The Design and Development of a Fully Dynamic Simulator for Renewable Energy Converters

Thesis submitted for the degree of
Doctor of Philosophy
at the University of Leicester

by

David Anthony Parker MSc(Strathclyde)
Department of Engineering
University of Leicester

March 2000

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

Design and Development of a Fully Dynamic Simulator for Renewable Energy Converters

David Anthony Parker

Abstract

This report describes the work undertaken to develop a real-time dynamic simulator for a renewable energy conversion system, focusing on wind energy converter systems (WECS). An assessment of existing simulators including hardware-in-the-loop simulation (HILS) has shown that there is a need for a hardware simulator with a comprehensive description of WECS aerodynamics.

The report reviews the modelling of WECS and establishes models for both the aerodynamics and drive train dynamics. Once the models have been established the hardware and software used for the simulator are introduced. The hardware consists of a DC drive controlling a DC motor, which acts as a prime mover and provides a shaft torque for a grid connected induction generator. The software used to model the WECS 'front end' and to provide a torque demand, via a serial communications link for the DC drive, is the Matlab/Simulink environment with the Real-Time Workshop.

A model of a 45kW WECS is developed from specifications provided, and implemented in Simulink as a software-only design. Verification of the model is obtained by comparing the performance of the simulation with measured site data.

Following the verification of the software-only model, the effects of including hardware in the simulation are modelled and assessed. Additionally, the effects of including the DC motor in the hardware test-bed are investigated and compensated for prior to the assessment of HILS.

The results of the HILS show that the simulator compares favourably with the measured site data and meets the objectives of the project brief. There are, however, some discrepancies between the simulated results and measured site data at high frequencies due to noise in the system. An alternative method of communication, between the PC and drive using a data acquisition card, is introduced to improve the response. The resulting simulations, with the card used for communications, show that the low to mid-frequency response is improved.

To emphasise the flexibility of the simulator a micro-hydro plant (MHP) is proposed and simulated, both in software and HILS. Results are favourable, showing the adaptability of the simulator to switch between different renewable energy converters with ease.

I would like to dedicate this thesis to the memory of my Dad, Harry Parker.

Acknowledgements

I would like to thank the teaching and technical staff of the University of Leicester Engineering Department for their help and friendship during my studies. Particular thanks go to Mr Adrian Payne for constructing the hardware test-bed and Dr G.A. Smith for his encouragement and interest in my research. Thanks also to Dr Randall Jones and Dr. J.A.M. Bleijs.

Acknowledgement goes to Dr G Dutton, Dr R Paynter and Dr A Ruddell of RAL for providing the WECS data which made this project possible.

Finally I would like to thank my partner Jo Swingler for her encouragement, understanding and friendship.

Contents

Glossary of Principle Symbols

β	Pitch angle
$\gamma_{\#}$, **damp#**	Damping coefficients
ρ	Density
λ	Tip speed ratio
$\theta_{\#}$	Angles of rotation
ϕ	Flux
τ	Time constant
$\omega_{\#}$	Rotational speeds
ω_e	Rotational speed error
ω_{ls}	Low speed shaft rotational speed
ω_{hs}	High speed shaft rotational speed
ω, ω_R	Rotational frequency of rotor
$\mathbf{A_0}$	Cross sectional area of upstream airflow
$\mathbf{A_1}$	Cross sectional area of the airflow at the actuator disc
$\mathbf{A_2}$	Cross sectional area of the downstream airflow
$\mathbf{C_L}$	Lift coefficient
$\mathbf{C_D}$	Drag coefficient
$\mathbf{C_p}$	Power coefficient
$\mathbf{C_q}$	Torque coefficient
$\mathbf{D_n}$	Data bits
$\mathbf{D_{\#}}$	Torque speed gradient
E	Peak line voltage
f, F	Force
$\mathbf{h_{\#}}$	Height
g	Acceleration due to gravity
$\mathbf{I_{\#}}$	Lumped inertias
$\mathbf{J_{\#}}$	Inertias
$\mathbf{J_g}$, $\mathbf{J_G}$	Generator inertia

J_R*	Effective rotor inertia
$K_\#$	Stiffness
m	Mass flow rate
N	Gearbox ratio
$P_\#$,$p_\#$	Power
$PL_\#$	Power losses
Q	Water flow rate
R	Radius of Rotor
rr	Rotor resistance
rs	Stator resistance
Δt	Time difference
$T_\#$	Torques
t_{ls}, T_{ls}	Low speed shaft torque
t_{hs}, T_{hs}	High speed shaft torque
T_a , T_A	Aerodynamic Torque
T_{rot}	Torque due to rotational sampling
T_w	Water time constant
v	Water velocity
V_0	Upstream airflow velocity
V_1,V	Airflow velocity at the actuator disc
V_2	Downstream airflow velocity
W	Power extracted by rotor
xlr	Rotor leakage reactance
xls	Stator leakage reactance
xm	Magnetising reactance
z_T	Tower movement

Chapter 1 Introduction

The overall aims of the project are best summarised with the following extract from the initial summary brief, provided by the project supervisor:

> 'Research into the development and application of renewable energy sources for the generation of electricity is often hampered by the inaccessibility of sites, the lack of instrumentation on commercial installations and the unpredictability of the energy source.
>
> However, the relations governing the conversion processes, such as the aerodynamic or hydrodynamic behaviour of turbines, are generally well established. To assess the potential of any electrical generator scheme for improving the overall efficiency of the energy extraction process, it is desirable to have access to a drive facility that can be programmed to simulate the characteristics of renewable energy converters under varying conditions in a controlled and repeatable manner. Such a facility can be designed and constructed using hardware already existing in the Electrical Power Engineering Section.
>
> The proposed research will involve the design of a test bed with the appropriate control hardware and software, and extensive testing to establish the ability of the rig to simulate the performance of wind, hydro and wave power turbines.' [1.1]

As the project developed it was decided to concentrate on the development of a fully dynamic simulator for a wind energy conversion systems (WECS). The provision in the initial development is that the simulator would have to be flexible enough to allow the development of other renewable energy converters without extensive redevelopment of the simulator software or hardware. To emphasise this flexibility the simulator should also include a micro-hydro plant (MHP) model once a WECS has been modelled.

1.1 Implications of the Brief

The initial implication of the brief is that an understanding of the physical structure of a WECS is necessary to establish what hardware facilities are required for the simulator. The hardware test-rig for electrical generation can then be designed and the communication and control software requirements of the rig determined.

Similarly, an understanding of the developments in the dynamic modelling of a WECS is necessary to assess the software requirements for modelling. Also, developments in modelling concerned with aerodynamic interaction between the WECS and the wind, will have to be investigated.

Other implications of the brief are that research and development is required in the following areas:

- An assessment of the current developments of both software-only simulations, and simulations where software is controlling some form of electrical generation hardware under closed loop control. This latter option is commonly referred to as hardware-in-the-loop simulation (HILS) [1.2].

- Project funds are limited. The development of software and hardware should, therefore, make use of equipment and packages available in the Department wherever it is possible. Using a standard software package, widely used in industry, would also reduce the development time for tasks such as modelling and control.

- The aim of the project is to produce electrical power from a generator comparable to the output of a WECS. It is therefore necessary to be able to drive the generator in real-time using a suitable prime mover, under software control, to produce a shaft torque on the generator. This torque will be comparable to the torque developed by a real WECS.

- Since the simulator will, eventually, be used to simulate other forms of renewable energy converters, it should be flexible enough to allow for different generator types.

- The development of a WECS simulator will have to be validated by comparing it with a real operational WECS. Therefore, some form of assessment criterion will have to be established. Data from a real WECS would need to be obtained for both modelling requirements and performance comparison.

1.2 Initial Simulator Design

The content and the analysis of the project brief indicate that a software package will be used to model part of the WECS and its interaction with the wind. The output of the model will be used to control the shaft torque of an electrical generator via a suitable interface and prime mover. Fig. 1.1 is a block diagram of the basic design.

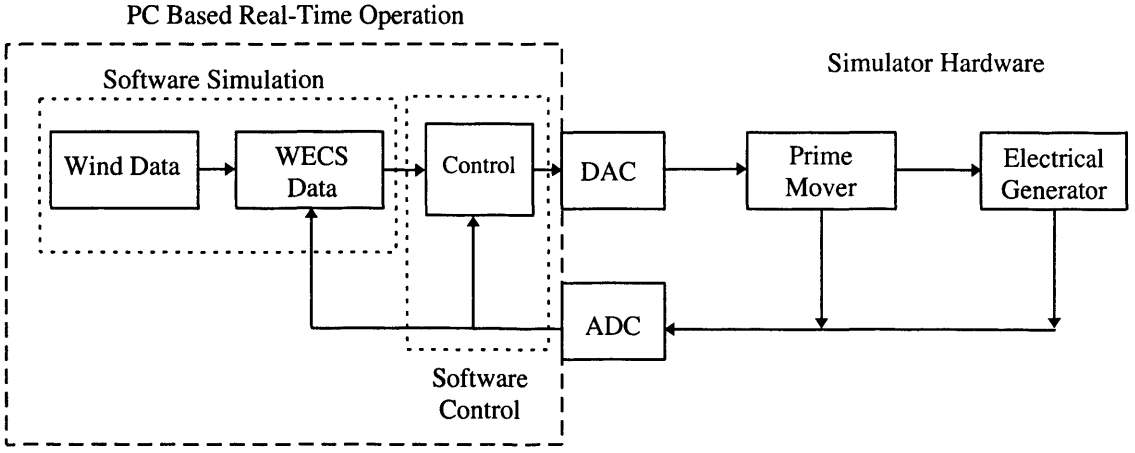


Fig. 1.1 Basic Design of the Lab Based WECS Simulator

The software side of the simulator will be PC based and model the 'front-end' of the WECS. This includes the wind data, the aerodynamic behaviour and the mechanical dynamics of the WECS. The hardware side of the simulator consists of the prime mover, which produces the drive torque for the generator, mimicking the actions of a WECS.

The output of the model in software will be used to derive a demand signal for the hardware via an interface. The nature of this demand depends on the requirements of the prime mover, which could be electrical, hydraulic or pneumatic. In general, an electrical signal output is required from the software. This is realised using a digital to analogue converter (DAC). Since the electrical output of an DAC is relatively small compared to that of an electrical generator, additional signal conditioning maybe required to ensure the output signal is suitable for the prime mover [1.3]. This would suggest that some form of power amplification is required.

Closed loop control in software will require the feedback of the status of the simulator to ensure optimum operation. Additionally, there may be possible interaction between the hardware parameters and the software required during modelling (see Chapter 5). These parameters need to be measured during the simulation and this data will, inevitably, need suitable conditioning for use with the software via an analogue to digital converter (ADC) interface. Again, some additional conditioning may be required on the measured signal, depending on the nature of the measurement.

The type of electrical generator used will depend on the final use of the output power. WECS may operate under a number of speed options. The most widely used methods are constant speed operation and variable speed operation.

1.2.1 Variable Speed and Constant Speed Operation - The Choice of Generator

Constant speed WECS employ an induction machine directly connected to the grid/mains supply. Although referred to as constant speed the operation will experience a variation of speed, but since this variation is small compared with variable speed operation, it is usually referred to as 'constant speed'. A variable speed WECS may use an AC synchronous or induction generator with a converter and a DC link. The variation of speed is over a much larger range compared with the constant speed operation. Alternatively, a DC generator connected to an inverter can be employed.

Each scheme has its advantages and disadvantages, the variable speed WECS are controlled to operate more efficiently and result in a higher energy yield, but this is not without a price. The controllers and the additional power conditioning such as the DC link and inverter, raise the capital costs of the WECS as well as the complexity of the system. The constant speed WECS, on the other hand, tend to be more robust and cheaper. The lack of complexity provides a higher level of reliability [1.4].

To limit the complexity of the system and simplify the analysis and validation, the design will concentrate on constant speed operation of a WECS. This will also have the added advantage of limiting the costs of the project. The selected generator for the lab-based simulator of Fig. 1.1 will therefore, be a grid-connected induction machine (IM). Development of the MHP model will also be based around a grid-connected IM. The simulator will be designed to be flexible enough to allow the adoption of alternative operating schemes, i.e. it will not be restricted to constant speed operation.

1.3 Current WECS Simulators - Literature Review

In order to assess the need for the proposed WECS simulator, it is necessary to assess the current simulators in this field and determine their limitations. Appendix 1a describes the findings of the study.

Five WECS simulators were assessed, namely:

- ECN IRFLET Project Test-rig
- Delft University of Technology DUWECS
- SMI, Technical University of Sofia, Combined Multiple Renewable Energy Sources System Simulator Facility.

- University of Rome, Microprocessor Controlled DC Drive as a Simulator of Wind Turbines

- University collaboration (Japan), Blade-Pitch-Angle-Controllable Windmill Simulator

The investigations do not indicate a definitive wind turbine simulator. Each project has its own notable innovative design, for example:

- The Japan simulator includes compensation in software for the presence of a DC motor acting as a prime mover.

- The IRFLET and University of Rome simulators both use a flywheel to simulate the effects of the rotor inertia. This has the advantage of including the rotor dynamics, but makes the simulator inflexible, e.g. for investigation of the effects of different rotor designs with different inertias.

- The SMI simulator includes both a WECS simulator and a photo-voltaic (PV) simulator connected to a bus with a diesel gen-set and dump load. The WECS simulator includes the effects of spatial distribution of the WECS within a wind park domain.

- The DUWECS simulator models both constant and variable speed WECS and includes some WECS aerodynamic and drive train dynamic effects.

Each simulator tends to include some dynamic modelling but none include all the relevant aerodynamics and drive train dynamics. For example, the DUWECS model includes rotational sampling and tower shadow effects but fails to include the effects of spatial filtering (see Chapter 2).

The exclusion of the dynamics is usually to make the simulator simpler since the simulator is used to test various hardware configurations. The IRFLET simulator, for example, is used to test control algorithms for variable speed operation at steady state. The need for complicated wind profiles is therefore, not required.

Various control algorithms and hardware were used on the simulators, these ranged from simple speed control for constant speed drives to control systems to moderate drive train resonance. Normally, control applications will vary from machine to machine, depending on machine type and optimal performance limits.

What was particularly apparent from the study were the problems encountered when attempting to compare the simulators with real systems since an exact reproduction of a particular wind regime was not obtained. This was evident with the DUWECS system and will obviously have to be considered during the development of the simulator rig.

All of the simulators are PC based which improves the flexibility of the system. Also the majority of the simulators sampled here boast 'user-friendly' designs for both software and hardware interaction. Not only will this improve the 'marketability' of the system but will allow any upgrade to the system to be implemented more easily.

A DC motor was used in all of the simulators where an 'external' drive facility was designed to model WECS drive trains. The relative ease of control of a DC motor, which will be discussed in Chapter 3, tends to explain this decision. A mixture of generators including three-phase asynchronous and synchronous, and DC were used on the rigs. This decision depended on the intended use of the simulator.

The overall impression of the literature review indicate that WECS simulators have met with success for the areas they were designed for, but the following points became apparent:

- There is a need for a flexible all-purpose WECS simulator rig incorporating all aspects of aerodynamics and wind characteristics.

- Such a rig will need to include a generator driven by a software-controlled prime mover to deliver 'real' power. This would allow the development of generator configurations and control schemes

- Such a system would need to be user friendly

- Extensive and realistic testing would be required. This would require direct comparison with an operational WECS.

- The simulator should be flexible enough to allow the development of additional renewable energy converters, such as an MHP, without major modifications to either the hardware or software.

1.4 Report Structure

In order to realise a fully dynamic simulator for WECS it is necessary to understand the characteristics of both the WECS aerodynamics and drive train and the interaction with the wind. **Chapter 2** describes the current development of WECS models, which include all the relevant dynamics, concentrating on grid connected I.M. machines.

Chapter 3 focuses on the development of the simulator hardware based on the design of Fig. 1.1. It will explain the selection of the particular hardware, for example the use of a DC motor as the 'prime mover'. Details are also given on the choice of using the Mentor II DC drive to perform the hardware closed loop control and the ADC and DAC operations. Additionally, attention is given to the serial communications software of the Mentor II and how this may be manipulated for novel communication with a PC.

The Mathwork's Matlab/Simulink software package is introduced in **Chapter 4** as the selected tool used for modelling the dynamics of a WECS. This package, along with the

software tool, the Real-Time Workshop, can be used to operate under real-time control and allows external communication. The chapter also describes the development of a unique design to establish real-time communication between the Mentor II drive and the Simulink software.

A WECS model with all the relevant aerodynamics, discussed in Chapter 2, is developed as a Simulink model in **Chapter 5**. The model is derived from specifications provided for a 45kW WECS. The development is accompanied by extensive comparisons between the model and measured site data to justify and validate the design.

A limitation in the size of the laboratory hardware means that a smaller generator is used with HILS. In **Chapter 6** the effects of using a smaller generator in the hardware simulator are investigated. Additionally, other effects of including HILS are assessed. This includes the introduction of delays and quantisation errors due to the hardware. As before the simulated results are compared with the measured site data to verify the design.

In **Chapter 7** compensation for the presence of the motor as the prime mover is developed and verified. This is followed by the results of including HILS and discusses the merits of the simulator as well as the problems encountered due to noise in the system.

In an attempt to improve the response of the simulator discussed in Chapter 7, an alternative method for communication between the PC and the DC drive, using a PC based data acquisition card, is developed and detailed in **Chapter 8**. Some improvements are noted and discussed.

Finally, **Chapter 9** describes the theory behind the modelling of a micro-hydro plant (MHP) and details the implementation and results of such a model on the simulator. This

is undertaken to show the flexibility of the simulator and emphasise its use for further development.

1.5 Summary

The proposal of the project is to develop a hardware based fully dynamic simulator for renewable energy converters. Initial studies indicated that efforts would be best served concentrating on a simulator for a wind energy converter system. At the conclusion of this simulation, a micro-hydro plant will be added to emphasise the flexibility of the lab-based simulator

Before development of the simulator was begun it was deemed appropriate to assess the current design and development of similar simulators. A literature review of current developments indicated that a definitive, dynamic and flexible simulator has not yet been realised. All the simulators have some novelty but they are limited to the hardware application they were developed for. The conclusion of the review is that there is a need for a flexible simulator, which would incorporate all the pertinent dynamics of a WECS drive train and the associated aerodynamics.

Unless stated or referenced, all the design and development work enclosed in this thesis is the original contribution of the author.

Chapter 2 Modelling of Wind Turbine Dynamics

A wind energy conversion system (WECS) converts the kinetic energy of the wind into rotational mechanical energy. The aerodynamic properties of the blades of the WECS interact with the air flow creating lift on the blades. The in-plane component of the lift forces combine to create a torque which will rotate the rotor and drive a mechanical load on the shaft. The larger the diameter of the turbine, and therefore the rotor, the more energy will be converted. Sometimes this conversion is used to drive mechanical systems directly, such as water pumps but, more usually, the mechanical power drives an electro-mechanical device or generator, to produce electrical power.

The choice of electrical generator used is dependent on the final use of the electrical energy produced. For example, some smaller WECS are used to charge batteries and use either DC generators or AC synchronous generators with some form of rectification. Induction machines (IM) are often used where the electrical energy produced is fed directly into the electrical grid. The choice of generator affects the design and performance of the WECS 'drive train'. Using an IM directly coupled to the grid demands that the rotational speed of the generator be maintained at or near synchronous speed (i.e. 3000rpm for a two-pole machine). Due to its size and the airfoil aerodynamics, it is not possible for the WECS's rotor to rotate at this speed, therefore a step-up gearbox is required to match the speeds of the rotor and the generator.

In order to develop a full model of the WECS dynamics, it is necessary to understand the dynamic interaction between the WECS drive train components; from the aerodynamic interaction between the wind and the blades to the interaction between the gearbox and the generator. This chapter discusses the various dynamics inherent with the drive train of a WECS, concentrating on a constant speed, grid connected WECS. The dynamics can be applied equally to a variable speed WECS but some of the simplifications may not be valid.

A 'constant speed' device is chosen with the hardware-in-the-loop requirements in mind. As mentioned in Chapter 1, constant speed control is less complex than variable speed control. Additionally, an IM connected directly to the grid is connected to a constant voltage and frequency, therefore an external load, with suitable control, is not required [2.1]. It was felt that this option would be preferable in order to reduce initial hardware development.

2.1 WECS Aerodynamics

As stated above, a WECS extracts the kinetic energy from wind and converts it into mechanical energy. The power available for a WECS is equal to the change in kinetic energy of the air as it passes through the rotor. The amount of available power can be analysed by assuming that the area swept by the WECS blades can be represented by a what is known as an actuator disc [2.2] and [2.3]. Fig. 2.1 shows the profile of a disc in a streamtube.

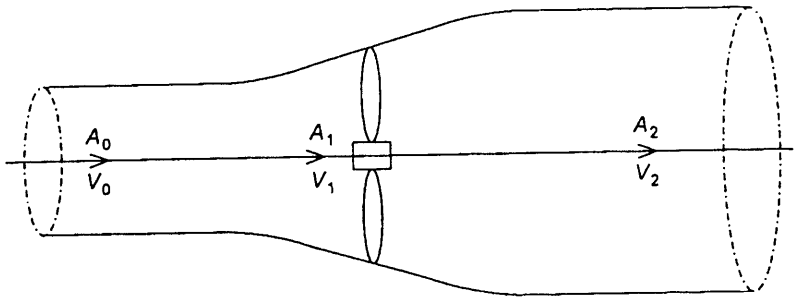


Fig. 2.1 Actuator Disc and Streamtube

If the airflow is regarded as incompressible, the mass flow rate of air in the streamtube must be equal throughout. Additionally, if considering only simple momentum theory, the air pressures immediately upstream and downstream of the rotor are constant over the rotor disk. Therefore, if the velocity of the airflow reduces at the rotor disc, the streamlines (cross sectional area of the flow) must diverge. In other words:

$$\text{Mass Flow Rate, } m = \rho A_0 V_0 = \rho A_1 V_1 = \rho A_2 V_2 \tag{2.1}$$

where A_n and V_n represent the cross sectional area of the streamtube and airflow velocity, respectively, at different points in the streamtube. Air density is represented by ρ.

The force, F, on the rotor disc is the rate of change of momentum through the streamtube:

$$F = m(V_0 - V_2) \tag{2.2}$$

and the power extracted by the rotor is:

$$W = m(\tfrac{1}{2}V_0^2 - \tfrac{1}{2}V_2^2) \tag{2.3}$$

Since the power extracted by the rotor is a product of the force on the rotor and the velocity, Eqn. (2.2) and Eqn. (2.3) can be rearranged to give:

$$V_1 = \tfrac{1}{2}(V_0 + V_2) \tag{2.4}$$

Defining a velocity factor, b, which relates the upstream velocity to the downstream velocity as the ratio:

$$b = \frac{V_2}{V_0} \tag{2.5}$$

the previous equations can be manipulated to give :

$$W = \frac{1}{2}\rho A_1 V_0^3 \times \frac{1}{2}(1 - b^2)(1 + b) \tag{2.6}$$

This can be rearranged to give:

$$W = C_p \frac{1}{2} \rho V_0^3 \pi R^2 \tag{2.7}$$

where C_p is the power coefficient and R is the radius (m) of the WECS rotor.

From Eqn. (2.6) and Eqn. (2.7) C_p can be defined as:

$$C_p = \frac{1}{2}(1-b^2)(1+b) \tag{2.8}$$

The maximum value of C_p can be calculated by differentiating C_p w.r.t. b resulting in a value of b = 1/3. Substituting this value into (2.8) gives :

$$C_p = \frac{16}{27} = 0.5926 \tag{2.9}$$

This is known as the Betz limit. In practice, the value of C_p is never greater than 0.5 due to the practical nature of slowly turning, low solidity rotors (total blade area as a ratio of swept rotor area) causing energy waste in wake rotation [2.4].

The value of C_p for a particular WECS varies non-linearly with the 'tip speed ratio' λ, which is defined as the ratio of tangential speed of the blade tips to the undisturbed wind speed:

$$\lambda = \frac{\omega R}{V} \tag{2.10}$$

where ω is the rotational velocity of the rotor (rad/s). A typical C_p - λ curve for a WECS is shown in Fig 2.2.

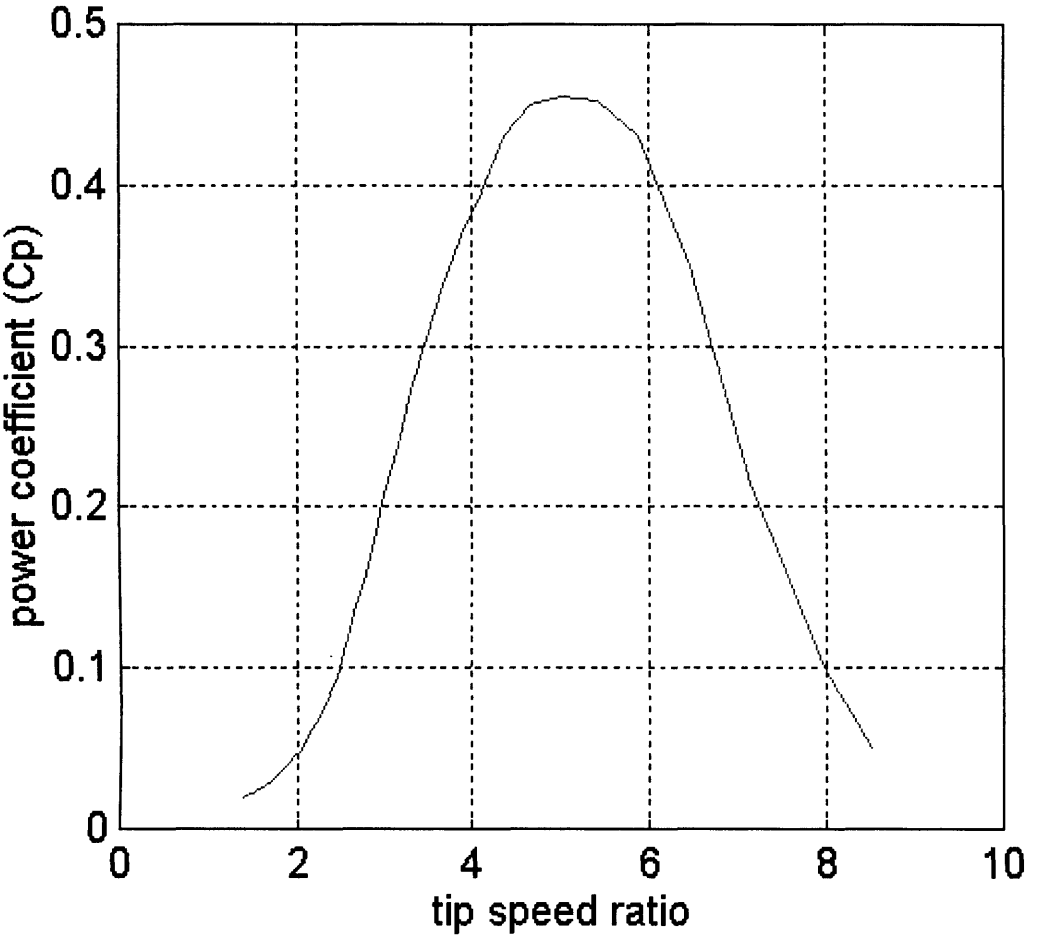


Fig. 2.2 The C_p - λ curve of the Windharvester 45kW WECS [2.5]

It will be seen later that the force developed from the interaction between the WECS and the wind, is best described as an aerodynamic torque. From Eqn. (2.7) and Eqn. (2.10), the aerodynamic torque T_a, can be defined as:

$$T_a = \frac{1}{\omega} C_p \frac{1}{2} \pi \rho V^3 R^2$$

$$T_a = C_q \frac{1}{2} \pi \rho V^2 R^3 \qquad (2.11)$$

where C_q is defined as the torque coefficient, and is related to C_p by:

$$C_{q=}\frac{C_p}{\lambda} \tag{2.12}$$

C_q - λ curves are blade/rotor dependant and since the relationship is non-linear, for simulation purposes the values are normally held in software as a look-up table [2.6]. There are, however, some examples where the curves have been described mathematically as a set of equations, e.g.

$$C_q = \alpha + \beta\lambda + \gamma\lambda^2 \tag{2.13}$$

where α, β and γ are constants which are device dependent [1.10]. The option chosen for any simulation, will have to depend on the information available for the WECS under investigation.

2.1.1 Pitch Regulated and Stall Regulated WECS

It was calculated earlier that the available power in the wind is directly proportional to the cubic value of wind speed. Above rated wind speeds, some form of power regulation is usually required due to the load limitation on either the structure of the WECS and/or the electrical generator. Power regulation relies on a change in the aerodynamics of the rotor blades. In other words, the C_q-λ data is modified. This can be further explained by considering Fig. 2.3 [2.3].

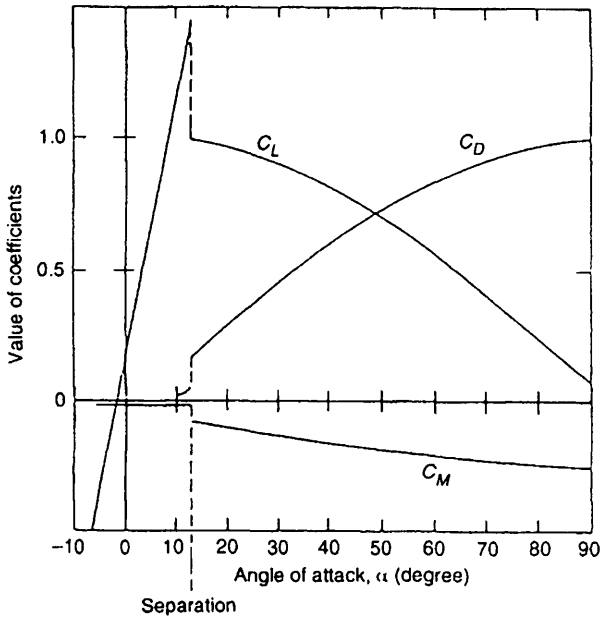


Fig. 2.3 Variation of Aerofoil Coefficients with Change in Angle of Attack

C_L, C_D and C_M represent the coefficients of lift, drag and pitching moment, respectively, experienced by a rotating blade. In this example, the angle of separation is approximately 13°, at this point the airflow over the blade is disrupted due to flow separation. Manipulating the characteristics of Fig. 2.3 lead to power regulation.

(a) Stall Regulation

Stall regulated rotors are blades that are designed to induce flow separation above a certain wind speed. Consider the cross sectional representation of a rotating blade and the velocity components on the blade shown in Fig. 2.4.

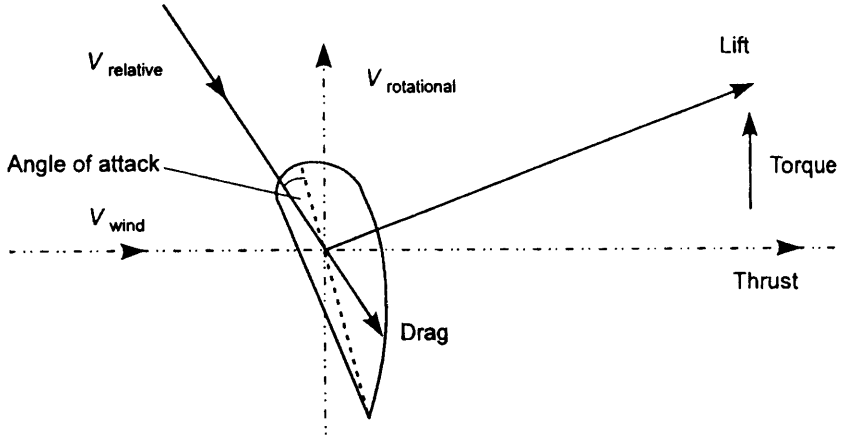


Fig. 2.4 Cross Sectional Profile of a Rotating Blade

The blade is rigidly fixed to the rotor and for a 'constant speed' WECS, the rotational velocity of the blade is practically constant. Assuming that the wind speed is below rated and the angle of attack is such that the air flow over the blade is not disrupted, the lift force will be greater than the drag force(i.e. below 13° on Fig. 2.3). This will result in a torque on the WECS shaft, as discussed previously. If the wind speed increases above rated, the angle between it and the relative wind velocity becomes smaller, due to the constant speed of rotation. This will inevitably, increase the angle of attack making it greater than the angle of separation, and leading to stall. The torque on the rotor will, therefore, be limited.

(b) Pitch Regulated

Whereas a stall regulated WECS relies on an increase in the angle of attack to induce stall, a pitch regulated WECS relies on a reduction in the angle of attack to limit the amount of lift on a blade [2.7]. This can be explained by consulting Fig. 2.3 and observing the lift characteristic from 0° up to the separation angle. The whole blade or part of it (i.e. full span or part span) is physically rotated to make the required reduction and, once again, limit the amount of torque produced.

Altering the aerodynamics during regulation affects the properties of the torque coefficient C_q. With stall regulation, the effect of stall is inherent in the C_q - λ data,

therefore, no extra provision is required in the modelling. Effects such as stall delay and dynamic stall are beyond the scope of this investigation and are not considered, but it should be noted that the model is flexible enough for these effects to be added if and when required. With pitch regulation, the values of C_q are effected by the pitch modifications. C_q is therefore, a function of both λ and the pitch angle, β. This relationship is usually modelled as a two dimensional 'look-up' table, the inputs being λ and β.

Since pitch regulation relies on the rotation of the blade position, the mechanism required to do this needs to be included in any simulation. This would have to include any delays in the measurement system used to assess the desired rotor position, along with any inherent delays and mechanism limitations.

2.1.2 Additional Aerodynamic Effects

The aerodynamic effects experienced by a WECS in a wind field are fairly complex. The profile of the wind varies both spatially and temporally as shown in Fig. 2.5. As the blades rotate in this non-uniform wind profile they effectively sample the wind at different points. This is known as rotational sampling. Additionally, variations in the wind stream, both upstream and downstream and the disruptive presence of the WECS in the stream, can cause additional torque and forces to be induced on the rotor.

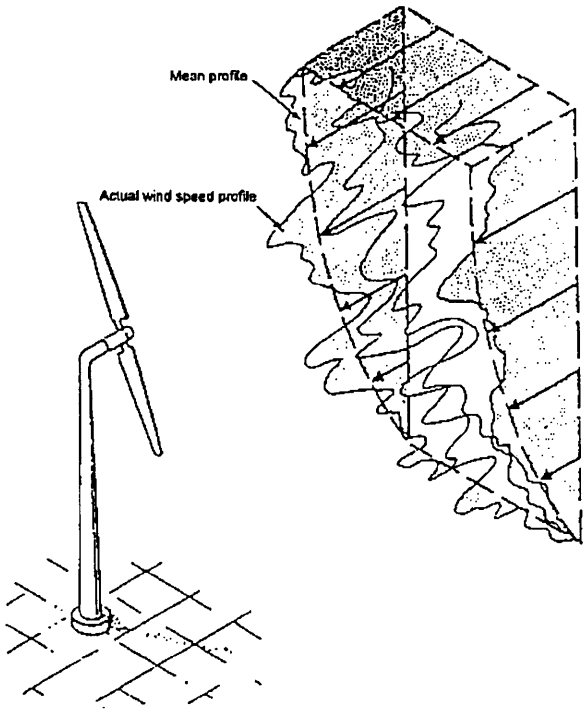


Fig 2.5 Typical Wind Velocity Profile

A number of these effects can cause structural problems for the WECS and are important for analysing the structural dynamics and hence fatigue life of a WECS [2.8]. From the drive train perspective, some of these dynamics have the effect of negating each other, such as equal but opposite oscillatory motion on the blades, resulting in a net motion of zero [2.8]. Since the project was concerned with simulation of the dynamics of the drive train, these effects are not discussed. Further details can be found in references [2.8] to [2.11]. This section will first detail the additional aerodynamics relevant to a WECS and then describe how they can be modelled.

(a) Wind Shear

One effect that contributes to rotational sampling is wind shear. It can be seen in Fig 2.5 that there is an increase of wind velocity with height. This is due to the effects of friction between the air and the surface of the ground. This 'boundary layer effect' decreases with height, until eventually it becomes zero (around 2km [2.3]). Turbulence is also present in

the profile. As the rotor blades rotate, each blade is effected by a different part of the profile (due to the shear). Each blade of the rotor, therefore, sees the profile as a periodical function of time. If the shear was linear across the rotor disk, the effect upon the induced torque would be the addition of a single harmonic at the blade passing frequency and be predominately deterministic. Because the profile is non-linear, due to the presence of turbulence, the effect tends to be stochastic having a bandwidth centred at the blade passing frequency [2.12].

The effect of wind shear becomes more prominent with increase in the radius of the rotor. The effects are negligible for machines with radii of less than 5m [2.3].

(b) Tower Shadow

The presence of the WECS support tower greatly affects the wind profile as it offers resistance to the flow past it and affects both the upstream and downstream profiles. The local effect is to slow the speed of the wind in front of the tower. As each blade of the WECS passes the tower, rotational sampling of this slower wind speed occurs. This 'tower shadow', like the wind shear, causes a torque to be induced on the drive train at frequencies centred at the blade passing frequency. Tower shadow is more pronounced for solid tower machines compared with lattice tower. Aerodynamically well designed lattice towers can alleviate the effects of tower shadow [2.13].

(c) Yaw Misalignment

The effects of yaw misalignment are very similar to the effects of wind shear. WECS use devices to direct the rotor into the direction of the prevailing wind. Because the wind direction is never constant, and the directional devices tend to have a delay in adapting to the new directions, the WECS can become misaligned. As with wind shear, where the blades sample the vertical differences in wind speed, yaw misalignment results in the turbine blades sampling the horizontal difference in wind speed. This, again results in

torque being induced in the drive train at frequencies centred at the blade passing frequency.

(d) Induced Torque at the Rotor Rotational Frequency

In addition to induced torques at the blade passing frequency, analysis of the torque spectrum of a WECS shows a predominant spectral peak at the rotor rotational frequency [2.5]. This effect is due to any asymmetry in the rotor blades and the effects of imbalance in aerodynamic and gravitational rotor loads. The effect of this induced torque is predominately deterministic and sinusoidal.

(e) Induction Lag

It was mentioned previously, that a WECS is influenced by the condition of both the upstream and downstream wind profiles [2.2]. The presence of a WECS in a wind flow can cause a wake rotation [2.2] and [2.12]. When the wind speed or the pitch angle of a blade changes (with pitch regulation), the airflow in the vicinity of the blade and downstream from the blade adjust to suit (see Fig. 2.1) . The complex wake cannot immediately change and it takes some time for the wind profile a distance downstream from the turbine to adjust in the appropriate way [2.14]. This effect is known as an induction lag.

During this lag, the induced aerodynamic torque is greater than expected. Experimental investigation into this phenomenon, indicate that the larger values of torque were found in the mid-frequency range (relative to the response of a WECS), with little change at low frequency.

The influence of induction lag is more prominent with pitch regulation systems. Induction lag can be negligible with stall regulated WECS [2.15].

(f) Tower Movement

Most of the WECS tower and hub movement has little affect on the induced torque on the drive train. One effect that can be influential is the fore-aft tower movement, z_T. The effect of the movement is to modify the effective wind speed by adding or subtracting z_T [2.15].

2.1.3 Modelling the Aerodynamic Effects

All the aerodynamic effects have to be modelled in a suitable manner for inclusion in any WECS simulation. Certain techniques can be used to 'lump' some of the afore mentioned aerodynamic effects without loss in accuracy.

(a) Simulating and Modelling the Wind

Realistic wind input to a simulation model, can be modelled in a number of ways. One method is to simulate a wind profile by imposing filtered, white noise on top of a measured mean wind speed to produce a Von Karman spectrum [2.15]. Another novel method is to generate a frequency spectra appropriate to real wind using a variable Weibull distribution [2.16]. With this particular project, a measured wind site data series was used. The wind speed was measured by an anemometer and is referred to as a 'point wind speed'

With either real or simulated wind data, it necessary to establish the effective wind speed to validate the use of the aerodynamic torque equation (Eqn. 2.11). This can be achieved by filtering the point wind speed data with a spatial filter. This compensates for the fact that the wind turbulence experienced by the rotor is only a local effect and only the spatial average wind speed variation influences the aerodynamic driving torque [2.14]. Without the filter there is too much power at higher frequencies in the simulation [2.17].

The transfer function, G(s), of a spatial filter, relating the point spectrum to the effective wind speed is:

$$G(s) = \frac{(\sqrt{2} + \beta s)}{(\sqrt{2} + \beta\sqrt{as})(1 + \frac{\beta}{\sqrt{as}})} \tag{2.14}$$

$$\beta = \frac{1.3R}{\bar{V}} \tag{2.15}$$

where a = 0.55 and $\bar{V}$ is the mean wind speed.

The tower motion, z_T, is simply added or subtracted from the effective wind speed, assuming z_T is known.

(b) Simulating and Modelling the Sampling Effects and Induction Lag

The effects of wind shear, tower shadow and yaw misalignment can be modelled by superimposing a torque component, due to the rotational sampling T_{rot}, on to the aerodynamic torque calculated from the effective wind speed (Eqn. 2.11). The effects due to sampling have both a deterministic part and a stochastic part. For a three bladed rotor, the third harmonic, due to the sampling effects of the blades, tends to be more prominent and more stochastic.

Reference [2.13] indicates that the spectral peak at the rotor frequency can be modelled using a simple sinusoid, while the effect of rotational sampling can be modelled as:

$$T_{rot} = (A + \varepsilon_1(t))\cos 3\omega t + (A + \varepsilon_2(t))\sin 3\omega t \tag{2.16}$$

where A is a constant, contributing to the deterministic component of the rotational sampling, while $\varepsilon_1(t)$ and $\varepsilon_2(t)$ are stochastic processes contributing the stochastic component. $\varepsilon_1(t)$ and $\varepsilon_2(t)$ are independent white noise processes coloured by first-order filters.

The induction lag is modelled using a lead-lag filter:

$$\text{Induction Lag} = \frac{(1+sa)}{(1+sb)} \tag{2.17}$$

where a > b and their values are found by comparing the simulated low speed shaft torque with the measured torque from a WECS, assuming that the appropriate measurements are available [2.17] and [2.18]. The value of b is usually less than 10, corresponding to the time constant, while the value of a is usually less than 15, depending on the mid frequency gain and the time constant [2.17].

2.2 Drive Train Dynamics

The modelling of the drive train dynamics involves the description of the WECS mechanical system and the interaction of the individual components of that system. Fig 2.6 shows the typical drive train of a 'constant speed' WECS. The drive train of a variable speed WECS may or may not have a similar construction, depending on the generator used and any power conditioning used.

The rotor is connected to a low speed shaft (LSS). Aerodynamic torque from the wind is transferred to the gearbox via the LSS and the LSS bearings are used to withstand the axial and radial loads transmitted by the rotor [2.19]. The flexible coupling at the end of the LSS, is sometimes used to compensate for misalignment problems on the shaft. Similarly, coupling is sometimes used on high speed shaft (HSS), which connects the generator to the gearbox.

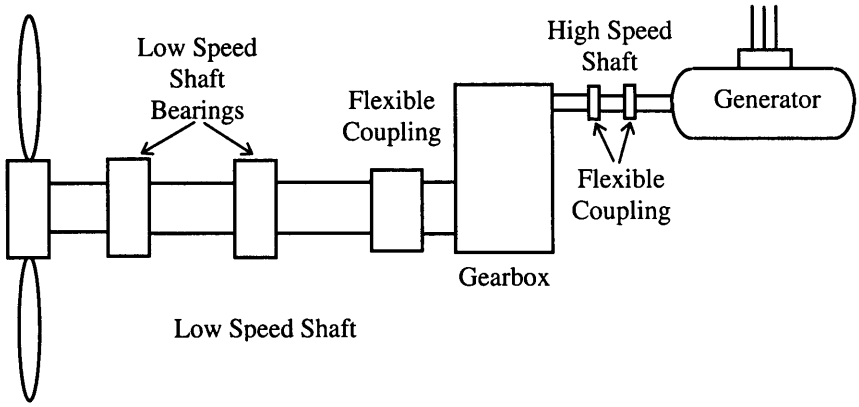


Fig 2.6 Typical Constant Speed WECS Drive Train and Generator

In addition to the drive train, the dynamic modes of tower, blades and hub and their interaction with the drive train, have to be considered. This can lead to a fairly complex, non-linear set of equations. Ideally, it is hoped that these complex dynamics could be represented by a simple model, which would accurately describe the approximate dynamic performance of the drive train. Additionally, this simple model would be suitable for all 'constant speed' wind turbines. This section evaluates the modelling of constant speed WECS, performed by Leithead and Rogers [2.8] and Sheinman and Rosen [2.18], in order to produce simple models.

2.2.1 Modelling the Drive Train Dynamics

Modelling the system shown in Fig 2.6, can be realised by representing the drive train as a chain of rotating inertias connected by torsional springs.

Most of the damping in the system is quite small and can be 'lumped' with the inertias and/or the stiffness components. The major source of damping in the drive train, is that associated with the induction machine connected to the grid. This may not be the case with a variable speed arrangement using a synchronous machine.

It is important to understand the interaction of the individual parts of the drive train, the tower, hub and blades of the WECS. This can be done by considering each part of the

drive train separately, and lumping the dynamics (inertia, damping and stiffness) of individual components together, to create a simplified model. The model is derived by assessing each part of the drive train and its interaction with the remaining parts. Dynamic equations of motion are established which eventually, through linearisation about an equilibrium point, allow some terms to be neglected. A model of a variable speed WECS would be subject to some modifications. The larger speed range could excite some structural dynamics not effected by constant speed operation [2.19]. Therefore, some of the assumptions stated for the constant speed WECS may not be valid for the variable speed WECS. The simplifications are used to speed up the development of the simulator but if the simulation software is flexible enough, any complexities such as non-linearities, can be included in the model at a later date.

The relationships between drive train components can be further simplified if some valid assumptions are made about individual components. For example, both the LSS and the HSS are assumed massless. This is valid if the shafts are uniform and their inertias are equally divided between the adjacent drive train components. Fig 2.7 shows a generalised, simple mechanical model of a constant speed drive train.

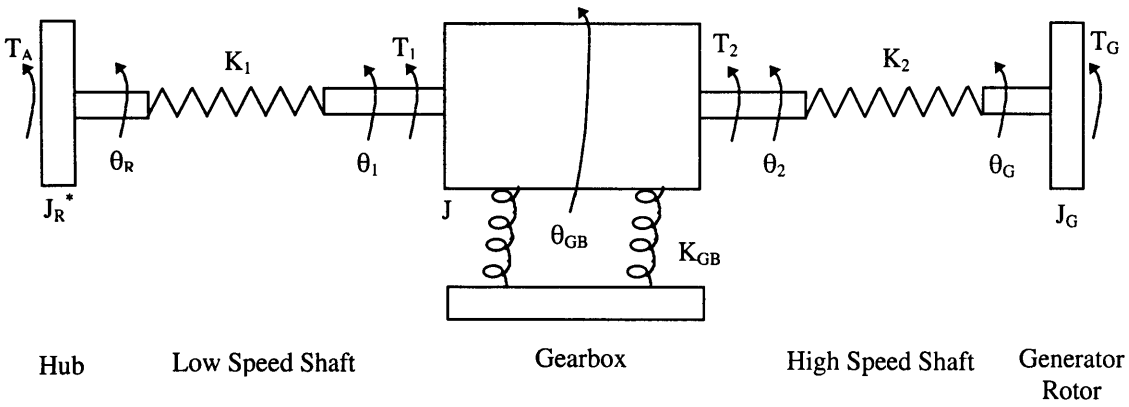


Fig 2.7 Simplified Model of a WECS Drive Train

T_A and T_G are the aerodynamic torque and the torque acting on the generator respectively, while T_1 and T_2 are torques acting on the gearbox. K_1, K_2 and K_{GB} represent

the stiffness of the LSS, the HSS and the gearbox respectively. θ_R, θ_1, θ_{GB}, θ_2, and θ_G represent the rotational position of the rotor, LSS at the gearbox, HSS at the gearbox and the generator respectively. J_R^* and J_G are the effective rotor inertia and generator inertia respectively. The derivation of the model in Fig. 2.7 is summarised in the following sections.

(a) Rotor, Hub and Low Speed Shaft Dynamics

J_R^*, the effective inertia of the rotor, is a 'lumped' parameter including the dynamics of both the rotor and the hub as follows:

$$J_R^* = J_R(1+\frac{J_H}{J_R}\frac{K_R}{(K_R+K_H)}) \tag{2.18}$$

where J_R is the rotor inertia, the subscript H denotes the hub properties and K denotes the stiffness. J_H includes half the inertia of the LSS, as mentioned previously.

The LSS stiffness, K_1, is another lumped parameter:

$$K_1 = \frac{K_R K_H}{(K_R+K_H)}(1+\frac{K_R}{K_R+K_H}\frac{J_H}{J_R}) \tag{2.19}$$

Both Eqn. (2.18) and Eqn.(2.19) are lumped parameters derived by analysis of the combined dynamics of the rotor and hub of the constant speed WECS [2.8].

(b) The Gearbox Dynamics

The gearbox shown in Fig. 2.7 can be either rigidly mounted or mounted on a compliant suspension. Fig. 2.8 shows the model of a rigidly mounted gearbox. N is the gearbox ratio while J_s is the inertia of the gearbox referred to the LSS.

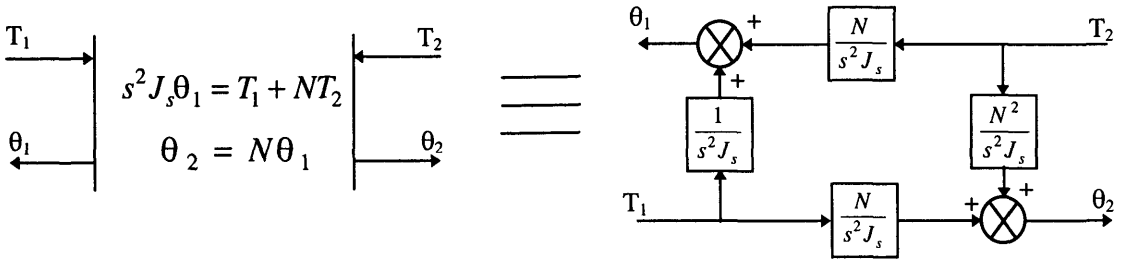


Fig. 2.8 Model of a Rigidly Mounted Gearbox

The compliant model is similar to the rigidly mounted model. The difference is that the relative movement of the gearbox is considered and the different stages of the gearbox have to be modelled. Fig. 2.9 shows the eventual model of the compliantly mounted gearbox.

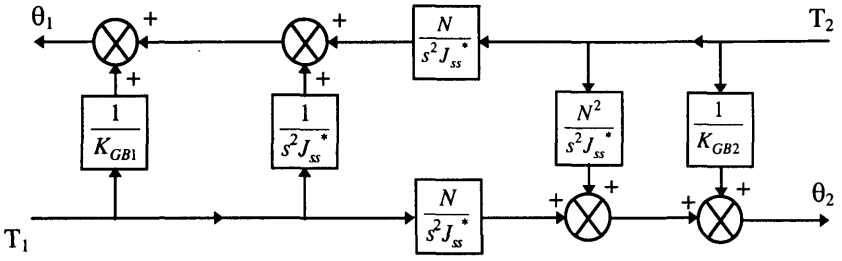


Fig. 2.9 Model of a Compliantly Mounted Gearbox

J_{ss}^* is the lumped gearbox inertia referred to the LSS, $\frac{1}{K_{GB1}}$ and $\frac{1}{K_{GB2}}$ are the lumped input and output compliance respectively. These parameters are related to the individual inertias and damping coefficients of the gearbox stages.

(c) Overall Drive Train Dynamics

The lumped gearbox dynamics of Fig. 2.8 and Fig. 2.9 are combined with the dynamics of the remaining drive train shown in Fig. 2.7 in order to establish a model of the overall drive train. Fig. 2.10 shows the model of a drive train with a compliantly mounted gearbox.

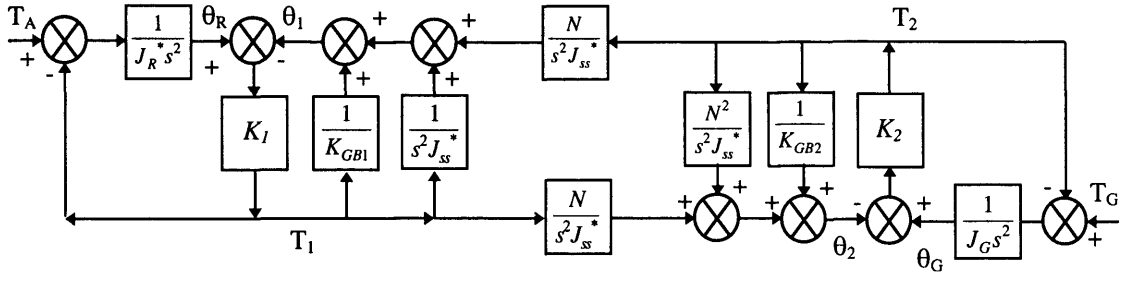


Fig. 2.10 Overall Drive Train Model with a Compliantly Mounted Gearbox

Further reductions can be obtained by additional 'lumping' of drive train components. In the case of a drive train coupled to an induction generator, high frequency dynamics may be neglected. Fig. 2.11 shows the simplified model of a constant speed WECS.

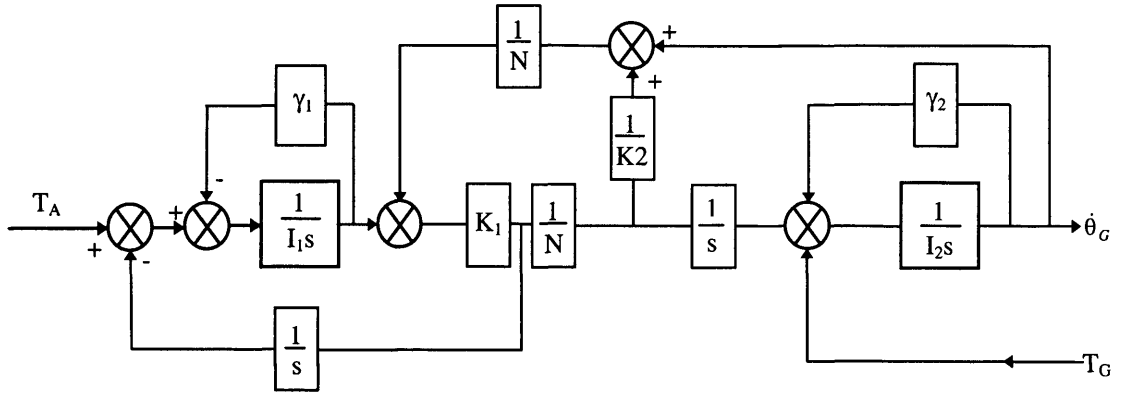


Fig 2.11 Simplified Model of a Constant Speed WECS

γ_1 and γ_2 are the LSS and HSS damping constants, respectively, while I_1 and I_2 are the lumped rotor and generator inertias respectively. The symbol I has been used for the drive train inertia terms in keeping with the said use in the reference [2.8].

The lumped parameters relate to the previous dynamics as follows:

$$I_1 = J_R^* + \frac{K_1}{K_1 + N^2 K_2} J_{ss}^* \tag{2.20}$$

$$I_2 = J_G + \frac{K_2}{K_1 + N^2 K_2} J_{ss}^* \tag{2.21}$$

This model can be used for both compliantly and rigidly mounted gearboxes and is valid if the effective LSS stiffness, K_1, for the compliant model, is redefined as [2.20]:

$$\frac{1}{K_1} = \frac{1}{K_1} + \frac{(N-1)^2}{N^2(I_3 s^2 + \gamma_3 s + K_3)} \tag{2.22}$$

where I_3 is the lumped gearbox inertia, γ_3 is the gearbox damping and K_3 is the stiffness of the gearbox mounting.

2.2.2 Modelling the Induction Generator Dynamics

A simple first order linear description of an IM is defined in reference [2.20]. The justification for using the simplified model is that the IM is connected to an strong grid (constant voltage and frequency) and it is stated that : 'Therefore, the electrical dynamics of the generator are enclosed in a strong feedback loop, minimising any difference between the dynamics of the simple model and a more complex, non-linear description'.

The first order model can be defined as:

$$\tau \dot{T}_G + T_G = D_e(\omega_R - \frac{\omega_S}{p}) \tag{2.23}$$

where τ is defined as the time constant of the machine, D_e is the gradient of the torque speed curve of the IM, ω_R is the rotor speed, ω_S is the grid frequency and p is the number of pole pairs of the IM. T_G is the generator reaction torque.

D_e and τ can be calculated using the equations of Appendix 2a, taken from reference [2.20].

2.2.3 Drive Train Model Validation

Combining Eqn. (2.23) with Fig. 2.11 allows the transfer function between the aerodynamic torque and the generator reaction torque to be established. Comparing the Bode plots of the simplified model with a more complete linear model shows that there is an agreement at both high and low frequencies for all wind speeds [2.20]. Further comparisons with the simplified model, and the response from an actual large scale WECS, show a favourable response at both high and low frequencies.

Earlier modelling by Wilkie et al [2.17] proposed and verified a comparison of the power spectral density of the LSS to validate any WECS model. The advantage of this form of comparison is that the response of any simulation would include the effects of the aerodynamic modelling, described at the beginning of this chapter, and not just the transfer function of the drive train and generator. Further more, the aerodynamic effects, such as the addition of rotational sampling and an induction lag, can be added in a modular fashion and any improvement in the response of the simulator observed accordingly. The result of this comparison showed that the simplified model was, again, a suitable representation of the WECS.

2.3 Summary

A wind energy conversion system converts the kinetic energy of the wind to a rotational, mechanical force. To develop a fully dynamic model of a WECS, an understanding of the aerodynamic properties of the wind, experienced by the device, is required. The wind profile varies both temporally and spatially, resulting in a complex, stochastic, driving function for the WECS. Additionally, the presence of the WECS in a wind profile produces additional aerodynamics which are experienced by the drive train of the WECS.

The components of the WECS drive train have to be modelled accurately in order to establish the dynamic interaction between each component and the aerodynamics. Previous research, assessing the spectral response of the simplified WECS model, shows that dynamics of complex systems can be reduced, with suitable 'lumping' of parameters and the neglecting of others, to simplified models with no deteriation in system response.

This chapter has detailed the dynamics of both the wind and the WECS drive train, which will result in the development of an accurate model. The model can be developed in the selected software package, and then validated using recognised techniques.

Chapter 3 Selection of Simulation Hardware

Chapter 2 detailed the development of a model of a dynamic, constant speed WECS. The aim of the project is to manipulate the model in software and provide real-time control of a grid-connected, induction machine (IM), as if it was being driven by the drive train of a WECS. This chapter details the development of the hardware required to realise the project objectives. Details include the design of a test-bed and a description of the operation of a DC drive, which will provide a real-time communications link to the simulation software.

3.1 Selection of Simulator Generator

The development of the WECS model in Chapter 2 requires a description of the particular WECS which is to be simulated. Details required for the description include blade size and characteristics and gearbox ratio. The data available for simulation (see Chapter 5) is for two WECS with ratings of 45kW and 330kW. The size of the hardware simulator would be limited due to the constraints of the laboratory and availability of machine. Within the lab, three phase supplies for both 415V and 250V are available with current ratings of 32A and 16A per phase respectively. An IM would have to be selected to be within these ratings therefore, simulation of a WECS at a higher rating would require appropriate scaling of mechanical power, in the simulation software, to drive the smaller machine. The selected machine, for both rating and availability, is a Brook Crompton Parkinson 11kW two pole device, rated at 2925rpm.

The proposed design of the simulator is such that the WECS model provides the real-time, mechanical power at the shaft of an IM connected to the grid. Simulation of the mechanical power, or prime mover, should be characteristic of the dynamic action of a WECS drive train under the influence of the wind. Software will simulate the 'front end' of the WECS, namely the aerodynamic torque and the dynamics of the low speed shaft and gearbox. The hardware arrangement therefore, has to include a device which can act

as the WECS's high speed shaft, controlled by the simulated 'front end', and drive the IM.

3.2 Simulation of the Prime Mover

A suitable device to act as the prime mover and provide the mechanical power to the shaft of the IM, would have to be able to respond fast enough to mimic the actions of the 'front end' of the simulator and be able to deliver the required mechanical power to the selected IM.

A 15kW DC motor was available in the lab, and it is anticipated that this could be used as a suitable prime mover and hence, reduce development time and costs. To justify its inclusion in the hardware simulator it is necessary to investigate any comparable alternatives

Pneumatic and hydraulic motors are not considered since a pneumatic or hydraulic source was not readily available. This suggests that an electrical source would be suitable since both a supply and a number of machines are available in the department. The next step is to assess the types of electrical motor available and establish their suitability for inclusion in the simulator.

Three electrical motor arrangements are considered, namely the AC induction motor, the AC synchronous motor and the DC motor. It has to be established which machine would be suitable for the simulator. The development time and costs of the selected device also have to be considered.

Selection of a suitable type of motor for the simulator, is dependent on the controllability of the mechanical power delivered by that motor. In other words are the torque and/or speed of the motor easily controlled ? A comparison of the three electrical motor types is required.

3.2.1. AC Induction and Synchronous Motors

The IM, and its use as a constant speed generator was introduced in Chapter 1. Induced voltages from a rotating magnetic field causes current, and therefore a flux, to be produced in the rotor. This causes an interaction between the two magnetic fields to produce a torque and hence, rotation. The speed of the rotation is dependent on the voltage and frequency of the supply to the stator, which is generally fixed.

Speed and torque control can be achieved if either the line voltage or frequency are changed. Control can be quite complex, susceptible to line fluctuations and the relationship is non-linear (e.g. torque developed is proportional to the square of the terminal voltage) [3.1].

The AC synchronous motor (ACSM) differs from the induction motor in that the flux in the machine is produced by a DC fed field winding on the rotor. One of the characteristics of the ACSM is that it is not self-starting. Starting can occur by either using a frequency converter to reduce the initial frequency of the stator flux, or by starting the motor as an induction motor. This can be achieved by leaving the rotor unexcited. The costs of the frequency converter can be quite prohibitive, while the latter option involves the development of hardware to 'switch-in' the rotor excitation at synchronisation.

Like the induction motor, the speed of an ACSM can be controlled by changing the frequency of the power supply. Common methods for speed control involve the use of either inverters or cycloconverters to control the supply frequency to the motor [3.1]. The torque of the motor can be controlled by adjusting the terminal voltage or the field current of the motor. Unfortunately, neither relationship is linear.

A number of commercial AC drives are available which provide variable voltage, variable frequency supplies. The drives can operate under both torque and speed control

using various control schemes. Examples include voltage-fed inverter drive using slip control to control the torque of an induction motor [3.2].

Unfortunately, no AC drives are available in the department and the budget of the project precludes the purchase of one.

3.2.2 DC Motor

In a DC motor the (stationary) flux is again produced by a DC field winding. The flux interaction between the stationary field and the armature currents in the rotor, produces torque and rotational motion. The advantage of the DC motor over the AC motors is that both the speed and the torque can be controlled relatively easily.

The developed torque is proportional to the field flux and the armature current while the speed is proportional to the back-emf and the field flux. If the field flux is kept constant, torque and speed can be controlled by controlling the armature current and voltage, respectively.

As mentioned earlier, a DC motor is available in the department. It is a Mawdsley's Ltd., 15kW, 3000rpm machine. The relative ease of control of the DC motor compared with the non-linear AC motors favours the use of a DC motor, reducing the development time and costs of using an AC arrangement.

3.3 Test-Bed Development

Following the selection of the motor and generator set, it is necessary to decide on an appropriate coupling between the two devices. Two options are considered for the test-bed arrangement. Firstly, an 'inline' coupling as shown in Fig. 3.1, is examined.

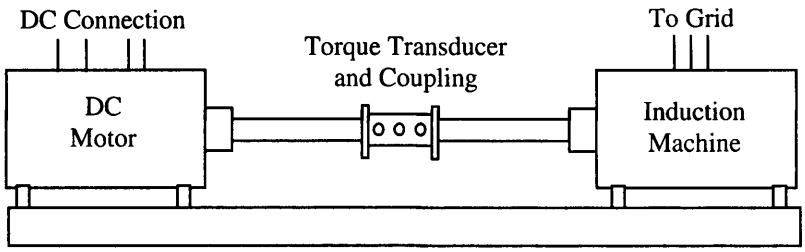


Fig 3.1 'In-Line' Coupling with Torque Transducer

The torque transducer can be used to provide a measurement of shaft torque if the simulation requires it. The disadvantages of using this arrangement will be the difficulty of changing generators, due to variable shaft heights etc. This would be necessary, for example, if the simulator is to be used for testing variable speed operation using a synchronous generator, or testing a multi-poled induction machine used in large hydro energy converters.

To ensure that the 'test-bed' can accommodate various generators, a 'side-by-side' arrangement is preferred, using pulleys and a belt. Fig 3.2 shows the suggested design.

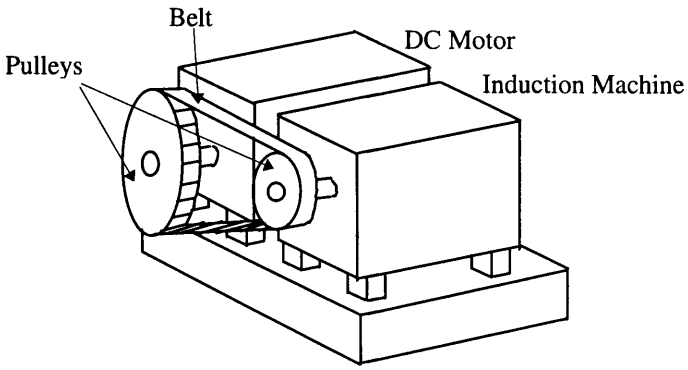


Fig 3.2 'Side-by-Side' Coupling

An estimation of the shaft torque could be acquired by placing the generator on a hinged plate mounted on load cells. This method would increase the complexity of the rig but since the measurement of the torque is not absolutely essential, the development time could not be justified (see Chapter 5).

The use of different size pulleys means that the 'test-bed' is, therefore, not restricted to the speed characteristics of the DC motor, allowing the use of IM's with different pole numbers. For example, the Brook Crompton is a two pole machine operating at approximately, 3000rpm. Although the DC motor can operate at 3000rpm, it is at its maximum operating condition and will require field weakening. This is undesirable, as will be discussed later. The DC motor can operate at 1500rpm without field weakening, and drive the 3000 rpm IM if pulleys with a 2:1 ratio, respectively, are placed on the machine.

For the IM and DC motor initially selected, the size of the pulley's and the length of belt have to be calculated to allow for rated power transfer.

3.3.1 Calculation of Pulleys, Belt and 'Test-bed' Sizes

There is available, within the department, a Fenner 96H (38.1mm width, 388.08mm diameter) pulley for a timing drive. It is anticipated that this could be used as pulley for the DC motor and hence reduce costs. For this to be acceptable, a pulley half this size (to give the desired speed ratio of 1:2) has to be selected for the IM, in this case a Fenner 48H, 97 mm pulley [3.3]. Once this initial procedure is complete, confirmation that the selected pulleys and timing belt could function with the power rating at the operating speed, has to be obtained. For this to be achieved, both the length and width of the belt are calculated from the information given in reference [3.3]. Appendix 3a details the required calculations for the belt as well as the measurements for the base of the test-bed.

3.4 Operation and Control of a DC Motor

Since the software simulation and drive will provide a demand signal to the motor, it is necessary to include details on the operation and control of a DC motor and understand any limitations this option offers.

It was mentioned earlier that a DC motor is doubly excited. A DC field is located on the stator while the rotor contains the armature. Fig. 3.3 shows the basic configuration.

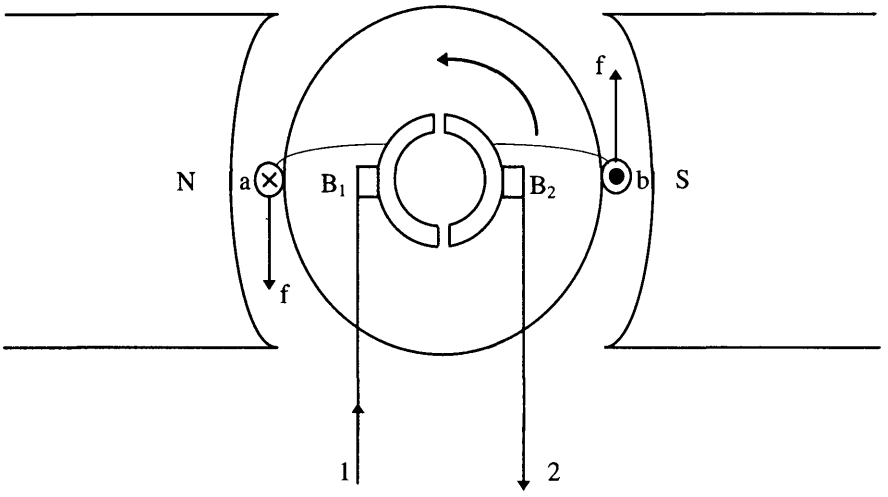


Fig. 3.3 Basic Configuration of a DC machine

The figure shows a two-pole device with one turn, a b, on the rotor with a current, i_a, flowing through it. B_1 and B_2 are brushes in contact with two commutator segments connected to the conductors. The field on the stator is excited by the current I_f, giving rise to a flux Φ. Φ is directly proportional to I_f up to saturation.

When a supply is connected to the armature, i_a produces a flux which interacts with the field flux and causes a force, f, to be experienced by the armature conductors. In Fig. 3.3, the force experienced on the rotor causes an anti-clockwise rotation due to the polarity of

the armature supply voltage. Direction of rotation can be altered by changing the polarity of either the field or armature supply.

3.4.1 Development of Motor Torque

The force experienced by an armature conductor in a magnetic field (known as Lorentz force), is defined as:

$$f = Bli_a \tag{3.1}$$

where B is the magnetic flux density (T) and l is the length of a conductor. In practice, many turns are placed on the armature, and the force on each conductor can be related to the total supply current, I_a:

$$f = Bl\frac{I_a}{a} \tag{3.2}$$

where a is the number of parallel paths on the armature. It follows that the average torque, $\overline{T}$, developed by a conductor is:

$$\overline{T} = \overline{B}l\frac{I_a}{a}r \tag{3.3}$$

where r is the radius of the armature. The average flux density can be found from the pole flux, Φ, and the circumferencial area A, which gives:

$$\overline{B} = \frac{\Phi}{A} = \frac{\Phi p}{2\pi rl} \tag{3.4}$$

where p is the number of field poles. Hence:

$$\overline{T} = \frac{\Phi p I_a}{2\pi a} \tag{3.5}$$

All conductors develop torque in the same direction and contribute to the average torque, therefore the total torque developed is:

$$T = 2N\overline{T} \tag{3.6}$$

where N is the number of armature turns.

From substitution, the developed torque in relation to the armature current and field flux is:

$$T = \frac{N\Phi p}{\pi a} I_a = K_a \Phi I_a \tag{3.7}$$

where K_a is defined as the motor or armature constant.

It can be seen from Eqn. (3.7) that if the field flux is kept constant, the developed torque is directly proportional to the armature current.

3.4.2 Speed Control of a DC Motor

When a conductor is moving linearly in a magnetic field, the armature conductors 'cut' lines of flux, inducing voltage in the conductors. The induced voltage, e, is proportional to the velocity of the conductor, v defined as:

$$e = Blv \tag{3.8}$$

When motoring, the armature conductors rotate in the magnetic field, causing a voltage to be induced in the armature. This 'back emf' opposes the applied current I_a. The average induced voltage on a turn, rotating in a field is:

$$\bar{e}_t = 2\bar{B}l\omega_m r \tag{3.9}$$

where ω_m is the mechanical rotational speed. Substituting Φ for B gives:

$$\bar{e}_t = \frac{\Phi p}{\pi}\omega_m \tag{3.10}$$

The average terminal voltage E_a, for all the turns connected in series is:

$$E_a = \frac{N}{a}\bar{e}_t \tag{3.11}$$

substituting for $\bar{e}_t$ gives:

$$E_a = \frac{Np}{\pi a}\Phi\omega_m \tag{3.12}$$

from eqn (3.7):

$$E_a = K_a\Phi\omega_m \text{ or } \omega_m = \frac{E_a}{K_a\Phi} \tag{3.13}$$

Therefore, the 'back emf' of the motor is directly proportional to the mechanical speed of the motor, if the field flux is kept constant.

Speed control can also be realised by adjusting the field flux while maintaining a constant voltage. This can become necessary if the rated voltage of the motor has been met but an

increase of speed is still required. Reducing the field flux, or field weakening, can increase the speed without increasing the voltage of the motor. This has the disadvantage of reducing the amount of mechanical torque developed (eqn 3.7). Additionally, varying the field current (flux) means that the relationship between the armature current and the torque is no longer linear.

3.4.3 DC Motor Field Connections

The field of a DC motor can be excited by connecting it as a separately excited machine, a shunt field winding or a series field winding. The shunt and the series terms refer to the mutual circuitry of the field and the armature. A shunt winding is the parallel connection of the field while a series winding connects the field in series with the armature. A separately excited winding uses a separate power source than that used for the armature circuit.

Each form of field winding has different torque-speed characteristics. A series motor, for example, characteristically can operate with a relatively high starting torque. For this project a separately excited DC motor is required. One of the major reasons for this choice is that both speed and torque operation can be more easily controlled if the field current (flux) is kept constant. From Eqn. (3.7) and Eqn. (3.13), speed and torque of the motor can be linearly controlled by controlling the 'back-emf' and armature current respectively.

3.4.4 Saturation and Armature Reaction in a DC Machine

It was stated earlier that the field flux in a DC motor increases linearly with increase in excitation. As the excitation increases, the reluctance of the iron core of the rotor becomes larger and saturation occurs. At saturation the flux increase is no longer linear with the field excitation.

Armature reaction is the effect caused when the field flux is disturbed by the flux caused by the armature current. The effect causes the flux under one half of a pole to be enhanced, while flux under the other half is diminished. If the 'enhanced' pole becomes saturated, then the overall effect is to reduce the total flux per pole. From the previous equations (Eqn. 3.7 and Eqn. 3.13), change of flux will obviously effect the operating speed and torque.

One method in reducing the armature reaction is to include compensation windings on the motor to reduce the flux caused by the armature. These windings tend to be expensive and therefore, tend only to be used on large machines.

Another way of reducing the armature reaction, is to limit the field current. This would reduce the chances of one of the poles being saturated, due to the armature reaction, and result in a reduction in flux.

3.4.5 Controlling the DC Motor from the Software Simulator

Once the 'test-bed' has been designed and constructed, it has to be decided how the DC motor, or wind energy prime mover, is to be controlled. The initial idea was that the WECS simulation in software would provide a variable armature voltage and current to the DC motor. The motor would, in turn, drive the grid connected IM and mimic the action of a WECS. The software, therefore, needs to be able to access the DC motor via a hardware interface.

Additionally, in order to successfully control the motor, the speed and/or torque status of the motor would need to be monitored by the software via a similar interface. This indicates the need for a Digital to Analogue Converter (DAC) and an Analogue to Digital Converter (ADC). However, the magnitude of the voltage/current required for a motor (e.g. up to 460V/49A for the Mawdsley [3.4]) would require additional signal manipulation (e.g. transformers or power electronics control). As mentioned earlier, the

costs of the project were limited, so it was hoped that additional equipment could be kept to a minimum and, therefore, also reduce the complexity of the simulator.

As an alternative to the transformer/power electronics arrangement, a DC motor drive system is available within the department and it was decided to make use of this. This particular drive was a Control Techniques Mentor II.

3.5 Control Techniques Mentor II DC Drive

The Mentor II drive allows speed and/or torque control of a DC motor and can operate as a four quadrant drive (i.e. reverse motoring to forward breaking). Operating parameters such as speed and/or torque operation, choice of form of speed feedback signal and maximum armature voltage can be set and varied via a menu system. These parameters are used to configure the Mentor II's dedicated on-board processor and control communications, security and other operational functions. Menus and parameters can be accessed via an on-board keypad or through a serial communications link.

3.5.1 The Mentor II Thyristor Operation

The Mentor II drive makes use of the relationships detailed in Eqn. (3.7) and Eqn. (3.13) to initiate closed loop control of the motor. The drive operates by controlling the firing angle of a thyristor bridge arrangements connected to both the armature and the field [3.5]. Fig. 3.4 shows the layout of the thyristor bridges which can be used for the four quadrant operation. The project does not require the reverse drive or braking options since it is not in the design of a WECS to change direction of rotation. This option, therefore, is simply disabled by setting the dedicated parameters in the Mentor II menu (parameters 04.15 and 04.16).

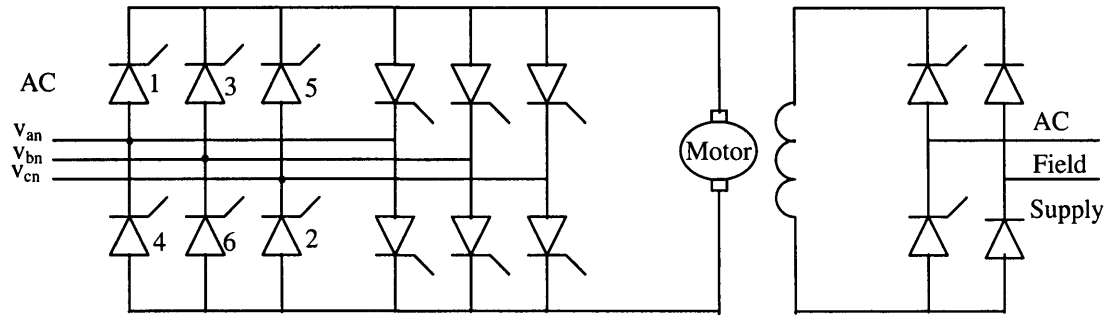


Fig. 3.4 Mentor II Parallel-Pair Thyristor Stack

The thyristors, 1 to 6, are used during forward motoring and are fired in numerical sequence (1, 3 and 5 during positive half cycle operation). The firing angle is calculated by the control action of the Mentor II. The reference for the firing angles are the zero crossing points of the line voltages (i.e. at $\pi/6$ of the phase voltage). Fig. 3.5 shows the example where the firing angle, α, is $\pi/6$ (30^{o}).

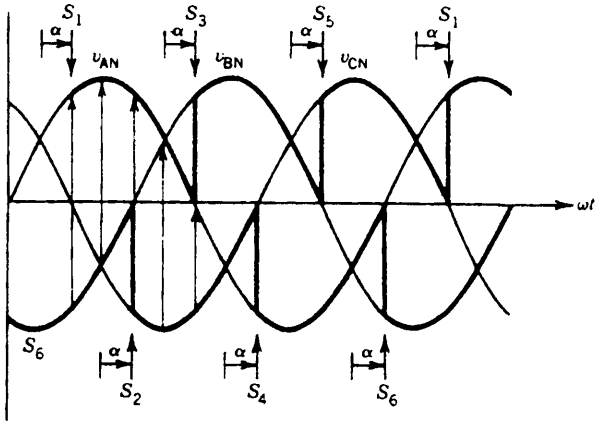


Fig. 3.5 Thyristor Voltage Waveforms at Firing Angle $\pi/6$

At $\omega t = \frac{\pi}{6} + \alpha$ thyristor 1 turns on. Before this, thyristor 6 is on. During the interval $(\frac{\pi}{6} + \alpha) < \omega t < (\frac{\pi}{6} + \alpha + \frac{\pi}{3})$, thyristors 1 and 6 conduct the output current and the motor terminals are connected to phase A and phase B and $v_o = v_{AB} = v_{AN} - v_{BN}$. The output

voltage v_o is the difference between the phase voltages v_{AN} and v_{BN}, as shown by the arrows in Fig. 3.5. This continues for each crossing point [3.1].

The average value of the output voltage is:

$$Vo = \frac{1}{2\pi / 3}\int_{\pi/6+\alpha}^{\pi/6+\alpha+\pi/3}(v_{AN} - v_{BN})d(\omega t) \tag{3.14}$$

$$= \frac{3\sqrt{6}}{\pi}V_p \cos\alpha \tag{3.15}$$

where V_p is the rms voltage.

Regenerative braking of the DC motor can be achieved by firing the thyristors in the second bridge in the correct sequence and deliver power back to the mains

3.5.2 Mentor II Control Menus

In order to ensure closed loop operation, it is necessary for the Mentor II to access the feedback variable, i.e. motor speed or motor torque/armature current. For speed control, the Mentor II allows this using three methods, namely, measurement of back emf, measurement taken from a tacho or measurement of a optical speed encoder. Under torque/current control, the Mentor's software monitors the motor's armature current via on board current transformers. Also, the Mentor II drive contains a number of on board ADC's and DAC's which can be used for additional signal measurement or demand.

Fig. 3.6 shows the basic control overview for speed control, including speed feedback selection and the interaction between menu parameters. It should be noted that the firing angle of the thyristor set uses current control regardless of whether the drive is operating under speed or torque control. Speed demands are converted into current demands using the on-board software.

The lettering on the left hand side of the diagram refers to the external connections of the Mentor II and are detailed in Appendix 3b. Table 3.1 details the Mentor II software menu. Each menu contains parameters which could be used for monitoring the status of the drive, or programming particular aspects of drive operation [3.5].

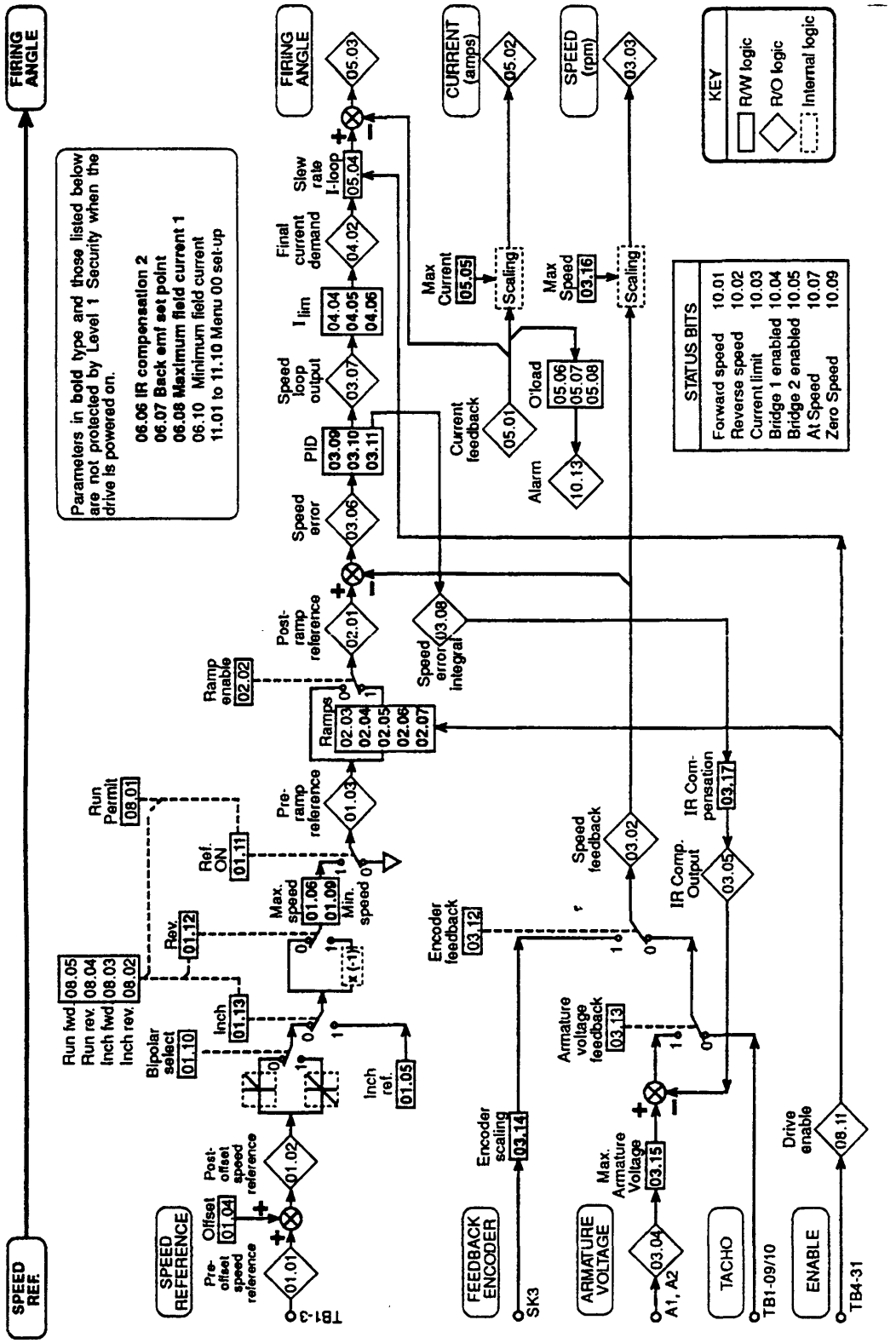


Fig. 3.6 Mentor II Control Overview

Menu	Description
00	User Menu - to give fast access to the most used parameters
01	Speed Reference - selection of source and limits
02	Acceleration and Deceleration Ramps
03	Speed Feedback Selection and Speed Loop
04	Current - selection and limits
05	Current Loop
06	Field Control
07	Analogue Inputs and Outputs
08	Logic Inputs
09	Status Outputs
10	Status Logic and Fault Information
11	Miscellaneous
12	Programmable Thresholds
13	Digital Lock
14	MD21 System Set up
15	Applications Menu 1
16	Applications Menu 2

Table 3.1 Mentor II Drive Menu List

Speed control, for example, could be accessed via menu 1. Various parameters could be adjusted to set speed demand and maximum speed etc. by manipulating the on-board keypad. Selection of ramps, to limit acceleration and deceleration, can be enabled in menu 2 to protect the motor from sudden, unforeseen changes in demand. Maximum, minimum and constant values of field current can be defined using menu 6.

3.5.3 Selection of Speed and Torque Control

As well as current selection, menu 4 also includes the quadrant enable (04.14 to 04.17) and selection of either speed control or torque control (04.12 and 04.13). Fig. 3.7 shows the interconnection of the relevant parameters.

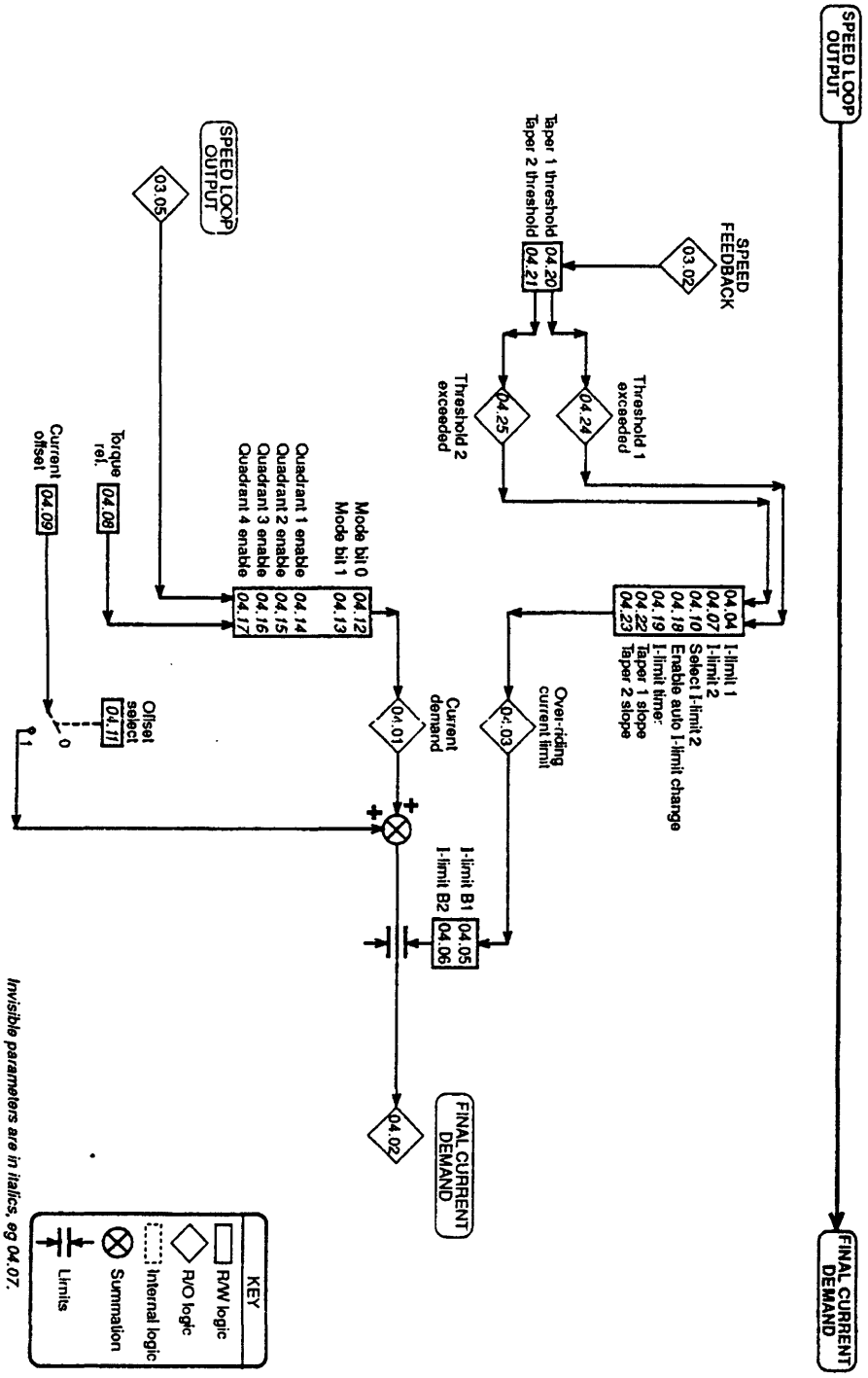


Fig. 3.7 Selection of Speed and Torque Control

Three forms of control are allowed with the Mentor II:

- Speed Control (04.12=0, 04.13=0)
- Basic Current or Torque Control (04.12=1, 04.13=0)
- Torque Control With Speed Override (04.12=0, 04.13=1)

With the speed control option, the conditioned demand is converted to a current demand and, subject to the current limits, becomes the final current demand. With the basic torque/current control, register 04.08 contains the current demand and, subject to the same limits as the first option, becomes the final current demand. The final option allows torque control up to the limit of the speed demand, again subject to the current limits.

The flexibility of the Mentor II indicates that it will be suitable for use with the hardware simulator as it allows both speed and torque control, with numerous options, and various ADC and DAC facilities.

It was mentioned that the Mentor II menus could be programmed manually or by use of a serial communications link. Manual programming of parameters is unrealistic for dynamic control of a WECS simulator, therefore the serial communication option has to be considered. The Mentor II makes use of a co-processor dedicated to serial communications. This unit is housed on the MD21 board.

3.6 MD21 Serial Communication Co-processor Board

The fact that the Mentor II microprocessor has to operate critical timing functions in real time imposes limitations on its ability to perform other duties such as serial communications [3.6]. A solution to this problem was to introduce the option of adding a second processor.

The MD21 is a single board microcomputer which can communicate with the main processor of the drive by means of a dual-port RAM. This provides memory which is accessible to both processors via their respective buses, allowing them to exchange

information without the need for interrupts or synchronous operation, and with minimal software overhead. The on-board serial port can operate under various communication protocols such as RS232 and RS422. Control of the operation of the serial communications is via The Mentor II's menu 14 (further details in Appendix 3b). Characteristics such as communication protocol, parity, baud rate and number of data bits can be set to suit the communications requirements.

3.6.1 Mentor II Operating System

The Mentor II Operating System (MOS) is the operating system of the MD21, as it is on the main processor, and exists as a package in the system EPROM.

The major function of the MOS on the MD21, is to interface it with the Mentor II and provide information, such as the amount of hardware that is available, and run application programmes stored in EPROM. Application programmes can be created by the user and stored in the user's EPROM. These programs are written in assembly language for both speed and concurrent operation with other programmes.

Two modes of operation are available with the MOS, namely 'Basic serial comms', which operates under interrupt control and supported handshaking, and 'Mentor II ANSI comms' which, similarly, operates under interrupt control, but without handshaking.

The Basic comms uses high level language instruction code, similar to BASIC. This allows ease of programming, but has the disadvantage of having a relatively slow processing time. Fig. 3.8 shows a timing diagram of a Basic comms. instruction to read data from one of the Mentor II registers. The C code used for this program is contained in Appendix 3b.

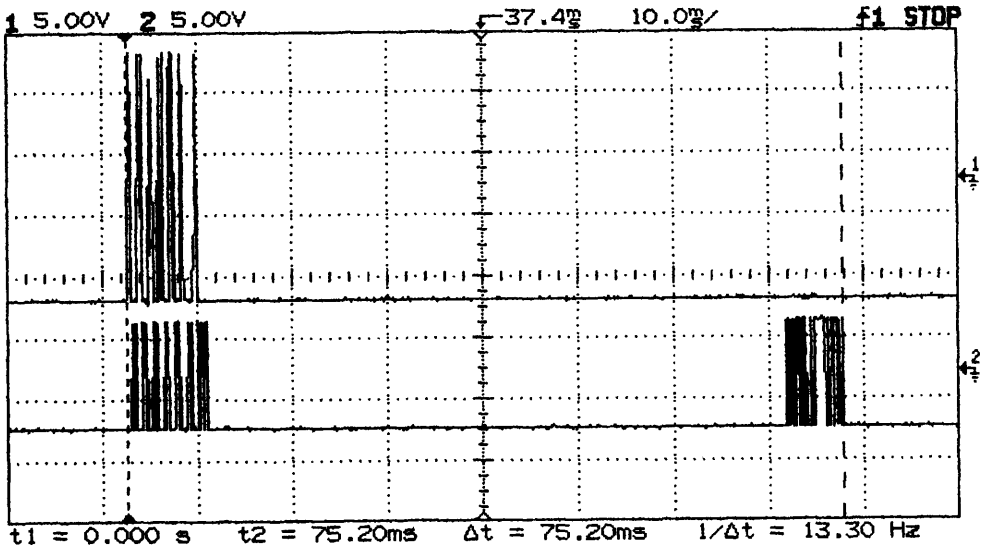


Fig. 3.8 Timing Diagram for a Basic Comms 'Read' Instruction

The data format of the serial communication will be discussed later in this section. The vertical 'bursts' indicated on the diagram correspond to the data exchange between the PC and the Mentor II. Channel 1 is the data transmitted by the PC, in this case the 'read' instruction. Channel 2 is the response from the Mentor II. This first 'burst' of data is the echo of the 'read' instruction, while the second data stream corresponds to the Mentor II register data. The total time taken for the information exchange, from request to completion, is 75.2ms, as indicated by the cursor measurement Δt on the diagram.

ANSI comms uses a lower level code, similar to assembly code. This increases the complexity of programming, but improves processing time compared with the Basic comms. It is shown in the next section that the time taken to complete a 'read' instruction using ANSI is 13.3ms. This is a vast improvement on the Basic comms. time of 75.2ms.

Initial appraisal of the two communication procedures, indicates that, due to improved time response, the ANSI comms option is the more favourable for communication with the PC-based simulation.

3.6.2 ANSI Comms

As mentioned before, Menu 14 of the Mentor II's software can be programmed to set the desired communications requirements. Setting the MD21 for ANSI comms, automatically sets some aspects of the communication format. Additionally, to maximise the communication speed, the baud rate is set at 19200bps. Table 3.2. indicates the format of communication selection.

Parameter	Format
Comms Selection	ANSI
Baud Rate	19200 bps
Data	7 Data Bits, 1 Stop Bit, Even Parity

Table 3.2 Desired Serial Communication Format of MD21

Both the MD21 and the selected simulation software, on the PC, must have the same communication format to ensure successful data transfer.

ANSI comms. makes use of a number of control characters (i.e. characters that require the use of the keyboard's control key). Instructions are sent as packages of data which are initiated and terminated with control characters. With an ANSI read command these control characters are ^***D*** (4h ASCII) and ^***E*** (5h ASCII) respectively.

The format of a 'read' instruction that is sent (in ASCII) from the PC to the Mentor II is:

^***D***N1N1N0N0M1M0P1P0^***N***

N1 and N0 are the drive address (used if more than one drive is connected to the PC - not needed in this case, therefore the drive is always designated as 01). These values are repeated for error control.

M1, M0, P1 and P0 are the required menu and parameter respectively.

For example, if the value of the armature voltage of the motor (03.04) is required by the PC's software, the following data would need to be issued by the PC:

^*D*00110304^*E*

The Mentor II responds by sending a package of data which includes the register selected, the value of that register and the initiation and termination control characters, in this case ^*B* and ^*C*. If, for example, the value contained in register 03.04 was 100 the response to a 'read' of that register would be:

^*B*0304+0100^*C*

ANSI comms has the ability to issue a 're-read' command. This is achieved by issuing ^*U*. The response of the Mentor II is the same.

Writing to the registers of the Mentor II using the write command has the format:

^*D*N1N1N0N0^*B*M1M0P1P0D4D3D2D1D0^*C*

N1 and N0 are the drive address

M1, M0, P0 and P1 are the menu and parameter selections

D0 to D4 is the value to be stored in the selected register where D4 is the sign of the data.

For example, to set the torque/current reference (04.08) of the motor at 555, the PC issues:

^*D*0011^*B*0408+0555^*C*

3.7 Serial Communications and Interfacing the PC and MD21

The serial communication device installed on the PC is the Intel 8250 UART (Universal Asynchronous Receiver/Transmitter) which, like the MD21, uses the RS232

communication protocol. RS232 is a two wire system which switches between ±15V but tends to be susceptible to noise over distances above 10m. Problems can also be encountered if a large bit rate is required, therefore RS232 is usually limited to bit rates of less than 20 kbits/s [3.7]. This is acceptable for communication between the PC and Mentor II since the baud rate will be no greater than 19.2kbits/s.

Fig. 3.9 shows the hardware connection between the UART of the PC and the MD21. Note that the 'handshaking' connections are shown but are not used with ANSI comms, they were used for initial testing of the Basic Comms. The only lines used are TX (transmit), RX (receive) and GND (ground).

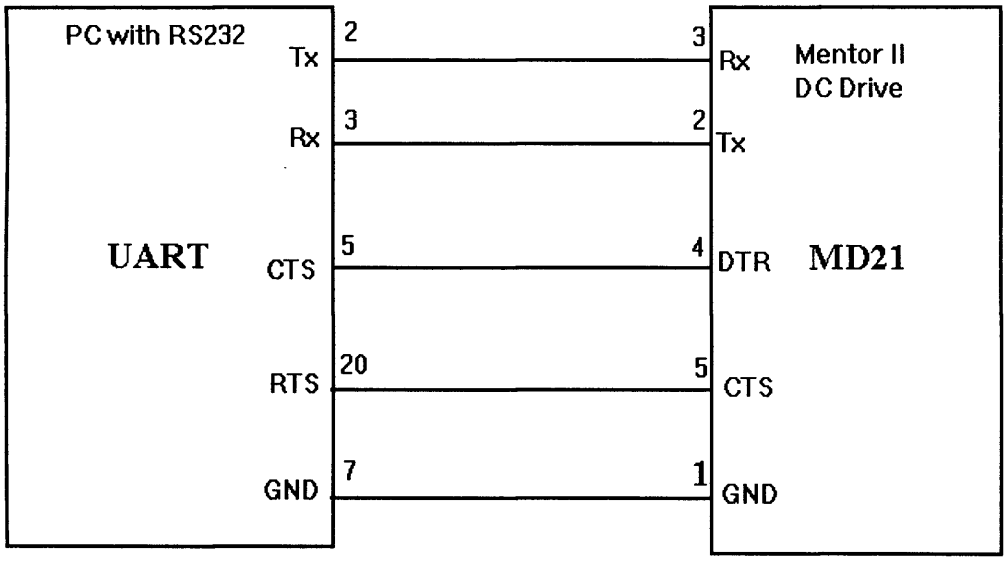


Fig. 3.9 Hardware Connection Between the PC and MD21

On the data lines TX and RX, positive voltages range from +3 to +15 volts and corresponded to logic '0', while negative voltages range from -3 to -15 volts corresponding to logic '1'. This arrangement is the inverse of the normal TTL voltage representation of logic.

3.7.1 Intel 8250 UART

The Intel 8250 UART allows both transmission and reception of 8-bit data via the serial port. The received and transmitted data, timing, monitoring and control of communications can be accessed via a number of registers within the UART. Table 3.3 shows the 8250 registers and their mapped addresses within the PC. Note that only the registers used for this particular project application are shown. Registers controlling interrupts and handshaking are omitted for the sake of brevity.

I/O Address	Register
2F8	TX Buffer (DLAB = 0)
2F8	RX Buffer (DLAB = 0)
2F8	Divisor Latch LSB (DLAB = 1)
2F9	Divisor Latch MSB (DLAB = 1)
2FB	Line Control Register (LCR)
2FD	Line Status Register (LSR)

Table 3.3 8250 Registers and I/O Address

As can be seen from Table 3.3, many of the registers are multipurpose and depend on the momentary action of the port (i.e. receiving, transmitting or port initialisation). DLAB (or the Divisor Latch Access Bit) is bit 7 of the Line Control Register and is used to select either data exchange or set the data exchange Baud rate.

With DLAB=1 the Baud rate of the UART can be set by sending the appropriate 2 byte value to the divisor latch registers (lowest significant byte (LSB) to 2F8H and most significant byte (MSB) to 2F9H). To obtain the Master clock which controls the Baud rate, the Reference clock of the UART is divided by the 16-bit divisor value. The bit rate (Baud rate) is then obtained by dividing the Master frequency by 16. Reference clock frequencies are typically 1.8432 MHz.

The control and status registers were used to set parity and stop/start bits (LCR) and monitor status of the receive and transmit registers (LSR). Appendix 3b contains a description of the relevant control and status registers.

When used during transmission, the UART can only accept a byte of data when the TX buffer is clear. This occurs when the transmission of the previous byte is complete, indicated by bits 5 and 6 of the LSR (transmitter holding register free) being set high. Similarly, when the UART is required to receive, a byte of data is available for reading when bit 0 of the LSR (received data ready) is high [3.7]. Software control of the communication has to allow for this continual 'polling' of the registers to check the status during both transmission and reception.

3.7.2 Communication Software Development

After defining the initialisation and operating parameters of both the MD21/Mentor II and the UART of the PC, it is necessary to develop software to test the serial link and confirm successful operation. Initially, simple programs were compiled using QBASIC, the Windows based BASIC language. The advantage of using QBASIC is that all the UART initialisation and control, is controlled automatically using simple instructions (i.e. the need to 'poll' status register during data exchange, is not required by the user's program). This means that the programmer can perform serial communication without knowledge of the structure of the PC's UART, other than it's port designation (i.e. COM1 or COM2). This 'simple' programming confirmed the validity of the hardware link and the programming structure of the QBASIC comms. Appendix 3b contains the details of the QBASIC programs used for reading from and writing to the MD21.

Once this initial test is confirmed, it is necessary to be able to perform the same tasks using C/C++ code. The reason for this will be explained in Simulink Real-Time Workshop section (see Chapter 4). Accomplishing these input/output (I/O) tasks using C/C++, requires the manual initialisation and manipulation of the PC's UART, as mentioned in the previous section. Although labourious, it does have the advantage of allowing greater control of the serial communication including control of handshaking, if required, unlike QBASIC.

List 3.1 shows a C program which is designed to send a ANSI instruction to the MD21. In this example, the Mentor II's torque/current demand register, 4.08, is programmed to be set at the value '555'. The '#include' statements at the beginning of the program, refer to 'header' files which contain 'declarations' required for mathematical (math.h) and I/O (conio.h) functions [3.8] and [3.9]. The UART is initialised by sending values corresponding to the required communications format to the appropriate UART registers. Initialisation is followed by the creation of the ANSI comms instruction array. Data transmission is controlled within the *for* loop by monitoring the status of the LSR, ensuring that the 'transmission register' is empty. Each byte of the ANSI instruction is transmitted using the *outp* command. The array *numb* is used to set up the output command. Further details regarding the C++/C programming language can be found in references [3.9] and [3.10].

```
#include <conio.h>     //for 'outp'
#include <iostream.h>  //for 'cout'
#define LCR 0x2fb
#define PORT 0x2f8
#define IER 0x2f9
#define MCR 0x2fc
#define MSR 0x2fe
#define LSR 0x2fd


int main()
    {
    /*Initialisation*/
    outp(LCR, 0x80);      //initiate baud set-up
    outp(PORT, 0x06);     //lsb baud
    outp(IER, 0x0);      //msb baud set to 19200
    outp(LCR, 0x1a);      //7e1
    outp(IER, 0x0);      //disable interrupts
    outp(MCR, 0x0);      //RTS=0
    /* end of init */
```

```
int numb;
char pc_instr[16] = {0x4,'0','0','1','1',0x2,'0','4','0','8','+','5','5','5',0x3,0xd}; // ANSI instr
here:for (numb=0;numb <= 15;numb++)          //create loop to o/p array
  {
    while ((inp(LSR)& 0x60) != 0x60);  //check for empty TXB
    outp(PORT, (int) pc_instr[numb]);    //o/p byte to UART
  }
cout << "This is the end of for loop";   //debug to check for end of loop
goto here;                          //continuous poll
return 0;
}
```

List 3.1 C Program to Write ANSI Instruction to MD21

Fig 3.10 is the result of monitoring the PC's TX line during program operation. The transmission of the instruction is continually 'looped' in order to aid capture on the oscilloscope.

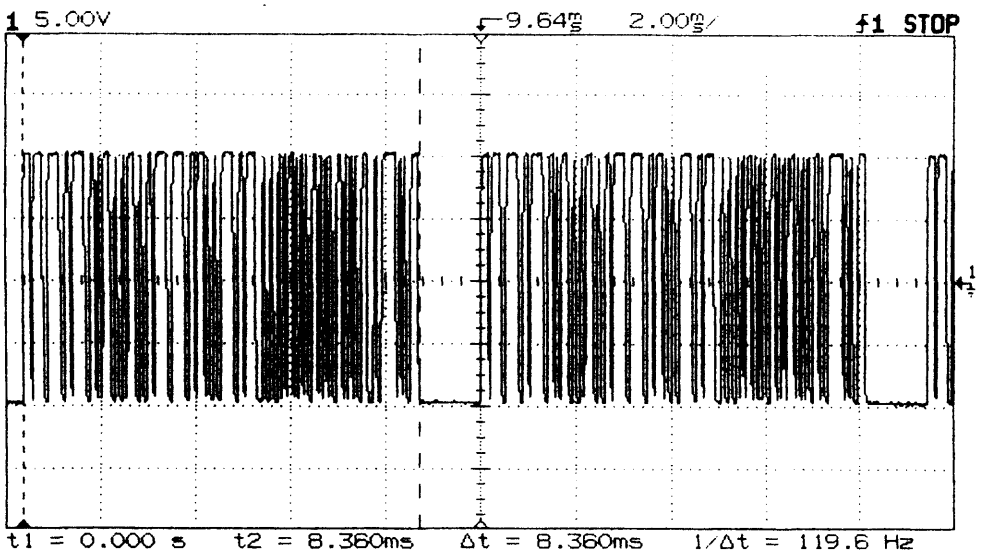


Fig 3.10 Timing Diagram for an ANSI 'Write' command

Each 'burst' of serial data corresponds to the ANSI 'write' instruction. The gap between data streams is due to a forced delay in programming to aid visualisation of the instruction.

It can be seen from the figure, that the total time to complete the transfer of the instruction is measured using the oscilloscope cursor function and is 8.36ms (Δt on the figure).

As indicated in ANSI comms. section, reading data from the Mentor II is more involved. The read instruction has to be transmitted by the PC, similar to a write instruction, and the response from the Mentor II has to be effectively processed. This processing involves reading each byte of the response from the 'receiver buffer' of the UART, and storing it in an array. As before, the LSR has to be monitored for the presence of data bytes in the 'receiver buffer'.

List 3.2 shows the main section of the C program used to read data from the Mentor II, and simply display it. The parameter *num* and array *from_mentor* are used to obtain the response from the drive one byte at a time. Note that the initialisation and declaration section is the same as shown in List 3.1 and has been omitted for the sake of brevity. Once again, the program runs as an infinite loop.

```
int numb, num=0;
int array=0;
char from_mentor[20];
char pc_instr[13] = {0x4,'0','0','1','1','0','3','0','4',0x5}; // read reg 03.04

/*section to send read instruction to Mentor II*/
here:for (numb=0;numb <= 9;numb++)          //create loop to o/p instr array
  {
    while ((inp(LSR)& 0x60) != 0x60);  //check for empty TXB
    outp(PORT, (int) pc_instr[numb]);    //o/p chr to port
  }

/*section to obtain response to read instruction from Mentor II*/
do{
  while((inp(LSR) & 0x1) != 1) ; //check for data ready in RXB
  from_mentor[num] = (char)inp(PORT);    //input data from RXB
```

```
  num++;                    //next array position
 } while ((int)from_mentor[num-1] != 0x3);   //check for end of data
 from_mentor[num] ='\0';          //place 'null' at end of input array
cout << from_mentor <<endl;       //print the input array
num=0;                     //reset count
goto here;
return 0;
```

List 3.2 C program to Read Data from a Mentor II Register

Fig 3.11 shows the TX (channel 1) and RX (channel 2) of the UART during the execution of the above C program. Channel 1 is the ANSI 'read' instruction data stream, while channel 2 is the register contents data stream. It can be seen from Δt, (the time difference between cursors) that the required processing time for a 'read' instruction is 13.3ms (c.f. 8.36ms for a 'write' instruction).

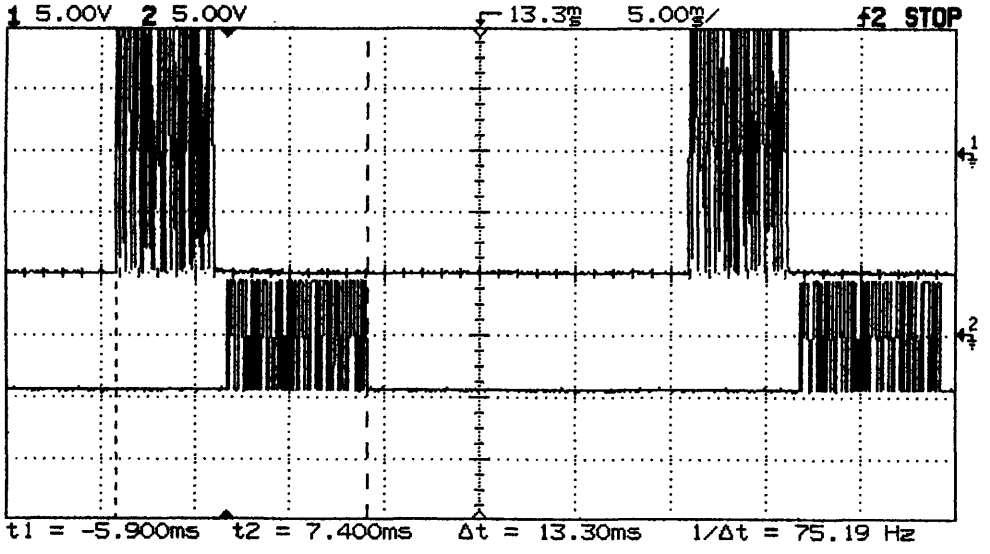


Fig 3.11 Timing Diagram for an ANSI 'Read' Command

The time required for a 'read' and a 'write' instruction indicates that any sampling required by the software simulation, would have been limited to 120Hz for a 'write' and 75Hz for a 'read'. It is assumed that realistically, the software simulation will require at least one 'read' and one 'write' per iteration, i.e. 'write' a torque demand to the Mentor II

followed by a 'read' of the motor's speed. This indicates that, at the minimum, the total time required to allow for the I/O commands, would be 21.66ms or 46Hz, which is relatively slow [3.11].

It is also necessary to estimate if the processing time of ANSI instructions will further effect the time delay of the serial communications. To achieve this, the armature current of the DC motor was monitored while the torque/current demand register of the Mentor II (04.08), was issued a 'step' change using a 'write' command from the PC. Fig. 3.12 shows the timing of the response. The lower data stream, shows the serial data which carries the 'step' command, while the upper most trace shows the change in average armature current. The middle, faint trace is the inverse of the armature current and not required for analysis.

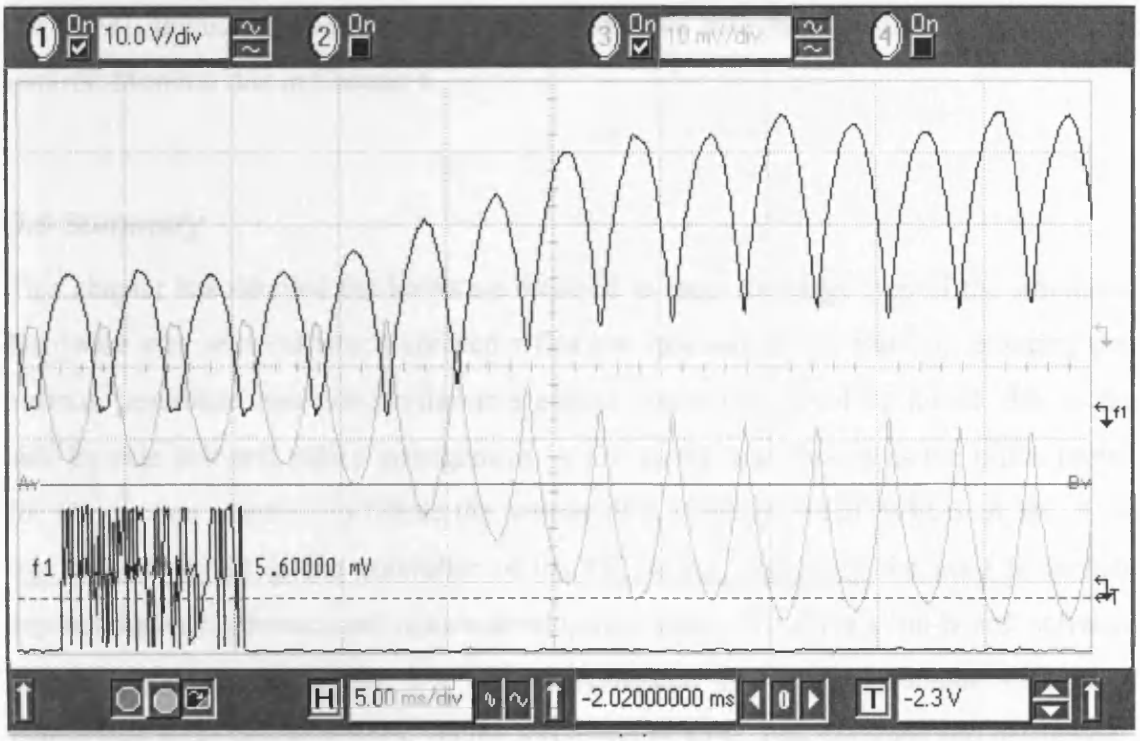


Fig 3.12 Change in Armature Current in response to a 'step' demand

It can be seen that the delay time, i.e. time taken to reach new current demand, is approximately, 17.5ms. This indicated that the total time taken for a 'write' instruction is ~25ms. A delay of this order was expected since the specifications listed in the Mentor II guide, stated that the 'current loop' resolution was 80Hz (12.5ms). The difference is due to the controller in the current loop of the Mentor II. This has been 'autotuned' by the Mentor II to improve the performance of the drive for the particular motor under control [3.5].

From intial studies, it is suggested that sampling rates in the order of 20-40ms are sufficient for the simulation of WECS: '.., the largest step length that provides numerically satisfactory integration is 0.02 seconds. Note that with a deterministic input only, the numerical integration could be performed with a step length of 0.1 seconds' [3.12]. This is due to the slow dynamics of the mechanical components of WECS and, therefore, allows the assumption that the serial communication link between the PC and

the MD21, including the processing delays, would be sufficient for the required test-bed control. More on this in Chapter 6.

3.8 Summary

This chapter has detailed the hardware required to meet the objectives of the simulator. Hardware was selected which allowed a flexible approach to the test-bed, ensuring that various generators and other renewable energy converters could be tested, due to the side-by-side belt and pulley arrangement. A DC motor was chosen as the prime mover for a induction machine to mimic the actions of a 'constant speed' WECS. A DC drive was chosen to act as the controller of the DC motor, removing the need to include explicit power electronics and reduce development costs. The drive's 'on-board' software and its serial communication link were investigated and a novel, unique strategy to manipulate these characteristics, via the serial port of a PC, was designed and developed.

Chapter 4 Selection of Simulation Software

Chapter 3 established the selection of the simulator hardware and the communication protocol required for exchange of data between the Mentor II and the PC. The next stage of the simulator development is to investigate suitable simulator software. The main criterion that has to be met is that the software must be able to successfully interface with the Mentor II DC drive via the serial port of the PC. This suggests that the software must be able to operate in real-time. Also the software will, ideally, have to allow for the development of simulation models in a modular manner. This allows the complexity of the model to be increased in stages and model parameters to be easily changed and to be interchangeable.

Ideally, the desired simulation software will be available as an industrial standard commercial package and be readily available within the department. This would reduce both the software development time and cost of the project.

This chapter introduces the software package selected to meet the project's objectives and the the interaction between it and the MD21, using the serial communications link.

4.1 Mathworks MATLAB/Simulink

One such package that meets all the above criteria, is the Mathworks' Matlab/Simulink software.

Matlab is a package specifically designed for scientific and engineering calculations. Its command structure is similar to a DOS or UNIX structure, using paths and directories. The 'base' version of Matlab contains simple mathematical functions which can be used in conjunction with matrices and/or statements and variables [4.1] [4.2]. In addition to this 'base' version of Matlab, there are available task specific 'toolboxes'. These

'toolboxes' include the Control Systems Toolbox, Simulink and the Real-Time Workshop.

Simulink is an interactive environment for modelling, analysing, and simulating a wide variety of dynamic systems, including discrete, analogue, and mixed signal systems. Simulink provides a graphical user interface for constructing block diagram models using "drag-and-drop" operations. With Simulink's large library of building blocks, a system can be modelled rapidly, without writing a single line of code [4.3]. Fig. 4.1 shows the available Simulink library.

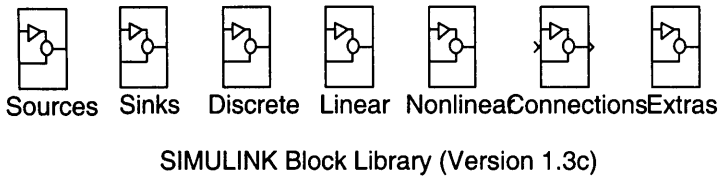


Fig 4.1 The Simulink Building Block Library

Some of the features of Simulink include:

- A comprehensive block library for creating linear, non-linear, discrete-time, continuous-time, hybrid and multirate systems.
- Convenient creation of hierarchical models and subsystems
- Mask facility for creating custom blocks and block libraries
- Model browser to view the decomposition of systems from highest-level through to component level
- Scalar and vector connections
- Signal and port labelling for clear and concise block diagrams
- Interactive simulation with live display
- Scopes, input sources, output sinks [4.3]

When a Simulink model is created by drawing a block diagram, Simulink uses the information in the block diagram to create an 's-function'. Each block diagram has a

corresponding s-function (normally transparent to the user) with the same name as the model. S-functions are basically Matlab functions with a special calling syntax which allows access to a models dynamic equations [4.4].

The strength of s-functions is their duality. Although Simulink automatically creates an s-function for any block diagram, the process can be reversed to produce a block diagram from a user defined C code s-function. The structure of this C code is very exact (see later in this chapter) and must be compiled as a MEX file (a Matlab executable file). Once in the form of a MEX file, the s-function can be used as a block diagram in Simulink. This option is usually preferred if a particular system is best described as a set of equations and it is simpler to enter them as C code.

Once the s-function C code is complete, it is compiled as MEX file using the Matlab '!cmex' instruction.

4.2 Simulink Real-Time Workshop

The Simulink Real-Time Workshop (RTW) is a toolbox that provides an integration between Simulink models and hardware facilities. It produces C code directly from Simulink graphical models and automatically builds programs that could be run in real-time as a 'stand alone' DOS executable. The major applications of RTW are [4.5]:

- Real-Time control - Control can be designed in Simulink/Matlab.
- Hardware-in-the-loop - Simulink is used to mimic real life measurements
- Interactive real-time parameter tuning
- Program building is fully automated

Fig. 4.2 shows the building process that is required to build a real-time 'stand alone' executable in RTW.

The Simulink model, shown in Fig. 4.2 produces a C coded s-function. The difference with the RTW is that the C code needs to be compiled to create the DOS executable. In order to do this, a C/C++ compiler, compatible with the Simulink s-functions, is required. One such compiler is the Watcom C/C++ compiler which, fortunately, is available on the University PCFS network.

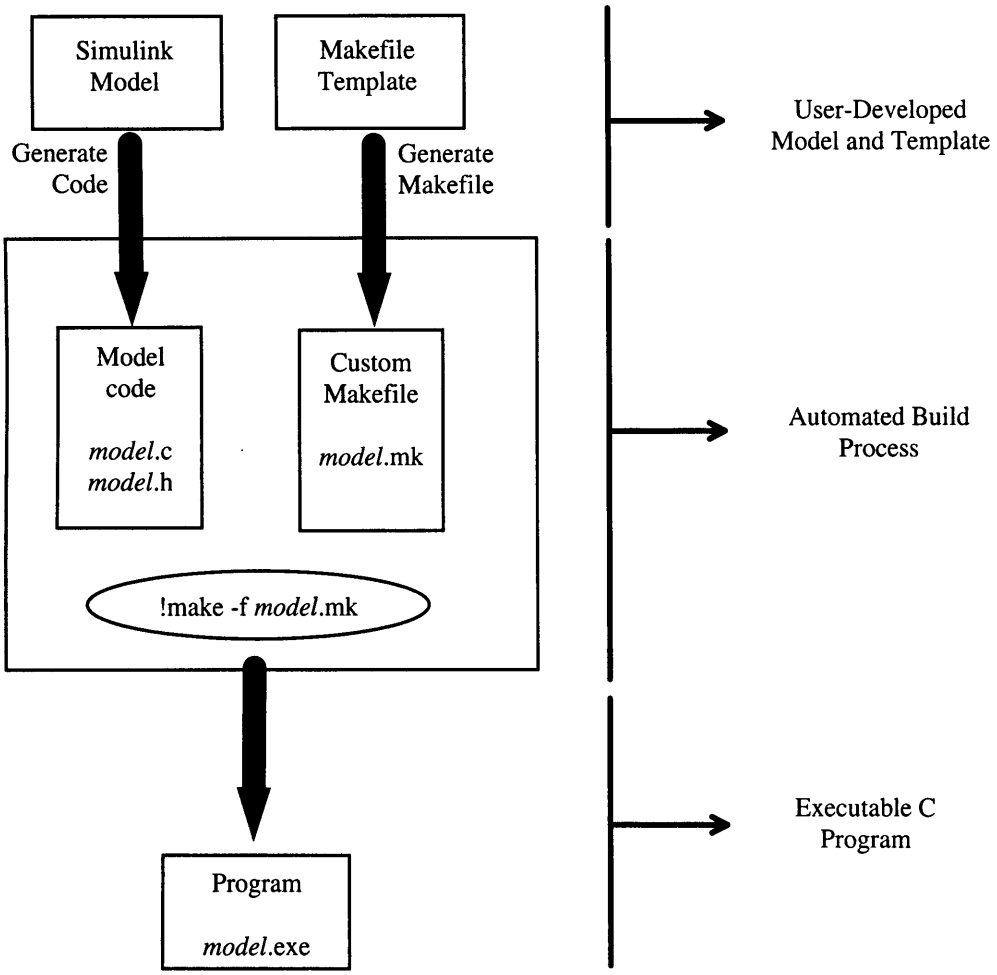


Fig 4.2 The RTW Build Process

In order to create an operating system dependent executable, i.e. a DOS executable file, the compiler has to be given certain information regarding the 'building process' of the program. Also Matlab has to be supplied with the location of the Watcom compiler and linker. This is achieved by creating a 'makefile' template, see Fig. 4.2, which has to be

specific for the user's particular compiler and target environment. The template contained such information as macro definitions, tool locations and compiler options. Appendix 4a contains the template makefile, which has been redesigned to control the executable building process, using the Watcom C/C++ V10.5 compiler on the university's PCFS network.

The '!make' command, seen in Fig 4.2, is a RTW instruction that combines the generated C code from the Simulink model and all the complier/linker options in the makefile, to produce the desired DOS executable (EXE) file.

The above process is valid for creation of executables regardless of the presence of I/O modules in the Simulink model. To include I/O and hence HILS, RTW requires the use of Device Driver Blocks.

4.2.1 Device Driver Blocks

In order to create a real-time EXE file using Simulink as a controller, or as HILS, there needs to be some I/O hardware control in software. This software can be implemented as a C code s-function, compiled as a MEX file and implemented as a Simulink block diagram. This block diagram is placed in the Simulink model, used for the real-time operation, wherever the particular I/O is needed. RTW contains 'ready made' MEX files for a number of commercially available DACs and ADCs. If the user plans to use these devices, the provided DDBs can be used within the RTW without further development of the devices' s-functions.

To include I/O modules other than the ones included with the RTW, the appropriate s-function C code has to be created and compiled as a MEX file (using '!cmex'). RTW provides an 's-function template' where the user can include the description and relevant information of the I/O device to be used. This template is called SFUNTMPL.C.

The C code s-function has to conform to a certain data structure, recognised by Simulink, for a MEX file to be successfully compiled. This structure is defined as 'SimStruct' and

has to be called from the device driver block (DDB) s-function. Specific functions also have to be included. Fig 4.3 shows the structure for a DDB s-function.

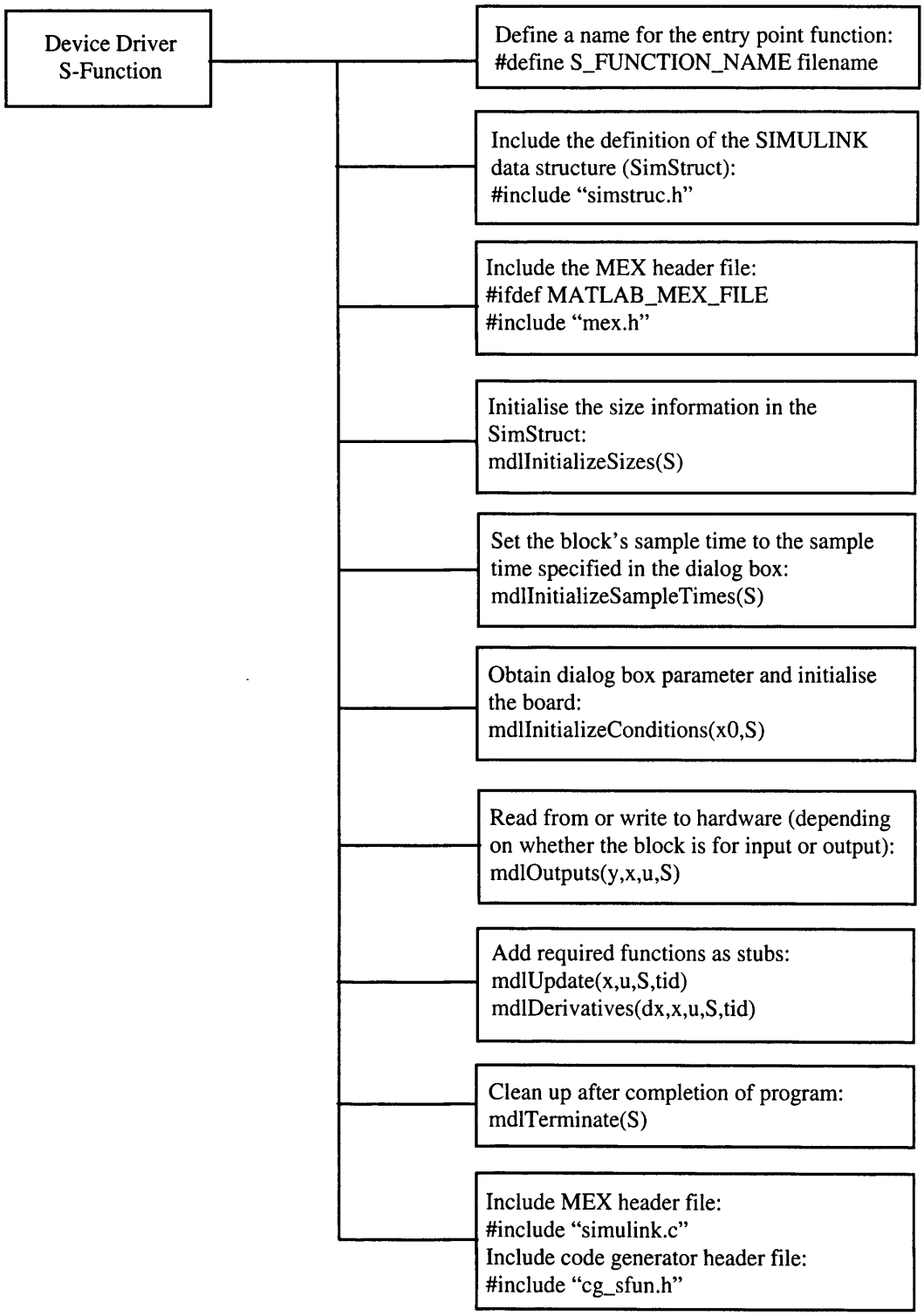


Fig 4.3 Format of a Device Driver S-Function

Each stage has a specific function in order to create the MEX file, namely:

- Initialising the SimStruct
- Initialising the I/O device
- Calculating the block outputs
- Terminating the program

Initialising the SimStruct is required on every block while the remaining operations are device dependent. Some of the functions such as mdlUpdate() and mdlTerminate() are required for structural purposes, but are generally unused.

The I/O initialisation (such as board memory location) in the DDB s-function can be rigidly set within the program or set via a pull down menu once the block has been created within Simulink. The latter option has to be set in the s-function while the menu parameter(s) has to be set in Simulink once the MEX file has been compiled. This is known as masking, and allows initialisation parameters to be varied without altering the DDB s-function.

4.2.2 Designing the Mentor II Device Driver Blocks

Two blocks have to be designed to realise the desired interface between Simulink and the Mentor II, one for reading data from the drive and one for writing data to the drive. The basic requirement is to convert the C programs that successfully communicated with the drive (discussed earlier in Chapter 3), into DDB s-functions. This involves manipulating the SFUNTMPL.C to include masking details, 'include' files and definitions, and ensuring that the imported C code is compatible with the s-function structure.

The DDB C code for reading data from the speed register (03.02) of the Mentor II (spdin.c) is shown in Appendix 4a. All the 'include' files and definitions are located prior to the mdlInitializeSizes() section (see Fig 4.3). The mdlInitializeSizes() section contain all the input and output details of the block. Since this block has one output (i.e. output to

the Simulink environment), it is set accordingly. Only one masking parameter is included, this is the sampling time and is defined in mdlInitializeSampleTime().

The 'main()' section of the imported C code makes up the mdlOutputs() section. This is the section that is continuously 'looped' when the Simulink model is run as a stand alone executable and which normally reads or writes to the ADC or DAC location, respectively. The s-function requires that these 'external' data transfers are assigned to the Simulink specified declaration variables. For example, a Simulink DDB input block (such as an ADC) has to assign the output value of that block to 'y', which is declared as a variable 'double float' in the mdlOutputs() section.

The imported C code has to mimic the action of an ADC and:

- Issue a ANSI 'read' command
- Receive the character string response from the Mentor for a 'read' command
- Separate the register contents from remaining data (this was done by ignoring the first six characters, ^***B***03.02, of the ANSI data stream)
- Convert this value to a 'double float' and assign it to 'y'.

List 4.1 shows how the above criteria are met by the 'imported' C code (mdlOutputs() section). The first character of the array *small* is programmed to point at the seventh character of the Mentor response array *from_mentor.*

```
static void mdlOutputs(double *y, double *x, double *u, SimStruct *S, int tid)
{
        int numb, num=0;
    int array=0, loop;
    char from_mentor[20], small[5];
        char pc_instr[13] = {0x4,'0','0','1','1','0','3','0','2',0x5}; //'read 03.02'
    for (numb=0;numb <= 9;numb++)          //create loop to o/p array
      {
        while ((inp(LSR)& 0x60) != 0x60);  //check for empty TXB
        outp(PORT, (int) pc_instr[numb]);    //o/p chr to port
```

```
    }

      do{
     while((inp(LSR)& 0x1) != 1);              //check for data ready in RXB
     from_mentor[num] = (char)inp(PORT);       //input data
     num++;                                    //next array position
    } while ((int)from_mentor[num-1] != 0x3);      //check for end of data

  from_mentor[num] ='\0';    // null array

  for (loop=num-6; loop<num-1; loop++)   //start loop to miss first 6 chars
  {
       small[array]=from_mentor[loop];
       array ++;
  }
  small[array] = '\0';       // NULL at end of array
  *y = atof(small);        // convert string to double


}
```

List 4.1 Imported C Code of spdin.c

The structure of the output ddb is, essentially, in the same format as the input ddb. The initialisation is the same as is the single mask parameter (sample time). The only difference, is that the mdlOutputs() section has to mimic the action of a DAC and the input to the output block is assigned to 'double float' 'u'.

The C code to mimic a DAC and write data to the Mentor II has to include the following:

- Receive data from the Simulink simulation (i.e. 'u')
- Determine the sign of the value 'u'
- Convert 'u' from a double float to a character array
- Include this data in an ANSI 'write' instruction
- Send this instruction to the Mentor II

The mdlOutputs() section used to achieve the above is shown in List 4.2 while the whole program is shown in Appendix 4a (torout.c). In this designed code the ddb writes data to the Mentor II torque/current demand register 04.08.

```
static void mdlOutputs(double *y, double *x, double *u, SimStruct *S, int tid)
{
 int numb;
 char pc_instr[17] = {0x4,'0','0','1','1',0x2,'0','4','0','8'}; // set 04.08;
 int intgr, loop, array=11, sign=10;


 /* the following for loop converts the int to a char array***
  ****NOTE: the input value would never be greater than 1000***/
 intgr = (int) *u;              //debug to test input data

 if (intgr>0)
          {
            pc_instr[sign] = '+';
          }
 else
          {
            pc_instr[sign] = '-';
          }

 intgr=abs(intgr);

if (intgr>1000)
 {
  intgr=1000;
 }

 for (loop=1000;loop>1;loop=loop/10)
 {
   if (loop>intgr)
   {
    pc_instr[array] = 0x30; //set bit to zero(=30h)
   }
   else
   {
    pc_instr[array] = (intgr/loop) + 0x30; //convert integer to ascii hex code
   }
   intgr = intgr - (intgr/loop)*loop;
   array++;
 }
 pc_instr[array] =  (intgr%10) + 0x30;     // uses remainder function
 pc_instr[++array] =0x3;              //^C
 pc_instr[++array] = 0xd;             // RETURN
```

```
for (numb=0;numb <= array;numb++)          //create loop to o/p array
 {
   while ((inp(LSR)& 0x60) != 0x60);  //check for empty TXB
   outp(PORT, (int) pc_instr[numb]);
 }
 array =11;
 pc_instr[array] = '\0';    //place 'null' at end of array

}
```

List 4.2 Imported C code of 'torout.c'

On receiving the double float value 'u' from Simulink, the program immediately converts it into an integer. This is because only integer values can be passed to Mentor II. The sign of the integer is then assessed by calculating if the value is greater than zero. The next stage is to convert the integer value, which must be less than 1000 since it is the maximum value that can be passed to the Mentor II, into a character array. This is done by comparing the integer value with 1000. If the value is greater than 1000 the value *intgr* is set to 1000. It is then divided by descending powers of 10, starting from 1000. The calculation results in single digit integer, which can be converted to a hexadecimal ascii code and is equivalent to that integer (this is done by adding 30H to it). Finally, any remainder left after the final division by 10 is calculated and converted to ascii. Each converted ascii character is stored, consecutively, in the instruction array *pc_instr*.

Both spdin.c and torout.c were successfully compiled as MEX files and used to create the corresponding Simulink graphical modules.These modules are seen to give similar timing characteristics to Fig. 3.10 and Fig. 3.11. Fig 4.4 shows a block diagram used to test the I/O modules (DDBs) during real-time operation, sending a torque demand to the DC motor (unloaded) while monitoring the speed.

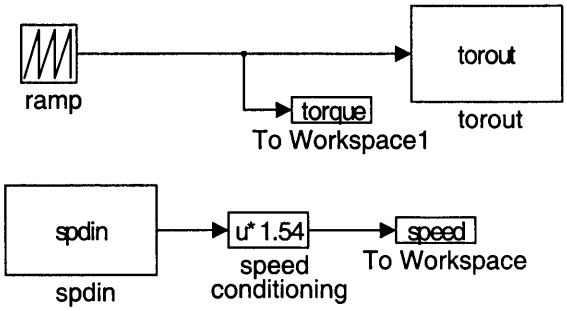


Fig 4.4 Block Diagram for Testing I/O DDBs

Fig 4.5 shows the result of the RTW executable. The 'torque demand' is shown without units as the number is unique to the Mentor II.

The linear ('ramp') torque demand to the drive, and hence motor, results in an acceleration on the motor. The relationship between the torque, T, and the motor speed, ω, is :

$$T = J\frac{d\omega}{dt} \tag{4.1}$$

where J is the inertia of the motor. Since the torque demand is a linear ramp and directly proportional to time, it can be replaced with At, where A represents the the gradient of the ramp. Rearranging eqn. (4.1)

$$\frac{d\omega}{dt} = \frac{At}{J} \tag{4.2}$$

Integrating both sides

$$\omega = \frac{1}{2}\frac{A}{J}t^2 \tag{4.3}$$

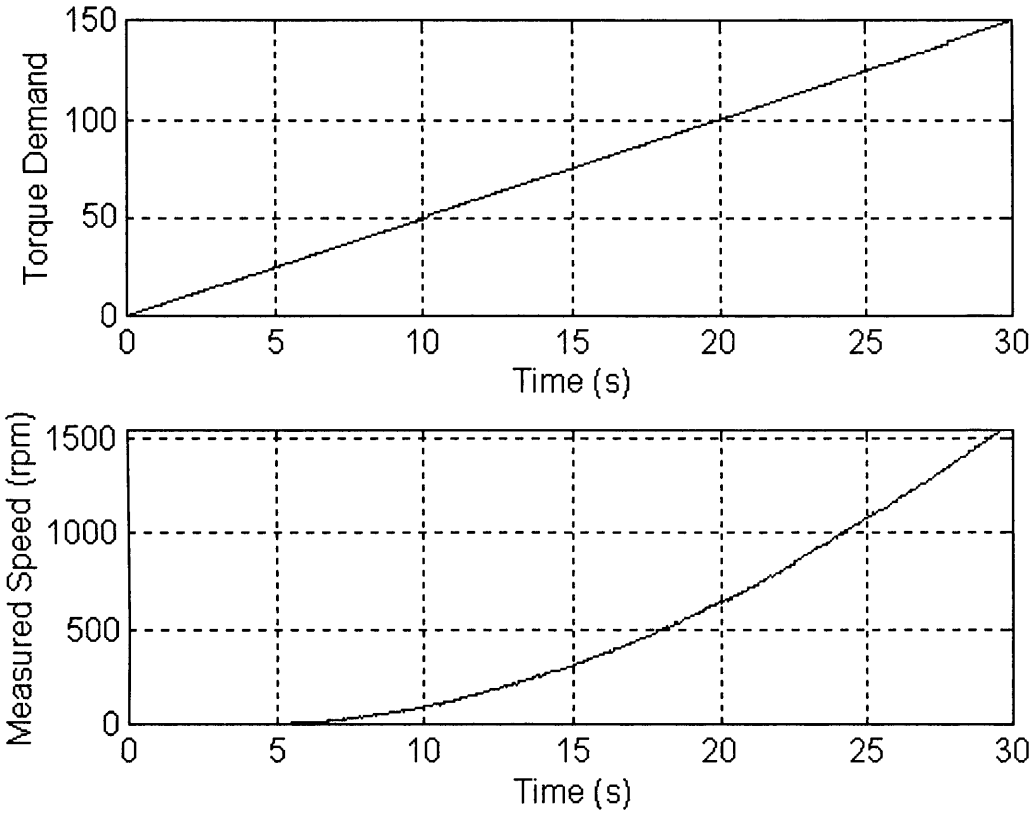


Fig 4.5 Torque Demand and Measured Speed of the DC motor

Fig. 4.5 indicates a 'square' relationship between the speed of the motor and the torque demand, confirming the theory stated in eqn. (4.3).

The above example suggests that the design and use of the DDBs, for real-time, serial communication between Simulink and the Mentor II, is valid.

4.3 Summary

The software selected to simulate the WEC and control the serial communications link in real-time, is the Mathwork's Simulink with the Real-Time Workshop. As with the hardware, this package was available within the department, reducing development time and costs. Software was designed to allow for the unique communication requirements of

both Simulink and the Mentor II, mimicking the action of both ADC's and DAC's. The development is unique and novel, following the in-depth study of both systems. The software, known as 'Device Driver Blocks', was developed as C code s-functions, a Simulink program format which can be linked as a graphical module within Simulink.

Following design, the driver blocks were extensively tested to validate their inclusion in any Simulink model requiring real-time communication with the Mentor II DC drive.

Chapter 5 Software Simulation

Having established the modelling requirements for the aerodynamics, relevant to a wind energy conversion system (WECS), and the drive train dynamics, it is necessary to implement the modelling in the selected software, Mathwork's Simulink. The advantage of Simulink is that the model can be developed in parts, particularly the aerodynamics, and each stage of the development assessed accordingly.

This chapter will detail the development of the WECS model in software and explain any difficulties encountered using Simulink. The 'software-only' simulation is required to assess and verify the model before including the simulation hardware for hardware-in-the-loop simulation (HILS).

Models of two WECS are created to confirm the generalisation of the theory and to show the flexibility of the simulator. The first model is a 330kW WECS, this model will be referred to as the 'Strathclyde' model [2.8] [2.20]. The second model is developed from the information available for a Windharvester 45kW WECS based at the Rutherford Appleton Laboratory (RAL) in Oxfordshire. Both machines are three bladed, the former is a pitch regulated device with a solid tower while the latter is stall regulated with a lattice tower. From the investigations of Chapter 2, it is suggested that the effects of induction lag and tower shadow should be less on the RAL WECS compared with the Strathclyde WECS.

The development of the WECS model will initially concentrate on the Strathclyde WECS, firstly detailing the drive train implementation followed by the generator. A transfer function relating the input of the model (aerodynamic torque) to the output (generator reaction torque) will be derived from the model, and compared with that given in the references. Once confirmation of the model is obtained, the parameters of the RAL model will then be calculated using the dynamic relationships derived in Chapter 2, and implemented in Simulink.

The RAL model will be used to develop all the aerodynamic properties relevant to a WECS. Each of the properties described in Chapter 2 such as the aerodynamic torque equation with C_q-λ data, rotational sampling, spatial filtering and induction lag, will be introduced to the model and simulated. A comparison of the low speed shaft (LSS) torque of the simulated WECS and the measured data of the actual WECS is undertaken to verify the accuracy of the simulator. This is performed as each aerodynamic property is added to the model, confirming the importance of including each effect in the simulation.

5.1 Implementing the Drive-Train Dynamics

Chapter 2 showed that the drive-train dynamics of a WECS can be accurately modelled by representing the WECS as a system of rotating masses and springs. The model 'lumps' together many of the dynamic characteristics to create the simplified model shown previously in Fig 2.11.

Fig. 5.1 shows how the drive train model, developed in chapter 2, is implemented in Simulink.

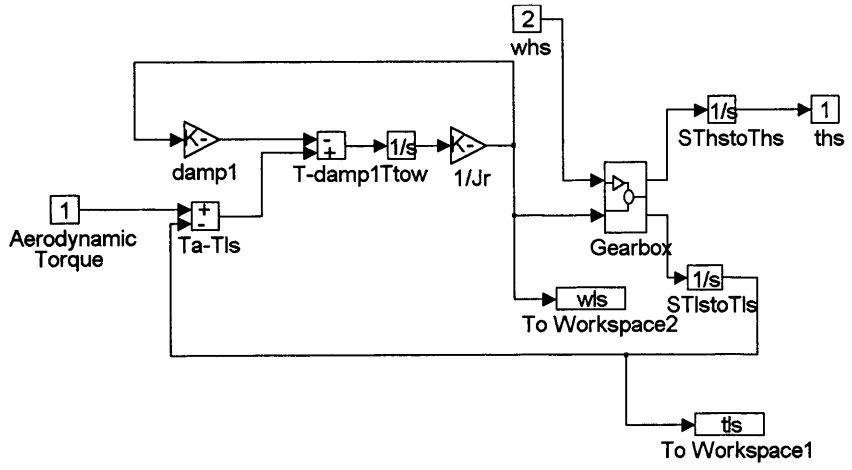


Fig. 5.1 Simulink Model of the WECS Drive-Train

Fig. 5.1 is the direct implementation of Fig. 2.11, where the '1/s' modules are Laplace integrators while the 'K-' modules are Simulink gains. The 'To Workspace' modules are used for monitoring all the relevant parameters required to assess the performance of the simulation, in this particular case the LSS torque, 'tls', and the LSS speed, 'wls'. The term 'damp1' is equivalent to the LSS damping constant, γ_1 of Fig. 2.11.

The gearbox dynamics, in the Simulink model, are 'grouped' together under the heading 'gearbox' (see Fig 5.1). Fig. 5.2 shows the details of the module.

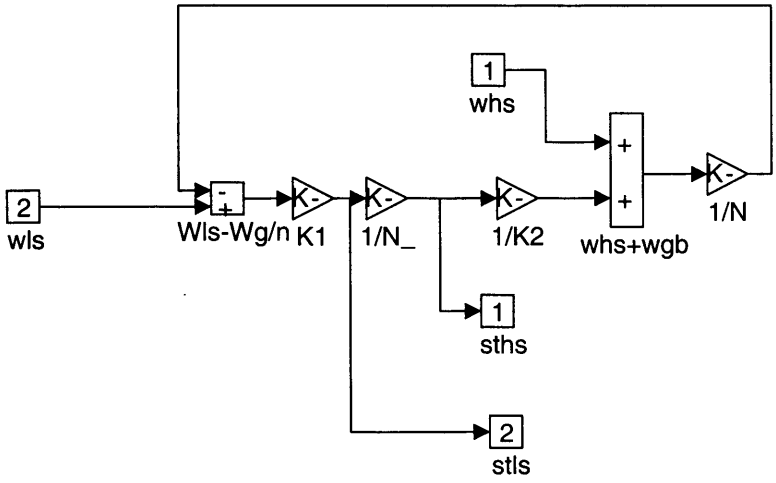


Fig. 5.2 Simulink Gearbox Module

The inputs to the module are the LSS and HSS speeds, 'wls' and 'whs' respectively. The outputs of the module are the derivatives of the LSS and HSS torques, 'stls' and 'sths' respectively. Once again, Fig. 5.2 compares directly with Fig. 2.11.

5.1.1 Algebraic Loops in Simulink

Algebraic loops occur when two or more blocks with direct feed-through of their inputs, such as gain blocks and non-linear blocks, form a feedback loop. When this occurs, Simulink performs iterations at each step of the simulation. This slows down the processing time and the loop may be unsolvable [4.4]. Additionally, the Real Time

Workshop (RTW) does not compile code that includes algebraic loops so they therefore, have to be removed from the model [4.5].

Preliminary assessment of the drive-train indicates that the gearbox contains an algebraic loop. Any simulation using the gearbox results in the Matlab warning shown in List 5.1.

Warning: The following blocks form an algebraic loop:
Gearbox/K1
*Gearbox/1/N_*
Gearbox/1/K2
Gearbox/whs+wgb
Gearbox/1/N
Gearbox/Wls-Wg/n
This will be solved at each iteration.

List 5.1 Matlab Algebraic Loop Warning

Obviously, the model has to be redesigned so that the loop is removed. Fig. 5.3 is the gearbox section of the drive-train, taken from Fig. 2.11.

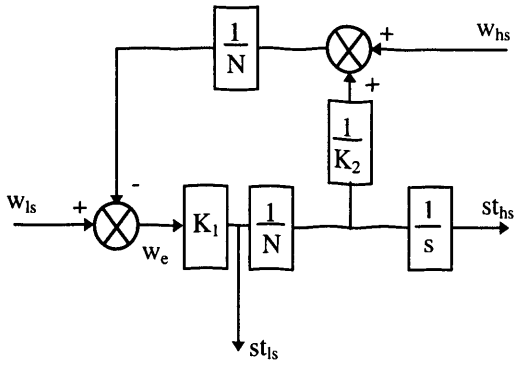


Fig 5.3 Gearbox Algebraic Loop

The equations relating the parameters shown in Fig. 5.3 are:

$$st_{ls} = w_e K_1 \tag{5.1}$$

$$st_{hs} = w_e K_1 \frac{1}{N} \tag{5.2}$$

$$w_e = w_{ls} - \frac{1}{N}(w_{hs} + \frac{w_e K_1}{N K_2}) \tag{5.3}$$

$$w_e = (w_{ls} - \frac{w_{hs}}{N})\left(\frac{1}{1+\frac{K_1}{N^2 K_2}}\right) \tag{5.4}$$

$$= (w_{ls} - \frac{w_{hs}}{N})\left(\frac{N^2 K_2}{N^2 K_2 + K_1}\right) \tag{5.5}$$

Eqn. (5.1), Eqn. (5.2) and Eqn. (5.5) are implemented in Simulink as shown in Fig. 5.4.

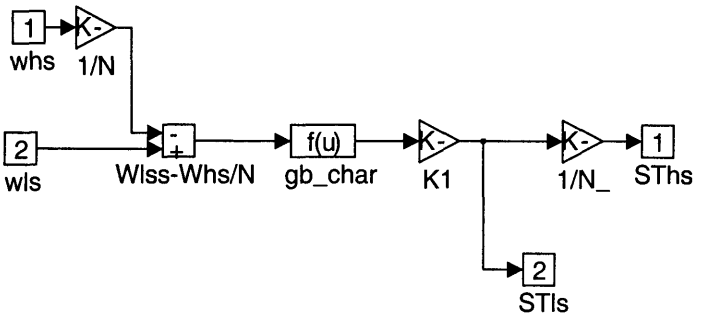


Fig 5.4 Alternative Gearbox Arrangement

'gb_char' module details the information contained in the larger brackets of Eqn. (5.5). The alternative arrangement removes the algebraic loop but retains all the relevant dynamic information.

5.2 Implementing the Generator Dynamics

The HSS mechanical dynamics of Fig. 2.11 are combined with the generator electrical dynamics of Eqn. (2.23) to create the model shown in Fig. 5.5

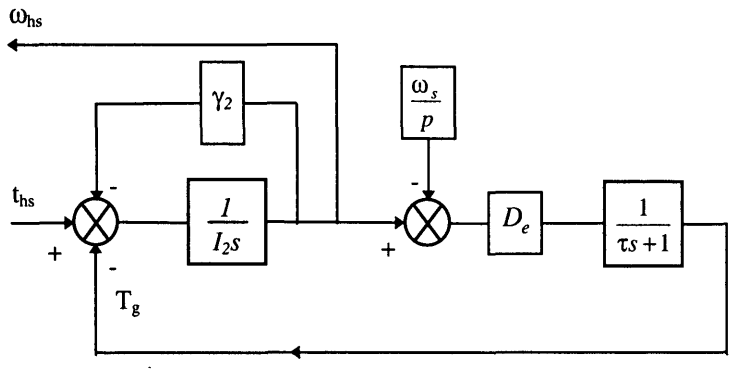


Fig 5.5 Combined HSS and Generator Dynamics

Fig 5.5 can be implemented in Simulink as shown in Fig 5.6

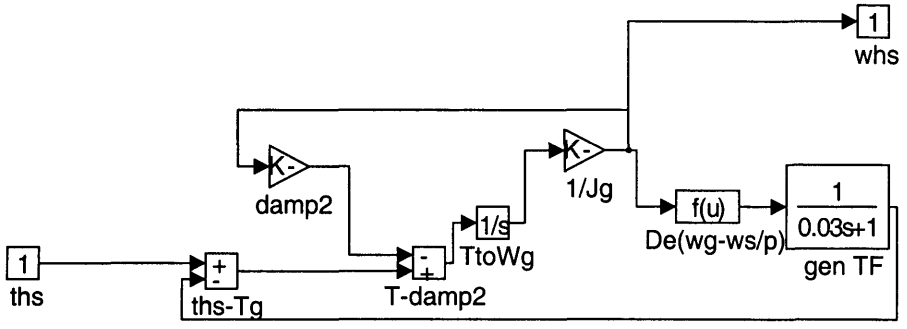


Fig 5.6 Simulink Implementation of the HSS and Generator Dynamics

The 'gen TF' module is the first order representation of the generator which, in this example, has a time constant of 30ms. The input to the module is the HSS torque, 'ths', while the HSS speed, 'whs', is the feedback, to the gearbox module of the drive train.

5.3 The Strathclyde WECS Model

The Strathclyde model contains the parameters for a 330kW WECS. Initial testing of the drive train and generator model can be performed using this data. List 5.2 includes the WECS parameters for the 330kW model [2.20].

$I_1 = 1.9x10^5\ kgm^2$
$I_2 = 3.8\ kgm^2$
$K_1 = 1.26x10^7\ Nm/rad$
$K_2 = 3.02x10^5\ Nm/rad$
$N = 40.65$
$\gamma_1 = 500$
$\gamma_2 = 0.3$
$D_e = 915.8\ Nm/rad/s$
$\tau = 0.03\ s$

List 5.2 330kW WECS Parameters

The values from List 5.2 are placed in the combined drive train and generator Simulink model (Fig. 5.1 and Fig. 5.5). To ensure that the design of the Simulink model is correct, it is necessary to establish the transfer function and poles of the model and compare it with the Strathclyde model.

The transfer function of the Strathclyde model, relating the aerodynamic torque, T_A, to the generator reaction torque, T_G is:

$$\frac{T_G}{T_A} = \frac{12783}{(s^2 + 6.72s + 52.92)(s^2 + 26.7s + 9825.7)}$$

$$= \frac{12783}{s^4 + 33.4s^3 + 10058.04s^2 + 67441.66s + 519976.04} \qquad (5.6)$$

Matlab can be used to derive the poles of the transfer function. The numerator and denominator of the transfer function can be set up as two arrays ('num' and 'den'). The following command is used to assess the zeros (z) , poles (p) and gain (k) of a transfer function:

[z,p,k] = TF2ZP(num,den)

The command for the transfer function of Eqn. (5.6) results in the following complex conjugate poles:

p1 = -13.34 + j98.22
p2 = -13.34 - j98.22
p3 = -3.36 + j6.4521
p4 = -3.36 - j6.4521

The root locus of these poles is shown in Appendix 5a.

5.3.1 Comparison of the Model Data with Provided Strathclyde Data

To ensure that the WECS model is compatible within the Simulink/Matlab environment, it is necessary to compare the Simulink model with the parameters of the Strathclyde model. The method used is to derive the transfer function and poles of the Simulink model and compare them with the values calculated above. To establish the transfer function, the state space representation of the model has to be first determined.

Matlab includes a facility which can be used to derive the linear state space model of a Simulink model. The 'linmod' instruction returns the values of the state matrix A, the

input matrix B, the output matrix C and the direct transmission matrix D of the following state space representation:

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du \qquad (5.7)$$

x is the state vector, u is the input vector and y is the output vector.

To use 'linmod' with a Simulink model requires the inclusion of an 'inport' and 'outport' module on the respective model input and output. The syntax of the 'linmod' command is:

[A,B,C,D]=LINMOD('SFUNC')

where SFUNC is the name of the Simulink model. Using the combined drive train and generator model (with the relevant port connections) the 'linmod' instruction results in Matlab returning the values of A, B, C and D. The response shown in Appendix 5a.

The state space model can be converted into a transfer function, for comparison with Eqn. (5.6), using the state space to transfer function instruction 'ss2tf', which has the syntax:

[num,den] = SS2TF(A,B,C,D)

As before, Matlab returns the values of 'num' and 'den' and the response to the instruction is again shown in Appendix 5a.

The coefficients resulting in the command refer to descending powers of the Laplace function, s. The last coefficient referring to s^0. From the appendix it can be seen that the

first coefficient refers to s^5 instead of the expected s^4. The transfer function, therefore has the form:

$$T.F.(s) = \frac{12782.71s}{s^5 + 33.41s^4 + 10057.94s^3 + 67427.69s^2 + 519958.98s} \tag{5.8}$$

Dividing Eqn. (5.8) through by 's' gives similar values seen in Eqn. (5.6). The slight discrepancies could be due to the Matlab processes, or rounding errors.

The poles of the equation are found to be:

p1 = -13.3481 + j98.2216
p2 = -13.3481 - j98.2216
p3 = -3.3593 + j6.4524
p4 = -3.3593 - j6.4524

A comparison of both the poles the transfer functions of the two models indicates that the Simulink model is comparable with the Strathclyde model.

5.4 Developing the Model Parameters of the RAL 45kW WECS

The RAL 45kW WECS is manufactured by Windharvester. It is a fixed-pitch, stall-regulated, three bladed machine and has a lattice tower support. Its gearbox arrangement consists of a 19.58:1 gearbox coupled to a 2:1 pulley and belt resulting, in a total ratio of 39.16. The generator used is a 45kW, two pole-pair induction machine (IM).

List 5.3 details the available machine parameters, for the WECS, which were obtain from RAL [5.1].

Rotor Inertia, J_R	*14145 kgm^2*
Hub Inertia, J_H	*22.8 kgm^2*

Gearbox Inertia (referred to LSS),	*34.2 kgm^2*
Large Pulley Inertia	*2.62 kgm^2*
Small Pulley Inertia	*0.376 kgm^2*
Generator Shaft Inertia	*0.378 kgm^2*
LSS Stiffness, K_1	*3.36x10^6 Nm/rad*
HSS Stiffness, K_2	*2.13x10^3 Nm/rad*
Total gearbox and Pulley Ratio	*39.16:1*
Low Speed Rotational Frequency	*4.01 rad/s*
First Flapwise Frequency	*16.34 rad/s*
Second Flapwise Frequency	*67.86 rad/s*
First Edgewise Frequency	*29.66 rad/s*
Rotor Diameter	*16.90 m*
Rotor Speed	*38.3 rpm*

List 5.3 Available RAL 45kW Data

The information in List 5.3 can be used to ascertain the simplified WECS model parameters as discussed in Chapter 2. In order to be compatible with the model developed for the Strathclyde WECS, the RAL parameters have to be manipulated to fit the requirements of the representation shown in Fig. 2.11. The following estimation of 'lumped' parameters, required for this particular representation, uses the dynamic relationships established in Chapter 2. Unfortunately, not all the data required for the development of the model (such as stiffness and damping characteristics) are available, so some assumptions have had to be made. Additionally, it has been reported that some of the estimated parameters may be erroneous and, therefore, open to debate [5.2].

It should also be reiterated that both 'I' and 'J' have been used for inertia parameters in keeping with the references.

5.4.1 Estimation of RAL Drive Train Model

(a) LSS Stiffness (K_1)

The LSS shaft stiffness, K_1, is given as 3.36×10^6 Nm/rad. It is not clear from the data how this was measured and if it is of the same format derived earlier. Comparing the data with the Strathclyde data given in List 5.2, it can be seen that the values are comparable. Since no other stiffness values (such as hub, rotor stiffness, see Eqn. (2.19)) have been measured and there is no available readings for the damping or stiffness of the gearbox mounting (Eqn. (2.22)), it is difficult to establish the effective value of K_1. It is assumed that the quoted value will, initially, suffice.

(b) HSS Stiffness (K_2)

As with K_1, no information has been given regarding the process of measurement for the HSS stiffness value, K_2. Similarly, since there is insufficient gearbox dynamic data, it is assumed the given value of 2.13×10^3 Nm/rad is sufficient for initial simulation purposes.

(c) Lumped Rotor Inertia (I_1)

From Eqn. (2.20), it can be seen that the equivalent lumped LSS inertia I_1, predominately consists of the effective rotor inertia, modified by the product of the effective gearbox inertia and stiffness ratio.

$$I_1 = J_R^* + \frac{K_1}{K_1 + N^2 K_2} J_{ss}^* \tag{2.20}$$

The effective rotor inertia, J_R^* can be assessed from Eqn. (2.18) and requires the stiffness values of both the rotor and the hub which are unavailable.

$$J_R^* = J_R\left(1 + \frac{J_H}{J_R}\frac{K_R}{(K_R + K_H)}\right) \tag{2.18}$$

The stiffness ratio can be assumed to be small since, in general, $K_H >> K_R$ [2.8]. Additionally, the inertia ratio can be calculated since the values of J_H and J_R have been measured (List 5.3).

$$\frac{J_H}{J_R} = \frac{22.8}{14145} = 1.612e^{-3} \tag{5.9}$$

Assuming that the stiffness ratio is approximately 1/10, substituting back into Eqn. (2.18)[2.8]:

$$J_R^* = 14145(1 + 1.612e^{-3} \times 0.1) = 14147.3 \text{ kgm}^2 \tag{5.10}$$

The effective gearbox inertia can be calculated using the formula:

$$J_{ss}^* = J_s + J_H \frac{K_R}{(K_R + K_H)} \tag{5.11}$$

where J_s is the gearbox inertia referred to the LSS. Using the same assumption used previously for the stiffness ratio:

$$J_{ss}^* = 34.2 + 22.8 \times 0.1 = 36.48 \text{ kgm}^2 \tag{5.12}$$

Substituting in Eqn. (2.20) to find I_1:

$$I_1 = J_R^* + \frac{K_1^*}{K_1^* + N^2 K_2^*} J_{ss}^*$$

$$I_1 = 14147.3 + \frac{3.36 \times 10^6}{3.36 \times 10^6 + 39.16^2 \times 2.13 \times 10^3} 36.48 \tag{5.13}$$

$$I_1 = 14165.8\ \text{kgm}^2$$

It should be noticed that comparison of this value and rotor inertia, J_R in List. 5.3, shows very little difference. The percentage change is minimal and indicates that little error is encountered if the value of the rotor inertia is adopted for I_1 in the model.

(d) Lumped Generator Inertia (I_2)

I_2 can be calculated from Eqn. (2.21):

$$I_2 = J_G + \frac{K_2^*}{K_1^* + N^2 K_2^*} J_{ss}^*$$

J_G is given in List 5.3 and is 0.378 kgm^2, therefore:

$$I_2 = 0.378 + \frac{2.13\times10^3}{3.36\times10^6 + 39.16^2 \times 2.13\times10^3} 36.48 \qquad (5.14)$$

$$I_2 = 0.3897\ \text{kgm}^2$$

Once again it can be seen that there will be little error if the generator inertia is used directly as the value of I_2.

(e) LSS and HSS Damping Constants (γ_1 and γ_2)

γ_1 and γ_2 are both dependent on the damping characteristics of the gearbox and the bearing losses of the respective shafts. Unfortunately, no data is available on any damping effects or shaft losses of the RAL WECS.

For preliminary simulation purposes, it will be assumed that the shaft losses, as a percentage of the rated power, are the same for the 45kW and 330kW machines. From

List 5.2 the values of γ_1 and γ_2 for the 330kW machine are 500 and 0.3 respectively. Assuming that the LSS and HSS speeds are 4 rad/s and 157.07 rad/s, respectively, the corresponding values for the RAL WECS model can be found as follows:

(i) Calculation of γ_1

Calculating the LSS power losses for the 330kW machine:

$$PL_{LSS-330} = \gamma_1 \omega_{ls}^{\ 2} \tag{5.15}$$

$$PL_{LSS-330} = 500 \times 4^2 = 8kW$$

expressing this as a percentage of the rated power:

$$\%loss = \frac{8}{330} = 2.42\% \tag{5.16}$$

The LSS damping factor for the 45kW machine would therefore be equal to:

$$\gamma_1 = \frac{45 \times 10^3 \times 2.42\%}{4^2} = 68.06 \tag{5.17}$$

(ii) Calculation of γ_2

Power loss on the HSS of the 330kW machine:

$$PL_{HSS-330} = \gamma_2 \omega_{hs}^{\ 2} \tag{5.18}$$

$$PL_{HSS-330} = 0.3 \times 157.07^2 = 7.4kW$$

the loss as a percentage of the rated power

$$\%loss = \frac{7.4}{330} = 2.24\% \tag{5.19}$$

from which the HSS damping factor of the 45kW machine can be estimated as:

$$\gamma_2 = \frac{45 \times 10^3 \times 2.24\%}{157.07^2} = 0.041 \tag{5.20}$$

The values selected for γ_1 and γ_2 can be assessed when the model is validated.

5.4.2 Estimating the First Order Model of the RAL Generator

Chapter 2 and Appendix 2a detailed the development of a first order model for an induction machine using the machine parameters of the equivalent circuit to establish both the torque/speed slope, D_e, and the machine time constant, τ (see Fig 5.5). The relevant parameters for the RAL 45kW machine are shown in List 5.4 [5.1].

Stator Resistance, rs	*0.1056* Ω
Rotor Resistance Referred to the Stator, rr	*0.0787* Ω
Stator Leakage Reactance, xls	*0.4356* Ω
Rotor Leakage Reactance Referred to the Stator, xlr	*0.4148* Ω
Magnetising Reactance, xm	*8.8095* Ω
Peak Line Voltage, E	$415\sqrt{2}$ *V*
Number of Pole Pairs, p	*2*

List 5.4 RAL 45kW Machine Parameters

A Matlab program, 'genral.m', was written to implement the equations of Appendix 2a. The listing of the program is shown in Appendix 5a.

Executing the program in Matlab results in:

$$D_e = 71.9838 \text{ Nm/rad/s}$$

$$\tau = 0.0365 \text{ s}$$

The equivalent values of the Strathclyde model, from List 5.2, show that the time constants are comparable, but there is a great difference in the values of D_e. This is to be expected considering the relative sizes of the generators. Chapter 6 assesses the effects of replacing large size generators with smaller ones. This will be necessary for the use of HILS in the laboratory.

The calculated generator values, along with those calculated for the drive train, can now be placed in the WECS model in Simulink. The characteristics of the model can be assessed and the transfer function and poles established.

5.4.3 Summary of Parameters and Transfer Function of the RAL Model

All of the calculated parameters for the RAL are shown in List 5.5

$I_1 = 14165.8 \text{ kgm}^2$
$I_2 = 0.3897 \text{ kgm}^2$
$K_1 = 3.36 \times 10^6 \text{ Nm/rad}$
$K_2 = 2.13 \times 10^3 \text{ Nm/rad}$
$N = 39.16$
$\gamma_1 = 68.06$
$\gamma_2 = 0.041$
$D_e = 71.9838 \text{ Nm/rad/s}$
$\tau = 0.0365 \text{ s}$

List 5.5 RAL Model Parameters

The transfer function of the model can be established in Matlab using the 'linmod' and 'ss2tf' commands (see section 5.3.1) and is:

$$T.F.(s) = \frac{15109.72}{s^4 + 27.5s^3 + 7952.13s^2 + 79184.4s + 592398.5} \tag{5.21}$$

A comparison of the RAL transfer function with the Strathclyde transfer function of Eqn. (5.8) shows that there is some similarity in the form of each.

5.5 Developing the WECS Aerodynamic Models in Simulink

Chapter 2 contains the description of the aerodynamics experienced by a WECS. The aerodynamic interaction between the wind and the WECS can be modelled in Simulink in a modular fashion. This will allow each particular aerodynamic effect to be assessed in turn and emphasise the importance of including it in the modelling. Initially, the development of aerodynamic torque due to the wind speed alone, will be considered to assess the dynamic response of the WECS drive train.

5.5.1 Development of Aerodynamic Torque from the Wind Speed

Eqn. (2.11) describes the available aerodynamic torque due to an effective wind speed. To recall, aerodynamic torque, T_a:

$$T_a = C_q \frac{1}{2} \pi \rho V^2 R^3 \tag{2.11}$$

The values of the torque coefficient, C_q, depend on the tip speed ratio, λ, of the WECS and are usually modelled as a 'look-up' table due to the non-linear relationship.

No data is available on the C_q-λ characteristics of the Strathclyde WECS, but C_p-λ data is available from RAL for the 45kW machine. The values of C_q can be calculated from the C_p data using the relationship shown in Eqn. (2.12):

$$C_q = \frac{C_p}{\lambda} \tag{2.12}$$

Appendix 5b includes tables of both the C_p-λ data provided by RAL and the calculated C_q-λ values. Fig. 5.7 is a graphical representation of the calculated C_q-λ values which are used to establish the aerodynamic torque in the Simulink model.

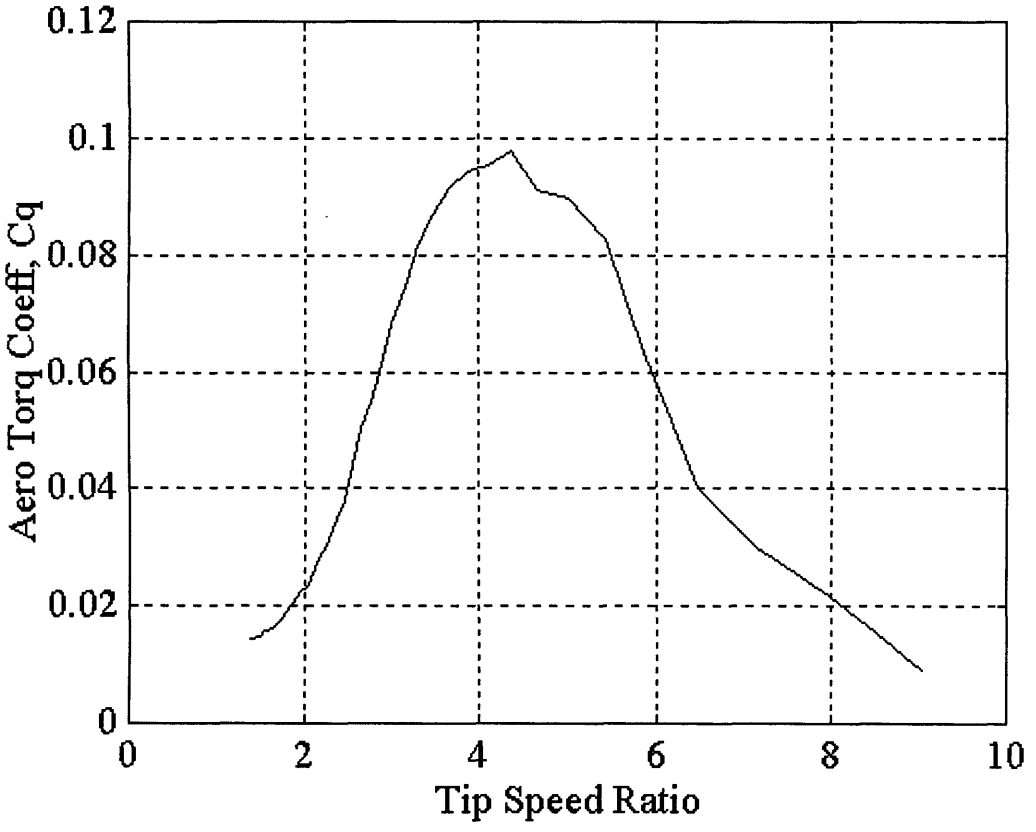


Fig 5.7 C_q-λ Curve for the RAL 45kW WECS

The C_q-λ values are stored in a 'look-up' table within the Simulink model. The model has to be able to calculate the value of the tip speed ratio, λ, and establish it as an input to the

table. The output, C_q, is used to calculate the available aerodynamic torque, T_a, from the relationship of Eqn. (2.11).

Fig. 5.8 shows the Simulink module used to derive T_a from the input wind speed data. The output 'Ta' connects to the input of the drive train module.

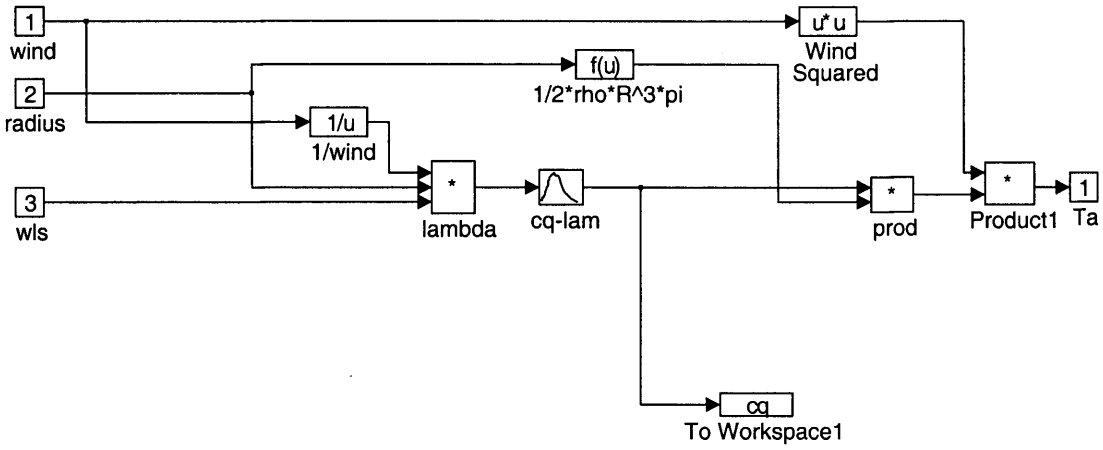


Fig. 5.8 Simulink Module to Calculate T_A from Wind Speed

The combined model of the aerodynamics, drive train and generator of the RAL WECS can be connected together and evaluated to see how the model compares with the actual WECS. Appendix 5c shows the combined model. Chapter 2 discusses how validation of the simulation is achieved by comparing the power spectral density of the modelled and measured LSS torque. To establish an accurately simulated LSS torque, a real driving function, in this case a series of measured wind speeds, is required for the model. Additionally, measurement of the actual LSS torque, with respect to the same wind speed series has to be available.

RAL has available a number of Matlab '.mat' files containing measured variables from the 45kW WECS test site. Measured variables include a wind speed series and the resultant LSS torque.

5.5.2 Measurement Data from the RAL 45kW WECS

Four Matlab data files ('.mat') containing measured data from the RAL test site are available. The measured parameters in each of the data files are:

- Wind Speed
- LSS Torque
- Wind Direction
- Electrical Power Output

The data files contains 600s (10 min) of 50Hz data with measurements over a wide range of wind speed. Table 5.1 lists each data file and its average wind speed. The variation allows the performance of the WECS to be assessed over a wide range of wind speed input.

RAL Data File	Average Wind Speed (m/s)
RL020702.mat	7.9057
RL060602.mat	8.6399
RL061302.mat	14.5213
RL101902.mat	17.1682

Table 5.1 Measured RAL Data Files [5.1]

With this information the model of the WECS can be compared with the measured data of the actual WECS through simulation in Simulink. Firstly, the simulation parameters of Simulink have to be selected. Variables such as the time step and integration method can be specified to suit the model under simulation.

5.5.3 Selection of Simulink Simulation Variables

Simulink allows the selection of a number of integration methods for use during simulation of models. Simulink recommends different integration methods depending on the characteristics of the model to be simulated [4.4].

(a) Linsim

Linsim is recommended for stiff systems (i.e. a model containing both fast and slow dynamics) and systems that are primarily linear models with a few non-linear blocks.

(b) Runge-Kutta rk23 and rk45

Recommended for highly non-linear and/or discontinuous systems but not stiff systems. Performs well for a mixed continuous and discrete model. Rk45 is faster and more accurate than the rk23 but produces fewer output points. Rk23, therefore, can be the preferred method since it provides a 'smoother' output.

(c) Gear

Recommended for use with smooth, non-linear models and stiff systems. Not good for systems with rapidly changing inputs.

(d) Adams

Used with models similar to those recommended for Gear but with a smaller variation of time constants.

(e) Euler

Recommends that Euler is only used to confirm the results of other integration methods as it requires much smaller step sizes than the other methods, therefore increasing processing time, to produce the same accuracy.

The Real-Time Workshop only allows the use of a small number of the above integration methods, namely, the Runge Kutta methods and the Euler.

Since the WECS model contains non-linearity's, such as the aerodynamic torque calculation and the alternative gearbox model, the Runge-Kutta methods will be used. Additionally, both methods are available for the normal simulation environment as well as real-time simulation. This will be useful for comparing the software-only simulation with the HILS, real-time simulation. The rk45 method has been selected for its accuracy.

(f) Selection of the Simulation Time Step

Minimum and maximum simulation step sizes can be selected according to the dynamics of the model under investigation. Additionally, a tolerance level can be set which controls the relative error of integration at each step of the integration. The tolerance level dictates the step size of the simulation, within the maximum and minimum limits. This can lead to a variable step size throughout the simulation, which is not acceptable for real-time operation where the step size has to be fixed.

Simulation with a fixed step size during normal simulation can be realised by setting both the maximum and minimum step size to the same value. This is the preferred option for WECS simulation since it is necessary to compare the real-time, fixed step simulation with the software-only simulation.

Since the data provided by RAL is measured at 50Hz the initial simulation step size is selected to be fixed at 0.02s. This step size may need to be increased during real-time operation due to the limitations of the serial communications (see Chapter 6).

5.5.4 Model Simulation Using the RAL Measured Data

Since the model of the drive train, generator and basic aerodynamics is complete and a realistic drive function is available in Matlab, simulation of the model can be initiated.

The wind speed series and the corresponding LSS torque have to be loaded from the Matlab data file. Initially, the RL020702.mat data is to be used. Simulink must have

access to the wind speed data during simulation. This can be achieved using a 'from_workspace' module which requires the name of the parameter to be accessed as well as the relevant sample time of the data. Therefore, a time series array, corresponding to the 50Hz sampling time, has to be devised.

List 5.6 shows the Matlab program designed to load the RAL data file and create the 50Hz time series array. The torque conditioning is necessary to convert the kNm measurements to Nm. Additionally, the torque data series is not complete, so the length of the array is not the same as the time array. Matlab and Simulink do not allow uneven arrays for certain processes such as the 'plot' and 'from_workspace' commands.

Comparison, with the measured electrical power data indicates that the torque values are missing from the end of the data series. Therefore, 'dummy' values are placed at the end of the torque array to ensure it is the same length as the other data arrays.

```
%Program to load all the required info for WECS model using RAL data
clear;                                  % clear memory
T1=0:0.02:600.08;                       % create time array
load rl020702.mat;                      % open RAL data
torque=torque*1000;                     % convert kNm to Nm
torque(30001:30005)=[6e3 6e3 6e3 6e3 6e3];  % dummy values to lengthen array
ral                                     % open RAL model in Simulink
```

List 5.6 Matlab Program to Condition RAL Data

The model is simulated using the wind speed data for 300s. The 300s simulation time was selected instead of the 600s to speed up processing time and reduce memory requirements.

Fig. 5.9 shows the resultant LSS torque time series, tls, after the 300s simulation and the measured LSS torque from data file RL020702.mat.

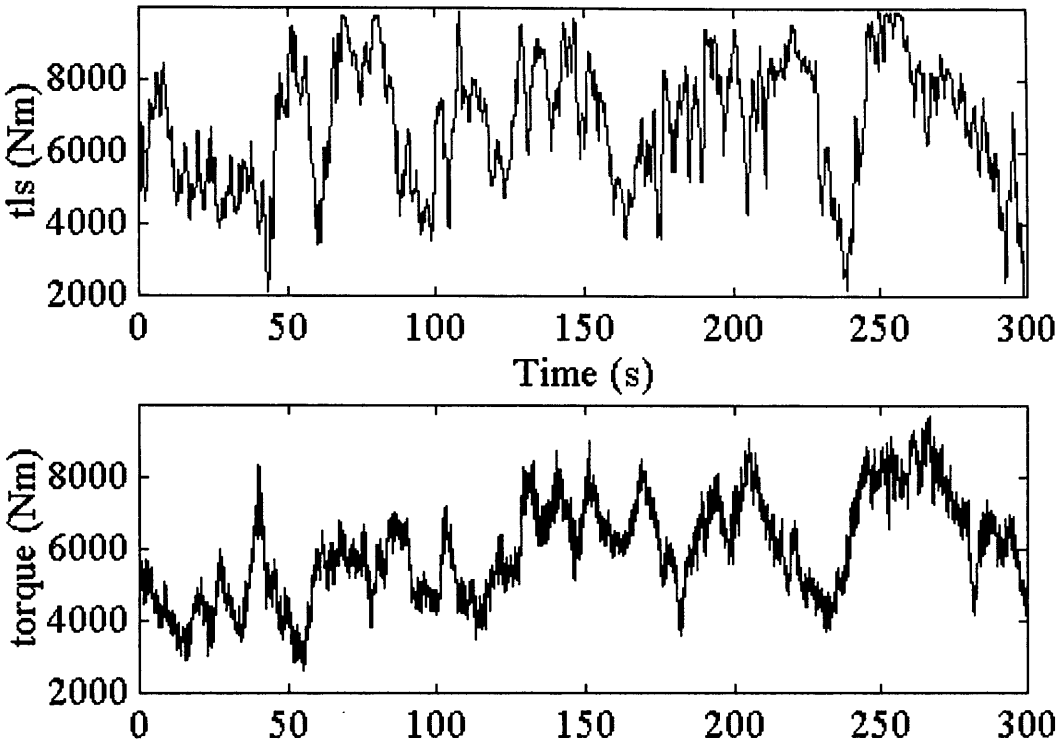


Fig 5.9 Measured and Simulated LSS Torque of the 45kW WECS

A dynamic comparison of the time series data is not possible. The main reason for this is that the wind speed is measured some distance away from the WECS and the LSS torque is a response to a delayed wind speed. The simulation, on the other hand, assumes that the WECS experiences the measured wind speed at the time of measurement. As an alternative, Wilkie et al proposed a comparison of the power spectral densities of the two LSS torque data series [2.18].

(a) Power Spectral Density

The power spectral density (PSD), Pxx, of a data array, X can be calculated in Matlab using the following command:

Pxx = PSD(X,NFFT,Fs)

where NFFT is the length of the Fast Fourier Transform (FFT) and Fs is the sampling frequency [5.3]. The resolution of the PSD is defined by length of the FFT. Initial investigations indicate that an FFT length of 1024 provides a suitable resolution to investigate dynamic characteristics such as rotational sampling. Additionally, Fs was chosen to be 50Hz to match the RAL data files.

The Matlab program designed to compare the PSD of the measured LSS torque with the simulated results is shown in Appendix 5c.

The resultant plot of the program of is shown in Fig. 5.10.

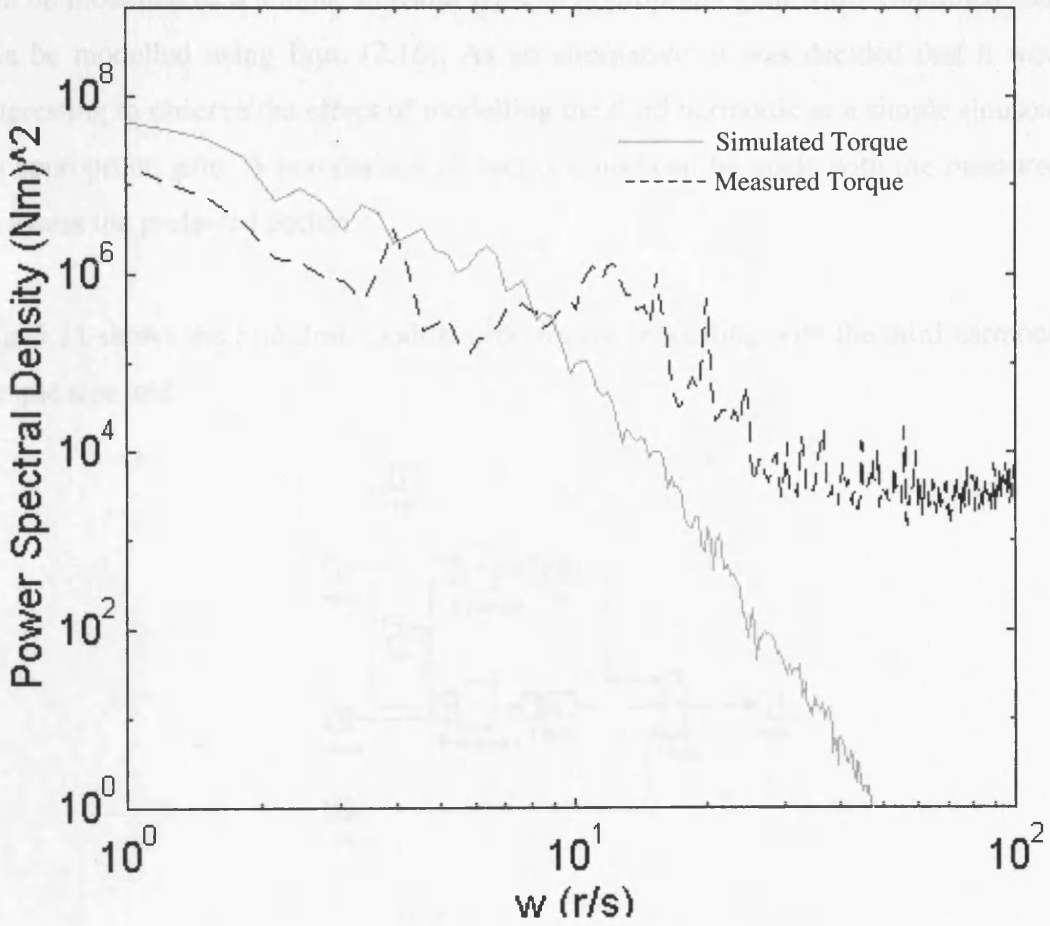


Fig 5.10 Power Spectral Density Comparison of Measured and Simulated Torque

The effect of rotational sampling on the measured site data can clearly be seen at ~4 rad/s and ~12rad/s, corresponding to the rotational speed, ω and 3ω (the blade passing frequency). The simulated torque is without the effects of rotational sampling, spatial filtering or induction lag. Additionally, the comparison shows that there is too much energy within the low to intermediate frequency range, as discussed in Chapter 2. There is a noticeable difference at the higher frequency, which again, is probably due to the omission of some of the dynamics and/or noise in the real system.

(b) Rotational Sampling

As stated in Chapter 2, the effects of the spectral peak at the rotor rotational frequency can be modelled as a simple sinusoid with an appropriate gain while rotational sampling can be modelled using Eqn. (2.16). As an alternative, it was decided that it would be interesting to observe the effect of modelling the third harmonic as a simple sinusoid with an appropriate gain. A comparison of each method can be made with the measured data to assess the preferred option.

Fig. 5.11 shows the Simulink module used for the modelling with the third harmonic as a simple sinusoid.

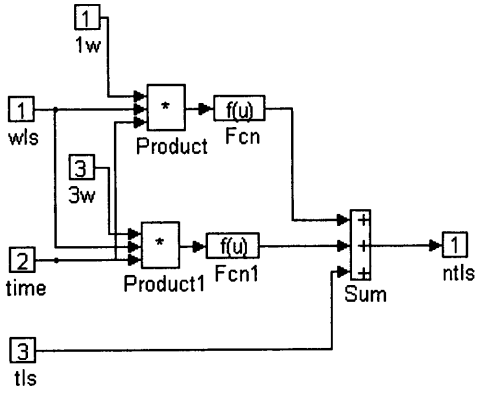


Fig. 5.11 Model of Rotational Sampling Using Simple Sinusoid Representation

The model adds two scaled sinusoids to the LSS torque, tls. The 'Fcn' component contains the following:

$$3e2*(\sin(u)+\cos(u)) \tag{5.22}$$

'u' is the input to 'Fcn', in this case the LSS rotational speed. The gain was selected iteratively. Similarly, 'Fcn1' has the format:

$$2.5e2*(\sin(u)+\cos(u)) \tag{5.23}$$

Once again, the gain was selected iteratively.

Fig. 5.12 shows the result of the PSD comparison between measured and simulated LSS torque with the above module. The comparison shows that there is an improvement at ω and 3ω on the simulated torque. Additionally, the higher frequency response has been improved.

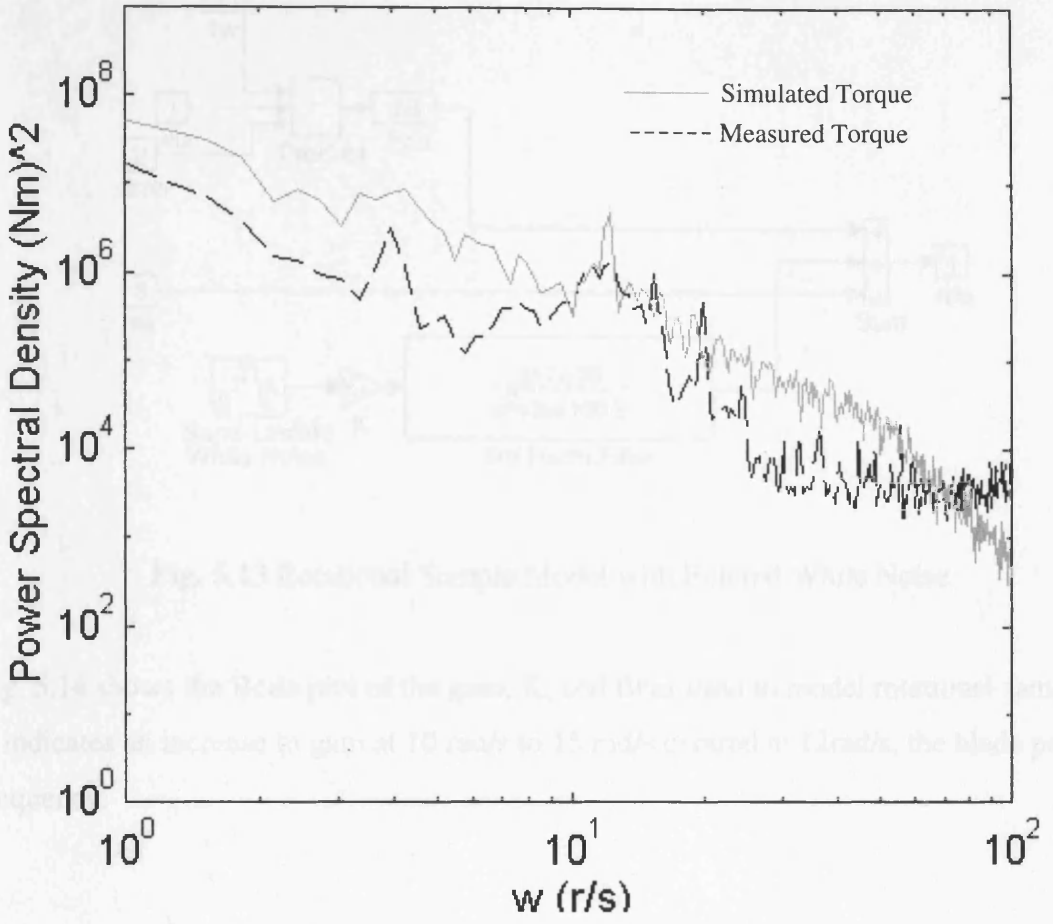


Fig 5.12 PSD Comparison with Rotational Sampling Included

Fig. 5.13 shows the model used to realise Eqn. 2.16 using filtered white noise to represent the third harmonic of the rotational sampling [2.14].

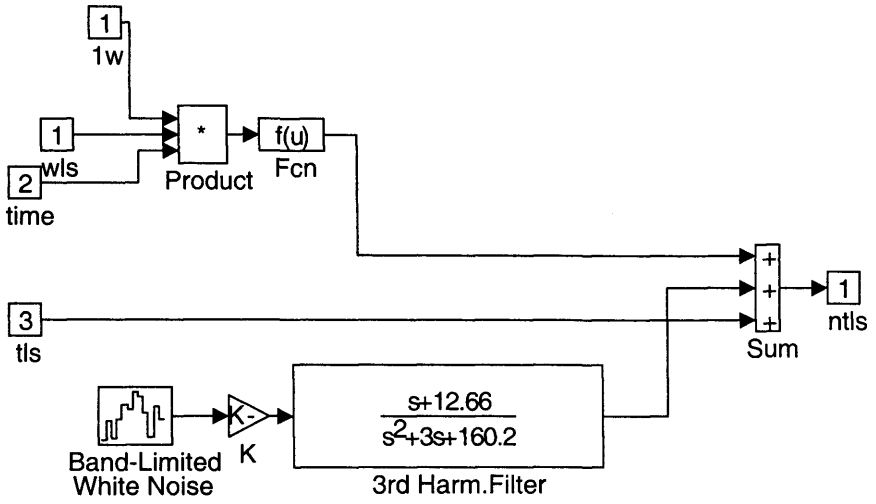


Fig. 5.13 Rotational Sample Model with Filtered White Noise

Fig. 5.14 shows the Bode plot of the gain, K, and filter used to model rotational sampling. It indicates an increase in gain at 10 rad/s to 15 rad/s centred at 12rad/s, the blade passing frequency.

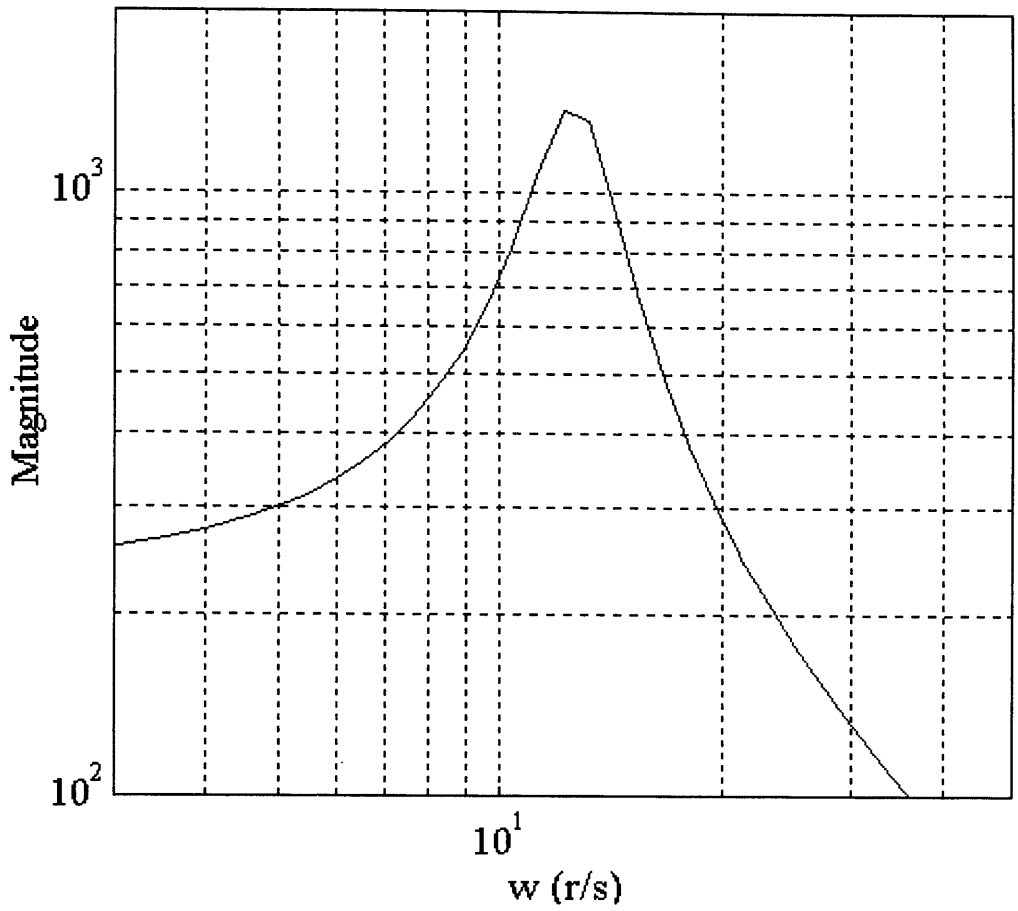


Fig 5.14 Bode Plot of the Third Harmonic Filter

Fig 5.15 shows the PSD of the measured and simulated torque using the filtered white noise model for the rotational sampling effects.

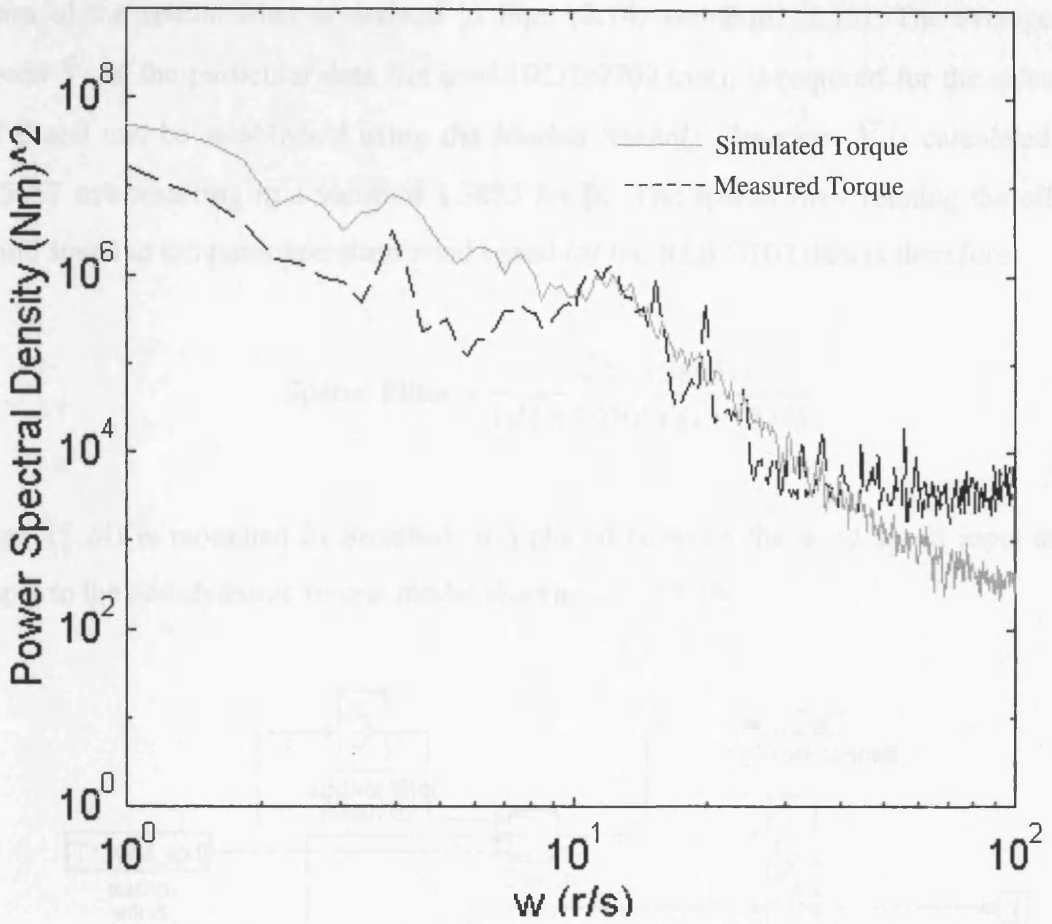


Fig 5.15 PSD Comparison Using Filtered White Noise for Rotational Sampling

Comparing Fig. 5.15 with Fig 5.12, it can be seen that the simulation with the filtered white noise gives a more accurate representation of the rotational sampling at approximately 12 rad/s. The simple sinusoid method of Fig. 5.12, does not have a large enough bandwidth at this point. The filtered white noise method is, therefore, more preferable.

(c) Spatial Filtering

The wind speed data used for the simulation is measured using an anemometer located near the WECS. The anemometer measures a point spectrum wind speed as mentioned in Chapter 2. To establish the effective wind speed experienced across the rotor of the

WECS, the model has to include spatial filtering of the point spectrum wind speed. The form of the spatial filter is derived in Eqn. (2.14) and Eqn. (2.15). The average wind speed $\overline{V}$ of the particular data file used (RL020702.mat), is required for the calculation of β and can be established using the Matlab 'mean()' function. $\overline{V}$ is calculated to be 7.9057 m/s resulting in a value of 1.3895 for β. The spatial filter relating the effective wind speed to the point spectrum wind speed for the RL020702 data is therefore:

$$\text{Spatial Filter} = \frac{(\sqrt{2}+1.3895s)}{(\sqrt{2}+1.0305s)(1+1.874s)} \tag{5.24}$$

Eqn. (5.24) is modelled in Simulink and placed between the wind speed input and the input to the aerodynamic torque model shown in Fig. 5.16.

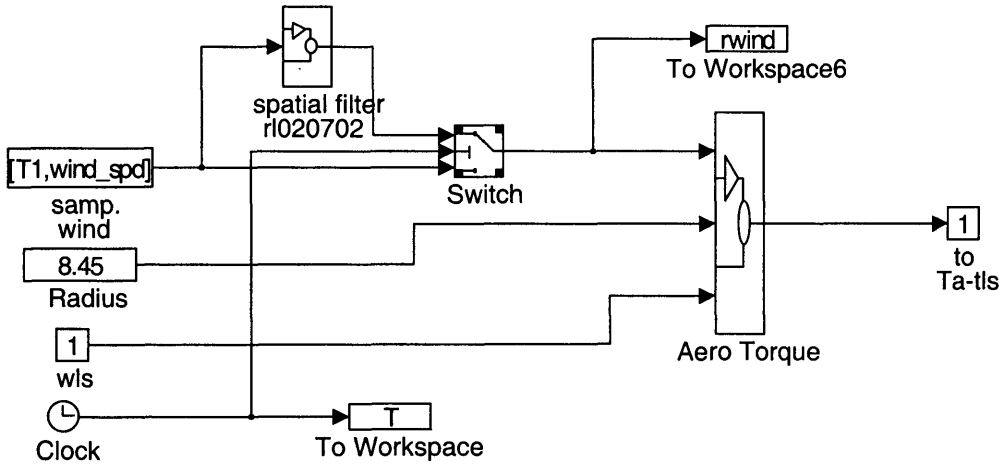


Fig. 5.16 Model Including Spatial Filter

The switch is used to avoid the initial effective wind speed being zero due to the integrating effect of the spatial filter. Without it, the simulation 'crashes' due to the divide by zero value of the wind when calculating the value of λ. Unfortunately, the initial conditions of the transfer function modules, used for modelling the spatial filter, can not be pre-set, hence the presence of the switch. The point spectrum is used for 2s before the filter is 'switched in'.

Simulation of the model with the spatial filter included, modifies the frequency response of the model. Fig 5.17 shows the PSD comparison.

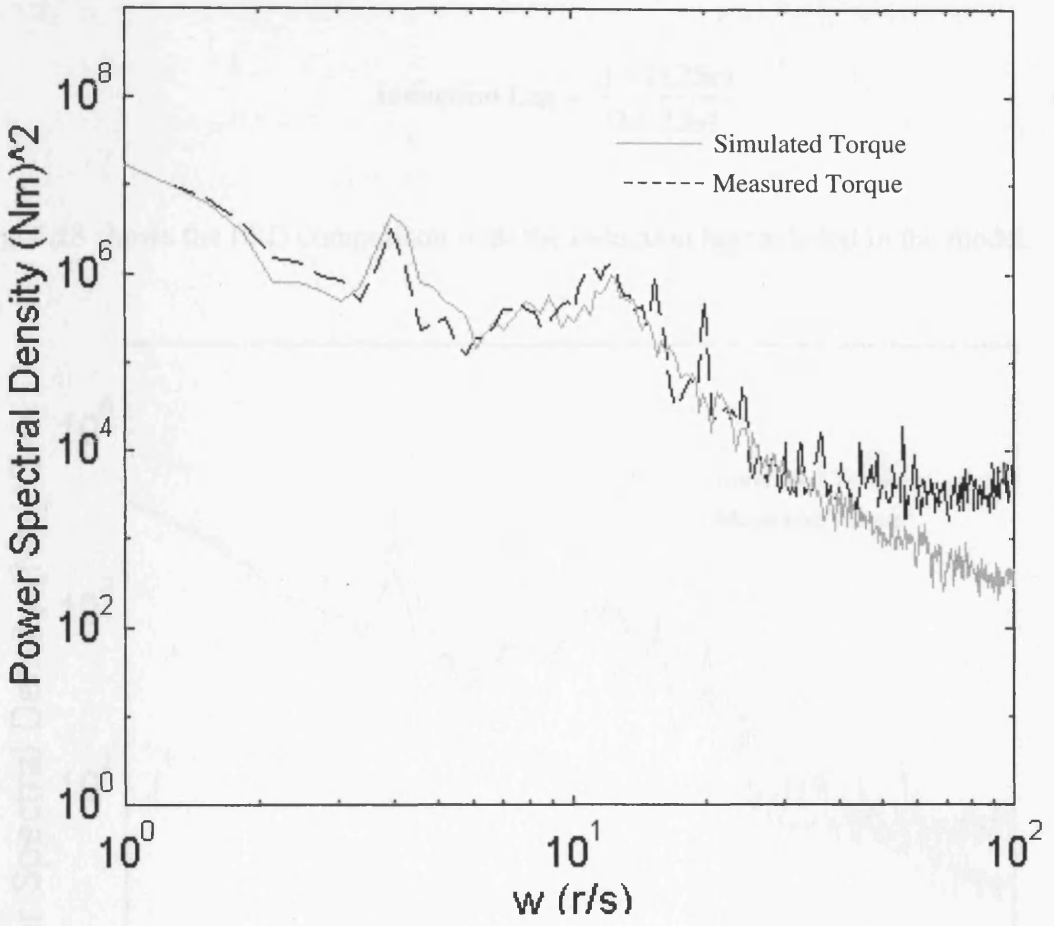


Fig. 5.17 PSD Comparison Including Spatial Filter

Comparison between Fig. 5.17 with Fig. 5.15 shows that the spatial filter has improved the frequency response of the model considerably. Adjusting the respective gains of the simulated rotational effects can improve the results further but this should be considered after the effects of the induction lag have been investigated.

(d) Induction Lag

Modelling the change of wake effect induction lag due to the presence of a WECS can be achieved by lead-lag filtering the simulated LSS torque. Eqn. (5.25) shows the values chosen for the lead-lag filter used to modify the torque [5.4]:

$$\text{Induction Lag} = \frac{(1+11.25s)}{(1+7.5s)} \tag{5.25}$$

Fig. 5.18 shows the PSD comparison with the induction lag included in the model.

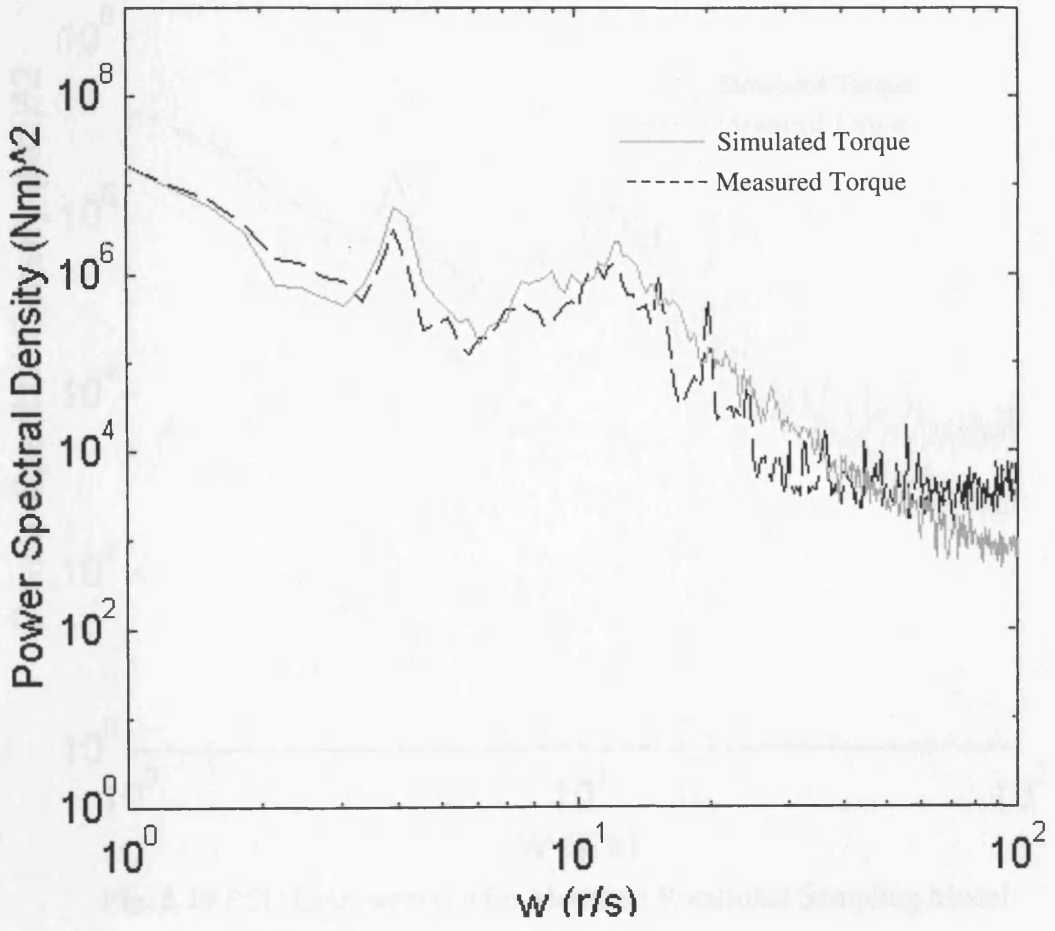


Fig 5.18 PSD Comparison Including Induction Lag Model

Comparing Fig 5.18 with Fig 5.17 shows that there is an overall increase in gain at mid to high frequencies as expected. The gain is particularly noticeable at the rotational sampling frequencies. This can be rectified by reducing the gains of the rotational sampling module 'noise harmonic'. Initially, the value of the first harmonic gain was, iteratively, set to 300. Using a similar technique, the gain found to suit the measured profile is 200. Similarly, the gain at the blade passing frequency was initially, set to 3000. Further experimental results suggests that it should be set to 2200. Fig. 5.19 shows the resultant comparison with the modified gains.

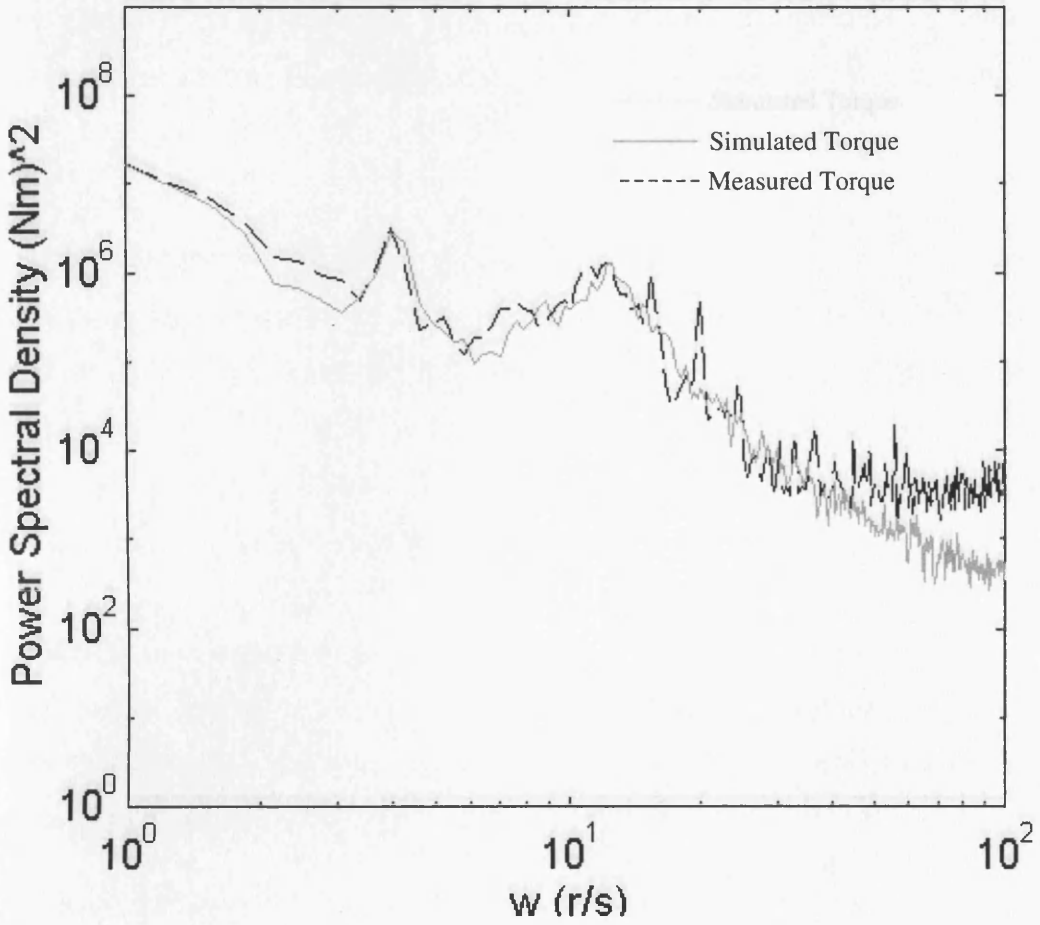


Fig. 5.19 PSD Comparison with Modified Rotational Sampling Model

Fig. 5.19 shows the gain improvement at the harmonic frequencies of rotational sampling. It is noticeable at the blade passing frequency that the simulated torque profile

is slightly shifted to the 'right' of the frequency scale. This is probably due to the characteristics of the white noise filter used to shape the response. The example used (Fig. 5.13 and Fig. 5.14) is designed for a harmonic frequency of $\sqrt{160.2}$ rad/s or 12.66 rad/s. The data from RAL states that the rotational frequency of the rotor is 4.01 rad/s, therefore, the third harmonic is 12.03 rad/s ($\sqrt{144.72}$ rad/s). Modifying the white noise filter to account for this difference in the centre frequency results in the PSD comparison shown in Fig. 5.20.

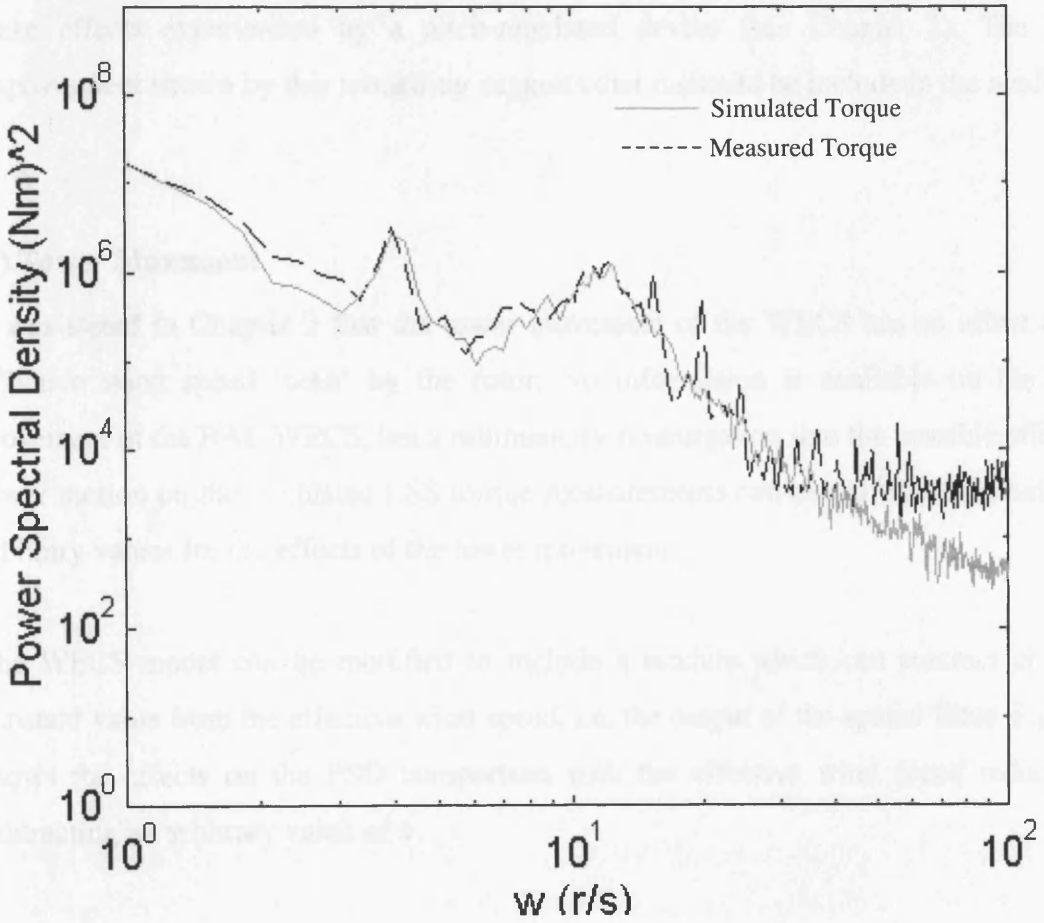


Fig 5.20 PSD Comparison With Modified White Noise Filter

Fig 5.20 suggests that the modification makes the comparison more acceptable, since the simulated blade passing frequency response compares more favourably with the measured data than that of Fig. 5.19.

It could be argued that the introduction of the induction lag does little to improve the response shown with the introduction of the spatial filter. Comparing Fig. 5.20 and Fig. 5.17 shows that there is very little difference. This could possibly be due to the fact that the RAL WECS under simulation is a stall-regulated device and not susceptible to the wake effects experienced by a pitch-regulated device (see Chapter 2). The minor improvement shown by this modelling suggests that it should be include in the model.

(e) Tower Movement

It was stated in Chapter 2 that the tower movement of the WECS has an affect on the effective wind speed 'seen' by the rotor. No information is available on the tower movement of the RAL WECS, but a rudimentary investigation into the possible effects of tower motion on the simulated LSS torque measurements can be performed by selecting arbitrary values for the effects of the tower movement.

The WECS model can be modified to include a module which can subtract or add a constant value from the effective wind speed, i.e. the output of the spatial filter. Fig. 5.21 shows the effects on the PSD comparison with the effective wind speed reduced by subtracting an arbitrary value of 1.

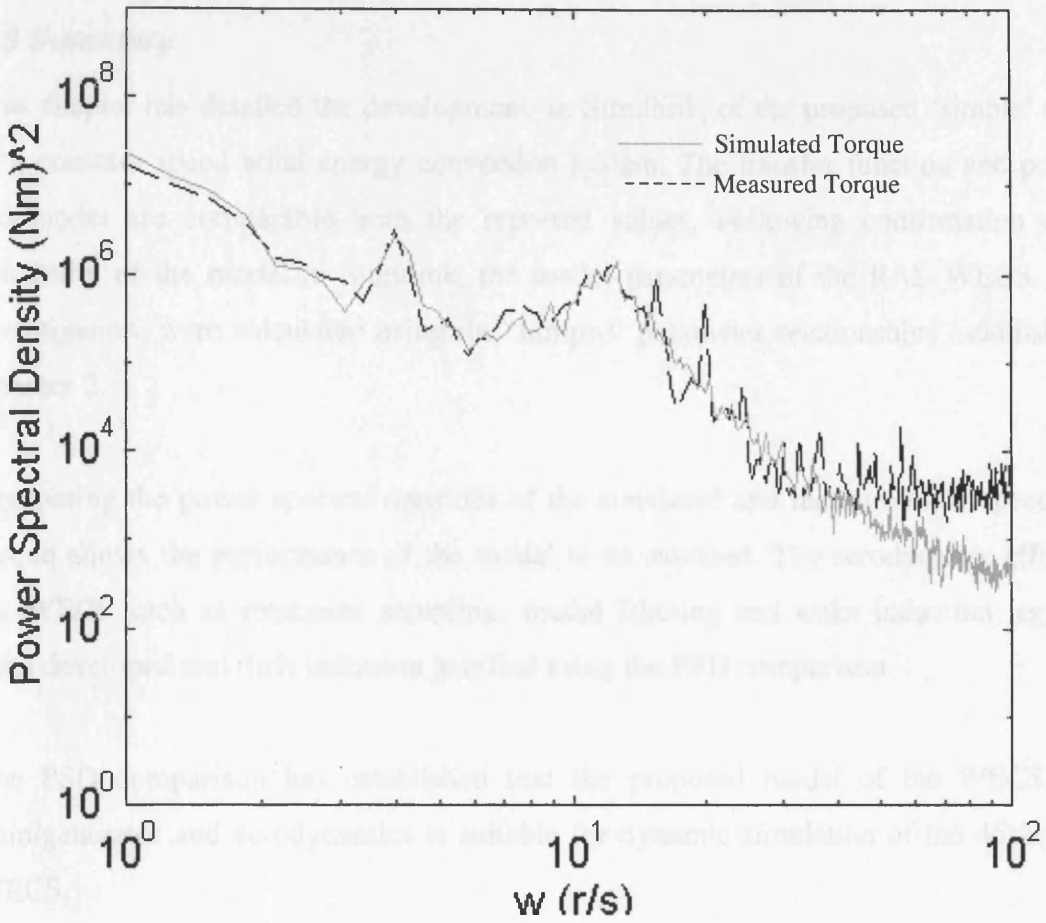


Fig. 5.21 PSD Comparison Including Simulation of Tower Motion

The effect of introducing the possible affects of tower motion have resulted in an increase in low frequency gain. The effect of adding an arbitrary value to the effective wind speed is to reduce the low frequency response.

It could be argued that the spatial filter could be adapted to compensate for the effects of tower motion at low frequency, but this would alter the higher frequency response of the model and affect the modelling of the rotational sampling. Since the values used here for the motion are totally arbitrary and no data is available on the influence of the motion, it is preferable to model the RAL WECS without any tower motion dynamics. It is assumed that any discrepancies between the simulated and measured data is due to the absence of the tower motion in the model.

5.6 Summary

This chapter has detailed the development, in Simulink, of the proposed 'simple' model of a constant speed wind energy conversion system. The transfer function and poles of the model are comparable with the reported values. Following confirmation of the suitability of the model in Simulink, the model parameters of the RAL WECS, under investigation, were calculated using the 'lumped' parameter relationships established in Chapter 2.

Comparing the power spectral densities of the simulated and measured low speed shaft torque allows the performance of the model to be assessed. The aerodynamic effects of the WECS such as rotational sampling, spatial filtering and wake induction lag, have been developed and their inclusion justified using the PSD comparison.

The PSD comparison has established that the proposed model of the WECS drive train/generator and aerodynamics is suitable for dynamic simulation of the 45kW RAL WECS.

The strength of using the selected simulation package has been reinforced. The addition and modification of modules to the WECS basis of the drive train and the generator, was performed with relative ease. Additionally, manipulation of the various tools available, such as the derivation of system poles and transfer functions from state space representation of the model, shows that the package has the added benefit of being a powerful analysing tool.

Chapter 6 Simulating the Effects of Hardware in the Loop

Before including hardware in the loop simulation (HILS) the effects of such an inclusion should be investigated in software. This will provide an indication of the limitations of the simulator and allow the development of suitable compensation. The previous chapters have described the design work required for the development of a fully dynamic wind energy conversion systems (WECS) simulator. A fully dynamic WECS model has been developed, using the Matlab/Simulink design tool, and validated using measured data from an actual WECS. Additionally, the simulator hardware, consisting of a Mentor II DC drive and DC motor driving a grid connected induction machine (IM), has been designed and constructed. A communication link, consisting of serial communications controlled via the Simulink Real-Time Workshop (RTW), has been established between Simulink and the Mentor Drive, allowing real-time control of the DC motor.

Knowing the nature of the hardware and the serial communication link, allows the simulation of effects such as the delay of the drive, to be modelled with relative ease in Simulink. The software will have to include any compensation required to cater for the particular hardware used. This includes compensating for a different size IM and the dynamics and losses of the DC motor, as well as the effects of the DC drive and signal digitisation. Throughout the development, comparisons will be made with the RAL site data and the software simulation of Chapter 5 to verify the modelling.

Initially, the effect of using a different generator size is investigated in the Simulink model to establish whether the concept is valid.

6.1 Replacing the 45kW IM with the 11kW IM in the Simulink Model

The WECS model developed in Chapter 5 is based on the information obtained from Rutherford Appleton Laboratory for a 45kW WECS and a 330kW WECS used by Strathclyde University. Within the laboratory, it is not practical to have a different

generator for simulation of different machines. Additionally, the constraints of the laboratory, mentioned in Chapter 3, limit the size of the IM that can be used. The particular IM used for the simulation is an 11kW two-pole machine.

The study investigates the effects of replacing a, generally, large IM with the smaller 11kW machine. It has to be established whether the substitution is valid with the appropriate compensation for the torque demand and speed difference. To achieve this, a model of the 11kW IM is required.

6.1.1 Establishing a Model of the 11kW IM

The development of a first order model of an IM was detailed in Chapter 5. Both the torque-speed slope and the time constant of the IM can be calculated from the equivalent circuit impedances. The equivalent circuit impedances of the 11kW machine are shown in Appendix 6a [6.1].

The Matlab calculations of the torque-speed slope, D_e, and the machine time constant, τ result in the following values:

$$D_e = 4.0832 \text{ Nms/rad} \quad \text{and} \quad \tau = 0.0256 \text{ s} \tag{6.1}$$

Comparing these values with those of the Strathclyde and RAL WECS shows that the time constants are very similar while the value of D_e reflects the relative size and the number of poles of each IM.

The damping factor, γ_2, and the inertia of the generator, I_2, have to be established for the 11kW model. γ_2 can be estimated using the same technique, shown in Chapter 5, which calculates the damping factor of the 45kW generator from percentage losses of the 330kW machine. The value of I_2 is found experimentally. Calculations and the

experimental measurements for the parameters, shown in Appendix 6a, result in the following :

$$\gamma_2 = 0.002 \text{ and } I_2 = 0.075 \text{kgm}^2 \tag{6.2}$$

In addition to the above modifications, the reference speed of the generator has to be changed to cater for the two-pole machine. The two-pole machine rotates at twice the speed of the 45kW four-pole machine. Compensation for the speed of the generator has to be included in the model and this is realised by including a 0.5 gain in the HSS speed, whs, feedback to the gearbox module.

A modification of the HSS driving torque is also required. The 11kW IM requires less driving torque than the 45kW IM. The reduction in torque is assumed to be linear and the ratio calculated as follows:

$$\text{Torque Reduction Ratio} = \frac{45}{11} \times \frac{2}{1} = 8.182 \tag{6.2}$$

The multiplication of two compensates for the 2:1 speed difference between the two generators (i.e. at rated power, a 3000 rpm machine requires half the torque of a 1500 rpm).

Modification of the model concerns the HSS side only. The LSS remains the same. The aim is that when HILS is included, the software will act as if a 45kW WECS is being simulated while the hardware appears to be simulating an 11kW WECS. Fig. 6.1 shows the modified HSS side of the WECS model which compensates for the 11kW machine.

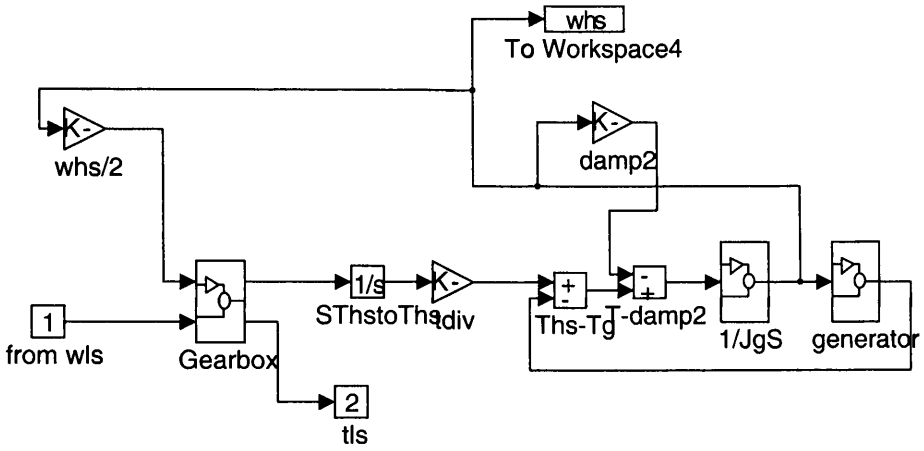


Fig. 6.1 WECS Model Including Compensation for the 11kW IM

The speed correction and torque reduction modules are 'whs/2' and 'tdiv' respectively.

The modified WECS model values are placed in the Simulink model and simulated to assess the change in the power spectral density (PSD) comparison of the low speed shaft (LSS) torques introduced in Chapter 5. Fig. 6.2 shows the LSS torque PSD comparison of the WECS model with the 11kW generator and the RAL site data.

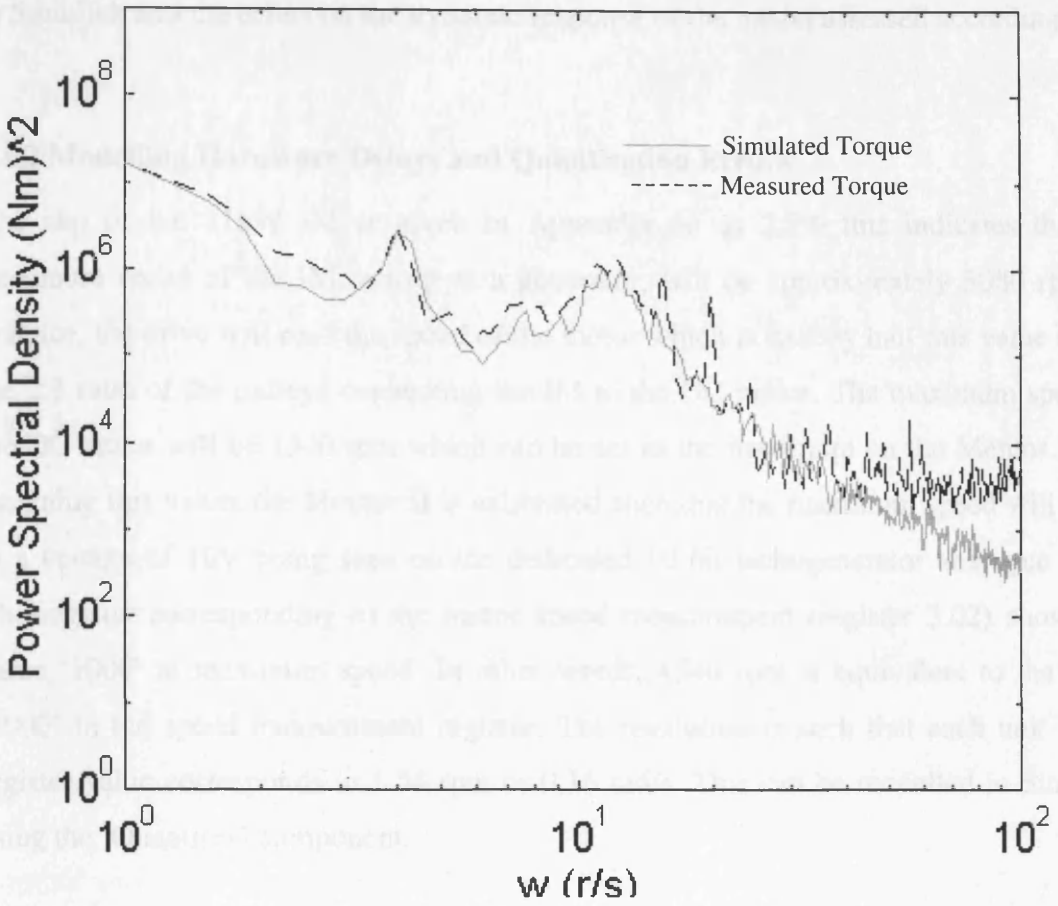


Fig. 6.2 PSD Comparison of the WECS with 11kW Generator

Comparing Fig. 6.2 with Fig. 5.20, which is the PSD comparison between the simulated 45kW WECS and the RAL site data, it is observed that the 11kW model compares favourably with the previous simulation. Both the low frequency response and the amplitudes at the rotational frequencies are very similar to the site data. This suggests that the inclusion of the 11kW generator, and therefore the hardware simulator, will not disrupt the dynamic response of the simulation.

The inclusion of the DC drive, DC motor and the serial communications link will cause delays in the closed loop as mentioned in Chapters 3 and 4. Additionally, real-time measurements via the registers of the drive will inevitably, be subject to noise due to

sampling and quantisation errors. The effects of the delays and the errors can be modelled in Simulink and the effect on the dynamic response of the model assessed accordingly.

6.1.2 Modelling Hardware Delays and Quantisation Errors

The slip of the 11kW IM is given in Appendix 6a as 2.5% this indicates that the maximum speed of the IM, acting as a generator, will be approximately 3080 rpm. In practice, the drive will read the speed of the motor which is exactly half this value due to the 2:1 ratio of the pulleys connecting the IM to the DC motor. The maximum speed of the DC motor will be 1540 rpm which can be set as the maximum on the Mentor II. On assigning this value, the Mentor II is calibrated such that the maximum speed will result in a voltage of 10V being seen on the dedicated 10 bit tachogenerator analogue input. The register corresponding to the motor speed measurement (register 3.02) shows the value '1000' at maximum speed. In other words, 1540 rpm is equivalent to the value '1000' in the speed measurement register. The resolution is such that each unit of the register value corresponds to 1.54 rpm or 0.16 rad/s. This can be modelled in Simulink using the 'Quantiser' component.

In addition to quantisation error, measurements of the tachogenerator (or encoder if selected) and the on-board ADC will be subject to further errors due to noise. A typical measurement of DC motor speed from the Mentor over a 20 s period, is captured by reading the dedicated speed register of the Mentor II from within Simulink. This is shown in Fig 6.3.

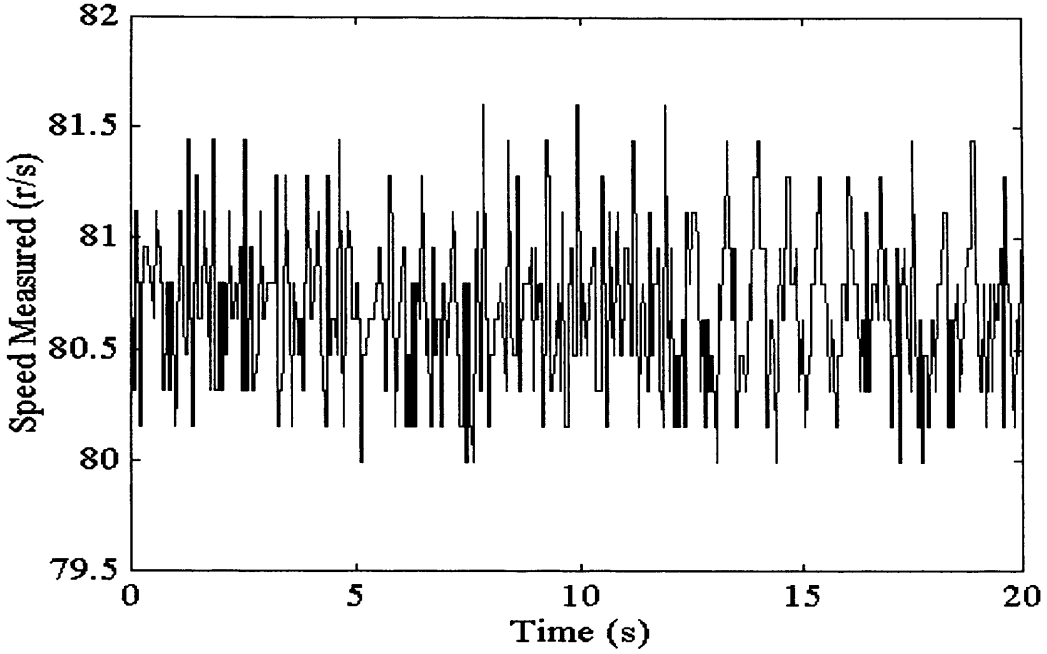


Fig. 6.3. Typical DC Motor Speed Measurement from the Mentor Drive

The speed demand from the drive is kept constant at '500' (80.63 rad/s) while the data is measured. The speed shown has been converted to rad/s from the Mentor II measurement. The presence of the noise can clearly be seen, although it is relatively small compared with the actual value (approx. ± 1%). This error is larger than the error expected from the effects of quantisation alone, therefore some other form of noise must be present in the Mentor register or ADC.

Although the relative value of the error is small, it will be 'fedback' to the gearbox module in the Simulink model and compared with the LSS speed, resulting in a speed error value. This error will be amplified by the relatively large values of shaft stiffness, therefore a small error can result in a large variation in LSS and HSS torques. This can lead to instability in the model.

The noise can be modelled using a band limited white noise module with an appropriate gain.

The delay in the closed loop due to the serial communications link can be modelled using a transport delay component programmed with the estimated delay time. Fig. 6.4 shows the module designed for the WECS model to include the errors inherent with HILS.

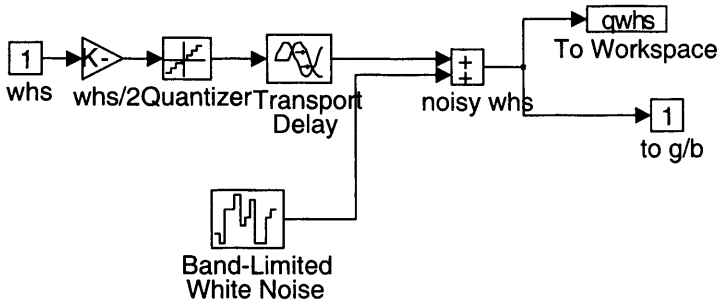


Fig 6.4 WECS Model Modification To Simulate The Actions of the HILS

The 'Quantizer' contains the value 0.16, the 'Transport Delay' simulates the delay of the drive (~0.03s). The 'BLW Noise' module has a gain corresponding to the speed measurement from the drive which, in this case, is 0.005 (see Appendix 6b).

Simulating the modified WECS model shows that after approximately 50s, instability occurs. Fig. 6.5 shows the LSS torque up to the point of instability as well as the effective wind speed output of the spatial filter. The relatively large change in the effective wind speed, just before the point of instability, suggests that the inclusion of the quantisation, noise and delay due to the proposed HILS moves the stable WECS model into an unstable state.

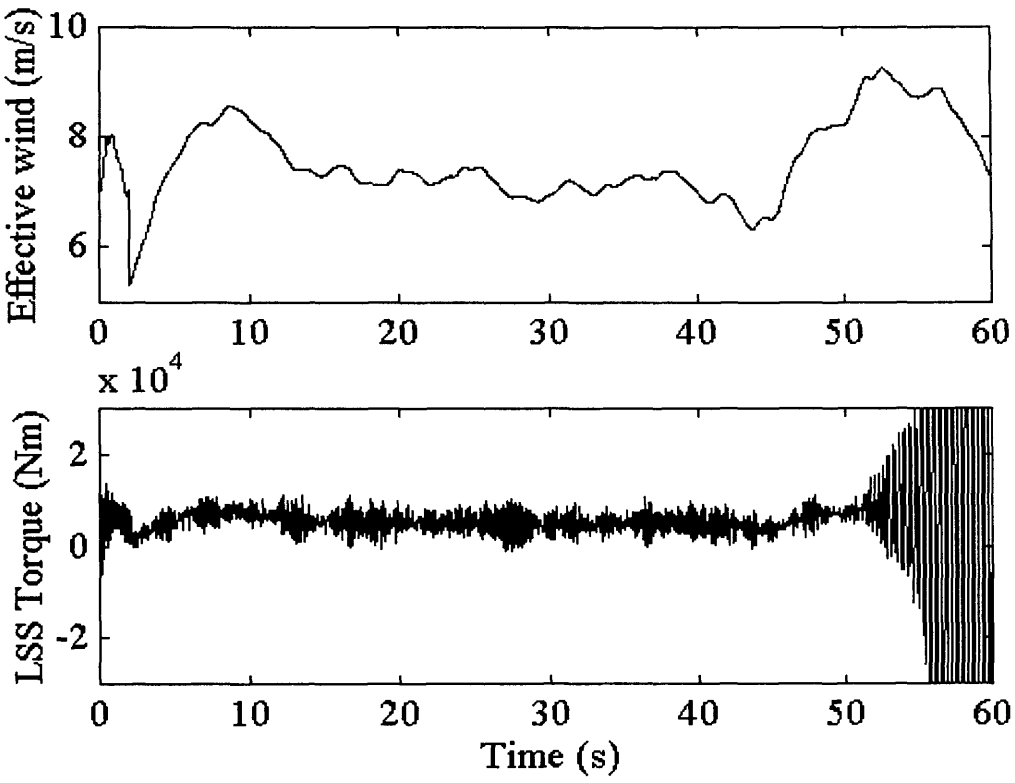


Fig. 6.5 Simulation Showing Instability due to HILS

It follows that values used to model HILS in the simulation will have to be modified to avoid instability. The delay and the quantisation can not be improved due to the limitations of the serial communications and drive, and the maximum speed setting of the drive. Further simulations show that if the noise power is reduced to below 0.0007, the simulation remains stable and produces a PSD comparison shown in Fig 6.6 (see Appendix 6b).

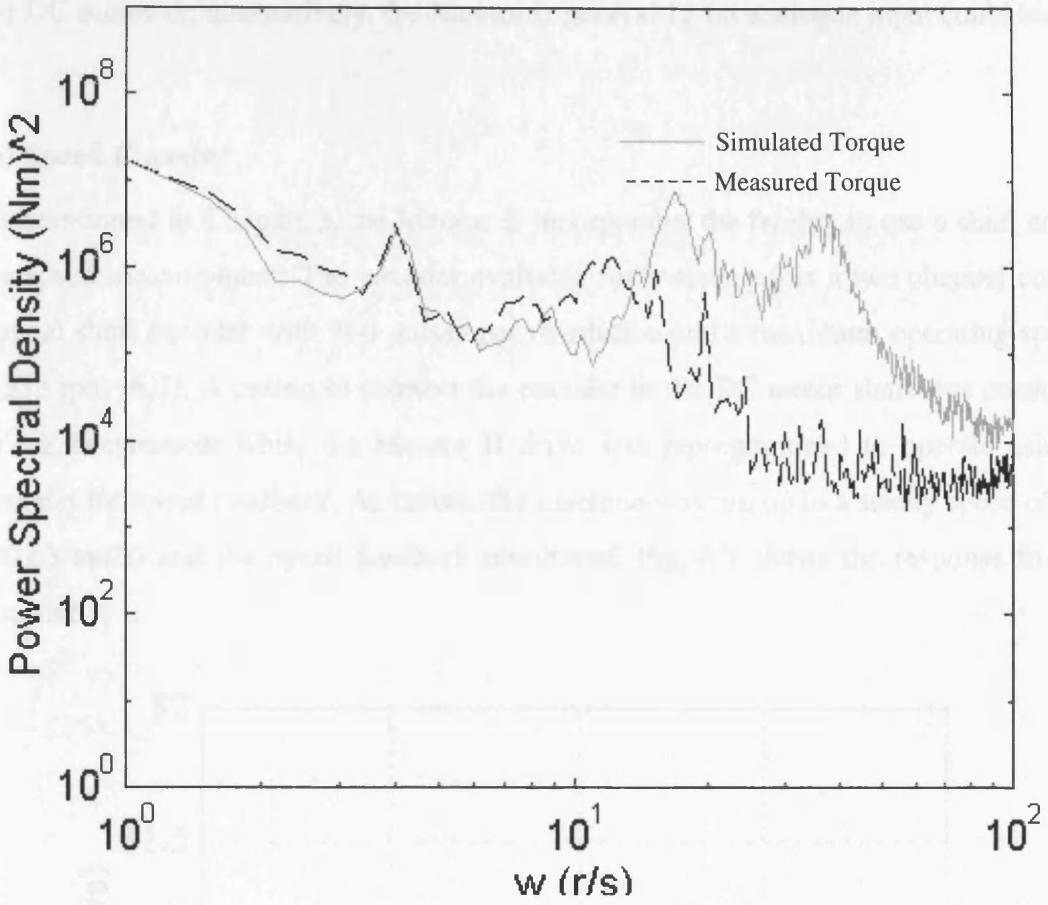


Fig. 6.6 PSD Comparison With HILS and a Noise Power of 0.0007

The lower noise power value ensures that the simulation is stable. Observation of the PSD comparison indicates that the lower frequency response of the simulator is valid, but there is too much energy at higher frequencies due to the delay and inherent sampling errors of the drive. Additionally, the effects of rotational sampling at the blade passing frequency, are swamped by the presence of the sampling noise.

The investigation undertaken in Appendix 6b indicates that reducing the noise on the speed feedback signal reduces the high frequency response of the LSS torque, leading to a more favourable comparison with the measured site data. An alternative speed measurement is therefore required where the effective noise power of the signal is less

than the 0.0007 limit of stability. One possibility is the use of a speed encoder attached to the DC motor or, alternatively, the Mentor II general 12 bit analogue input could be used.

(a) Speed Encoder

As mentioned in Chapter 3, the Mentor II incorporates the facility to use a shaft encoder for speed measurement. The encoder available for assessment is a two channel enclosed optical shaft encoder with 360 pulses per revolution and a maximum operating speed of 8,333 rpm [6.2]. A casing to connect the encoder to the DC motor shaft was constructed in the Department while the Mentor II drive was reprogrammed to operate using the encoder for speed feedback. As before, the machine was run up to a steady speed of '500' (80.63 rad/s) and the speed feedback monitored. Fig. 6.7 shows the response from the encoder.

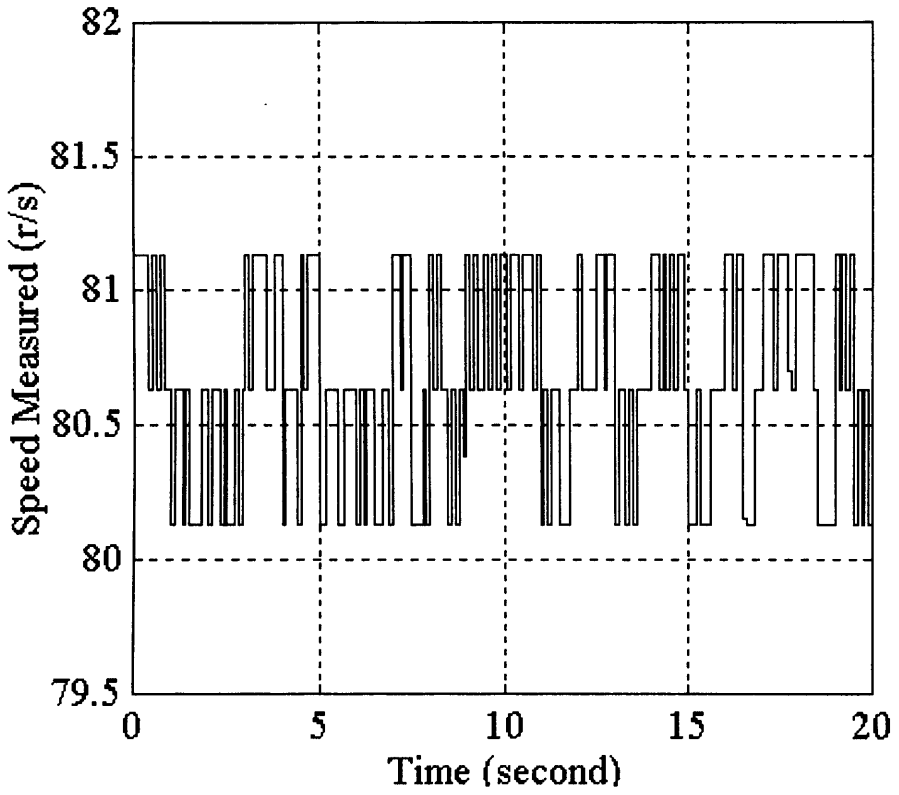


Fig. 6.7 Speed Response from the Optical Shaft Encoder

Comparing Fig. 6.7 with Fig. 6.3, it can be observed that the 'noise' inherent with the measurement is smaller with the encoder option. Further studies (see Appendix 6b) indicate that the 'noise power' of the encoder measurement corresponds to the value 0.004. This value, as previously mentioned, will result in instability.

An encoder with a higher resolution per revolution could have improved the situation, but none was available in the department at the time of the experiment. Therefore, an alternative method was sought.

(b) The Mentor II 12-bit Analogue Input

The Mentor II has a 12-bit analogue input which, in general, is connected to an on-board potentiometer and used as a manual speed demand. Alternatively, it can be used as a general input. One alternative to using the 10 bit speed register connected to the DC tachogenerator is to linearly attenuate the voltage from the tachogenerator and connect it to the 12-bit analogue input. The increase in resolution should reduce the apparent noise appearing on the speed feedback signal.

To perform the attenuation on the signal, a simple potential divider circuit is suggested. From the name plate, the tachogenerator is defined as having a resolution of 0.06V per rpm. This corresponds to a voltage of 92.4V at the defined maximum speed of 1540rpm. The 92.4V should, therefore, correspond to 10V at the analogue input which results in an attenuation of 9.24. Additionally, the input impedance of the analogue input is given as 100kΩ. Fig. 6.8 shows the attenuation circuit used.

One of the Mentor II's general purpose registers can be used to indicate the voltage across the analogue input (in this case, register 07.05). The value in the register is a maximum of '1000' which corresponds to the maximum input of 10V. Therefore, any voltage across the analogue input appears, in register 07.05, as that voltage multiplied by 100.

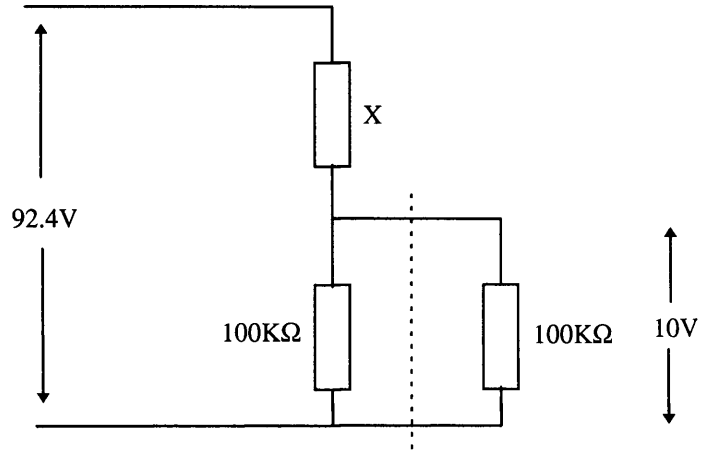


Fig. 6.8 Voltage Attenuation Circuit

To find the value of X:

$$\frac{92.4}{10} = \frac{X+50}{50}$$

$$X = 412\text{k}\Omega \text{ (N.P.V. = 430k}\Omega) \qquad (6.3)$$

Using the nearest preferred value results in a maximum input voltage across the analogue input of 9.625V. The relationship between the measured voltage and the speed of the motor, in rpm or rad/s, can be defined in Simulink as a gain module. At 1500rpm the Mentor II register will show 962 or 963, the conversion ratio would be approx. 1.56. This value would depend on the tolerance of the resistors so the ratio would have to be determined experimentally. Appendix 6a details how the conversion ratio is established practically to be 1.59.

Fig. 6.9 shows the measured steady state speed using the potential divider circuit via the general purpose register.

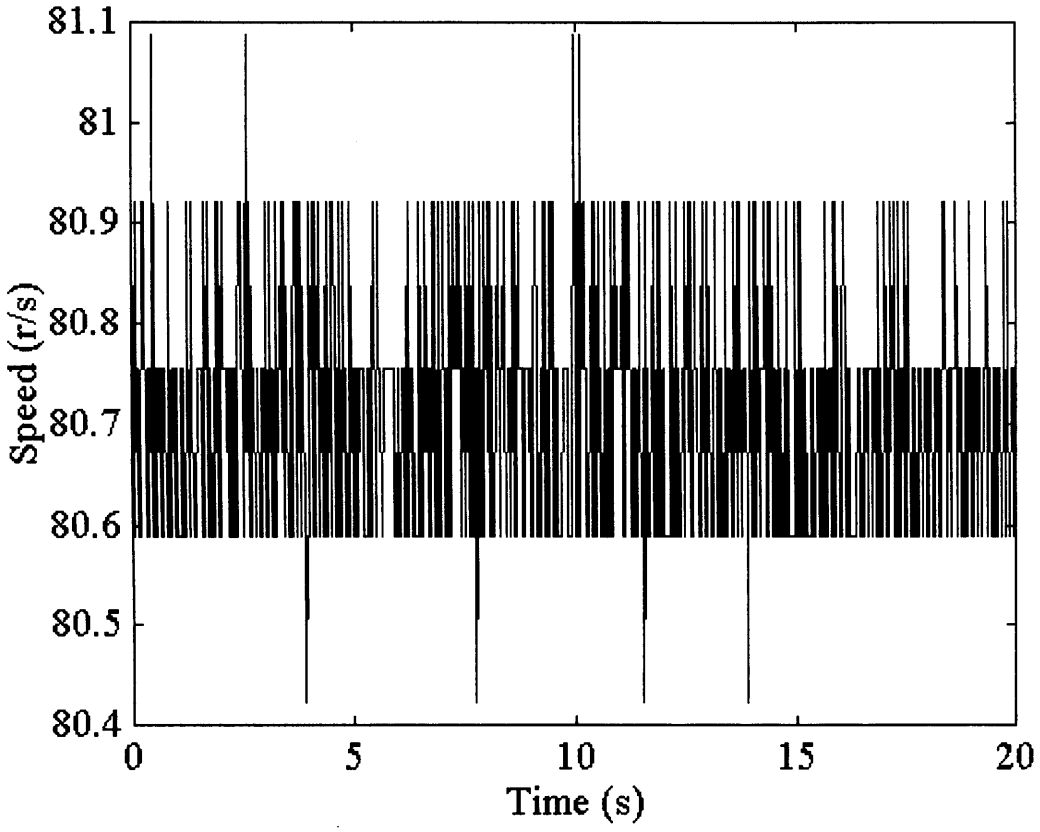


Fig. 6.9 Speed Measurement Using the Potential Divider

Simulations in Simulink using the 'BLW Noise' module indicate that the inherent noise using this method is equivalent to a noise power of '0.0005'. Placing this value into the model simulating the inclusion of HILS gives a stable response. Fig 6.10 shows the PSD comparison with the alternative method of speed measurement. Comparing Fig. 6.10 with Fig. 6.6, the energy at higher frequencies has been reduced but the effect of rotational sampling at the blade passing frequency is still not discernible. Further reduction of the noise gain would improve the higher frequency response and make the effects of rotational sampling more pronounced (see Appendix 6b).

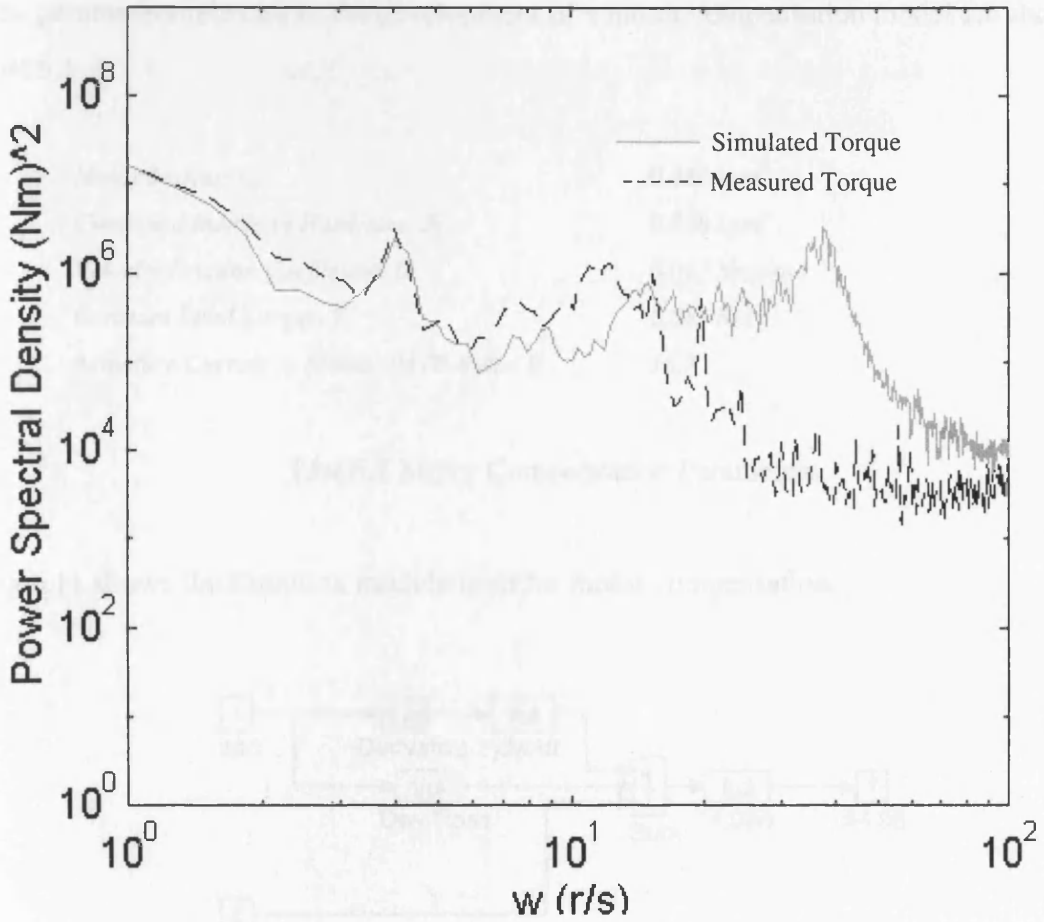


Fig. 6.10 PSD Comparison with HILS and a Noise Power of '0.0005'

Once the speed measurement method is calculated to remain stable, it is necessary to investigate the effect the presence of the motor, as part of the HILS, has on the simulation.

6.2 Compensating for the Presence of the DC Motor in the Simulator

The presence of the DC motor will influence the torque output from the simulator [1.10]. The inertia, friction and torque losses will all contribute to modifying the simulated driving torque on the generator. The dynamics of the motor have to be compensated for in the Simulink model to ensure that the simulated driving torque output from the model is the actual torque 'seen' by the generator shaft. Calculation and measurement of the

inertia and losses of the DC motor, pulleys and the generator are detailed in Appendix 6a. The parameters relevant to the development of a motor compensation model are shown in List 6.1

Motor Inertia, J_m	*0.488 kgm^2*
Combined Inertia of Hardware, J_T	*0.788 kgm^2*
Velocity Friction Coefficient, D	*0.017 Nms/rad*
Constant Load Torque, T_c	*2.085 Nm*
Armature Current to Mentor 04.08 Ratio, K	*16.7*

List 6.1 Motor Compensation Parameters

Fig 6.11 shows the Simulink module used for motor compensation.

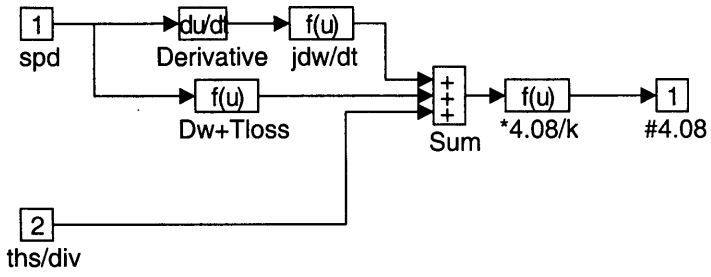


Fig. 6.11 Simulink Motor Compensation Module

The derivative of the speed measurement is multiplied by the calculated motor inertia and added to the losses due to the friction and the constant load torque. These values are added to the output torque calculated for the 11kW generator. The modified torque demand is then converted to an equivalent Mentor II torque demand for register 04.08.

6.2.1 Testing The Motor Compensation Module

The motor compensation module was set up in Simulink to test the calculated values. The inertia is initially set at 0.788 kgm^2 to include the inertia of the generator and pulleys. A

speed profile containing both ramps and a constant value was input to the module and the resultant output current monitored. The same speed profile is then output via an RTW executable to the Mentor II drive and the actual armature current measured. Fig. 6.12 shows the simulated response of the armature as well as the speed profile.

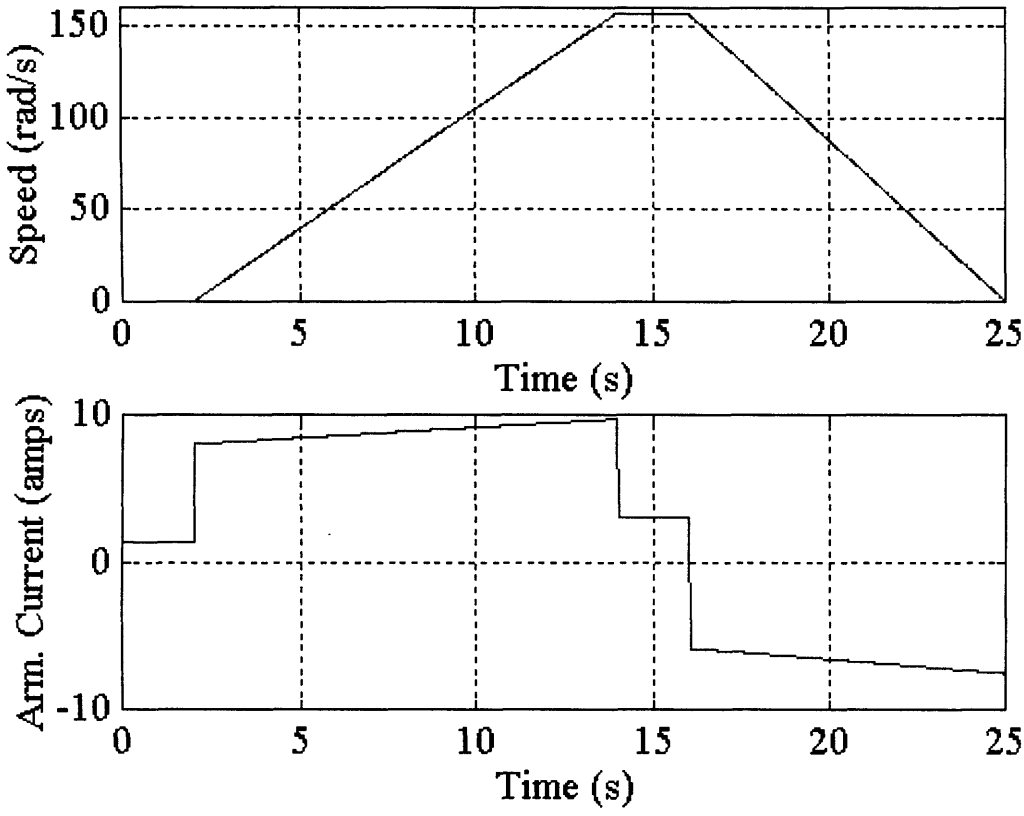


Fig. 6.12 Simulated Response of the DC Motor to a Speed Demand

The armature current measurement clearly shows the effect of acceleration, constant speed and deceleration. The RTW executable contains the same speed demand profile, modified for use with the Mentor II and a module to read the armature current. Fig. 6.13 shows the measured current.

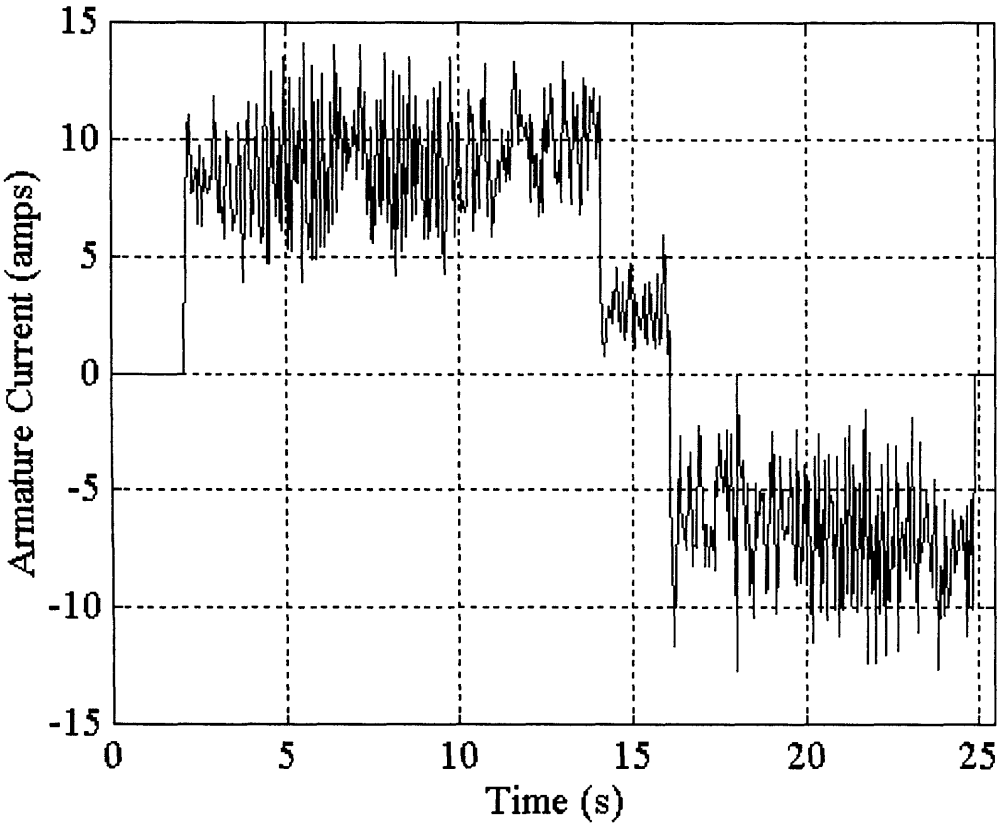


Fig. 6.13 Measured Armature Current for Specified Speed demand

Fig. 6.13 is very noisy making it difficult to analyse. One possible reason for the presence of the noise is the switching nature of the drive output due to the firing action of the drive's thyristors (at approx. 300Hz), introduced in Chapter 3. The relatively low impedance of the DC motor results in large ripples being present on the output voltage of the drive. To combat this possible source of error, a choke (35mH) was placed in the armature circuit in an attempt to 'smooth' the ripples. The test was then repeated. Observation of the new results indicated that there was no noticeable improvement on the current measurement.

The noise on the measurement of the current could be due to errors in the current register of the Mentor II. To investigate this, the register was read without any drive output. The response to this test showed that the measured current was constantly '0'.

Another alternative, is that the controller of the drive is not set correctly. The drive has an 'autotune' facility which sets the optimum gains for the connected motor. There is no information on the nature of this optimisation so therefore, it is difficult to assess the process followed. This leads to the possibility that these values are erroneous, so they were adjusted in an attempt to improve the current response. Unfortunately, various adjustment failed to make any improvements.

To establish an estimation of the armature current, without the noise of Fig. 6.13, an ammeter was placed in the armature circuit. The result of this indicate that, visually, the measurements compare favourably to the simulated armature current shown in Fig. 6.12. The time duration of the speed profile is long enough to allow this and was chosen specifically for this purpose. During acceleration, the measured value start at 8 amps rising to approximately, 9.5 amps. Constant speed demand resulted in 3 amps while the deceleration starts at -5 amps, ending at -7 amps. Since the comparison is favourable, it suggests that the calculated and measured motor values are acceptable for the compensation module.

It should be noted that the simulation of the WECS model with HILS includes the inertia of the 11kW generator and small pulley which is 0.075kgm^2 (0.3kgm^2 at the motor side). The motor compensation module must therefore only include the inertia of the motor and the large pulley which was calculated to be 0.488 kgm^2.

Since both the motor compensation and an alternative speed measuring technique have been designed, the simulator hardware can now be controlled using the simulation software in real-time and its performance assessed.

6.4 Summary

Before the simulator hardware could be used for HILS, the effects of using a smaller size generator had to be assessed in software. The objective is to ensure that the simulation in software, when operating with HILS, will appear to be simulating a WECS model with a

45kW generator while an 11kW generator in hardware is being used to return power to the mains. A model of the 11kW generator had to be developed and suitable scalars placed on the torque demand to the generator and the speed from the generator. This ensured that the driving torque to the generator was within its operational capability and the speed fedback to the gearbox was of the same order as that expected from the 45kW machine.

Once the use of a different size generator was validated, it was necessary to assess the effects of including HILS. The inherent delay due to the serial communications link and drive, the speed measurement quantisation and the noise present on the speed signal were modelled in software. Initial investigations discovered that the speed measurement from the Mentor II dedicated speed measurement register using both tachogenerator feedback and an optical speed encoder, would cause the simulation to become unstable. Further investigations indicated that using one of the Mentor II general purpose input registers, connected to the voltage (after attenuation) from the tachogenerator, would allow stable operation. Subsequent simulations indicate that the HILS results will compare with the site data in the low to mid-frequency range, at higher frequencies, however, the results will diverge due to the inherent HILS characteristics.

The relatively slow response of the motor had to be compensated for in software. The dynamics of the motor, generator and pulleys were calculated from test results and a suitable compensation model designed and validated in Simulink.

Chapter 7 Real-Time Hardware in the Loop Simulation

The next stage of the development of a real-time WECS simulator is to ensure that the simulator hardware can be controlled by the Simulink model. In other words, the WECS model needs to be modified to act as the 'front end' of the WECS and provide the appropriate control signals to the DC drive. The simulator would then need to be compared with the site data to verify the design.

Chapters 5 detailed the design of the model based in a software-only environment. The effects of including hardware in the simulation and compensation for various aspects of the hardware were modelled in Chapter 6. To realise the desire for hardware in the loop simulation (HILS), modifications to the WECS model are required so that the HSS side of the model is removed and the relevant compensation is included. Additionally, the device driver blocks, developed in Chapter 4, will be introduced to provide an interface between the software and hardware.

The complete real-time HILS WECS model is shown in Fig. 7.1. The operation of the closed -loop HILS simulator can be summarised as follows:

- The software simulates the aerodynamics of the WECS as well as the LSS and the gearbox dynamics, producing a torque demand for the grid connected IM generator.

- This torque is modified to compensate for the 11kW generator, the presence of the DC motor and convert the demand into the format used by the Mentor II.

- The driving torque on the DC motor results in the grid connected IM being driven at a speed higher than synchronous and therefore, generating three-phase power into the mains.

- The simulator monitors the speed of the DC motor which is half that of the IM due to the 2:1 pulley ratio.

- This speed measurement is reduced according to the gearbox ratio and compared with the simulated LSS speed. Any error is multiplied by the respective LSS and HSS stiffness to derive the derivative of torque for the respective shafts. This results in closed loop operation.

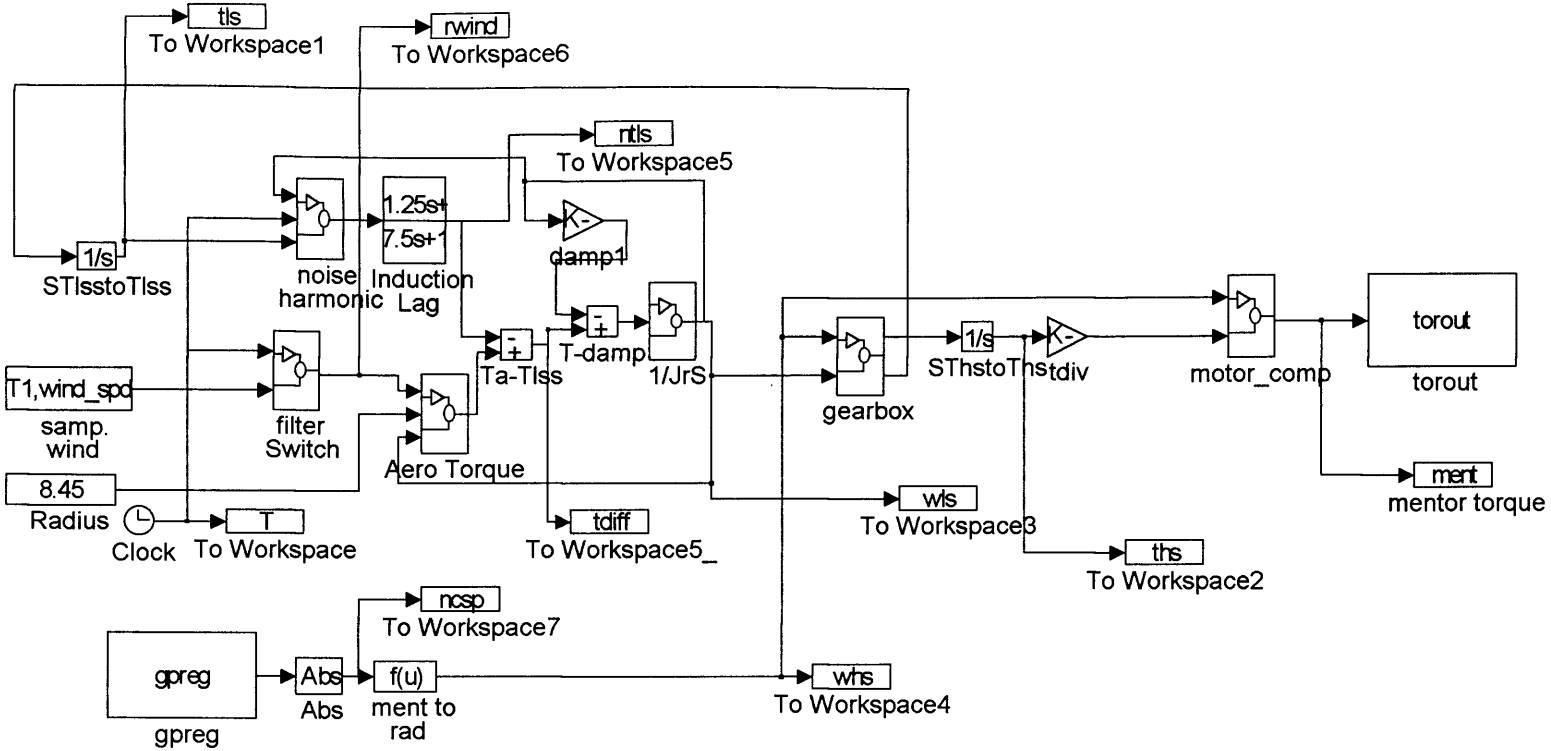


Fig. 7.1 HILS WECS Model

7.1 WECS Simulation Using HILS

The Mentor II drive is set up for closed loop torque control while the IM generator is connected to the laboratory three-phase 415V power supply. A three phase variac is used between the IM and the mains to limit the initial start-up current of the machine.

The real-time simulation is desired to run for 300s (5 min) so that the data established during experimentation is comparable with the RAL measured data used for the PSD comparison. Unfortunately, the real-time data-log buffer in Matlab is programmed to the default size of 2048 data values. This buffer size would correspond to 40.96s of data using a sample rate of 20 ms which is not appropriate. The buffer size can be reprogrammed by editing the Matlab C code file 'rt_log.c' and setting the 'Kilodouble' to half the size required in the definition section of the file (the program doubles the set value during processing). The value has to be an exponential of two to ensure optimum processing time [5.3]. The nearest exponential value to ensure 300s of data at 20 ms (15000 points) can be logged is 2^{14} (16384).

Chapter 4 described the serial communication procedure between the drive and Simulink for both reading from and writing to the Mentor II registers. The total time taken to complete a write (torque demand) instruction and a read (speed measurement) instruction is 21.66 ms. The maximum sampling period will obviously, be tied to this value. Initially, the sampling period will be set to 25 ms to allow for any unforeseen overlap.

Fig. 7.2 shows the LSS torque of the software-only simulation of the RAL WECS and the LSS torque of the HILS RTW executable. It was mentioned in Chapter 5 that a time domain comparison between the simulated and measured LSS torque was not possible due to the inherent time delay in measured wind speed and the instantaneous torque measurement of the RAL site data. A time domain comparison of the HILS and the software-only simulation is possible in this example since both simulations are not subject to the delay.

The results are the same regardless of whether the Mentor II is programmed to operate with or without regenerative braking. This is because negative torque is not required under the operating conditions shown in this example. However, wind regimes at lower wind speed will result in a negative shaft torque causing the IM to motor. Regenerative braking may then be required on the motor.

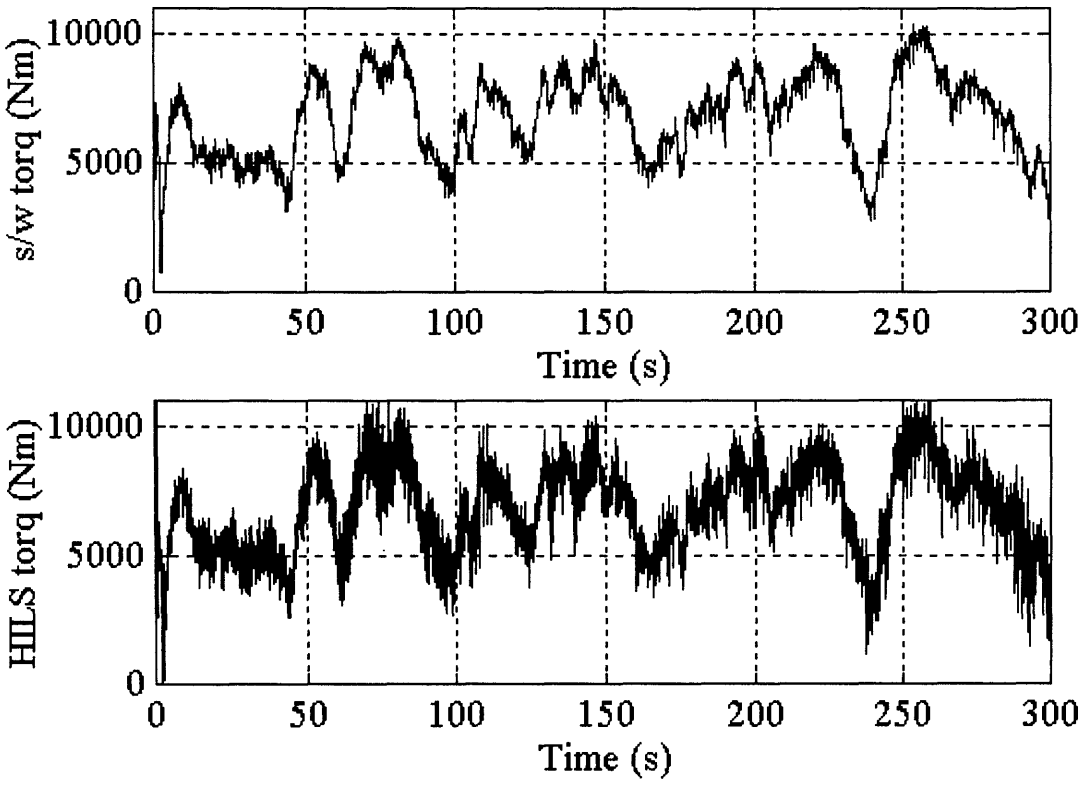


Fig. 7.2 LSS Torque Analysis of the Software Simulation and HILS

The analysis of the HILS torque shows that it is relatively noisy compared with the software-only simulated torque. The software simulation of the inclusion of the hardware in the simulation suggests that this would be the case. The comparison does show, however, that the response of both systems is very similar if the effects of the sampling noise are neglected.

Previously, the PSD comparison between the measured data and simulated data involved both data series sampled and simulated at 50Hz. The HILS executable operates at a sampling rate of 40Hz (25 ms), therefore to ensure that the same parameters are used throughout for comparison, a simple conversion can be performed on the HILS data series. Fig. 7.3 shows the Simulink circuit to achieve this. The sampling period of the model is set at 20ms so the LSS torque data is input to the model as a 25ms data series ('ntls') and converted, through interpolation, to a 20ms data series ('nntls').

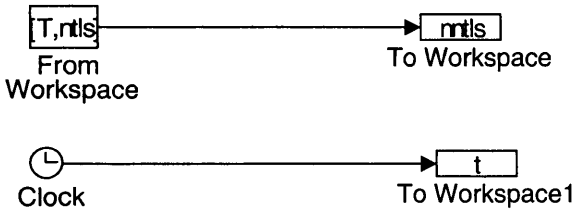


Fig. 7.3 Simulink Model to Convert 25ms Data to 20ms

The converted data series can now be compared with the PSD values of the measured site data. Fig. 7.4 shows the PSD comparison.

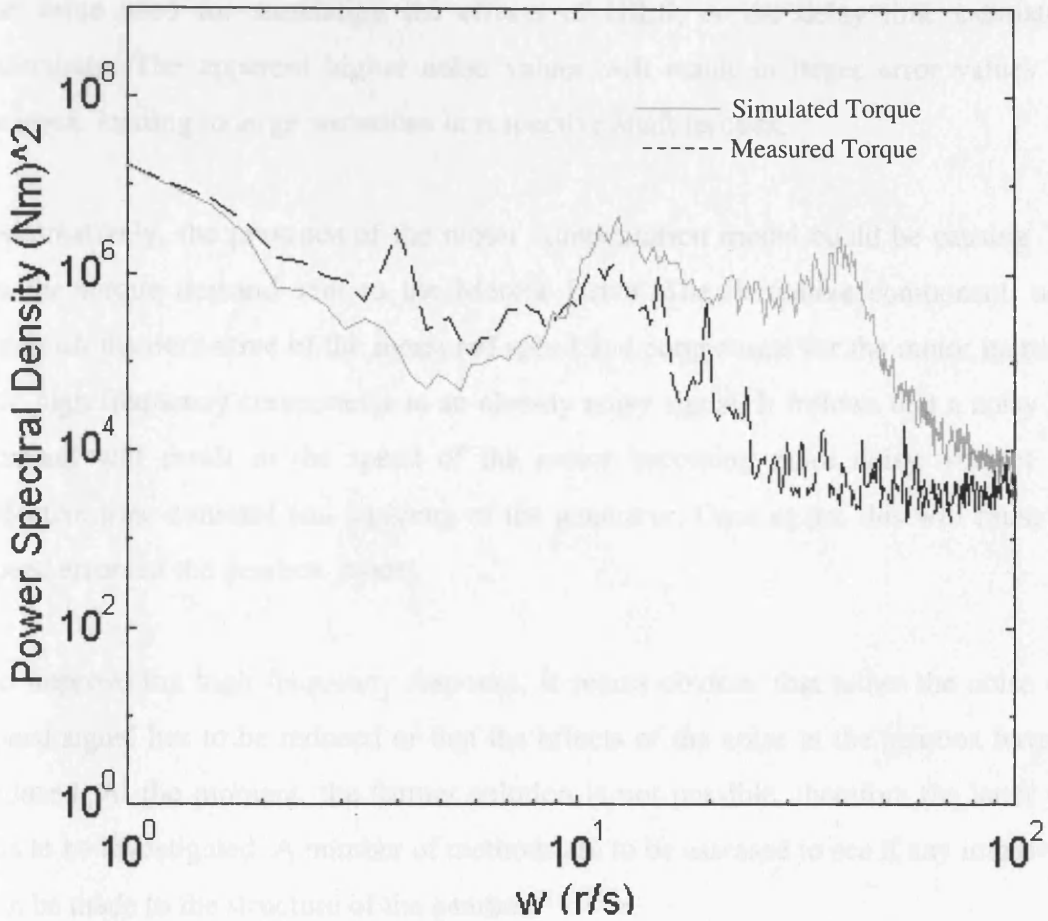


Fig. 7.4 PSD Comparison of Site Data and HILS

It can be seen from Fig. 7.4 that the low frequency response is very encouraging and is comparable to the simulation PSD comparisons of Chapter 5 however, there is too much energy at higher frequencies. This to be expected when considering the high frequency noise present on the time domain analysis of Fig. 7.2. This outcome was also suggested when the effects of HILS were simulated and discussed in Chapter 6 (see Fig. 6.6 and Fig. 6.10). The effects of the rotational sampling are no longer prominent as they are 'swamped' by the noise on the measurement.

Further observation of the higher frequency response, of Fig. 7.4, indicates that the energy levels are larger than that suggested for the 'general purpose input' speed measurement method (Fig. 6.10). This could be due to a number of reasons. Firstly, the

'noise power' estimate of the chosen speed measurement method, could be larger than the value used for simulating the effects of HILS, or the delay time estimates are inaccurate. The apparent higher noise values will result in larger error values in the gearbox, leading to large variations in respective shaft torques.

Alternatively, the presence of the motor compensation model could be causing 'noise' on the torque demand sent to the Mentor Drive. The derivative component, used to establish the derivative of the measured speed and compensate for the motor inertia, will add high frequency components to an already noisy signal. It follows that a noisy torque demand will result in the speed of the motor becoming more noisy, subject to the effective time constant and damping of the generator. Once again, this will cause larger speed errors in the gearbox model.

To improve the high frequency response, it seems obvious that either the noise on the speed signal has to be reduced or that the effects of the noise in the gearbox have to be reduced. At the moment, the former solution is not possible, therefore the latter option has to be investigated. A number of methods are to be assessed to see if any improvement can be made to the structure of the gearbox.

7.1.1 First Alternative Gearbox Design

With the original gearbox, the speed measurement from the drive is converted to rad/s and then the value is further reduced by the gearbox ratio, N, which, in the RAL case is 39.16. This division can further reduce the resolution of the measurement. An alternative to this method is to multiply the LSS speed by N before the error value is calculated. This can be explained by considering the gearbox equation for establishing the speed error, Eqn. (5.5).

$$w_e = \left(w_{ls} - \frac{w_{hs}}{N} \right) \left(\frac{N^2 K_2}{N^2 K_2 + K_1} \right) \tag{5.5}$$

Rearranging this equation to remove the division of the measured speed by N gives:

$$w_e = (Nw_{ls} - w_{hs})\left(\frac{NK_2}{N^2K_2 + K_1}\right) \tag{7.1}$$

This can be implemented in Simulink without affecting any other part of the HILS model. The modified RTW executable, simulated for 300s, results in the LSS torque data series shown in Fig. 7.5.

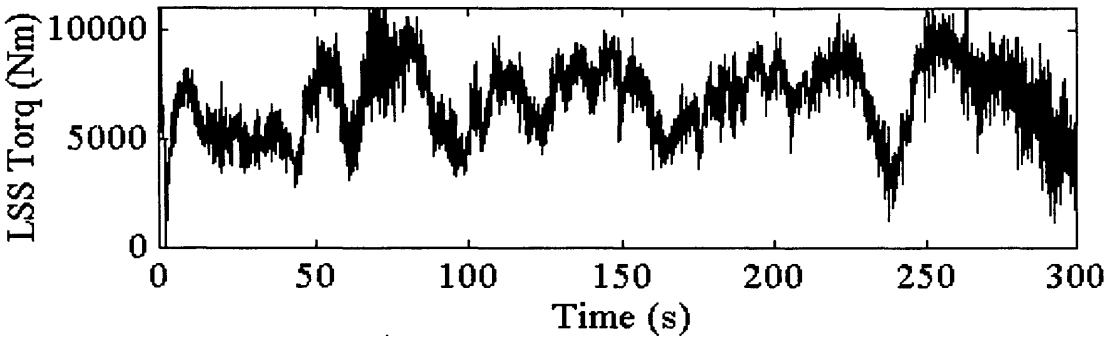


Fig. 7.5 LSS Torque of HILS Using First Alternative Gearbox Arrangement

Comparing the LSS torque of Fig. 7.5 and the original HILS measurement of Fig. 7.2 shows that there is no improvement on the waveform. Indeed, the data appears to be slightly more noisy than the original.

7.1.2 The Second Alternative Gearbox Arrangement

The second arrangement attempts to improve the error difference by keeping the speed measurement data in the form used by the Mentor II register, i.e. it is not converted into rad/s. The LSS speed would need to be converted to this value for the speed difference calculation in the gearbox. This can be further explained by considering Eqn. (7.1) and adding the appropriate conversion. Let the Mentor II speed be M and the conversion from the Mentor II value to rad/s be C where C is:

$$C = \frac{1.59 * 2\pi}{60}$$

$$w_e = \left(Nw_{ls} - \frac{M}{C} \right) \left(\frac{NK_2}{N^2 K_2 + K_1} \right)$$

$$w_e = \frac{1}{C} \left(CNw_{ls} - M \right) \left(\frac{NK_2}{N^2 K_2 + K_1} \right) \tag{7.2}$$

This alternative model is implemented in the HILS and simulated for 300s as before.

In addition to the modified gearbox the motor compensation module can be modified to cater for the Mentor II values. Eqn. (7.3) shows the structure of the existing motor compensation which is used to calculate the modified torque T_{ment}.

$$T_{ment} = \frac{16.7}{1.57} \left[0.017 w_{hs} + 0.488 \frac{dw_{hs}}{dt} + 2.085 + t_{hs11} \right] \tag{7.3}$$

t_{hs11} is the HSS torque converted for the 11kW IM. Adapting the equation to cater for the Mentor II speed value M and the conversion to rad/s, C:

$$T_{ment} = \frac{16.7}{1.57C} \left[0.017 M + 0.488 \frac{dM}{dt} + C2.085 + C t_{hs11} \right] \tag{7.4}$$

Implementing both these modifications in the HILS model and simulating for 300s, results in the LSS torque time series shown in Fig. 7.6.

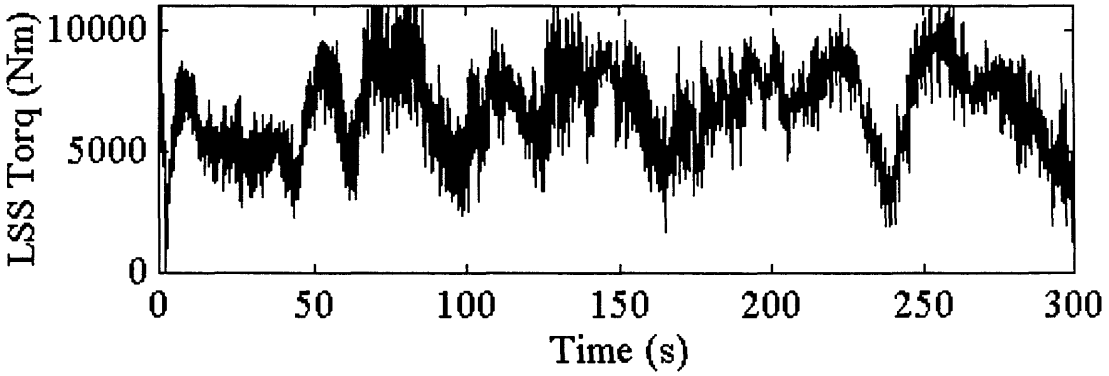


Fig. 7.6 LSS Torque of HILS Model with Second Alternative Gearbox Arrangement

Comparing Fig. 7.6 with Fig. 7.2 it can be observed that, once again, there is no noticeable improvement in response. This gearbox arrangement seems to be the most noisy of the three options.

Another alternative design to the model in an attempt to reduce the noise in the simulation involves the modification of the motor module.

7.1.3 Neglecting the Inertia Compensation in the Motor Model

It was mentioned earlier in this section that the noisy speed measurement from the Mentor II is subject to further high frequency components due to the speed derivative calculation required for motor inertia compensation.

Since the IM is, essentially, kept at constant speed due to the connection to the mains, the variation of speed is very small. At the estimated maximum speed of 1540 rpm, the variation from synchronous speed (slip) is 2.5%. It can then be argued that since the variation in speed, and therefore the derivative of the speed, is very small, the effects of the motor inertia can be neglected. This will allow the removal of the derivative action from the motor compensation module and possibly reduce the noise in the system.

The original HILS model with the initial gearbox arrangement, is modified to remove the derivative action and simulated for the standard 300s. Fig. 7.7 shows the simulated LSS torque during the 300s simulation.

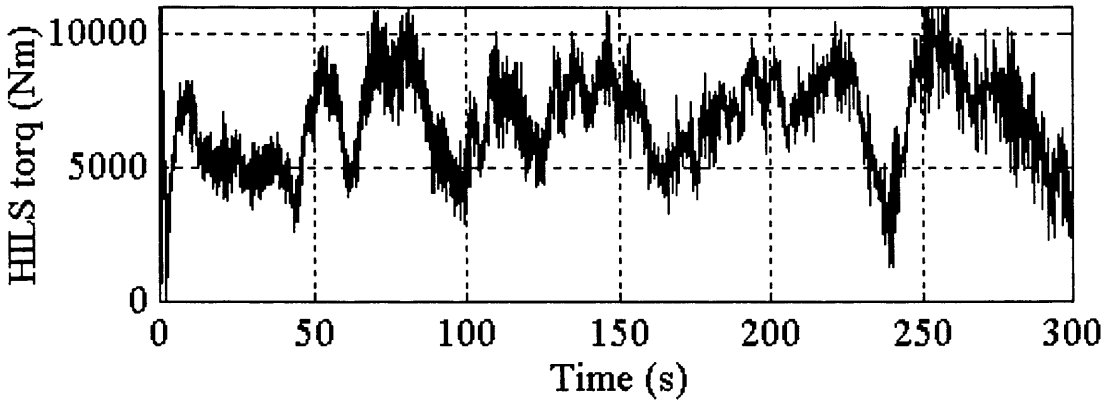


Fig. 7.7 HILS LSS Torque with Modified Motor Compensation

It can be observed, when comparing Fig. 7.7 to Fig. 7.2 that there is a very slight improvement in the torque response. In order to assess if there is any improvement in the PSD comparison, the torque data series is converted into a 20ms series as before. Fig. 7.8 shows the PSD comparison for the modification.

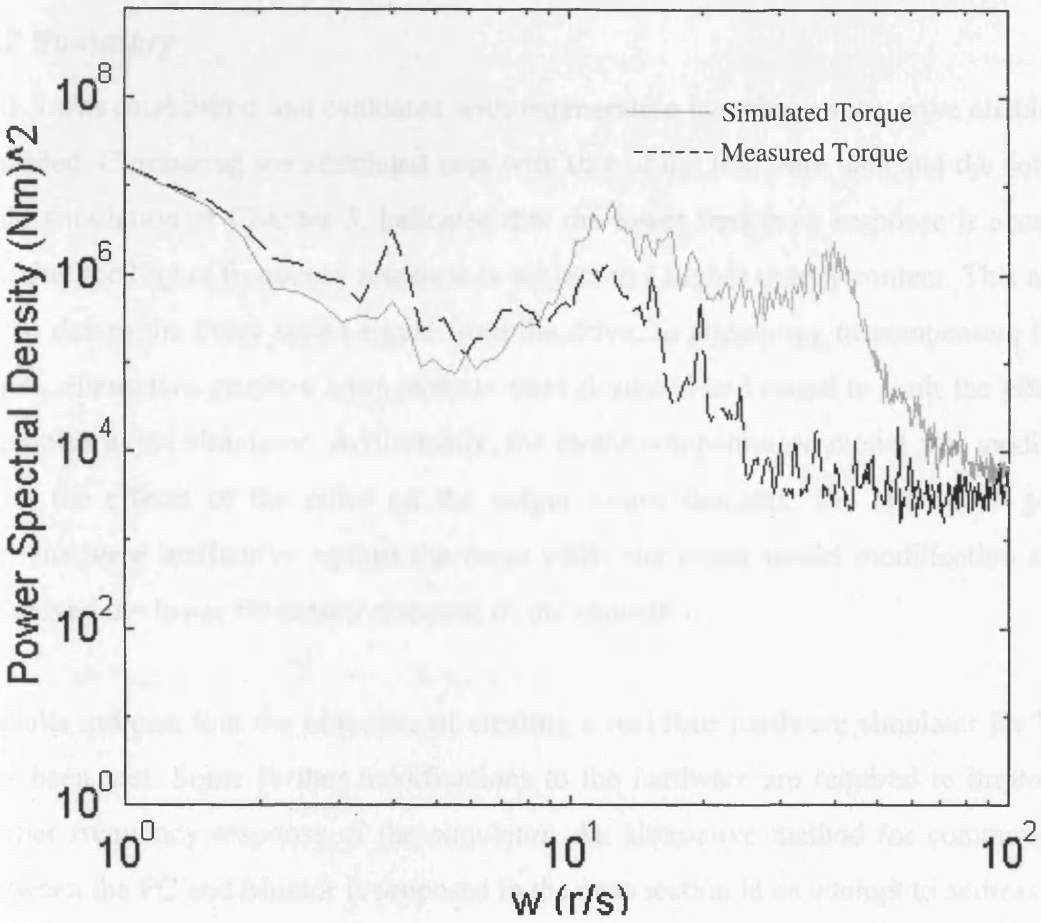


Fig. 7.8 PSD Comparison with Modified Motor Model

Comparing Fig. 7.8 with Fig. 7.4 shows that there is some improvement to the response. It can be observed that the peak at the rotational frequency is more discernible but the energy at higher frequencies has increased slightly. This suggests that the modification does have some benefits at lower frequencies.

In general, it appears, from the investigations of Chapter 6, that the only way to improve the overall response of the simulator is to improve the resolution and accuracy of the speed measurement signal. Introducing a choke to the armature circuit of the DC motor (as before in Chapter 6) did not make any noticeable improvement to the response. Reaffirming the belief that the resolution of the speed measurement needs to be improved.

7.2 Summary

HILS was established and evaluated with regenerative breaking on the drive enabled and disabled. Comparing the simulated data with that of the RAL site data and the software-only simulation of Chapter 5, indicates that the lower frequency response is acceptable but that the higher frequency response is subject to a higher energy content. This appears to be due to the noisy speed signal from the drive. In attempting to compensate for this noise, alternative gearbox arrangements were designed and tested to limit the effects of the noise in the simulator. Additionally, the motor compensation model was modified to limit the effects of the noise on the output torque demand. The alternative gearbox designs were ineffective against the noise while the motor model modification slightly improved the lower frequency response of the simulator.

Results indicate that the objective of creating a real-time hardware simulator for WECS has been met. Some further modifications to the hardware are required to improve the higher frequency response of the simulator. An alternative method for communication between the PC and Mentor is proposed in the next section in an attempt to address this.

Chapter 8. Using a PC Based Data Acquisition Card for HILS

To improve the response of the HILS, an alternative method for communication between the PC and the Mentor II is suggested and investigated. The serial link has proven to be capable for communication for the HILS, but there are concerns about the sampling rate used. The maximum sampling rate is 25ms (40Hz) and although it has been mentioned previously that this should be acceptable for the simulation for a grid connected, 'constant speed' WECS, a communication link with a faster sampling rate could help to reduce the inherent noise in the system. This is especially true if the noise is due to any aliasing effects.

The selected, alternative method is an data acquisition card housed in the PC. This chapter describes the structure of the card and the steps taken to integrate it into the simulator's existing software and hardware set-up. For example, the card will need to be accessed from Simulink in real-time and minor modifications to the Mentor and the speed measurement circuit will apply. Simulations, both HILS and software-only, are performed to assess the response of the simulator and compare the two communication methods.

Initially, the structure of the selected card is discussed in order to assess its programming methodology. This will result in the design of C code to be used as a RTW device driver blocks in Simulink.

8.1 The Amplicon PC30FA Data Acquisition Card

The particular data acquisition card used is the Amplicon PC30FA consisting of 16 single-ended or eight differential 12-bit input channels (analogue to digital converters), as well as four 12-bit digital to analogue converters. The board also consists of digital input and output channels, which will not be used for this project. The maximum sampling rate for the card is 330kHz [8.1].

The input and output ranges of the card can be set to either ±5V or ±10V, depending on the application and both this selection and the choice of single-ended or differential input, are programmed via the card's on-board software. Additionally, the address of the card on the PC's bus, is set using a series of dip switches.

The following sections describe the register arrangement and programming requirements of the PC30FA. This is necessary to establish what is required for the development of programs, using C code, to control ADC and DAC. The C code programs control all aspects of conversion and manipulate the data formats required for the PC30FA registers and the C environment.

8.1.1 PC30FA Register Layout and the Integration of the Card into the Simulator

The PC30FA is controlled via a series of 32 8-bit software registers some of which have a dual read/write purpose. Table 8.1 shows the layout of the register relevant to the project [8.2].

Register Address	Register Name	
	Read	*Write*
700	ADC Low Byte - ADDATL	Block Count - BLKCNT
701	ADC Data/Status - ADDSR	-
702	Control/Channel (ADDCCR)	
703	ADC Mode Register	
70C	-	DAC0 Low Byte - DADATL0
70D	-	DAC0 High Byte - DADATH0
718	Gain Read - GAINREG	Gain Memory 0 - GMEM0
71C	-	ADC Configuration - ADCCFG
71D	-	DAC Configuration - DACCFG

Table 8.1 PC30FA Register Layout

The 'Register Address' column refers to the address of each register and assumes that the card is configured to have a base default of 700H (factory setting). Any change of the on-board DIP switches will alter the register addresses to correspond to the base value.

(a)ADDATL - ADC Data Low Byte (Read Only) (700H)

This register contains the low byte (D_0-D_7) on the completion of an analogue to digital conversion.

(b)BLKCNT - Block Counter (700H)

Indicates the number of conversions to be performed on each ADC strobe. The (Hex) value written to the register is calculated from:

256-(No. of conversions per block)

(c)ADDSR - ADC Data/Status Register (701H)

Lowest four bits contains the high 4-bit data (D_{11}-D_8) of the ADC conversion. The remaining bits contain ADC status information. Bit 6 is the 'ADC Done' status bit set at the end of each conversion and reset on the reading of ADDATL.

(d)ADCCR -ADC Control/Channel Register (702H)

The highest four bits of the register contain the 4-bit channel address which is to be manipulated. Only channel 0 is used for this project. The remaining bits are used for control, bit 1 is used to select software strobe for the conversion (this will be required for use in Simulink) while bit 0 is the strobe itself. A software strobe is initiated by a 'negative going edge' (i.e. bit 0 is set followed by a reset).

(e)ADMDE - ADC Mode Register (703H)

Used for the selection of DMA and error monitoring. Bits 0 and 1 are used to set up, or clear, the channel list on the board. Since only channel 0 will be used, only needs to be cleared at 'start up'.

(f)DADATLO - DAC0 Register (70CH)

Holds four lower bits of 12-bit code loaded into DAC0 for DAC conversions. The 12-bit data is transferred to the output as soon as this register is loaded.

(g)DADATH0 - DAC0 Register High Byte (70DH)
Holds the eight higher bits of the 12-bit code for DAC0 DAC conversion.

(h)GMEMO - Gain Memory 0 Register (718H)
Holds the programmable gain settings for a number of the channels, including channel 0. Bits 0 and 1 must both be reset for a gain of one.

(i)ADCCFG - ADC Configuration Register (71CH)
Holds the interrupt source, input signal range and input signal mode. Bit 1 is set for ±10V range, while bit 0 is set for differential mode. All other bits are reset to disable interrupts and ensure conventional binary coding.

(j)DACCFG -DAC Configuration Register (Write only) (71DH)
This is the mode register in the DAC converter and controls the output gain settings. For ±10V output for channel 0, bits 3 and 7 must be set.

Knowing the requirements and format of the PC30FA, C code will be designed to manipulate card initialisation and control data transfer. Initially, development of the control software will assess the needs of the existing simulator and the communication between the PC and the Mentor. In other words, a C program will be designed to initialise the card knowing that the Mentor ADC and DAC channels operate over the range ±10V and that programs will integrate into the Simulink Real-Time Workshop, as a DDB (device driver block). This will follow the same design format used for the serial communication method, discussed in Chapter 4.

8.1.2 Configuring and Controlling the PC30FA for ADC

Knowing the register structure and address of the PC30FA allows the design of a suitable C program to control these registers and measure a voltage source. Fig. 8.1 shows the pin-out of the card.

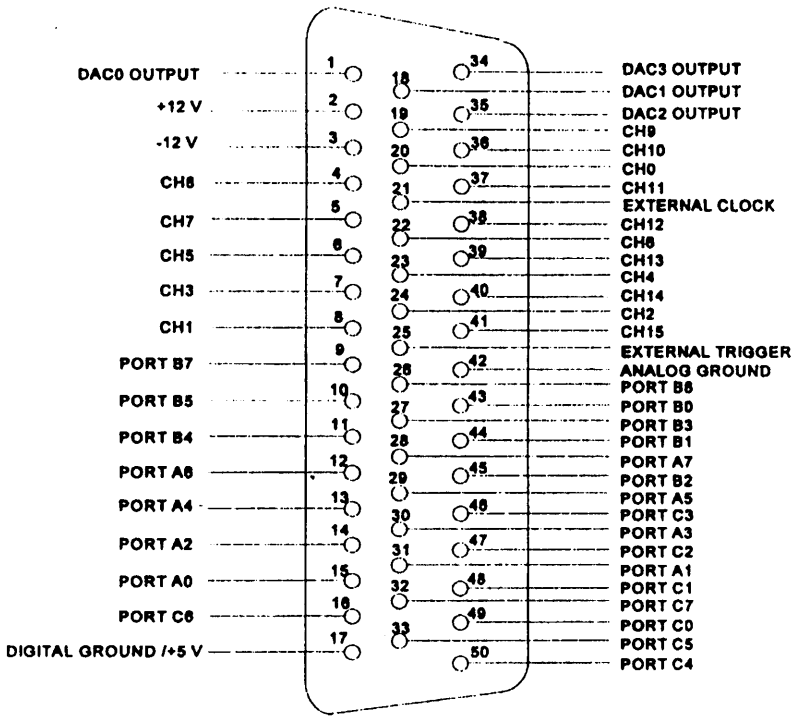


Fig 8.1 Pin-out of the PC30FA Board

The single-ended input is stated to be sensitive to noise when lead lengths exceed 45cm [8.1]. This will be the case with the simulator, therefore the differential input mode is chosen and Fig. 8.2 shows the input arrangement required for differential mode. CH8 is used as the 'RET' input for CH0.

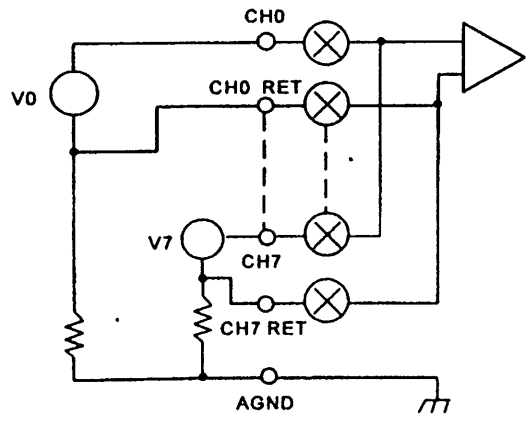


Fig 8.2 Differential Input Connections for CH0 and CH7

V0 and V7 denote a voltage source.

8.1.3 C Program to Establish ADC

For the selected conversion method of one channel, one conversion under software strobe control and differential input of ±10V, the following initialisation is required on the PC30FA card:

- Allow for one conversion per strobe by placing the value of 256-1 (FFH) in to the BLKCNT register.
- Clear the list register by placing 96H in the ADMDE register.
- Select CH0, disable interrupts, enable software strobe control and reset strobe bit by placing 02H in the ADCCR register.
- Ensure that the gain of CH0 is set to unity by placing 00H in the GMEM0 register
- Finally, configure the conversion to operate with ±10V range, in differential mode with no interrupt and no inversion on DAC binary coding. This is achieved by placing 03H into the ADCCFG register.

To realise these needs C code has been designed and developed to perform the above initial operations and is shown in List 8.1

```
/*Initialisation*/
    outp(blkcnt, 0xff);          //1 pulse per conv
    outp(admde, 0x96);           //clr list
    outp(adccr, 0x02);           //ch0, dis int, set s/w strobe, clr str bit
    outp(gmem0, 0x00);           //gain of ch = 1
    outp(adccfg, 0x03);          //no dac inv, no int, diff mode, +-10v
```

List 8.1 Initialisation C Code for ADC Conversion

After the initialisation, the card is ready for ADC conversion. The designed C code will need to include the conversion process which consists of the following procedures:

- Issue a software strobe (negative going pulse) by setting and resetting the strobe bit. This is achieved by sending 03H then 02H to the ADCCR register.
- Poll the 'end of conversion' bit until it is set, indicating that data is ready (monitoring bit 6 of the ADDSR register).
- Once data is ready, read the highest 4-bits of the input data (lowest nibble of ADDSR), masking out the remaining bits in the register. Also read the low byte input data to another location.
- The high data nibble input will need to be added to the low data byte input. A method has to be devised in which the high data nibble can be combined with the low data byte. The selected method will shift the high data nibble 8-bits 'to the left' creating 12-bit data. In other words, the '<<8' C command will be used to, effectively, convert the high data nibble from D_3-D_0 to D_{11}-D8. This method resets all the other bits so that the low data byte can be 'OR-ed' together with the high data nibble to create the 12-bit data.
- Convert the 12-bit data into a voltage. From the PC30FA data manual [8.2]:

$$Voltage = \frac{(12_{bitdata} - 2048) \times 10}{2048}$$

The C code designed and developed for the above procedure is shown in List 8.2. Note that the code will be placed in a loop to ensure continuous conversion.

```
outp(adccr,0x03);                        //start of strobe
outp(adccr,0x02);                        //strobe
while ((inp(addsr)& 0x40) != 0x40);      //poll bit 6 for eoc
d811=(inp(addsr)& 0x0f);                 //mask out control data
d07=inp(addatl);                         //get data 0 to 7
d011= d811<<8;                           //shift left 8bits
d011=d011+d07;                           //combine values
cout<< hex <<d011<<endl;                 //print out
voltage=(d011-2048)*10/2048;
```

List 8.2 C Code used for ADC Conversion

Tests examining the whole ±10V range showed that the correct voltage value was read by the program from the voltage source connected to the ADC input.

8.1.4 C Program for DAC

Having established ADC, the next stage is to ensure the card can be used for DAC. The voltage source is removed and a DVM is connected between DAC0 and the analogue ground (see Fig. 8.1).

The initialisation for DAC is the same as for ADC but with one addition:

- To establish the ±10V range for the output, bits 7 and 3 have to be set in the DACCFG register. All other bits are reset.

The DAC is, basically, the reverse process seen above. Tasks to be performed are:

- The voltage level required has to be converted using the following:

$$\text{code} = \frac{(\text{voltage} \times 2048)}{10} + 2047$$

- The code, automatically converted to hex, is placed in the appropriate register with the correct positioning. The method devised to do this is a series of bit 'shifts' and masking. First, the code is masked to isolate the high data byte, 'shifted to the right' by 4 bits and the result placed in the DADATH0 register. Secondly, the code is again masked, this time to isolate the lower data nibble, 'shifted to the left' and placed in the DADATL0 register. The board places the corresponding value on DAC0 as soon as the low data nibble has been placed.

The C code designed and developed to perform the above conversion is shown in List 8.3. As with the ADC, the code is placed in a loop for continuous operation.

```
fcode =(voltage*2048/10)+2047;          //convert voltage
code=int(fcode);                        //conv float to int
hbyte = (code & 0xff0) >>4;             //mask and shift
lbyte = (code & 0xf) <<4;               //mask and shift
cout<< "hbyte....."<<hex <<hbyte<<endl;  //print out - debug
cout<< "lbyte....."<<hex <<lbyte<<endl;  //print out - debug
outp(dadatho,hbyte);                    //place H data byte
outp(dadatlo,lbyte);                    //place L data nibble
```

List 8.3 C Code for DAC

Monitoring the DVM over the whole ±10V range indicated that the program performed as desired.

The next stage in incorporating the PC30FA into the simulator is to convert the existing C code, for both ADC and DAC, into Simulink Real-Time Workshop DDBs.

8.2 Creating DDBs for the PC30FA

The format of the DDBs is exactly the same as the DDBs created for communication using the serial comms. link, discussed in Chapter 4. Modifications to the initialisation and output sections of the C code s-function are made to include the unique code developed above. The process for compiling the code for use within the Simulink environment, is also as shown previously. The code for ADC (pc30ad.c) and DAC (pc30da.c) is shown in Appendix 8a and differs slightly from the C code shown above. This is due to the requirements of the s-function, which use the 'double float' variables 'y' and 'u', and to adapt the voltages, in Simulink, to the Mentor format, e.g. 10V is 1000.

8.2.1 Controlling the PC30FA from the Simulink Real-Time Workshop

Once the DDBs have been created the data acquisition board can be used in the Simulink environment. To assess the performance of the board, models are created for the input and output of data. Fig. 8.3 shows the basic model for monitoring inputs to the PC.

Fig. 8.3 Simulink Model to Monitor the ADC of the PC30FA

The input is connected to a DC power supply between CH0 and 'analogue ground', as before, and the simulation run for 10s. Fig. 8.4 shows the measurement of the input voltage.

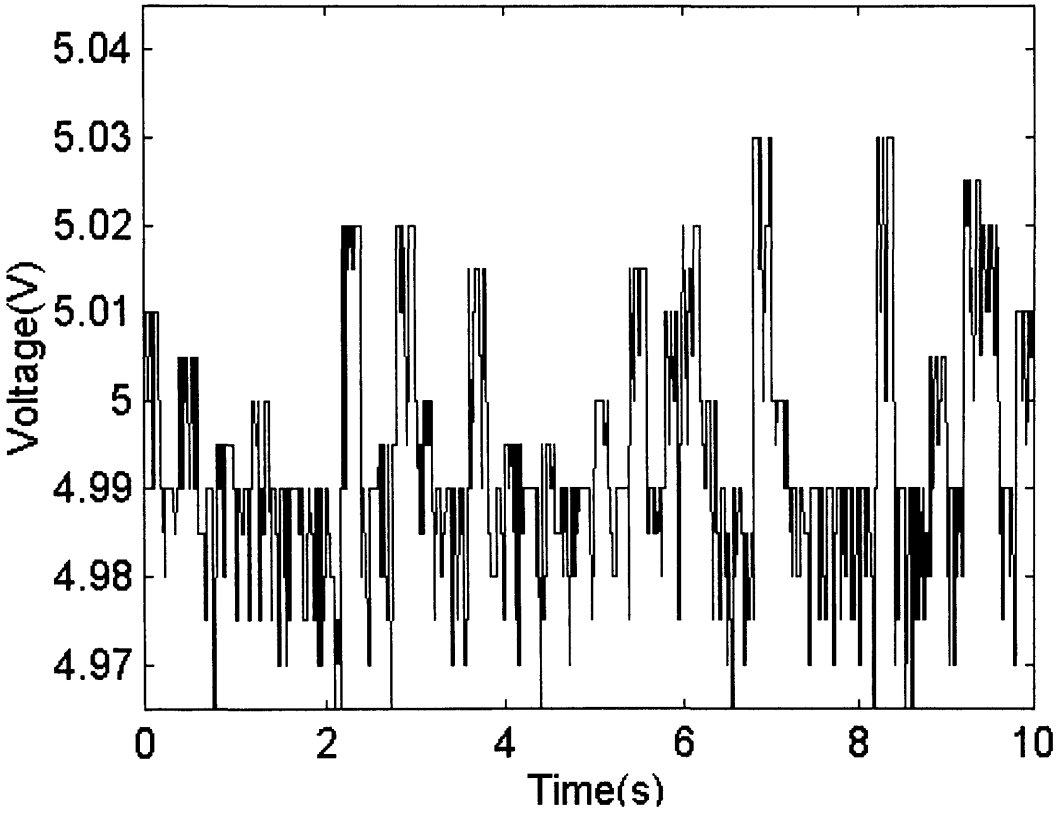


Fig. 8.4 Measurement of DC Voltage Source

From the figure it can be seen that there is a noticeable amount of noise, which has the value ±0.04V (0.8%). It is a reasonable assumption that this is an acceptable noise value for many applications, but this may not be the case for the simulator (see Chapter 6).

In an attempt to improve the noise it was decided to incorporate a number of decoupling capacitors across the differential inputs. Fig. 8.5 shows the proposed arrangement.

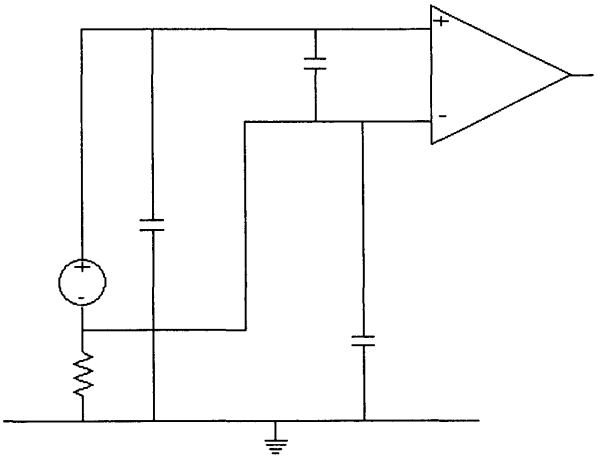


Fig. 8.5 Differential Input Incorporating Decoupling Capacitors

The input voltage, via Simulink, with the modified input circuit is shown in Fig. 8.6.

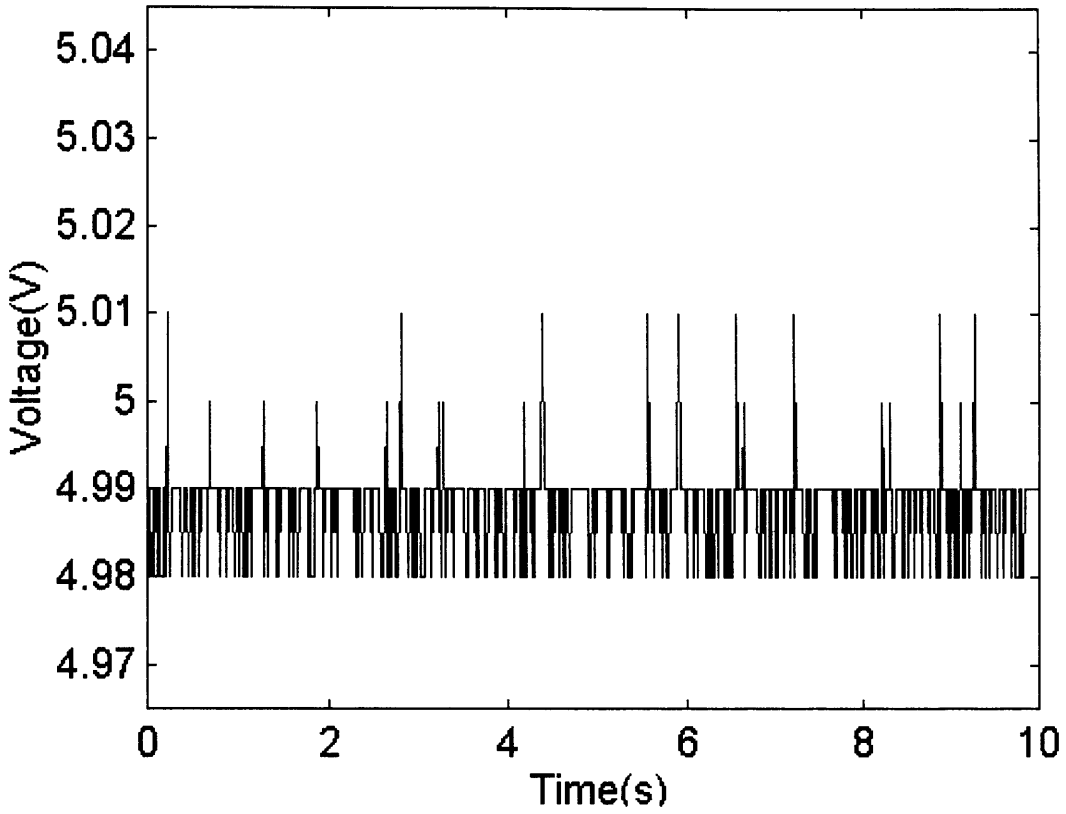


Fig 8.6 Voltage with Modified Input Circuit

Fig 8.6 shows that there has been an improvement in the measurement as the noise reduces to ±0.02V (0.4%).

The next stage of the development is to look at the measurement of the voltage across the tacho. With the serial communication link discussed earlier (Chapter 6), the feedback from the tacho was scaled down, using a potential divider, before being input to one of the Mentor's ADC channel. With the data acquisition card the feedback will, again, need to be scaled down using a potential divider, before being connected directly to the differential input. It is stipulated in the PC30FA manual that the source impedance of the devices connected to the analogue inputs must be ≤ 1kΩ. The impedance of the tacho is measured at 500Ω so the circuit will need to compensate for this. Additionally, the

resistors available are rated at 0.6W so the circuit had to be designed with the power requirements in mind. Fig. 8.7 shows the circuit used.

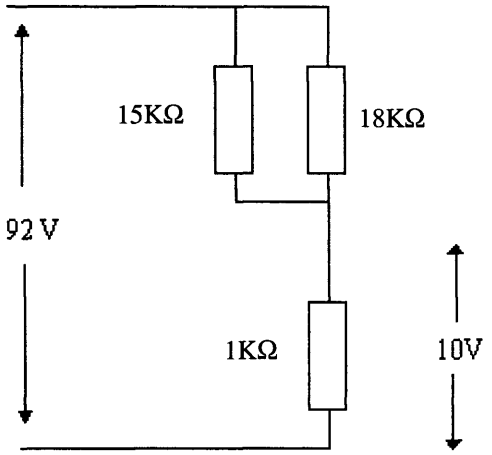


Fig. 8.7 The Tacho Feedback Conditioning Circuit

Measuring the effective resistance across the differential input gives 893Ω, which is within specification. Calculations show that the voltage division and power requirements are also acceptable.

Connecting the output of the above circuit to the ADC while the input is connected to analogue tacho, allows the performance of the tacho to be assessed in Simulink. The Simulink model shown in Fig. 8.3 is used again. Fig. 8.8 shows the measured voltage when the motor is run at synchronous speed.

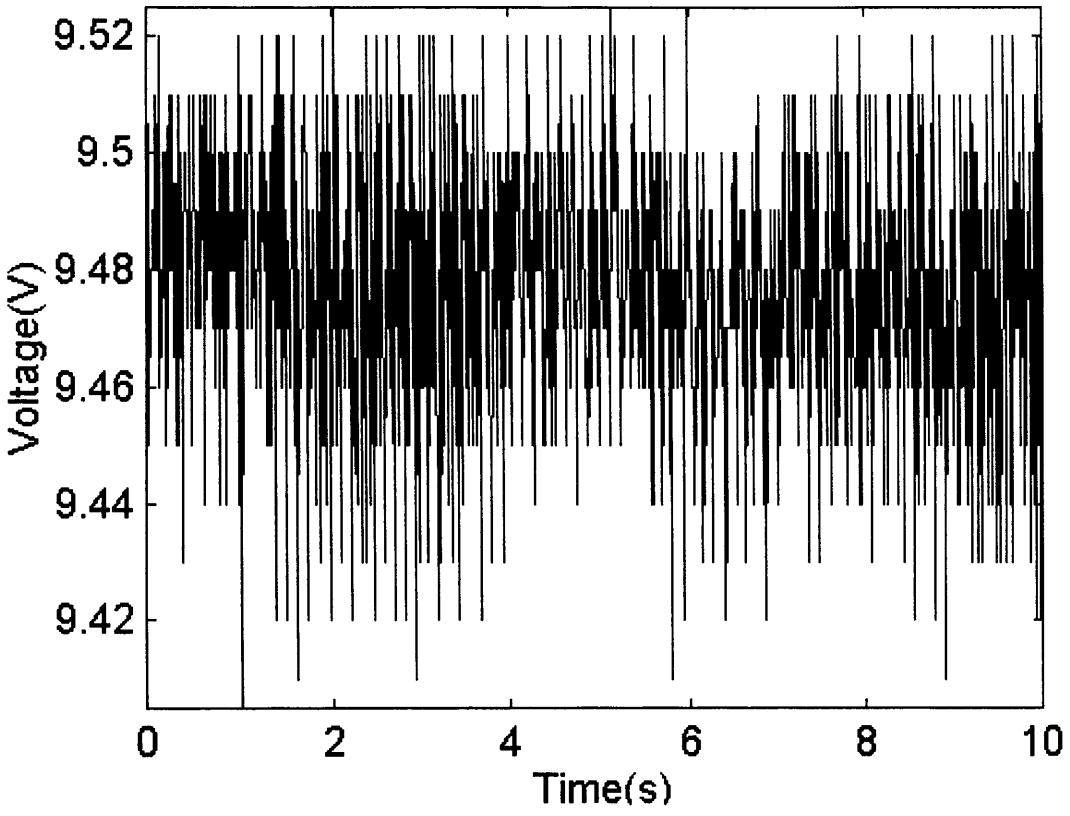


Fig. 8.8 Voltage Across the Tacho at Synchronous Speed

(Note that the measurement is actually a negative voltage, but is shown positive for comparison purposes).

The measurement is noisier than the DC voltage source (0.74% c.f. 0.4%). Direct measurement of the tacho output, using an oscilloscope, shows a similar noise level, suggesting the additional noise must be due to the tacho itself. Chapter 6 and Appendix 6b discussed this and the implications of the noise on the effect on the stability of HILS. Suitability of the use of this particular method of speed feedback may be too noisy for HILS, therefore, is was decided to reassess the use of the digital encoder for speed measurement. Previously (Chapter 6), it was noted that the encoder would lead to instability in HILS using the slower sampling rate and serial communications link. To establish whether this is still the case with the faster sampling rate, the encoder was re-attached to the rig. The Mentor configuration was adjusted to allow for the change from

tacho to encoder. A speed measurement was taken from the encoder, via the Mentor and PC30FA. Fig. 8.9 shows the response.

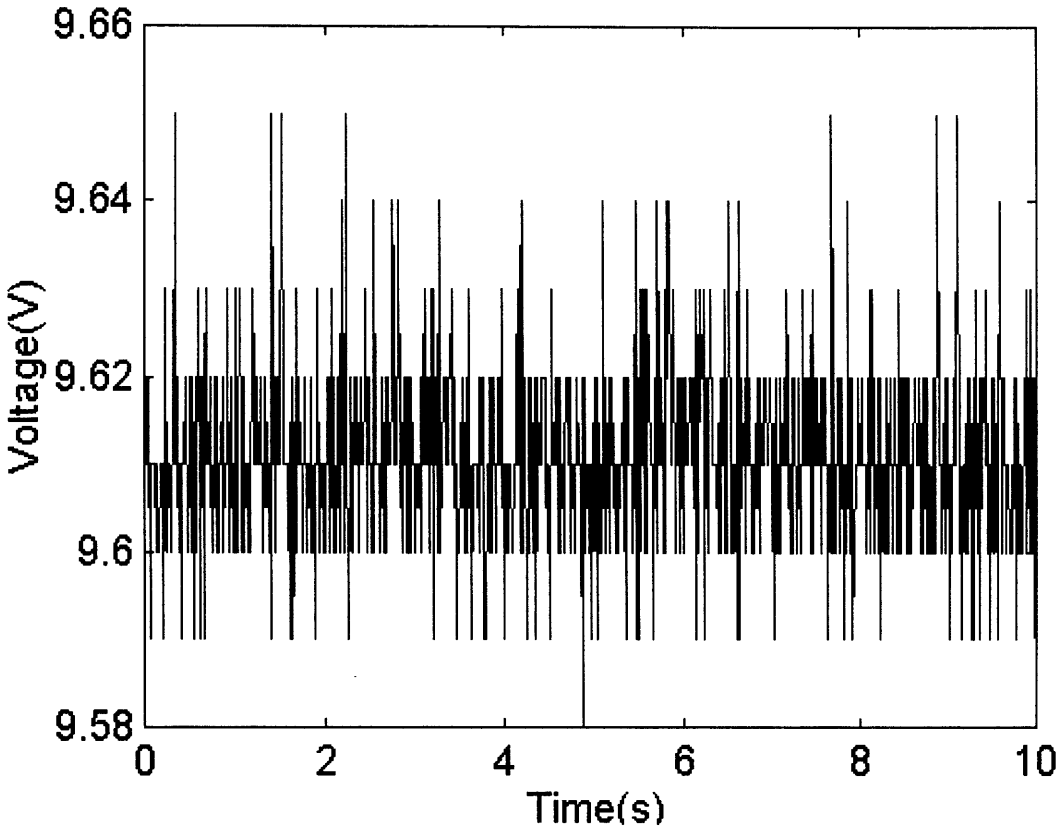


Fig. 8.9 Speed Measurement Using Encoder

Comparison of the measurements of the tacho and the encoder show that the encoder provides a more stable and less noisy (0.4%) reading than the tacho and should be the preferred choice for speed feedback for HILS.

8.2.2 Using the PC30FA with HILS

Before assessing the HILS with the new communication link, it is necessary to observe any change in the software-only simulation with the faster sampling rate of 80Hz (0.0125s). This is chosen since the Mentor current loop is specified to have a bandwidth of 80Hz [3.5]. The same model is used, as in Chapter 5, except that the sampling rate has been increased from 40Hz to 80Hz. Fig. 8.10 shows the power spectral density comparison of the model and the RAL site data.

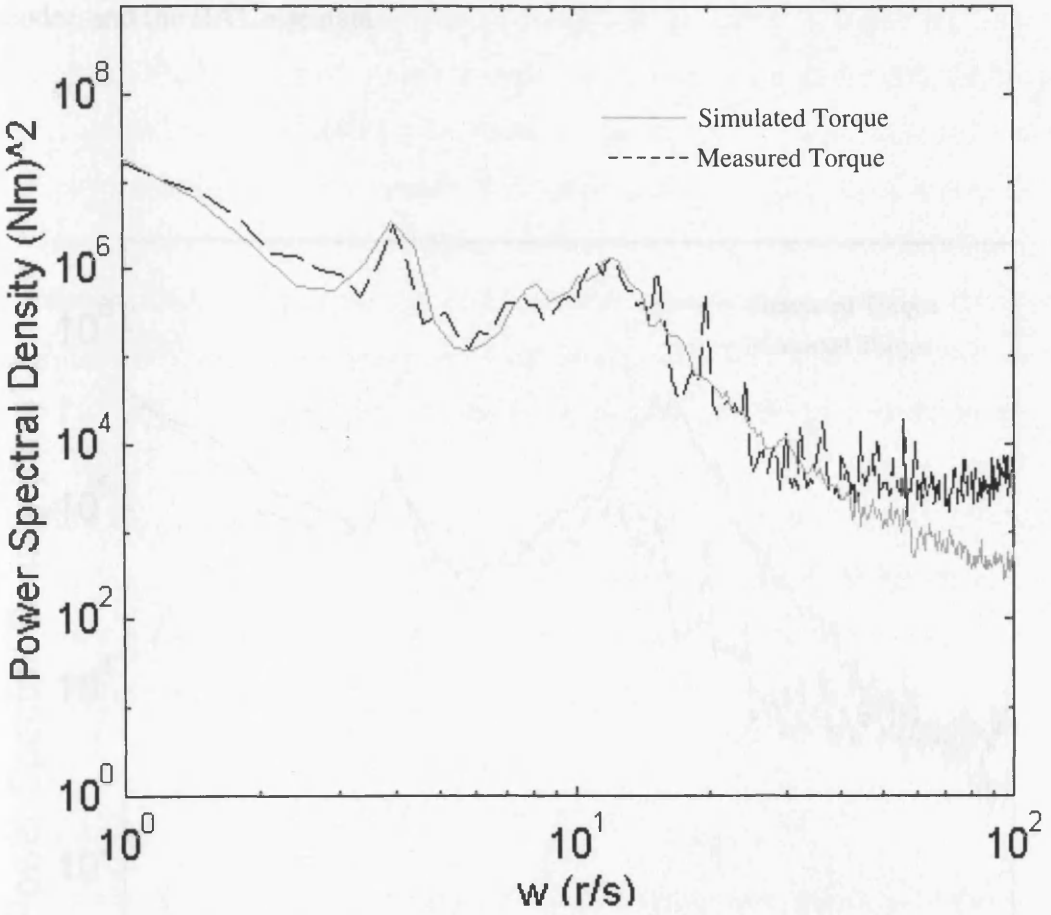


Fig. 8.10 PSD Comparison of the Site Data and 80Hz Simulation

Comparing the above figure and Fig. 5.20 shows that the responses are very similar for the different sampling rates with only a slight variation at the rotational frequency of 4 rad/s The next stage is to investigate the response of the HILS and compare it with the simulation of Chapter 7. The output of the data acquisition card (DAC0) was connected

to one of the Mentors 10-bit input channels and the Mentor was reprogrammed to read this channel and set it as the torque demand of the drive (register 04.08). Similarly, the input of the digital encoder was directed to one of the Mentor's 10-bit output channels (after decoding) and connected to one of the data acquisition card's differential inputs (CH0). This is the motor speed measurement which is 'fedback' into the Simulink model. As before in Chapter 7, a DOS executable was created via the RTW.

Fig. 8.11 shows the PSD comparison of the HILS, using the data acquisition card and encoder, and the RAL site data.

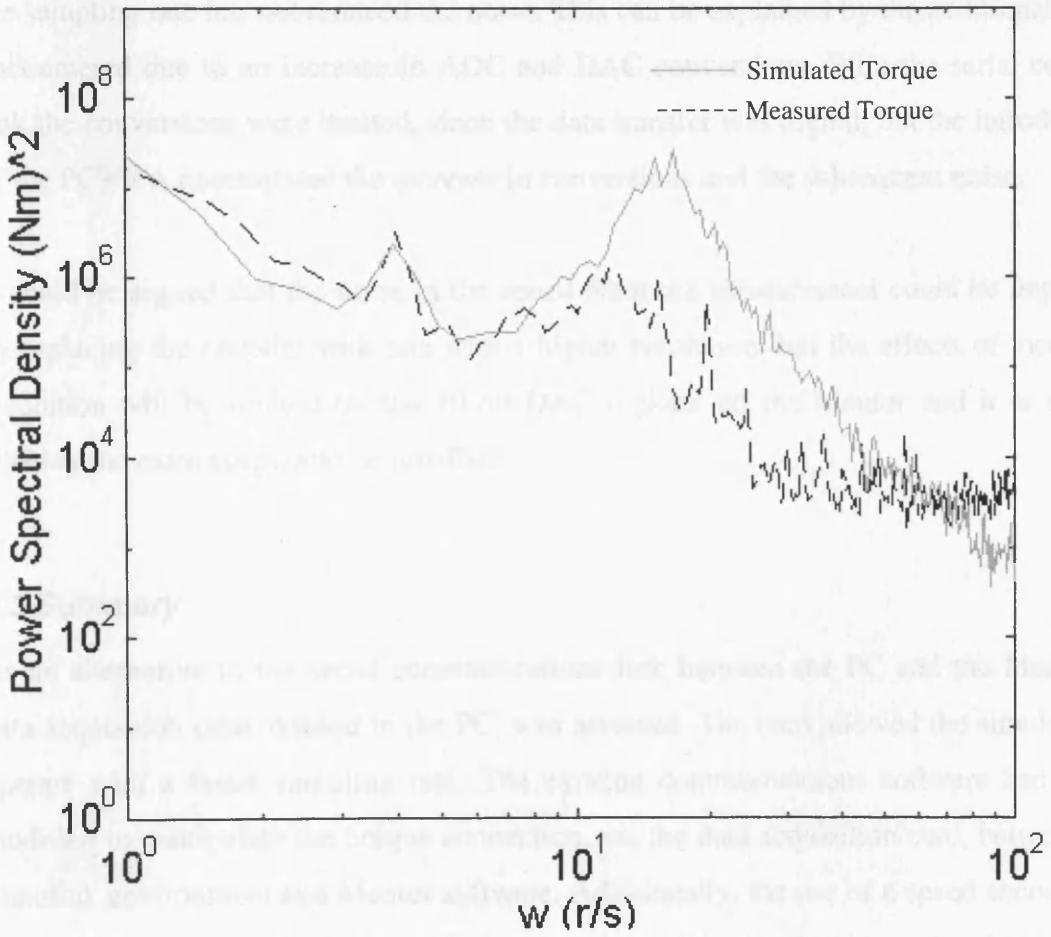


Fig. 8.11 PSD Comparison of HILS and Site Data

It is noticeable that there is excess energy in the mid to high frequency range but the low frequency comparison is very close. The comparison at the rotational frequency is very encouraging . Comparing this response to the HILS response with the serial comms. link (Fig. 7.4) shows that there is definitely an improvement in response at lower to mid-frequency, but there is still too much energy in the higher frequencies. This is noticeable at the rotational sampling period (~12 rad/s). In an attempt to limit the energy, the white noise model used to emulate the rotational sampling is removed, but no improvement in response is noticed. As before, it is assumed that the effect is due to the noise in the HILS circuit. Both the speed measurement from and the torque demand to the Mentor are subject to noise, which appears to resonate through the system. Although the low-frequency response has been improved, improving the sampling method and, therefore, the sampling rate has not reduced the noise. This can be explained by the additional noise encountered due to an increase in ADC and DAC conversions. With the serial comms. link the conversions were limited, since the data transfer was digital, but the introduction of the PC30FA necessitated the increase in conversions and the subsequent noise.

It could be argued that the noise in the speed feedback measurement could be improved by replacing the encoder with one with a higher resolution, but the effects of increased resolution will be limited by the 10-bit DAC register on the Mentor and it is unsure whether the extra cost could be justified.

8.3 Summary

As an alternative to the serial communications link between the PC and the Mentor, a data acquisition card, housed in the PC, was assessed. The card allowed the simulator to operate with a faster sampling rate. The existing communications software had to be modified to manipulate the unique connection, via the data acquisition card, between the Simulink environment and Mentor software. Additionally, the use of a speed encoder for feedback was reassessed since the inclusion of the data acquisition card, subsequently, introduced more noise to the system and made the use of the tacho unfavourable.

Assessment of the modifications to the simulator showed that there was an improvement in its response at low to mid frequencies. Higher energy levels, present at higher frequencies, are, once again, attributed to sampling noise. It is suggested that this particular arrangement has resulted in the noise being enhanced by the additional DAC and ADC conversions, required due to the presence of the data acquisition card. Further improvements to the response could be possible with the introduction of a higher resolution encoder, but limitations of the Mentor DAC channels suggest that the costs of investigating this, can not be justified.

Another positive aspect to the modifications is that, although major in concept, they were relatively straightforward once the specification and operation of the data acquisition card was known. This, once again, indicates that whether the modifications are in software or hardware, the simulator is flexible enough to easily adapt.

Chapter 9 Simulating a Micro-Hydro Plant

Having successfully modelled and simulated a WECS, both in software and HILS, a further renewable energy converter will be simulated, specifically a micro-hydro plant (MHP). The simulation is required to further emphasise the flexibility of the simulator and justify it being referred to as a dynamic simulator for renewable energy converters.

This chapter will detail the theory of a MHP and describe the design and development of the model in Simulink, including both software-only and HILS modelling.

9.1 Theory of Developing Power from Water

A MHP makes use of the basic relationship between a mass of water dropping in height, under gravity, and the force it exerts as it falls. In other words, a MHP converts the potential energy of a mass of water at height into kinetic energy to drive a water turbine. The drop in height, or the gross 'head', the mass of the water and the acceleration due to gravity, relate to the energy released as follows:

$$E = mgh_{gross} \tag{9.1}$$

Where m is the mass of water (Kg), g acceleration due to gravity ($9.81 m/s^2$) , h_{gross} the head (m). Replacing the mass of water by the product of its density, ρ ($1000 kg/m^2$), and its volume, V (m^3) gives:

$$E = V\rho gh_{gross} \tag{9.2}$$

To convert the energy released to gross available power, the volume flow rate, Q (m^3/s), replaces the volume to give:

$$P_{gross} = Q\rho gh_{gross} \tag{9.3}$$

The 'gross' subscript shows the ideal situation where there the are no losses in the MHP. This is not the case (e.g. frictional losses in the penstock, inefficiency of the generator) and the efficiency of the MHP can be as low as 50%. Fig. 9.1 shows the typical arrangement of a MHP and includes the typical losses in various parts of the system [9.1].

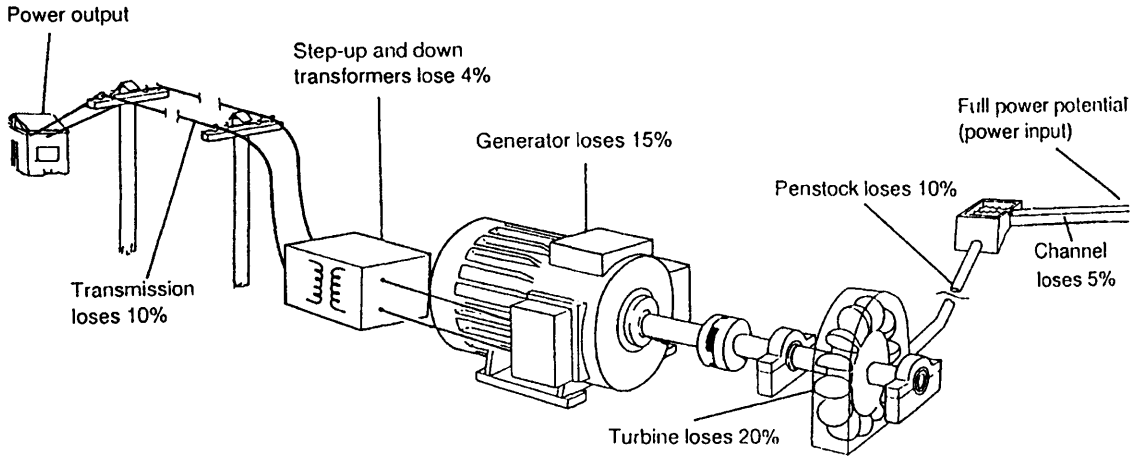


Fig. 9.1 Typical Construction and Losses of a MHP

Some of the processes shown in Fig. 9.1 are omitted in some MHPs. For example, some run of the river schemes do not use a channel and the penstock, with or without a gate, connects direct to the head, while many MHPs use grid-connected generators without the transformer [9.2]. In addition to simplifying the circuit, minimising the complexity improves the overall efficiency. For simplicity purposes, the rest of the study will concentrate on such a system.

The next stage is to assess the inefficiencies of the system, especially the 'front-end' losses such as those associated with the penstock and water turbine, and develop a suitable MHP model in Simulink.

9.2 Modelling the Penstock

The penstock is a length of piping connecting the head to the water turbine, driving an electrical generator. The material used for the penstock can be steel, PVC, even concrete, the selection depends on the cost and the expected pressure requirements of the MHP. Of concern to the modelling of the MHP is the 'roughness' of the inside wall of the pipe and the length of the pipe. The 'roughness' is defined as a factor 'k', in mm, and can range from values of 0.003mm for new PVC piping to 20mm for old, corroded cast iron [9.1]. The effect of the roughness on the penstock model is that it increases the friction, and therefore losses, in the pipe. Longer pipe lengths, L_{pipe}, also increase the losses. These losses are modelled as the 'wall losses', $h_{wallloss}$:

$$h_{wallloss} = \frac{fL_{pipe}0.08Q^2}{d^5} \tag{9.4}$$

f is the friction factor and d is the internal diameter of the pipe. The friction factor is determined from a 'Moody Chart', which depends on the values of k, d and Q. Appendix 9a shows the roughness values for various pipes and the Moody Chart required to establish f.

Additionally, the velocity, v, of the water in the penstock and any bends or obstructions in the pipe, K, contributes to 'turbulence losses', $h_{turbloss}$. Eqn. (9.5) shows the relationship between the velocity and flow rate while Eqn. (9.6) shows the relationship between v and $h_{turbloss}$.

$$v = \frac{4Q}{\pi d^2} \tag{9.5}$$

$$h_{turbloss} = \frac{v^2}{2g}K \tag{9.6}$$

Both the wall and turbulence losses combine to reduce the effective head, h, of the MHP:

$$h = h_{gross} - h_{turbloss} - h_{wallloss} \tag{9.7}$$

9.2.1 Modelling Gate Position

As mentioned before, many MHP systems are simply 'run of the river' types where the penstock is directly connected to the head. These systems tend not to have any gate control and are controlled via the electrical load [9.1]. Some systems do include a gate at the entrance of the penstock and a change in the position of the gate greatly affects the modelling of the penstock.

The gate is normally controlled by an governor, which monitors the speed of the generator. Any deviation of speed and, therefore output power, results in a change in gate position via the governor [9.2]. The control action of the governor can range from proportional-only control to PID control [9.3].

Studies have shown that hydro-turbines exhibit an initial inverse response characteristic of turbine torque to gate changes [9.4]. In other words, a positive step in gate position results in an initial 'transient droop' in the mechanical power. This effect can be modelled using a first order transfer function as follows:

$$\frac{\Delta T_m}{\Delta G} = \frac{1 - T_w s}{1 + 0.5 T_w s} \tag{9.8}$$

ΔT_m is the change in unit torque, ΔG is the change in unit gate position ('1' indicates that the gate is fully open) and T_w is known as the 'water time constant' (s) and is typically '1' [9.3]. The transfer function contains a 'right-hand plane zero' making it potentially unstable [4.2]. It should be noted that this representation is often deemed sufficient, although Ref. [9.4] argues that the first order transfer function is inadequate and should

be replaced with a fourth order model. For the sake of simplicity, the first order model, Eqn.(9.8), will be used for simulation purposes.

9.3 Developing the Model in Simulink

Having established the 'front-end' of an MHP it is now necessary to develop the rest of the MHP model in Simulink and test it. The model will be designed for the grid connected 11kW IM generator, so a similar sized MHP will be proposed and developed. It should be noted that a larger size could be designed with the appropriate torque scaling, as seen in the development of the WECS model. The model will be designed to include the turbine characteristics as well as drive-train and generator dynamics used before with the WECS. Appropriate parameter alterations will be required to suit the MHP requirements.

9.3.1 Modelling the 'Front-End' Dynamics of the MHP

Since no data is available for the simulation and comparison of an existing MHP, it was decided that a suitable model could be derived and designed from the information contained in the references. As stated above, an 11kW system was required, so knowing the approximate losses of the turbine an initial idea of the mechanical power at the turbine, and therefore the flow rate and gross head, can be estimated. Appendix 9b details the estimation of parameters and describes any assumptions made. The design is based on the assumption that a Pelton water turbine, with a efficiency of 75%, will be used [9.1].

The assumptions and calculations from Appendix 9b show that the mechanical power should be 14.67kW, corresponding to a flow rate of 0.05m^3/s and net head (after penstock losses) of 30m. Additionally, the transfer function relating the initial inverse response characteristic of turbine torque to gate changes will need to be combined with these values. To ensure that the complexity of the model is kept to a minimum, no attempt will be made to simulate the effects of a gate position governing system. A change in gate position will be represented as a step demand to the input of the transfer function. This

will suffice to show the characteristics of the penstock and the effect on the model and is similar to the method of modelling in references [9.3]. Fig. 9.2 shows the Simulink model of 'front-end' characteristics. Note that the gate position is represented by a 'step input' module which can be designed to 'step' at anytime in the simulation. The 'switch' is used to by-pass the gate transfer function at 'start-up'. Instability occurs if the switch is not used.

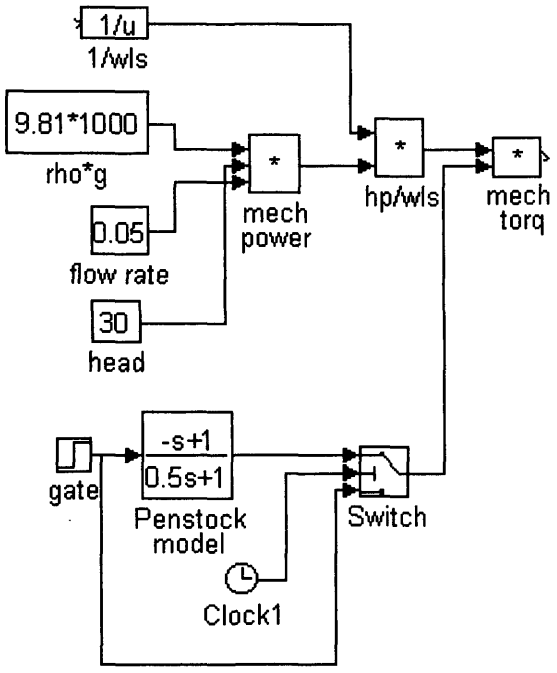


Fig. 9.2 Simulink model for MHP 'Front-End'

The '1/wls' module is used, along with 'hp/wls', to convert the mechanical power into a driving torque. The next stage is to modify the existing WECS drive-train and generator model for use with MHP.

9.3.2 Modelling the Drive-Train and Generator Dynamics

The first adjustment to the drive-train model will be a change in the gearbox ratio. Due to the characteristics of the MHP the gearbox ratio is limited to 3:1 [6.1]. Choosing this value means that the maximum allowed speed for the low speed shaft (LSS), at

synchronous, will be $\frac{157.08}{3} = 52.36 \frac{rad}{s}$. If this is the case, the driving torque to establish the required mechanical power of 14.67kW is $\frac{14.67kW}{52.36} = 280.2Nm$.

The losses of the Pelton turbine can be included in the existing drive train model by adjusting the damping factor, γ_1. The loss can be represented by a reduction of the LSS torque, namely, $Loss = 280.18 \times 0.25 = 70.05Nm$. Since γ_1 relates the loss in torque to the LSS speed, it will have the value $\gamma_1 = \frac{70.045}{52.36} = 1.33$. This estimate does not take into account the losses due to the shaft stiffness and assumes that the generator is running at synchronous speed. Further analysis will need to be performed during simulation.

Since no other drive-train parameters were available for a typical MHP, adjustments to the existing model were performed in an iterative manner, the values being chosen to ensure stability and reduce the initial oscillation at the start of the simulation. It should be emphasised that any modification of these parameters to correspond to any measured data, can be easily implemented.

The derivation of the parameters for the 11kW generator is detailed in Chapter 6. List 9.1 shows the chosen parameters for the MHP drive-train and generator.

$I_1 = 50\ kgm^2$

$I_2 = 0.075\ kgm^2$

$K_1 = 2x10^4\ Nm/rad$

$K_2 = 2.13x10^3\ Nm/rad$

$N = 3$

$\gamma_1 = 1.33$

$\gamma_2 = 0.002$

$D_e = 4.0832\ Nms/rad$

$\tau = 0.0256s$

List 9.1 MHP Drive-Train and Generator Parameters

9.3.3 Software-Only Simulation of MHP

Having established the MHP model parameters, it is simulated in Simulink. Initially, the simulation is performed without the penstock model in order to analyse the mechanical power measurements and assess the choice of damping factor, γ_1.

Table 9.1 shows the simulated LSS power, p_{ls}, and the HSS power, p_{hs}, taken for various values of γ_1.

γ_1	p_{ls} (W)	p_{hs} (W)
1.33	10,990	8,740
1	11,908	9,655
0.5	13,307	11,050

Table 9.1 Power Analysis Measurements

The measurements show that the initial calculation of 1.33 for γ_1 is too high and should be replaced by 0.5 to give a more realistic performance.

To complete the simulation, the penstock model is included and its effect monitored. An arbitrary 'step' in gate response from 60% 'open' to 80% is set 10s into the simulation. Fig. 9.3 shows the response of the LSS torque, t_{ls}, to the step.

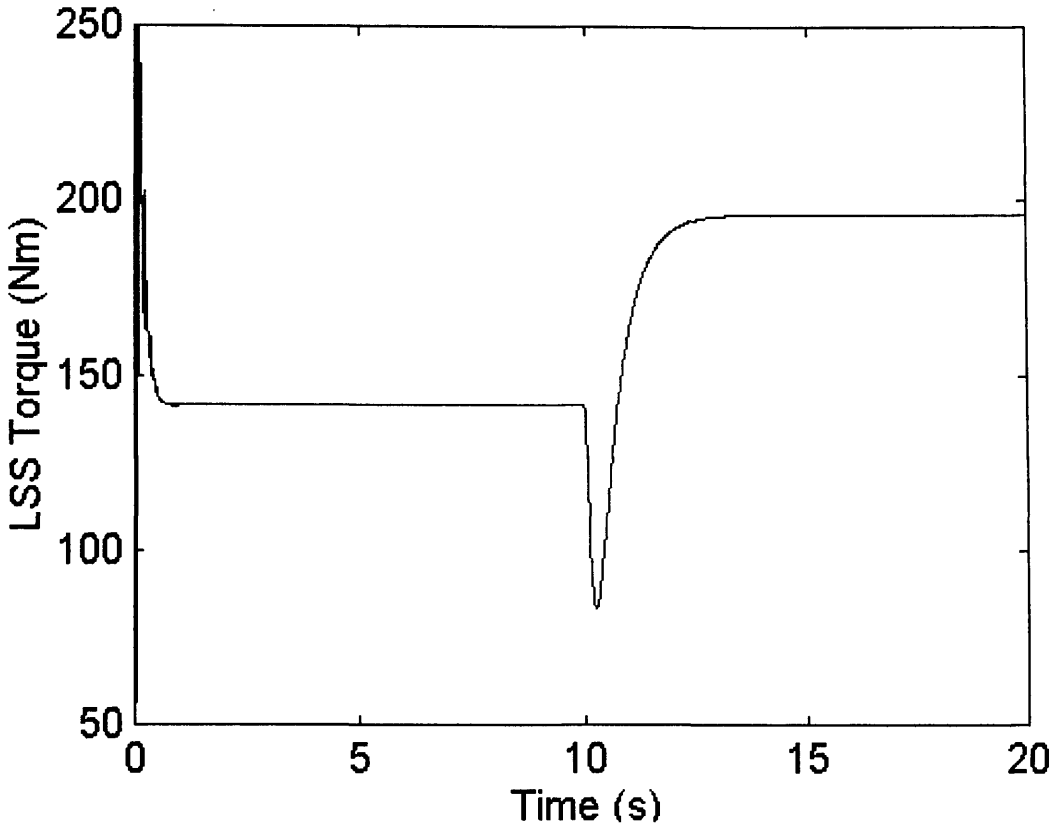


Fig 9.3 Response of t_{ls} to a Gate Step Change

The figure clearly shows that the effect of the change in gate position results in the inverse response characteristic expected from a MHP. This 'transient droop' compares to the results of modelling in references [9.3]. The initial disturbance at the beginning of the simulation is due to the 'initial values' chosen for some of the Simulink modules, as seen with the WECS model.

9.3.4 HILS of MHP

Initial simulations showed that the system was very susceptible to noise and it was difficult to distinguish any changes in the simulation with a change in gate position. Since the parameters of the model were assumed, it was decided to alter the LSS damping factor, which represents the loss of the turbine, from 0.5 back to 1.33 (see Table

9.1). This ensured a stable simulation and Fig 9.4 shows the response of the simulator to a gate step from 60% open to 80%, as before. The measurement is affected by noise, as expected, but comparing Fig. 9.4 and Fig. 9.3, there is a direct comparison between the software-only and HILS simulations.

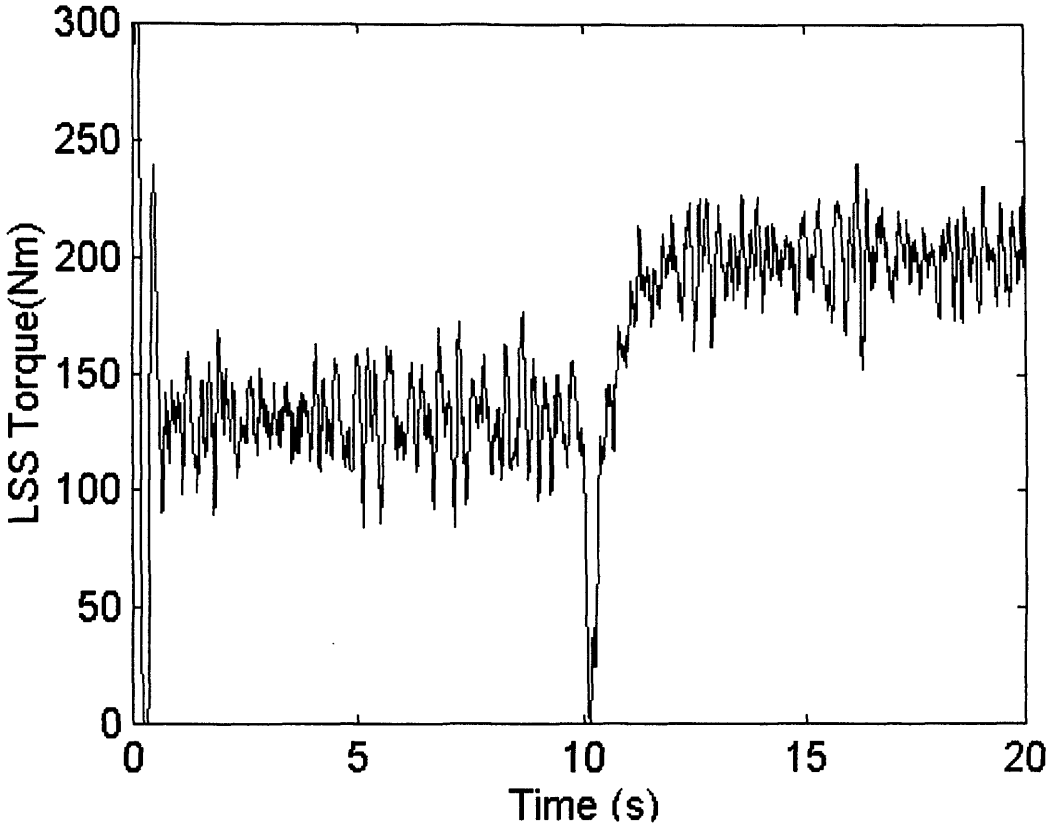


Fig. 9.4 HILS Response to Gate Step

The inverse response characteristic is clearly visible at the point of the gate change (10s). This implies that the simulator provides an accurate, if noisy, response validating its use.

As with the WECS, a number of methods were addressed in order to improve the response, but like the WECS, the noise was predominately due to the speed measurement and could not be improved upon.

9.4 Summary

To emphasise the flexibility and adaptability of the simulator, a micro-hydro plant (MHP) was designed and simulated both as software-only and HILS. Since no 'real' system was available for simulation, system parameters were derived, and a model designed, from information contained in the references. The model defined a new 'front-end', corresponding to the hydrodynamics of a MHP, while the drive-train and generator models retained the same format used in the WECS model. Parameters were altered to suit the MHP.

Unique to the MHP was a model of the penstock representing changes in gate position. An inverse response characteristic was observed at a gate position change, as expected from references, and this was evident both with the software-only and HILS simulations, validating the model. The simulator has been shown to successfully model both a wind energy and hydro energy converter and has emphasised its flexibility to adapt to modifications, regardless of the complexity.

Chapter 10 Conclusions and Recommendations for Further Work

10.1 Design Review

The proposal of the project was to develop a hardware based, fully dynamic simulator for renewable energy converters. Initial studies indicated that efforts would be best served concentrating on a simulator for a wind energy converter system (WECS).

A literature review of current developments in the field of WECS simulators indicated that a definitive, dynamic and flexible simulator had not yet been realised. In other words, there was still a need for a WECS simulator incorporating all the pertinent dynamics of a WECS drive train and the associated aerodynamics. The desire to simulate various renewable energy sources and different schemes of operation, suggested that both the hardware and the software used for simulation needed to be easily adaptable to these requirements.

Once the modelling of the dynamics of the WECS and the relevant aerodynamic effects had been investigated, it was then necessary to assess suitable simulator hardware and software.

Selected hardware consisted of a DC drive and motor connected to a grid-connected induction machine (IM). The DC motor and the IM were mounted side-by-side and connected via a belt and pulley arrangement. The side-by-side arrangement was selected specifically to ensure that the generator could be easily replaced without major modifications to the test-bed.

The selected software for modelling also had to be able to control communication between the PC and the drive. Initially, a serial communication link, manipulating the DC drive's 'on-board' software and communication capability, via the serial port of a PC, was selected. The software chosen to simulate the WECS and control the serial communications was the Mathwork's Simulink with the Real-Time Workshop.

Additional software was designed to provide the novel and unique communication requirements between Simulink and the DC drive.

Having established the hardware, software package and required communication between the two, it was necessary to develop, in Simulink, the proposed software-only model of a constant speed WECS. A model, based on a 45kW WECS, was developed and validated by comparing the performance of the simulation with measured data from the WECS site. All the relevant dynamics and aerodynamics of the WECS were modelled successfully.

Before the simulator hardware could be included for HILS, the effects of using a smaller size generator was assessed in software. The objective was to ensure that the simulation in software was for a 45kW WECS while an 11kW generator was used to return power to the mains. A model of the 11kW generator had to be developed and suitable gain values placed on the torque demand from the software and the speed from the generator. This ensured that the generator driving torque was within its operational capability, and the speed feedback to the gearbox was of the same order as that expected from a 45kW machine.

Once the use of a different size generator was validated, it was necessary to assess the effects of including HILS. The inherent delay due to the serial communications link between Simulink and the Mentor II, the speed measurement quantisation and the noise present on the speed signal were modelled in software. Initial investigations discovered that the speed measurement from the Mentor II using either tachogenerator feedback or an optical speed encoder would cause the simulation to become unstable. Further investigations indicated that connecting the tachogenerator, via an attenuator, to one of the Mentor II general purpose input registers, would allow stable operation.

The response of the motor, due to losses and its inertia, had to be compensated for in software. The dynamics of the motor, generator and pulleys were calculated from test results and a suitable compensation model designed and tested in Simulink.

HILS was established and evaluated. Comparing the simulated data with that of the RAL site data and the software-only simulation indicates that the lower frequency response matched that of the real system. However, the higher frequency response was subject to a higher energy content swamping out the effects of the rotational sampling. This appeared to be due to the noisy speed signal from the drive. To compensate for this noise, alternative gearbox arrangements were designed and tested to limit the effects of the noise in the simulator. Additionally, the motor compensation model was also modified and a first-order digital low pass filter was designed and implemented in Simulink (higher order resulted in instability). Although, these modifications provided some improvement to the lower frequency response, the higher frequency response was still swamped by noise.

In an attempt to improve the response of the simulator an alternative communication method, between the PC and the drive, was developed. A PC based data acquisition card was connected to one of the drive's ADC and DAC channels and communication controlled, once again, by the development of unique driver blocks in Simulink. This arrangement reduced the complexity, and therefore improved the speed of the communication, allowing the sampling rate during simulation to be increased. Once the communication requirements were established, the WECS simulator with HILS could be tested. Results showed that, compared to the serial comms. method, the low to mid-frequency response was improved, especially at the rotational frequency, while the high frequency response was not. The additional ADC and DAC conversions, required due to the presence of the data acquisition card, were blamed for the increase of high frequency energy content.

Finally, to prove the flexibility of the simulator a micro-hydro plant (MHP) was modelled and simulated, both software-only and HILS. Unlike the WECS simulation, no real system was available for comparison, but a realistic model was designed and developed from references. The 'front-end' of the model was established to include the

hydrodynamic properties of a MHP and provide a driving torque for the mechanical section of model. The drive-train and generator models were the same format as that of the WECS, with only minor changes to the parameter values to suit the MHP characteristics. Software-only and HILS simulations gave comparable results.

10.2 Achievements of the WECS Simulator

The main selling point of the Simulator is its flexibility. Throughout the design and development, attention was always given to the fact that the simulator should incorporate various renewable energy converters as well as have the ability to cater for different operating strategies. The design of the hardware is such that the selected generator could easily be replaced without major modifications to the test-bed. For example, larger hydro systems tend to use multi-poled generators operating at a relatively slow speed, compared with the two-pole generator used in this study. If simulation of such a device was required, the new generator could be mounted on the existing test-bed and a suitable pulleys ratio chosen depending on the shaft torque requirements of the generator. The only modifications to the test-bed would be the possible need for new fixing points for the generator. The height of the generator is not of great importance as it would be with an in-line arrangement. The only restriction is that the selected pulley on either or both the motor or generator is not greater than the 96H (388.08 mm) pulley used in this study.

The flexibility of the selected software package of Simulink and Matlab has been exhibited throughout the study. This was especially apparent during the development and validation of the aerodynamic effects such as rotational sampling and spatial filtering for the WECS model. The vast component Simulink library allowed the implementation of modular development of the model with relative ease. Various phenomena, such as the aerodynamic effects could be created as subsystems, and the influence of each assessed prior to the introduction of further dynamics. This modular approach also ensured that model fault-finding during development could be quickly narrowed down to subsystem level. This modular approach was seen to be advantageous when modelling another renewable energy converter, the MHP, where the existing WECS model was easily

modified. Many of the model parts, such as the low speed shaft, gearbox and generator were retained, subject to some parameter adjustment.

Flexibility was further emphasised when adapting the simulator to include the data acquisition card for communication between the PC and drive. Modifications to the Simulink model and, additionally the simulator hardware, were shown to be relatively easy once the particular communication requirements were established and designed in Simulink.

Another advantage of using the selected package is the comprehensive collection of analysing tools. This is particularly evident with the power spectral density comparison of the various simulated data and the measured data, and the assessment of the WECS model transfer function and pole position. The ease of use of such tools aided the simulator development since it prevented the need to create complex algorithms to perform the desired tasks.

One of the most novel aspects of the simulator development is the design of the communications modules for use with the Real-Time Workshop. These modules are essential for HILS in real-time and the ability to include such modules as graphical components in the Simulink models allowed the software-only model to be easily converted to allow for HILS.

The main achievement of the simulator is that it has met the objectives of the initial project brief, in that it has provided a test-bed that can be used to simulate renewable energy converters. Analysis of the response of the WECS simulator using a measured wind speed profile, has shown that its performance is very similar to that of a real, operational WECS under the influence of the same wind profile, while the software-only and HILS simulations of the MHP are also similar. Some work is required to improve the high frequency response of the simulator to match that of real systems, but the cause of the existing difference has been identified.

10.3 Further Developments

The immediate requirement for further development of the simulator is to improve its high frequency response. This could be achieved with the introduction of a high resolution speed encoder in an attempt to reduce the noise on the speed feedback signal, but a digital encoder will be limited by the 10-bit output DAC used to transfer the data to the PC so the advantages may be minimal. As an alternative, an analogue encoder could be used, but again, it is unsure if noise inherent in such a system would cause similar problems.

Another immediate development of the simulator could be the simulation of a variable speed or two speed operation WECS. The drive train of the model used may need to be modified to realise the requirements of variable speed operation. The development of the 'simple' constant speed model neglected some dynamics that may be required during variable speed operation. The relatively large damping inherent with the use of the grid connected induction machine, for example, may not be valid for a synchronous machine, effecting some of the modelling assumptions (see Chapter 2). Also effects such as tower and hub resonant frequencies may be excited by the variable speed operation leading to induced torque in the drive train. These effects, would therefore need to be included in the model.

As mentioned in the previous section, further renewable energy converters, such as wave devices, could be simulated using the simulator. The hardware can be easily modified to allow for such a requirement while the software is flexible enough to ensure that the renewable energy converter can be easily modelled, as shown with the introduction of the MHP. Although further study is required on the modelling of additional converters it is assumed that, as with the WECS and MHP, the outcome of any 'front-end' simulation in software is to provide a driving torque demand for the DC motor and receive a speed feedback from the drive. This would ensure that further development of models would be minimal.

The maximum sampling frequency, at present, is limited to the maximum bandwidth of the drive which is 80Hz. Although acceptable for both the WECS and MHP simulators, it may not be the case for further R.E. converters. If not, an alternative interface between the Real-Time Workshop and the DC motor will be required, by-passing the DC drive. This will, obviously, lead to the development of additional hardware and control algorithms, increasing the complexity of the simulator. Fortunately, this development would not affect either the Simulink modelling or the hardware test-bed.

10.4 Original Contributions

There are a number of novel aspects of the research that should be emphasised. Chapter 1 discussed some existing WECS simulators and highlighted the need for a HILS simulator which should be fully dynamic, including all the mechanical dynamics and aerodynamics, which effect the driving torque of the device. Reviews of current developments, on-going since the initiation of the project has shown that simulators being used are still suffering from the limitations of not including all the dynamic and if so, are failing to compare the simulators with operational WECS [10.1]. This project has addressed these needs and implemented them successfully. This has resulted in a unique simulator which can be used for the development of WECS control schemes, or alternative drive techniques, in the laboratory and not in the hostile environments usually associated with the location of WECS.

Equally unique is the development of the simulator to simulate a MHP system. The simulator includes all the dynamics associated with the mechanical system and hydrodynamics. Reviews of the existing modelling, for such devices, have shown that they are not fully dynamic and they are all software based. Including all the dynamics and ensuring HILS operation makes this area of the project equally unique, once again allowing for the development of MHP control and drive schemes in a laboratory environment.

Although the developments of the WECS and MHP simulators are unique in themselves, the fact that both these schemes are implemented on the same simulator and are readily interchangeable with no modifications, emphasises the unrivalled singularity the simulator represents and confirms the flexibility of the design which was sought from the project conception.

Another novel aspect of the project which needs to be highlighted is the development of the communications between the drive and the PC. The serial communication link which manipulated the programming structure of the Mentor and Simulink software was unique in conception and design. A thorough knowledge of both systems was required before a suitable protocol could be designed and implemented. The resulting code, used for both reading and writing data, provided a reliable and effective means of communication and initially, avoided the need for an expensive data acquisition card. Although limited by its maximum sampling rate of 40Hz, the communication method is a powerful option allowing control of the Mentor DC drive from Simulink. This, therefore, has the potential of being a very useful tool if such an arrangement would be needed for further research. Equally, the inclusion of a data acquisition card also required a similarly unique development of control software in Simulink. The characteristics and operation of the card, again, had to be thoroughly researched before this software could be developed. Adjustments to the Mentor software also had to made to ensure that the correct voltage values were placed at or read from the appropriate DAC and ADC registers respectively. This once again emphasises the flexibility and novelty of the chosen simulator arrangement in that it adapts easily to changes whether in hardware or software.

The final aspect of the simulator that has to be addressed, is the hardware test-rig arrangement. The review of all other simulators, that included some form of HILS, showed that each had a fixed motor-generator arrangement some even having a fly-wheel to simulate the inertia of a WECS. These systems are not designed for flexibility and limit the simulators to testing one or a specific type of particular generator without major re-design of the test-bed. The novelty of the side-by-side arrangement used in this

project, with a pulley-belt system and sliding base, is that a whole range of generator sizes and types can be included in any simulation without any modification to the test bed. This, once again, re-emphasises the flexibility of the simulator and confirms its status as a powerful simulation tool for renewable energy converters.

Chapter 11 References

[1.1] Personal Correspondance with Dr J.A.M. Bleijs, Leicester, Jan. 1995

[1.2] 'A Comparative Analysis of Dynamic Models for Performance Calculation of Grid-Connected Wind Turbine Generators', Chedid R, LaWhite N, Ilic M, Wind Engineering, Vol. 17, No.4, 1993

[1.3] 'Electrical Power Equipment and Measurements - With Heavy Current Electrical Application', Symonds A, McGraw-Hill Book Company (UK) Limited, 1980

[1.4]. 'Basic Control Aspects of WECS', Freris LL, taken from 'Wind Energy Conversion Systems', Freris LL (Editor), Prentice Hall International (UK) Ltd, 1990

[1.5]. 'Rotorsimulator voor de IRFLET - Proefopstelling' (in Dutch), Baltus CWA, Overtoom ATJM, Pierik JTG, ECN 1991

[1.6]. 'Design and Test of the Controller for a Variable Speed Wind Turbine', Leithead WE, Rogers MCM, Pierik JTG, van Engelen TG, ETSU 1994

[1.7]. 'Modelling and Identification of Flexible Wind Turbines and a Factorization Approach to Robust Control', Bongers PMM, Delft University, 1994

[1.8]. 'Combined Multiple Renewable Energy Sources System Simulator Facility' Astinov I L, Bopp G, Consoli A, Lalas D P, Morgana B, Wrixon G T, EWEC 1994, Thessaloniki, Greece, 10-14 October, pp1154-1158

[1.9]. 'A Microprocessor Controlled DC Drive as Simulator of Wind Turbines' Di Napoli A, Crescimbini F, Noia, G, EWEC 1989, Glasgow, UK, 10-13 July, pp687-691

[1.10]. 'Blade-Pitch-Angle-Controllable Windmill Simulator' Toumiya T, Sakakibara T, Suzuki T, International Journal of Energy Research, Vol. 17, p89-104, 1993

[2.1]. 'Generation of Electricity', Freris LL, taken from 'Wind Energy Conversion Systems', Freris LL (Editor), Prentice Hall International (UK) Ltd, 1990

[2.2]. 'Wind Turbine Aerodynamics', Sharpe DJ, taken from 'Wind Energy Conversion Systems', Freris LL (Editor), Prentice Hall International (UK) Ltd, 1990

[2.3]. 'Wind Energy Technology', Walker JF, Jenkins N, UNESCO, John Wiley and Sons, 1997

[2.4]. 'Wind Turbine Engineering Design', Eggleston DM, Stoddard FS, Van Nostrand Reinhold, 1987

[2.5]. Personal correspondance from Dr J Dutton, Rutherford Appleton Labratory, Oxfordshire, UK

[2.6]. 'Final Report for CEC Contract JOUR-0078 Vol. 3 - Jodymod Dynamic Simulation Software Package: Model Description' RAL-94-003

[2.7]. 'Horizontal Axis WECS Design', Armstrong J, Brown A, taken from 'Wind Energy Conversion Systems', Freris LL (Editor), Prentice Hall International (UK) Ltd, 1990

[2.8]. 'Drive-Train characteristics of Constant Speed HAWT's: Part 1 - Representation by Simple Dynamic Models', Leithead WE, Rogers MCM, Wind Engineering, Vol. 20, No. 3, 1996

[2.9]. 'Dynamic Analysis of Wind Turbines for Fatigue Life Prediction' Garrad AD, Hassan U, Garrad Hassan & Partners

[2.10]. 'Development of a Wind Turbine Systems Dynamic Model Using the Automatic Dynamic Analysis of Mechanical Systems (ADAMS) Software', Wright AD, Buhl ML, Elliott AS, National Renewable Energy Laboratory

[2.11]. 'Modeling and Identification of Flexible Wind Turbines and a Factorizational Approach to Robust Control', Bongers PMM, Delft University of Technology, Faculty of Mechanical Engineering and Marine Technology, 1994

[2.12]. 'Forces and Dynamics of Horizontal Axis Wind Turbines', Garrad AD, taken from 'Wind Energy Conversion Systems', Freris LL (Editor), Prentice Hall International (UK) Ltd, 1990

[2.13]. 'Classical Control of Active Pitch Regualtion Of Constant Speed Horizontal Axis Wind Turbines', Leithead WE, De La Salle SA, Reardon D, International Journal of Control, Vol. 55, No. 4, 1992

[2.14]. 'Unsteady Wake Effects Caused By Pitch Angle Changes', Stig-Øye AFM, IEA R&D WECS Joint Action on Aerodynamics of Wind Turbines Symposium, Report of the Technical University of Denmark, 15 October 1986 (s.oye@afm.dtu.dk)

[2.15]. 'Wind Turbine Simulation Model - User Guide and Software Report', Leithead WE, de la Salle S, Reardon D, Department of Energy contract no. E/SA/CON/5108/1851, 1990

[2.16]. 'Simulation of Wind With a Variable 'K' Parameter', Keller JG, Wind Engineering, Vol. 16, No. 6 1992

[2.17]. 'Modelling of Wind Turbines by Simple Models', Wilkie J, Leithead WE, Anderson C, Wind Engineering, Vol. 14, No. 4, 1990

[2.18]. 'A Dynamic Model for Performance Calculations of Grid-Connected Horizontal Axis Wind Turbines, Part 1 - Description of the Model', Sheinman Y, Rosen A, Wind Engineering, Vol. 15, No. 4, 1991

[2.19]. 'Role and Objectives of Control for Wind Turbines', Leithead WE, de la Salle S, Reardon D, IEE Proceedings-C, Vol. 138, No.2, March 1991

[2.20]. 'Drive-Train characteristics of Constant Speed HAWT's: Part II - Simple Characterisation of Dynamics', Leithead WE, Rogers MCM, Wind Engineering, Vol. 20, No. 3, 1996

[3.1]. 'Principles of Electric Machines and Power Electronics', Sen PC, John Wiley and Sons Inc. 1989

[3.2]. 'Control Techniques Drives and Servos Yearbook 1990-1', Control Techniques Plc., 1989

[3.3]. 'Fenner Drive Belts and Pulleys Catalogue', J H Fenner and Co., 1988

[3.4]. Mawdsley data sheet - Personal Correspondence

[3.5]. 'User's Guide for the Mentor II DC Drives', Control Techniques plc 1991

[3.6]. 'Mentor II Supplementary Information', Control Techniques Drives Limited, Jan. 1992

[3.7]. 'Interfacing the IBM-PC to Medical Equipment - The art of Serial Communication', Nickalls RWD, Ramasubramian R, Cambridge University Press, 1995

[3.8]. Watcom C/C++ On-line Help Facility, Version 10.50

[3.9]. 'The C++ Programming Language', Second Edition, Stroustrup B, Addison-Wesley Publishing Company, 1991

[3.10]. 'Software Engineering with C++ and Case Tools', Pont MJ, Addison-Wesley Publishing Company, 1996

[3.11]. 'Digital and Analog Communication Systems', Leon W, Couch II, Macmillan Publishing Company 1990

[3.12]. 'Modelling of Wind Turbines by Simple Models', Wilkie J, Leithead WE, Anderson C, Wind Engineering Vol. 14 No. 4, 1990

[4.1]. 'Matlab User's Guide', The Mathworks Inc., Natick, Massachusetts, 1993

[4.2]. 'Modern Control Systems Analysis and Design Using Matlab', Bishop RH, Addison-Wesley Publishing Company, Inc., 1993

[4.3]. http://www.mathworks.com/products/simulink/, 1998

[4.4]. 'Simulink Dynamic System Simulation Software - User's Guide' The Mathworks Inc., December 1993

[4.5]. 'Simulink Real-Time Workshop'. The Mathworks Inc. 1990-1993

[5.1]. Personal Correspondance with G. Dutton, July 1997, Rutherford Appleton Laboratory, Oxfordshire, UK.

[5.2]. Personal Correspondance with Paynter R, August 1997, Rutherford Appleton Laboratory, Oxfordshire, UK.

[5.3]. 'Signal Processing Toolbox User's Guide', The Mathworks Inc. 1990-1993

[5.4]. 'Wind Turbine Control Systems Modelling and Design Phase I and II - Main Report', Leithead WE, de la Salle SA, Reardon D, Grimble MJ

[6.1]. Personal Correspondance from Greenwood D, Brook Hansen Design Office Huddersfield, 8 Dec 1997.

[6.2] Data Sheet 10748, RS Components 1991

[8.1] 'Amplicon Liveline Catalogue', 1997, Vol. 2

[8.2] 'User Manual for the PC30F and PC30G Series Boards', Eagle Technology, Fifth Edition, 1996

[9.1] 'Micro-Hydro Design Manual - A Guide to Small-Scale Water Power Schemes', Harvery A, Intermediate Technology Publications, 1993

[9.2] 'Assessment of Hydroturbine Models for Power-Systems Studies', Smith JR, McLean R, Robbie JF, IEE Proc., Vol.130, Pt. C, No.1, Jan. 1983

[9.3] 'HydraulicTurbine and Turbine Control Models for System Dynamic Studies', Working Group on Prime Mover and Energy Supply Models for System Dynamic Performance Studies, IEEE Trans. on Power Systems, Vol.7, No. 1, Feb. 1992

[9.4] 'Accurate Low Order Model for Hydraulic Turbine-Penstock', Sanathanan CK, IEEE Trans. on Energy Conversion, Vol. EC-2, No. 2, June 1987

[10.4] 'Control Structures Analysis for a Real Time Wind System Simulator', Nichita C, Diop AD, Belhache J, Dakyo B, Protin L, Wind Engineering, Vol. 22, No. 6, 1998

Appendix 1a

The following is an outline of the technology and designs that are being incorporated in some of the WECS simulators being developed at the moment. Various aspects of aerodynamics and the wind speed properties have been given particular attention and will be used as a comparison base.

The Energy Centre of the Netherlands IRFLET Project

The IRFLET test-rig was constructed under the Dutch Department of Energy FLEXHAT programme which had a stated aim of attaining a reduction in the cost of wind power by 30% [1.5].

The test-rig for the IRFLET project is shown in Fig. A1a.1 [1.6]. It consists of a variable speed drive-train, i.e. a flexible shaft, gearbox, synchronous generator and DC-link, together with a DC machine which mimics the aerodynamic torque corresponding to a specified wind input. The DC machine is computer controlled in order to simulate the aerodynamics of the rotor with the rotor inertia represented by a flywheel. It drives the variable speed drive-train through a gearbox. The rig is rated at 30kW.

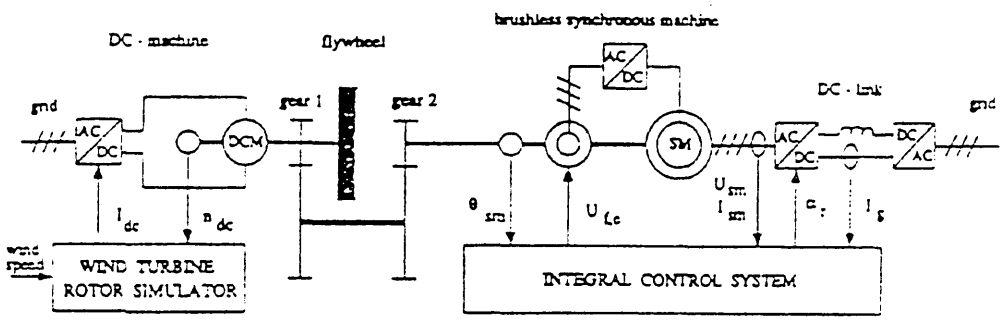


Fig. A1a.1 IRFLET Test Rig

Fig. A1a.2 shows the aerodynamic algorithm used for the test-bed. The diagram shows that the model relies on the kinetic energy relationship between power and wind speed

and not on any further aerodynamics. Subsequent studies have shown that the rig operates on 'stepped' wind speed inputs and is not concerned with any effects due to turbulence.

The main failing of the rig is its inflexibility. As stated above, the rig uses a flywheel to model the inertia. This implies that the rig is only designed to simulate one WECS, unless the flywheel is easily replaced. Even if this were the case, it is very impractical.

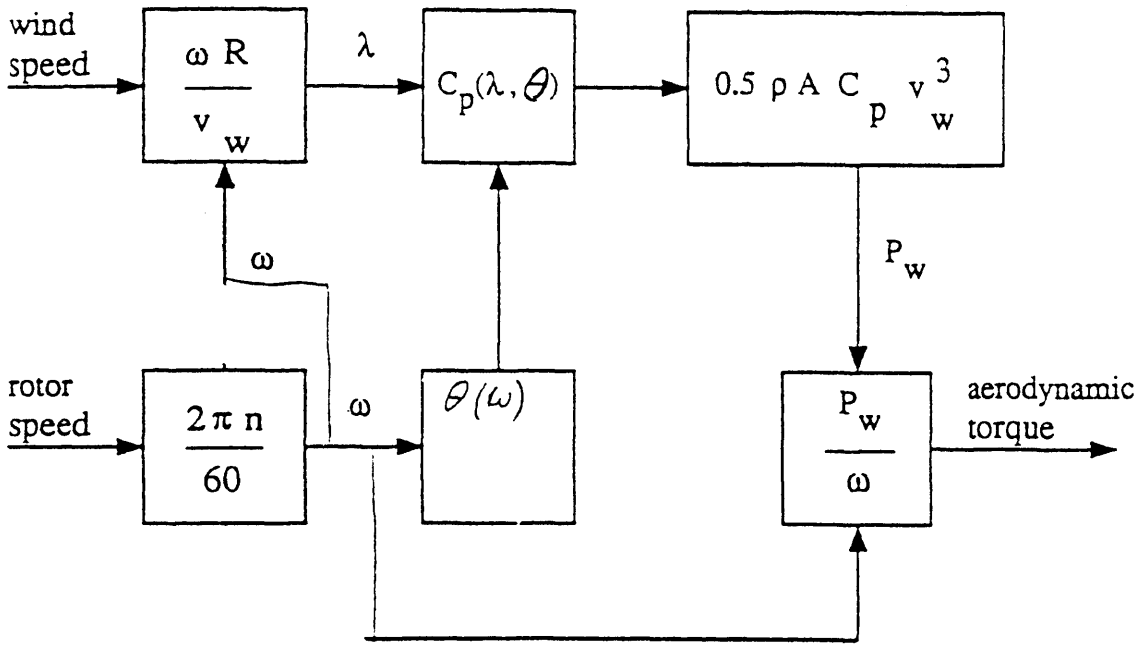


Fig. A1a.2 IRFLET Aerodynamic Algorithm

Delft University of Technology DUWECS

DUWECS, which stands for Delft University Wind Energy Conversion Simulation is a dedicated wind energy PC based simulation package using the Fortran 77 language. The package is such that a complete wind turbine is divided into modules.

A library of modules is contained within the package and can be easily exchanged to form different wind turbine models, for example a rigid tower instead of a flexible tower.

Up to 12 different rotor modules are available with varying degrees of aerodynamic properties. Modules include:

- No rotor dynamics
- Oscillatory rotational sampling (tower shadow)
- Wind shear and yaw
- Two and three bladed models

Reference [1.7] includes results of a DUWECS simulation of a real wind turbine. The simulation includes a rotor module which includes tower shadow and wind shear. Unfortunately, the simulation assumes an average wind profile and neglects turbulence. Comparisons with the measured data from the wind turbine and the simulator are favourable but are only valid for 'stationary values'. In other words, because actual wind fields are unknown, no clear statements about the simulation model could be made [1.7].

SMI Multiple Renewable Energy Sources System Simulator

The Simulation Modelling in Industry (SMI) laboratory based in the Technical University of Sofia, Bulgaria contains a facility to simulate combined multiple renewable energy sources including wind and PV.

The facility is comprised of a 25kWp photovoltaic simulator, a 50kVA wind turbine simulator, two diesel gen-sets with a total power of 80kVA and a battery set with a total capacity of 135kWh. This facility supplies a simulated load of 70kVA. Various power electronic circuitry is used to control the facility [1.8].

The system can be configured to work as a real hybrid plant, having the capability to be configured as four different layouts, and has the ability to vary the size of the components to simulate as many as possible different hybrid systems. All components are connected to supply a stand-alone grid.

The wind turbine simulator is realised by a DC motor driving a three-phase generator and is a constant speed device. The motor is driven by a controlled rectifier linked to a main control computer (Apple Macintosh). This link provides the controller of the rectifier with a value of torque calculated by dedicated software, used to simulate the action of the wind. This torque value is continuously compared with measured torque on the motor/generator shaft.

The torque software is generated using a particular wind turbine's power curves. Detailed aerodynamics do not appear to be included in the simulation and no details of testing are included in the documentation.

In general, the simulation facility is quite large and this, to some extent, limits its flexibility.

University of Rome Wind Simulator

The Department of Electrical Energy, University of Rome, has developed a Wind turbine simulator as seen in Fig. A1a.3 [1.9].

As with the majority of the simulators mentioned in this section, this particular rig uses software to drive a DC motor. Similar to the IRFLET simulator, it includes a flywheel to model the rotor inertia and simulates a constant speed system.

The action of the wind is modelled on the computer by imposing gust models on top of fixed, average wind speeds. No further dynamics are included.

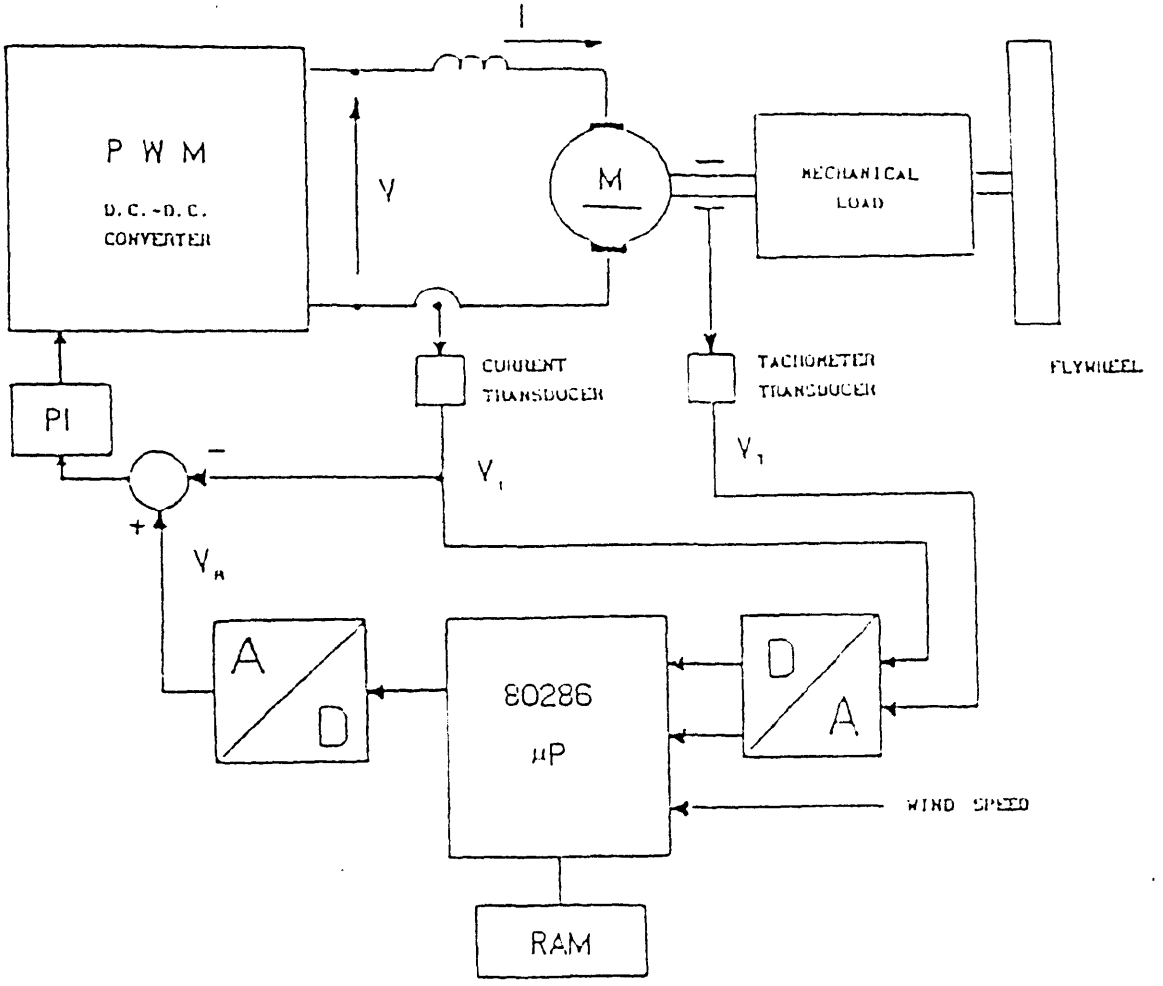


Fig. A1a.3 The University of Rome Wind Turbine Simulator

Blade-Pitch-Angle-Controllable Windmill Simulator

The authors in reference [1.10] have developed a WECS simulator which uses a software controlled DC motor to drive a DC generator. Fig. A1a.4 shows the hardware arrangement.

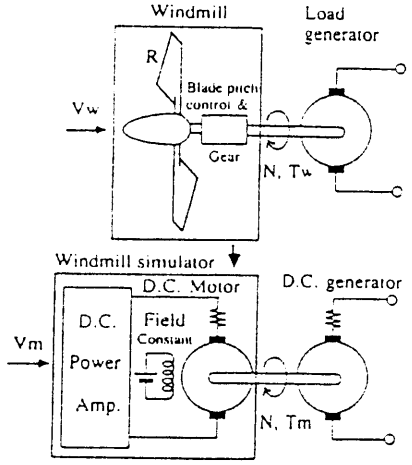


Fig. A1a.4 Hardware Arrangement for 'Windmill Simulator'

The model is constant speed and the facility to change system parameters, such as rotor inertia, is available. Software also includes a model that compensates for the use of the DC motor simulating part of the drive train.

Initially, only the average wind speed values can be used for the determination of the turbine speed/torque characteristics. Further tests indicated that the rig could simulate both transient and dynamic responses. The software also includes a program to generate 'natural wind' with the use of random number generation and appropriate filters. Further tests apparently show that the simulator responds favourably to the 'natural wind'.

The simulator does not appear to include aerodynamic effects such as tower shadow, wind shear and yaw misalignment.

Appendix 2a

Calculation of the First Order Dynamics of an Induction Generator

The time constant, τ, and the torque speed slope, D_e, of a first order model of an induction generator, can be shown to have the following form:

$$\tau = \frac{[K_3 + K_1(K_1K_3 - K_2K_4)]}{\omega_S[K_3^2 + (K_1K_3 - K_2K_4)^2 - (1 + K_1^2)\varepsilon^2]} \qquad \text{(A2a.1)}$$

where the slip, ε is:

$$\varepsilon = \frac{(p\omega_R - \omega_S)}{\omega_S} \qquad \text{(A2a.2)}$$

The K values are defined as:

$$K_1 = \frac{r_s(X'_{lr} + X_m)}{a}, \quad K_2 = \frac{r_s X_m}{a}, \quad K_3 = \frac{r'_r(X_{ls} + X_m)}{a}, \quad K_4 = \frac{r'_r X_m}{a} \qquad \text{(A2a.3)}$$

$$\text{and} \qquad a = X_{ls}X'_{lr} + X_m X'_{lr} + X_m X_{ls} \qquad \text{(A2a.4)}$$

where r_s is the stator resistance, r'_r is the rotor resistance, X_{ls} is the stator leakage reactance, X'_{lr} is the rotor leakage reactance and X_m is the magnetising reactance.

The torque speed slope, D_e, is defined as the rate of change of steady state generation reaction torque, T_{Go} with the slip of the generator. T_{Go} is derived as:

$$T_{Go} = \frac{3}{2}\frac{p}{(\omega_s r_s)}\frac{K_2K_4\varepsilon E^2}{[(K_1K_3 - K_2K_4)^2 + K_3^2 - 2K_2K_4\varepsilon + (1 + K_1^2)\varepsilon^2]} \qquad \text{(A2a.5)}$$

where E is the peak line voltage.

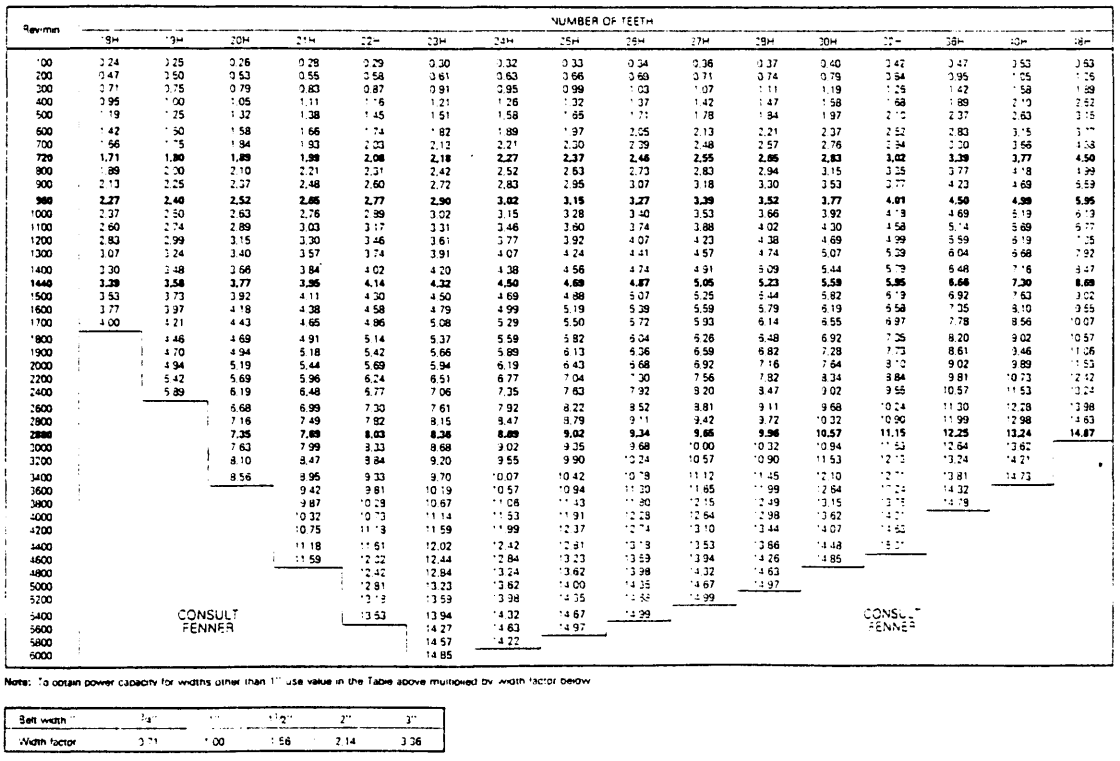
D_e can therefore be defined as:

$$D_e = \frac{\frac{dT_G}{d\varepsilon}}{\frac{d\varepsilon}{d\omega_r}} = \frac{pT_{Go}}{(\omega_s \varepsilon)} \frac{[K_3^2 + (K_1K_3 - K_2K_4)^2 - (1 + K_1^2)\varepsilon^2]}{[(K_1K_3 - K_2K_4)^2 + K_3^2 - 2K_2K_4\varepsilon + (1 + K_1^2)\varepsilon^2]} \qquad \text{(A2a.6)}$$

Appendix 3a

Calculation of Test-bed Dimensions and Pulley Sizes

For the selected 3000rpm, 11kW Brook Crompton Parkinson IM, a suitable pulley has to be chosen to give the desired speed ratio 2:1 (since motor could operate up to 2000rpm without field weakening). As stated in Chapter 3, it is desirable to use a Fenner 96H (38.1mm width, 388.08mm diameter) pulley connected to the DC motor. The Fenner Drives and Pulley's manual is consulted to give the desired ratio. From the manual, it is noted that the selected pulley needs to be 48H [3.3].



Rev/min	NUMBER OF TEETH														
	18H	19H	20H	21H	22H	23H	24H	25H	26H	27H	28H	30H	32H	36H	48H
100	0.24	0.25	0.26	0.28	0.29	0.30	0.32	0.33	0.34	0.36	0.37	0.40	0.42	0.47	0.63
200	0.47	0.50	0.53	0.55	0.58	0.61	0.63	0.66	0.69	0.71	0.74	0.79	0.84	0.95	1.26
300	0.71	0.75	0.79	0.83	0.87	0.91	0.95	0.99	1.03	1.07	1.11	1.19	1.26	1.42	1.89
400	0.95	1.00	1.05	1.11	1.16	1.21	1.26	1.32	1.37	1.42	1.47	1.58	1.68	1.89	2.52
500	1.19	1.25	1.32	1.38	1.45	1.51	1.58	1.65	1.71	1.78	1.84	1.97	2.10	2.37	3.15
600	1.42	1.50	1.58	1.66	1.74	1.82	1.89	1.97	2.05	2.13	2.21	2.37	2.52	2.83	3.77
700	1.66	1.75	1.84	1.93	2.03	2.12	2.21	2.30	2.39	2.48	2.57	2.76	2.94	3.30	4.38
720	**1.71**	**1.80**	**1.89**	**1.99**	**2.08**	**2.18**	**2.27**	**2.37**	**2.46**	**2.55**	**2.65**	**2.83**	**3.02**	**3.39**	**4.50**
800	1.89	2.00	2.10	2.21	2.31	2.42	2.52	2.63	2.73	2.83	2.94	3.15	3.35	3.77	4.99
900	2.13	2.25	2.37	2.48	2.60	2.72	2.83	2.95	3.07	3.18	3.30	3.53	3.77	4.23	5.59
960	**2.27**	**2.40**	**2.52**	**2.65**	**2.77**	**2.90**	**3.02**	**3.15**	**3.27**	**3.39**	**3.52**	**3.77**	**4.01**	**4.50**	**5.95**
1000	2.37	2.50	2.63	2.76	2.89	3.02	3.15	3.28	3.40	3.53	3.66	3.92	4.19	4.69	6.19
1100	2.60	2.74	2.89	3.03	3.17	3.31	3.46	3.60	3.74	3.88	4.02	4.30	4.58	5.14	6.77
1200	2.83	2.99	3.15	3.30	3.46	3.61	3.77	3.92	4.07	4.23	4.38	4.69	4.99	5.59	7.35
1300	3.07	3.24	3.40	3.57	3.74	3.91	4.07	4.24	4.41	4.57	4.74	5.07	5.39	6.04	7.92
1400	3.30	3.48	3.66	3.84	4.02	4.20	4.38	4.56	4.74	4.91	5.09	5.44	5.79	6.48	8.47
1440	**3.39**	**3.58**	**3.77**	**3.95**	**4.14**	**4.32**	**4.50**	**4.69**	**4.87**	**5.05**	**5.23**	**5.59**	**5.95**	**6.66**	**7.30**	**8.69**
1500	3.53	3.73	3.92	4.11	4.30	4.50	4.69	4.88	5.07	5.25	5.44	5.82	6.19	6.92	7.63	9.02
1600	3.77	3.97	4.18	4.38	4.58	4.79	4.99	5.19	5.39	5.59	5.79	6.19	6.58	7.35	8.10	9.55
1700	4.00	4.21	4.43	4.65	4.86	5.08	5.29	5.50	5.72	5.93	6.14	6.55	6.97	7.78	8.56	10.07
1800		4.46	4.69	4.91	5.14	5.37	5.59	5.82	6.04	6.26	6.48	6.92	7.35	8.20	9.02	10.57
1900		4.70	4.94	5.18	5.42	5.66	5.89	6.13	6.36	6.59	6.82	7.28	7.73	8.61	9.46	11.06
2000		4.94	5.19	5.44	5.69	5.94	6.19	6.43	6.68	6.92	7.16	7.64	8.12	9.02	9.89	11.53
2200		5.42	5.69	5.96	6.24	6.51	6.77	7.04	7.30	7.56	7.82	8.34	8.84	9.81	10.73	12.42
2400		5.89	6.19	6.48	6.77	7.06	7.35	7.63	7.92	8.20	8.47	9.02	9.55	10.57	11.53	13.24
2600			6.68	6.99	7.30	7.61	7.92	8.22	8.52	8.81	9.11	9.68	10.24	11.30	12.28	13.98
2800			7.16	7.49	7.82	8.15	8.47	8.79	9.11	9.42	9.72	10.32	10.90	11.99	12.98	14.63
2880			**7.35**	**7.69**	**8.03**	**8.36**	**8.69**	**9.02**	**9.34**	**9.65**	**9.96**	**10.57**	**11.15**	**12.25**	**13.24**	**14.87**
3000			7.63	7.99	8.33	8.68	9.02	9.35	9.68	10.00	10.32	10.94	11.53	12.64	13.62	
3200			8.10	8.47	8.84	9.20	9.55	9.90	10.24	10.57	10.90	11.53	12.13	13.24	14.21	
3400			8.56	8.95	9.33	9.70	10.07	10.42	10.78	11.12	11.45	12.10	12.71	13.81	14.73	
3600				9.42	9.81	10.19	10.57	10.94	11.30	11.65	11.99	12.64	13.26	14.32		
3800				9.87	10.28	10.67	11.06	11.43	11.80	12.15	12.49	13.15	13.78	14.78		
4000				10.32	10.73	11.14	11.53	11.91	12.28	12.64	12.98	13.62	14.21			
4200				10.75	11.18	11.59	11.99	12.37	12.74	13.10	13.44	14.07	14.65			
4400				11.18	11.61	12.02	12.42	12.81	13.18	13.53	13.86	14.48	15.07			
4600				11.59	12.02	12.44	12.84	13.23	13.59	13.94	14.26	14.85				
4800					12.42	12.84	13.24	13.62	13.98	14.32	14.63					
5000					12.81	13.23	13.62	14.00	14.35	14.67	14.97					
5200					13.19	13.59	13.98	14.35	14.69	14.99						
5400			CONSULT		13.53	13.94	14.32	14.67	14.99				CONSULT			
5600			FENNER			14.27	14.63	14.97					FENNER			
5800						14.57	14.92									
6000						14.85										

Note: To obtain power capacity for widths other than 1" use value in the Table above multiplied by width factor below

Belt width "	3/4"	1"	1 1/2"	2"	3"
Width factor	0.71	1.00	1.56	2.14	3.06

Table A3a.1 Power Ratings (kW) of Pulleys Against Speed

The next stage is to confirm that the rating of the pulleys is compatible with the rating of the 11kW IM. Table A3a.1 shows the rating of the 'Heavy' drives. For a 38.1mm width, 194.04mm diameter 48H pulley the power rating is calculated to be:

$$\text{Rating} = 14.87\text{x}10^3 \text{ x } 1.56$$

$$= \mathbf{23.197kW}$$

This is obviously, more than efficient for the desired test-bed.

An estimation of the belt length is then required. Before this is possible, the design of the base of the test-bed is considered. As stated in the main text, it is anticipated that the base, would be able to cater for a number of generators. It was decided that the best option is to rigidly fix the DC motor and have the generator mounting as a movable plate. The generators would be fixed to the plate and the plate adjusted until the required tension of the belt is achieved. Table A3a.2 shows the dimensions of three machines available in the department which are considered for the test-bed, as well as the DC motor.

Machine	Feet Width (mm)	Feet Length (mm)	Front to Feet (mm)
15kW Synchronous	280	275	215
10kW Induction	220	240	170
11kW Induction	250	215	215
15kW DC Motor	215	510	160

Table A3a.2 Machine Dimensions

From the table it is deduced that the length of the base is dependant on the length of the motor since it is the longest. The eventual length of the base is selected to be 1m in order to ensure stabilisation of the base with the machines mounted on it.

Since the DC motor has the larger pulley attached to its shaft, the mounting for it has to account for the 388.08 mm diameter. This is achieved by placing the motor 100mm higher than the base.

When considering the width of the base, not only is the distance between the feet mountings required, but also the diameter of the pulleys. The generator with the largest feet width is the synchronous generator with a width of 280mm. This, and the motor width gives a total of 495mm. The width of the pulleys totals 582.12mm. To ensure stability, the total width of the base is selected to be 1m.

With this data, the length of the timing belt required for use with the motor and the 11kW IM is calculated. The following is the calculation recommended by Fenner.

Firstly, the required distance between the centre points of the two machines, with pulleys and the belt, is required. To save costs, the length of the belt is to be as small as possible without affecting performance. Initial estimations indicate a distance of 450mm is acceptable. This value is assigned the variable, C, and related to the other pulley and belt characteristics, as follows:

$$L = 2C + \frac{(D+d)^2}{4C} + 1.57(D+d) \qquad \text{(A3a.1)}$$

Where

L = Required pitch length of belt in mm

D = Pitch diameter of large pulley

d = Pitch diameter of smaller pulley

In this example,

$$L = (2*450) + \frac{(388.08-194.04)^2}{4*450} + 913.93$$

$$L = 1834.8 \text{ mm}$$

The belt size, is found by multiplying the above value by 2.54. This gives:

$$L_{corr} = 722.38$$

According to the Fenner information, the nearest sizes to this were 700 (1778 mm) or 750 (1905 mm). The values of C corresponding to these two values is calculated as follows

$$C = A + \sqrt{A^2 - B}$$

where

$$A = \frac{L}{4} - 0.3925(D + d) \text{ and } B = \frac{(D-d)^2}{8}$$

The two values of L gave C = 431.98 mm and C = 495.49 mm. The former value is sufficient so the 700H150 (700, 38.1 mm, heavy duty) belt is selected.

Appendix 3b

The Input and Output Connections of the Mentor II

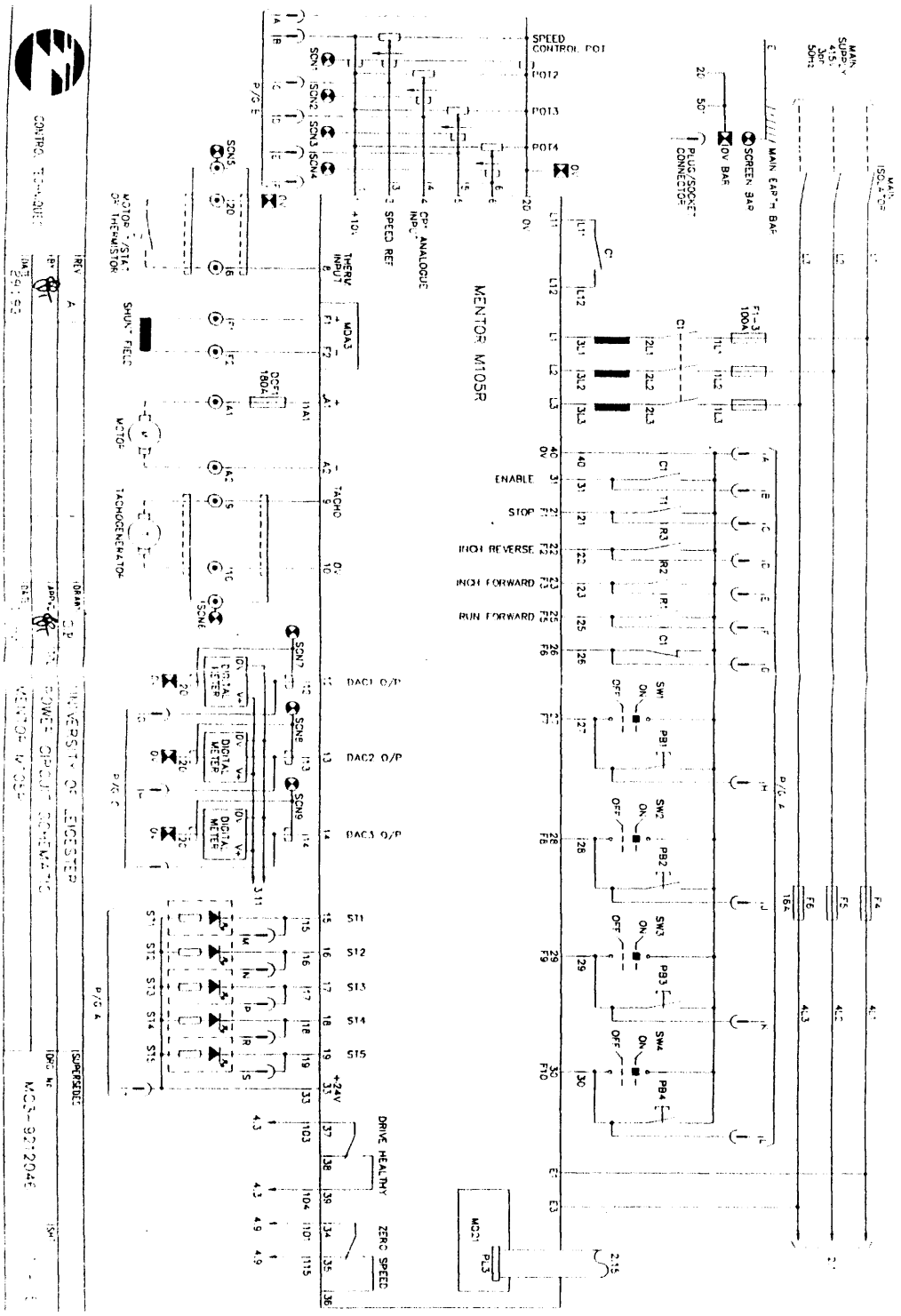


Fig. A3b.1 The External Connections of the Mentor II

The Mentor II Menu 14 - MD21 Control

Menu Parameter	Description
14.01	Drive number (usually set to 1 when using one drive)
14.02	0 - ANSI comms 1 - Basic serial comms
14.03	Baud rate (x100), i.e. '192' for 19200 baud rate
14.04	Line pacing character (Basic serial comms)
14.05	Enable or disable autobooting
14.06	Line feed enable (Basic comms)
14.07	Enable or disable '>' prompt (Basic serial comms), enable or disable checksum (ANSI comms)
14.08	1st bit of data format (Basic serial comms) set to '0' for ANSI to operate with Mentor II
14.09	2nd bit of data format (Basic serial comms)
14.10	Enable or disable Intel Basic (disabled for ANSI)
14.11 to 14.17	Enable or disable application programmes

Table A3b.1 Serial Communications Configuration Menu

C Code for Basic Comms 'Read' Command

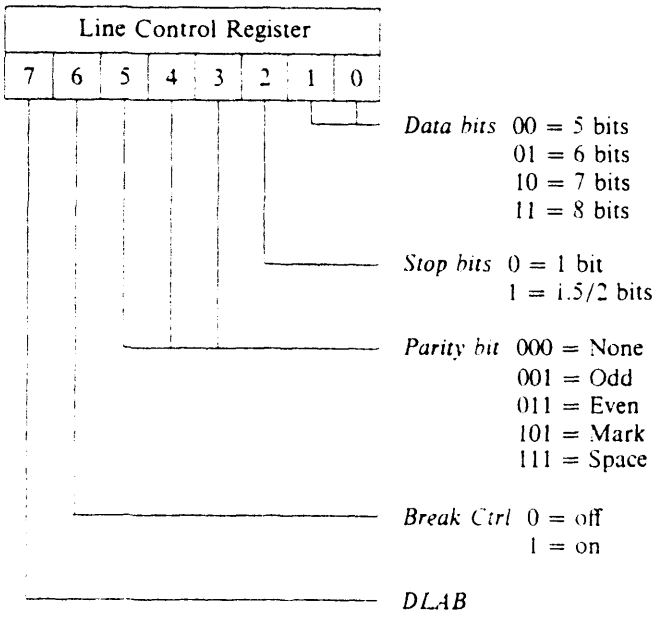
```
#include <conio.h>
#include <math.h>
#include <iostream.h>
#include <iomanip.h>
#define LCR 0x2fb
#define PORT 0x2f8
#define IER 0x2f9
#define MCR 0x2fc
#define MSR 0x2fe
#define LSR 0x2fd

int readp(int, char* const);
int echo(int, char* const);
int writep();

int main()
    {
```

```
    /*Initialisation*/
    outp(LCR, 0x80);      //initiate baud set-up
    outp(PORT, 0x06);     //lsb baud
    outp(IER, 0x0);       //msb baud set to 19200
    outp(LCR, 0x1a);      //7e1
    outp(IER, 0x0);       //disable interrupts
    outp(MCR, 0x0);       //RTS=0
    /* end of init */

   here:writep();
   goto here;  //loop
return 0;
    }

/*****************************************************************/
/*******************FUNCTIONS*************************************/
/*****************************************************************/

int writep()
    {
      int numb, element=0;
      char f_m_[50];
      char pc_instr[7] = {'#','0','3','.','0','4'}; // set reg 1.0 to value
      pc_instr[6] = (int) 0xd;      // terminate array with a carriage return
      for (numb=0;numb <= 6;numb++)          //create loop to o/p array
        {
          while ((inp(LSR)& 0x60) != 0x60); //check for empty TXB
          outp(PORT, (int) pc_instr[numb]);    //o/p chr to port
          echo(element, &f_m_[0]);  //read characters back from the MII
          element++; //next location of array
        }
      readp(element, &f_m_[0]); //get rest of data from MII
      return 0;
    }

/************Function to read echoed characters from the mentor**********/
int echo(int num, char* const from_mentor)
    {
      while((inp(LSR)& 0x1) != 1); //check for data ready in RXB
      from_mentor[num] = (char)inp(PORT); //input data
      return 0;
    }

/*******************Function to read data from Mentor*********************/
int readp(int num, char* const from_mentor)
    {
      do{
        while((inp(LSR)& 0x1) != 1); //check for data ready in RXB
        from_mentor[num] = (char)inp(PORT);    //input data
        num++;                    //next array position
       } while ((from_mentor[(num -1)] != '>') && num < 50);//check for end of data

      from_mentor[num] = '\0'; // put NULL character at end of array
```

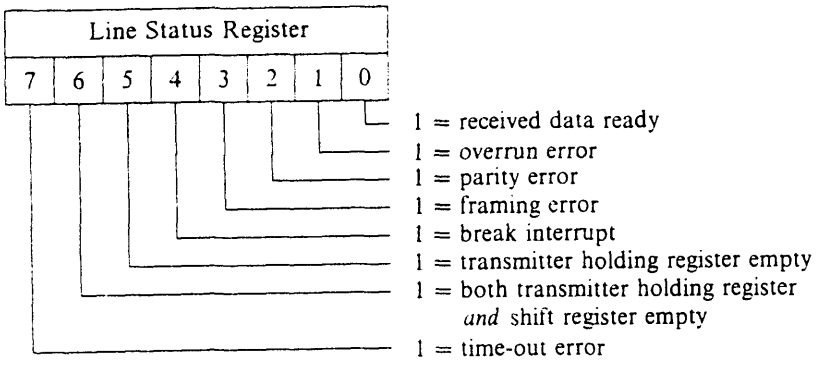
```
    cout<< from_mentor;
    return 0;
}
```

The UART 8250 Control and Status Registers

Line Control Register



Line Status Register



Baud Rate Divisor

Baud Rate Divisor															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
high byte register (BDRH)								low byte register (BDRL)							

Communicating Between the PC and the Mentor II

Writing Data to the Mentor II

```
10 OPEN "COM2:2400,N,8,1,CD,CS5000,DS,OP5000,RS,RB2048" FOR RANDOM ACCESS READ
WRITE AS #1
20 DO
30    PRINT #1, "#1.20"; CHR$(44); "300"
40 FOR I = 1 TO 1000
50 NEXT I
60 PRINT "SENDING DATA"
70 LOOP
80 END
```

Reading Data from the Mentor II

```
10 OPEN "COM2:2400,N,8,1,CD,CS5000,DS,OP5000,RS,RB2048" FOR RANDOM ACCESS READ
WRITE AS #1
20 DO
40    PRINT #1, "#03.04"
50    AN$ = " "
80    DO
90          BN$ = INPUT$(LOC(1), #1)
100         AN$ = AN$ + BN$
105   LOOP UNTIL BN$ = ">"
110   PRINT AN$
120 LOOP
```

Appendix 4a

Watcom C/C++ Template Makefile

```
# File    : drt_wat2.tmf
# Abstract:
#       Template makefile for building a DOS-based real-time
#       version of a SIMULINK model using generated C code
#       and the Watcom C/386 Compiler with Watcom WMAKE
#
#       Note that this template can be automatically customised using
#           make_nrt.m and the "Generate and Build Real-Time" option under the
#           "Code" menu heading.


#------------------------ Macros Read By make_rt -------------------------
#
# Note: These macros are parsed by make_rt.  Thus they should not contain
#       other macros, as these macros will not be expanded.
#
MAKE = wmake
QUOTE = "
HOST = PC
BUILD = yes

#------------------------- Customisation Macros -------------------------
#
# The following set of macros are customised by the make_rt program.
#
MODEL         = |>MODEL_NAME<|
MAKEFILE      = |>MAKEFILE_NAME<|
S_FUNCTIONS   = |>S_FUNCTION_FILENAMES<|
S_FUNCTIONS_OBJ = |>S_FUNCTION_OBJ_FILENAMES<|
INTEGRATOR    = |>INTEG_SRC_FILENAME<|
INTEGRATOR_OBJ  = |>INTEG_OBJ_FILENAME<|
LOGGER        = |>LOG_SRC_FILENAME<|
LOGGER_OBJ    = |>LOG_OBJ_FILENAME<|
COMM_LINK     = |>COMM_LINK_FILENAME<|

INTEG_DEFINES  = |>INTEG_DEFINES<|
LOGGING_DEFINES = |>LOGGING_DEFINES<|

MATLAB_ROOT   = |>MATLAB_ROOT<|

#------------------------------ Tool Locations ------------------------------
#
# Modify the following three macros to reflect where you have installed
# MATLAB, the Watcom C/386 Compiler, and the Phar Lap Assembler.  The
# Phar Lap Assembler is not required.  It is only used to compile rt_fpu.asm,
# which is also supplied in object file form.
#

WATCOM_ROOT = v:\watcom\v10.50
```

```
#PHARLAP_ROOT = c:\phar386

#---------------------------- Tool Definitions ---------------------------

!ifeq %OS Windows_NT
CC = $(WATCOM_ROOT)\binnt\wcc386
LD = $(WATCOM_ROOT)\binnt\wcl386
!else
CC = $(WATCOM_ROOT)\binw\wcc386
LD = $(WATCOM_ROOT)\binw\wcl386
!endif

#AS = $(PHARLAP_ROOT)\bin\386asm

#------------------------------ Include Path -----------------------------

CODEGEN_ROOT = $(MATLAB_ROOT)\codegen

MATLAB_INCLUDES = &
 -I$(MATLAB_ROOT)\simulink\include &
 -I$(CODEGEN_ROOT)\common\include &
 -I$(CODEGEN_ROOT)\rt\common &
 -I$(CODEGEN_ROOT)\rt\dos\os

COMPILER_INCLUDES = -I$(WATCOM_ROOT)\h

INCLUDES = $(MATLAB_INCLUDES) $(COMPILER_INCLUDES)

#-------------------------------- C Flags --------------------------------

REQ_OPTS = -fpi87 -3s
OPT_OPTS = -oaxt
DBG_OPTS = -d2
OPTS =
CC_OPTS = $(REQ_OPTS) $(OPT_OPTS) $(DBG_OPTS) $(OPTS)

CPP_REQ_DEFINES = -DMODEL_NAME=$(MODEL)

CFLAGS = $(CC_OPTS) $(INCLUDES) $(CPP_REQ_DEFINES) $(CPP_DEFINES) &
 $(INTEG_DEFINES) $(LOGGING_DEFINES)

ASFLAGS = -twocase -nolist
LDFLAGS = -l=dos4g -x

#---------------------------- Source Files -----------------------------

REQ_SRCS = $(MODEL).c rt_main.c rt_keybd.c rt_sim.c simstruc.c rt_cpu.c
OPT_SRCS = timer.c
S_FCN_SRCS = $(S_FUNCTIONS)
INT_SRCS = $(INTEGRATOR)
LOG_SRCS = $(LOGGER)

C_SRCS = $(REQ_SRCS) $(OPT_SRCS) $(S_FCN_SRCS) $(INT_SRCS) $(LOG_SRCS)

REQ_OBJS = $(MODEL).obj rt_main.obj rt_keybd.obj rt_sim.obj simstruc.obj rt_cpu.obj
```

```
OPT_OBJS = timer.obj
S_FCN_OBJS = $(S_FUNCTIONS_OBJ)
INT_OBJ = $(INTEGRATOR_OBJ)
LOG_OBJ = $(LOGGER_OBJ)
C_OBJS = $(REQ_OBJS) $(OPT_OBJS) $(S_FCN_OBJS) $(INT_OBJ) $(LOG_OBJ)

ASM_SRCS = rt_fpu.asm modf.asm frexp.asm fft387.asm
ASM_OBJS = rt_fpu.obj modf.obj frexp.obj fft387.obj

OBJS = $(C_OBJS) $(ASM_OBJS)

LIBS =

# Source Path
.c :
$(MATLAB_ROOT)\simulink\src;$(CODEGEN_ROOT)\rt\common;$(CODEGEN_ROOT)\rt\dos\os;$(C
ODEGEN_ROOT)\rt\dos\devices

.asm : $(CODEGEN_ROOT)\rt\dos\os

#---------- Exported Environment Variables ----------
#
# Because of the 128 character command line length limitations in DOS, we
# use environment variables to pass additional information to the WATCOM
# Compiler and Linker
#
!ifeq %OS Windows_NT
PATH = $(WATCOM_ROOT)\binnt;$(WATCOM_ROOT)\bin;$(WATCOM_ROOT)\binb
!else
PATH = $(WATCOM_ROOT)\bin;$(WATCOM_ROOT)\binw;$(WATCOM_ROOT)\binb
!endif
WATCOM = $(WATCOM_ROOT)

#---------- Rules ----------

.BEFORE
        @set path=$(PATH)
        @set WATCOM=$(WATCOM)
        @set WCM_VER=100
        @if exist $(MODEL).lnk @del $(MODEL).lnk
        @for %i in ($(OBJS)) do @echo FILE %i >> $(MODEL).lnk

$(MODEL).exe : $(OBJS)
        $(LD) /fe=$(MODEL).exe $(LDFLAGS) @$(MODEL).lnk $(LIBS)
        @echo * * * * * *  Make of $(MODEL).EXE complete * * * *
        del $(MODEL).lnk

.c.obj:
        *$(CC) $(CFLAGS) $<

.asm.obj:
        @if exist $(AS).exe $(AS) $(ASFLAGS) $< -o $@
        @if not exist $(AS).exe copy $[*.obj $@

fft387.obj: $(CODEGEN_ROOT)\rt\dos\os\fft387.obj
```

```
	copy $[*.obj $@

#---------------------------- Dependencies ----------------------------

$(OBJS) : $(MAKEFILE) .AUTODEPEND

rt_main.obj : $(MODEL).c .AUTODEPEND
```

Spdin.c

```
/*
 *  spdin.c
 *
 * Copyright (c) 1994 by The MathWorks, Inc.
 * All Rights Reserved
 */

/*
 * STEP1 The following #define is used to specify the name of your S-Function.
 *
 * You should change the define to include the name of your S-function.
 */

#define S_FUNCTION_NAME spdin

/* STEP2
 * Need to include simstruc.h for the definition of the SimStruct and
 * its associated macro definitions.
 */

#include "simstruc.h"

/*STEP3
 *Include file for MEX file
*/

#ifdef MATLAB_MEX_FILE
#include "mex.h"
#endif

#include <conio.h>     //for outp & inp
#include <string.h>    //for str* commands
#include <stdlib.h>    //for atoi
#include <ctype.h>     //for isdigit
#define LCR 0x2fb
#define PORT 0x2f8
#define IER 0x2f9
#define MCR 0x2fc
#define MSR 0x2fe
#define LSR 0x2fd


/* STEP4
 * mdlInitializeSizes - initialize the sizes array
```

```
*
* The sizes array is used by SIMULINK to determine the S-function block's
* characteristics (number of inputs, outputs, states, etc.).
*/

static void mdlInitializeSizes(SimStruct *S)
{
   ssSetNumContStates(    S, 0);     /* number of continuous states */
   ssSetNumDiscStates(    S, 0);     /* number of discrete states */
   ssSetNumInputs(        S, 0);     /* number of inputs */
   ssSetNumOutputs(       S, 1);     /* number of outputs */
   ssSetDirectFeedThrough(S, 0);     /* direct feedthrough flag */
   ssSetNumSampleTimes(   S, 1);     /* number of sample times */
   ssSetNumInputArgs(     S, 1);     /* number of input arguments */
   ssSetNumRWork(         S, 0);     /* number of real work vector elements */
   ssSetNumIWork(         S, 0);     /* number of integer work vector elements */
   ssSetNumPWork(         S, 0);     /* number of pointer work vector elements */
}

/*STEP5
 * mdlInitializeSampleTimes - initialize the sample times array
 *
 * This function is used to specify the sample time(s) for your S-function.
 * If your S-function is continuous, you must specify a sample time of 0.0.
 * Sample times must be registered in ascending order. If your S-function
 * is to acquire the sample time of the block that is driving it, you must
 * specify the sample time to be INHERITED_SAMPLE_TIME.
 */

static void mdlInitializeSampleTimes(SimStruct *S)
{
   ssSetSampleTimeEvent(S, 0, mxGetPr(ssGetArg(S,0))[0]);
   ssSetOffsetTimeEvent(S, 0, 0.0);

   /*
    * SET OTHER SAMPLE TIMES AND OFFSETS HERE
    */
}

/*STEP6
 * mdlInitializeConditions - initialize the states
 *
 * In this function, you should initialize the continuous and discrete
 * states for your S-function block. The initial states are placed
 * in the x0 variable. You can also perform any other initialization
 * activities that your S-function may require.
 */

static void mdlInitializeConditions(double *x0, SimStruct *S)
{
        /*Initialisation*/
  int clear;
  outp(LCR, 0x80);       //initiate baud set-up
  outp(PORT, 0x06);      //lsb baud
  outp(IER, 0x0);       //msb baud set to 19200
```

```
 outp(LCR, 0x1a);      //7e1
 outp(IER, 0x0);       //disable interrupts
 outp(MCR, 0x0);       //RTS=0
 clear=inp(PORT);


}

/*
 * mdlOutputs - compute the outputs
 *
 * In this function, you compute the outputs of your S-function
 * block.  The outputs are placed in the y variable.
 */

static void mdlOutputs(double *y, double *x, double *u, SimStruct *S, int tid)
{
          int numb, num=0;
        int array=0, loop;
        char from_mentor[20], small[5];
          char pc_instr[13] = {0x4,'0','0','1','1','0','3','0','2',0x5}; //'read 03.02'
        for (numb=0;numb <= 9;numb++)            //create loop to o/p array
           {
             while ((inp(LSR)& 0x60) != 0x60);  //check for empty TXB
             outp(PORT, (int) pc_instr[numb]);    //o/p chr to port
           }

            do{
           while((inp(LSR)& 0x1) != 1); //check for data ready in RXB
           from_mentor[num] = (char)inp(PORT);    //input data
           num++;                       //next array postion
         } while ((int)from_mentor[num-1] != 0x3);      //check for end of data

        from_mentor[num] ='\0';     // null array

        for (loop=num-6; loop<num-1; loop++)   //start loop to miss first 6 chars
        {
            small[array]=from_mentor[loop];
            array ++;
        }
        small[array] = '\0';       // NULL at end of array
        *y = atof(small);       // convert string to double


}

/*STEP 8 - PART 1
 * mdlUpdate - perform action at major integration time step
 *
 * This function is called once for every major integration time step.
 * Discrete states are typically updated here, but this function is useful
 * for performing any tasks that should only take place once per integration
 * step.
 */
```

```
static void mdlUpdate(double *x, double *u, SimStruct *S, int tid)
{
  /*
   * YOUR CODE GOES HERE
   */
}

/*STEP8 - PART 2
 * mdlDerivatives - compute the derivatives
 *
 * In this function, you compute the S-function block's derivatives.
 * The derivatives are placed in the dx variable.
 */

static void mdlDerivatives(double *dx, double *x, double *u, SimStruct *S, int tid)
{
  /*
   * YOUR CODE GOES HERE
   */
}

/* STEP9
 * mdlTerminate - called when the simulation is terminated.
 *
 * In this function, you should perform any actions that are necessary
 * at the termination of a simulation.  For example, if memory was allocated
 * in mdlInitializeConditions, this is the place to free it.
 */

static void mdlTerminate(SimStruct *S)
{
  /*
   * YOUR CODE GOES HERE
   */
}
// STEP10
#ifdef  MATLAB_MEX_FILE   /* Is this file being compiled as a MEX-file? */
#include "simulink.c"     /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"      /* Code generation registration function */
#endif
```

Torout.c

```
/*
 * torout.c
 *
 *  Copyright (c) 1994 by The MathWorks, Inc.
 *  All Rights Reserved
 */

/*
 * STEP1 The following #define is used to specify the name of your S-Function.
 *
 * You should change the define to include the name of your S-function.
 */

#define S_FUNCTION_NAME torout

/* STEP2
 * Need to include simstruc.h for the definition of the SimStruct and
 * its associated macro definitions.
 */

#include "simstruc.h"

/*STEP3
 *Include file for MEX file
*/

#ifdef MATLAB_MEX_FILE
#include "mex.h"
#endif

#include <conio.h>     //for outp & inp
#include <string.h>    //for str* commands
#include <stdlib.h>          //for abs
#define LCR 0x2fb
#define PORT 0x2f8
#define IER 0x2f9
#define MCR 0x2fc
#define MSR 0x2fe
#define LSR 0x2fd


/* STEP4
 * mdlInitializeSizes - initialize the sizes array
 *
 * The sizes array is used by SIMULINK to determine the S-function block's
 * characteristics (number of inputs, outputs, states, etc.).
 */

static void mdlInitializeSizes(SimStruct *S)
{
   ssSetNumContStates(    S, 0);      /* number of continuous states */
```

```
  ssSetNumDiscStates(   S, 0);    /* number of discrete states */
  ssSetNumInputs(       S, 1);    /* number of inputs */
  ssSetNumOutputs(      S, 0);    /* number of outputs */
  ssSetDirectFeedThrough(S, 1);   /* direct feedthrough flag */
  ssSetNumSampleTimes(  S, 1);    /* number of sample times */
  ssSetNumInputArgs(    S, 1);    /* number of input arguments */
  ssSetNumRWork(        S, 0);    /* number of real work vector elements */
  ssSetNumIWork(        S, 0);    /* number of integer work vector elements */
  ssSetNumPWork(        S, 0);    /* number of pointer work vector elements */
}

/*STEP5
 * mdlInitializeSampleTimes - initialize the sample times array
 *
 * This function is used to specify the sample time(s) for your S-function.
 * If your S-function is continuous, you must specify a sample time of 0.0.
 * Sample times must be registered in ascending order. If your S-function
 * is to acquire the sample time of the block that is driving it, you must
 * specify the sample time to be INHERITED_SAMPLE_TIME.
 */

static void mdlInitializeSampleTimes(SimStruct *S)
{
  ssSetSampleTimeEvent(S, 0, mxGetPr(ssGetArg(S,0))[0]);
  ssSetOffsetTimeEvent(S, 0, 0.0);

  /*
   * SET OTHER SAMPLE TIMES AND OFFSETS HERE
   */
}

/*STEP6
 * mdlInitializeConditions - initialize the states
 *
 * In this function, you should initialize the continuous and discrete
 * states for your S-function block.  The initial states are placed
 * in the x0 variable.  You can also perform any other initialization
 * activities that your S-function may require.
 */

static void mdlInitializeConditions(double *x0, SimStruct *S)
{
        /*Initialisation*/
 outp(LCR, 0x80);      //initiate baud set-up
 outp(PORT, 0x06);     //lsb baud
 outp(IER, 0x0);      //msb baud set to 19200
 outp(LCR, 0x1a);      //7e1
 outp(IER, 0x0);      //disable interrupts
 outp(MCR, 0x0);      //RTS=0


}

/*
 * mdlOutputs - compute the outputs
```

```
*
* In this function, you compute the outputs of your S-function
* block.  The outputs are placed in the y variable.
*/

static void mdlOutputs(double *y, double *x, double *u, SimStruct *S, int tid)
{
 int numb;
 char pc_instr[17] = {0x4,'0','0','1','1',0x2,'0','4','0','8'}; // set 04.08;
 int intgr, loop, array=11, sign=10;


 /* the following for loop converts the int to a char array***
  ****NOTE: the input value would never be greater than 1000***/
 intgr = (int) *u;              //debug to test input data

 if (intgr>0)
          {
            pc_instr[sign] = '+';
          }
 else
          {
            pc_instr[sign] = '-';
          }

 intgr=abs(intgr);

 for (loop=1000;loop>1;loop=loop/10)
 {
   if (loop>intgr)
   {
    pc_instr[array] = 0x30; //set bit to zero
   }
   else
   {
    pc_instr[array] = (intgr/loop) + 0x30;
   }
   intgr = intgr - (intgr/loop)*loop;
   array++;
 }
 pc_instr[array] =  (intgr%10) + 0x30;     // uses remainder function
 pc_instr[++array] =0x3;                   // ^C
 pc_instr[++array] = 0xd;                  // RETURN

 for (numb=0;numb <= array;numb++)         //create loop to o/p array
 {
    while ((inp(LSR)& 0x60) != 0x60);  //check for empty TXB
    outp(PORT, (int) pc_instr[numb]);
 }
 array =11;
 pc_instr[array] = '\0';
```

```
}

/*STEP 8 - PART 1
 * mdlUpdate - perform action at major integration time step
 *
 * This function is called once for every major integration time step.
 * Discrete states are typically updated here, but this function is useful
 * for performing any tasks that should only take place once per integration
 * step.
 */

static void mdlUpdate(double *x, double *u, SimStruct *S, int tid)
{
  /*
   * YOUR CODE GOES HERE
   */
}

/*STEP8 - PART 2
 * mdlDerivatives - compute the derivatives
 *
 * In this function, you compute the S-function block's derivatives.
 * The derivatives are placed in the dx variable.
 */

static void mdlDerivatives(double *dx, double *x, double *u, SimStruct *S, int tid)
{
  /*
   * YOUR CODE GOES HERE
   */
}

/* STEP9
 * mdlTerminate - called when the simulation is terminated.
 *
 * In this function, you should perform any actions that are necessary
 * at the termination of a simulation.  For example, if memory was allocated
 * in mdlInitializeConditions, this is the place to free it.
 */

static void mdlTerminate(SimStruct *S)
{
  /*
   * YOUR CODE GOES HERE
   */
}
// STEP10
#ifdef  MATLAB_MEX_FILE   /* Is this file being compiled as a MEX-file? */
#include "simulink.c"     /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"      /* Code generation registration function */
#endif
```

Appendix 5a

Calculation of Root Locus of a Transfer Function

The root locus of a transfer function can be calculated and plotted in Matlab using the command:

RLOCUS(num,den)

'num' and 'den' are the numerator and denominator of the transfer function, respectively. Fig. A5a.1 shows the root locus constructed using information provided on the Strathclyde WECS.

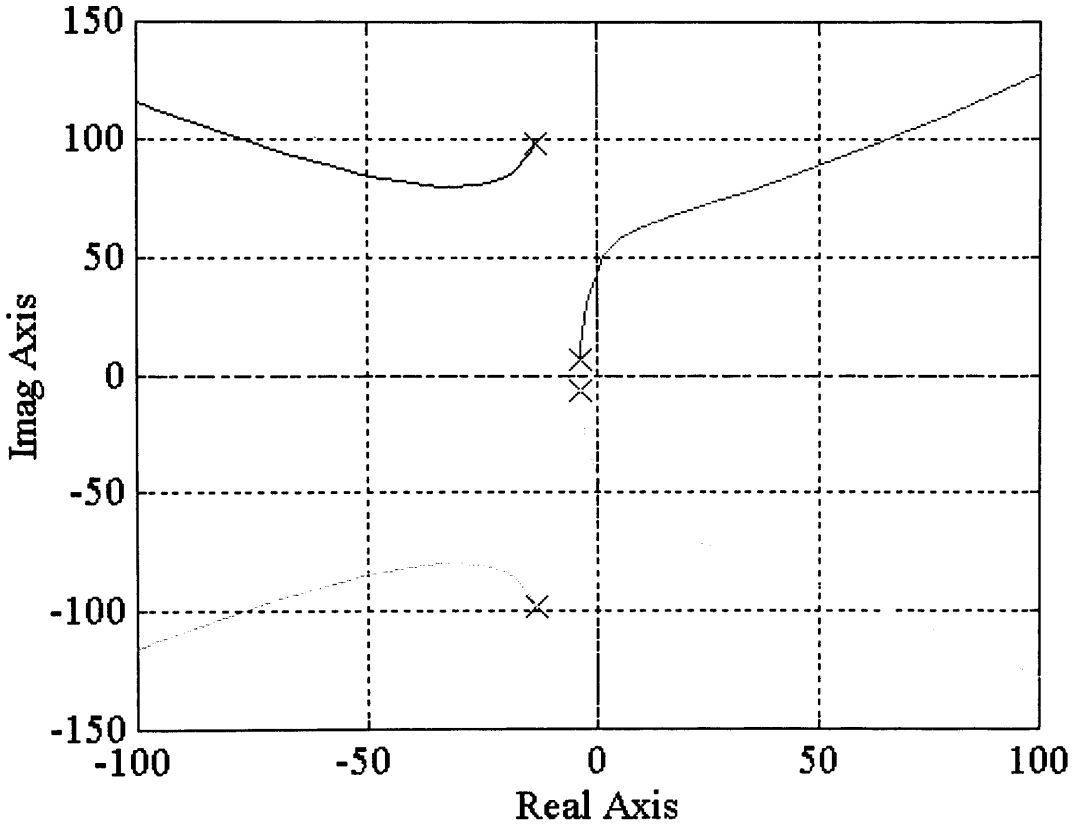


Fig. A5a.1 Root Locus of the Strathclyde WECS

Matlab Response to a 'linmod' Command

List A5a.1 shows the response of the 'linmod' command used to determine the state space representation of the Strathclyde Simulink model.

```
A =
        1.0e+004 *
         -0.0033   0.0241        0       0       0
         -0.0033   0.0000        0       0  0.0001
         0         0  0.0000  -0.0001        0
         0  -7.9560   0.0065        0       0
         0  -0.1957   0.0002        0       0

        B =
          0
          0
          1
          0
          0

        C =
         33.3333        0       0       0       0

        D =
        0
```

List A5a.1 Matlab State Space Model of the WECS Model

Matlab Response to a 'ss2tf' Command

List A5a.2 shows the response to 'ss2tf' for the Simulink based Strathclyde model

```
NUM =
    1.0e+004 *
    Columns 1 through 4
     0   0.00000000000000   0.00000000000000  -0.00000000000001
    Columns 5 through 6
      1.27827079519259  -0.00000000000009

    DEN =
      1.0e+005 *
     Columns 1 through 4
    0.00001000000000   0.00033414911924   0.10057941405578   0.67427690671342
    Columns 5 through 6
    5.19958980409994   0.00000000000002
```

List A5a.2 Matlab Transfer Function of the Simulink WECS Model

Matlab Program to Estimate Parameters of the RAL IM

```
xm=8.8095*3; % Parameters *3 to convert from 'star'
xls=0.4352*3;
xlr=0.4148*3;
rs=0.1073*3;
rr=0.0787*3;
u=0.02667; % u is the slip
ws=50*2*pi; % supply frequency
p=2;
a=(xls*xlr)+(xm*(xlr+xls));
k1=rs*(xlr+xm)/a;
k2=rs*xm/a;
k3=rr*(xls+xm)/a;
k4=rr*xm/a;
E=415*1.4142;
dem=(k1*k3-k2*k4)^2+k3^2-2*k2*k4*u+(1+k1^2)*u^2; %denominator for tg0 and De
tg0=(3/2)*p*k2*k4*u*E^2/(ws*rs*dem);
```

```
De=p*tg0*(k3^2+(k1*k3-k2*k4)^2-(1+k1^2)*u^2)/(ws*u*dem)
t=(k3+k1*(k1*k3-k2*k4))/(ws*(k3^2 +(k1*k3-k2*k4)^2-(1+k1^2)*u^2))
```

List 5.6 Matlab Program 'genral.m' for RAL Generator First Order Model

Appendix 5b

C_q-λ Values for the 45kW WEC

λ	C_q
1.3693	0.0142
1.3976	0.0145
1.4270	0.0143
1.4577	0.0147
1.4897	0.0149
1.5232	0.0154
1.5582	0.0158
1.5949	0.0159
1.6333	0.0161
1.6736	0.0165
1.7160	0.0172
1.7606	0.0180
1.8075	0.0190
1.8570	0.0201
1.9094	0.0214
1.9647	0.0224
2.0233	0.0233
2.0856	0.0248
2.1518	0.0276
2.2224	0.0291
2.2977	0.0312
2.3783	0.0349
2.4648	0.0381
2.5578	0.0442
2.6581	0.0505
2.7666	0.0551
2.8843	0.0614
3.0125	0.0684
3.1526	0.0747
3.3064	0.0815
3.4760	0.0870
3.6639	0.0914
3.8733	0.0944
4.1080	0.0955
4.3730	0.0977
4.6746	0.0910
5.0209	0.0899
5.4226	0.0827
5.8941	0.0622
6.4554	0.0402
7.1349	0.0297
7.9743	0.0219
9.0376	0.0092

C_p-λ Values for the 45kW WECS

λ	C_p
1.3693	0.0195
1.3976	0.0203
1.4270	0.0204
1.4577	0.0214
1.4897	0.0222
1.5232	0.0234
1.5582	0.0246
1.5949	0.0254
1.6333	0.0263
1.6736	0.0276
1.7160	0.0295
1.7606	0.0317
1.8075	0.0344
1.8570	0.0373
1.9094	0.0408
1.9647	0.0440
2.0233	0.0471
2.0856	0.0516
2.1518	0.0593
2.2224	0.0647
2.2977	0.0718
2.3783	0.0829
2.4648	0.0939
2.5578	0.1129
2.6581	0.1342
2.7666	0.1523
2.8843	0.1771
3.0125	0.2059
3.1526	0.2356
3.3064	0.2695
3.4760	0.3025
3.6639	0.3350
3.8733	0.3656
4.1080	0.3922
4.3730	0.4273
4.6746	0.4253
5.0209	0.4512
5.4226	0.4482
5.8941	0.3667
6.4554	0.2593
7.1349	0.2122
7.9743	0.1744
9.0376	0.0832

Appendix 6a

Equivalent Circuit Parameters of the 11kW IM

List A6a.1 shows the impedance parameters provided by the Manufacturer

Stator Resistance, rs	*0.695 Ω*
Stator Leakage Reactance, xls	*1.249 Ω*
Rotor Resistance (Referred to Stator), rr	*0.420 Ω*
Rotor Leakage Reactance (Referred to Stator), xlr	*1.977 Ω*
Magnetising Reactance, xm	*52.76 Ω*
Number of Pole Pairs, p	*1*
Peak Line Voltage, E	$415\sqrt{2}\ V$
Slip, u	*0.025*

List A6a.1 Equivalent Circuit Parameters of the 11kW IM

Calculating the Damping Factor of the 11kW IM

Using the same format of section 5.4.1(eii) where the losses of the generator are assumed to be the same, i.e. 2.24%, gives the following:

$$\gamma_2 = \frac{11\times10^3\times2.24\%}{314.16^2}$$

$$\gamma_2 = 0.002$$

Estimating the Relationship Between the General Purpose Register and the Motor Speed

Table A6a.1 shows the measurements of the general purpose register and the corresponding speed of the motor measured with an optical tachometer.

Register Value	Measured Speed (rpm)	Conversion Ratio
-315	500	1.5873
-365	580	1.5890
-433	687	1.5866
-628	997	1.5876
-828	1316	1.5894
-964	1535	1.5923

Table A6a.1 Comparison of Speed Measurement

The average conversion value is 1.5887.

DC Motor Model

Fig. A6a.1 shows the electrical and mechanical model of a DC Motor and load

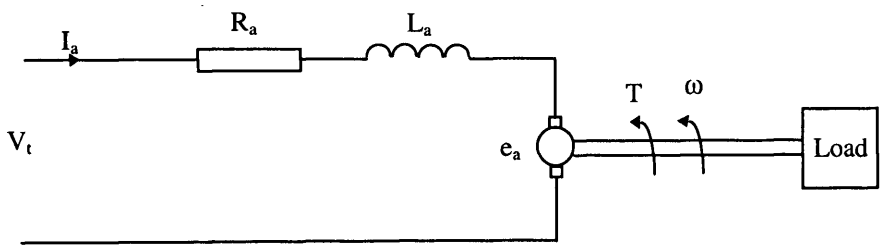


Fig A6a.1 DC Motor Model

The model assumes that the field current is constant.

From the model:

$$V_t = R_a I_a + L_a \frac{dI_a}{dt} + K\Phi\omega \qquad \text{(A6a.1)}$$

and

$$T = J\frac{d\omega}{dt} + D\omega + T_c \qquad \text{(A6a.2)}$$

where J is the inertia of the motor, D is the velocity friction coefficient and T_c is the constant load torque.

The machine parameters R_a and L_a, provided by Mawdsley are 0.52 Ω and 8.25mH respectively. Assume that K'= KΦ then at steady state:

$$K' = \frac{V_t - I_a 0.52}{\omega} \qquad \text{(A6a.3)}$$

and from Eqn. (A6a.2)

$$T = D\omega + T_c \qquad \text{(A6a.4)}$$

Estimating the DC Motor and Induction Machine Characteristics

The mechanical machine parameters of the motor can be estimated by measuring the armature voltage, current and the motor speed. Initially, the IM is disconnected from the motor so that the inertia of the motor and large pulley can be estimated. The IM and small pulley can then be reconnected and the inertia and losses of the combined hardware estimated. All calculations and measurements are taken with the regenerative braking enabled.

The drive has been set up for speed control with the field current, initially, fixed at 4 amps. Table A6a.2 shows the measurements and calculations

Speed (rpm)	V_t (Volts)	I_a (Amps)	K' (Nm/a)	T=(K'I_a) (Nm)
62.83	146	1.35	2.31	3.132
83.78	195	1.45	2.32	3.364
104.72	244	1.55	2.32	3.596
125.66	293	1.65	2.32	3.828
146.61	341	1.75	2.32	4.06
157.08	365	1.8	2.32	4.176
167.55	390	1.85	2.32	4.292

Table A6a.2 Measurements and Calculations of DC Motor Characteristics

The inertia of the motor and the large pulley can be estimated by measuring the rate of change of speed w.r.t. to the torque. At 157.08 rad/s T = 4.176 Nm if the armature supply is removed and with the field still connected, the time taken for the speed to fall 16.7 rad/s is 2.1 s. Therefore

$$\frac{d\omega}{dt} = \frac{16.7}{1.95} = 8.564$$

and the inertia, J can be estimated to be

$$J = \frac{T}{\frac{d\omega}{dt}} = \frac{4.176}{8.564} = 0.488 kgm^2$$

The IM and small pulley is reconnected to the DC motor and large pulley in order to estimate the combined inertia and losses of the two machines and the belt and pulley arrangement. This time the field current was set to lower value of 2 amps, as explained in Chapter 3 to avoid the effects of armature reaction. Also the drive was set for torque control in order to establish the relationship between the torque demand register (04.08)

and the measured armature current. Table A6a.3 shows the measured values and calculations

04.08	I_a (Amps)	V_t (Volts)	Speed (r/s)	K'(Nm/a)	Torq(Nm)
27	1.62	44	27.44	1.57	2.5434
29	1.74	60	37.7	1.57	2.7318
31	1.85	90	56.55	1.57	2.9045
34	2.03	125	78.54	1.58	3.1871
42	2.5	194	122.1	1.58	3.925
51	3.04	249	157.29	1.57	4.7728

Table A6a.3 Measurements and Calculations of DC Motor and IM

The relationship between the torque/current demand register, 04.08, and the armature current is shown to be linear and have the average gain of 16.7.

To calculate the constant losses and velocity friction coefficient consider the two cases where the armature current is 1.62 amps and 3.04 amps. From Eqn. (A6a.4) the measurements at these points can be used to form two simultaneous equations:

when $I_a = 1.62$ amps

$$27.44D + T_c = 2.543 \qquad \text{(A6a.5)}$$

when $I_a = 3.04$ amps

$$157.29D + T_c = 4.7728 \qquad \text{(A6a.6)}$$

$$157.29D + 2.543 - 27.44D = 4.7728$$

$$D = 0.017 \qquad \text{(A6a.7)}$$

from Eqn.(A6a.5)

$$T_c = 2.076Nm$$

from Eqn.(A6a.6)

$$T_c = 2.099Nm$$

Taking an average value for T_c gives the value 2.085Nm.

Calculating the inertia of the whole system using the same method as before. The 'rig' is run up to 157.29 rad/s and the armature supply removed. Measuring the response shows that a speed change of 14.9854 rad/sec occurs during 2.51 sec. Therefore

$$\frac{d\omega}{dt} = \frac{14.9854}{2.474} = 6.057$$

$$J = \frac{T}{\frac{d\omega}{dt}} = \frac{4.7728}{6.057} = 0.788kgm^2$$

Comparing the inertias of the combined system and the motor with the large pulley, the difference in inertia, which is the inertia of the generator and small pulley referred to the motor side, is:

$$0.788 - 0.488 = 0.3kgm^2$$

Referring this value to the generator side results in the generator inertia, J_g:

$$J_g = \frac{0.3}{N^2} = \frac{0.3}{2^2} = 0.075kgm^2$$

This value is used for simulation of the 11kW generator and small pulley in Simulink.

Appendix 6b

Simulating the Noise Content of the Speed Measurement from the Mentor II

Fig. A6b.1 shows the Simulink model used to simulate the noise inherent on the speed measurement from the drive.

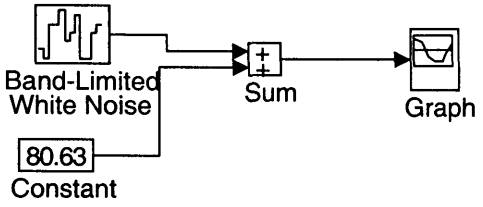


Fig. A6b.1 Noise Simulation Model

The gain of the 'BLW Noise' component can be varied and added to the constant value 80.63, which represents the expected speed measurement from the Mentor II.

Simulating the Measurement from the Mentor II

Fig. A6b.2 to Fig. A6b.4 show the response of the model to different gains in the 'BLW Noise' component. Each plot has the same axis as the speed measurement from the Mentor II (Fig. 6.3) for comparison purposes. Enlarging the plots (not shown here) show that the response of the noise model with the gain set at 0.005 is very similar to the speed measurement register of Fig. 6.3.

Similarly, the noise content of the signal from the optical speed encoder measurement is equivalent to a noise power gain of 0.004.

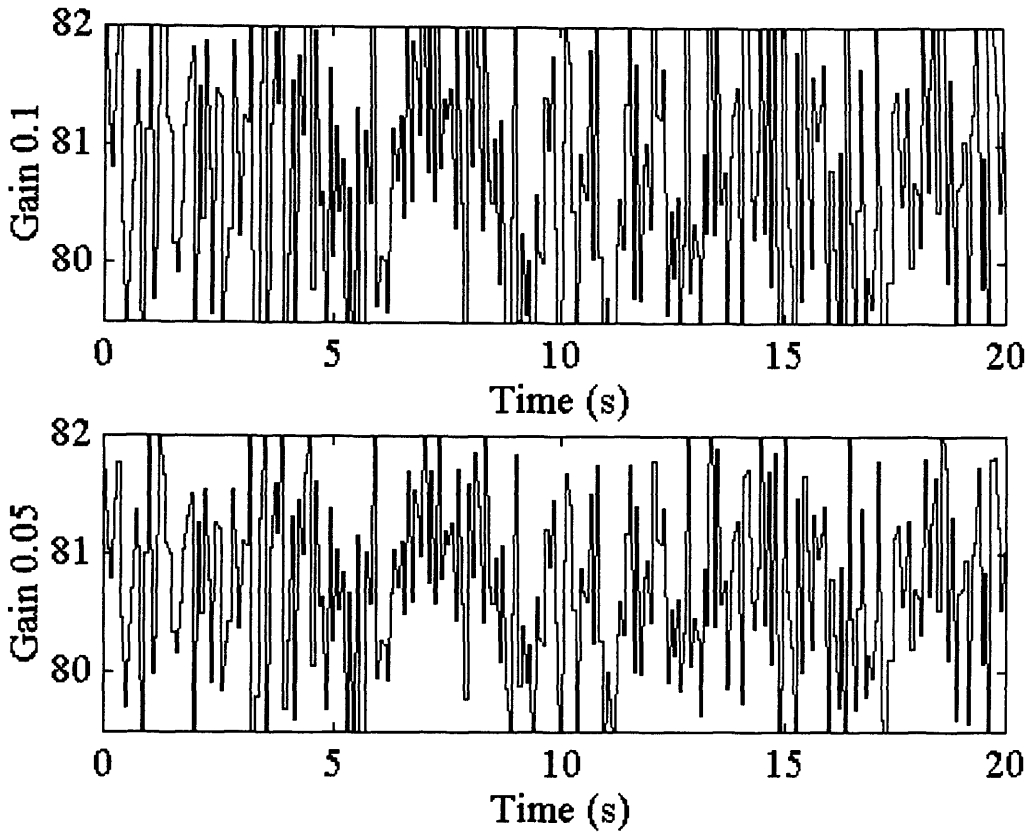


Fig. A6b.2 Response to Noise Gains of 0.1 and 0.05

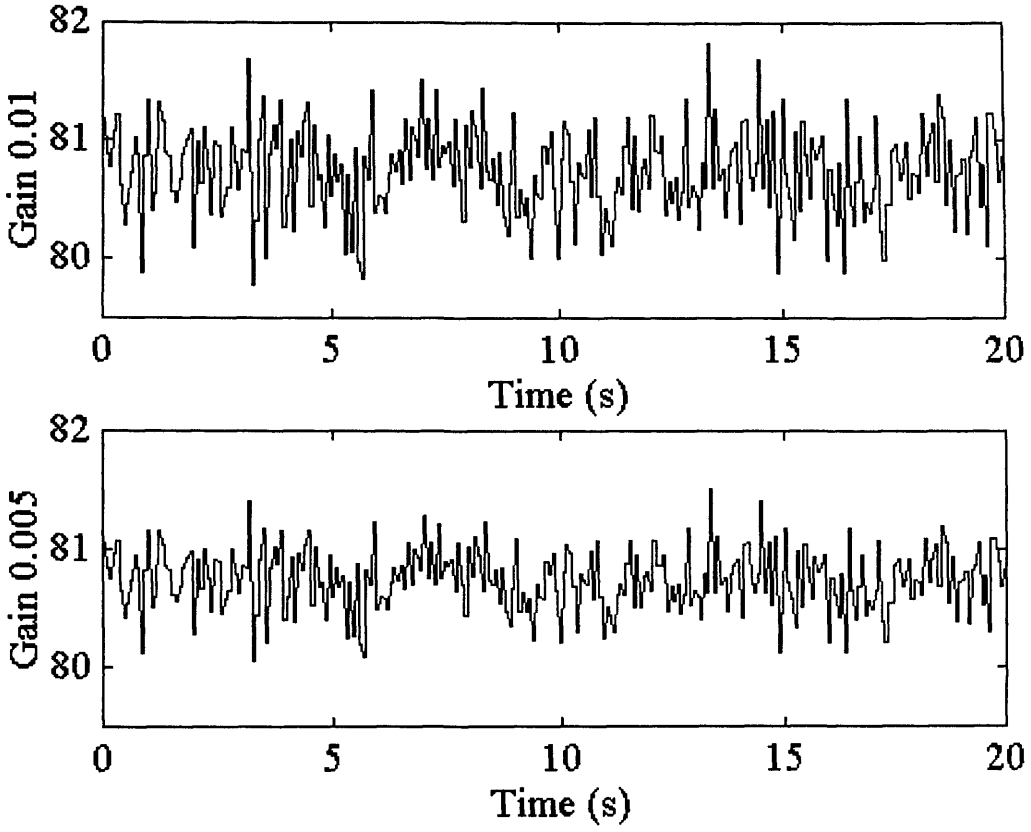


Fig. A6b.3 Response to Noise Gains 0.01 and 0.005

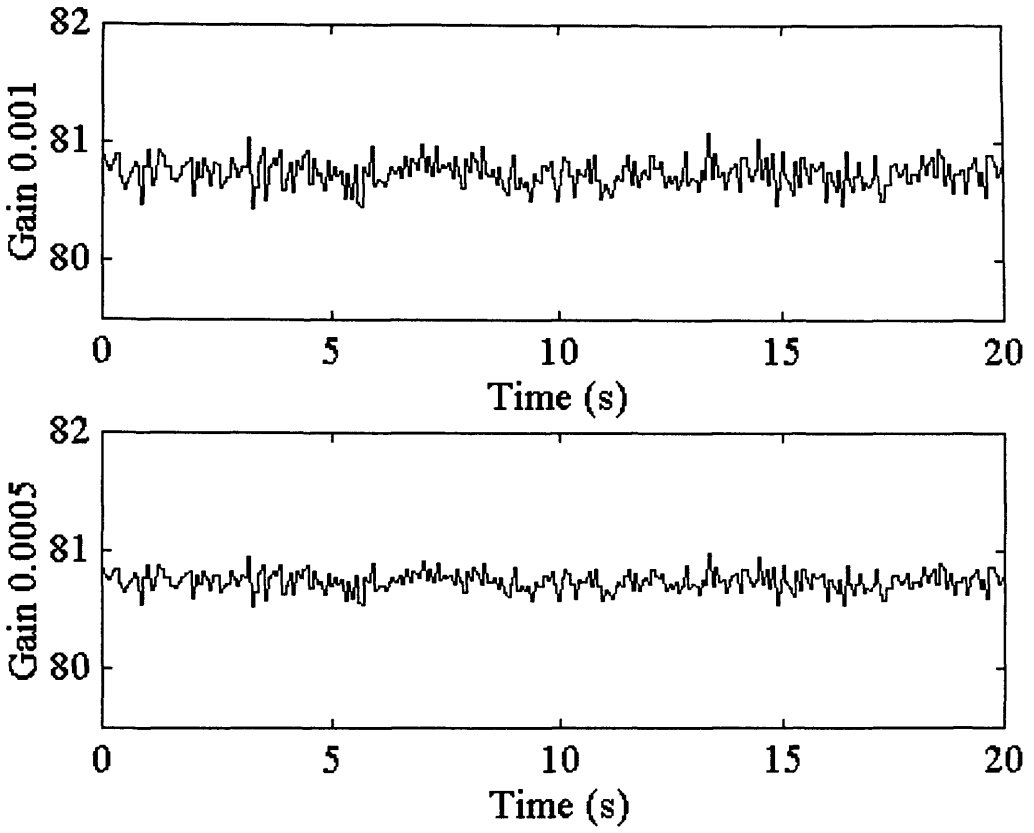


Fig. A6b.4 Response to Noise Gains 0.001 and 0.0005

Determining the Noise Gain Limit for Stable Operation of WECS HILS

Variable gain values are used to establish the value of noise at which the WECS model, incorporating the HILS, becomes unstable. Once this value is established, any steady speed measurement can be assessed for both its noise content and whether it can be used for HILS. The module used for determining the influence of HILS is shown in Fig. 6.4. Fig. A6b.5 and Fig. A6b.6 shows the time response of the simulation associated with varying noise gain values. The limit of stability occurs when the noise gain is 0.0007. Therefore, the noise of the speed measurement must be lower than this value.

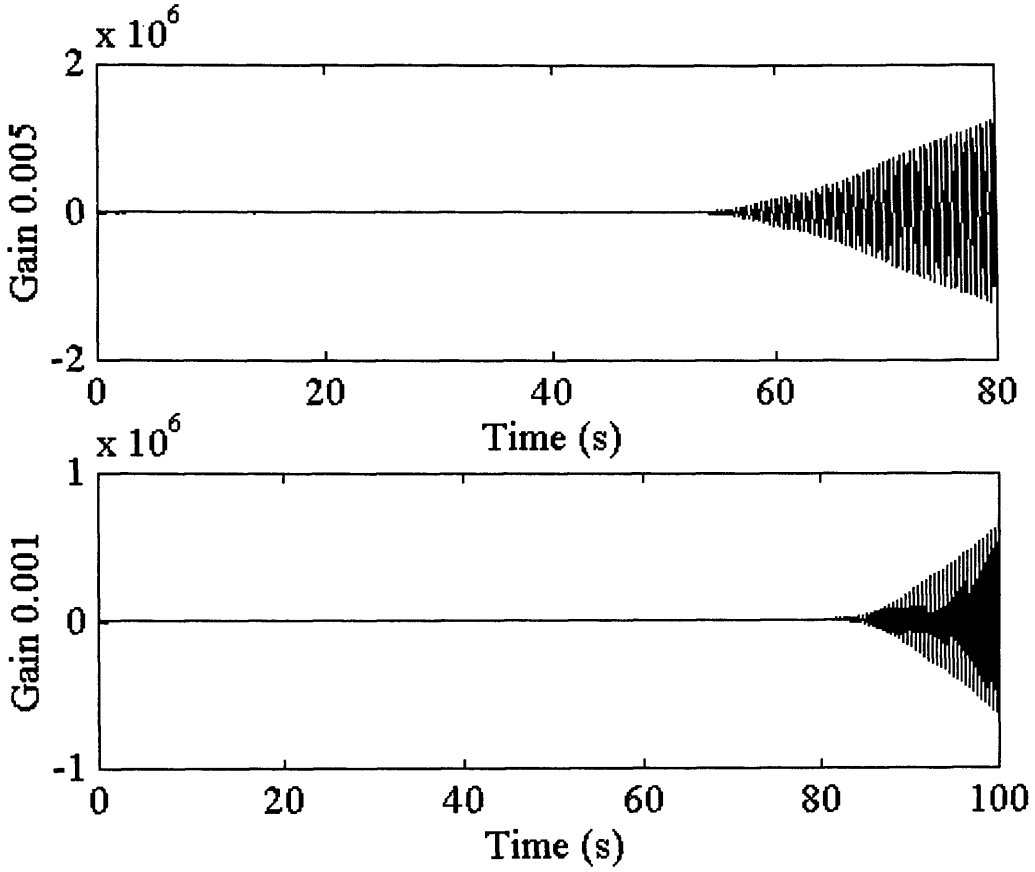


Fig. A6b.5 Response of Model to Noise Gains of 0.005 and 0.001

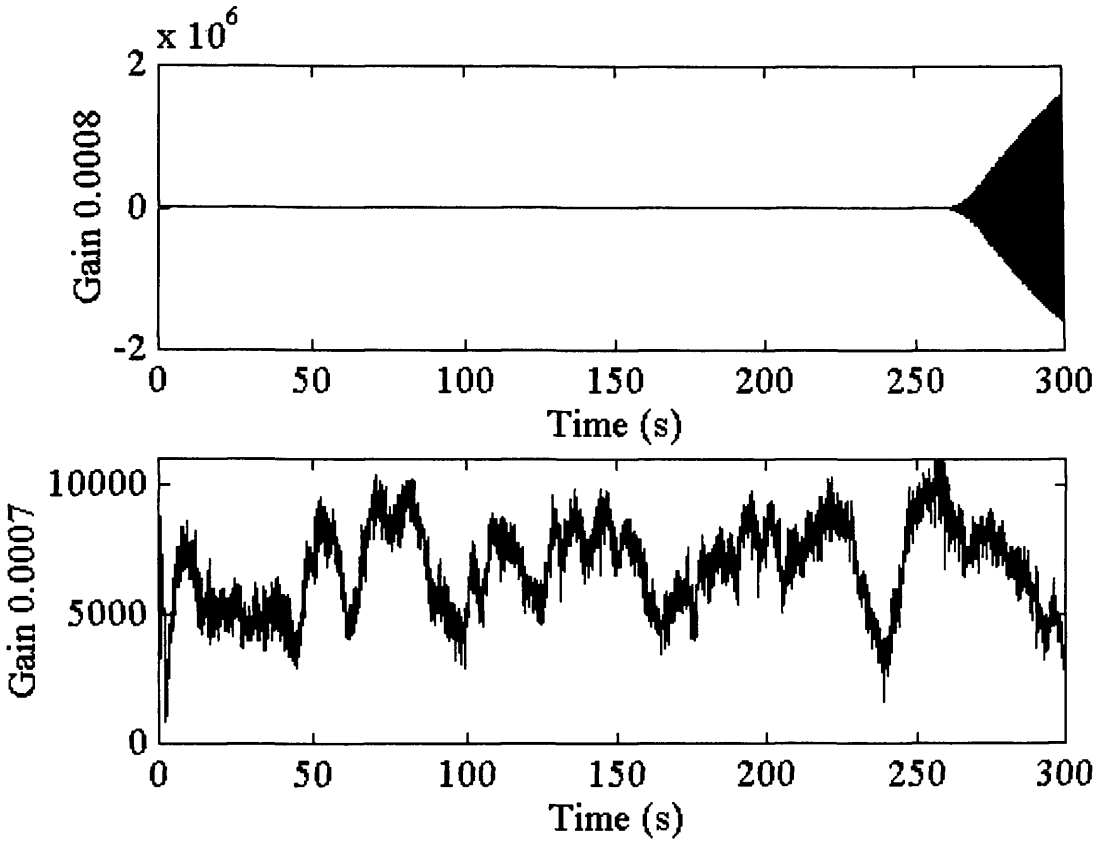


Fig. A6b.6 Response of Model to Noise Gains of 0.0008 and 0.0007

Reducing the Noise Power Further

Once the limit of stability is calculated it is necessary to assess the improvement of the PSD comparison with the reduction in noise gain. Fig. 6.6 and Fig. 6.10 show the PSD comparison for gains of 0.0007 and 0.0005 respectively. Fig. A6b.7 and Fig. A6b.8 show how the PSD comparison is improved, in particular the blade passing frequency, with further reduction in noise gain.

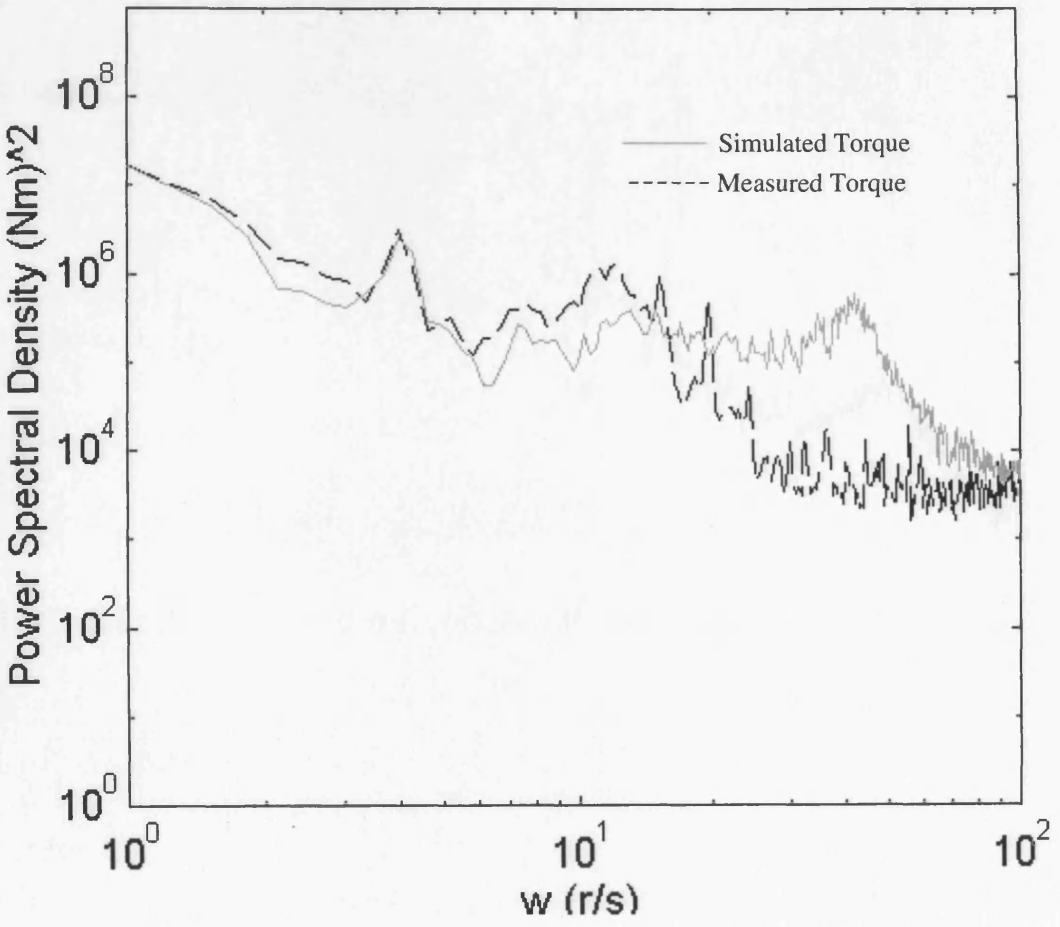


Fig. A6b.7 PSD Comparison with Noise Gain 0.0003

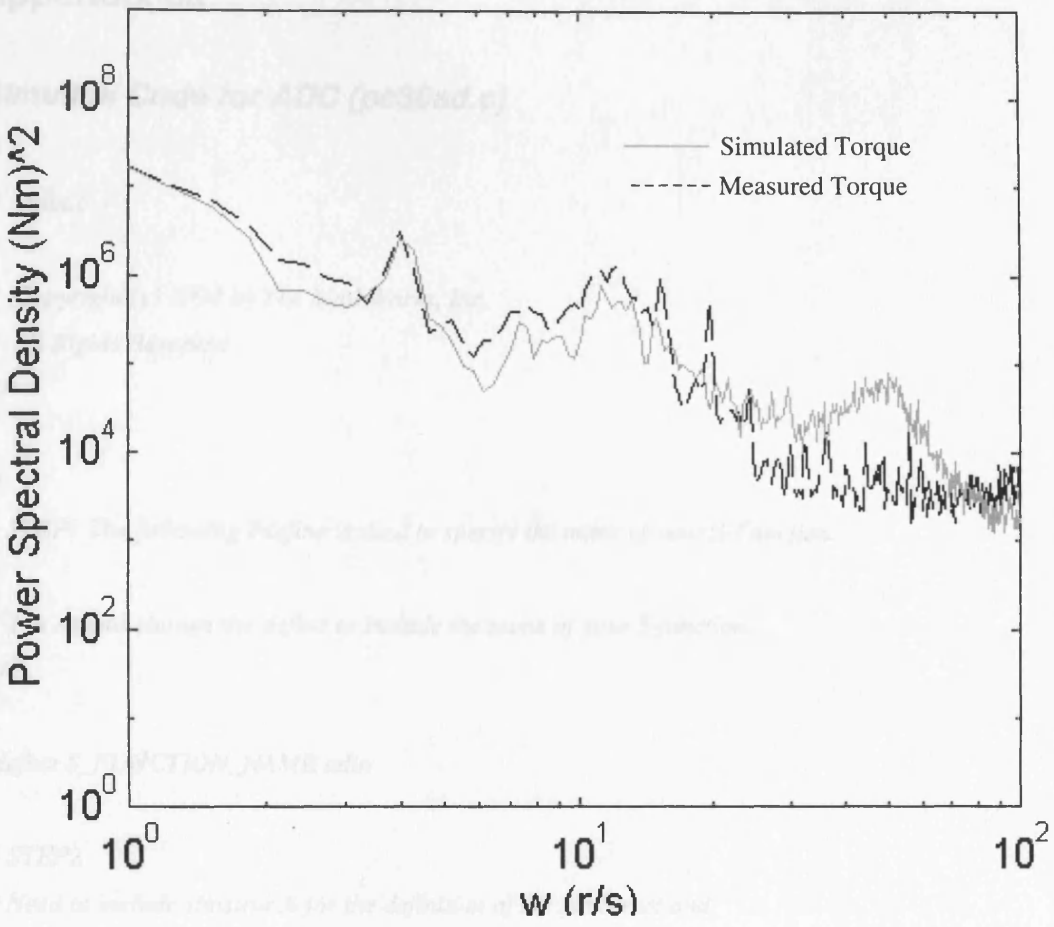


Fig. A6b.8 PSD Comparison with Noise Gain 0.00005

Appendix 8a

Simulink Code for ADC (pc30ad.c)

```
/*
 *  adin.c
 *
 *  Copyright (c) 1994 by The MathWorks, Inc.
 *  All Rights Reserved
 */

/*
 * STEP1 The following #define is used to specify the name of your S-Function.
 *
 * You should change the define to include the name of your S-function.
 */

#define S_FUNCTION_NAME adin

/* STEP2
 * Need to include simstruc.h for the definition of the SimStruct and
 * its associated macro definitions.
 */

#include "simstruc.h"

/*STEP3
 *Include file for MEX file
*/

#ifdef MATLAB_MEX_FILE
#include "mex.h"
#endif

#include <conio.h>      //for outp & inp
//#include <iostream.h>   //for cout
```

```
#define blkcnt 0x700
#define adccr 0x702
#define admde 0x703
#define gmemo 0x718
#define adccfg 0x71c
#define addsr 0x701
#define addatl 0x700


/* STEP4
 * mdlInitializeSizes - initialize the sizes array
 *
 * The sizes array is used by SIMULINK to determine the S-function block's
 * characteristics (number of inputs, outputs, states, etc.).
 */

static void mdlInitializeSizes(SimStruct *S)
{
   ssSetNumContStates(    S, 0);     /* number of continuous states */
   ssSetNumDiscStates(    S, 0);     /* number of discrete states */
   ssSetNumInputs(        S, 0);     /* number of inputs */
   ssSetNumOutputs(       S, 1);     /* number of outputs */
   ssSetDirectFeedThrough(S, 0);     /* direct feedthrough flag */
   ssSetNumSampleTimes(   S, 1);     /* number of sample times */
   ssSetNumInputArgs(     S, 1);     /* number of input arguments */
   ssSetNumRWork(         S, 0);     /* number of real work vector elements */
   ssSetNumIWork(         S, 0);     /* number of integer work vector elements */
   ssSetNumPWork(         S, 0);     /* number of pointer work vector elements */
}

/*STEP5
 * mdlInitializeSampleTimes - initialize the sample times array
 *
 * This function is used to specify the sample time(s) for your S-function.
 * If your S-function is continuous, you must specify a sample time of 0.0.
 * Sample times must be registered in ascending order.  If your S-function
```

```
 * is to acquire the sample time of the block that is driving it, you must
 * specify the sample time to be INHERITED_SAMPLE_TIME.
 */

static void mdlInitializeSampleTimes(SimStruct *S)
{
   ssSetSampleTimeEvent(S, 0, mxGetPr(ssGetArg(S,0))[0]);
   ssSetOffsetTimeEvent(S, 0, 0.0);

   /*
    * SET OTHER SAMPLE TIMES AND OFFSETS HERE
    */
}

/*STEP6
 * mdlInitializeConditions - initialize the states
 *
 * In this function, you should initialize the continuous and discrete
 * states for your S-function block.  The initial states are placed
 * in the x0 variable.  You can also perform any other initialization
 * activities that your S-function may require.
 */

static void mdlInitializeConditions(double *x0, SimStruct *S)
{
        /*Initialisation*/
      outp(blkcnt, 0xff);      //1 pulse per conv
      outp(admde, 0x96);       //clr list, set ch0 as only ch
      outp(adccr, 0x02);       //ch0, dis int,set s/w strobe, clr str bit
      outp(gmemo, 0x00);       //gain of ch = 1
      outp(adccfg, 0x03);      //no dac inv, no int, diff mode, +-10v


}

/*
```

```
 * mdlOutputs - compute the outputs
 *
 * In this function, you compute the outputs of your S-function
 * block.  The outputs are placed in the y variable.
 */

static void mdlOutputs(double *y, double *x, double *u, SimStruct *S, int tid)
{
        int d07, d811, d011;
        outp(adccr,0x03);      //start of strobe
      outp(adccr,0x02);      //strobe
      while ((inp(addsr)& 0x40) != 0x40);      //poll bit 6 for eoc
      d811=(inp(addsr)& 0x0f);      //mask out control data
      d07=inp(addatl);            //get data 0 to 7
      d011= d811<<8;              //shift left 8bits
      d011=d011+d07;              //combine values
      *y=(d011-2048)*1000/2048;

}

/*STEP 8 - PART 1
 * mdlUpdate - perform action at major integration time step
 *
 * This function is called once for every major integration time step.
 * Discrete states are typically updated here, but this function is useful
 * for performing any tasks that should only take place once per integration
 * step.
 */

static void mdlUpdate(double *x, double *u, SimStruct *S, int tid)
{
   /*
    * YOUR CODE GOES HERE
    */
}
```

```
/*STEP8 - PART 2
 * mdlDerivatives - compute the derivatives
 *
 * In this function, you compute the S-function block's derivatives.
 * The derivatives are placed in the dx variable.
 */

static void mdlDerivatives(double *dx, double *x, double *u, SimStruct *S, int tid)
{
   /*
    * YOUR CODE GOES HERE
    */
}

/* STEP9
 * mdlTerminate - called when the simulation is terminated.
 *
 * In this function, you should perform any actions that are necessary
 * at the termination of a simulation.  For example, if memory was allocated
 * in mdlInitializeConditions, this is the place to free it.
 */

static void mdlTerminate(SimStruct *S)
{
   /*
    * YOUR CODE GOES HERE
    */
}
// STEP10
#ifdef    MATLAB_MEX_FILE    /* Is this file being compiled as a MEX-file? */
#include "simulink.c"      /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"       /* Code generation registration function */
#endif
```

Simulink Code for DAC (pc30da.c)

```
/*
 * daout.c
 *
 * Copyright (c) 1994 by The MathWorks, Inc.
 * All Rights Reserved
 */

/*
 * STEP1 The following #define is used to specify the name of your S-Function.
 *
 * You should change the define to include the name of your S-function.
 */

#define S_FUNCTION_NAME daout

/* STEP2
 * Need to include simstruc.h for the definition of the SimStruct and
 * its associated macro definitions.
 */

#include "simstruc.h"

/*STEP3
 *Include file for MEX file
 */

#ifdef MATLAB_MEX_FILE
#include "mex.h"
#endif

#include <conio.h>     //for outp & inp
#define blkcnt 0x700
#define adccr 0x702
#define admde 0x703
#define gmemo 0x718
```

```
#define adccfg 0x71c
#define addsr 0x701
#define addatl 0x700
#define daccfg 0x71d
#define dadatho 0x70d
#define dadatlo 0x70c


/* STEP4
 * mdlInitializeSizes - initialize the sizes array
 *
 * The sizes array is used by SIMULINK to determine the S-function block's
 * characteristics (number of inputs, outputs, states, etc.).
 */

static void mdlInitializeSizes(SimStruct *S)
{
   ssSetNumContStates(    S, 0);     /* number of continuous states */
   ssSetNumDiscStates(    S, 0);     /* number of discrete states */
   ssSetNumInputs(        S, 1);     /* number of inputs */
   ssSetNumOutputs(       S, 0);     /* number of outputs */
   ssSetDirectFeedThrough(S, 1);     /* direct feedthrough flag */
   ssSetNumSampleTimes(   S, 1);     /* number of sample times */
   ssSetNumInputArgs(     S, 1);     /* number of input arguments */
   ssSetNumRWork(         S, 0);     /* number of real work vector elements */
   ssSetNumIWork(         S, 0);     /* number of integer work vector elements */
   ssSetNumPWork(         S, 0);     /* number of pointer work vector elements */
}

/*STEP5
 * mdlInitializeSampleTimes - initialize the sample times array
 *
 * This function is used to specify the sample time(s) for your S-function.
 * If your S-function is continuous, you must specify a sample time of 0.0.
 * Sample times must be registered in ascending order.  If your S-function
 * is to acquire the sample time of the block that is driving it, you must
```

```
 * specify the sample time to be INHERITED_SAMPLE_TIME.
 */

static void mdlInitializeSampleTimes(SimStruct *S)
{
   ssSetSampleTimeEvent(S, 0, mxGetPr(ssGetArg(S,0))[0]);
   ssSetOffsetTimeEvent(S, 0, 0.0);

   /*
    * SET OTHER SAMPLE TIMES AND OFFSETS HERE
    */
}

/*STEP6
 * mdlInitializeConditions - initialize the states
 *
 * In this function, you should initialize the continuous and discrete
 * states for your S-function block.  The initial states are placed
 * in the x0 variable.  You can also perform any other initialization
 * activities that your S-function may require.
 */

static void mdlInitializeConditions(double *x0, SimStruct *S)
{
          /*Initialisation*/
          outp(blkcnt, 0xff);       //1 pulse per conv
      outp(admde, 0x96);      //clr list, set ch0 as only ch
      outp(adccr, 0x02);      //ch0, dis int,set s/w strobe, clr str bit
      outp(gmemo, 0x00);       //gain of ch = 1
      outp(adccfg, 0x03);       //no dac inv, no int, diff mode, +-10v
      outp(daccfg, 0x88);       //dac +-10v


}

/*
```

```
 * mdlOutputs - compute the outputs
 *
 * In this function, you compute the outputs of your S-function
 * block.  The outputs are placed in the y variable.
 */

static void mdlOutputs(double *y, double *x, double *u, SimStruct *S, int tid)
{
 int hbyte,lbyte, code;

        code =((*u)*2048/1000)+2047;      //convert voltage
     hbyte = (code & 0xff0) >>4;    //mask and shift
     lbyte = (code & 0xf) <<4;      //mask and shift
     outp(dadatho,hbyte);
     outp(dadatlo,lbyte);
}

/*STEP 8 - PART 1
 * mdlUpdate - perform action at major integration time step
 *
 * This function is called once for every major integration time step.
 * Discrete states are typically updated here, but this function is useful
 * for performing any tasks that should only take place once per integration
 * step.
 */

static void mdlUpdate(double *x, double *u, SimStruct *S, int tid)
{
   /*
    * YOUR CODE GOES HERE
    */
}

/*STEP8 - PART 2
 * mdlDerivatives - compute the derivatives
 *
```

```
 * In this function, you compute the S-function block's derivatives.
 * The derivatives are placed in the dx variable.
 */

static void mdlDerivatives(double *dx, double *x, double *u, SimStruct *S, int tid)
{
  /*
   * YOUR CODE GOES HERE
   */
}

/* STEP9
 * mdlTerminate - called when the simulation is terminated.
 *
 * In this function, you should perform any actions that are necessary
 * at the termination of a simulation.  For example, if memory was allocated
 * in mdlInitializeConditions, this is the place to free it.
 */

static void mdlTerminate(SimStruct *S)
{
  /*
   * YOUR CODE GOES HERE
   */
}
// STEP10
#ifdef    MATLAB_MEX_FILE   /* Is this file being compiled as a MEX-file? */
#include "simulink.c"     /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"      /* Code generation registration function */
#endif
```

Appendix 9a

Table 9a.1 shows typical roughness values for various pipe materials and conditions [9.1].

Material	Age/ Condition		
	Good (<5 years)	Normal (5-15 years)	Poor (>15 years)
Smooth Pipes (PVC, HDPE, MDPE, Glass fibre)	0.003	0.01	0.05
Concrete	0.06	0.15	1.5
Mild Steel			
-Uncoated	0.01	0.1	0.5
- Galvanised	0.06	0.15	0.3
Cast Iron			
New	0.15	0.3	0.6
Old - Slight Corrosion	0.6	1.5	3.0
- Moderate Corrosion	1.5	3.0	6.0
- Severe Corrosion	6.0	10.0	20.0

Table 9a.1 Pipe Roughness Values (mm)

Fig. 9a.1 shows the Moody Chart which is used to establish the friction factor, f. Q is the flow rate, d is the internal diameter of the pipe, k is the pipe roughness [9.1].

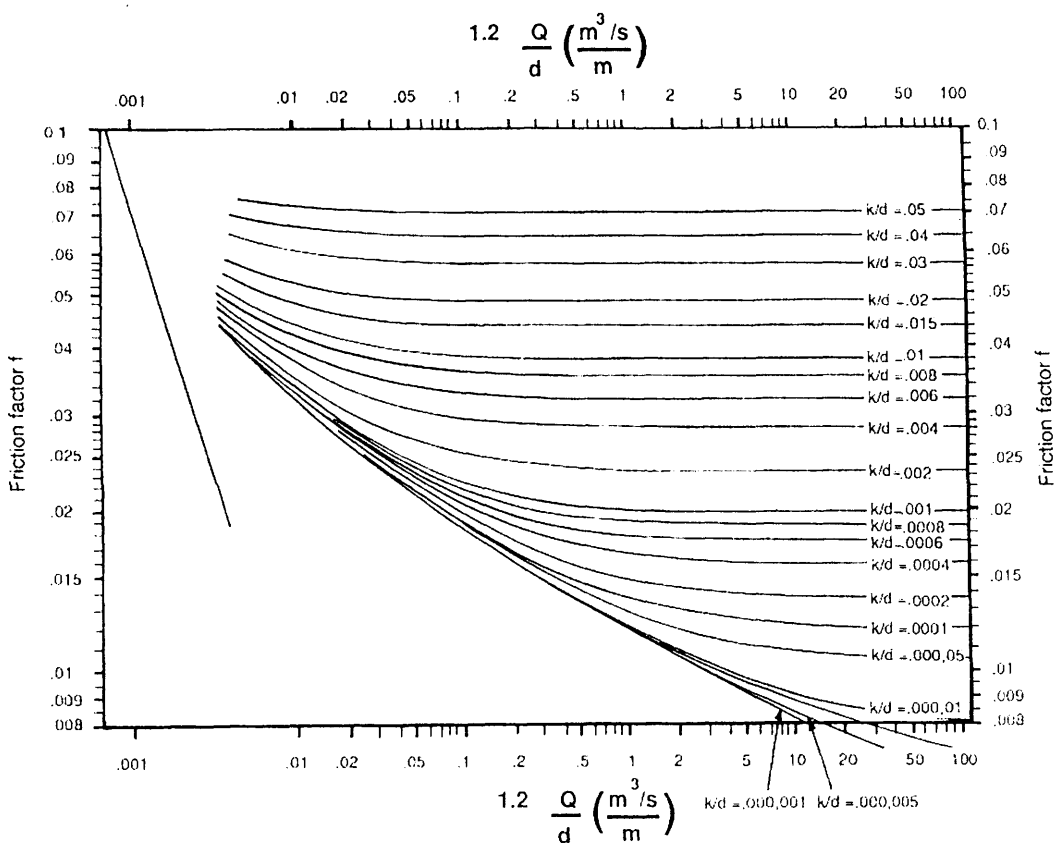


Fig. 9a.1 The Moody Chart

Appendix 9b

Estimating the 'Front-End' Parameters for an 11kW MHP

To ensure that the 11kW IM operates near to rated, the 'front-end' parameters of the MHP have to be estimated to allow for losses in the penstock and turbine. The simulator includes a model of the drive train so dynamics in this section will be accounted for.

The first assumption to be made is that a Pelton water wheel will be used as the turbine. References state that the Pelton has an efficiency of 75% and assuming that the losses in the drive-train are negligible, compared to those of the turbine, the mechanical power, P_m, at the turbine will be [9.1]:

$$P_m = \frac{11kW}{0.75} = 14.67kW$$

Assuming that the net head will be 30m, the flow rate, Q, can be estimated from Eqn. (9.3)

$$Q = \frac{P_m}{\rho g h_{net}}$$

$$Q = \frac{14.67kw}{1000 \times 9.81 \times 30} = 0.05 \frac{m^3}{s}$$

In order to estimate the friction factor, the roughness value, k and internal diameter, d, have to be assumed. Assuming that the penstock is galvanised, mild steel and 5-15 years old, k is 0.00015m while the diameter of the pipe is assumed to be 0.2m [9.1]. Appendix 9a includes the Moody Chart necessary for estimating f. First the following have to be calculated:

$$1.2\frac{Q}{d} = 1.2 \times \frac{0.05}{0.2} = 0.3$$

$$\frac{k}{d} = \frac{0.00015}{0.2} = 0.00075$$

These co-ordinates on the Moody Chart correspond to value of f = 0.024. Assuming a pipe length of 35m, the pipe wall losses can be calculated from Eqn. (9.4):

$$h_{wallloss} = \frac{0.024 \times 35 \times 0.08 \times (0.05)^2}{(0.2)^5} = 0.525m$$

The velocity, v, of the water in the pipe, required for calculating the turbulence losses, can be calculated from Eqn. (9.5)

$$v = \frac{4 \times 0.05}{\pi \times (0.2)^5} = 1.59 \frac{m}{s}$$

Assuming that the pipe has a head loss co-efficient, K, of 0.5, the turbulence losses can be calculated from Eqn. (9.6)

$$h_{turbloss} = \frac{(1.59)^2}{2 \times 9.81} 0.5 = 0.06m$$

The total head losses are:

$$h_{loss} = 0.525 + 0.06 = 0.585m$$

The gross head will therefore, be 30.585m to ensure a net of 30m This corresponds to a loss of 2% and compares to loss measurements in the references [9.1].

Bibliography

'CAMO:Cypros ESIM, User's Manual, Version 3.' CAMO A/S, Jarleveien 4, N-7041 Trondheim, Norway, 1991

'Final Report for CEC Contract JOUR-0078. Volume 3 - Jodymod Dynamic Simulation Software Package: Model Description', RAL-94-003

'Final Report for CEC Contract JOUR-0078. Volume 3 - Jodymod Dynamic Simulation Software Package: User's Guide', RAL-94-005

'Wind Turbine Engineering Design' Eggleston D M, Stoddard F S, VNR 1987

'Principles of Electromechanical-Energy Conversion' Meisel J, McGraw-Hill 1966

'Hughes Electrical Technology' McKenzie Smith I, Longman Scientific & Technical 7th Edition 1995

'Structural Dynamics, Stability and Control of High Aspect Ratio Wind Turbine Generators' Stoddard F S, University of Massachusetts 1979

''Front-End' Aerodynamic Control of Horizontal Axis Wind Turbines- Final Report' Anderson C G, Campbell T J, ETSU WN 6037 1993

'Modeling [sic] and Control of Variable-Speed Wind-Turbine Drive-System Dynamics' Novak P, Ekelund T, Jovik I, Schmidtbauer B, IEEE Control Systems August 1995

'Modelling and Output Power Optimisation of a Wind Turbine Driven Double Output Induction Generator' Uctug M Y, Eskanderzadeh I, Ince H, IEE Proc.-Electr. Power Appl., Vol. 141 No.2, March 1994

'Comparison of Wind Tunnel Airfoil Performance Data with Wind Turbine Blade Data', Butterfield C P, Scott G, Musial W, Journal of Solar Energy Engineering, May 1992, Vol. 114

'Theoretical and Experimental results from the Operation of Riso's Experimental Wind/Diesel System', Lundsager P, Hauge Madsen P, Aagraard Madsen H, European Wind Energy Association Conference and Exhibition 7-9 October 1986

'A Short Term Dynamic Simulation Model for Wind/Diesel Systems' Uglen K, Oyvin S, BWEA Conference 1988

'Simulation of a Radial MV/LV Electricity Grid Voltage Associated with Wind Power Induction Generator', Rakkolainen J, Vilkko M, Lautala P, BWEA Conference 1995

'Development of Dynamic Models for no Storage Wind/Diesel Systems', Jeffries W Q, McGowan J G, Manwell J F, BWEA Conference 1995

'Dynamic Modelling of a Wind-Diesel System for Analysis and Design', Asbach-Cullen R C, Freris L L, BWEA Conference 1995

'Improvement in Performance of a Passive Pitch Wind Turbine with Variable Speed Operation' Bleijs J A M, EWEC 1994

'A Norwegian Wind/Diesel Autonomous System', Toftevaag T, Uhlen K, Skarstein O, Wigren J, Wind Energy 1989

'The Integration of a Wind Turbine and Hydraulic Accumulator Energy Store with a Diesel Generator to Supply Electricity in a Remote Location', Slack G, University of Reading 1985

'Harmonic Analysis of an Electric Utility System with a High Penetration Level of Wind Generation', Tang L, Zavadil R M

'Development of a Wind Turbine Systems Dynamics Model Using the Automatic Dynamic Analysis of Mechanical Systems (ADAMS) Software', Wright A D, Buhl M L, Elliot A S

'Wind Power Modeling and Application in Generating Adequacy Assessment', Billinton R, Gan L, IEEE Proc. 1993

'Energy Conversion Problems in a Wave Power Station', Beattie W C, UPEC 1993

'Computer Modelling of the Islay Wave Power Generator' Linden B M, Beattie W C, Whitmaker T J T, UPEC 1993

'Dynamic Control Options for Variable Speed Wind Turbines', Iqbal M T, Coonick A H, Freris L L, Wind Engineering, Vol. 18, No. 1

'Dynamic Response of Offshore Wind Turbines', Wastling M A, Quarton D C, Schellin T E, Wind Engineering, Vol. 17, No. 5

'A Dynamic Model of the Influence of Turbulence on the Power Output of a Wind Turbine', Sheinman Y, Rosen A, Journal of Wind Engineering and Industrial Aerodynamics, Vol. 39, 1992

'Software Engineering with C++ and Case Tools', Pont MJ, Addison-Wesley Publishing Company, 1996

'Engineering Mathematics', Stroud KA, MacMillan Education, 1987, 3rd Edition

'Further Engineering Mathematics', Stroud KA, MacMillan Education, 1987

'Solutions of Problems in Advanced Electrical Engineering', Atkinson GH, Stevens RA, Sir Isaac Pitman & Sons Ltd. 1967

'Modern Control Engineering', Ogata K, Prentice Hall International, Inc. 1990, 2nd Edition

'Computer Modelling of Electrical Power Systems', Arrillaga J, Arnold CP, Harker BJ, John Wiley & Sons, 1991

'Electric Circuits', Nilsson JW, Addison-Wesley Publishing Company, 1983

'System Modelling and Control', Schwarzenbach J, Gill KF, Edward Arnold, 1988

'Engineering Mechanics - Volume 2 Dynamics', Meriam JL, Kraige LG, John Wiley & Sons, 1987, 2nd Edition

'Large-Eddy Simulation: A Critical Review of the Technique', Mason PJ, Quarter Journal of the Royal Meteor Society, pp120-126, 1994

'Some Aspects of Small Aerogenerator Design and Testing', Buehring IK, Freris LL, Third International Symposium on Wind Energy Systems, 1980

'Flexible Wind Turbine Model Validation', van Baars GE, Bongers PMM, Wind Engineering, Vol. 16, No. 4, 1992

'Calculations of Dynamic Wind Turbine Blade Loads from Simple Meteorological Data', Smedman AS, Wind Engineering, Vol. 16, No. 4, 1992

'Torque Measurements Using Strain Gauges', http://apo.mech.nwu.edu/mbrown-lib/Master...is/Section4/Section4.2.1/master4.2.2.html

'Torque Measuring Platform', Personal Correspondence with Mullins M, Arrow Engineering Designs, May 1997

'Electric Machines and Drives', Slemon GR, Addison-Wesley Publishing, 1992

'NMAKE.wri File', Documentation for the NMAKE Utility for Microsoft Visual C++, Version 1.0, 1993

'I/O Port Access in Microsoft Visual C++',
http://www.doc.ac.uk/~ih/doc/par/doc/data/vc.html

'How to Change COM1 or COM2 Parameters While Port is Open',
http://emwac.faf.cuni.cz/mir...velopr/BASIC/KB/Q39/2/55.txt

'The Role of Converters & Their Control in the Recovery of Wave Energy', Childs JF, IEE, 1997

'Flood & Coastal Defence', No. 10, June 1997, Ministry of Agriculture, Fisheries and Food

'Design Charts for Wells Air Turbine', Kotb MA, Wind Engineering, Vol. 20, No. 4, 1996

'New Approach for Simulating an Energy Limited Hydro Unit', Ahsan Q, Bhuiyan MR, IEE Proceedings, Vol. 137, Pt. C, No. 5, 1990

'A Wind/Diesel System with Variable Speed Flywheel Storage', Ruddell AJ, Bleijs JAM, Freris LL, Infield DG, Smith GA, Wind Engineering, Vol. 17, No. 3, 1993

'Interconnection Issues Concerning Consumer-Owned Wind Electric Generators', Park GL, Zastrow OW, IEEE Trans. on Power Apparatus and Systems, Vol. Pas-101, No. 7, 1982

'High Quality Mains Power from Variable-Speed Wind Turbines', Jones R, Smith GA, Wind Engineering, Vol. 18, No. 1

'Propagation and Elimination of Torque Ripple in a Wind Energy Conversion System', Dessaint L, Nakra H, Mukhedkar D, IEEE Trans. on Energy Conversion, Vol. EC-1, No. 2, 1986

'On Wind Turbine Power Measurements', Frandsen S, Christensen CJ, International Symposium on Wind Energy Systems, 1980

'Autonomous Wind Energy Conversion Systems with a Simple Controller for Maximum-Power Transfer', Ermis M, Ertan HB, Akpinar E, Ulgut F, IEE Proceedings-B, Vol. 139. No. 5, 1992

'Control Strategies for Variable-Speed Wind Energy recovery', Goodfellow D, Smith GA, Gardner G, BWEC 1988

'Some Control Aspects of a Small Isolated Wind Turbine', Iqbal MT, Coonick AH, Freris LL

'Amplicon Liveline Catalogue', 1997, Vol. 2

'User Manual for the PC30F and PC30G Series Boards', Eagle Technology, Fifth Edition, 1996

'Amplicon Liveline Catalogue', 1997, Vol. 2

'User Manual for the PC30F and PC30G Series Boards', Eagle Technology, Fifth Edition, 1996

'Micro-Hydro Design Manual - A Guide to Small-Scale Water Power Schemes', Harvery A, Intermediate Technology Publications, 1993

'Assessment of Hydroturbine Models for Power-Systems Studies', Smith JR, McLean R, Robbie JF, IEE Proc., Vol.130, Pt. C, No.1, Jan. 1983

'HydraulicTurbine and Turbine Control Models for System Dynamic Studies', Working Group on Prime Mover and Energy Supply Models for System Dynamic Performance Studies, IEEE Trans. on Power Systems, Vol.7, No. 1, Feb. 1992

'Accurate Low Order Model for Hydraulic Turbine-Penstock', Sanathanan CK, IEEE Trans. on Energy Conversion, Vol. EC-2, No. 2, June 1987

'Control Structures Analysis for a Real Time Wind System Simulator', Nichita C, Diop AD, Belhache J, Dakyo B, Protin L, Wind Engineering, Vol. 22, No. 6, 1998