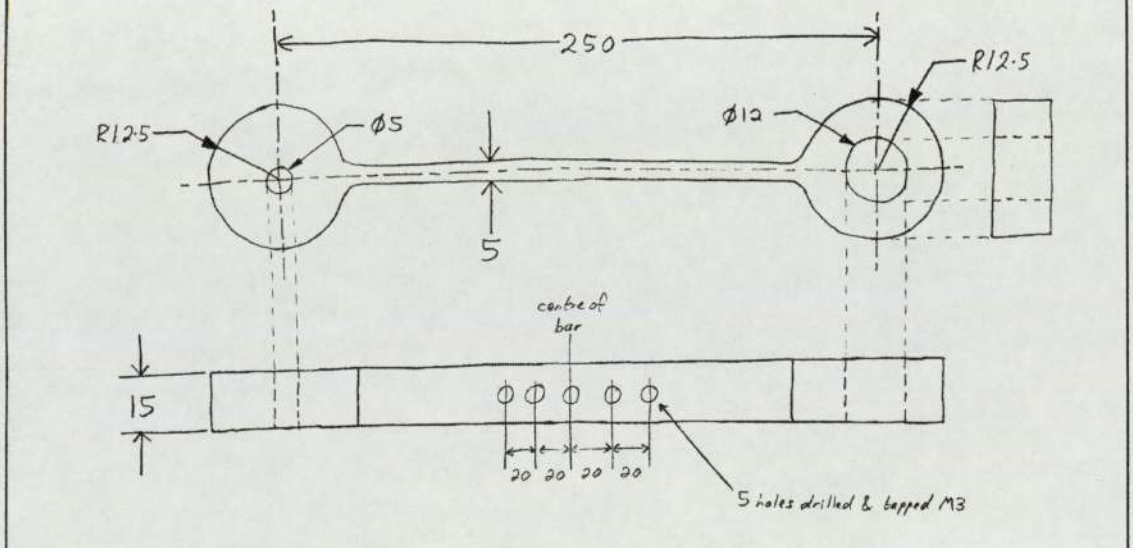
TOLERANCE	MATERIAL	PROJECTION	ORIGINAL SCALE	DRAWN: *DW*	DO NOT SCALE ALL DIMENTIONS IN mr	
DIMENSIONAL ±0.1 ANGULAR ±5° UNLESS OTHREWISE STATED.	SILVER STEEL		1 : 1	DATE: *3/9/96* CHECKED: DATE:	TITLE: Shafts	DRG NO. 1



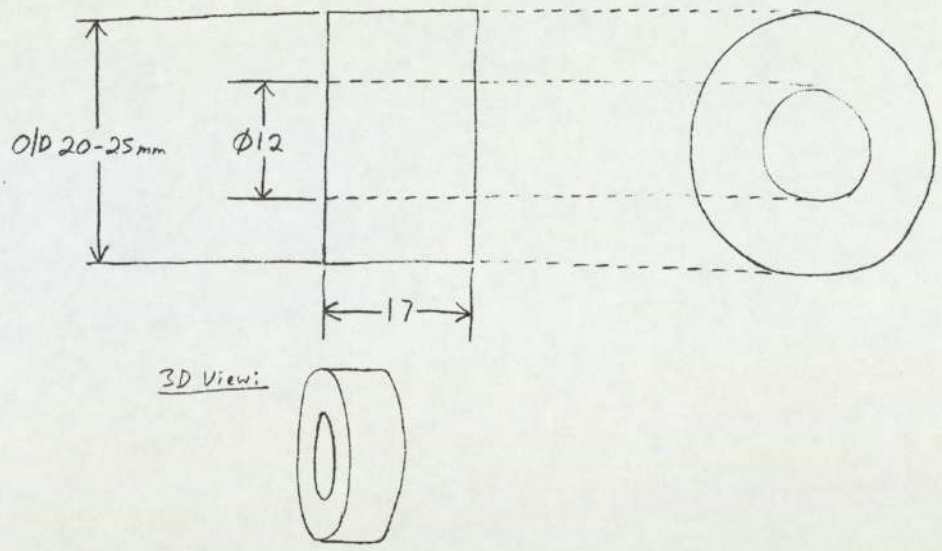
TOLERANCE	MATERIAL	PROJECTION	ORIGINAL SCALE	DRAWN: *DW*	DO NOT SCALE ALL DIMENTIONS IN mr	
DIMENSIONAL ±0.1 ANGULAR ±2° UNLESS OTHREWISE STATED.	MILD STEEL		1 : 1	DATE: *3/9/96* CHECKED: DATE:	TITLE: Rotating Arms	DRG NO. 2



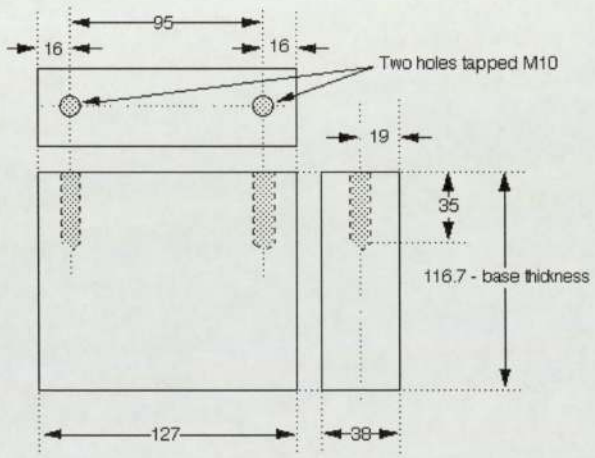
TOLERANCE	MATERIAL	PROJECTION	ORIGINAL SCALE	DRAWN:	DO NOT SCALE ALL DIMENTIONS IN mm	
DIMENSIONAL ±0.2 ANGULAR ±2° UNLESS OTHREWISE STATED.	Mild Steel		Not to Scale	DATE: 2/9/96 CHECKED: DATE:	TITLE: Middle Link	DRG NO. 6



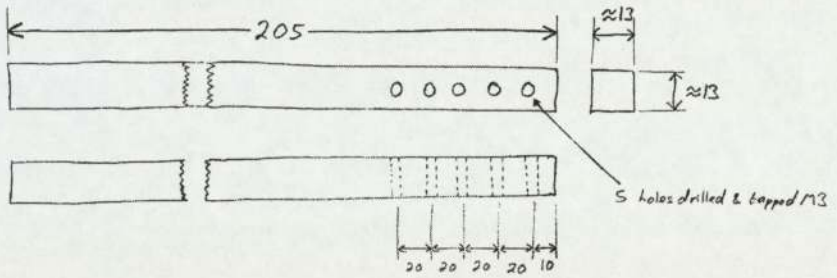
TOLERANCE	MATERIAL	PROJECTION	ORIGINAL SCALE	DRAWN:	DO NOT SCALE ALL DIMENTIONS IN mm	
DIMENSIONAL ±0.2 ANGULAR ±2° UNLESS OTHREWISE STATED.	Mild Steel		Not to Scale	DATE: 2/9/96 CHECKED: DATE:	TITLE: Driven Bars (2 off)	DRG NO. 7



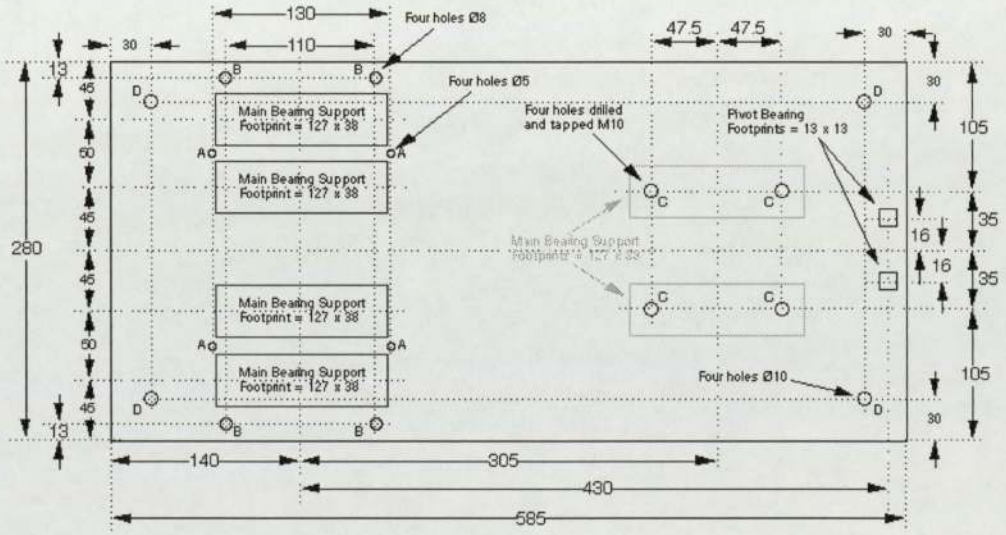
TOLERANCE	MATERIAL	PROJECTION	ORIGINAL SCALE	DRAWN:	DO NOT SCALE ALL DIMENTIONS IN mm	
DIMENSIONAL ±0.2 ANGULAR ±2° UNLESS OTHREWISE STATED.	Mild Steel		Not to Scale	DATE: 2/9/96 CHECKED: DATE:	TITLE: Spacer	DRG NO. 8



TOLERANCE	MATERIAL	PROJECTION	ORIGINAL SCALE	DRAWN: DW	DO NOT SCALE ALL DIMENTIONS IN mr	
DIMENSIONAL ±0.2 ANGULAR ±2° UNLESS OTHREWISE STATED.	MILD STEEL		1 : 1	DATE: 29/8/96 CHECKED: DATE:	TITLE: Main Bearing Support	DRG NO. 13



TOLERANCE	MATERIAL	PROJECTION	ORIGINAL SCALE	DRAWN: MJ	DO NOT SCALE ALL DIMENTIONS IN mm	
DIMENSIONAL ±0.2 ANGULAR ±2° UNLESS OTHREWISE STATED.	Mild Steel		Not to scale	DATE: 3/9/96 CHECKED: DATE:	TITLE: Spring Supports	DRG NO. 15

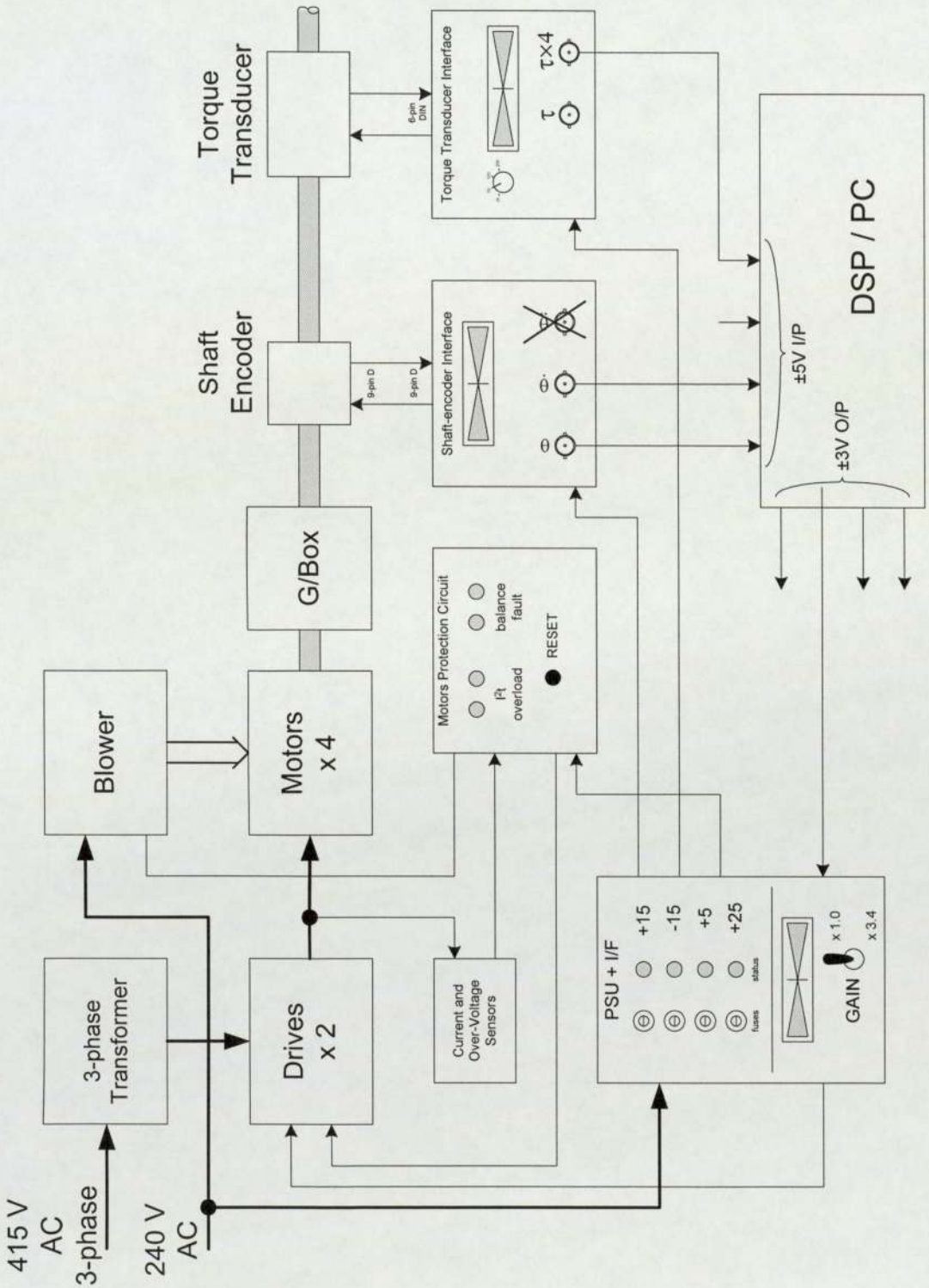


TOLERANCE	MATERIAL	PROJECTION	ORIGINAL SCALE	DRAWN: DW	DO NOT SCALE ALL DIMENTIONS IN mm	
DIMENSIONAL ±0.2 ANGULAR ±2° UNLESS OTHREWISE STATED.	MILD STEEL ≈12mm thick		1 : 3	DATE: 3/9/96 CHECKED: DATE:	TITLE: Base Plate	DRG NO. 16

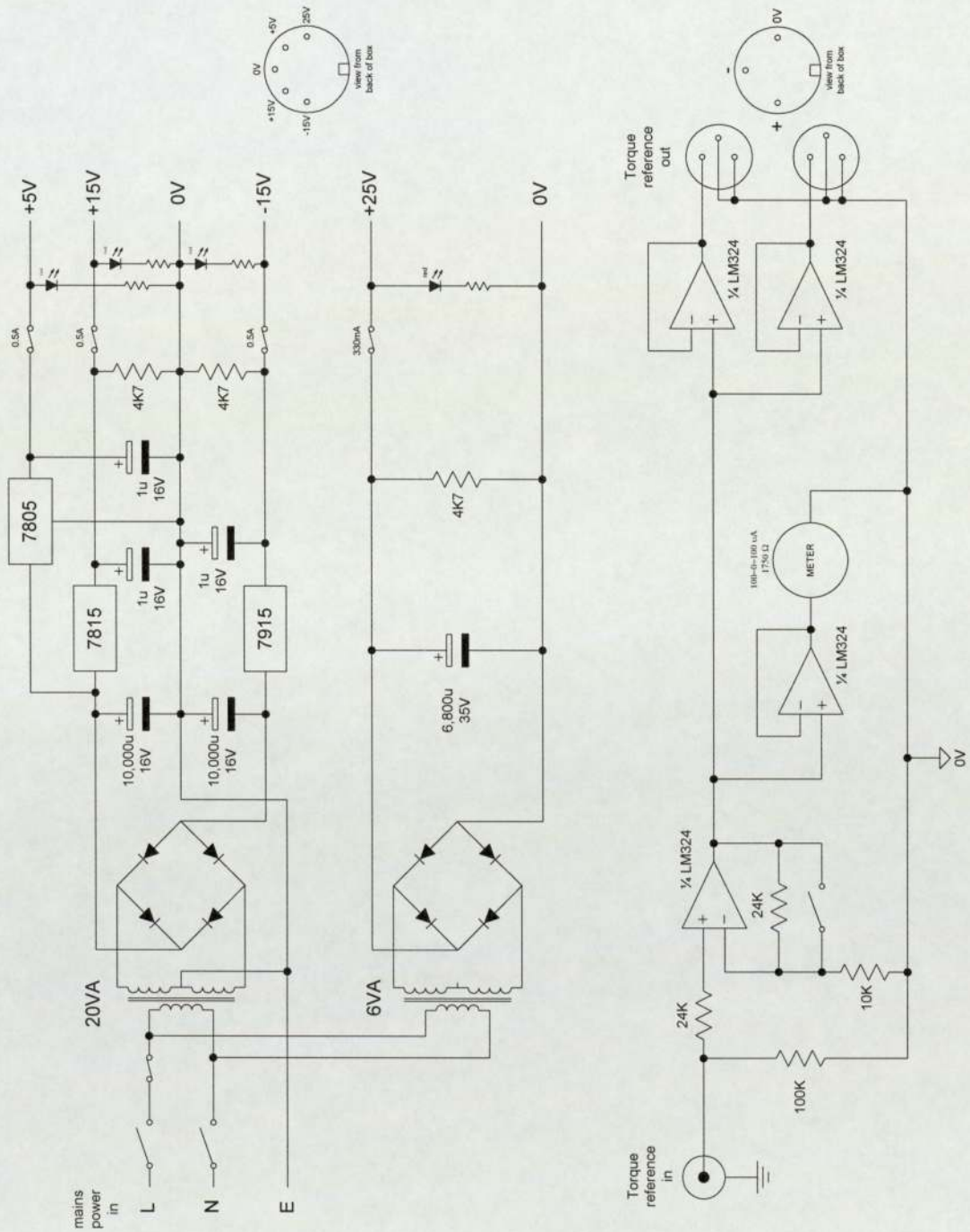
APPENDIX E

Appendix E Electrical / Electronic Interfacing Circuit Diagrams

This appendix contains the circuit diagrams referred to in chapter 7. Below is a diagram showing the overall connection of the individual electrical / electronic parts.

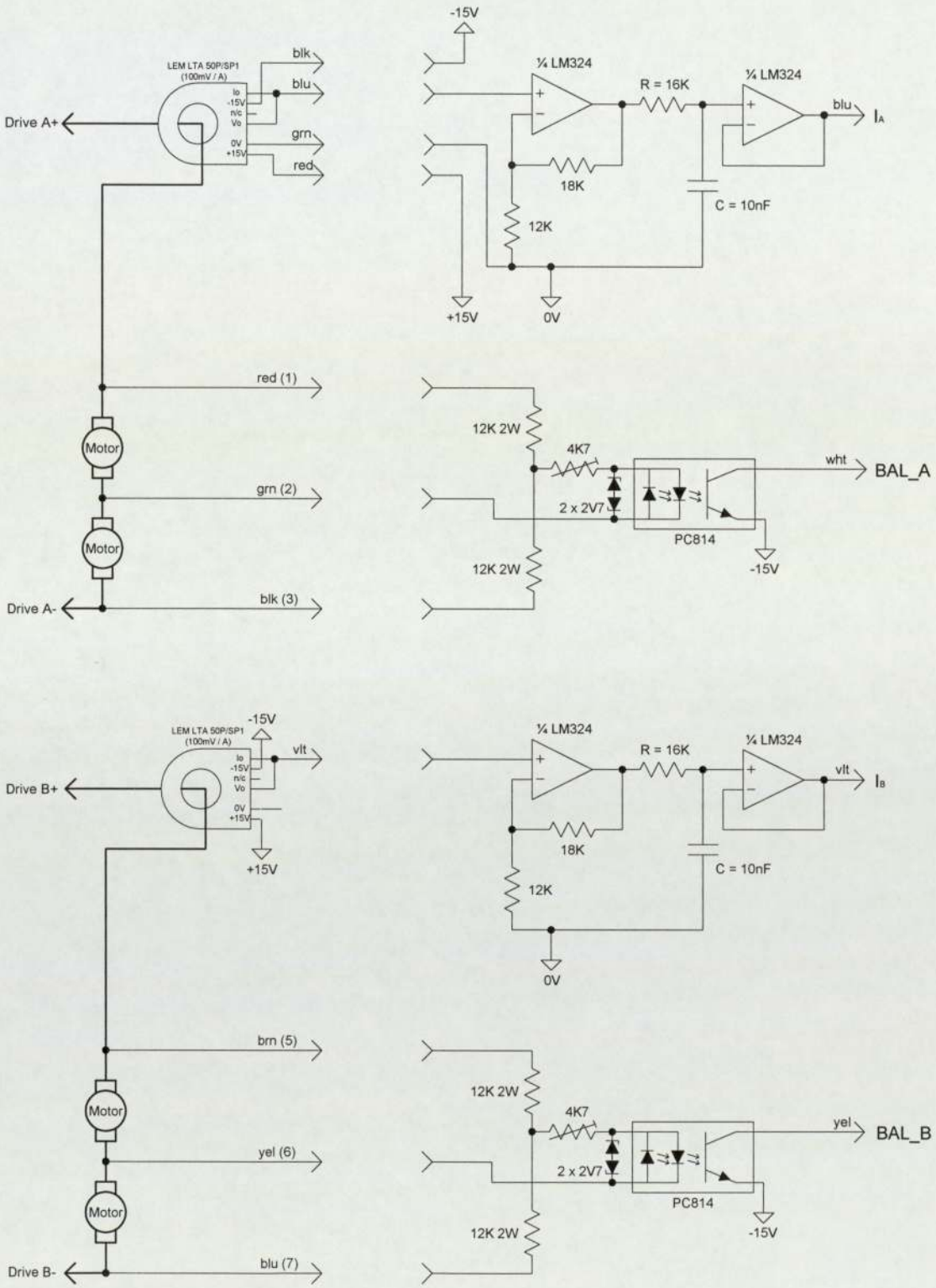


E.1 Electronic Interfacing Power Supply and Drives Interface

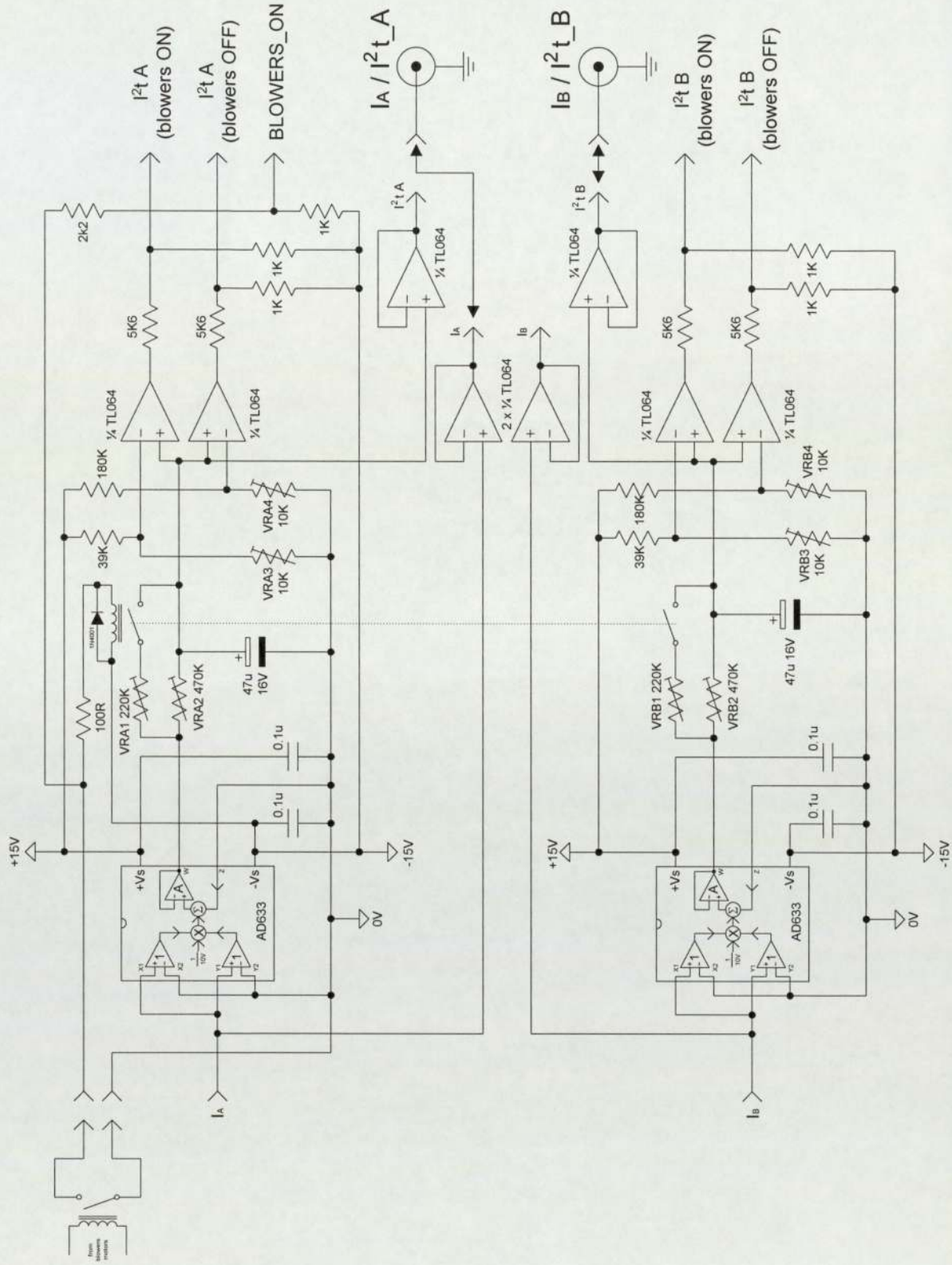


E.2 Motors Protection Circuit

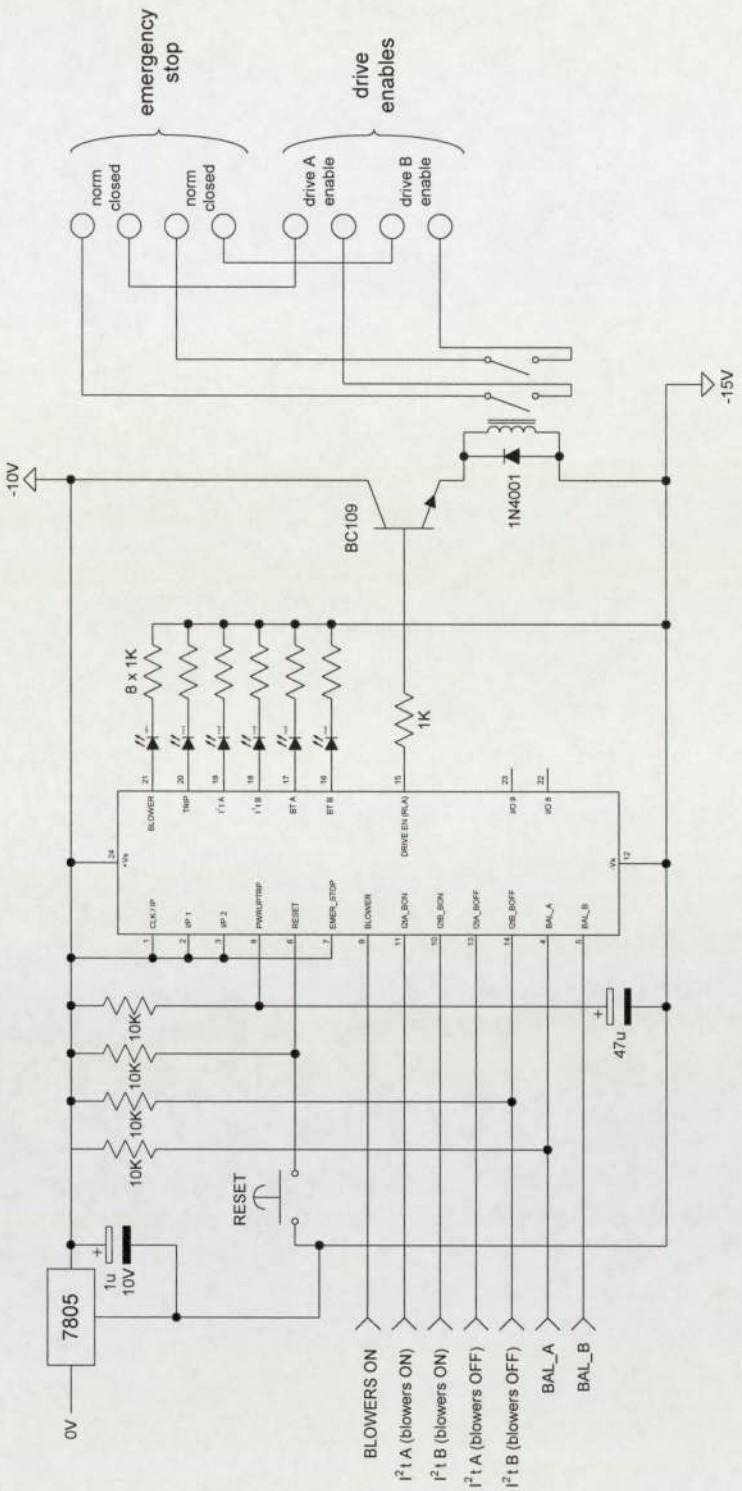
E.2.1 Current, Over-Voltage and Imbalance Detection



E.2.2 Motors Temperature Estimation



E.2.3 Logic Control

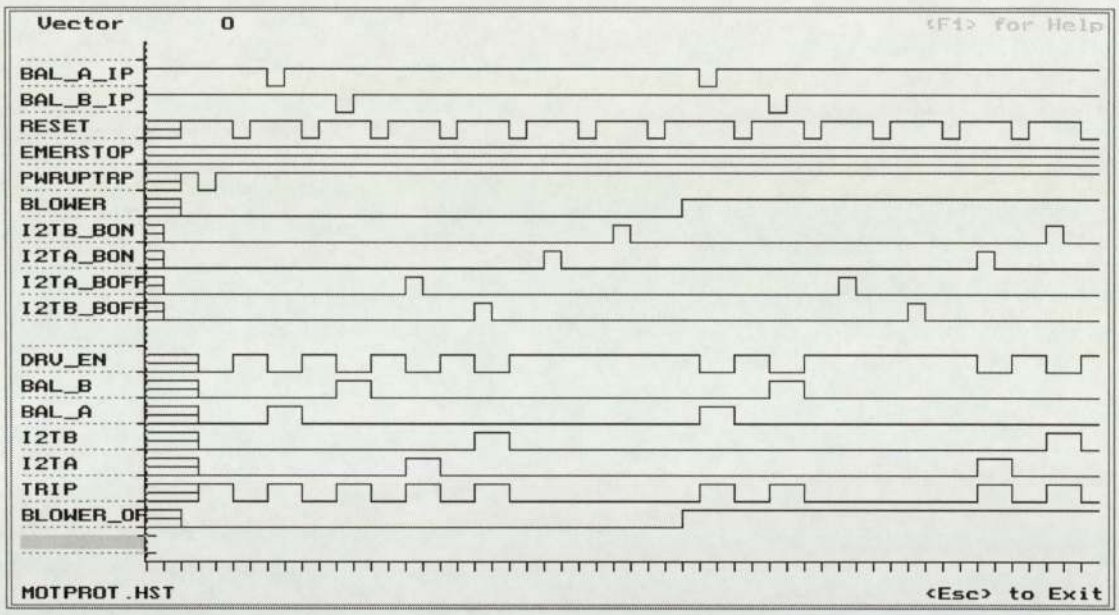


E.2.4 Control PLD code

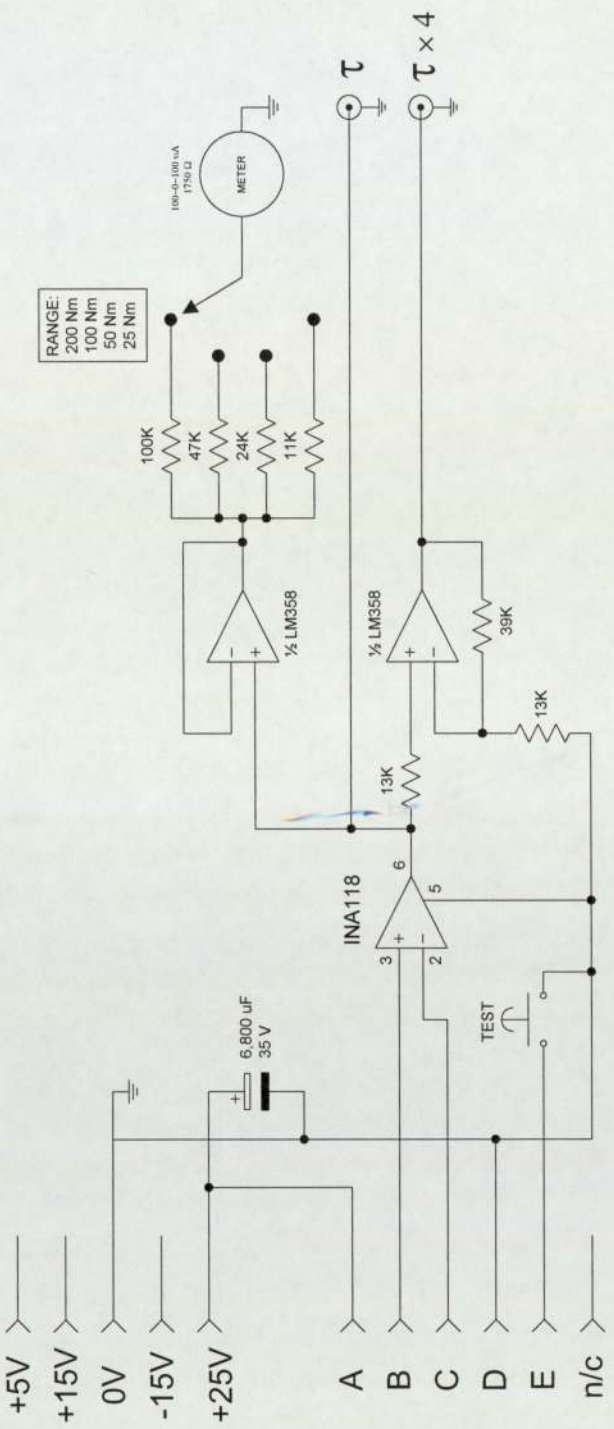
```
Title           Motors Protection Circuit Logic
Pattern         PDS
Revision        1.0
Author          David Wells
Company         Aston University
Date            30/4/96

OPTIONS         TURBO = OFF
                SECURITY = OFF

CHIP            MOTPROT    22V10

   ;Pin Declarations

   ;inputs
;    PIN     1       SPARE_IP_1    ;
;    PIN     2       SPARE_IP_2    ;
;    PIN     3       SPARE_IP_3    ;
    PIN     4       BAL_A_IP      ;Balance trip A
    PIN     5       BAL_B_IP      ;Balance trip B
    PIN     6       RESET         ;Reset switch I/P (0=RESET)
    PIN     7       EMERSTOP      ;Not used
    PIN     8       PWRUPTRP      ;Power-up trip (0=TRIP)
    PIN     9       BLOWER        ;Blower ON I/P (0=ON, 1=OFF)
    PIN     10      I2TB_BON      ;I^2*t trip A, blowers off
    PIN     11      I2TA_BON      ;I^2*t trip A, blowers off

   ;outputs
    PIN     13      I2TA_BOFF     ;I^2*t trip A, blowers on
    PIN     14      I2TB_BOFF     ;I^2*t trip A, blowers on
    PIN     15      DRV_EN        ;Drive enable relay
    PIN     16      BAL_B         ;O/P latch & led
    PIN     17      BAL_A         ;O/P latch & led
    PIN     18      I2TB          ;O/P latch & led
    PIN     19      I2TA          ;O/P latch & led
    PIN     20      TRIP          ;O/P latch & led
    PIN     21      BLOWER_OP     ;O/P latch & led
    PIN     22      SPARE_OP_1    ;
    PIN     23      SPARE_OP_2    ;

EQUATIONS

    BLOWER_OP = BLOWER

    I2TA = ((I2TA_BOFF * /BLOWER) + (I2TA_BON * BLOWER) + I2TA) *
             RESET * PWRUPTRP

    I2TB = ((I2TB_BOFF * /BLOWER) + (I2TB_BON * BLOWER) + I2TB) *
             RESET * PWRUPTRP

    BAL_A = (/BAL_A_IP + BAL_A) * RESET * PWRUPTRP

    BAL_B = (/BAL_B_IP + BAL_B) * RESET * PWRUPTRP

    TRIP = (/PWRUPTRP + I2TA + I2TB + BAL_A + BAL_B + TRIP) * RESET

    DRV_EN = /TRIP

    SPARE_OP_1 = GND
    SPARE_OP_2 = GND

SIMULATION

   SETF BAL_A_IP BAL_B_IP
   SETF /I2TA_BOFF /I2TB_BOFF /I2TA_BON /I2TB_BON
   SETF RESET /BLOWER PWRUPTRP

   CLOCKF PWRUPTRP

   CLOCKF RESET

   FOR n := 1 TO 2 DO
       BEGIN
           CLOCKF BAL_A_IP
           CLOCKF RESET

           CLOCKF BAL_B_IP
           CLOCKF RESET

           CLOCKF I2TA_BOFF
```

```
            CLOCKF RESET

            CLOCKF I2TB_BOFF
            CLOCKF RESET

            CLOCKF I2TA_BON
            CLOCKF RESET

            CLOCKF I2TB_BON
            CLOCKF RESET

            SETF BLOWER
         END

;end simulation
```

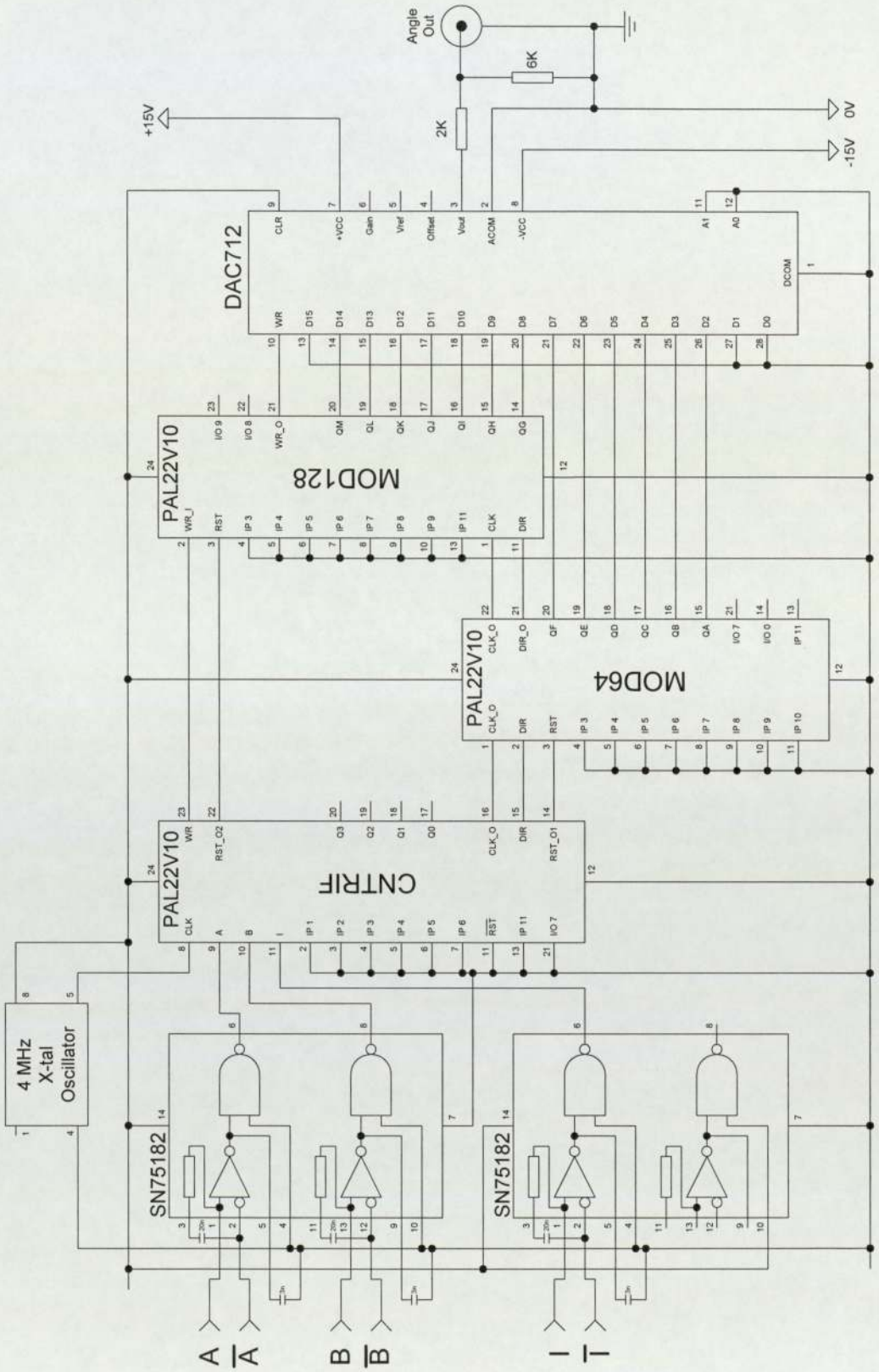
E.2.4.1 Control PLD code simulation



E.3 Torque Transducer Interface



E.4 Shaft Encoder Interface



E.4.1 Shaft Encoder Interface PLD Code – "CNTRIF"

```
Title           Encoder_Interface
Pattern         PDS
Revision        6.0
Author          David Wells
Company         Aston University
Date            1/10/96

;
;This design takes A & B pulses from an 'n' line per revolution encoder and
;outputs /UP and /DOWN signals to a counter to count '4n' lines per revolution.
;Also includes timing signal generation for the D/A convertor and digital
;filtering of the input signals - see lab book.
;

CHIP            ENCDR_IF   22V10


;Pin Declarations

    ;inputs
    PIN     1       CLK     ;clock for state machine and mealy variable (DIR)
    PIN     8       A       ;from encoder
    PIN     9       B       ;from encoder
    PIN     10      I       ;from encoder (not used in this design)
    PIN     11      RST     ;power-up reset I/P

    ;outputs
    PIN     23      WR          ;write O/P to D/A
    PIN     22      RST_O2      ;RST O/P to 2nd counter
    PIN     20      Q3          ;state variable
    PIN     19      Q2          ;state variable
    PIN     18      Q1          ;state variable
    PIN     17      Q0          ;state variable
    PIN     16      CNTR_CLK    ;mealy machine O/P, counter clock
    PIN     15      DIR         ;mealy machine O/P, direction of count (0=down, 1=up)
    PIN     14      RST_O1      ;RST O/P to 1st counter


STATE MOORE_MACHINE

DEFAULT_BRANCH      HOLD_STATE

;state assignments

   S0   = /Q3 * /Q2 * /Q1 * /Q0  ;0000  - powerup state
   S1   = /Q3 * /Q2 * /Q1 *  Q0  ;0001
   S2   = /Q3 * /Q2 *  Q1 * /Q0  ;0010
   S3   = /Q3 * /Q2 *  Q1 *  Q0  ;0011
   S4   = /Q3 *  Q2 * /Q1 * /Q0  ;0100
   S5   = /Q3 *  Q2 * /Q1 *  Q0  ;0101
   S6   = /Q3 *  Q2 *  Q1 * /Q0  ;0110
   S7   = /Q3 *  Q2 *  Q1 *  Q0  ;0111
   S8   =  Q3 * /Q2 * /Q1 * /Q0  ;1000
   S9   =  Q3 * /Q2 * /Q1 *  Q0  ;1001
   S10  =  Q3 * /Q2 *  Q1 * /Q0  ;1010
   S11  =  Q3 * /Q2 *  Q1 *  Q0  ;1011
   S12  =  Q3 *  Q2 * /Q1 * /Q0  ;1100
   S13  =  Q3 *  Q2 * /Q1 *  Q0  ;1101
   S14  =  Q3 *  Q2 *  Q1 * /Q0  ;1110
   S15  =  Q3 *  Q2 *  Q1 *  Q0  ;1111


;state transitions

   S0  := (/A * /B)  -> S0
        + (/A *  B)  -> S1
        + ( A * /B)  -> S8
        + ( A *  B)  -> S1

   S1  := VCC        -> S2

   S2  := (/A * /B)  -> S14
        + (/A *  B)  -> S2
        + ( A * /B)  -> S3
        + ( A *  B)  -> S3

   S3  := VCC        -> S4

   S4  := (/A * /B)  -> S5
        + (/A *  B)  -> S12
```

```
        + ( A * /B)   -> S5
        + ( A *  B)   -> S4

   S5  := VCC         -> S6

   S6  := (/A * /B)   -> S7
        + (/A *  B)   -> S7
        + ( A * /B)   -> S6
        + ( A *  B)   -> S10

   S7  := VCC         -> S0

   S8  := (/A * /B)   -> S8
        + (/A *  B)   -> S0
        + ( A * /B)   -> S9
        + ( A *  B)   -> S9

   S9  := VCC         -> S10

   S10 := (/A * /B)   -> S6
        + (/A *  B)   -> S11
        + ( A * /B)   -> S10
        + ( A *  B)   -> S11

   S11 := VCC         -> S12

   S12 := (/A * /B)   -> S13
        + (/A *  B)   -> S13
        + ( A * /B)   -> S4
        + ( A *  B)   -> S12

   S13 := VCC         -> S14

   S14 := (/A * /B)   -> S15
        + (/A *  B)   -> S14
        + ( A * /B)   -> S15
        + ( A *  B)   -> S2

   S15 := VCC         -> S8

EQUATIONS

    Q0.CLKF = CLK   ;state machine clock
    Q1.CLKF = CLK   ;
    Q2.CLKF = CLK   ;
    Q3.CLKF = CLK   ;

    Q0.RSTF = RST   ;resets
    Q1.RSTF = RST   ;
    Q2.RSTF = RST   ;
    Q3.RSTF = RST   ;

    WR.D := /CNTR_CLK     ;/WR signal same as CNTRCLK, but delayed
    WR.CLKF = CLK
    WR.RSTF = RST

    DIR = Q3
    CNTR_CLK = Q0

    RST_O1 = RST
    RST_O2 = RST


SIMULATION

   VECTOR status := [Q3, Q2, Q1, Q0]
   SETF /A /B /CLK /RST
   PRLDF /Q3 /Q2 /Q1 /Q0

CLOCKF RST

   FOR j:=0 TO 3 DO
       BEGIN
           CLOCKF CLK
           CLOCKF CLK
           CLOCKF CLK
           CLOCKF CLK
           CLOCKF CLK
           CLOCKF CLK
           ;toggle A
           IF (A = VCC) THEN
```

```
            BEGIN
                SETF /A
            END
        ELSE
            BEGIN
                SETF  A
            END
        CLOCKF CLK
        CLOCKF CLK
        CLOCKF CLK
        CLOCKF CLK
        CLOCKF CLK
        CLOCKF CLK
        ;toggle B
        IF (B = VCC) THEN
            BEGIN
                SETF /B
            END
        ELSE
            BEGIN
                SETF  B
            END
    END


FOR j:=0 TO 3 DO
    BEGIN
        CLOCKF CLK
        CLOCKF CLK
        CLOCKF CLK
        CLOCKF CLK
        CLOCKF CLK
        CLOCKF CLK
        ;toggle B
        IF (B = VCC) THEN
            BEGIN
                SETF /B
            END
        ELSE
            BEGIN
                SETF  B
            END
        CLOCKF CLK
        CLOCKF CLK
        CLOCKF CLK
        CLOCKF CLK
        CLOCKF CLK
        CLOCKF CLK
        ;toggle A
        IF (A = VCC) THEN
            BEGIN
                SETF /A
            END
        ELSE
            BEGIN
                SETF  A
            END
    END

;end simulation
```

E.4.2 Shaft Encoder Interface PLD Code – "MOD64" and "MOD128"

The file below is "MOD64". This is a modulo-64 binary up-down counter written to compile for a 22V10 programmable logic device. The state assignments and state transitions were created by C-code which is quicker than typing it by hand. "MOD128" is the same type of counter, the only difference being the number of states (128) and state variables (7). Because of its large size and similarity to "MOD64", it has not been included in this text.

```
Title           modulo 64 up-down counter with wrap-around
Pattern         PDS
Revision        1.0
Author          David Wells
Company         Aston University
Date            1/10/96

CHIP            MOD_64    22V10

   ;Pin Declarations

   ;inputs
    PIN     1       CLK     ;clock input
    PIN     2       DIR     ;count UP or /DOWN
    PIN     3       RST     ;clear input

   ;outputs
    PIN     23      CLK_O   ;borrow output
    PIN     22      DIR_O   ;carry output
    PIN     20      QF      ;counter O/P msb
    PIN     19      QE      ;counter O/P
    PIN     18      QD      ;counter O/P
    PIN     17      QC      ;counter O/P
    PIN     16      QB      ;counter O/P
    PIN     15      QA      ;counter O/P

STATE MOORE_MACHINE

DEFAULT_BRANCH      HOLD_STATE

;state assignments

    S0   = /QF * /QE * /QD * /QC * /QB * /QA  ;000000
    S1   = /QF * /QE * /QD * /QC * /QB *  QA  ;000001
    S2   = /QF * /QE * /QD * /QC *  QB * /QA  ;000010
    S3   = /QF * /QE * /QD * /QC *  QB *  QA  ;000011
    S4   = /QF * /QE * /QD *  QC * /QB * /QA  ;000100
    S5   = /QF * /QE * /QD *  QC * /QB *  QA  ;000101
    S6   = /QF * /QE * /QD *  QC *  QB * /QA  ;000110
    S7   = /QF * /QE * /QD *  QC *  QB *  QA  ;000111
    S8   = /QF * /QE *  QD * /QC * /QB * /QA  ;001000
    S9   = /QF * /QE *  QD * /QC * /QB *  QA  ;001001
    S10  = /QF * /QE *  QD * /QC *  QB * /QA  ;001010
    S11  = /QF * /QE *  QD * /QC *  QB *  QA  ;001011
    S12  = /QF * /QE *  QD *  QC * /QB * /QA  ;001100
    S13  = /QF * /QE *  QD *  QC * /QB *  QA  ;001101
    S14  = /QF * /QE *  QD *  QC *  QB * /QA  ;001110
    S15  = /QF * /QE *  QD *  QC *  QB *  QA  ;001111
    S16  = /QF *  QE * /QD * /QC * /QB * /QA  ;010000
    S17  = /QF *  QE * /QD * /QC * /QB *  QA  ;010001
    S18  = /QF *  QE * /QD * /QC *  QB * /QA  ;010010
    S19  = /QF *  QE * /QD * /QC *  QB *  QA  ;010011
    S20  = /QF *  QE * /QD *  QC * /QB * /QA  ;010100
    S21  = /QF *  QE * /QD *  QC * /QB *  QA  ;010101
    S22  = /QF *  QE * /QD *  QC *  QB * /QA  ;010110
    S23  = /QF *  QE * /QD *  QC *  QB *  QA  ;010111
    S24  = /QF *  QE *  QD * /QC * /QB * /QA  ;011000
    S25  = /QF *  QE *  QD * /QC * /QB *  QA  ;011001
    S26  = /QF *  QE *  QD * /QC *  QB * /QA  ;011010
    S27  = /QF *  QE *  QD * /QC *  QB *  QA  ;011011
```

```
    S28  = /QF *  QE *  QD *  QC * /QB * /QA  ;011100
    S29  = /QF *  QE *  QD *  QC * /QB *  QA  ;011101
    S30  = /QF *  QE *  QD *  QC *  QB * /QA  ;011110
    S31  = /QF *  QE *  QD *  QC *  QB *  QA  ;011111
    S32  =  QF * /QE * /QD * /QC * /QB * /QA  ;100000
    S33  =  QF * /QE * /QD * /QC * /QB *  QA  ;100001
    S34  =  QF * /QE * /QD * /QC *  QB * /QA  ;100010
    S35  =  QF * /QE * /QD * /QC *  QB *  QA  ;100011
    S36  =  QF * /QE * /QD *  QC * /QB * /QA  ;100100
    S37  =  QF * /QE * /QD *  QC * /QB *  QA  ;100101
    S38  =  QF * /QE * /QD *  QC *  QB * /QA  ;100110
    S39  =  QF * /QE * /QD *  QC *  QB *  QA  ;100111
    S40  =  QF * /QE *  QD * /QC * /QB * /QA  ;101000
    S41  =  QF * /QE *  QD * /QC * /QB *  QA  ;101001
    S42  =  QF * /QE *  QD * /QC *  QB * /QA  ;101010
    S43  =  QF * /QE *  QD * /QC *  QB *  QA  ;101011
    S44  =  QF * /QE *  QD *  QC * /QB * /QA  ;101100
    S45  =  QF * /QE *  QD *  QC * /QB *  QA  ;101101
    S46  =  QF * /QE *  QD *  QC *  QB * /QA  ;101110
    S47  =  QF * /QE *  QD *  QC *  QB *  QA  ;101111
    S48  =  QF *  QE * /QD * /QC * /QB * /QA  ;110000
    S49  =  QF *  QE * /QD * /QC * /QB *  QA  ;110001
    S50  =  QF *  QE * /QD * /QC *  QB * /QA  ;110010
    S51  =  QF *  QE * /QD * /QC *  QB *  QA  ;110011
    S52  =  QF *  QE * /QD *  QC * /QB * /QA  ;110100
    S53  =  QF *  QE * /QD *  QC * /QB *  QA  ;110101
    S54  =  QF *  QE * /QD *  QC *  QB * /QA  ;110110
    S55  =  QF *  QE * /QD *  QC *  QB *  QA  ;110111
    S56  =  QF *  QE *  QD * /QC * /QB * /QA  ;111000
    S57  =  QF *  QE *  QD * /QC * /QB *  QA  ;111001
    S58  =  QF *  QE *  QD * /QC *  QB * /QA  ;111010
    S59  =  QF *  QE *  QD * /QC *  QB *  QA  ;111011
    S60  =  QF *  QE *  QD *  QC * /QB * /QA  ;111100
    S61  =  QF *  QE *  QD *  QC * /QB *  QA  ;111101
    S62  =  QF *  QE *  QD *  QC *  QB * /QA  ;111110
    S63  =  QF *  QE *  QD *  QC *  QB *  QA  ;111111

;state transitions

    S0   := ( DIR) -> S1              ;counting up, next state = S1
          + (/DIR) -> S63             ;counting down, next state = S255
    S1   := ( DIR) -> S2
          + (/DIR) -> S0
    S2   := ( DIR) -> S3
          + (/DIR) -> S1
    S3   := ( DIR) -> S4
          + (/DIR) -> S2
    S4   := ( DIR) -> S5
          + (/DIR) -> S3
    S5   := ( DIR) -> S6
          + (/DIR) -> S4
    S6   := ( DIR) -> S7
          + (/DIR) -> S5
    S7   := ( DIR) -> S8
          + (/DIR) -> S6
    S8   := ( DIR) -> S9
          + (/DIR) -> S7
    S9   := ( DIR) -> S10
          + (/DIR) -> S8
    S10  := ( DIR) -> S11
          + (/DIR) -> S9
    S11  := ( DIR) -> S12
          + (/DIR) -> S10
    S12  := ( DIR) -> S13
          + (/DIR) -> S11
    S13  := ( DIR) -> S14
          + (/DIR) -> S12
    S14  := ( DIR) -> S15
          + (/DIR) -> S13
    S15  := ( DIR) -> S16
          + (/DIR) -> S14
    S16  := ( DIR) -> S17
          + (/DIR) -> S15
    S17  := ( DIR) -> S18
          + (/DIR) -> S16
    S18  := ( DIR) -> S19
          + (/DIR) -> S17
    S19  := ( DIR) -> S20
          + (/DIR) -> S18
    S20  := ( DIR) -> S21
          + (/DIR) -> S19
```

```
S21  := ( DIR) -> S22
      + (/DIR) -> S20
S22  := ( DIR) -> S23
      + (/DIR) -> S21
S23  := ( DIR) -> S24
      + (/DIR) -> S22
S24  := ( DIR) -> S25
      + (/DIR) -> S23
S25  := ( DIR) -> S26
      + (/DIR) -> S24
S26  := ( DIR) -> S27
      + (/DIR) -> S25
S27  := ( DIR) -> S28
      + (/DIR) -> S26
S28  := ( DIR) -> S29
      + (/DIR) -> S27
S29  := ( DIR) -> S30
      + (/DIR) -> S28
S30  := ( DIR) -> S31
      + (/DIR) -> S29
S31  := ( DIR) -> S32
      + (/DIR) -> S30
S32  := ( DIR) -> S33
      + (/DIR) -> S31
S33  := ( DIR) -> S34
      + (/DIR) -> S32
S34  := ( DIR) -> S35
      + (/DIR) -> S33
S35  := ( DIR) -> S36
      + (/DIR) -> S34
S36  := ( DIR) -> S37
      + (/DIR) -> S35
S37  := ( DIR) -> S38
      + (/DIR) -> S36
S38  := ( DIR) -> S39
      + (/DIR) -> S37
S39  := ( DIR) -> S40
      + (/DIR) -> S38
S40  := ( DIR) -> S41
      + (/DIR) -> S39
S41  := ( DIR) -> S42
      + (/DIR) -> S40
S42  := ( DIR) -> S43
      + (/DIR) -> S41
S43  := ( DIR) -> S44
      + (/DIR) -> S42
S44  := ( DIR) -> S45
      + (/DIR) -> S43
S45  := ( DIR) -> S46
      + (/DIR) -> S44
S46  := ( DIR) -> S47
      + (/DIR) -> S45
S47  := ( DIR) -> S48
      + (/DIR) -> S46
S48  := ( DIR) -> S49
      + (/DIR) -> S47
S49  := ( DIR) -> S50
      + (/DIR) -> S48
S50  := ( DIR) -> S51
      + (/DIR) -> S49
S51  := ( DIR) -> S52
      + (/DIR) -> S50
S52  := ( DIR) -> S53
      + (/DIR) -> S51
S53  := ( DIR) -> S54
      + (/DIR) -> S52
S54  := ( DIR) -> S55
      + (/DIR) -> S53
S55  := ( DIR) -> S56
      + (/DIR) -> S54
S56  := ( DIR) -> S57
      + (/DIR) -> S55
S57  := ( DIR) -> S58
      + (/DIR) -> S56
S58  := ( DIR) -> S59
      + (/DIR) -> S57
S59  := ( DIR) -> S60
      + (/DIR) -> S58
S60  := ( DIR) -> S61
      + (/DIR) -> S59
S61  := ( DIR) -> S62
      + (/DIR) -> S60
```

```
    S62  := ( DIR) -> S63
          + (/DIR) -> S61
    S63  := ( DIR) -> S0
          + (/DIR) -> S62


EQUATIONS

   CLK_O = /QF  ;next counter active on rising edge

   DIR_O = DIR

   QA.CLKF = CLK
   QB.CLKF = CLK
   QC.CLKF = CLK
   QD.CLKF = CLK
   QE.CLKF = CLK
   QF.CLKF = CLK

   QA.RSTF = RST
   QB.RSTF = RST
   QC.RSTF = RST
   QD.RSTF = RST
   QE.RSTF = RST
   QF.RSTF = RST


SIMULATION

   VECTOR status := [QF, QE, QD, QC, QB, QA]   ;to see counter in hex
   SETF /RST /CLK DIR                          ;set to count up

   CLOCKF RST

   FOR j:=0 TO 65 DO  ;count up
       BEGIN
           CLOCKF CLK
       END

   SETF /DIR          ;set to count down

   FOR j:=0 TO 67 DO  ;count down
       BEGIN
           CLOCKF CLK
       END

;end simulation
```

E.4.3 Shaft Encoder Interface PLD Code – "MOD64" and "MOD128"

```
Title       Direction-toggle V-F inhibitor
Pattern     PDS
Revision    2
Author      David Wells
Company     Aston University
Date        2/9/98

CHIP        FV_Inhibit  22V10

;When a stationary shaft is at a transition boundary the position will be
;observed to be oscillating between adjacent
;This design inhibits the F->V converter input from CNTRIF for seven state-changes
;after a change of shaft direction.

;Pin Declarations

;inputs
PIN         1     CLK
PIN         2     SPARE1
PIN         3     CLK_EDGE    ;1 for +ve edge, 0 for -ve edge
PIN         4     DIR
PIN         5     CNTR_CLK

;outputs
PIN         23    NEW_CLK
PIN         22    D0
PIN         21    D1
PIN         20    D2
PIN         19    D3
PIN         18    D4
PIN         17    D5
PIN         16    D6
PIN         15    TO_V_F
PIN         14    DIR_STBL


EQUATIONS

      D0.CLKF = CLK
      D1.CLKF = CLK
      D2.CLKF = CLK
      D3.CLKF = CLK
      D4.CLKF = CLK
      D5.CLKF = CLK
      D6.CLKF = CLK

      D0.RSTF = GND
      D1.RSTF = GND
      D2.RSTF = GND
      D3.RSTF = GND
      D4.RSTF = GND
      D5.RSTF = GND
      D6.RSTF = GND

      D0.SETF = GND
      D1.SETF = GND
      D2.SETF = GND
      D3.SETF = GND
      D4.SETF = GND
      D5.SETF = GND
      D6.SETF = GND

      D0.D := DIR
      D1.D := D0
      D2.D := D1
      D3.D := D2
      D4.D := D3
      D5.D := D4
      D6.D := D5

      NEW_CLK = CLK * /CLK_EDGE + /CLK * CLK_EDGE

      DIR_STBL = (( DIR *  D0 *  D1 *  D2 * D3 * D4  * D5  *  D6)
                 + (/DIR * /D0 * /D1 * /D2* /D3 * /D4 * /D5 * /D6))

      TO_V_F = DIR_STBL * CNTR_CLK

; end of design
```

```
SIMULATION

;simulate an oscillation on a near stationary shaft in one quadrant
;(see state graph appendix E)
; S0 -> S8 -> S9 -> S10 -> S6 -> S7
; ^_____________________________________|

      SETF /CLK
      SETF /CNTR_CLK /DIR     ;S0
      CLOCKF CLK              ;put shift regiseters into a known state
      CLOCKF CLK
      CLOCKF CLK
      SETF /CNTR_CLK  DIR     ;S8
      CLOCKF CLK
      SETF  CNTR_CLK  DIR     ;S9
      CLOCKF CLK
      SETF /CNTR_CLK  DIR     ;S10
      CLOCKF CLK
      SETF /CNTR_CLK /DIR     ;S6
      CLOCKF CLK
      SETF  CNTR_CLK /DIR     ;S7
      CLOCKF CLK
      SETF /CNTR_CLK /DIR     ;S0
      CLOCKF CLK

;normal clockwise rotation
      SETF  CNTR_CLK /DIR     ;S1
      CLOCKF CLK
      SETF /CNTR_CLK /DIR     ;S2
      CLOCKF CLK
      SETF  CNTR_CLK /DIR     ;S3
      CLOCKF CLK
      SETF /CNTR_CLK /DIR     ;S4
      CLOCKF CLK
      SETF  CNTR_CLK /DIR     ;S5
      CLOCKF CLK
      SETF /CNTR_CLK /DIR     ;S6
      CLOCKF CLK
      SETF  CNTR_CLK /DIR     ;S7
      CLOCKF CLK
      SETF /CNTR_CLK /DIR     ;S0
      CLOCKF CLK

;end simulation
```