# The application of self-tuning control to rolling mill systems.

GROVES, Christopher N.

# The Application of Self-Tuning Control to Rolling Mill Systems

Christopher Nigel Groves

A Thesis Submitted in Partial Fulfilment of
the Requirements of
Sheffield Hallam University
for the Degree of Master of Philosophy

May 1993

Collaborating Organisation: Davy International Sheffield

## Abstract

The introduction of self-tuning control techniques to the Automatic Gauge Control (AGC) system of a rolling mill is considered. This is a new application of pole-placement self-tuning control and no comparable examples have been found in the literature.

Initially, an existing ACSL model was investigated and a simpler version, suitable for on-line use was developed. A simpler MATLAB model of the system was also produced.

A data logging exercise was then carried out to allow PRBS testing and provide data for model validation. System identification and correlation analysis were then used to obtain a simplified parametric model. Included in this stage was a thorough investigation into the practical aspects of the PRBS method for system identification and the requirements for successful implementation noted.

Through extensive simulation studies it was successfully demonstrated that a pole-placement self-tuning controller using simple least squares estimation can be used to control the position loop of the AGC system. There appears no good reason why this procedure could not be applied to other rolling mill control loops.

## Acknowledgements

This project was carried out under a Teaching Company Scheme between Sheffield Hallam University (SHU) and Davy International Sheffield. I am extremely grateful for the assistance given to me by these three organisations and for that provided by Dr. K. Dutton (SHU) and Mr. M.T. Clark (Davy).

## Table of contents

# 1.0 INTRODUCTION

This project was carried out in collaboration with Davy International Sheffield who supply manufacturing and processing equipment to the worlds metals industries. One of their main areas of expertise is Automatic Gauge Control (AGC) systems applied to rolling mills. As a result of Davy completing a program to develop a DDC (Direct Digital Control) system, the foundations existed on which to develop a more sophisticated controller. The current DDC basically does the same job as an analogue system and as such, has no real advantages over the old analogue loops, except that the problem of drifting signals is removed and it is easier to log data from the digital system. Some form of adaptive control would hopefully tackle non-linearities and reduce the time spent on commissioning. In addition to these benefits, such a system would also improve control over a wide range of materials.

## 1.1 Background (History)

Processes have been controlled by conventional Proportional plus Integral plus Derivative (PID) [1,2,3] controllers for over half a century [4,5]. This strategy has remained unchanged despite the introduction of microprocessor based control systems. However, the control calculations involved in a microprocessor implementation of a PID controller algorithm are fairly trivial [6]. With an increase in computer power and a decrease in costs, the next step was to provide further features. Although it is generally recognised that the conventional PID controller is very effective, in practice its initial tuning and any subsequent tuning on a plant with many control loops can be very time consuming. The effectiveness of a PID controller is also reduced on plants with varying parameters. Therefore one obvious method for improvement was to provide self-tuning and adaptive capabilities [7,8,9,10].

A self-tuning controller has the ability to calculate its own control parameters, assumed unknown but constant, at the time of commissioning. An adaptive controller extends this concept by altering its control parameters on-line to suit changing process conditions. These two ideas are virtually the same, incorporating similar estimation algorithms, see section 1.2.1 below. Potential candidates for the application of adaptive control are processes that are non-linear and stochastic, thus, they are difficult to analyse and control. If they were linear they could be controlled by classical linear controllers. If they were not stochastic or uncertain in some way there would be no requirement to learn in the form of self adjustment or coefficient estimation. There are no general design procedures for non-linear stochastic processes.

Adaptive control theory has been developed over the last twenty years and several thousand papers have been written [11,12,13,14]. Although there appears to be a great number of practical applications for such techniques, and developments in computer technology have resulted in making implementation of the algorithms possible, the number of successful applications actually reported is very limited. The main reason for this is that much of the information, for competitive reasons, is kept "in-house" thus, specific details remain unpublished.

## 1.2 Adaptive schemes

Three classes of theoretical adaptive controllers exist [15,16]:

In a *gain scheduling* controller the designer makes use of his knowledge of the process non-linearities to develop a schedule for the controller gain (plus derivative and integral gains as required). The schedules are then used to program the controller coefficients as a function of one or more of the process variables. Gain scheduling is an open loop compensation and can be thought of as a feedback control system in which the gains are adjusted by feed forward compensation. There is no feedback from the performance of the closed loop system to compensate for an incorrect schedule. It is contentious whether gain scheduling should be considered as an adaptive system because the parameters are changed in open loop.

In *model reference* adaptive control systems [17] the user defines at the outset, the closed loop system response. The desired response to set point changes is expressed in mathematical terms and then implemented as a dynamic model. The model is subjected to the actual set point changes and calculates in real time, the corresponding idealised output. Within the model reference system is a conventional feedback controller with adjustable coefficients. The process and model outputs are compared and the differences are used to adjust the coefficients of the feedback controller.

The two schemes above are known as *implicit* (or direct) methods because the adjustment rules tell directly how the controller parameters should be updated. A different scheme is obtained if the process model is updated and then the updated controller coefficients are obtained from the solution of a design problem, this is often termed an *explicit method*. A self-tuning controller uses a highly simplified and linear mathematical model of the process. The technique uses a feedback control system together with an identification system which continually monitors the process inputs and outputs to produce a mathematical representation of the process dynamics. This model is in the form of a generalised equation with a finite number of parameters which are estimated recursively. The model is then used to calculate the coefficients of the controller based on a defined performance requirement. Thus, there are two separate functions being carried out, the first is a learning process (model generation) and the second is the actual control of its behaviour [15]. It is useful at this stage to provide brief details of the methods available for each function.

### 1.2.1 Estimation

In adaptive controllers observations of the system input and output signals are obtained sequentially in real time. It is desirable to make calculations recursively in order to save computation time [10]. The algorithms are therefore arranged in such a way that the results already obtained can be used to obtain the new estimate of the current dynamics of the process.

The most commonly used family of algorithms is that of least squares [11,18,19] as they are easy to program and work well. If the same input is applied to both the system and the mathematical model the outputs can be compared, giving rise to an error which is dependent on how poor the estimates are. This gives an indication of how close the model is to the actual system. In an attempt to obtain a better model, the sum of the squares of the errors is minimised by the adjustment of the model parameters.

### 1.2.2 Design

Self-tuning controllers are flexible with respect to a design method. Virtually any design technique can be accommodated. Minimum variance control is especially useful in the design of regulators which maintain an output value constant despite process disturbances. It is most effective when the process is not changing. The main disadvantage of this method is that it does not perform well when the set point is being varied.

A simple and direct design procedure is that of pole placement. The objective here is to locate the closed loop system poles at pre specified locations in order to obtain the desired response. Designs based on gain and phase margins and on linear quadratic Gaussian control have also been developed [15,16].

## 1.3 Automatic Gauge Control (AGC) [20,21]

The gauge or thickness of a rolled piece of metal may vary across its width or, along its length. Normally variations across the width are associated with shape or flatness control and variations along the length, with gauge control. Both these factors must be controlled effectively in real time [22], to meet customers' requirements for higher tolerances, increased rolling speeds for higher productivity and to reduce "off-gauge" material, that would be scrapped or need re-working.

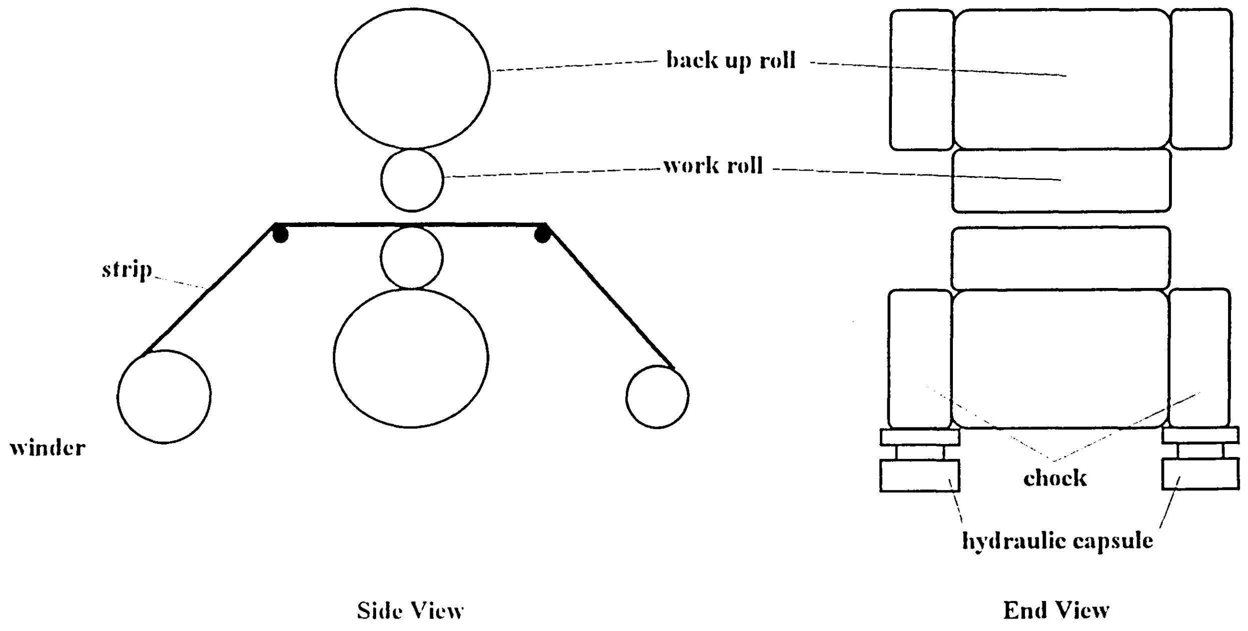


**Figure 1.1, Mill stand.**

The system in question is a single stand cold reversing mill which uses fast acting hydraulic cylinders or capsules for roll adjustment. Figure 1.1 shows the mill stand configuration, consisting of, a pay-off winder, two work rolls, two back up rolls and a take-up winder. The back up rolls are housed in chocks (as are the work rolls normally) and the hydraulic cylinders are placed between these chocks and the stand housings on either side of the mill, also see section 3.2.2.

Before rolling can begin, the mill must be threaded. This involves slowly taking strip off the pay-off winder and passing it through the mill between the work rolls, to the take-up winder. The mill is now set up to roll the strip which will be carried out at a greater speed. Almost the full length of strip is then passed between the work rolls, set to give the required gauge reduction, leaving the mill threaded and the end of the strip in what was the pay-off winder. The gap between the work rolls is then reduced further and the process repeated in the reverse direction. For operational purposes the number of passes will generally be an odd number. Both the winders are driven and controlled by the mill control system of which the AGC is only a part. In the roll gap, the metal is plastically elongated in the direction of rolling. As the strip passes between the work rolls it speeds up, thus, the strip speed is quicker on the exit side of the mill.

The control objective of the AGC system is to ensure minimal variation in out going strip thickness, which is controlled by varying the gap between the work rolls. Such a system uses one basic mode of control around the hydraulic capsules, either position control or load (pressure) control. Other more complex control loops produce trims which adjust the reference to the basic loop. For example, the out going strip thickness can be easily measured and this value fed back for comparison with the desired thickness. However, the thickness can only be measured “down stream” of the roll gap which leads to a transport delay. As a result such measurements can only be used as slow trims, leaving the major control effort to the fast acting position or load control loop around the capsule. A simple extension to this principle is to fit a thickness measuring device to the entry side of the mill and thus allow the control system to anticipate errors before the strip enters the roll gap and correct them using feedforward control.

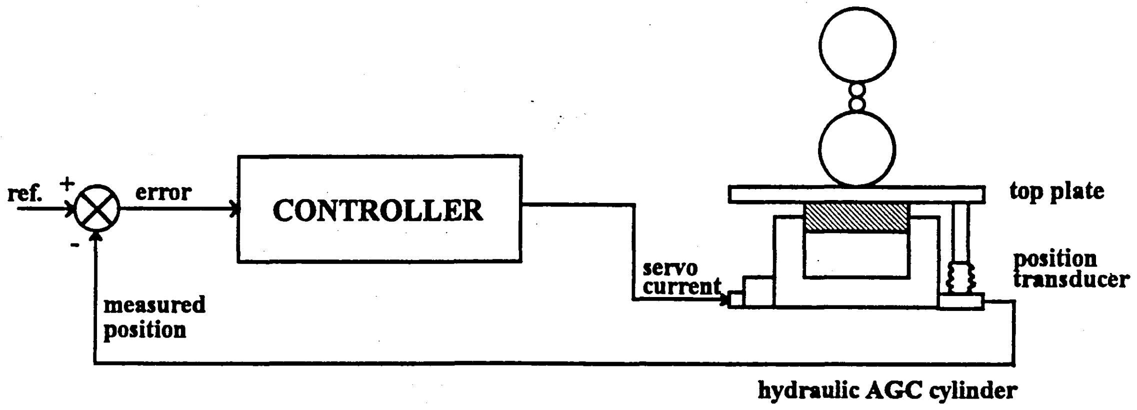


**Figure 1.2, Position loop of AGC.**

Figure 1.2 shows the basic position loop of the hydraulic AGC. Note 1, Appendix F. In order to convert to a load loop, the position transducer would be replaced with a pressure transducer. In practice there would be two servo valves and two transducers of each type to improve accuracy. In either mode, the major parameters affecting the exit gauge are tension, entry gauge disturbances, material hardness changes and roll eccentricity. In general, only proportional control is used. The servo valve and hydraulic cylinder provide a natural integrator. Flow is proportional to input current and flow integrates to give a load change. An additional integral term may sometimes be added to remove any remaining steady state errors. In general, load control loops are more difficult to stabilise than position loops.

## 1.4 Project Objectives

The overall aim of the project is to investigate some form of adaptive control for an hydraulic AGC system applied to a single stand cold reversing mill. Initially, effort will be directed towards self-tuning and will only be concerned with the position loop controlling the hydraulic capsules. The response of this loop is well defined and the loop can be modelled relatively easily from its basic components. If successful, the development of an adaptive system should:

- reduce commissioning time
- improve control over a wide range of materials
- be a selling point
- allow re-tuning by the mill operator.

A further advantage is that the lessons learnt and algorithms developed here could be applied to other loops and systems at a later stage.

## 1.5 Major Contributions

This is a new application and the proposed control technique has been shown to be possible through extensive simulation studies; in addition, no comparable applications have been found in the literature. In addition to the simulation work, logging of plant data was also carried out to aid in the modelling of the system. The practical aspects of the PRBS method for system identification, which are omitted from the texts, have been thoroughly investigated and the requirements for successful implementation noted.

# 2.0 MODELLING OF A SINGLE STAND COLD REVERSING MILL

The suggested self-tuning controller of chapter one requires a simplified, linear model of the system to be controlled. In addition to this model, a more complex model would be extremely useful in the development stage for simulating the controller's response. The difficulty in building a mathematical model is balancing two opposing factors; including all the relevant factors affecting system response and minimising computing time. The question of overall accuracy must also be answered, ie what are the "relevant factors"? These problems are obviously amplified under the constraints of real time computing.

A block diagram of a simple mill stand model based on physical elements, relating servo valve current to hydraulic cylinder position is shown in figure 2.1. Note 2, Appendix F.

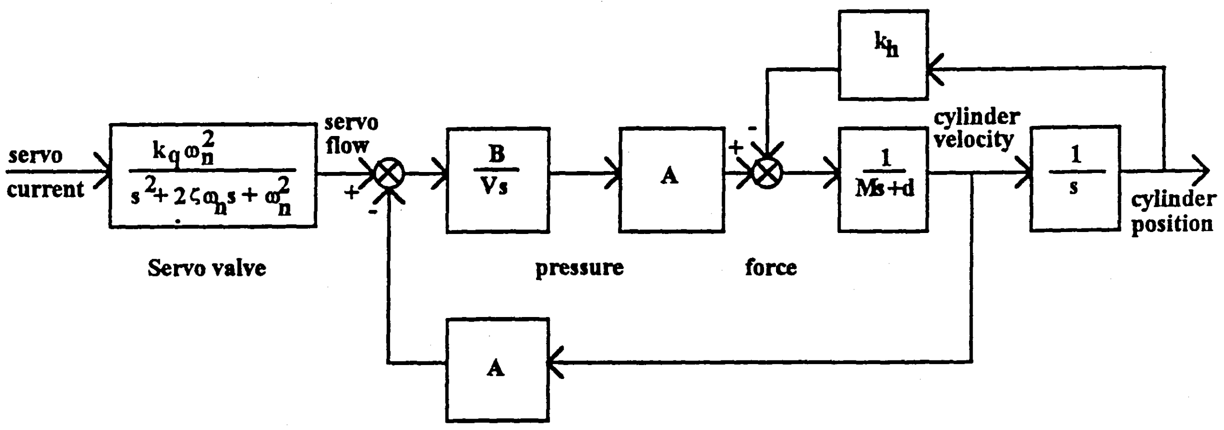


**Figure 2.1. Block diagram of mill stand model.**

where:

| | |
|---|---|
| $\omega_n$ | natural frequency of servo valve (rad.$s^{-1}$) |
| $\zeta$ | damping ratio for servo valve |
| $k_q$ | flow gain of servo valve ($m^3$/s per A) |
| B | oil compressibility ($Nm^{-2}$) |
| A | cylinder area ($m^2$) |
| V | cylinder volume ($m^3$) |
| M | moving mass of mill (kg) |
| $k_h$ | mill housing stiffness ($Nm^{-1}$) |

continued...

d damping term = $2.\mu.\sqrt{(k_h.M)}$ ($\mu$ = damping factor ~.1)

The servo valve is represented by a second order transfer function, with natural frequency $\omega_n$, damping ratio $\zeta$ and gain $k_q$. The output of the servo valve is oil flow to the hydraulic cylinder. Flow into the cylinder will obviously be affected by any movement of the ram, altering the available volume and hence the pressure. The flow is integrated to give the pressure change and thus, a change in the force acting on the ram. This force is also dependent on the ram's current position. The actual cylinder is then represented by a first order transfer function with the force acting on the ram as the input producing a movement. This cylinder model contains a damping term, d, which is calculated from the mill housing stiffness, the moving mass of the mill and a damping factor, $\mu$, to produce the required response.

## 2.1 The ACSL mill model

The ACSL (Advanced Continuous Simulation Language) [23] mill model is a computer model of a rolling mill designed for the simulation and testing of gauge control systems. This model was developed and used previously by Davy. The model is two dimensional in the sense that it does not consider variations across the roll bite. A complete mill stand model can be constructed from basic blocks, or modules, that include, mill stand with hydraulic capsule, main motor drive with speed control and pay-off and take-up winders. Each block being a complex non-linear ACSL model of that particular component.

In order to understand the principles involved, a mill model was constructed by connecting together the major modules. Once initialised, the model ran satisfactorily. However, difficulties were experienced in maintaining tension between the roll bite and the winders on either side of the stand. The model had been set up using data from a number of previous projects and as such was not related to any one particular real mill. The model thus required tailoring to a specific mill, it was hoped that this step would reduce the stability problems being experienced.

## 2.1.1 Enhancements and simplifications

The existing model was found to be extremely complex and slow in execution and so not suitable for any real time simulations that may be required later. The electrical drive modules were particularly detailed and were used in three sections of the overall mill stand model: in the main drive and in both the pay-off and take-up winders. Thus, any improvements made here would be amplified.

The main drive system was isolated as an ACSL model with two inputs (rolling torque and reference speed) and one output (roll speed). A frequency response and a number of step tests were carried out on the model in order to understand its operation. These results would also act as a reference when testing any alterations. Throughout the tests the armature voltage was monitored for saturation which would introduce a non linearity into the system.

The step responses were found to be basically second order. For small steps, up to base speed, a second order model was fitted, working with percentage overshoot and time to cross final value. It was assumed that by alteration of the gain at higher speeds and loads, a similar response would be obtained, and that this was representative of real drive control systems. To remove any steady state error between the reference speed and the actual speed an integral term was added to the proportional controller and a provision made to avoid integral wind-up. Figure 2.2 shows the block diagram for the simplified drive model.

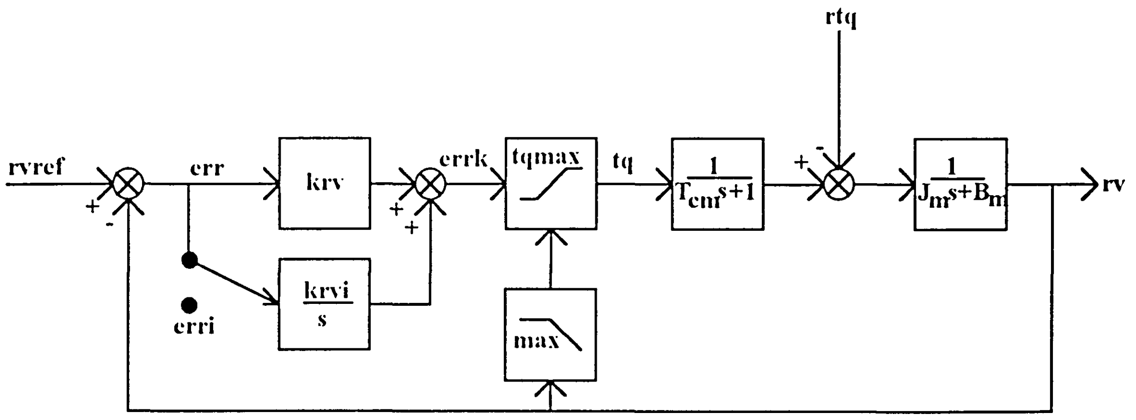


Figure 2.2. Simplified drive model

where:

| | |
|---|---|
| rvref | speed reference (rad.$s^{-1}$) |
| rtq | rolling torque (Nm) |
| rv | mill speed (rad.$s^{-1}$) |
| Jm | moment of inertia (kg$m^2$) |
| Bm | Damping coefficient (5% of max. torque) |
| krv | torque speed gain (Nm/rad.$s^{-1}$) |
| krvi | integral gain (Nm.$rad^{-1}$) |
| tqmax | maximum torque (Nm) (dependent on speed) |
| Tcm | time constant (s) |

The system has two inputs, reference speed, rvref and rolling torque (load), rtq, and one output, mill speed, rv. The difference between the actual speed and the reference produces an error, err, for the proportional plus integral controller to give the control signal, errk (the term erri is used in the prevention of integral wind-up). The maximum torque of the motor, tqmax, is limited by the motor's speed, so the value of tqmax is altered as the motor speed changes once it is greater than the base speed of the motor. A lag term is introduced to provide a phase lag between a speed change and a torque change. The rolling torque, rtq, is the load on the motor, this is subtracted from the torque produced by the motor to give the torque available for acceleration. This torque is then transferred to a speed through a first order transfer function. The ACSL code for this simplified model can be found in appendix A1.

The response of the modified drive model to a step in reference speed of amplitude of .05 rad.s$^{-1}$, from an initial speed of 5 rad.s$^{-1}$, (approximately the base speed of the motor being modelled) is shown in figure 2.3. Figures 2.4 and 2.5 also show step responses of the modified drive model, in both cases the amplitude of the step is much greater, 5 rad.s$^{-1}$. Figure 2.4 shows the step from 5 rad.s$^{-1}$ to 10 rad.s$^{-1}$ and figure 2.5 a similar step but from a much greater initial speed of 30 rad.s$^{-1}$. These two responses although unrealistic, (a ramp would be more appropriate for larger changes in speed), clearly show the effect of torque available for acceleration limiting the motor's speed response.

The other area for improvement was the complete winder models. Although there are two separate models, they are basically the same. The winders are used to pay out the metal strip in to the mill for rolling and to coil the strip after its gauge has been reduced. As above, the ACSL code was isolated (the take-up reel was used). On investigation this model was found to be very oscillatory and almost impossible to set up as a stand alone model. It was therefore decided to start again. The major problem in modelling this system was where to introduce damping. It is not known where exactly the damping comes from in the actual system. Two methods were investigated; damping as a result of frictional and other forces acting on the winder against the direction of rotation, and damping as a result of the tension within the strip. Another major problem is that of inertia. Obviously, as a winder pays-off or takes-up strip its radius and therefore its moment of inertia alters.

A model was built which included a damping term based on the tension of the strip, figure 2.6.

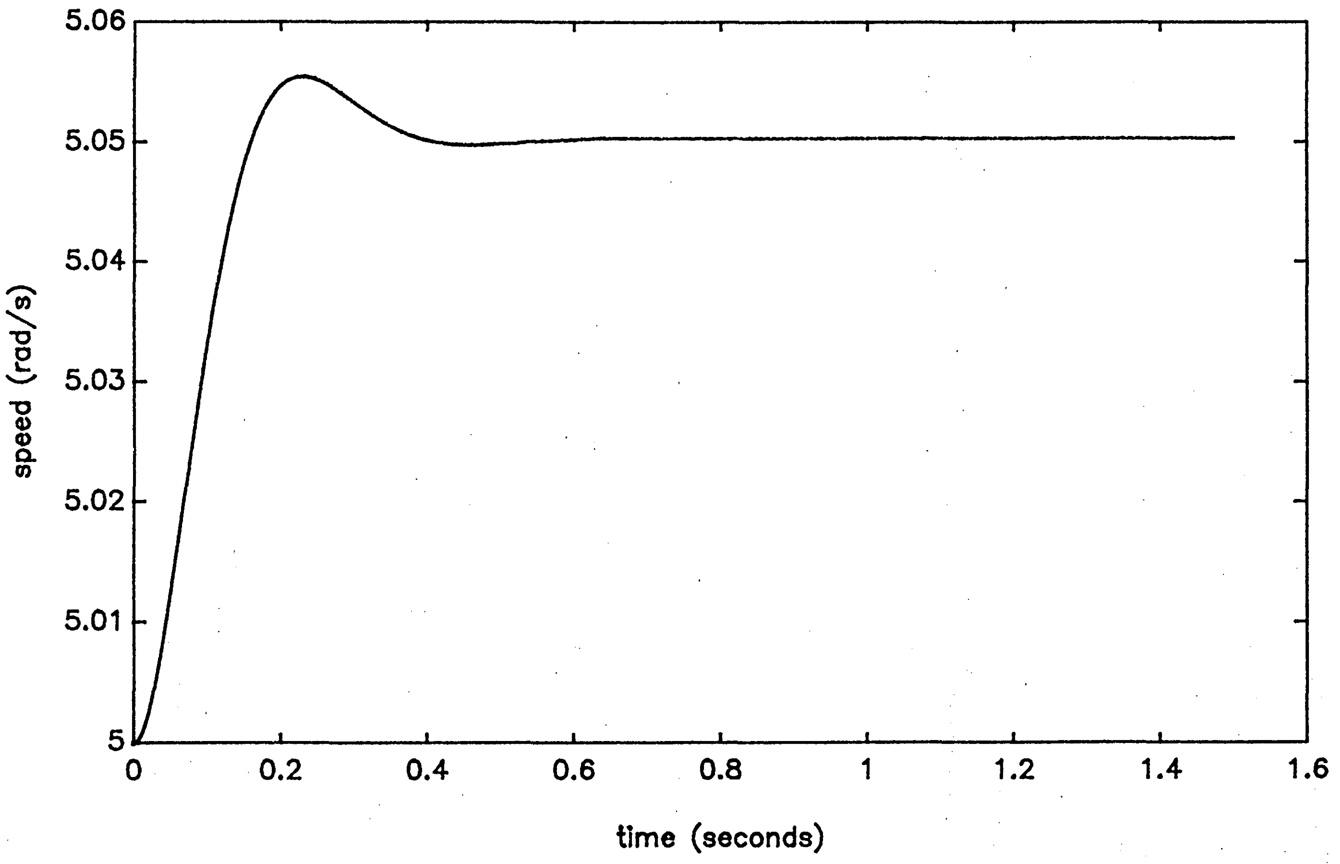


**Figure 2.3, Simplified drive model, step response (i).**

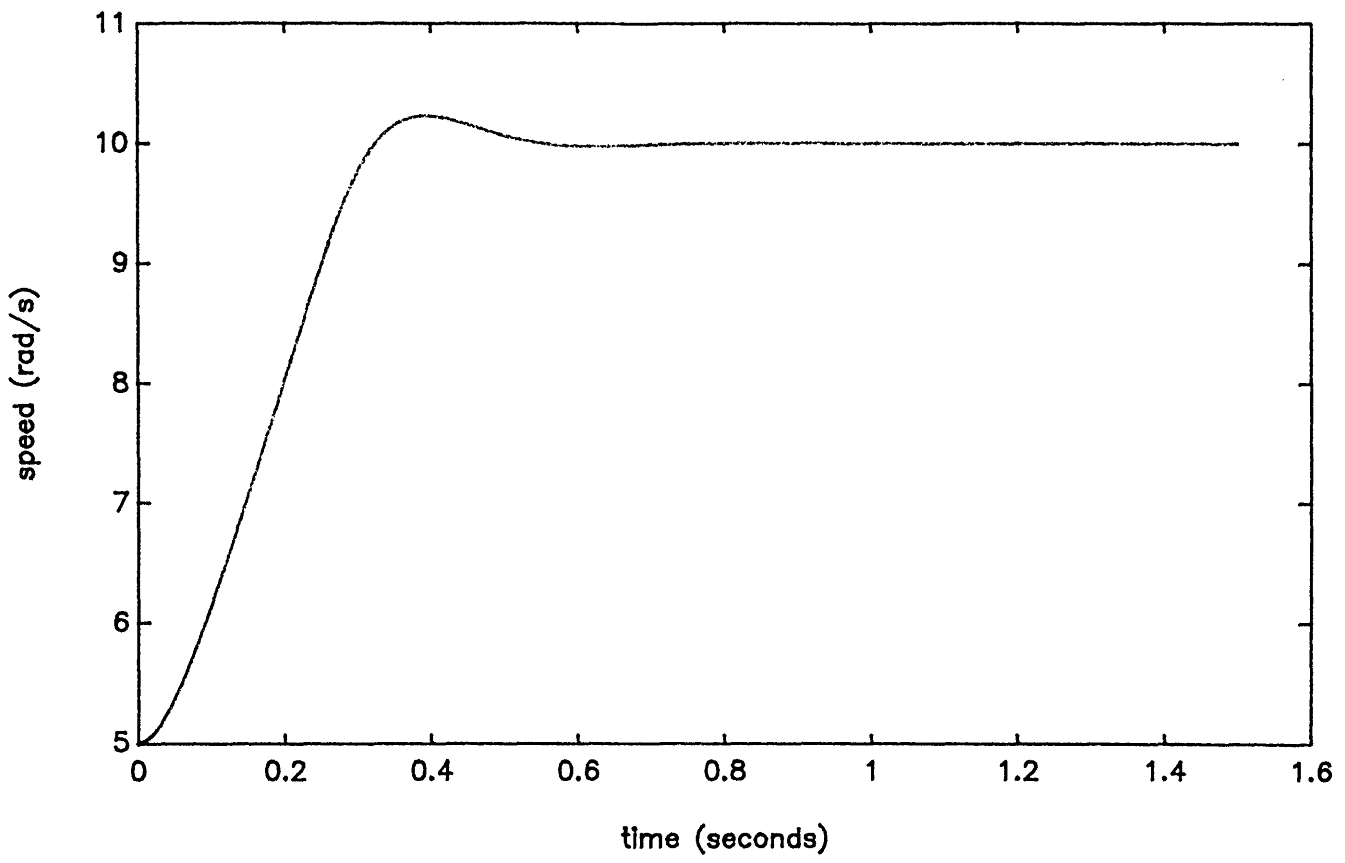


**Figure 2.4, Simplified drive model, step response (ii).**

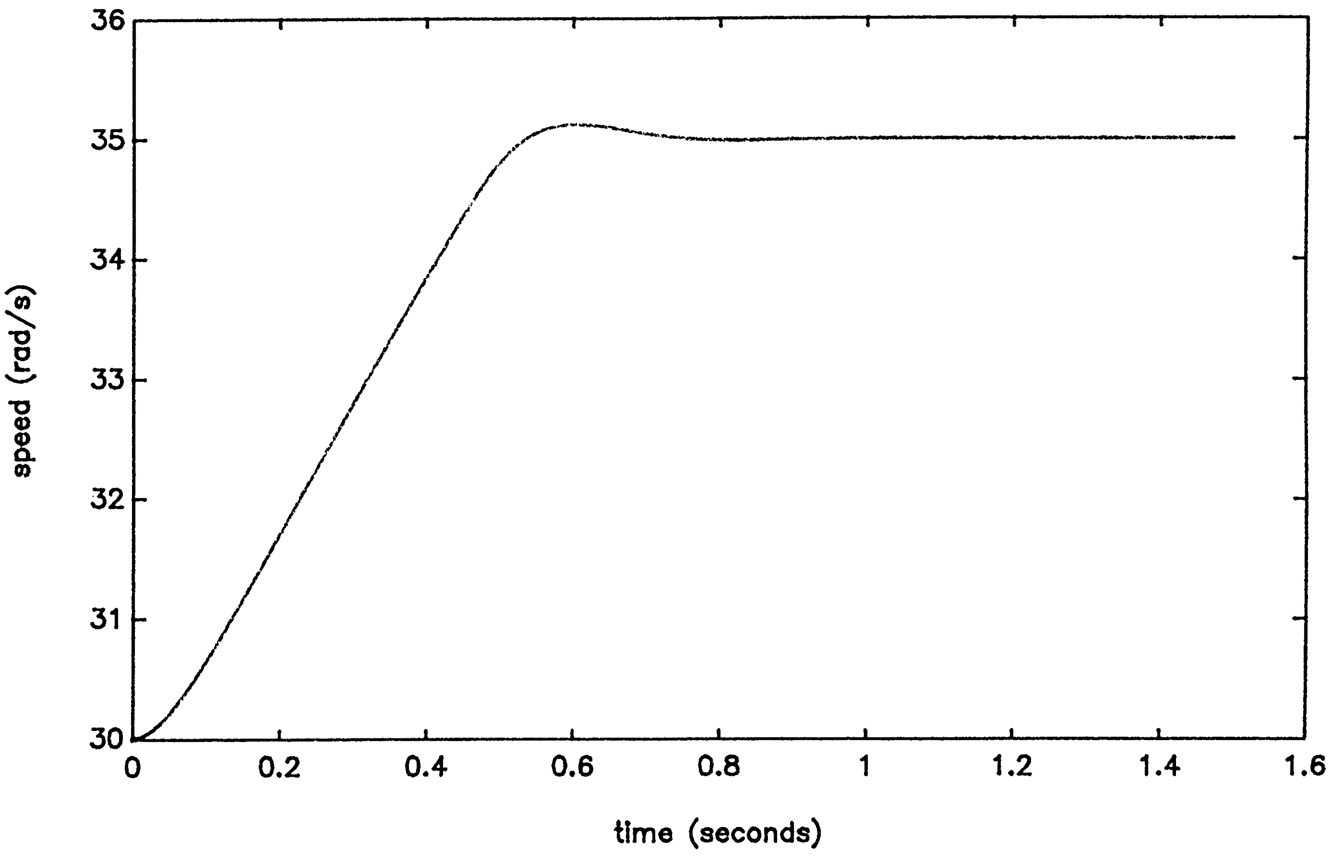


**Figure 2.5, Simplified drive model, step response (iii).**

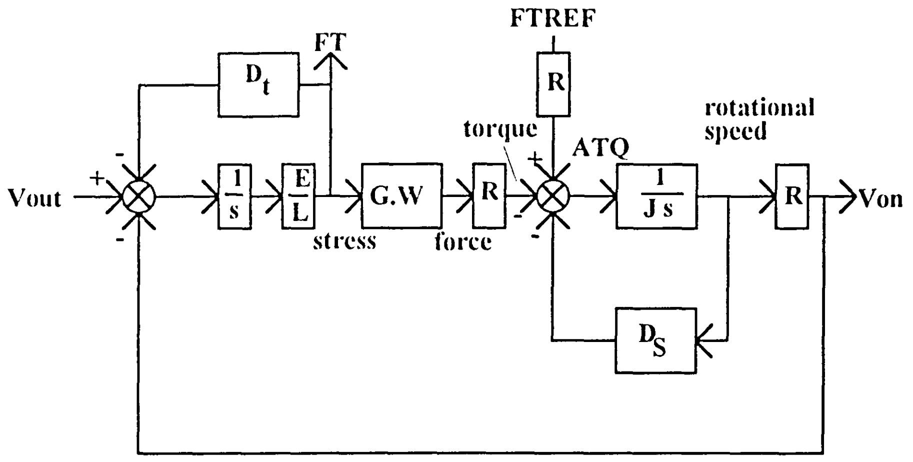


Figure 2.6, Simplified winder model block diagram.

where:

| | |
|---|---|
| Vout | speed out of mill ($m.s^{-1}$) |
| FTREF | Ref. tension force (N) |
| Von | speed onto winder ($m.s^{-1}$) (winder peripheral speed) |
| FT | Front tension ($N.m^{-2}$) |
| E | Young's modulus ($Nm^{-2}$) |
| L | stand-winder distance (m) |
| $D_t$ | tension damping term ($ms^{-1}/Nm^{-2}$) |
| G | gauge out of stand (m) |
| W | width out of stand (m) |
| R | coil radius (m) |
| J | moment of inertia of coil and motor (kgm2) |
| $D_S$ | speed damping term ($Nm/rad.s^{-1}$) |
| ATQ | accelerating torque (Nm) |

The aim of the above model is to relate two linear speeds, the speed out of (or in to) the mill and the speed on to (or off) the winder. The difference between the two speeds integrated with respect to time gives $\Delta L$, which when converted to a strain and multiplied by Young's modulus gives the tension within the strip. This tension is then easily converted to a torque acting on the winder. This torque is only valid in one direction, ie the strip will not push the winder round, it will only act to oppose the winder drive. Thus, figure 2.6 requires the addition on a non-linear block before the summation to calculate ATQ. The mill operator will have set a desired tension in the strip, FTREF, there will therefore be a resultant torque attempting to prevent the winder from turning. Integration of the resulting acceleration leads to the rotational speed. The major problem with this model is deciding where damping is introduced in to the system. It was decided to introduce a damping term based on the tension in the strip as a result of the speed difference between winder and mill stand. As the model stands, the alteration in moment of inertia of each winder as strip passes through the mill is not taken in to account.

Figure 2.7 shows the response of the take-up winder peripheral speed to a step change in strip exit speed from the roll gap. The response is highly damped, giving no overshoot and has a rise time of approximately 75ms. Figure 2.8 shows the corresponding variation in strip tension. The strip becomes "slack" while the winder speeds up to maintain the required tension in strip. The strip becomes "slack" because the strip is leaving the roll gap much quicker than it is being coiled. The tension in the strip is recovered as the winder speed increases. It should be noted that the linear speed of the strip onto the winder is slightly greater than the speed out of the mill, this difference is needed to maintain the required tension. The damped response is highly desirable as any oscillation could alter the exit gauge by drawing the metal out of the roll gap.

Unfortunately, despite the large amount of effort expended, this model failed to improve on the performance of the previous attempt when combined with the other modules to give a complete stand model, still being very difficult to set up and stabilise. This was due to lack of understanding about the real winder control system. The ACSL code for the simplified winder model is listed in appendix A2.

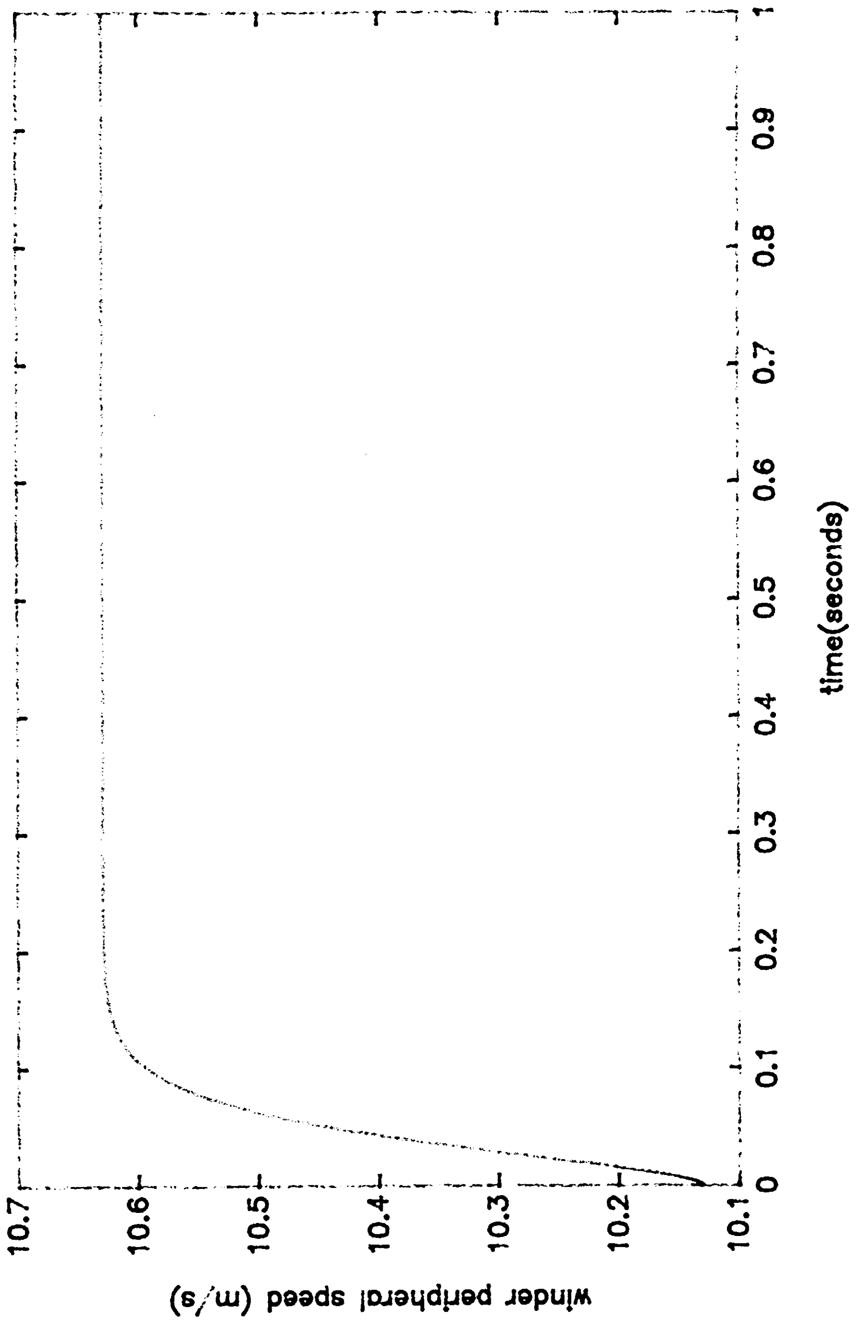


**Figure 2.7. Simplified winder model, speed step response.**

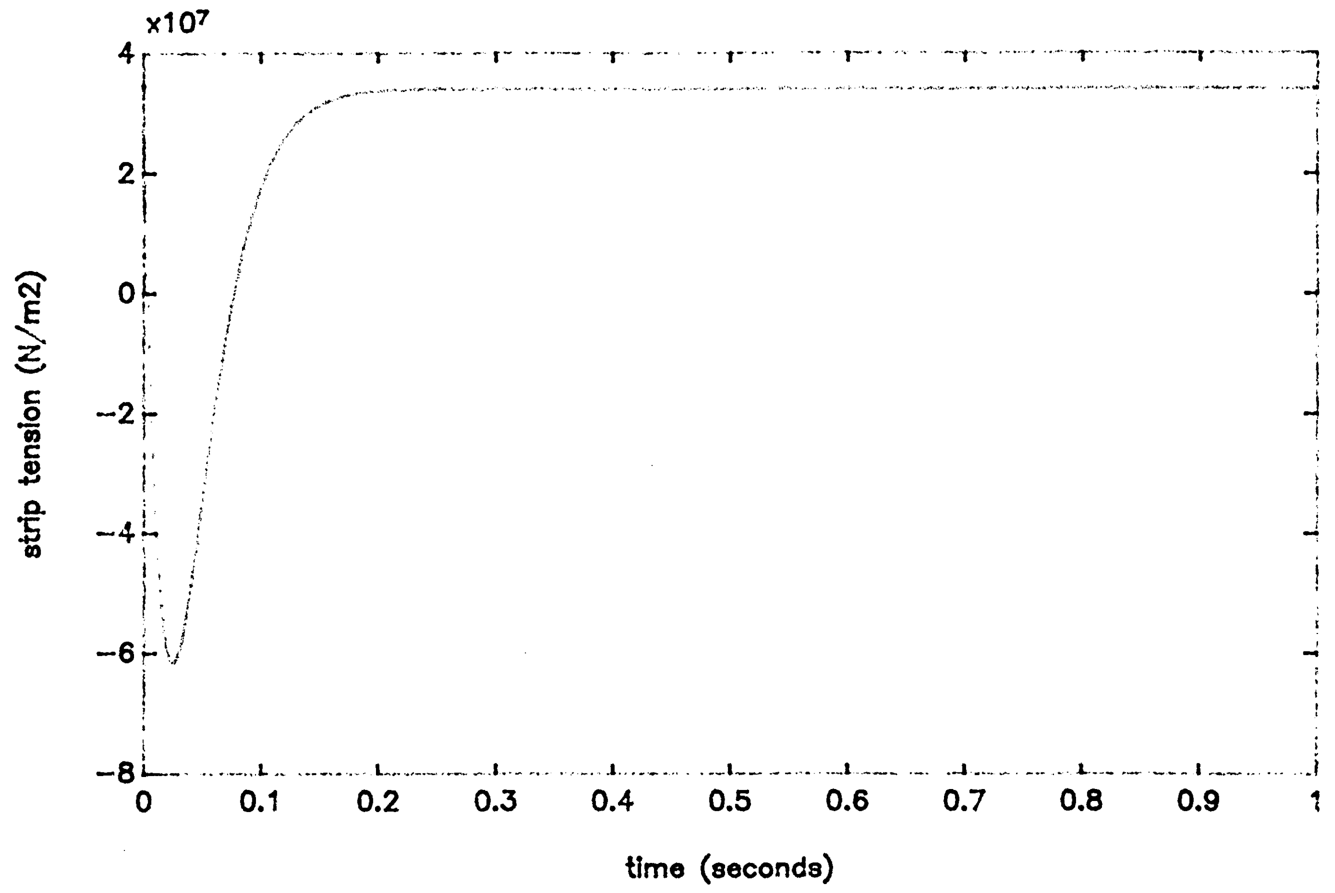


Figure 2.8, Simplified winder model, tension response.

## 2.1.2 Model structure

Although considerable simplifications have been made, the overall mill stand model remains fairly complex. Figure 2.9 attempts to show all the interactions taking place in the mill stand model by showing the individual modules and the interactions between them. In general the data flows from top to bottom. At the centre of the model is the actual roll gap where the strip gauge is reduced, the actual distance between the work rolls being determined by the servo valves in the stand model.

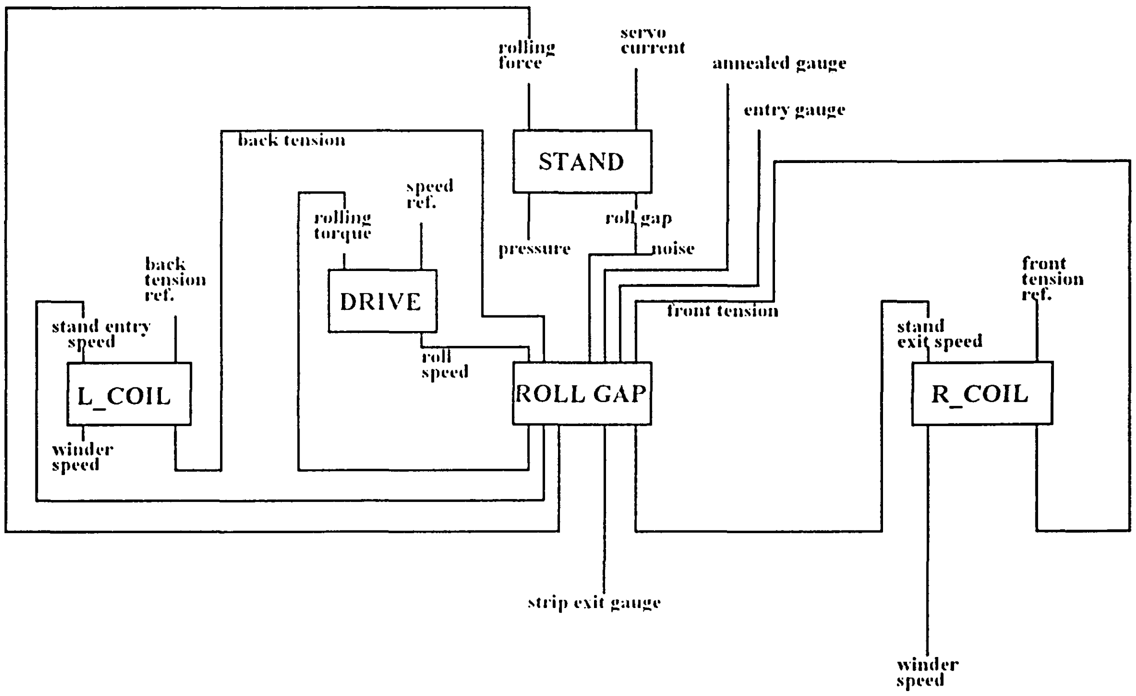


Figure 2.9. The structure of the complete mill stand model.

The stand module represents a mill stand with a bottom mounted capsule. The top rolls and the top of the mill housing are assumed to move together as one mass. As detailed above, the servo valve dynamics are represented as a second order transfer function. The model also takes in to account mill housing stretch and known non-linearities such as hysteresis in the capsule and limiting of the servo valve spool travel. L_COIL and R_COIL represent the left and right hand winder models respectively. Figure 2.9 is drawn for strip being rolled from left to right.

The roll gap module uses a FORTRAN subroutine to calculate rolling load, rolling torque, deformed roll radius and forward slip [24]. The annealed gauge is used to introduce a hardness variable into the model. The main output of this module is the strip exit gauge. As stated earlier, the major factor affecting winder speed is tension. The tension within the strip also has an effect on the exit gauge. If tensions are high, the strip is drawn plastically out of the mill, thus giving a thinner gauge than would be expected from the rolling load alone.

## 2.2 The MATLAB model

As mentioned above, the model required tailoring to a specific mill, so a data logging exercise was carried out (actual details in section 3.2.1, below). Due to difficulties experienced in identifying a model from the recorded data (see below), a model based on the block diagram in figure 2.1, was developed within MATLAB [25]. MATLAB stands for matrix laboratory and is a powerful interactive software package for numerical analysis. This simple model was later used heavily, in preference over the ACSL model, being quicker and easier to use for developing identification algorithms (section 3.3.2) and designing controllers (section 4.3). Once coded in MATLAB, it is possible to alter the form of the model eg from transfer function to state space form [26]. A number of block diagram transformations were required as the original model was found to be ill-conditioned. This problem arose from the difference in orders of magnitude between microns ($10^{-6}$m) and the forces within the model ($10^{9}$N).

Figure 2.10 shows the step response on the MATLAB model. The MATLAB code for this model, modbld.m, can be found in appendix B1. The model is seen to have a small delay of approximately 5ms and a rise time of 15ms. Obviously, the times will be affected by altering the natural frequency or damping ratio of the servo valve or any of the

MATLAB model step response

MATLAB model freqency response

6op OSDLJd

other parameters of the model. The frequency response of this model is given in figure 2.11, the frequency range of interest is 0-30Hz.

## 2.3 Conclusions

The motor drive components of the existing ACSL mill stand model have been simplified. Unfortunately, the attempt at improving the winder models was not as successful as had been hoped. During the course of the remaining work the MATLAB model became the preferred model for simulation providing a reasonable representation of the real plant. This model is much simpler and as a result easier and quicker to use.

# 3.0 DATA ACQUISITION AND SYSTEM IDENTIFICATION

As previously stated, a number of problems had been experienced in building a simulation model using physical insight alone. Further, the existing model, written in the ACSL software had been constructed via a number of different projects and as a result, did not represent any one particular mill. By carrying out a data acquisition and identification exercise on a real mill, the simulation model could be tailored to a specific mill and an attempt made to improve those aspects of the model known to be poor. In addition, the data would be required in the initial testing of any control system designed. Note 3, Appendix F.

## 3.1 System Identification Options

System identification is the process by which a mathematical model of a dynamic system may be derived from system input and output measurements. Application of the procedure requires three elements [27,28];

- the observed input / output data
- a set of possible model structures
- a selection criterion, (the identification method) based on the information recovered from the data.

The identification process amounts to repeatedly selecting a model structure, fitting the best model to the structure and then evaluating the model's properties to see if they are satisfactory.

Identification methods can be classified according to the type of model that results:

- non parametric methods, eg correlation analysis
- parametric methods, eg parameter estimation techniques.

### 3.1.1 Correlation Analysis

Correlation techniques are used in a wide variety of applications and give a measure of the similarity between two signals. Correlation analysis produces a non parametric model [29,30], to which a parametric model may be fitted [31,32].

Correlation analysis is used to determine an estimate of the impulse response of a linear system who's output is corrupted with noise. This technique requires that an external signal be input to the system, as the input during normal operation does not generally contain enough information. One of the most common signals used for this purpose is a Pseudo Random Binary Sequence (PRBS) [29,33,34,35]. PRBS signals are simple to generate, (see Appendices B2 and C1 for MATLAB and C [36] listings) and can easily be injected into a system. The signal intensity is low, with energy spread over a wide range of frequencies. As a result the PRBS signal appears as noise on the normal input signal and causes very little additional disturbance as long as its amplitude is carefully selected. Also refer to appendix B3 for the correlation function MATLAB code.

### 3.1.2 Parametric Estimation

Probably the most commonly used models in system identification are parametric models. Such structures are based on previous values of the output, current and previous values of the input and usually current and previous values of a disturbance signal (noise). The general parametric model structure is:

$$A(z^{-1})y(t) = \frac{B(z^{-1})}{F(z^{-1})}u(t\text{-}nk) + \frac{C(z^{-1})}{D(z^{-1})}e(t)$$

where:

| | |
|---|---|
| y(t) | output data sequence |
| u(t) | input data sequence |

continued...

$e(t)$ white noise sequence

$nk$ pure time delay of nk samples between input and output

$A(z^{-1})$ polynomial in the delay operator $z^{-1}$ of order na

$$A(z^{-1}) = 1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_{na} z^{-na}$$

$B(z^{-1})$, $C(z^{-1})$, $D(z^{-1})$, $F(z^{-1})$ are polynomials in the delay operator of order nb, nc, nd and nf respectively.

The simplest parametric model having the above form is the ARX (Auto Regressive with eXogenous input) [27,33] structure, where nc=nd=nf=0.

$$A(z^{-1})y(t) = B(z^{-1})u(t-nk) + e(t)$$

The main disadvantage of this very simple structure is the simplicity of the disturbance term which is assumed to be a white noise sequence. A slightly more complex model, the ARMAX (Auto Regressive Moving Average with eXogenous input) structure corresponds to nf=nd=0. Model structures of the above form are often referred to as "black box" models. For further discussion the reader is referred to [27,28,33].

The most useful class of methods for parameter estimation is that of Prediction Error Methods. The same input is applied to both the system and the model, the resulting outputs are compared giving rise to an error which is dependent on how poor the estimate of the model is. Thus, this error gives an indication of how close the model is to the actual system. This error is known as the prediction error or loss function, its magnitude and not the sign of the error is of greatest use. Therefore, the square of the error is used and this forms the basis of the majority of identification procedures.

Before commencing parameter estimation a model structure must be selected. There are two further choices to be made, the model order (number of parameters to be estimated) and the time delay. Parameter estimation is based on minimising the loss function, the difference between the actual output and the predicted output. It would therefore seem reasonable to determine the model structure from variations in the loss function caused by varying the model order and delay [31,32]. For a single value of delay, the loss function first decreases then remains approximately constant or increases slightly as the model order is increased. It is also possible to get an idea of the time delay from the estimated impulse response. Further details of model order selection can also be found in [37,38].

## 3.2 Data Acquisition

There were three main aims of the data acquisition exercise:

- obtain plant data during normal rolling
- enable the ACSL simulation model to be tailored to a specific mill and an attempt to answer a number of remaining questions concerning the adequacy and accuracy of the model.
- carry out PRBS tests on various loops of the control system

By analysing the available signals during normal rolling it was expected that it would be possible to determine if the signals contained enough information for on-line identification without having to inject an external signal [39].

### 3.2.1 Plant Data

The available mill was a Sendzimir, or 'Z' mill, and not of the 4-high configuration originally envisaged, figure 1.1 [24,40,41]. A 4-high mill consists of four horizontal rolls, one above the other, the direction of which may be reversed to pass the metal forwards and backwards. The 4-high configuration is a common special case of a simple 2-high mill in an attempt to lower work roll diameter (which allows greater reductions), whilst avoiding roll bending with the use of the two larger back up rolls. The Sendzimir mill is an extension to this idea. It uses a cluster of rolls of increasing diameter to support very thin work rolls, and is most often used for rolling exceptionally hard material. It was not anticipated that this would cause any problems as the differences in mill construction were not in the main drive, winder and gap control areas. Also, the mill was highly instrumented and all the required signals were available. The data recorded were the same for the majority of the test records and the details of these experiments can be found in appendix D.

The data were recorded using a Bakker Electronics 2570 Signal Analyser. However, before analysis, the data had to be transferred to an Hewlett Packard (HP) workstation. This step proved extremely time consuming as the two machines have very different operating systems. The majority of the data analysis was carried out using the MATLAB software.

Initial investigation of the data showed that some of the tests had been unsuccessful (in terms of the usefulness of their results), however, other results were encouraging. The delay and rise times of the estimated impulse response, obtained by correlation analysis, were as expected but, the amplitudes were low and the waveforms noisy. Thus, misleading results were obtained and the estimated impulse response of the system was generally poor. These problems were thought to be due to noise. However, the use of time averaging and maximum PRBS amplitude, without saturating the servo valve, yielded no improvements. The data sequences were normalised by the subtraction of the mean value from each sample to remove any offset and the division by the standard deviation. This improves numerical conditioning as the input and output signals are then of the same order of magnitude. After normalisation quantisation was seen to be a problem as the data was only utilising six levels out of the possible twelve of the Analogue to Digital Converter (ADC).

Little success was achieved at fitting simple parametric models to data obtained from testing the position loop. This loop was chosen for simplicity and because it had a more or less known response. The System Identification Toolbox for MATLAB [28] was used for these first identification tests. Due to their apparent failure a number of Least Squares algorithms [27,33,42,43] were developed as .m files [25] (.m files are simply text (ASCII) files that contain MATLAB statements. They are particularly useful for writing user defined functions for use within MATLAB. It is easiest to think of .m files as computer programs written within MATLAB). However, the simulated responses of the models remained poor. Indeed it was possible to fit a more successful model to the estimated impulse response by hand, figures 3.1 and 3.2.

Figure 3.1 shows an attempt to fit a fifth order ARMAX model to the estimated impulse response of the position loop. It was assumed that there was a delay between the input and output signals of approximately 10ms and so the first section of the curve was ignored. However, it was later found, section 3.4, that a delay just shifts the complete curve to the right so this assumption was incorrect. Figure 3.2 shows the same estimated impulse response with a smooth curve fitted to the data by hand. The curve is basically the sum of a phase shifted exponentially decaying cosine and an exponential. The equation is of the form

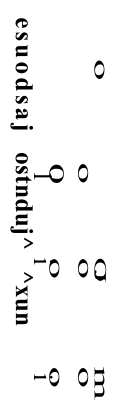
$$Ae^{-Bt}\cos(Ct + \phi) - De^{-Et}$$

The five parameters A,B,C,D and E were estimated from approximations made of the noisy waveform with the amplitude, A, and phase shift, $\phi$, chosen to give the best fit.

A number of tools were employed through MATLAB in an attempt to improve the results. Filtering of the data to remove any high frequency noise made little improvement nor did effectively re-sampling the data at a lower frequency.

The signals during normal operation were found not to contain enough information (sufficient frequencies within the range of interest) for identification. Therefore, an on-line implementation of the identification algorithms requires the injection of an external signal such as a PRBS.

fifth order arroax model {dBlay - 9mS) Position Loop



.046#exp (-.00265*t) .#cos (.0209#t+pi/3) - . 0225#exp (-.00259#t)

### 3.2.2 Test Rig

A fairly simple experimental test rig with known responses was also available, figure 3.3. The test rig consists of a stand housing with bottom mounted hydraulic cylinder with load and position loop AGC, full details can be found in appendix E. A number of the previous tests were repeated on this rig and the effects of altering the parameters of the PRBS were examined. Similar problems to those experienced with the actual mill occurred and the small changes (altering the period and reducing the clock frequency slightly) made to the PRBS had little effect. The problems experienced throughout the data acquisition exercise indicated problems with the data recorded as all the identification methods produced poor results. The tests were carried out at a pressure approximately equal to half the full working pressure (equivalent to a rolling load of 600 tonnes). This value was chosen to give the same response whether the load was moved up or allowed to fall.

## 3.3 Identification

Primarily as a result of the work detailed above, and after using the author's own parameter estimation routines (section 3.3.2), it became obvious that there was a major problem with the set-up of the PRBS tests.

### 3.3.1 Discussion

The bit interval of the PRBS should be short compared to the shortest feature of interest (based on the expected response), and its period should be longer than the system settling time [33]. The bit interval should be less than half the smallest time constant, preferably less than a quarter [30]. The system was expected to have an approximate response consisting of a delay of 10ms and a rise time of 10ms. The PRBS was set up

| | | |
|---|---|---|
| clock frequency, | $F_c$=3kHz | $d_t$=.3x10$^{-3}$ s |
| No. of bits, | p=1023 | |
| period, | $pd_t$= .34 s | |

Figure 3.3. The Daw test rig.

Therefore, the set up would appear to be reasonable, the aim being to record sufficient data points to obtain an approximate impulse response. However, it is required to have a flat power spectrum up to the highest frequency of interest [35], assumed to be $F_h$=30Hz. Assuming the power spectrum is required to be flat to within 1dB before falling away, this occurs at a frequency equivalent to .25$F_c$ [35].

$$\therefore 4F_h=F_c \qquad \Rightarrow \underline{F_c=120\text{Hz.}}$$

With a much slower clock frequency it is necessary to sample data at a greater rate in order to obtain enough data points to give a smooth response curve. If the system was linear the clock frequency being too high would be less important as a linear system would act a low pass filter and its output would look like the response to a PRBS with a lower effective clock frequency. The system was however, highly non-linear and, being plant data, the signals were contaminated with noise. Also, there were many other loops affecting the position loop, it was possible that the signal had not been injected into the inner most loop. As $F_c$ was too high the system could not respond. The PRBS was seen as general noise due to the non-linearities spoiling the required properties and the parameter estimation algorithms could not cope as they are generally only used to fit very simple noise models. Therefore, it had not been immediately obvious that there was a problem.

### 3.3.2 Algorithm Development

When trying to identify models from the mill data many problems were encountered. During the analysis a number of algorithms were written for use within MATLAB in an attempt to obtain reasonable models. By far the most common estimation technique [11,12] is that of least squares. A number of implementations of recursive least squares and recursive extended least squares [16,43] were developed and shown to work satisfactorily in simulation, appendices B4 and B5. The practical problem of the build up of round-off errors within the recursive estimator was overcome by using U-D factorisation [44], appendix B5.

In order to carry out Correlation analysis successfully it was found that a number of restrictions must be placed on the data record. The sampling frequency must be an integer multiple of the clock frequency. If this is not the case the correlation curve is seen to be noisy. Better results are obtained if the data record contains an integer number of PRBS periods.

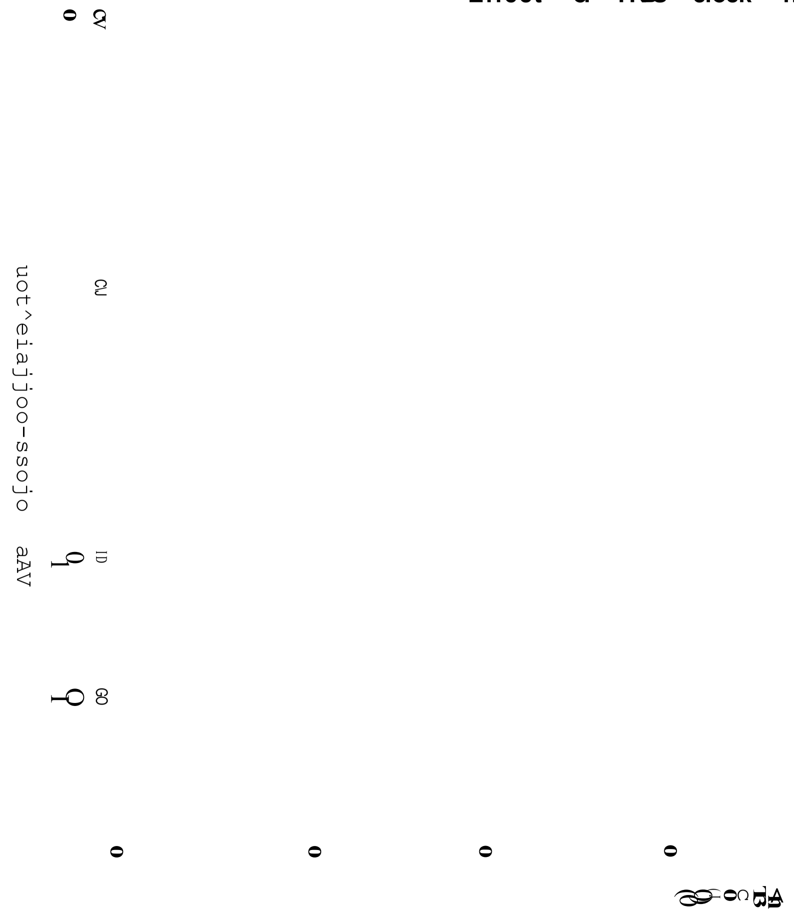
## 3.4 Discussion of findings

Due to the restrictive settings available on the PRBS generator only two reasonably valid clock frequencies were available, 100Hz and 300Hz. A number of tests were carried out on the test rig at these clock frequencies, varying the number of bits in the PRBS and the sample frequency. A much slower PRBS with a clock frequency of 30Hz was also investigated.

For these tests the original data logging equipment was unavailable and so a PC based system was used. The board used was an RTI-850, having 8 channels and a 16 bit ADC. This was used in conjunction with Snapshot storage scope software. The data were stored in an ASCII format which made transfer to the HP system much simpler. (The previous data were stored in DOS binary format and required conversion to HP binary then to ASCII before analysis.) The sampling rate was not always exactly that requested due to timing limitations within the ADC. Although there was a slight difference between sample frequency and the PRBS clock frequency, the amount of noise introduced to the correlation analysis was seen to be minimal.

In the first instance, correlation techniques were used to look at the test results. Figure 3.4 shows the effect of altering the clock frequency. As expected a clock frequency of 30Hz is too slow, there is little correlation between the two signals as the response is

**Effect of PRBS clock frequency**



over within a few samples. A clock frequency of 300Hz appears to be a little too fast and the correlation curve does not settle. As the clock frequency increases the output response amplitude decreases. The most useful result is achieved with a clock frequency of 100Hz. It was found that altering the length of the PRBS had little effect and that sampling at twice the clock frequency, ie 200Hz, was sufficient.

A number of further tests varying the PRBS amplitude were also carried out with a clock frequency of 100Hz, figure 3.5. The smallest amplitude, equivalent to 12μm, is basically lost in noise. For small steps much of the response will be lost due to friction. The largest amplitudes, 100μm and 200μm, show as expected, saturation of the servo valves. Within the range of the test, the most appropriate amplitude of 25μm give the best results.

The signals from the test using a 100Hz clock frequency and 200Hz sample frequency were taken and used for identification. It was found that the system could be represented by a very simple 5th order ARX model, figure 3.6, containing 10 parameters. This model only requires the simplest Least Squares algorithm as there is no noise model. More complex models were investigated but no significant improvement in the simulated responses were obtained. It is important to remember that the model structure dictates the number of parameters to be estimated and hence computation time. This structure is the same as that obtained from the simple mill model developed in MATLAB earlier, section 2.2.

The frequency response of the data and the model can be easily compared, figures 3.7 and 3.8. There is seen to be good correspondence over the frequency range of interest, 0-30Hz. Cross validation [45] was used successfully in testing the models. This technique involves using two different data sets, with the same sampling frequency. The first is used to identify the system and the second is then used to validate the model.

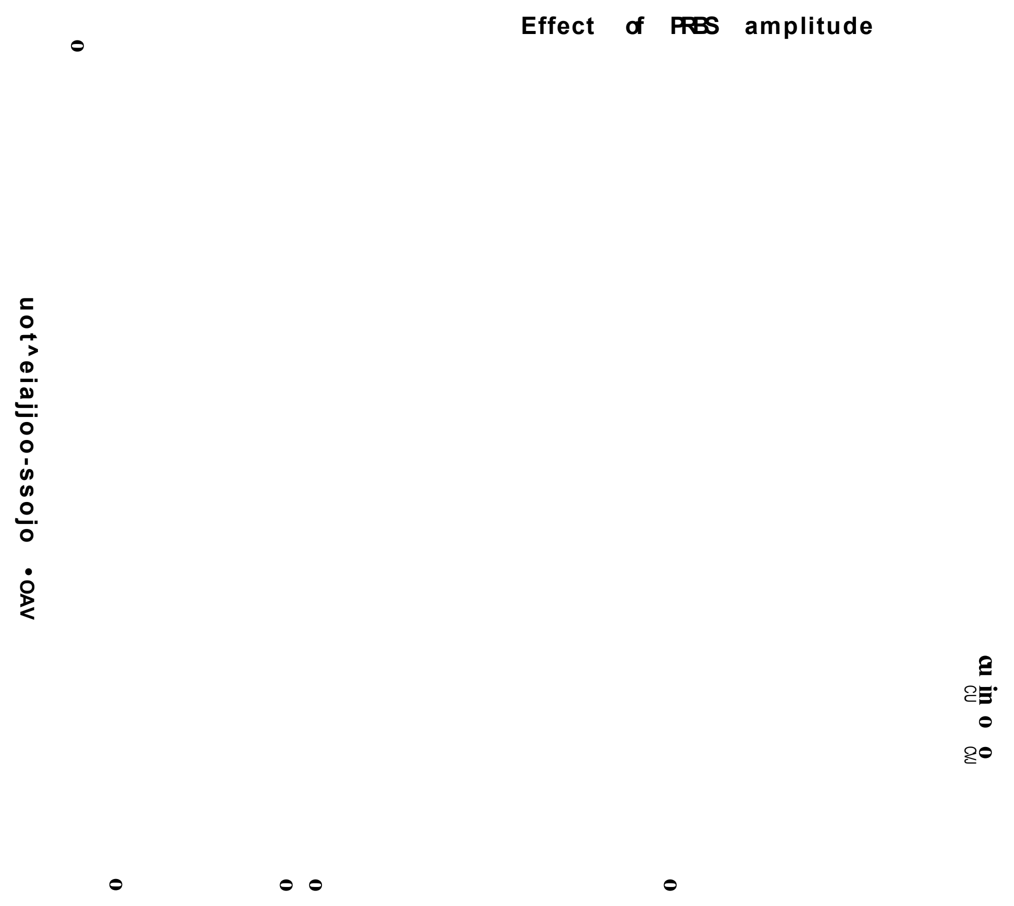


Figure 3.5, PRBS amplitude variations.

rnmpari »-d actual responses.

AMPLITUDE PLOT

-LLJJ-X 1—I__L I

**Figure 3.7. Identified model amplitude response.**

PHASE PLOT

**Figure 3.8. Identified model phase response.**

As in all digital control applications it is important to have proper conditioning of the signals [46]. Due to problems of aliasing it is necessary to eliminate all frequencies above the Nyquist frequency before sampling (the Nyquist frequency is defined as being equal to half the sampling frequency.). A number of methods are available when initialising the estimator, the simplest is to take the minimum number of samples required to calculate an estimate before actually starting the estimator. If employing this method it is good practice to allow the system to settle before starting to record samples. If the estimator has been used previously it could be initialised with the final settings from a previous run.

The data during normal rolling was seen to contain insufficient information and was thus unsuitable for identification purposes. This is the reason for using a PRBS. Estimator "blow up" or parameter divergence can occur if the input to the system is not sufficiently and persistently excited. The parameter estimator is the critical element of a self-tuner. If the model is not being improved the results should be discarded. The management of the estimator [11,12,47] is often called jacketing. In an extreme case this may consist of an "off" switch for an operator.

## 3.5 Conclusions

The AGC position loop can be successfully identified using recursive least squares but this requires the injection of a test signal. Such a signal is a PRBS with a clock frequency of 100Hz being sampled at 200Hz. If the sample frequency is not an integer multiple of the clock frequency the results from any correlation analysis will be noisy. The period of the PRBS was found to have less effect as long as it was greater than the system settling time. The results imply that there is only a small band of suitable clock frequencies. However, this is heavily influenced by the restrictive settings of the PRBS generator used for the tests. This could be a problem if the system was found to be so non-linear that its response altered the validity of the choice of clock frequency.

# 4.0 CONTROLLER DESIGN

Having successfully demonstrated that the position loop of the ACG system could be identified the next stage was to decide on the control strategy and develop the self-tuning controller. Note 4, Appendix F.

## 4.1 Background

As stated in section 1.2, there are a number of options when deciding on the form of an adaptive controller. The majority of papers published are of a theoretical nature and suggest a large number of applications. The adaptive systems that have been implemented are at least as good as the original control scheme. For the most part, applications have been to fairly slow systems, in for example, the chemical industry [48,49] and ship steering [50], in such cases the sampling times are of the order of minutes. The majority of papers do not give specific details but only note the techniques used. No detailed examples of adaptive systems applied to gauge control in the rolling of metals were found and the number of applications to any kind of "fast" plant is very limited.

By far the most popular estimation techniques are those based on least squares. As stated earlier a number of design methods can be catered for. Examples of pole-placement [48,49], minimum variance [50,51,52] and the use of multivariable ideas [53,54,55] have been published. Suggested general texts are [14,15,16].

## 4.2 Options

Intuitively, a "one-shot" self-tuning controller able to tune on request, applied to the position or load control loop of the hydraulic cylinders would be the simplest scheme to adopt initially. Such a system could be later extended to allow continuous tuning (adaptive control) and be applied to other loops. The time constraint will require a very simplified model in order to minimise the number of parameters requiring estimation. If the computing time is found to be too great it should be possible to use a batch (off-line) algorithm. Such algorithms process all the observations simultaneously and produce a single estimate of the parameters. Such a system would work by recording a number of measurements in real time and then calculate the controller parameters, for example every second.

Minimum variance [16,56] could be used to minimise exit strip gauge variations. However this requires the minimisation of a complicated cost function which would require a large amount of computing time. Also, minimum variance does not work well when the set point is being varied, which is the case when considering the reference to the position loop which is altered constantly by other loops.

The other main option is that of pole-placement [15,16] which is simpler. A further advantage is that the desired response can be specified in terms of a second order transient response. Also, this method allows for combined regulation and servo control. The disadvantage of this method is that excessive control signals may be produced as the controller tries to drive the system too hard in an attempt to meet ambitious control objectives.

## 4.3 Pole Placement Design

The pole placement design method was chosen primarily because of its relative simplicity and the fact that the desired response can be specified in terms of a second order system [16]. Note 5, Appendix F.

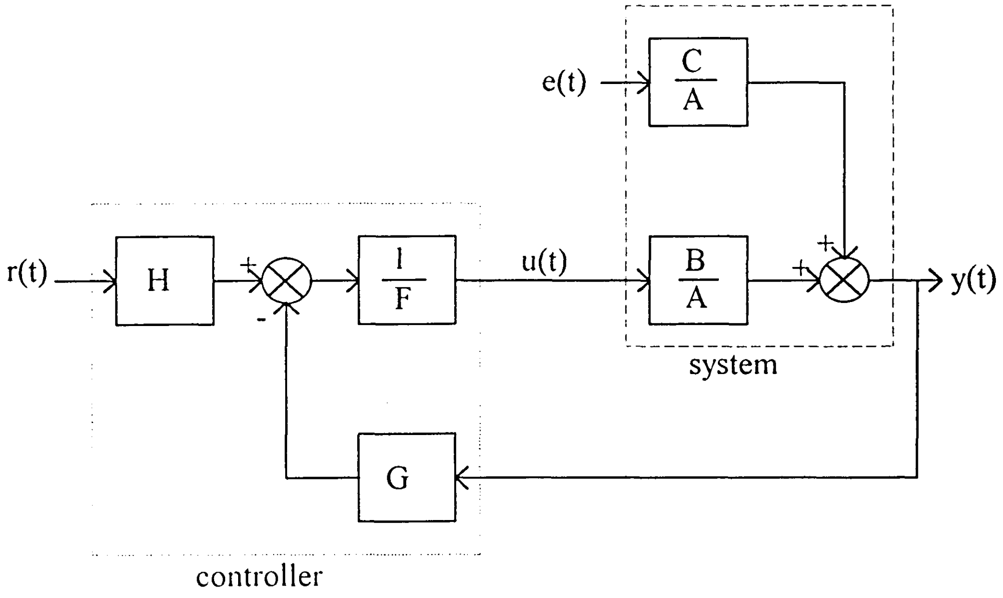


**Figure 4.1 Pole Placement Self-Tuning System.**

The system output y(t) is required to follow the reference r(t) rejecting random disturbances, according to a specified design rule. The problem therefore is one of regulation and servo following. Consider the system [16], figure 4.1, defined by

$$\begin{aligned} Ay(t) &= Bu(t-1) + Ce(t) \\ &= Bz^{-1}u(t) + Ce(t) \end{aligned}$$

this model assumes a delay of one sample between system input and output. With controller

$$Fu(t) = Hr(t) - Gy(t)$$

where:

$F = 1 + f_1z^{-1} + \dots + f_{nf}z^{-nf}$ $\quad A = 1 + a_1z^{-1} + \dots + a_{na}z^{-na}$

$G = g_0 + g_1z^{-1} + \dots + g_{ng}z^{-ng}$ $\quad B = b_0 + b_1z^{-1} + \dots + b_{nb}z^{-nb}$

$H = h_0 + h_1z^{-1} + \dots + h_{nh}z^{-nh}$ $\quad C = 1 + c_1z^{-1} + \dots + c_{nc}z^{-nc}$

the closed loop equation is then

$$y(t)(AF + BGz^{-1}) = BHz^{-1}r(t) + FCe(t)$$

The required closed loop poles are then specified by $T = 1 + t_1z^{-1} + \dots + t_{nt}z^{-nt}$ and the controller parameters in F and G are calculated according to the polynomial identity

$$AF + BGz^{-1} = TC$$

The polynomial T specifies the desired poles. These poles may be expressed in terms of complex conjugate pairs, being roots of T. For a unique solution the degrees nf and ng should be selected as

$$nf = nb$$

$$ng = na-1 \quad (na \neq 0)$$

inserting in the system equation

$$y(t) = \frac{HB}{TC} r(t-1) + \frac{F}{T} e(t)$$

note that the noise polynomial has been cancelled in the disturbance term. The precompensator H is selected to achieve low frequency gain matching. In the simplest form H is chosen so that C is cancelled from the pole set, ie:

$$H = C\left[\frac{T}{B}\right]_{z=1}$$

Therefore, the closed loop equation is

$$\underline{y(t) = \left[\frac{T}{B}\right]_{z=1} \frac{B}{T} r(t-1) + \frac{F}{T} e(t)}$$

### 4.3.1 Simulations

Having successfully identified a model from the test rig data the MATLAB code for a pole placement controller was developed, appendix B6. The desired response was specified in terms of a second order system, ie natural frequency and damping ratio. Figure 4.2 shows the control signal u(t) and system output y(t) from the identified model of section 3.4, in response to a 4Hz square wave reference. Initially, the controller is switched off with only the identification algorithm in operation. With no controller in operation, the control signal (system input) is the 4Hz reference signal. After 0.5 seconds the model has been successfully identified and the controller is switched on, a change in the control signal is clearly observed. After initial transients, the response is as required.

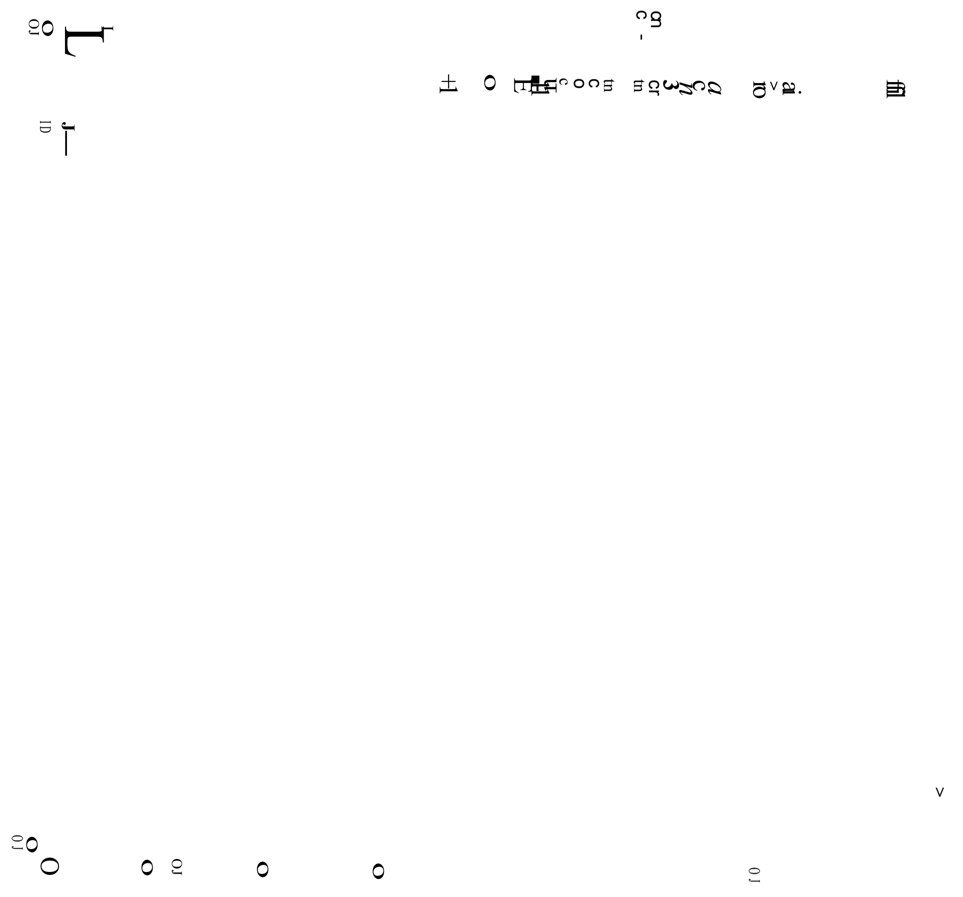


Figure 4.2, Self-tuning controller response.

This value of 0.5 seconds was chosen arbitrarily to demonstrate the controller principle. The identification algorithm can successfully construct a usable model in less than half this time. Note 6, Appendix F.

As more unrealistic demands are put on the system, in terms of harsher desired responses, the amplitude of the control signal becomes excessive. In a real system there is a danger that the controller will try and drive the system too hard. It is therefore important to monitor the control signal and take care when specifying the desired response. Other implementation aspects are considered later in section 4.4.

### 4.3.2 Location of the other poles

In specifying the desired response in terms of a second order system the control algorithm is placing two of the closed loop poles in the required positions. The control system toolbox [26] was used to investigate the location of the other poles. The pole-placement algorithm moves the dominant poles from their open loop positions, figure 4.3, to the required closed loop positions, figure 4.4, and the other poles are moved so as to have no visible effect on the system response. In the z-plane this involves moving the remaining poles to the neighbourhood of the origin. (In the s-plane the other poles are moved far to the left and are grouped very close to the negative real axis).

The closed loop poles are detailed in table 4.1, below. The desired second order system response corresponding to a natural frequency of 200 rad.s$^{-1}$ with a damping ratio of .7, are clearly seen as the dominant poles.

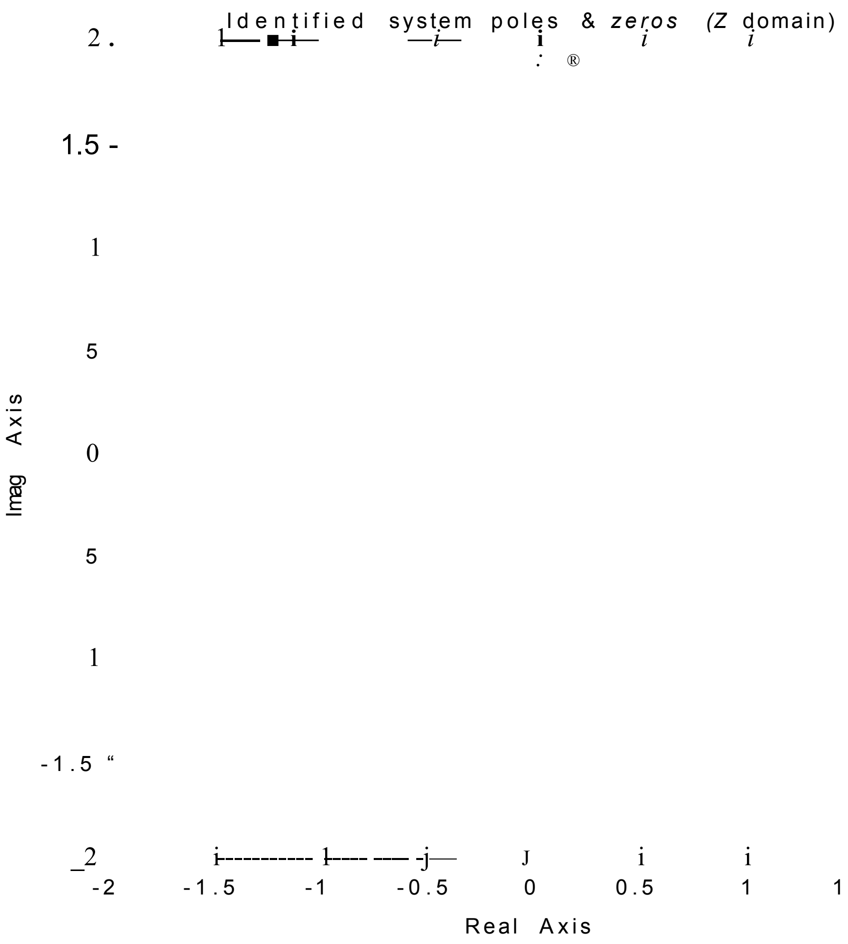


**Figure 4.3. Open loop system poles and zeros.**

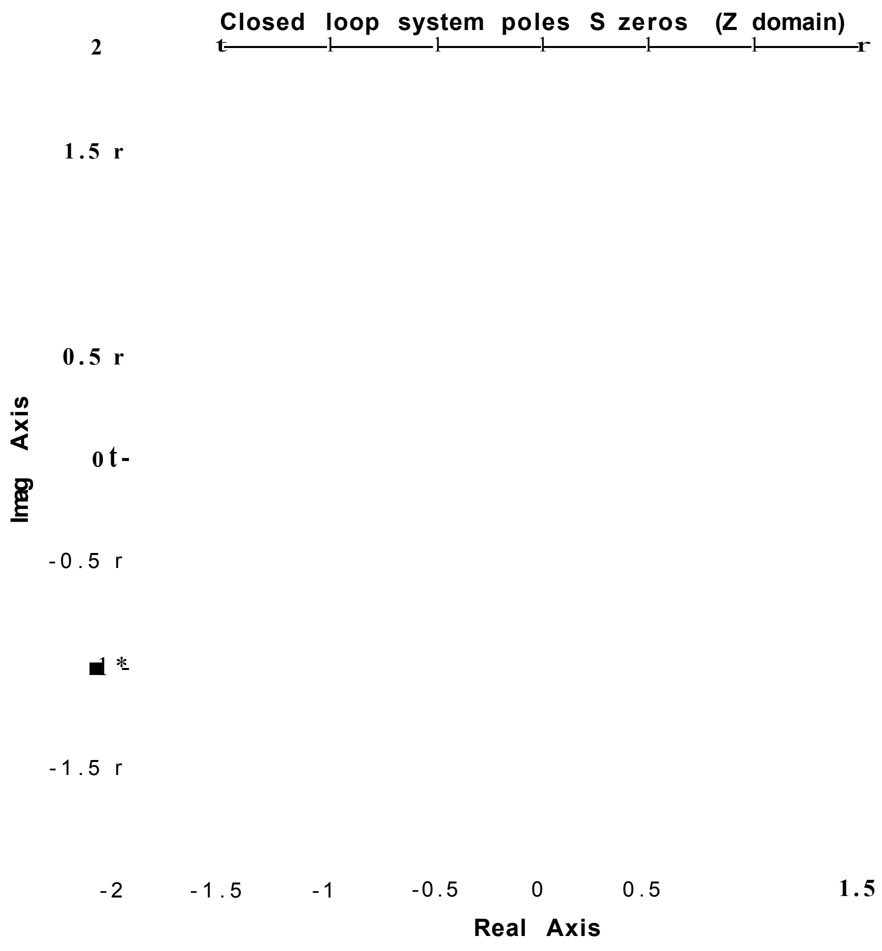


**Figure 4.4. Closed loop system poles and zeros.**

| $\omega_n$ rad.s$^{-1}$ | $\zeta$ |
|---|---|
| 200.000 | .700 |
| 200.000 | .700 |
| 670.948 | .996 |
| 670.948 | .996 |
| 691.475 | .968 |
| 691.475 | .968 |
| 730.175 | .919 |
| 730.175 | .919 |
| 783.530 | .858 |
| 783.530 | .858 |
| 848.140 | .794 |
| 848.140 | .794 |
| 921.361 | .731 |

Table 4.1, Closed loop poles.

## 4.4 Discussion

There is the possibility of ill-conditioning within the estimated model. It is highly unlikely that an exact pole / zero cancellation will occur within the identified model. This problem is overcome by using Kucera's algorithm [16,57] before controller synthesis, appendix B7. As a result of applying Kucera's algorithm the minimal realisation of the system is achieved by the detection of any approximate pole / zero cancellation. The decision on how approximate the cancellation should be requires further experimentation with the algorithm and is expressed in terms of a tolerance. This algorithm has also been developed and incorporated into the MATLAB version of the controller, appendix B8.

There are a number of factors to consider in the implementation of such a controller [11,12,46]. When initialising the controller it is possible to set it up with proportional or proportional plus integral gain. If the system has been controlled before, the previous values of the controller parameters can be used. Before allowing the controller loose on the system its output can be compared to that of the existing controller, thus, avoiding any problems due to excessive control signals. Although the possibility of this problem occurring should be minimised through simulation studies carried out at the design stage. The implementation aspects of the identification algorithms, section 3.4, must also be considered. These requirements are just as important as the controller will only ever be as good as the model it utilises to control the process.

## 4.5 Conclusions

A pole placement  self-tuning controller using the model successfully identified previously in chapter 3, has been shown through simulation, to be possible. This controller is based on a second order system response of 200 rad.$s^{-1}$ natural frequency and with a damping ratio of .7.

# 5.0 CONCLUSIONS AND RECOMMENDATIONS

## 5.1 Conclusions

It has been demonstrated that a relatively simple, fifth order parametric model, to represent the dynamics of the position loop of an AGC system can be identified. This procedure requires the injection of a PRBS signal into the loop, as the input during normal operating conditions does contain sufficient information. A recursive least squares estimation algorithm can then used to calculate the parameters of the model.

In order to use the PRBS successfully, it was discovered that a good deal of information about the system response must be known before identification can begin. This data is required to enable the correct choices to be made about the sampling frequency and parameters of the PRBS. Incorrect choices can lead to misleading results.

Through computer simulations it has been further shown that the system identified above (the position loop of an AGC) can be successfully controlled by a pole-placement self-tuning controller. This controller was to chosen to have a desired response based on a second order system having a natural frequency of 200 rad.s$^{-1}$ and a damping ratio of .7.

## 5.2 Recommendations for further work

As stated above, the self-tuning controller has been shown to work in simulation. Obviously, the next major stage in the project will be to develop a prototype system. The development of the C code has already started and awaits completion and implementation by Davy personnel. Although the fundamental code is relatively straight forward, (this basically requires the translation of the MATLAB code already developed) the real-time and supervisory aspects still require careful examination.

A number of the practical issues concerning the implementation of the proposed controller have already been discussed in sections 3.4 and 4.4, and suggestions made on the handling of these aspects. The constraints of real time computing may mean that the estimation algorithms cannot be used at every sample interval. If this is the case, some form of off-line (or batch processing) of the least squares algorithm will be required, that for example, stores and then processes the data every second.

The magnitude of the control signal applied to the system must be monitored at all times, as problems could easily occur due to saturation, and abrupt changes in this signal could cause the metal strip to break. This potential problem must also be considered when selecting the desired closed loop response. Although, this should be carried out in simulation at the design stage, prior to the testing the controller.

The proposed controller is based on a self-tuning assumption that the system has unknown but constant parameters. Once the controller is achieving satisfactory control, the estimator algorithms can be "switched off". It should however be possible to monitor performance and restart the self-tuning procedure at some later stage as required. In any extension to adaptive control, provision must be made for parameter variations. With the system in continuous operation, it is essential [58] to monitor the performance of the estimator. It is vital that the estimator can react quickly to sudden changes, but does not allow the parameter estimations to diverge.

The continued development of the self-tuning controller, as outlined above, will require further experimentation and comparison with the current control systems at all stages.

## 6.0 REFERENCES

1 Pemberton T.J., PID: The logical control algorithm II, Control Engineering, p61-3, July 1972.

2 Dorf R.C., Modern control systems, 6th Ed., Addison Wesley, 1992.

3 Deobelin E.O., Control system principles and design, Wiley, 1985.

4 Ziegler J.G., Nichols N.B., Optimum settings for automatic controllers, TRANS. ASME, p750-68, 1942

5 Pemberton T.J., PID: The logical control algorithm, Control Engineering, p66-7, May 1972.

6 Bennett S., Real-time computer control: An Introduction, Prentice-Hall,1988.

7 SIRA Research and Development Division, A study of adaptive and self-tuning control, Sira Ltd., 1986.

8 Astrom K.J., Wittenmark B., Computer controlled systems: Theory and design, Prentice-Hall, 1984.

9 Warwick K., Rees D., (Eds.), Industrial digital control systems, IEE Control Engineering Series 29, Peter Peregrinus, 1986.

10 Bennett S., Linkens D.A., (Eds.), Computer control of industrial processes, IEE Control engineering series 21, Peter Peregrinus, 1982.

11 Astrom K.J., Theory and applications of adaptive control - A survey, Automatica, vol 19, p471-86, 1983.

12 Wellstead P.E., Zanker P., Techniques of self-tuning, Optimal control applications and methods, vol 3, p305-22, 1982.

13 Clarke D.W., Gawthrop P.J., Self-tuning control, IEE proceedings, vol 126, No.6, p633-40, 1979.

14 Harris C.J., Billings S.A., (Eds.), Self-tuning and adaptive control: Theory and applications, IEE Control Engineering Series 15, Peter Peregrinus, 1981.

15 Astrom K.J., Wittenmark B., Adaptive control, Addison Wesley, 1989.

16 Wellstead P.E., Zarrop M.B., Self-tuning systems; Control and signal processing, Wiley, 1991.

17 Landau Y.D., Adaptive Control: The model reference approach, Marcel Dekker, 1979.

18 Plackett R.L., Some theorems in least squares, Biometrika, vol 37, 149-57, 1950.

19 Young P.C., Recursive estimation and time series analysis, Springer Verlag, 1984.

20 Jackman R., Gronbech R.W., Forster G.A., The position-controlled hydraulic mill, from - Hydraulic Control of Rolling Mills and Forging Plant, The Iron and Steel Institute, 1972.

21 Young J.A., Hydraulic automatic gauge control (AGC), Institute of Metals book 354, Advances in Cold Rolling Technology, 1985.

22 Bennett S., Linkens D.A., (Eds.), Real-time computer control, IEE control engineering series 24, Peter Peregrinus, 1984.

23 Advanced Continuous Simulation Language (ACSL), Reference manual, Mitchell & Gauthier Associates, Mass. USA.

24 Ginzburg V.B., Steel-rolling technology, Marcel Dekker, 1989.

25 Moler C., Little J., Bangert S., ProMATLAB - User's guide, The Mathworks, Natick, Ma. USA.

26 Control systems toolbox user's guide, The Mathworks, Natick, Ma. USA.

27 Ljung L., System identification - Theory for the user, Prentice-Hall, 1987.

28 Ljung L., System identification toolbox user's guide, The Mathworks, Natick, Ma. USA, 1991.

29 Godfrey K.R., Correlation methods, Automatica, Vol. 16, p527-34, 1980.

30 Godfrey K.R., The theory of the correlation method of dynamic analysis and its application to industrial processes and nuclear power plant, Measurement and Control, Vol. 2, May 1969, pT65-T72.

31 Baur U., Isermann R., On-line identification of a heat exchanger with a process computer - A case study, Automatica, Vol. 13, p487-96, 1977.

32 Isermann R., Baur U., Two step process identification with correlation analysis and least squares parameter estimation, Journal of Dynamic Measurement and Control, ASME 96G(4), p426-32, 1974.

33 Norton J.P., Introduction to identification, Academic Press, 1984.

34 Graupe D., Identification of systems, Van Nostrand, 1972.

35 Solartron JM1861, Pseudo random signal generator, Technical Manual, Solartron Electronics, 1971.

36 Harbison S.P., Steele Jr G.L., C: a reference manual, Prentice Hall 1987.

37 Van den Boom A.J.W., Van den Enden A.W.M., The determination of the orders of process and noise dynamics, Automatica, Vol. 10, p245-56, 1974.

38 Unbehauen H., Gohring B., Tests for determining model order in parameter estimation, Automatica, Vol. 10, p233-44, 1974.

39 Soderstrom T., Stoica P., System identification, Prentice-Hall, 1989.

40 Cottrel A., An introduction to metallurgy, Arnold, 1975.

41 Harris J.N., Mechanical working of metals, Pergamon Press, 1983.

42 Isermann R., Baur R., Bamberger W., Kneppo P., Siebert H., Comparison of six on-line identification and parameter estimation methods, Automatica, vol. 10, p81-103, 1974.

43 Eykhoff P., System identification: Parameter and state estimation, Wiley, 1974.

44 Bierman G.J., Factorisation methods for discrete sequential estimation, Academic Press, 1977.

45 Isermann R., Practical aspects of process identification, Automatica, vol. 16, p575-87, 1980.

46 Wittenmark B., Astrom A.J., Practical issues in the implementation of self-tuning control, Automatica, vol. 20, no 5, p595-605, 1984.

47 Warwick K., Implementation of self-tuning controllers, IEE Control Engineering Series 35, Peter Peregrinus, 1988.

48 Najim K., Youlal H., Regularised pole placement adaptive control of a liquid-liquid extraction column, International journal of systems science, v21, n7, p1313-23, 1991.

49 Najim K., Youlal H., Irving E., Najim M., Linear quadratic self-tuning of a liquid-liquid extraction column, optimal control applications and methods, v9, n3, p273-84, 1988.

50 Jia X.L., Leng D.Q., Yang C.E., Hou J.M., Song Q., Implementation of an adaptive autopilot unit, IFAC Adaptive systems in control and signal processing, Glasgow, 1989.

51 Dash P.K., Gupta A.P., New methods for improvement of dynamic performance of wind turbine generators, Electric machines and power systems, v11, n6, p485-98, 1986.

52 Oliveira A.L., Coito F.V., Silveira L.M., Lemos J.M., Marques J.S., A microprocessor implementation of a self-tuning controller, IFAC Adaptive systems in control and signal processing, Lund, Sweden, 1986.

53 Tham M.T., Vagi F., Morris A.J., Wood R.K., Multivariable and multirate self-tuning control: A case study, IEE proceedings D, v138, n1, 1991.

54 De Savio F., Ricci M., Visioli A., Multivariable adaptive control: An industrial drying process application, IFAC Adaptive systems in control and signal processing, Glasgow, 1989.

55 Omatu S., Yamamoto T., Hotta M., Adaptive temperature control of a water bath by using multivariable self-tuning control, IFAC Adaptive systems in control and signal processing, Glasgow, 1989.

56 Wellstead P.E., Edmunds J.M., Prager D., Zanker P., Self-tuning pole/zero assignment regulators, International journal of control, vol 30, no1, p1-26, 1979.

57 Kucera V., Discrete linear control: The polynomial equation approach, Czechoslovak Academy of sciences, 1979.

58 Lingarkar R., Lui L., Elbestani M.A., Sinha N.K., Knowledge-based computer control in manufacturing systems: A case study, IEEE Transactions on systems, man, and cybernetics, vol 20, no 3, 1990.

# APPENDIX A - ACSL code

This appendix lists the ACSL code for the simplified drive macro, SDRIVE, and for the simplified winder macro, SCOILR.

## A1 Simplified drive macro

```
macro sdrive(rv,rvref,rtq,id)

"This macro represents mill drive system"
"this is basically a 2nd order model with a motor torque"
"limit dependent on mill speed"

"in   rvref - ref. speed (rad/sec)"
"     rtq   - rolling torque (Nm)"

"out  rv    - mill speed (rad/sec)"

"id is an identifier eg F1"
"used so macro can be call a number of times"
"with locally generated variables"

"macro redefine tq,err,erri,errk,atq,tqmax"
"ensures these variables are unique to this macro call"


"-----parameters-----"

constant jm_id    = 19000      $"m of i of system (kgm2)"
constant bm_id    = 100        $"damp coeff 5% of max torq"
constant krv_id   = 275000     $"torque-speed gain"
constant krvi_id  = 27500      $"integral term"
constant tmax_id  = 450000     $"max torque (Nm)"
constant tgrd_id  = -6700      $"torque-speed gradient"
constant tcm_id   = .05        $"a time constant"
constant vcut_id  = 5          $"torque cut off speed"


"-----initial conditions-----"

constant rvi_id   = 5          $"ic for roll speed"
constant mvi_id   = 500        $"ic for motor speed"
                                "times damping"
constant inti_id  = 0          $"ic for integral term"
constant tqi_id   = 120500     $"ic for torque"
constant errk     = 0

"-----the model-----"

procedural
"cos speed both an input and calculated output"

"torque varies with speed"
tqmax = tmax_id
if (rv .le. vcut_id) go to label1
```

```
    tqmax = tgrd_id * rv + 483500

label1..continue

err = rvref - rv

"check for integral limit"
erri = err
if (errk**2 .le. tqmax**2) go to label2
    if (err*errk .le. 0) go to label2
        erri = 0

label2..continue

errk = krv_id*err + krvi_id*integ(erri,inti_id)

tq = bound(-tqmax,tqmax,errk)

atq = realpl(tcm_id,tq,tqi_id) - rtq
    "available torque"

rv = realpl(jm_id/bm_id,atq,mvi_id) / bm_id
   "speed"



end   $"of procedural"

macro end   $"sdrive"
```

**A2 Simplified winder macro**

```
macro scoilr(von,rstres,vout,ftref)

"version  : 1.0"
"date     : 7-8-91"
"author   : cng"
"filename : scoilr_1.0"

"this macro represents a coiler system RH or takeup"
"basically 2nd order with damping coming from"
"rotational speed of coiler and strip tension"

"in   vout   - speed out of stand (ms-1)"
"     ftref  - front tension ref force (N)"

"out  von    - coiler peripheral speed (ms-1)"
"     rstres - strip tension (N/m2)"

"-----parameters-----"

constant re     = 2.1E+10  $"E / stand-coiler dist"
constant rgauge = .002     $"gauge out of stand (m)"
constant rwidth = 1.27     $"width out of stand (m)"
constant rdamps = 200      $"speed damping term"
constant rmandr = .36       $"mandrel radius (m)"
constant rdampt = 3.8E-09   $"tension damping term"
constant rrad   = .4        $"coil radius (m)"
constant rinrti = 5000      $"m of i RH motor+coiler (kgm2)"

"set at end of initial section"
"rspdi = vout / rrad"
"rdli  = (ftref*rrad-rspdi*rdamps)/(
rrad*rgauge*rwidth*re)"


"-----the model-----"

procedural

rtfbk = rdampt * rstres

rstres = re * integ(von-rtfbk-vout,rdli)

rforce = rstres * rgauge * rwidth

rmtq = ftref * rrad

rtorq = rmtq - rforce*rrad - rdamps*rspeed

rinrt = rinrti + rwidth*8000*(rrad**4-rmandr**4)

rspeed = integ(rtorq/rinrt,rspdi)

von = rspeed * rrad


end  $"of procedural"

macro end  $"scoilr"
```

# APPENDIX B - MATLAB code

This appendix contains listings of useful .m files.

## B1 Code for building the model

```
% modbld.m
% define constants
kgap=25;                         % original 50;
B=1.4e9;
kmill=4.5e8;
area=.58;
v=.04*area;
dd=5e6;
m=1e5;

wn=60*2*pi;                     % original 60;
damp=1.1;

% define blocks
n1=kgap;d1=1;
n2=wn^2;d2=[1 2*damp*wn wn^2];
n3=1;d3=[1 0];
n4=B*area;d4=[v*m dd*v v*kmill+B*area^2];
nblocks=4;
blkbuild
disp(' ')
disp('finished blkbuild')
% interconnections
q=[
1 -4 0 0
2  1 0 0
3  2 0 0
4  3 0 0];
iu=[1];             % input block
iy=[4];             % output block
% interconnect to create SS model
[ac bc cc dc]=connect(a,b,c,d,q,iu,iy);
% minimise states
[acm bcm ccm dcm]=minreal(ac,bc,cc,dc);
[num,den] = ss2tf(acm,bcm,ccm,dcm,1);
[ad,bd] = c2d(acm,bcm,1/200);
[numd dend] = ss2tf(ad,bd,ccm,dcm,1);
b = numd; b(1) = [];
a = dend; a(1) = [];
c = 0;
```

## B2 PRBS generator

```
function p=prbs(n,h)
%
% p = prbs(n,h)
%
% generates a prbs sequence of length 2^n -1
% h = no. of samples each prbs level held for
% if want a prbs sampled at twice the clock frequency h=2
%
% n=2,3,4,5,6,7,9,10,11.

% feedback connections from - Godfrey KR, Correlation
Methods,
% Automatica 16 (1980)

% Chris Groves 5-3-92

if n<2 | n>11 | n==8, error('Must use 2,3,4,5,6,7,9,10,11
register stages'),end
if nargin==1, h=1;end
if abs(h-fix(h)) > eps, error('h must be integer'),end

reg=ones(n); p=[];fbki=0;hld=ones(h,1);

if n==2, fbki=1; end
if n==3, fbki=2; end
if n==4 | n==5, fbki=3; end
if n==6 | n==9, fbki=5; end
if n==7, fbki=4; end
if n==10, fbki=7; end
if n==11, fbki=9; end
if fbki==0, break, end

for period=1:2^n-1
     v=reg(n);
     if v==0, v=-1; end
     p=[p;v*hld];
     fbk=eor(reg(n),reg(fbki));
     for shift=n-1:-1:1
          reg(shift+1)=reg(shift);
     end
     reg(1)=fbk;
end
```

## <u>**B3 Correlation function**</u>

```
function a=corr(u,y)
%
% a = corr(u,y)
% u&y nx1 vectors
% calculates cross correlation between u (input) and y
(output)
% y=u for auto correlation.


nu=length(u);
for k=1:nu
     if k==1
          ushift=u;
     else
```

```
          ushift=[ushift(nu);ushift(1:nu-1)];
     end
     a(k)=sum(y.*ushift)/nu;
end
a=a';
```

## B4 Single step extended least squares

```
function[theta,error,p] =
ssels(y,u,err,N,na,nb,nc,nk,oldp,oldtheta,l)
%
%single step extended least squares
%
n = na+nb+nc;
x = zeros(n,1);
for ky=1:na, x(ky)=-y(N-ky);end
for ku=1:nb, x(na+ku)=u(N-ku-nk+1);end
for kc=1:nc, x(na+nb+kc)=err(N-kc);end
err1 = y(N) - x'*oldtheta;
p = oldp/l*(eye(n)-(x*x'*oldp/(1+x'*oldp*x)));
theta = oldtheta + p*x*err1;
error = zeros(n,1);
error(1) = err1;
```

## B5 Comparison of matrix inversion algortihms and UD factorisation

```
%   test2.m
% compare two matrix inversion algorithms and
% UD factorisation
% z = [y u] the output input data in column vectors
% nn = [na nb nc nk]
% more tests required - not very robust - B E W A R E !!!

[Ncap,nz] = size(z); nu = nz-1;
if nu>1, error('only one input allowed!!!!'),return,end
na = nn(1); nb = nn(2); nc = nn(3); nk = nn(4);
n = na + nb + nc;
if n == 0, return, end

k = max([na+1 nb+nk nc+1]);
jj = (k:Ncap);
x1 = zeros(n,1);
x2 = x1;
x3 = x1;
p1 = 10000*eye(n);
p2 = p1;
theta1 = zeros(n,1);
theta2 = theta1;
theta3 = theta1;
matrix1 = zeros(n,length(jj));
matrix2 = matrix1;
```

```
matrix3 = matrix1;
f = zeros(n,1); g = zeros(n,1); beta = zeros(n,1);
u = []; for j = n:-1:1, u = [u;zeros(1,n-j) 1 ones(1,j-
1)*10000]; end
d = ones(n,1)*10000; mu = zeros(n,1); v = zeros(n,1);
lambda = 1;

e1 = zeros(Ncap,1);
e2 = e1;
e3 = e1;

for fred = 1:length(jj)
        N = jj(fred);
     % MIL v1
     for ky=1:na, x1(ky)=-z(N-ky,1);end
        for ku=1:nb, x1(na+ku)=z(N-ku-nk+1,2);end
     for kc=1:nc, x1(na+nb+kc)=e1(N-kc);end
     e1(N) = z(N,1)-x1'*theta1;
     oldp1 = p1;
     p1 = oldp1-(oldp1*x1*x1'*oldp1)/(1+x1'*oldp1*x1);
     theta1 = theta1 + p1*x1*e1(N);
     matrix1(:,fred) = theta1;

     % MIL v2
     x2(1:na+nb) = x1(1:na+nb);
     for kc=1:nc, x2(na+nb+kc)=e2(N-kc);end
     e2(N) = z(N,1)-x2'*theta2;
     oldp2 = p2;
     l = oldp2*x2/(1+x2'*oldp2*x2);
     theta2 = theta2+l*e2(N);
     p2 = oldp2-l*(oldp2*x2)';
     matrix2(:,fred) = theta2;

     % UD factorisation !!!!!
     x3(1:na+nb) = x1(1:na+nb);
     for kc=1:nc, x3(na+nb+kc)=e3(N-kc);end
     e3(N) = z(N,1)-x3'*theta3;
     oldu = u; oldd = d;
     f = oldu'*x3;
     g = diag(oldd)*f;
     beta0 = lambda;
     beta(1) = beta0+f(1)*g(1);
     d(1) = oldd(1)/beta(1);
     v(1) = g(1);
     mu(1) = -f(1)/beta0;
     for j_ud = 2:n
          beta(j_ud) = beta(j_ud-1)+f(j_ud)*g(j_ud);
          d(j_ud) = beta(j_ud-
1)*oldd(j_ud)/(beta(j_ud)*lambda);
          v(j_ud) = g(j_ud);
          mu(j_ud) = -f(j_ud)/beta(j_ud-1);
          for i_ud = 1:j_ud-1
               u(i_ud,j_ud) =
oldu(i_ud,j_ud)+v(i_ud)*mu(j_ud);
               v(i_ud) = v(i_ud)+oldu(i_ud,j_ud)*v(j_ud);
          end
     end
     lud = v/beta(n);
     theta3 = theta3+lud*e3(N);
     matrix3(:,fred) = theta3;

end

% MIL v1 and MIL v2 from Wellstead & Zarrop SELF-TUNING
SYSTEMS
% also UD factorisation.

[theta1 theta2 theta3]
```

**B6 Pole-placement controller synthesis**

```
function theta_c = synth(a,b,c,t)
%
% function to synthesise controller coeffs
% using pole placement method
%
%       theta_c = synth(a,b,c,t);
%
%  theta_c - returned controller coeffs
%  a - A polynomial coeffs
%  b - B polynomial coeffs
%  c - C polynomial coeffs
%  t - T polynomial coeffs (desired pole set)
%
% poly's should be columns !
% don't require leading 1's im A C T

if c == [], c = 0; end

na = length(a);
nb = length(b) - 1;        % fiddle !!
n = na + nb;
A = zeros(n);

for j=1:nb
     A(j,j) = 1;
     A(j+1:j+na,j) = a;
end
for j=nb+1:n
     A(j-nb:j,j)=b;
end

B = conv([1;t],[1;c]);
B(1) = [];
for j = length(B)+1:n
     B(j) = 0;
end
B(1:na) = B(1:na)-a;
theta_c = inv(A) * B;
```

**B7 Kucera's algorithm**

```
function [newa,newb] = kucera(a,b)
% ***********************************************************
% * basics of a function to carry out Kucera`s algorithm  *
% * used before controller synthesis. Requires a & b as   *
% * column vectors based on                                *
% *                                                         *
% *      H(Z^-1) = Z^-1(b0 b1 b2 ....)                      *
% *                    --------------                       *
% *                    1 a1 a2 .....     as usual           *
% *                                                         *
% * b = [b0 b1 b2 ....]'                                    *
% * a = [a1 a2 ....]'                                       *
% * answer given in same format.                            *
% * assumes na >= nb                                        *
% * cng 10/8/92  from wellstead and zarrop                  *
% ***********************************************************
% 24/8/92

na = length(a);
nb = length(b);
degdiff = na-nb+1;                  % must be >=0

F = [[1;a] [0;b;zeros(degdiff-1,1)]];
V = [[zeros(nb,1);1] zeros(na+1,1); zeros(nb+1,1)
[zeros(na,1);1]];

nrowsV = na+nb+2;
finish = 0;
% main loop
while finish == 0
    [nrows ncols] = size(F);
    % calculate degdiff
    nna = nrows;
    for i = 1:nrows
     if F(i,1) == 0, nna = nna-1; end
    end
    nnb = nrows;
    for i =1:nrows
     if F(i,2) == 0, nnb = nnb-1; end
    end
    degdiff = abs(nna-nnb);

    if F(1,1) ~= 0
        lambda = F(1,1)/F(1+degdiff,2); high = 1; low = 2;
    else
        lambda = F(1,2)/F(1+degdiff,1); high = 2; low = 1;
    end

    % subtract columns
    % shift low column up by degdiff elements = mult by
Z^degdiff
```

```
    lowshiftF = zeros(nrows,1);
    for i = 1:nrows-degdiff
        lowshiftF(i) = F(i+degdiff,low);
    end
    F(:,high) = F(:,high) - lambda*lowshiftF;
    % same operation on V
    lowshiftV = zeros(nrowsV,1);
    for i = 1:nb+1-degdiff
        lowshiftV(i) = V(i+degdiff,low);
    end
    for i = nb+2:nrowsV-degdiff
        lowshiftV(i) = V(i+degdiff,low);
    end
    V(:,high) = V(:,high) - lambda*lowshiftV;

    % check for "zero" polynomial
    % must do before delete a row
    suma = 0; sumb = 0;
    SOME_NUMBER = .0000001;   % !!!!!!!!!! choice of this
number ?????????
                            % if 'true' too early will get
                            % more than one pole-zero
                            % cancelation !!!!!!!!!!!!!!!!!!

    for i = 1:nrows
     suma = suma + F(i,1)^2;
        sumb = sumb + F(i,2)^2;
    end
    if high == 1
        if suma/sumb < SOME_NUMBER, finish = 1; end
    else
     if sumb/suma < SOME_NUMBER, finish = 1; end
    end

    % remove zeros row = lose power of Z
    if sum(F(1,:)) == 0
        F(1,:) = [];
    end

 %   F, V, degdiff, disp('Any key'), pause
end
%disp('Result =')

% sort out answer
result = V(:,high);
% find highest power of Z in a
finish = 0; i = 1;
while finish == 0
    i = i+1;
    if result(nb+i) ~= 0, finish = 1; end
    if nb+i == nrowsV, finish = 1; end       % error !!
end
result = result/result(nb+i);
newb = -result(i:nb+1);
%newb
newa = result(nb+i+1:nrowsV);
```

```
%newa
```

## B8 Self-tuning controller

```
% bubble_k.m
% 26/8/92  now uses kucera's algorithm
% cng
clc
clear
disp('clear ')
disp('Building model........Please wait')
modbld
disp(' '),disp(' ')
disp('         1. Square wave @ 4 Hz')
disp('         2. Sine wave @ 4 Hz')
disp('         3. PRBS N = 9, fc = 100 Hz')
disp('         4. Low amplitude white noise')
disp('         5. Step')
disp(' '),disp('         (sample frequency = 200 Hz)')
disp(' ')
choice = input('    Choose your reference ');
if choice < 1 | choice > 5, break, end
if choice == 1
     ref = [ones(25,1);-ones(25,1)];
     ref = [ref;ref;ref;ref;ref;ref;ref;ref];
end
if choice == 2
     t = (0:399)'/200;
     ref = sin(2*pi*5*t);
end
if choice == 3
        ref = prbs(9,2);

end
if choice == 4
     rand('normal')
     ref = rand(400,1)/1000;
end
if choice == 5
     ref=[zeros(100,1);ones(300,1)/1000];
end
disp(' ')
choice = input('    Use Kucera (y/n), default = y ','s');
if  choice == 'n' | choice == 'N'
     use_kucera = 0;
else
     use_kucera = 1;
end

dd = .7; wn = 200;
t1 = -2*exp(-dd*wn/200)*cos(wn/200*(1-dd^2)^.5);
t2 = exp(-2*dd*wn/200);
t = [t1 t2]';
```

```
%t = -.6
if c == [0], c = []; end
rand('normal')
e = rand(length(ref),1)'/10;
na = length(a); nb = length(b);
nc = length(c);
nk = 1;
nt = length(t); nf = nb-1; ng = na;
N = length(ref);
u = zeros(N,1);
y = zeros(N,1);
ran=rand/10;
aest = ones(na,1)*1e-15; best = ones(nb,1)*1e-15; cest =
ones(nc,1)*1e-15;
f = zeros(nf,1); g = zeros(ng,1); gy = 0;
n = na + nb + nc;
oldp = 10000*eye(n); oldtheta = [aest;best;cest];
matrix = zeros(n,N); err = zeros(N,1); l = 1;
matrixc = zeros(nf+ng,N);
ptrace = zeros(N,1);

if nc == [0],
      e = zeros(N,1);             % comment out to add
noise;
      c = 0; cest = 0;
end
hr = 0; gy = 0; fu = 0;
for time=max([na+1 nb+nk nc+1]):N
qq=rand*10+35;disp([blanks(qq) 'o'])
     for i=1:nb, y(time) = y(time) + b(i)*u(time-i-nk+1);
end
     for i=1:na, y(time) = y(time) - a(i)*y(time-i); end
     y(time) = y(time) + e(time);
     for i=1:nc, y(time) = y(time) + c(i)*e(time-i); end
     [theta ce p] =
ssels(y,u,err,time,na,nb,nc,nk,oldp,oldtheta,l);
     matrix(:,time) = theta;
     ptrace(time) = trace(p);
     oldp = p;
     err(time) = ce(1);
     % reset estimator ?
     if abs(y(time) - y(time-1)) >.5
          [time y(time)-y(time-1)]
          oldp = 10000*eye(n);
          disp('P RESET')
     end

     oldtheta = theta;
     aest = theta(1:na);
     best = theta(na+1:na+nb);
     if nc > 0
          cest = theta(na+nb+1:na+nb+nc);
     end
     u(time) = ref(time);

     if time >= 100
     if use_kucera == 1
          [kaest,kbest] = kucera(aest,best);
     else
          kaest = aest; kbest = best;
     end
     knf = length(kbest)-1;
     kng = length(kaest);
          Q = synth(kaest,kbest,cest,t);
          f = Q(1:knf);
          g = Q(knf+1:knf+kng);
          nfdiff = nf-knf;
          ff = [f;zeros(nfdiff,1)];
          ngdiff = ng-kng;
          gg = [g;zeros(ngdiff,1)];
          Q = [ff;gg];
          matrixc(:,time)=Q;
%    end
          hr = 0;
          h = (1+sum(t))/sum(best)*[1 cest'];
          for i=1:length(h), hr = hr + h(i)*ref(time-i+1);
end
          gy = 0;
          for i=1:kng, gy = gy + g(i)*y(time-i+1); end
          fu = 0;
          for i=1:knf, fu = fu + f(i)*u(time-i); end
          u(time) = hr - gy - fu;
     end
end
clc
reflen = length(ref);
time = (0:reflen-1)/200;
index = 100:reflen;
disp(' '),disp('ref and system output')
plot(time,[ref y])
pause
disp(' '),disp('model parameters')
for i=1:10
     plot(time,matrix(i,:))
     pause
end
disp(' '),disp('control parameters')
for i=1:9
     plot(time(index),matrixc(i,index))
     pause
end
```

# APPENDIX C - C code

Below are listed three files containing C code.

- prbs.c - a prbs generator
- cng3.c - program to identify a fifth order ARX model using least squares estimation
- cng.h - header file containg functions for the above program

## <u>C1 PRBS generator</u>

```
/*
************************************************************
*
*** prbs.c
***
*** 6/11/92
***
***
***
*** requires register length as command line argument
*** *** can also give no. of periods
*** *** output currently sent to screen
*** ***
***
*** Godfrey KR, Correlation Methods, Automatica 16
(1980)***                                          ***
***
************************************************************
*
*/

#include<stdio.h>
#include<math.h>
#include<stdlib.h>

struct prbs_reg /* used to produce sequence */
{
   int value;
   struct prbs_reg *next;
   struct prbs_reg *last;
};

struct prbs_seq /* used to store sequence */
{
   int value;
   struct prbs_seq *next;
};

main(argc,argv)
   int argc;
   char *argv[];
{
   int get_val();
   void shift_reg();
   struct prbs_reg
*reg_start,*reg_last,*reg_temp,*reg_next;
   struct prbs_seq *seq_start,*seq_temp,*seq_next;
   int n,i,reg_size,fbk_stage,period,fbk,seq_size,ok,seqs;

   /* error checking and set up */
   if (argc<2 || argc>3)
   {
      printf("\nE R R O R - Integer argument(s)
required\n");
      printf("\nFORMAT - prbs reg_length no_of_periods
(Default 1 period)\n");
      exit(1);
   }
   n=atoi(argv[1]);
   if (n<2 || n>11 || n==8)
   {
      printf("\nE R R O R - only 2 3 4 5 6 7 9 10 or 11
register stages\n");
      exit(1);
   }
   period=((int)pow(2.0,(double)n))-1;    /* 'cos pow
requires doubles */
   if (n==2)
      fbk_stage=1;
   if (n==3)
      fbk_stage=2;
   if (n==4 || n==5)
      fbk_stage=3;
   if (n==6 || n==9)
      fbk_stage=5;
   if (n==7)
      fbk_stage=4;
   if (n==10)
      fbk_stage=7;
   if (n==11)
      fbk_stage=9;

   if (argc==3)
   {
      seqs=atoi(argv[2]);
      if (seqs<1)
      {
      printf("\nE R R O R - No. of periods must be
positive\n");
      exit(1);
      }
   }
   else
      seqs=1;

   /* initialise */
   reg_size=sizeof(struct prbs_reg);
   seq_size=sizeof(struct prbs_seq);
   reg_start=(struct prbs_reg *)malloc(reg_size);
   reg_temp=(struct prbs_reg *)malloc(reg_size);
   seq_start=(struct prbs_seq *)malloc(seq_size);
   seq_temp=(struct prbs_seq *)malloc(seq_size);
   reg_start->value=1;
   reg_start->next=reg_temp;
   reg_start->last=NULL;
   seq_start->value=1;
   seq_start->next=seq_temp;
   reg_last=reg_start;
   for (i=1;i<n;i++)
   {
      if (i!=(n-1))
      reg_next=(struct prbs_reg *)malloc(reg_size);
      else
      reg_next=NULL;
      reg_temp->value=1;
      reg_temp->next=reg_next;
      reg_temp->last=reg_last;
      reg_last=reg_temp;
      reg_temp=reg_next;
   }/* shift register initialised */

   /* calc rest of sequence */
   for (i=0;i<period-1;i++)
   {

fbk=get_val(reg_start,n)^get_val(reg_start,fbk_stage);
      shift_reg(reg_start);
      reg_start->value=fbk;
      seq_next=(struct prbs_seq *)malloc(seq_size);
      seq_temp->value=get_val(reg_start,n);
      seq_temp->next=seq_next;
      if (i==period-2)
      /* last time so, loop bcak to start */
      seq_temp->next=seq_start;
      seq_temp=seq_next;
   }/* sequence complete */

   /* output sequence */
   printf("Built sequence\n");
   seq_temp=seq_start;
   for (i=0;i<seqs;i++)
   {
      ok=1;
      printf("tigger..");
      while(ok)
      {
      fprintf(stdout,"%d ",seq_temp->value);
      seq_temp=seq_temp->next;
      if (seq_temp==seq_start)
         ok=0;
      }
      printf("\n");
   }
```

```
} /* end of main */

int get_val(reg,x)
   /* get value in element x of register reg        */
   /* x=1 gives first element, x=2 gives second etc */
   struct prbs_reg *reg;
   int x;
{
   struct prbs_reg *temp;
   int i;
   temp=reg;
   for (i=1;i<x;i++)
      temp=temp->next;
   return(temp->value);
}

void shift_reg(reg)
   /* shift operation on register reg */
   struct prbs_reg *reg;
{
   struct prbs_reg *last,*prev,*temp;
   while((reg=reg->next)!=NULL)
      temp=reg;
   last=temp;
   while ((prev=last->last)!=NULL)
   {
      last->value=prev->value;
      last=prev;
   }
}
```

**C2 Least squares estimation**

```
/* cng3.c */
#include<stdio.h>
#include"cng.h"

#define Na 5        /* because of way data currently
stored*/
#define Nb 5        /* Na must equal Nb
!!!!!!!!!!!!!!!!!!!!*/
#define N  Na+Nb

typedef double vector[N];
typedef double matrix[N][N];

main()
{
   /* ********** initialise ********** */
   FILE *file_ptr;
   int i,j,count;
   double u[Nb+1],y[Na+1],dummy;
   double e,y_estimate,den,next;
   vector X,theta,v1,v2;
   matrix oldp,p,m1,num;

   for (i=0;i<N;i++)
      oldp[i][i]=10000.0;
   file_ptr=fopen("test_data","rt");          /* 95 samples
*/
   /*file_ptr=fopen("oct_test1.mat","rt");*/ /* 995 samples
*/
   if (file_ptr==NULL)
   {
      printf("Unable to open file\n");
      exit(1);
   }
   for (i=0;i<Na;i++)
      /*fscanf(file_ptr,"%le %le %le %le %le",&dummy,&u[Nb-
i],&y[Na-i],&dummy,&dummy,&dummy);*/
      fscanf(file_ptr,"%le %le",&y[Na-i],&u[Nb-i]);


   /* ********** start of main loop ********** */
   for (count=0;count<95;count++)
   {
      /* read next sample */
      /*fscanf(file_ptr,"%le %le %le %le
%le",&dummy,u,y,&dummy,&dummy,&dummy);*/
      fscanf(file_ptr,"%le %le",y,u);

      /* make X vector */
      for (i=0;i<Na;i++)
      X[i]=y[i+1]*-1.0;
      for (j=i;j<N;j++)
      X[j]=u[j-i+1];

      /* calc error */
      y_estimate=vvtmult(X,theta,N);
      e=(*y)-y_estimate;

      /* calc p */
      mvmult(oldp,X,v1,N);
      vvmult(v1,X,m1,N);
      mmmult(m1,oldp,num,N);
      vmmult(X,oldp,v2,N);
      den=vvtmult(v2,X,N);
      den+=1.0;
      mdivd(num,den,N);
      msubtract(oldp,num,p,N);

      /* update theta */
      mvmult(p,X,v1,N);
      for (i=0;i<N;i++)
      theta[i]=theta[i]+v1[i]*e;
      shift(u,Nb+1);
      shift(y,Na+1);
      mcopy(p,oldp,N);
   }
   printf("theta");
   vdisp(theta,N);
   fclose(file_ptr);
}/* ********** end of main ********** */
```

**Header file**

```
/* cng.h */

void shift(x,m)
   double *x;
   int m;
{
   double *last,*prev;
   int i;

   last=x+m-1;
   prev=x+m-2;
   for (i=0;i<m;i++)
      (*last--)=(*prev--);
   *x=0;
}

void vdisp(x,m)
   double *x;
   int m;
{
   int i;
   printf("\n");
   for (i=0;i<m;i++)
      printf("%.15e\n",*x++);
}

double vvtmult(x,y,m)
   double *x,*y;
   int m;
{
   int i;
   double z;

   for (i=z=0;i<m;i++)
      z+=(*x++)*(*y++);
   return(z);
}

void mdisp(x,m)
   double *x;
   int m;
{
   int i,j;
   for (i=0;i<m;i++)
   {
      for (j=0;j<m;j++)
      printf("%e ",*x++);
      printf("\n");
      }
   }

   void mmmult(x,y,z,m)
      double *x,*y,*z;
      int m;
   {
      int i,j,k;
      double *xrow,*ycol;
      for (i=0;i<m;i++)
      {
         xrow=x;
         ycol=y;
         for (j=0;j<m;j++)
         {
         for (k=(*z)=0;k<m;k++)
         {
            (*z)+=(*xrow)*(*ycol);
            xrow++;
            ycol+=m;
         }
         xrow=x;
         ycol=y+j+1;
         z++;
         }
         x+=m;
      }
   }

   void vvmult(x,y,z,m)
      double *x,*y,*z;
      int m;
   {
      double *ycol;
      int i,j;
      for (i=0;i<m;i++)
      {
         ycol=y;
         for (j=0;j<m;j++)
         (*z++)=(*x)*(*ycol++);
         x++;
      }
   }

   void vmmult(x,y,z,m)
      double *x,*y,*z;
      int m;
   {
      double *xcol,*ycol;
      int i,j;
      ycol=y;
      for (i=0;i<m;i++)
      {
         xcol=x;
         for (j=(*z)=0;j<m;j++)
         {
```

```
        (*z)+=(*xcol)*(*ycol);
        xcol++;
        ycol+=m;
        }
        ycol=y+i+1;
        z++;
    }
}

void mvmult(x,y,z,m)
    double *x,*y,*z;
    int m;
{
    double *xcol,*ycol;
    int i,j;
    xcol=x;
    for (i=0;i<m;i++)
    {
        ycol=y;
        for (j=(*z)=0;j<m;j++)
        {
        (*z)+=(*xcol)*(*ycol);
        ycol++;
        xcol++;
        }
        xcol=x+m*(i+1);
        z++;
    }
}

void msubtract(x,y,z,m)
    double *x,*y,*z;
    int m;
{
    int i;
    for (i=0;i<(m*m);i++)
        (*z++)=(*x++)-(*y++);
}

void mcopy(x,y,m)
    double *x,*y;
    int m;
{
    int i;
    for (i=0;i<(m*m);i++)
        (*y++)=(*x++);
}

void mdivd(x,y,m)
    double *x,y;
    int m;
{
    int i;
    for (i=0;i<(m*m);i++)
        (*x)=(*x++)/y;
}
```

## APPENDIX D - Data logging details

Below is an example of the information that was stored as a data record.

```
filename : prbs1
header   : header2
date     : 23/10/91
subject  : system identification - PRBS testing AGC
clock F  : 3kHz
register : 10
amplitude: 50mv
control  : gap
channels : 25
sample   : 7kHz
length   : 16k
notes    : were rolling. dither present. no chan 16
         :

   SIGNAL                     NAME     CALIBRATION      ADC RANGE
1  position reset             POS RE   -1v/mm           5
2  stand speed                ST SPD   10v=350m/min     10
3  left tension               LTNSN    10v=196kN        10
4  right tension              RTNSN    10v=196kN        10
5  entry gauge error          ENTGER   10v=100u thick   10
6  exit gauge error           EXTGER   10v=100u thick   10
7  exit gauge ref             EXTREF   1v/mm            5
8  entry gauge ref            ENTREF   1v/mm            5
9  entry speed                ENTSPD   10v=350m/min     10
10 exit speed                 EXTSPD   10v=350m/min     10
11 load                       LOAD     5v=200tonnes     5
12 rack pull (pressure)       PRESS    1v=1tonne        10
13 true position              TRUPOS   -2v/mm           10
14 position ref               POSREF   -1v/mm           10
15 servovalve current         SERVOI   1v=20mA          2.5
16 rot speed LH winder        RSLHW    10v=1325rpm      10
17 rot speed RH winder        RSRHW    10v=1325rpm      10
18 stand speed ref            STDREF   10v=350m/min     10   +ve
roll right
19 left tension ref           LTREF    -10v=196kN       10
20 right tension ref          RTREF    -10v=196kN       10
21 main drive field I         FIELDI   5v=18A           5
22 main drive armature I      ARM I    5v=1200A         5
23 main drive armature V      ARM V    -5v=580v         5
24 prbs                       CHAN24                    2.5
25 trigger o/p                CHAN25                    10
```

## APPENDIX E - Test rig details

### Mill Stand

| | |
|---|---|
| Total mass | 150 tonnes |
| Mill spring | 450 tonnes/mm @ 200 tonnes load |
| | 700 tonnes/mm @ 2000 tonnes load |
| Housing posts | 1020mm x 780mm |
| Height of housing | 7600mm |
| Window size | 1524mm x 5072mm |
| Maximum load | 1100 tonnes |

### Roll / Chock blocks

Two steel blocks of mass 25 tonnes each to simulate the roll / chock masses.

### Pump set

Maximum pressure greater than 400 bar (6000psi)
Flow: 230 l/min (50gpm) @ 190 bar (2500psi)
160 l/min (25gpm) @ 276 bar (4000psi)

### Capsule installation

Hydraulic rams to raise or lower roll / chock blocks plus hydraulic ram to slide capsule in and out of housing. Capsule stroke changed by adding or subtracting packers.

### Development capsule

850mm bore single acting capsule with maximum 300mm stroke. Designed for either top or bottom mounting with different manifold arrangements.

## APPENDIX F - Additional notes

These notes have been included to clarify a number of points raised by the examiners.

1. Figure 1.2 shows the overall control loop considered in the majority of this thesis. It is the position loop of the hydraulic AGC system, with capsule position reference as input and measured capsule position as output.

2. Figure 2.1 shows a model of the hydraulic AGC capsule and servo valve which is shown pictorially in figure 1.2.

3. Chapter 3 is primarily concerned with the identification of the position loop of the AGC system, figure 1.2. There are a number of sources (including gauge variations, roll eccentricity and trims from other control loops) resulting in a random disturbance affecting this loop.

4. Chapter 4 looks at the development of a self-tuning controller, based on the pole-placement technique, to control the position loop of the AGC by utilising the model identified in chapter 3. The desired response of the controller was specified in terms of a second order system.

5. Figure 4.1 shows the block diagram of the pole-placement controller and the plant. r(t) is the same as the reference signal of figure 1.2, u(t) corresponds to servo valve current and y(t) to measured capsule position.

6. Figure 4.2 relates to figures 1.2 and 4.1. The position reference signal r(t) is not shown but is a ±10 microns square wave. Although unrealistic, this reference was chosen to demonstrate the self-tuning action. u(t), the control signal, is the servo valve current, a suitably scaled version of this signal is included on the same axis as the output signal, again to demonstrate the self-tuning action. y(t), the system output (measured capsule position), is seen to follow the reference with the desired response, giving no steady state error, once the controller is active.