REAL TIME EMULATION ENVIRONMENT FOR DIGITAL CONTROL DEVELOPMENT

by

H.J.SLATER B.Eng A.M.I.E.E.

Thesis submitted for the Degree of

Doctor of Philosophy in the Faculty of Engineering.

Department of Electrical and Electronic Engineering

The University of Newcastle Upon Tyne

October 1996

ABSTRACT

Simulation is a powerful tool for developing electric drive systems. Simulations allow the designer to experiment with control algorithms and hardware systems in a safe environment. To this end simulation is becoming increasingly popular. Off-line simulation does have its limitations in that the controller developed during the simulation period has eventually to be transferred to the target processor which will operate in the actual drive system. If, however, a real-time simulation environment could be realised, then the actual controller running in the actual target processor could be included in the simulation. Therefore no translation of code would be required once the controller had been developed and tested within the simulation. This would obviously lead to a reduction in development time and eliminate any possibility of introducing errors due to the translation between the simulated and actual controllers.

This thesis describes the development of such a system using a multiple digital signal processing environment. The real-time simulated drive is operated in parallel with an experimental drive to allow a direct comparison between the two.

The ability of the multiple processing system to operate in real-time has allowed the whole concept of simulation to be taken a stage further by the development of a real-time power level simulator. This simulator is capable of emulating a machine and load in real-time with real level of voltage and current. It is designed to replace a real machine during the development and testing stages of drive manufacture. This Virtual Machine is a controllable source/sink which is driven by the real-time simulation, and because of this the Virtual Machine takes on the characteristics of any choice of model within the real-time simulation. Moreover, because of its ability to handle bi-directional power flow, the Virtual machine can be programmed to emulate motors or generators. The Virtual Machine also includes the emulation of loads, thus making it extremely flexible and of interest to applications such as machine tools, electric vehicles, and wind generators, to name but a few.

ACKNOWLEDGEMENTS

I would like to thank my tutors Professor Alan Jack and Dr. David Atkinson for their support and assistance throughout the project. I would also like to thank all of the workshop staff at the department especially John and Jack whose friendly chats about the ups and downs of a certain football team have always brought a smile to my face.

I would like to thank EPSRC for funding the project and also Control Techniques and Cegelec for their generous supply of equipment.

During the three years that I have spent working in the U.G. Lab I have made many friends, Andy, Pete, Chris, Jim H., Gavin, Jim K., Ken, Behnard (1966), Volker, Chris with the hair and Christian to name but a few. They are all exceptional individuals and I wish them well with their futures wherever they may end up. Thanks go to Jim for having the noisiest rig ever invented and for positioning it three feet from my desk.

I would like to thank Julie my wife for her constant support and patience during the last three years and my parents for constantly asking "How's it going?" when the look on my face must have given them a clue. They seemed genuinely interested in discussing the timing problems of Texas C40s or anything else that was weighing heavy on my mind.

TABLE OF CONTENTS

Chapter 1: INTRODUCTION

Chapter 2: INDUCTION MOTOR MODEL

Chapter 3: INVERTER SIMULATION

Chapter 4: CONTROLLER

Chapter 5: DIGITAL SIGNAL PROCESSING SYSTEM

Chapter 6: REAL TIME SIMULATION

Chapter 7: THE VIRTUAL MACHINE

Chapter 8: CONCLUSIONS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

ia	a-phase current
ib	b-phase current
ic	c-phase current
$\overline{i_s}$	Stator Current Vector
$\overline{i_r}$	Rotor Current Vector
$\overline{\psi_r}$	Rotor Flux Vector
V_{ds}, V_{ds}^s	D-axis stator voltage in stator reference frame
V_{qs}, V_{qs}^s	Q-axis stator voltage in stator reference frame
V_{ds}^e	D-axis stator voltage in rotor flux reference frame
V_{qs}^e	Q-axis stator voltage in rotor flux reference frame
V_{dr}	D-axis rotor voltage in stator reference frame
V_{qr}	Q-axis rotor voltage in stator reference frame
i_{ds}, i_{ds}^s	D-axis stator current in stator reference frame
i_{qs}, i_{qs}^s	Q-axis stator current in stator reference frame
i_{ds}^e	D-axis stator current in rotor flux reference frame
i_{qs}^e	Q-axis stator current in rotor flux reference frame
i_{dr}	D-axis rotor current in stator reference frame
i_{qr}	Q-axis rotor current in stator reference frame
ψ_{ds}	D-axis stator flux in stator reference frame
ψ_{qs}	Q-axis stator flux in stator reference frame
ψ_{dr}	D-axis rotor flux in stator reference frame
ψ_{qr}	Q-axis rotor flux in stator reference frame
ω_r	Rotor angular velocity
ω_1	Synchronous angular velocity
ω_2	Slip angular velocity
f_{slip}	Slip frequency
$f_{carrier}$	Carrier frequency
f	Supply frequency
V	Supply voltage

ψ_{ag}	Air gap flux
$\overline{\psi_{mag}}$	Magnetising flux vector
$\overline{i_{mag}}$	Magnetising current vector
L_{mag}	Magnetising inductance
V_{dc}	DC link voltage
V_{AN}	A phase voltage
V_{BN}	B phase voltage
V_{CN}	C phase voltage
Ta	'on' time for a-phase upper switching device
Tb	'on' time for b-phase upper switching device
Tc	'on' time for c-phase upper switching device
to	zero voltage state time (all upper off)
ta	Switching state 'a' time
tb	Switching state 'b' time
t7	zero voltage state time (all upper on)
L_s	Stator total inductance
L_r	Rotor total inductance
L_m	Mutual inductance
R_s	Stator resistance
R_r	Rotor resistance
R_l	Stator effective resistance
X_l	Stator leakage reactance
V_l	Stator terminal voltage
X_m	Magnetising reactance
R_m	Magnetising resistance
L_l	Stator leakage inductance
L_2	Rotor leakage inductance
E_l	Back emf
X_2'	equivalent rotor leakage reactance
R_2'	equivalent rotor resistance
J	Inertia
T_e	Electromagnetic torque

T_{load}	Load torque
R_{dump}	DC link dump resistance
I_{link}	Current through link inductor
$I_{inverter}$	Current taken by inverter
V_{ss}	Rectified 3-phase mains voltage
θ_r	Angle between stator d-axis and rotor d-axis
θ_2	Angle between rotor current vector and rotor d-axis
θ_f	Angle between rotor flux vector and stator d-axis

1. INTRODUCTION

1.1 Introduction

The field of Electrical drives encompasses many disciplines including motor technology, power electronics and digital microcontrollers. Since its invention by Nikola Tesla in 1891 and the further development to a cage rotor design two years later by Dobowolsky, the induction machine has proved to be one of the most popular machines in industry. Its simple construction gives it an inherent reliability which cannot be matched by DC or synchronous machines. It is this robustness and the fact that it is relatively inexpensive to produce that has made the induction machine suitable for many applications.

The induction machine is by nature a constant speed machine and it has taken the advancement of power electronic switching technology to allow the induction machine to be used in variable speed applications. The development of power electronic converters has been centred on the advances made in the field of semiconductor switching devices. As faster switching devices are developed, capable of switching more power, their popularity increases and hence their cost reduces, the popularity of the IGBT was predicted by Lipo [*Lipo*, 1988]. An overview of modern power switching devices is included in the publication by Bose [*Bose*, 1992]. The development and availability of converters has meant that the induction machine is now a viable option in many variable speed applications.

The one area in which the DC machine has always triumphed over the induction machine is that of control. It is only recently with the introduction of more advanced control strategies, namely field orientated control, that the induction machine has been able to compete with the DC machine in terms of versatile control. These complex

control strategies are now achievable because of the dramatic development of microprocessor technology. The cost and availability of processing power has allowed controllers to be implemented in software as opposed to analogue hardware. The implementation of software controllers has given a greater flexibility to the control algorithm designer who can now realise complex control algorithms which were previously impossible with analogue circuits.

The increased complexity of the machine, converter, controller drive system has necessitated the use of simulations which model the behaviour of the machine, converter and controller. Simulations provide a safe environment in which the behaviour of the drive system can be observed and in which complex control strategies can be experimented with. Simulations have also benefited from the rapid advancements made in microprocessor technology. As faster more powerful processors become available simulations can be carried out in less time and to a greater level of detail resulting in increased accuracy.

Electric Drives have developed in such a way that they now encompass many diverse technologies and disciplines. The induction machine, now coupled to a voltage source inverter and controlled by a microcontroller, remains as popular as ever. As the personal computer continues to snowball into a faster, more powerful tool, simulations will become commonplace and will cease to be time consuming over night runs.

1.2 Field Orientated Control

Field orientated control or vector control was first introduced by Hasse in 1969 and Blaschke in 1972 [*Blaschke*, 1972]. It has become an increasingly popular method of controlling the induction machine, as its versatility allows the machine to be used in many and varied applications [*Leonhard*, 1988] [*Finch, Atkinson, Acarnley*, 1990]. The area of field orientated control has, for some time, been the subject of considerable research. The basic principle of field orientated control is to allow the induction machine to be controlled in a similar manner to a separately excited DC machine, that is it gives

individual control of the flux and torque of the machine. In a separately excited DC machine, and ignoring cross saturation effects (i.e. armature reaction), the armature current is the torque component and the field current is the flux component, therefore the field current can be set to maintain the rated flux of the machine and the armature current can be controlled to directly control its torque. By having separate armature and field coils in the machine the armature and field currents are decoupled and this allows maximum torque response. In a cage induction machine the only winding available is that of the stator, therefore it is not obvious how two decoupled machine currents which separately influence the flux and torque of the machine can be controlled.

Field orientated control overcomes the problem of having just the stator winding available by extracting two components of the stator current which represent the decoupled components of the flux and torque within the machine. These decoupled components are derived using knowledge of the position and magnitude of the machine flux vector. Two methods of obtaining the required information about the machine flux vector are available, which are known as direct and indirect. In direct field oriented control the magnitude and position of the machine flux are obtained by either explicit measurement or by calculations based on directly measurable machine values such as currents, terminal voltages and rotor speeds together with a flux model, both these types of direct field orientated control are described and compared by Gabriel et al [*Gabriel, Werner and Nordy,* 1980]. Direct field orientated control is also discussed by Jansen et al, in which a comparison of observer-based direct field oriented controllers is made [*Jansen, Lorenz, Novotny,* 1994]. In indirect field orientated control the machine flux is predicted using demanded currents and no measured values are used.

Measuring the machine flux has proved unpopular due to the necessity to have flux sensors, to apply a field orientated controller to an existing machine would require the addition of such sensors within the machine which proves to be both difficult and expensive. Direct field oriented control which uses readily available measurable values of current, voltage and speed or position is now a popular approach, this type of controller requires a model which can estimate the position and magnitude of the machine flux which is used to obtain the decoupled components of stator current. The model of the

machine flux can be calculated within various reference frames, the book by Vas [*Vas*, 1990] gives a comprehensive guide to obtaining the magnitude and position of the machine flux in the stator flux, the rotor flux and the magnetising flux reference frames. The rotor flux reference frame has proved popular as it provides simple decoupling of the torque and flux of the machine, in this reference frame the stator current component controlling the flux is in line with the rotor flux reference frame and the stator current component controlling the torque is perpendicular to the rotor flux reference frame.

The flux model requires information regarding the machine parameters and therefore has the disadvantage that parameter variation degenerates the performance of the controller. If the parameters used by the flux model are inaccurate then the estimate of the flux vector becomes inaccurate and true decoupling between the torque and flux producing components is not achieved. For this reason much research is being carried out into parameter estimation techniques [*Garces*, 1980] [*Lorenz*, 1986] and adaptive control techniques [*Liaw, Chao, Lin*, 1992], notably compensation for parameter variation due to saturation [*Levi*, 1994a], [*Moreira*, 1993] and compensation for iron loss [*Levi*, 1994b,1995]. These attempts aim to make the flux modelling more accurate and thus improve the performance of the field orientated controller. Parameter estimation and adaptive control schemes have benefited from the advances made in microprocessor technology. Cheaper, more powerful processors which are now available, allow complex control ideas to be realised in which parameter modelling can be included.

A further indication of how eager drive manufacturers are to eliminate the requirement of sensors is the amount of research being carried out into the field of position estimation [*Jansen, Lorenz*, 1993]. The elimination of a position sensor would provide a further cost reduction and mean that vector controlled drives could be used for existing machines to which adding a position sensor would prove difficult.

It can be seen that the future of the field orientated drive looks to be a drive which uses only measurable values of current, which are anyway required for protection and closed loop control. The drive will include an adaptive control strategy in which

machine parameters can be varied, thus maintaining the performance of the controller and estimate machine speed accurately down to zero speed.

1.3 Simulation

As controllers become more complex and the performance of the drive system more critical, it is becoming increasingly important to be able to predict the exact behaviour of the drive and controller when subjected to various conditions. It is because of this that simulating the controller and drive system, before it is implemented in the actual hardware and control processor, has become more widespread amongst drive manufacturers. Simulation provides the designer of the control algorithms and drive hardware a safe environment in which he can experiment with controller parameters and also observe the effect of variations in machine and load parameters. An overview of computer simulation of variable speed drives is presented by Finney [*Finney*, 1994].

The simulation environment has normally been provided by two methods, either by writing dedicated software in a high level language such as 'C', or by using a commercial "off the shelf" simulation package such as Matlab, Saber or Spice. The first method has the obvious disadvantage that a detailed knowledge of programming is required and the environment takes longer to develop, the result, however, may be more appropriate to the required task. The second method, using commercially available packages to build the simulation, has the advantage that they are extremely user friendly and the operator can quickly become familiar with them. Modern simulation packages such as Matlab allow the user to quickly put together a simulation from a library of general building blocks provided with the package or a more dedicated library purchased separately. The packages also provide tools for graphing and mathematical functions which can be useful to the user when analysing results. One of the country's largest drive manufacturers recently adopted this approach to obtaining a useful simulation environment [*Slater, Armstrong,* 1995]. A simulation of a simple drive system is presented by Chowdhury and Giesselmann in which the drive is modelled using an evaluation version of MicroSim's Design Centre 3. This package uses the PSPICE

simulation engine and also a schematic editor for graphical circuit input [*Chowdhury, Giesselmann,* 1994]. Another example of a PC based simulation of an extremely basic drive system is presented by Nigim [*Nigim,* 1994] in which the drive system is simulated using the popular commercial mathematics package MathCad. This has drawbacks, in that the package chosen is not specifically designed for simulation and therefore all of the dedicated simulation features available with a package such as Matlab are not available.

Dedicated simulation environments do have the advantage that, because they do not have to be as general as commercial simulation packages, they can have faster executable times and can be more adaptable to the designers specific requirements. An example of a recently developed dedicated simulation environment is the simulation package SIMUVEC [*Vas, Li,* 1993]. SIMUVEC is a PC based simulation package which models an induction machine together with various field orientated controllers. The controllers that are simulated can be linear or adaptive and the machine models are those presented by Vas [*Vas,* 1990]. Another dedicated simulation environments is CASED which is demonstrated by simulating a current source inverter fed induction motor by Kleinhans et. al. [*Kleinhans et al,* 1994]. Finney describes a PC simulation which contains a simulation of eight different drive systems [*Finney,* 1994], the simulator is a menu driven, user friendly environment, but the simulations are for steady state operation only. The publication states that even today's fast personal computers are incapable of simulating the behaviour of the drive system in anything close to real-time.

A different approach to simulation and drive development is that presented by Chhaya and Bose [*Chhaya, Bose,* 1993]. In this publication an expert system is proposed which provides an automated simulation and design aid for a voltage source inverter for an induction machine. The system works by having a consultation period with the drive designer who provides the expert system with information regarding the inverter power rating and performance specification. The expert system then designs the inverter and performs a simulation of the system. The simulated results are used by the expert system to optimise the drive design. The simulation of the drive design which is

performed by the expert system is carried out on a personal computer using the PC-SIMNON simulation language. An approach which is similar to the above is that discussed by Virk and Liang [*Virk, Liang,* 1993], the simulation of an induction motor drive is carried out in this publication by using a neural network rather than a set of ordinary differential equations. The authors claim that this method is 875 times faster than the differential equation method. The traditional method with which they compare the neural network is a Matlab simulation of an induction machine, this will be slow when compared to a dedicated PC simulation solving the differential equations. The neural network does, however, require a training period but this is claimed to be just 120 seconds.

All of the above approaches to simulation have one thing in common, that is that the controller used in the simulation environment is itself a simulation. Once the performance of the drive design and the control algorithms have been verified then the control algorithms must be translated into the code which will run in the actual control processor. This translation of code is another delay in realising the actual controller and a step in which human error can be introduced. Systems are being developed such as Dspace and a system originated at Newcastle University for CAPEC [*French,* 1994], in which an easy to use commercial simulation package is used to provide the simulation environment and then the controller code is generated in a format which can be down loaded to the target control processor. A further improvement on this approach would be to use the actual control algorithms running in the actual control processor within the simulation. This approach allows the actual controller code to be developed immediately within the target hardware. This would obviously require the simulation environment to operate in real time in order to keep up with the control processor.

A real time simulation environment can be developed by using a multiple parallel processor system in which the control processor interacts with other processors which carry out real time simulations of the other components which make up the drive system. This system has the advantage that the actual control code can be developed directly in the simulation environment and there is no translation of the control algorithms at all, it

also has the advantage of execution time. This approach is presented by Bosga et al [*Bosga et al,* 1995] and is in use as a teaching aid.

1.4 The Virtual Machine

When developing a drive or control algorithm eventually the drive and controller must be tested with a real machine, this gives rise to several worries. If the control algorithm fails to perform as expected i.e. it loses control or fails to control correctly it could cause serious damage to the drive and/or machine, if the drive develops a fault it too could cause damage to itself and also possibly the machine it is being tested with.

It is possible to replace the actual machine with a "Virtual Machine". This Virtual Machine emulates at actual power levels the behaviour of an induction machine and can been designed to allow the drive to be tested at actual power levels without the need of a real machine, as far as the drive is concerned it is connected to a real machine. The Virtual Machine controls the currents drawn from the drive to match the currents which would be drawn if it were connected to a real machine, these current demands are obtained from the real time machine model simulation which is driven directly from the output of the drive. The machine model parameters can easily be changed to make the Virtual Machine behave as a complete range of different machines. The Virtual Machine provides a safe, flexible emulation environment in which drives can be tested to their full capacity without the need for any actual hardware.

1.5 Overview Of The Project

The research carried out and documented in this thesis investigates the ability of a multiple, parallel processing system to perform accurate real time simulations of an inverter and induction machine drive system. In order to achieve this a multiple digital signal processing (D.S.P) system is developed, in which the real time simulation can be implemented.

An actual field orientated controller is developed which operates in the target processor. A typical three phase voltage source inverter which feeds a 3.5KW induction machine is developed based on IGBT switching devices. This laboratory drive is controlled by the field orientated controller. In order to demonstrate the accuracy of the real time simulation, the simulation is run in parallel with the actual drive and induction machine, this allows the results of the simulation to be compared directly to the actual measured values.

An graphical user interface (GUI) is developed which allows the operator to enter variables such as machine parameters, controller gains and demands, and also provides the user with a graphical output of captured real time data. The user interface is implemented in Windows based software, providing the user with a familiar environment of menus, dialogue boxes, slide bars etc.

A Virtual Machine is developed which can replace the actual machine and be used to test the drive system. The performance of this Virtual Machine when used to test a commercial drive is presented and compared to the performance of the same commercial drive when connected to an actual machine.

1.6 Thesis Layout

The main body of the thesis is contained in six chapters, chapters two to seven. The first three of these chapters explains the three main components which make up a drive system namely the machine, the inverter and the controller. These components are described in chapters two, three and four respectively. These chapters explain how the components of the drive system are simulated.

Chapter five describes the multiple digital signal processing system which is used to achieve real-time simulation. Chapter six explains how the simulation and controller code which was described in chapters two to four is implemented within the multiple parallel processing environment and reports on the performance comparison between the actual drive and the real-time simulation of the drive when both systems are run in parallel and fed from exactly the same controller.

Chapter seven introduces the idea of the Virtual Machine. The chapter includes a simulation of the Virtual machine and discusses the hardware and software which is required to develop a Virtual Machine. The chapter also reports results taken from testing a commercial inverter with the Virtual Machine. A comparison is made between testing the inverter with an actual machine and testing the inverter with the Virtual Machine.

2. INDUCTION MOTOR MODEL

2.1 Introduction

The induction machine was first proposed in 1891 by Nikola Tesla and two years later a cage rotor machine was described by Dobrowolsky [*Say*, 1983], since then the induction machine has gained immense popularity and has been used in a wide variety of applications, for example pumps, steel mills and hoist drives [*Krause*, 1986]. The main reason for the induction machines popularity can be attributed to its simple construction which makes it relatively inexpensive and extremely reliable [*Fitzgerald*, 1990].

The machine has gained even more popularity with the introduction of the variable speed drive, this allows the machine to be used in many variable speed applications where it previously was not suitable. With the advances made in the area of microprocessor technology more elaborate control algorithms are capable of being implemented and the drive system has become extremely complex. This complexity has led to the necessity to be able to simulate the drive system and all of its constituent parts. One major part of the drive system is the machine which must therefore be simulated accurately so that its effect on the drive system can be confidently predicted.

The well documented per phase equivalent circuit of a three phase induction machine is a recognised method of analysing the steady state performance of the machine. When the machine is used in a drive system however, it is the dynamic behaviour of the machine which must be considered [*Bose*, 1986]. In order to produce an accurate simulation of the induction machine a model must be developed which is capable of simulating the dynamic behaviour of the machine accurately. The dynamic model is based on the circuit equations of the machine but since the circuits may be magnetically coupled these equations may become complicated. It is because of this

complication that the twin axis approach is used. The representation of a three phase winding with an equivalent two phase winding was first introduced and applied to synchronous machines by Park in the late 1920s [*Jones*, 1967][*Krause*, 1986] and later developed by Kron to deal with all rotating machines [*Say*, 1983]. The two axis d-q approach is a recognised method of representing and analysing the dynamic behaviour of a three phase machine [*Bose*, 1986]. It has the advantage that there will be no magnetic coupling between the two axes and therefore the machine equations will be simplified.

The following twin axis model assumes that all magnetic circuits are linear i.e. free from saturation. It also assumes that magnetomotive forces (mmfs) and fluxes can be represented by the fundamental component of their spatial distribution. Any higher order harmonic components of these fundamentals are therefore ignored. The machine windings are assumed to be evenly distributed around the machine and slot effects are ignored. These previous assumptions mean that effects inside the machine such as crawling and cogging cannot be modelled and as such are lost.

The following chapter will determine the equations for the currents in the machine and the electromechanical torque produced by the machine in terms of the machine parameters. The method used for machine parameter identification will be reported and the performance of the machine model will be investigated with emphasis on the execution time of the simulation.

2.2 Two Phase Equivalent Windings

The machine equations developed for the motor model used in this thesis will be based on the twin axis d-q theory. A graphical representation of the a three phase machine and its equivalent two phase machine is shown in Fig. 2.2.1.

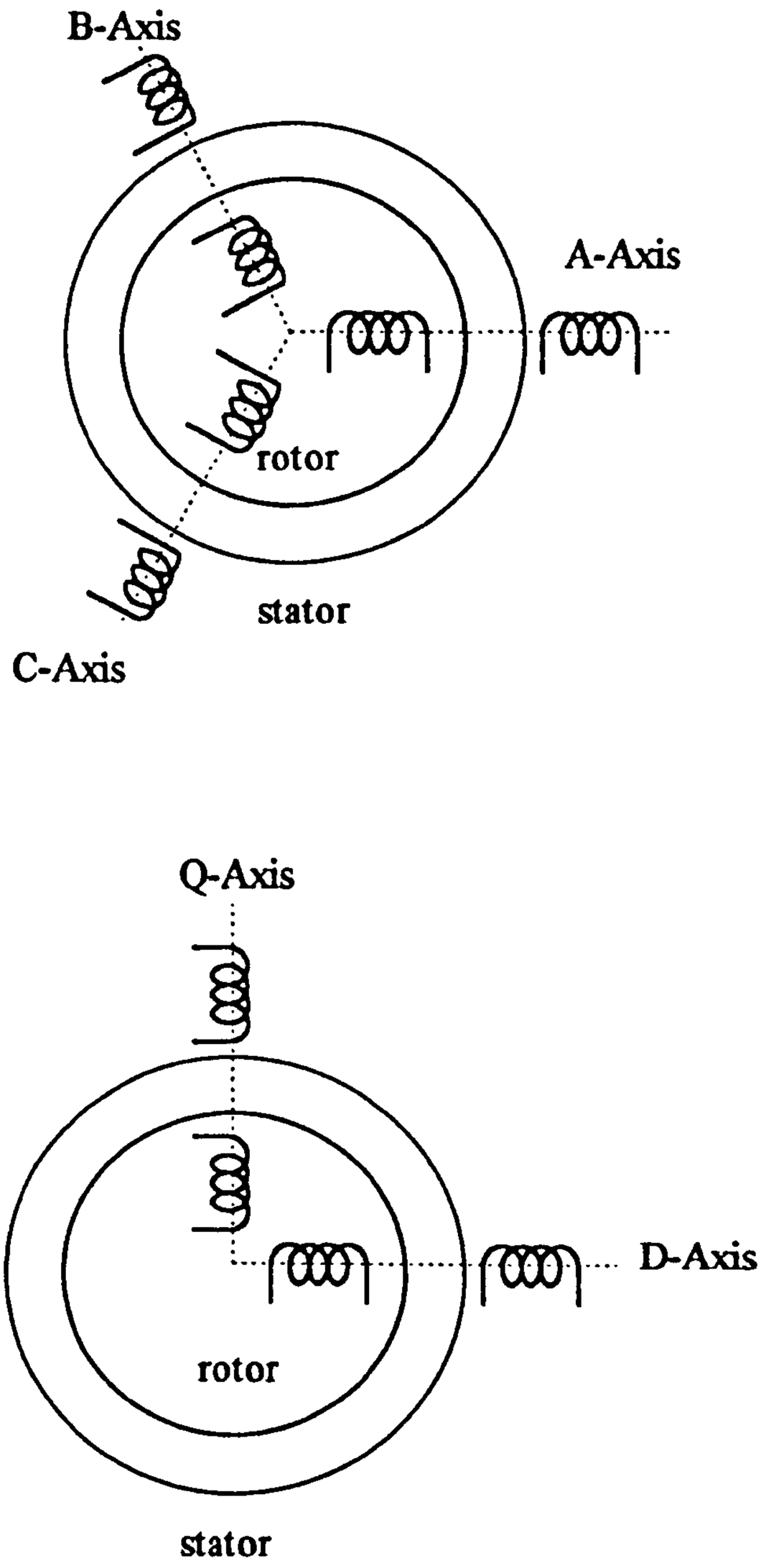


Figure 2.2.1: Equivalent Three and Two Phase Windings

Consider the three phase stator winding of Fig. 2.2.1. Each coil carries a current which results in spatial mmf distribution in the air gap, the resultant mmf distribution of the three phase windings is a stepped wave which will have a near sinusoidal fundamental [*Fitzgerald,* 1990]. Since the three phase currents are contributing to one resultant mmf which can be thought of as a space vector, then the three phase stator currents can be thought of as vectors, $\overline{ia}, \overline{ib}, \overline{ic}$, which can be summed to produce a resultant space vector stator current $\overline{i_s}$. Fig. 2.2.2 shows three phase current vectors and the resultant stator current vector for one instant in time and how the same resultant stator current vector can be produced from a two phase winding.

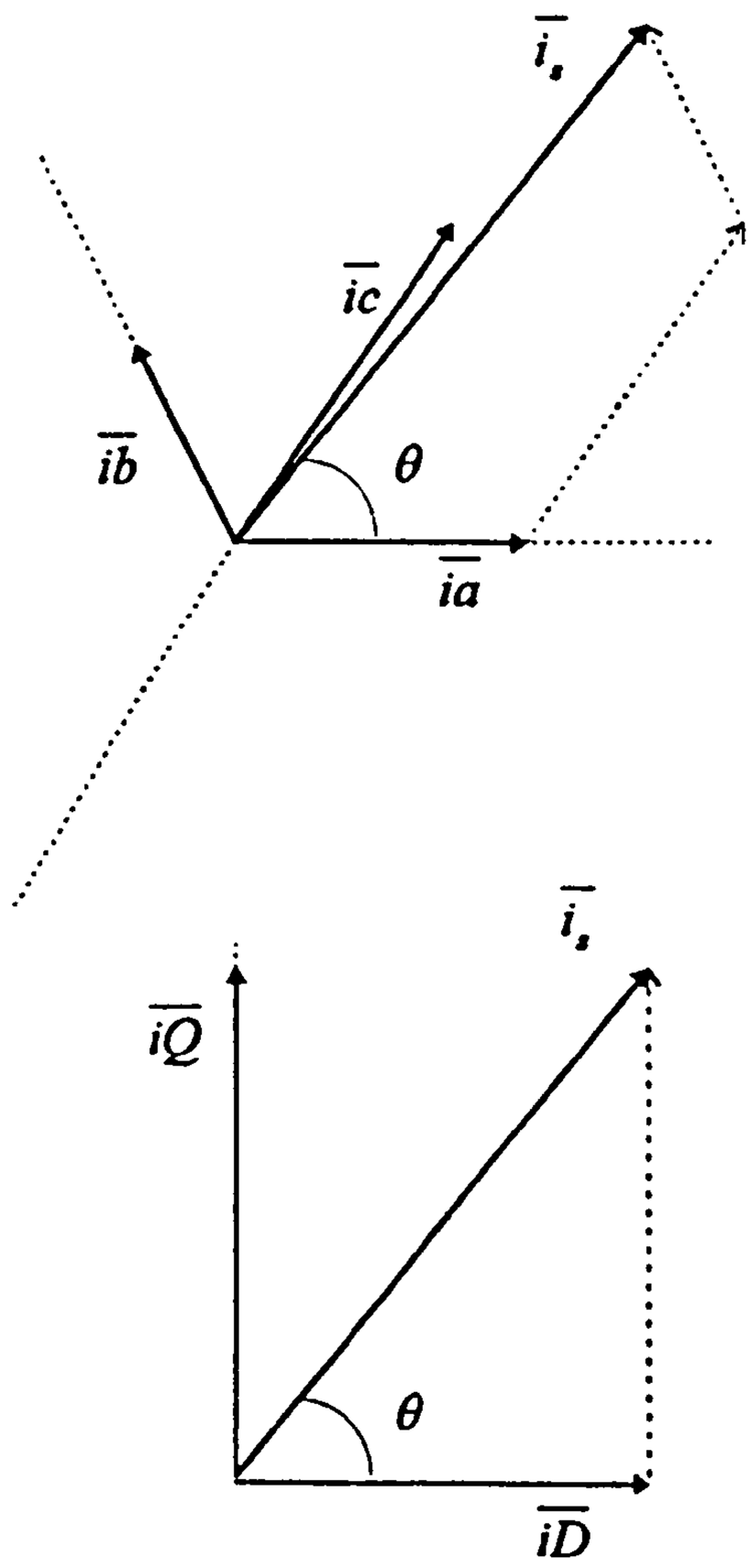


Figure 2.2.2: Stator Current Vector from Three and Two Phase System

2.3 Axis Transformations

Consider the vector diagram of Fig. 2.3.1. This shows the orthogonal two phase vectors offset from the axis of the three phase vectors by an arbitrary angle of θ.

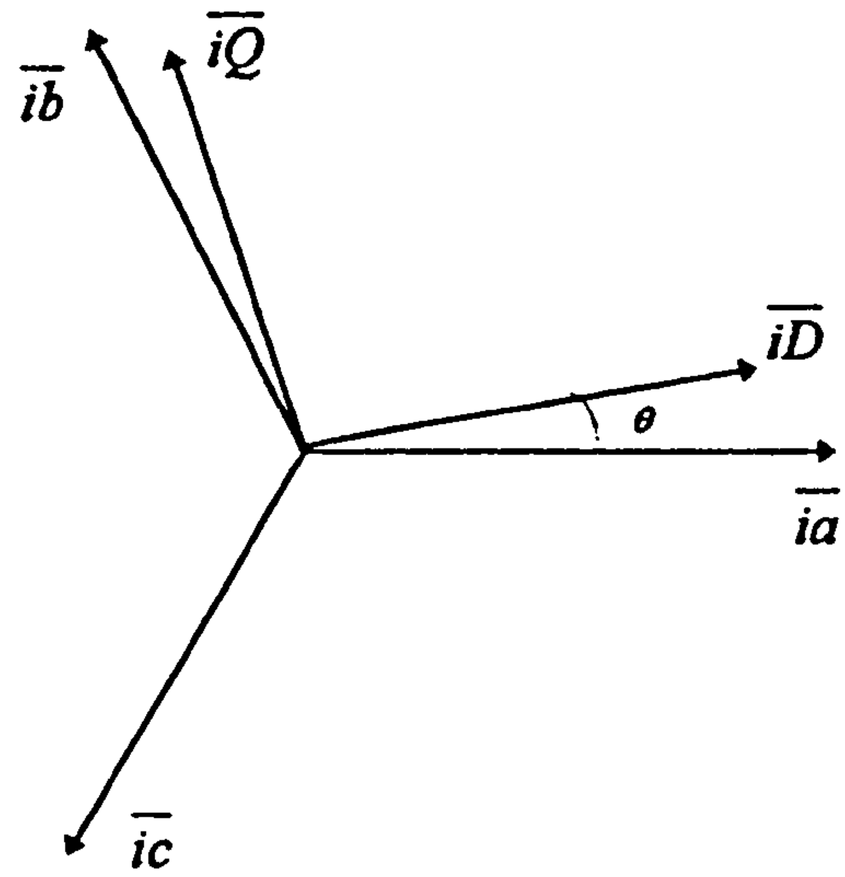


Figure 2.3.1: Axis Transformation

It is common practice to neglect the zero sequence component of the two phase system because the neutral of the machine is generally isolated and therefore no zero sequence current can flow. The phase values can be described in terms of the D-Q components as:-

$$ia = iD\cos\theta - iQ\sin\theta \qquad (2.3.1)$$

$$ib = iD\cos(120^0 - \theta) + iQ\sin(120^0 - \theta) \qquad (2.3.2)$$

$$ic = iD\sin(120^0 - \theta) - iQ\cos(120^0 - \theta) \qquad (2.3.3)$$

and the D-Q values can be described in terms of the phase components as:-

$$iD = \frac{2}{3}\left(ia\cos\theta - ib\sin(30^0 - \theta) - ic\sin(30^0 - \theta)\right) \qquad (2.3.4)$$

$$iQ = \frac{2}{3}\left(-ia\sin\theta + ib\cos(30^0 - \theta) - ic\cos(30^0 + \theta)\right) \qquad (2.3.5)$$

It is common practice to align the two sets of axis such that $\theta = 0$ and the D-axis is coincident with the a-axis. This results in the following transformations:-

$$ia = iD \tag{2.3.6}$$

$$ib = -\frac{1}{2}iD + \frac{\sqrt{3}}{2}iQ \tag{2.3.7}$$

$$ic = -\frac{1}{2}iD - \frac{\sqrt{3}}{2}iQ \tag{2.3.8}$$

$$iD = \frac{2}{3}ia - \frac{1}{3}ib - \frac{1}{3}ic \tag{2.3.9}$$

$$iQ = \frac{1}{\sqrt{3}}ib - \frac{1}{\sqrt{3}}ic \tag{2.3.10}$$

Quantities of voltage, flux and current can be transformed using these transformations but it must be noted that these transformations are not power invariant. The product of the three phase voltages and currents will not equal the product of the transformed two phase voltages and currents, and for this reason the power and torque calculated from transformed quantities must be adjusted by a factor of 3/2. The non-power invariant transformation has advantages for speed of execution in the processor.

2.4 Machine Equations

2.4.1 Electrical Equations

In order to perform a simulation of the machine, equations must be derived which represent the variables within the machine. These equations can then be solved by mathematical methods in a microprocessor. Consider the two phase representation of the three phase stator winding described in the previous section and graphically represented in Fig. 2.2.1, if the two phase axis is considered to be stationary and attached to the stator axis then the voltage equations for the D and Q axis stator windings are:-

$$V_{ds} = R_s i_{ds} + \frac{d\psi_{ds}}{dt} \qquad (2.4.1)$$

$$V_{qs} = R_s i_{qs} + \frac{d\psi_{qs}}{dt} \qquad (2.4.2)$$

Consider the rotor winding also as its two phase equivalent as shown in Fig. 2.2.1 and that the rotor is rotating at some angular velocity ω_r , the voltage equations for the rotor D and Q rotor windings are:-

$$V_{dr} = R_r i_{dr} + \frac{d\psi_{dr}}{dt} + \omega_r \psi_{qr} \qquad (2.4.3)$$

$$V_{qr} = R_r i_{qr} + \frac{d\psi_{qr}}{dt} - \omega_r \psi_{dr} \qquad (2.4.4)$$

Consider now the flux linkages of the stator and rotor in the D and Q axis:-

$$\psi_{ds} = L_s i_{ds} + L_m i_{dr} \qquad (2.4.5)$$

$$\psi_{qs} = L_s i_{qs} + L_m i_{qr} \qquad (2.4.6)$$

$$\psi_{dr} = L_r i_{dr} + L_m i_{ds} \qquad (2.4.7)$$

$$\psi_{qr} = L_r i_{qr} + L_m i_{qs} \qquad (2.4.8)$$

Substituting the equations for the flux linkages (2.4.5)-(2.4.8) into the voltage equations (2.4.1)-(2.4.4) gives:-

$$V_{ds} = R_s i_{ds} + L_s \frac{di_{ds}}{dt} + L_m \frac{di_{dr}}{dt} \qquad (2.4.9)$$

$$V_{qs} = R_s i_{qs} + L_s \frac{di_{qs}}{dt} + L_m \frac{di_{qr}}{dt} \qquad (2.4.10)$$

$$V_{dr} = R_r i_{dr} + L_r \frac{di_{dr}}{dt} + L_m \frac{di_{ds}}{dt} + \omega_r L_r i_{qr} + \omega_r L_m i_{qs} \qquad (2.4.11)$$

$$V_{qr} = R_r i_{qr} + L_r \frac{di_{qr}}{dt} + L_m \frac{di_{qs}}{dt} - \omega_r L_r i_{dr} - \omega_r L_m i_{ds} \qquad (2.4.12)$$

It is now necessary to make the currents the subject of the equations since it is these that will be solved for given values of voltages, angular velocities etc.

$$L_s \frac{di_{ds}}{dt} = -R_s i_{ds} - L_m \frac{di_{dr}}{dt} + V_{ds} \qquad (2.4.13)$$

$$L_s \frac{di_{qs}}{dt} = -R_s i_{qs} - L_m \frac{di_{qr}}{dt} + V_{qs} \qquad (2.4.14)$$

$$L_r \frac{di_{dr}}{dt} = -R_r i_{dr} - L_m \frac{di_{ds}}{dt} - \omega_r L_r i_{qr} - \omega_r L_m i_{qs} + V_{dr} \qquad (2.4.15)$$

$$L_r \frac{di_{qr}}{dt} = -R_r i_{qr} - L_m \frac{di_{qs}}{dt} + \omega_r L_r i_{dr} + \omega_r L_m i_{ds} + V_{qr} \qquad (2.4.16)$$

Since the model represents a squirrel cage induction machine in which the rotor windings are made up of conductor bars embedded in the rotor which are short circuited with end rings it can be assumed that the rotor voltages will be zero, therefore:-

$$V_{dr} = V_{qr} = 0 \qquad (2.4.17)$$

Substituting these values of rotor voltage into (2.4.13)-(2.4.16) and making appropriate substitutions to force the equations to contain only like differential currents gives:-

$$L_s \frac{di_{ds}}{dt} = -R_s i_{ds} - \frac{L_m}{L_r}\left(-R_r i_{dr} - L_m \frac{di_{ds}}{dt} - \omega_r L_r i_{qr} - \omega_r L_m i_{qs}\right) + V_{ds} \quad (2.4.18)$$

$$L_s \frac{di_{qs}}{dt} = -R_s i_{qs} - \frac{L_m}{L_r}\left(-R_r i_{qr} - L_m \frac{di_{qs}}{dt} + \omega_r L_r i_{dr} + \omega_r L_m i_{ds}\right) + V_{qs} \quad (2.4.19)$$

$$L_r \frac{di_{dr}}{dt} = -R_r i_{dr} - \frac{L_m}{L_s}\left(-R_s i_{ds} - L_m \frac{di_{dr}}{dt} + V_{ds}\right) - \omega_r L_r i_{qr} - \omega_r L_m i_{qs} \quad (2.4.20)$$

$$L_r \frac{di_{qr}}{dt} = -R_r i_{qr} - \frac{L_m}{L_s}\left(-R_s i_{qs} - L_m \frac{di_{qr}}{dt} + V_{qs}\right) + \omega_r L_r i_{dr} + \omega_r L_m i_{ds} \quad (2.4.21)$$

The differential currents can now be made the subject of the equations:-

$$L_s \frac{di_{ds}}{dt} - \frac{L_m^2}{L_r}\frac{di_{ds}}{dt} = -R_s i_{ds} + \frac{L_m R_r i_{dr}}{L_r} + \frac{L_m \omega_r L_r i_{qr}}{L_r} + \frac{L_m^2 \omega_r i_{qs}}{L_r} + V_{ds} \quad (2.4.22)$$

$$L_s \frac{di_{qs}}{dt} - \frac{L_m^2}{L_r}\frac{di_{qs}}{dt} = -R_s i_{qs} + \frac{L_m R_r i_{qr}}{L_r} - \frac{L_m \omega_r L_r i_{dr}}{L_r} - \frac{L_m^2 \omega_r i_{ds}}{L_r} + V_{qs} \quad (2.4.23)$$

$$L_r \frac{di_{dr}}{dt} - \frac{L_m^2}{L_s}\frac{di_{dr}}{dt} = -R_r i_{dr} + \frac{L_m R_s i_{ds}}{L_s} - \frac{L_m V_{ds}}{L_s} - \omega_r L_r i_{qr} - \omega_r L_m i_{qs} \quad (2.4.24)$$

$$L_r \frac{di_{qr}}{dt} - \frac{L_m^2}{L_s}\frac{di_{qr}}{dt} = -R_r i_{qr} + \frac{L_m R_s i_{qs}}{L_s} - \frac{L_m V_{qs}}{L_s} + \omega_r L_r i_{dr} + \omega_r L_m i_{ds} \quad (2.4.25)$$

rearranging yields:-

$$\left(L_s - \frac{L_m^2}{L_r}\right)\frac{di_{ds}}{dt} = -R_s i_{ds} + \frac{L_m R_r i_{dr}}{L_r} + \frac{L_m \omega_r L_r i_{qr}}{L_r} + \frac{L_m^2 \omega_r i_{qs}}{L_r} + V_{ds} \quad (2.4.26)$$

$$\left(L_s - \frac{L_m^2}{L_r}\right)\frac{di_{qs}}{dt} = -R_s i_{qs} + \frac{L_m R_r i_{qr}}{L_r} - \frac{L_m \omega_r L_r i_{dr}}{L_r} - \frac{L_m^2 \omega_r i_{ds}}{L_r} + V_{qs} \quad (2.4.27)$$

$$\left(L_r - \frac{L_m^2}{L_s}\right)\frac{di_{dr}}{dt} = -R_r i_{dr} + \frac{L_m R_s i_{ds}}{L_s} - \frac{L_m V_{ds}}{L_s} - \omega_r L_r i_{qr} - \omega_r L_m i_{qs} \quad (2.4.28)$$

$$\left(L_r - \frac{L_m^2}{L_s}\right)\frac{di_{qr}}{dt} = -R_r i_{qr} + \frac{L_m R_s i_{qs}}{L_s} - \frac{L_m V_{qs}}{L_s} + \omega_r L_r i_{dr} + \omega_r L_m i_{ds} \quad (2.4.29)$$

$$\frac{di_{ds}}{dt} = -R_s\left(\frac{L_r}{L_sL_r - L_m^2}\right)i_{ds} + \left(\frac{L_mR_r}{L_sL_r - L_m^2}\right)i_{dr} + \left(\frac{L_mL_r\omega_r}{L_sL_r - L_m^2}\right)i_{qr} + \left(\frac{L_m^2\omega_r}{L_sL_r - L_m^2}\right)i_{qs} + \left(\frac{L_r}{L_sL_r - L_m^2}\right)V_{ds}$$

(2.4.30)

$$\frac{di_{qs}}{dt} = -R_s\left(\frac{L_r}{L_sL_r - L_m^2}\right)i_{qs} + \left(\frac{L_mR_r}{L_sL_r - L_m^2}\right)i_{qr} - \left(\frac{L_mL_r\omega_r}{L_sL_r - L_m^2}\right)i_{dr} - \left(\frac{L_m^2\omega_r}{L_sL_r - L_m^2}\right)i_{ds} + \left(\frac{L_r}{L_sL_r - L_m^2}\right)V_{qs}$$

(2.4.31)

$$\frac{di_{dr}}{dt} = -R_r\left(\frac{L_s}{L_sL_r - L_m^2}\right)i_{dr} + \left(\frac{L_mR_s}{L_sL_r - L_m^2}\right)i_{ds} - \left(\frac{L_sL_r\omega_r}{L_sL_r - L_m^2}\right)i_{qr} - \left(\frac{L_sL_m\omega_r}{L_sL_r - L_m^2}\right)i_{qs} - \left(\frac{L_m}{L_sL_r - L_m^2}\right)V_{ds}$$

(2.4.32)

$$\frac{di_{qr}}{dt} = -R_r\left(\frac{L_s}{L_sL_r - L_m^2}\right)i_{qr} + \left(\frac{L_mR_s}{L_sL_r - L_m^2}\right)i_{qs} + \left(\frac{L_sL_r\omega_r}{L_sL_r - L_m^2}\right)i_{dr} + \left(\frac{L_sL_m\omega_r}{L_sL_r - L_m^2}\right)i_{ds} - \left(\frac{L_m}{L_sL_r - L_m^2}\right)V_{qs}$$

(2.4.33)

In matrix format these are:-

$$\begin{pmatrix} \frac{di_{ds}}{dt} \\ \frac{di_{qs}}{dt} \\ \frac{di_{dr}}{dt} \\ \frac{di_{qr}}{dt} \end{pmatrix} = \begin{pmatrix} -\frac{L_rR_s}{K} & \frac{\omega_rL_m^2}{K} & \frac{L_mR_r}{K} & \frac{\omega_rL_mL_r}{K} & \frac{L_r}{K} & 0 \\ -\frac{\omega_rL_m^2}{K} & -\frac{L_rR_s}{K} & -\frac{\omega_rL_mL_r}{K} & \frac{L_mR_r}{K} & 0 & \frac{L_r}{K} \\ \frac{L_mR_s}{K} & -\frac{\omega_rL_sL_m}{K} & -\frac{L_sR_r}{K} & -\frac{\omega_rL_sL_r}{K} & -\frac{L_m}{K} & 0 \\ \frac{\omega_rL_sL_m}{K} & \frac{L_mR_s}{K} & \frac{\omega_rL_sL_r}{K} & -\frac{L_sR_r}{K} & 0 & -\frac{L_m}{K} \end{pmatrix} \times \begin{pmatrix} i_{ds} \\ i_{qs} \\ i_{dr} \\ i_{qr} \\ V_{ds} \\ V_{qs} \end{pmatrix}$$

where:- $K = (L_sL_r - L_m^2)$

(2.4.34)

2.4.2 Electromechanical Equations

The angular velocity of the rotor ω_r in the electrical equations above, is not a constant and a solution for it must also be sought. It can be related to the electromechanical torque produced by the machine and the load torque applied to the shaft of the machine. The equation for the change in angular velocity is:-

$$\frac{d\omega_r}{dt} = \frac{1}{J}\left(T_e - T_{load}\right) \qquad (2.4.35)$$

where:- J is the inertia of the system.

T_e is the electromagnetic torque produced by the machine.

T_{load} is the load torque.

This is a simple mechanical equation in which the load is assumed to be a single inertia joined to the machine by an infinitely stiff shaft. It can be seen from the above equation that if the electromagnetic torque is greater than the load torque the change in angular velocity will be positive and the machine will speed up, conversely if the load torque is greater than the electromagnetic torque the change in angular velocity will be negative and the machine will slow down.
Details of the inertia of the machine set and its loading are given in Appendix B.

The electromagnetic torque is produced by the machine by an interaction of the air gap flux and the rotor mmf. In vector form:-

$$T_e = -\frac{3}{2}\overline{\psi}_r \times \overline{i}_r \qquad (2.4.36)$$

$$T_e = -\frac{3}{2}\left(L_r\overline{i}_r + L_m\overline{i}_s\right) \times \overline{i}_r \qquad (2.4.37)$$

$$T_e = -\frac{3}{2}L_m(i_{qr}i_{ds} - i_{dr}i_{qs}) \qquad (2.4.38)$$

2.5 Machine Parameter Identification

In order to solve the machine equations which have been developed for the stator and rotor currents, it is necessary to have actual values for the machines various parameters of resistance and inductance. The machine used throughout the research was a GEC 3.0KW, three phase, single cage induction motor, the machine was mounted on a laboratory test bed and had its rotor coupled mechanically to that of a d.c. machine which could be used to apply a load to the induction machine. In order to identify the machines parameters various tests were performed on the motor. Three tests are required, a no-load test, a blocked rotor test and measurement of the stator resistance. During the tests measurements of the power taken by the motor were taken using the two wattmeter technique described in Appendix A of this thesis.

2.5.1 No-Load Test

The induction machine was driven by the d.c. machine at rated speed (1500 rpm) and was supplied with rated voltage at rated frequency. With the d.c. machine forcing the induction machine to rotate at synchronous speed, the slip is zero and the rotor resistance referred to the stator is infinite resulting in the rotor being effectively open circuit. The equivalent circuit of the induction machine can now be reduced to that shown in Fig. 2.5.1 below.

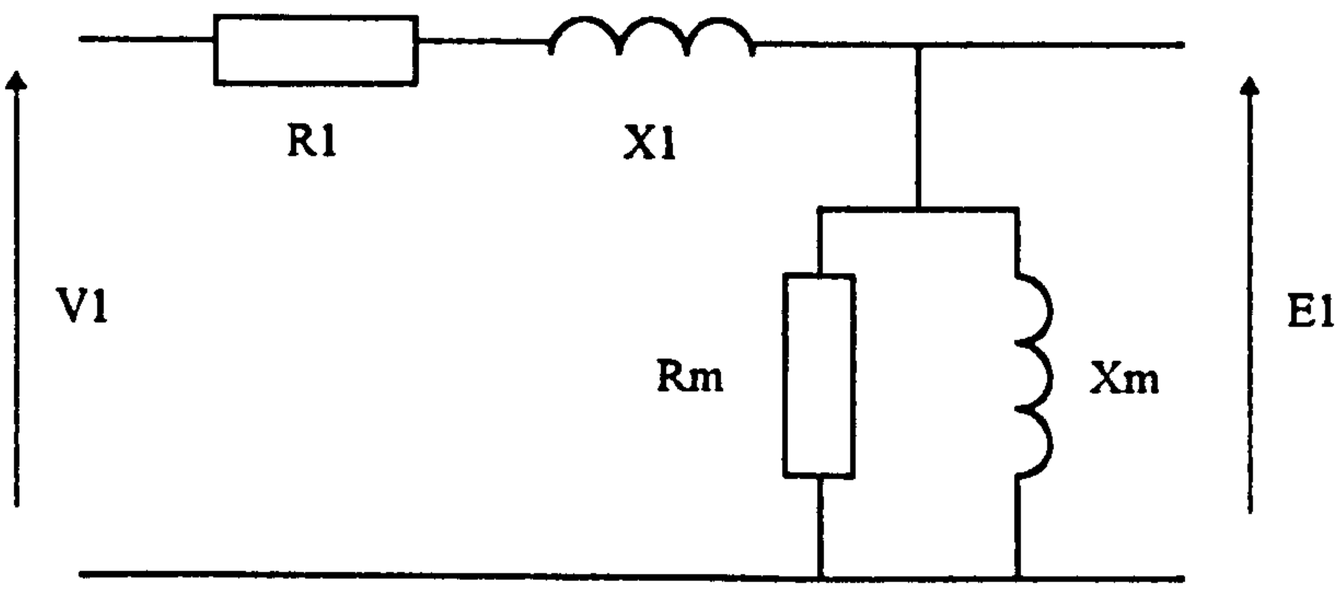


Figure 2.5.1: No Load Equivalent Circuit

It can be seen from Fig. 2.5.1 that the reactance measured at the stator terminals for the no load test will equal the stator leakage reactance plus the magnetising reactance, this is known as the stator self reactance:-

$$X_{11} = X_1 + X_m = X_{no-load} \tag{2.5.1}$$

The stator self reactance was calculated from measurement of the no load impedance and the no load resistance.

$$Z_{no-load} = \frac{V_{no-load}}{I_{no-load}} \tag{2.5.2}$$

$$R_{no-load} = \frac{P_{no-load}}{3I^2_{no-load}} \tag{2.5.3}$$

$$X_{no-load} = \sqrt{Z^2_{no-load} - R^2_{no-load}} \tag{2.5.4}$$

$$Z_{no-load} = \frac{240}{3.36} = 71.429\Omega \tag{2.5.5}$$

$$R_{no-load} = \frac{270}{33.87} = 7.972\Omega \tag{2.5.6}$$

$$X_{no-load} = \sqrt{5102.102 - 63.553} = 70.983\Omega \tag{2.5.7}$$

Information regarding the leakage reactance must now be obtained in order to determine the magnetising reactance from the stator self reactance.

2.5.2 Locked Rotor Test

The objective of the locked rotor test is to identify the leakage impedance parameters of the machine. The locked rotor test was performed by locking the rotor of the induction machine so that its shaft cannot rotate and applying a reduced but balanced voltage to the stator. The equivalent circuit for the locked rotor test is shown in Fig. 2.5.2 below.

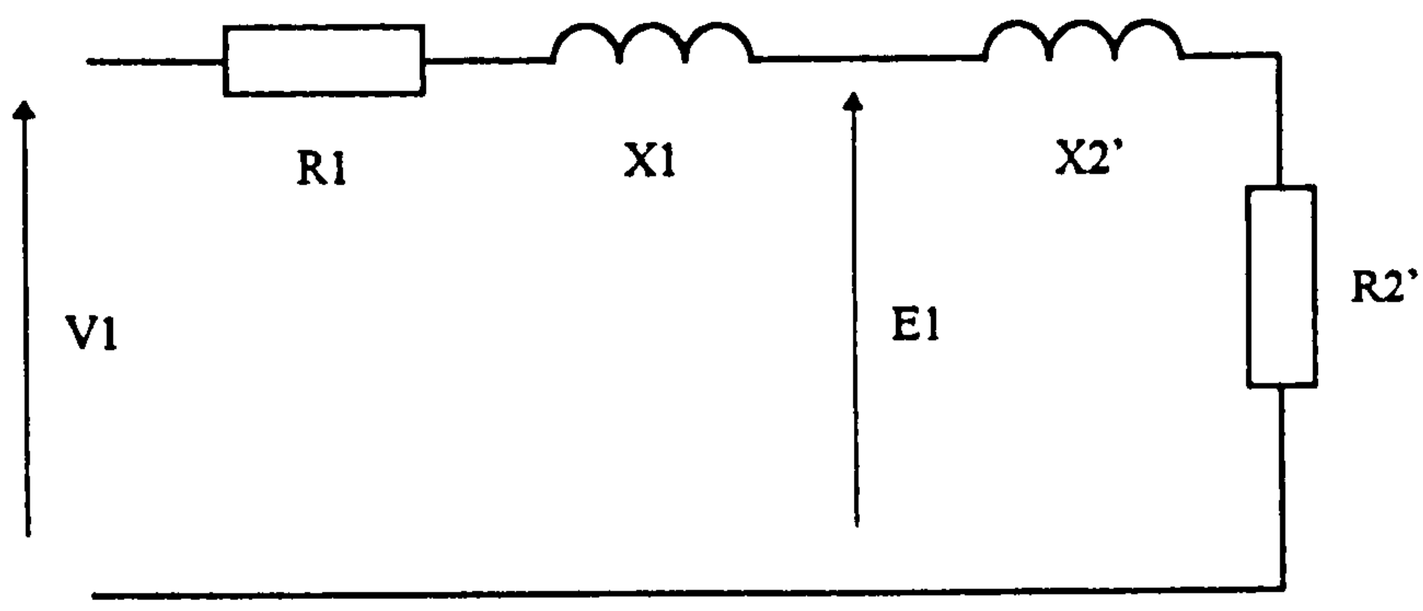


Figure 2.5.2: Locked Rotor Equivalent Circuit

It can be seen from Fig. 2.5.2 above that the reactance measured at the stator terminals for the locked rotor test is equal to the sum of the stator and rotor leakage reactance s.

$$X_{locked} = X_1 + X_2 \qquad (2.5.8)$$

The IEEE Test Procedure recommends that the locked rotor reactance can be evenly distributed between the stator and rotor leakage reactance s for a class A motor. The motor under test is a class which are defined as machines with normal starting current and torque, therefore:-

$$X_1 = X_2 = \frac{X_{locked}}{2} \qquad (2.5.9)$$

The locked rotor reactance was calculated from measurement of the locked rotor impedance and the locked rotor resistance.

$$Z_{locked} = \frac{V_{locked}}{I_{locked}} \tag{2.5.10}$$

$$R_{locked} = \frac{P_{locked}}{3I^2_{locked}} \tag{2.5.11}$$

$$X_{locked} = \sqrt{Z^2_{locked} - R^2_{locked}} \tag{2.5.12}$$

$$Z_{locked} = \frac{44.9}{5.8} = 7.74\Omega \tag{2.5.13}$$

$$R_{locked} = \frac{405}{100.92} = 4.013\Omega \tag{2.5.14}$$

$$X_{locked} = \sqrt{59.908 - 16.104} = 6.618\Omega \tag{2.5.15}$$

$$X_1 = X_2 = \frac{6.618}{2} = 3.309\Omega \tag{2.5.16}$$

The magnetising reactance can now be calculated by:-

$$X_m = X_{no-load} - X_1 = 70.983 - 3.309 = 67.674\Omega \tag{2.5.17}$$

The rotor resistance now remains to be calculated, the resistance calculated for the locked rotor test R_{locked} is a sum the stator resistance and a resistance R which is the combination of $R_2 + jX_2$ in parallel with jX_m .

$$R_{locked} = R_1 + R \tag{2.5.18}$$

$$R = R_2\left(\frac{X_m^2}{R_2^2 + (X_2 + X_m)^2}\right) \tag{2.5.19}$$

The above equation can be simplified if the self reactance of the rotor is much greater than the resistance of the rotor, [*Fitzgerald*, 1990]

$$R \approx R_2\left(\frac{X_m}{X_2 + X_m}\right)^2 \qquad (2.5.20)$$

$$R_2 = (R_{locked} - R_1)\left(\frac{X_2 + X_m}{X_m}\right)^2 \qquad (2.5.21)$$

The stator resistance was measured directly as 2.39Ω, therefore the rotor resistance could be calculated as:-

$$R_2 = 1.623 \times 1.1 = 1.79\Omega \qquad (2.5.22)$$

The parameters determined from the preceding tests are as follows:-

$$Lm = \frac{X_m}{2\pi f} = \frac{67.674}{314.159} = 215.413mH \qquad (2.5.23)$$

$$L_1 = \frac{X_1}{2\pi f} = \frac{3.309}{314.159} = 10.533mH \qquad (2.5.24)$$

$$L_2 = \frac{X_2}{2\pi f} = \frac{3.309}{314.159} = 10.533mH \qquad (2.5.25)$$

$$R_1 = 2.39\Omega \qquad (2.5.26)$$

$$R_2 = 1.79\Omega \qquad (2.5.27)$$

2.6 Performance of Induction Motor Model

The Electrical and mechanical equations which have been derived in the preceding sections can be implemented in a computer simulation of the machine. The differential equations can be solved in a time stepping fashion using mathematical techniques. The main objective of this part of the project was to see how possible real-time simulation is with present hardware. The manner in which the equations of the motor model are solved will obviously have a direct impact on the accuracy and execution speed of the simulation. Therefore it was necessary to make a good choice for the time stepping algorithm which solves the equations. A choice was made early on to try only explicit methods which do not require solution of simultaneous equations at each time step. This can be seen to be justified if one notes that the time step inevitably has to be small to follow digital sampling which loses the advantage of larger time steps offered by implicit methods. This chapter observes the performance of the motor model when executed in a standard PC and investigates the limitations of the PC in providing real-time simulation. As an initial step the two extremes of explicit solvers were examined in the guise of the simple Euler scheme and the more complex fourth order Runge Kutta algorithm. An investigation of how the time step length affects the accuracy of the two methods will be made. Also a measure of the relative execution speeds of the two methods will be made, and the time step length required to allow the model to be execute in real-time within the PC, stated. The two solving techniques were implemented in a C code program which was executed on an IBM compatible 486 33 MHz personal computer.

2.6.1 Steady State Performance

The motor model was tested in the steady state with three phase sinusoidal 240V rms. 50Hz volts applied to the stator and the load torque was zero. The simulations were allowed to run for a sufficient length of time for the machine to run

up to almost synchronous speed. The Runge Kutta solver was used in parallel with the Euler solver to allow a direct comparison of the two methods. The following results show the phase current which was calculated by both methods. The time step was varied to show what effect this would have on the accuracy of both mathematical solvers. The results of the steady state tests are given in Fig. 2.6.1.

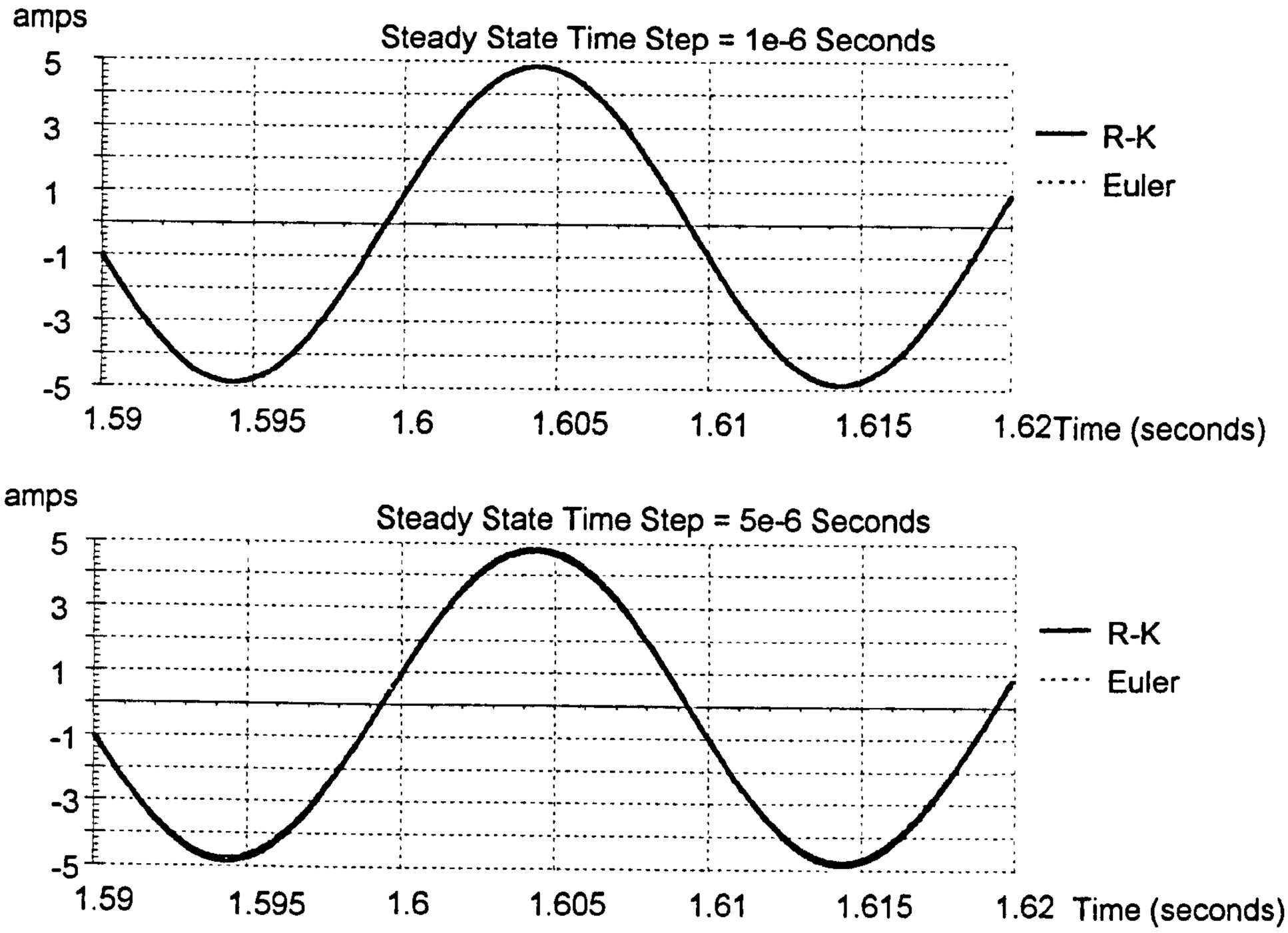


Figure 2.6.1: Steady State Comparison of Solvers

The following results show the actual phase voltage and the resultant phase currents when the real machine was connected directly to the three phase mains supply.

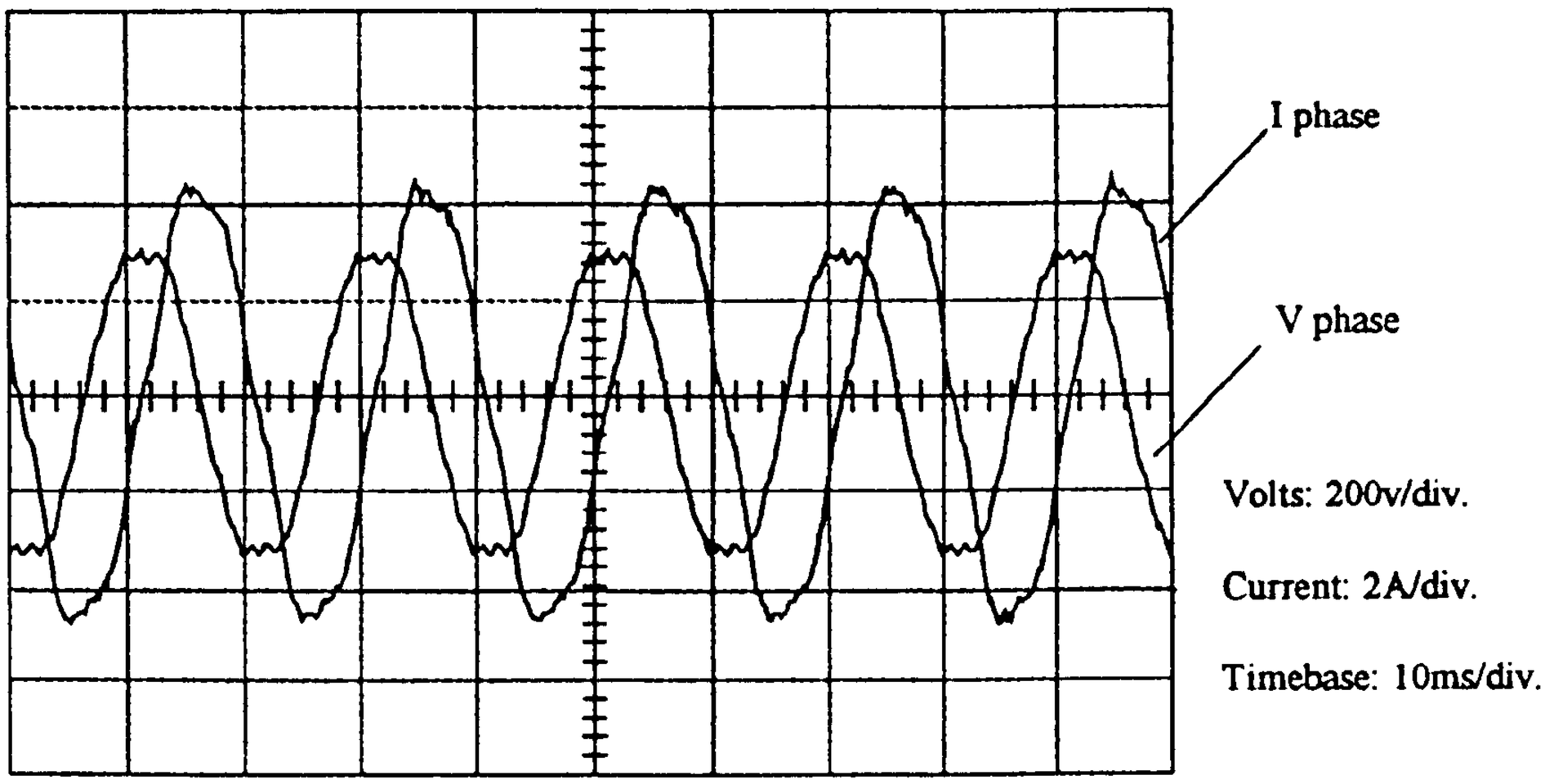


Figure 2.6.2: Actual Steady State Phase Voltage and Current

The calculated results below show the simulation of the same conditions using the Runge-Kutta algorithm with a time step of 50μs.

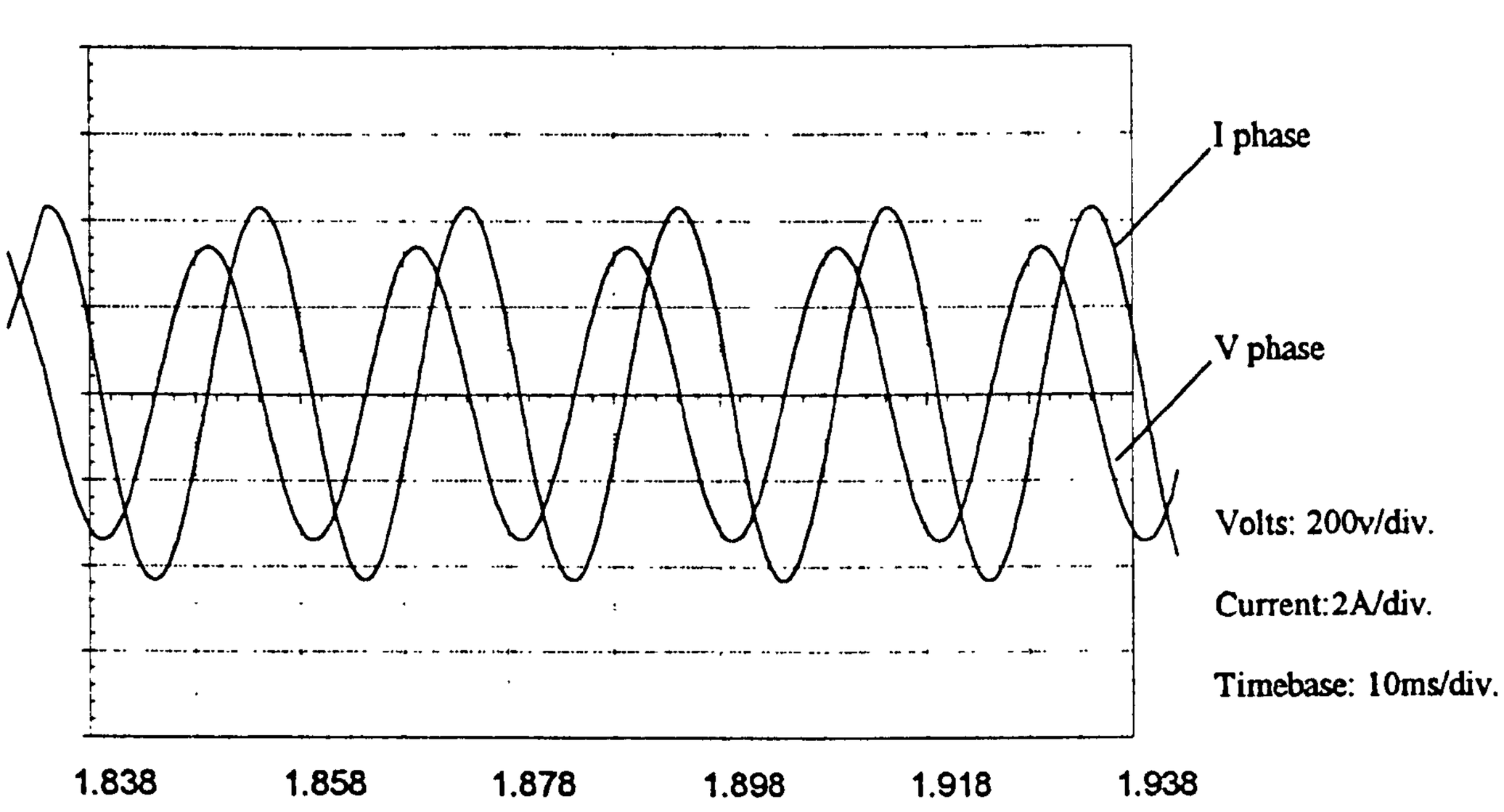


Figure 2.6.3: Simulated Steady State Phase Voltage and Current

Comparison of Fig. 2.6.2 with 2.6.3 show the calculation to be in good agreement with test give or take the harmonics ignored in the simulation.

2.6.2 Transient Performance

In order to observe the transient response of the two mathematical solvers the model, to begin with, was run as in the previous steady state tests at a constant stator voltage and no load torque, then a severe load torque of 50Nm was suddenly applied. This test is not one which would be carried out with the real machine since the magnitude of the load torque would cause the motor to stall. The test was again repeated with various solver time steps to observe and compare the two mathematical methods. The results of the transient tests are given in Fig. 2.6.4.

2.6.3 Execution Time

It is clear from the steady state and transient results that the Runge Kutta solver maintains its accuracy as the time step is increased, this is as expected as the Runge Kutta is a fourth order method as compared to the first order Euler method. The execution time for the two methods is compared in Table 2.6.1.

Number of Executions	Runge-Kutta Total Time (seconds)	Euler Total Time (seconds)	Runge-Kutta Time/Solution (μ seconds)	Euler Time/Solution (μ seconds)
6000	2.36	0.44	393.33	73.33
30000	12.16	2.49	405.33	83.00
60000	24.53	5.08	408.33	84.67
90000	36.86	7.67	409.56	85.22
120000	49.23	10.17	410.25	84.75

Table 2.6.1: Time Comparison of Solvers

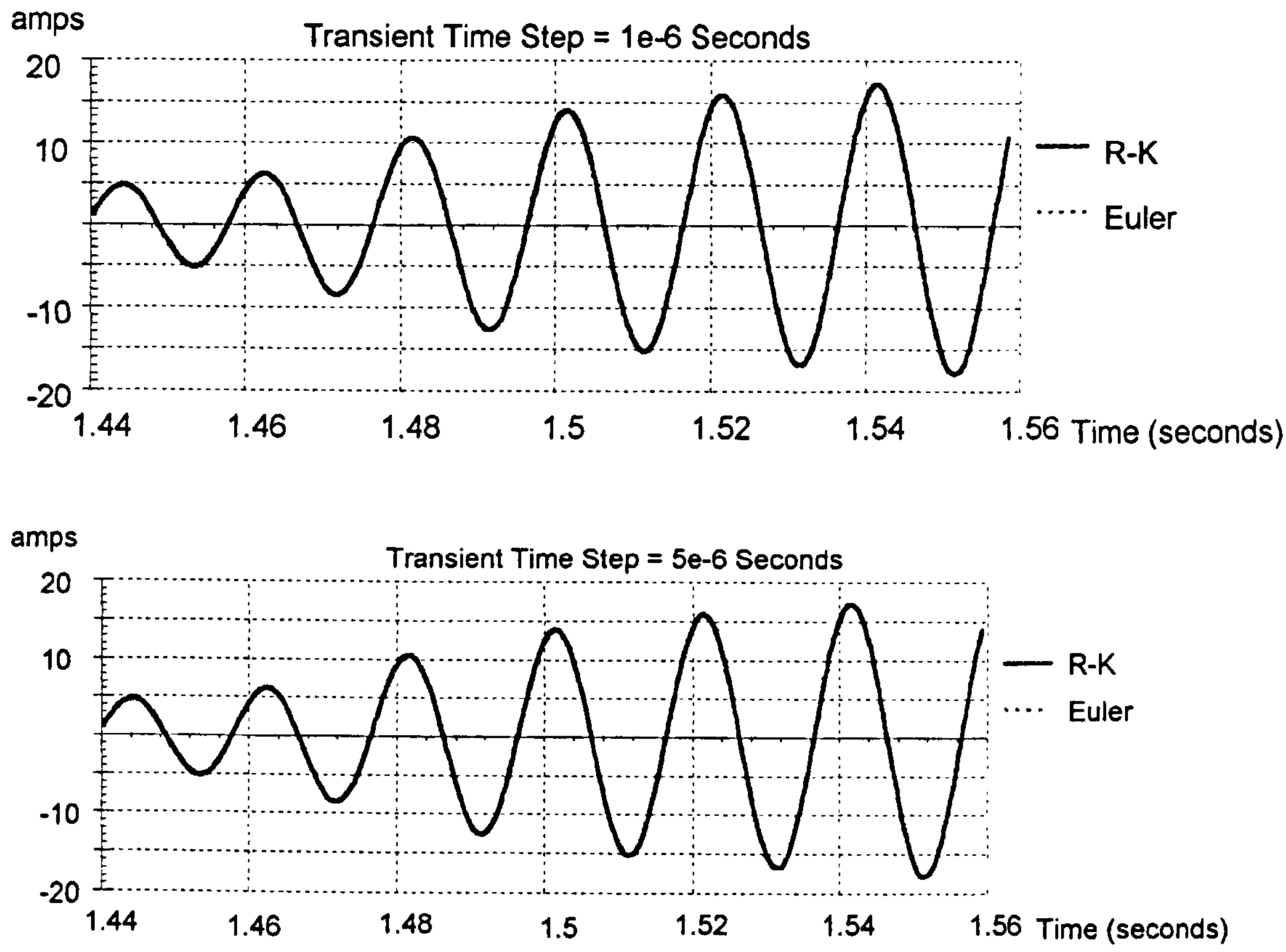


Figure 2.6.4: Transient Comparison of Solvers

It can be seen from the Table 2.6.1 that the average execution time for the Runge Kutta solver was 405.356μs and the average execution time for the Euler solver was 82.194μs. The Runge Kutta is therefore approximately five times slower than the Euler which is understandable when the mathematical methods are analysed. The Runge Kutta method solves the set of machine differential equations four times (a forth order solver) for each time step and then performs an averaging calculation based on the four solutions, the Euler method solves the differential equations just once.

A comparison can be made between the two solvers to see how close they approach real time simulation when run on a standard personal computer. The following table shows the execution time that each solver takes to perform 1 second of simulation for various time steps.

Time Step	Runge Kutta	Euler
1μs	6 min. 46 sec.	1 min. 22 sec.
5μs	1 min. 21 sec.	16.43 sec.
10μs	40.54 sec.	8.22 sec.
50μs	8.12 sec.	1.64 sec.

Table 2.6.2: Execution Time for 1 Second of Real Time

It is clear that for the Runge Kutta solver to simulate in real time it must have a time step of at least its execution time which is 405.356μs and similarly the Euler method must also have a time step equal to or greater than its execution time which is 82.194μs. The performance of the two solvers for these respective time steps is now examined. Three simulations were run, one using a Runge Kutta solver with a time step of 1μs to be used as a reference, one using a Runge Kutta solver with a time step of 406μs and finally one using a Euler solver with a time step of 83μs. The chart of Fig. 2.6.5 shows the results obtained.

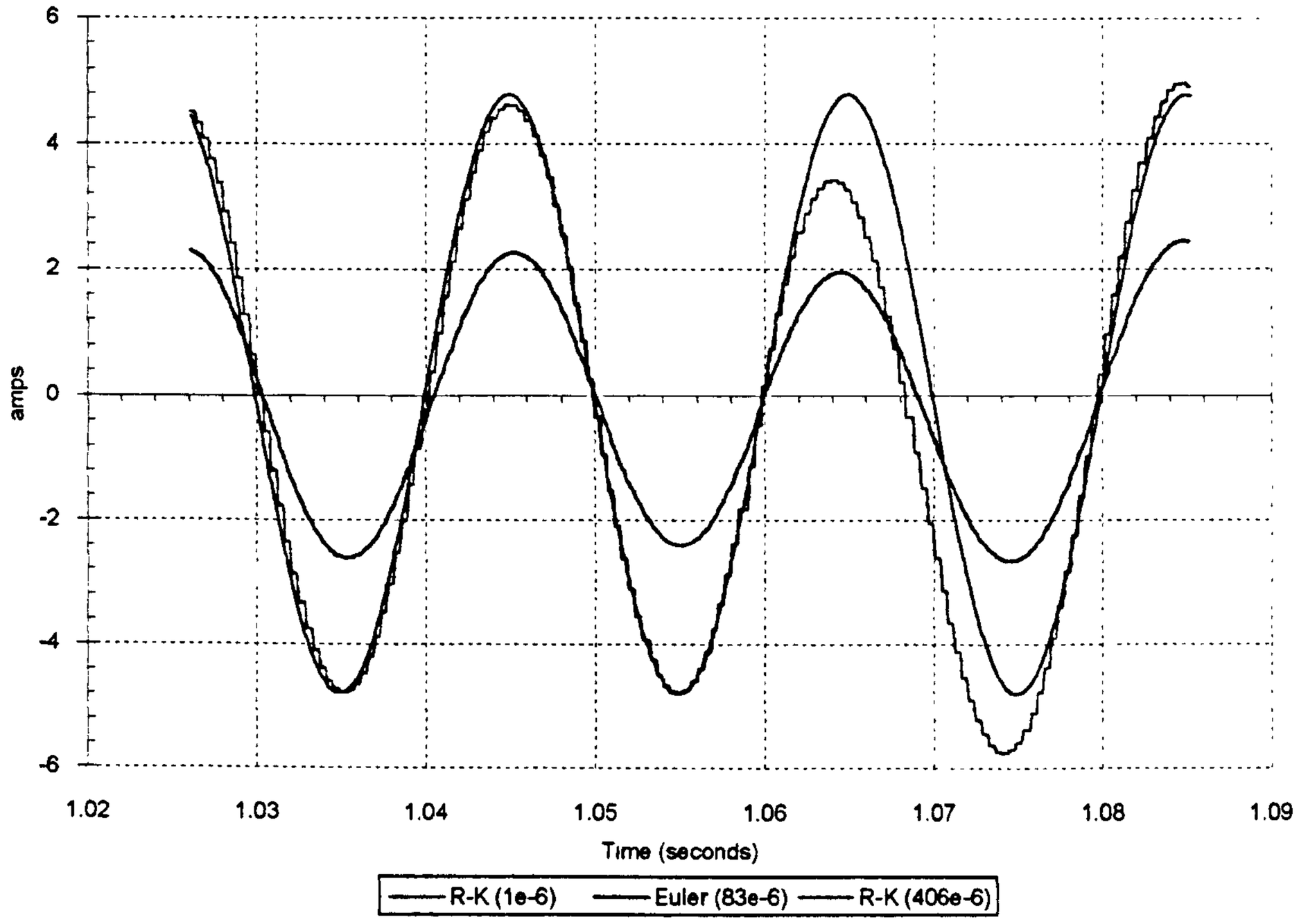


Figure 2.6.5 Performance of 'Real Time' PC Simulations

The above results clearly show that the personal computer simulations cannot approach real time simulation with any degree of accuracy. The simulation times given are only for the solution of the machine differential equations, if the system to be simulated is a more complete one, for example one which consists of a controller, an inverter and the motor, then the possibility of achieving real time simulation with a personal computer becomes even less likely.

An interesting recent publication by Moreno-Equilaz et. al. [*Moreno-Equilaz*, 1994] discussed the implementation of a real time vector controller in a standard 486 50MHz personal computer. The processor solved two differential equations for the magnetising current and the rotor flux angle which were required by the vector controller. Although the author was not attempting a full simulation of the drive system, the implementation within a PC of a controller in real time indicates the advances which are currently being made in the speed of personal computers. As personal computers become faster and more powerful how long will it be until the

processor can perform a Euler function to solve the machine equations in under 10μs, therefore making real time simulation a possibility within the PC?

2.7 The Commercial Option

The previous sections have all dealt with the behaviour of a dedicated C code program written specifically to simulate an induction machine. It is becoming increasingly common to use commercial simulation packages to quickly build up system simulations, one of the most popular is MATLAB. MATLAB when used in conjunction with SIMULINK provides a versatile simulation platform on which simulations can be built up quickly and easily in block diagram fashion. Results can be analysed using easily used built in maths functions and plotted using built graphing functions. This all leads to a user friendly environment in which blocks can be added, removed or modified with little effort. There is, however, one major drawback with these commercial packages and that is execution speed.

The following simulation block diagram was created from a drives simulation library, [*Slater and Armstrong*, 1995]

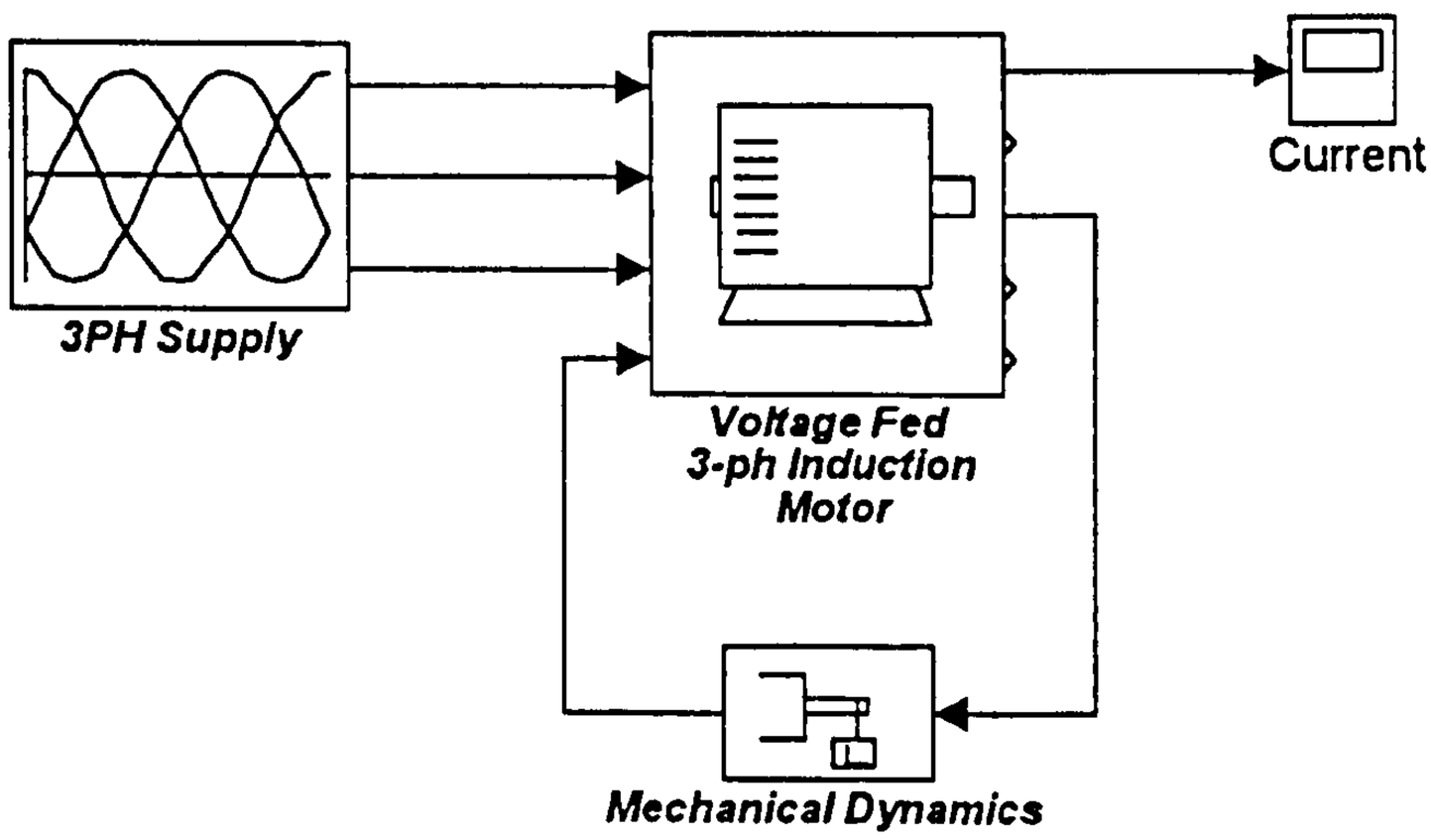


Figure 2.7.1: Matlab Simulation Block Diagram

The block diagram of Fig. 2.7.1 was simulated with Matlab, Matlab has various mathematical solvers provided and to compare its performance to the PC C code simulation a Runge Kutta fifth order and Euler algorithms have been used. The following results for the machine current was obtained using Runge Kutta with a time step of 50μs.

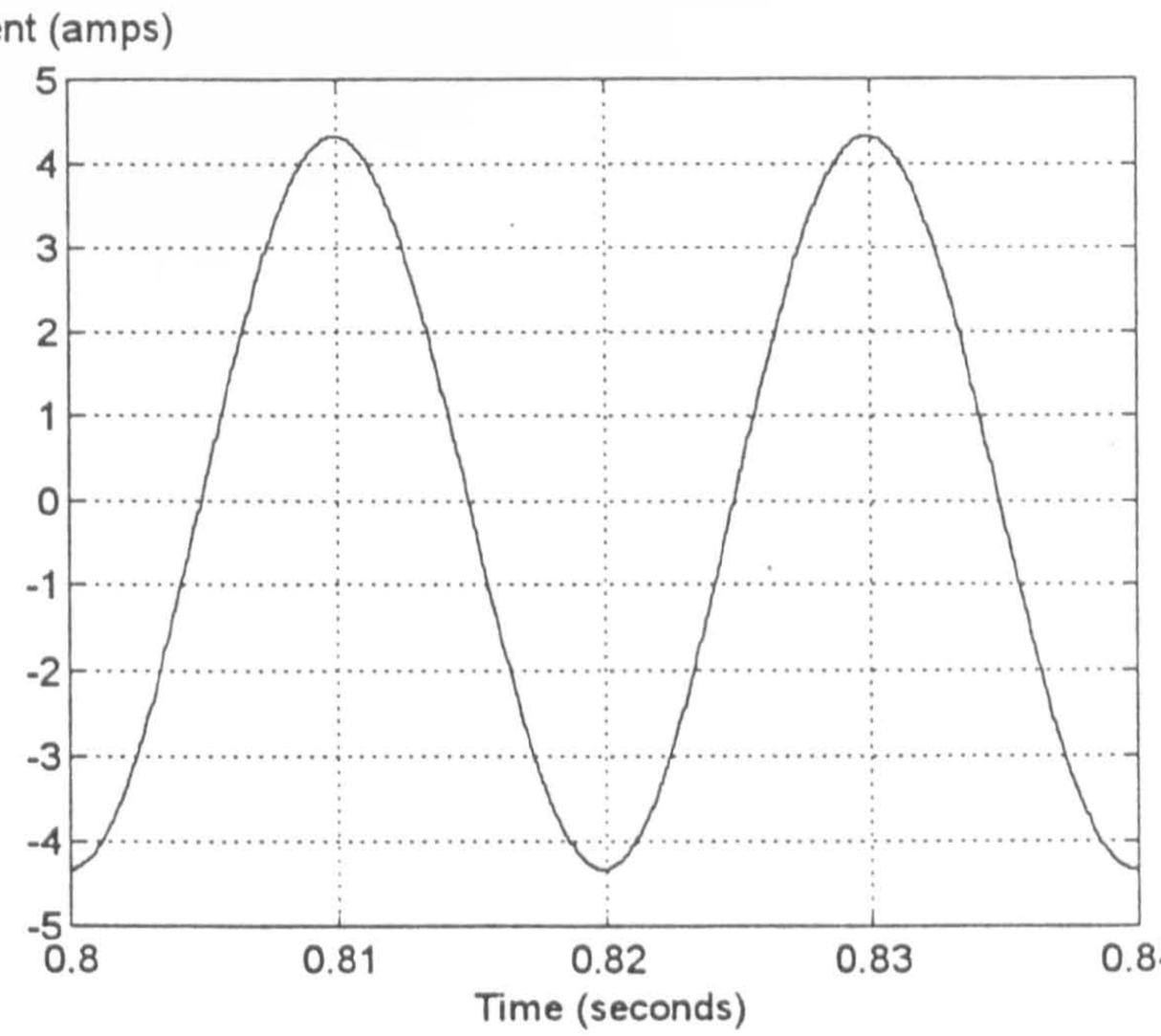


Figure 2.7.2: Matlab Simulation Current Waveform

The execution times for the Runge Kutta and the Euler methods were monitored for various time steps. The following Table 2.7.1 shows the execution times for Matlab to perform 1 second of simulation time, the times for the PC C code simulation given in Table 2.6.2 are repeated here for comparison.

Time Step	Runge Kutta Matlab	Euler Matlab	Runge Kutta PC	Euler PC
1μs	1 hour 21 min. 40 sec.	20 min.	6 min. 46 sec.	1 min. 22 sec.
5μs	16 min. 20 sec.	4 min.	1 min. 21 sec.	16.43 sec.
10μs	8 min. 10 sec.	2 min.	40.54 sec.	8.22 sec.
50μs	1 min. 38 sec.	24 sec.	8.12 sec.	1.64 sec.

Table 2.7.1: Execution Time for 1 Second of Real Time, Matlab Comparison

It can be seen from the previous timings that the Matlab simulation runs approximately 14 times slower for the Euler solver and approximately 12 times slower for the Runge Kutta solver. A recent publication [*Wade, Dunnigan and Williams*, 1994] about using Matlab to simulate a vector controlled induction machine indicates similar long execution times, in this publication the authors report a time of 20 minutes to simulate 0.3 seconds of real time using a 486 66 MHz PC with a time step of 100μs. This time step could only be used if no modulator was included in the simulation, i.e. the system simulation only included a model of the controller feeding sinusoidal demands to the motor model. If a model of a modulator was included then the authors had to reduce the time step to 2μs to maintain simulation accuracy. Increasing the time step to this level meant that the simulation time to perform 0.3 seconds of real time rose to a very disappointing 10 hours.

2.8 Summary

This chapter has developed the motor model equations, this motor model can now be included in the simulation of the overall drive system. The chapter reported the methods used to identify the parameters of the machine that are used in the model equations and also investigated the simulation time of the model when the equations were solved by both simple and complex mathematical methods.

The previous results have shown that the accuracy of the Euler solver decreases as the time step increases whereas the accuracy of the Runge Kutta solver remains reasonably high. The Euler method, being a first order solver, also suffers from the disadvantage that it is not unconditionally stable. All Runge Kutta methods are strongly stable [*Maron*, 1987]. For a non-real time PC simulation, the Runge Kutta solver is preferred to the Euler method. The investigation into the execution speed of both solvers showed that the dedicated PC software executed at a much faster rate than the commercial software simulation. This is expected because of the general nature of the commercial package. The execution speed of both solvers within the PC also showed that the Euler required a time step of at least 83μs, and the Runge Kutta a

time step of at least 406μs, to achieve real-time simulation within the PC. With these time steps both solving methods gave inaccurate results and at this stage only the motor model is included in the simulation; introducing a modulator would further reduce the allowable time step. A further investigation will be made into which solver is best suited to achieving real-time simulation within a multiple processing environment. This will be dealt with within the real-time simulation section, Chapter 6 of this thesis.

3. INVERTER SIMULATION

3.1 Introduction

The Induction Machine is by nature a constant speed machine, the speed of the machine is directly proportional to the frequency of the voltage applied to its stator windings. In order to control the machine efficiently it must therefore be supplied by a variable frequency supply. The object of the inverter is to provide this variable frequency supply, the demand for which comes from the drive system controller. The heart of the inverter is a network of power switching devices usually connected in a bridge formation which can be controlled to apply the desired voltage to the machines terminals. It has taken the advancement of power electronic switching technology to allow the Induction Machine to be used in variable speed applications. The development of power electronic converters has been centred on the advances made in the field of semiconductor switching devices. As faster switching devices are developed, capable of switching more power, their popularity increases and hence their cost reduces. The development and availability of converters has meant that the Induction Machine is a viable option in many variable speed applications.

The complexity of modern drives and their controllers, makes it necessary to simulate the drive system to allow experimentation with control algorithms, drive design etc. The motor simulation was discussed in the previous chapter of this thesis, this chapter will investigate the simulation of a three phase voltage source inverter which provides the interface between the controller and the motor.

A standard three phase voltage source inverter is represented in schematic form below in Fig. 3.1.1

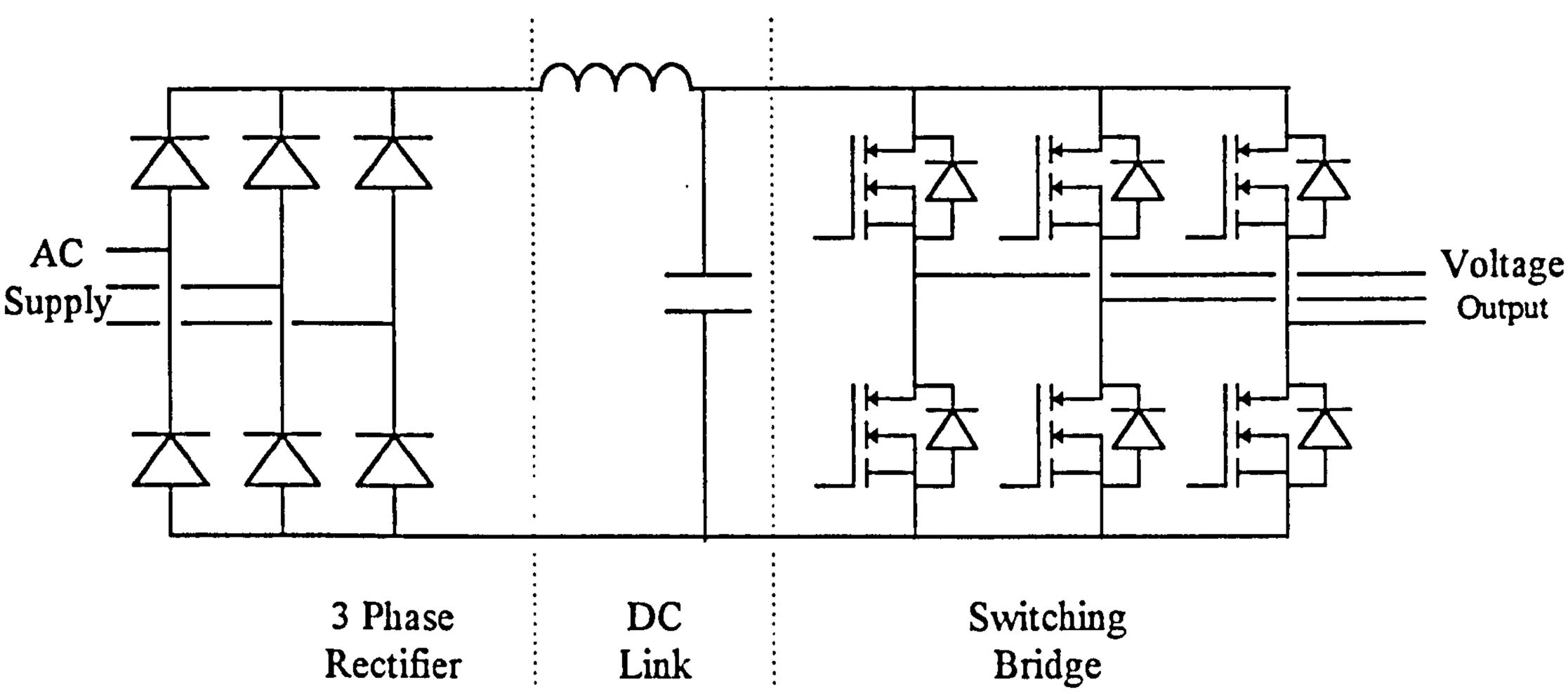


Figure 3.1.1: Three Phase Inverter Schematic

To enable the verification of the simulation software developed during the project a three phase voltage source inverter was designed and constructed. The inverter was based on IGBT switching devices and designed to operate at up to 20KHz switching frequency. The drive was processor controlled and contained a PWM ASIC for modulation. Details of the inverter construction are included in the Appendix of this thesis.

3.2 Modulator

To control the induction machine it must be supplied with a voltage of variable magnitude and variable frequency, ideally this voltage would be sinusoidal. The inverter applies the variable voltage by means of high frequency switches which pulse a DC voltage onto the motor terminals in such a manner as to try and replicate the desired sinusoidal demand. The object of the modulator is to provide the switching device signals given the sinusoidal demands from the controller. Two forms of modulation technique are commonly used, sine-triangle pulse width modulation and space vector modulation.

3.2.1 Sine-Triangle PWM

In sine-triangle modulation the demand signal from the controller is modulated by comparing it to a triangular signal which is also known as the carrier waveform. The intersection points of the two waveforms are then used as the switching moments for the bridge devices. Each phase of the inverter has its own demand signal, a single carrier signal can be used for all three phases and the comparison of this carrier with the phase demand signal determines the upper and lower device switching time, i.e. if the demand signal is greater than the triangle waveform then turn on the upper device and if the demand signal is lower than the triangle waveform turn on the lower device.

Fig. 3.2.1 shows the triangle waveform and one phase demand signal, the resultant switching signal for that phase is shown in Fig. 3.2.2. When the switching signal is high the upper device will be on and when the switching signal is low the lower device will be on. The resulting line voltage will switch between positive and negative DC link voltage and is shown in Fig. 3.2.3. The phase voltage will switch between 2/3 and 1/3 of the DC link voltage, this is shown in Fig. 3.2.4.

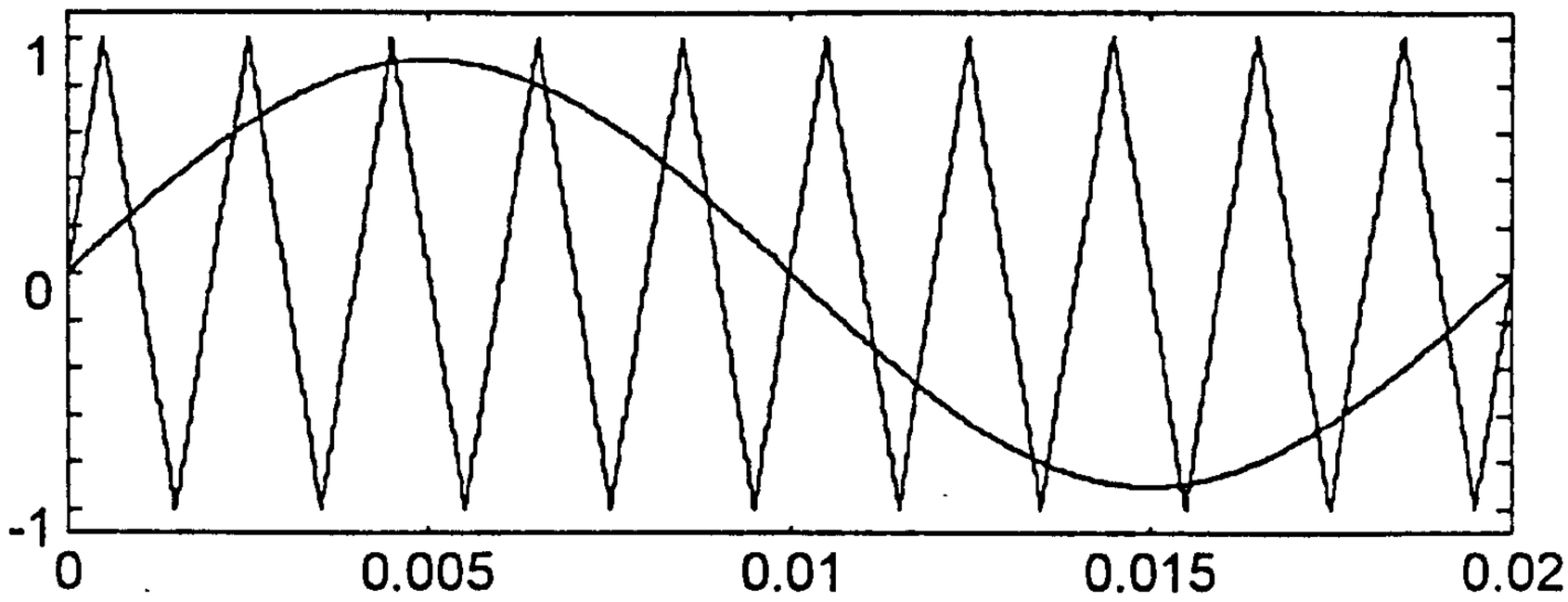


Figure 3.2.1: Sine-Triangle PWM

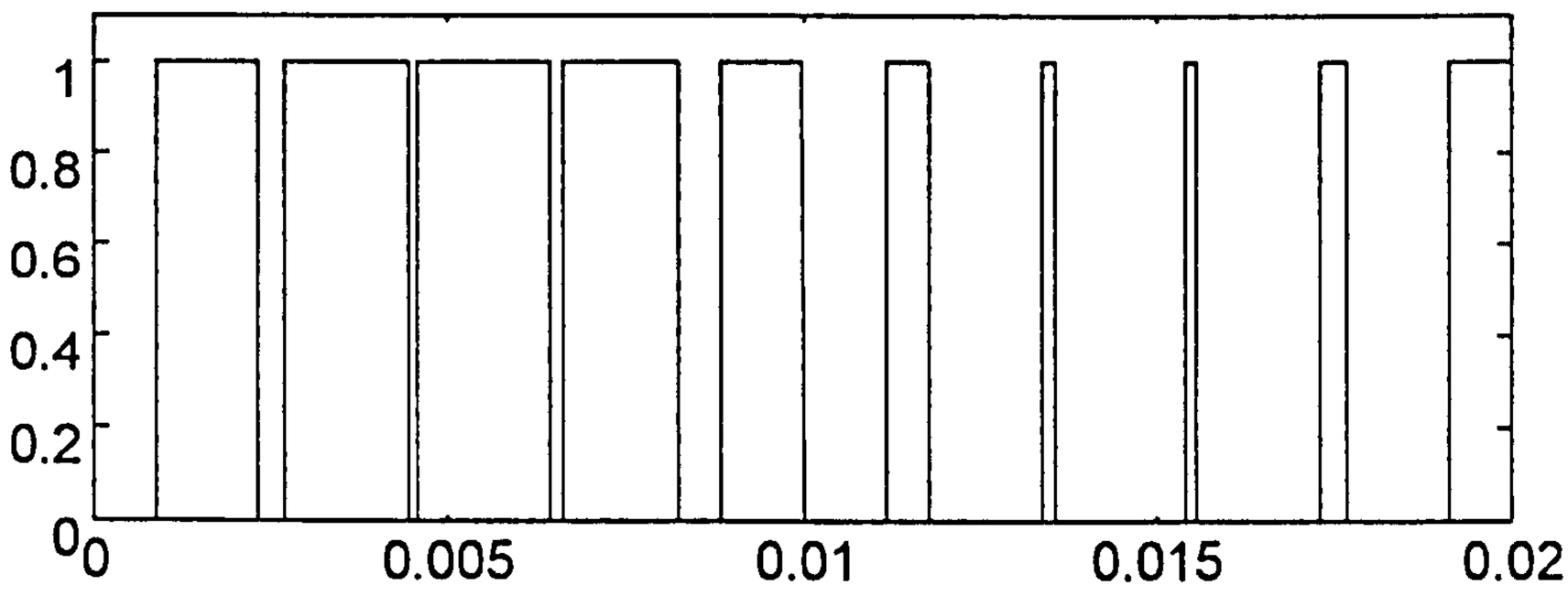


Figure 3.2.2: Sine-Triangle PWM Switching Signal

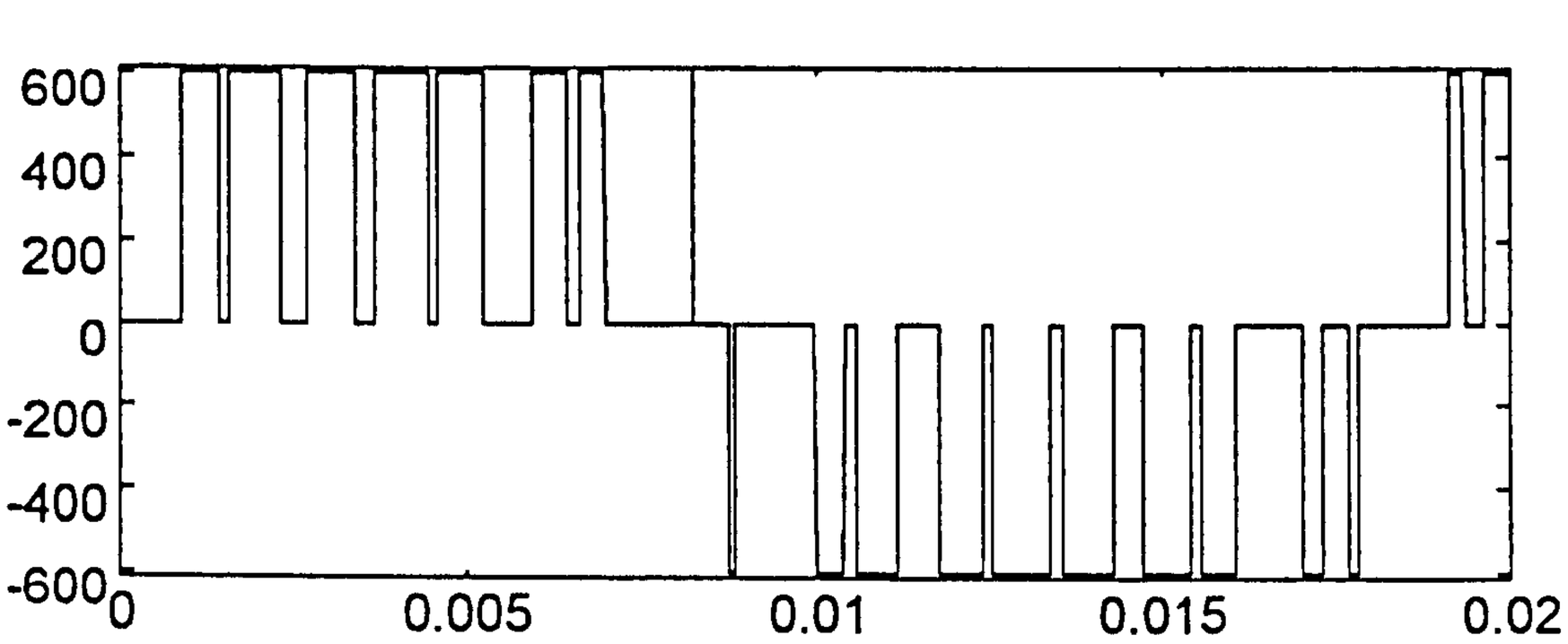


Figure 3.2.3: Sine-Triangle PWM Line Voltage

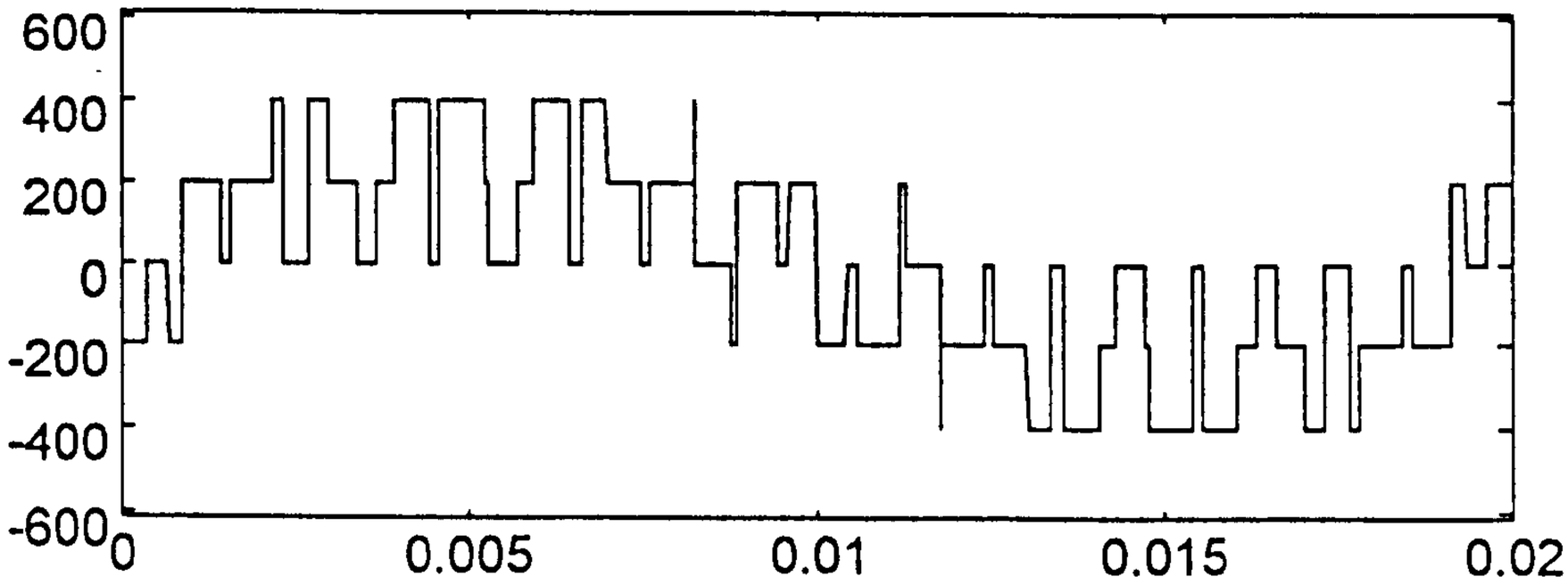


Figure 3.2.4: Sine-Triangle PWM Phase Voltage

3.2.2 Space Vector Modulation

Sine-triangle PWM is a continuous process, i.e. the demand is continuously compared to the carrier and at the moment of intersection the devices are switched. As digital drive control has become increasingly popular, software modulation techniques have been developed. One such technique is Space Vector Modulation. [*Handley and Boys*, 1992], for instance, reported on the practicalities of generating real-time, software based modulation waveforms using low cost microprocessors and they used the Space Vector Modulation method. Space Vector Modulation is an alternative to sine-triangle PWM in which the switching instants are pre-calculated for each switching period. A switching period is one cycle of the carrier waveform and is therefore equal to $1/f_{carrier}$.

Space Vector Modulation uses the three phase voltage demand as a vector which can be represented in two dimensions. The inverter bridge which the modulator is to control is capable of producing certain fixed voltage vectors. Space Vector Modulation decides which combination of these bridge vectors should be used to produce the required voltage vector.

Since there are only six switching devices that make up the inverter bridge, there are only a finite number of possible switching states. There are six possible 'voltage' states and two possible 'null' states. 'Voltage' states are those which result in a voltage being applied to the stator of the machine and 'null' states are those that do not result in a voltage being applied. The following Fig. 3.2.5 shows the eight possible switching configurations and the resultant voltage vector which they produce.

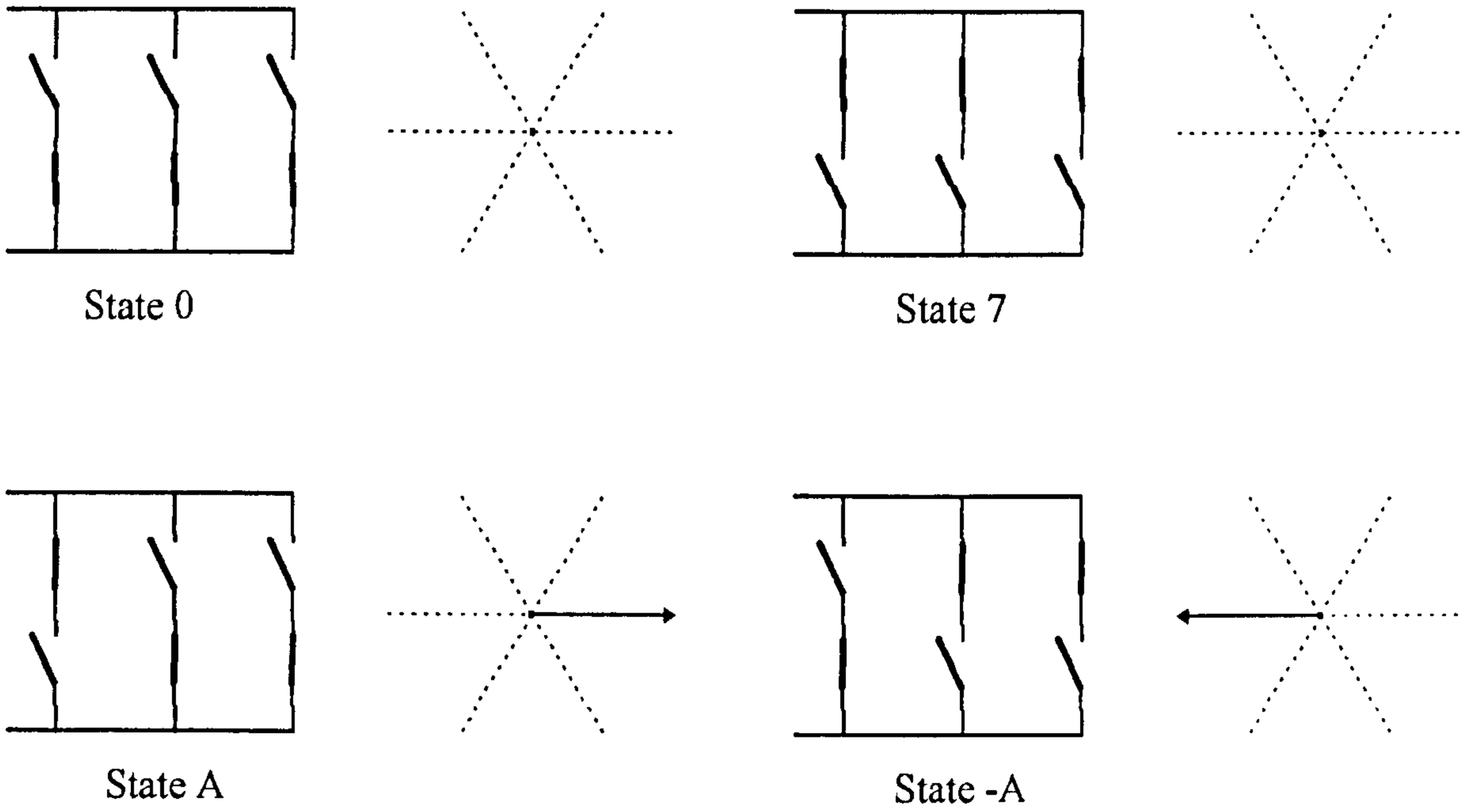


Figure 3.2.5: Bridge Switching States

Space Vector Modulation uses the voltage demand from the drive controller and regards it as a vector, since each switching state produces a fixed voltage vector any in between vector can be achieved by the combination of the two adjacent switching states. The modulator calculates which two 'voltage' switching states are required to produce the required voltage vector and how long these states need to be applied to produce the required vector. Fig 3.2.6 below shows a demanded voltage vector $\overline{V}$ and the switching vectors $\overline{V_A}$, and $\overline{V_{-C}}$ which would be applied to create it.

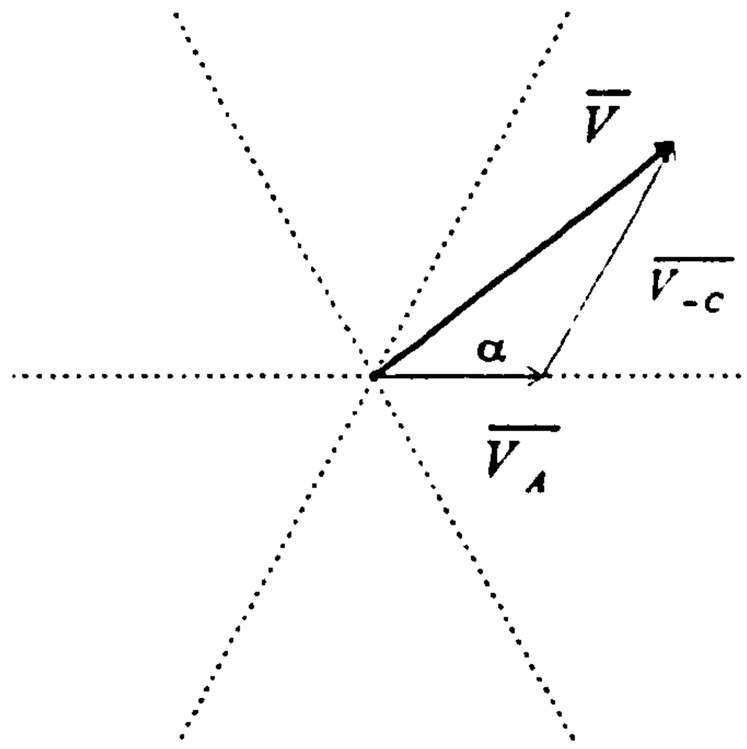


Figure 3.2.6: Demanded Voltage Vector

The modulator calculates which two switching states to apply and the duration of these states from the following equations:-

$$t_1 = \frac{3}{2}|V|T(\cos\alpha - \frac{1}{\sqrt{3}}\sin\alpha) \qquad (3.2.1)$$

$$t_2 = \sqrt{3}|V|T\sin\alpha \qquad (3.2.2)$$

$$t_0 = t_7 = T - (t_1 + t_2) \qquad (3.2.3)$$

where:-

$|V|$ is the magnitude of the demanded vector

α is the displacement angle of the demanded vector between two adjacent switching states.

T is half of the switching period

t_1 is the 'on' time for the first 'voltage' vector

t_2 is the 'on' time for the second 'voltage' vector

t_0 and t_7 are the times for the two 'null' vectors

The magnitude of the demanded vector $|V|$ is normalised into a per unit value where the DC link voltage Vdc is defined as 1 per unit.

Once the modulator has calculated which two switching states are to be applied during the switching period and how long each switching state is to be on for, it then implements them in the following pattern:-

$$\overline{V_0} \Rightarrow \overline{V_1} \Rightarrow \overline{V_2} \Rightarrow \overline{V_7} \Rightarrow \overline{V_2} \Rightarrow \overline{V_1} \Rightarrow \overline{V_0} \qquad (3.2.4)$$

where:-

$\overline{V_0}$ is the null vector with all lower devices on.

$\overline{V_1}$ is the first voltage vector with one upper and two lower devices on.

$\overline{V_2}$ is the second voltage vector with two upper and one lower device on.

$\overline{V_7}$ is the null vector with all upper devices on.

The above pattern enables Space Vector Modulation to be compared directly to sine-triangle PWM and highlights the equivalence of the two modulation techniques [*Handley and Boys,* 1992] . Consider one carrier cycle of a 5KHz triangle and three phase voltage demand references.

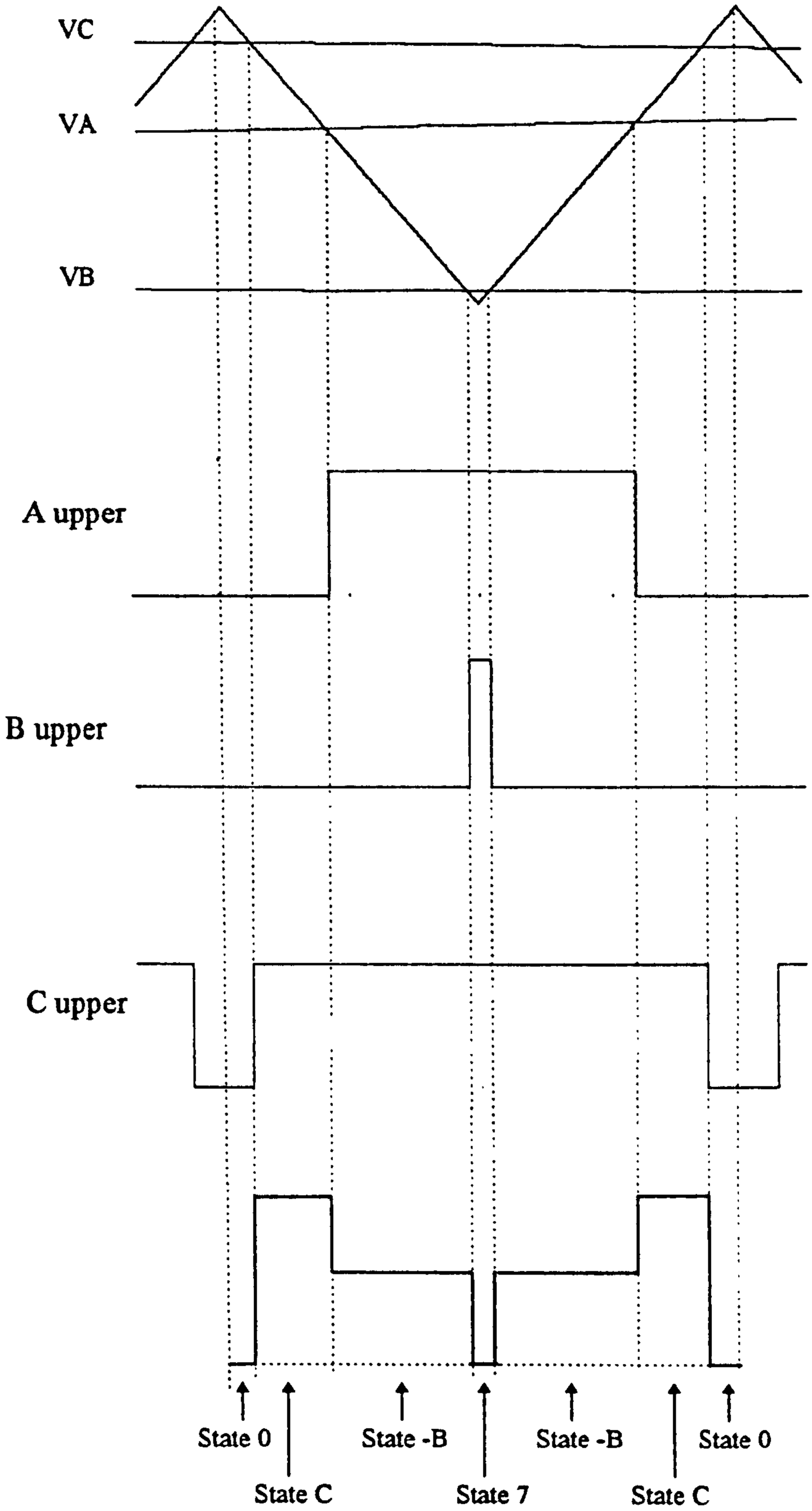


Figure 3.2.7: Sine-Triangle PWM, One Switching Period

If the same voltage demands were used by the Space Vector Modulator the resultant demand vector $\overline{V}$, which is made up of the three phase demands $\overline{V_A}$, $\overline{V_B}$ and $\overline{V_C}$, can be represented by two switching vectors. Figure 3.2.8 below shows the demand vector from the three phase voltage demands and the two switching vectors which can be used to achieve this demand.

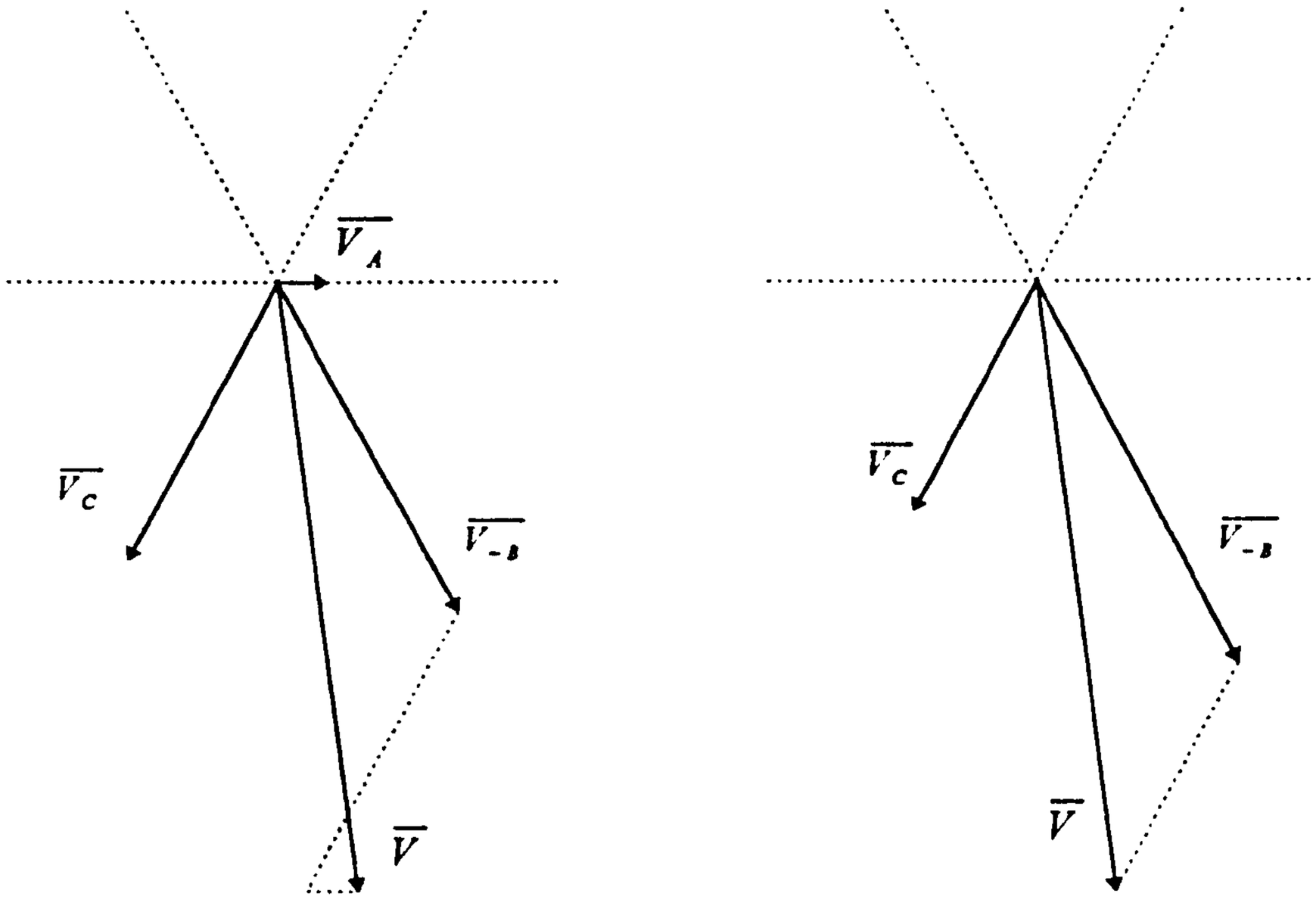


Figure 3.2.8: Space Vector Modulator, One Switching Period

The resultant switching pattern would therefore be:-

$$\overline{V_0} \Rightarrow \overline{V_C} \Rightarrow \overline{V_{-B}} \Rightarrow \overline{V_7} \Rightarrow \overline{V_{-B}} \Rightarrow \overline{V_C} \Rightarrow \overline{V_0} \qquad (3.2.5)$$

Comparison of the above switching pattern to that of the sine-triangle PWM pattern shown in Fig 3.2.7 clearly shows the similarity between the two modulation techniques.

Space Vector Modulation has a considerable advantage over sine-triangle PWM when it comes to simulating the two modulation techniques. Because sine-triangle modulation is continuous the switching instant could occur at any time, therefore the comparison between the carrier and the demand signal must be made as frequently as possible to ensure that the bridge is switched at the correct time. Or a search routine needs to be implemented to find the crossing point. This leads to either a very small time step or to extra search steps and therefore an excessive simulation time. This is why the publication [*Wade, Dunnigan and Williams,* 1994] reported the increase in simulation time from 20 minutes to 10 hours when the modulator was included in the simulation. With Space Vector Modulation the instants at which the bridge switching occurs are calculated at the start of each modulation period and so the time step can be, on average, larger.

3.2.3 PWM ASIC

The modulator simulated in the research carried out during the project was a commercial PWM ASIC (application specific integrated circuit). The actual ASIC was implemented in the inverter which was used throughout the project to validate the simulations. The use of ASICs for PWM is common amongst modern drives, it removes the modulation task from the processor which is carrying out the control, thus allowing more processor time to implement complex control algorithms.

The simulation of the ASIC consisted of calculating the switching times of the inverter bridge at the start of each switching period. The equations used to calculate the switching times were provided in the manufacturers data which was supplied with the ASIC. There were three equations to solve, one for each phase demand:-

$$T_a = (255 + V_a) \times \frac{switching\_period}{510} \qquad (3.2.6)$$

$$T_b = (255 + V_b) \times \frac{switching\_period}{510} \qquad (3.2.7)$$

$$T_c = (255 + V_c) \times \frac{switching\_period}{510} \qquad (3.2.8)$$

where:- *switching_period = 1/switching frequency*

V_a, V_b, V_c are the demanded phase voltages

The ASIC had nine, eight bit control registers which were used to program the various functions provided by the ASIC such as switching frequency (carrier frequency), deadtime length, voltage demands etc. Each of the voltage demands were in the form of an eight bit data register representing magnitude and a further bit representing polarity. The magnitudes could be in the range of 0-254.

The simulation of the PWM ASIC received the simulation equivalent of the same voltage demands as the actual ASIC, it then calculated the 'on' times of the devices for each phase from the above equations. From the magnitudes of the voltage demands the simulation decides which two switching states are to be applied during the switching period and from the 'on' times for each phase the duration of the switching periods can be calculated.

3.3 Inverter Bridge

The switching devices used in the real inverter are IGBTs, these devices have a turn-on time of approximately 0.5μs and a turn-off time of approximately 1.5μs. If the switching characteristics of these devices were to be modelled with any degree of accuracy then a time step would have to be chosen which was smaller than 0.5μs. One objective of the project is to see how achievable real-time simulation is with presently available hardware. The TMS320C40 processors used to perform the simulation have a single instruction cycle time of 50ns. This means that only 10 single cycle instructions could be performed in the 0.5μs turn-on period, this is unrealistic and so, for the purpose of the simulations carried out during this research, the switching devices were considered as ideal. The conclusion of this thesis will mention new processor technology which will could make the modelling of switching devices possible in real-time.

The simulation of the inverter bridge receives the states which are to be applied from the simulation of the modulator, it then applies the correct proportion of the DC link voltage to the motor model simulation.

The motor model, as has already been discussed, is formed in the stationary D-Q axis attached to the stator with the D axis aligned with the A phase axis of the actual three phase stator winding. The switching states which are possible were shown in Fig. 3.2e, the motor model requires the D axis voltage V_{ds} and the Q axis voltage V_{qs}, these two voltages can be obtained by applying the transformations discussed in section 2.3 to the three phase voltages. The three phase voltages for the six voltage switching states can be deduced from the bridge pattern and are shown below in Fig. 3.3.1.

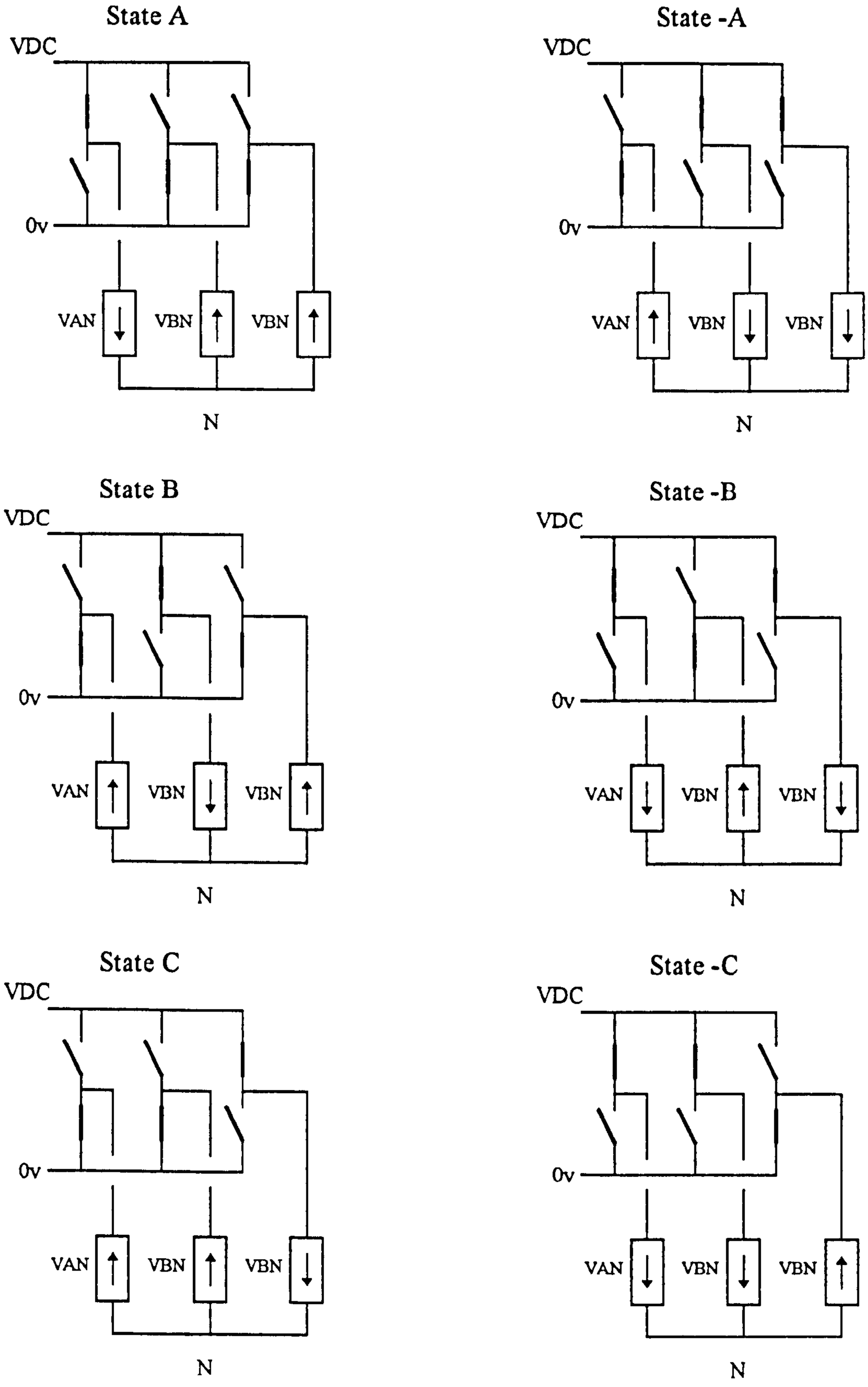


Figure 3.3.1: Phase Voltages for the Six Switching States

It can be seen from Fig. 3.3.1 that the phase voltages can be expressed in terms of the DC link voltage as follows:-

$$\text{State A} \Rightarrow V_{AN} = 2/3V_{DC} \quad V_{BN} = -1/3V_{DC} \quad V_{CN} = -1/3V_{DC} \quad (3.3.1)$$

$$\text{State -A} \Rightarrow V_{AN} = -2/3V_{DC} \quad V_{BN} = 1/3V_{DC} \quad V_{CN} = 1/3V_{DC} \quad (3.3.2)$$

$$\text{State B} \Rightarrow V_{AN} = -1/3V_{DC} \quad V_{BN} = 2/3V_{DC} \quad V_{CN} = -1/3V_{DC} \quad (3.3.3)$$

$$\text{State -B} \Rightarrow V_{AN} = 1/3V_{DC} \quad V_{BN} = -2/3V_{DC} \quad V_{CN} = 1/3V_{DC} \quad (3.3.4)$$

$$\text{State C} \Rightarrow V_{AN} = -1/3V_{DC} \quad V_{BN} = -1/3V_{DC} \quad V_{CN} = 2/3V_{DC} \quad (3.3.5)$$

$$\text{State -C} \Rightarrow V_{AN} = 1/3V_{DC} \quad V_{BN} = 1/3V_{DC} \quad V_{CN} = -2/3V_{DC} \quad (3.3.6)$$

Transforming these voltages to the twin axis D-Q reference frame gives values for V_{ds} and V_{qs} in terms of V_{DC}.

$$\text{State A} \Rightarrow V_{ds} = 2/3V_{DC} \quad V_{qs} = 0 \quad (3.3.7)$$

$$\text{State -A} \Rightarrow V_{ds} = -2/3V_{DC} \quad V_{qs} = 0 \quad (3.3.8)$$

$$\text{State B} \Rightarrow V_{ds} = -1/3V_{DC} \quad V_{qs} = (1/\sqrt{3})\, V_{DC} \quad (3.3.9)$$

$$\text{State -B} \Rightarrow V_{ds} = 1/3V_{DC} \quad V_{qs} = -(1/\sqrt{3})\, V_{DC} \quad (3.3.10)$$

$$\text{State C} \Rightarrow V_{ds} = -1/3V_{DC} \quad V_{qs} = -(1/\sqrt{3})\, V_{DC} \quad (3.3.11)$$

$$\text{State -C} \Rightarrow V_{ds} = 1/3V_{DC} \quad V_{qs} = (1/\sqrt{3})\, V_{DC} \quad (3.3.12)$$

The simulation of the inverter bridge can now simply use the state number from the modulator simulation and supply the correct D and Q axis voltage to the motor model which can be time stepped.

3.4 Dead Time

The previous sections arrived at models of the modulator and the inverter bridge. These models can now be compared to the performance of the actual inverter when connected to the actual motor. Consider the simulation of the inverter connected to the simulation of the motor, figure 3.4.1 shows the simulated motor phase current. It may be noted that the current ripple caused by the modulation has been effectively removed due to the high switching frequency and the motor's inductance.

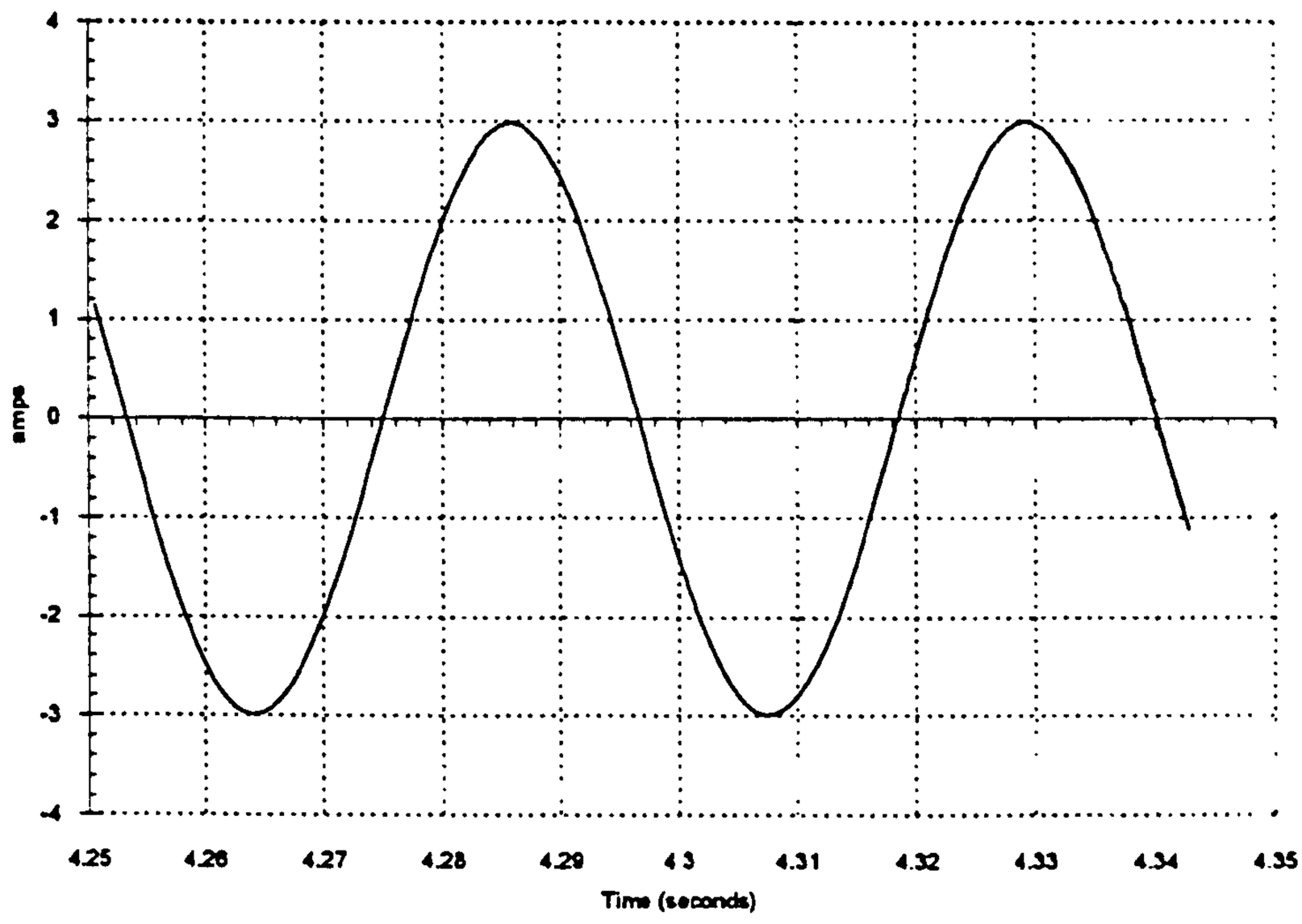


Figure 3.4.1: Simulated Motor Current

Now consider the actual system, figure 3.4.2 shows the actual motor phase current which results when the motor is fed from the inverter.

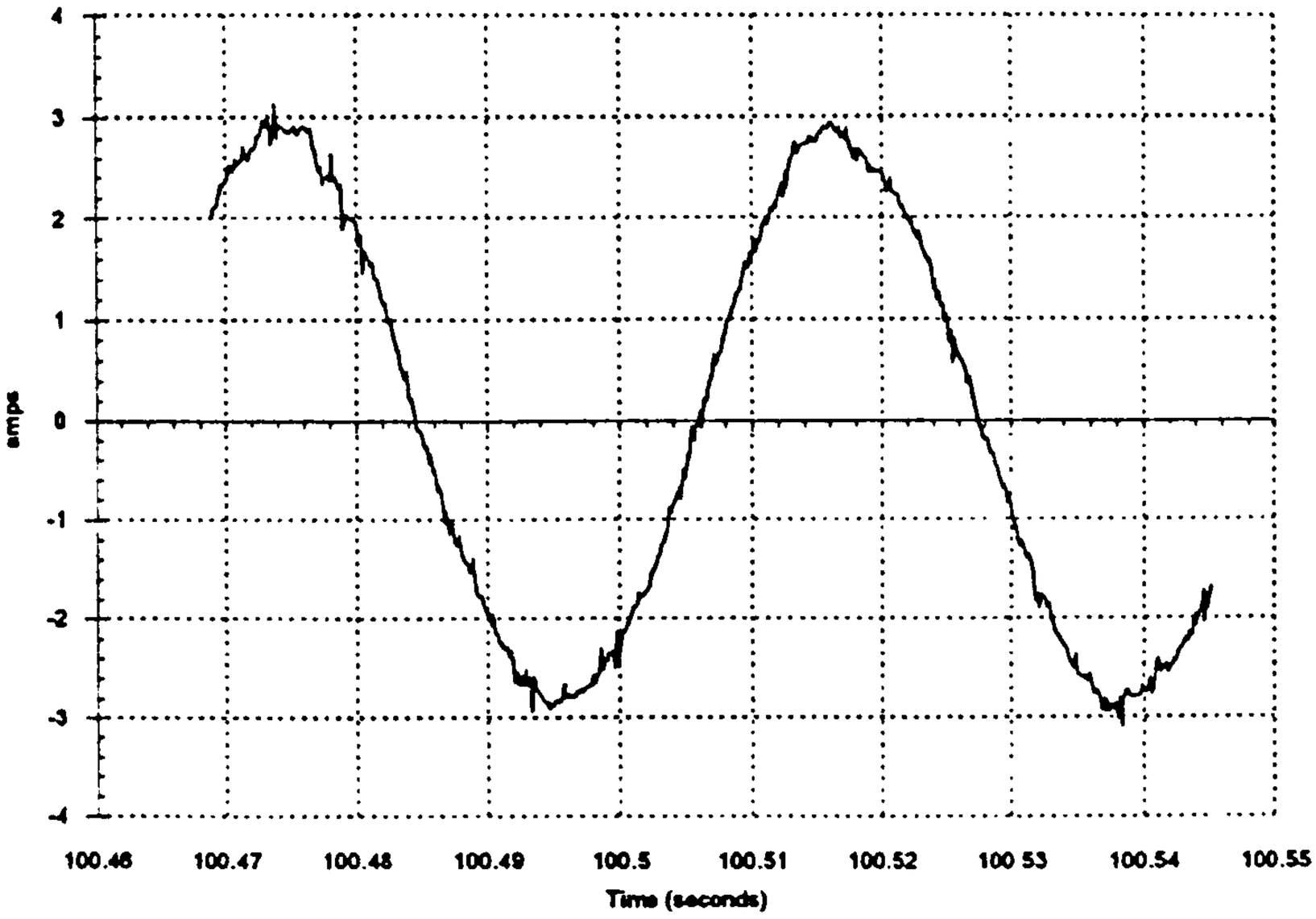


Figure 3.4.2: Measured Motor Current

Comparing the two waveforms shows some obvious difference with the simulation. In addition to the measurement noise there is some more consistent difference particularly in the region of the peaks and the zero crossing. In the simulation of the inverter bridge the switching devices were modelled as ideal and so one switch in a particular phase could switch on simultaneously as the other device in that phase switched off. In order to avoid a 'shoot-through' fault occurring in which both the upper and lower devices in a phase are on at the same instant resulting in a direct short circuit across the DC link, the actual modulator inserts a delay, this delay is known as a deadtime delay. The modulator delays the turn on of one device in a particular phase after the turn off of the corresponding device in the same phase, so for the deadtime period both the upper and lower devices in that phase are off. This delay is necessary because semiconductor switching devices are not ideal and take a finite time to turn on and off.

The delay in turning on the device results in the distorted current waveform. The distortion is due to the motor current flowing through the anti-parallel, or free-wheeling, diodes across the switching devices during the dead time period and effectively connecting the motor phase terminal to either the upper or lower of the DC

link. The direction of the current flow through the phase determines which rail the motor terminal connects to [*Mohan,Underland,Robbins*, 1989]. Consider the switching waveforms of Fig. 3.4.3. The ideal switching patterns of the upper and lower devices are shown together with the switching pattern which includes dead time. The turning on of the device is delayed by T_{dead} after the turning off of the other device. The result is the period when both upper and lower devices are off, this period is indicated by the shaded areas on the waveforms.

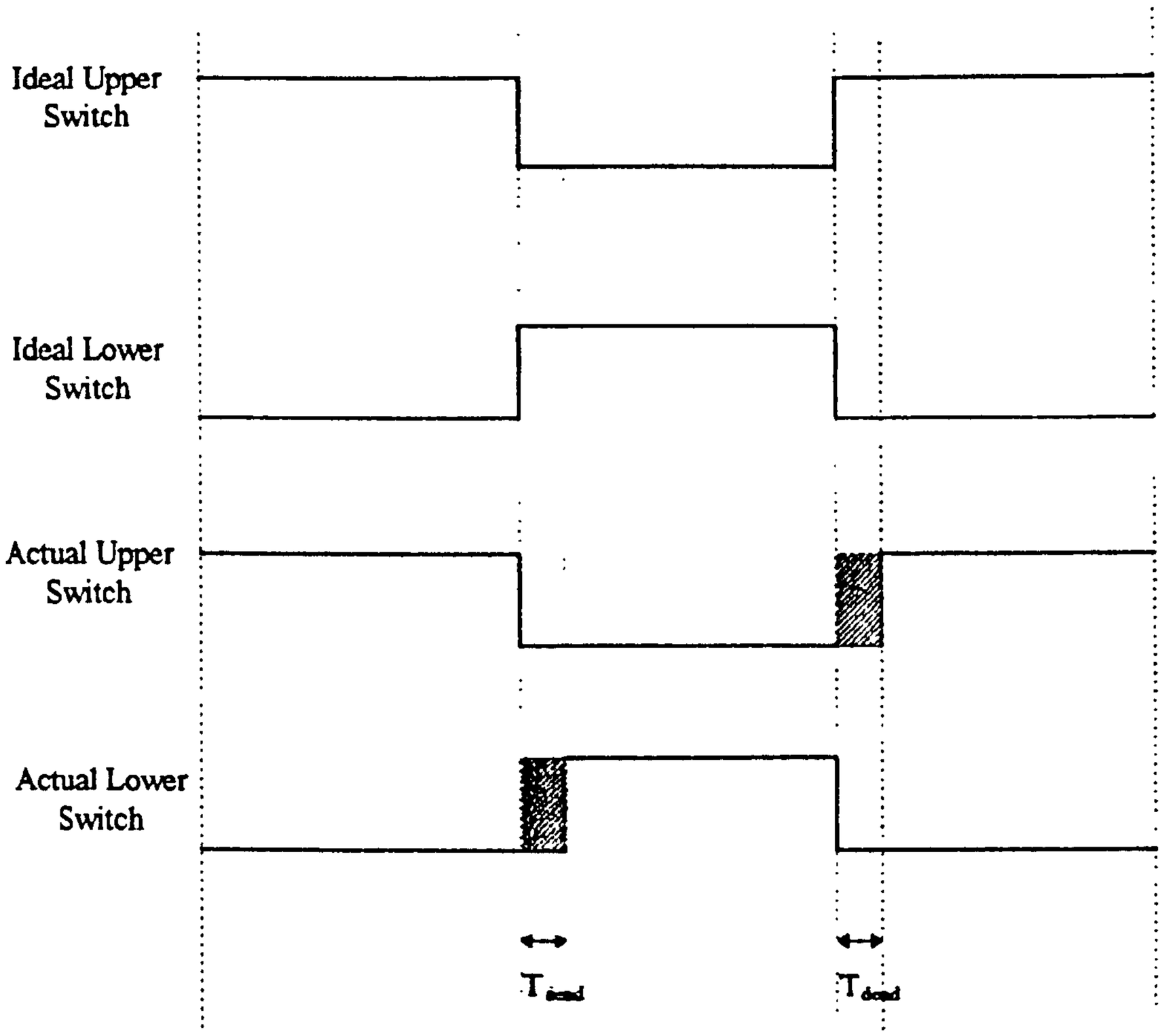


Figure 3.4.3: Ideal and Actual Device Switching Times

Consider now the current flow through the devices for the above switching pattern. Fig. 3.4.4a shows the upper device on and the lower device off , with the load current flowing in a positive direction through the upper device. The next switching instant requires the upper device to be turned off but the lower device not to be turned on until a time T_{dead} has elapsed, the current flow will remain in a positive direction and will therefore flow through the free-wheeling diode of the lower device, this is shown in Fig. 3.4.4b.

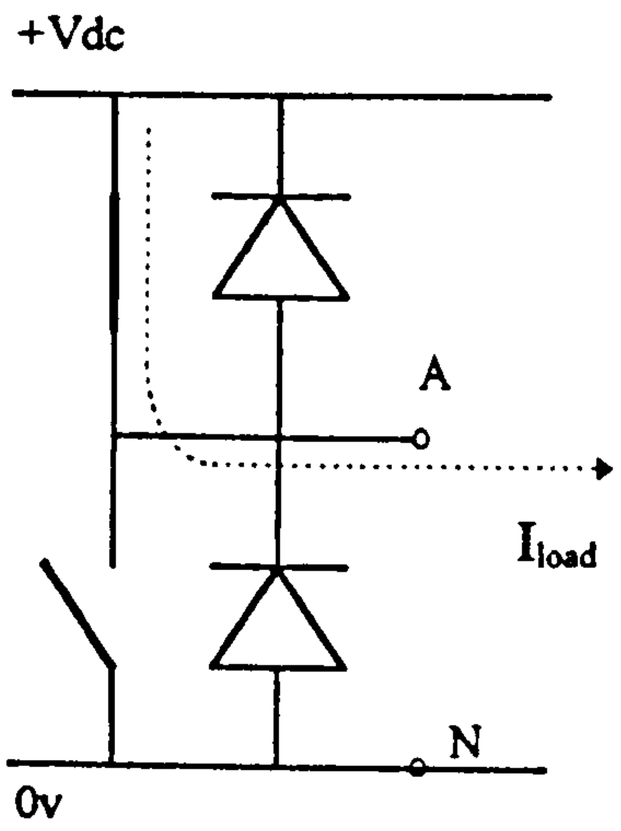


Figure 3.4.4a

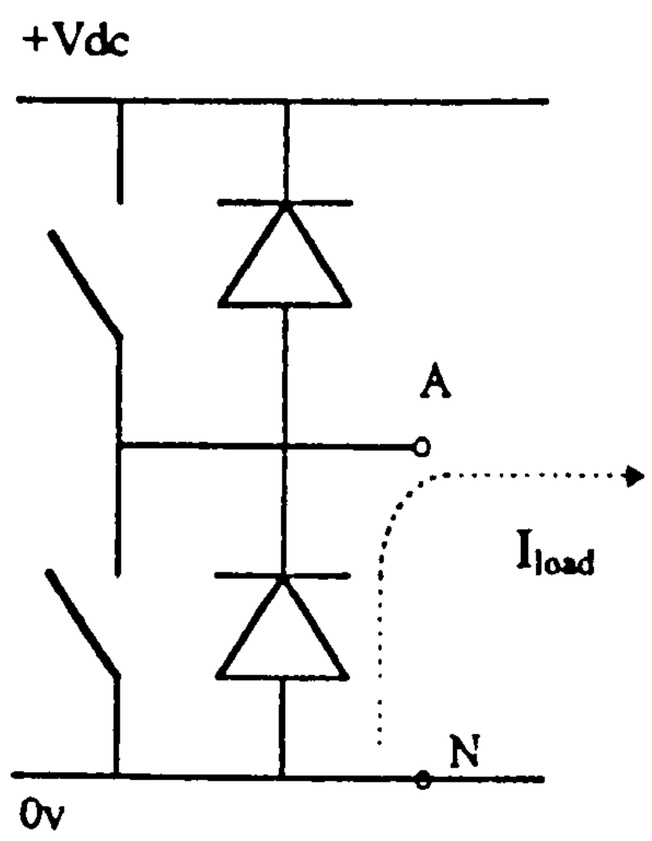


Figure 3.4.4b

The voltage V_{AN} will switch between +Vdc and 0v, this is the same voltage swing which would have occurred if the lower switch had been switched on simultaneously with the upper device switching off i.e. the dead time period does not change the effective switching pattern.

Consider the next dead time period, this occurs as the lower device is switched off and the turn on of the upper device is delayed by T_{dead}. Fig. 3.4.4c shows the lower device on and the upper device off, if the load current is considered to remain positive then it flows through the free-wheeling diode of the lower device. The next switching instant turns the lower device off but delays the turn on of the upper device. The load current will continue to flow through the free-wheeling diode of the lower device as shown in Fig. 3.4.4d.

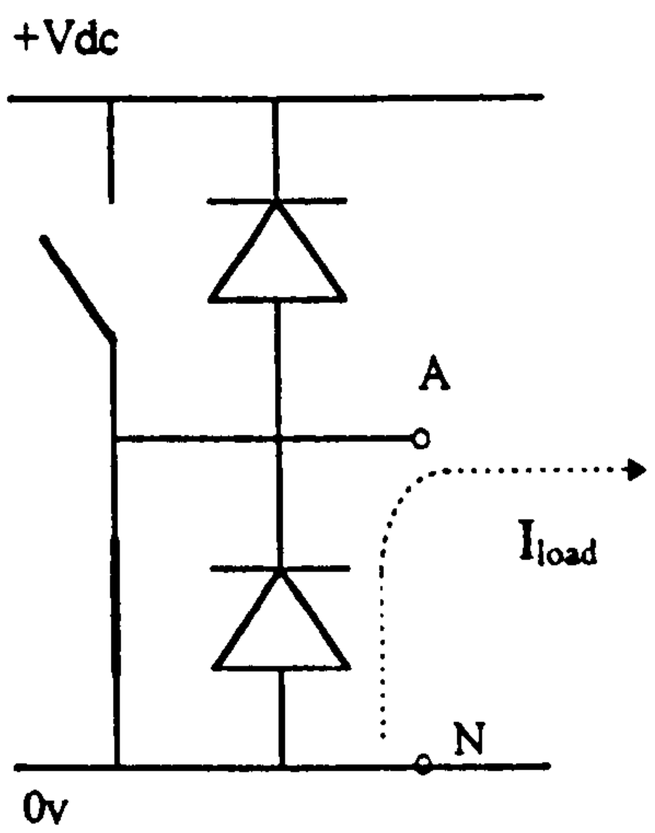


Figure 3.4.4c

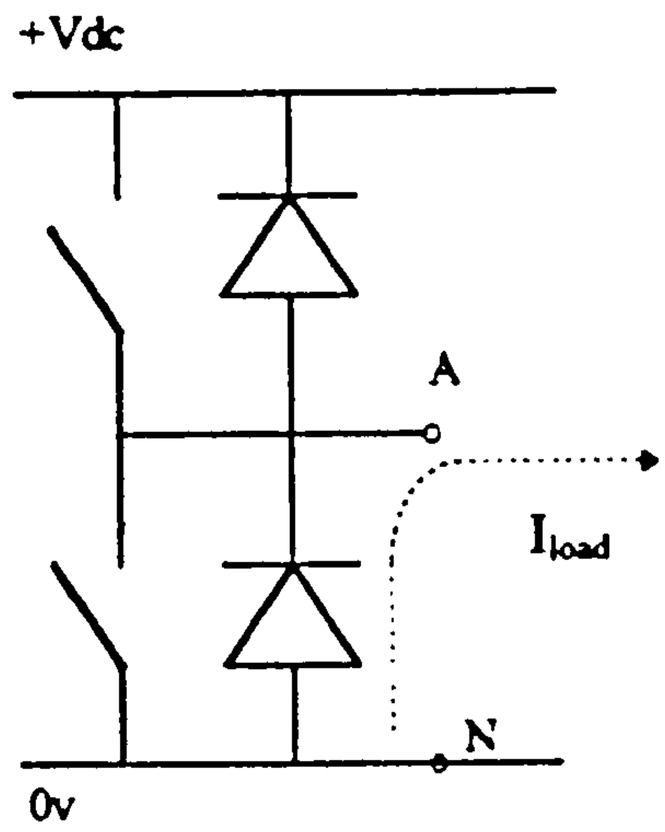


Figure 3.4.4d

Figure 3.4.4 Dead Time Current Flow, Positive Load Current

The voltage V_{AN} will remain unchanged at 0v. if however, the dead time delay had not been inserted then the upper device would have been turned on simultaneously with the lower device turning off and the voltage V_{AN} would have switched to +Vdc. Therefore the dead time delay during this period has effectively extended the length of time that the lower device is on for.

Consider now the current flow through the devices for the same switching pattern but with the load current flowing in a negative direction. Fig. 3.4.5a shows the upper device on and the lower device off with the load current flowing through the free-wheeling diode of the upper device. The first dead time period occurs as the upper device is turned off and the turning on of the lower device is delayed until a time T_{dead} has elapsed, the current flow will remain in a negative direction and continue to flow through the free-wheeling diode of the upper device, this is shown in Fig. 3.4.5b.

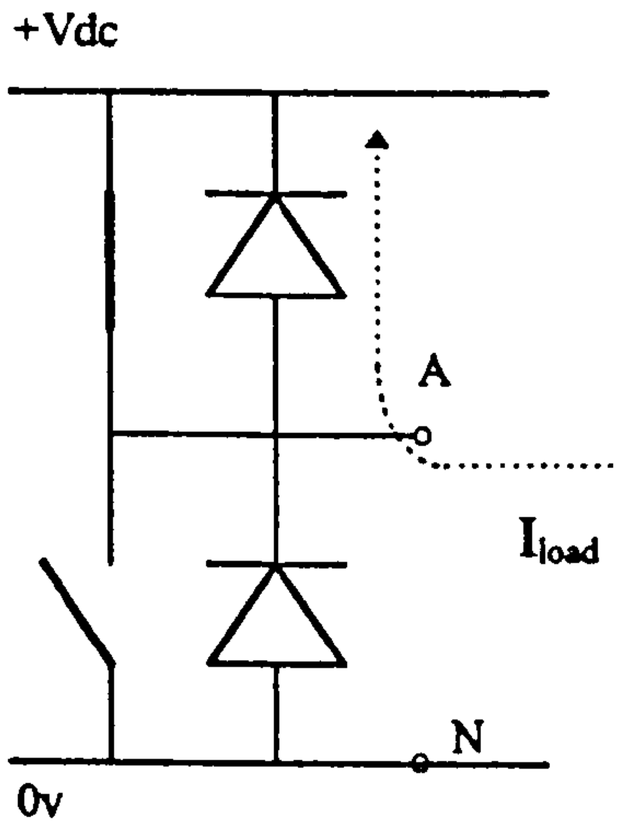


Figure 3.4.5a

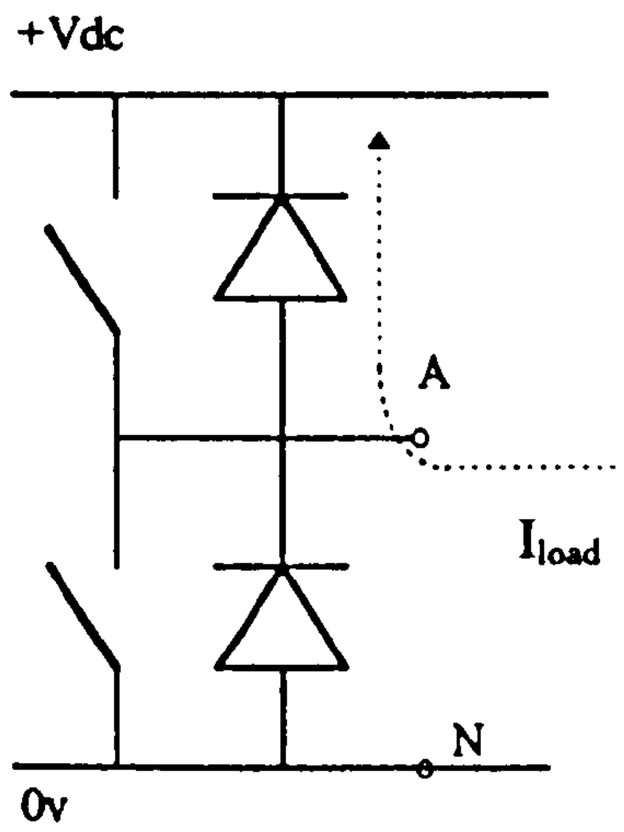


Figure 3.4.5b

The voltage V_{AN} will remain unchanged at +Vdc, if the dead time period was omitted and the lower device had been switched on at the moment the upper device switched off then the voltage V_{AN} would have switched instantly to 0v, the dead time period has therefore effectively extended the period of time that the upper device is on.

Consider the next dead time period, this occurs as the lower device is switched off and the turn on of the upper device is delayed by T_{dead}. Fig. 3.4.5c shows the lower device on and the upper device off, if the load current is considered to remain negative then it flows through the lower device. The next switching instant turns the lower device off but delays the turn on of the upper device. The load current will continue to flow in a negative direction but will now flow through the free-wheeling diode of the upper device as shown in Fig. 3.4.5d.

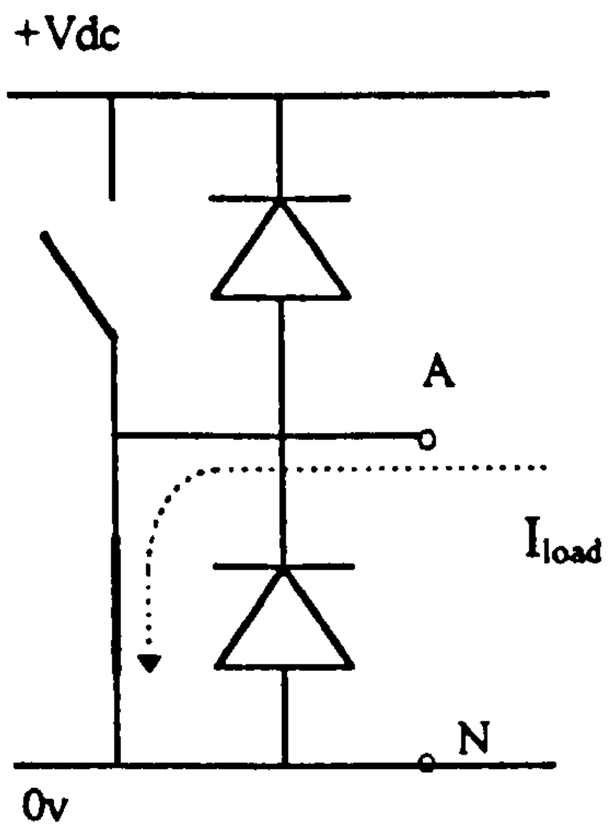


Figure 3.4.5c

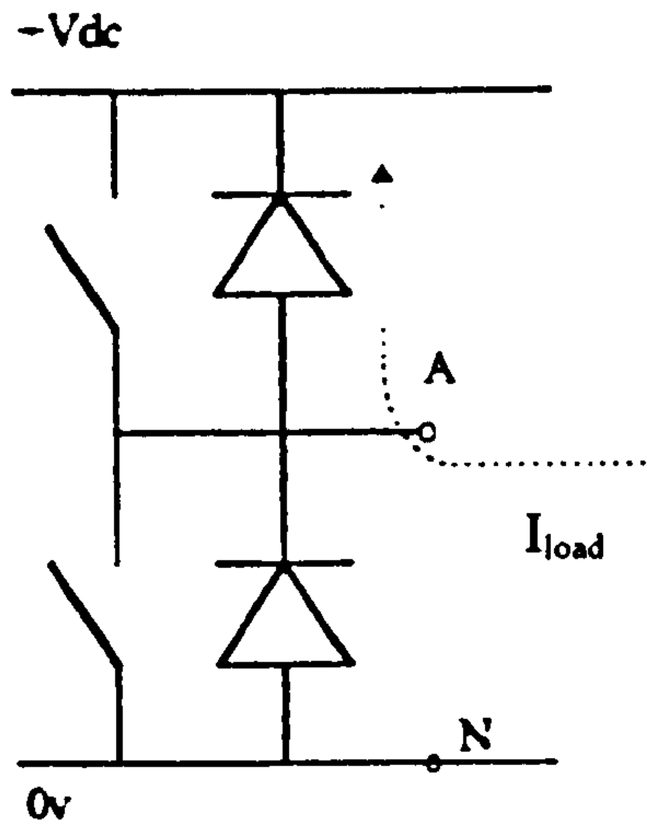


Figure 3.4.5d

Figure 3.4.5 Dead Time Current Flow, Negative Load Current

The voltage V_{AN} will switch between 0v and +Vdc, this is exactly the same voltage swing which would have occurred if the dead time delay had not been inserted and the upper device had been turned on simultaneously with the lower device turning off. Therefore dead time during this period does not change the effective switching pattern.

The above cases highlight how the dead time delay effects a single phase, obviously there is a dead time delay inserted on all three phases and they must all be considered when arriving at a suitable method of simulating the overall effect. The previous section arrived at the seven step switching pattern which is applied to the motor model. This switching pattern must now be modified to include the dead time delays. The following timing diagrams show the effect of the dead time delays on the inverter switching pattern. The effect that the dead time has, is to either increase or decrease the switching periods within the switching pattern. The direction of the current flow in each of the three phases determines whether the times are increased or decreased.

One example of a particular switching pattern is examined here, the other cases are presented in the Appendix of this thesis which cover all possible current directions. This example starts with all of the upper devices off and all of the lower devices on. In the first half of the switching pattern, the turn on of the upper devices is delayed by the dead time after the lower devices have turned off. In the second half of the switching pattern the turn on of the lower devices is delayed by the dead time after the upper devices have turned off. Consider the three phase inverter bridge shown below with the three phase currents as indicated.

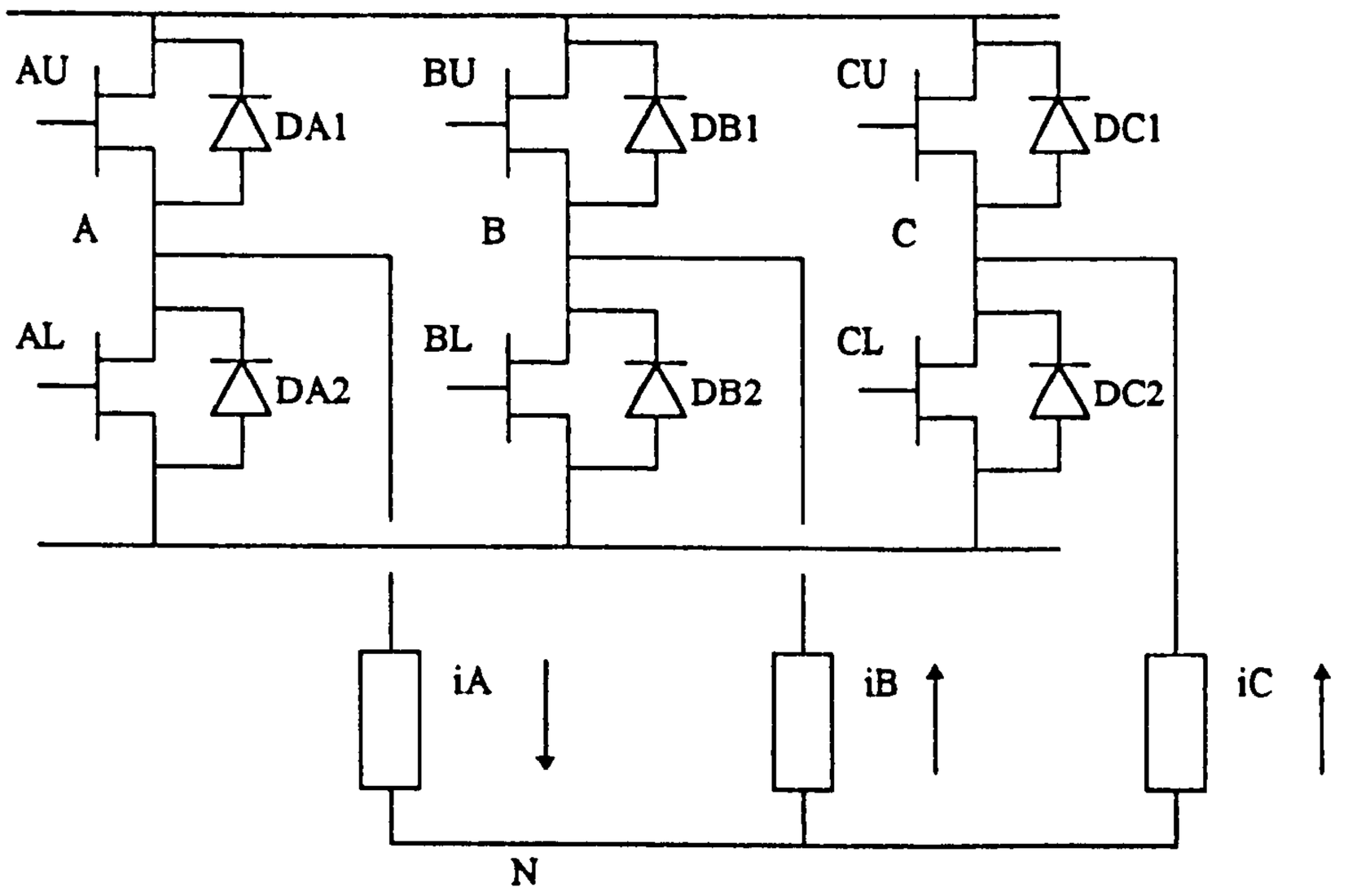


Figure 3.4.6: Instantaneous Current Flow in Inverter

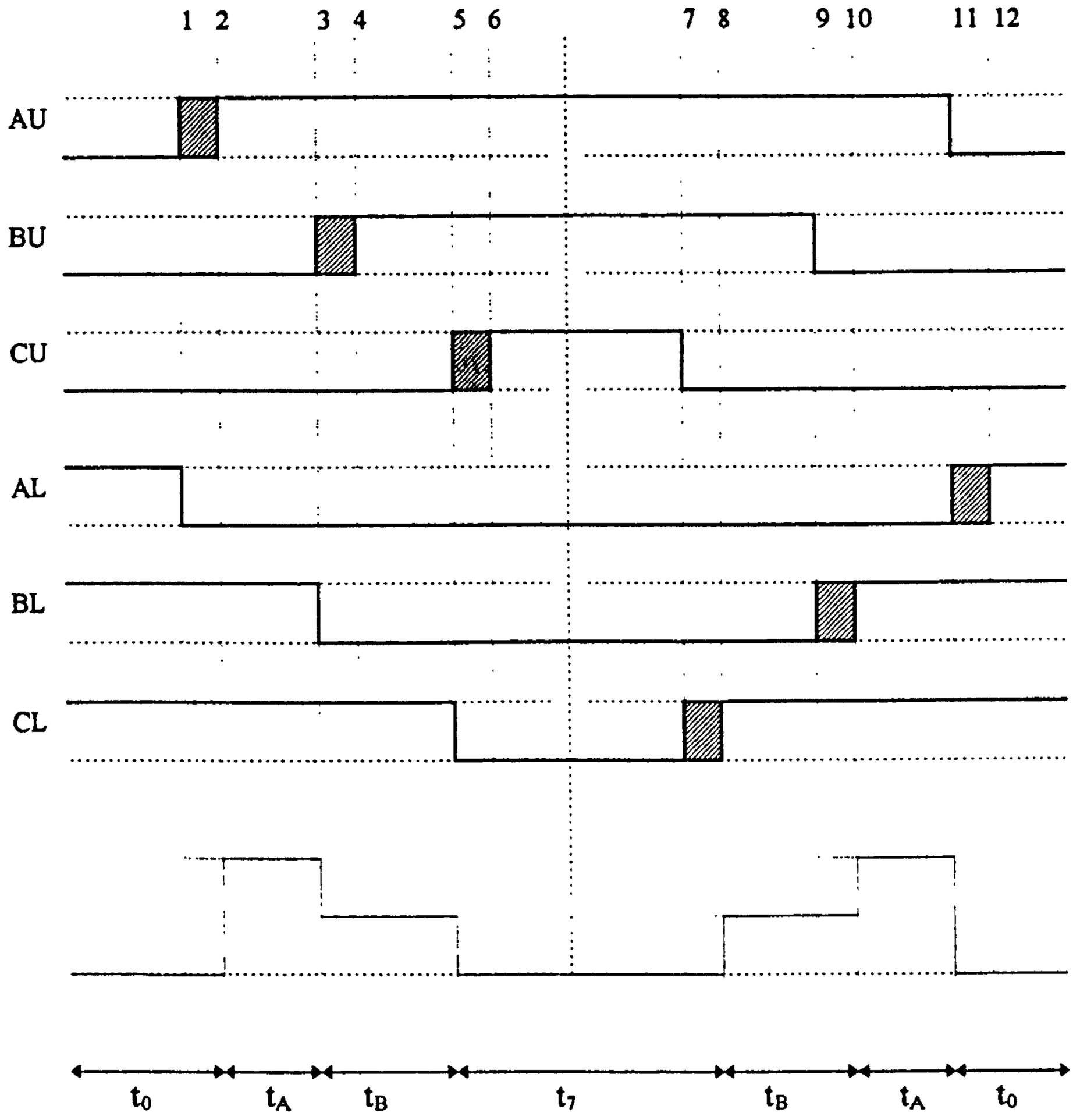


Figure 3.4.7: Modified Switching Pattern

At the start all the lower devices are on, DA2, BL and CL are conducting:-

1. AL Opens iA continues to flow via DA2, point A remains at 0v, zero state continues

2. AU Closes iA flows via AU, point A goes to Vdc, state A starts

3. BL Opens iB flows via DB1, point B goes to Vdc, state B starts

4. BU Closes iB continues to flow via DB1, point B remains at Vdc, state B continues

5. CL Opens iC flows via DC1, point C goes to Vdc, zero state starts

6. CU Closes iC continues to flow via DC1, point C remains at Vdc, zero state remains

7. CU Opens iC continues to flow via DC1, point C remains at Vdc, zero state remains

8. CL Closes iC flows via CL, point C goes to 0v, state B starts

9. BU Opens iB continues to flow via DB1, point B remains at Vdc, state B remains

10. BL Closes iB flows via BL, point B goes to 0v, state A starts

11. AU Opens iA flows via DA2, point A goes to 0v, zero state starts

12. AL Closes iA continues to flow via DA2, point A remains at 0v, zero state continues

The overall result of the dead time in the three phases is to modify the switching pattern as shown, the first zero voltage state is increased by T_{dead}, the first voltage state, state A, is reduces by T_{dead}, the second voltage state, state B, remains unchanged, the overall middle zero voltage state is increased by T_{dead}, the second state B is unchanged, the second state A is reduced by T_{dead} and finally the last zero state is unchanged. The simulation of the modulator can therefore modify the lengths of the switching states, depending on the direction of the current flow, to include the effect of dead time delays and the current flow through the free-wheeling diodes.

The following figures show the result of including the dead time delay in the simulation. Fig. 3.4.8 shows the no-load motor phase current with a dead time of 2μs. Fig. 3.4.9 shows the actual motor current for this dead time value. The simulated waveform now exhibits a similar amount of distortion to the actual phase current give or take the measurement noise.

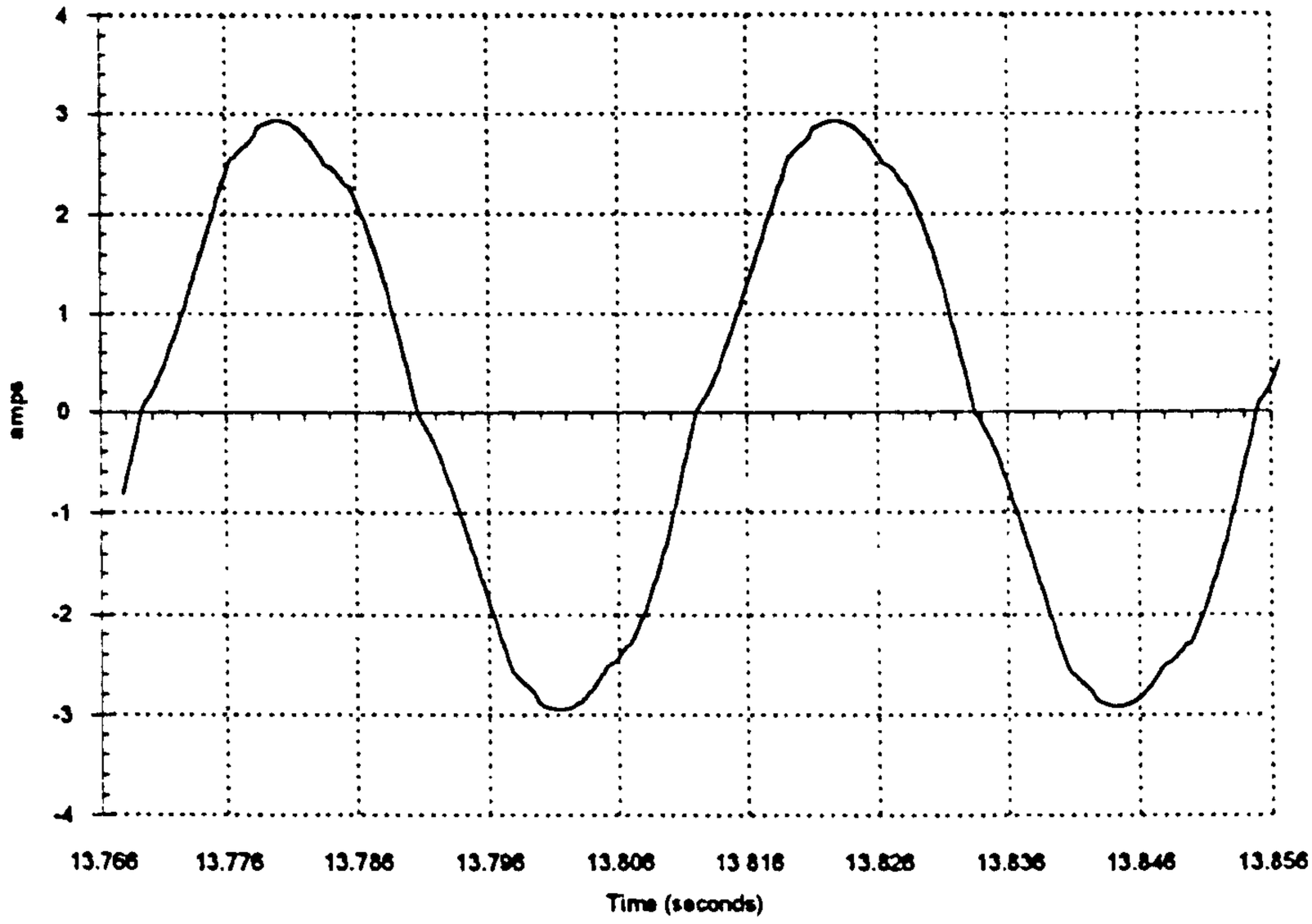


Figure 3.4.8: Simulated Current Waveform with 2μs of Dead Time

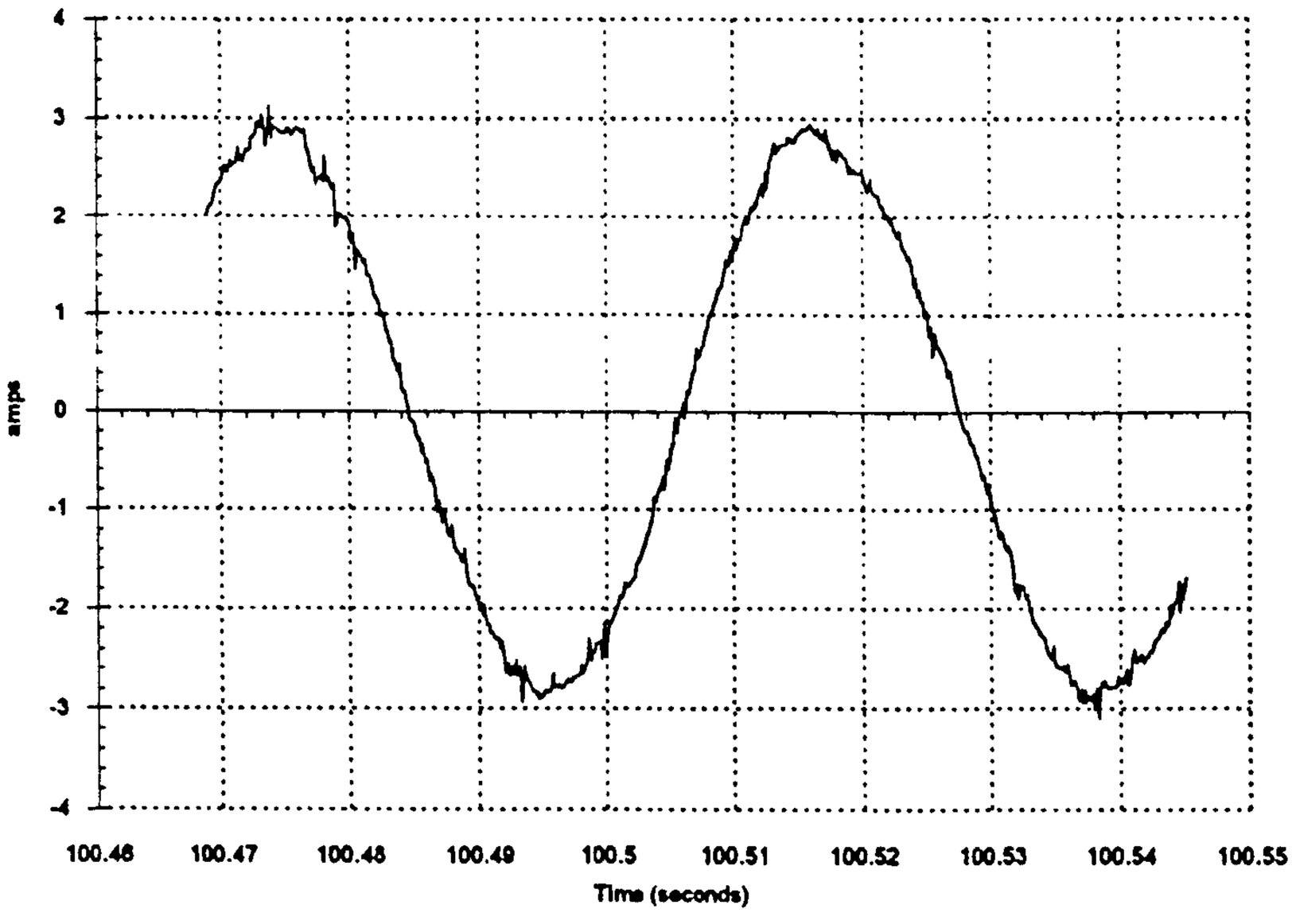


Figure 3.4.9: Actual Current Waveform with 2μs of Dead Time

Research has been carried out into methods which compensate for the distortion caused by the dead time. [*Dodson et. al.* 1990] proposed a method which changed between two gate firing signals depending on the direction of the phase current, [*Sukegawa et. al.* 1991] proposed a feed forward method which compensated for the dead time in the rotating D-Q reference frame and [*Murai*, 1987] used a method of comparing actual switching instances with demanded and then modifying the next demanded instant to take into account the previous error. No compensation methods were employed in the simulation since the PWM ASIC used in the actual drive had no compensation functions. The amount of distortion will increase for lower voltage demands since the dead time will be a larger percentage of the device on times.

3.5 DC Link

The DC link provides the voltage which the bridge switches across the motor windings. The DC link is formed by rectification and smoothing of the three phase supply. The components which make up the link and have to be simulated are shown below. The smoothing of the rectified voltage is achieved by the link capacitor and the link inductor is included to reduce the chances of discontinuous current occurring for low level inverter currents. The resistor R_{dump} is switched across the DC link if the link voltage rises above a pre-set level to quickly reduce the link voltage. The dump resistor will be switched in during braking periods when the machine generates power back into the DC link. Since the rectifier used in this inverter is uncontrolled, power cannot be fed back into the mains supply during these braking periods, hence the link voltage will rise and damage the link components if the dump resistor is not switched across it.

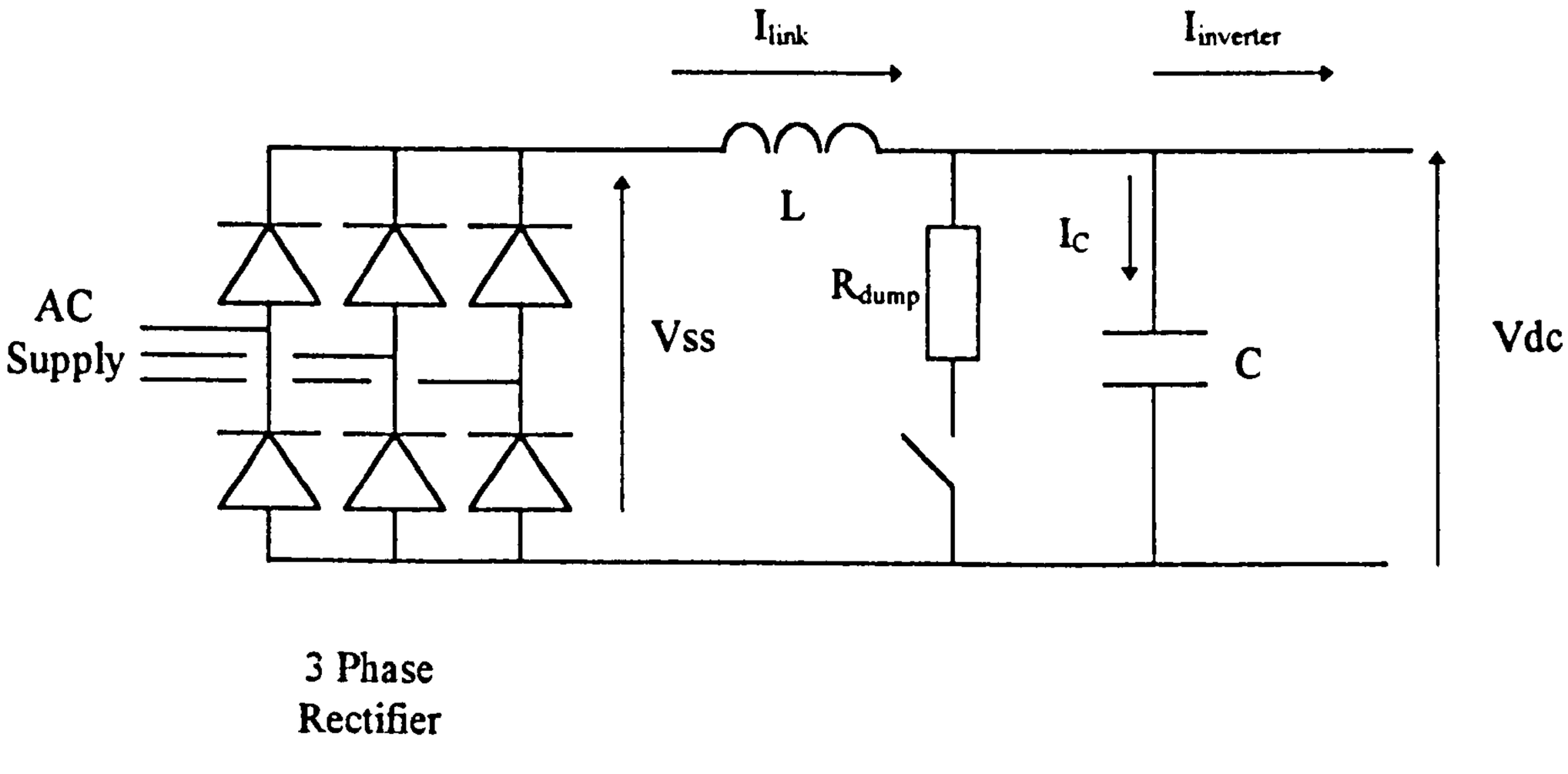


Figure 3.5.1: DC Link Schematic

The rectified voltage Vss is a full wave rectification of the three phase supply, and is calculated mathematically, the blocking effect of the rectifier is simulated by effectively inserting a diode in line with the inductor as shown:-

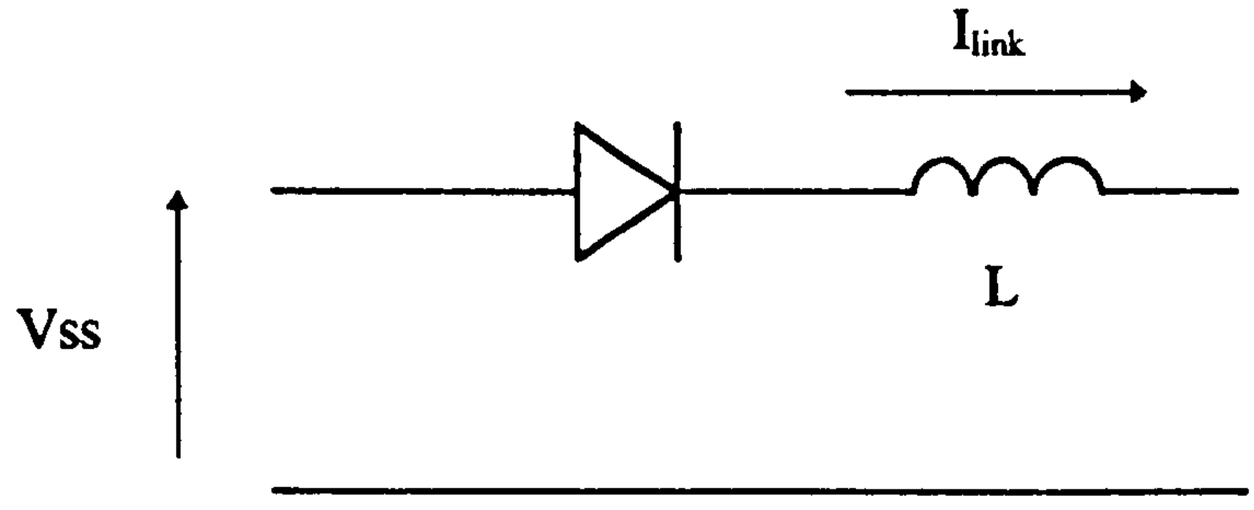


Figure 3.5.2: Rectifier Representation

The voltage across the link capacitor is simulated by solving the following differential equation:-

$$\frac{dV_{dc}}{dt} = \frac{1}{C}(I_{link} - I_{inverter}) \qquad (3.5.1)$$

The link current is also solved by a differential equation:-

$$\frac{dI_{link}}{dt} = \frac{1}{L}(V_{ss} - V_{dc} - R_l I_{link}) \qquad (3.5.2)$$

where R_l is the series resistance of the link inductor.

It can be seen that the link voltage is dependent upon the inverter current, this inverter current can be calculated if the bridge state and the phase currents are known. It assumes, for example, that if the upper switch of the 'A' phase is closed and the lower switches of the 'B' and 'C' phases are closed, then the current from the link is equal to the 'A' phase current. Similarly if two upper devices are closed and one lower, then the current taken from the link is equal to the sum of the two upper phase currents, which for a balanced system will equal the current in the lower phase.

Tests were carried out on the link simulation with the following parameters:-

$C = 2000\mu f$

$L = 3.5mH$

$R_l = 0.1\Omega$

$R_{dump} = 10\Omega$

Dump upper threshold = 650V

Dump lower threshold = 630V

AC supply 415V three phase

The first result of Fig. 3.5.3 shows the rectified three phase input voltage and the resultant steady state DC link voltage. Fig. 3.5.4 shows the motor phase current and the link current I_{link} for the same steady state no-load condition, the no-load condition means that there is very little current taken from the link as shown. A load of 10Nm was then applied to the motor model and the same results taken. Fig. 3.4.5 shows that as the load is applied the DC link voltage reduces from approximately 560 to 530 volts. Fig. 3.4.6 shows the increase in the link current due to the applied load, the link current is discontinuous during the no-load case but as the load is applied the current will eventually start to become continuous, higher link inductance would make it more continuous. The final test performed on the DC link simulation involved applying a speed reversal to the motor. The speed demand to the motor was reversed and Fig. 3.5.7 shows the motor phase current during the reversal. The DC link plot of Fig. 3.5.8 shows how the link voltage rises as the motor begins to generate back into the DC link as it slows down, the dump resistor switches in at the pre-defined voltage level of 650 volts and rapidly reduces the DC link voltage until it reaches the dump resistors lower threshold of 630 volts. The Dump resistor continues to switch in and out of the circuit removing energy from the motor until it reaches zero speed, as the motor increases velocity in the opposite direction the link voltage reduces as the motor begins to take power from the link again. Fig. 3.5.9 shows the motor speed during the reversal.

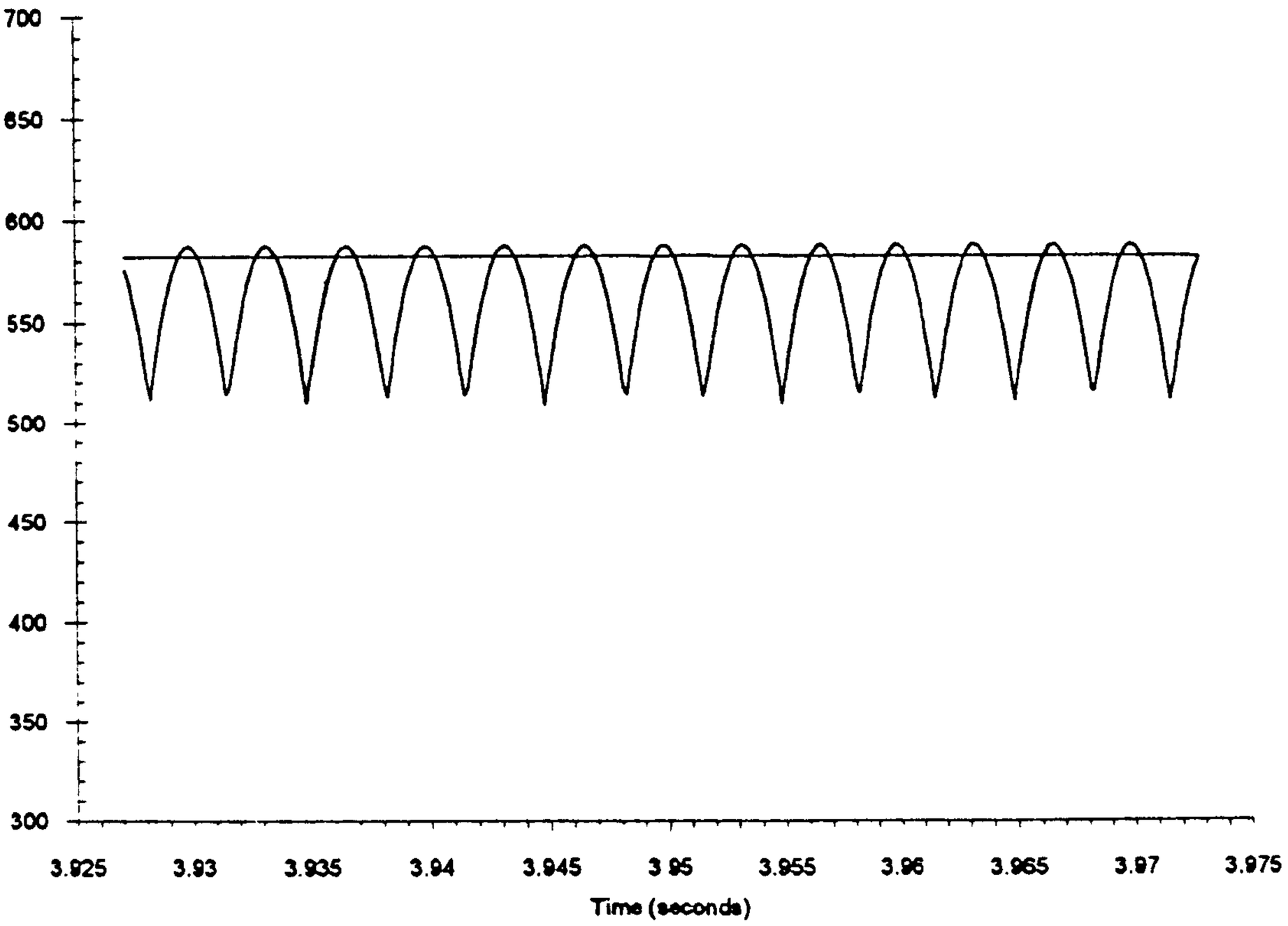


Figure 3.5.3: Steady State Rectifier and DC Link Voltages

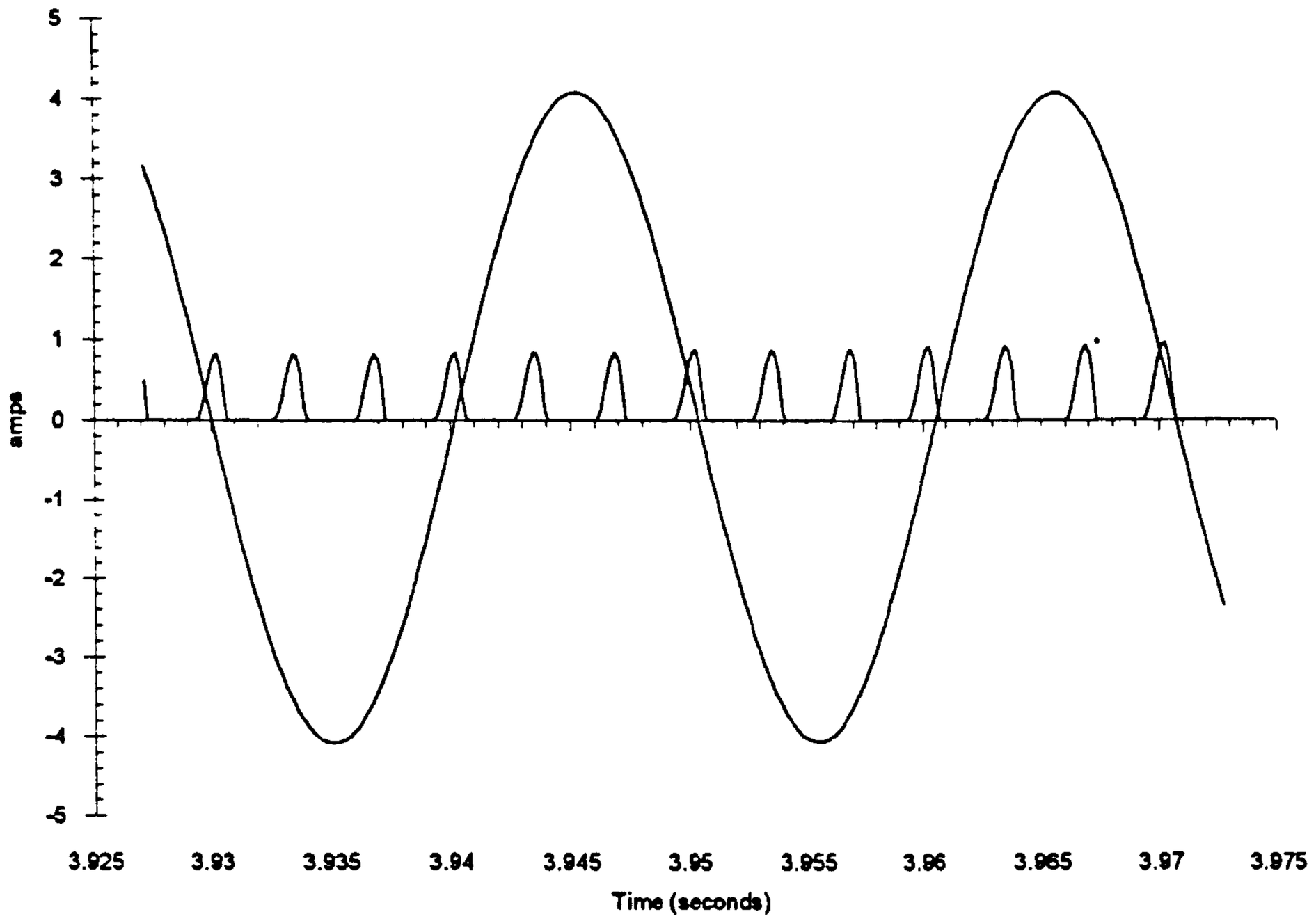


Figure 3.5.4: Steady State Phase and DC Link Currents

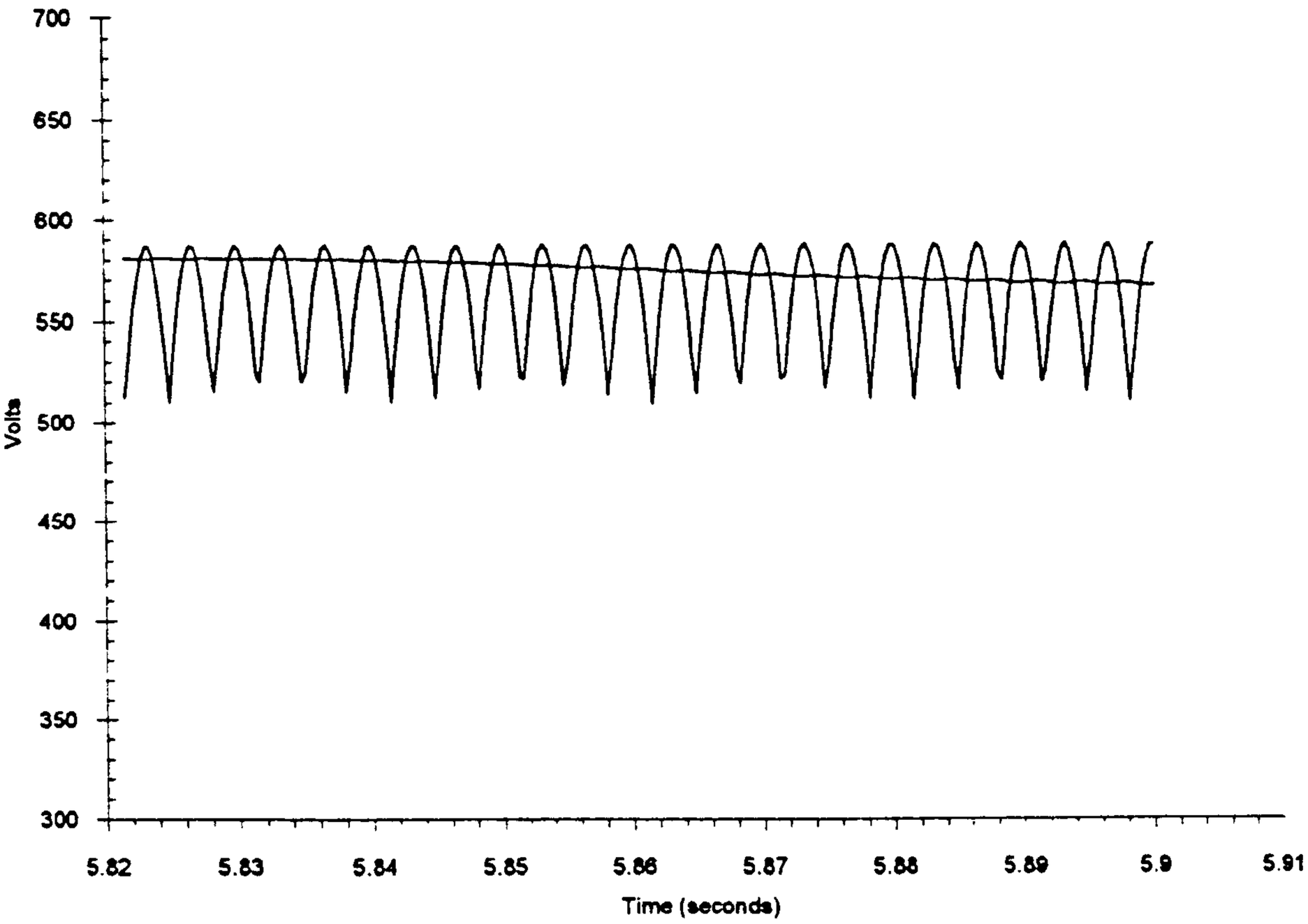


Figure 3.5.5: Rectifier and DC Link Voltages For Applied Load

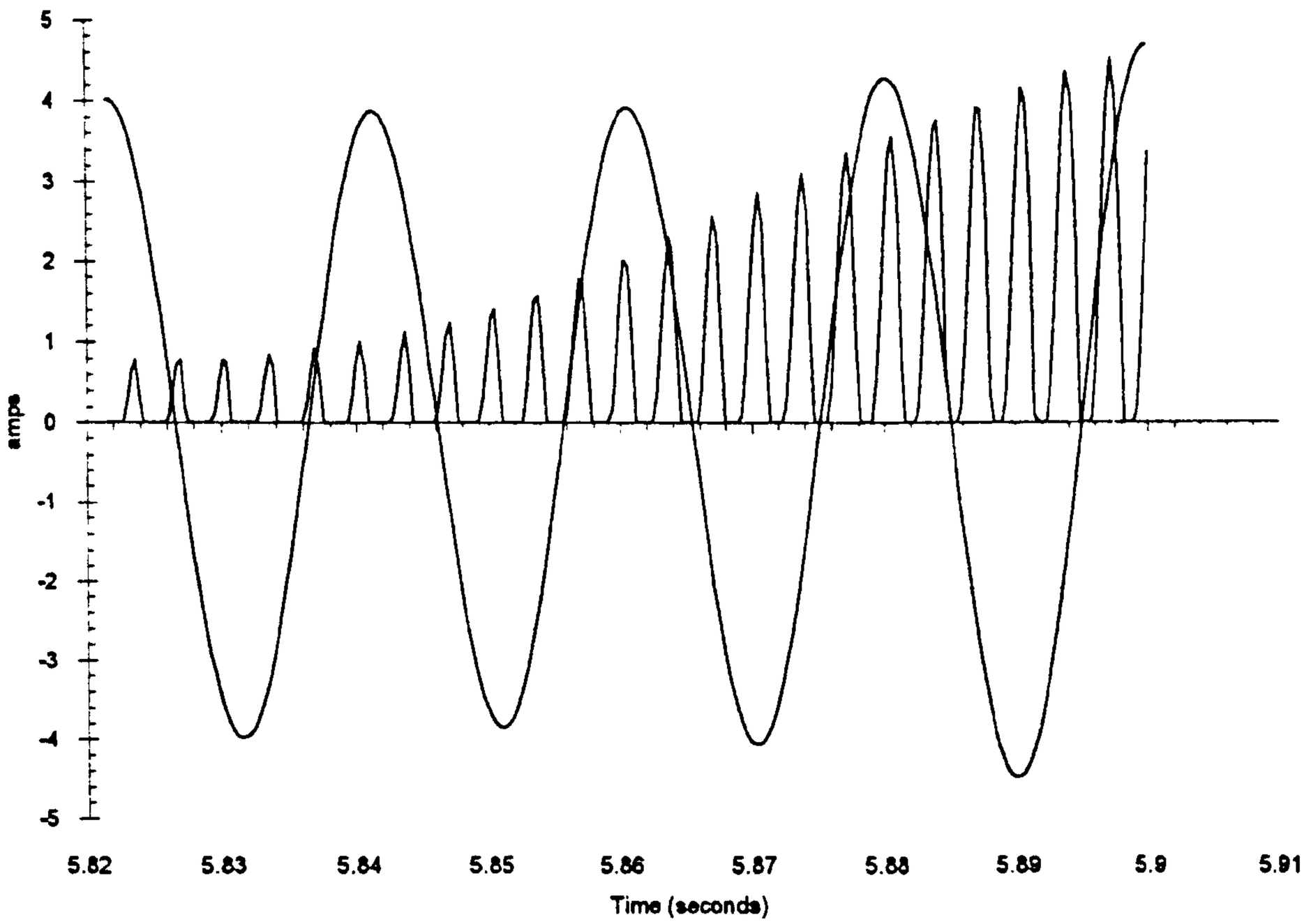


Figure 3.5.6: Phase and DC Link Currents For Applied Load

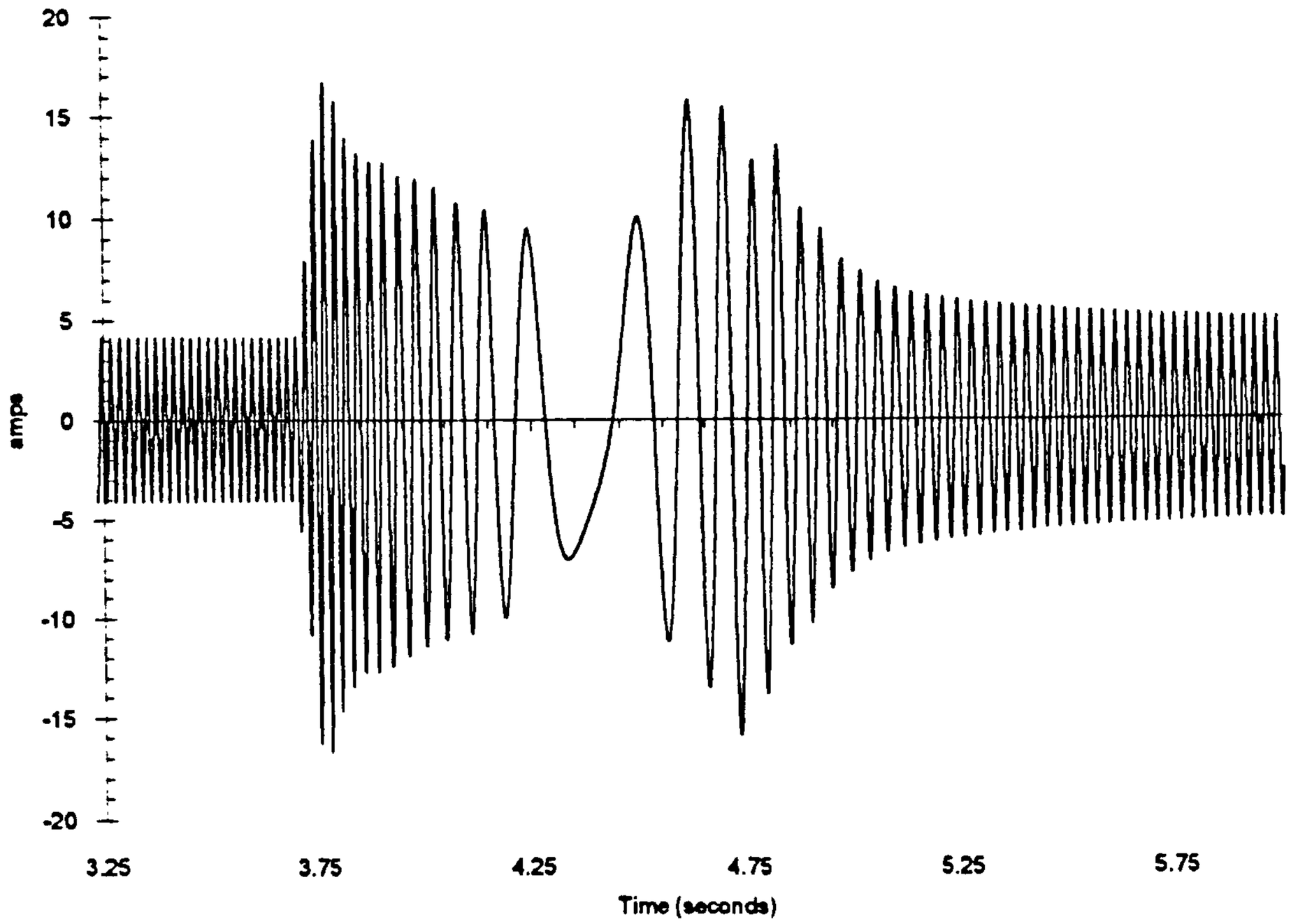


Figure 3.5.7: Phase Current During Speed Reversal

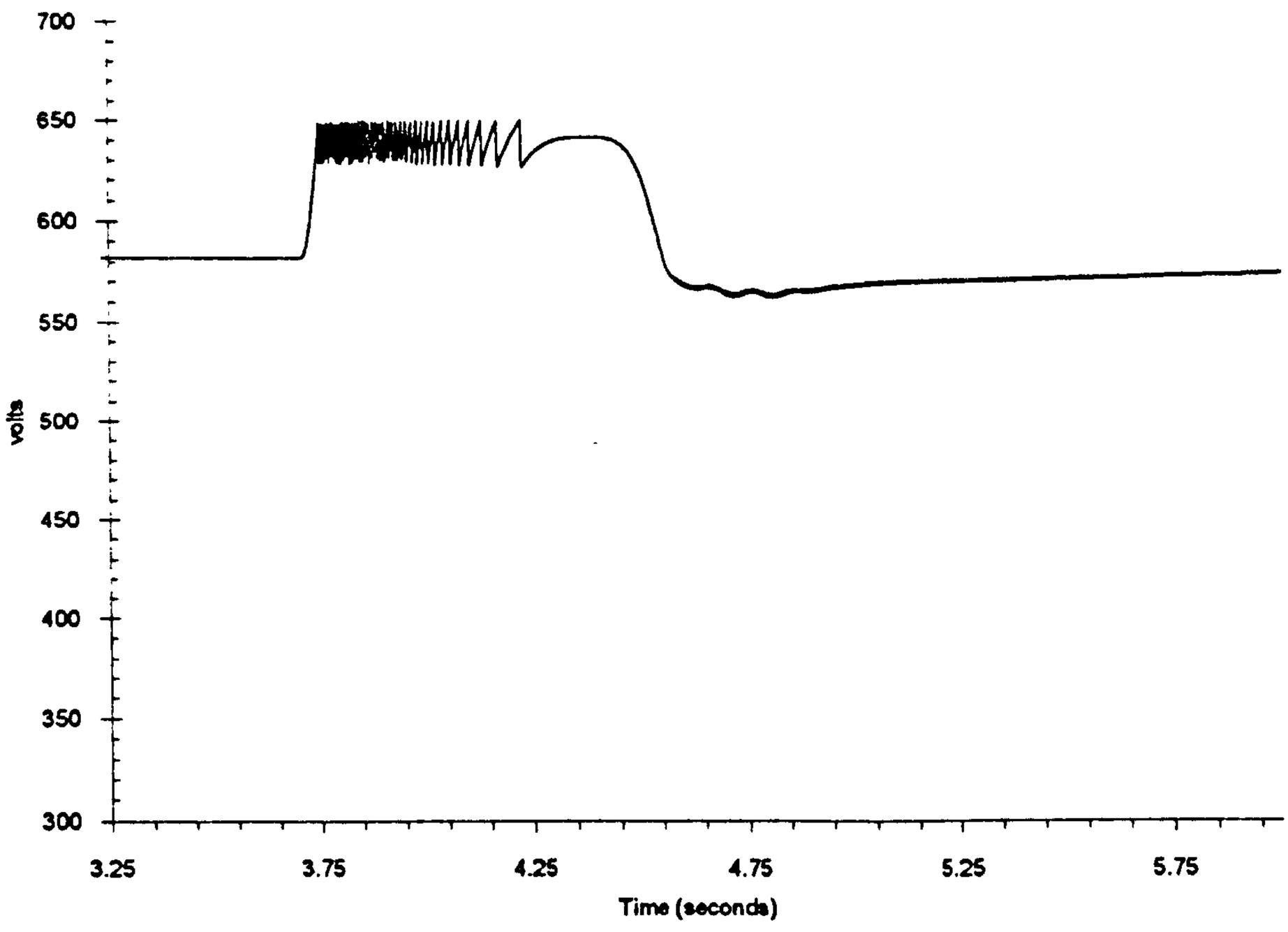


Figure 3.5.8: DC Link Voltage During Speed Reversal

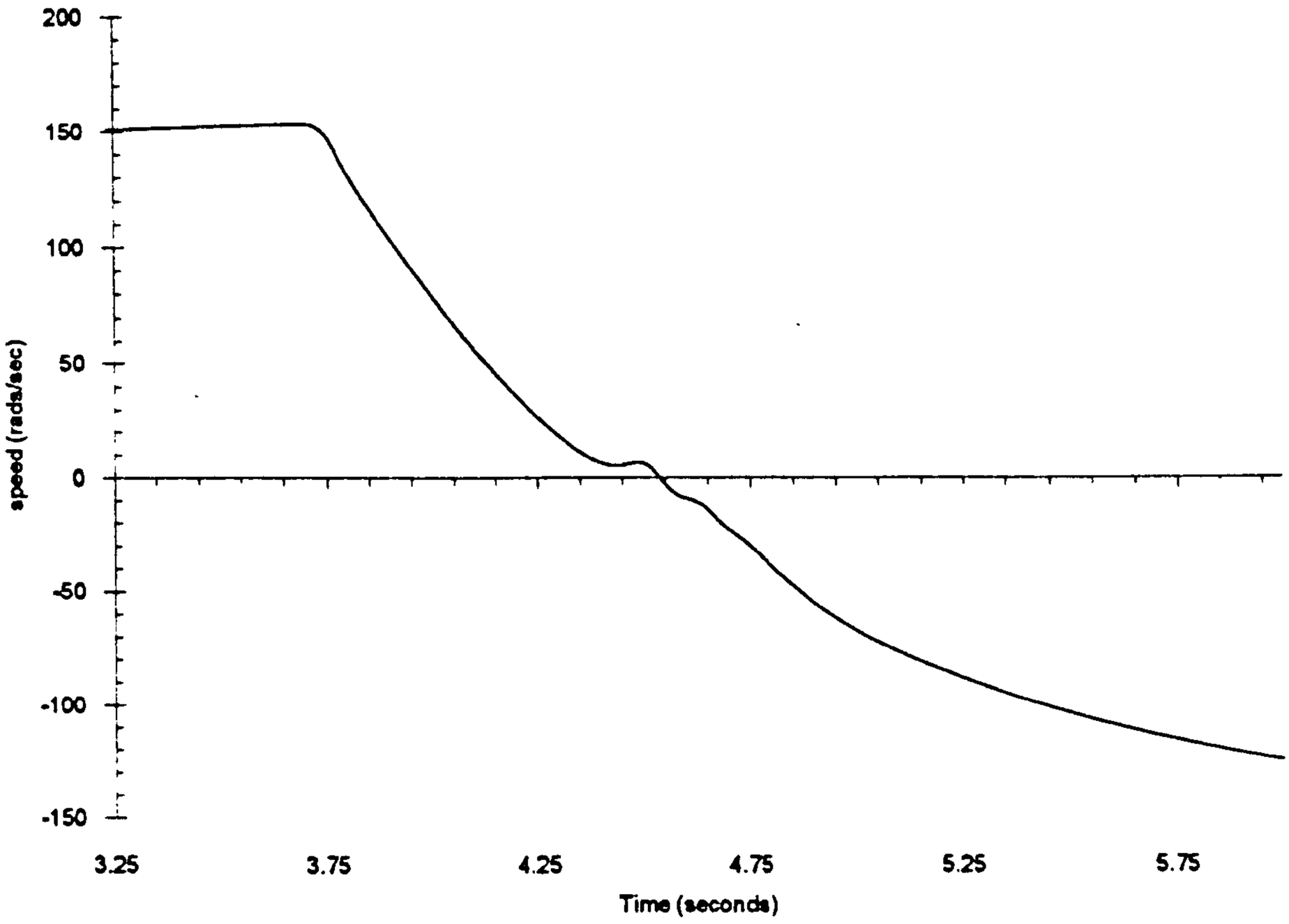


Figure 3.5.9: Rotor Speed During Speed Reversal

The following results were taken from the actual drive and show the DC link voltage and the link current for a steady state case. The first, Fig. 3.5.10, shows the actual DC link voltage and actual link current, the second, Fig. 3.5.11, shows the simulated DC link voltage and currents and also the simulated rectifier output voltage. The variations in the magnitudes of the link current pulses are due to the imbalance of mains supply voltage. In order to include this in the simulation the actual three line voltages were measured and used to produce the simulated rectifier output voltage, this is an instantaneous measurement and any variation in the mains supply during the tests cannot be included in the simulation. The variation in the mains supply accounts for the changes in the link current magnitudes.

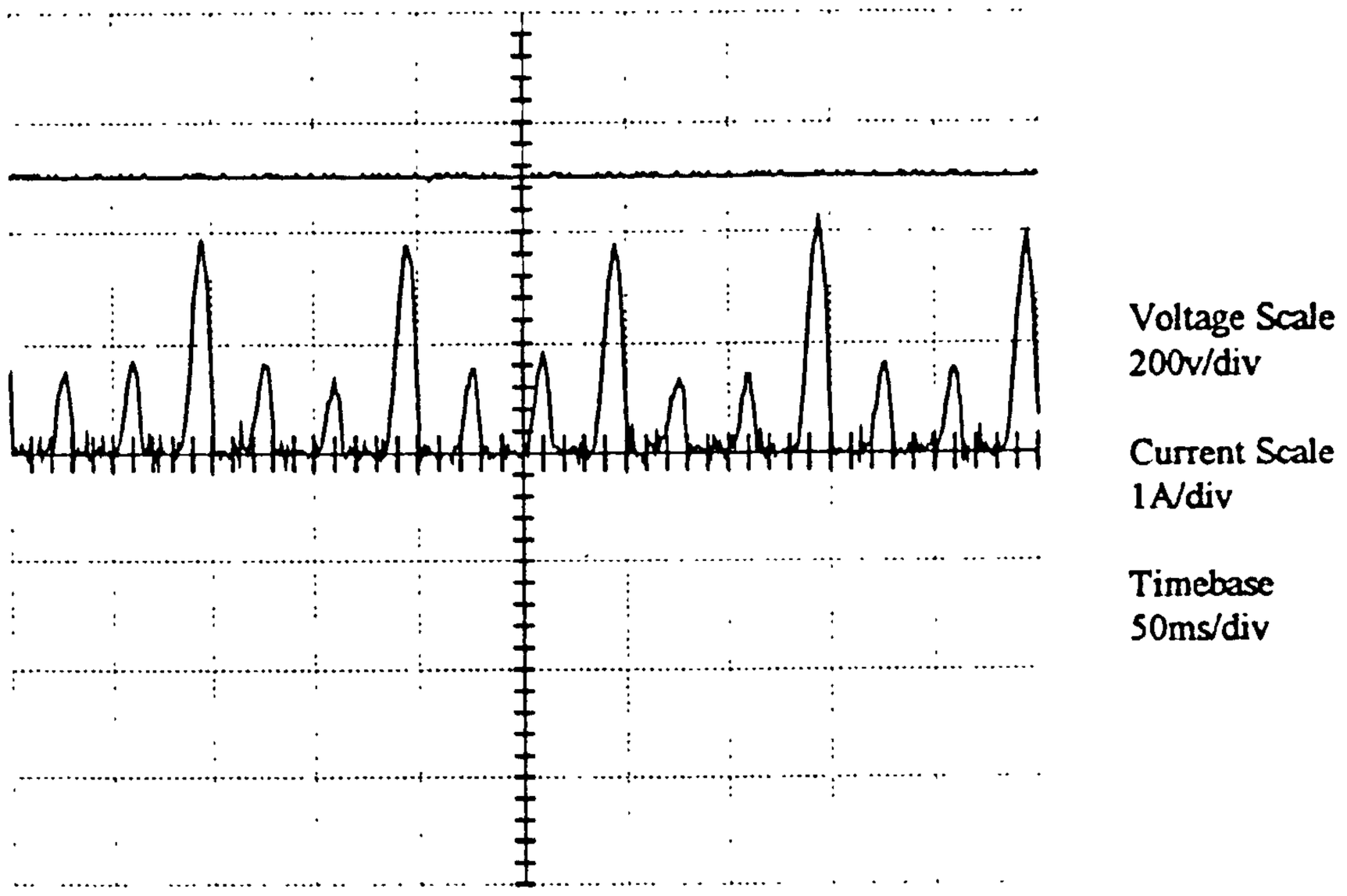


Figure 3.5.10: Actual DC Link Voltage and Current

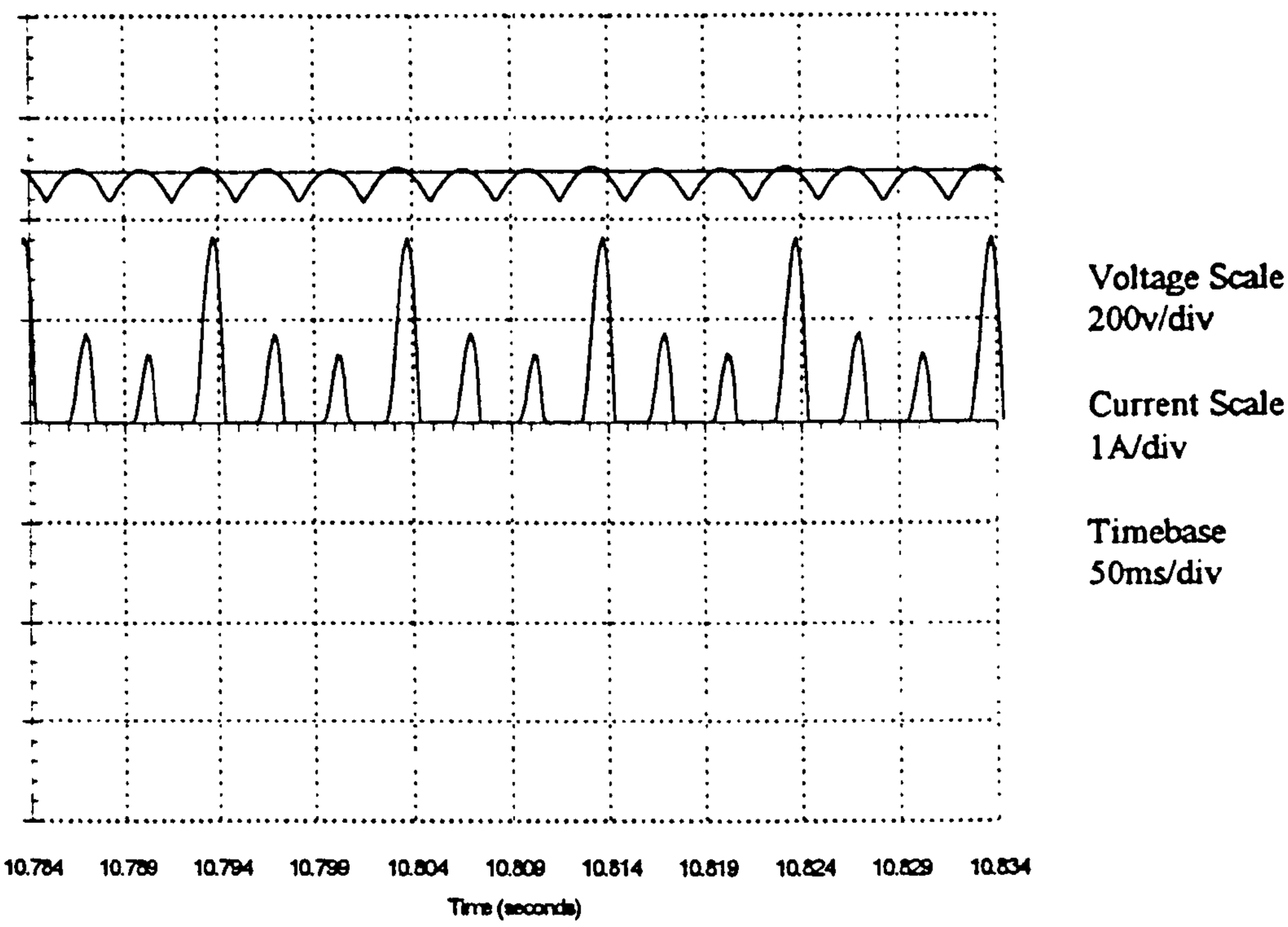


Figure 3.5.11: Modelled Rectifier Voltage, DC Link Voltage and Current

3.6 Variable Time Stepping

3.6.1 Theory

The modulator simulation calculated the bridge states and the duration of these bridge states over the switching period. The inverter bridge simulation calculated the amount of voltage to be applied to the motor model due to these bridge states. The motor model must now be time stepped over the switching period. The drawbacks of using extremely fine time steps and checking for a change in the bridge state at each time step, have previously been highlighted i.e. excessive execution times. It would therefore be advantageous to solve the motor model equations in such a manner which did not require small time steps, thus increasing simulation speed, but did not overlook or incorrectly apply any switching instants to the model equations.

Since the actual moments when the bridge is to be switched are pre-calculated by the simulation of the modulator, this information could be used by the motor model to avoid the necessity to have fine time steps. Using this information introduces the idea of a variable time step model in which the time steps are dependent upon the duration of the switching states.

Consider the state information for a single switching period for a three phase demand of :-

Va = -241.4 v

Vb = 188.69 v

Vc = 52.72 v

Applying these demands to the modulator equation with a switching frequency of 6.5KHz :-

$$T = (255 + V) \times \frac{switching\_period}{510} \qquad (3.6.1)$$

gives the on times for each phase upper device:-

$$Ta = 4.1\mu s$$

$$Tb = 133.8\mu s$$

$$Tc = 92.83\mu s$$

The following figure shows these times and the resultant states which will occur together with the approximate state times:-

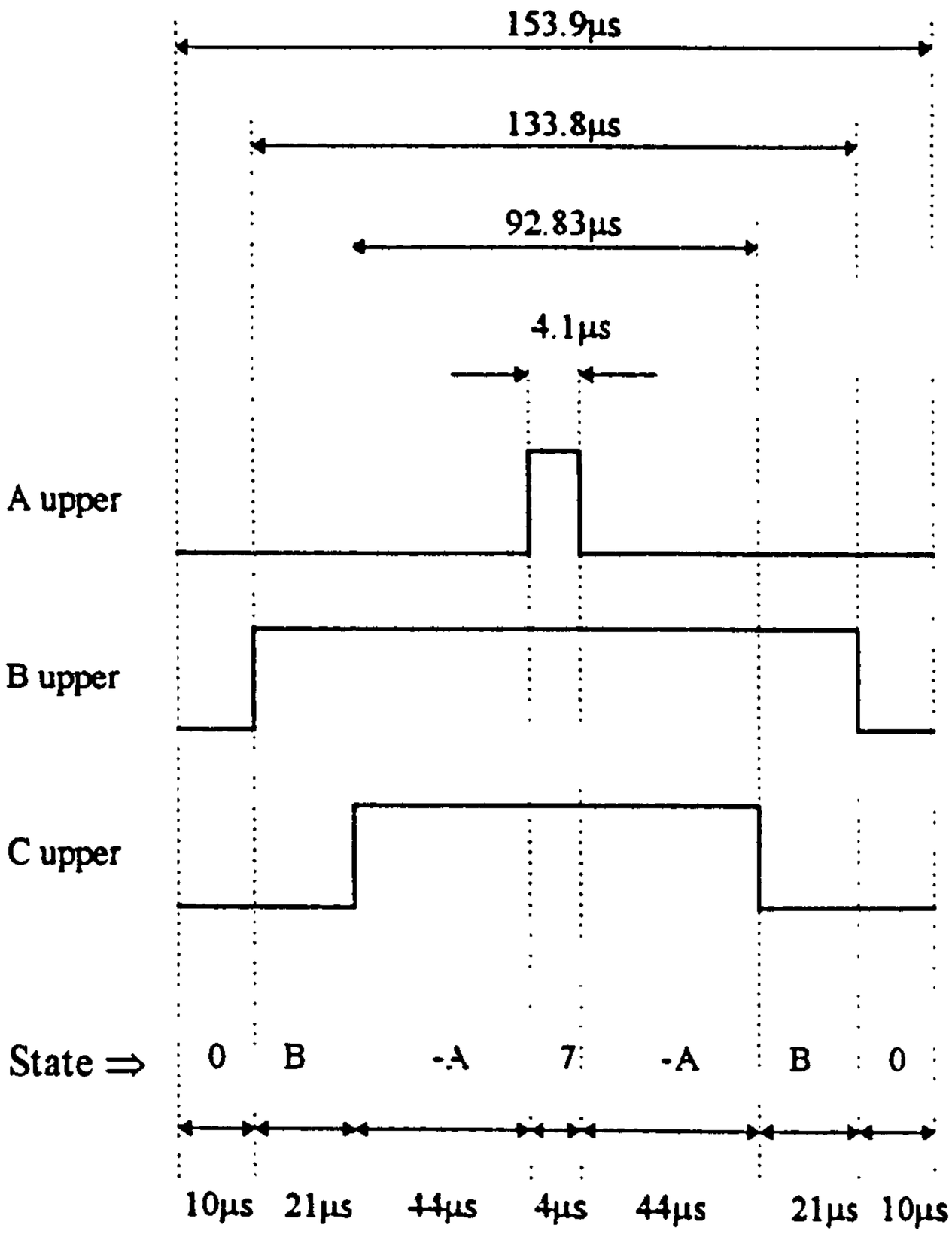


Figure 3.6.1: State Times for Instantaneous Three Phase Demand

The actual phase voltages applied to the motor and the D and Q axis voltages which would be applied to the motor model, are shown below in Fig. 3.6.2, the phase volts switch between ± 2/3 Vdc and ± 1/3 Vdc , the D axis voltage also switches

between ± 2/3 Vdc and ± 1/3 Vdc and the Q axis voltage switches between ± 0.57735 Vdc. The following figures are for a DC link voltage of 600 volts.

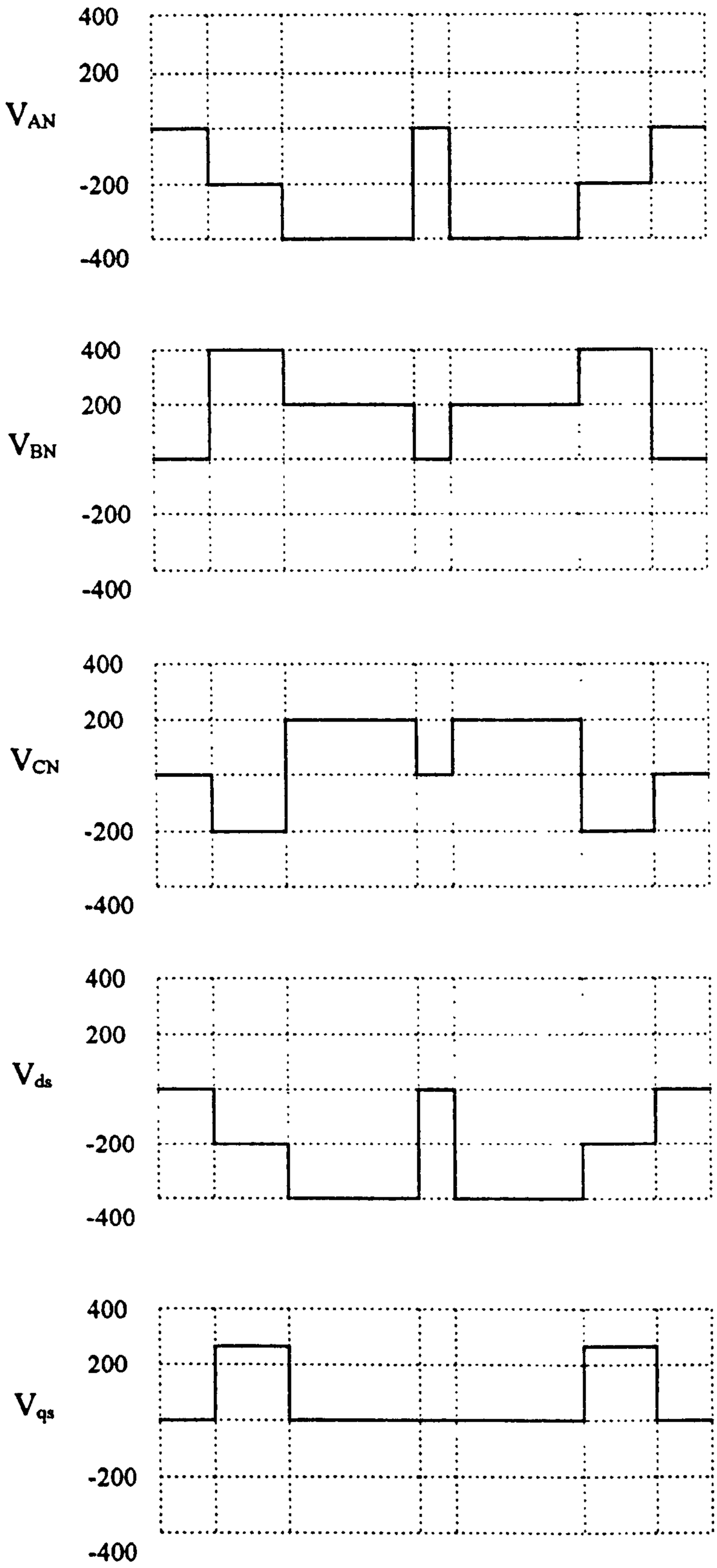


Figure 3.6.2: Phase and D-Q Voltages for One Switching Period

The above example of instantaneous voltage applied to the modulator simulation would result in the motor model equations being solved seven times as follows:-

Time Step	Vds	Vqs
10μs	0	0
21μs	-200	346.4
44μs	-400	0
4μs	0	0
44μs	-400	0
21μs	-200	346.4
10μs	0	0

Table 3.6.1: Time Steps for One Switching Period

In the proposed variable time stepping approach the motor model equations are always solved seven times this corresponds to the seven step pattern output from the modulator. The time steps for each of these seven steps constantly change depending upon the demand to the modulator.

3.6.2 Performance

In order to validate the variable time stepping model it was compared to a constant time step model which checks for a change in bridge state at each time step. The following figures show the D-axis voltage which was supplied by the simulation of the modulator and the bridge, Fig 3.6.3 shows the desired voltage switched at the correct switching instants which the modulator demanded. Fig. 3.6.4 shows the voltage applied to the motor model for a constant time step of 1μs. Fig. 3.6.5 shows the same voltage for a constant time step of 2μs. Fig. 3.6.6 shows the same voltage for a constant time step of 5μs.

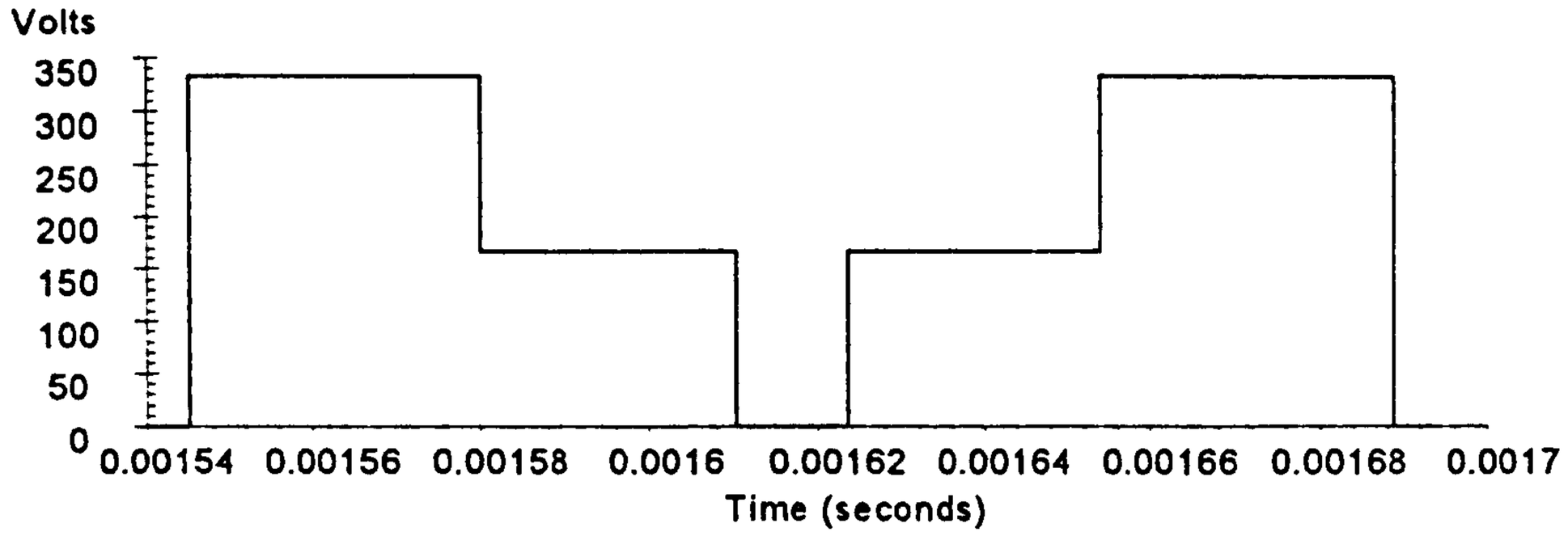


Figure 3.6.3: Desired Voltage Pattern

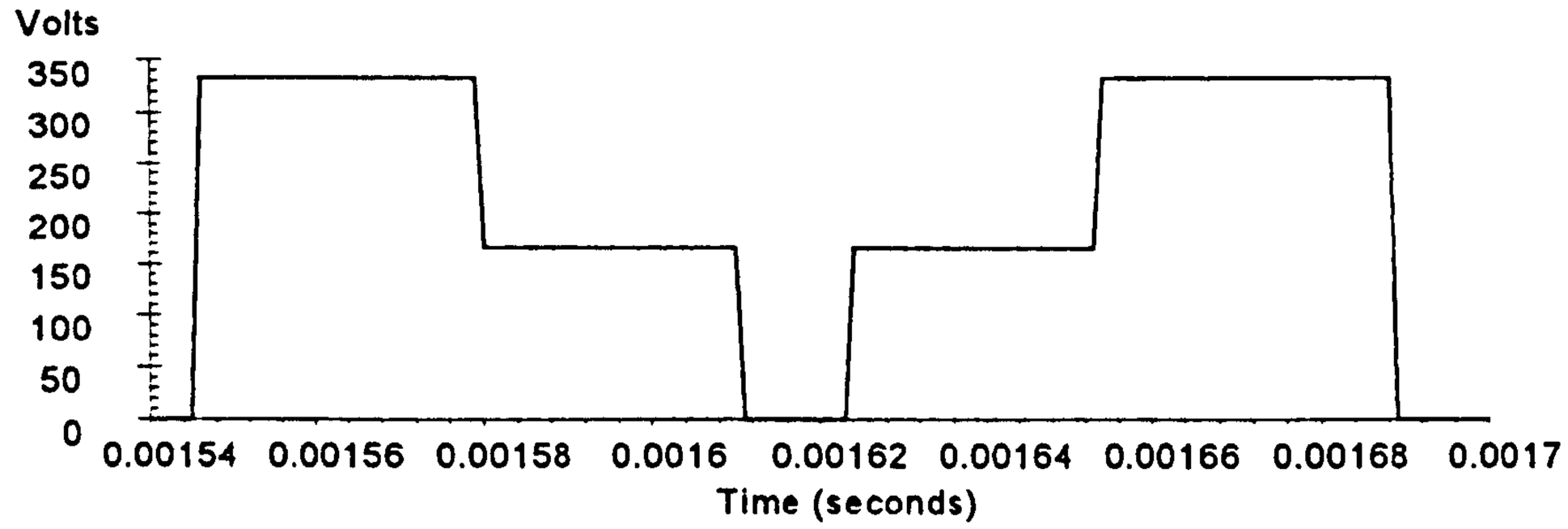


Figure 3.6.4: Constant 1μs Time Step

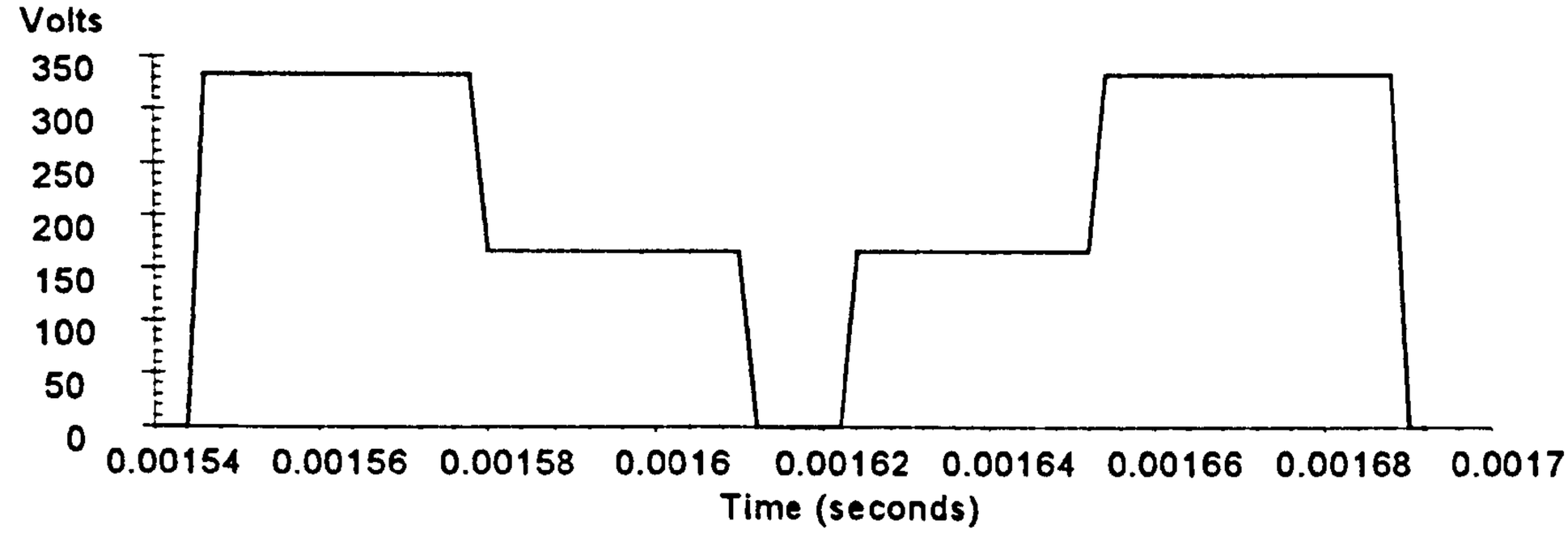


Figure 3.6.5: Constant 2μs Time Step

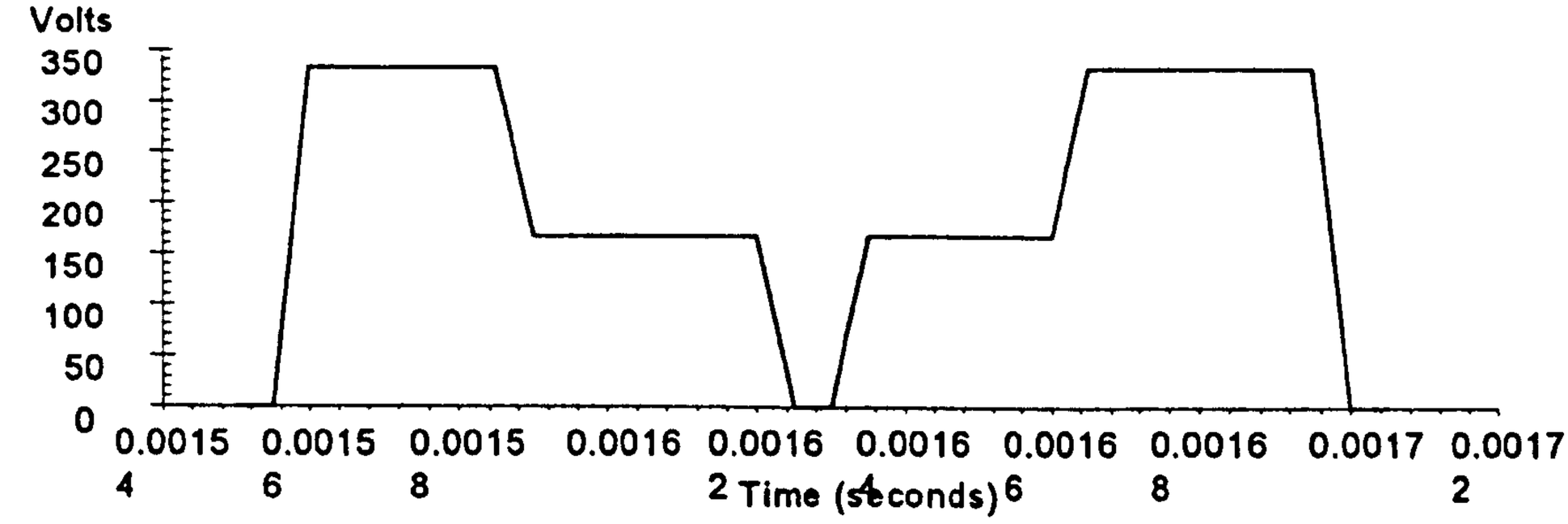


Figure 3.6.6: Constant 5μs Time Step

The necessity to maintain an extremely fine time step when constantly monitoring for changes in bridge state is clearly shown by the previous figures, at a time step of 1μs the voltage pattern that is applied to the motor model is close to the desired output voltage pattern from the modulator, but increasing the time step to 5μs, which is still relatively small, has led to a considerable degeneration in the voltage pattern.

Section 3.6.1 concluded with the seven step pattern which is supplied by the modulator, the motor model must now be time stepped over this pattern. It is clear that the minimum number of times the motor model equations must be solved is seven, however, depending upon the switching frequency the time steps could be large. For the results taken during this project a switching frequency of 6.5KHz was used and this frequency will be used throughout the following tests.

The minimum execution time for the variable time step model would correspond to solving the model equations seven times using the Euler solver. The Euler solver has already been highlighted as causing inaccuracies at large time steps, the maximum time step which can be produced by the variable time step model is approximately 75μs, this is only valid for the 6.5KHz switching frequency which gives a switching period of 154μs. The following figure shows the D-Axis stator and rotor currents produced by the variable time step model and the constant 1μs model. The test involved running the motor at almost synchronous speed and then applying the three phase stator voltage, thus resulting in a short start up transient. The figure 3.6.7 shows that there is good agreement between the two time stepping methods during the large transient period but as the currents reduce then the results of the two methods begin to deviate.

Figure 3.6.8 shows the steady state case for the same currents. The difference between the two methods is clear, the stator current has reduced for the variable time step method and the rotor current is increased with respect to the constant time step method.

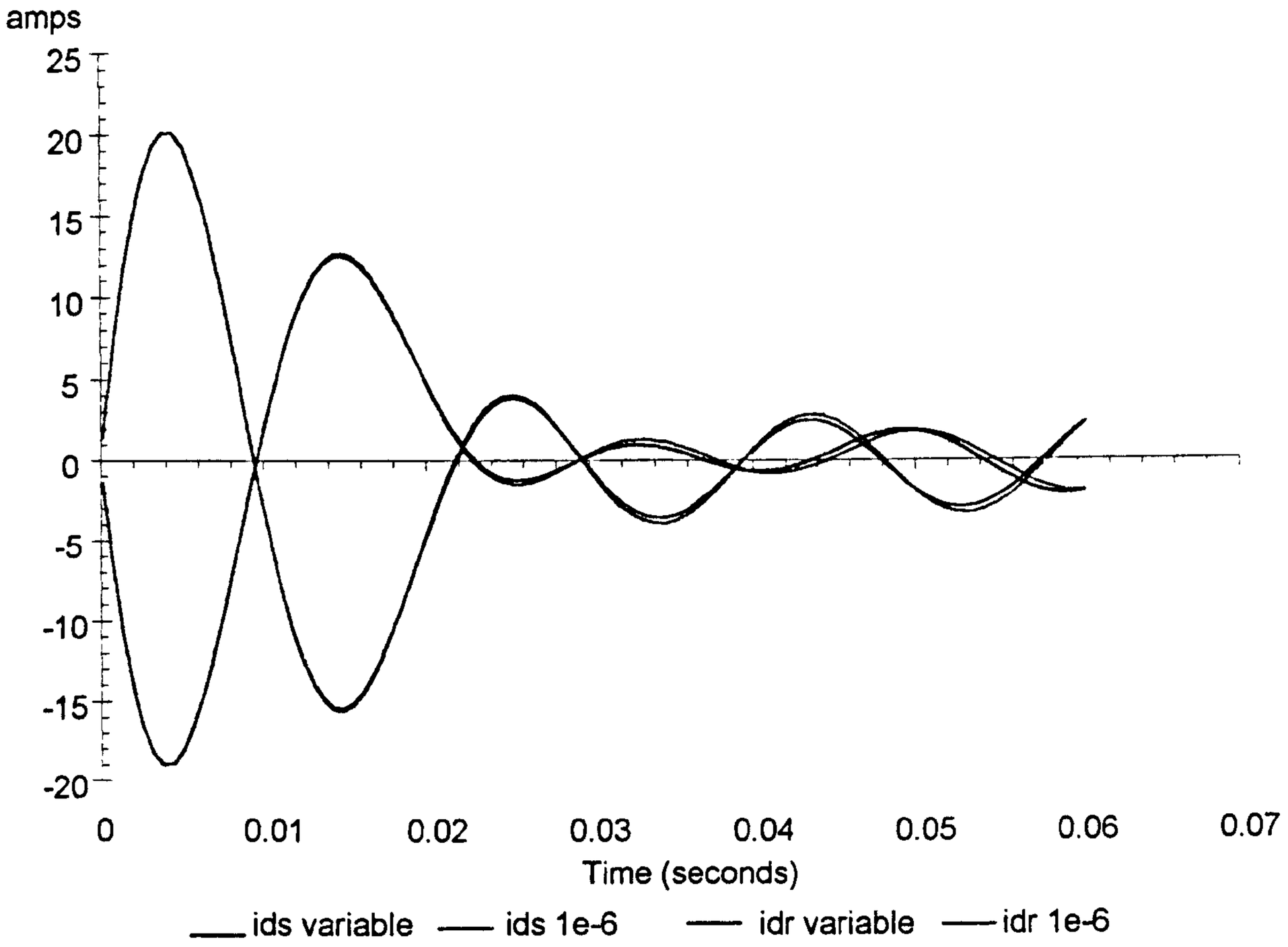


Figure 3.6.7: Transient Response of Variable Time Step Model Using Euler

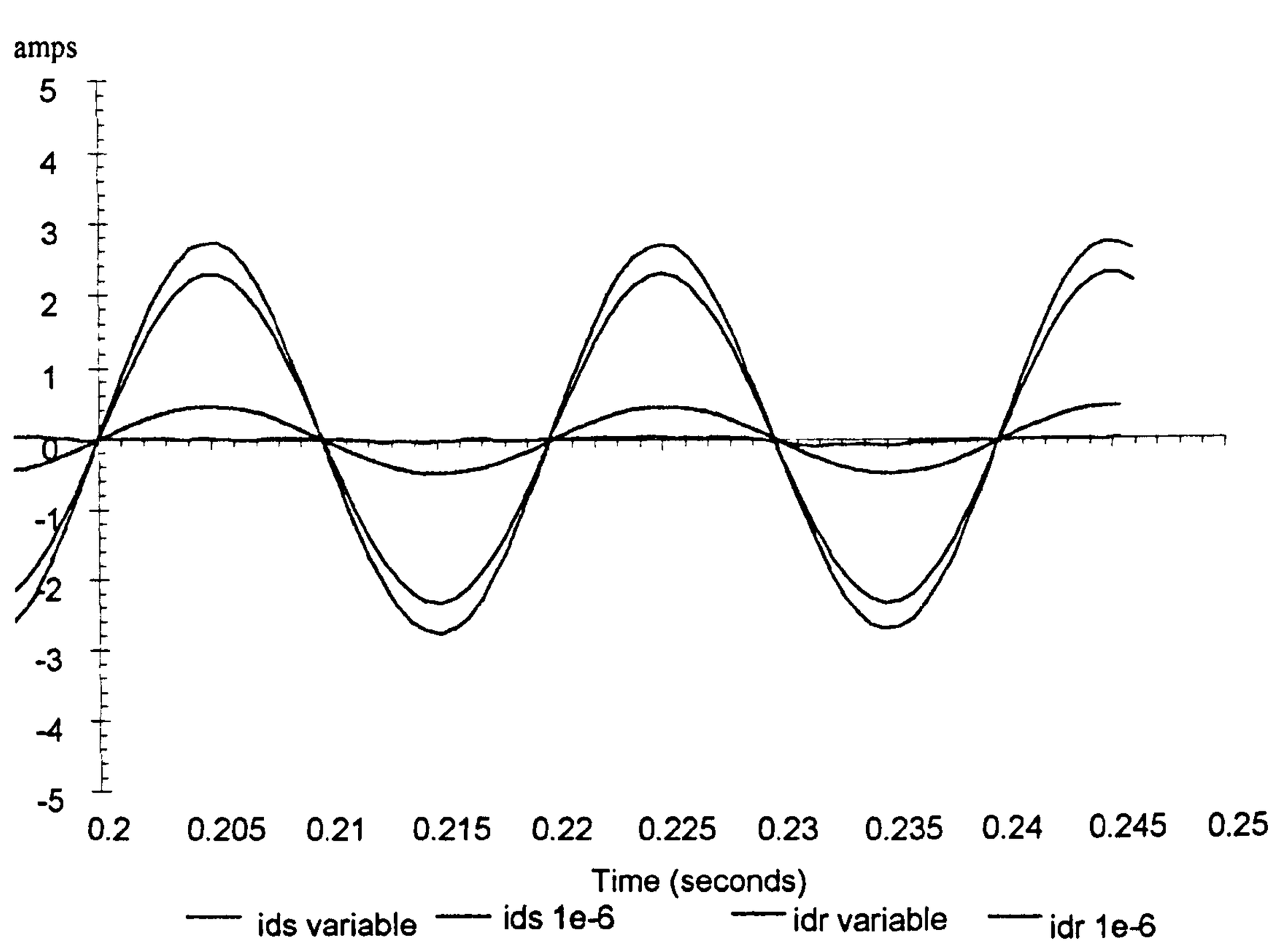


Figure 3.6.8: Steady State Response of Variable Time Step Model Using Euler

A detailed examination of the rotor currents for the steady state case clearly shows the differences between the two methods. Fig. 3.6.9 shows a close view of the rotor currents for the constant time step model and this can be compared to the similar view of the rotor currents for the variable time step model shown in Fig. 3.6.10. It is suggested that these errors occur due to inaccuracies introduced by the Euler method when solving large time steps. A record was made of the size of the switching states which the modulator produced during this steady state test. The voltage demands that were fed to the modulator were 50Hz with a peak of 220v.

State	Minimum Time	Maximum Time
0	8.3μs	19μs
A	1μs	44μs
B	1μs	44μs
7	8.3μs	19μs

Table 3.6.2: Minimum and Maximum Switching Times

The 44μs time steps for the two voltage states will cause the Euler solver to introduce the errors i.e. small di/dt errors. These errors are more noticeable for small values of current, for example the steady state no-load case, where the error is a greater percentage of the overall current.

In order to prove that the Euler method was introducing these errors at the large time steps a Runge Kutta solver was used in place of the Euler solver for the variable time step model. The equations were still only solved seven times Fig. 3.6.12 shows the D-axis rotor currents for the variable time step model incorporating the Runge Kutta solver. Fig. 3.6.11 is the constant 1μs time step model repeated here for comparison. The use of the Euler method for the constant 1μs time step model is adequate due to the small time step length.

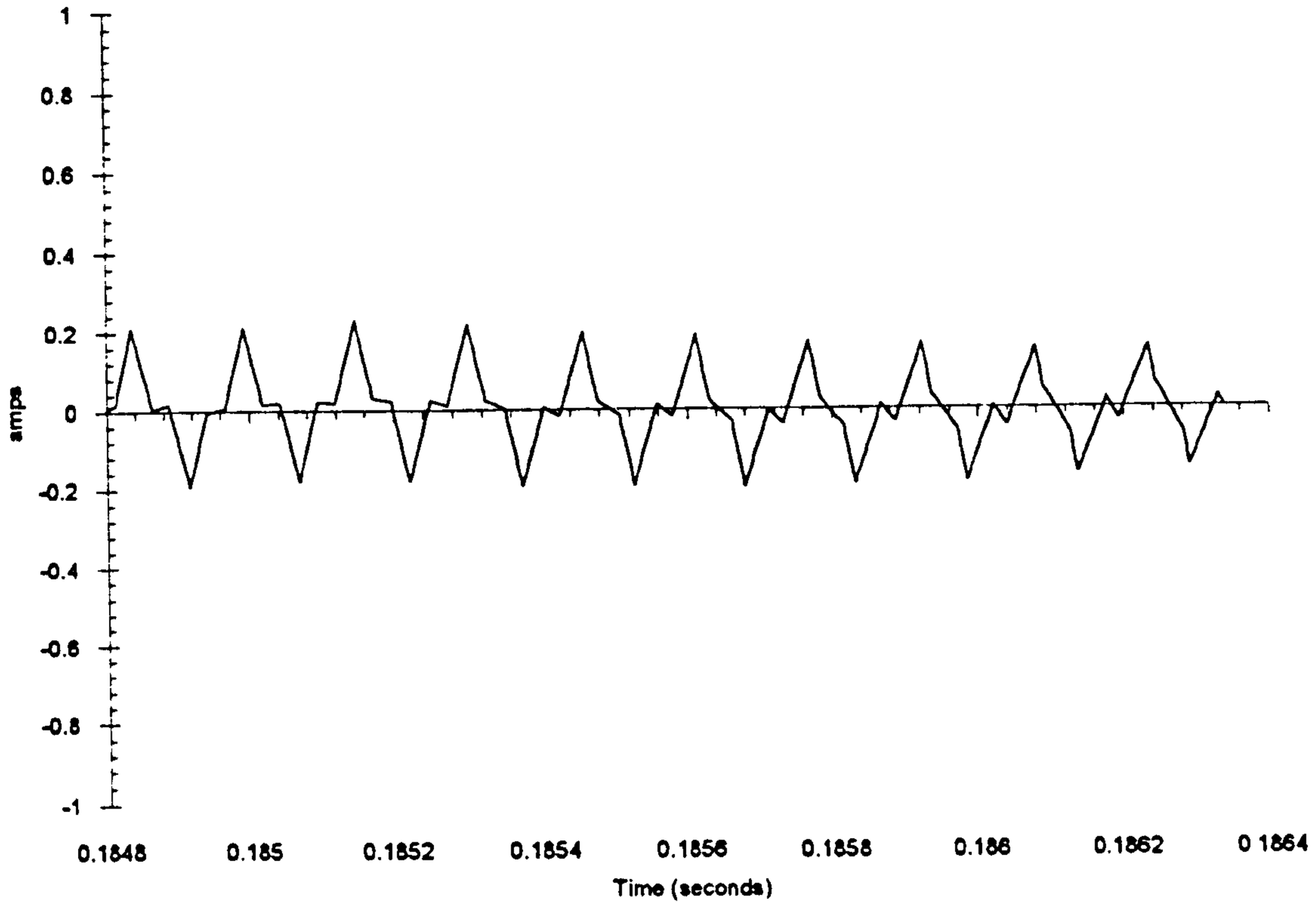


Figure 3.6.9: D-Axis Rotor Current Constant 1μs Time Step

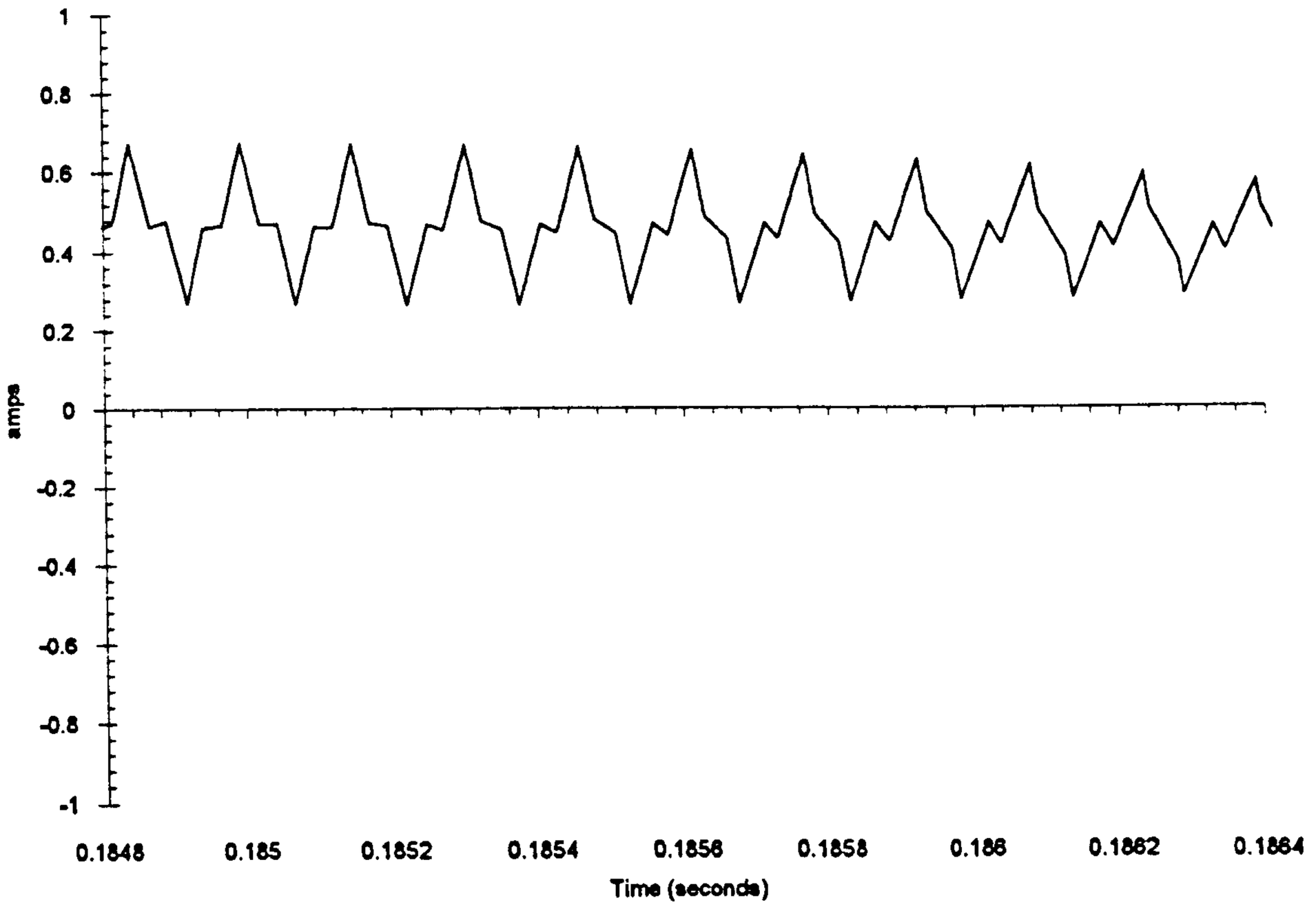


Figure 3.6.10: D-Axis Rotor Current Variable Time Step Model Using Euler

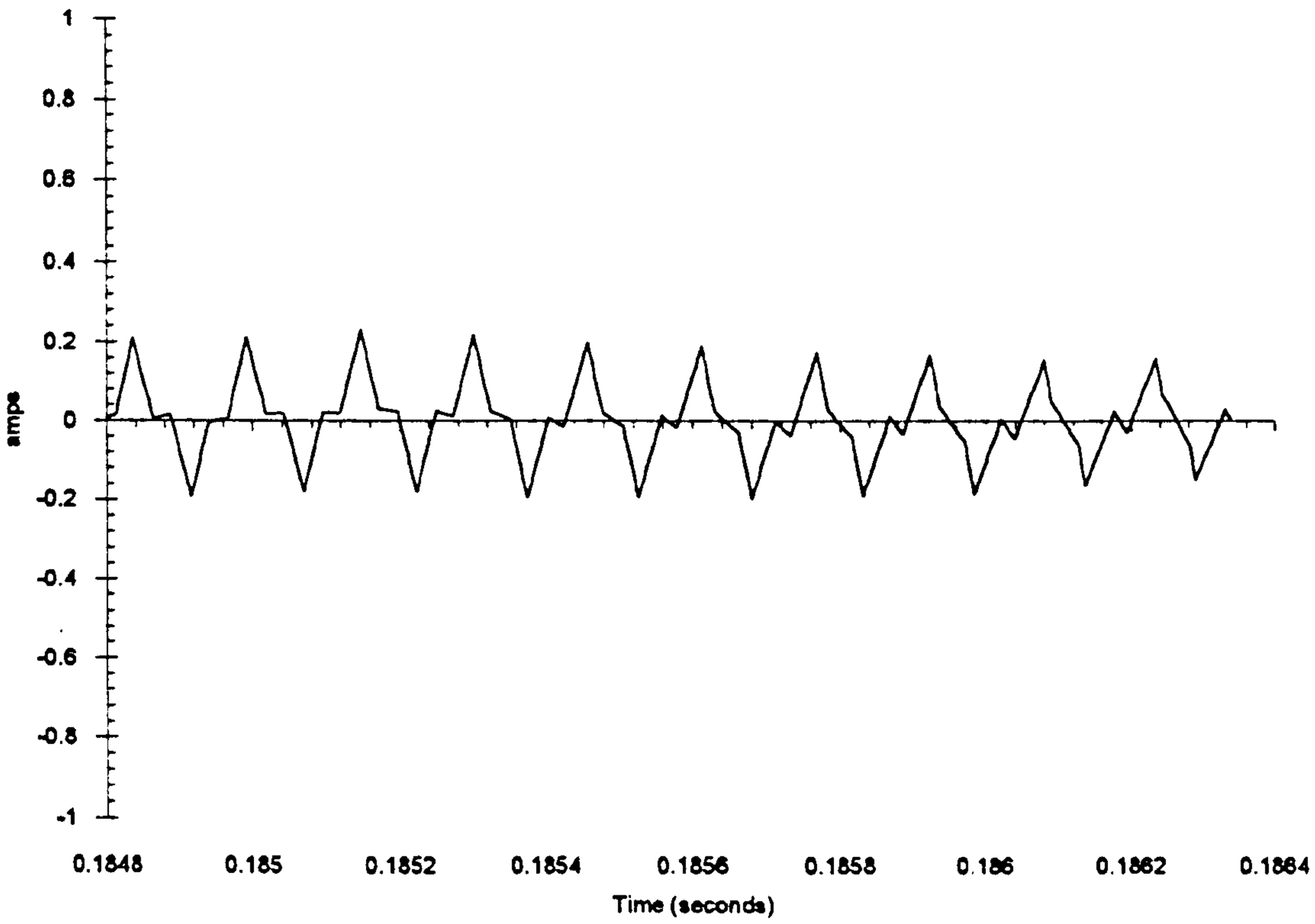


Figure 3.6.11: D-Axis Rotor Current Constant 1μs Time Step

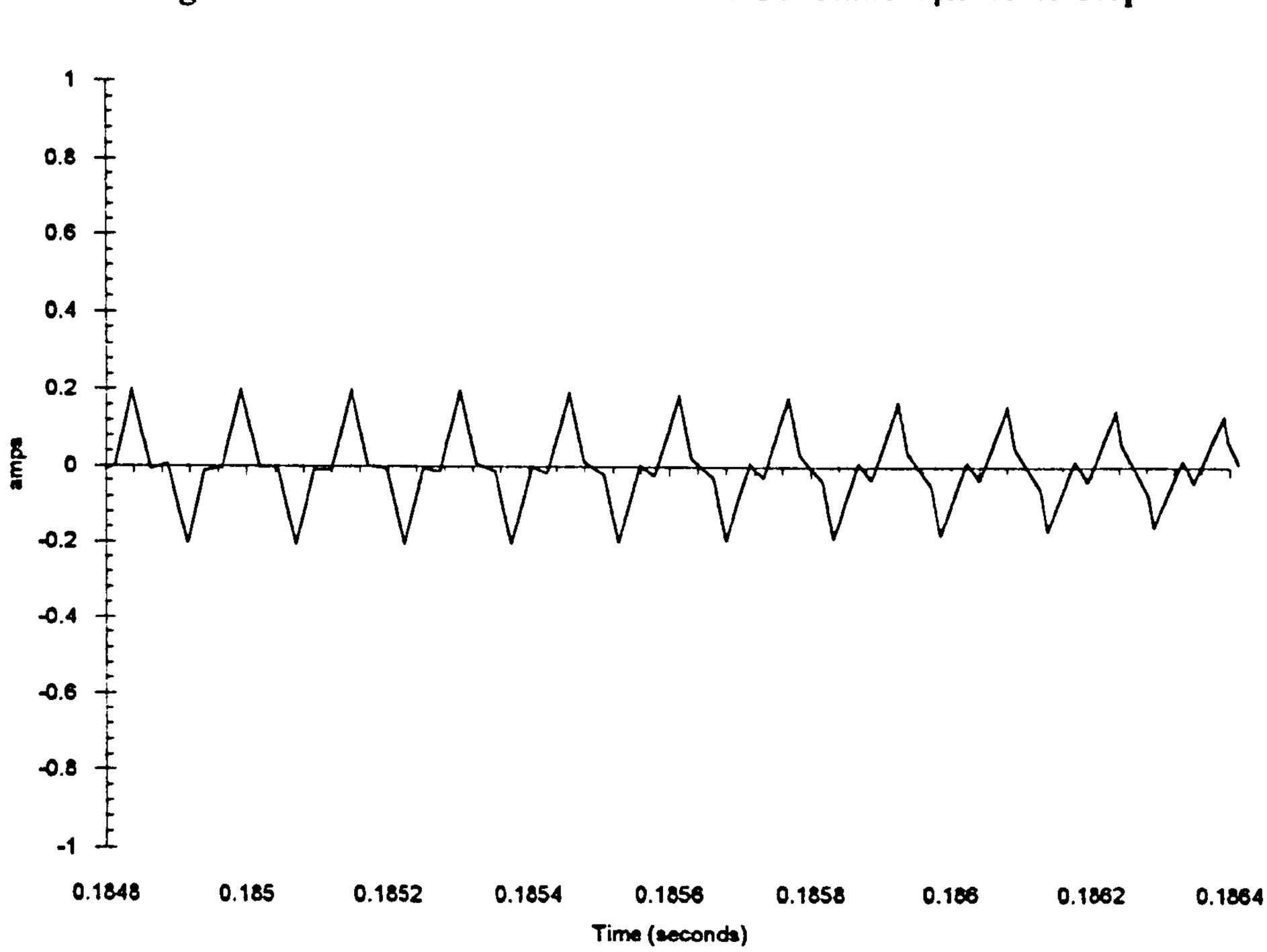


Figure 3.6.12: D-Axis Rotor Current Variable Time Step Model Using R-K

The previous figures clearly show that the variable time step model can perform equally as well as a constant time step model, the advantage that the variable time step model has is in its execution time. Over the switching period of 154μs, the motor differential equations have to be solved 154 times for fixed time step of 1μs. Over the same switching period using the variable time step approach the motor differential equations are only solved 7 times, thus the variable time step method executes approximately 22 times faster if the same solver is used for both. If the Runge Kutta method is employed for the variable time step model this gain reduces to approximately 4.5 times faster, which is still a considerable gain. The simulation could be accelerated even further by incorporating the Runge Kutta routine only when the switching state times were considered too large for the Euler method.

3.7 Summary

This Chapter has discussed the modelling techniques used to simulate the three phase voltage source inverter. The three main parts of the inverter i.e. the modulator the DC link and the bridge have all been simulated. The inclusion of deadtime modelling has been made to the bridge simulation and the idea of a variable time step model has been introduced. The advantage of the variable time step model is useful when the simulation is implemented into a real time environment and this will be demonstrated in chapter six.

4. CONTROLLER

4.1 Introduction

The ideal control properties of an electric drive are considered to be independent control of the flux and torque. These properties are readily available with a DC machine since the torque can be controlled by the armature current, and the flux can be controlled by the field current. The DC machine, however, has the inherent mechanical disadvantage of its commutator and brushes which require periodic maintenance and prohibit the use of the machine in certain applications. The AC machine does not suffer these disadvantages, and in the case of a cage induction machine offers a much lower cost than a DC machine. An AC machine may have an advantage over DC machines in terms of cost and construction but it fails to provide the flexibility in terms of control of the DC machine. For this reason much work has been carried out into developing a control method which allows AC machines to be controlled in the same ideal manner as the DC machine, thus providing all the advantages of machine simplicity together with flexibility of control.

The two most popular methods of controlling the induction machine are Volts/Hertz Control and more recently Vector Control (Field Orientated Control). One of the objects of this project is to provide a real time simulation environment in which drive controllers may be developed. In order to observe the behaviour of the simulation both a simple Volts/Hertz Controller and a more complex Vector Controller have been developed and tested. This chapter will discuss both of these controllers.

4.2 Volts/Hertz Control

Volts/Hertz is a simple open loop control method in which the speed of the induction machine is controlled by the frequency of the supply to the machine. The amplitude of the applied voltage to the machine is varied with the frequency so that the ratio of volts to hertz is essentially constant. The ratio of the voltage to frequency is maintained constant in order to maintain a constant flux in the machine. Consider the following equations:-

$$V = K_1 \psi_{ag} f \qquad (4.2.1)$$

$$T_e = K_2 \psi_{ag} f_{slip} \qquad (4.2.2)$$

It can be seen from the above equation 4.2.1 that if the ratio of V to f is kept constant then the air gap flux ψ_{ag} will remain constant, if the air gap flux is constant then from 4.2.2 it can be seen that a linear relationship between the electromagnetic torque T_e and the slip frequency f_{slip} results. The torque slip curve of the motor can be repeated along the frequency axis as shown in the following figure.

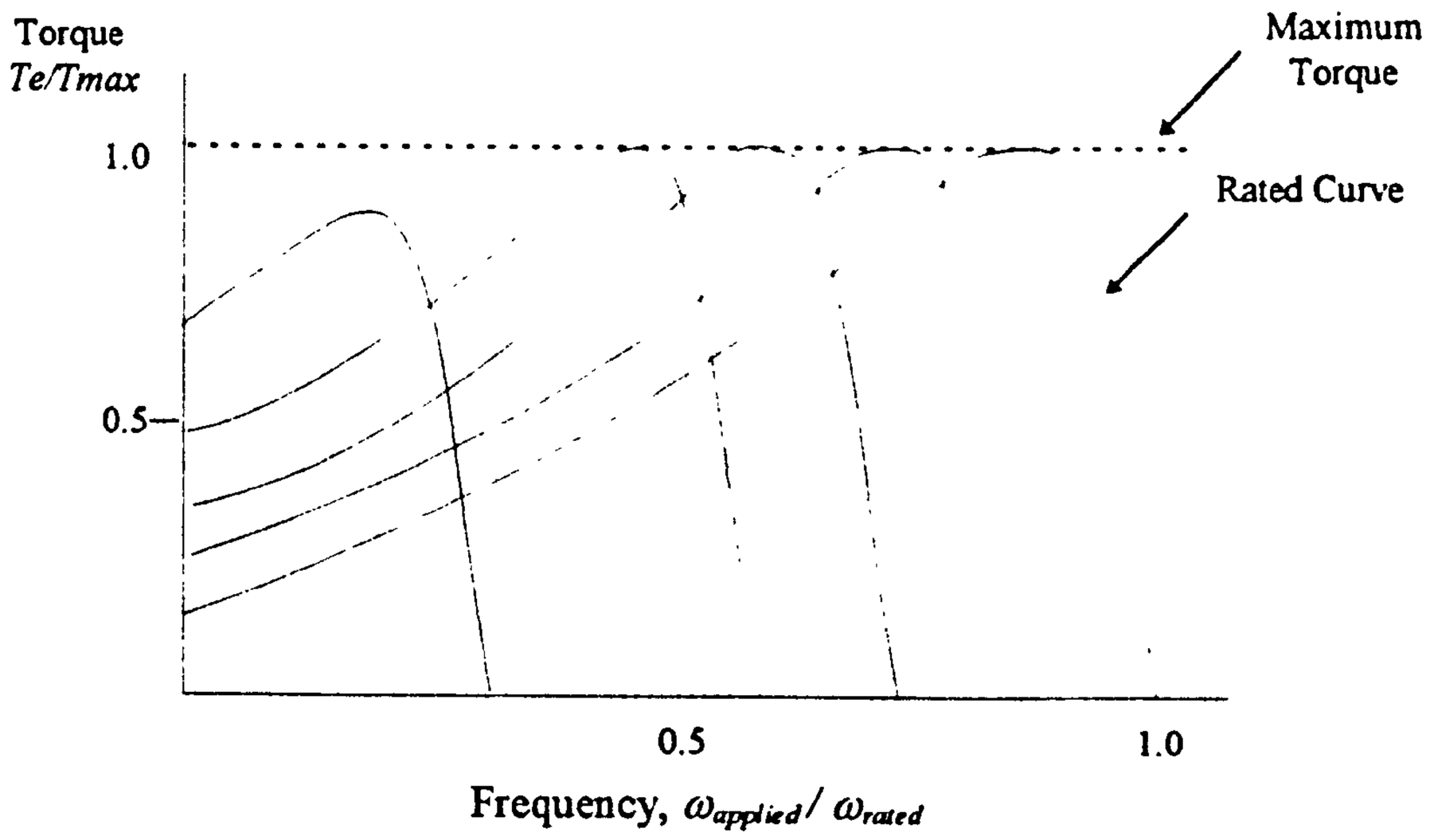


Figure 4.2.1: Volts/Hertz Control Torque Slip Curves

The curves shown in the above figure show that the maximum torque is maintained as the applied frequency is reduced to a certain level, below this level the torque is reduced. This reduction in the torque is due to the stator resistance becoming significant as the applied stator volts are reduced. To overcome this a boost voltage is applied at low speeds so that maximum torque can continue to be produced.

A simple Volts/Hertz control scheme with a low frequency boost voltage is shown in the figure below.

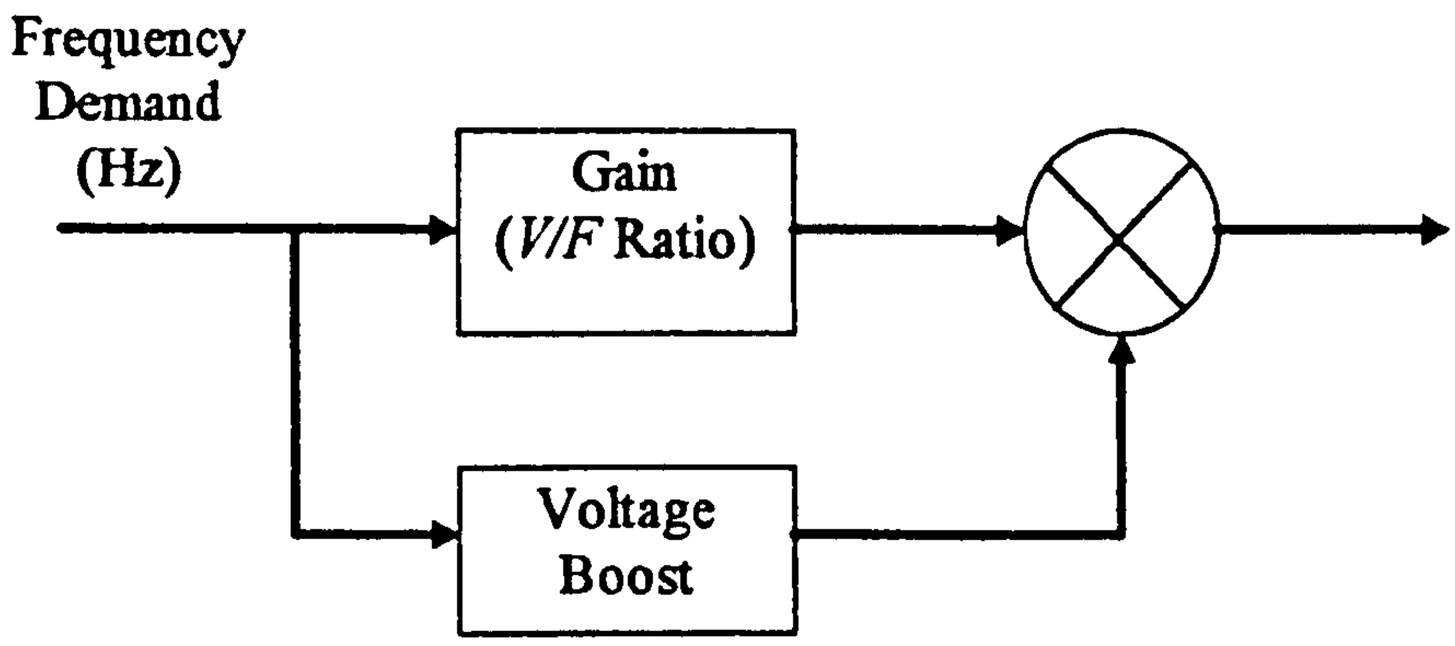


Figure 4.2.2: Volts/Hertz Control Block Diagram

The voltage boost in this controller is dependent upon the frequency demand and produces the following *V/F* relationship.

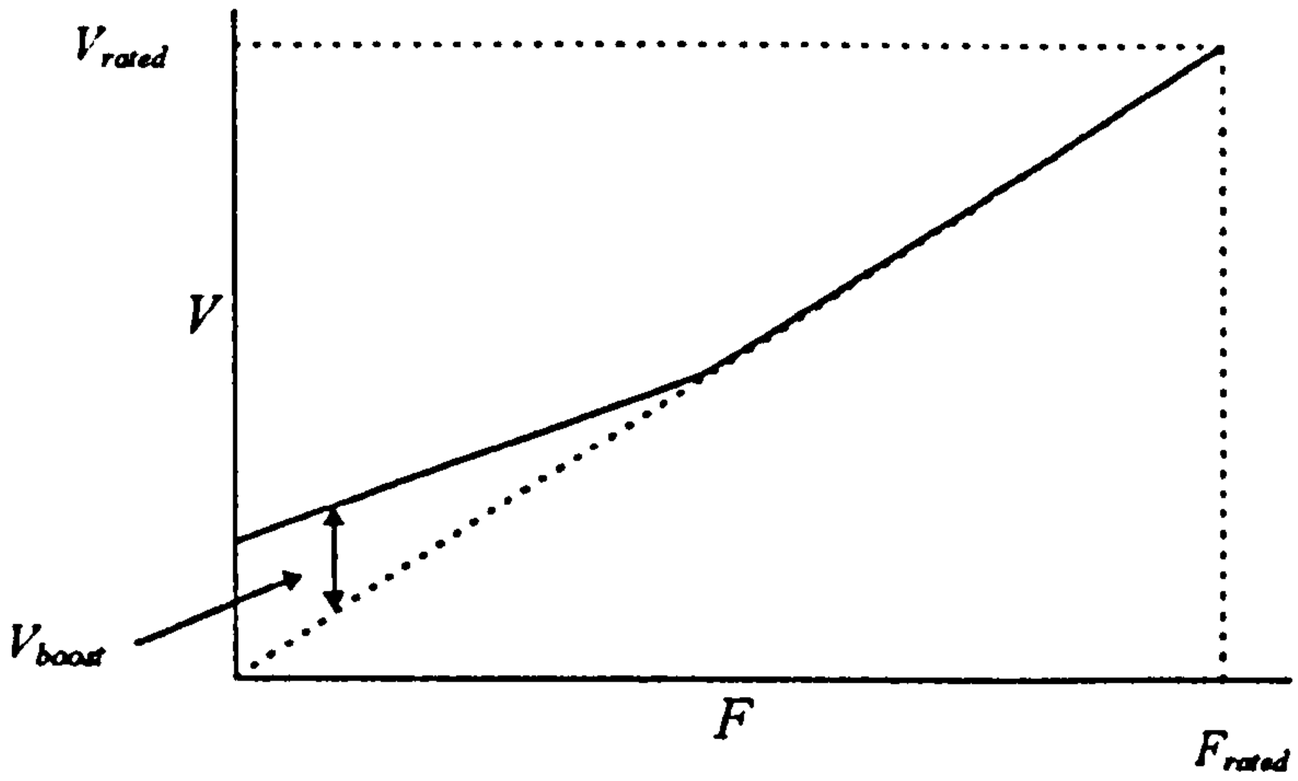


Figure 4.2.3: Volts/Hertz Control V/F Ratio

Figure 4.2.4 shows the phase voltage applied to the motor model for an increasing demand in frequency, the effect of the boost at the low frequency is highlighted.

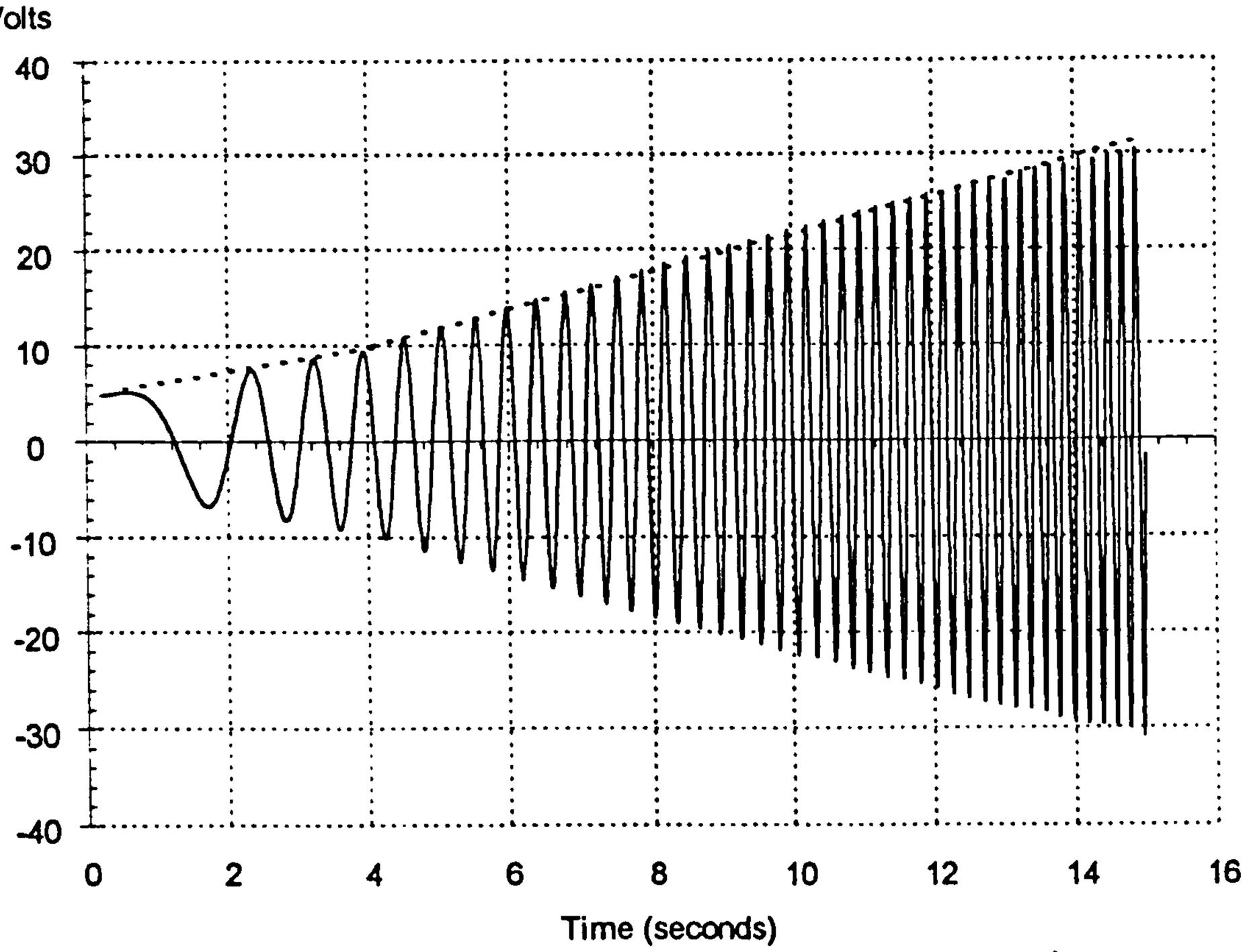


Figure 4.2.4: Volts/Hertz Control Phase Voltage During Frequency Increase

The following results are taken from the simulation of the motor model driven from the open loop Volts/Hertz controller, at this stage no inverter simulation is included and the three phase output demand of the controller is applied directly to the motor model. The results show the response of the machine to a sudden step change in demand frequency from 3.33Hz to 16.66Hz, the figures show the applied phase voltage, the resultant phase current and the rotor angular velocity. Because the controller is open loop i.e. no actual values are returned to the controller, the sudden change in demand results in high phase currents.

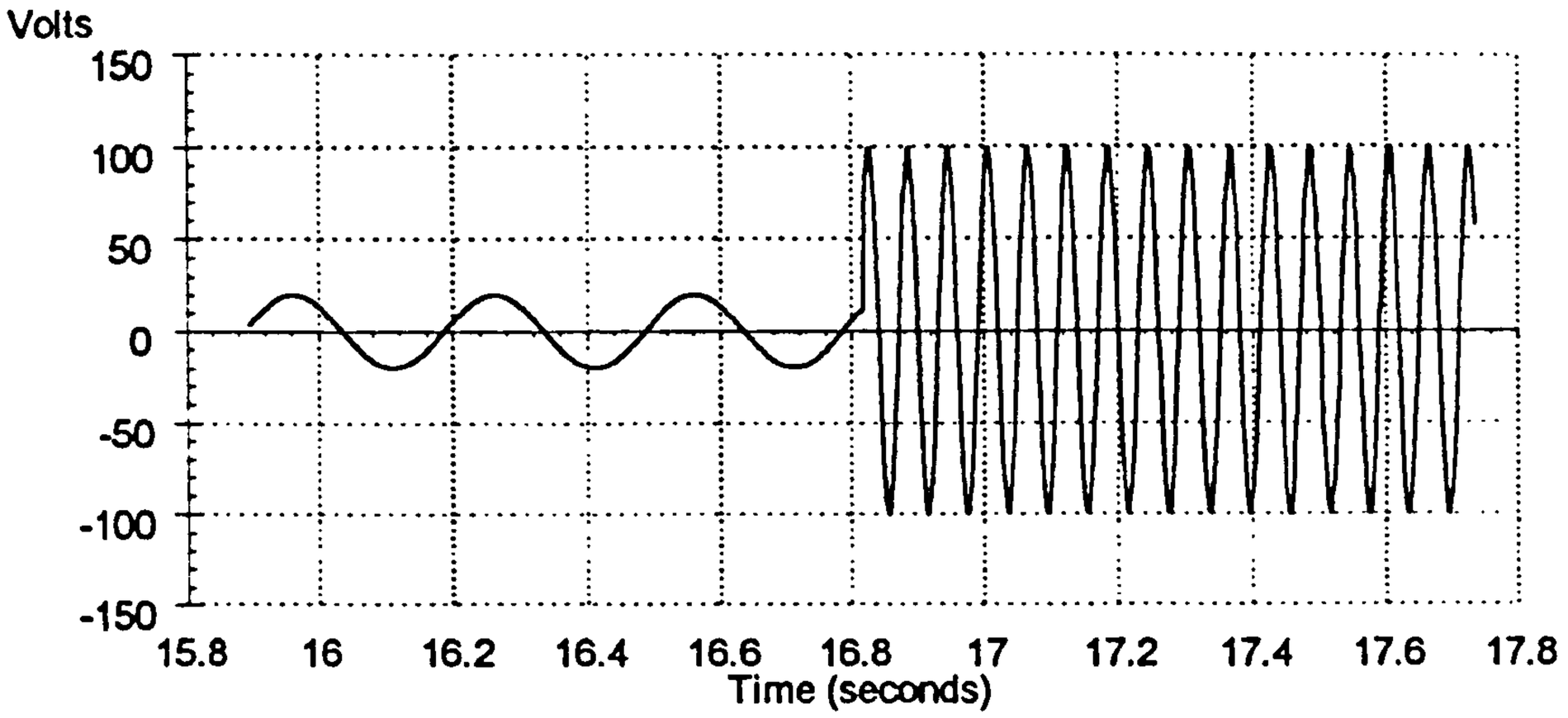


Figure 4.2.5: Phase Voltage for 3.33Hz to 16.66Hz Step in Demand

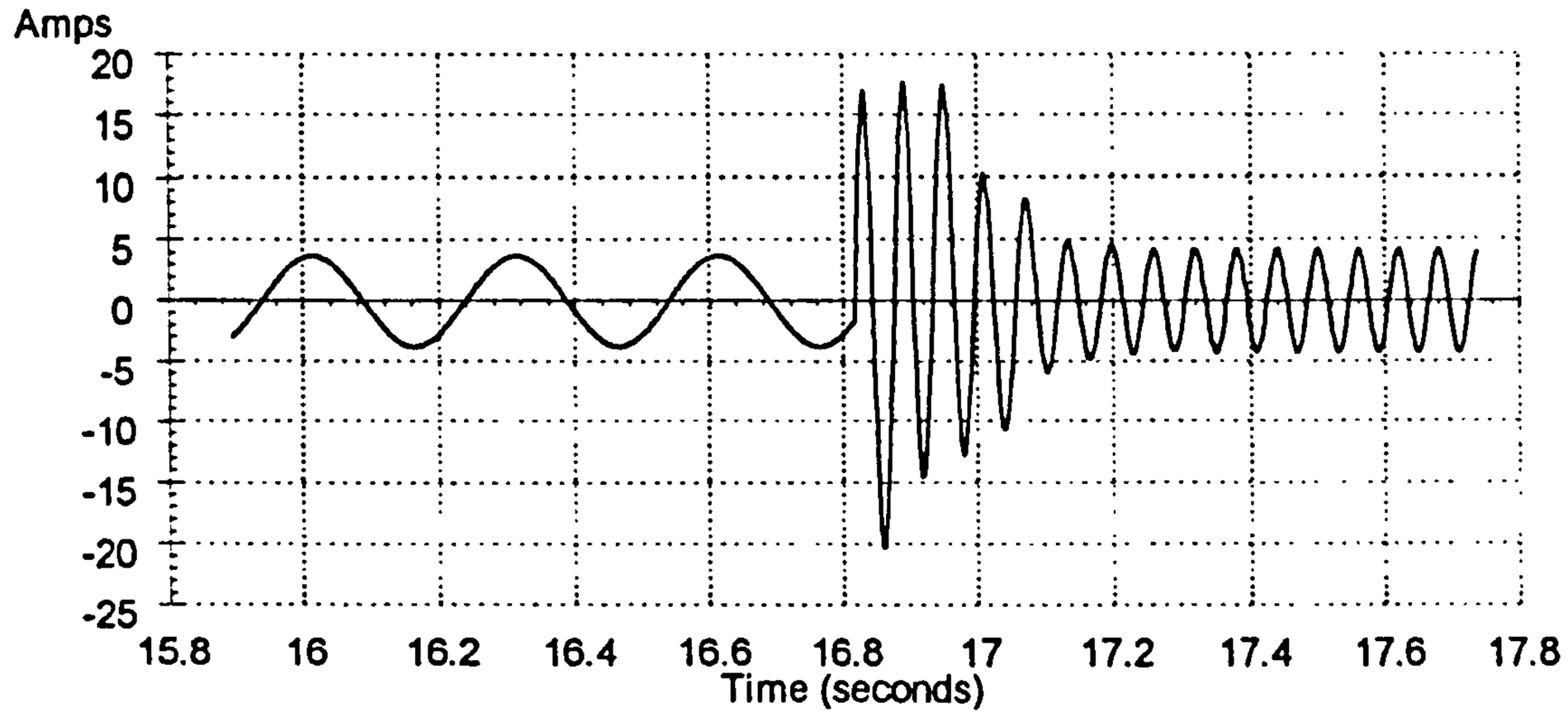


Figure 4.2.6: Phase Current for 3.33Hz to 16.66Hz Step in Demand

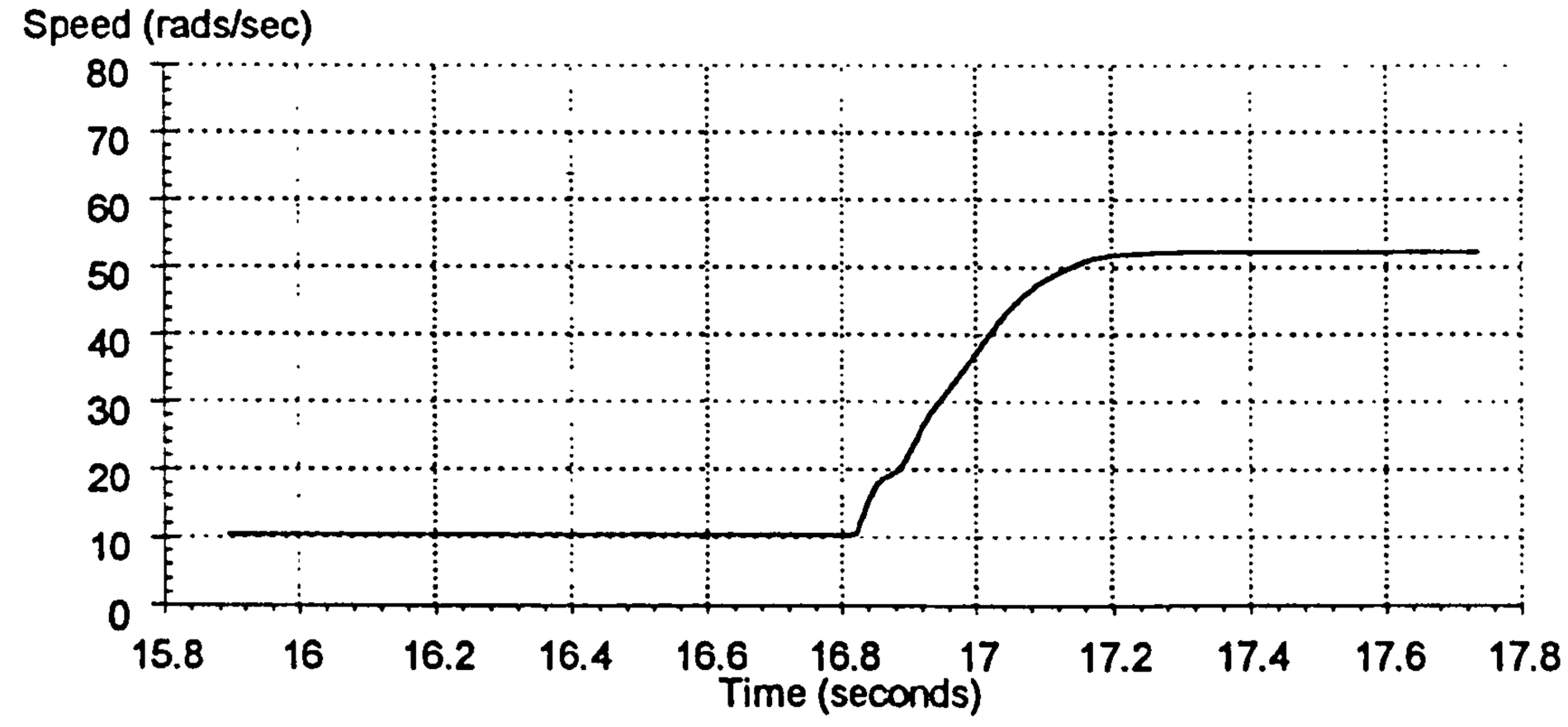


Figure 4.2.7: Rotor Speed for 3.33Hz to 16.66Hz Step in Demand

The controller can now be connected to the motor model via the simulation of the three phase inverter which was described in chapter three. This will allow the simulation results to be compared to the actual values obtained from the real drive system. The following results show the steady state response of the simulated and the actual systems.

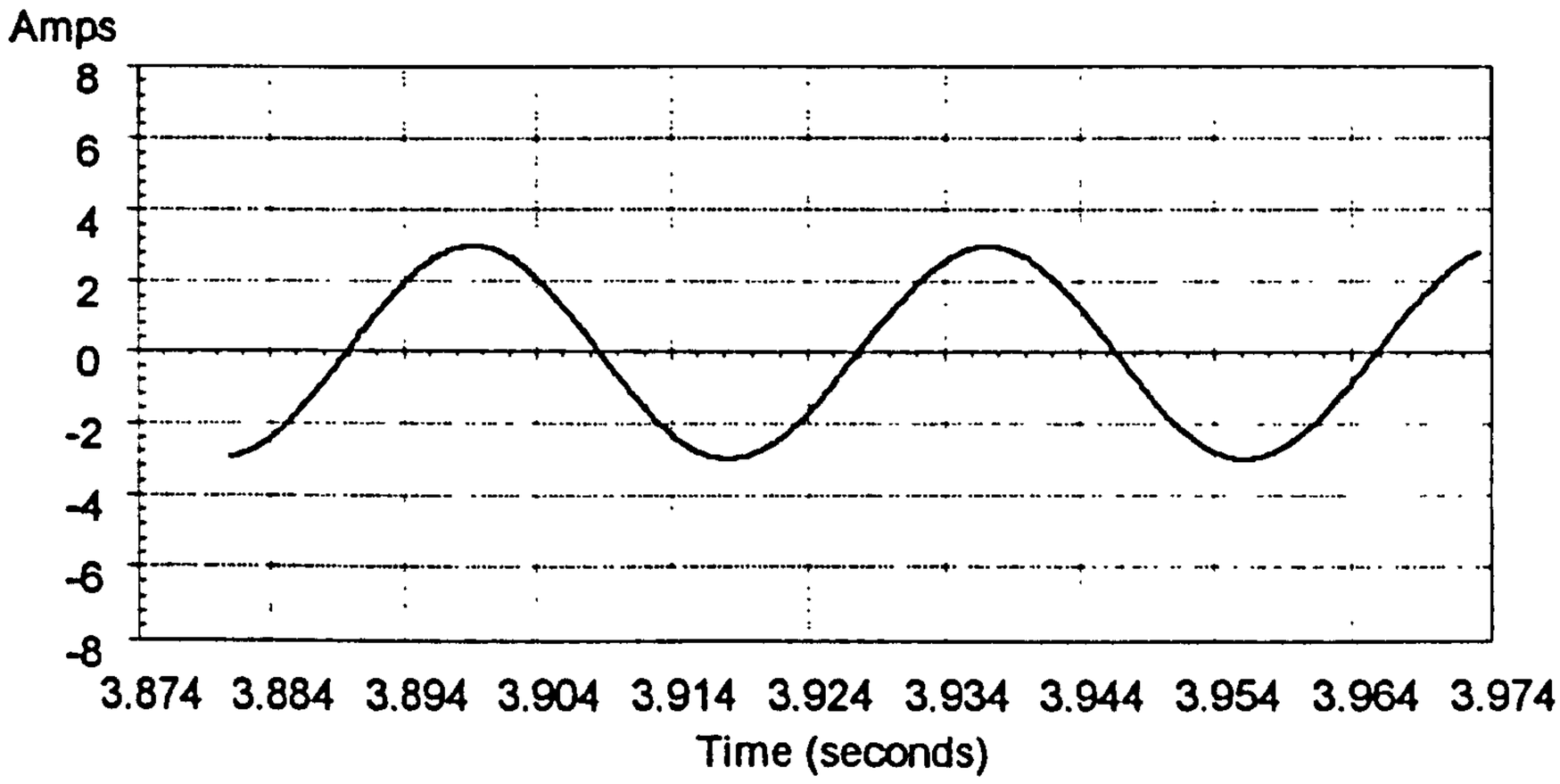


Figure 4.2.8: Volts/Hertz Steady State Simulated Current 26.66Hz Demand

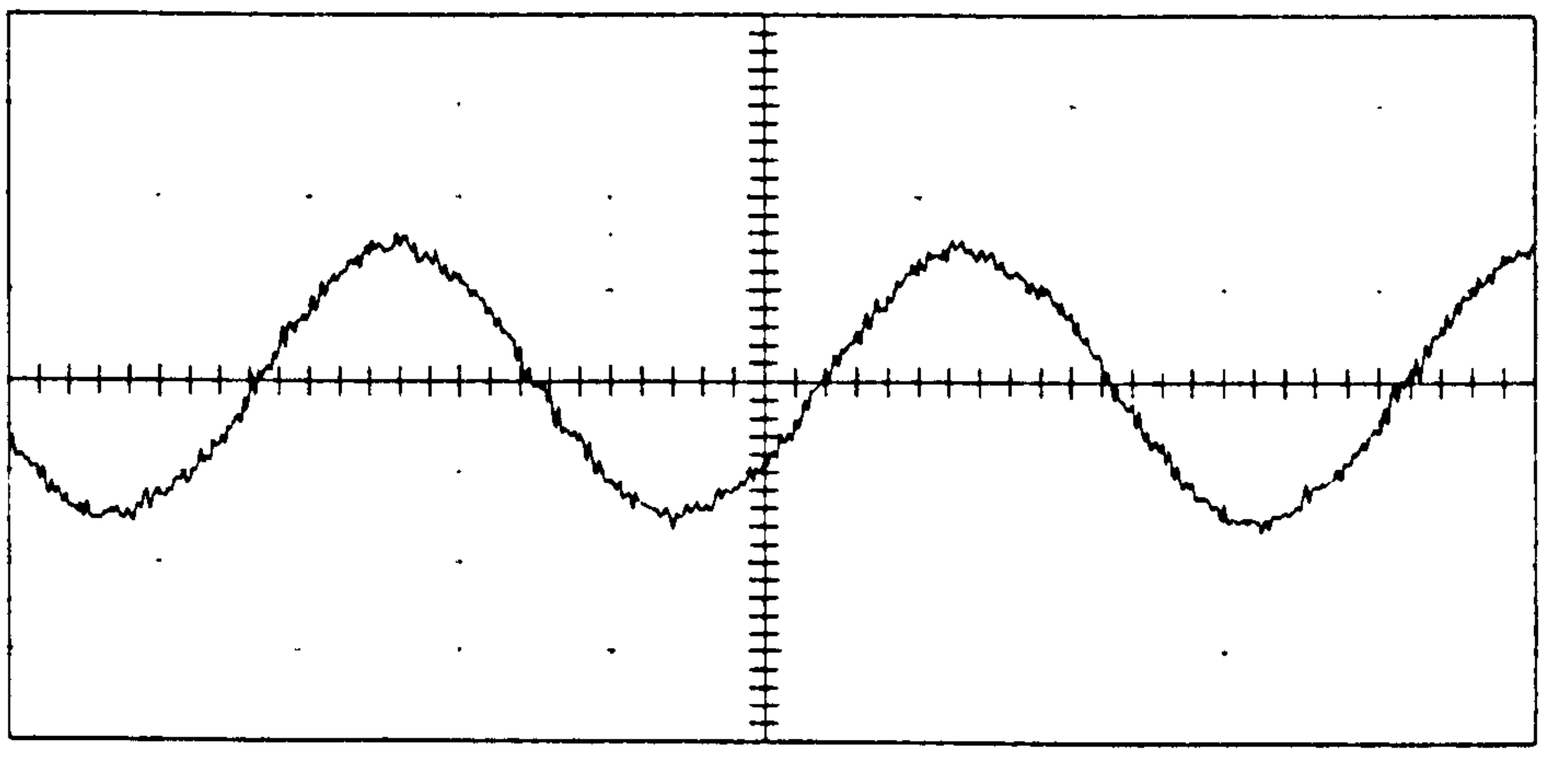


Figure 4.2.9: Volts/Hertz Steady State Actual Current 26.66Hz Demand

The steady state case showed the simulation to be close to the actual. The transient response of the two drive systems can now be compared. The transient results were taken for a step increase in the input frequency demand of 13.33Hz to 26.66Hz and a step decrease in demand of 26.66Hz to 13.33Hz. During the transient tests the phase current, rotor angular velocity and the DC link voltage were measured.

With no-load and therefore small slip the rotor angular velocity should approximately follow the demand frequency as follows:-

$$Speed\ (rads/sec) = (Supply\ Frequency \times 2\pi) / pole\ pairs \qquad (4.2.3)$$

The machine is a four pole motor, the rotor velocity should therefore be approximately 41.88 rads/sec for a frequency demand of 13.33Hz and 83.75 rads/sec for a demand of 26.66Hz.

As with the steady state case the simulation of the drive system provides a reasonably accurate representation of the actual drive system during the transient period. During both transients the phase current increases from approximately 3 amps to approximately 9 amps. The simulated and actual rotor angular velocities both increase from approximately 42 rads/sec to 72 rads/sec during the acceleration transient, and during the deceleration transient both angular velocities decrease from approximately 84 rads/sec to 48 rads/sec. The DC link voltage shows a slight reduction during the acceleration period as the machine current increases and more power is taken from the DC link. During the deceleration period when the machine is generating back into the DC link, the link voltage increases to the point at which the brake resistor is switched across the link to avoid any excessive over-voltage . The link voltage then reduces as the machine returns to the motoring mode of operation.

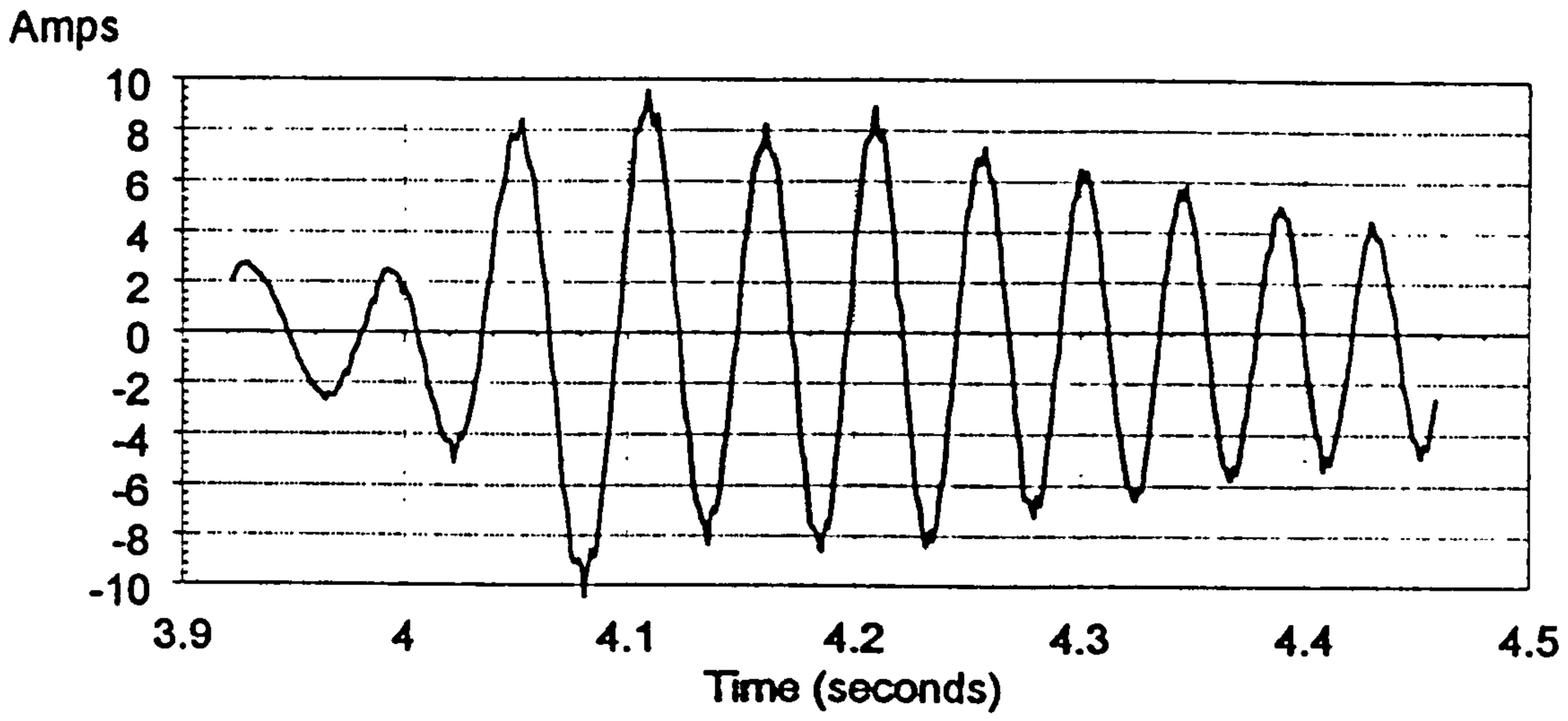


Figure 4.2.10: Transient Actual Current 13.33Hz to 26.66Hz Demand

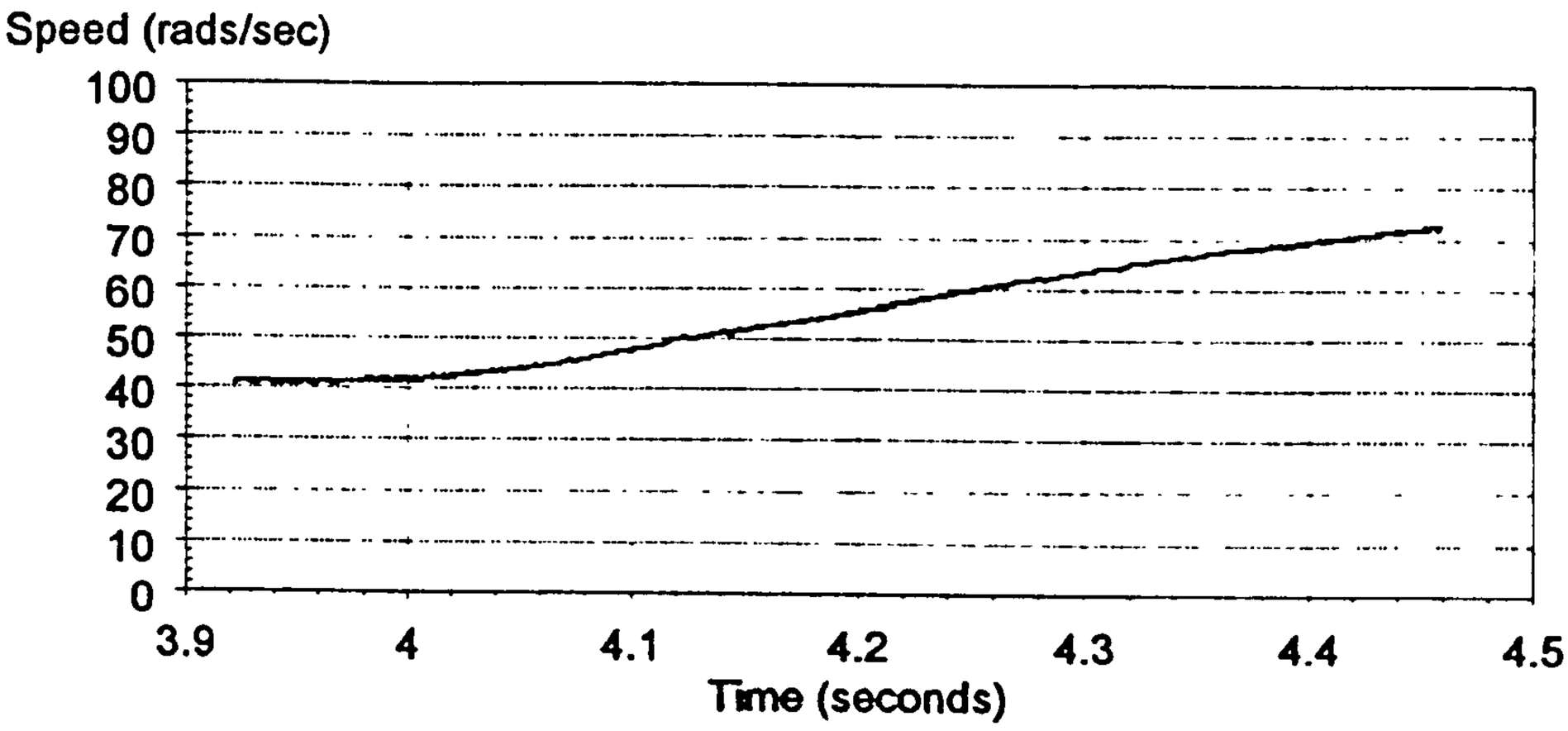


Figure 4.2.11: Actual Rotor Angular Velocity 13.33Hz to 26.66Hz Demand

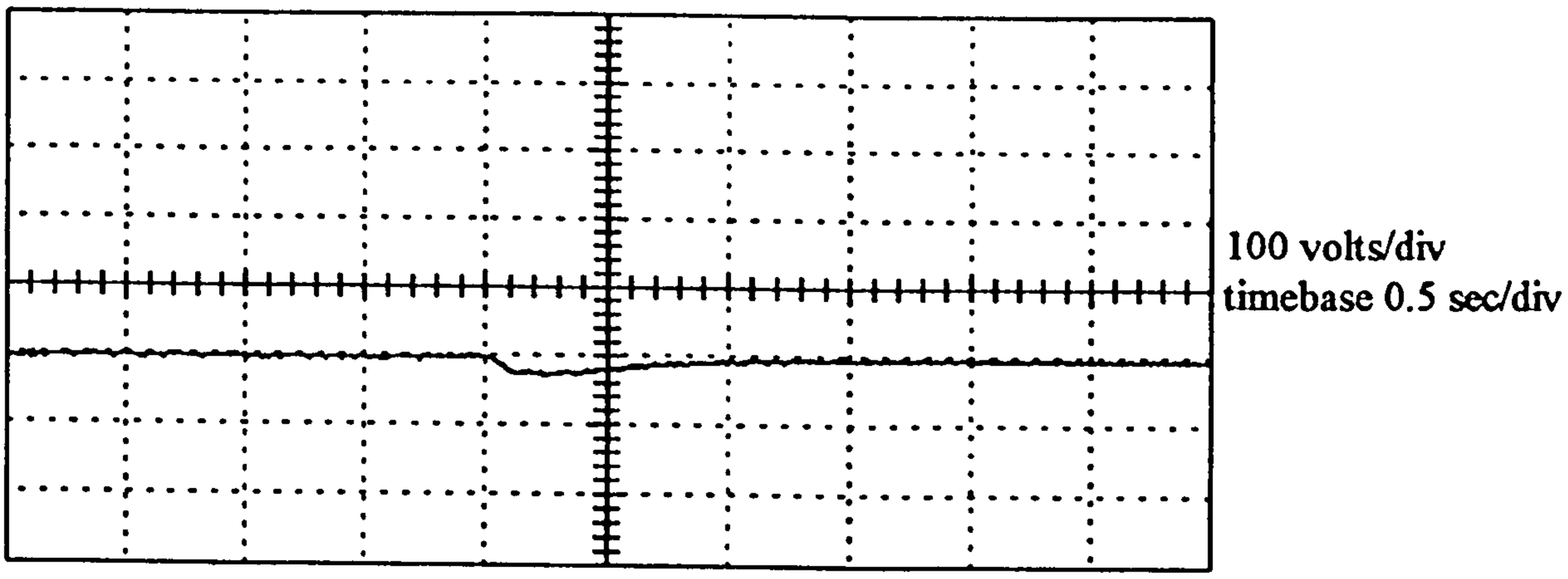


Figure 4.2.12: Actual DC Link Voltage 13.33Hz to 26.66Hz Demand

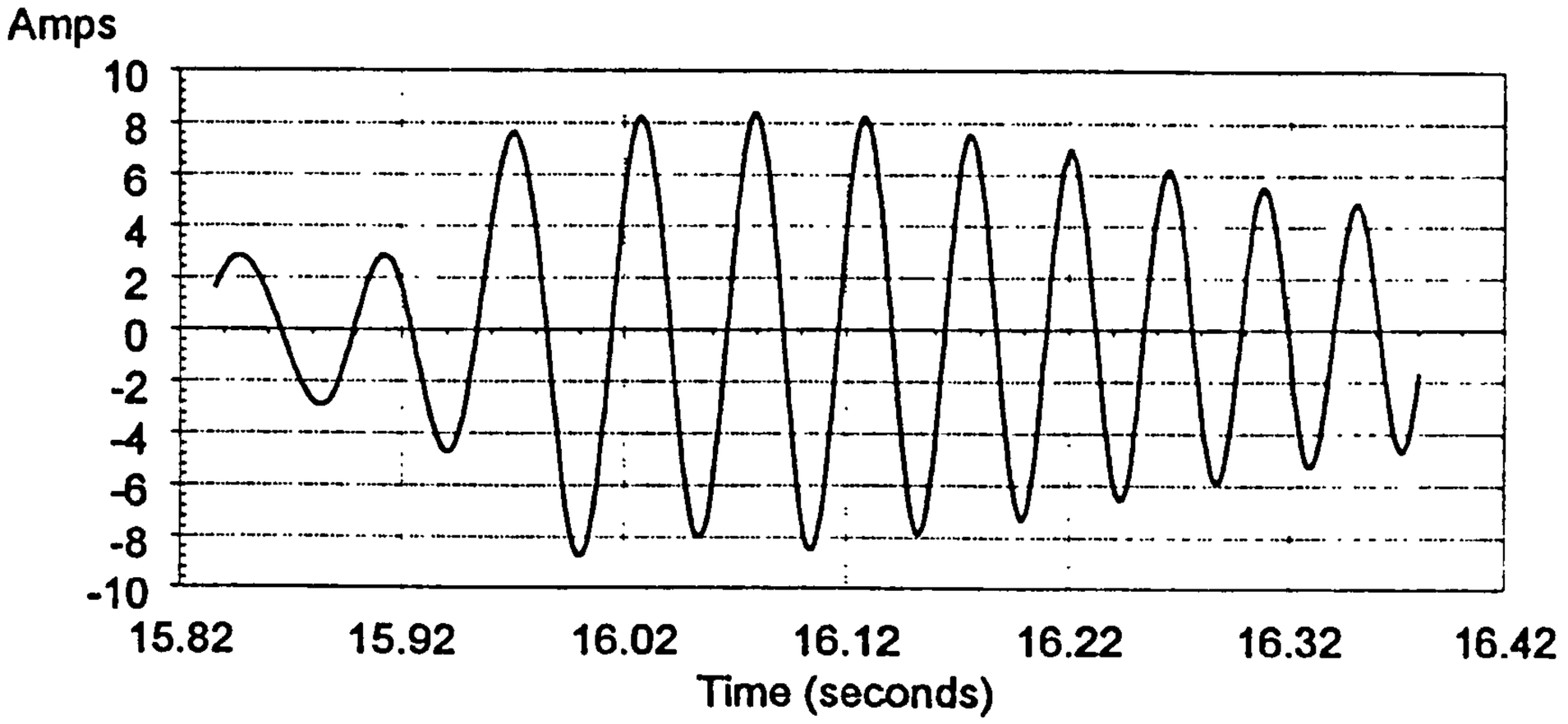


Figure 4.2.13: Transient Simulated Current 13.33Hz to 26.66Hz Demand

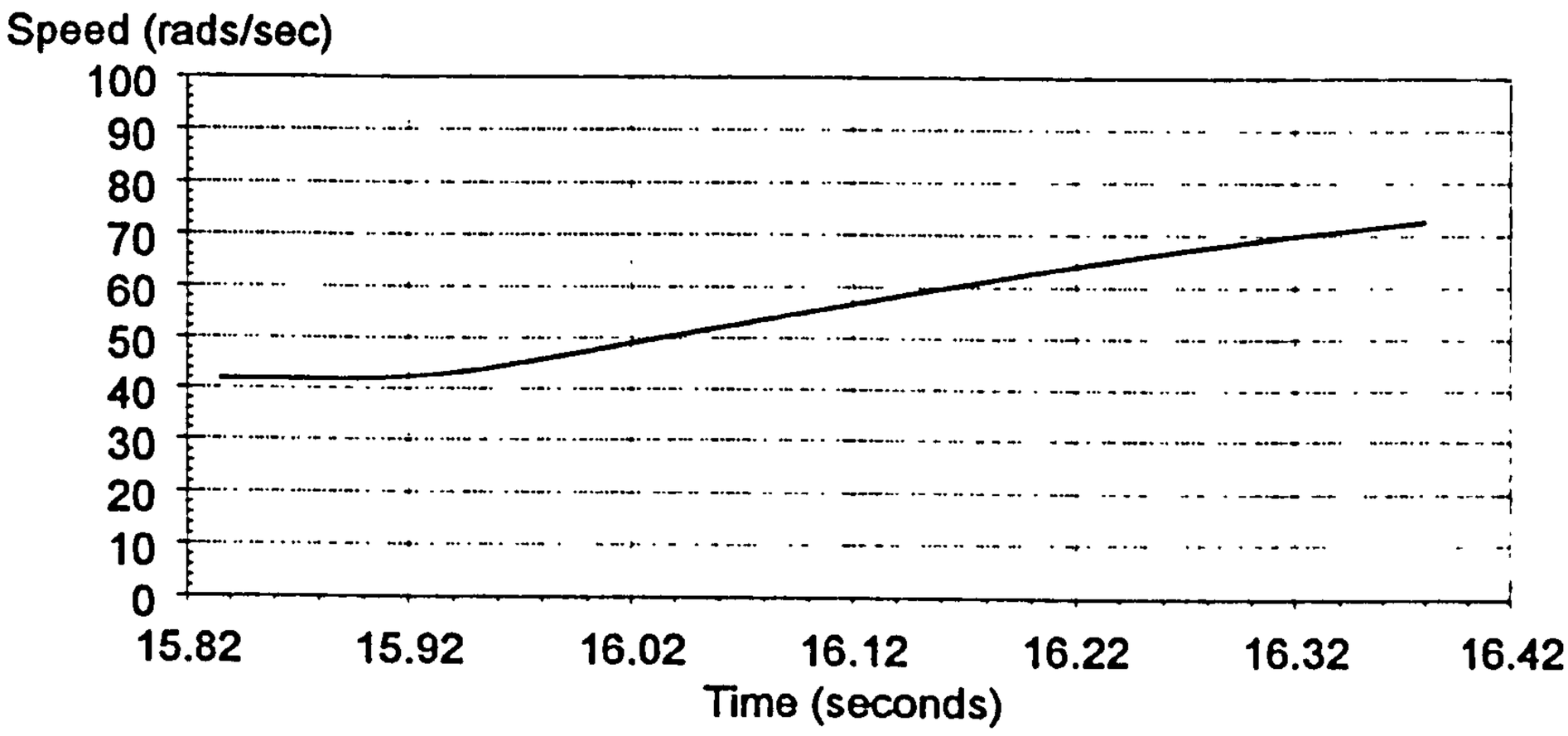


Figure 4.2.14: Simulated Rotor Angular Velocity 13.33Hz to 26.66Hz Demand

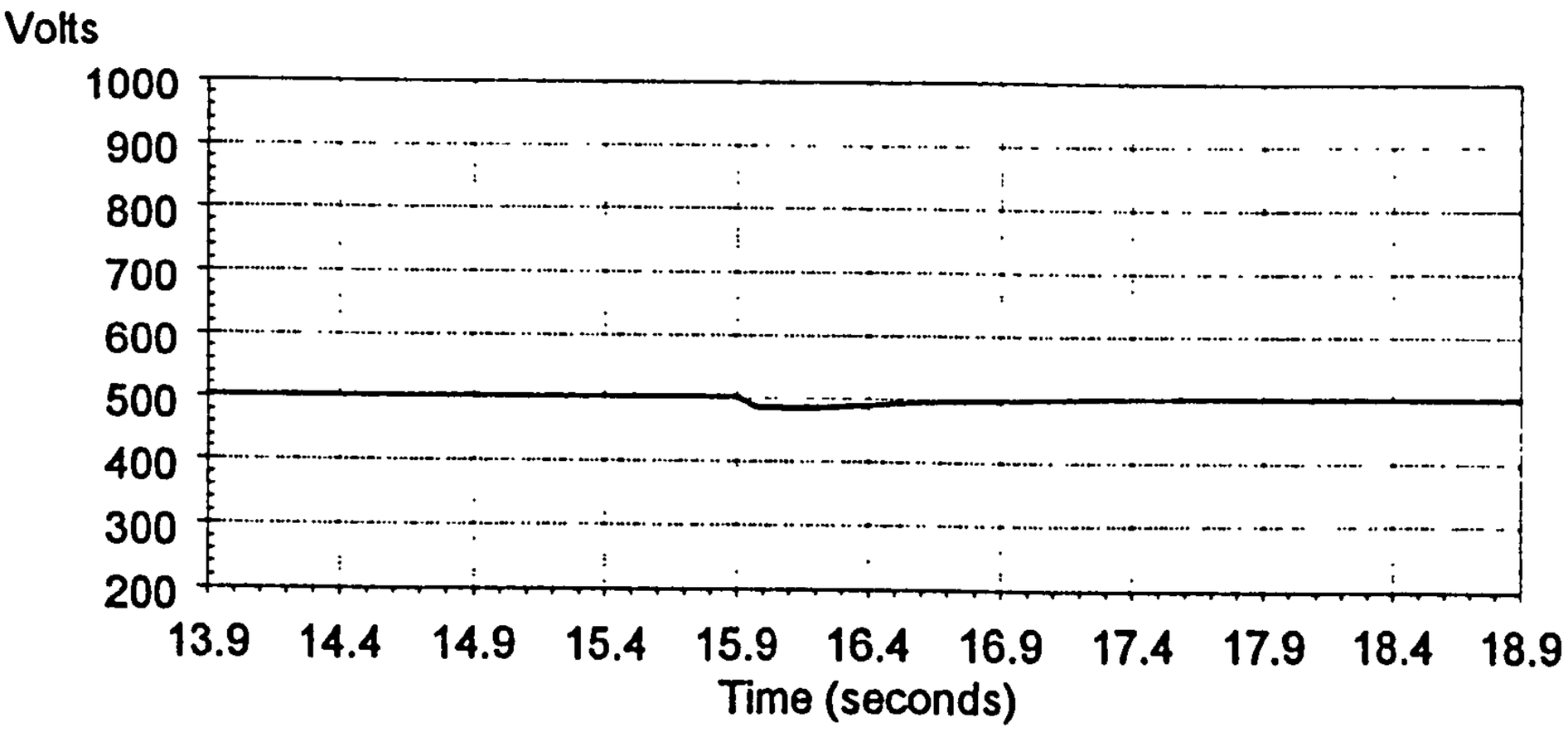


Figure 4.2.15: Simulated DC Link Voltage 13.33Hz to 26.66Hz Demand

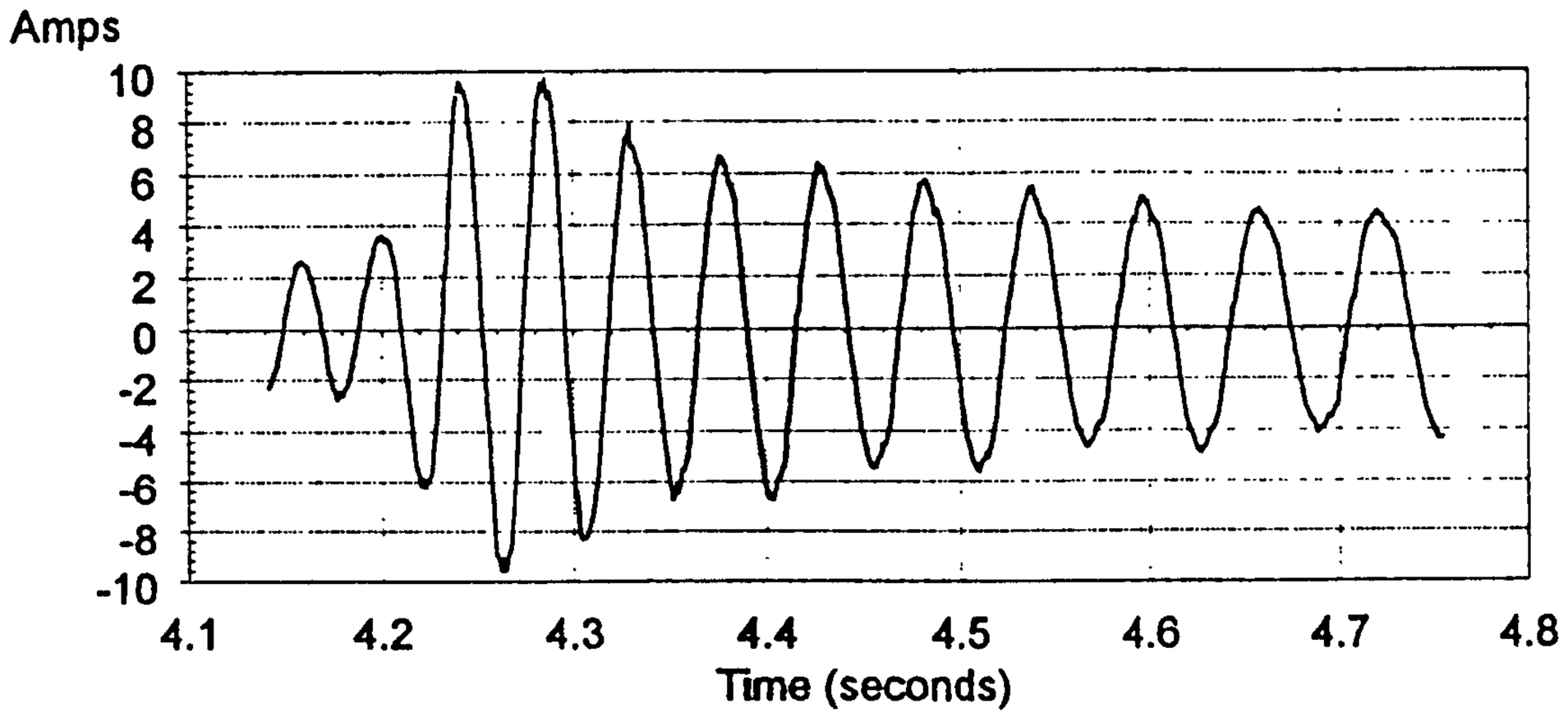


Figure 4.2.16: Transient Actual Current 26.66Hz to 13.33Hz Demand

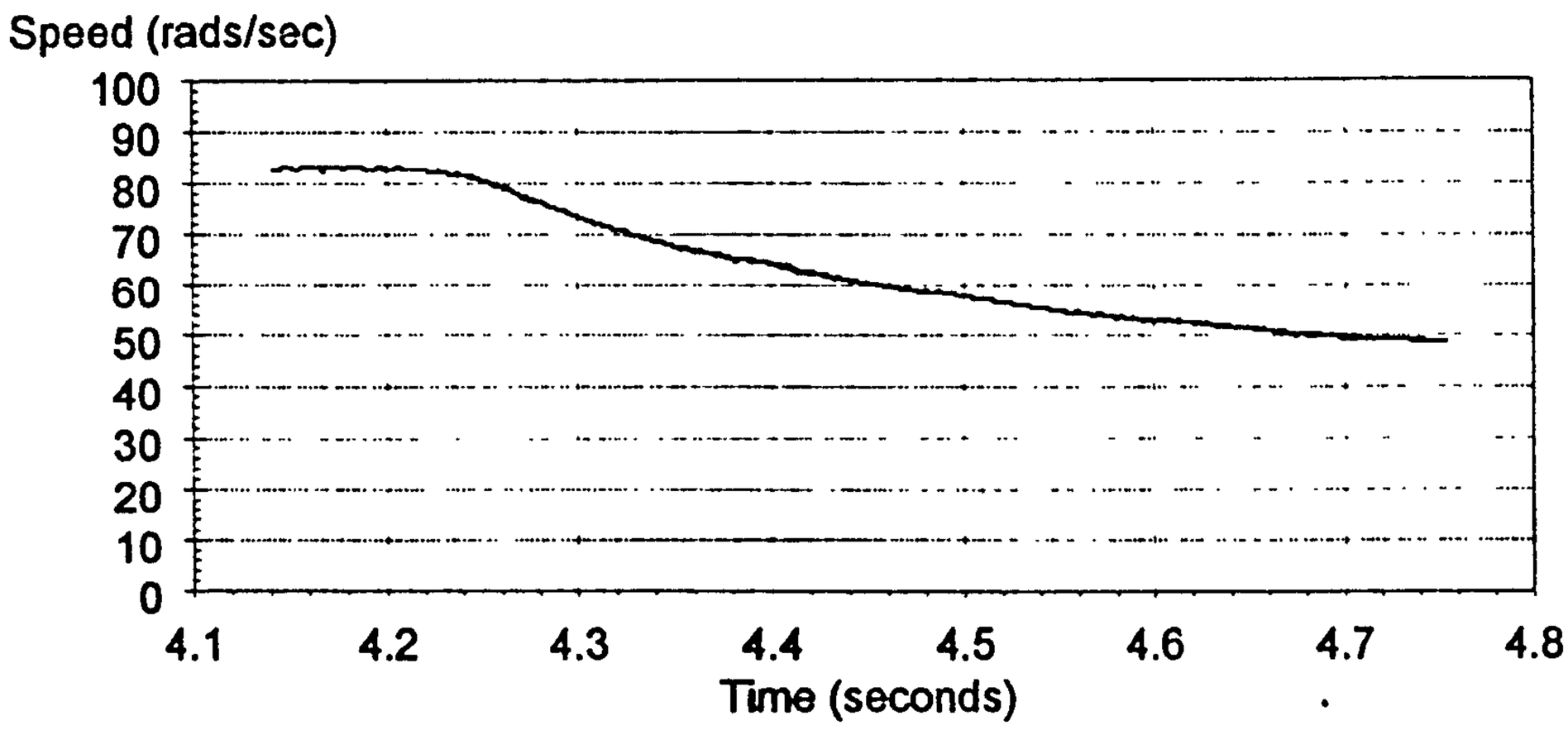


Figure 4.2.17: Actual Rotor Angular Velocity 26.66Hz to 13.33Hz Demand

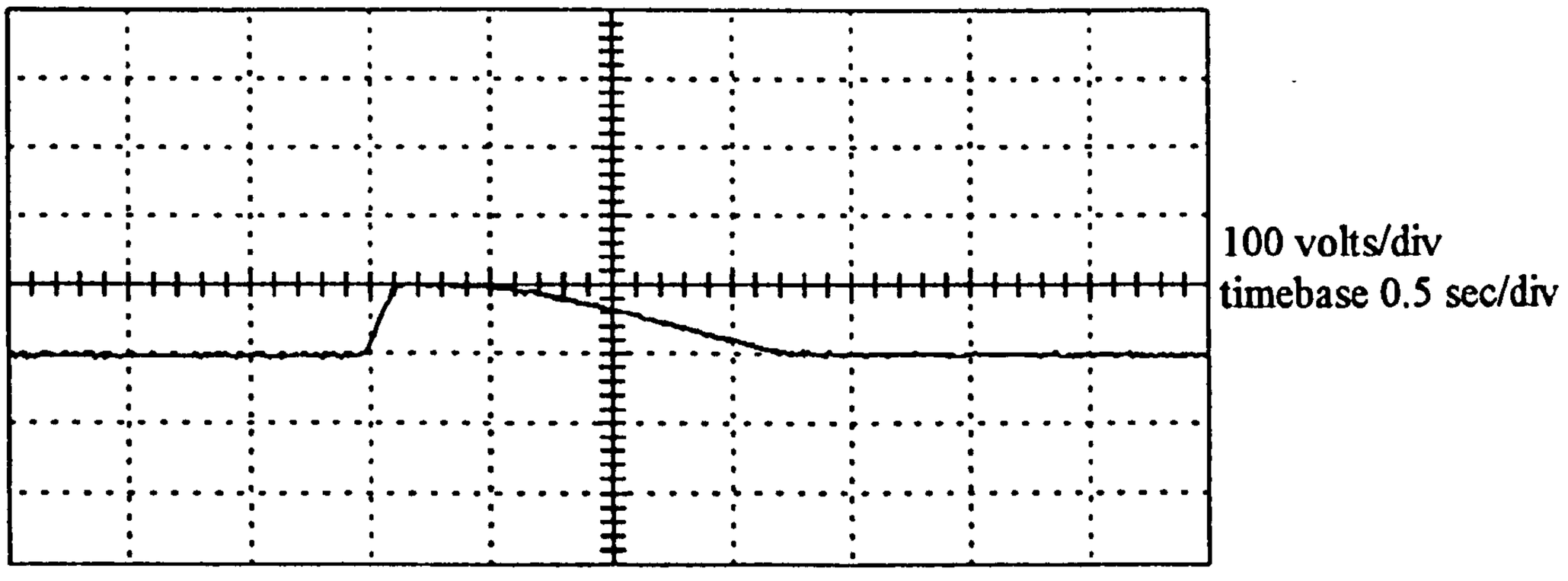


Figure 4.2.18: Actual DC Link Voltage 13.33Hz to 26.66Hz Demand

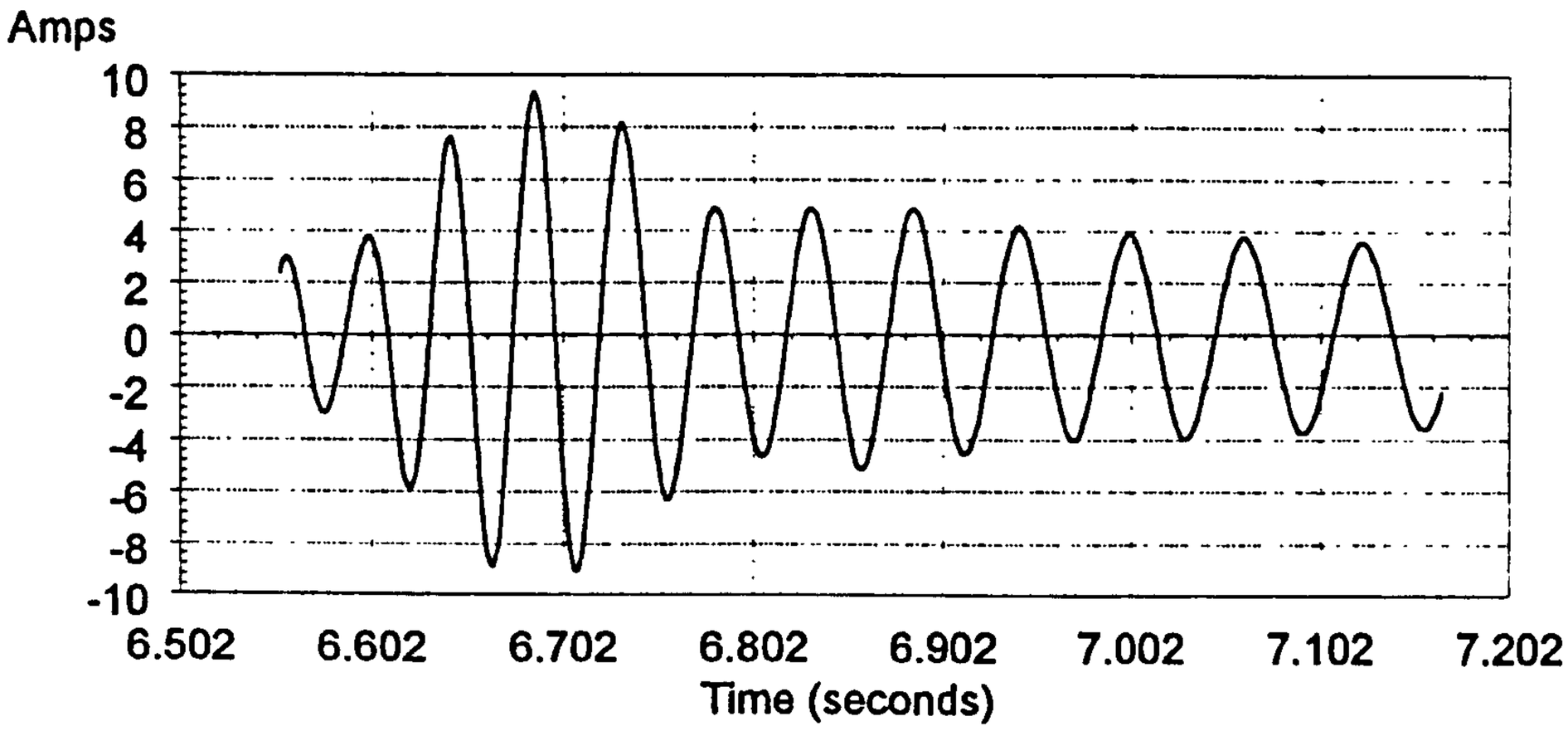


Figure 4.2.19: Transient Simulated Current 26.66Hz to 13.33Hz Demand

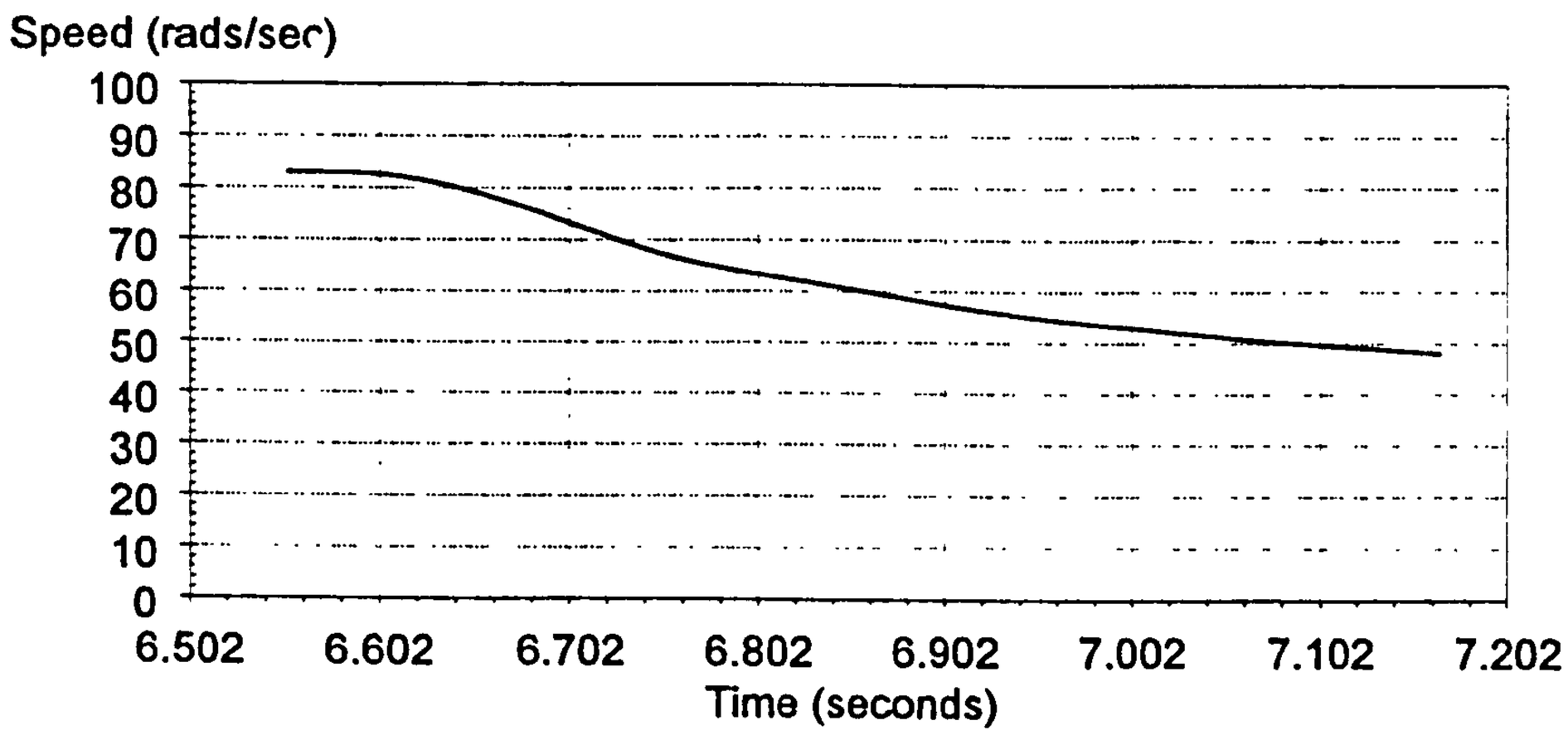


Figure 4.2.20: Simulated Rotor Angular Velocity 26.66Hz to 13.33Hz Demand

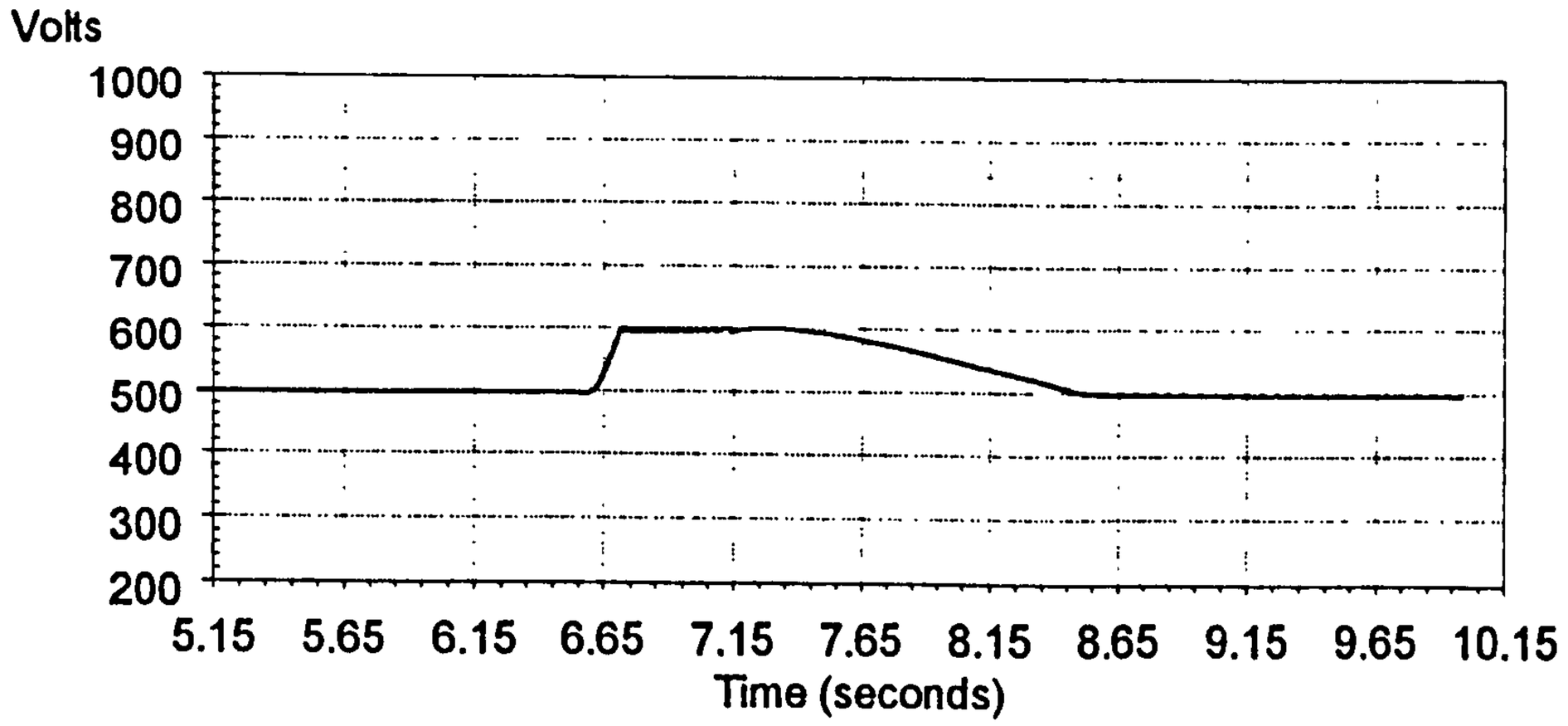


Figure 4.2.21: Simulated DC Link Voltage 26.66Hz to 13.33Hz Demand

The manner in which the simulated current and the actual current reduce after the initial increase, however, shows up error in the simulation. The actual phase current for the acceleration transient appears to reduce after the initial increase at a faster rate than the simulation, conversely during the deceleration transient the actual current reduces at a slower rate than the simulation. Consider the envelope formed by the peaks of both the simulated and actual currents during the deceleration transient:-

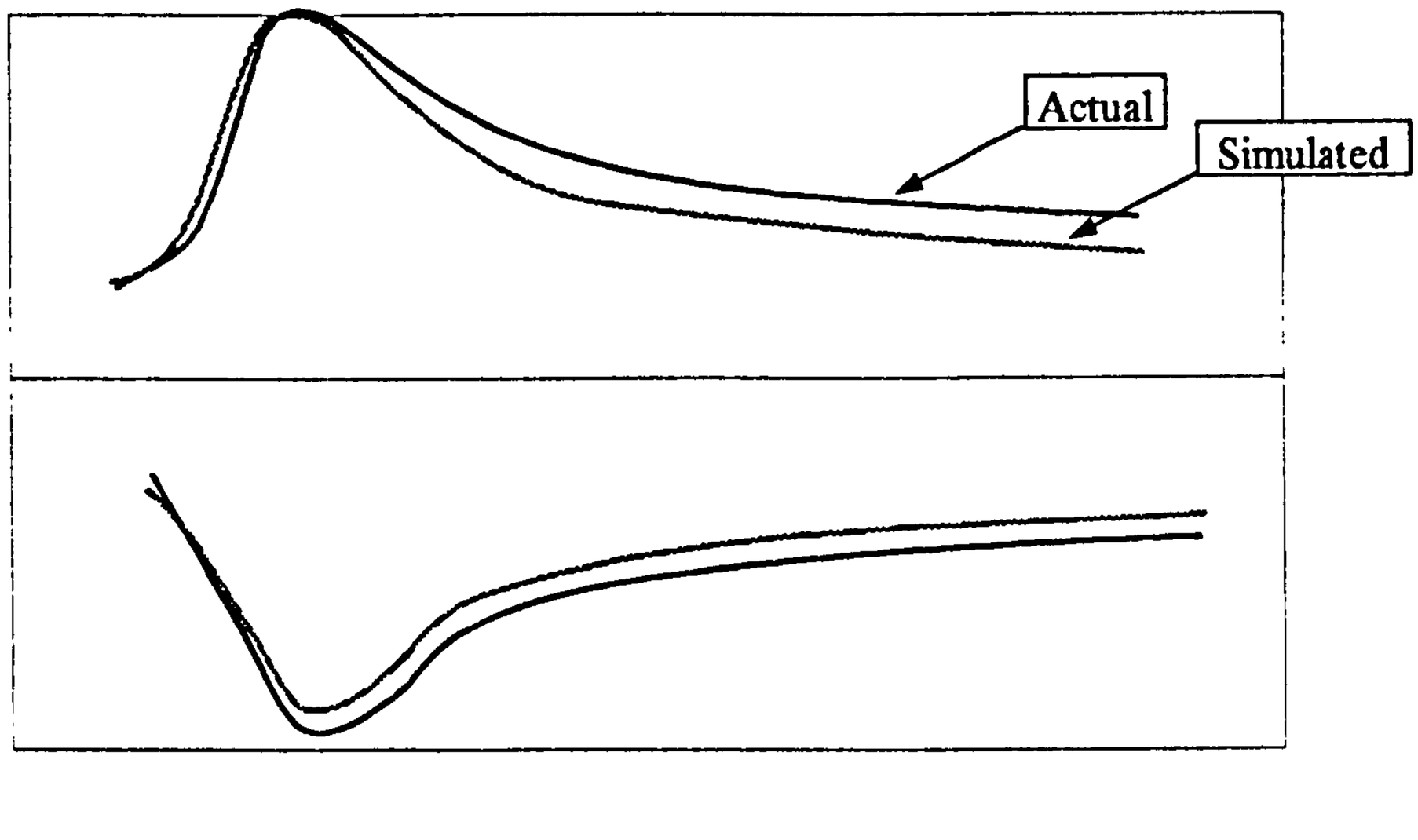


Figure 4.2.22: Actual and Simulated Current Envelope During Deceleration

The model used in the simulation is linear, no parameter variations are taken into account, the actual machine on the other hand, behaves in a non-linear manner due to effects such as saturation. The result of saturation of the main flux path in the machine is that the magnetising inductance varies with the level of saturation [*Vas*, 1990]. Therefore if the flux in the machine is not constant during the transient period then the magnetising inductance will not be constant during the transient, as is the case with the linear model. Under normal circumstances there is a linear relationship between the magnetising flux and the magnetising current as follows:-

$$\overline{\psi_{mag}} = L_{mag}\overline{i_{mag}} \qquad (4.2.4)$$

As the flux increases, however, and the machine exhibits a higher level of saturation the magnetising inductance varies in a non-linear fashion and so the relationship of (4.2.4) is no longer linear. Consider the magnitude of the magnetising flux vector during the deceleration transient:-

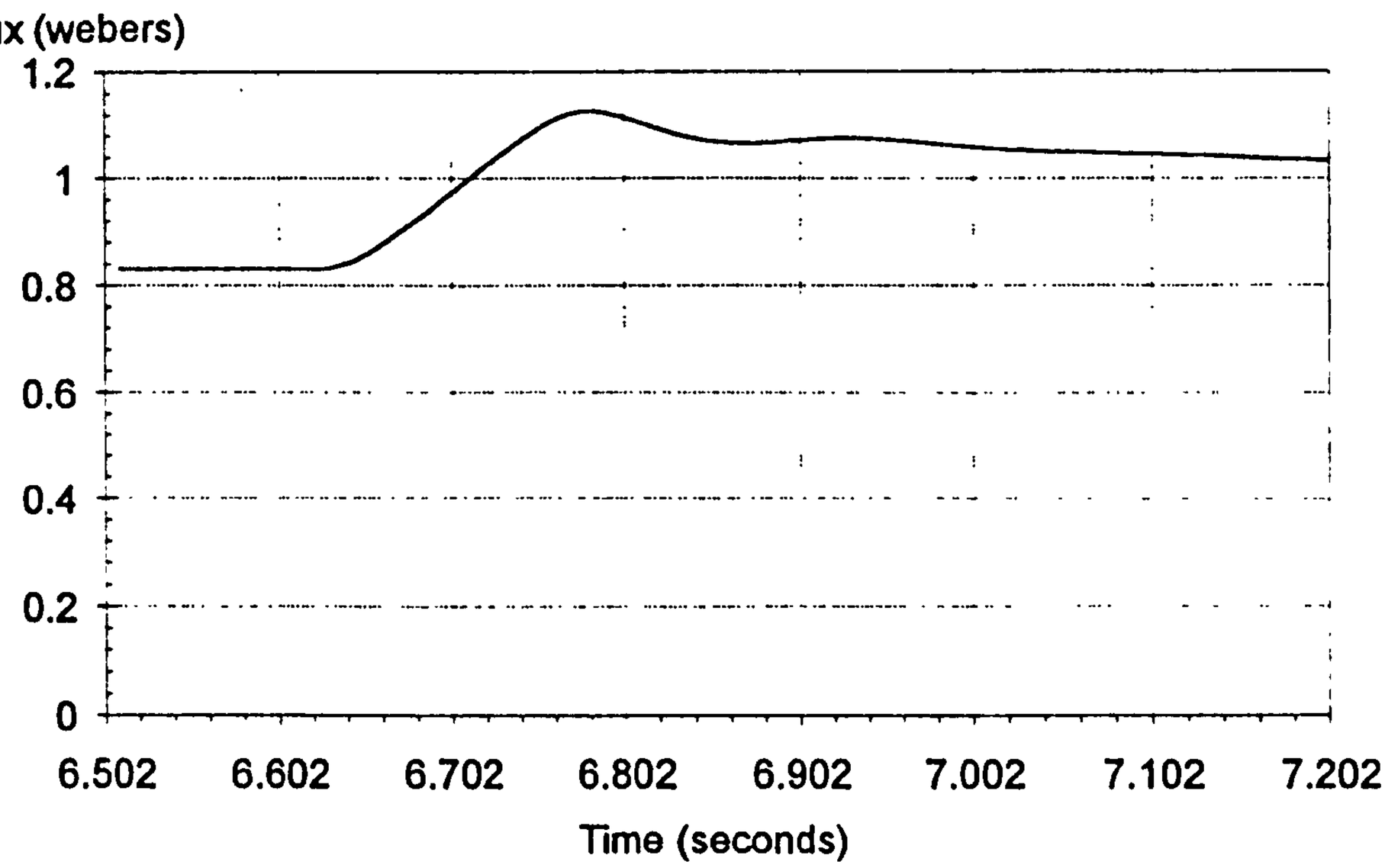


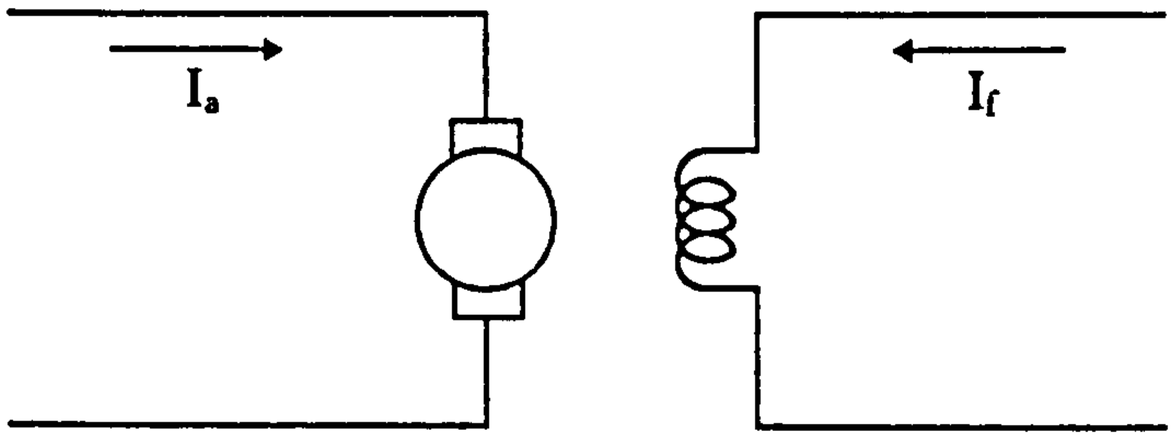
Figure 4.2.23: Magnetising Flux During Deceleration

The above figure clearly shows how the magnetising flux is far from constant but increases during the transient period. This increase in flux would cause the actual magnetising inductance to reduce and thus differ from the constant value used within the simulation.

4.3 Vector Control

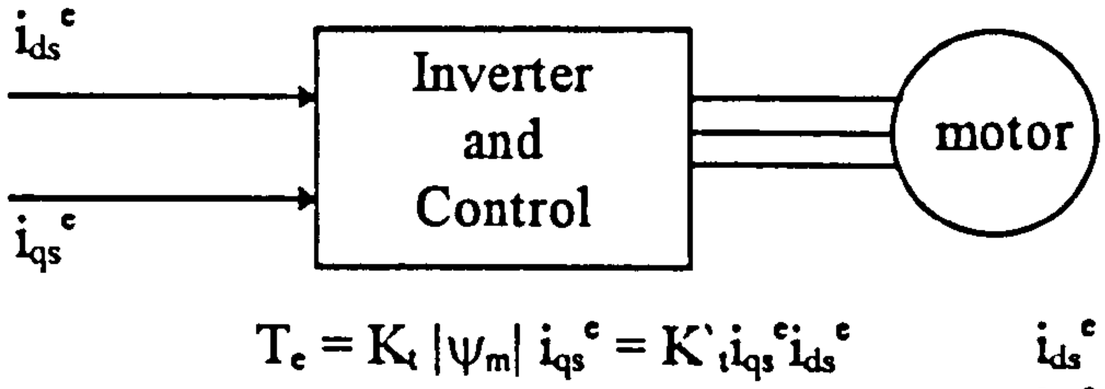
4.3.1 Theory

The vector controller developed for this drive system is a direct field orientated controller, a direct field orientated controller uses measurable values to produce the angle and magnitude of the rotor flux vector. The information regarding the rotor flux vector is required by the control algorithms to achieve independent control of the machine's torque and flux. In Vector Control the torque and flux of the machine are controlled by separate decoupled current components. This is an analogy to the manner in which a separately excited DC machine is controlled, in this case the armature current controls the torque and the field current controls the flux. Consider the figure below [*Bose*, 1986].



$T_e = K_t \psi_g I_a = K'_t I_a I_f$

I_f - flux component
I_a - torque component



$T_e = K_t |\psi_m| i_{qs}^e = K'_t i_{qs}^e i_{ds}^e$

i_{ds}^e - flux component
i_{qs}^e - torque component

Figure 4.3.1: Analogy Between DC Machine and Induction Machine

If the two current components i_{ds}^e and i_{qs}^e can be controlled individually then i_{ds}^e can be used to achieve constant rated flux, and i_{qs}^e can be used to achieve maximum torque response.

Consider the vector diagram below, it shows the stationary D-Q stator axis and the d-q rotor axis which is rotating at $\omega_r = d\theta_r/dt$ with respect to the stator axis. The rotor current vector is also shown, the currents induced in the rotor rotate at a frequency $\omega_2 = d\theta_2/dt$ with respect to the rotor axis.

$$\omega_2 = \omega_1 - \omega_r \qquad (4.3.1)$$

where:-

ω_1 = synchronous angular velocity

ω_r = rotor angular velocity

ω_2 = slip angular velocity

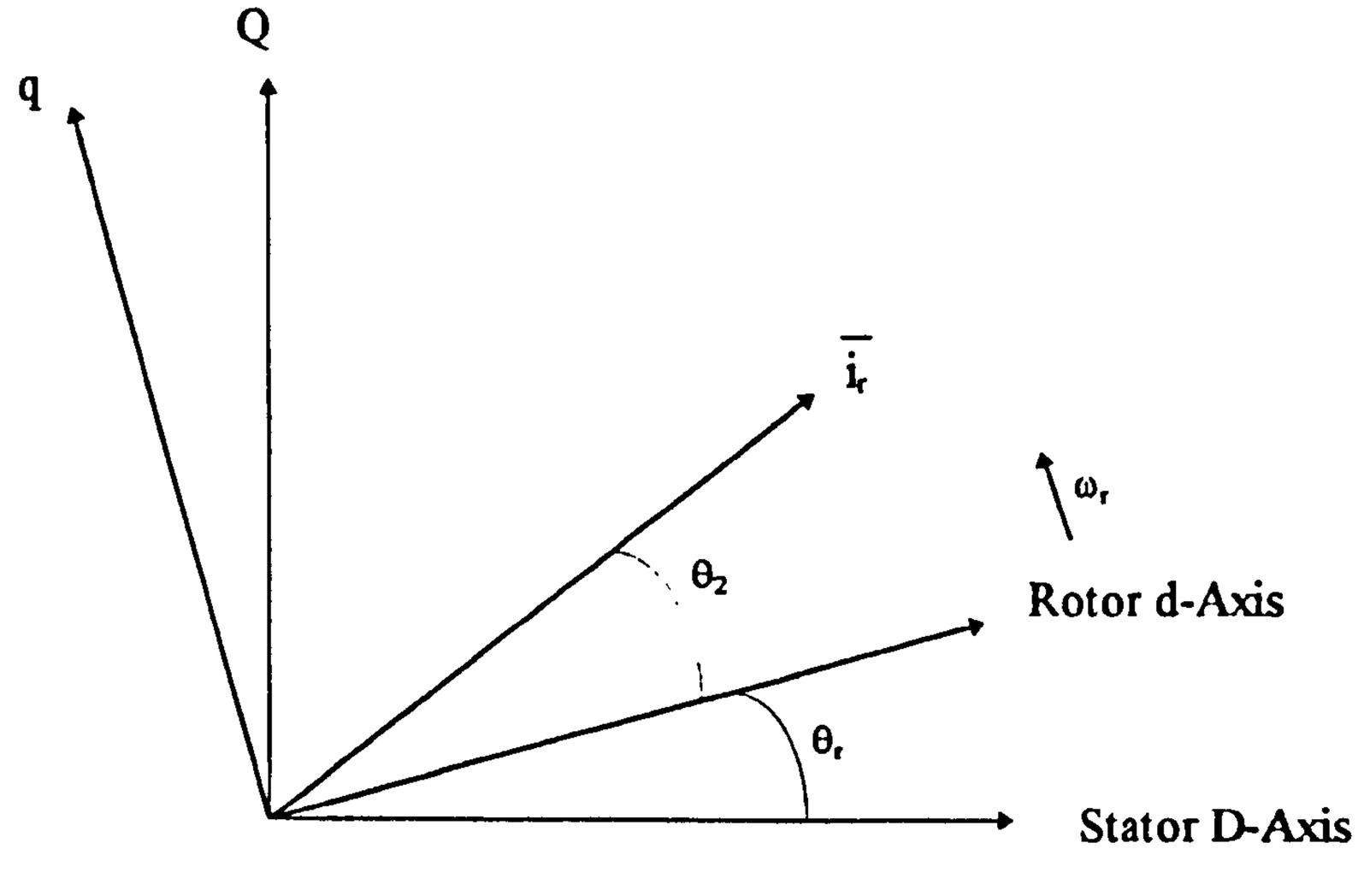


Figure 4.3.2: Rotor Current Vector

The rotor current vector can be expressed in terms of the rotor axis:-

$$\overline{i_r} = i_{dr} + ji_{qr} = i_r e^{j\omega_2 t} \qquad (4.3.2)$$

and expressed in terms of the stator axis:-

$$\overline{i_r} = i_r e^{j(\omega_2+\omega_r)t} \qquad (4.3.3)$$

$$\overline{i_r} = i_r e^{j\omega_1 t} \qquad (4.3.4)$$

therefore it can be seen that the rotor current vector rotates at the stator frequency with respect to the stator axis.

The following vector diagram shows a reference frame attached to the rotor flux vector which is in line with the rotor current vector. This reference frame therefore rotates at synchronous speed with respect to the stator axis.

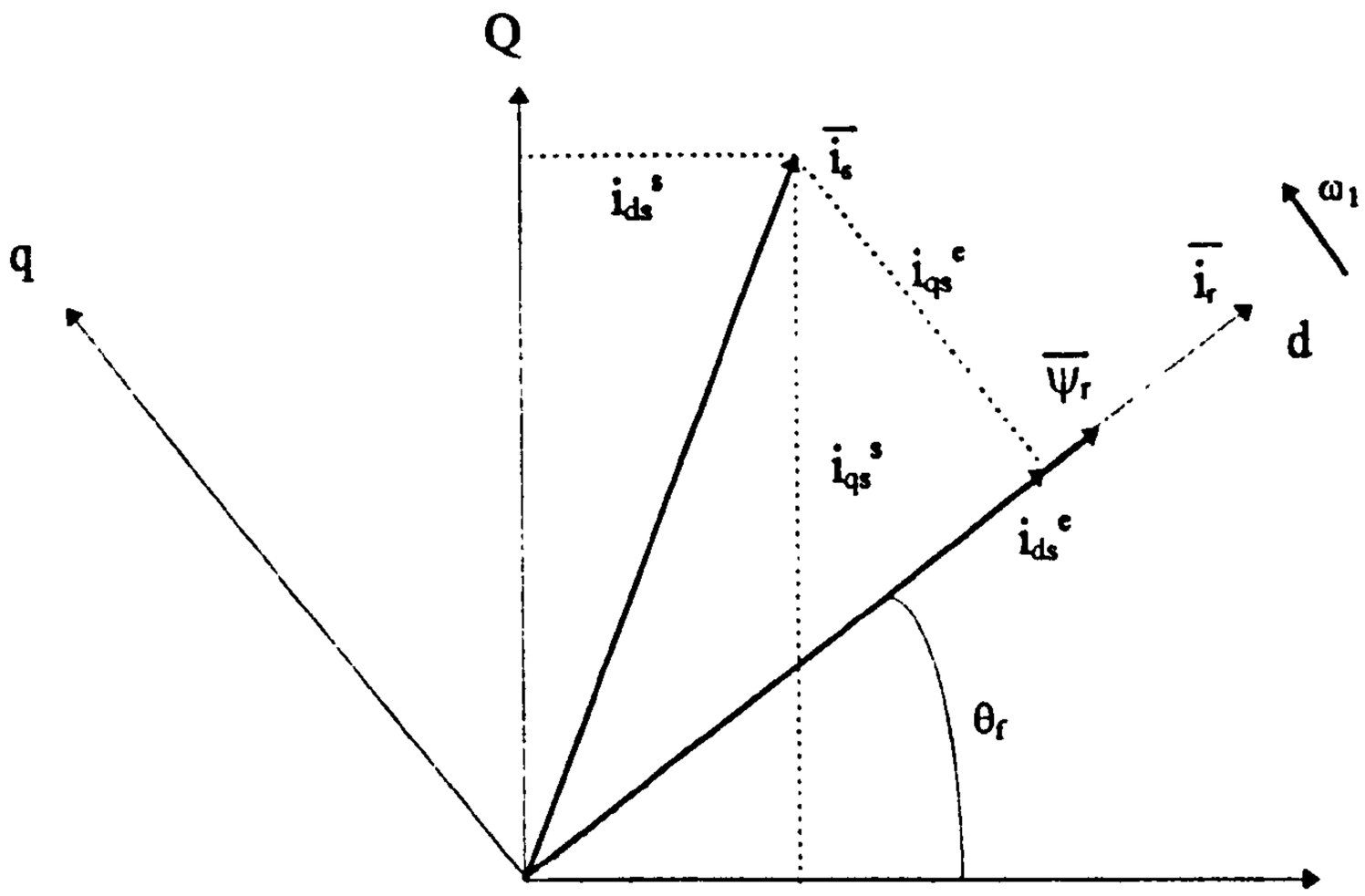
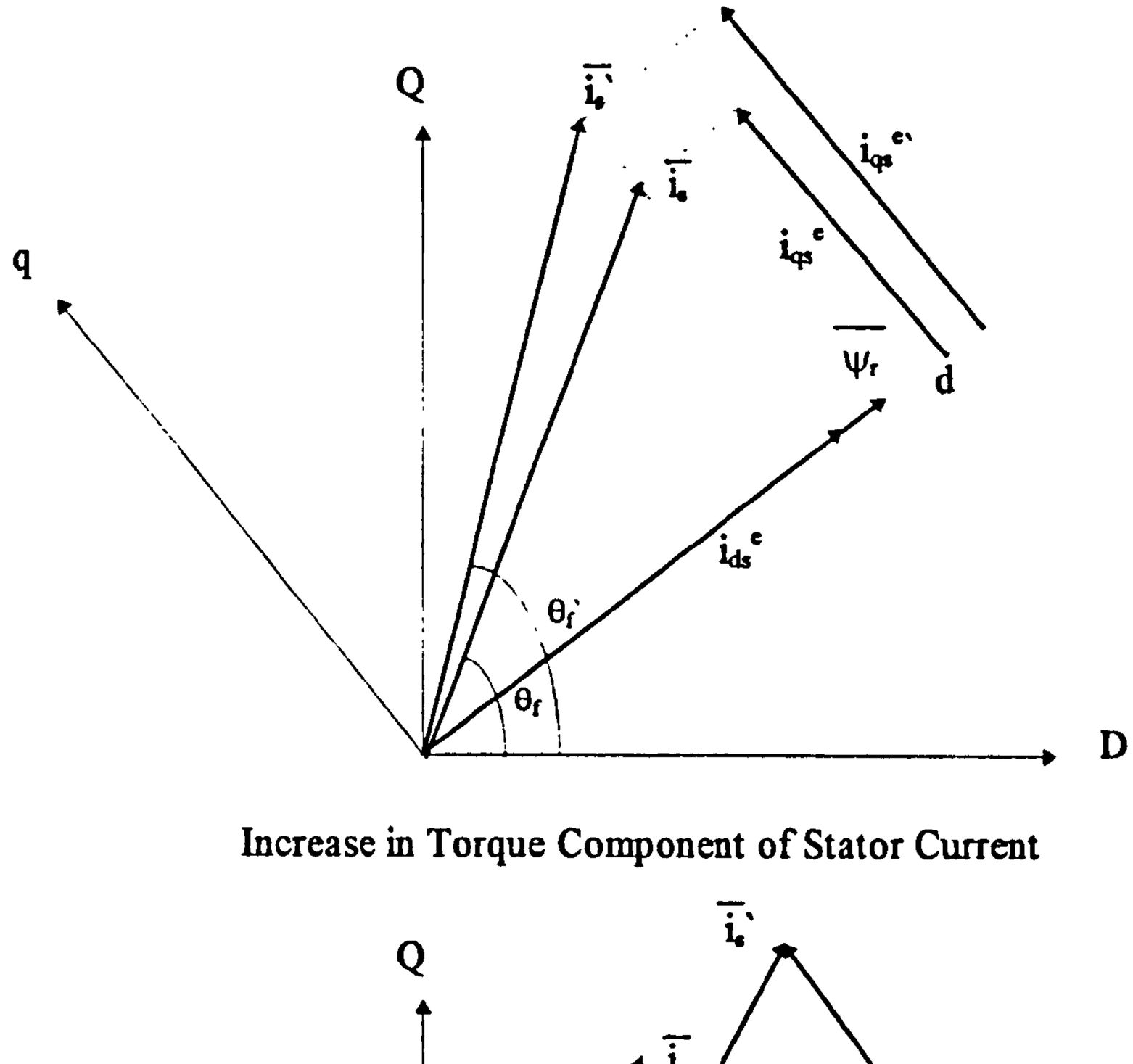


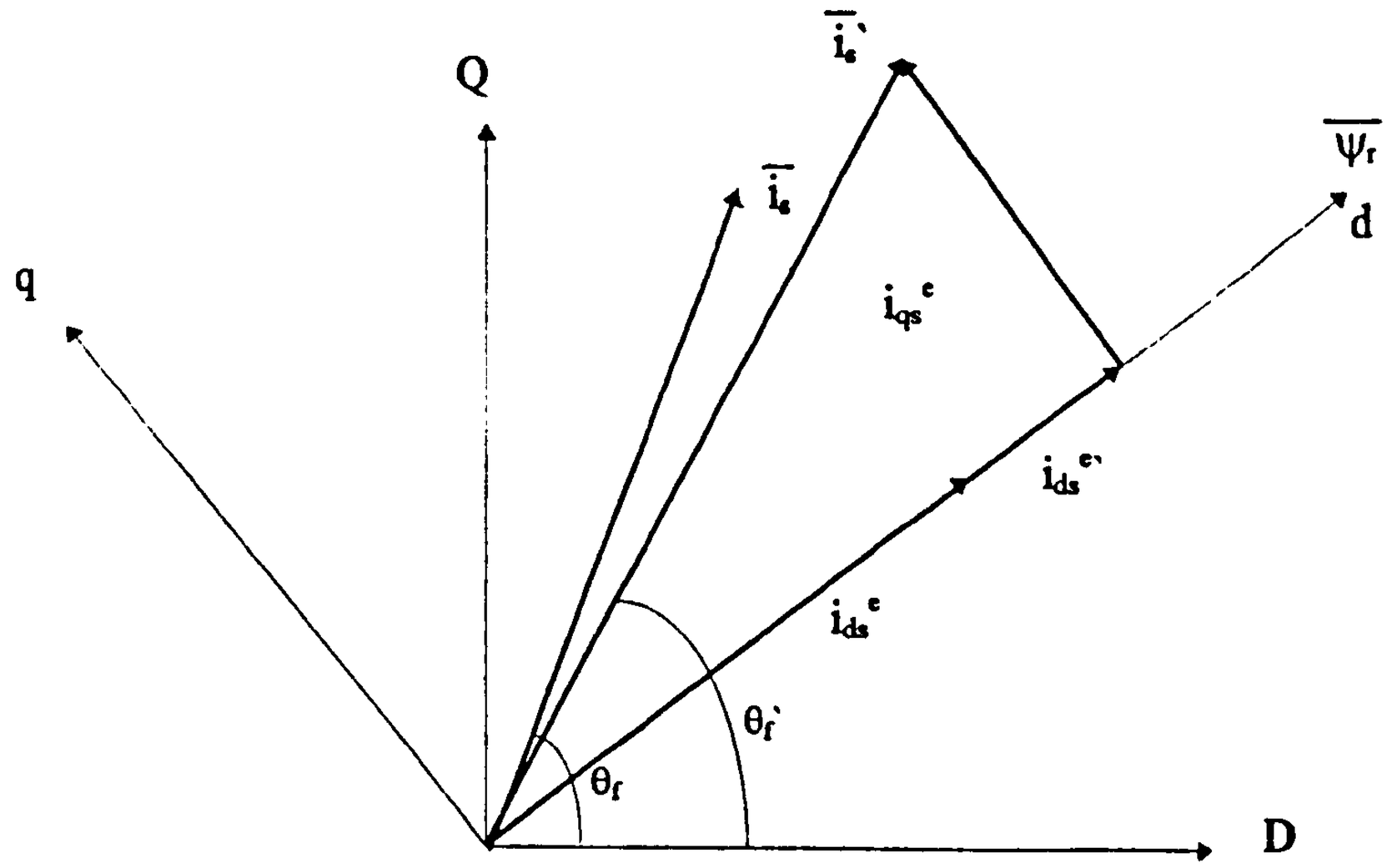
Figure 4.3.3: Stator Current Vector in the Rotating Reference Frame

Unlike the DC machine which has direct access to the armature and field windings, the cage induction motor only provides access to the stator current. The stator current must therefore be broken down into parts which directly affect the torque and flux. The vector diagram above shows the stator current vector and the D and Q axis components i_{ds}^s and i_{qs}^s which make up this vector. It can be seen from the above diagram that the stator current vector can also be represented by two currents in the rotating reference frame attached to the rotor flux vector, i_{ds}^e and i_{qs}^e.

The current component i_{ds}^e of the stator current is in line with the flux vector and therefore directly influences it, the current component i_{qs}^e of the stator current is perpendicular to the flux vector and therefore influences the torque of the machine. An indication of how the stator current vector can be controlled so that the components vary according to the torque and flux demands is given in the diagrams below.



Increase in Torque Component of Stator Current



Increase in Flux Component of Stator Current

Figure 4.3.4: Torque and Flux Components of Stator Current Vector

The first figure shows that by controlling the stator current vector from i_s to i_s' by increasing the torque demand component from i_{qs}^e to $i_{qs}^{e'}$, the torque is increased but the flux component i_{ds}^e can be maintained at a constant value thus maintaining rated flux. Likewise, the second figure shows how the stator current vector can be controlled from i_s to i_s' in such a manner as to keep the torque component i_{qs}^e constant and varying the flux component for a variation in the flux demand from i_{ds}^e to $i_{ds}^{e'}$.

In order to be able to control the currents i_{ds}^e and i_{qs}^e they must be obtained from the available stator current which is in the stationary stator reference frame. A transformation is therefore required between the stator reference frame and the rotor flux reference frame. From Fig. 4.3.3 it can be seen that the transformation of the stator reference frame currents i_{ds}^s and i_{qs}^s which make up the stator current vector, to the rotor flux reference frame currents i_{ds}^e and i_{qs}^e is made by the angle between the two reference frames θ_f. The transformation from the stator to the rotor flux reference frame is as follows:-

$$\begin{vmatrix} i_{ds}^e \\ i_{qs}^e \end{vmatrix} = \begin{vmatrix} \cos\theta_f & \sin\theta_f \\ -\sin\theta_f & \cos\theta_f \end{vmatrix} \begin{vmatrix} i_{ds}^s \\ i_{qs}^s \end{vmatrix} \qquad (4.3.5)$$

The transformation from the rotor flux reference frame to the stator reference frame is:-

$$\begin{vmatrix} i_{ds}^s \\ i_{qs}^s \end{vmatrix} = \begin{vmatrix} \cos\theta_f & -\sin\theta_f \\ \sin\theta_f & \cos\theta_f \end{vmatrix} \begin{vmatrix} i_{ds}^e \\ i_{qs}^e \end{vmatrix} \qquad (4.3.6)$$

The above transformations can be used to change the available stator current components into the rotor flux reference frame in which they can be controlled. The transformed stator currents i_{ds}^s and i_{qs}^s will appear as DC quantities i_{ds}^e and i_{qs}^e in the rotor flux reference frame since they are rotating at synchronous speed in the stator reference frame and are transformed into a reference frame which is also rotating at synchronous speed.

The angle θ_f of the rotor flux vector with respect to the stator must be calculated or measured in order to perform the transformations. As previously mentioned the Vector Controller which is developed here is a direct field orientated controller, in a direct field orientated controller the information about the rotor flux vector is either measured directly using sensors or calculated using real measurable quantities. In this controller the angle θ_f is calculated by measured real quantities, in this case real values of stator current i_{ds}^s and i_{qs}^s and rotor angular velocity ω_r are used. The controller uses a flux model to calculate the magnitude and position of the rotor flux vector using the measurable quantities and known machine parameters.

4.3.2 Flux Model

The object of the flux model is to use real measurable quantities and estimate the magnitude and angle θ_f of the rotor flux vector. Consider the rotor voltage equations in the stator D-Q axis reference frame:-

$$V_{dr} = R_r i_{dr} + \frac{d\psi_{dr}}{dt} + \omega_r \psi_{qr} \qquad (4.3.7)$$

$$V_{qr} = R_r i_{qr} + \frac{d\psi_{qr}}{dt} - \omega_r \psi_{dr} \qquad (4.3.8)$$

$$V_{dr} = V_{qr} = 0 \qquad (4.3.9)$$

substituting (4.3.9) into (4.3.7) and (4.3.8) gives:-

$$\frac{d\psi_{dr}}{dt} = -R_r i_{dr} - \omega_r \psi_{qr} \qquad (4.3.10)$$

$$\frac{d\psi_{qr}}{dt} = -R_r i_{qr} + \omega_r \psi_{dr} \qquad (4.3.11)$$

and:-

$$\psi_{dr} = L_r i_{dr} + L_m i_{ds} \qquad (4.3.12)$$

$$\psi_{qr} = L_r i_{qr} + L_m i_{qs} \qquad (4.3.13)$$

rearranging (4.3.12) and (4.3.13) gives:-

$$i_{dr} = \frac{\psi_{dr} - L_m i_{ds}}{L_r} \qquad (4.3.14)$$

$$i_{qr} = \frac{\psi_{qr} - L_m i_{qs}}{L_r} \qquad (4.3.15)$$

substituting (4.3.14) and (4.3.15) into (4.3.10) and (4.3.11) gives:-

$$\frac{d\psi_{dr}}{dt} = -R_r \frac{\psi_{dr} - L_m i_{ds}}{L_r} - \omega_r \psi_{qr} \qquad (4.3.16)$$

$$\frac{d\psi_{qr}}{dt} = -R_r \frac{\psi_{qr} - L_m i_{qs}}{L_r} + \omega_r \psi_{dr} \qquad (4.3.17)$$

Equations (4.3.16) and (4.3.17) can be represented in a block diagram of the flux model as shown below:-

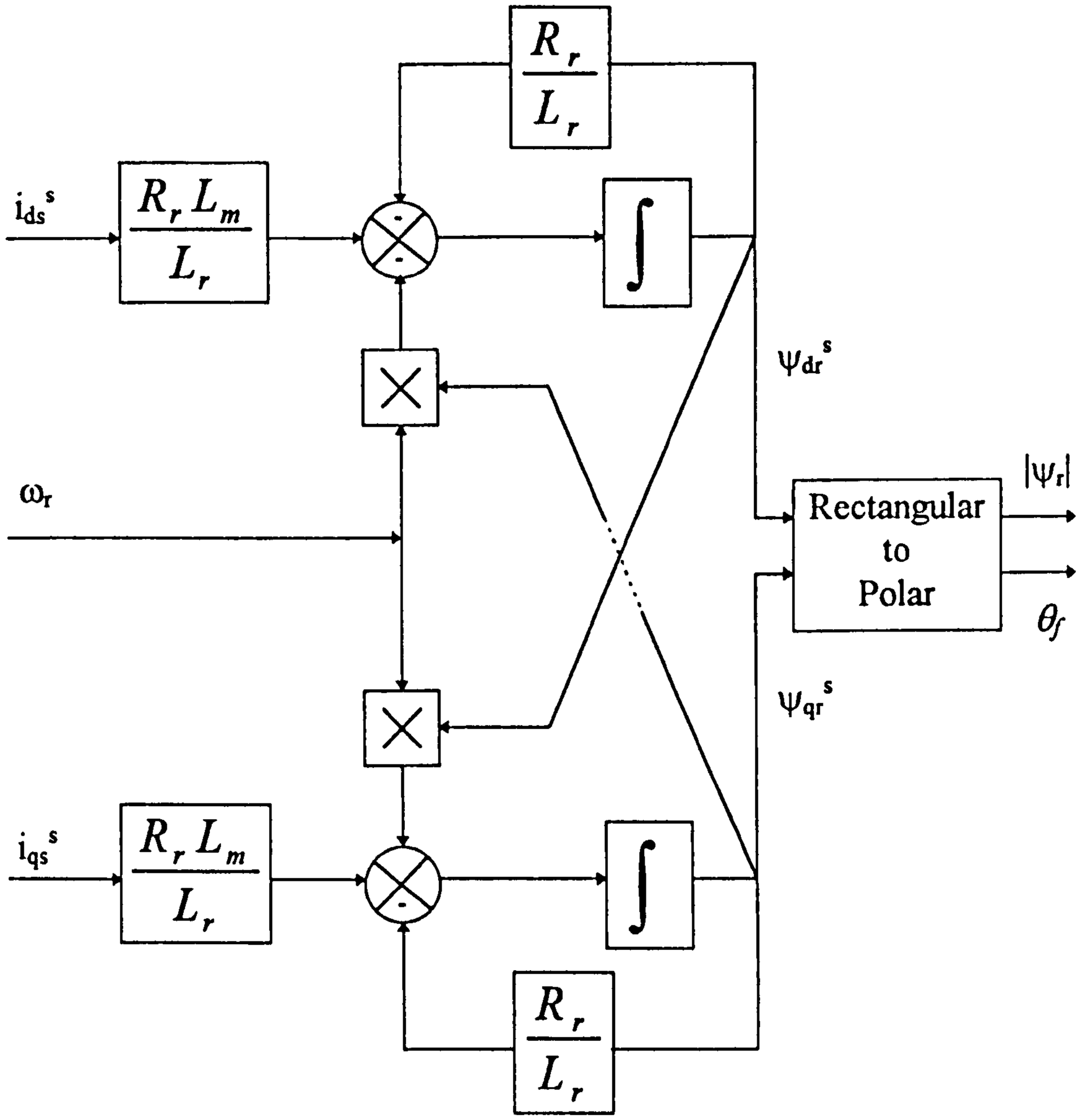


Figure 4.3.5: Flux Model

The following results were obtained from the flux model using the D and Q axis stator currents of the motor model which was fed from a three phase sinusoidal 50Hz supply, the motor was rotating at 157 rads/second. The results show the estimated D and Q axis rotor flux components, the resultant rotor flux magnitude and angle, and also the D and Q axis stator currents transformed into the rotor flux reference frame.

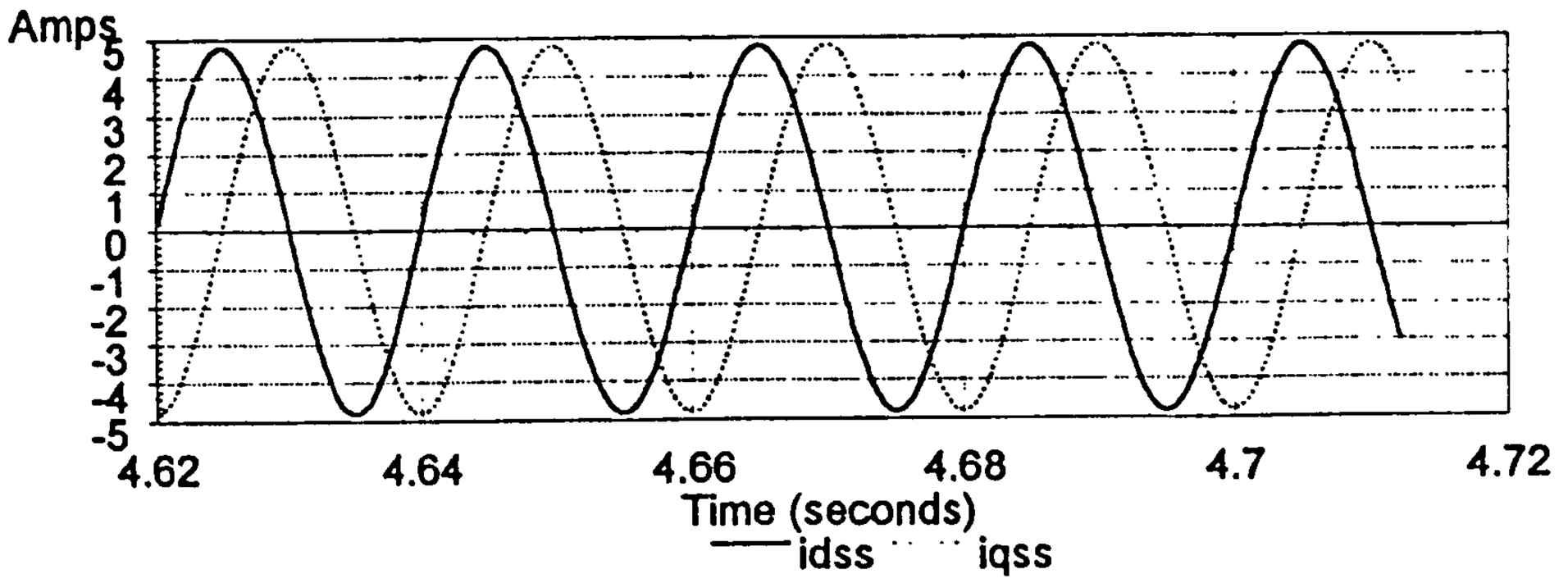


Figure 4.3.6 D-Q Stator Currents in Stator Reference Frame

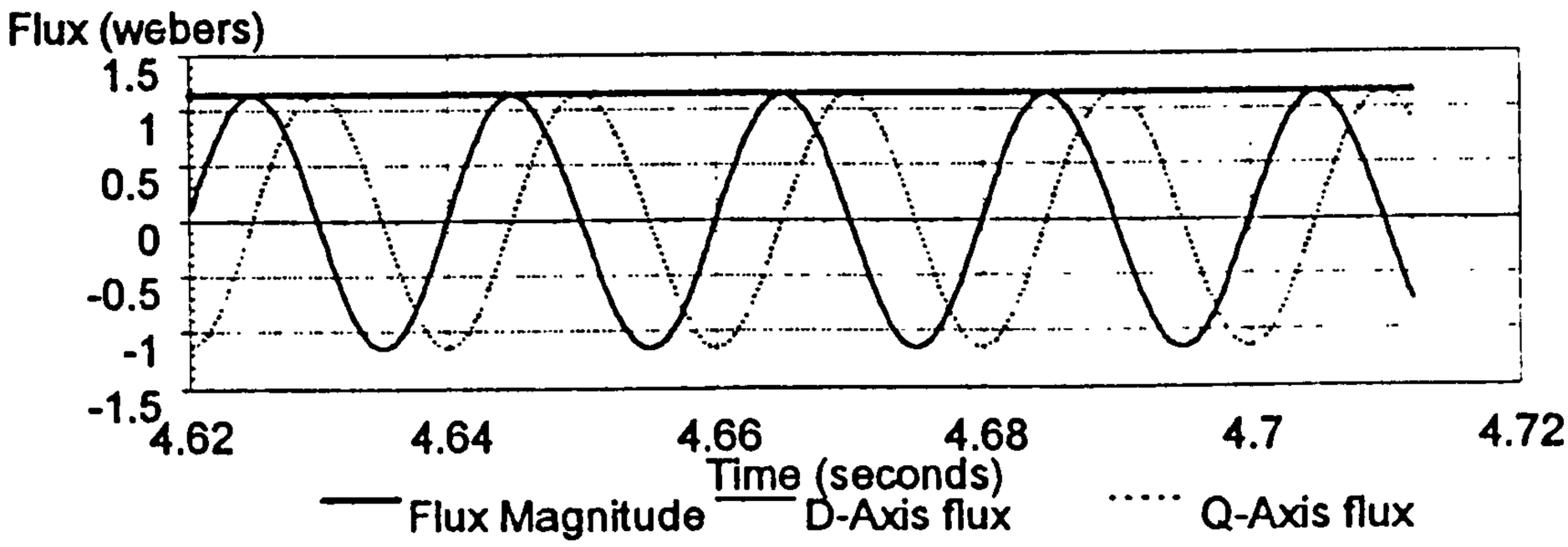


Figure 4.3.7 D-Q Rotor Flux Components and Resultant Flux Magnitude

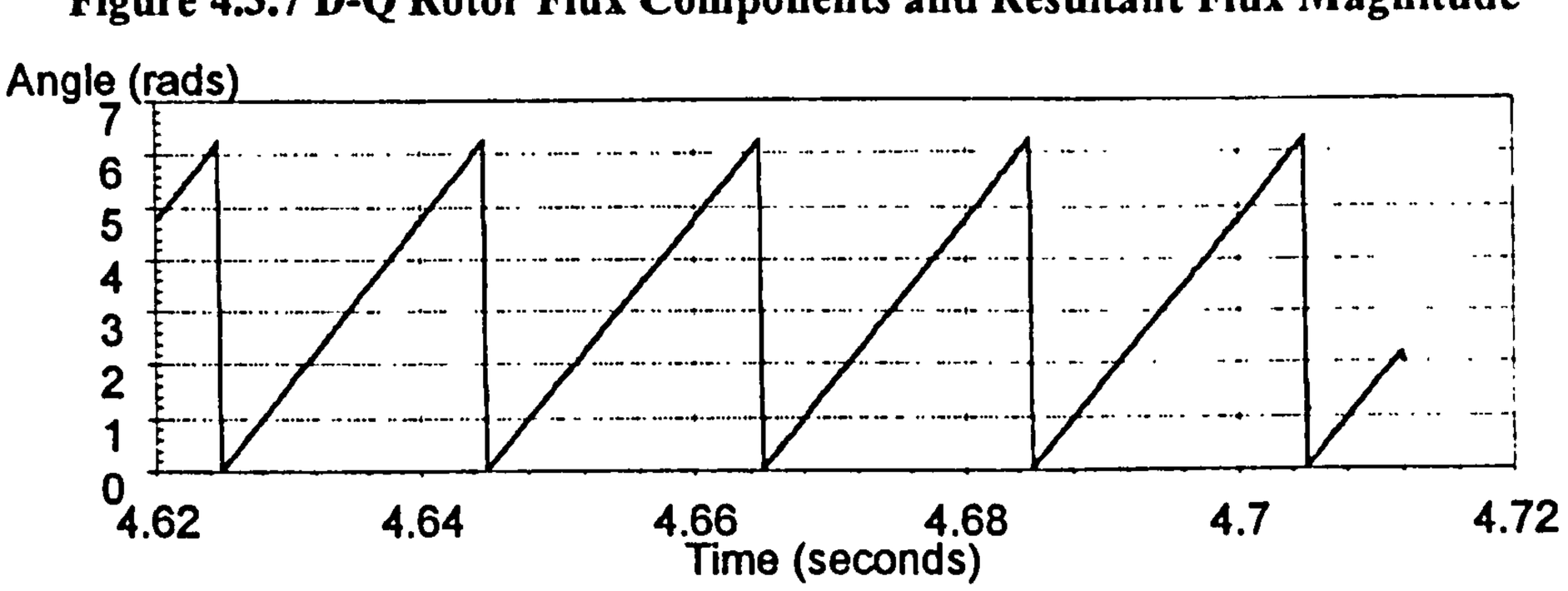


Figure 4.3.8 Angle of Rotor Flux Vector

Figure 4.3.9 D-Q Stator Currents Transformed to Rotor Flux Reference Frame

It can be seen from Figure 4.3.9 that the sinusoidal stator currents of Figure 4.3.6 when transformed by the estimated rotor flux angle of Figure 4.3.8 give DC quantities, these can then be controlled by a closed loop current controller. This flux model is common and is described along with other flux modelling methods in [*Vas*, 1990].

4.3.3 The Control System

The flux model and the reference transformations previously discussed are part of the overall control system. A block diagram of the control system which is made up of these and other parts is shown below.

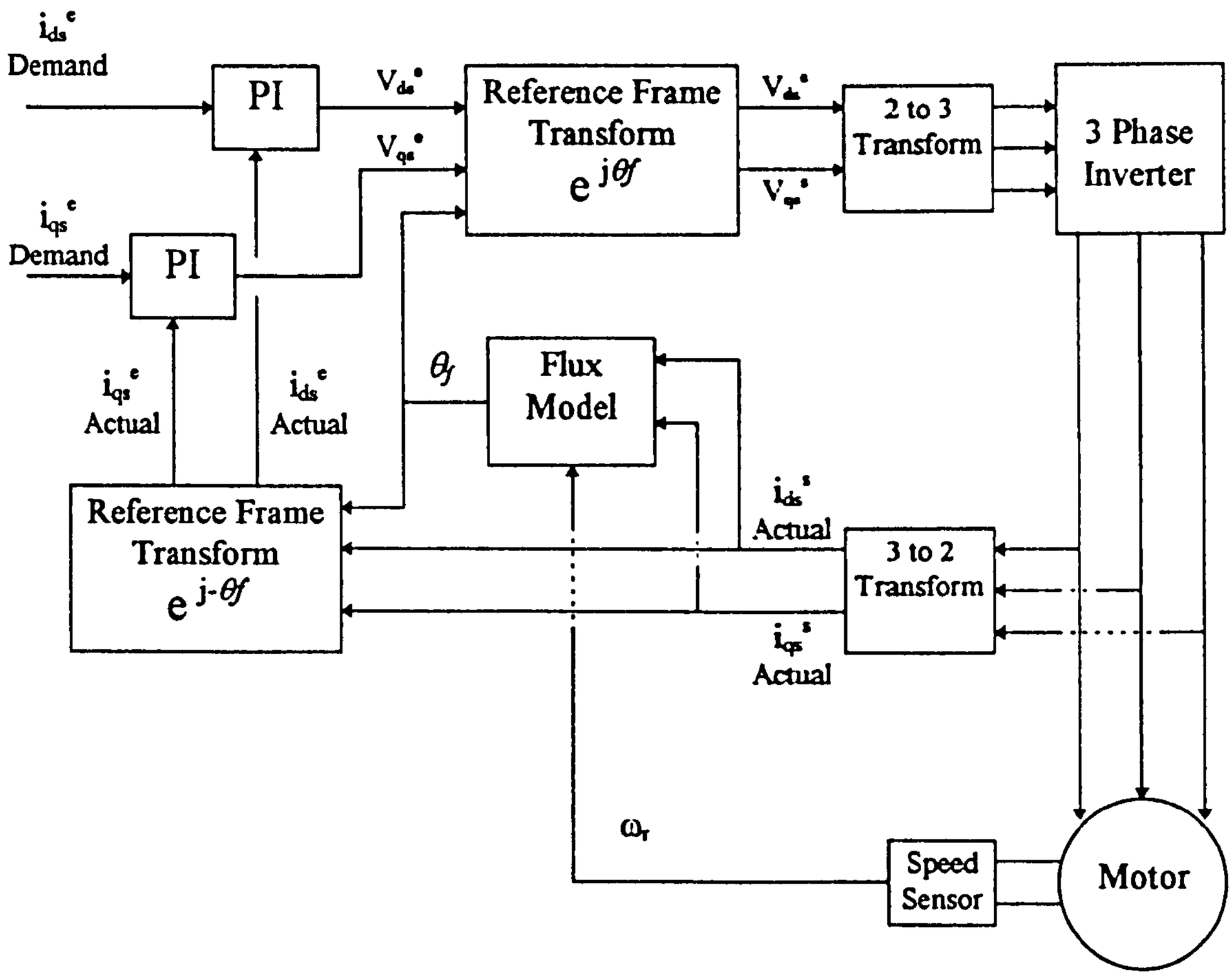


Figure 4.3.10: Vector Control Block Diagram

The demanded values of i_{ds}^e and i_{qs}^e are supplied by the operator, the flux in the machine is governed by i_{ds}^e and torque by i_{qs}^e. The PI controllers then produce the demanded values of V_{ds}^e and V_{qs}^e from the demanded and actual currents. These two voltages are transformed to the stator reference frame and transformed to the three phase voltage demands that the voltage source inverter requires. The actual motor three phase currents are measured and transformed to the D-Q axis currents i_{ds}^s and i_{qs}^s, these stator reference frame currents are then transformed to the rotor flux reference frame and fed back to the PI current controllers.

The following results show how the flux and torque of the machine are changed by varying the two components of the stator current in the rotor flux reference frame i_{ds}^e and i_{qs}^e. The first case examined is that of an increase in flux component i_{ds}^e from 1 to 5 amps. Figure 4.3.11 shows the actual value of i_{ds}^e.

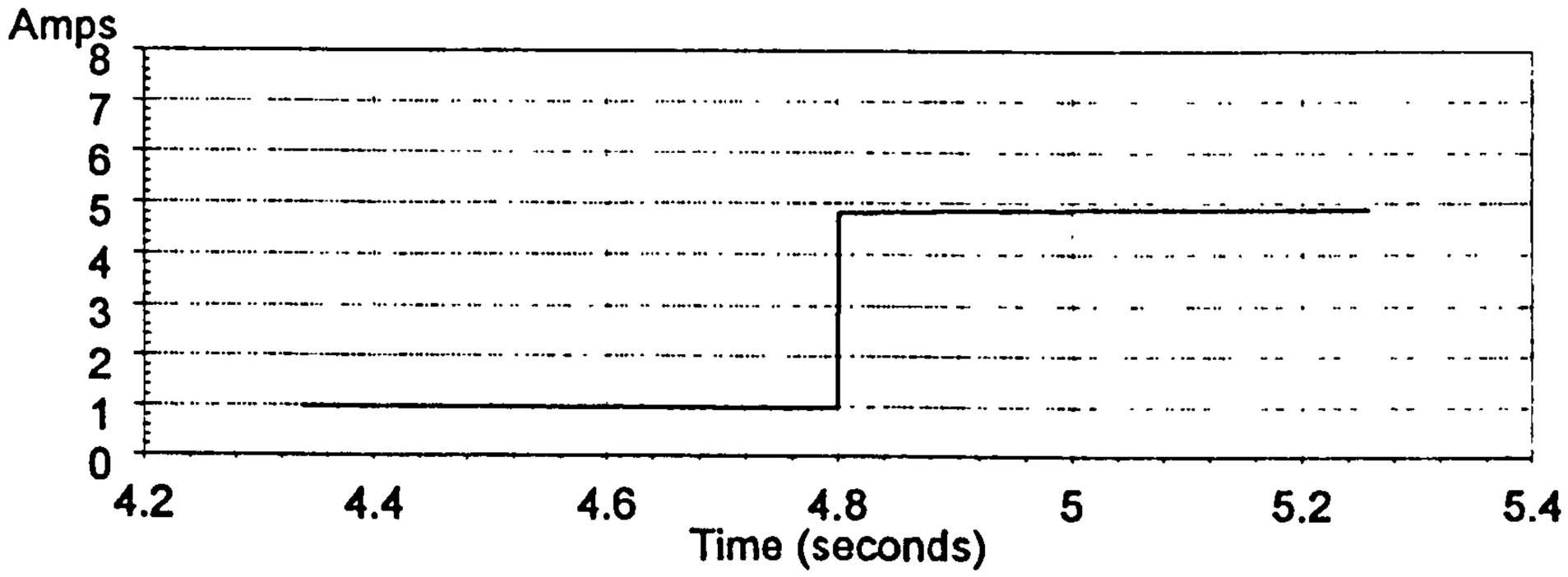


Figure 4.3.11: D-Axis Stator Current in Rotor Flux Reference Frame i_{ds}^e

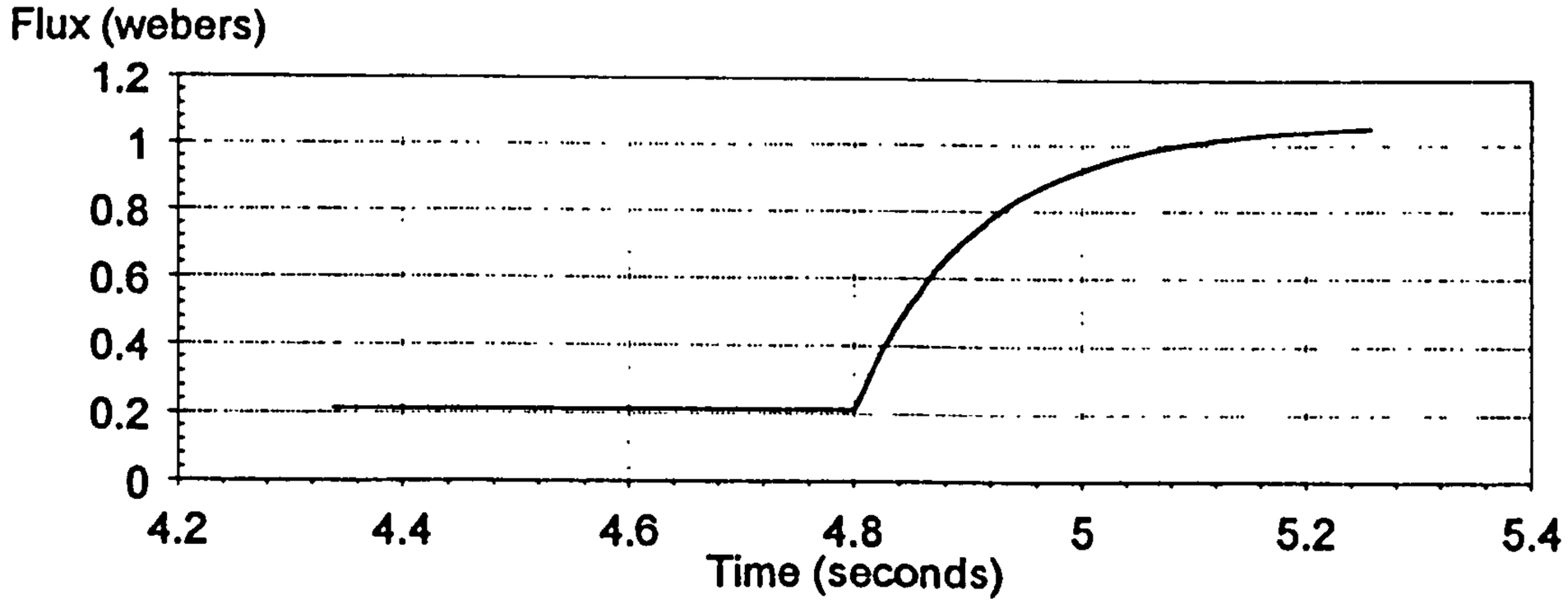


Figure 4.3.12: Magnitude of Rotor Flux Vector

The previous figure displayed the magnitude of the rotor flux vector. It is also possible to display the rotor flux vector in the form of a locus as shown below.

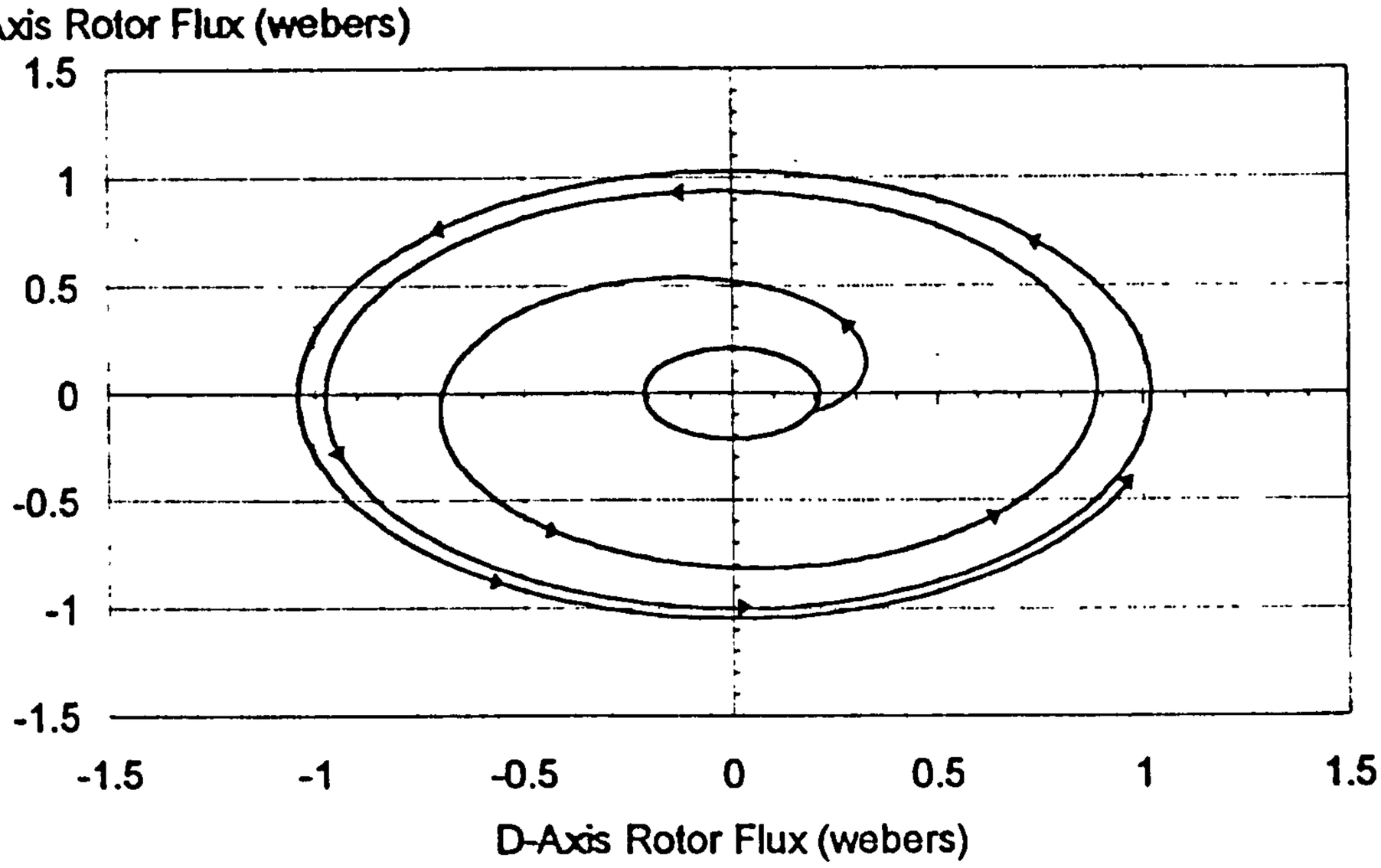


Figure 4.3.13: Locus of Rotor Flux Vector

The effect of increasing the current demand i_{ds}^e is to increase the magnitude of the rotor flux vector, This is as expected since the D axis stator current, in the rotor flux reference frame, is in line with the rotor flux vector.

The following results show the effect of increasing the torque component of stator current i_{qs}^e. The demand current i_{qs}^e was increased from 0 to 5A , the flux component i_{ds}^e was held at 3A. The first figure shows the increase in the actual value of i_{qs}^e, the second figure shows the electromagnetic torque produced by the machine and the final figure shows the rotor angular velocity of the machine. The figures clearly show the rapid increase in the machine torque, directly controlled by the current component i_{qs}^e.

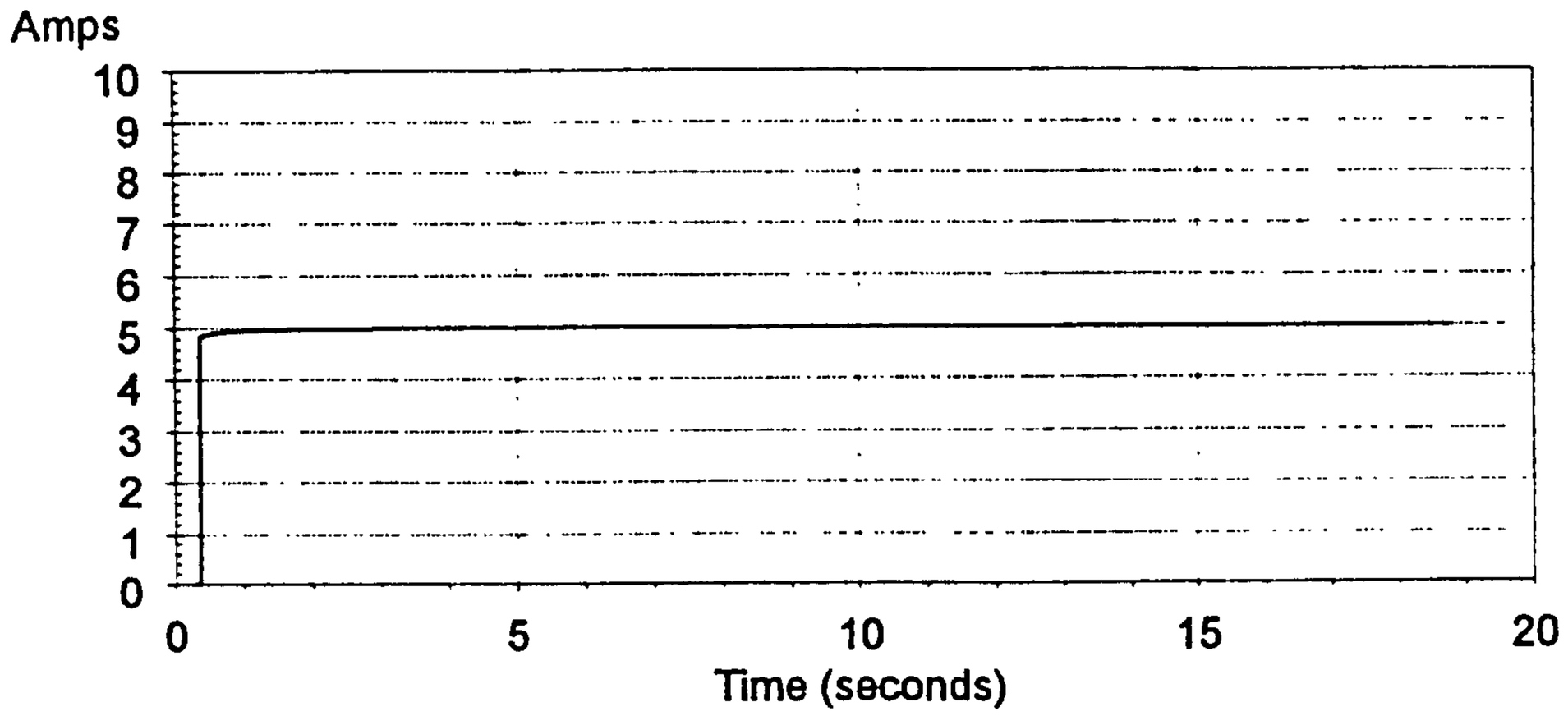


Figure 4.3.14: Q-Axis Stator Current in Rotor Flux Reference Frame i_{qs}^{e}

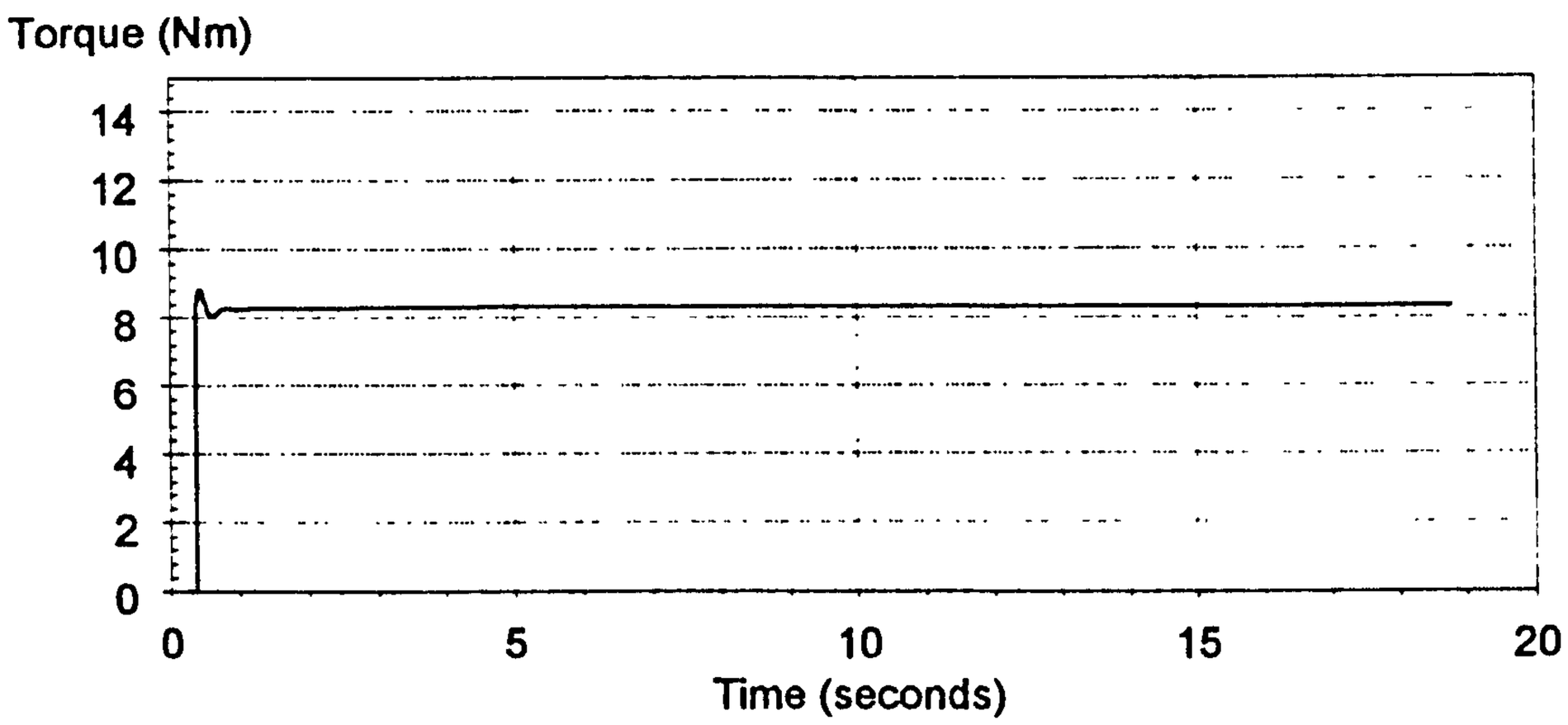


Figure 4.3.15: Electromagnetic Torque

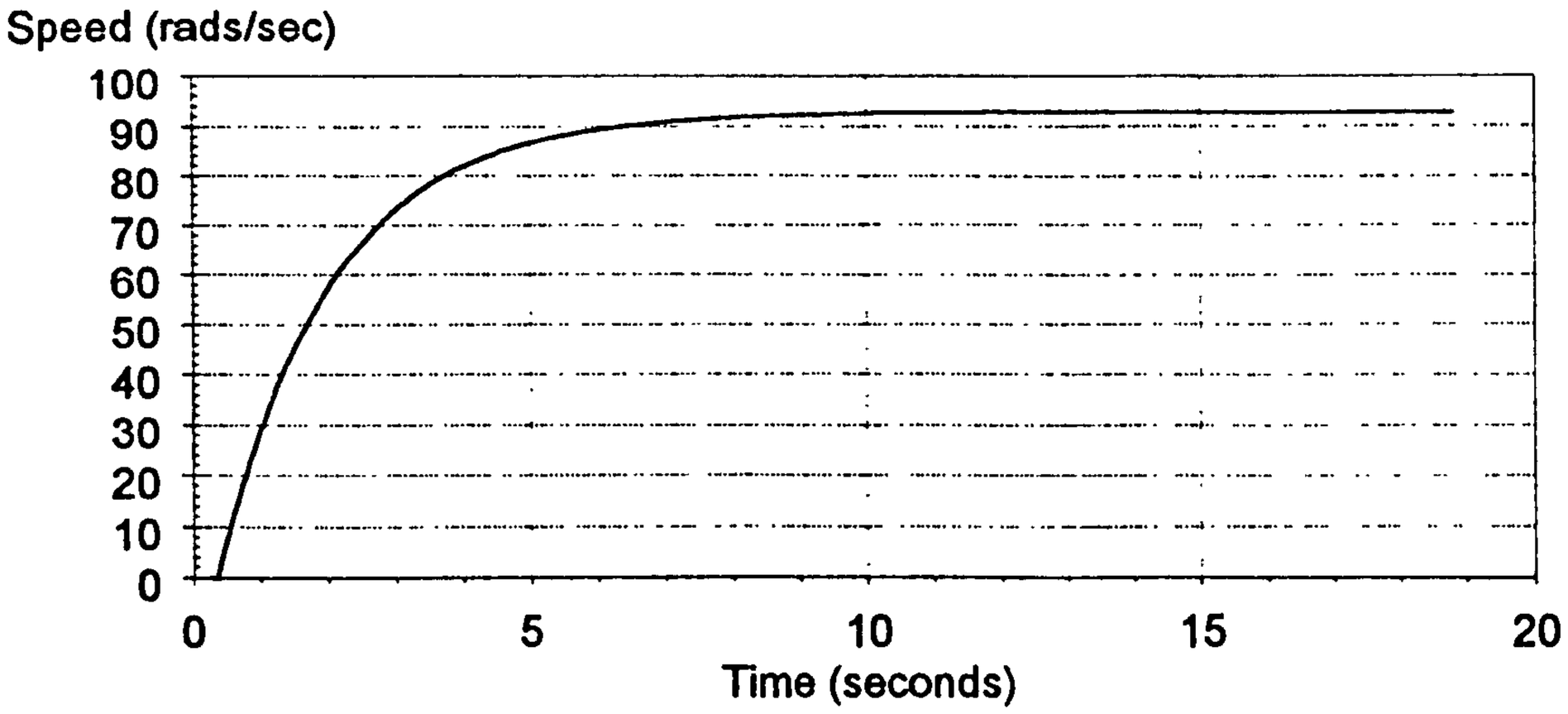


Figure 4.3.16: Rotor Angular Velocity

The controller is now included in a complete simulation of the drive system including the machine model and the inverter model. The following results compare the simulation of the drive system including the Vector controller described above, to the actual drive system. The results show the steady state case for different values of i_{qs}^{e} the torque component, with a constant flux producing current component i_{ds}^{e} .

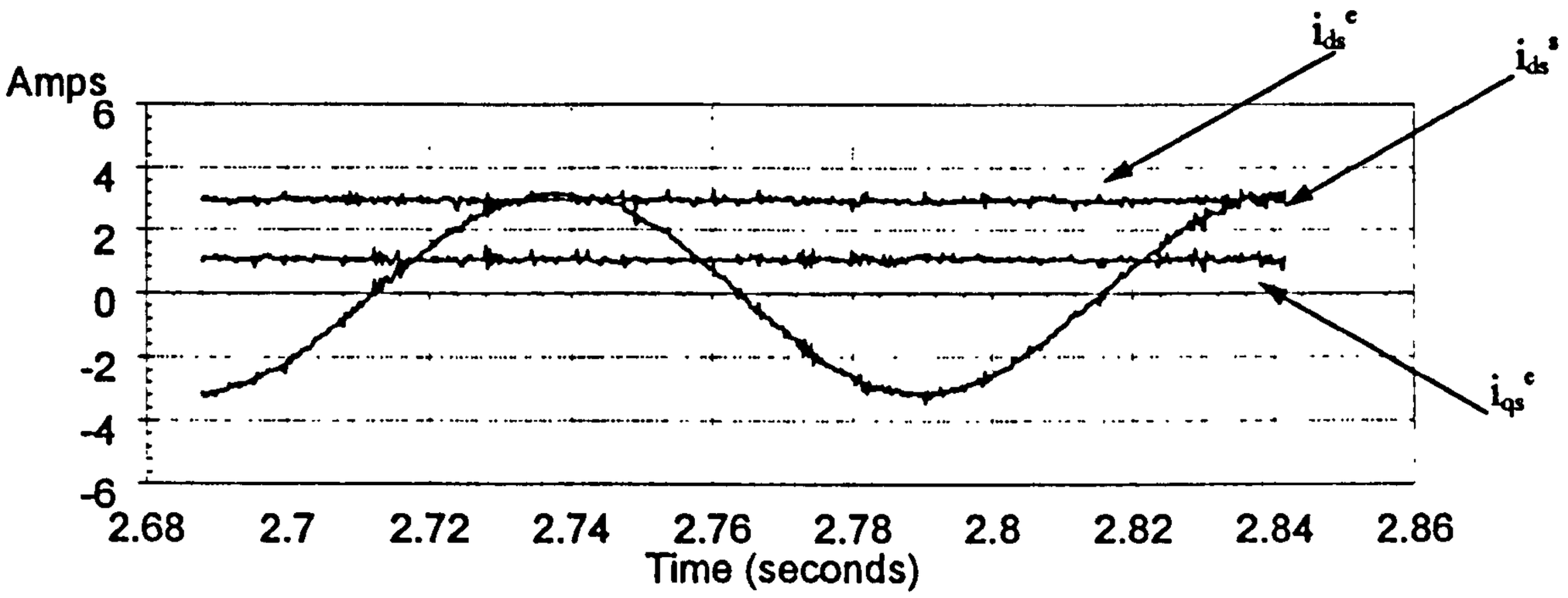


Figure 4.3.17: Actual Currents $i_{d_s}^{e}$=3A $i_{q_s}^{e}$=1A

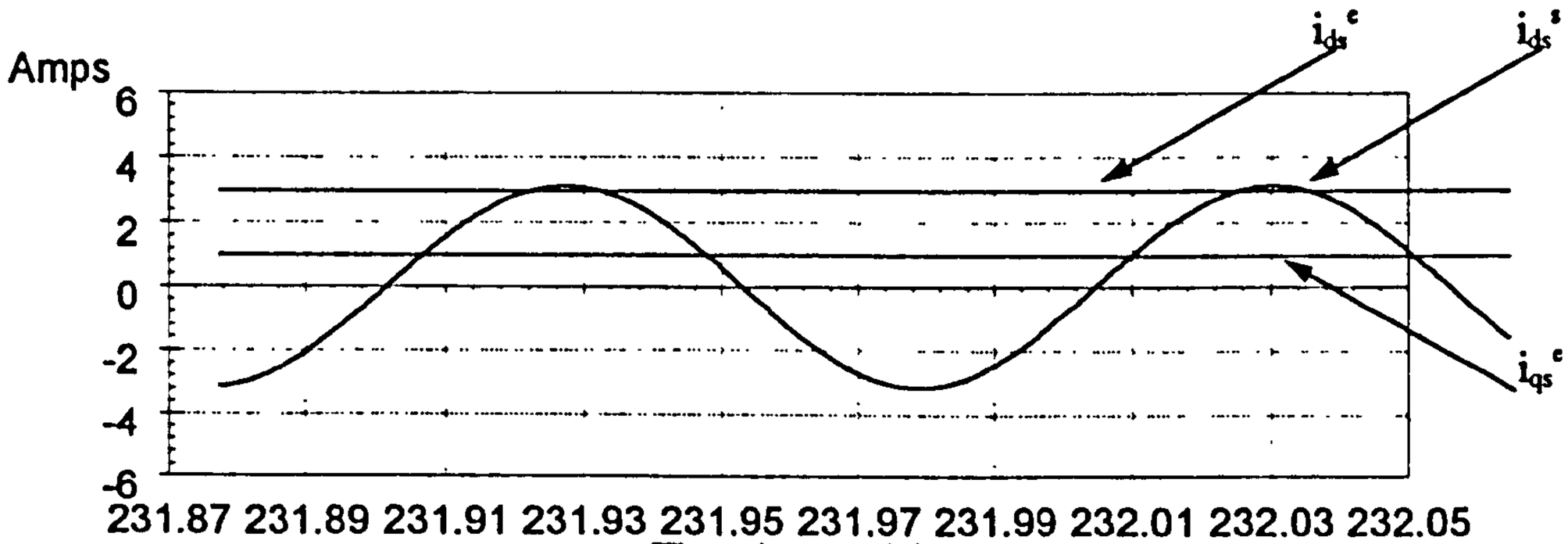


Figure 4.3.18: Simulated Currents $i_{d_s}^{e}$=3A $i_{q_s}^{e}$=1A

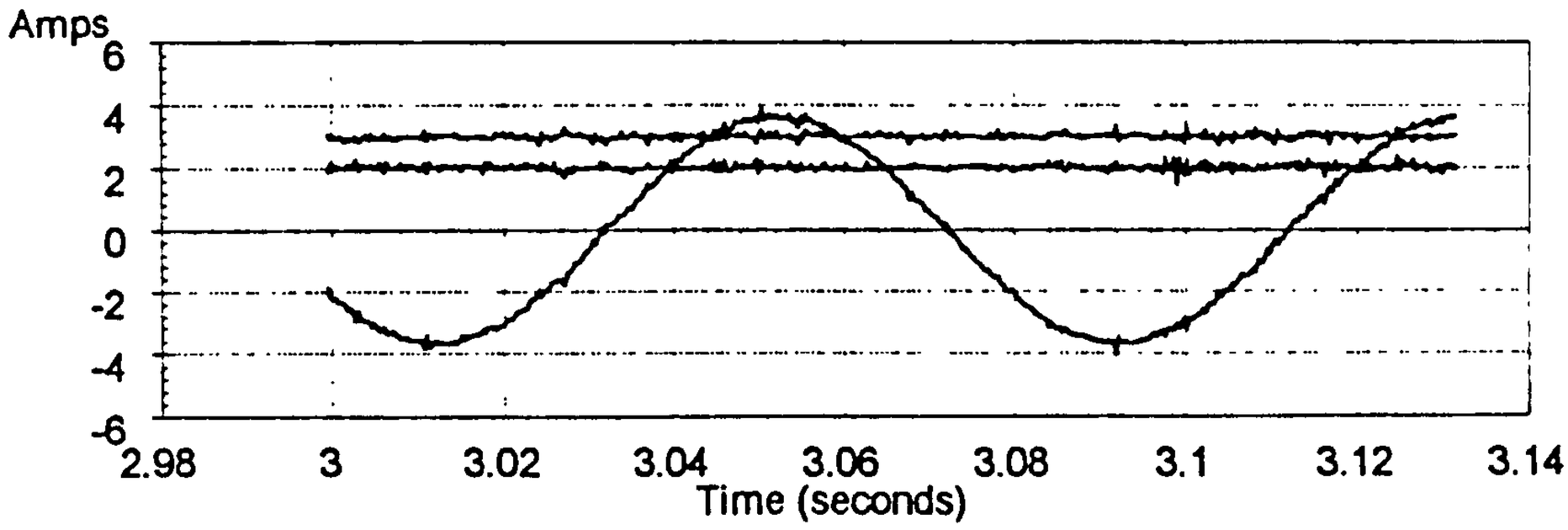


Figure 4.3.19: Actual Currents $i_{d_s}^e$=3A $i_{q_s}^e$=2A

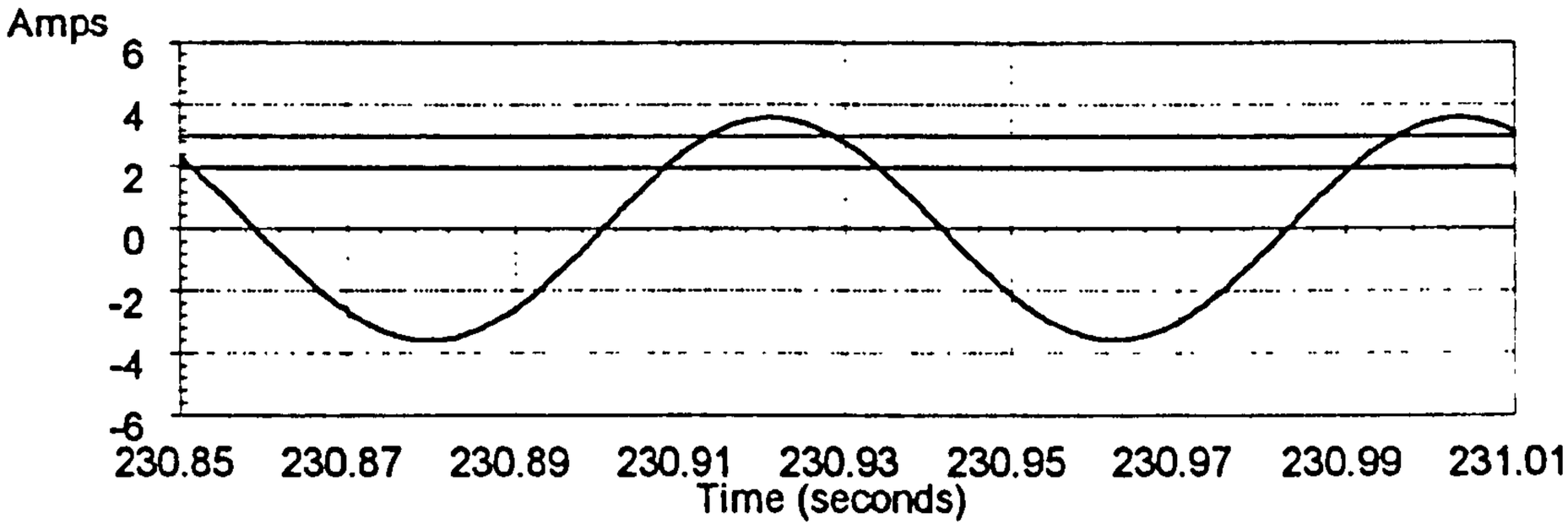


Figure 4.3.20: Simulated Currents $i_{d_s}^e$=3A $i_{q_s}^e$=2A

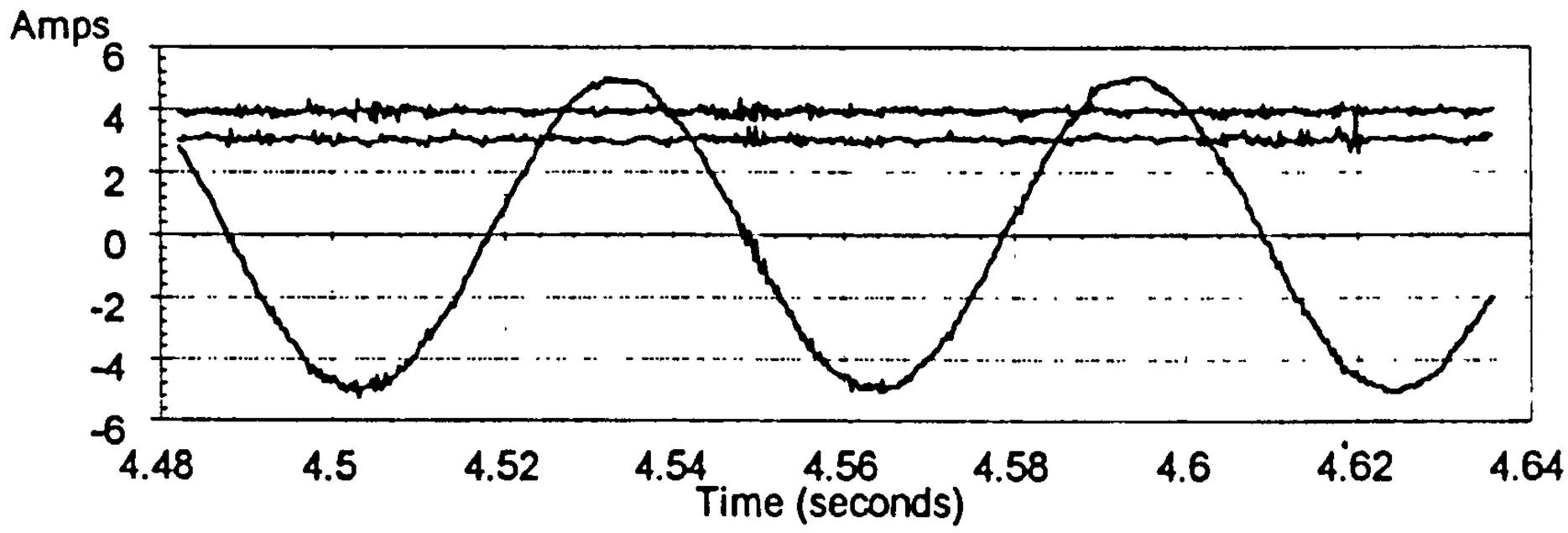


Figure 4.3.21: Actual Currents $i_{d_s}^e$=3A $i_{q_s}^e$=4A

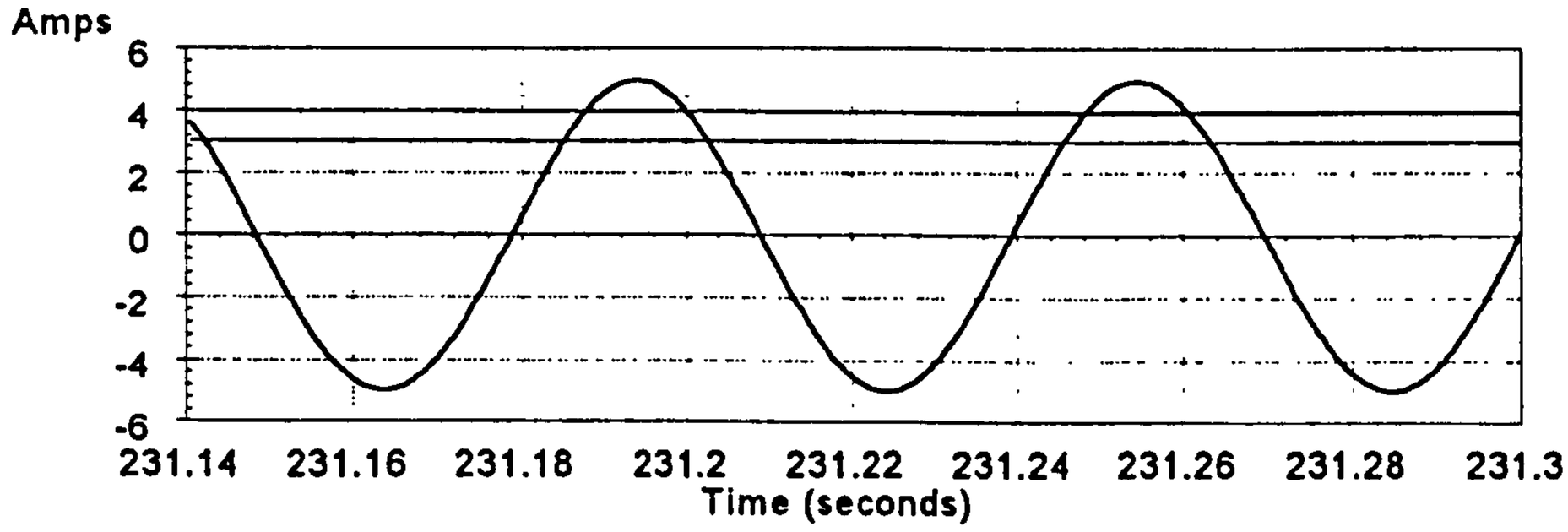


Figure 4.3.22: Simulated Currents $i_{d_s}^e$=3A $i_{q_s}^e$=4A

As with the open loop Volts/Hertz control simulation the steady state results for the vector control show that the off line non-real time simulation is in good agreement with the actual measured values of the real drive. The actual values contained measurement noise.

The following results compare the transient behaviour of the simulated drive to that of the actual drive. The first transient applied is a step increase in the torque component of stator current i_{qs}^{e} of 1 amp to 7 amps, with the flux component held constant at 3 amps. The following four figures show the D-axis stator current in the stator reference frame i_{ds}^{s} and also the D and Q axis stator currents in the rotor flux reference frame i_{ds}^{e} and i_{qs}^{e} which resulted from the simulation. The actual transient was repeated with the actual drive and the results of the actual current values which resulted are displayed together with those of the simulation.

The second transient applied is a step change in torque component of stator current i_{qs}^{e} of 7 amps to 1 amp. The second set of four figures show the simulated and actual currents which resulted from this test.

The simulated transient results for the vector controlled drive show close agreement with the measured values from the actual drive. They exhibit a higher degree of accuracy than the transient results taken from the open-loop controlled drive. In a vector drive the flux producing current component i_{ds}^{e} is held constant thus a rated flux is maintained within the machine. Because of this saturation of the main flux path is avoided and therefore the behaviour of the linear model is closer to that of the real machine. The model also resembles the actual more closely due to the closed loop control.

The results of the simulated vector controlled drive show how a simulation can provide a useful tool for predicting the behaviour of a drive system.

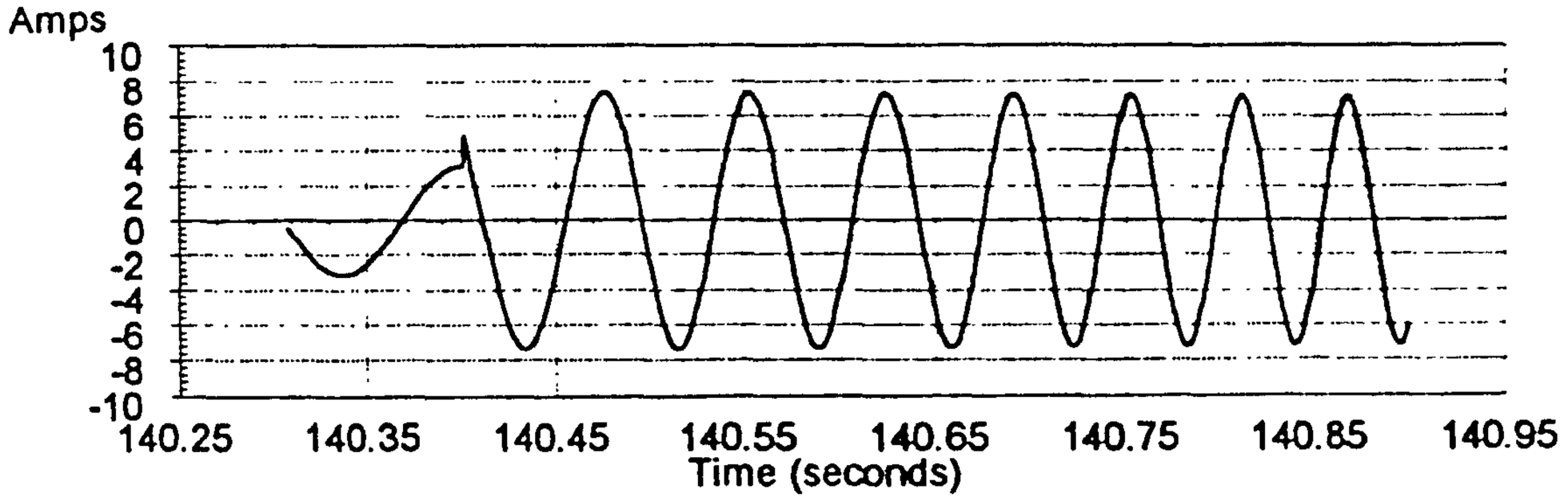


Figure 4.3.23: Simulated Stator Current i_{ds}^{s}

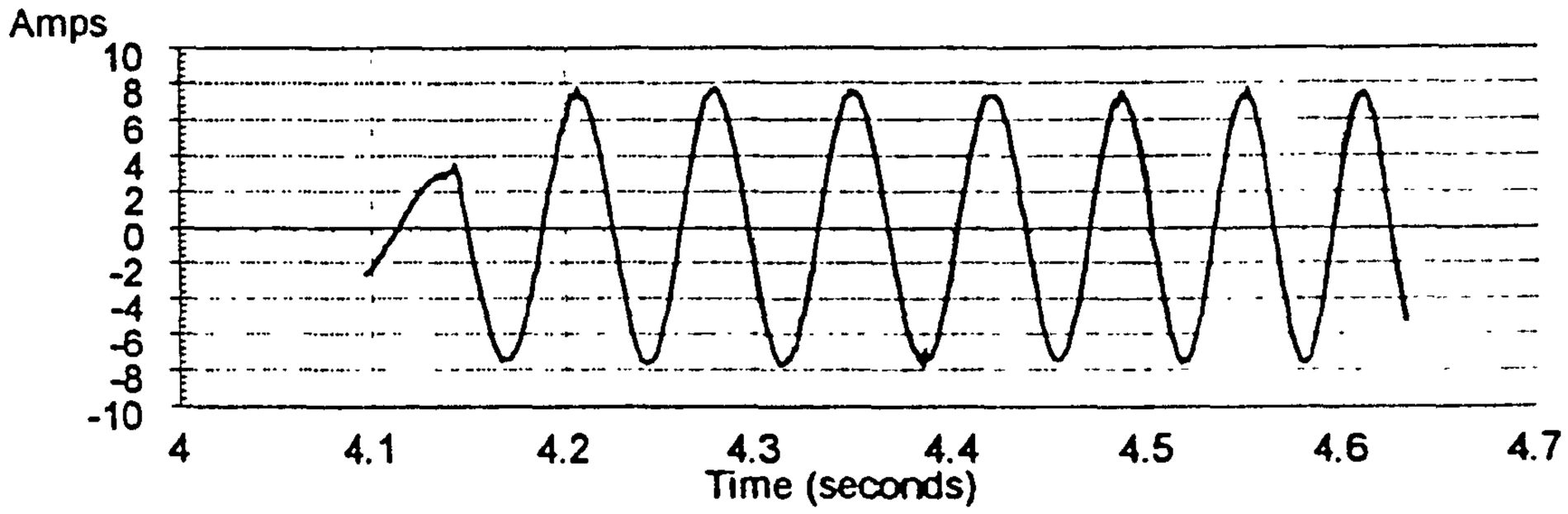


Figure 4.3.24: Actual Stator Current i_{ds}^{s}

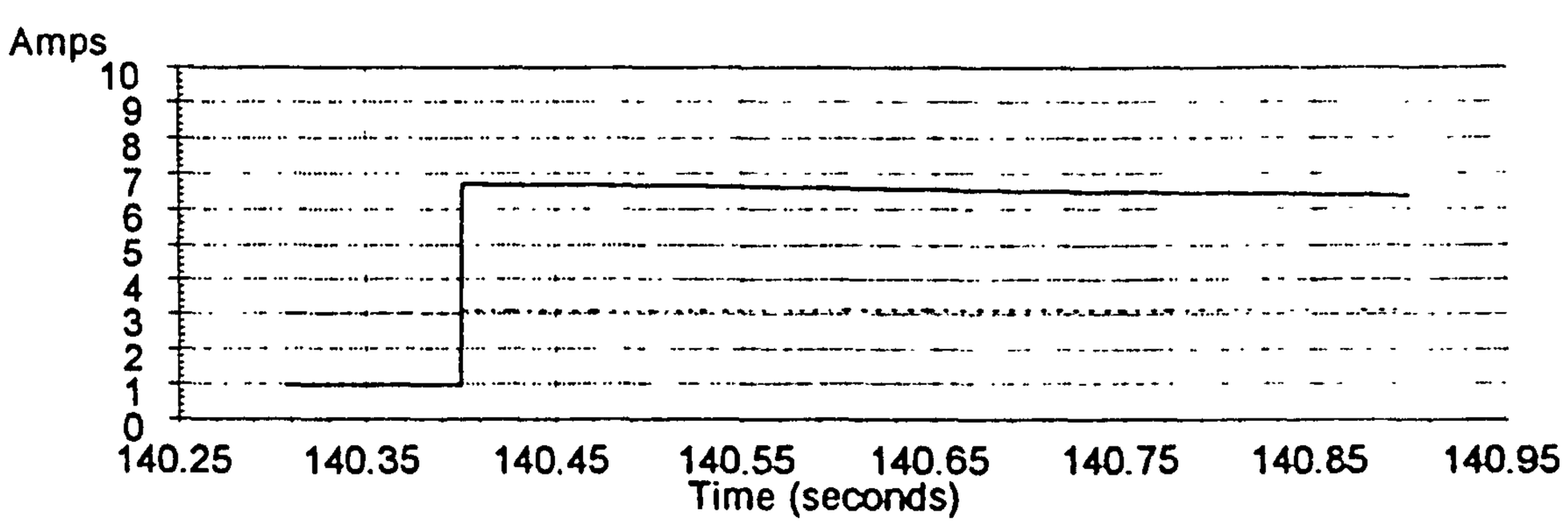


Figure 4.3.25: Simulated Currents $i_{d_s}^{e}$=3A $i_{q_s}^{e}$=1 to 7A

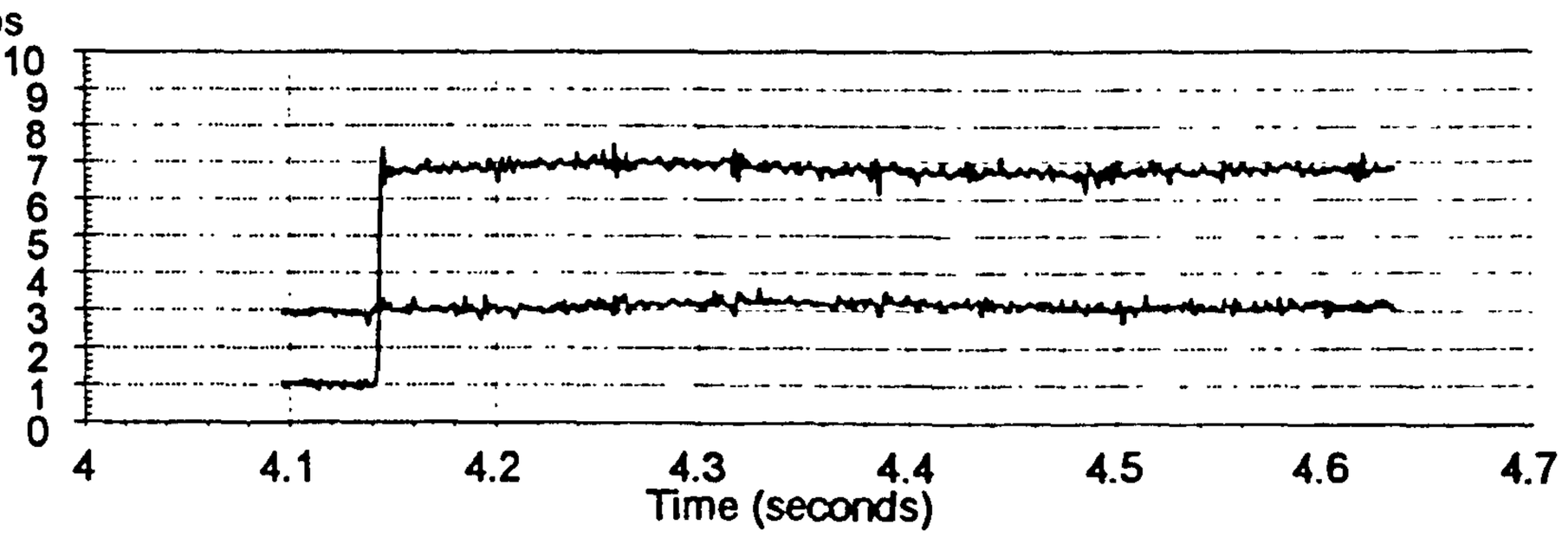


Figure 4.3.26: Actual Currents $i_{d_s}^{e}$=3A $i_{q_s}^{e}$=1 to 7A

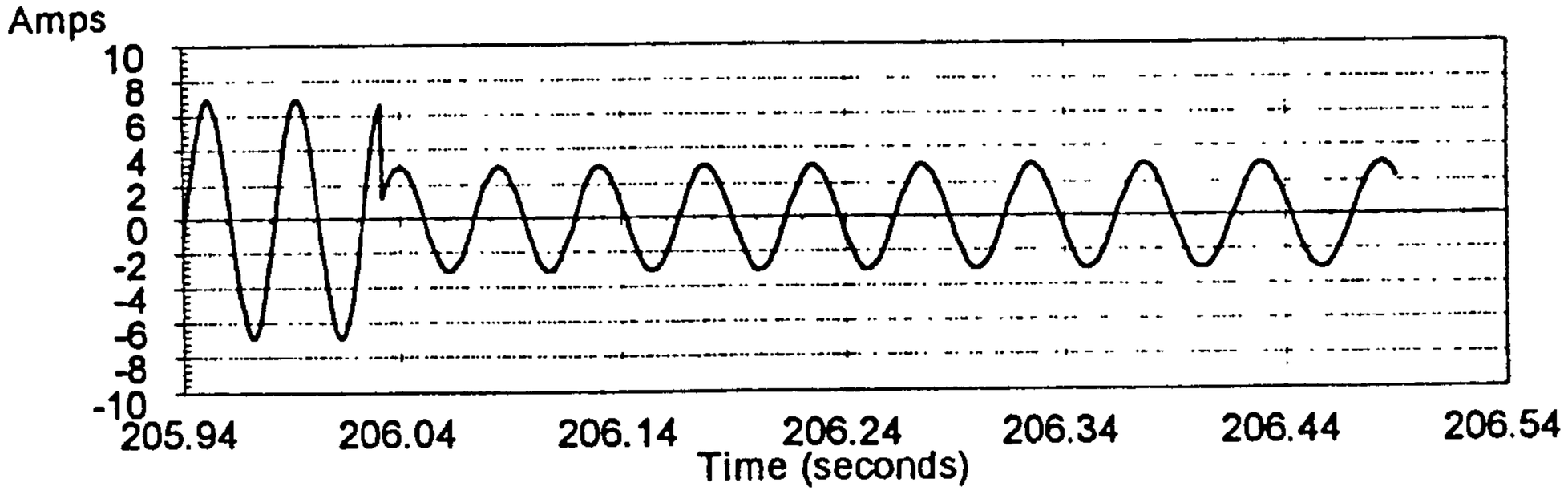


Figure 4.3.27: Simulated Stator Current i_{ds}^{e}

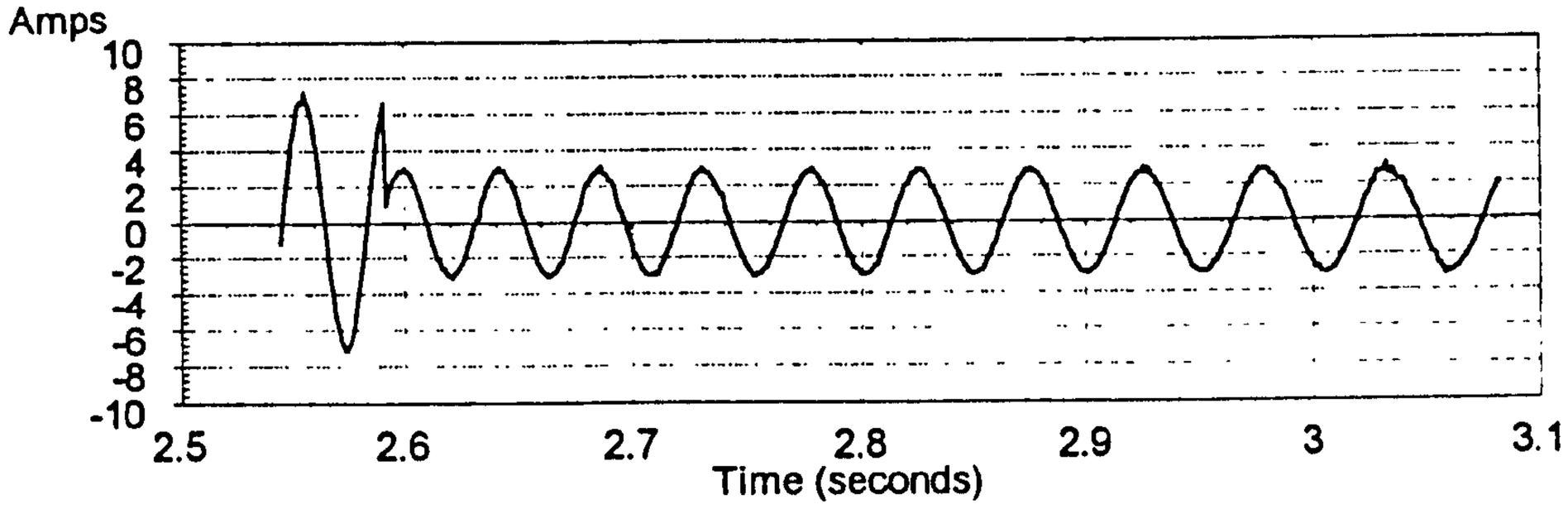


Figure 4.3.28: Actual Stator Current i_{ds}^{e}

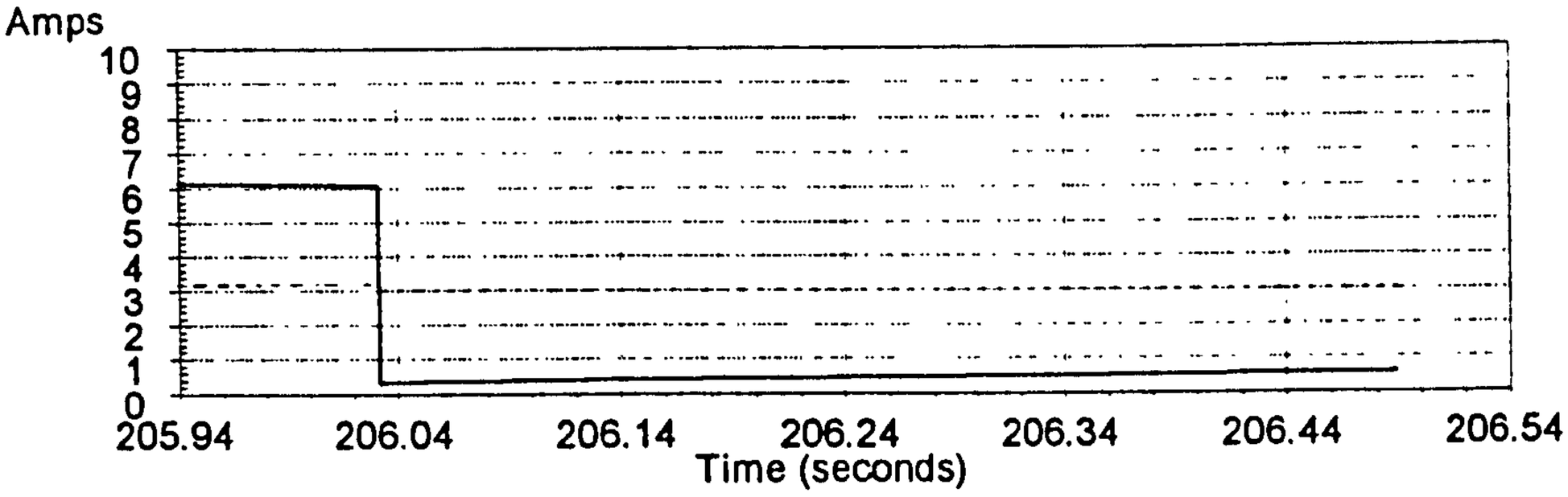


Figure 4.3.29: Simulated Currents i_{ds}^{e}=3A i_{qs}^{e}=7 to 1A

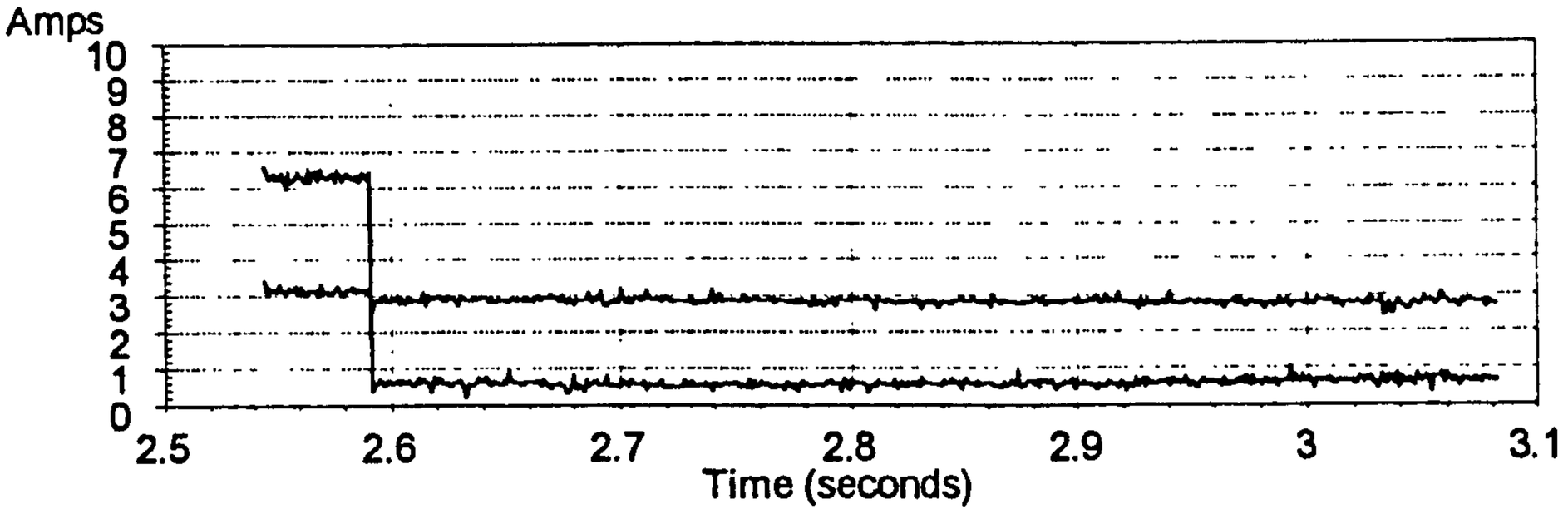


Figure 4.3.30: Actual Currents i_{ds}^{e}=3A i_{qs}^{e}=7 to 1A

4.4 Summary

This chapter has described two control algorithms suitable for use in the induction machine drive system. The first is a simple, open loop Volts/Hertz controller and the second is a more complex Vector controller. The two controllers are the most commonly employed algorithms in modern induction motor drive systems.

The two controllers were used to control the simulations of the inverter and machine, both of which have been detailed in the previous two chapters. This chapter has arrived at the point at which the simulation of the complete drive system can be compared with results taken from the actual drive system.

The limitations of the linear model used to simulate the machine were discussed with regard to the open loop Volts/Hertz controller, and these limitations were used to explain some of the discrepancies between the simulated results and the actual results taken for transient cases. Since it is one of the objectives of the Vector controller to maintain a constant level of flux in the machine, main flux path saturation will be avoided and the consequential variation of magnetising inductance should therefore be minimised. The Vector control results thus exhibited fewer inaccuracies than the open loop case.

In reality, small variations of magnetising inductance will occur in a Vector controlled drive system, and these variations can affect the performance of the controller and thus the behaviour of the complete drive system. Because the magnetising inductance is used as a parameter in the flux model within the Vector controller, any variation of this parameter will affect the estimate of the rotor flux vector which is used in the reference frame transformations. These transformations are crucial to the effective decoupling of the flux and torque producing components of stator current, and so a deterioration in the overall performance of the drive system will result from any variation of magnetising inductance. Recent work by [*Levi*, 1994] and [*Levi*, 1995] has demonstrated how main flux path saturation affects the behaviour of a Vector controlled drive. In this work a non-linear model of the induction machine

is used in a simulation of a Vector controlled drive, and the drawbacks of the standard flux model are highlighted as the machine exhibits saturation. A more complex flux model based on the non-linear machine model, which offers superior performance to the linear model, is proposed by Levi. This non-linear model uses the machine's magnetisation curve to obtain a time variant value of magnetising inductance dependent upon the value of magnetising current.

This chapter, together with the previous two, has described the main three blocks that make up the drive system, the machine, the inverter and the controller. The simulations of these three blocks have been connected and simulated within the PC non-real time environment. The results of the simulation have compared favourably with results from an actual drive system. The following chapter will investigate the possibility of realising this simulation in real time.

5. DIGITAL SIGNAL PROCESSING SYSTEM

5.1 Introduction

One of the main objectives of this project was to investigate the possibility of implementing a real time simulation of a drive system. This real time simulation would then provide a useful tool for the development of the drive's digital controller. Chapter two of this thesis reported on the inability of modern personal computers to achieve anywhere near real time simulation at the level that is required for an effective drive simulation. Another platform on which to build the simulation must therefore be considered.

The drive system is a collection of individual parts, the controller, the inverter and the motor. These individual parts are operating in parallel with each other and not in the sequential manner in which they are modelled in normal simulation software. They are simulated in this way because the personal computers used to run these simulations are incapable of performing parallel computation. If the simulation environment was capable of true parallel processing then the individual components of the drive system could be modelled as they appear in the actual drive arrangement. The parallel processing approach not only mimics the actual system more closely but also reduces the execution time of the simulation.

This chapter will detail the platform on which the real time simulator developed during this project is based. The system incorporates multiple digital signal processors in which parallel processing is exploited to achieve the goal of real time simulation.

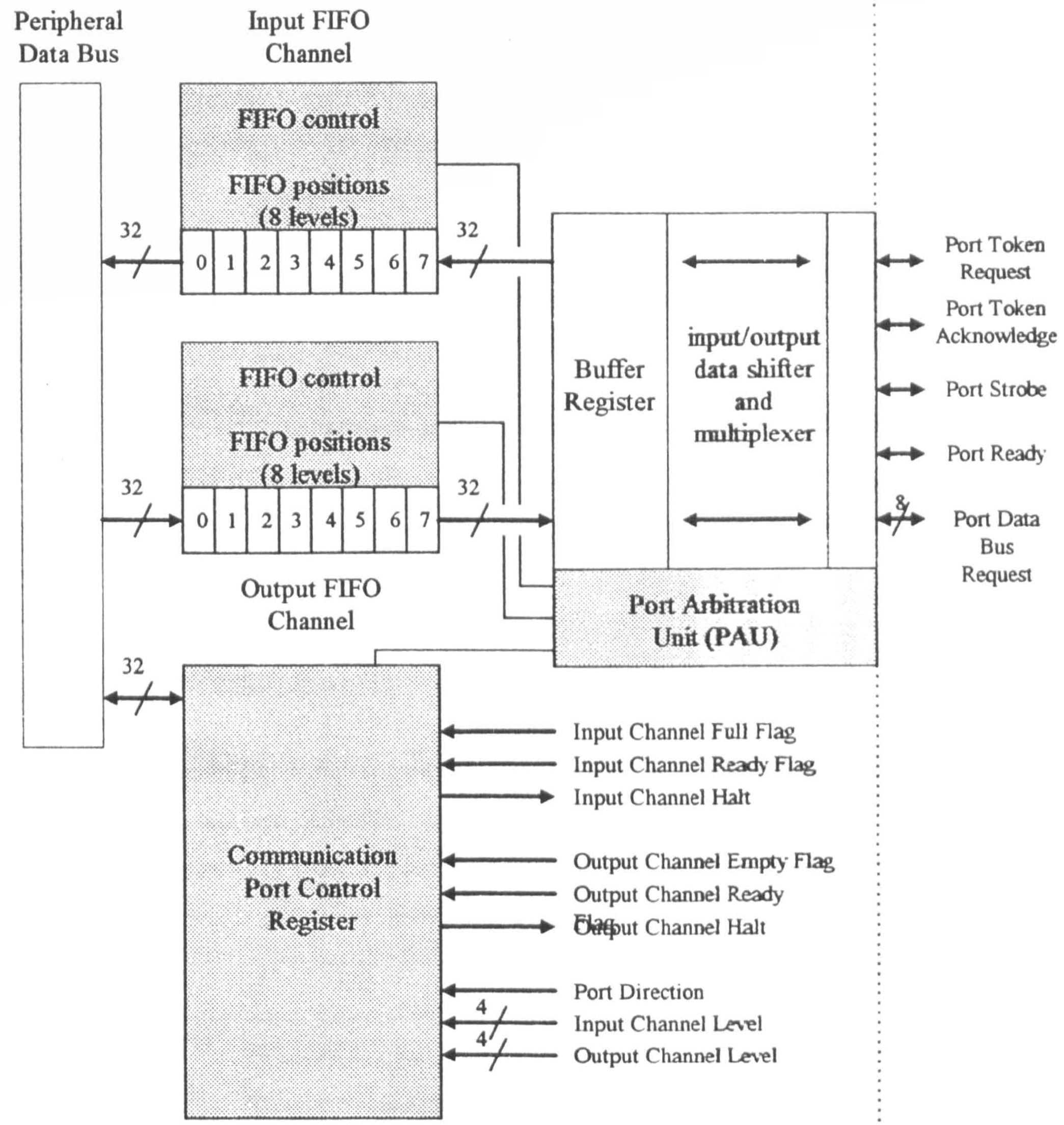
5.2 The Digital Signal Processor (D.S.P)

5.2.1 TMS320C40

The digital signal processor (D.S.P) used to perform the simulation is the Texas TMS320C40. The TMS320C40 (C40) belongs to the TMS320 generation of D.S.P.s designed by Texas. The C40 is an advancement of the C30 series that were the first floating point processors in the TMS320 generation of digital signal processors [*Texas Instruments*, 1991]. The C40 incorporates built in hardware designed specifically to promote high speed interprocessor communication.

The D.S.P. central processing unit (CPU) is a 32 bit floating point processor capable of 275 MOPS (millions of operations per second). The C40 has an instruction cycle time of 40 or 50ns depending on the clock frequency and the majority of its instructions are carried out in a single cycle. It also has hardware divide and inverse square root functions thus further speeding mathematical tasks.

The primary feature of the C40 is its parallel processing capability, the C40 has six communication ports dedicated to providing direct processor to processor connection. These communication ports allow bi-directional data transfer between processors at a rate of 20 Mbytes per second. They provide the ability to construct a network of interconnected C40s, producing a flexible parallel processing environment. The communication ports include buffering of all data transfers and have built in handshaking to synchronise communication. The following diagram shows a block diagram of a single communication port.



[*Texas Instruments*, 1991]

Figure 5.2.1: Communication Port Block Diagram

The block diagram of the communication port shows that each port has an input channel and an output channel. Each channel provides an 8 level, 32-bit first in first out buffer which isolates the data bus of the C40 from the data bus of the communication port. The Port Arbitration Unit (PAU) handles the handshaking and data transfer between the two communication ports which are connected together. The PAU uses four control lines and eight data lines, the data transfer cycle between the two ports is as follows:-

The PAU which wants to send data down the link but does not have possession of the link, requests the use of the link via the port token request line, the PAU which does have possession of the link then relinquishes it via the port token acknowledge line. The sending PAU can now write data to the link and signal that it has done this to the receiving PAU via the port strobe line. The receiving PAU then reads the data into its FIFO (first in first out) input channel and uses the port ready line to indicate to the sending PAU that the data has been received.

The FIFO buffers are extremely important to the parallel processing system, one processor can write 8 x 32-bit words to the buffer and then the PAU of that port will take over the data transfer allowing the processor to carry on uninterrupted. The communication ports are controlled via the Communication Port Control Register. Each of the six ports has a control register which contains control and status bits for that port. The communication port provides four interrupts:-

Input Channel Ready	(ICRDY)
Input Channel Full	(ICFULL)
Output Channel Ready	(OCRDY)
Output Channel Empty	(OCEMPTY)

These interrupts can be used to interrupt the processor when data is available from another processor or when it is clear to write data to another processor. This allows the tasks performed in multiple processors to be synchronised.

The port control register also allows the processor to halt data transfer via the input or output channels, the data transfer will remain halted until the processor frees it or a system reset occurs. The control register can provide information about the state of the input and output FIFO buffers, this is done via the Input Channel Level and Output Channel Level lines. Each channel level line is four bits wide and represents the number of full levels in the buffers, i.e. '0000' means the buffer is empty, '0010' means that two levels are occupied and '1111' means that the buffer is full.

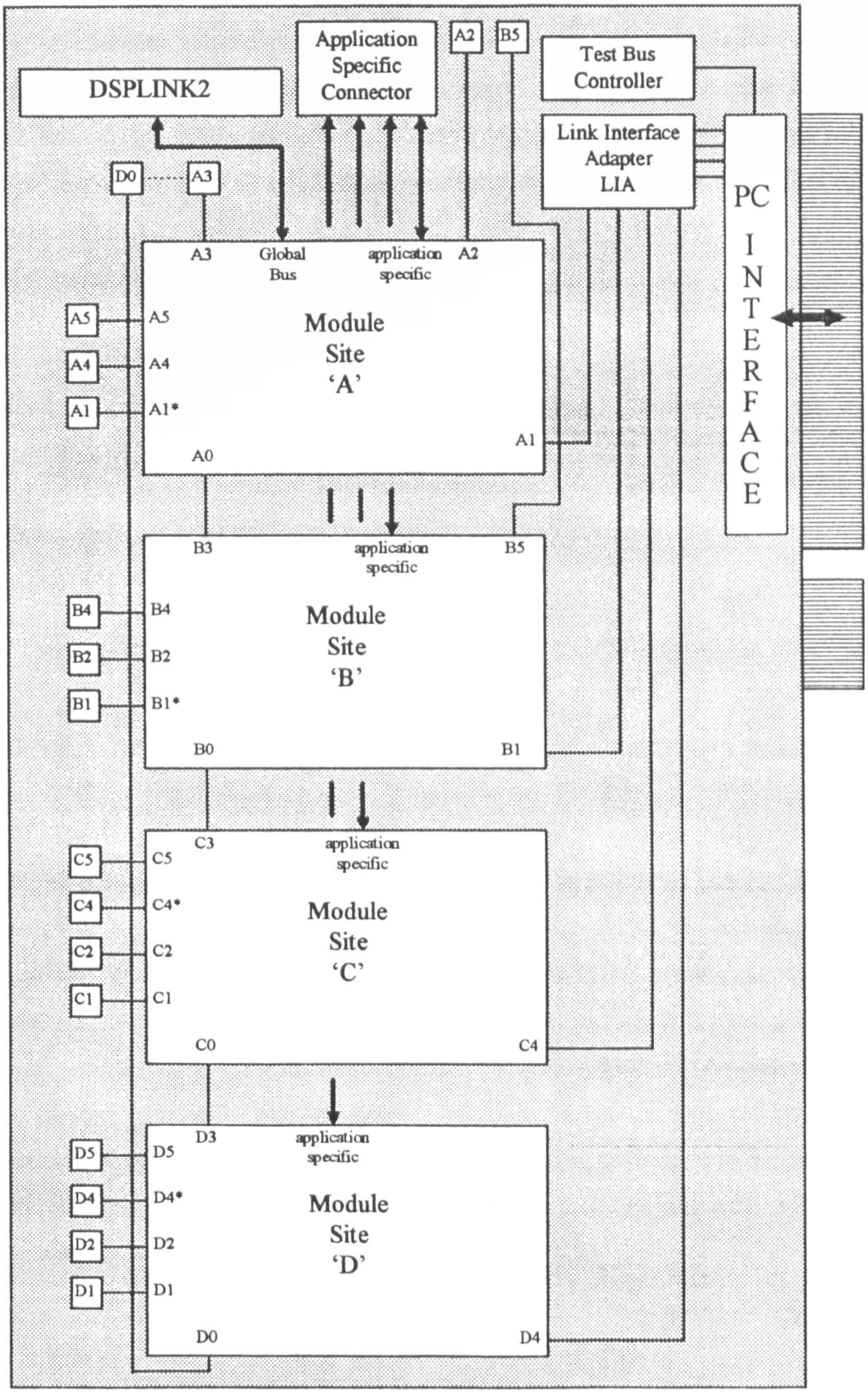
5.2.2 TIM-40 Module

The multiple D.S.P system is based on a single "mother" board. This mother board is a commercial item which has four processor sites. Each processor site is suitable for one processor module (TIM-40 module). The TIM-40 module specification was designed by Texas to give manufacturers of C40 based products a standard to comply to when designing mother and "daughter" boards. The TIM-40 modules used in the multiple processor system are commercial items from Loughborough Sound Images. Each module has a single TMS320C40 processor, together with three banks of zero wait state static RAM and a small identification ROM which holds hardware dependent set-up values accessed by the processor.

5.3 The Mother Board

5.3.1 QPC-C40

The mother board is a standard commercial item designed by Loughborough Sound Images and is called the QPC-C40. The QPC-C40 has sites for up to four TIM-40 modules, it is designed to be mounted in a standard PC and interfaces to the PC via the PC AT bus. The following figure shows a block diagram of the QPC-C40 mother board.



* only available if LIA is not used

Figure 5.3.1: QPC-C40 Mother Board Block Diagram

The four module sites are interconnected using ports 0 and 3 via the mother board, one other port of each site is connected via the Link Interface Adapter to the PC AT bus. The remainder of the communication ports are brought out to headers on the QPC-C40 board thus allowing the user to match the interprocessor connection to his specific requirements.

5.3.2 Test Bus Controller

The D.S.P module sites can be accessed by the PC in two ways, either the Test Bus Controller can be used or the Link Interface Adapter. The Test Bus Controller has the same functionality as the Texas Instruments XDS510 emulation system. The Test Bus Controller is used by the debugging system DB40 to allow the user to download and run code within the debugger environment.

5.3.3 Link Interface Adapter (LIA)

The Link Interface Adapter (LIA) interfaces the PC to each module site via one of that sites communication ports, if the LIA is used to access the D.S.Ps then the communication port that the LIA is using will not be available for interprocessor connection. The LIA is the main interface between each D.S.P module site and the host PC, it allows code to be downloaded and executed to each D.S.P using library functions which can be embedded into standard PC software. The LIA emulates the action of a communication port and because of this it can transfer data and interrupt the processor via the port in the same manner as any other processors in the parallel system. Any code within the D.S.P which talks to other D.S.Ps via a communication port can also talk to the LIA.

5.3.4 DSPLINK2

If a TIM-40 module equipped with a global bus connector is used in module site 'A' then the D.S.P in that module will have access to the DSPLINK2 interface on the motherboard. This provides the D.S.P with a direct connection to the outside world without having to go via the PC AT bus. DSPLINK2 is a 32-bit parallel, high speed, bi-directional interface which is mapped directly into the global memory map of the D.S.P located in module site 'A'. The DSPLINK2 can be used to interface to other commercial boards which are equipped with a similar interface or a user specific board can be designed to interface to the link.

5.4 The Present System

The multiple processing system used throughout this research consists of three C40 D.S.Ps mounted on one QPC-C40 motherboard. The motherboard is mounted inside an IBM compatible PC and interfaces to the outside world via the DSPLINK2 connection of the D.S.P module located in module site 'A' of the motherboard. The D.S.Ps are interconnected as shown in Figure 5.4.1. The diagram shows the interconnections of the communication ports of the three D.S.Ps, each communication port used for interprocessor data transfer is dedicated to either input or output.

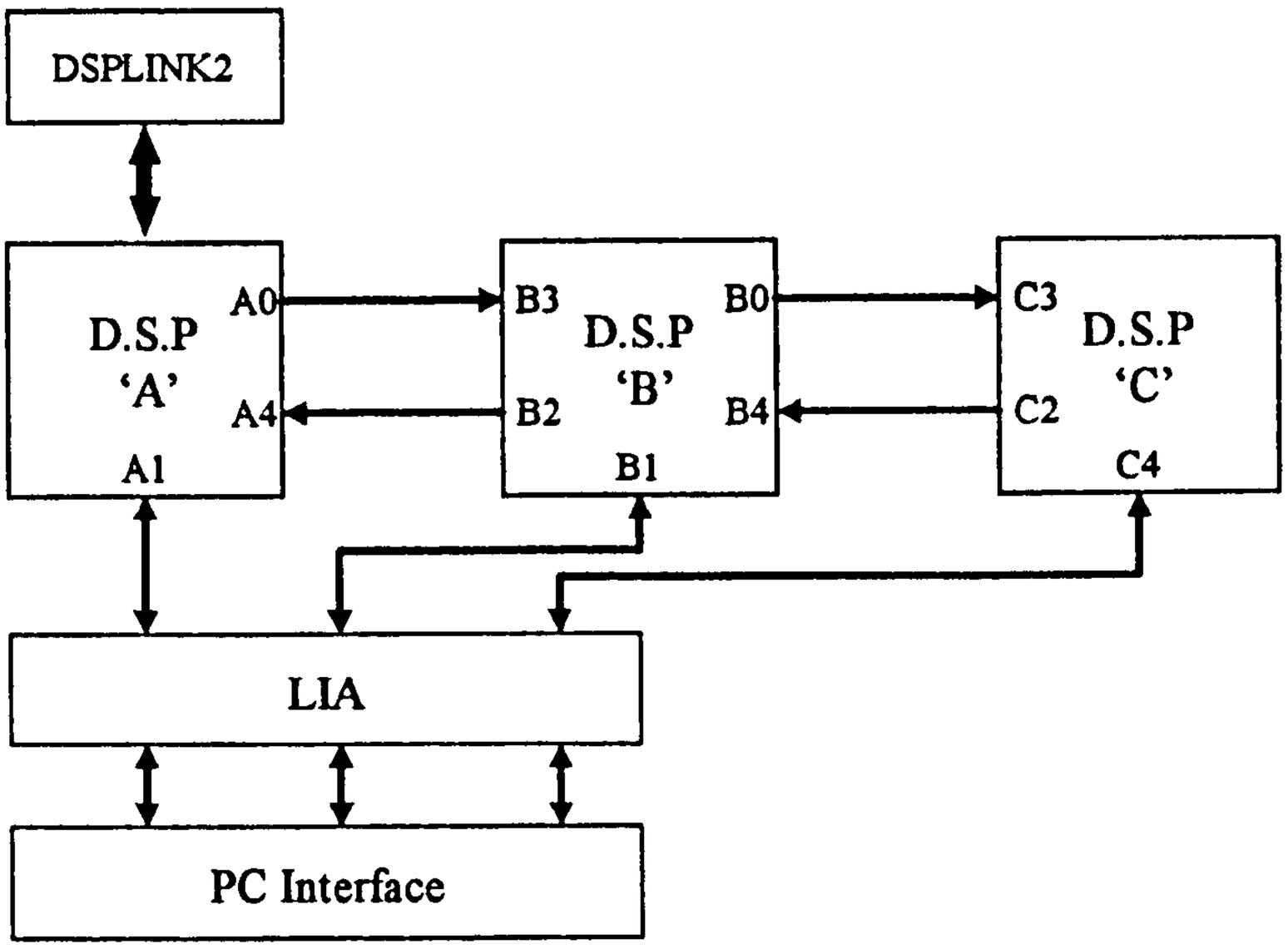


Figure 5.4.1: Block Diagram of Present System

The simulation code must be broken down into logical blocks and distributed between the three processors. One of the advantages of achieving real time simulation is that the controller which is included in the simulation can be the actual controller which is to control the real drive. In the present system the controller is one of the D.S.P modules and it is dedicated to the control functions and no other. If the TIM-40 module is to be used as the controller for a commercial drive then the actual TIM-40 module can be inserted into the simulation environment and become part of the simulation, this allows the actual control algorithms to be experimented with and modified within the safe environment of the simulation. Once the controller is tuned and has proved to perform correctly the module can be removed from the simulation environment and inserted into the actual drive with no translation of control software.

The laboratory set up has a real time simulation of the drive system running in parallel with the actual drive. The same controller is used to feed the actual and simulated drives. This allows a direct comparison to be made between the simulated and actual drives.

Within the present system the controller D.S.P is located in module site 'A' of the Quad board inside the PC. The controller sends its demand signals to the actual inverter using the DSPLINK2 connection on the Quad board, these demand signals are also sent to the simulation of the inverter and induction machine. The remaining D.S.Ps are used to perform the simulation routines and real time data acquisition tasks. The PC acts as the user interface to the whole system, it can send or receive data to or from any of the D.S.Ps. The following figure shows the hardware of the present system, the voltage source inverter contains a PWM ASIC which receives the 3 phase voltage demands from the controller and then produces the device switching signals which are required for those demands. The switching devices of the inverter then apply the modulated DC link voltage to the Induction Machine.

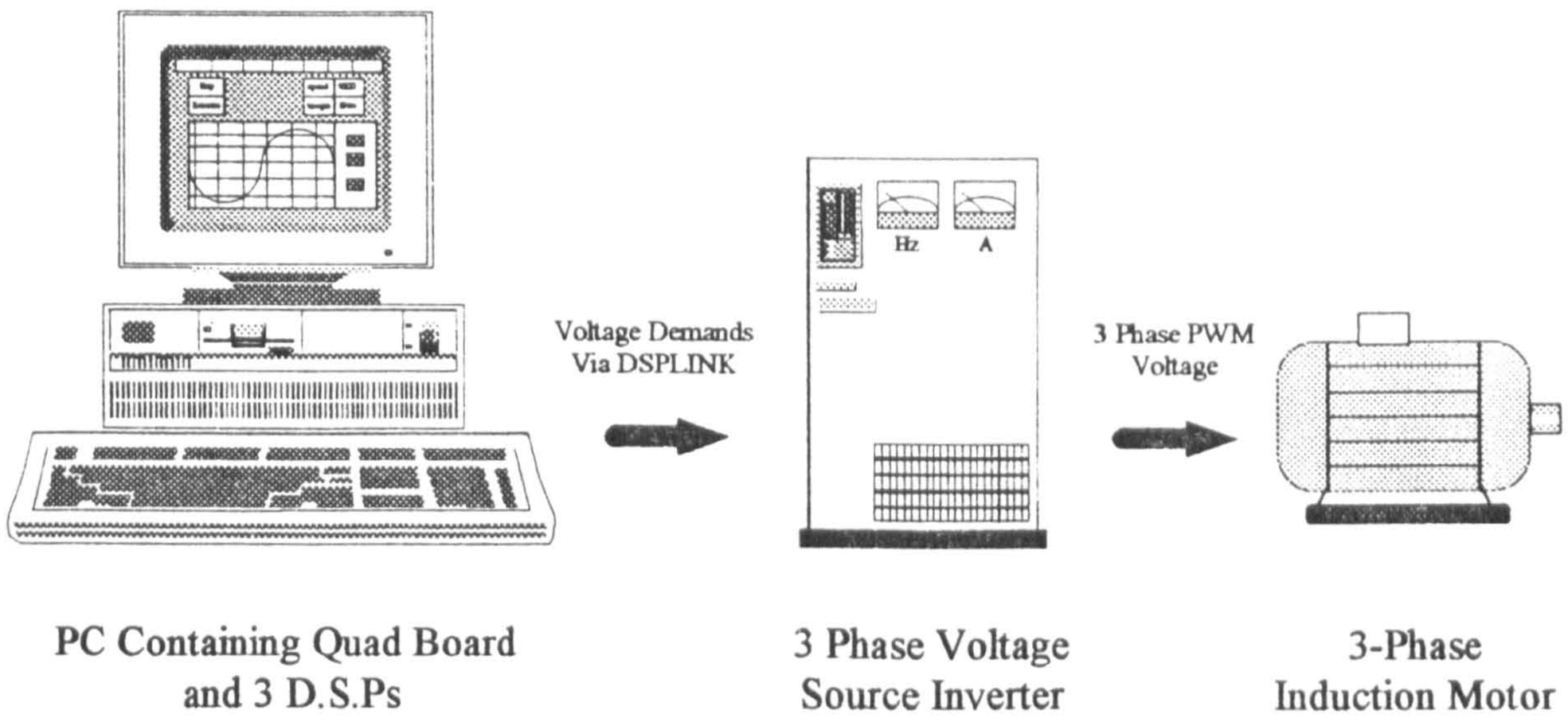


Figure 5.4.2: Present System Hardware

The multiple processor system is interrupt driven, two types of interrupt are used to synchronise the processors. Firstly the internal timer interrupt is used in the controller D.S.P to accurately time the control loop, and secondly the communication port which has received data interrupt is used to key data transfer between the D.S.Ps and also the PC. Each D.S.P is set up to service two interrupt routines as follows.

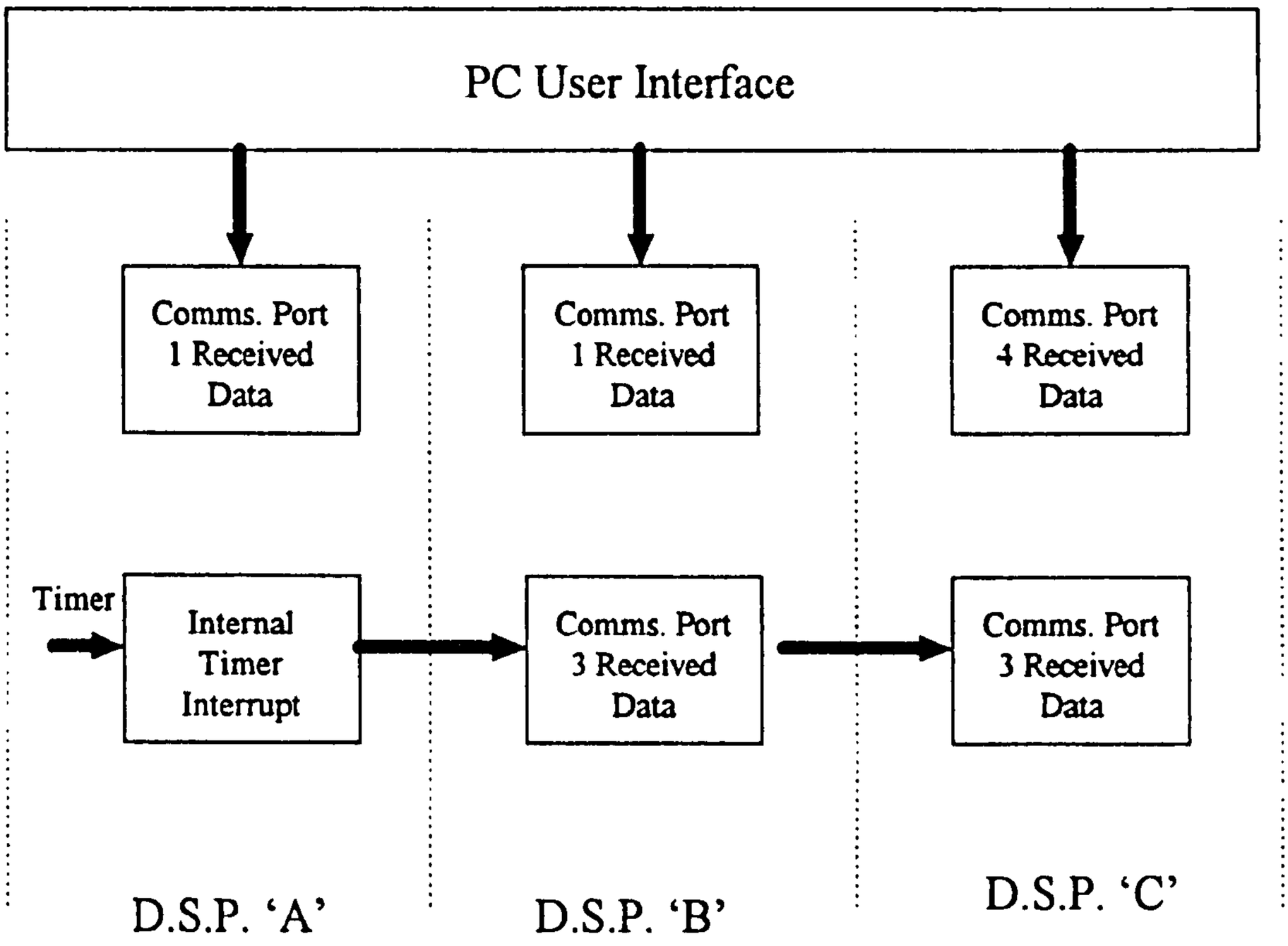


Figure 5.4.3: Multiple Processor Interrupt System

It can be seen from the interrupt layout above that D.S.P.'A' is interrupted by its internal timer which is set to the control frequency of the system. D.S.P.'B' is interrupted by D.S.P.'A' via its communication port 3 and D.S.P.'C' is interrupted by D.S.P.'B' also via its communication port 3.

The interrupt to D.S.P.'B' is made from within the timer interrupt routine of D.S.P.'A' and the interrupt to D.S.P.'C' is made from within the port 3 interrupt

routine of D.S.P.'B' therefore the interrupt routines of all three processors are synchronised to a single internal timer. By using the interrupt system outlined here D.S.P.'A' acts as the master processor and the routines carried out within the remaining D.S.Ps are slaves to it. The user interface software which is executed within the PC processor can interrupt any of the three D.S.Ps, these interrupts depend upon the users commands.

Details of the code executed in each of these interrupt service routines, together with a description of the user interface will be given in the following chapter of this thesis.

5.5 Summary

This chapter has described the hardware used in setting up the multiple digital signal processing environment. This environment has been developed in order to achieve real time simulation of the drive system. The chapter has given an insight into the key elements of the chosen processor and motherboard which make them ideal for this particular application. The hardware set-up of the present system has been outlined and the interconnection of the communication ports of the multiple processors has been described. Finally the interrupt system which dictates the execution of software within the each processor has been highlighted. The hardware system described in this chapter will be used to implement the simulations outlined in the previous three chapters and the performance of the real-time simulation will be discussed in Chapter 6.

6. REAL TIME SIMULATION

6.1 Introduction

Chapters 2, 3 and 4 of this thesis have described the simulation of an induction motor drive system which runs in a sequential manner in a standard PC system. They have also described how a multiple parallel processing system can be built up using Texas TMS320C40 digital signal processors located on a common mother board mounted inside a PC. The objective of this chapter is to implement the simulation software and the controller software which has been described, within the multiple digital signal processing system and investigate the possibility of achieving real time simulation with this system. The controller will be implemented in a single processor, as is the case in a standard drive system, and will be used to control both actual induction machine via a three phase voltage source inverter and also a simulation of the same system. The simulation software will be implemented in a second processor, and a third will be used for data acquisition. A description of which tasks are carried out by each of the processors within the parallel system will be given and a comparison of the simulation to the actual drive will be presented by executing the simulation in parallel with the actual drive system.

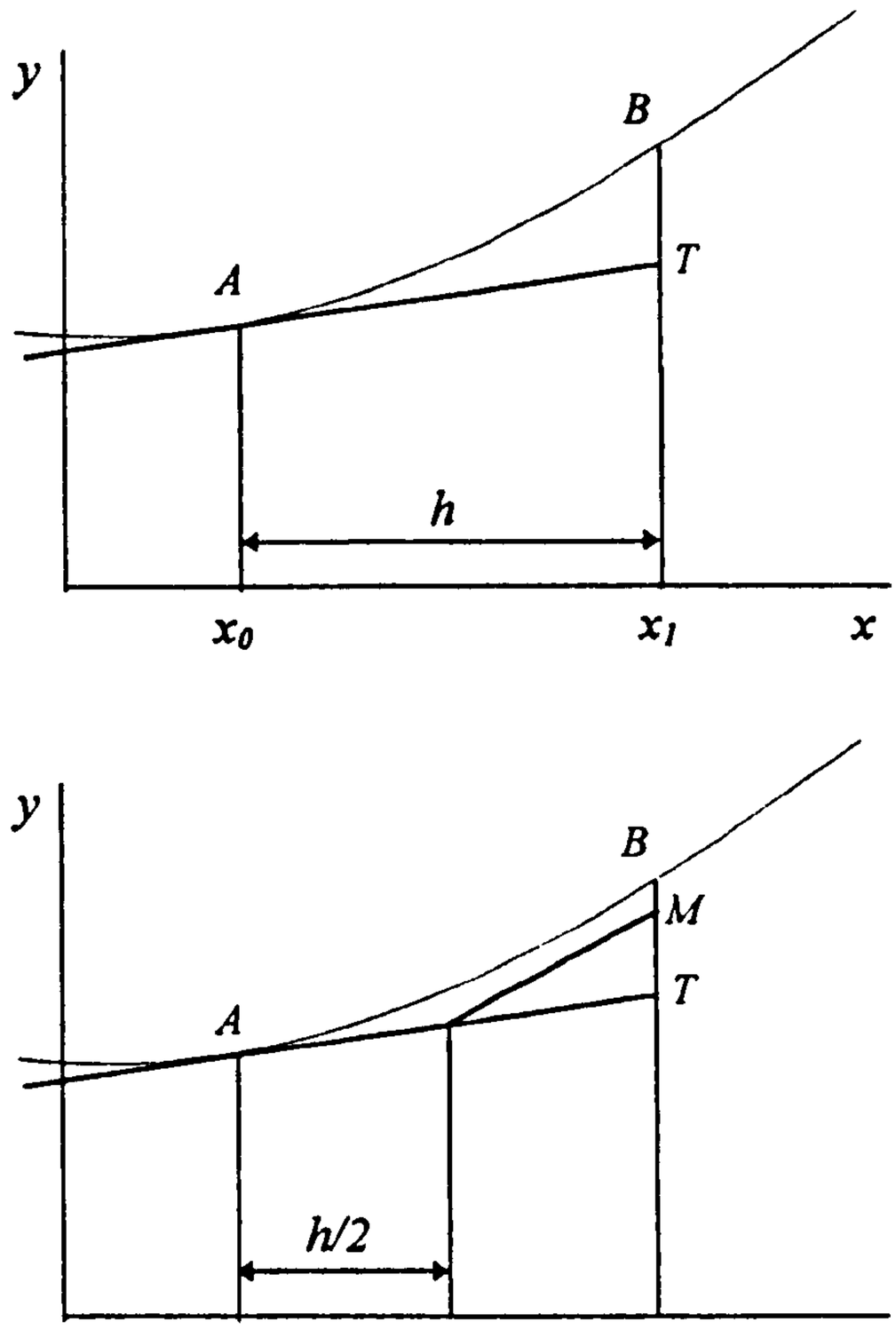
6.2 Euler-Cauchy Method

Two numerical methods were considered in chapter two of this thesis for solving the motor model differential equations, the very simple Euler method and the more complex Runge-Kutta fourth order method. These two methods were discussed because they represent extremes of accuracy and execution time, the Euler method offering very fast computation but low accuracy and the Runge-Kutta offering high accuracy but long computation time. When implementing the simulation code of the

previous chapters in the digital signal processor, a solver had to be chosen which would allow the code to be executed in real time. This meant that all of the simulation had to be executed faster than the control period imposed by the chosen switching frequency.

The majority of the computation carried out in the simulation involves the solution of the motor model electrical differential equations, for this reason a fast solver is required. The Euler method , which is the quickest, has the drawback of accuracy but can be executed in the D.S.P. in approximately 55μs, Runge-Kutta fourth order is five times slower and so could be executed in approximately 275μs. The chosen switching frequency of 6.5KHz means that the control period is 153.8μs which immediately rules out Runge-Kutta unless the switching frequency is dropped substantially. This echoes the point made previously that synchronisation with the digital cycle implies short time steps and hence simple time stepping schemes.

The Euler-Cauchy method, or improved Euler method, of solving differential equations is based on the Euler method but offers considerably higher degrees of accuracy over the simple Euler method [*Jeffrey,* 1986]. Euler is a 1st order method whereas Euler-Cauchy is a 2nd order method, the Euler-Cauchy could be described as Runge-Kutta 2nd order [*Maron,* 1987]. In the standard Euler method the slope of the function is used over the whole interval to predict the result after this interval. In the Euler-Cauchy method the slope of the initial values is used together with the slope at the first Euler solution to the equations, the average of these two slopes is then used to obtain a more accurate solution. The following figures show graphically the standard and improved Euler methods:-



[*Stroud*, 1986]

Figure 6.2.1: Euler-Cauchy Method

From the above figure it can be seen that the standard Euler method gives an error of *BT* by using the slope at x_0 over the whole interval *h*. In the Euler-Cauchy method this slope is used for the first half of the interval only and then the estimate of the slope of the curve at x_1 is used for the remainder of the period. This results in the error *BM* which is greatly reduced when compared to the first method. For the function $y`=f(x,y)$ with initial condition $x=x_0$, $y=y_0$ the point y_1 can be calculated as:-

$$y_1 = y_0 + 0.5h\,(y`_0 + f(x_1 , \overline{y_1})) \qquad (6.2.1)$$

where $\overline{y_1}$ is the prediction of y_1 gained from the standard Euler method i.e.

$$\overline{y_1} = y_0 + h\,(y`_0) \qquad (6.2.2)$$

In order to practically illustrate the error levels typical of the three methods using models and time steps typical for the applications under consideration the following figures show comparisons executed on the PC simulation, the simulation is for the motor model only, being fed from a sinusoidal three phase 50Hz voltage with a fixed time step of 50μs. Figure 6.2.2 shows the D-axis stator current for the three methods, the maximum error between the standard Euler and the Runge-Kutta can be measured to be approximately 1 amp. The error between the Euler-Cauchy method and the Runge-Kutta is extremely small and is shown magnified in the figure 6.2.3. This figure shows the maximum error to be approximately 5mA which is 0.14% of the peak Runge-Kutta current.

Figures 6.2.4 and 6.2.5 show the D-axis rotor current, the motor model was running on no-load and so the rotor currents should be zero. The Euler method estimates a significant rotor current as shown in the figure 6.2.4. Figure 6.2.5 shows the rotor current predicted by Euler-Cauchy and Runge-Kutta in more detail. Runge-Kutta predicts a peak current of 0.8mA the Euler-Cauchy method predicts a peak current of 6mA which, against a stator current of 2.5 A is negligibly small.

The execution time for the Euler-Cauchy method within the D.S.P. is 76.4μs which is an increase of 21.4μs over the standard Euler method, however, the increase in accuracy over the standard Euler method justifies this additional computation time. The Euler-Cauchy method has therefore been chosen for the real time simulation to solve the motor model electrical equations.

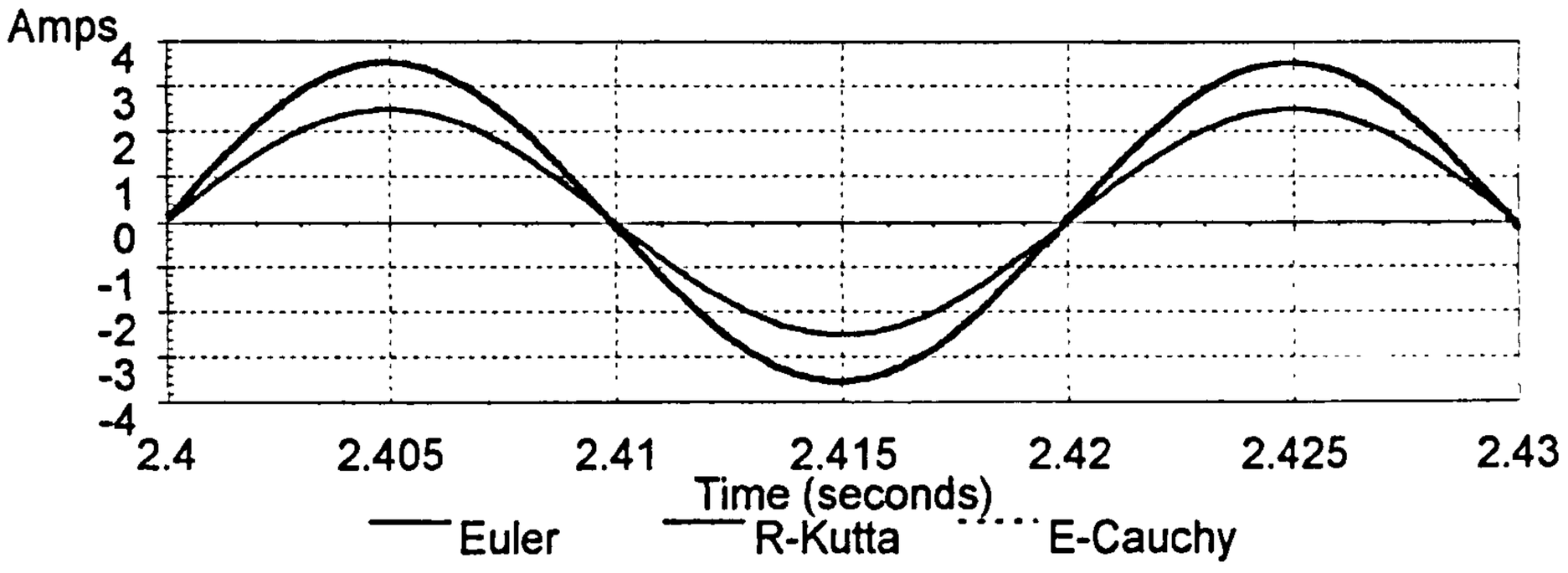


Figure 6.2.2: D-Axis Stator Current for Euler, Runge-Kutta and Euler-Cauchy

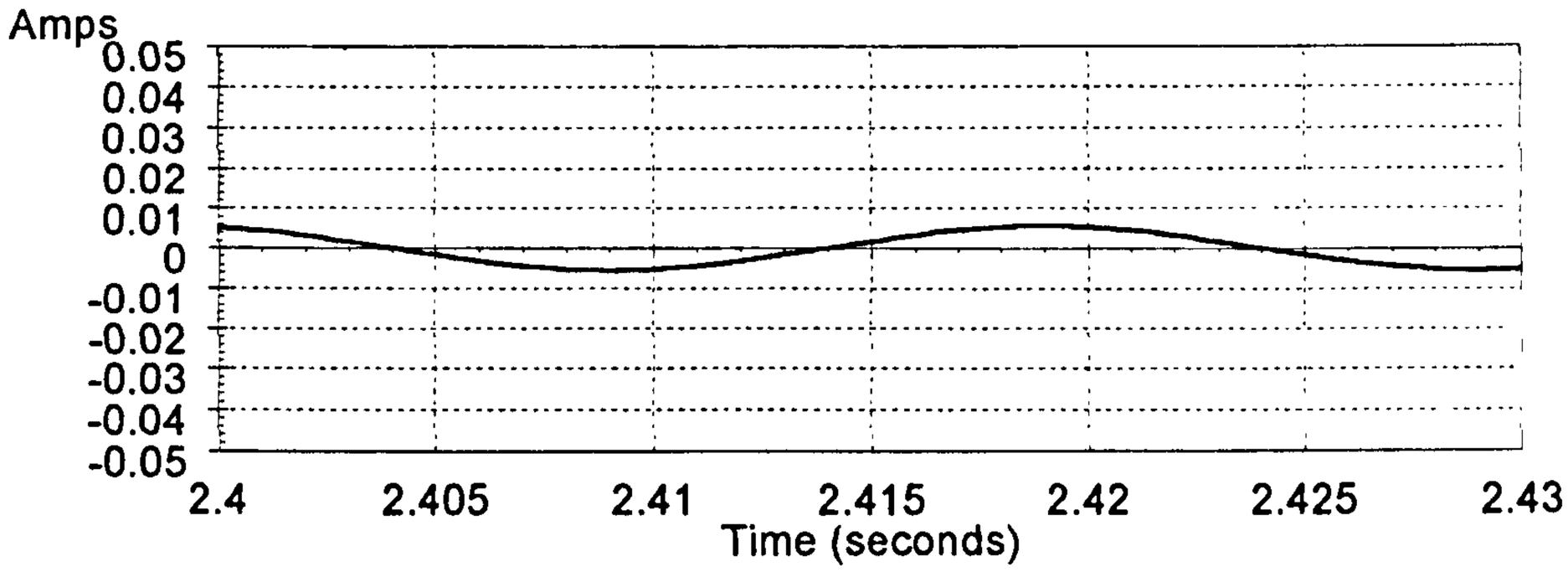


Figure 6.2.3: Error Between Runge-Kutta and Euler-Cauchy

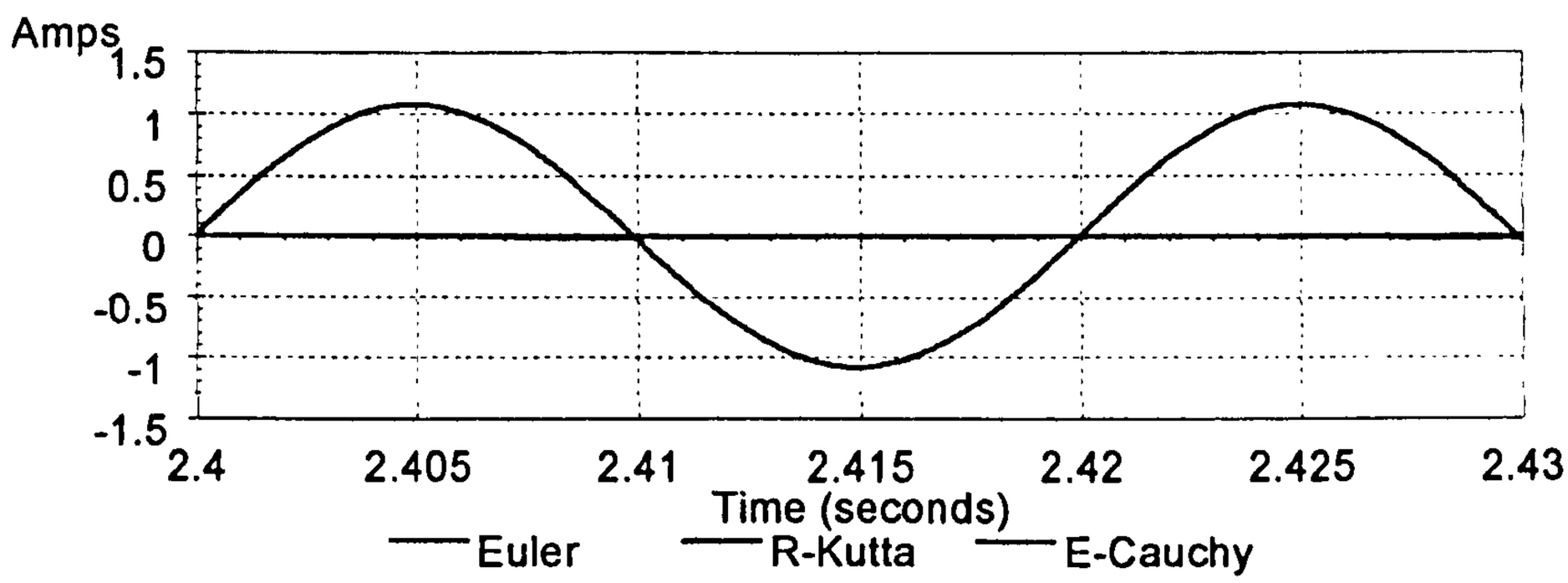


Figure 6.2.4: D-Axis Rotor Current for Euler, Runge-Kutta and Euler-Cauchy

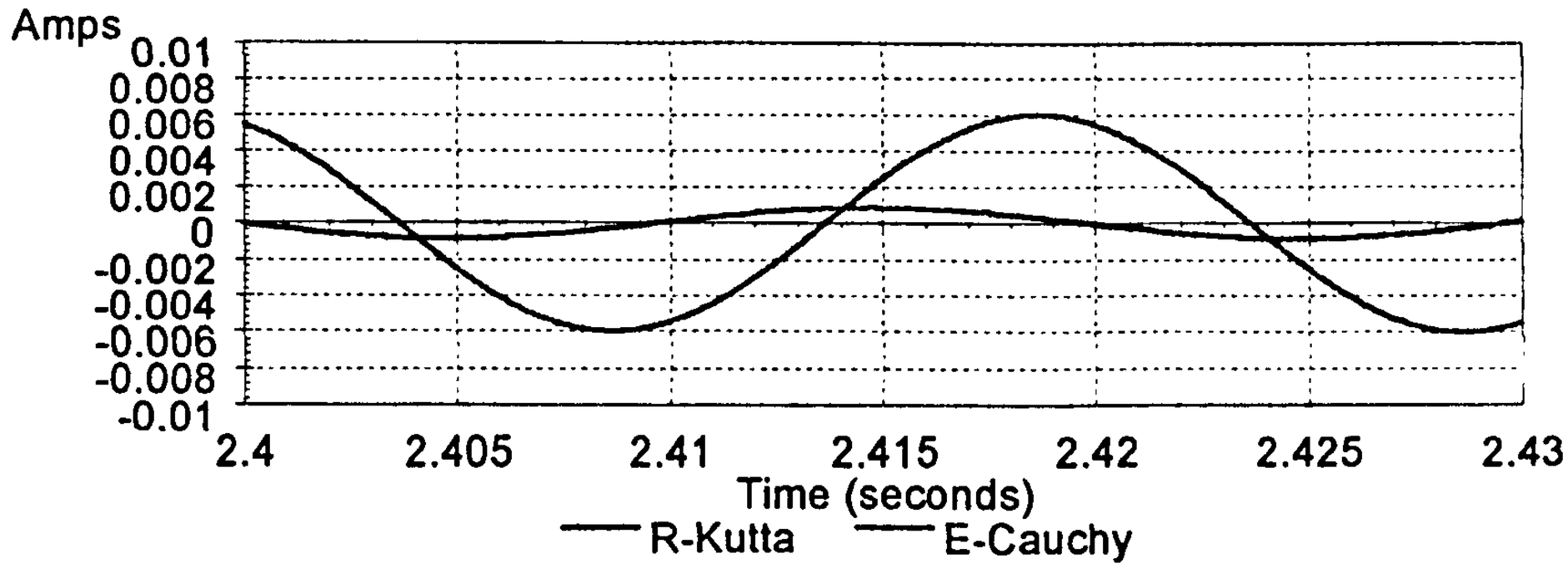


Figure 6.2.5: D-Axis Rotor Current for Runge-Kutta and Euler-Cauchy

6.3 Code Distribution

Three digital signal processors and the PC make up the multiple processor system and so four separate programs are required, one for each processor. The tasks carried out by each processor are as follows:-

D.S.P.'A'.....Drive Controller Software

D.S.P.'B'.....Inverter and Motor Simulation Software

D.S.P.'C'.....Data Acquisition Software

PC.....Graphical User Interface Software

Each of these programs interacts with the others via the communication port interrupt system described in chapter five, this ensures that the execution of the three D.S.P. programs *is* synchronised.

6.3.1 Drive Controller

Two types of drive controller have been implemented, an open loop Volts/Hertz controller and a field orientated (Vector) controller, the operation of both of these controllers was discussed in chapter four. The controller is located in D.S.P.'A' and consists of two interrupt service routines.

The first interrupt is received from the user interface via the PC software. This interrupt is received whenever the user changes parameters which affect the controller, such as gains or demands. On receiving the PC interrupt the processor enters the interrupt service routine which simply consists of reading the new data value sent by the PC, the processor then returns to performing the control algorithms and the changes made by the user are implemented. The second interrupt that the controller D.S.P. receives is that of its own internal timer. This interrupt is set up to operate at the switching frequency of the dive system, i.e. a switching frequency of 10KHz would result in a timer interrupt occurring every 100µs.

On receiving the timer interrupt the control processor enters the interrupt service routine which contains the control algorithm. The processor first of all writes the demanded voltages produced by the previous control cycle to both the actual and the simulated drives, it then reads the values of current from either the actual system or the simulated system. The processor then performs the control algorithm on these currents and produces the new demands which are applied at the start of the next interrupt routine, the processor then exits the interrupt service routine and awaits the next timer interrupt. This control strategy ensures that the instant that the demands are applied is fixed and is not dependent upon the length of the control algorithm, the simulation processor therefore has the complete control period to perform the simulation algorithms.

In order to allow a direct comparison of the actual drive system and the simulated drive system the two systems are run in parallel, with the control processor controlling the actual and the simulated systems. The control processor therefore sends the voltage demands to the actual modulator via the DSPLINK2 connection and also the simulation of the modulator, which is carried out in processor 'B', via communication port 0. A flowchart of the control processor software is shown in the following figure.

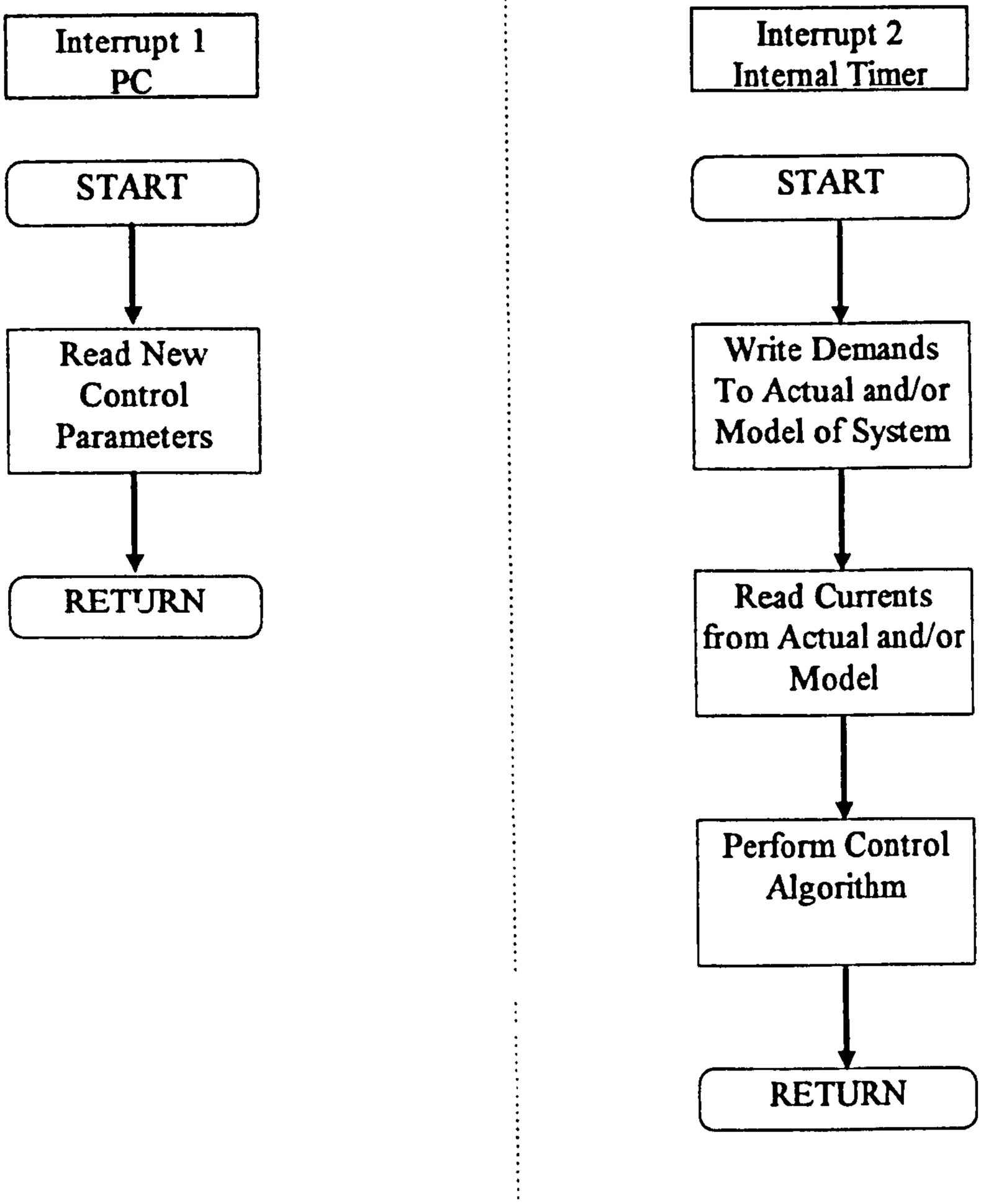


Figure 6.3.1: Control Processor Interrupt Routines

6.3.2 Inverter and Motor Simulation

The software which simulates the inverter and the machine is executed within the second processor, D.S.P.'B'. The simulations performed for the machine and inverter are the same as those described in chapters two and three of this thesis respectively. The code within the processor consists of two interrupt service routines.

The first interrupt, as is the case with the first D.S.P., is received from the user interface via the PC software. Whenever the user makes a change to the motor model parameters such as stator resistance or leakage inductance, or parameters of the inverter such as link capacitance or inductance, the second processor is interrupted. On receiving the PC interrupt the processor enters the interrupt service routine and reads the new parameters via communication port 1. The simulation can then continue and these new parameters take effect.

The second interrupt within the simulation processor is received from communication port 3. This port is connected to the output communication port of the first D.S.P. which is performing the control algorithm. The interrupt is flagged whenever the control D.S.P. writes to the port the new demanded values from the controller. On receiving this interrupt the processor enters the service routine and reads the data that has been transmitted by the control processor, it then performs the simulation of the inverter and machine for these demands. Once the simulation is complete the processor writes the values of machine current and speed to the communication port connected to the control D.S.P. so that they can be read and used by the controller.

During the second interrupt routine the simulation processor sends the third D.S.P., which is carrying out real time data acquisition, data which the user can have displayed via the graphical user interface. This interrupt is dependent upon the communication port being empty. The transfer of the real time data between the third D.S.P. and the PC processor may hold up the third D.S.P. and so to ensure that the

control and simulation processors are not held up, the data transfer between the second and third processors only takes place if the third processor has already cleared the transmission line of old data. A flowchart of the simulation processor software is shown in the figure below.

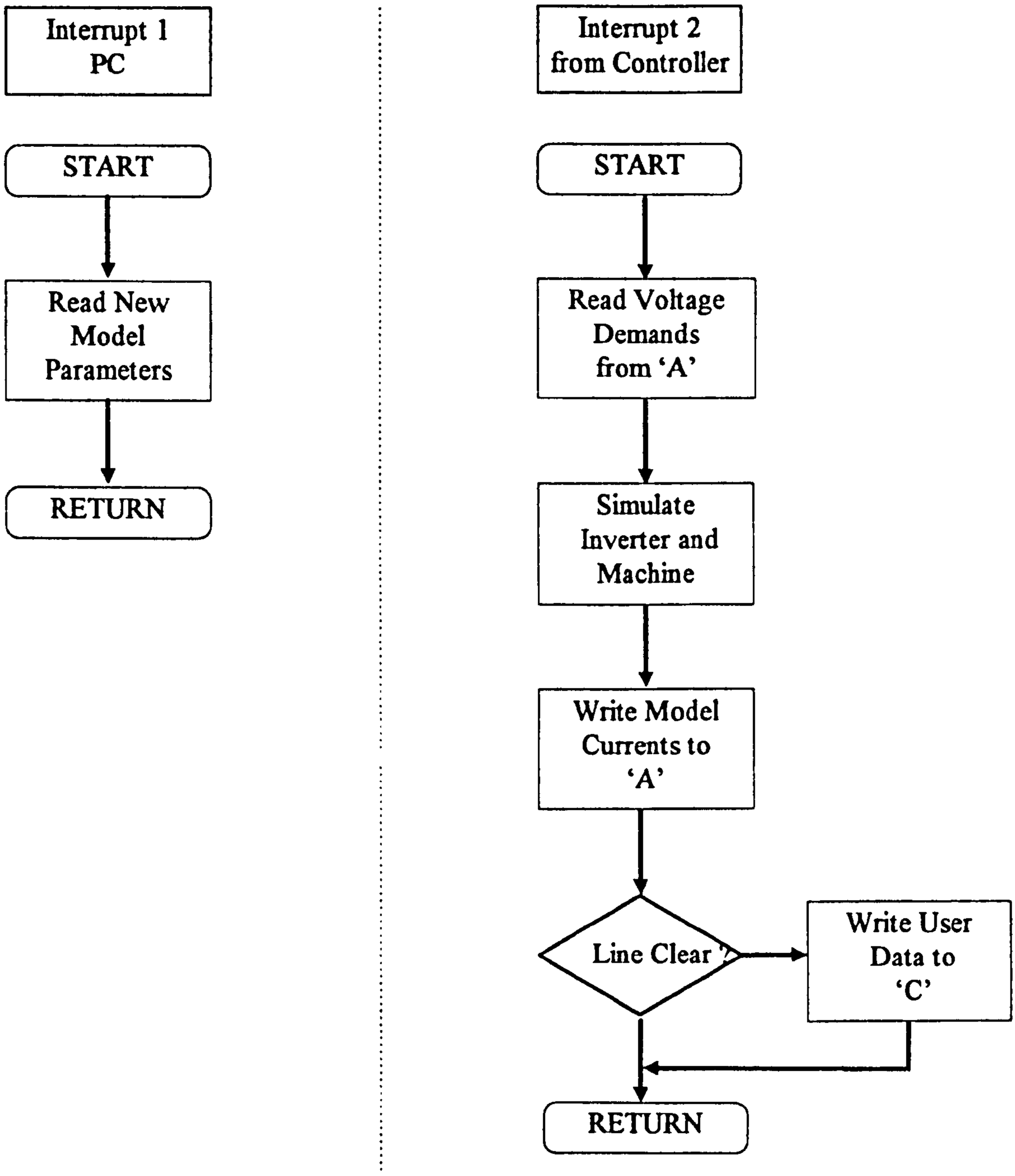


Figure 6.3.2: Simulation Processor Interrupt Routines

6.3.3 Data Acquisition

The third D.S.P. of the multiple processing system located in module site 'C' of the motherboard, is used to record real time data and then transfer this data to the PC processor where it can be displayed, logged etc. The data is stored in four arrays each array represents a single variable within the system and contains 500 real time measurements of that variable. The data can be displayed on a four channel software oscilloscope within the graphical user interface once it has been transferred to the PC.

The data acquisition processor services two interrupt routines, the first as with the previous processors is initiated by the PC user interface software. This interrupt is flagged when the user requires different variables to be stored or the data acquisition to be performed in a different manner. The data can be stored in three possible ways, Roll, Acquire, or Single Shot. The Roll function simply stores the first 500 points and then overwrites these with the next 500 and so on, the Acquire function stores the 500 data points immediately after the users command and does not overwrite them. Finally the Single Shot command works by comparing one of the real time values and not commencing the save routine until this value is equal to a pre-defined level, this allows transients or edges to be captured.

The second interrupt that the data acquisition processor deals with is received from the simulation processor. This interrupt is flagged when the simulation processor writes new data to communication port 3 of the data acquisition processor. This data may be real motor currents, simulated motor currents or any other variable available from the control or simulation processors. Together with variables from the controller and simulator a value of real time at which each data point was measured is also transmitted to the data acquisition processor. The value of real time is obtained from the internal timer of the control processor. A flowchart of the data acquisition software is given in the following figure.

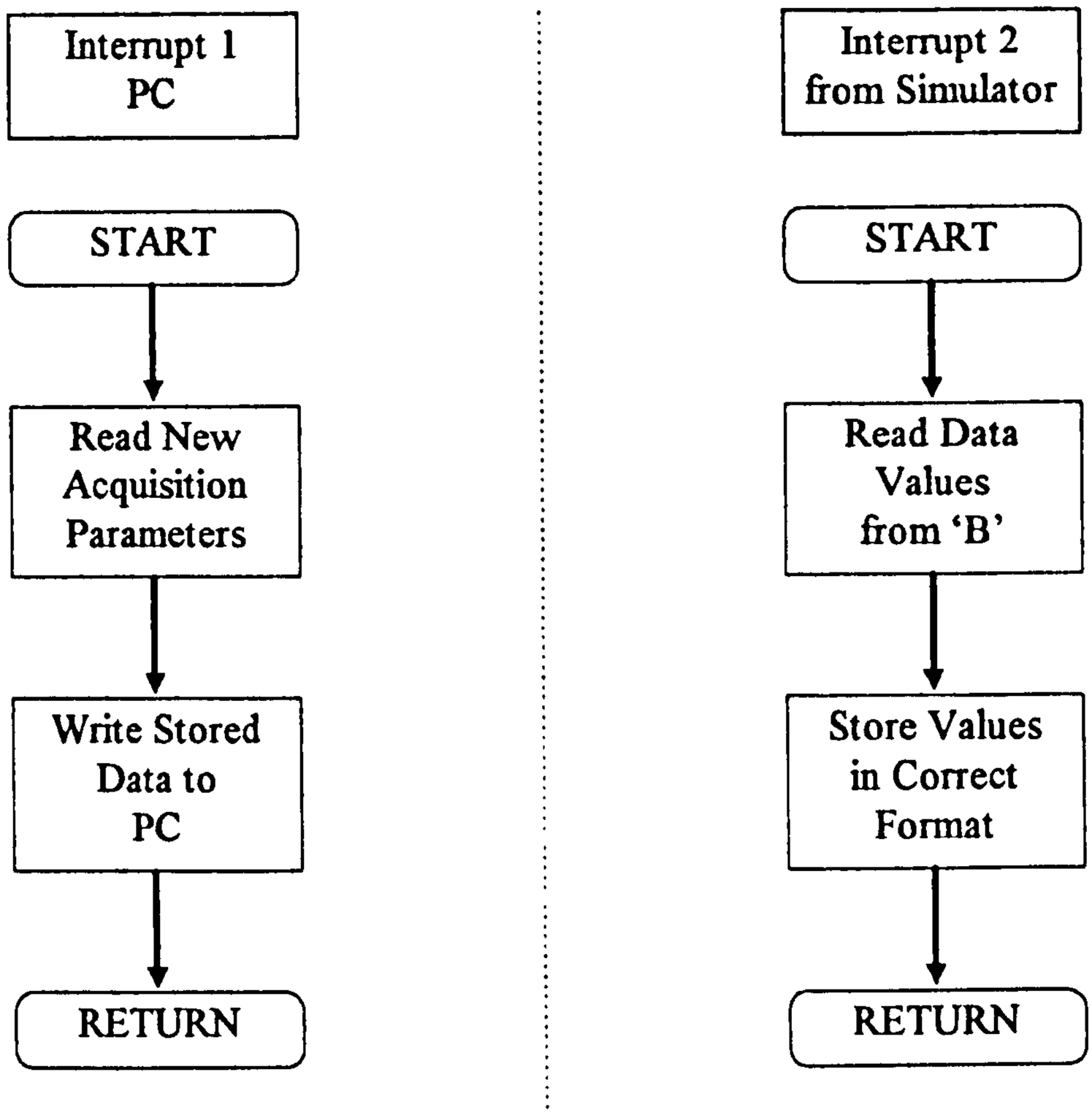


Figure 6.3.3: Data Acquisition Processor Interrupt Routines

6.3.4 Graphical User Interface

The Graphical User Interface (G.U.I.) is the interface provided to allow the operator of the development system to control both the real time simulation of the drive system and also the actual drive system itself. The G.U.I. enables the operator to send demands to both simulation and actual drives, modify control parameters or simulation parameters and interrogate the drive system by acquiring real time data from both systems. The main objective of the G.U.I. must therefore be to provide a clear, uncomplicated yet flexible environment.

The G.U.I. is executed by the processor of the host PC which contains the multiple digital signal processing system. It consists of a program written in the high level language 'C' which allows the user to address the resident quad board and download the D.S.P. programs to each of the individual digital signal processors. Parameters can then be sent to the relevant D.S.P.s and captured real time data can be transferred back to the PC to allow the operator to view real and simulated variables and store this data for importing into other common software applications for instance word processors or spreadsheets.

In order to provide an interface which is familiar to most operators who have experience working with personal computers the program executes in the Windows operating system environment. This environment is made up of windows, menu bars, dialogue boxes etc. and is the most common operating system used by personal computers. The familiar interface allows the operator to quickly change parameters by entering values into dialogue boxes, acquire data via a software oscilloscope which offers similar features to a real oscilloscope and even control all functions of the PWM ASIC resident in the real inverter.

The following figure shows the simulation and drive control desktop which the user interacts with.

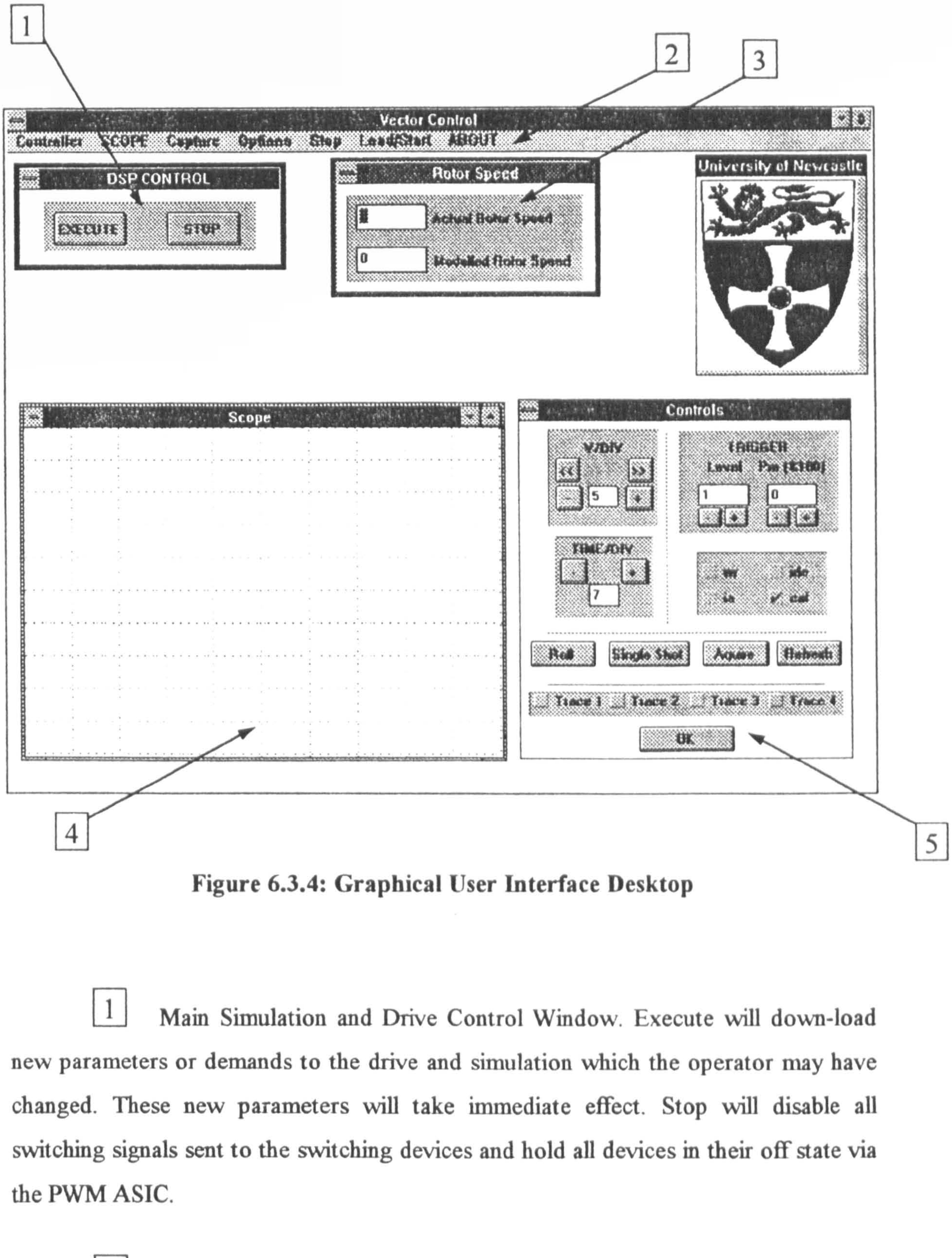


Figure 6.3.4: Graphical User Interface Desktop

1 Main Simulation and Drive Control Window. Execute will down-load new parameters or demands to the drive and simulation which the operator may have changed. These new parameters will take immediate effect. Stop will disable all switching signals sent to the switching devices and hold all devices in their off state via the PWM ASIC.

2 Menu Bar, allows user to access parameter input dialogue boxes, controller gains, demands etc.

3 Rotor Angular Velocity Display Window, provides the user with information regarding the actual rotor velocity measured from an encoder, and also the rotor speed of the simulated machine.

4 Software Oscilloscope Output Window, provides the graphical output of the oscilloscope. Allows the operator to observe real time data from both the actual and simulated drive systems captured by the data acquisition D.S.P.

5 Software Oscilloscope Control Window, allows the operator to control the method in which the data is acquired by the D.S.P. Functions which are common on real oscilloscopes have been provided such as single shot. variable time base etc.

The following two figures give examples of how the dialogue boxes, which appear once the operator has selected a particular menu item. are used to input variables to the drive system. The two examples given are the input demands for a Vector Controlled drive and a Volts/Hertz Controlled drive.

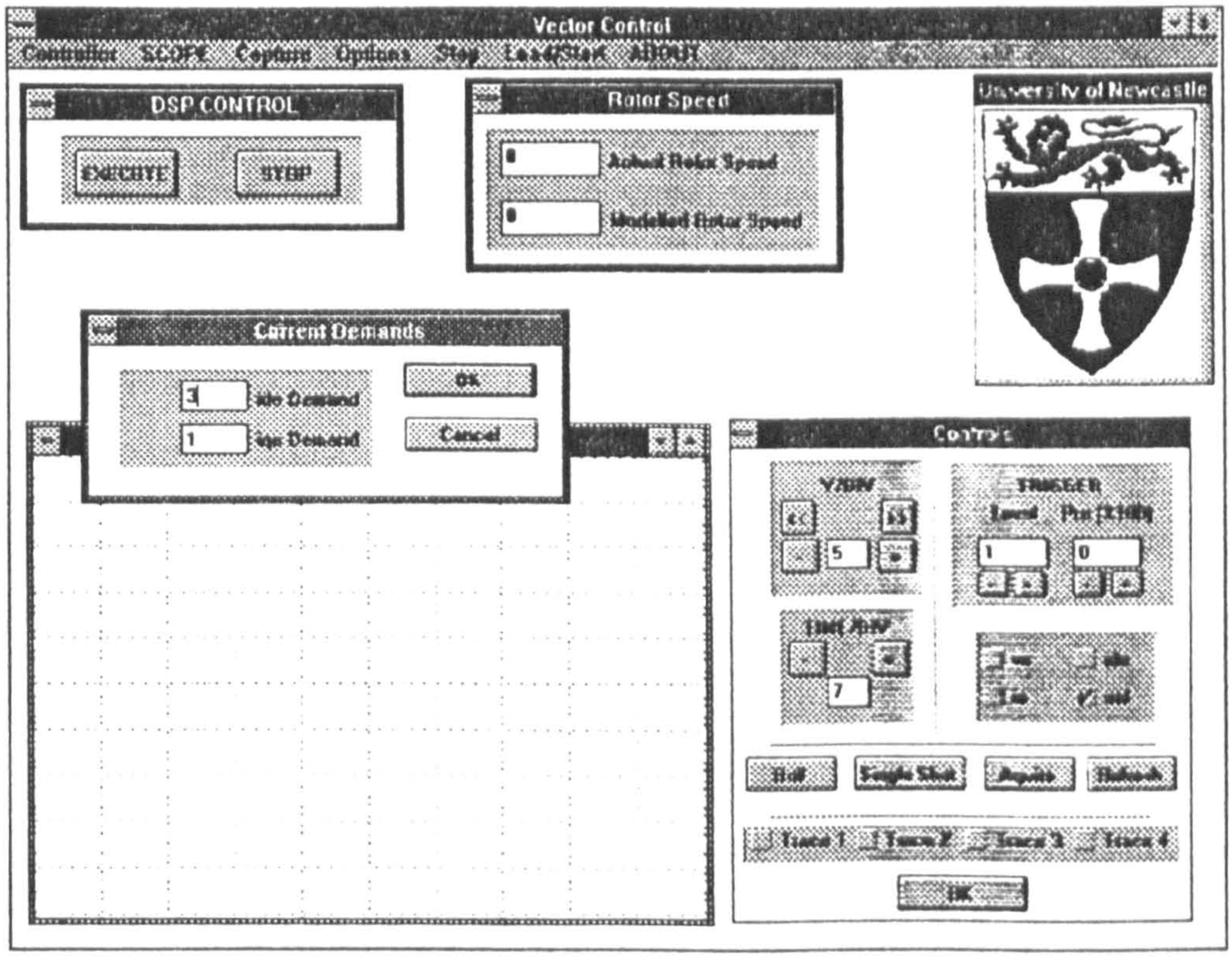


Figure 6.3.5: Current Demand Input for Vector Controlled Drive

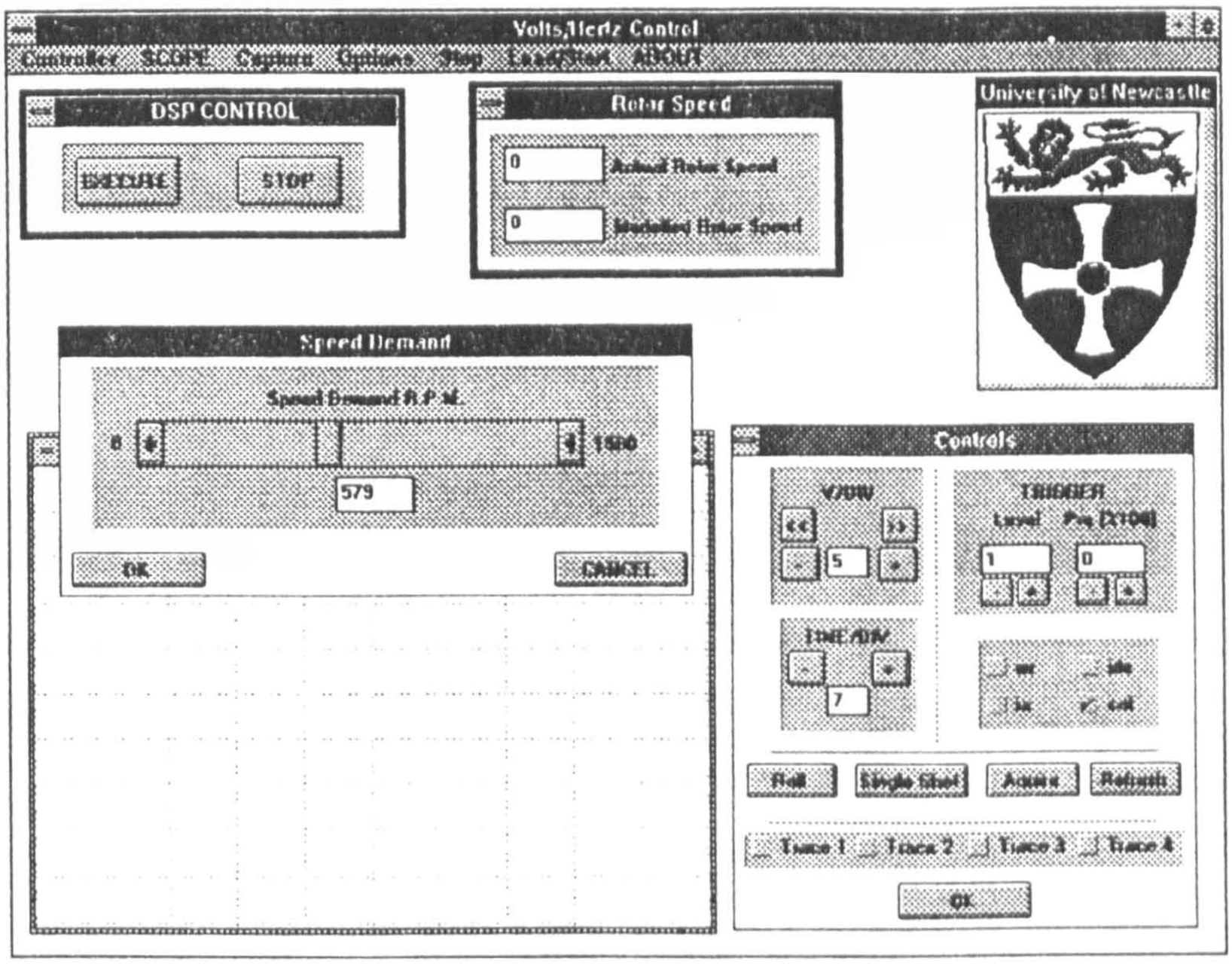


Figure 6.3.6: Speed Demand Input for Volts/Hertz Controlled Drive

The inputs to the Vector drive are simply the torque component i_{qs}^{e} and the flux component i_{ds}^{e}. The input to the Volts/Hertz drive is a slider bar which the operator can use to select the desired speed. Both demands take effect once the operator has selected OK on the dialogue box and then selected EXECUTE from the main control window.

6.4 Timings of Control and Simulation Code

The execution time of each part of the control and simulation software must be known to determine how much processing time is being used and how much is available. In order to accurately time each part of the software a data line of the DSPLINK of D.S.P.'A' was used to flag the beginning and end of each algorithm. The switching frequency of the drive was chosen to be 6.5KHz which equates to a control period of 153.8μs. All control software executed in D.S.P.'A' must therefore be carried out within this period, likewise all simulation software carried out in the second processor D.S.P.'B' must also be executed within this 153.8μs interval.

Figure 6.4.1 shows that both of the control algorithms can be computed well within the limit imposed by the chosen switching frequency. As well as the actual control algorithm, the processor must also read actual variables i.e. the three machine currents and the rotor angular velocity, this takes 20μs which gives a total controller time of 50μs for the Vector Controller and 41μs for the Volts/Hertz Controller.

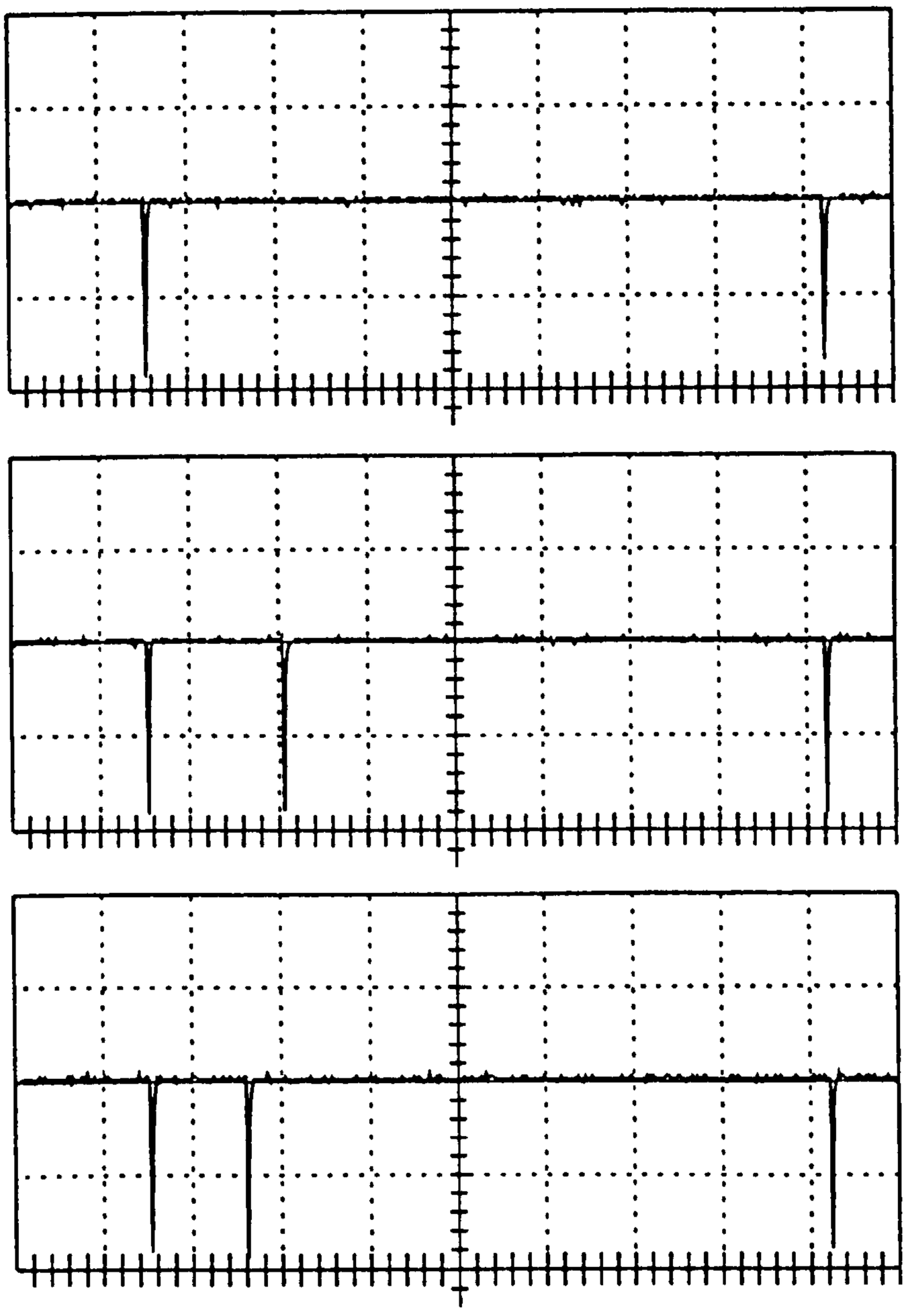


Figure 6.4.1: Control Algorithm Timing

Figure 6.4.2 shows three timing pulses, the first is the time at which the controller starts to send data to the simulation, the second is the time when this ends and the controller waits to read data from the simulation, and the third is the time when the controller has received the data from the simulation. The difference between the second two pulses is therefore the time that the simulation processor has taken to perform its algorithms.

The first graph shows the total control period 'A', the time to transfer data from the controller to the simulation 'B' and the time for the simulation to read this data and to transfer data from the simulation back to the controller 'C'. The remainder of the figures show the additional time taken by the simulation processor as more simulation routines are added to it.

Period D - Simulation sends data to data acquisition processor

Period E - Simulation includes Modulator

Period F - Simulation includes Motor Model

Period G - Simulation includes Mechanical Model

Period H - Simulation includes DC Link

Period I - Simulation includes Two to Three Phase Conversion

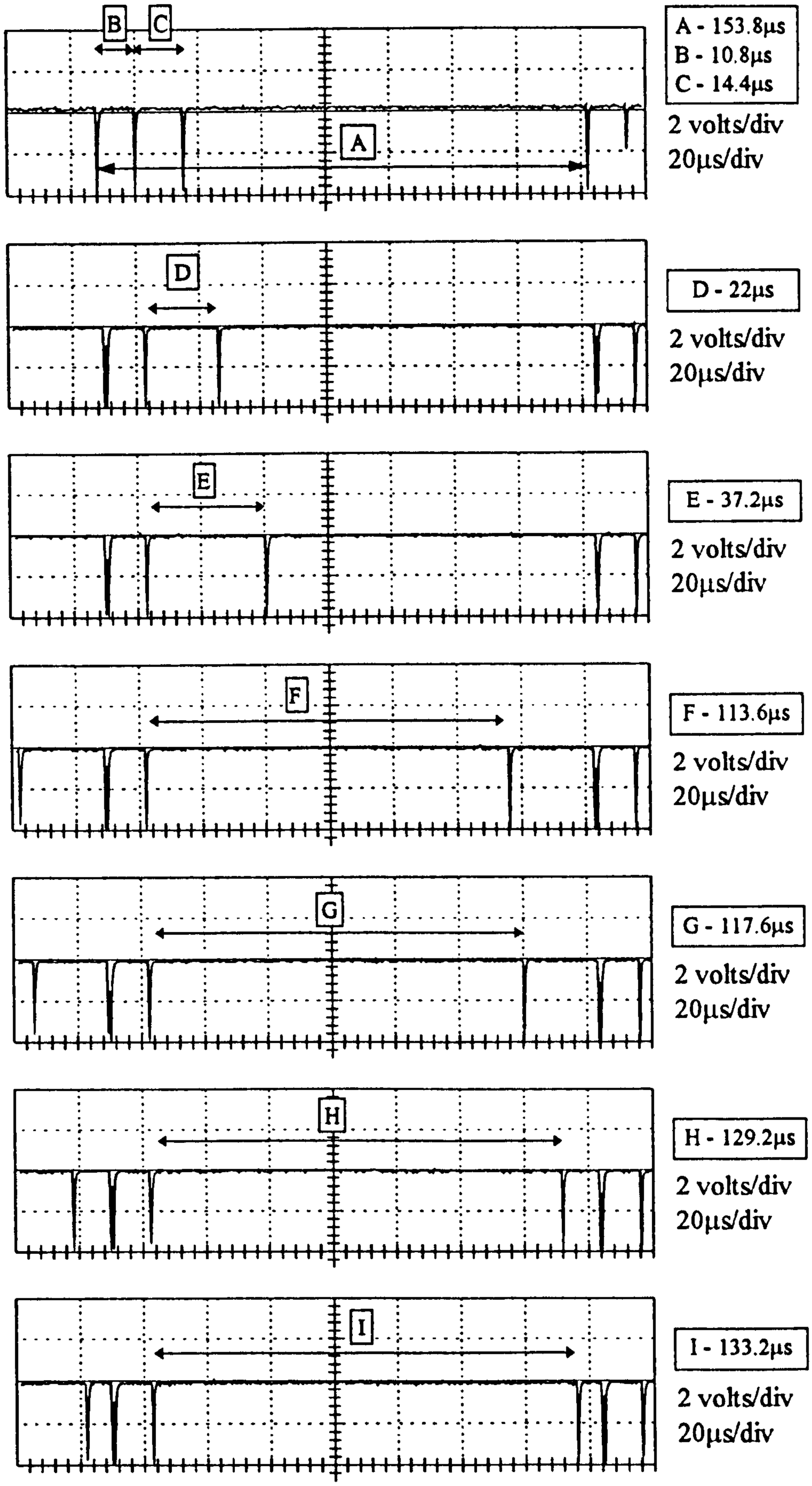


Figure 6.4.2: Simulation Algorithm Timing

Table 6.4.1 shows the extra tasks that were added to the simulation processor and the subsequent additional time taken.

DSP	Period	Algorithm	Time	Process Time
A	A	Total Control Period	153.4μs	
A	B	Controller to Simulation Transfer	10.8μs	
B	C	Simulation Read and Write to Controller	14.4μs	14.4μs
B	D	Simulation to Data Acquisition Transfer	22μs	7.6μs
B	E	Modulator Simulation	37.2μs	15.2μs
B	F	Motor Model Electrical Equations	113.6μs	76.4μs
B	G	Motor Model Mechanical Equation	117.6μs	4μs
B	H	DC Link Simulation	129.2μs	11.6μs
B	I	Two phase to Three phase Conversion	133.2μs	4μs
		Total		133.2μs

Table 6.4.1: Simulation Algorithm Timing Table

The timing flowchart, figure 6.4.3, shows the algorithms carried out in each of the three processors and the interaction between the processors. Alongside each algorithm is the execution time of the process. It can be seen from the timing results that the simulation algorithms together with the data transfer routines take 133.2μs. This means that the second processor carrying out the simulation is using 86.6% of the available processing time. This is within the control cycle time of 153.8μs and therefore real-time simulation is possible.

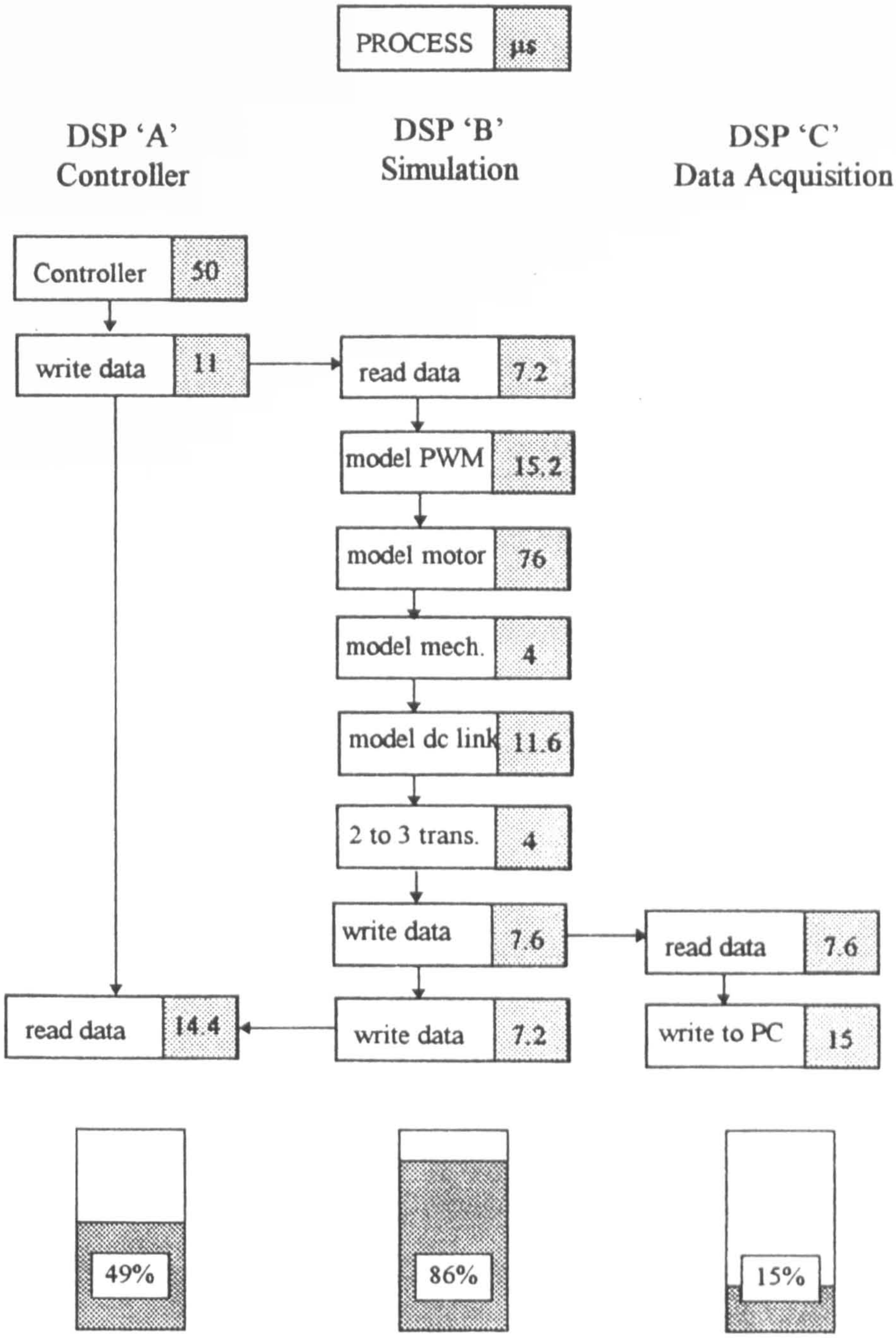


Figure 6.4.3: Algorithm Timing Flowchart

6.5 Performance of Real Time Simulation

The performance of the real time simulation of the drive system can be compared directly to the performance of the actual drive system by running the two systems in parallel. One of the crucial advantages of using a real time simulation is that the controller used for the simulation can be the actual controller of the real drive system running in the target processor. In this laboratory demonstration and validation this is precisely what is happening although in a real application there will be little point in running the simulation and the real drive simultaneously. The overall block diagram system used to validate the real time simulation is shown below.

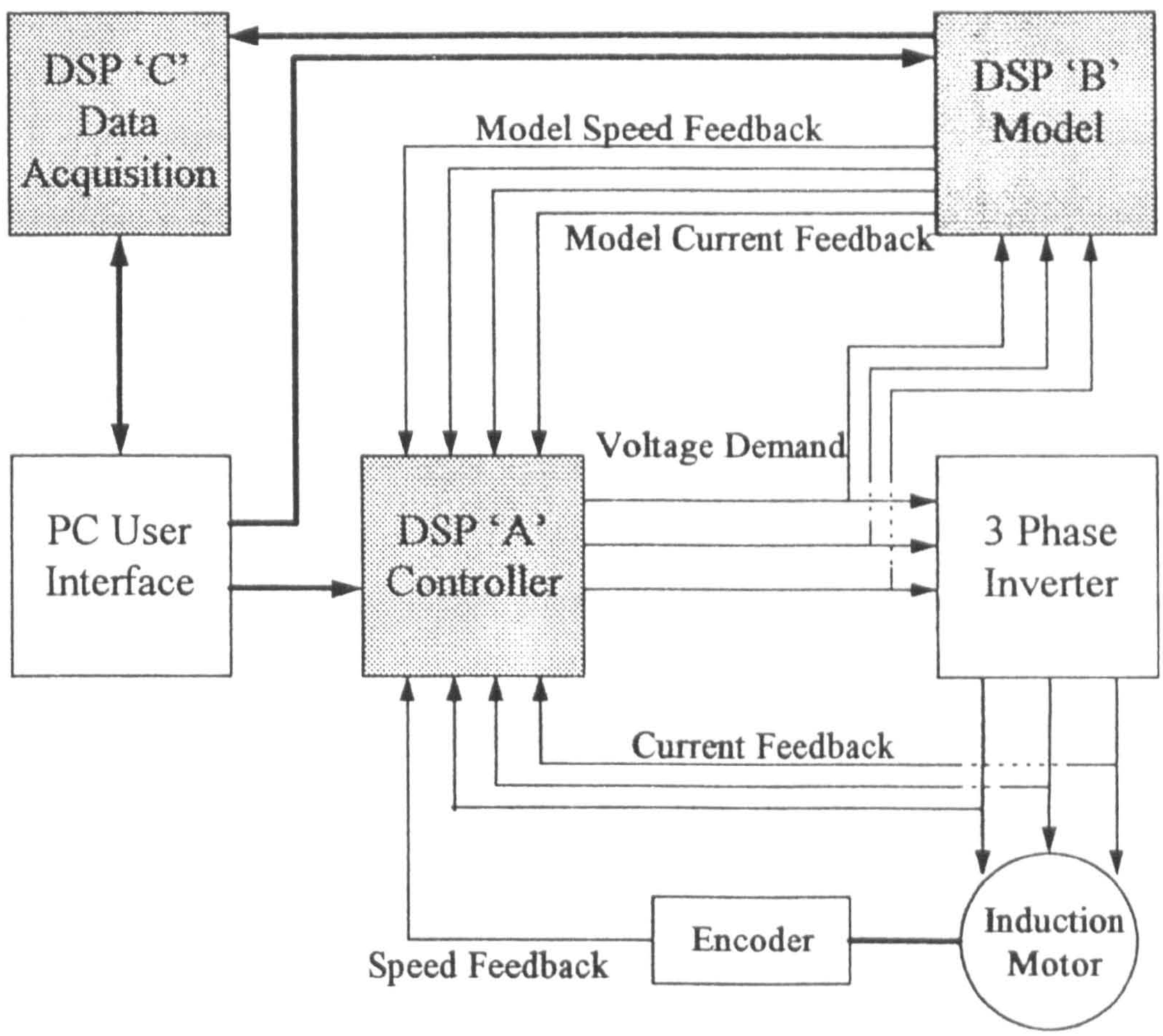


Figure 6.5.1: Parallel Simulation and Actual Drive System

The previous figure shows how the controller D.S.P.'A' sends the three phase voltage demands to both the actual inverter and also the simulation of the inverter carried out in D.S.P.'B'. The actual inverter feeds the real motor and the simulation of the inverter feeds the simulation of the motor which is also carried out in D.S.P.'B'.

The controller has two possible sources of feedback, the actual values from the real motor and the simulated values from the simulation of the motor. The controller requires the currents to be fed back as part of the closed loop Vector Control scheme and the rotor angular velocity for use in the flux model which is required for the reference frame transformations within the Vector Controller. If the system is being run as a simulation only, i.e. there is no real inverter or motor to be controlled, then the controller acts on the variables fed back from the simulation. If, however, the actual inverter and motor is connected then the controller will act on the variables fed back from the actual motor and the simulation simply runs as a slave in parallel with the actual drive system.

6.5.1 Volts/Hertz Control Results

The following figures show the results of running the real time simulation in parallel with the actual drive when the actual drive is controlled by an open loop Volts/Hertz Controller. Figures 6.5.2 and 6.5.3 show the steady state stator current of the simulated motor and the actual motor for constant frequency demands to the inverter. Figure 6.5.2 shows the current for no-load with a deadtime of 2μs, the deadtime with which the drive is normally operated. Figure 6.5.3 shows the current for an increased deadtime of 5μs, this figure is included to show the current distortion which occurs as a result of the deadtime period. Both show good correlation between actual and simulated. The experiment with dead time is a good example of the power of the simulator in the study of the performance of the drive.

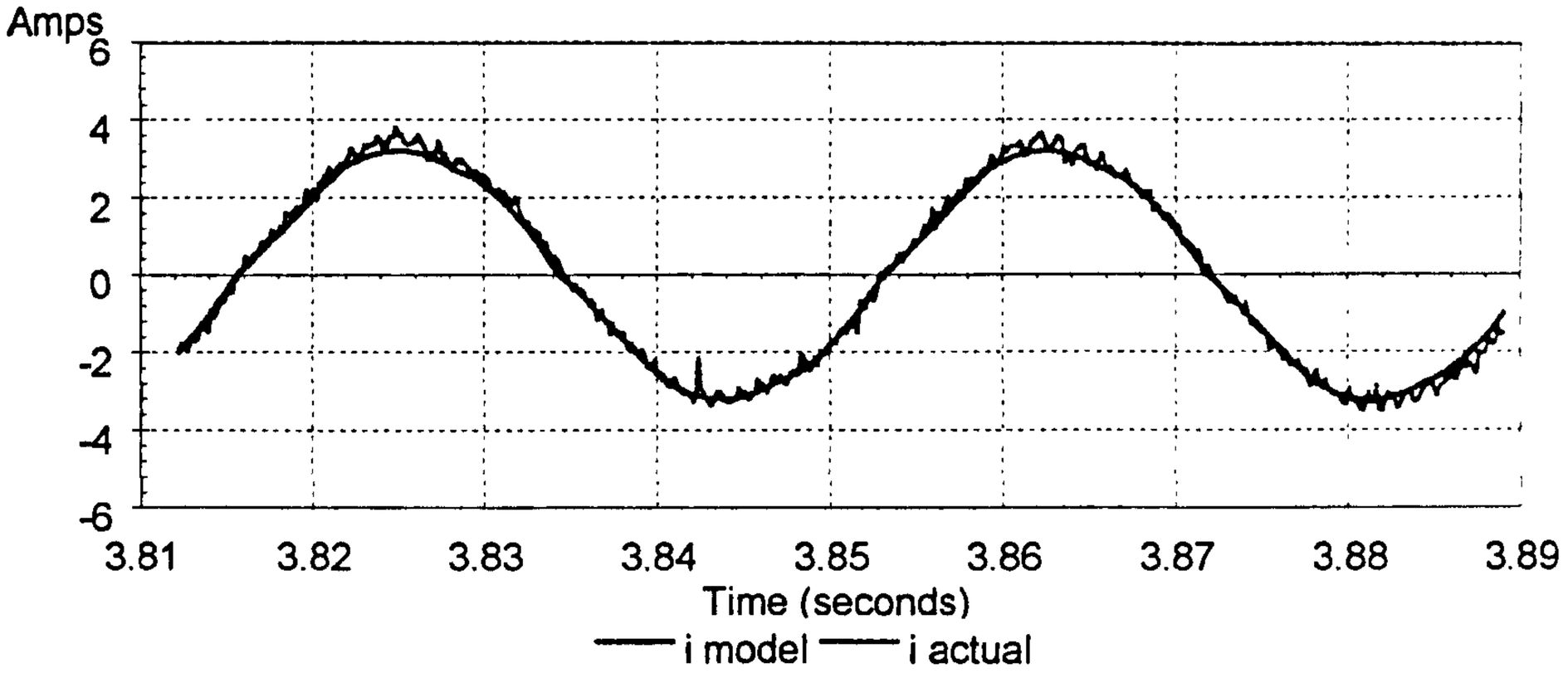


Figure 6.5.2: Steady State Volts/Hertz Control (Deadtime=2μs)

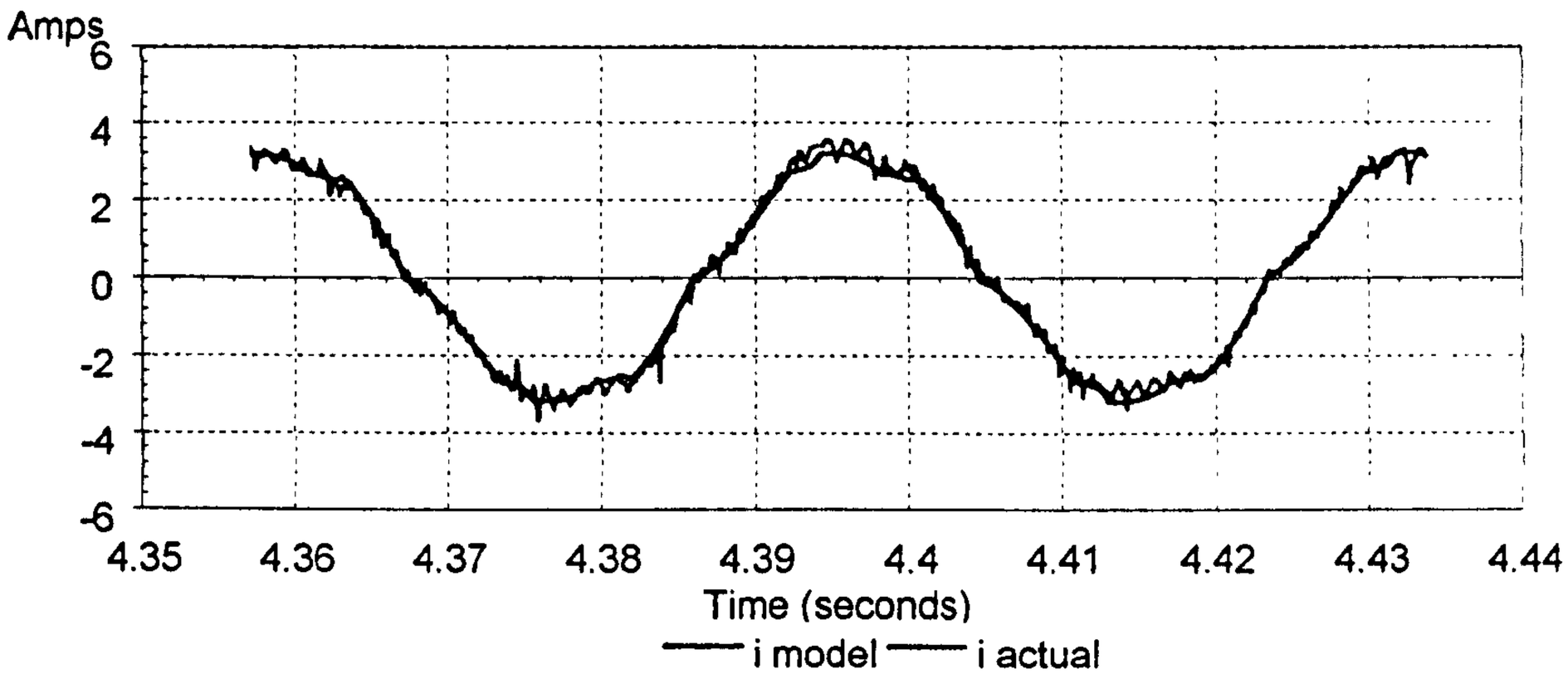


Figure 6.5.3: Steady State Volts/Hertz Control (Deadtime=5μs)

Figures 6.5.4 to 6.5.7 show the transient behaviour of the real-time simulation when controlled by the open loop Volts/Hertz Controller. The transient applied was a step in frequency demand between 13.33 Hz (400 r.p.m) and 26.66 Hz (800 r.p.m) up or down.

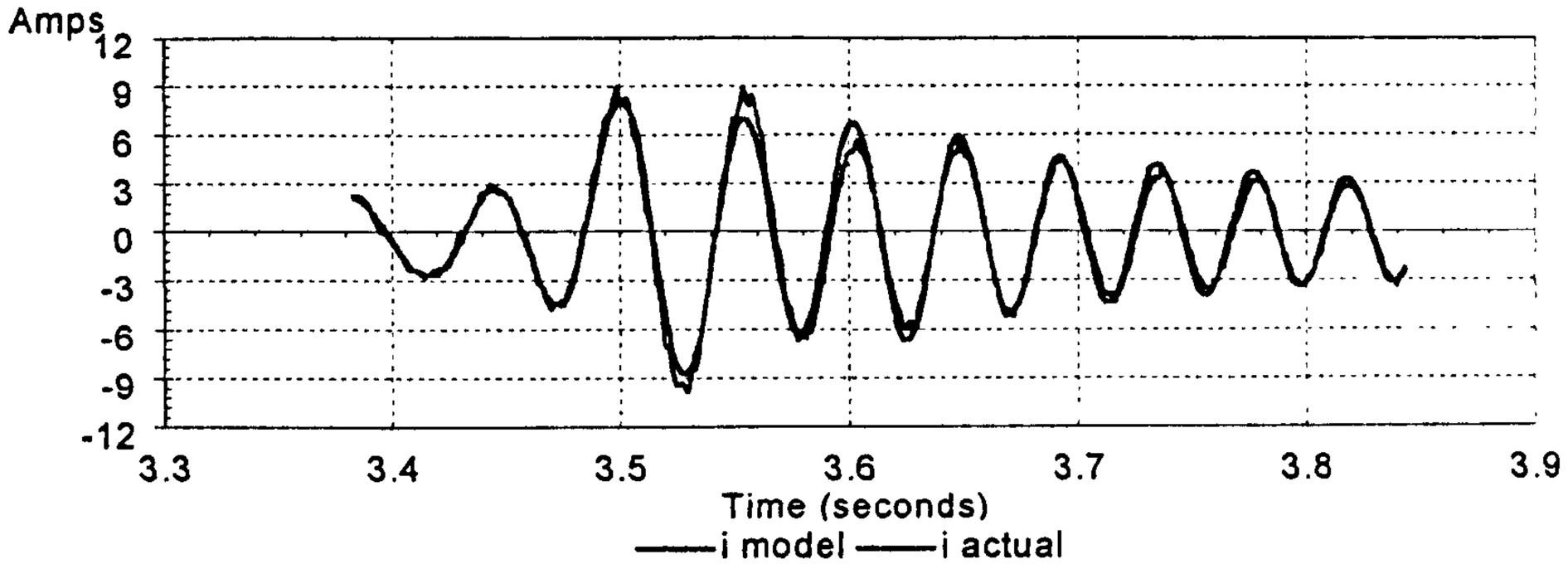


Figure 6.5.4: 13.33Hz to 26.66Hz Transient Volts/Hertz Control, Phase Current

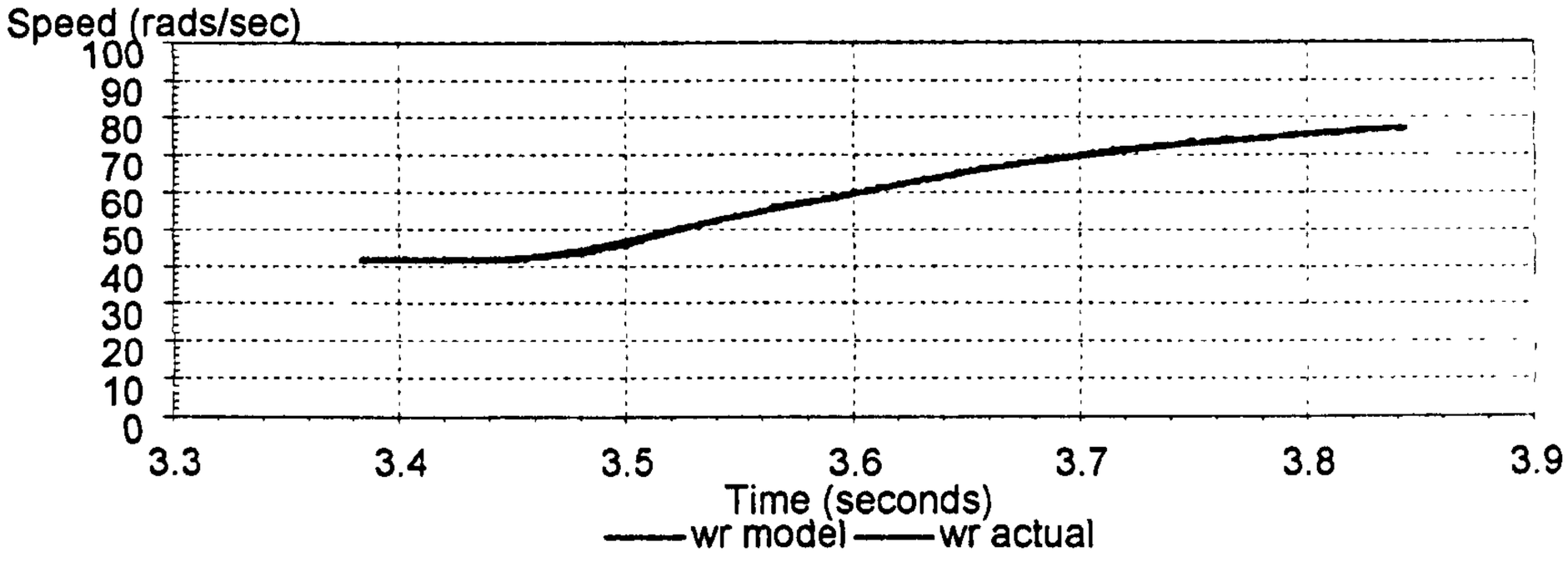


Figure 6.5.5: 13.33Hz to 26.66Hz Transient Volts/Hertz Control, Rotor Speed

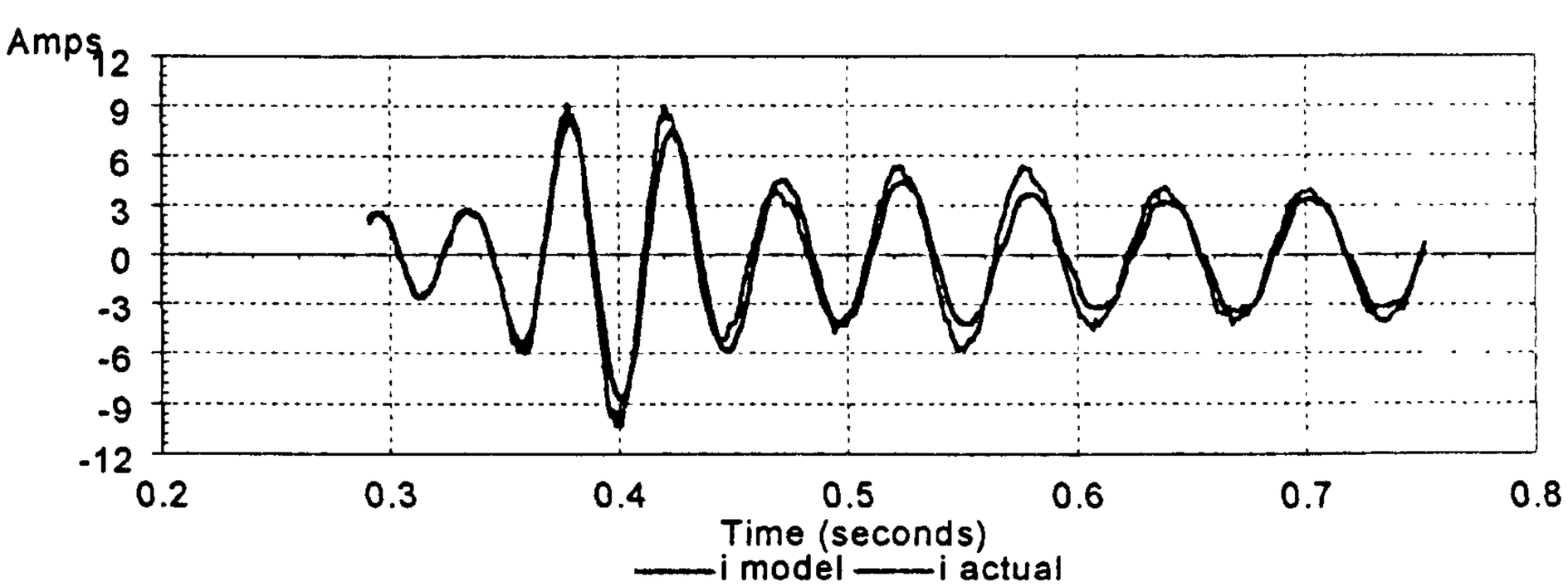


Figure 6.5.6: 26.66Hz to 13.33Hz Transient Volts/Hertz Control, Phase Current

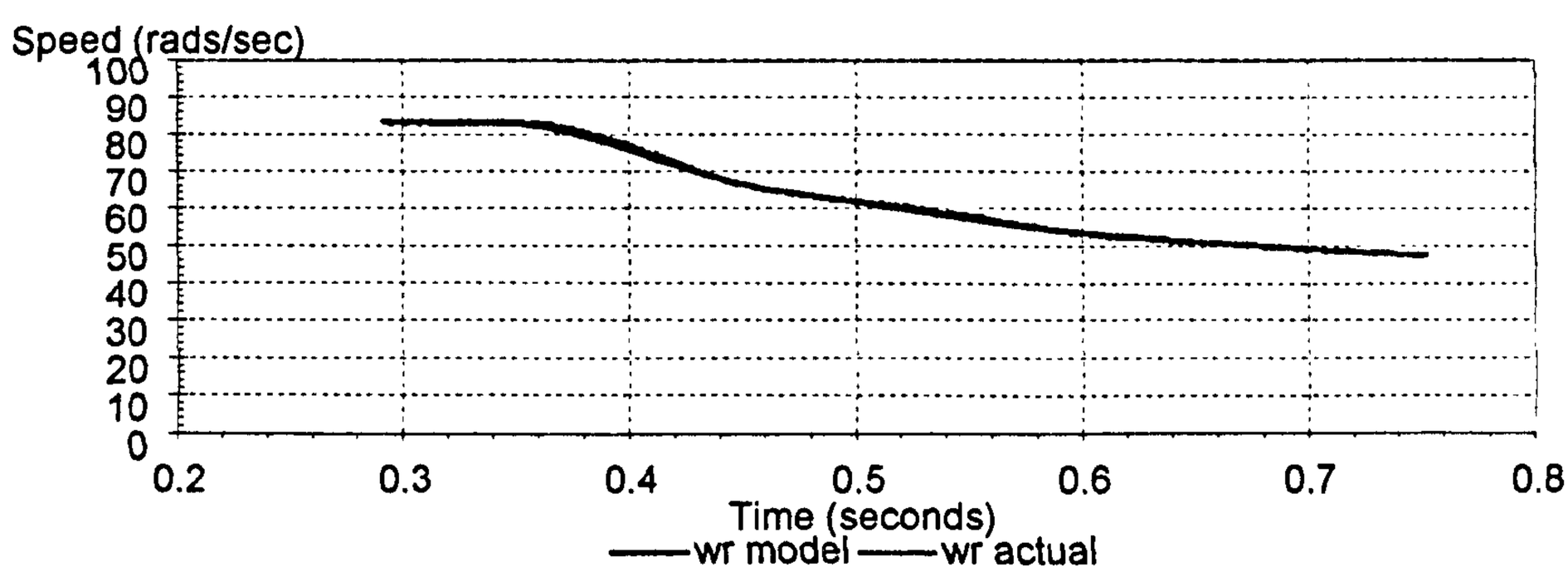


Figure 6.5.7: 26.66Hz to 13.33Hz Transient Volts/Hertz Control, Rotor Speed

The previous results show that the real-time simulated current matches the actual current well during both transients, the speed reduction transient, however, appears to exhibit more errors than the speed increase transient. In order to investigate this discrepancy further the actual measured rotor speed was fed to the model, thus eliminating the mechanical model and any errors that are introduced because of it. Figures 6.5.8 and 6.5.9 are for the same transient but with the motor model using the actual rotor angular velocity.

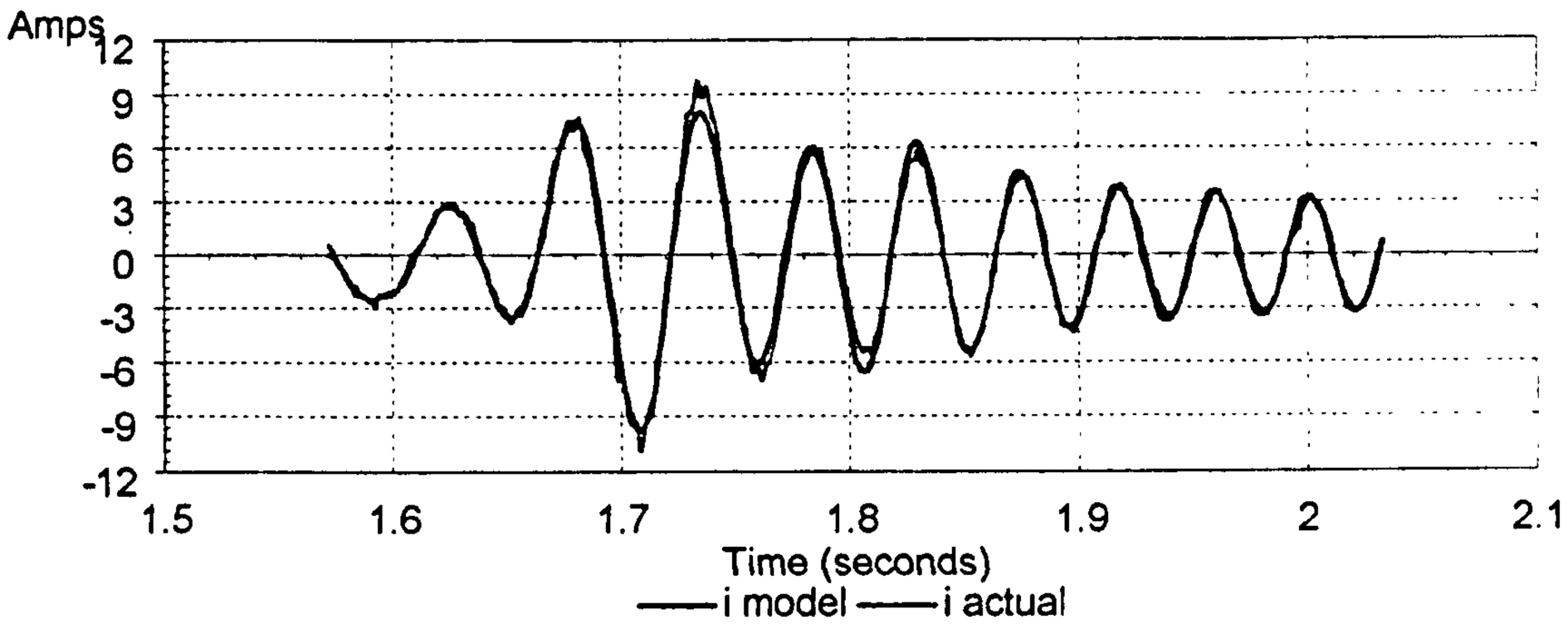


Figure 6.5.8: 13.33Hz to 26.66Hz Transient, Phase Current Actual Speed Used

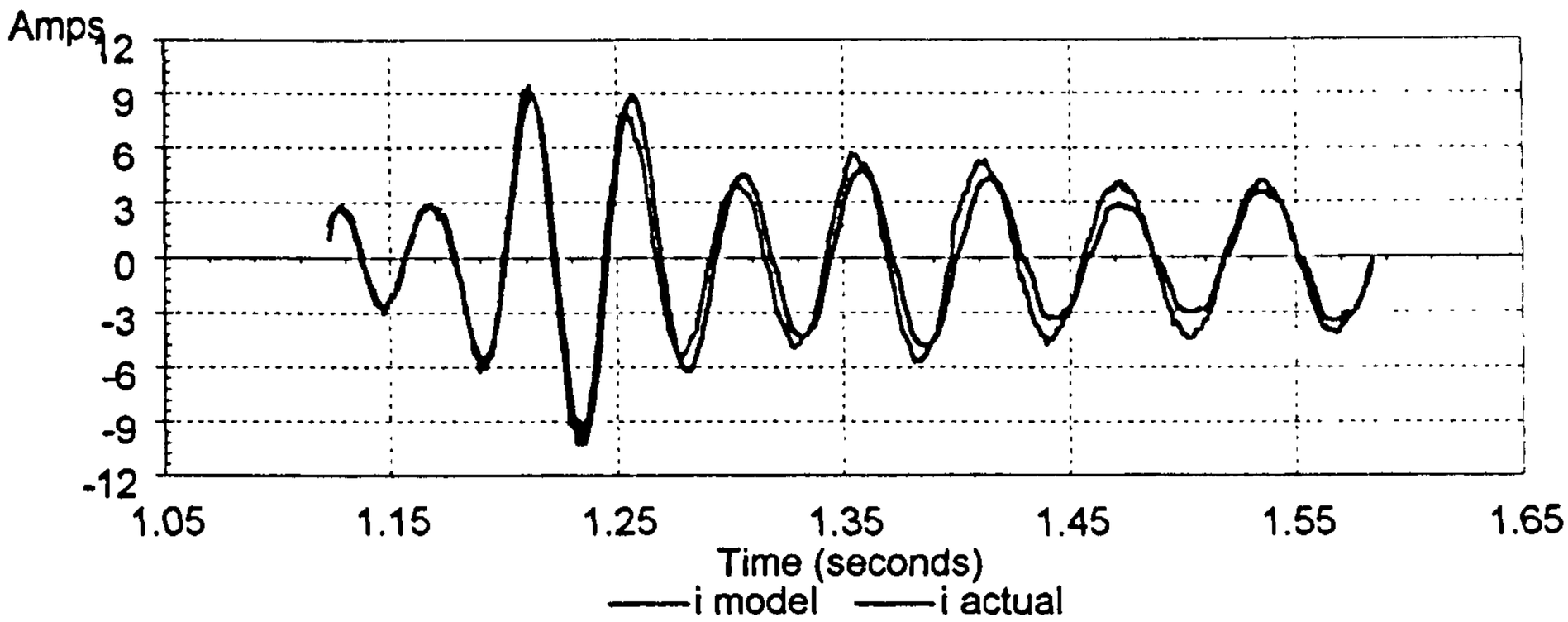


Figure 6.5.9: 26.66Hz to 13.33Hz Transient, Phase Current Actual Speed Used

The figures above show that very little improvement has occurred when using the actual speed signal, the speed reduction transient still exhibits more errors than the speed increase transient. Chapter four of this thesis compared a non-real time simulation to the actual drive, it showed how the flux in the machine increases during

the speed reduction transient. This increase in the flux leads to a degree of saturation in the machine, this saturation results in a variation in the magnetising inductance of the machine. The motor model uses fixed parameters, this will lead to differences between the simulated and actual currents.

In order to try to improve the simulation an attempt is made to simulate the non-linearity of the actual machine within the processing time available. The timing results indicated that the D.S.P. carrying out the simulation has just 20μs processing remaining, this means that it is impossible to implement a detailed saturation model of the induction machine. There have been many attempts to define saturation models, examples are [*Brown, Kovacs, Vas*, 1983] [*Vas, Alakula, Brown, Hallenius*, 1988],[*Melkebeek*, 1983],[*Melkebeek, Novotny*, 1983] [*Boldea, Nasar*, 1987]. Detailed saturation models tend to use the magnetisation curve and also the dynamic inductance which is the first derivative of the magnetisation curve. These models contain a greater number of coefficients of greater complexity than the fixed parameter model due to inclusion of cross-saturation effects. These cross-saturation effects are described in detail in [*Vas, Hallenius, Brown*, 1986], although the existence of cross-saturation was contested by Kovacs [*Kovacs*, 1984]. The complexity of these saturation models means that they cannot be implemented in real-time in the present system, however, future expansion of the system may provide the extra processing power required.

A method of accounting for saturation of the main flux path which can be implemented in real-time within the restrictions of the available processing time is to simply vary the magnetisation inductance as a function of the magnetising current. This method is similar to that described by [*Onbilgin, Senlik*, 1995] , the magnetising current is calculated each control cycle and a new value of magnetisation inductance is obtained from the magnetisation curve for this value of current.

The following results show the same transient with the model using the actual speed, but with a variable value of magnetisation inductance obtained from the magnetisation curve.

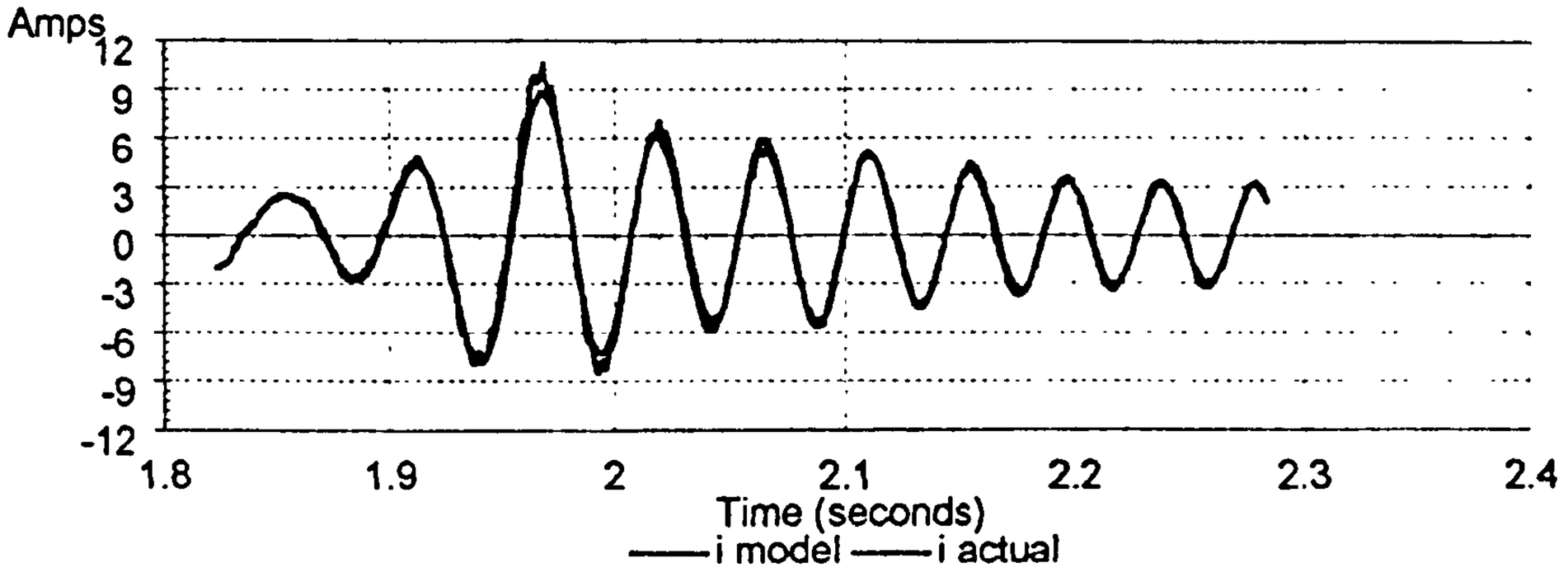


Figure 6.5.10: 13.33Hz to 26.66Hz, Actual Speed and Varying Lm Used

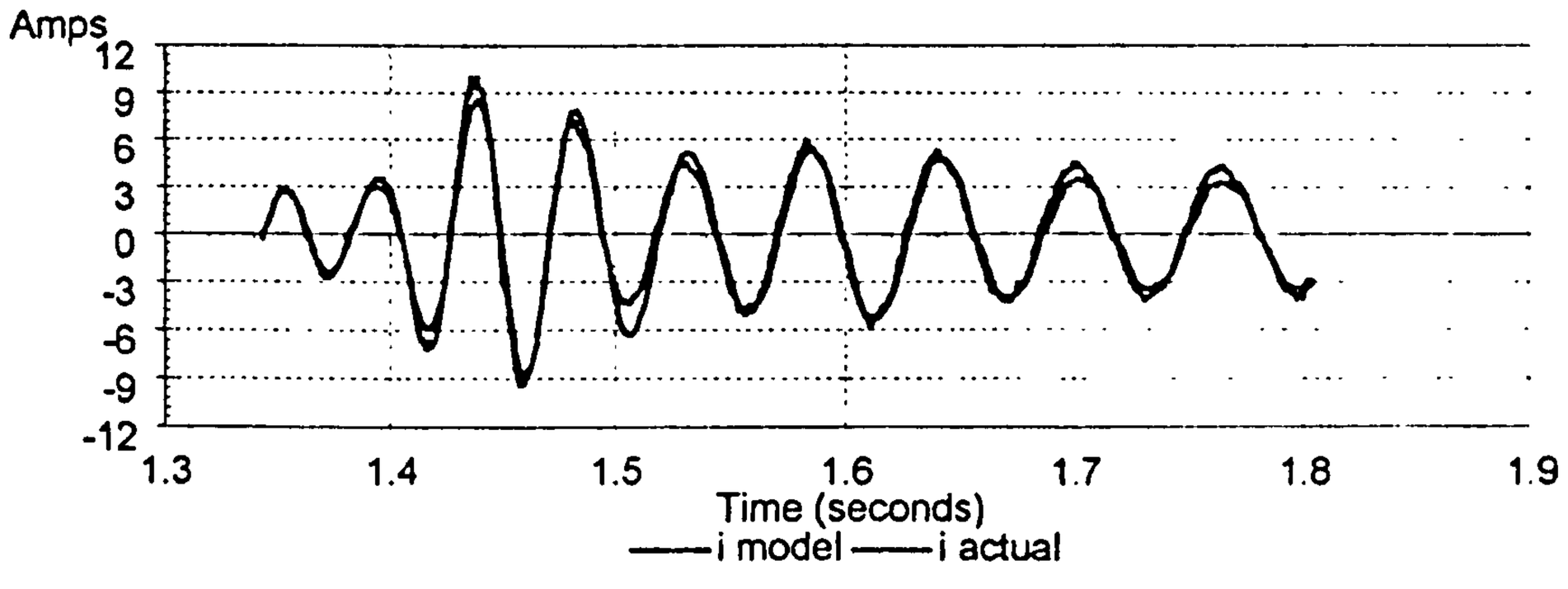


Figure 6.5.11: 26.66Hz to 13.33Hz, Actual Speed and Varying Lm Used

The above results show an improvement over the results of the same transient with a constant value of inductance, the currents, however still exhibit some errors which are likely to be due to saturation. Figures 6.5.12 to 6.5.15 are for the same transient but with the model of the speed included as well as the simple saturation model. They, as expected, are similar to the previous results which had the model speed equal to the actual speed measurement since the modelled speed is in good agreement with the actual speed.

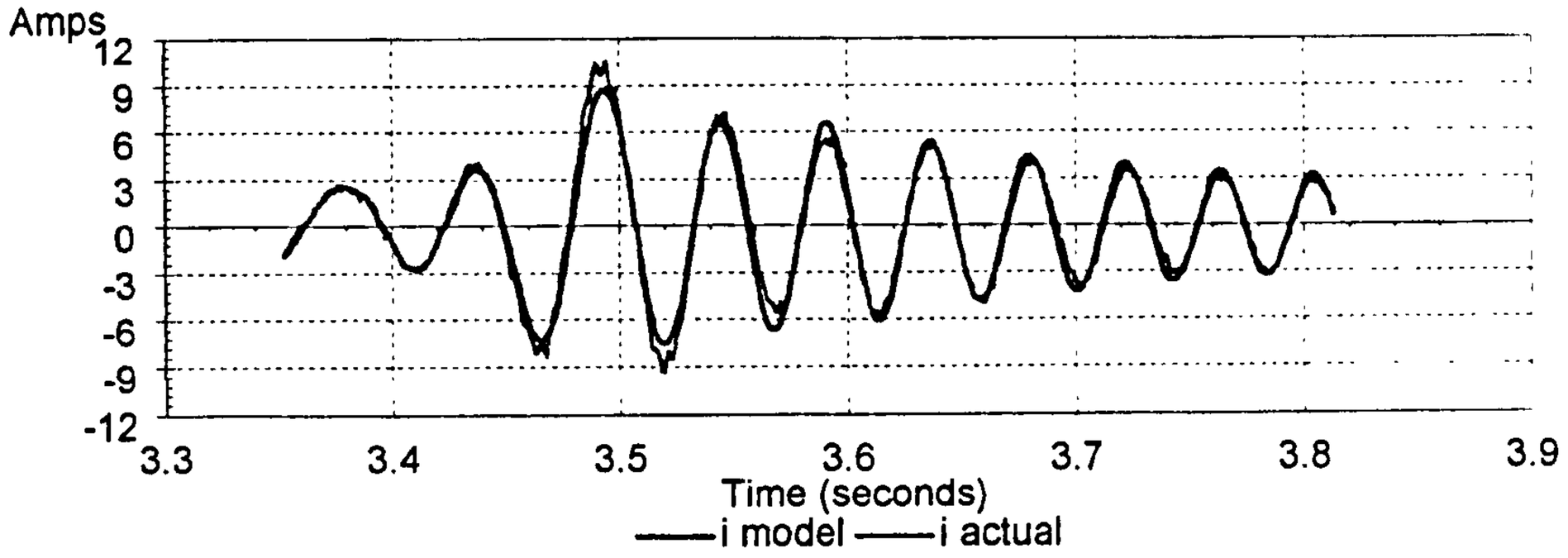


Figure 6.5.12: 13.33Hz to 26.66Hz, Phase Current, Speed Modelled, Varying Lm

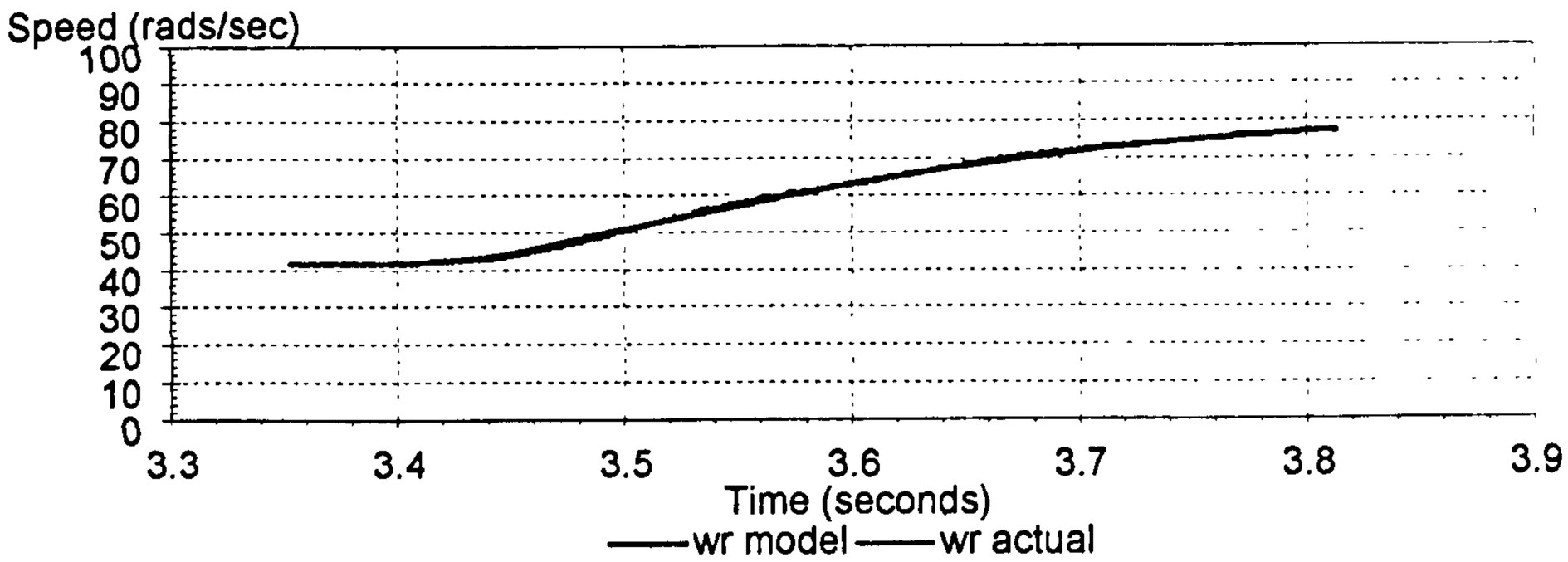


Figure 6.5.13: 13.33Hz to 26.66Hz, Model/Actual Rotor Speed, Varying Lm

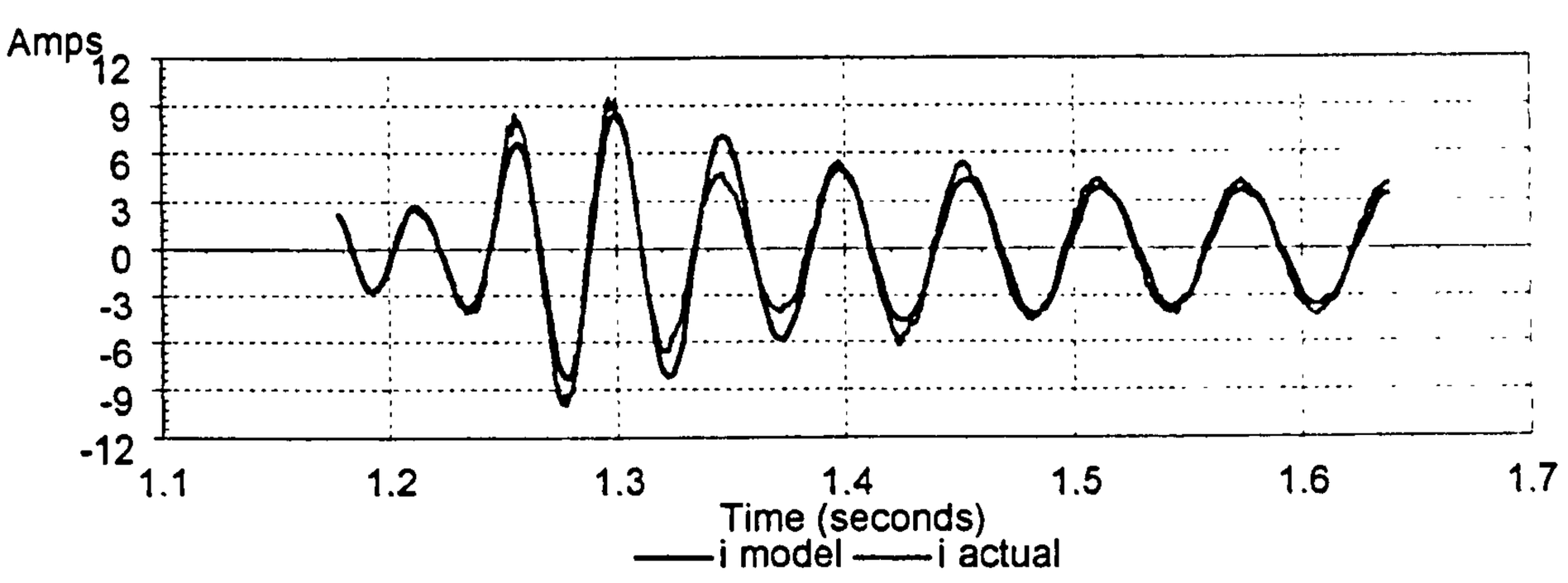


Figure 6.5.14: 26.66Hz to 13.33Hz, Phase Current, Speed Modelled, Varying Lm

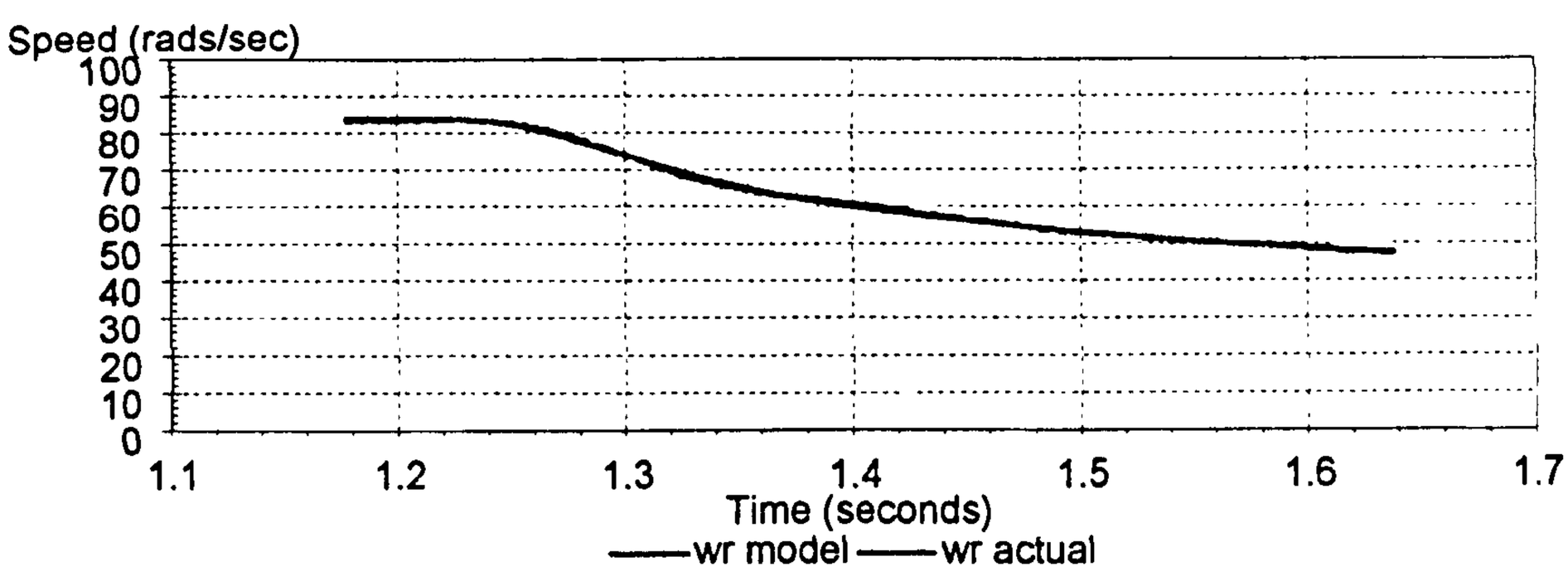


Figure 6.5.15: 26.66Hz to 13.33Hz, Model/Actual Rotor Speed, Varying Lm

6.5.2 Vector Control Results

Figures 6.5.16 to 6.5.19 show the results of running the real time simulation in parallel with the actual drive when the actual drive is controlled by a closed loop Vector Controller. The first show the steady state stator current of the simulated motor and the actual motor for constant demands of flux component i_{ds}^{e} and torque component i_{qs}^{e}.

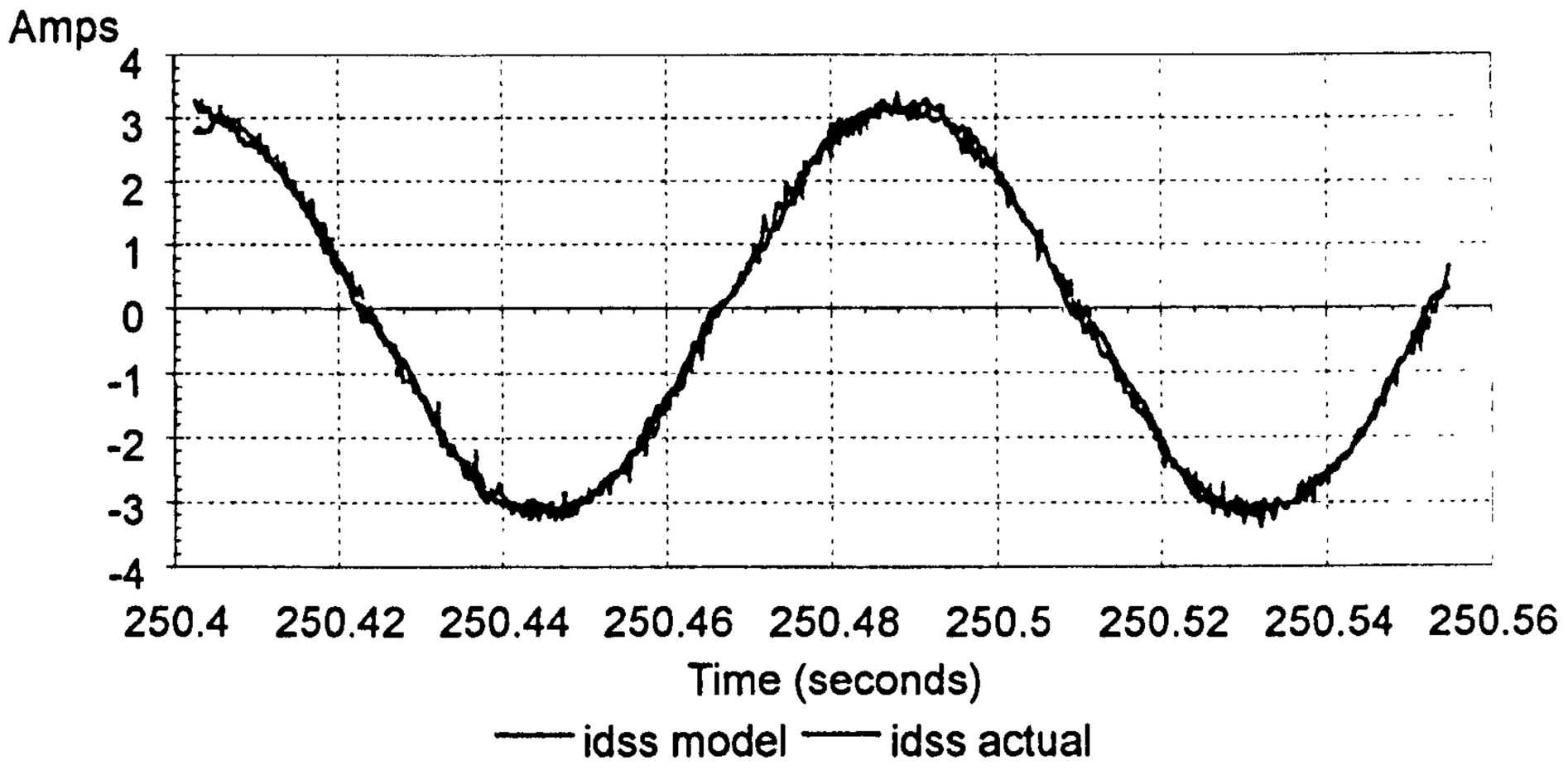


Figure 6.5.16: Model/Actual Stator Current Vector Control, i_{ds}^{e} =3A i_{qs}^{e} =1A

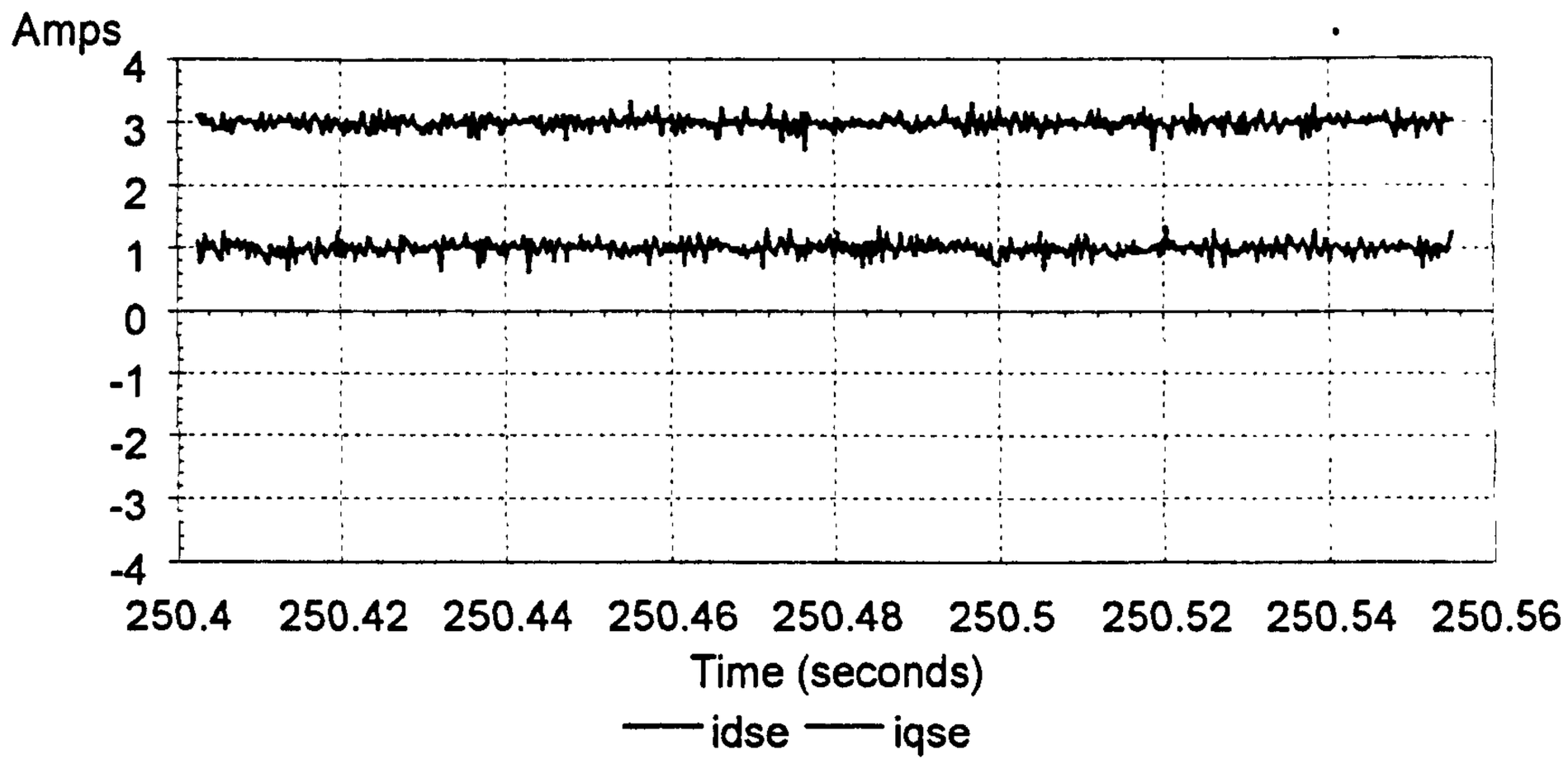


Figure 6.5.17: Actual i_{ds}^{e} and i_{qs}^{e}, for Demands i_{ds}^{e} =3A i_{qs}^{e} =1A

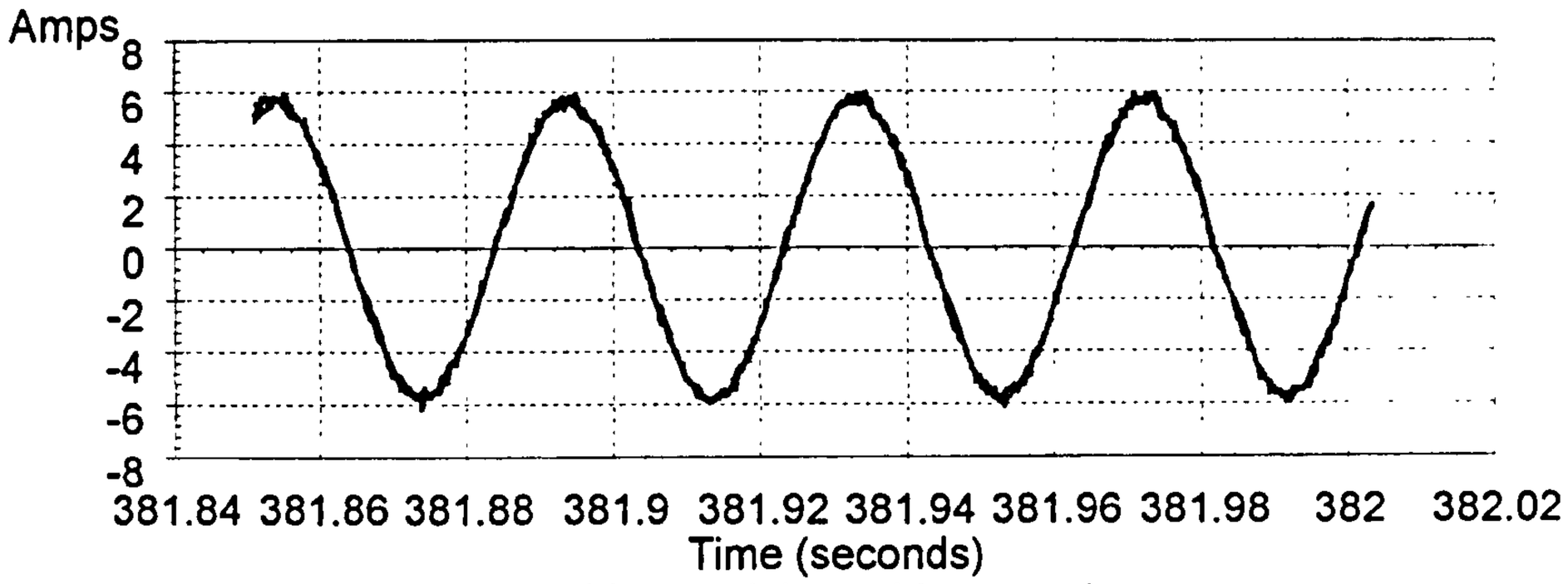


Figure 6.5.18: Model/Actual Stator Current Vector Control, i_{ds}^{e} =3A i_{qs}^{e} =5A

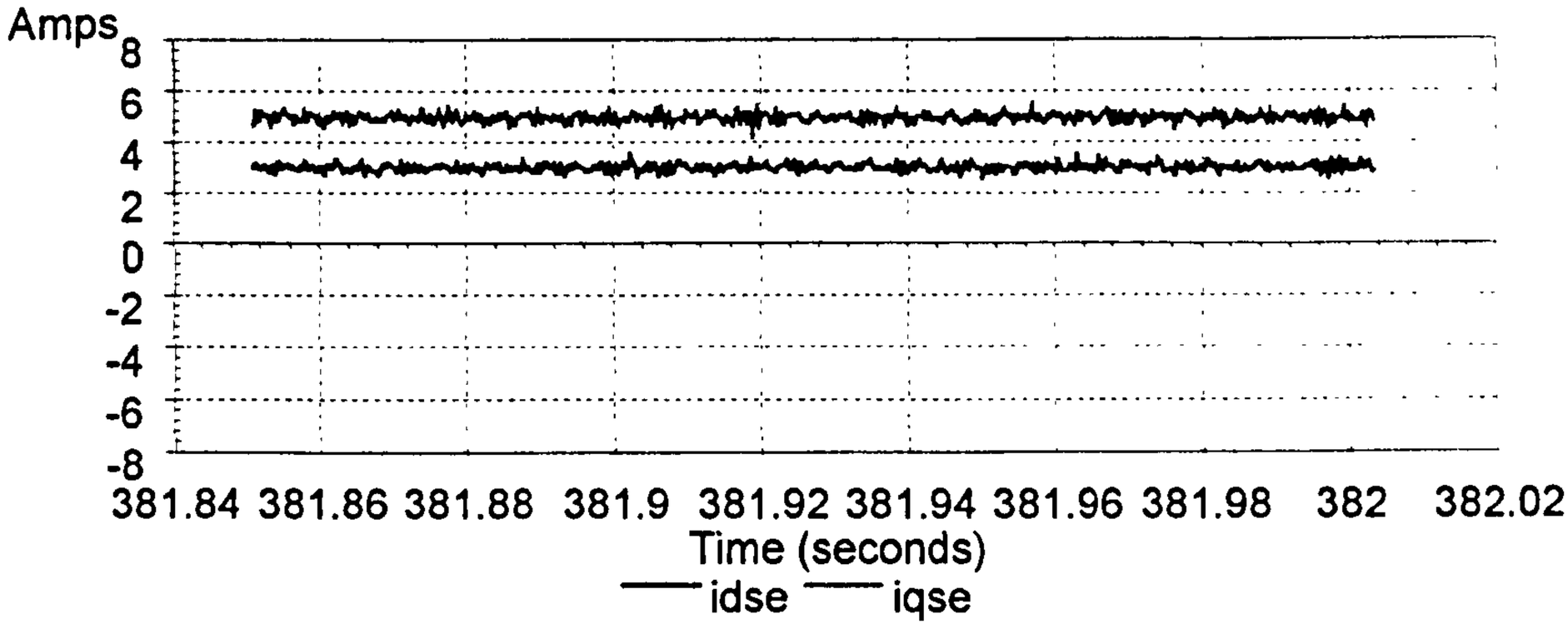


Figure 6.5.19: Actual i_{ds}^{e} and i_{qs}^{e}, for Demands i_{ds}^{e} =3A i_{qs}^{e} =5A

The steady state results show a close agreement between the actual stator current and the real-time simulated stator current. Figures 6.5.20 to 6.5.23 compare the real-time model to the actual system during a transient. The transient chosen is a sudden change in the torque producing current i_{qs}^{e} whilst holding the flux component i_{ds}^{e} constant. The first set of results are for a constant value of magnetisation inductance and the actual speed being used by the simulation. Both simulated currents for the increase and decrease in the torque demand follow the actual currents well. Both simulated currents are closer to the actual currents than in the open loop Volts/Hertz Control results, this is due to the fact that, fundamentally, the Vector Controller is trying to maintain a constant level of flux within the machine and so the fixed parameter model copes well with the simulation of the transients. The results of the actual stator currents in the rotor flux reference frame, i_{ds}^{e} and i_{qs}^{e}, show how the Vector Controller controls these DC currents resulting in a constant flux and a step in machine torque.

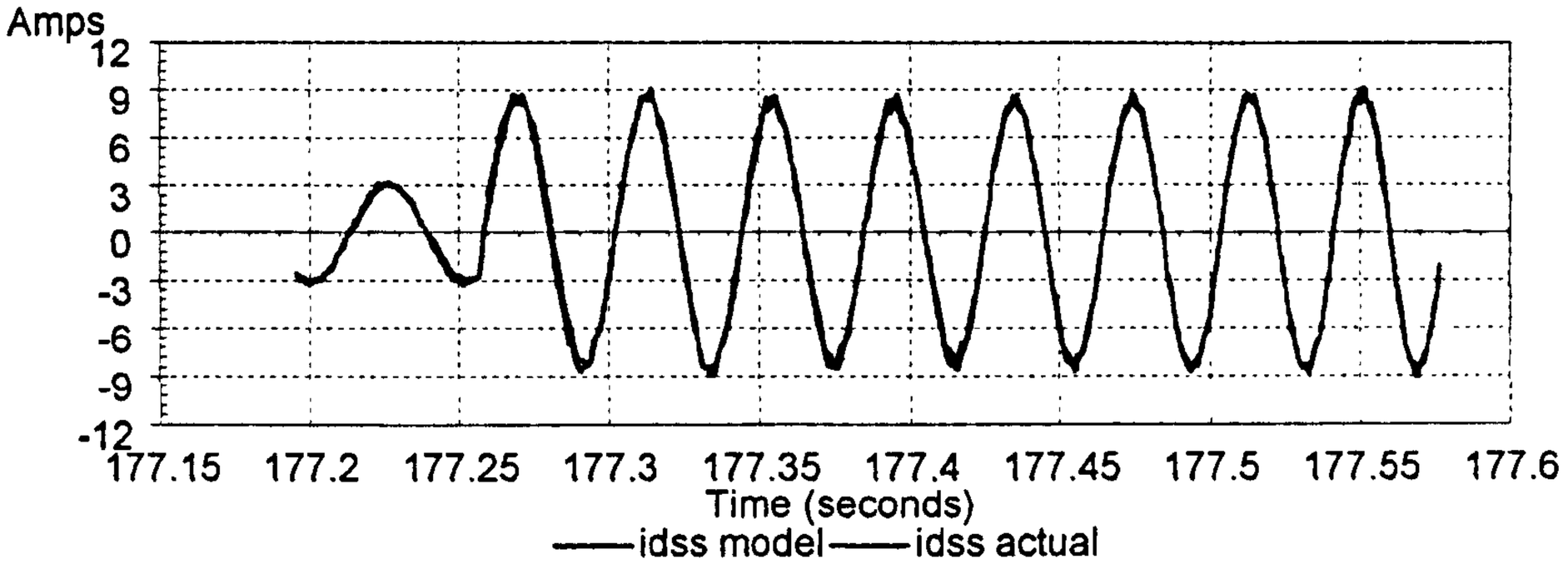


Figure 6.5.20: Model/Actual Stator Current Vector Control, i_{ds}^{e} =3A i_{qs}^{e} =1 to 8A

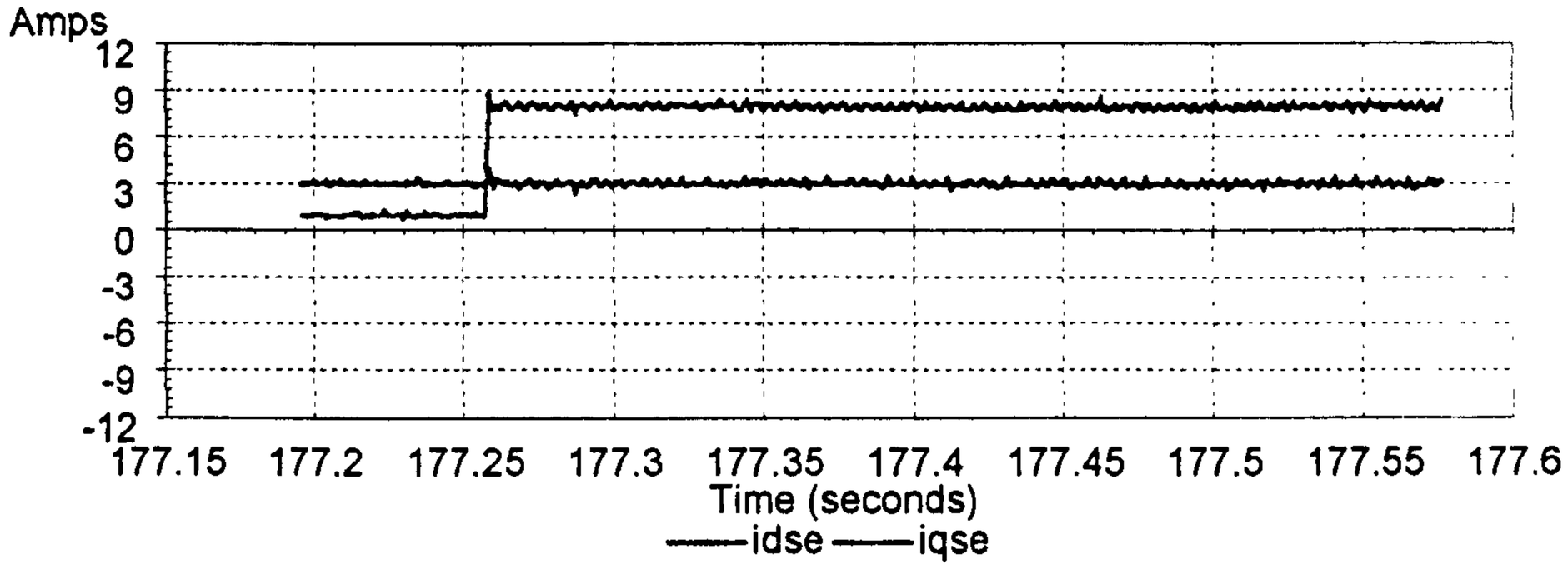


Figure 6.5.21: Actual i_{ds}^{e} and i_{qs}^{e}, for Demands i_{ds}^{e} =3A i_{qs}^{e} =1 to 8A

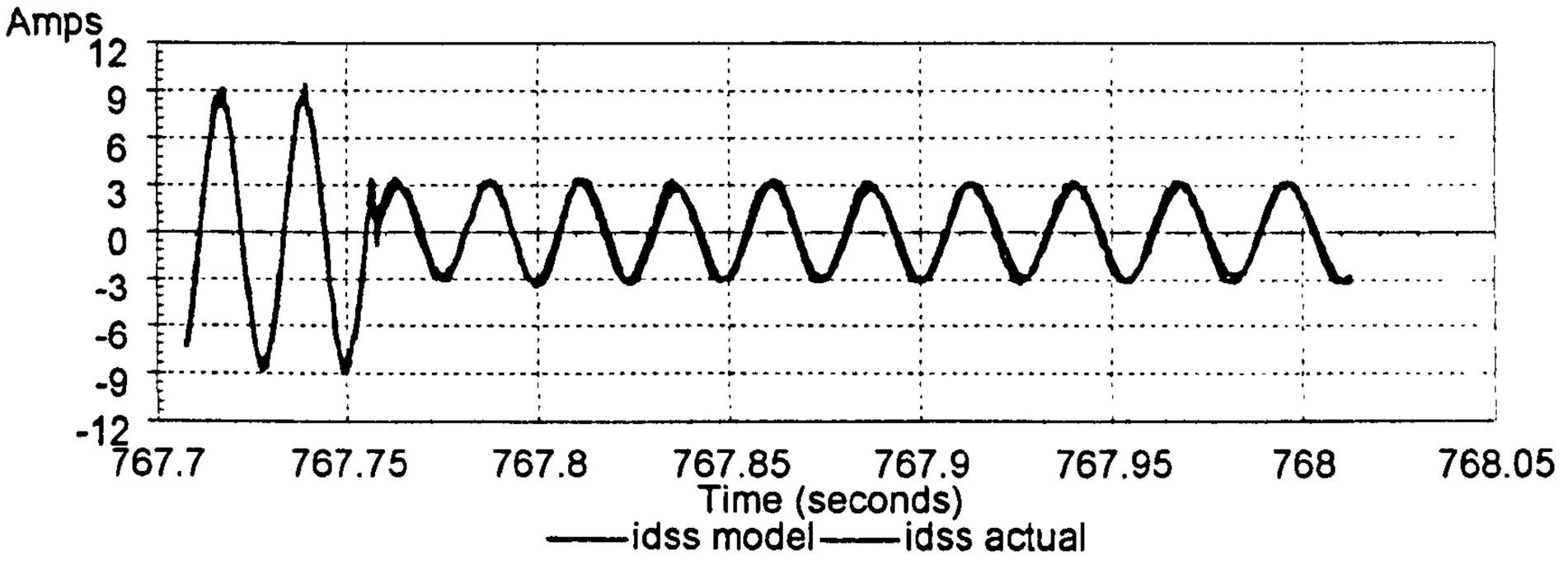


Figure 6.5.22: Model/Actual Stator Current Vector Control, i_{ds}^{e} =3A i_{qs}^{e} =8 to 1A

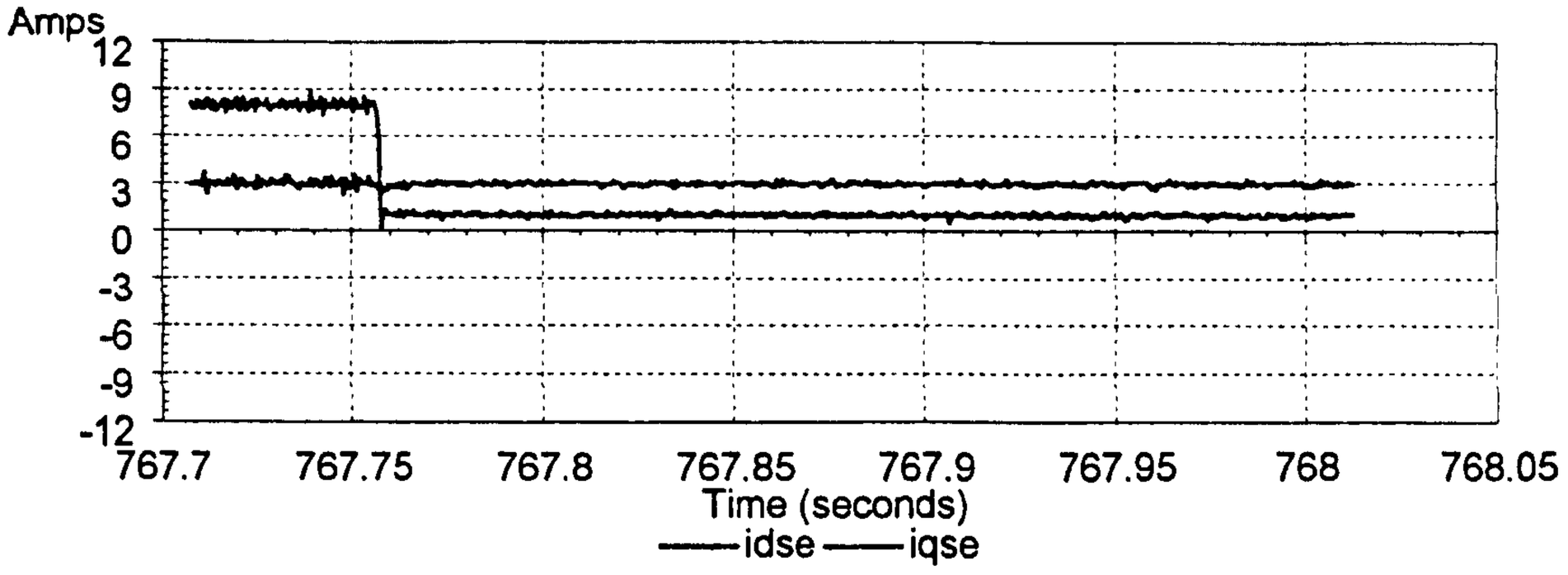


Figure 6.5.23: Actual i_{ds}^{e} and i_{qs}^{e}, for Demands i_{ds}^{e} =3A i_{qs}^{e} =8 to 1A

The following set of results, 6.5.24 to 6.5.27, are for the same transients but the simulation uses the simple saturation model described in the Volts/Hertz section. The stator currents are very similar to the results of the fixed parameter model, this is understandable since the magnetising current, which is also shown, varies by only a small amount which will result in only a small change in the magnetisation inductance.

The remaining set of results 6.5.28 to 6.5.31 show the torque demand transient with the simulation using the modelled speed as opposed to the actual speed . Both speed signals are also shown. Again the simulation ties up well with the actual currents during the transient.

A closer view of these results is given in 6.5.32 to 6.5.35, the controlled currents i_{ds}^{e} and i_{qs}^{e} are shown in the same detail. The response of the controller can be seen during the step in i_{qs}^{e} demand, and a slight interference in the flux component i_{ds}^{e} is also apparent during the sudden transient indicating that the Vector Controller has not achieved perfect decoupling between the D and Q axis in the rotor flux reference frame. No de-coupling terms are used in the controller which leads to coupling of the voltage equations in the rotor flux reference frame.

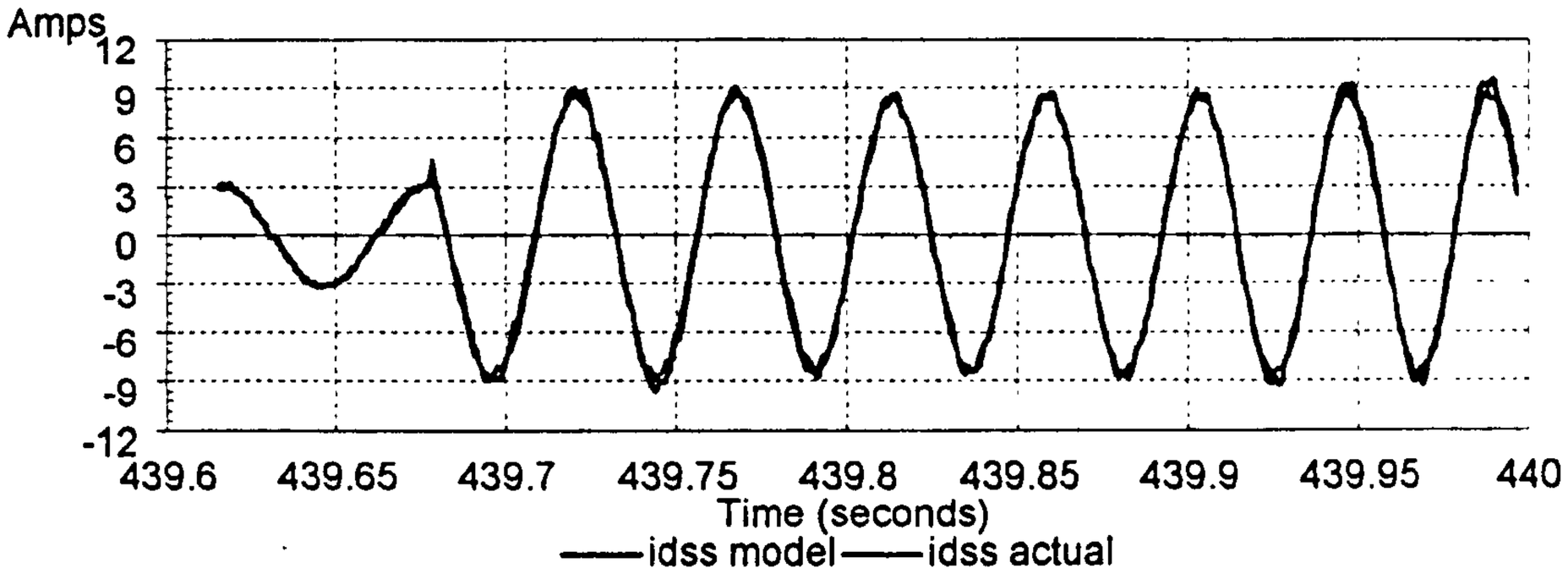


Figure 6.5.24: Model/Actual Stator Current, Varying Lm, i_{ds}^{e} =3A i_{qs}^{e} =1 to 8A

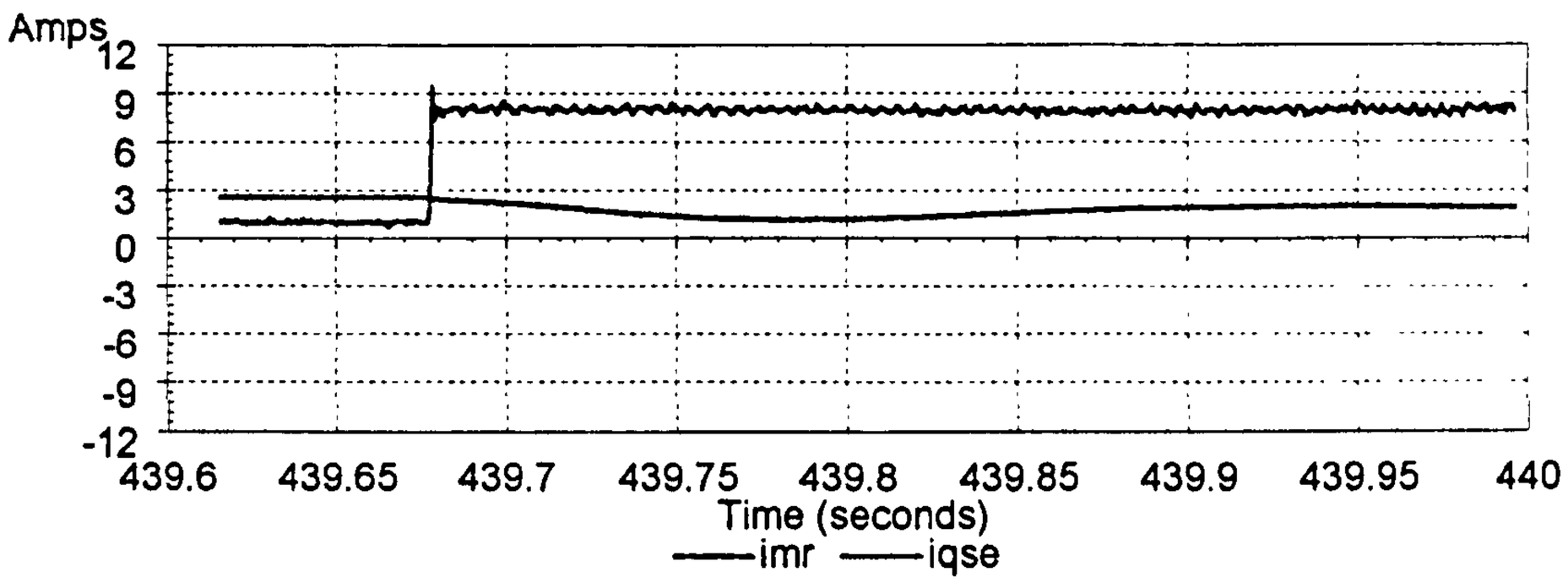


Figure 6.5.25: Actual i_{qs}^{e} and Modelled i_{mr}, for Demands i_{ds}^{e} =3A i_{qs}^{e} =1 to 8A

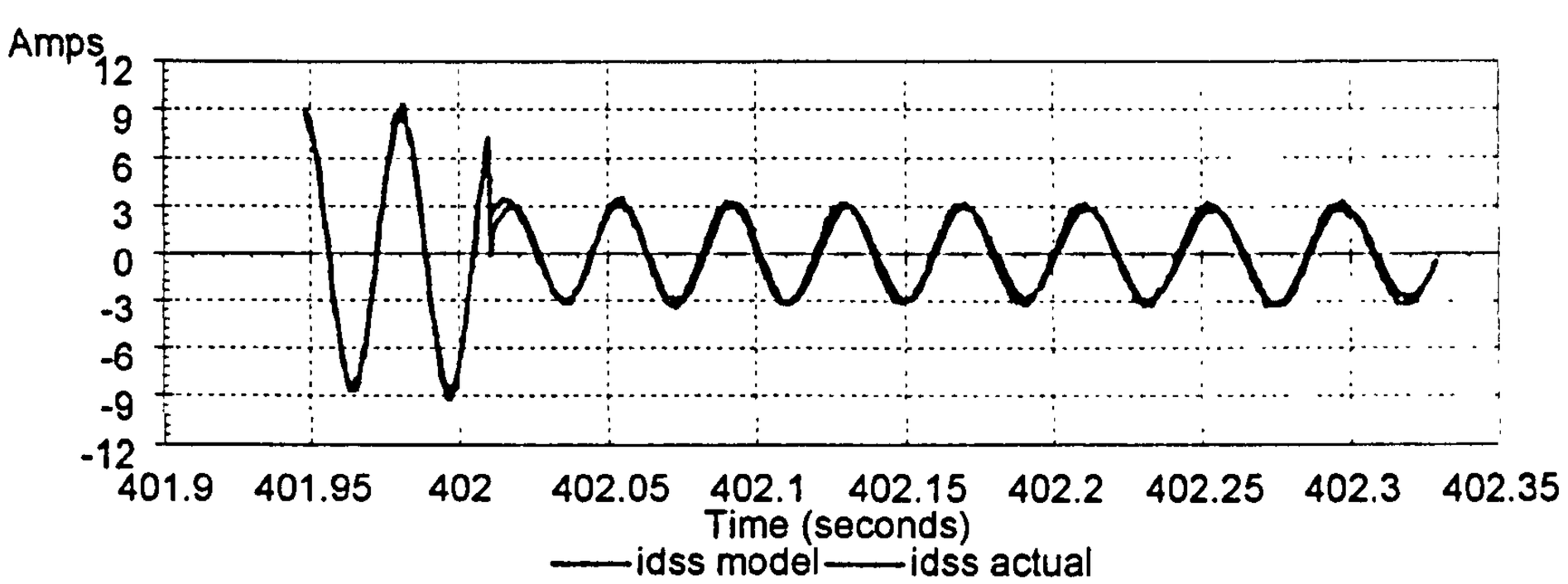


Figure 6.5.26: Model/Actual Stator Current, Varying Lm, i_{ds}^{e} =3A i_{qs}^{e} =8 to 1A

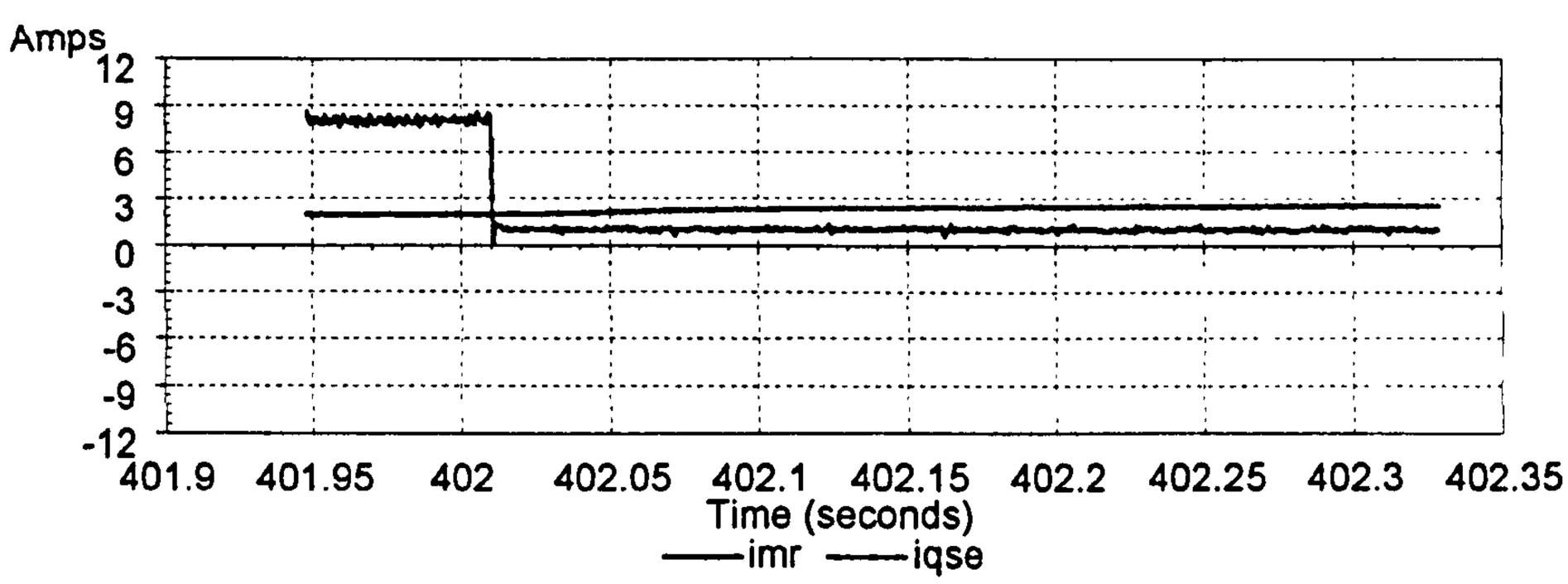


Figure 6.5.27: Actual i_{qs}^{e} and Modelled i_{mr}, for Demands i_{ds}^{e} =3A i_{qs}^{e} =8 to 1A

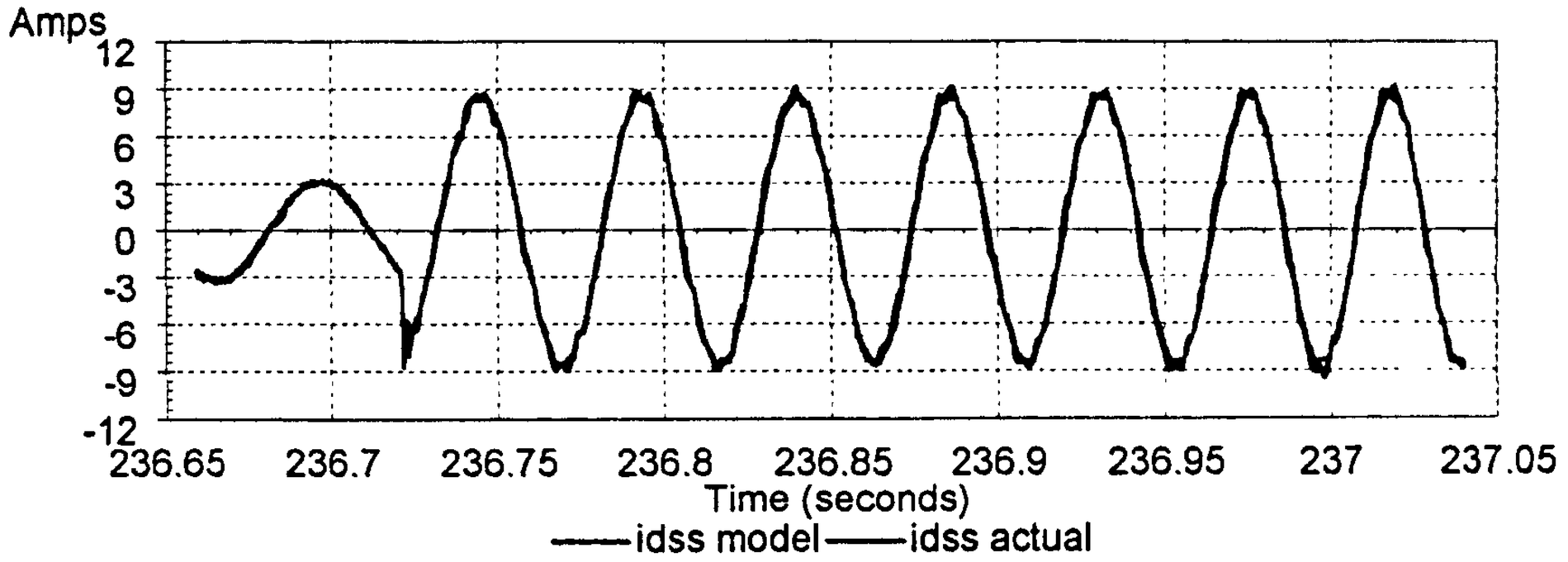


Figure 6.5.28: Model/Actual Current, Speed Modelled , i_{ds}^{e} =3A i_{qs}^{e} =1 to 8A

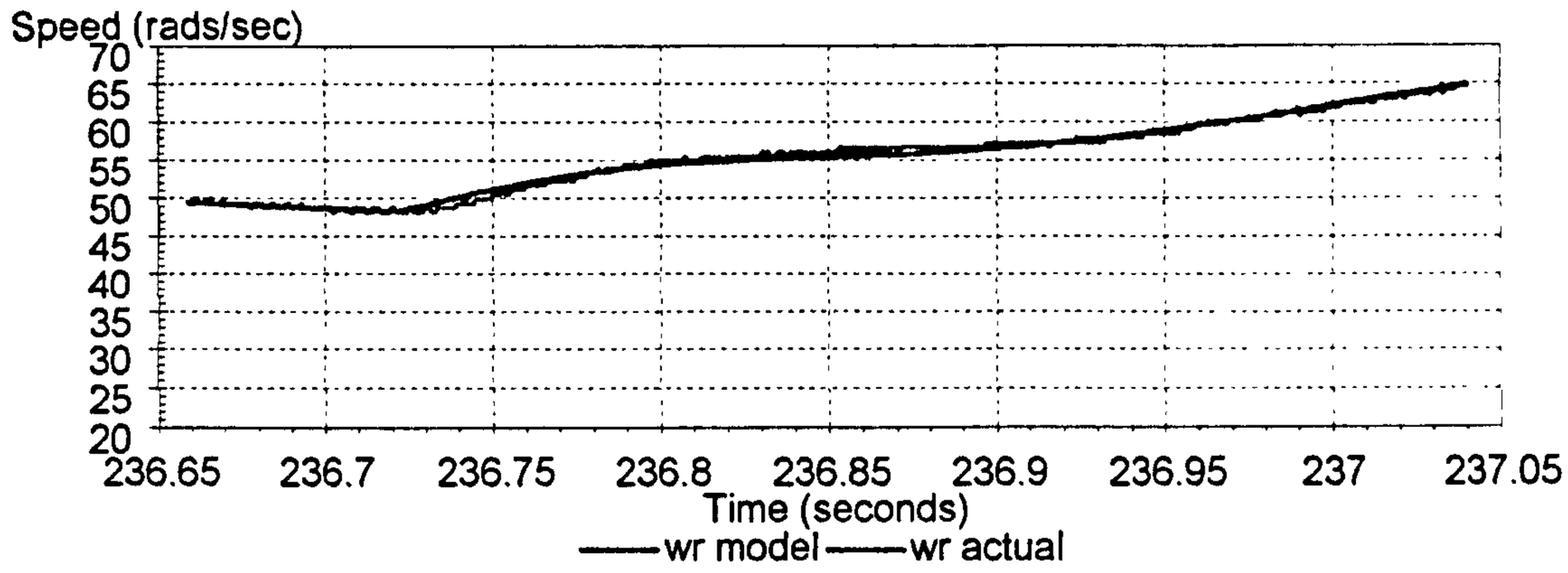


Figure 6.5.29: Model/Actual Rotor Speed , i_{ds}^{e} =3A i_{qs}^{e} =1 to 8A

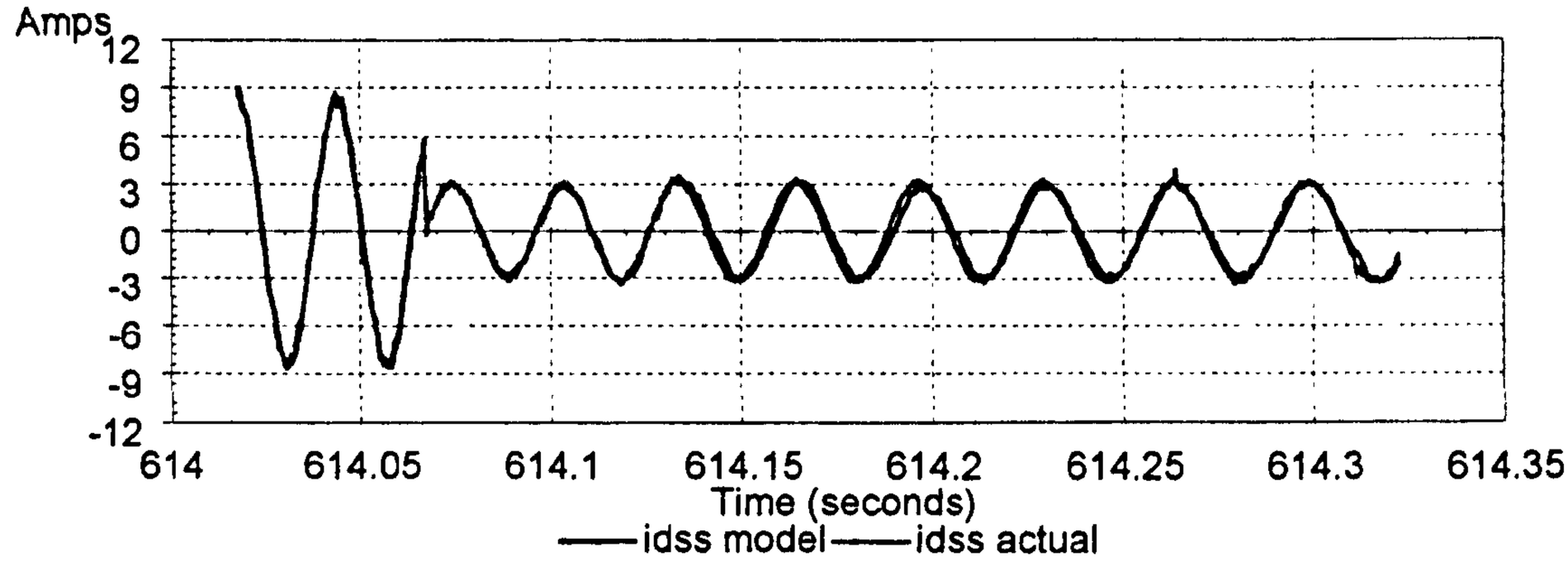


Figure 6.5.30: Model/Actual Current, Speed Modelled , i_{ds}^{e} =3A i_{qs}^{e} =8 to 1A

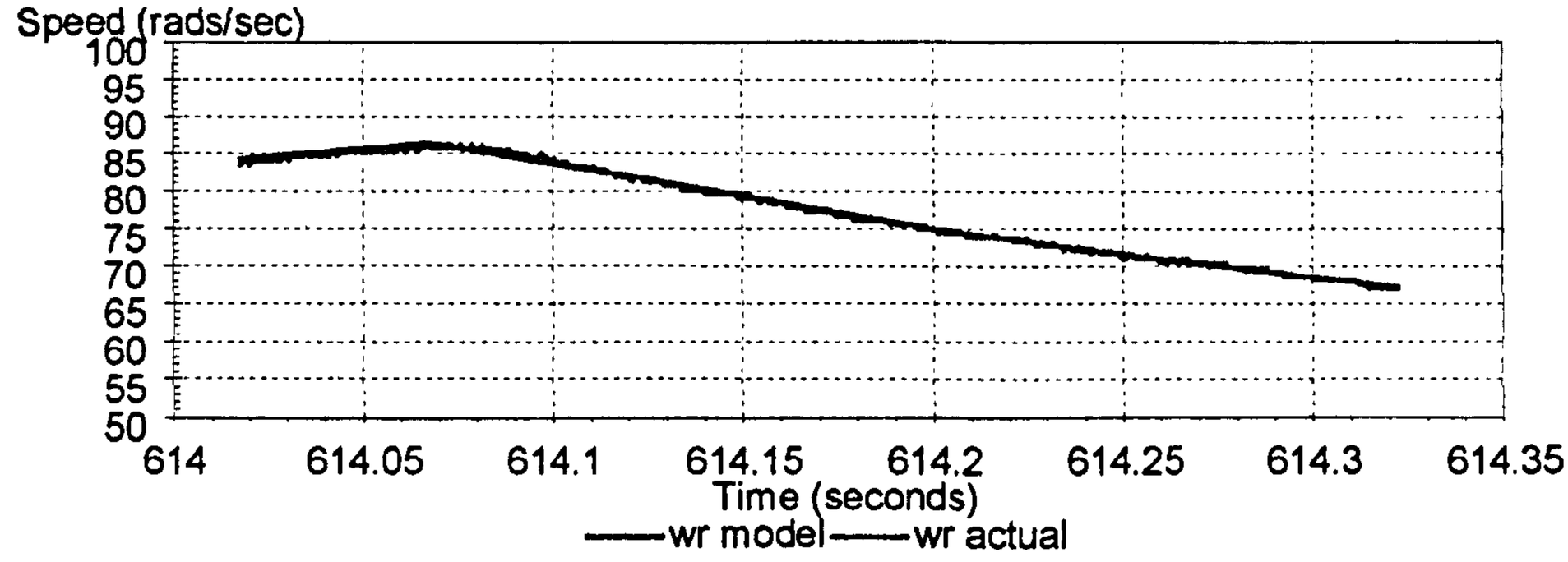


Figure 6.5.31: Model/Actual Rotor Speed , i_{ds}^{e} =3A i_{qs}^{e} =8 to 1A

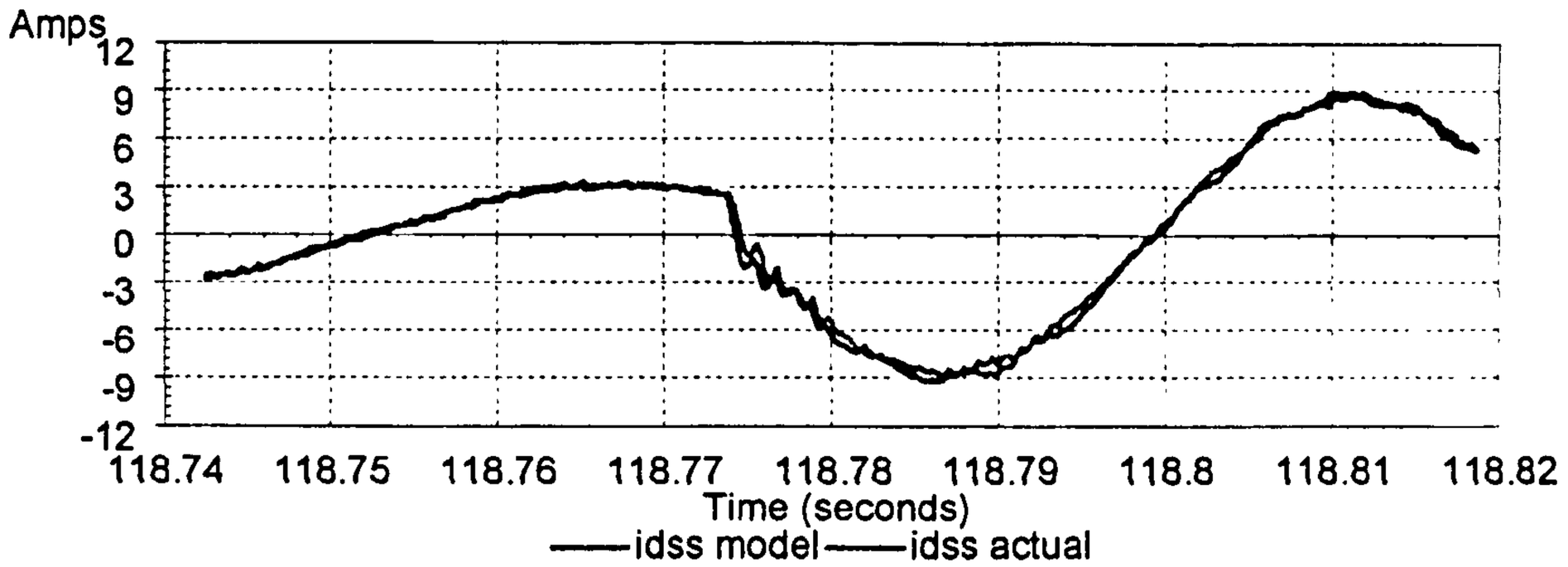


Figure 6.5.32: Model/Actual Current, Speed Modelled , i_{ds}^{e} =3A i_{qs}^{e} =1 to 8A

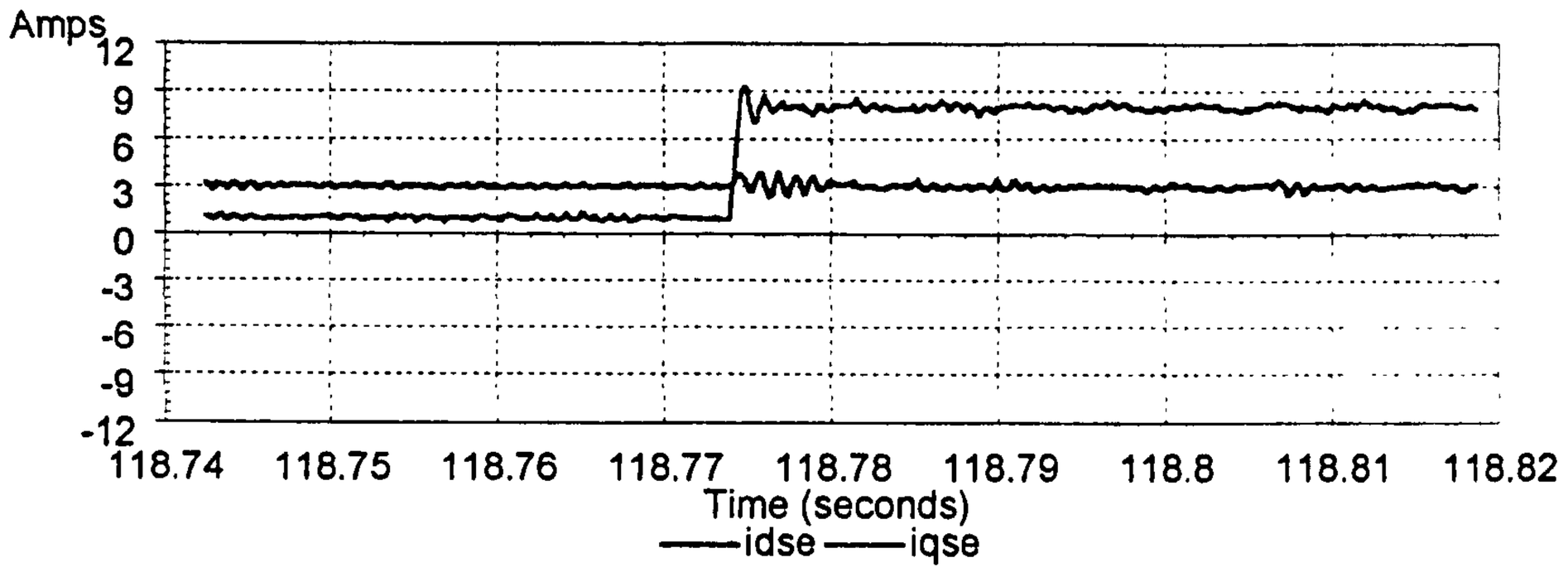


Figure 6.5.33: Actual i_{ds}^{e} and i_{qs}^{e}, Speed Modelled , i_{ds}^{e} =3A i_{qs}^{e} =1 to 8A

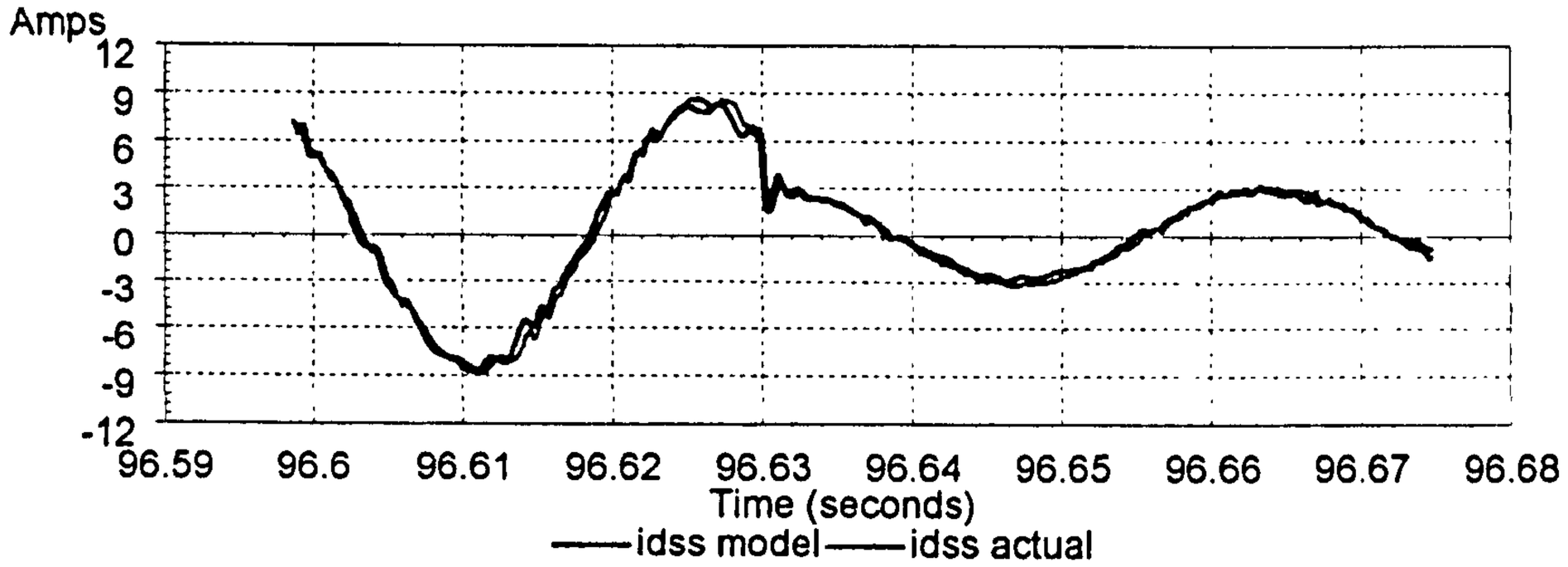


Figure 6.5.34: Model/Actual Current, Speed Modelled , i_{ds}^{e} =3A i_{qs}^{e} =8 to 1A

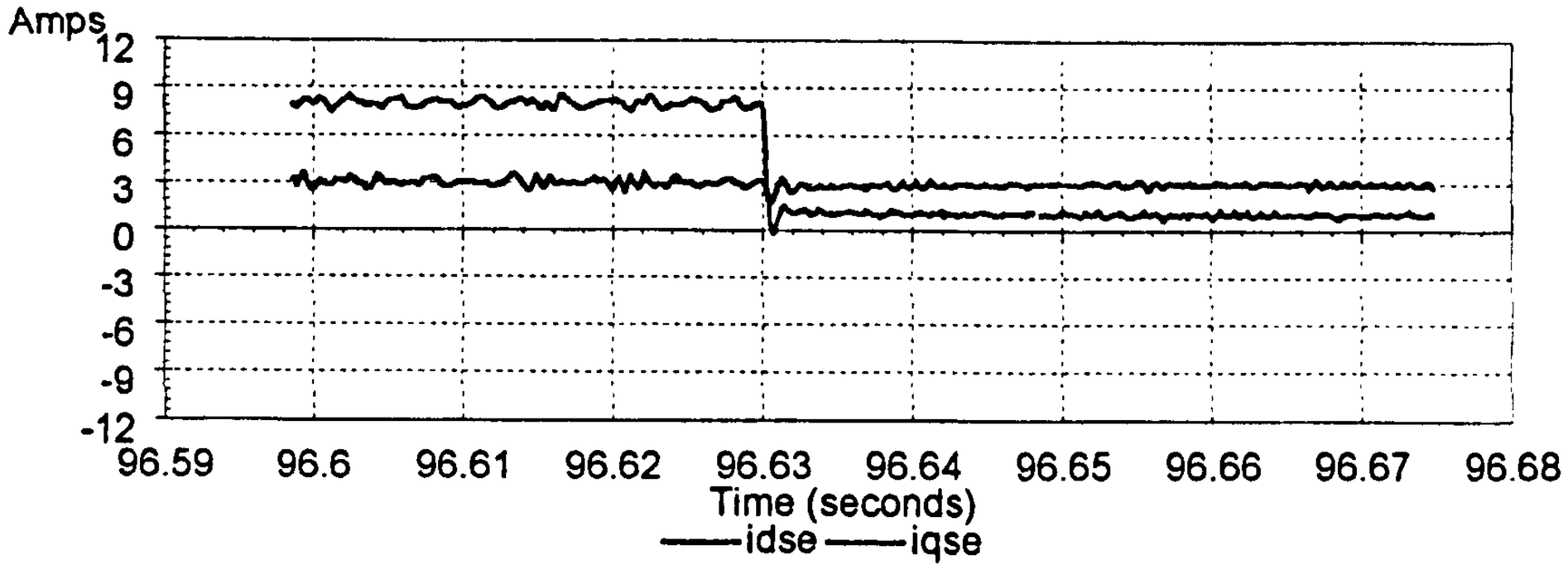


Figure 6.5.35: Actual i_{ds}^{e} and i_{qs}^{e}, Speed Modelled , i_{ds}^{e} =3A i_{qs}^{e} =8 to 1A

6.6 Summary

This chapter has taken the simulation of the induction machine described in chapter two, the simulation of the inverter described in chapter three and the controller described in chapter four. It has then demonstrated how these can be implemented in the multiple digital signal processing hardware set-up described in chapter five to achieve a simulation of the overall drive system which is capable of operating in real-time. Simulation of drives is normally approached in one of two ways, either a commercial simulation package is used such as [*Slater,Armstrong*, 1995],[*Nigim*, 1994], [*Chowdhury,Giesselmann*, 1994], or a dedicated simulation is designed such as [*Vas, Li*, 1993],[*Finney*, 1994]. The commercial option gives flexibility, useful data processing tools, and the ability to quickly build up simulations, it does, however, incur speed problems. A detailed comparison between a dedicated PC simulation and commercial simulation was included in chapter two, highlighting the faster execution times of the dedicated software approach. What this chapter is proposing is a different approach to drive development, one in which the controller developed within the simulation is the actual controller, running in the target processor, used in the final drive design. What makes this possible is the ability of the simulation to run in real-time.

Other approaches to drive development environments include those in which a commercial simulation package is used for simulation purposes and then the code is automatically generated for the target processor for example, Dspace, Link-RT by LSI and that designed for CAPEC at Newcastle University [*French*, 1994] . Although this approach is not attempting real-time simulation, it is recognising the need to automate the translation process from simulation to actual code. Chhaya and Bose, [*Chhaya, Bose*, 1993], offers yet another approach to drive development in which an expert system is used to design and simulate the drive system. It takes information such as the inverter power rating and produces a suitable drive design. This attempts to remove even more human involvement from the development stages.

A similar real-time simulation system is described in [*Bosga et al,* 1995] which is used as a teaching aid. In this system two C40s are used for control and simulation. The system uses a control panel featuring potentiometers, switches and LEDs. The control panel is used to input demands, and also change parameter values. This approach is taken ahead of the computer GUI approach because, although it is much less flexible, it is less dangerous for its application as a teaching aid. The system uses oscilloscopes and voltmeters to display measured or simulated variables since the system is not fast enough to display the data on a computer screen. The system uses one DSP for the controller and one for the motor model and assumes the modulator to be ideal. If the modulator is simulated then one of the DSPs is dedicated to this task and the other performs not only the control but also the motor model. This has drawbacks in that the control DSP is no longer performing as it will in the final drive design, i.e.control only. Unfortunately few details are given for the motor model and no results are presented which compare the real time simulation performance to a real drive.

The system developed for this project differs in that the variable time stepping approach allows the modulator and the motor to be modelled in real-time in a single DSP. The system also has the ability to acquire real time data by using a third DSP. The ability of the simulation to execute in real-time has allowed it to be operated in parallel with the actual system that it is trying to mimic and thus allowed direct comparison in real-time of the two systems. The direct comparison showed the real-time simulation to perform well in steady state and transient situations for both an open and closed loop controller. The comparison of the simulation to the actual system has demonstrated limitations of the fixed parameter model and an attempt was made to introduce some level of saturation modelling. There will always be a compromise, however, between simulation accuracy and simulation execution time. The timing section of this chapter has shown what level of simulation can actually be achieved in real-time with the current hardware. The main limitation of real-time simulation is obviously the available processing time. A limit has been reached with the present number of processors on the amount of simulation which can be achieved. Reducing the switching frequency would instantly give more time to the simulation as would the

addition of further processors or the introduction of more powerful processors. Included in the conclusion of chapter eight is a description of the type of processor which is soon to be available from Texas Instruments. As processor technology advances, it is not difficult to envisage a parallel processing system with the capability to simulate, in real-time, the switching characteristics of the inverter devices. Such processors would also allow the real-time implementation of far more advanced motor models, ones which accurately modelled such effects as saturation, cogging and crawling.

7. THE VIRTUAL MACHINE

7.1 Introduction

Simulators are commonly used to experiment with inverter design and control algorithms. A real time emulation environment has been described in previous chapters which uses a multiple digital signal processor system in which each processor simulates a different part of the overall drive system. Because of the ability of the simulator to operate in real time the actual controller running in the target processor can be used to control the simulation. The advantage of this system is that the controller code developed for the simulation is the same code as that used in the real drive and therefore development time is greatly reduced. When developing an inverter or control algorithm, however, eventually it must be tested with a real machine, this gives rise to several worries. If the control algorithm fails to perform as expected i.e. it loses control or fails to control correctly it could cause serious damage to the inverter being tested and / or the machine that it is being tested with. Also a complete range of machines is needed to test a range of inverters of different ratings. The Virtual Machine controls the currents drawn from the drive to match the currents which would be drawn if it were connected to a real machine, these current demands are obtained from the real-time motor model simulation which is driven from the output of the inverter under test. The motor model and load model parameters can easily be changed to make the Virtual Machine behave as a complete range of different motors and loads.

The Virtual Machine which has been developed is designed to allow the drive to be tested at full power levels without the need of a real machine, as far as the drive is concerned it is connected to a real machine. The fundamental idea of the Virtual Machine is to provide a simulation environment in which an inverter can be tested at real power levels without the requirement of an actual machine. The Virtual Machine

replaces the actual machine during the testing and development stages of the inverter design, thus providing a safer and more flexible development environment.

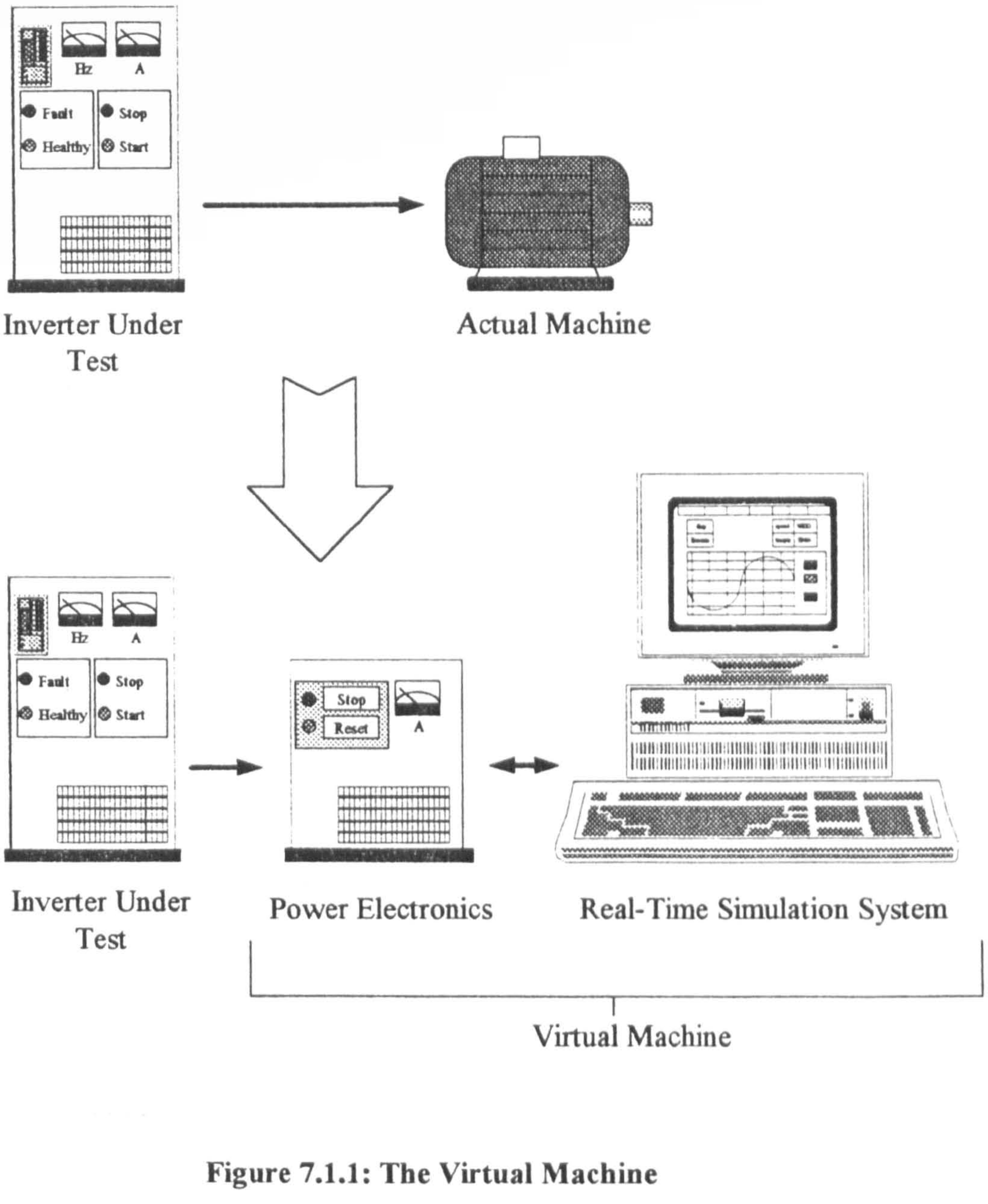


Figure 7.1.1: The Virtual Machine

7.2 Driving Simulation From Actual Inverter

The first step in achieving the Virtual Machine is for the simulation to have the ability to be driven directly from the output of the inverter under test. In order to drive the simulation from the actual inverter a method of sampling the output of the inverter had to be designed. The output of the inverter is a pulse width modulation signal, the switching frequency of this signal is dictated by the inverter under test. It is made up of high frequency pulses of magnitude equal to the DC link voltage of the inverter. It is therefore impossible to simply sample the output voltage at fixed time intervals and then apply this to the motor model and expect sensible results. It is necessary to have an accurate measure of the average voltage that the inverter would apply to the motor windings for the sample interval and then apply this to the motor model which can then time step this interval using the average volts.

The sampling system used for the Virtual Machine consists of three continuous hardware integrators which integrate the line to line voltage outputs of the inverter under test. Each integrator is sampled and reset at each control period of the Virtual Machine. The switching frequency of the Virtual Machine was chosen as 13KHz which gives a control period of approximately 77μs. The integrators, therefore, integrate the output of the inverter under test for 77μs and are then reset to zero by the processor controlling the Virtual Machine. This sampling system is completely independent from the switching frequency of the inverter under test and no connection to the inverter under test is necessary other than the three voltage outputs which are normally connected to the real machine.

In order to test the concept a Matlab/Simulink simulation of the overall system was used. Figure 7.2.1 shows the Matlab/Simulink simulation of the sampling system.

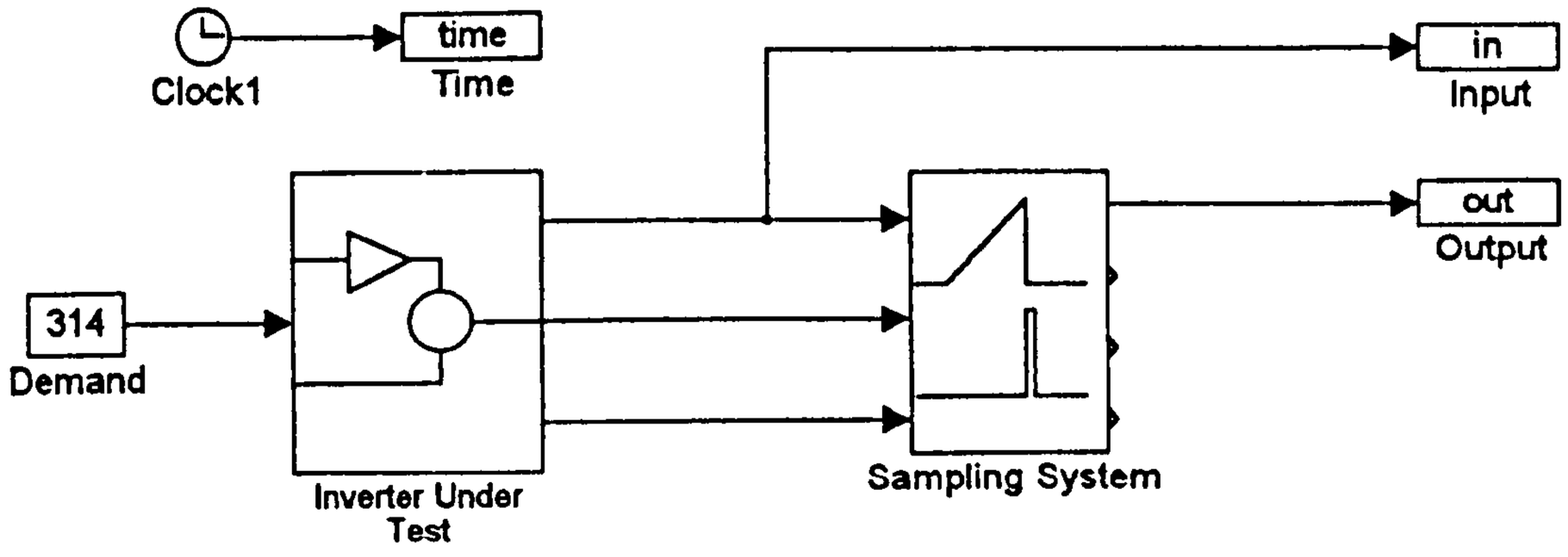


Figure 7.2.1: Simulation of Sampling System

The block entitled 'Inverter Under Test' is simply a simulation of a voltage source inverter controlled by an open loop Volts/Hertz controller. The following figure shows the contents of this block.

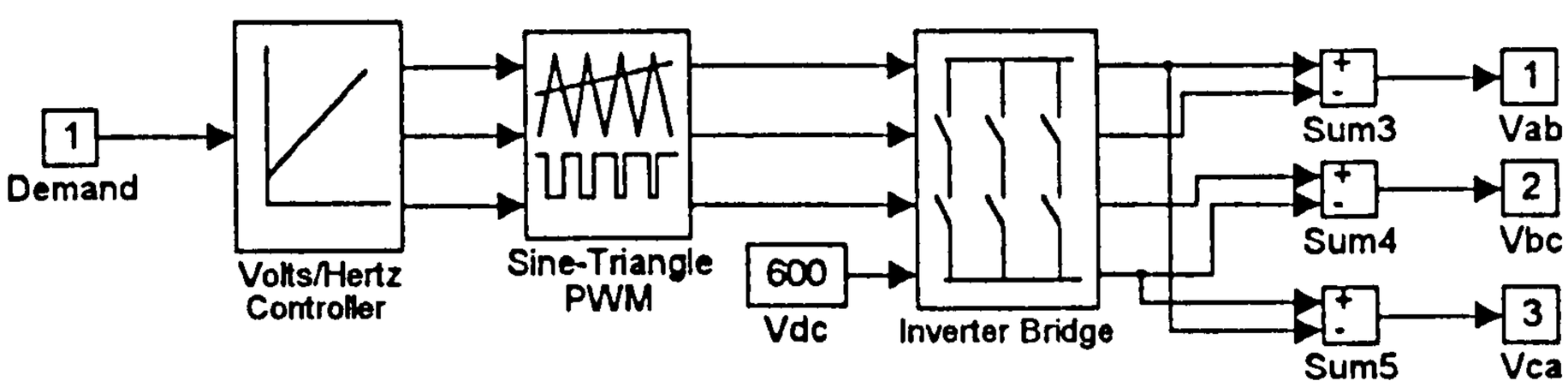
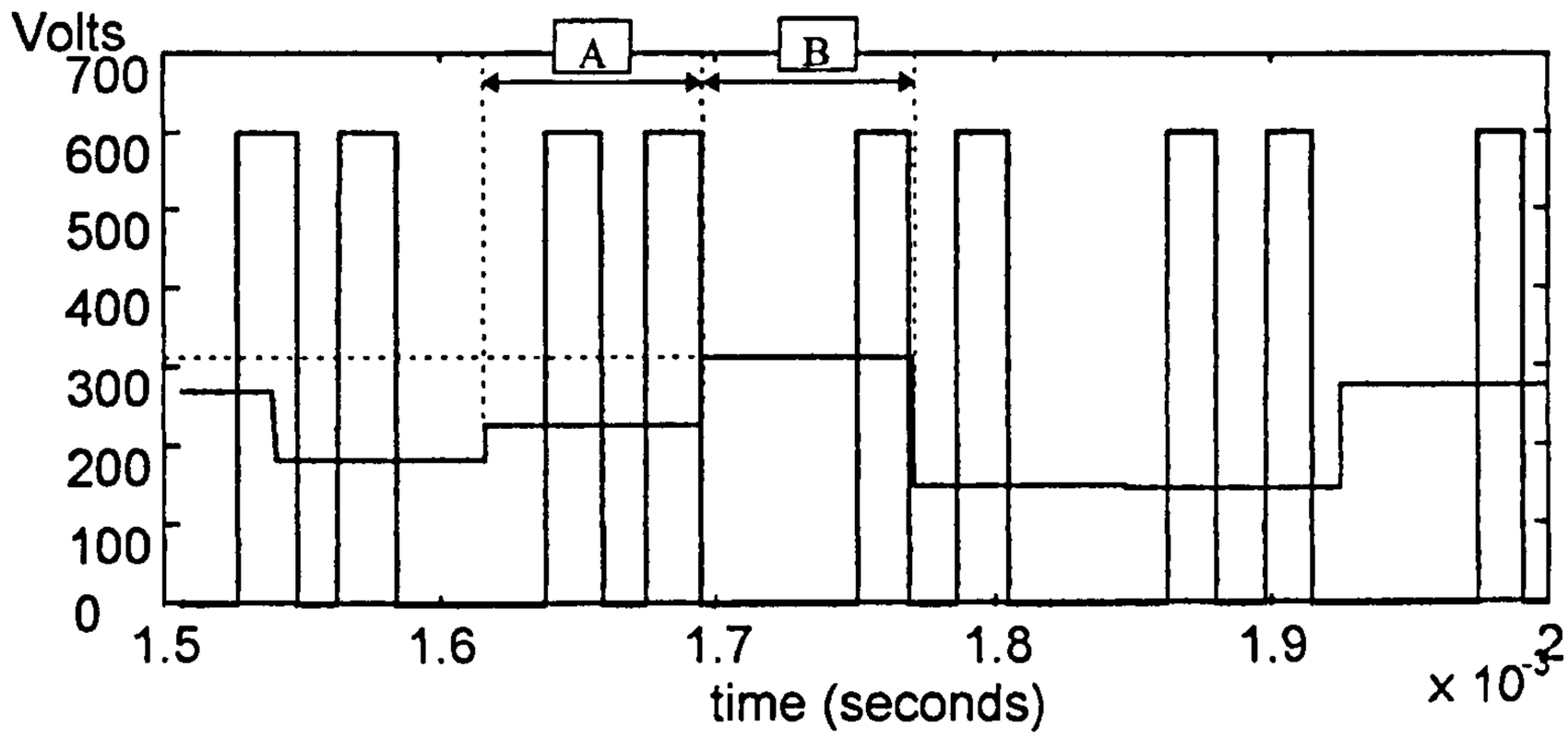


Figure 7.2.2: Inverter Under Test Simulation Block

The switching frequency of the inverter under test was set at 9KHz , figure 7.2.3 shows the output of the inverter under test and the corresponding output of the sampling system. If the output of the inverter under test during the sampling period 'A' is considered, it can be seen that two pulses, each of 600v and of 19.56μs duration occur. If these two pulses are averaged over the sampling period of 77μs the result is approximately 305v. It can be seen from the second period 'B', that the output of the sampling system matches this value, therefore the sampling system produces the average voltage which would have been applied to the actual motor if the inverter under test was driving an actual motor for the sampling period.



—— Output of Inverter Under Test —— Output of Sampling System

Figure 7.2.3: Sampling System Input and Output

In order to observe what effect driving the motor model from the sampled voltage as opposed to the actual PWM signal has, two motor models were run in parallel. One motor model was fed from the normal PWM output of the inverter under test, and the second was fed from the output of the sampling system.

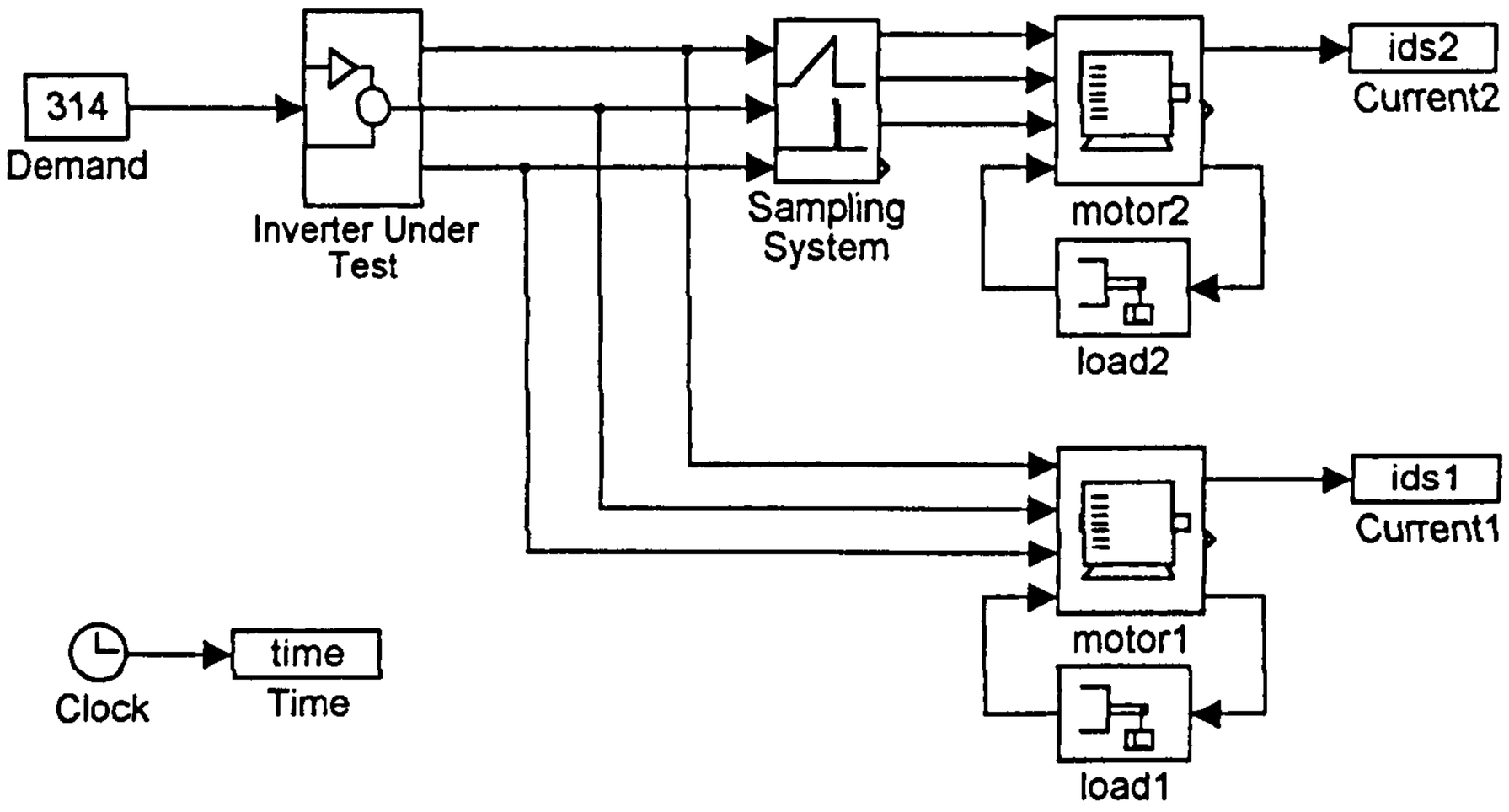


Figure 7.2.4: Simulated Comparison of Sampling System

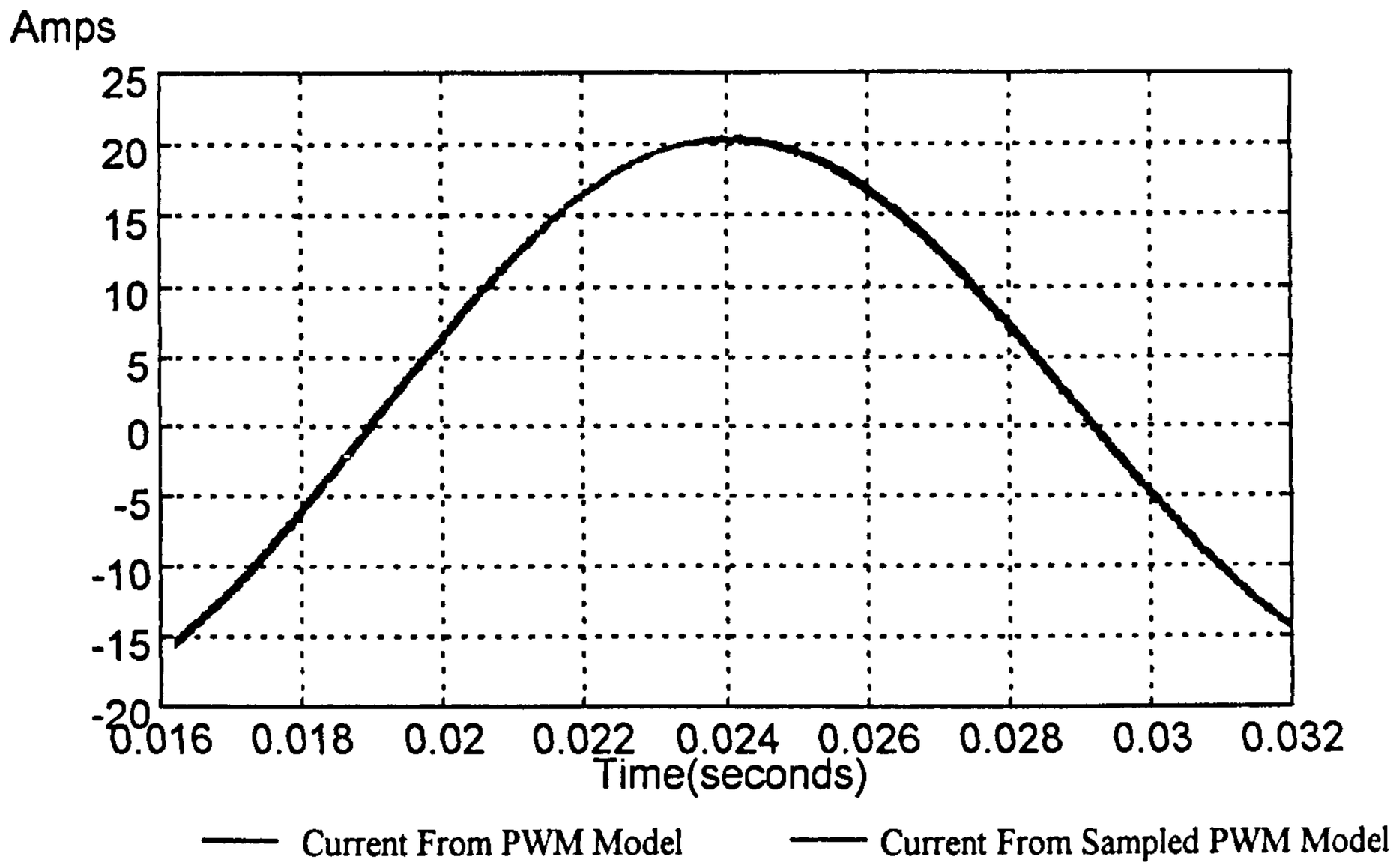


Figure 7.2.5: Resultant Phase Current from Sampling System

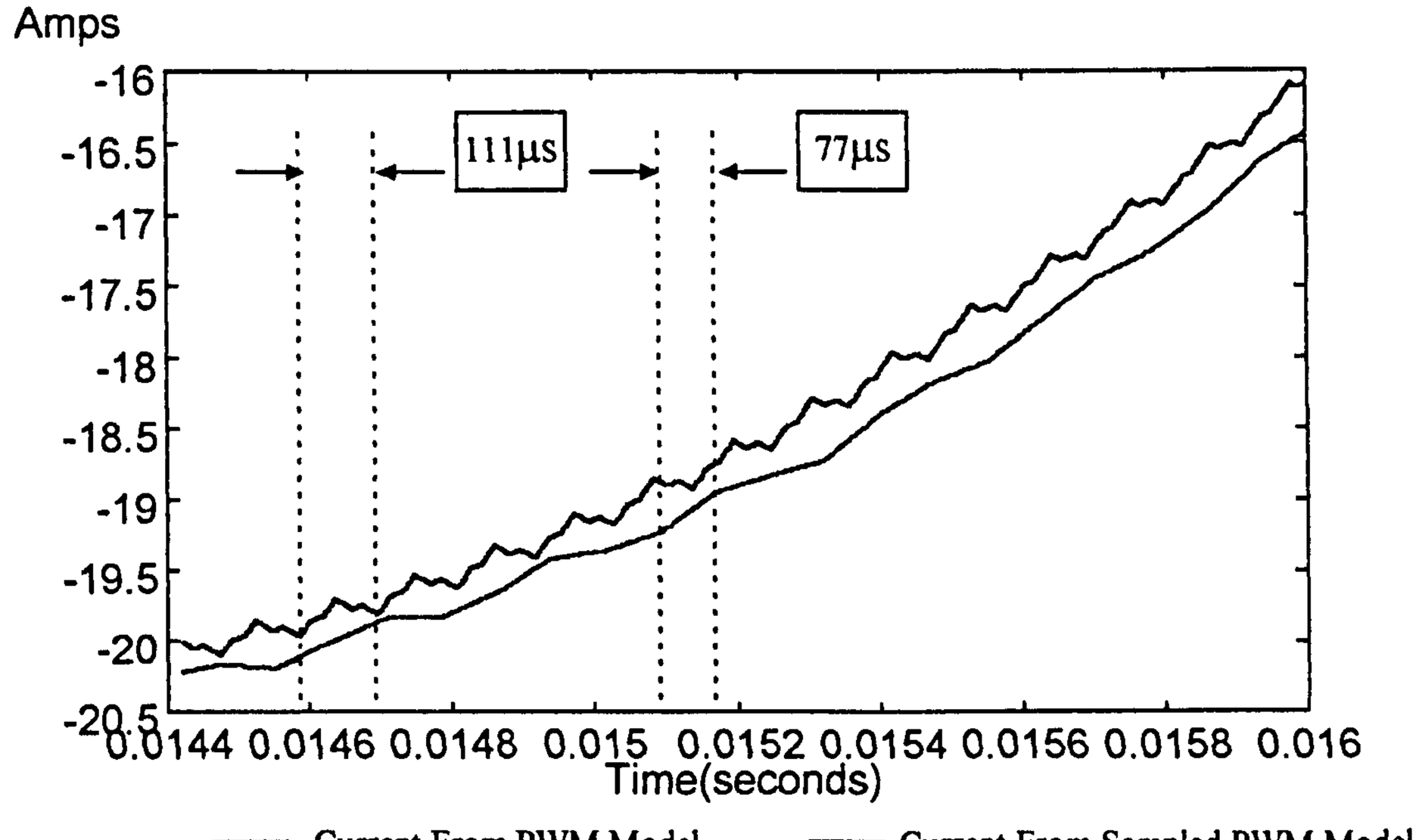


Figure 7.2.6: Resultant Phase Current from Sampling System (Detailed View)

Figure 7.2.4 shows the system modelled and figures 7.2.5 and 7.2.6 show the currents from the two motor models, figure 7.2.6 showing detail from figure 7.2.5.

Figure 7.2.6 shows two periods, the PWM period of the inverter under test corresponding to a switching frequency of 9KHz which results in the period of 111μs and the period of the sampling system 77μs. The current from the model fed from the sampled PWM has a phase shift equal to the sampling period therefore it is important to ensure that the sampling period is kept small. The advantage of the integration method of sampling is that it is continuous i.e. the output PWM signal is constantly being integrated and averaged. The only part of the PWM signal which is lost is that which occurs during the reset period of the integrators, in the present system this is 1.4μs.

The following system was used to observe the performance of the real-time simulation of the machine when fed from a standard inverter. The inverter, at this stage, was connected to an actual machine to allow comparison between actual currents and real-time modelled currents. Obviously, with this set-up, the simulation of the inverter has been replaced by the actual inverter and so the processor carrying out the simulation only has to perform the motor model simulation and not a simulation of the complete drive. The processor which had been used as the drive controller in the real-time simulation, is only used to read actual currents from the machine and also read and reset the hardware integrators. The third processor is still performing data acquisition routines and interfacing to the GUI.

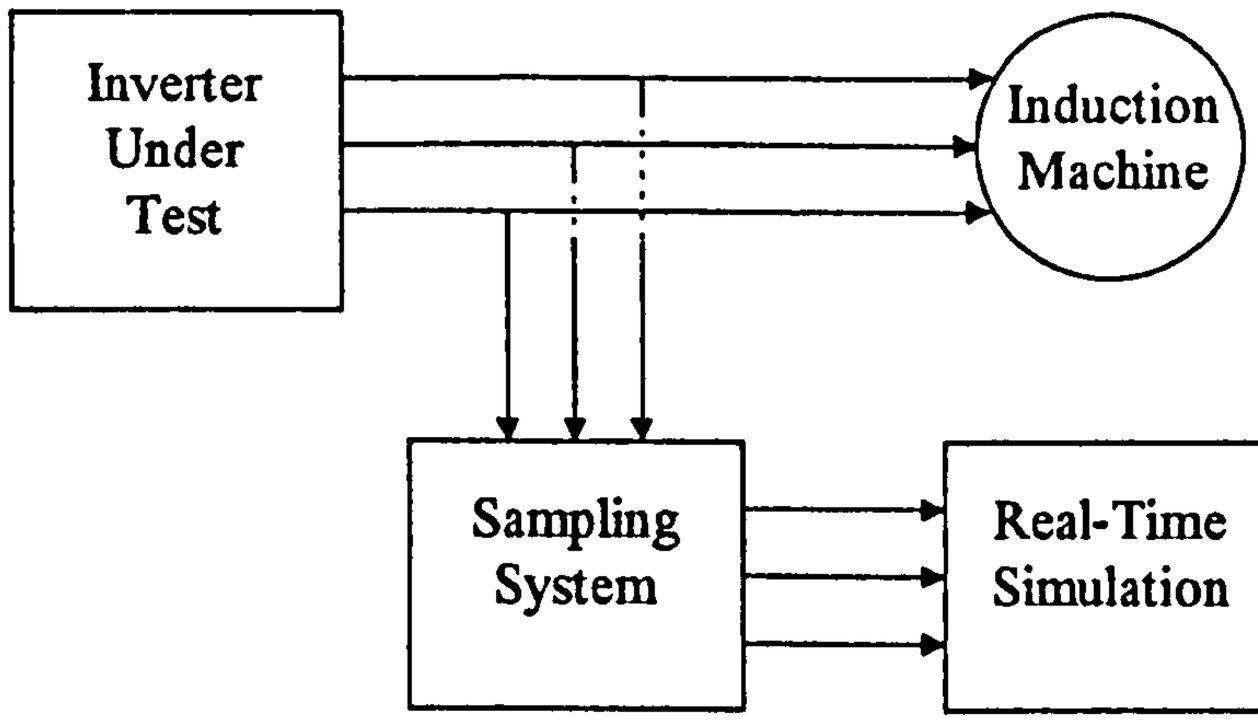


Figure 7.2.7: Real-Time Simulation Fed from Inverter Under Test

The inverter under test in these experiments was a Control Techniques CDE 400 4KW drive. This inverter has four possible switching frequencies, 3,6,9 and 12KHz. The following results were taken with the switching frequency set at 12KHz with the inverter operating with open loop control. Figure 7.2.8 shows the output of one of the three hardware integrators, and the output voltage of the inverter which was the input to the integrator.

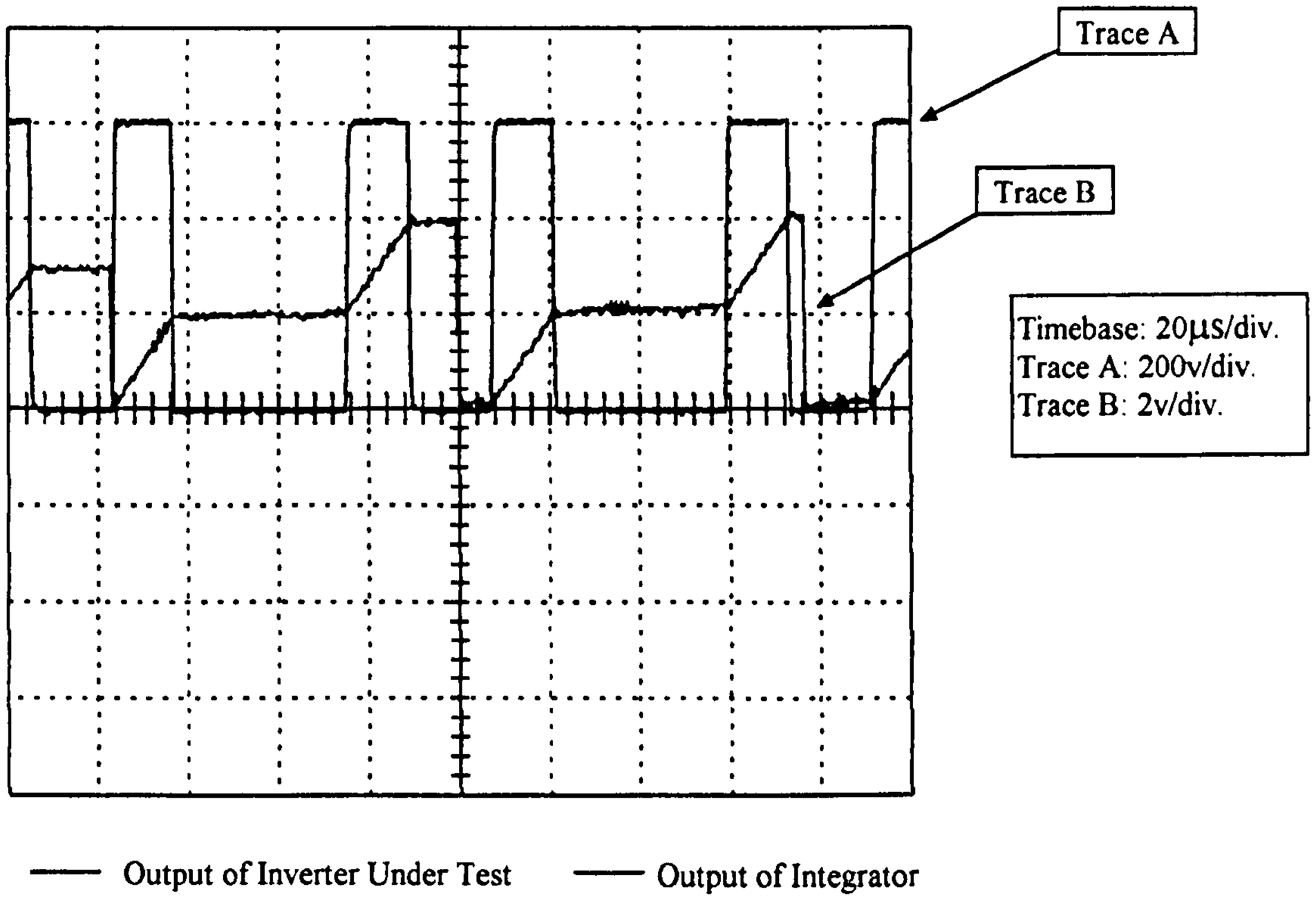


Figure 7.2.8: Input and Output of Hardware Integrator

A direct comparison can now be made between the actual machine phase current and the real-time simulation phase current fed from the sampled PWM. The following figures (7.2.9 and 7.2.10) show the steady state and transient comparison of the two currents. The steady state current is for a frequency demand to the inverter under test of 30Hz. The transient results were taken for a full speed reversal from 50Hz to -50Hz. As can be seen the agreement is very good.

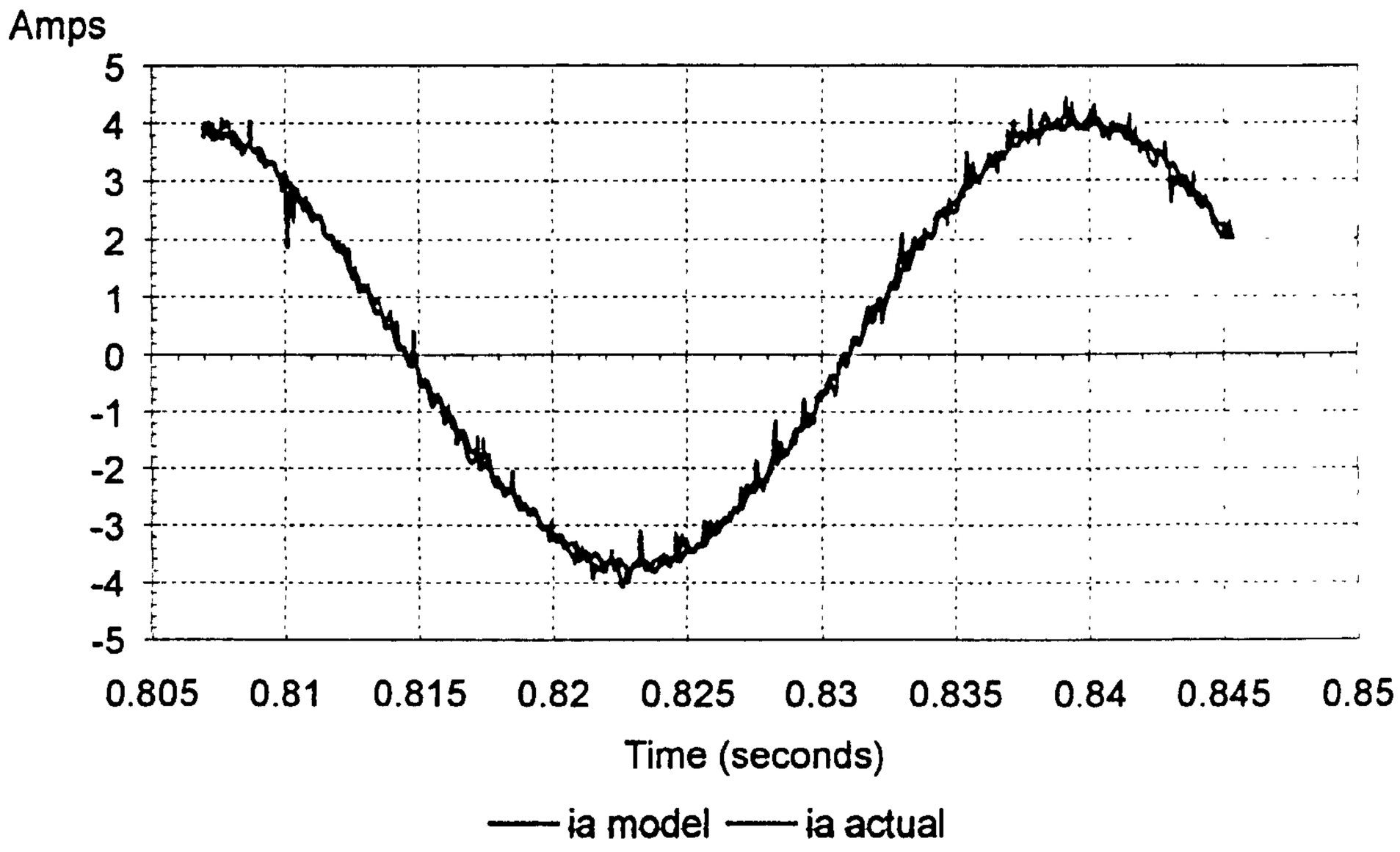


Figure 7.2.9: Steady State Comparison of Actual and Real-Time Model Currents

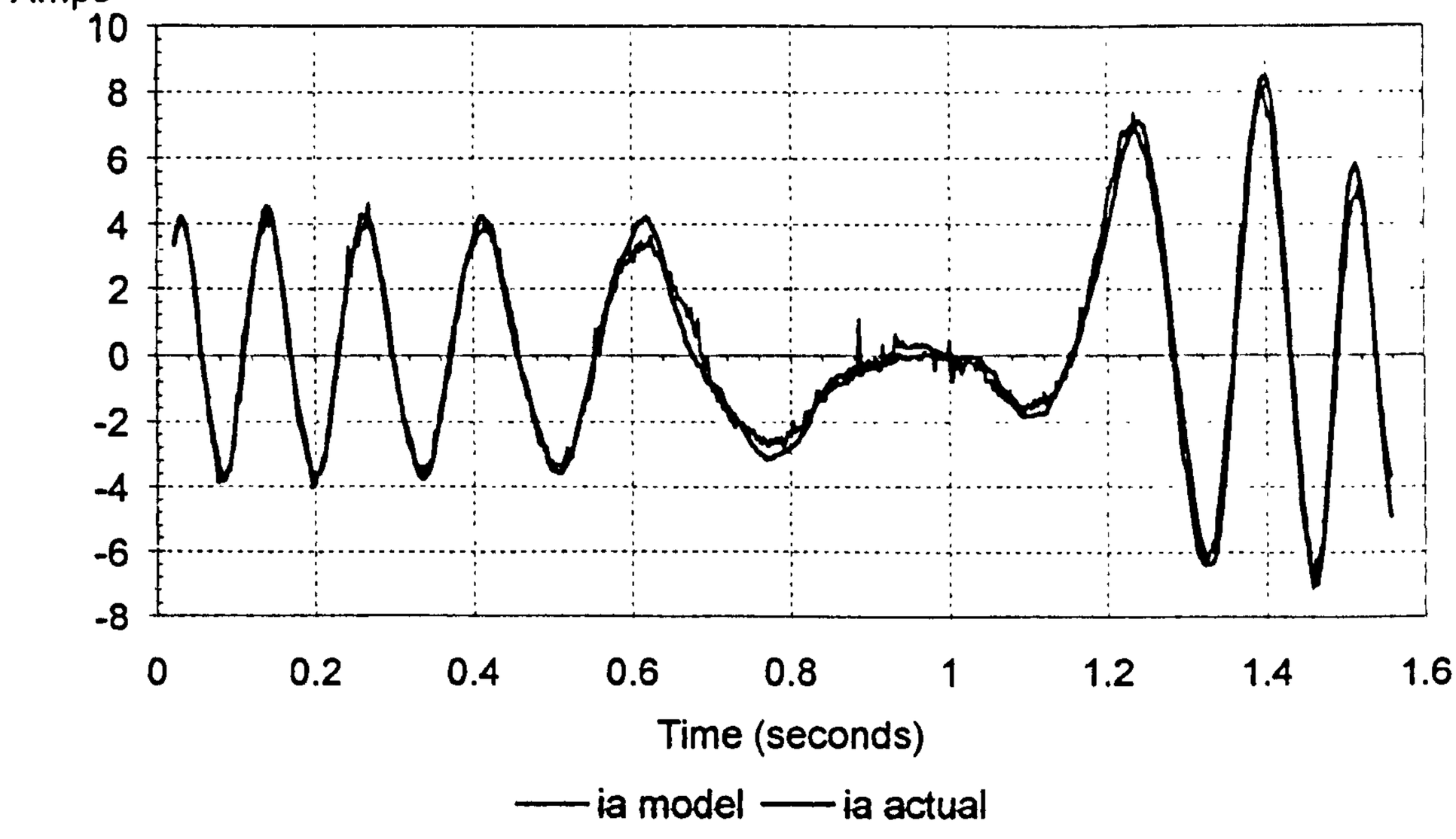


Figure 7.2.10: Transient Comparison of Actual and Real-Time Model Currents

7.3 Achieving the Virtual Machine

Two important steps have been made towards achieving the Virtual Machine, first the ability of the simulation to execute in real-time and secondly the ability of the simulation to be driven from the output of any real inverter. The next step is to control the current drawn from the inverter under test in such a manner as to match the current which results from the real-time simulation when driven from the inverter under test. The inverter under test is no longer connected to a real machine, it is, in fact, connected to a Virtual Machine.

The current drawn from the inverter under test is controlled by the use of a power electronics converter connected to the output of the inverter under test via link inductors. Figure 7.3.1 shows the overall block diagram of the Virtual Machine.

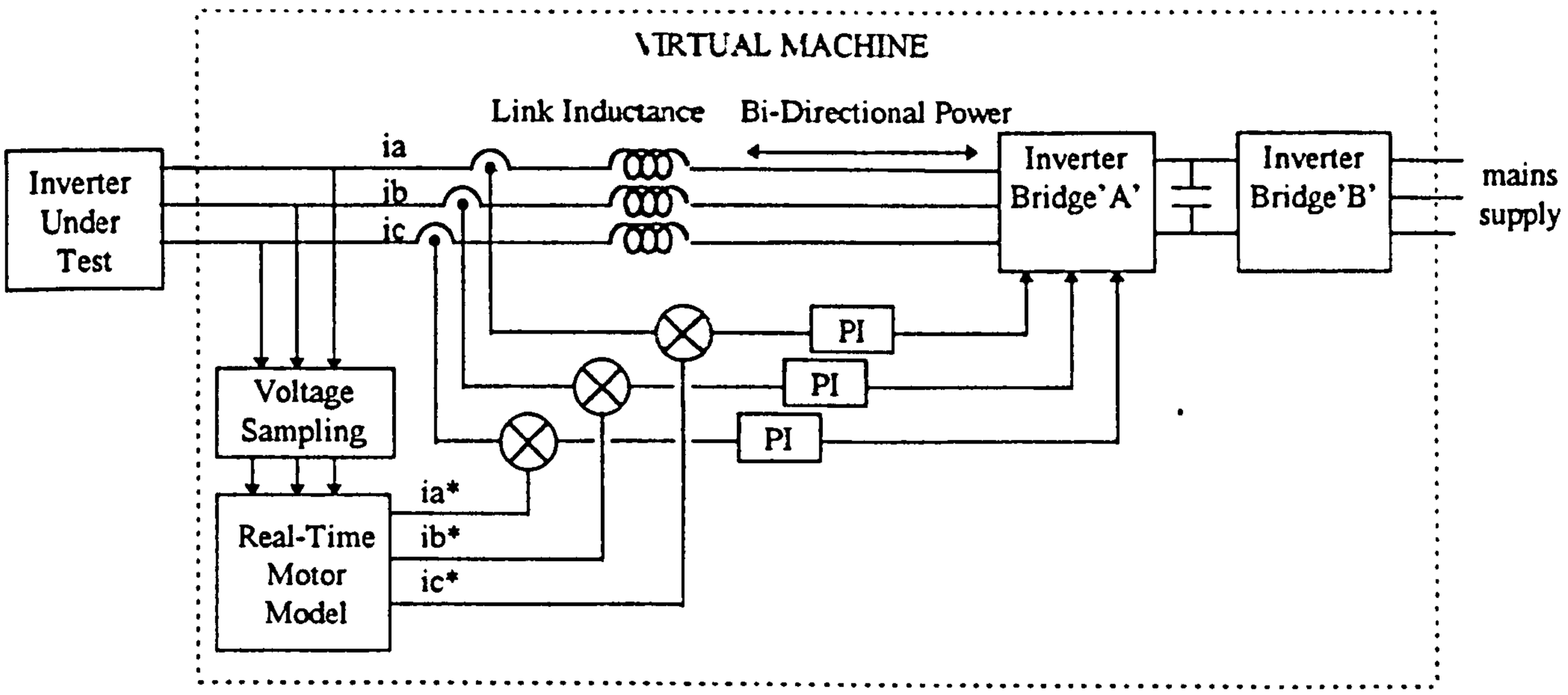


Figure 7.3.1: The Virtual Machine Block Diagram

The power electronics of the Virtual Machine, marked in the block diagram of figure 7.3.1 as inverter bridges A and B, simply consists of a three phase inverter of standard design and interfaces to the inverter under test as follows:-

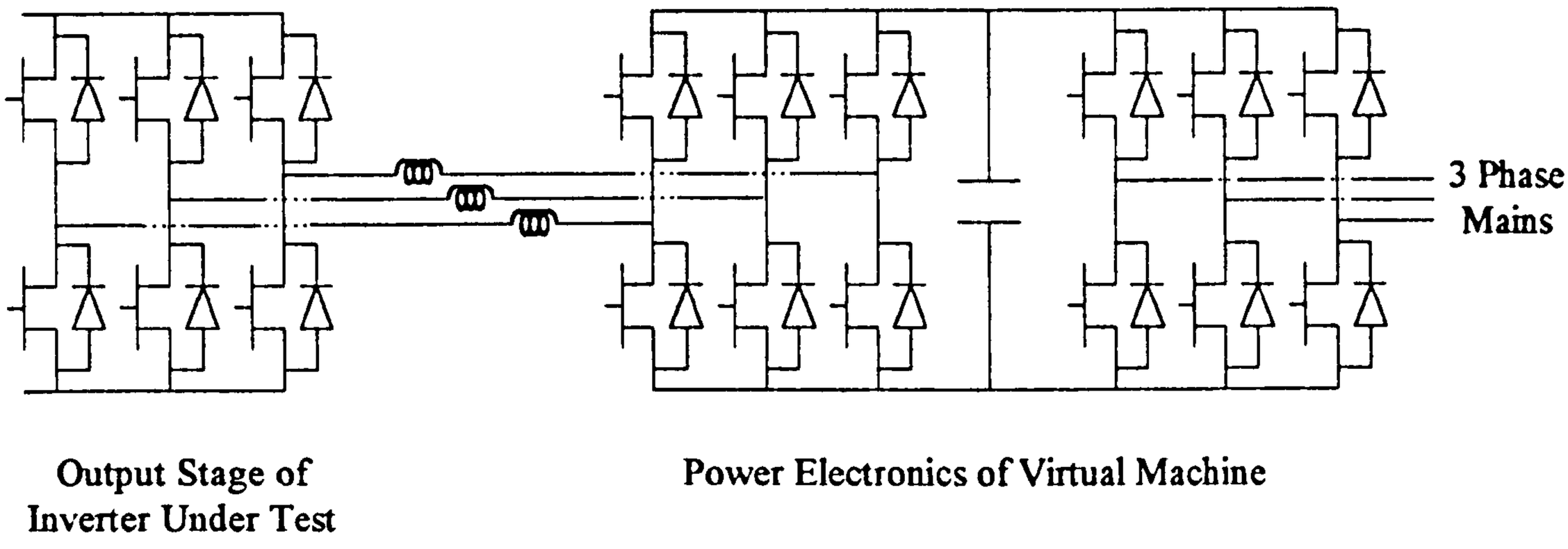


Figure 7.3.2: Power Electronics of the Virtual Machine

The active mains interface of the Virtual Machine allows power flow back into the mains supply.

The overall system works as follows, the first processor reads the values of the sampled voltages, these are then past to the second processor. The second processor applies the sampled voltages to the motor model and solves the equations of the model for one time step, the resultant D and Q axis stator currents are then returned to the first processor to be used as demands for the actual current. The first processor also reads in the actual values of the three phase currents, it can then control the voltage demand given to the power electronics of the Virtual Machine to force the actual currents drawn from the inverter under test to match the demanded currents received from the motor model.

7.3.1 Results Of Computer Simulation Of The Virtual Machine

Figure 7.3.3 shows a Matlab/Simulink block diagram which was used to simulate the Virtual Machine. The line outputs of the inverter under test are connected to three inductors, the other side of which are connected to the power electronics of the Virtual Machine. The sampling system samples the line to line voltages of the inverter under test and then uses these to drive the real-time simulation. The PID Controller block then controls the power electronics of the Virtual Machine so that the current which is actually drawn from the inverter under test matches the current predicted by the real-time simulation.

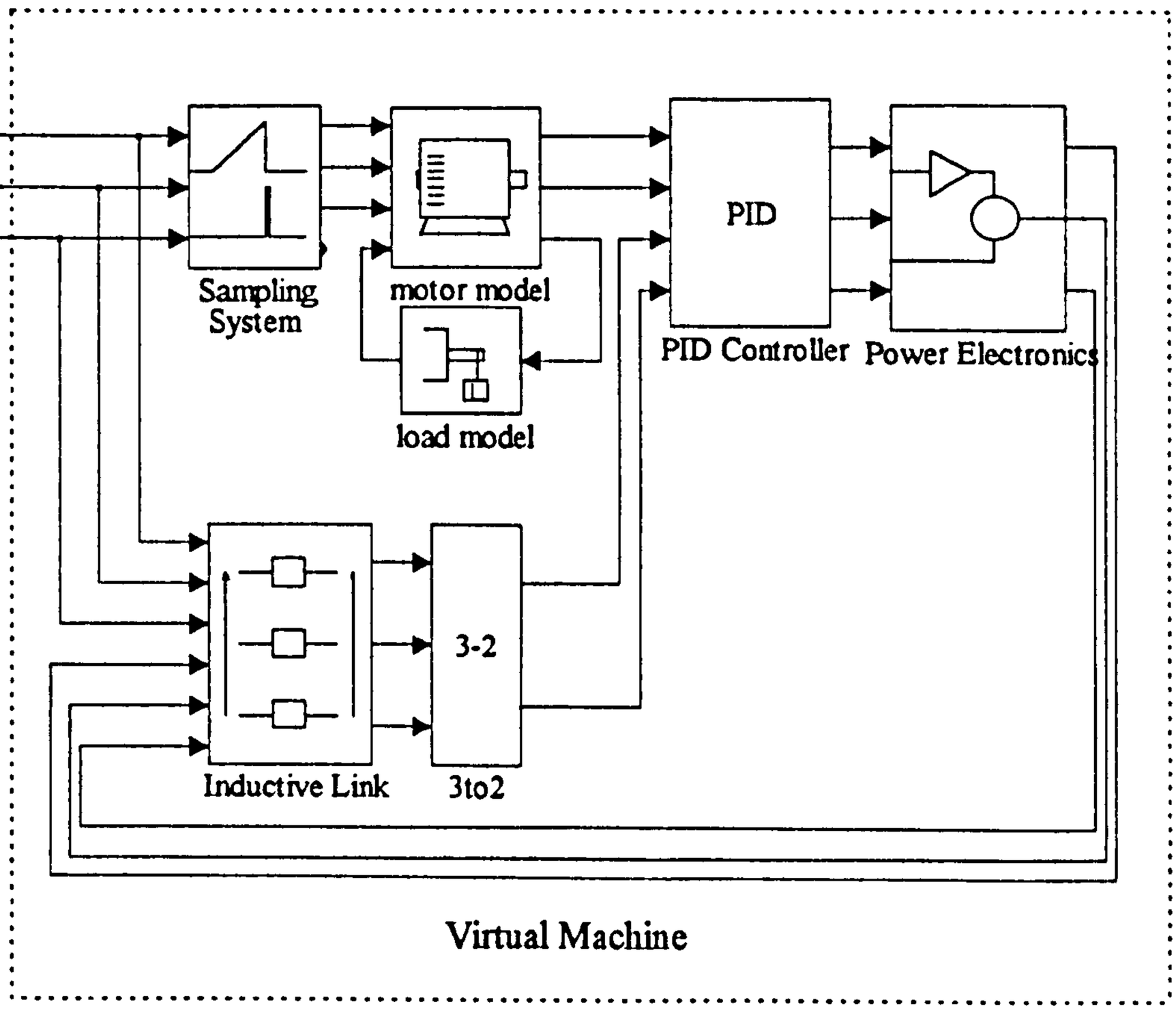


Figure 7.3.3: Matlab/Simulink Simulation of the Virtual Machine

Figure 7.3.4 shows the predicted phase current from the motor model driven from the sampled voltage, and the resultant phase current which occurs when the inverter under test is connected to the Virtual Machine. For comparison purposes the current which results when the inverter under test is connected to a simulation of a real machine is also shown in figure 7.3.5. The inverter under test was initially chosen to have a switching frequency of 9KHz and the line inductors of the Virtual Machine were 40mH.

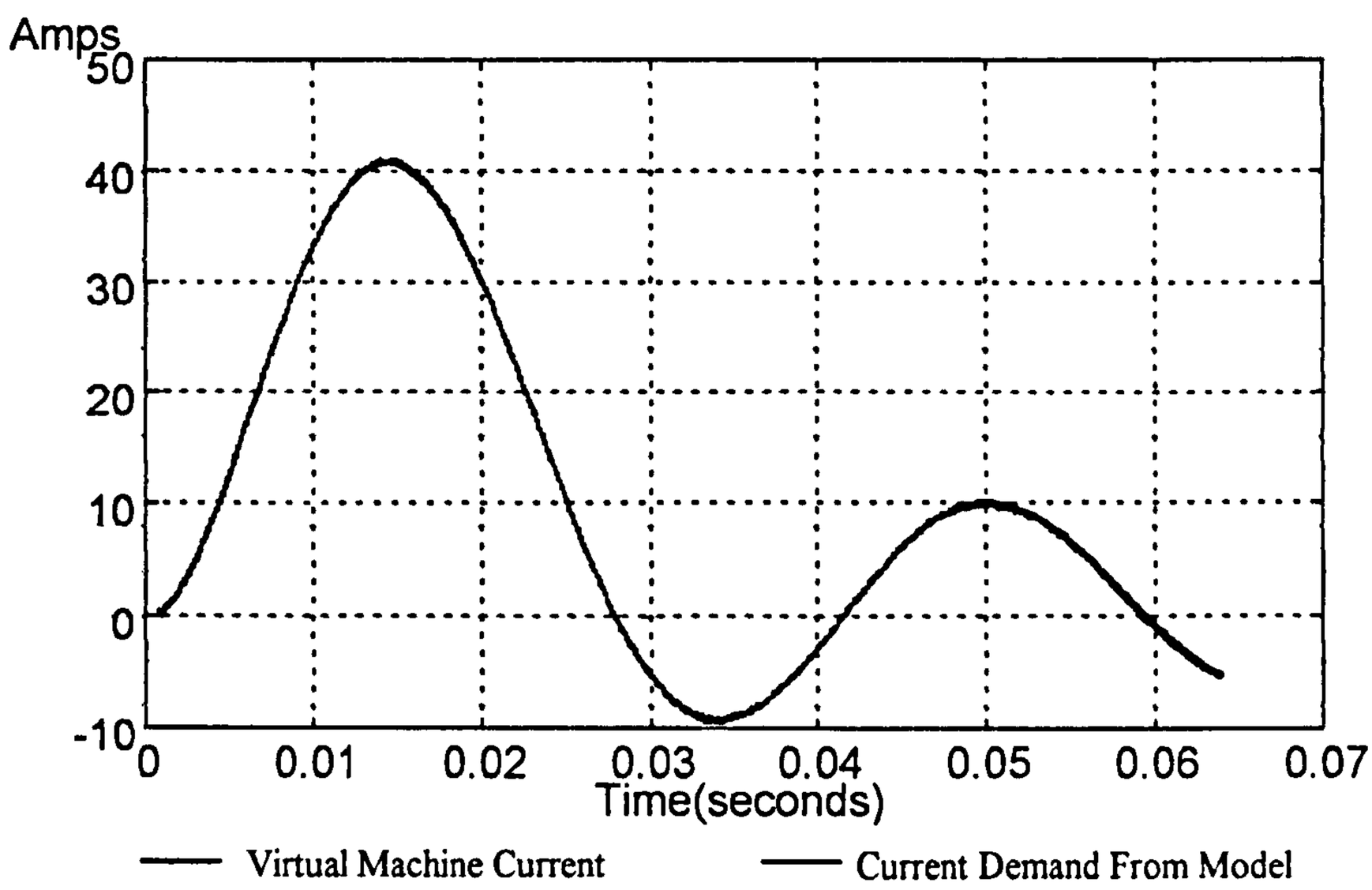


Figure 7.3.4: Current Demand from Model and Actual Virtual Machine Current

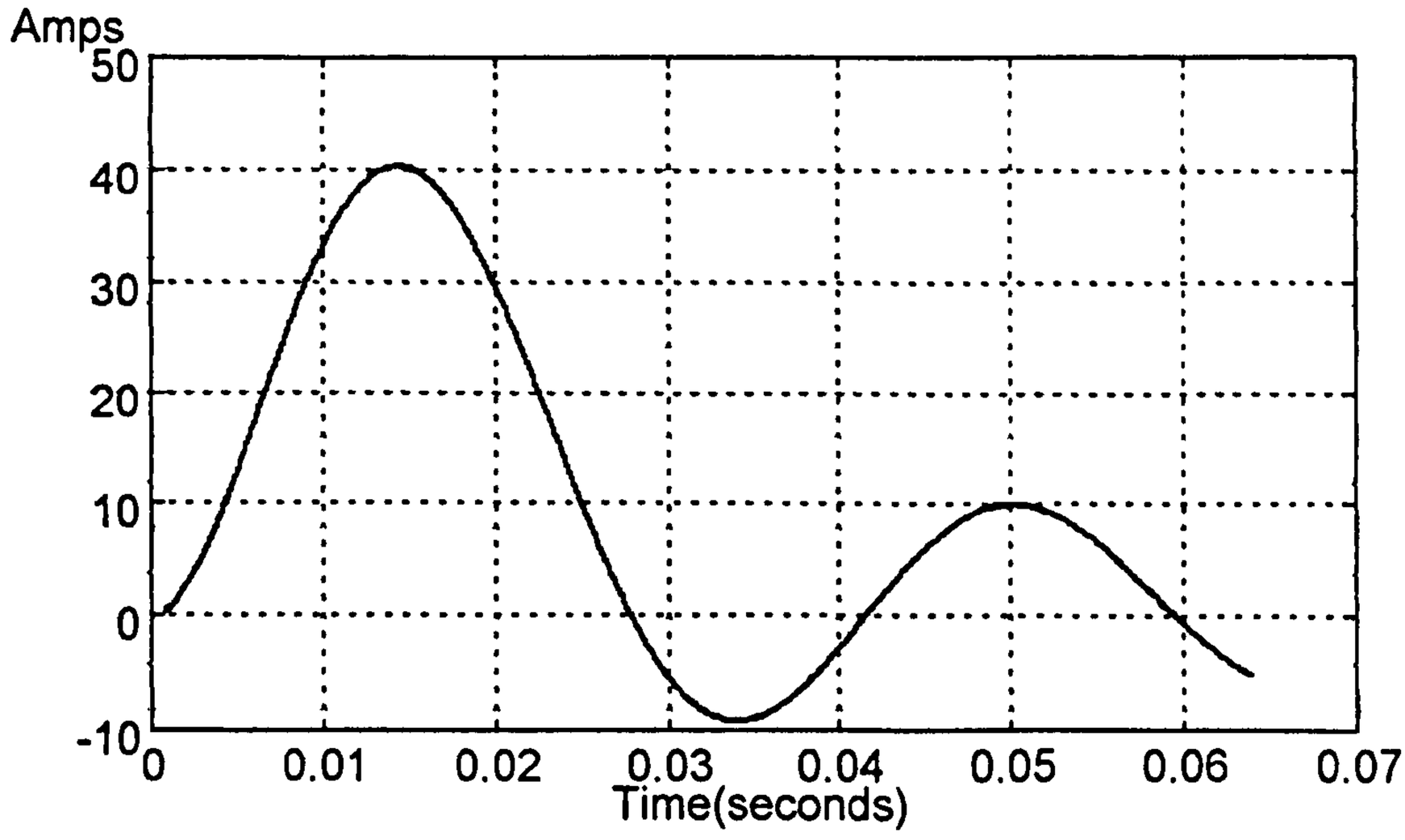


Figure 7.3.5: Current from Simulation of Real Machine

Figures 7.3.6 to 7.3.11 take a closer look at the above currents together with the voltages that cause them. The voltage applied to the inverter under test side of the link inductors has been designated as the primary voltage, and the voltage applied to the remaining side of the link inductors by the power electronics of the Virtual Machine has been designated as the secondary voltage.

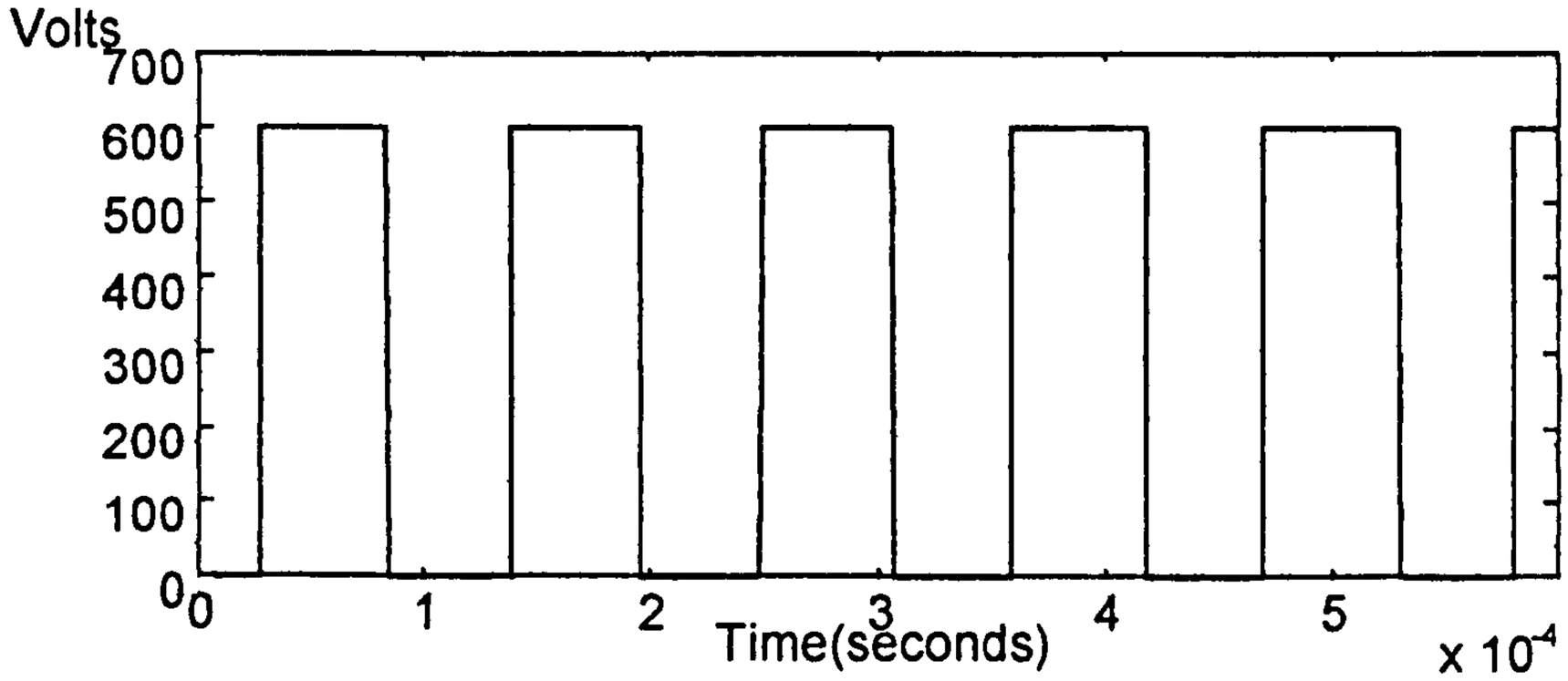


Figure 7.3.6: Primary Voltage Applied by Inverter Under Test

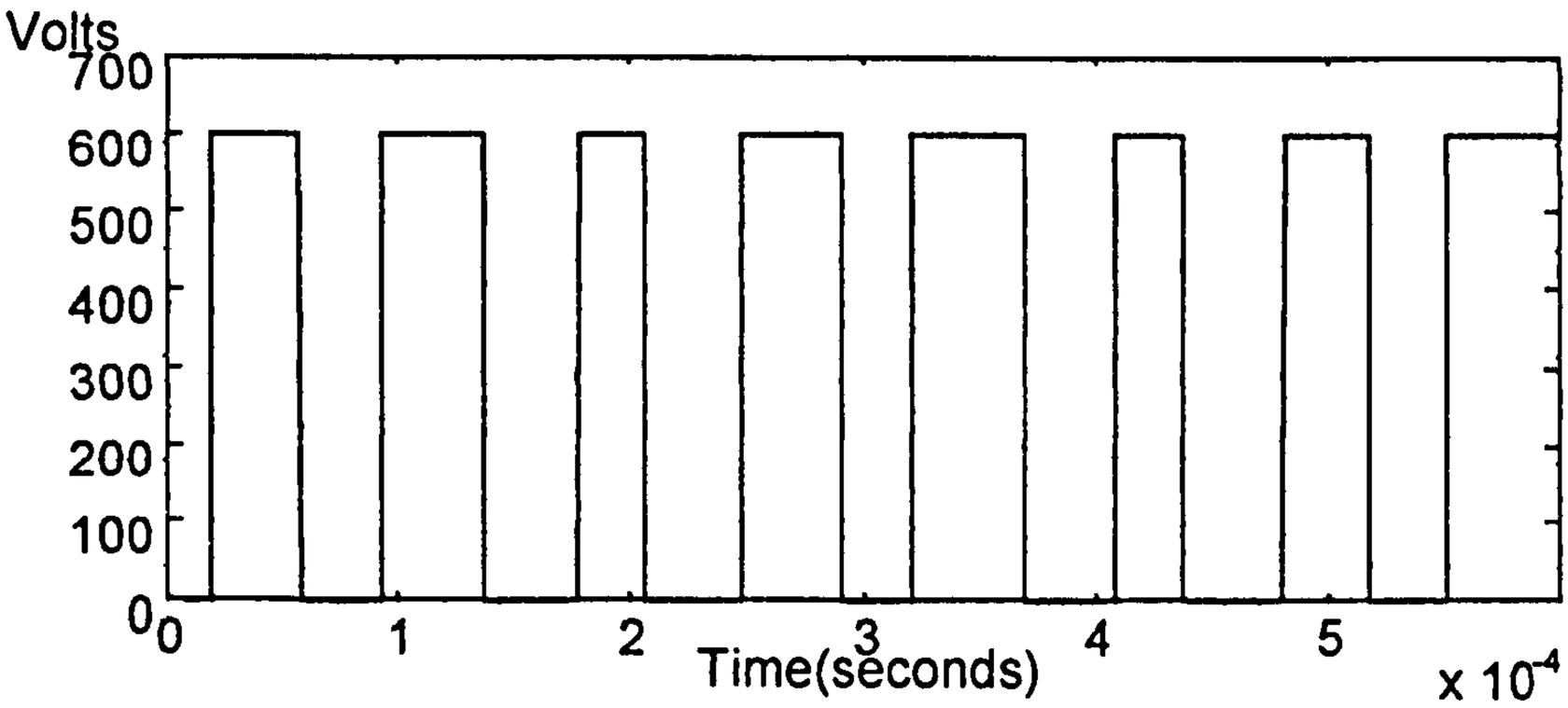


Figure 7.3.7: Secondary Voltage Applied by Virtual Machine

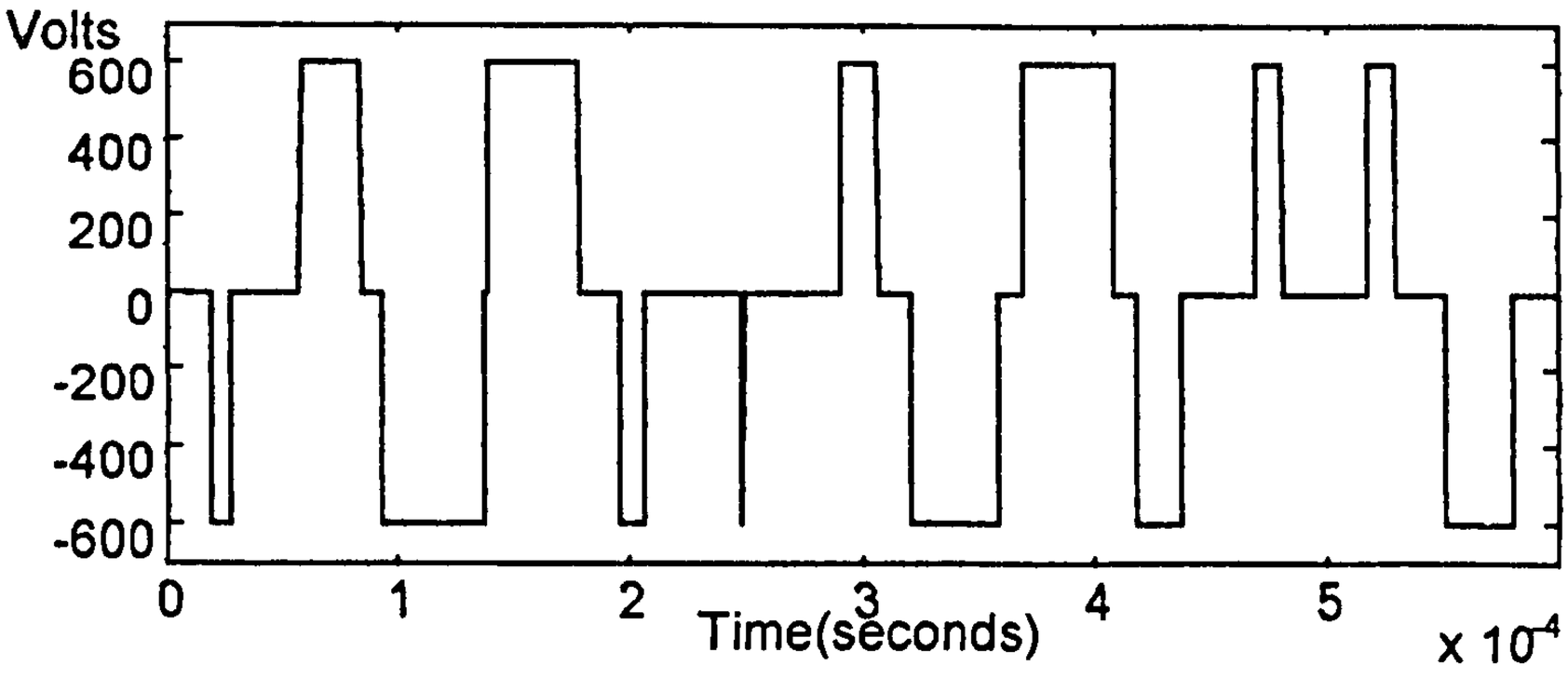


Figure 7.3.8: Net Voltage Across Link Inductor

The current from the model of the machine which is used as the demand is as follows:-

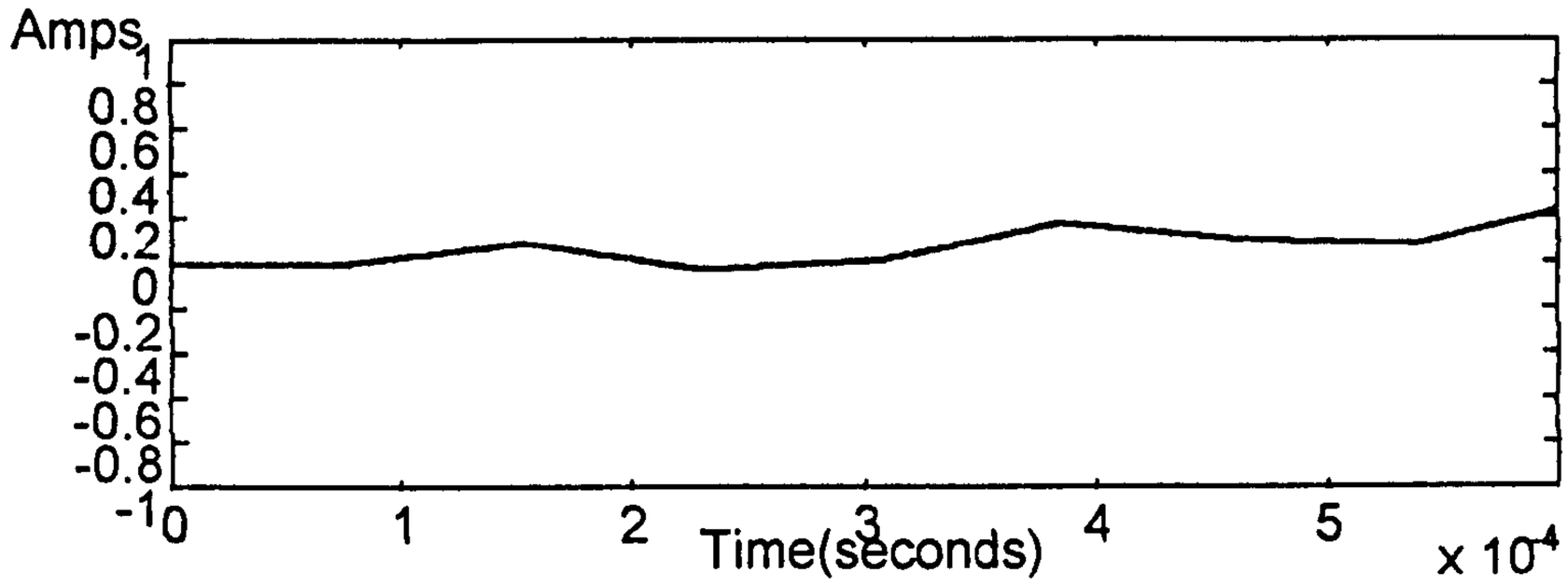


Figure 7.3.9: Current Demand from Model fed from Sampled PWM

The current which actually flows through the link inductor due to the net voltage across it, which was shown in Fig. 7.3.8, is as follows:-

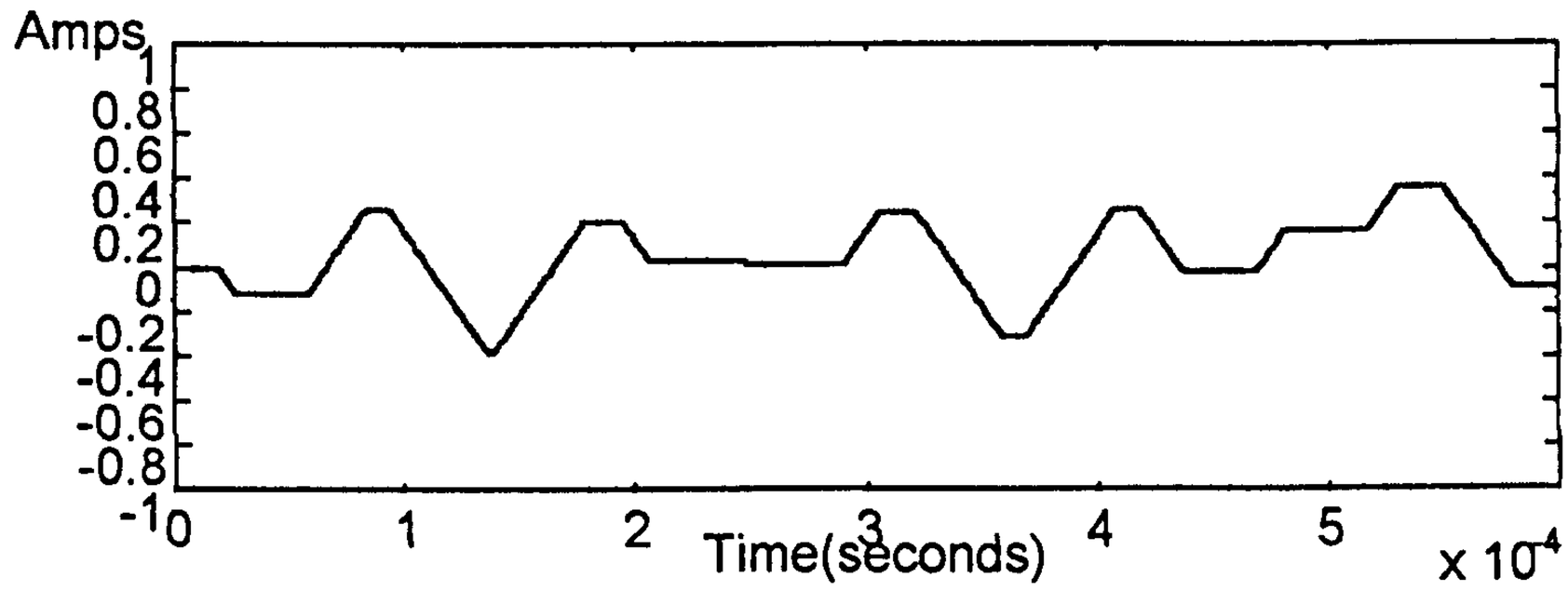


Figure 7.3.10: Current Through Link Inductor

This can be compared to the current which occurs if the inverter under test is connected to a real machine:-

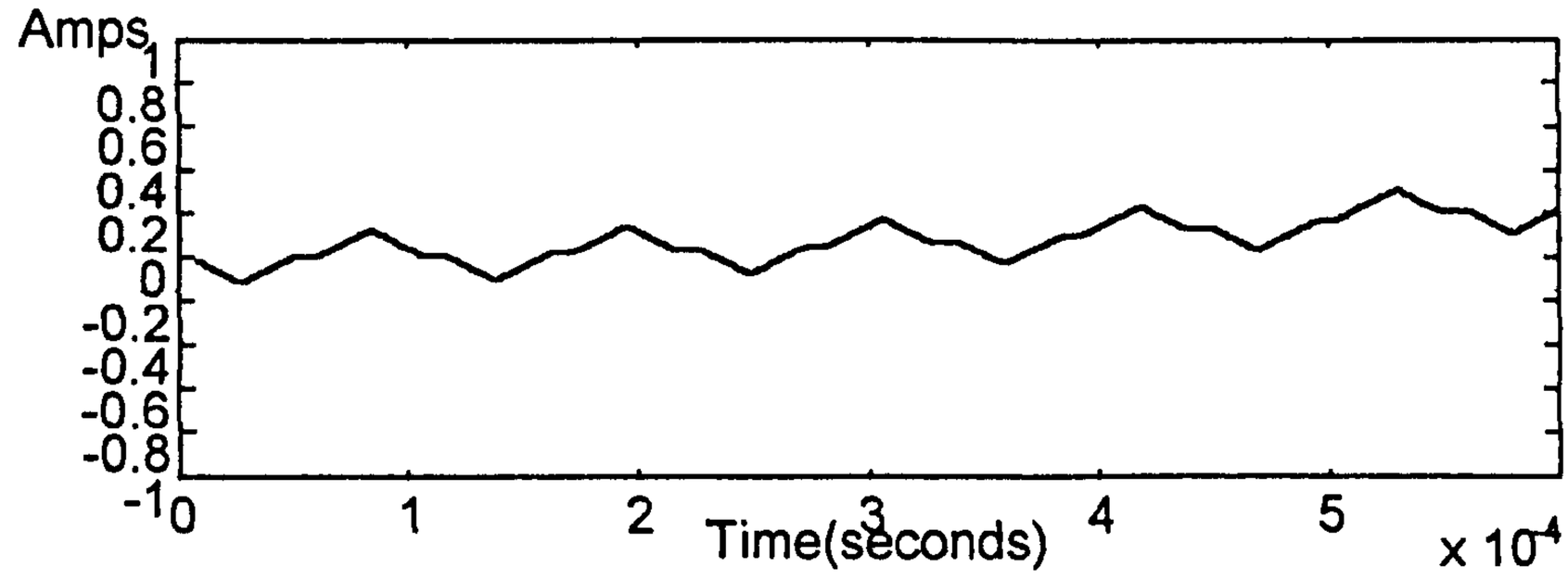


Figure 7.3.11: Phase Current for Inverter Under Test Connected to Real Machine

If the demanded current of figure 7.3.9 is compared to the Virtual Machine Current of figure 7.3.10 and the actual machine current of figure 7.3.11 it can be seen that the Virtual Machine current exhibits some of the switching pattern of the real machine and does not follow the smoother demanded current. This occurs even though the Virtual Machine is trying to control this current to match the value obtained from the model of the machine because the current through the link inductors is affected directly by the real PWM voltage of the inverter under test. The current waveform with the Virtual Machine, however, does not match the current with the real machine because the switching pattern of the virtual machine also has an effect on the waveform.

Comparing the current for the Virtual Machine, figure 7.3.10, with the current for the actual machine, figure 7.3.11, it can be seen that the Virtual Machine current exhibits a faster ramp rate. The ramp rate of the Virtual Machine current is dictated by the link inductance. The inductance's that link the inverter under test to the Virtual Machine have two major effects on the performance of the Virtual Machine. Firstly they directly affect the response of the current control of the Virtual Machine, for this reason small values of inductance would seem necessary. However, they also affect how the current ripple of the current drawn from the inverter under test. is made up. When the inverter under test is connected to a real machine the current contains a ripple which is caused by the switching action of the inverter under test and so is dependent on the switching frequency of the inverter under test. When the inverter under test is connected to the Virtual Machine the current now contains a ripple which is dependent upon the switching frequency of the Virtual Machine as well as that of the inverter under test. For this reason larger values of inductance mean that the switching frequency of the Virtual Machine affects the inverter under test to a lesser extent. Figure 7.3.12 shows the same test with the link inductors increased to 80mH.

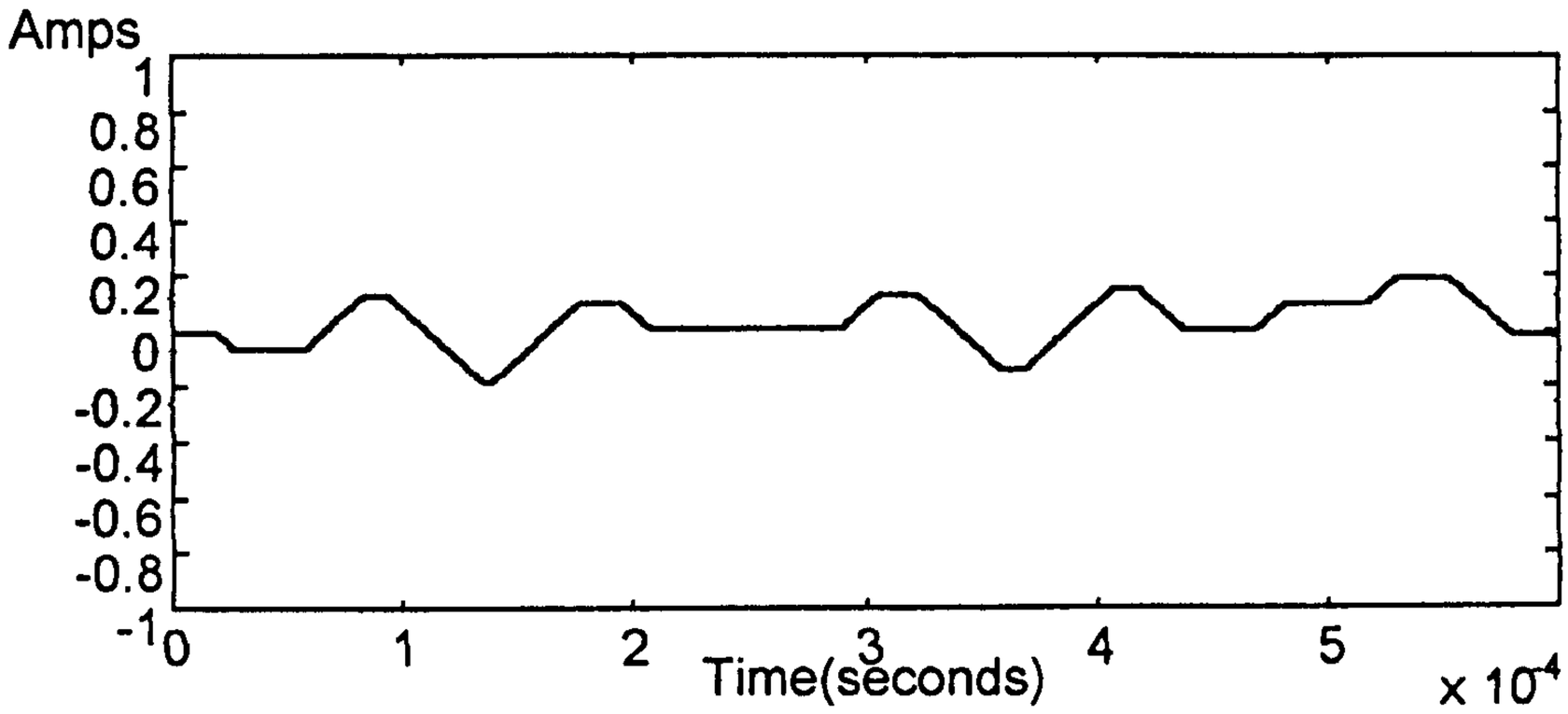


Figure 7.3.12: Current Through Link Inductor, Increased Inductance

Increasing the link inductance has obviously reduced the ramp rate of the current. If this current waveform is overlaid with the current waveform of Fig. 7.3.11 then a direct comparison can be made between the Virtual Machine current and the real machine current. The current waveform is now a closer representation of the actual current drawn by a real machine.

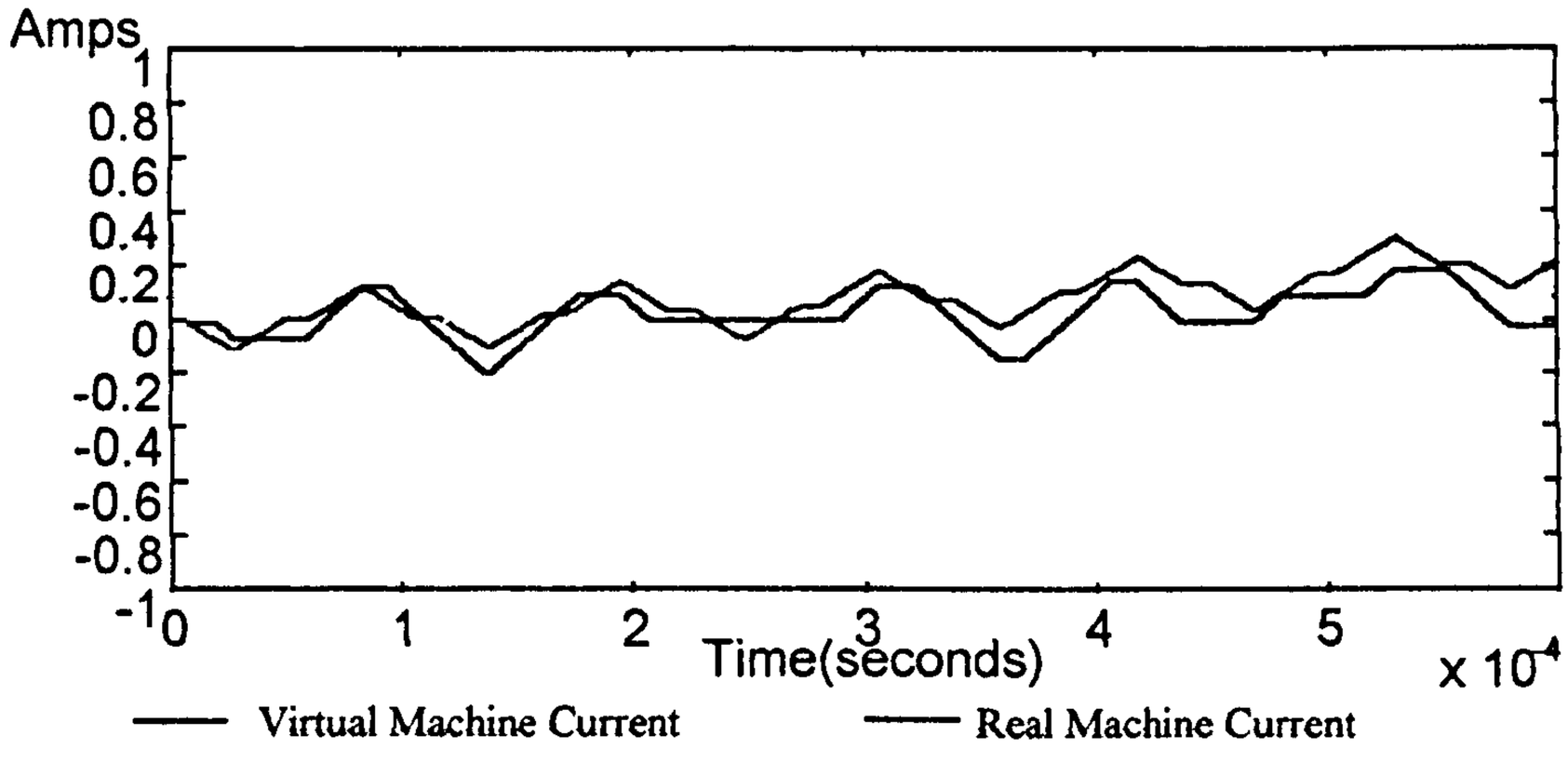


Figure 7.3.13: Comparison of Virtual and Real Machine Current

The previous results investigated the effect of the link inductance on the performance of the Virtual Machine. The following results investigate the effect of the switching frequency of the inverter under test on the Virtual Machine. Two frequencies were considered, figures 7.3.14 to 7.3.16 show results for 3KHz and figures 7.3.17 to 7.3.19 show results for 12KHz.

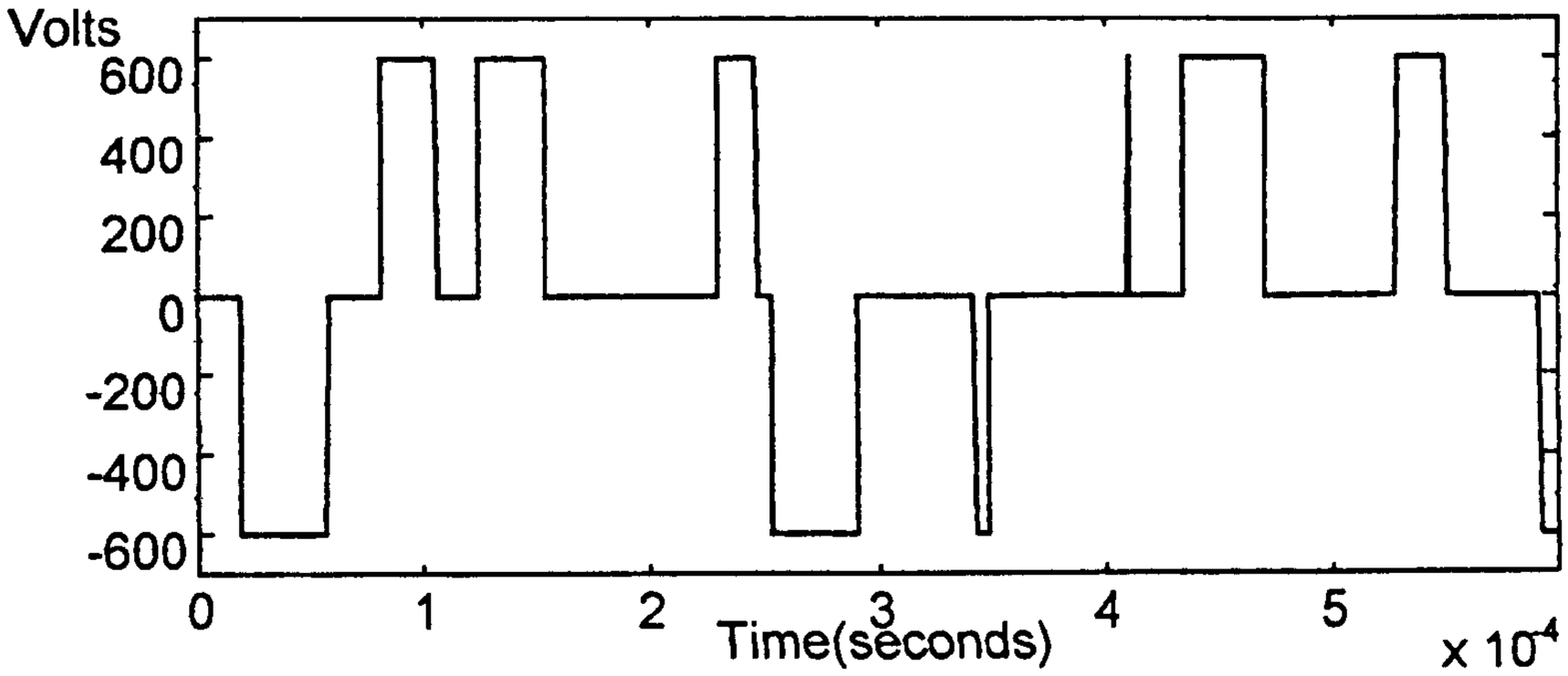


Figure 7.3.14: Net Inductor Voltage, Inverter Under Test Switching at 3KHz

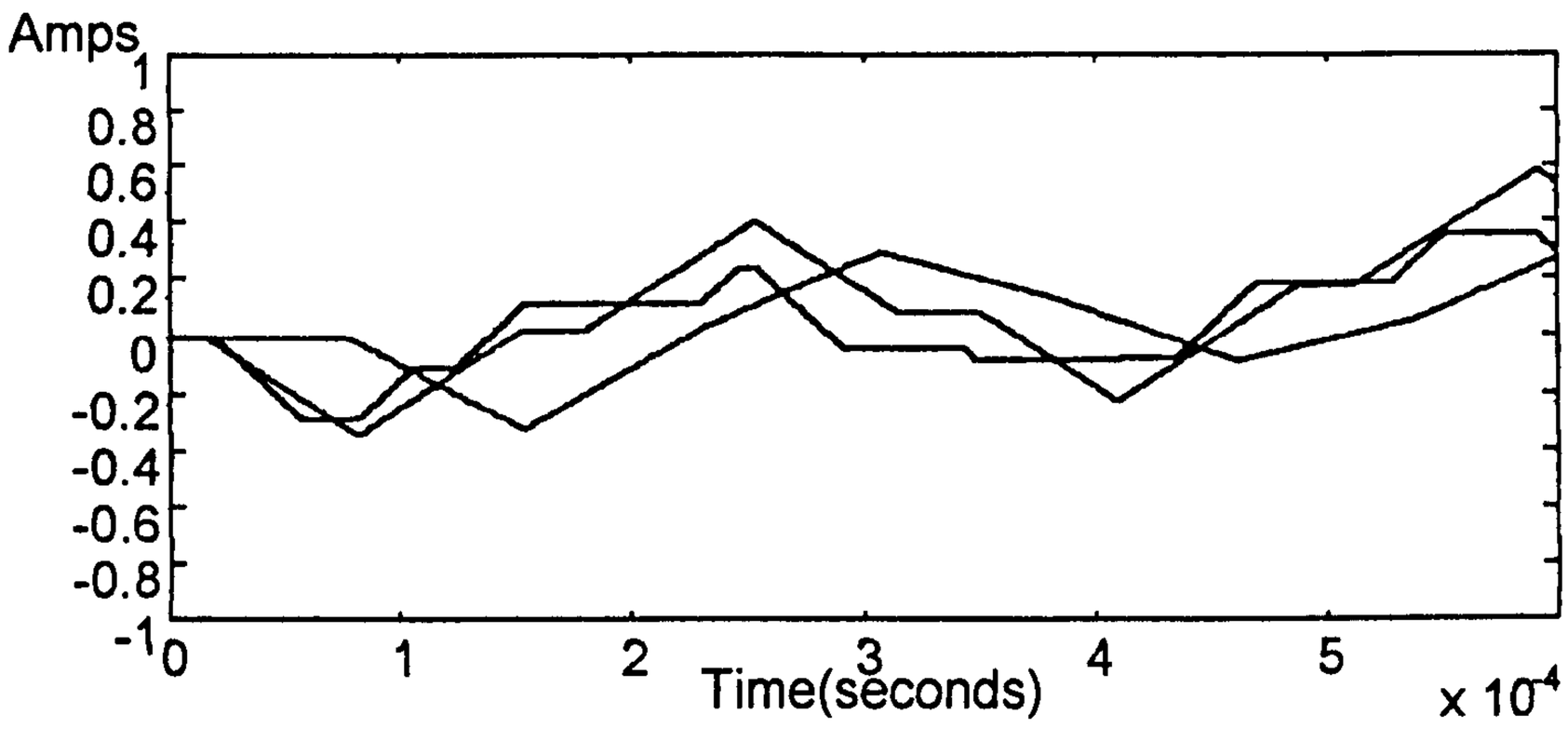


Figure 7.3.15: Phase Currents, Inverter Under Test Switching at 3KHz

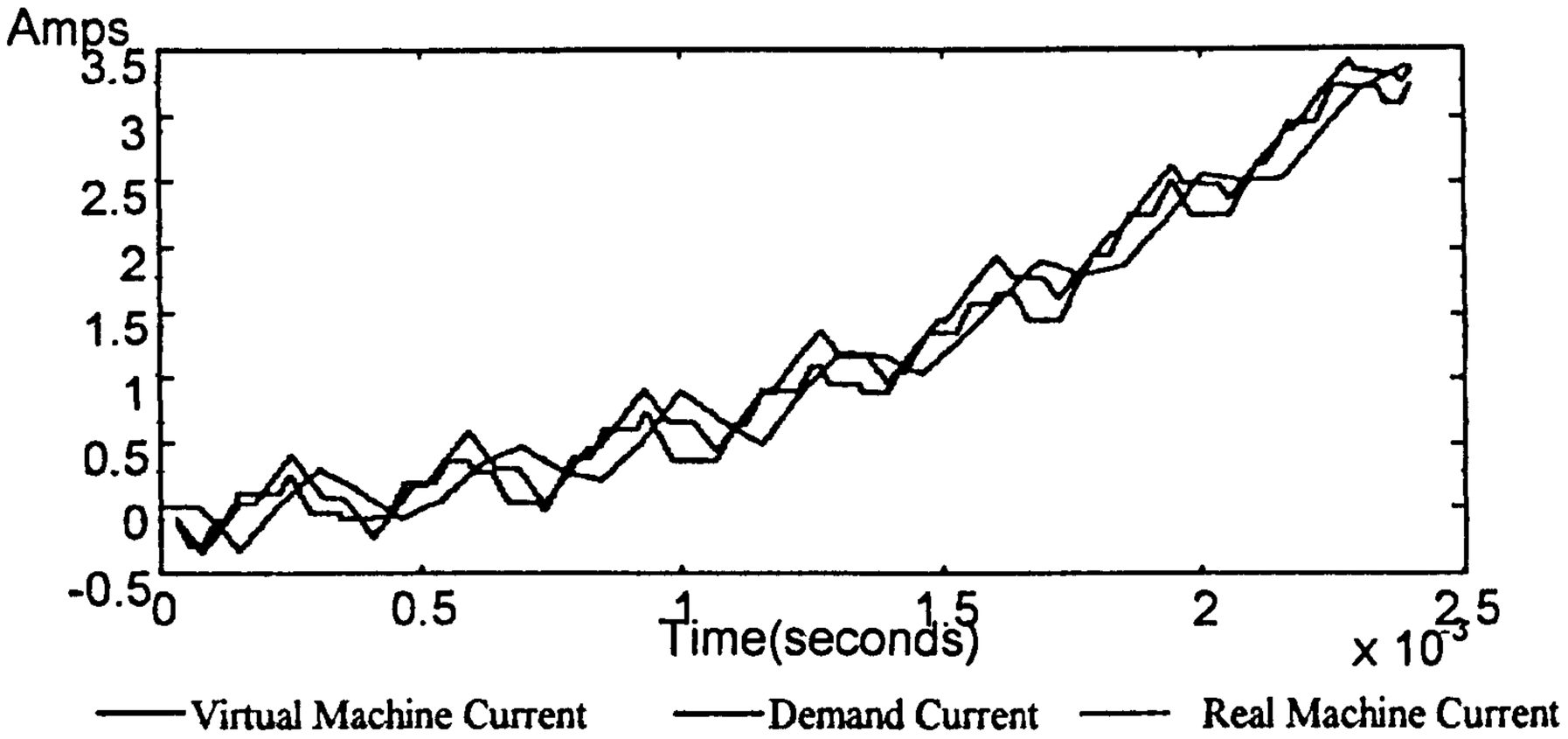


Figure 7.3.16: Phase Currents, Inverter Under Test Switching at 3KHz

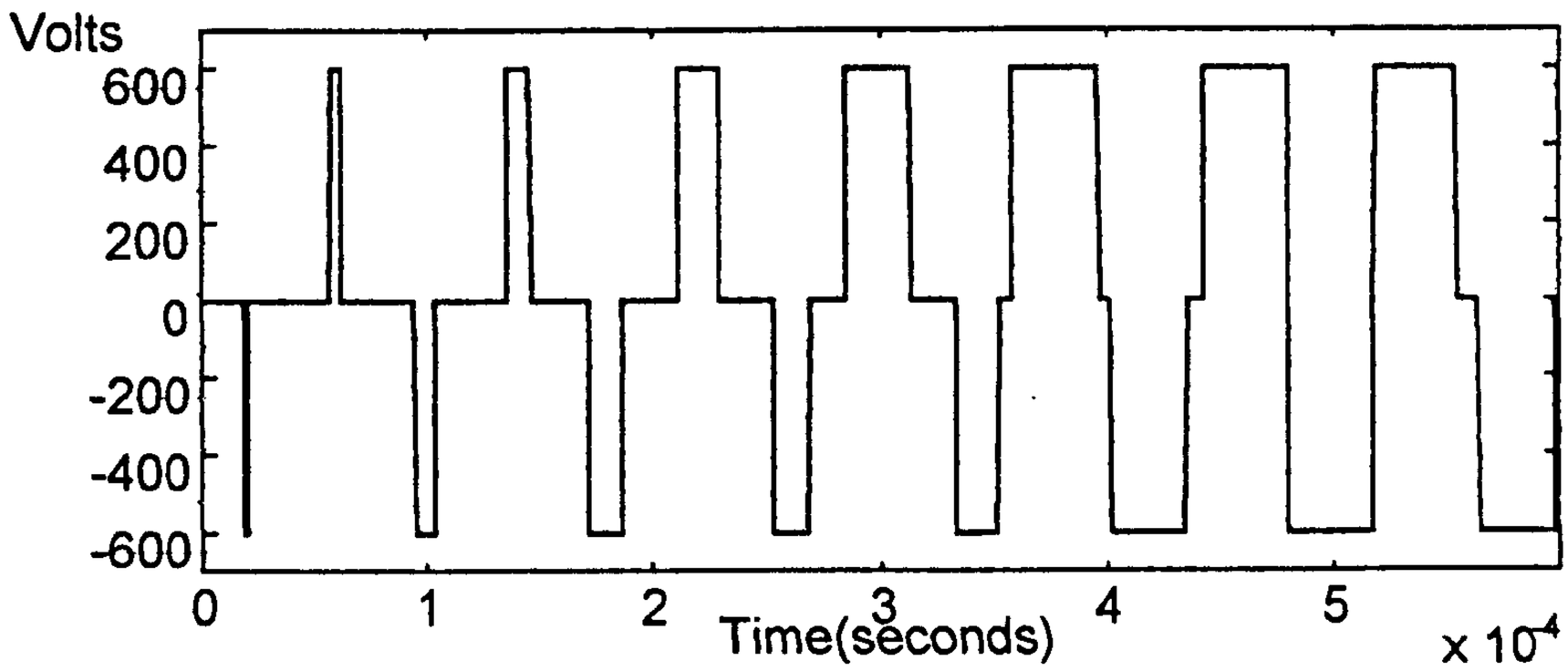


Figure 7.3.17: Net Inductor Voltage, Inverter Under Test Switching at 12KHz

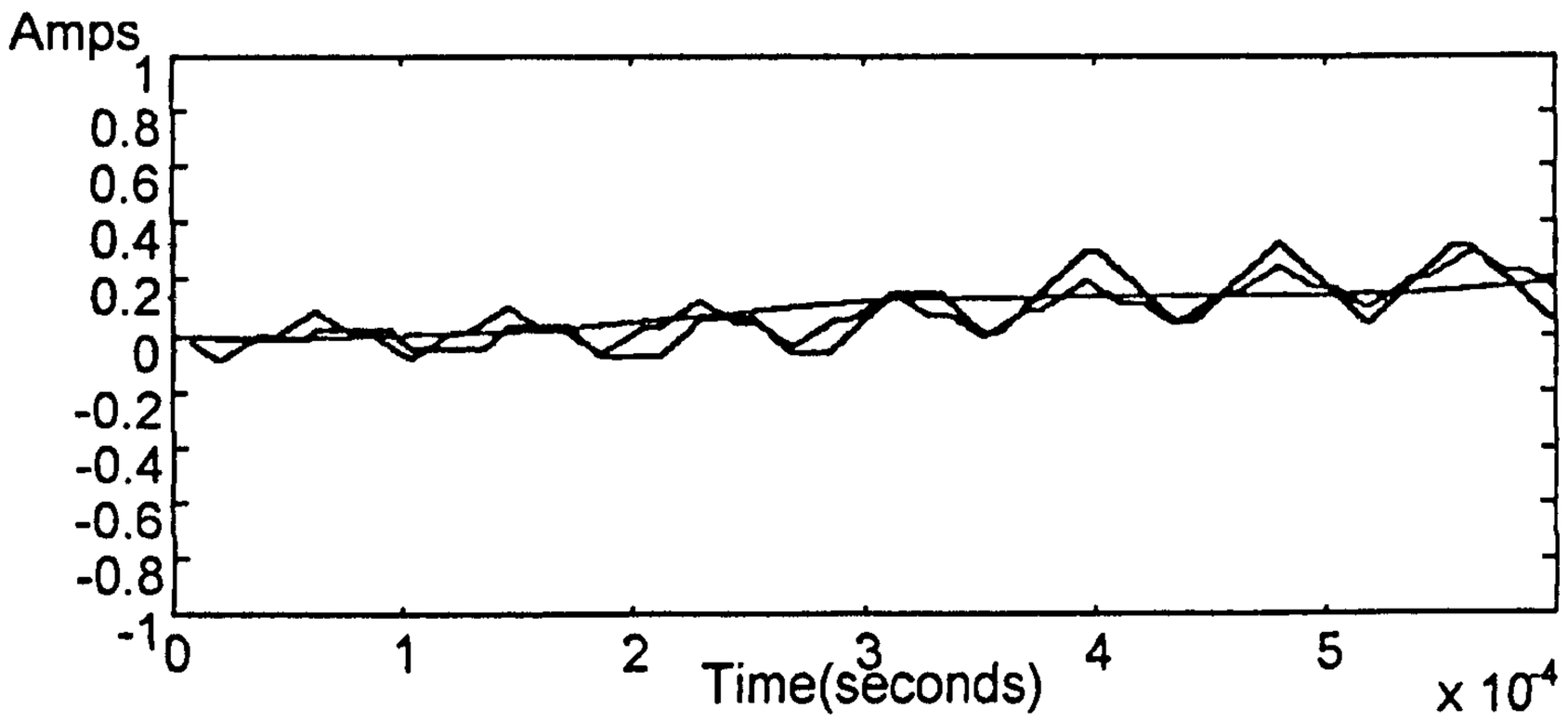


Figure 7.3.18: Phase Currents, Inverter Under Test Switching at 12KHz

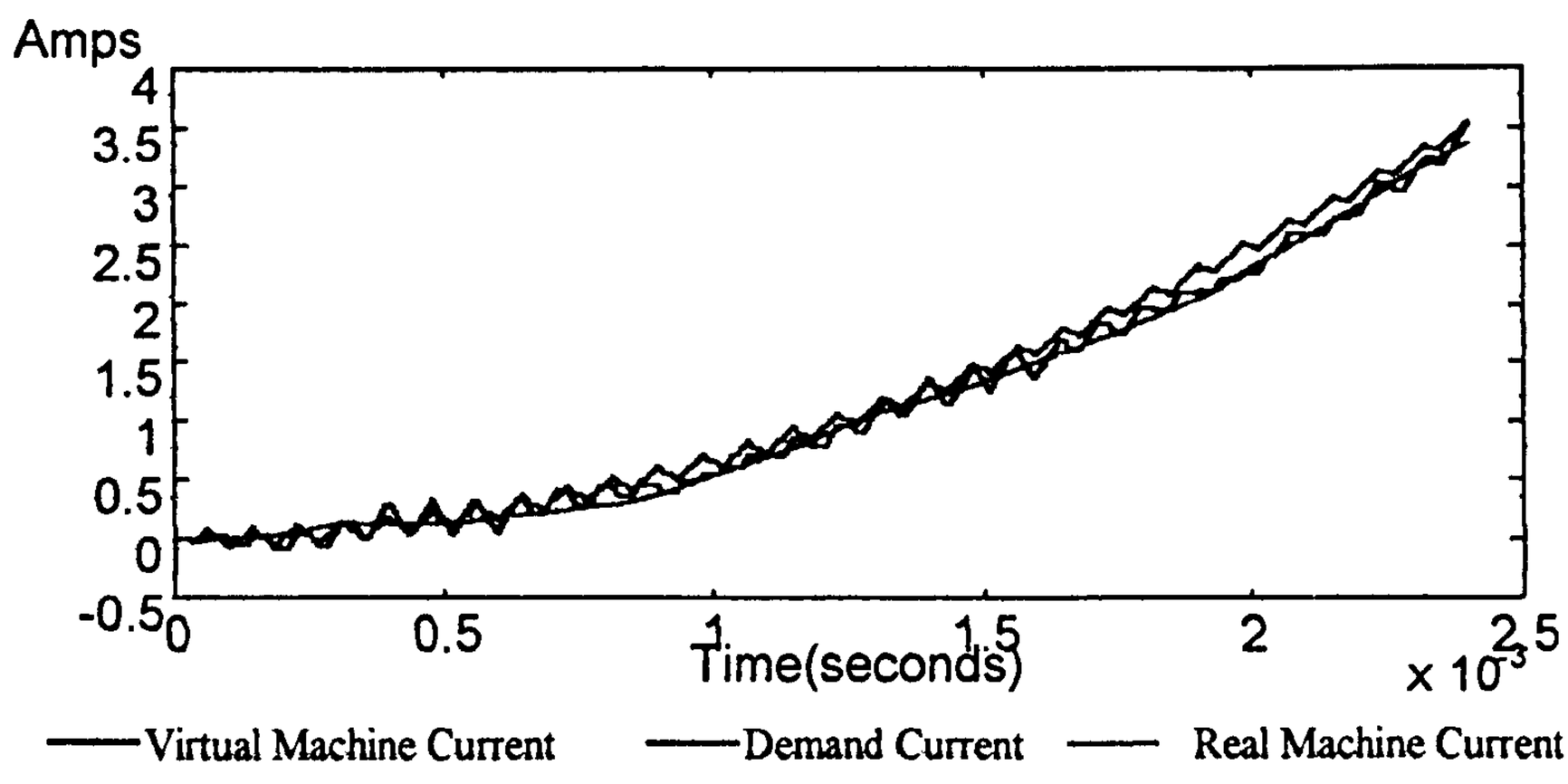


Figure 7.3.19: Phase Currents, Inverter Under Test Switching at 12KHz

The previous results show that the Virtual Machine copes well when the switching frequency of the inverter under test is both low and high. The detailed current plots of figures 7.3.15 and 7.3.18 show that the demanded current waveform obtained from the real-time motor model shows a closer resemblance to the actual machine current when the switching frequency of the inverter under test is low. This is due to the sampling period of the Virtual Machine remaining the same. If the sampling period of the Virtual Machine was infinitely small compared to the switching frequency of the inverter under test then none of the switching pulses of the inverter under test would be averaged out, instead every switching edge would be captured and applied to the motor model, thus producing a current demand which matched the real machine currents exactly. However, because this is not possible with the present hardware, the fact that the Virtual Machine loses some detail of the PWM output of inverter under test must be expected.

7.3.2 Steady State Results Of The Virtual Machine

The following results were taken with the Control Techniques inverter (mentioned earlier) used as the inverter under test. This inverter was connected to the three phase voltage source inverter designed and used for the real time simulation results which was used, in this case, as the Virtual Machine. Three cases of steady state demand are considered, 0Hz ,10Hz and 30Hz.

The inverter under test operates with a DC link voltage of 600 volts, with a frequency demand of zero the inverter switches on the upper switching device in each phase for half the switching period and the lower switching device for the other half of the switching period. Figure 7.3.20 shows one output line voltage of the inverter under test and the corresponding output line of the Virtual Machine when the inverter under test is initially switched on with a zero demand. The switching frequency of the inverter under test is 12KHz and the switching frequency of the Virtual Machine is 13KHz. These two voltages are connected together through the link inductance.

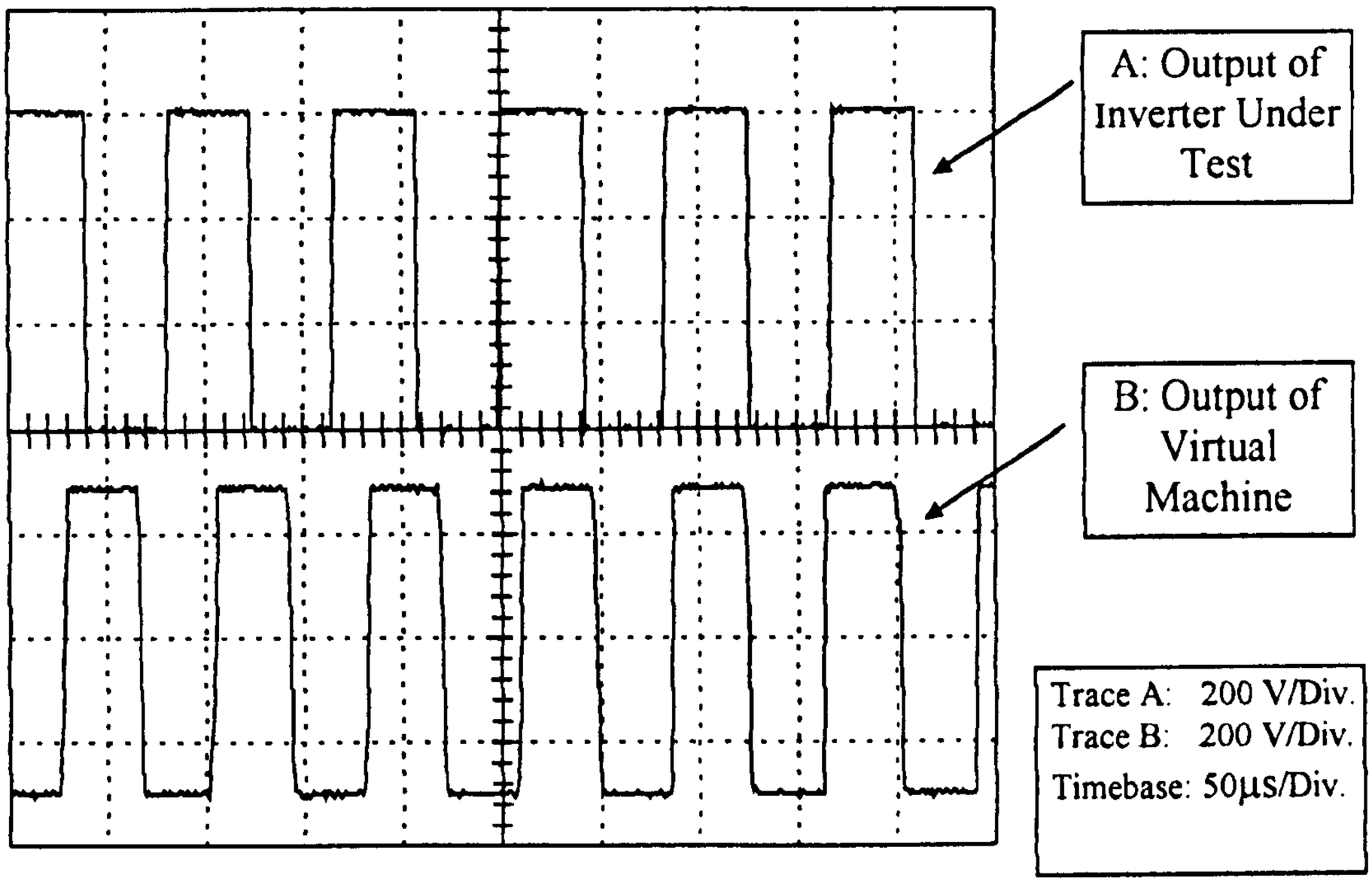


Figure 7.3.20: Voltage from Inverter Under Test and Secondary Voltage Produced by the Virtual Machine

Figure 7.3.21 shows the current demand produced from the real-time model of the machine fed from the sampled output of the inverter under test, and the actual current that is drawn from the inverter under test.

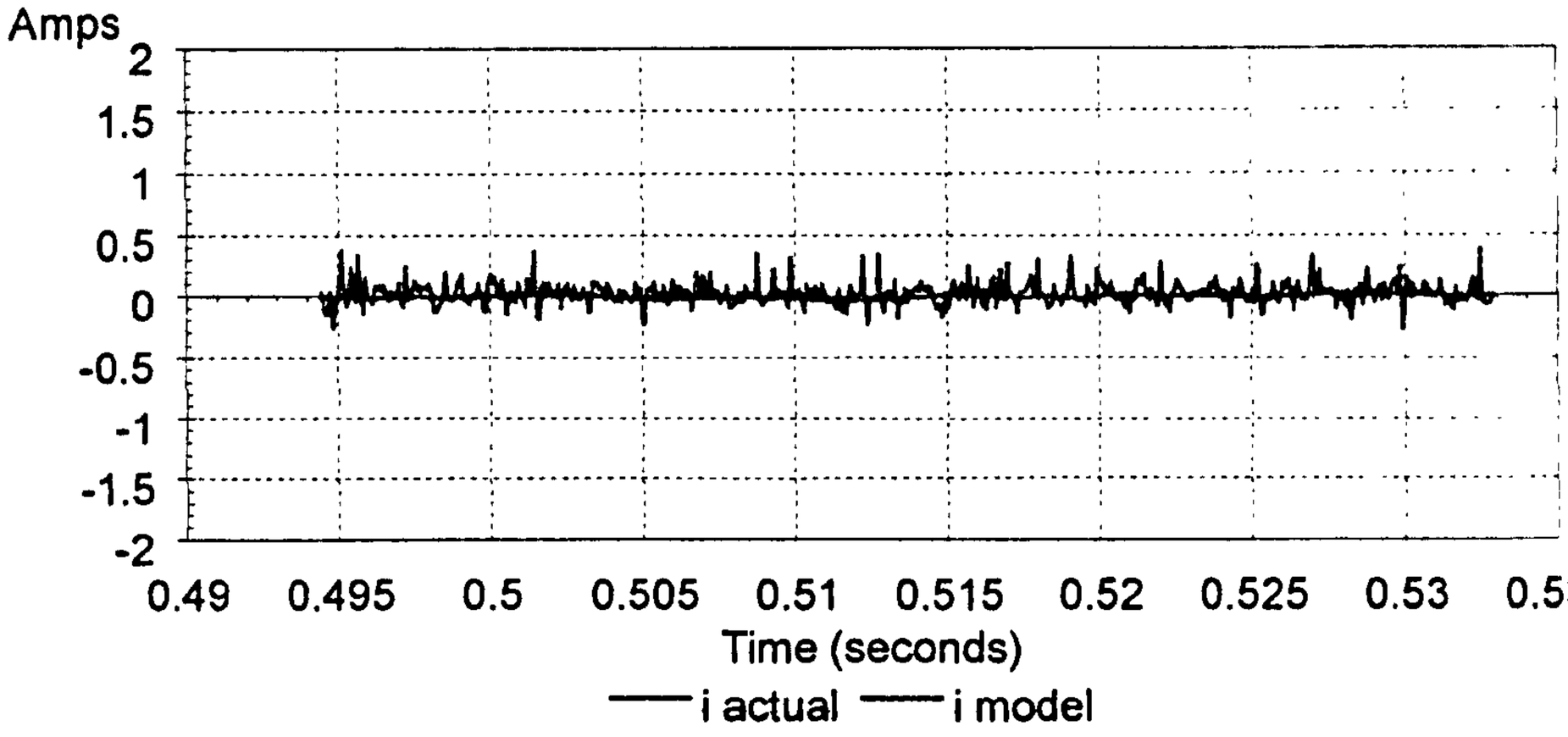


Figure 7.3.21: Demand / Actual Current for 0Hz Demand to Inverter Under Test

With a demand of 0Hz the motor model current is 0A, figure 7.3.21 shows how the Virtual Machine controls the actual current drawn from the inverter under test to be

approximately 0A. Figure 7.3.22 shows detail of the actual current waveform and the net voltage across the link which causes this current, i.e. the voltage produced by the inverter under test minus the voltage produced by the Virtual Machine.

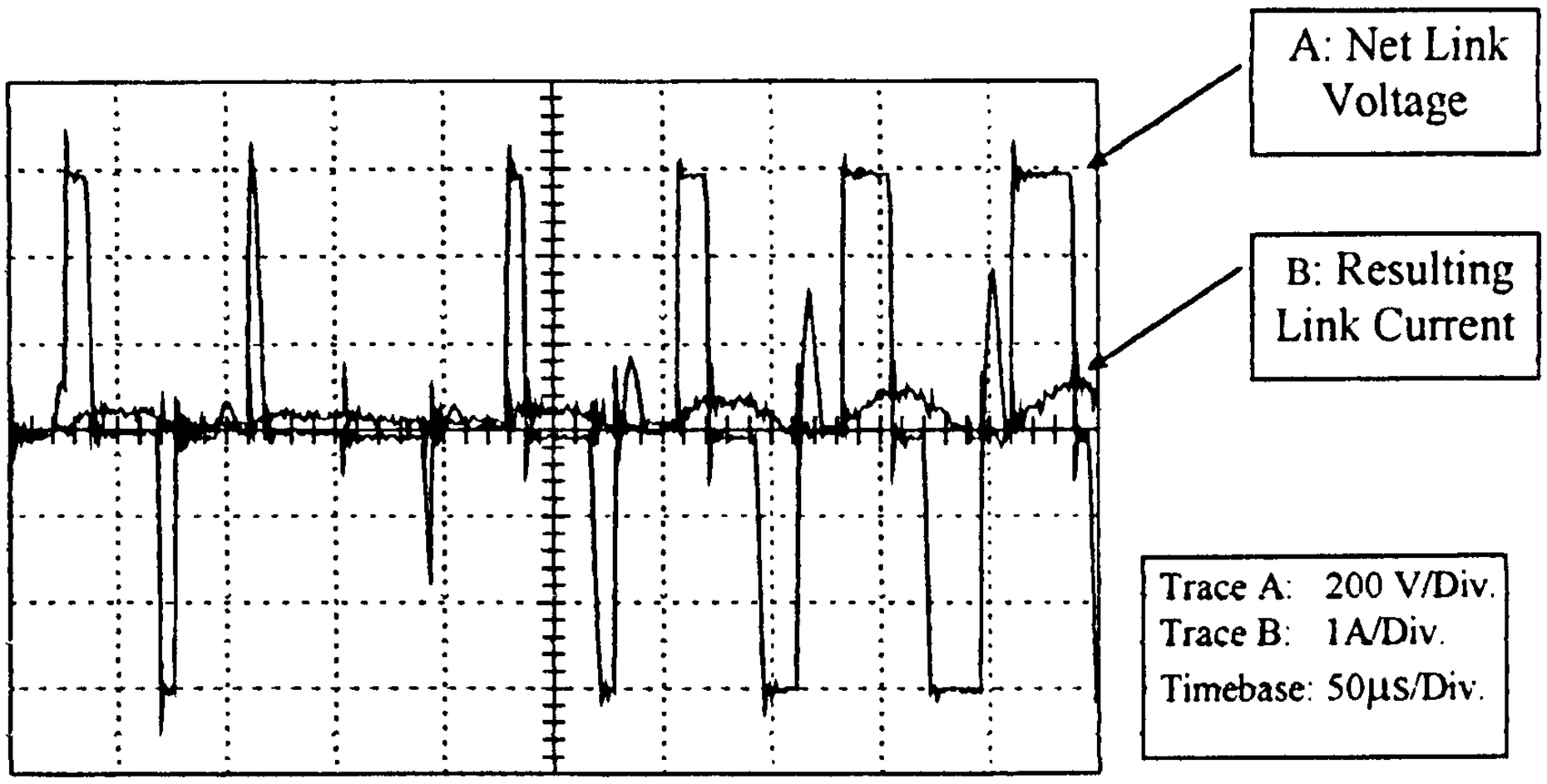


Figure 7.3.22: Net Voltage Across Link and Resultant Current 0Hz Demand

The frequency demand to the inverter under test was increased and the output voltage of the inverter became a modulated sinusoid. The motor model responded to this voltage and produced a current demand for the Virtual Machine. The Virtual Machine attempts to control the current taken from the inverter under test to match the current demand produced by the real-time model. In order to compare the behaviour of the Virtual Machine to that of a real machine two types of results are shown, firstly a comparison of the real machine currents to the real-time motor model currents and secondly the real-time model currents to the Virtual Machine currents. This is included since the Virtual Machine cannot be connected to the inverter under test at the same time as a real machine. Figures 7.3.23 to 7.3.24 show the real machine current and real-time model current for a 10Hz demand to the inverter under test. Figures 7.3.25 and 7.3.26 show the real-time model currents and the Virtual Machine currents for the same 10Hz demand to the inverter under test. The model current tends to lag the actual machine current, this delay is similar to that shown in the simulation results of figure 7.2.4. The delay is introduced by the sampling system, the sampling

system has a frequency of 13KHz which results in a delay of the model current by 77μs.

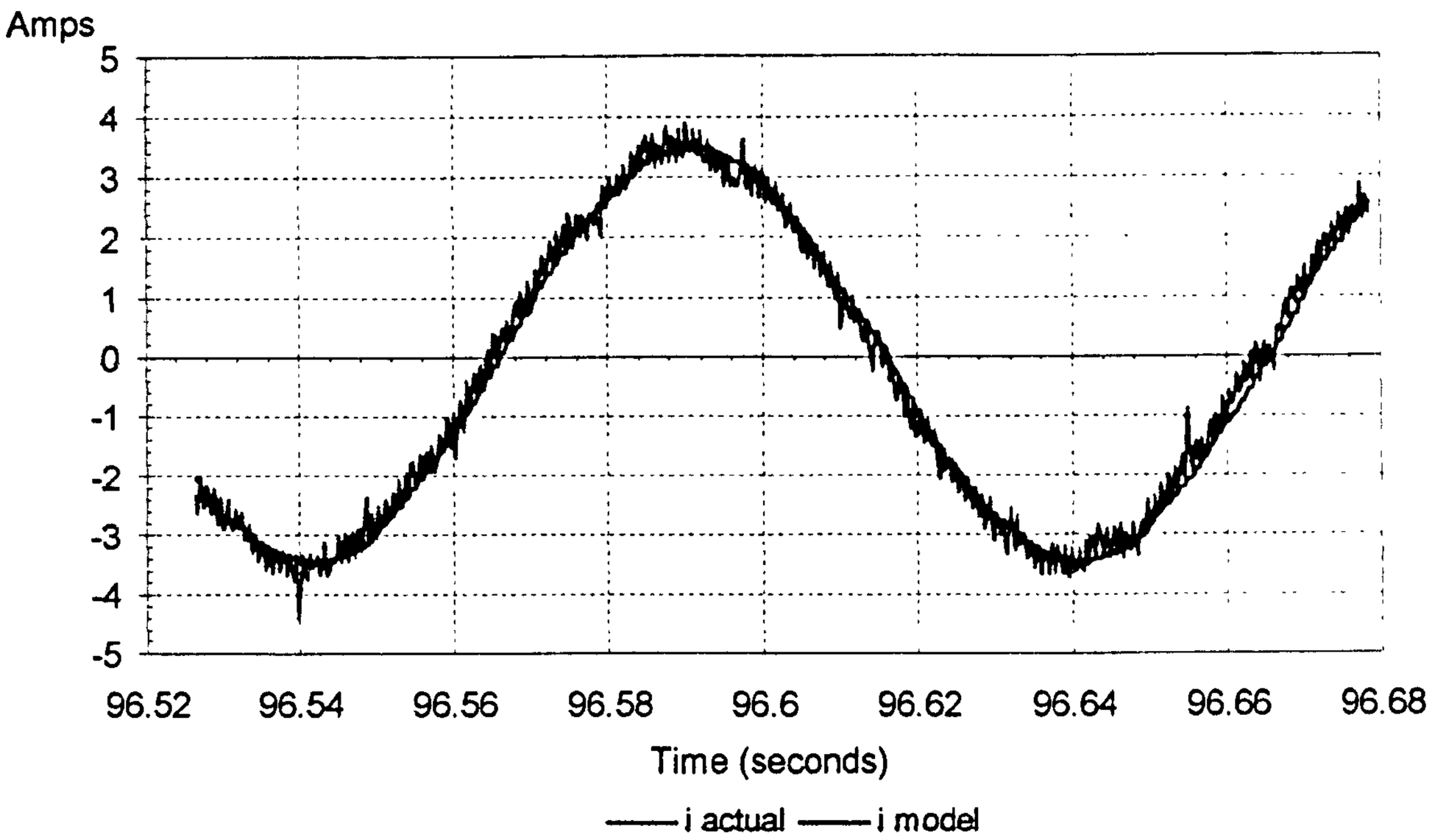


Figure 7.3.23: Actual Machine Current and Real-Time Model Current for 10Hz Demand to Inverter Under Test

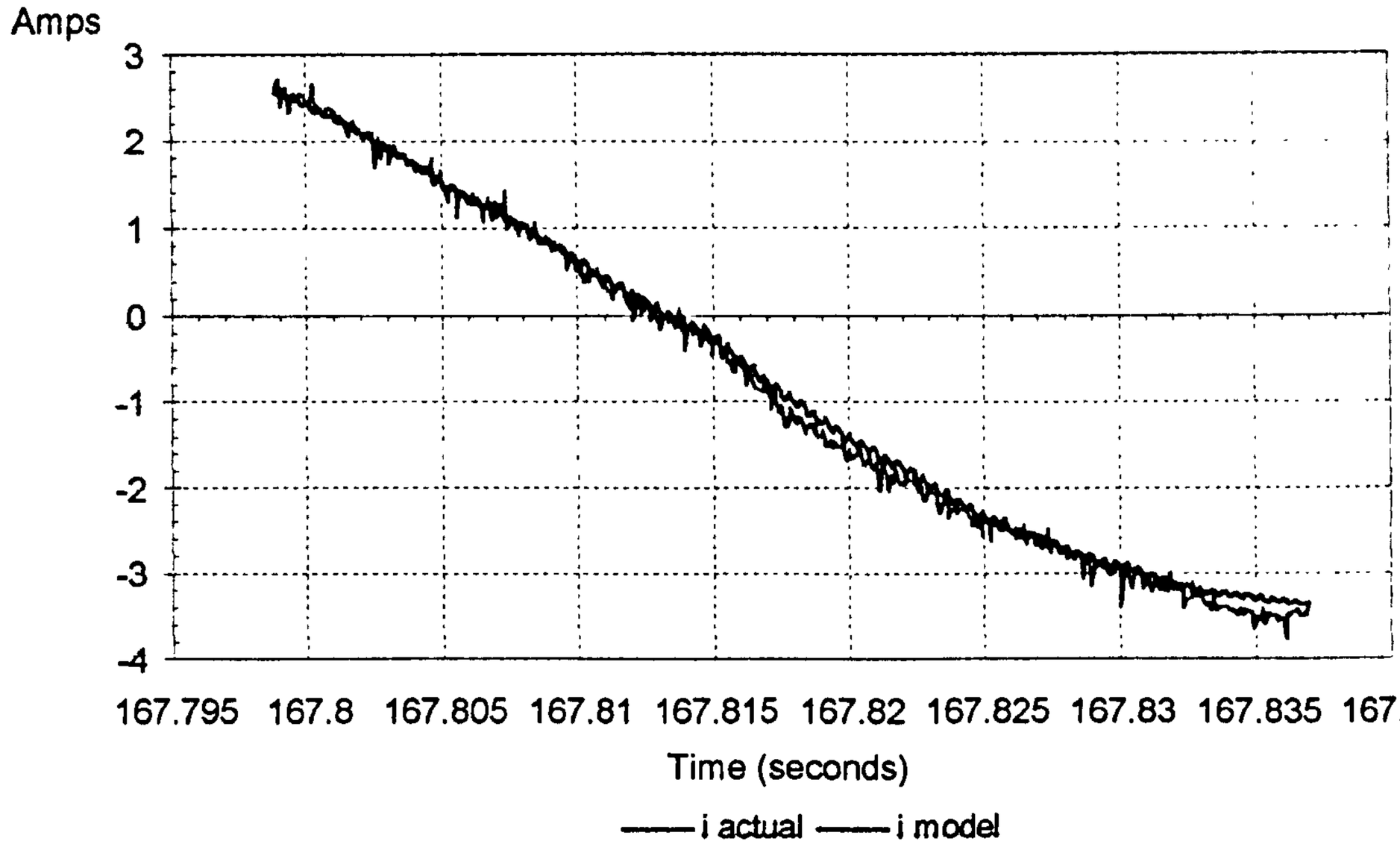


Figure 7.3.24: Actual Machine Current and Real-Time Model Current for 10Hz Demand to Inverter Under Test , Detailed View

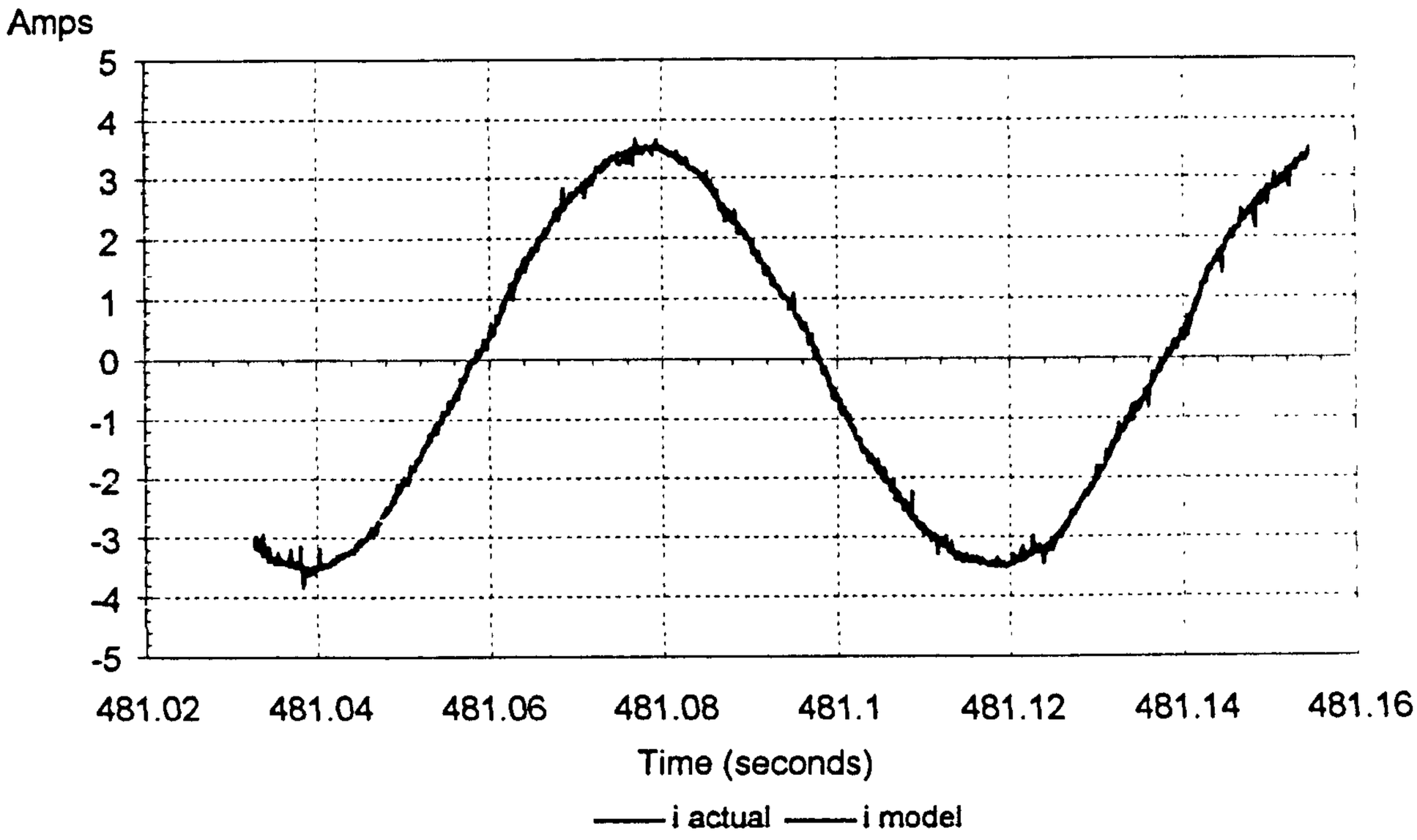


Figure 7.3.25: Demand / Actual Currents Drawn by Virtual Machine, 10Hz

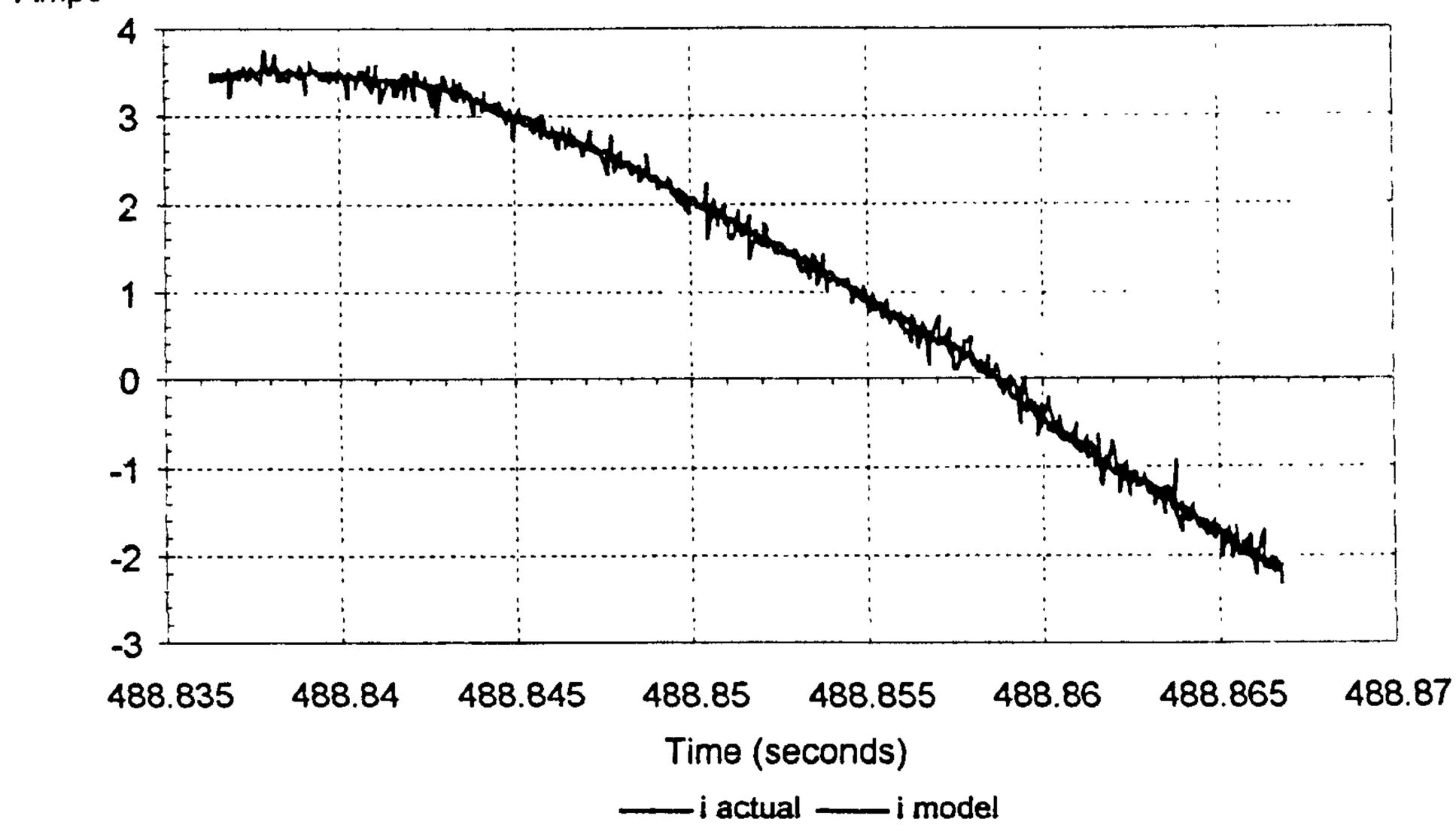


Figure 7.3.26: Demand / Actual Currents Drawn by Virtual Machine, 10Hz

The frequency demand to the inverter under test was then increased further to 30Hz. A set of results similar to those given for the 10Hz demand are now presented for the 30Hz demand. Figure 7.3.27 shows the real machine and real-time model currents. Figures 7.3.28 shows the real-time model and Virtual Machine currents.

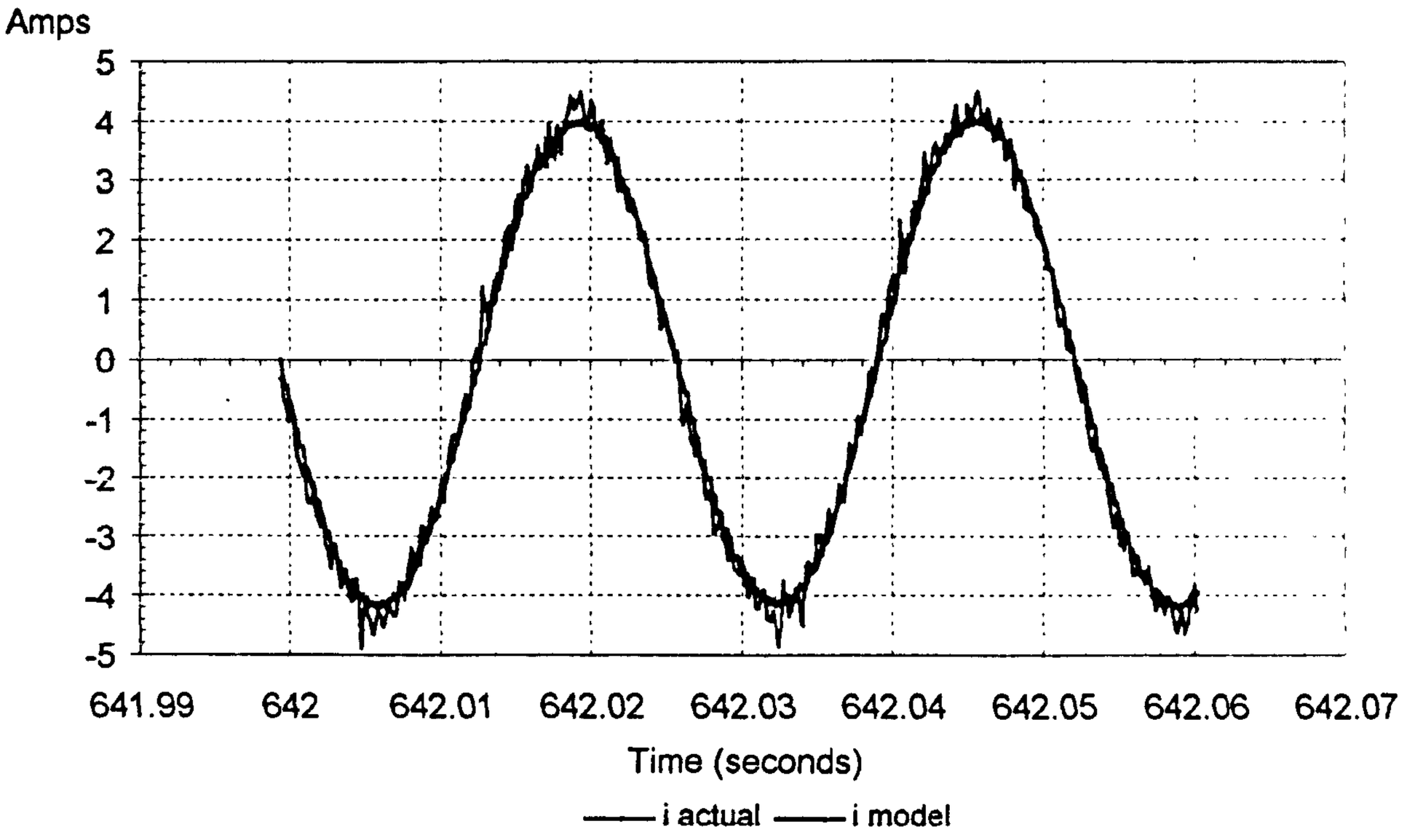


Figure 7.3.27: Actual Machine Current and Real-Time Model Current for 30Hz Demand to Inverter Under Test

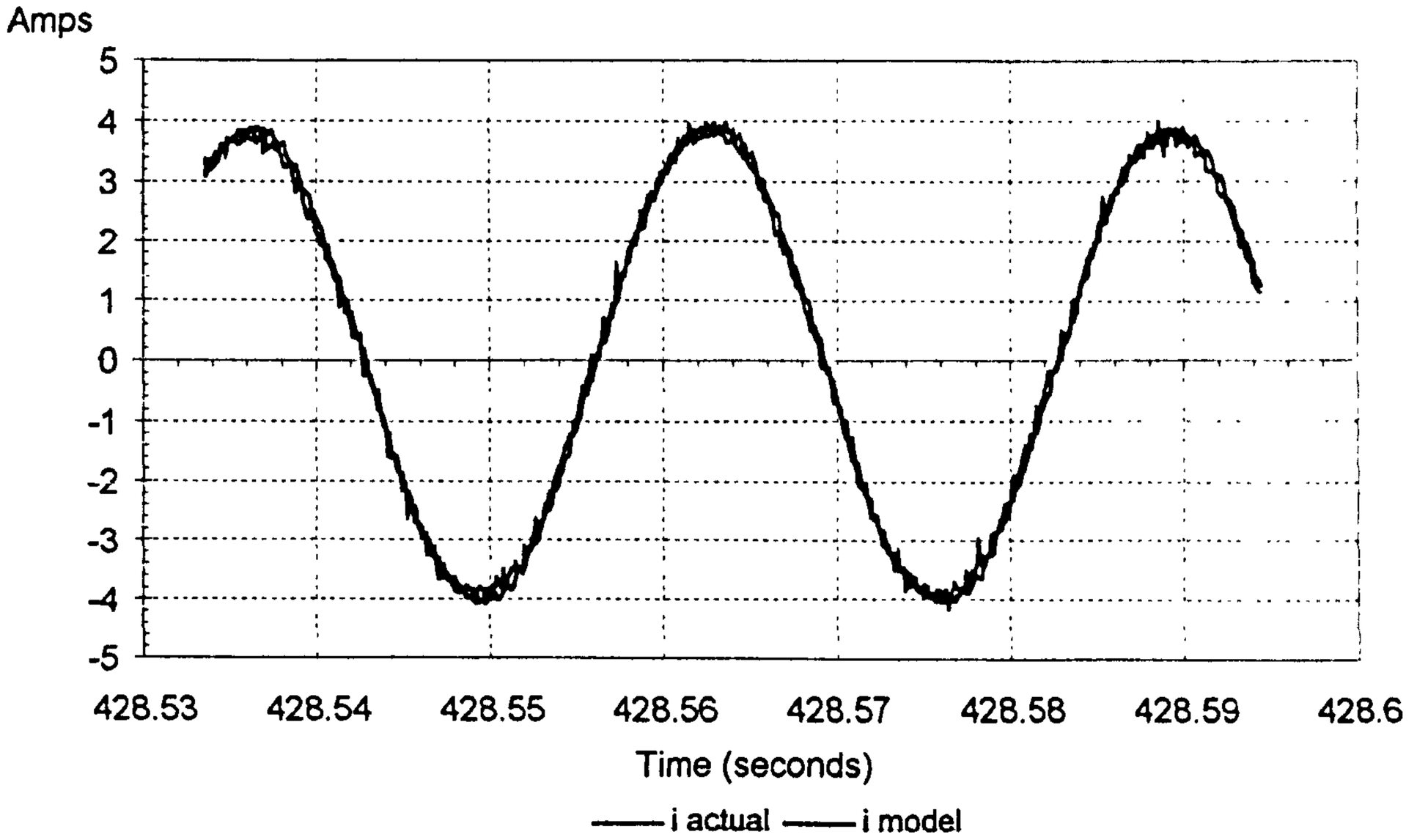


Figure 7.3.28: Demand / Actual Currents Drawn by Virtual Machine, 30Hz

The real-time model currents exhibit a delay when compared to the actual machine currents, again this is due to the sampling period and emphasises the importance of keeping the sampling period as short as possible. Comparing the Virtual Machine

current to the real-time model current, it can be seen that at both steady state frequency demands the Virtual Machine controls the current drawn from the inverter under test extremely well. Figure 7.3.29 shows detail of the Virtual Machine current and the net voltage across the link which is responsible for this current.

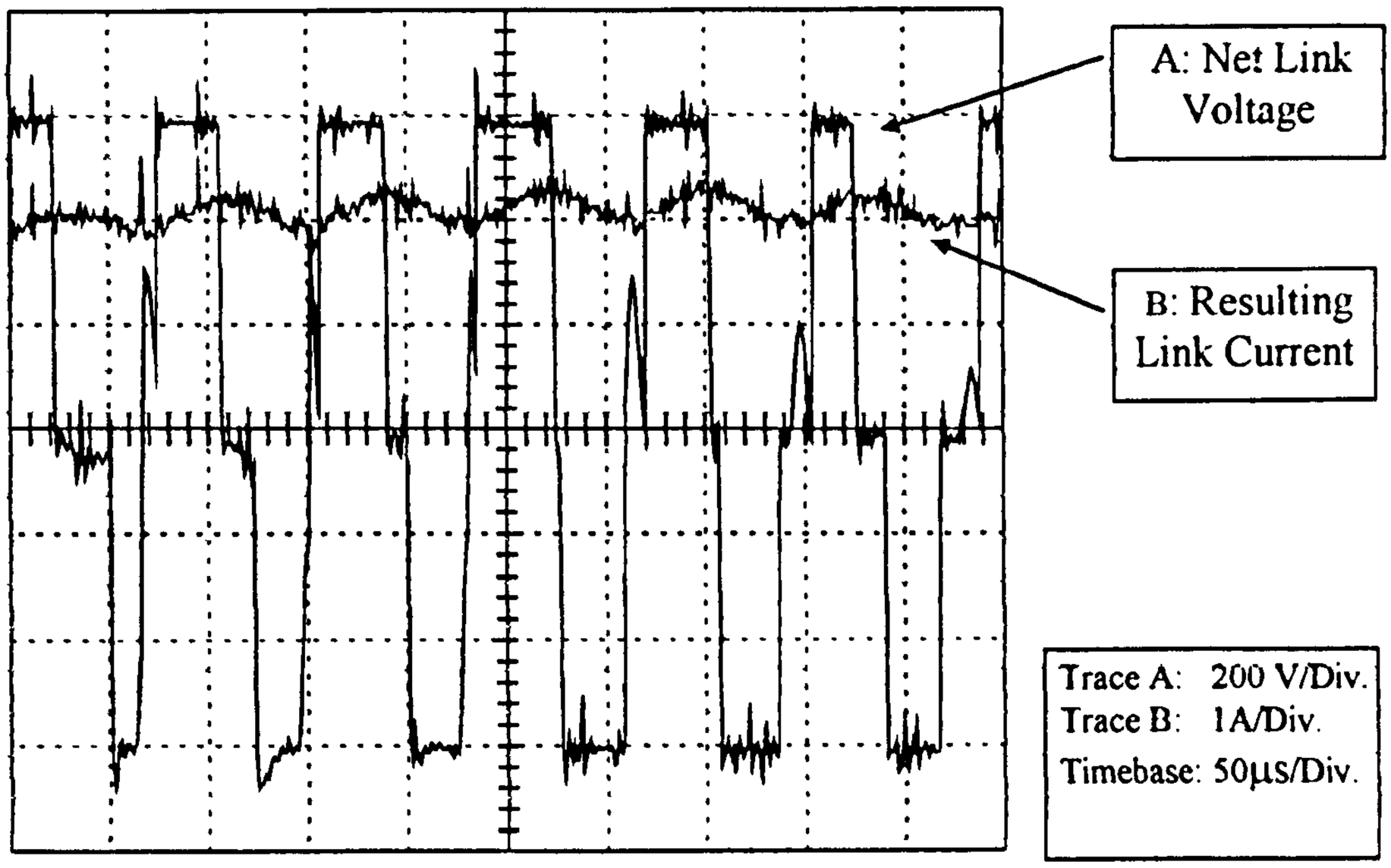


Figure 7.3.29: Net Voltage Across Link and Resultant Current 30Hz Demand

7.3.3 Transient Results Of The Virtual Machine

In order to investigate the transient behaviour of the Virtual machine a speed reversal was applied to the inverter under test. Figures 7.3.30 and 7.3.31 show results for the real machine and figures 7.3.32 and 7.3.33 show results for the Virtual Machine. The following results show the transient behaviour of the Virtual Machine and its comparison to the behaviour of the inverter under test when it is attached to an actual machine. The following figure shows the current drawn from the inverter under test when connected to an actual machine during a full speed reversal. Figure 7.3.30 shows the D and Q axis stator currents drawn from the inverter under test when connected to a real machine for a 50Hz speed reversal demand.

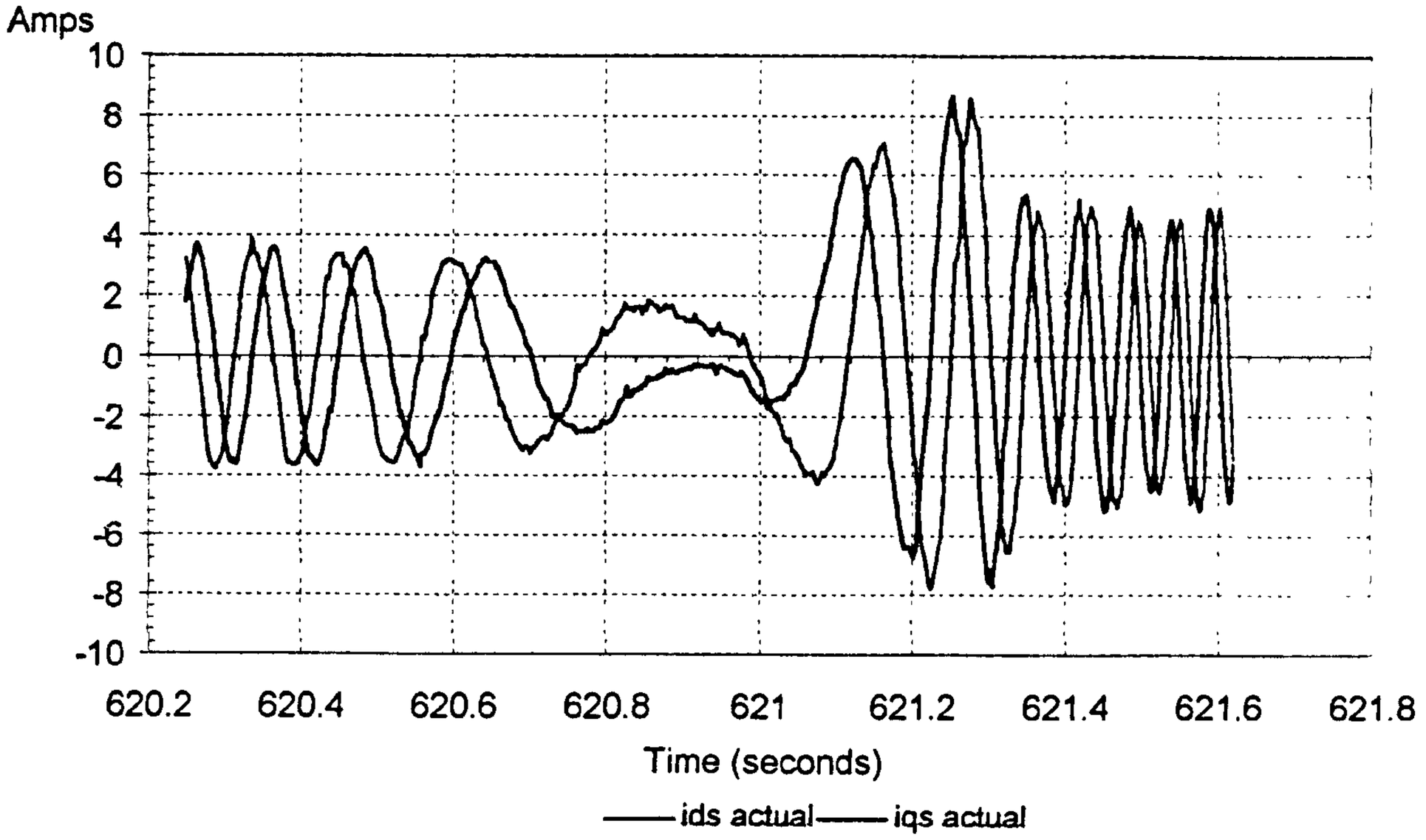


Figure 7.3.30: Actual Machine Currents During 50Hz Speed Reversal

Figure 7.3.31 shows the measured rotor angular velocity of the actual machine during the speed reversal.

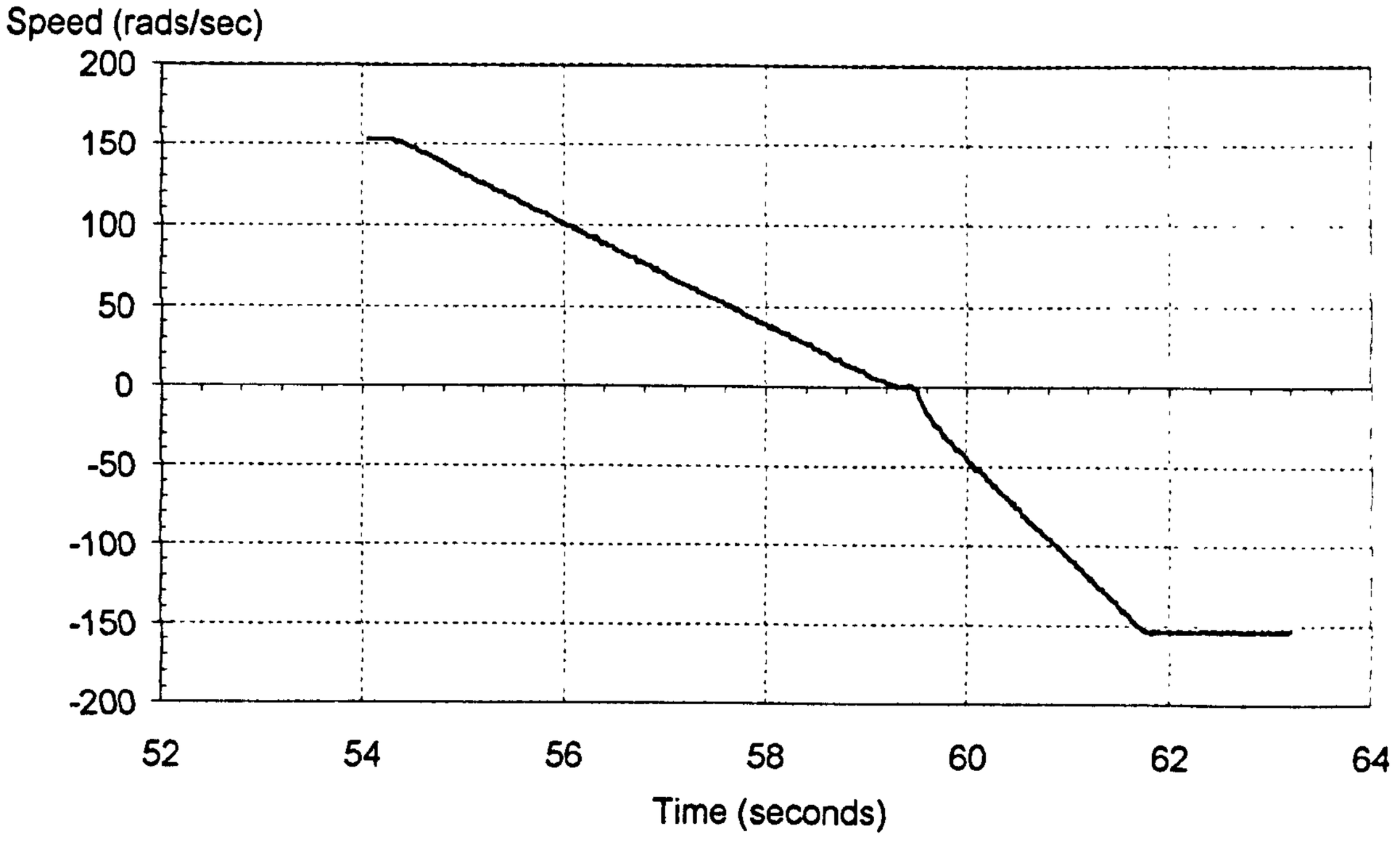


Figure 7.3.31: Rotor Angular Velocity of Actual Machine During 50Hz Speed Reversal

Figure 7.3.32 shows the current drawn from the inverter under test when it connected to the Virtual Machine for the same speed reversal applied to the inverter under test. The real-time model currents, which are the demands to the Virtual Machine, are also shown.

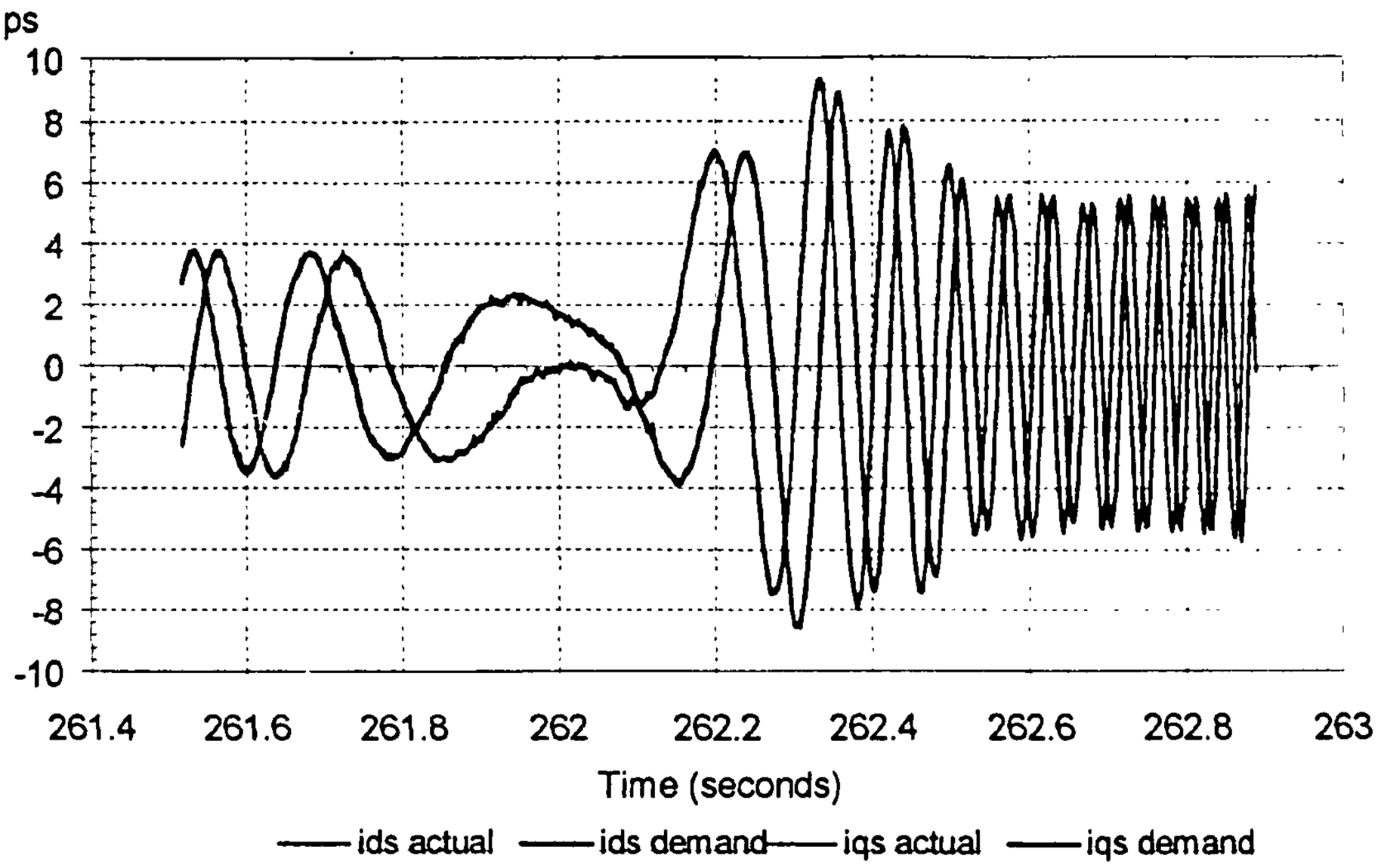


Figure 7.3.32: Actual / Demanded Virtual Machine Currents during 50Hz Speed Reversal

Figure 7.3.33 shows the rotor angular velocity of the Virtual Machine.

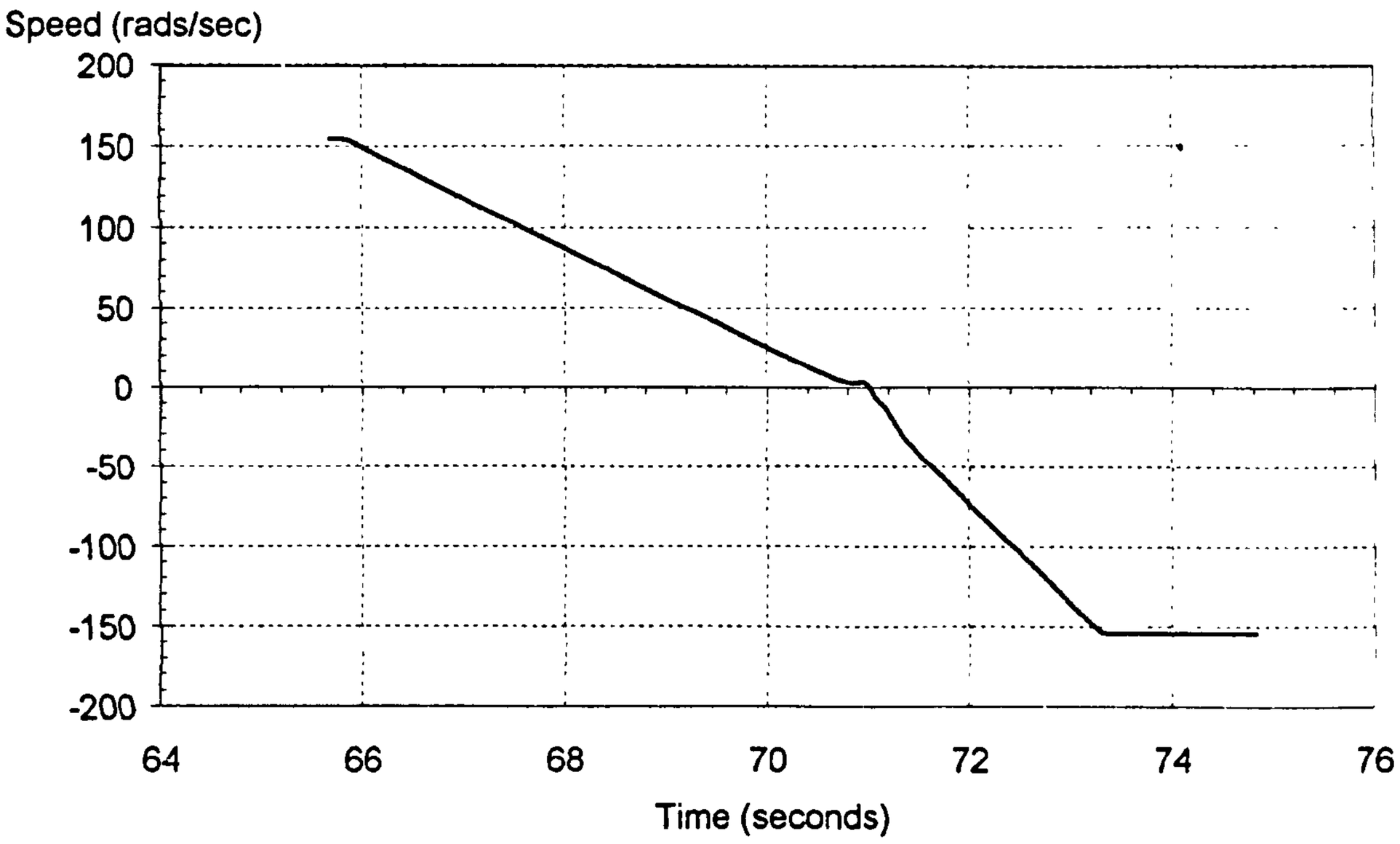


Figure 7.3.33: Rotor Angular Velocity of Virtual Machine During 50Hz Speed Reversal

Considering figure 7.3.32 it can be seen that the Virtual Machine controls the current drawn from the inverter under test to match the real time motor model currents extremely well during the transient period. Therefore, if the real-time motor model currents are producing a reasonably accurate representation of the currents that would flow in a real machine, the Virtual Machine performs an accurate power level emulation of a real machine.

Figures 7.3.34 to 7.3.37 show the behaviour of the Virtual Machine and the inverter under test during a 50Hz speed reversal with the Virtual Machine applying a load of approximately 14 Nm at full speed. The deceleration time is reduced to 1.25 seconds which causes the DC link of the inverter under test to rises sufficiently to activate the dynamic break of the inverter under test. Figures 7.3.34 and 7.3.35 show the electromagnetic torque and the angular velocity of the Virtual Machine respectively.

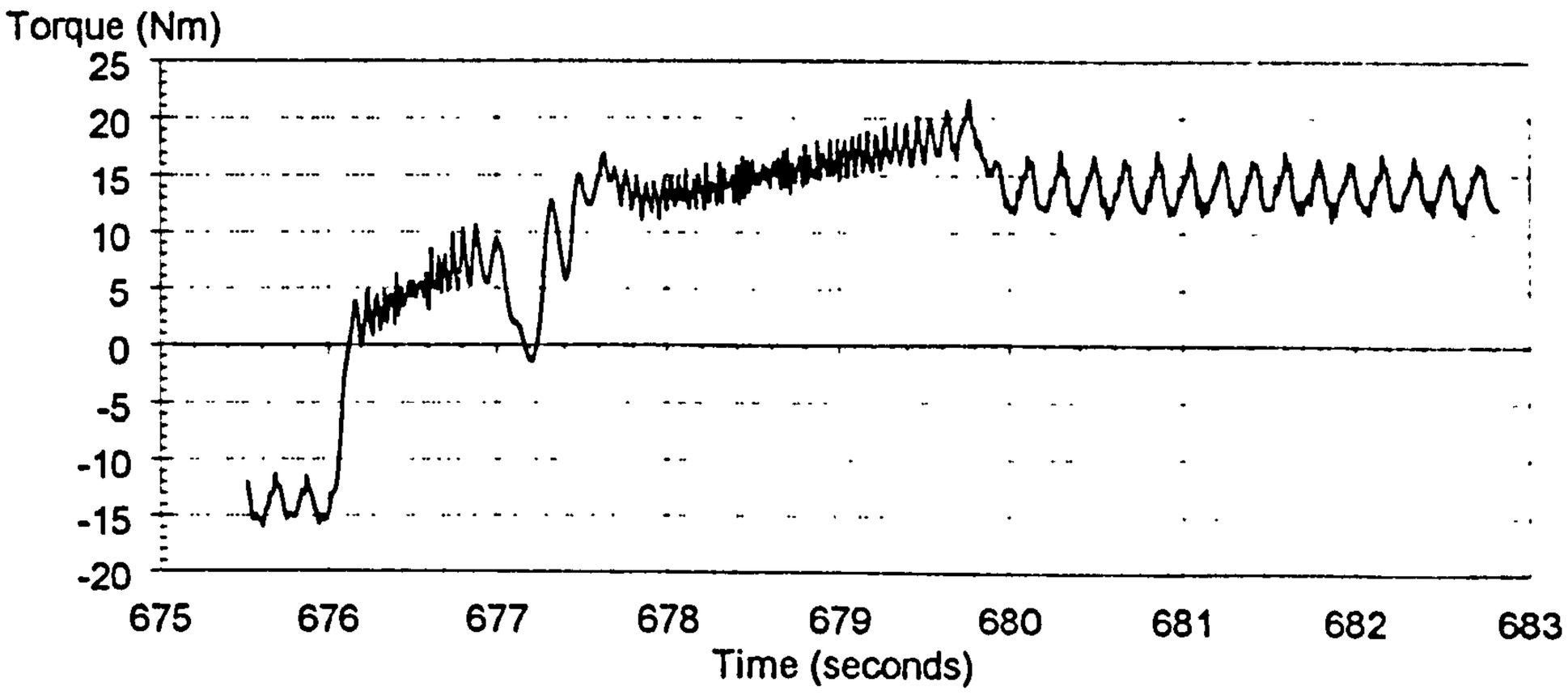


Figure 7.3.34: Electromagnetic Torque of Virtual Machine

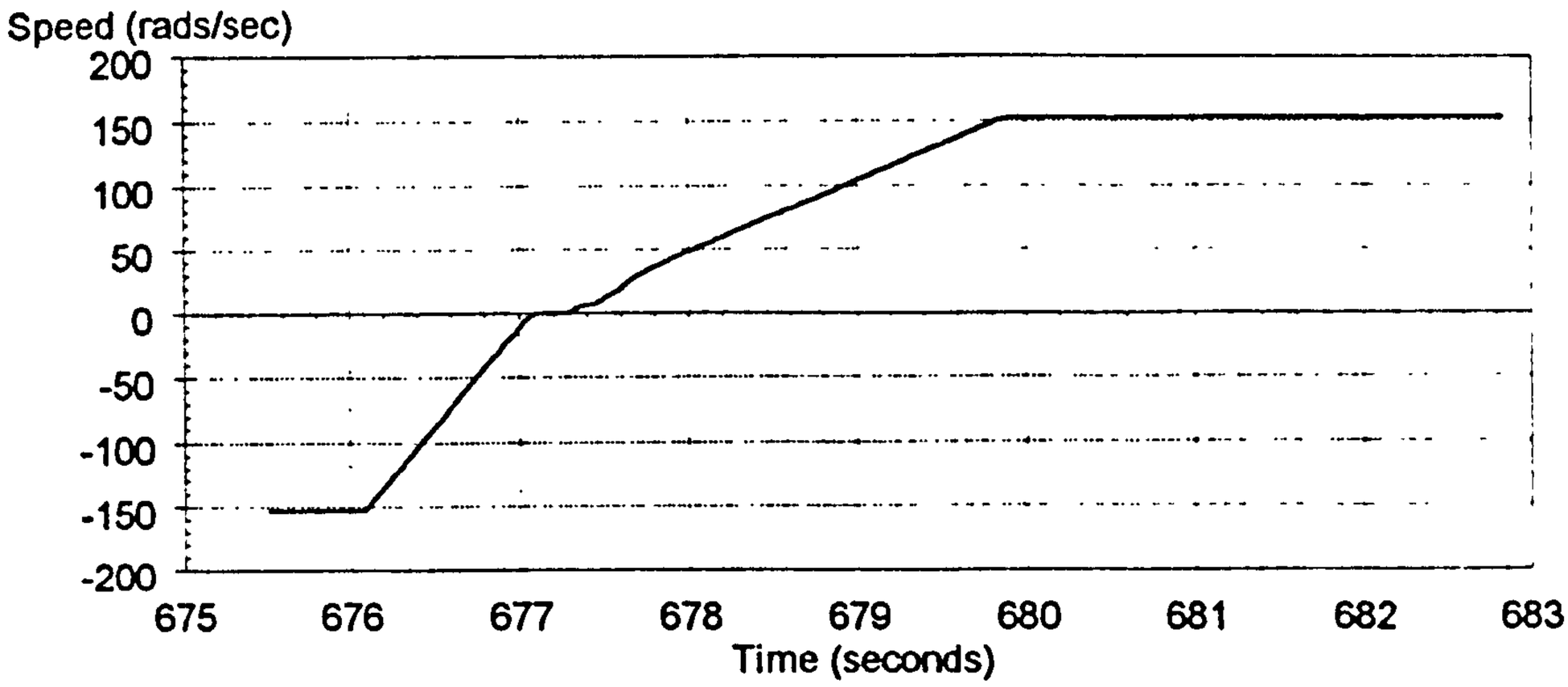


Figure 7.3.35: Angular Velocity of Virtual Machine

The Virtual Machine is connected to the three phase mains supply via a regeneration unit. This unit transfers power from the Virtual Machine back into the mains supply when the DC link of the Virtual Machine rises above a pre-defined value. In this way power which is taken from the inverter under test, and is usually used up rotating a real machine, is returned to the mains supply. Figures 7.3.36 and 7.3.37 show the DC link voltage of the inverter under test during the speed reversal and a measure of the power returned to the mains supply via the regeneration unit of the Virtual Machine. The plot of the DC link voltage shows the period during the speed reversal when the dynamic brake of the inverter under test is in operation i.e. when the DC link voltage rises above 780v. The rise in the DC link voltage is caused by the power flow back into the inverter under test, during this period, power is no longer flowing from the Virtual Machine back into the mains but from the mains into the Virtual Machine and then to the inverter under test.

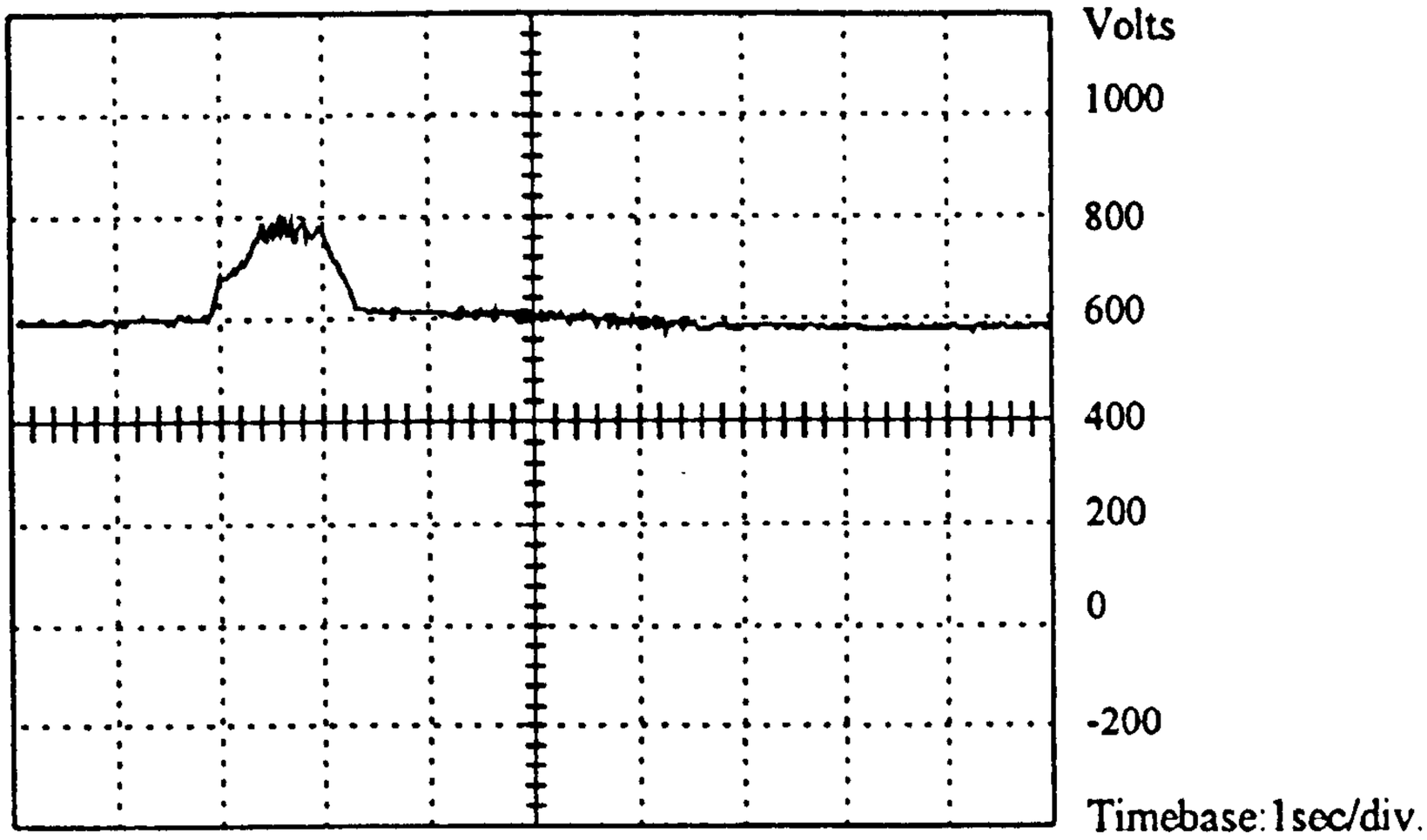


Figure 7.3.36: DC Link Voltage of Inverter Under Test

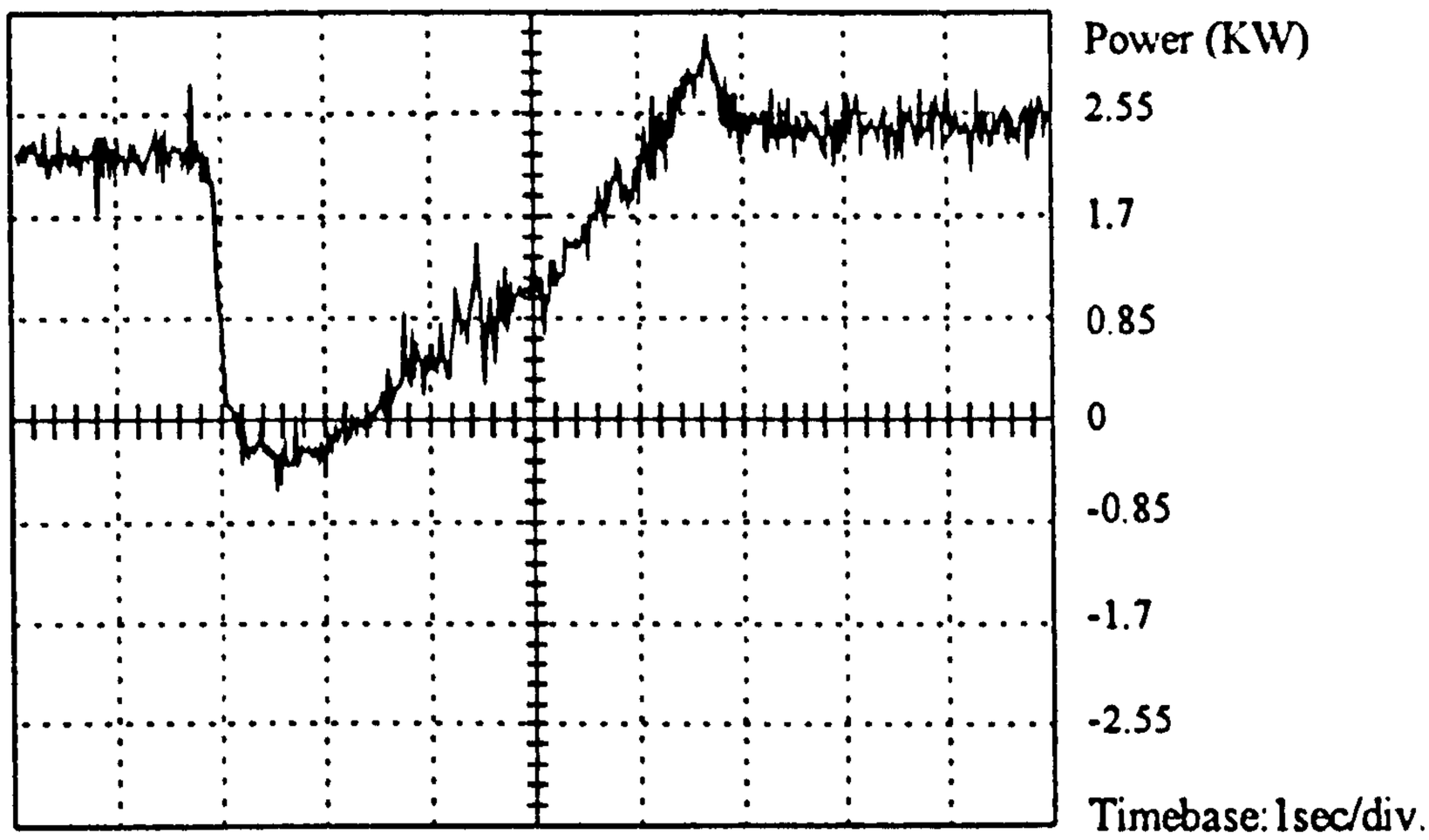


Figure 7.3.37: Power Returned to Mains Supply Via Regeneration Unit of Virtual Machine

7.4 Summary

The ability to simulate in real-time, which has been demonstrated in the previous chapters of this thesis, has been used to create a power level simulation which may be termed a Virtual Machine. It is clear that if a simulation has the ability to run in real-time, then it can be driven by an actual inverter. This idea was hinted by Bosga: [*Bosga et al*, 1995]

"It would even be possible to use a real converter to generate voltages which are then input to the real-time model of the machine."

The Virtual Machine drives the real-time model from the output of the inverter under test. It then goes further and uses the resulting currents as demands for the currents which are drawn from the inverter under test by the power electronics of the Virtual Machine. By actually controlling the power flow to and from the inverter under test the simulation environment is no longer purely signal level but is now at real power levels. The Virtual Machine is capable of simulating, at real levels of voltage and current, the behaviour of an actual induction machine. The results of testing a standard 'off the shelf' inverter with the Virtual Machine compare favourably with those of testing the same inverter with an actual machine. The inverter under test can be used with the Virtual Machine without any problems arising. Using a Virtual Machine to test an inverter offers a number of significant advantages over using an actual machine.

The Virtual Machine can be programmed to behave like any motor or generator, the parameters of the motor can be entered quickly and easily instantly making it appear to the inverter under test that it has been connected to a different machine. One Virtual Machine could be used to test a complete range of inverters. The need for the inverter manufacturer to have a range of motors of different types and ratings to test a range of inverters is avoided. It allows the inverter manufacture to experiment and observe how their inverters behave when attached to motors and drive trains of different characteristics. Because the load characteristics can be programmed into the Virtual Machine it can test the inverter within a simulation of the intended drive application. During the testing period of a drive intended for an electric vehicle

application, the Virtual Machine could represent particular driving conditions. urban cycles for example. Considering another drive application of wind generation, the Virtual Machine could be used to simulate the wind generator which is driven by a randomly varying wind source. The Virtual Machine offers the drive designer a way of easily subjecting the drive to the conditions which it will finally be used, conditions which are normally difficult to replicate in the laboratory.

The Virtual Machine allows the user to test both the hardware and the software of the inverter. The Virtual Machine can provide different load characteristics to test the control algorithms of the inverter against. The flexibility of the Virtual Machine allows the designer of the control algorithms to experiment with different designs in the safe knowledge that if something goes wrong then expensive damage to the inverter and machine will not result. A rotating machine can not be stopped instantly, it has an inertia which can cause problems if a fault occurred with the design of the inverter or the control algorithms within the inverter. The Virtual Machine contains no rotating parts, it is made up of fast acting power electronics which can handle a fault situation and prevent unnecessary damage to the inverter.

The lack of rotating parts within the Virtual Machine also provides a safety benefit. Motor shafts rotating at high velocities create hazardous environments which can endanger engineers in the testing and development areas. Also, because of the Virtual Machine has no moving parts, maintenance of test machines is no longer necessary. The Virtual Machine also has regeneration capability, the power flow from the inverter under test can be returned to the mains supply. When testing an inverter with an actual machine, the machine uses this power for rotation and it is therefore lost, the Virtual Machine offers a far more ecological option. Having the ability to control power flow bidirectionally allows the Virtual Machine to simulate both motors and generators.

8. CONCLUSIONS

8.1 Summary

This section is aimed at summarising all of the preceding chapters. Chapters two to four identified the three main constituent parts which make up a complete drive system and identified suitable mathematical models and algorithms by which they can be represented. These parts are the machine, which was discussed in chapter two, the inverter, which was discussed in chapter three and finally the controller which was discussed in chapter four. The motor model which is used to simulate the machine is based on twin axis theory and is carried out in the stationary reference frame attached to the stator of the machine. A comparison of the execution times for this model, programmed in 'C' and carried out in a personal computer, was made with a standard commercial simulation package, highlighting the drawbacks of such a general package typical timing results were given in table 2.7.1.

Chapter five explained the multiple digital signal processing system which was used throughout the research. This parallel processing system, based on the Texas TMS320C40 D.S.P., was used to implement the simulation and control code. Chapter six showed that real-time simulation of the complete drive system was now possible using the multiple processing system. The performance of the real-time simulation was demonstrated by running an actual drive in parallel with the real-time simulation of the drive. Both the simulation and the actual drives were driven from the same control processor. The results of the real-time simulation proved to be very good in both the steady state and the transient cases.

Chapter seven took the whole concept of simulation a stage further. This chapter explained how the ability to simulate in real-time can be used to develop a power level simulator. This power level simulator, which has come to be known as the

Virtual Machine, is designed to replace an actual machine during the development and testing stage of inverter manufacture. The Virtual Machine consists of power electronics, a real-time model and a controller. The Virtual Machine can then control the power flow from the inverter which is being tested in such a manner that it appears to the inverter to be just like an actual machine. Comparison of the results from testing an inverter with a real-machine to those with a Virtual Machine showed the Virtual Machine to perform well. The Virtual Machine offers a great number of advantages over a real machine during the testing stage of the inverter design, not only because of its ability to behave like any machine or load, but also because of its safety and ecological advantages.

8.2 Conclusions

As the processing power available to drive manufacturers is rapidly increasing, the opportunity to implement more complex control algorithms in cost effective drives is opening up. This increase in complexity, however, also increases the chances of controller errors occurring. Such errors could cause costly damage to expensive power electronic hardware. Because of this, simulation is becoming a powerful tool with which the designer of the control algorithms and drive hardware can experiment, safe in the knowledge that an unstable control loop or parameter error will not result in damage to the drive

Simulation has always been one remote step away from the real drive and the real controller. By developing a real-time simulation, the controller used within the simulation during the development stages of the drive design can be exactly the same controller as that which is to be used with the actual drive system. This controller can be implemented in the same processor that is to be used in the final product and interface to the simulation in exactly the same way as it will with the actual drive. Because the code developed during the simulation stage is the actual code to be used in the final system, there is no need for code translation between the simulation environment and the actual drive. This allows the control algorithms to be developed once only, which means that the development time for the drive is reduced and also

that there is no possibility of introducing errors during the translation period between simulation and actual systems. The necessity to simplify the translation between simulation and actual code is underlined by the current trend in simulation packages which can generate executable code for the target processor such as Link-RT by LSI and a system based on Matlab/Simulink and the Texas C31 processor developed at Newcastle University [*French*, 1994]. This approach simplifies programming the processors and reduces translation errors but the simulation is still carried out in non real-time.

The results of the real-time simulation compared favourably with the real drive indicating that the real-time simulation environment can provide a useful tool for controller development. It was proved with both a simple open loop controller and a more complex vector controller. It does, however, have limitations. The main limitations are the amount of processing time available and the execution speed of the processors carrying out the simulation code. The present system uses a control cycle of 6.5KHz, which gives a control period of 153.83μs. All of the simulation code which represents the inverter and the machine must therefore be executed within this 153.83μs period. This has been shown to be achievable with the present multiple processing system, but there is not enough remaining time to allow any further increases of the switching frequency, at present there is only 20μs processing time available.

The timing section indicated how much processing power is still available with the present system. The spare time available in the controller DSP has not been used to perform any simulation since this processor is a dedicated controller for the actual drive. The QPC-C40 board which is located within the PC has locations for four C40 TIM modules. The addition of the fourth processor would immediately provide more processing power, and the number of processors could further be increased by the addition of more QPC-C40 boards. The addition of further processors would mean that simulation of tasks which were not dependent upon each other could be carried out in parallel; tasks such as the four electrical equations solved for the motor model.

By carrying out each of these four equations in separate processors the execution time for the motor model would be quartered to 19.1μs.

The switching devices are treated as ideal switches with no attempt to simulate the turn-on, turn-off characteristics of the devices. The switching devices used in the actual drive are IGBTs which have a turn-on time of 0.5μs and a turn-off time of 1.5μs. It must, therefore, be apparent that to simulate the switching characteristics of this type of device in real-time would require the simulation to be carried out in under 0.5μs. The C40 DSP which is currently used to perform the simulation has a clock frequency of 40MHz, which gives it an instruction cycle time of 50ns. This means that only 10 single cycle instructions could be executed to simulate the turn-on of the device. This is not a realistic option and it will take the introduction of faster processors to attempt this level of real-time simulation, the following paragraph discusses such advances in processor technology. The current method of simulating switching devices at the signal level is to use commercial packages such as PSPICE and libraries of micromodels [*Salazar, Joos,* 1994]. These simulations allow modelling of snubber circuits, gate drive circuitry and analysis of switching device losses, but are far from real-time simulations.

As the specification of available processors continues to increase, the instruction cycle time of 50ns for the present TMS320C40s will begin to look slow. Texas Instruments already offers the TMS320C80 DSP which combines four DSPs and a RISC controller on a single chip and is capable of two billion operations per second i.e. an instruction every 0.5 ns. This is the most powerful DSP available today and with instruction time down this low, the possibility of modelling switching devices in real-time can easily be envisaged. Furthermore, a recent press release by Texas Instruments [*Texas Instruments,* 1996] , outlined their new 0.18 micron processor technology. The new microchip is to have 125 million transistors packed into a single device using their new Timeline Technology. With this new technology many DSPs could be combined onto one chip offering significant increases in performance. The large transistor count per microchip will also overcome some of the inherent limitations of the DSPs presently available. The limitation with the present DSP technology is due

to the processor communicating with separate memory devices. The ability to have so many transistors on the same piece of silicon will allow static RAM to be built onto the same chip as the processor, thus allowing much faster instruction cycle times and opening up the possibility of real-time simulation of faster processes such as switching characteristics of power electronic devices.

The Virtual Machine provides a new type of emulation environment, one in which simulation is performed in real-time at real power levels. The Virtual Machine is designed to replace the actual machine and load when testing inverters, thus providing a safer more flexible testing environment. The Virtual Machine developed was based on the real-time simulation environment. It used a real-time motor model to predict the behaviour of a machine and then power electronics to control the current taken from the inverter being tested with the Virtual Machine. The Virtual Machine does not suffer, as much as the real-time simulation environment, from the time limitations imposed by the performance of the present multiple digital signal processing system. This is due to the fact that the Virtual Machine does not require any simulation of the inverter, which includes modelling the PWM etc. The Virtual Machine samples the output of the actual inverter being tested; therefore there is no requirement for a simulation of the inverter and the only simulation code that needs to be executed is the simulation of the machine itself. Ideally the Virtual Machine would respond to every output pulse of the inverter under test in exactly the same manner as a real machine. The current drawn from the inverter would exhibit exactly the same switching harmonics as when connected to a real machine. If the sampling frequency of the Virtual Machine was extremely high, compared to the switching frequency of the inverter under test, and the inductance linking the inverter under test to the Virtual Machine kept low, then the current drawn would match that of a real machine. The Virtual Machine, however, requires a finite time to perform the real-time model and control the currents. Because of this time some detail of the PWM is lost. It can be seen then, that benefits could be gained by introducing more processing power, or more importantly, faster processors.

The sampling system of the Virtual Machine at present introduces a delay equal to the sampling period, in the current system this is 77μs for a switching frequency of 13KHz. It would of course be desirable to reduce this delay as much as possible. Reducing this delay would allow the Virtual Machine to respond to the modulated output of the inverter under test in a manner which is closer to the real machine. The introduction of the sort of processor technology mentioned above, regarding the real-time simulation environment, would allow far more complex models of the machine to be used. The machine model could be enhanced so that effects such as saturation, including cross saturation, could easily be modelled in real-time. The sampling system could also be operated at a much higher frequency which would reduce the delay introduced and, together with a faster control loop for the current drawn from the inverter under test, improve the accuracy of the overall Virtual Machine.

The Virtual Machine offers many advantages over an actual machine and its associated load, fundamentally it can be programmed to behave as any machine and load. The present system incorporates a graphical user interface (GUI), which is programmed in the Windows™ environment. This familiar type of user interface allows the machine and load parameters to be quickly and easily modified by a dialogue box accessed by a pull down menu as the Virtual Machine is running. Drive manufacturers can experiment with machine and load parameters and observe how their control algorithms perform and cope with variations in these machine parameters, thus testing the robustness of their controllers. It is also possible to produce abnormalities such as winding shorts or open circuits with the system in a controlled manner.

The power rating of the Virtual Machine is limited by the rating of the power electronics incorporated within it, i.e. the power electronics which sink and source power from and to the inverter under test. The present system uses a 15KW inverter as the power electronics heart of the Virtual Machine and the largest machine which it can therefore emulate is 15KW. The current real-time machine model is programmed to behave as a 3KW induction machine. This one Virtual Machine could be used to test a complete range of inverters up to 15KW. Normally separate machines would be required for each rating of inverter but the Virtual Machine avoids this requirement.

The Virtual Machine could offer great advantages not just to the drives manufacturers, but also to the clients of the drives manufacturers. If, for example, the customer has a specific process which requires the implementation of a drive system, then the Virtual Machine can be programmed to model this whole process. The drive system which is supplied by the drive manufacturer can then be tested with the Virtual Machine acting as the overall process, alleviating the need and risk associated with using the real system in the initial testing. If the drive system fails to respond or perform as required, then no damage, which could prove costly, will result to the process.

Apart from the flexibility that the Virtual Machine offers the drive manufacturer during the testing stages of drive development, the Virtual Machine also provides a safer test bed for use with drives. An actual machine has inertia, which means that once it is running it cannot be stopped instantly, for it has stored energy which must be removed. If the inverter that is driving the machine fails to perform then the inertia of the test machine may cause even more damage to the inverter under test. The Virtual Machine has no moving parts and therefore no inertia or stored energy, so if a fault occurs with the inverter under test, the Virtual Machine can react by recognising the fault and stopping the power flow between the inverter under test and the Virtual Machine. This ability to react quickly to a fault situation can avoid any additional, possibly costly, damage to the inverter under test. The Virtual Machine must respond faster than the inverter under test to allow it to protect the inverter under test in this manner.

Not only is the lack of rotating parts within the Virtual Machine beneficial in terms of a fault situation occurring with the inverter under test, but also the Virtual Machine provides a safer test environment for the development engineers working on the inverter. The lack of rotating machinery within the test area would mean less space would be required, because a complete range of actual machines would no longer be required. The Virtual Machine produces less audible noise than an actual machine which is also beneficial to the development engineers. The lack of rotating parts within

the Virtual Machine also means that no maintenance of test machinery would be required.

The ability of the Virtual Machine to return the energy from the inverter under test to the mains supply has ecological benefits as compared with the actual machine. In the case of the latter, the energy taken from the inverter under test is simply used to rotate the machine and is lost. This ability to return the power taken from the inverter to the mains supply means that the power supply requirements to the laboratory could also be reduced.

8.3 Future Work

The advantages that faster and more powerful processors could have on the real-time simulation environment have already been highlighted. Any future development of the real-time simulation environment would require the addition of more, similar, processors i.e. TMS320C40s, or the move to a different processing system such as the TMS320C80. The new Texas Timeline technology processors will most likely be desirable when released, but the date of release is yet to be announced. The development of a real-time drive simulation based on this new processor technology should allow the real-time simulation environment to be taken to further levels of complexity and therefore accuracy.

The Virtual Machine presently only includes a model of the three phase induction machine. Future work could include the development of further real-time models which allow the Virtual Machine to simulate other machines.

The Virtual Machine has been demonstrated to a number of leading drives manufacturers. All have shown a great deal of interest in the Virtual Machine, and all recognised it as being beneficial to the development of their products. One future development which has been specifically requested by one of the companies that witnessed a demonstration of the Virtual Machine is the addition of a speed/position encoder output. This means that the Virtual Machine should be able to provide signals

which emulate those produced by a shaft mounted speed or position encoder. This emulated encoder signal could then be fed back to the inverter under test, which would require the encoder signal for a closed loop control scheme. One of the world's leading drives manufacturers has requested prototype Virtual Machines for use in the development of a new product. This will lead to the Virtual Machine, which was developed simply as a prototype during the project, to be further developed into a more commercial tool.

9. REFERENCES

Blaschke,F. 1972
"The Principle of Field Orientation as Applied to the New Transvektor Closed-Loop Control System for Rotating Machines"
Siemens Review, Vol.34 , pp.217-220, May 1972

Boldea, I., Nasar, S.A.1987
"Unified Treatment of Core Losses and Saturation in the Orthogonal-Axis Model of Electric Machines"
IEE Proceedings,
Vol. 134, Pt. B, No.6, Nov. 1987, pp.355-363

Bose, B.K.1986
"Power Electronics and AC Drives"
Prentice-Hall, New Jersey, 1986

Bose, B.K. 1995
"Recent Advances in Power Electronics"
IEEE Transactions on Power Electronics,
Vol. 7, No. 1, Jan. 1992, pp.2-15

Bosga, S.G., v.d.Burgt, J.J.A., Duarte, J.L., Vandenput, A.J.A. 1995
"A Multi-DSP System for Real-Time Simulation and Control of Electric Drive Systems"
EPE Sevilla, Vol. 3, 1995, pp.971-975

Brown, J.E., Kovacs, K.P., Vas, P. 1983
"A Method of Including the Effects of Main Flux Path Saturation in the Generalised Equations of A.C. Machines"
IEEE Transactions on Power Apparatus and Systems,
Vol. PAS-102, No. 1, Jan. 1983, pp.96-103

Chhaya, S.M., Bose, B.K. 1993
"Expert System Based Automated Simulation and Design Optimisation of a Voltage-Fed Inverter for Induction Motor Drive"
IECON Proceedings, Industrial Electronics Conference,
Vol. 2, 1993, pp.1065-1070

Chowdhury, G., Giesselmann, M. 1994
"Development of Induction Motors Drives with Real-Time PWM Control and Dynamic Modelling of Drive Performance with Graphical User Interface"
Proceedings Intersociety Energy Conversion Engineering Conf., Vol. 1, 1994, pp.190-195

Dodson, R.C., Evans, P.D., Tabatabaei Yazdi, H. 1990
"Compensating for Dead Time Degradation of PWM Inverter Waveforms"
IEE Proceedings, Vol. 137, Pt. B, No. 2, Mar. 1990, pp.73-81

Finch, J.W., Atkinson, D.J., Acarnley, P.P. 1990
"Scalar to Vector: General Principles of Modern Induction Motor Control"
IEE Conference Power Electronics and Variable Speed Drives, 1990 ,pp.364-369

Finney, D. 1994
"Computer Simulation of Variable Speed Drives"
IEE Colloquium (Digest)
Digest No. 020, Jan. 20th 1994

Fitzgerald, A.E., Kingsley, C., Umans, S.D. 1990
"Electric Machinery"
5th Edition, McGraw-Hill, 1990

French, C.D. 1994
"Flexible DSP Controller"
CAPEC Report on Task One, Aug. 1994
University of Newcastle Upon Tyne

Gabriel,R. , Leonhard,W. , Nordy,C.J., 1980
"Field-Oriented Control of a Standard AC Motor Using Microprocessors"
IEEE Transactions on Industry Applications,
Vol. IA-16, No. 2, Mar/Apr 1980, pp.186-192

Garces, L.J. 1980
"Parameter Adaptation for the Speed-Controlled Static AC Drive with a Squirrel-Cage Induction Machine"
IEEE Transactions on Industry Applications,
Vol. IA-16, No. 2, Mar/April 1980, pp.173-178

Handley, P.G., Boys, J.T. 1992
"Practical Real-Time PWM Modulators, an Assesment"
IEE Proceedings-B, Vol. 139 No. 2, Mar. 1992, pp.92-102

Jansen, P.L., Lorenz, R.D. 1993
"Accuracy Limitations of Velocity and Flux Estimation in Indirect Field Oriented Induction Machines"
EPE Brighton 1993, pp.312-318

Jansen, P.L., Lorenz, R.D., Novotny, D.W. 1994
"Observer-Based Direct Field Orientation: Analysis and Comparison of Alternative Methods"
IEEE Transactions on Industry Applications,
Vol. 30, No. 4, Jul/Aug 1994, pp.945-952

Jeffrey, A. 1986
"Mathematics for Engineers and Scientists"
Van Nostrand Reinhold (UK) Co. Ltd, 1986

Jones, C.V. 1967
"The Unified Theory of Electrical Machines"
Butterworths, London 1967

Kleinhans, C.E., Diana, G., Harley, R.G., McCulloch, M.D., Randelhoff, M., Woodward, D.R. 1994
"Analysing a CSI-Fed Field Oriented Controlled Induction Motor using a New Simulation Package CASED"
IECON Proceedings Vol.1, 1994, pp.192-197

Kovacs, K.P. 1984
"On the Theory of Cylindrical Rotor A.C. Machines, Including Main Flux Path Saturation"
IEEE Transactions on Power Apparatus and Systems,
Vol. PAS-103, No. 4, Apr. 1984, pp.754-761

Krause, P.C. 1986
"Analysis of Electric Machinery"
McGraw-Hill, 1986

Leonhard, W. 1985
"Control of Electrical Drives"
Springer-Verlag, 1985

Leonhard, W. 1988
"Field-Orientation for Controlling AC-Machines - Principle and Application, a Tutorial"
IEE Conference Power Electronics and Variable Speed Drives
1988, pp.277-282

Levi,E. 1994a
"Magnetic Saturation in Rotor-Flux-Oriented Induction Motor Drives: Operating Regimes, Consequences and Open-Loop Compensation"
European Transactions on Electrical Power Engineering (ETEP)
Vol. 4, No. 4, 1994, pp.277-286

Levi,E. 1994b
"Impact of Iron Loss on Behaviour of Vector Controlled Induction Machines"
Conference Record IEEE Industry Applications Society Annual Meeting IAS, Denver Colorado,
IEEE Press Cat. No. 94CH34520 1994, pp.74-80

Levi, E. 1995
"Comparative Study of Detuning Effects in Indirect Rotor Flux Oriented Induction Machines Due to Iron Losses"
Proceedings of IEE Int. Conf. on Power Electronics and Drive Systems, PEDS
Cat. No. 95TH8025, 1995, pp.639-644

Liaw, C., Chao, K., Lin, F. 1992
"A Discrete Adaptive Field-Oriented Induction Motor Drive"
IEEE Transactions on Power Electronics,
Vol. 7, No. 2, Apr. 1992, pp.411-419

Lipo, T.A. 1988
"Recent Progress in the Development of Solid-State AC Motor Drives"
IEEE Transactions on Power Electronics,
Vol. 3, No. 2, Apr. 1988, pp.105-116

Lorenz, R.D. 1986
"Tuning of Field-Oriented Induction Motor Controllers for High Performance Applications"
IEEE Transactions on Industry Applications,
Vol. IA-22, No. 2 Mar. 1986, pp.293-297

Maron, M.J. 1987
"Numerical Analysis A Practical Approach"
Macmillan Publishing Company, 1987.

Melkebeek, J.A.A., Novotny, D.W. 1983
"The Influence of Saturation on Induction Machine Drive Dynamics"
IEEE Transactions on Industry Applications,
Vol. IA-19, No. 5, Sep./Oct. 1983, pp.671-681

Melkebeek, J.A.A. 1983
"Magnetising-Field Saturation and Dynamic Behaviour of Induction Machines - Part1: Improved Calculation Method for Induction Machine Dynamics"
IEE Proceedings, Vol. 130, Pt. B, No. 1, Jan. 1983, pp.1-9

Melkebeek, J.A.A. 1983
"Magnetising-Field Saturation and Dynamic Behaviour of Induction Machines - Part2: Stability Limits of a Voltage-Fed Induction Motor and of a Self-Excited Induction Generator"
IEE Proceedings, Vol. 130, Pt. B, No. 1, Jan. 1983, pp.10-17

MicroSoft 1990
"Windows Guide to Programming Version 3"
Microsoft Press, 1990

Mohan, N., Undeland, T.M., Robbins, W.P. 1989
"Power Electronics: Converters, Applications, and Design"
John Wiley and Sons, 1989

Moreira, J.C.1993
"A New Method for Rotor Time Constant Tuning in Indirect Field Oriented Control"
IEEE Transactions on Industry Applications,
Vol. 8, No. 4, Oct. .1993, pp.626-631

Moreno-Eguilaz, J.M., Brodonau, J. 1994
"Comparison of Differential Equation Resolution Methods in Real-Time Microprocessor-Based Vector Control System for an Induction Drive"
IEEE Workshop on Computers in Power Electronics,
1994, pp.319-322

Murai, Y., Watanabe, T., Iwasaki, H. 1987
"Waveform Distortion and Correction Circuit for PWM Inverters with Switching Lag-Times"
IEEE Transactions on Industry Applications,
Vol. IA-23, No. 5, Sept./Oct. 1987, pp.881-886

Nigim, K.A. 1994
"PC Based Single and Three Phase Induction Motor Drive Performance Simulation"
Mediterranean Electrotechnical Conference MELECON
Vol. 3, 1994 pp.1255-1258

Onbilgin, G., Senlik, I, Sezgin, A. 1995
"Magnetic Saturation in Dynamic Behaviour Modelling of Three Phase Induction Motors with Space Phasors"
International Aegean Conference on Electrical Machines and Power Electronics ACEMP ,
Vol. 1, 1995 pp.78-82

Petzold, C. 1990
"Programming Windows"
Microsoft Press, 1990

Salazar, L., Joos, G. 1994
"PSPICE Simulation of Three-Phase Inverters by Means of Switching Functions"
IEEE Transactions on Power Electronics,
Vol. 9, No. 1, Jan. 1994, pp.35-42

Say, M.G. 1983
"Alternating Current Machines"
Fifth Edition, Pitman, London, 1983

Slater, H.J., Armstrong, G. 1995
"Drives Simulation Library"
University of Newcastle Upon Tyne, Machines and Drives Group, 1995

Sukegawa, T., Kamiyama, K., Mizuno, K., Matsui, T., Okuyama, T. 1991
"Fully Digital, Vector-Controlled PWM VSI-Fed ac Drives with an Inverter Dead-Time Compensation Strategy"
IEEE Transactions on Industry Applications
Vol. 27, No. 3, May/June 1991, pp.552-559

Texas Instruments Ltd. 1991
"TMS320C40 User's Manual"
Revision A, May 1991

Texas Instruments Ltd. 1996
"TI's 0.18 Micron Process Technology Packs 125 Million Transistors on a Single Chip"
Web Address: http://www.ti.com/corp/docs/pressrel/1996/96025b.htm

Vas, P., Hallenius, K.E., Brown, J.E.1986
"Cross-Saturation in Smooth-Air-Gap Electrical Machines"
IEEE Transactions on Energy Conversion,
Vol. EC-1, No. 1 Mar. 1986, pp.103-109

Vas, P., Alakula, M., Brown, J.E., Hallenius, K.E. 1988
"Field-Oriented Control of Saturated A.C. Machines"
IEE Conference Power Electronics and Variable Speed Drives,
1988, pp.283-286

Vas, P. 1990
"Vector Control of AC Machines"
Clarendon Press, Oxford, 1990

Vas, P. Li, J. 1993
"Simulation Package for Vector Controlled Induction Motor Drives"
IEE sixth Annual Conference on Electrical Machines and Drives
Conference Publication 376, 1993, pp.265-270

Virk, G.S., Liang, D.T.W. 1993
"Intelligent Simulation of Induction Motor Drives"
IEE 2nd International Conference on Advances in Power System
Control, Operation and Management,
Hong Kong, Dec. 1993, pp.757-762

Wade, S., Dunnigan, M.W., Williams, B.W. 1994
"Simulation of Induction Machine Vector Control and Parameter
Identification"
IEE Conference Publication, Vol. 399, 1994, pp.42-47

Appendix A

Determination of Machine Parameters

In order to determine the values of the machine parameters a no load and a locked rotor test was carried out on the machine. Both tests were carried out with the following circuit.

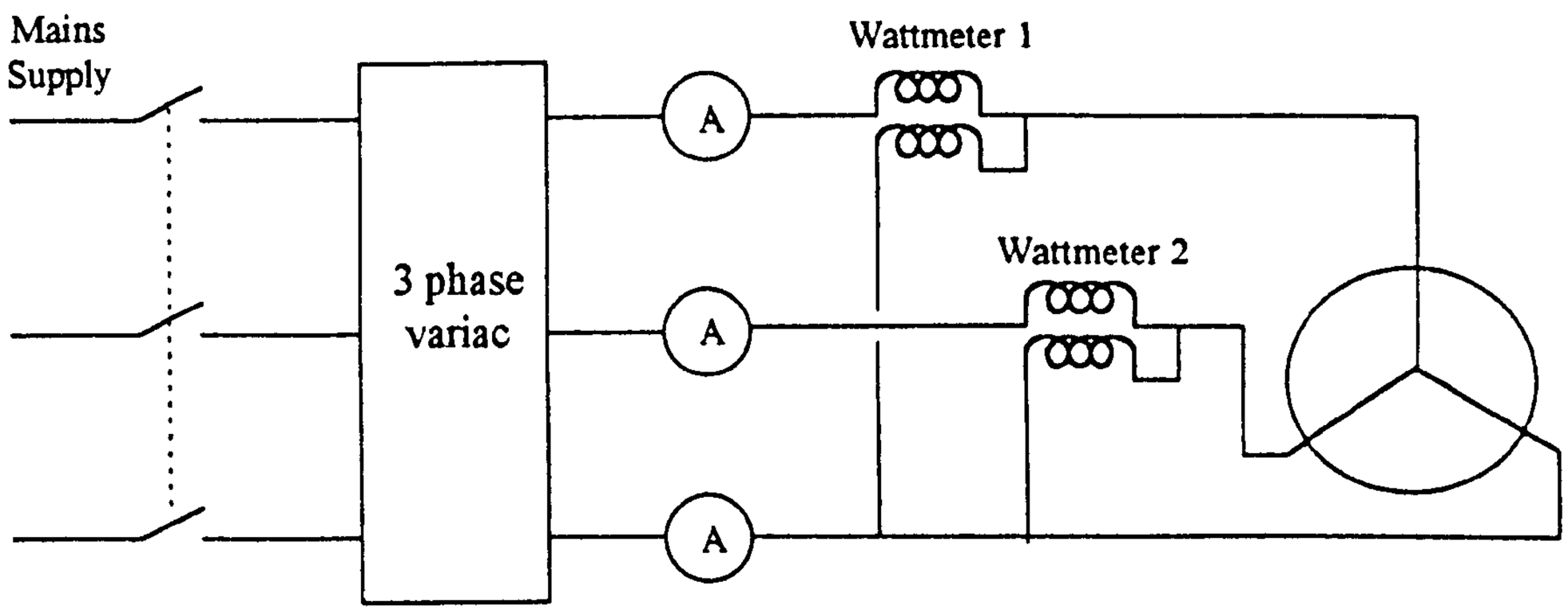


Figure A.1: Two Wattmeter Connection

This connection allows the measurements required by the no load and locked rotor tests described in Chapter 2 to be taken.

Appendix B

Experimental Determination of Inertia

The following discusses the experimental procedure used to determine the inertia of the laboratory test rig which was used throughout the project.

The laboratory test rig is made up of the three phase 3KW induction machine which is directly coupled to a Mawdsley DC work machine as follows:-

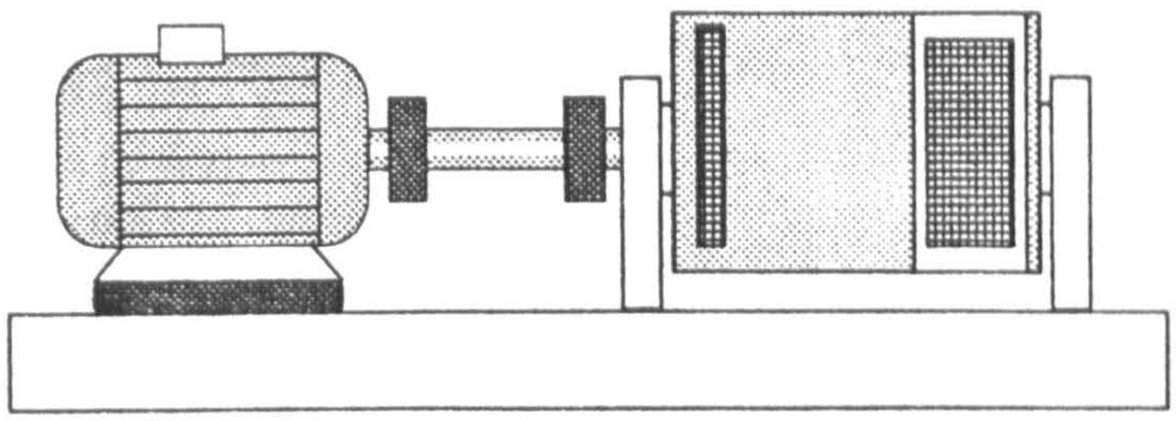


Figure B.1: Laboratory Machine Set-Up

The load is applied to the induction machine by exciting the field winding of the DC work machine and connecting a resistive load bank across the armature of the DC work machine.

The mechanical equation is:-

$$\frac{d\omega_r}{dt} = \frac{1}{J}(T_e - T_{load})$$

where:- J is the inertia of the system.

T_e is the electromagnetic torque produced by the machine.

T_{load} is the load torque.

In order to determine the value of J two test are required.

It can be seen from the electromechanical equation that under steady state conditions, when there is no change in speed, the electromagnetic torque of the induction machine must equal the total load torque. The first test ran the machine at different angular velocities and the steady state torque was measured. The second test required is a run-down test in which the machine is driven to a certain speed and then the power is switched off. The machine then decelerates due to load torque at a rate dictated by the inertia. The following figures show the results of the two tests.

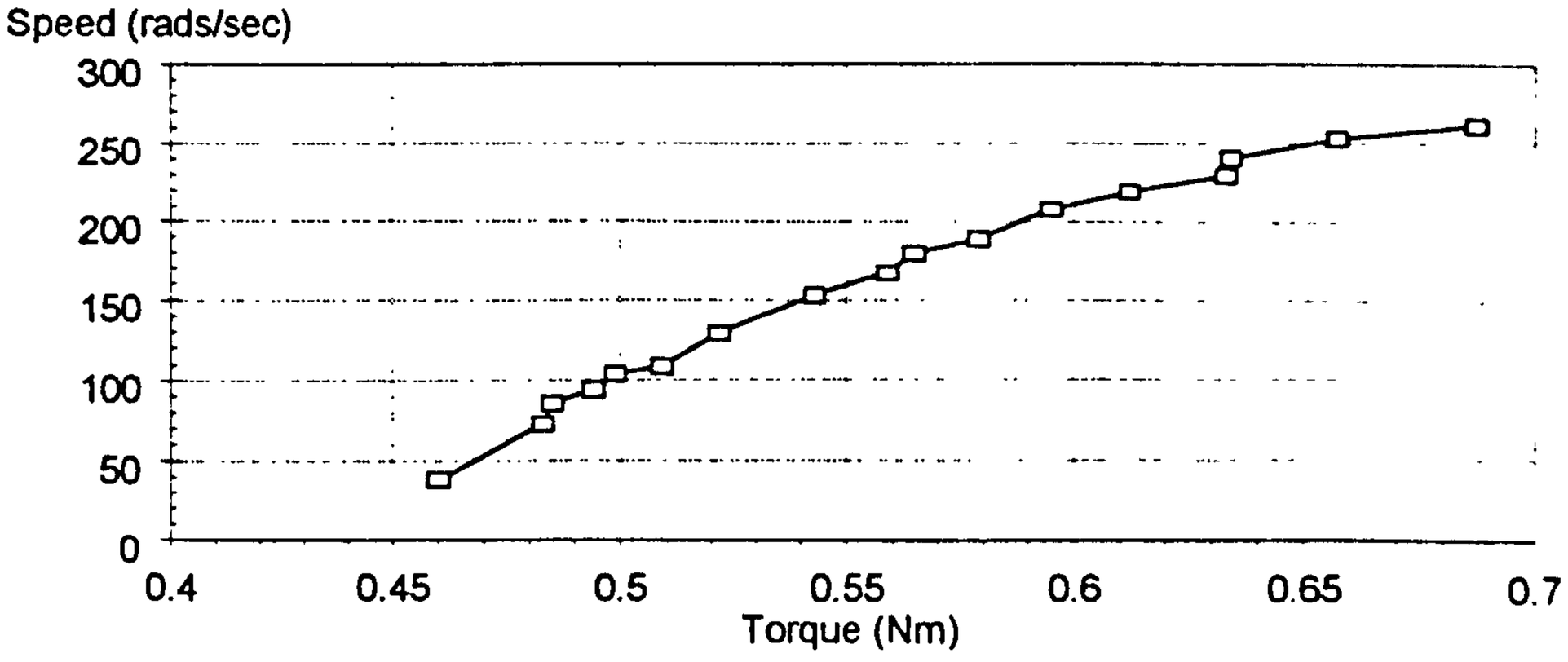


Figure B.2: Steady State Test

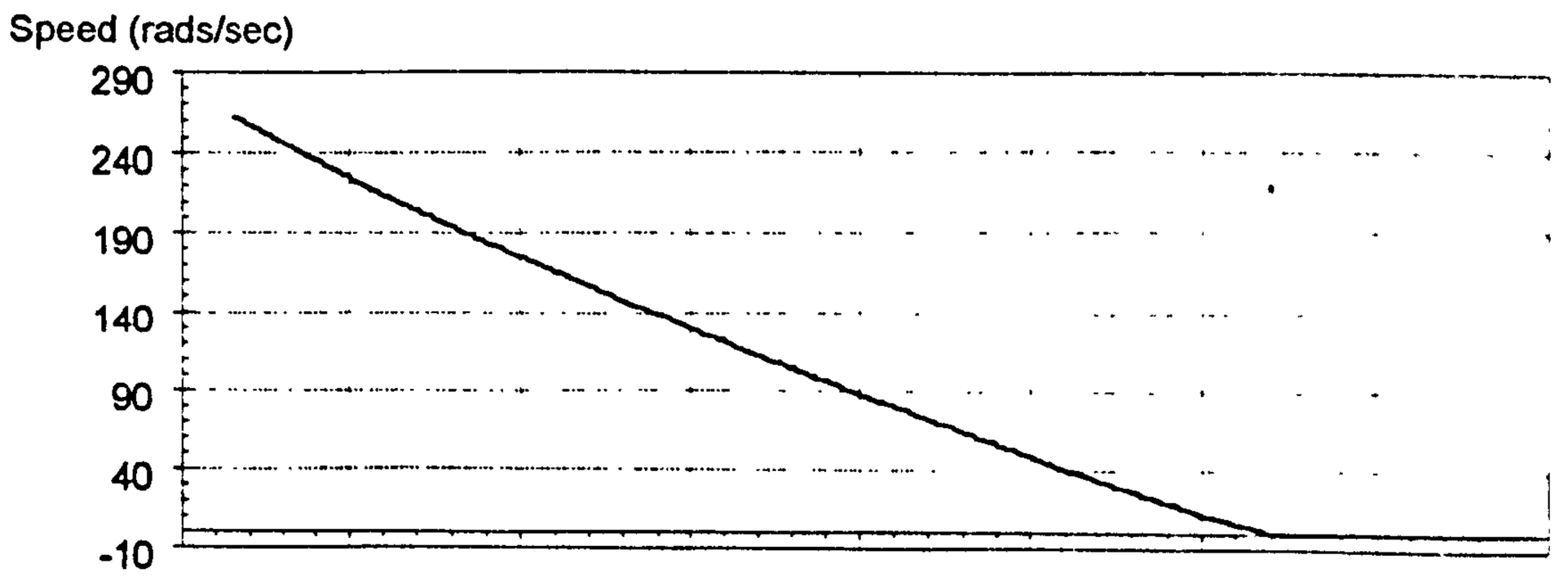


Figure B.3: Run-Down Test

with T_e=0, $$J = \frac{-T_{load}}{\frac{d\omega_r}{dt}}$$

The slope of the run-down curve is measured at a particular speed giving $d\omega/dt$ and at the same speed on the steady state curve the load torque is calculated. This allows the inertia to be obtained.

Three speeds were considered:-

@40 rads/sec T_{load} = 0.46 Nm , $d\omega/dt$ = -4.0 rads/sec/sec $\therefore J$ = 0.115 Kgm^2

@150 rads/sec T_{load} = 0.54 Nm , $d\omega/dt$ = -4.75 rads/sec/sec $\therefore J$ = 0.114 Kgm^2

@250 rads/sec T_{load} = 0.65 Nm , $d\omega/dt$ = -5.7 rads/sec/sec $\therefore J$ = 0.114 Kgm^2

[*Leonhard*, 1985]

Appendix C

Dead Time

The following timing diagrams show the effect of dead time on the inverter switching pattern. The effect that the deadtime has, is to either increase or decrease the switching periods within the switching pattern. The direction of the current flow in each of the three phases determines whether the times are increased or decreased. Six cases are presented which cover all possible current directions. All cases start with all of the upper devices OFF and all of the lower devices ON. In the first half of the switching pattern, the turn ON of the upper devices is delayed by the deadtime after the turning OFF of the lower devices. In the second half of the switching pattern the turn ON of the lower devices is delayed by the deadtime after the turning OFF of the upper devices.

The resultant Voltage across one phase of the machine (VAN) is obtained for each case.

Case One:-

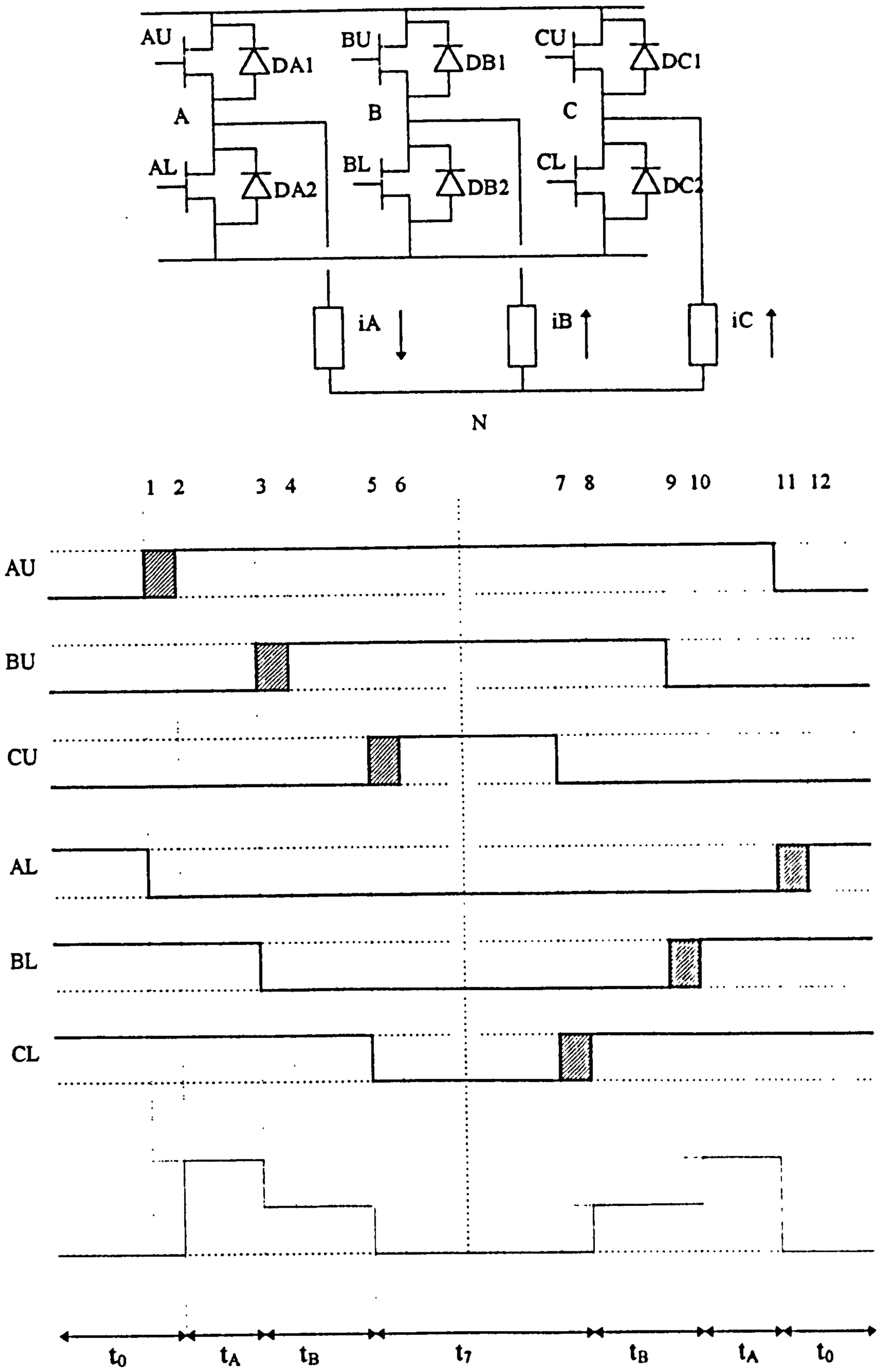


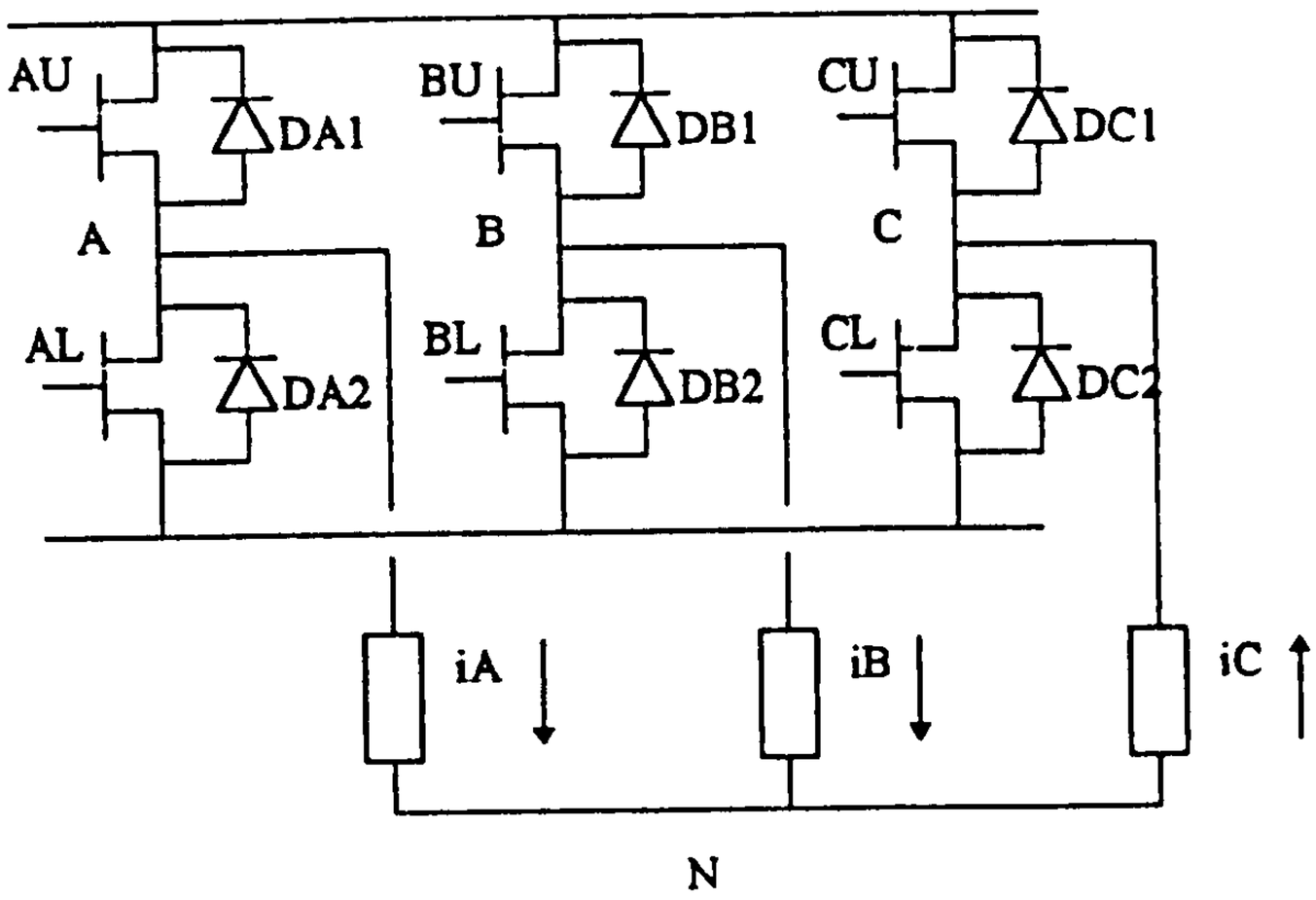
Figure C.1: DeadTime Case One

To start all lower devices ON.

DA2,BL,CL conducting:-

1. AL OPENS iA continues to flow via DA2, point A remains at 0v, zero state continues
2. AU CLOSES iA flows via AU, point A goes to Vdc, state A starts
3. BL OPENS iB flows via DB1, point B goes to Vdc, state B starts
4. BU CLOSES iB continues to flow via DB1, point B remains at Vdc, state B continues
5. CL OPENS iC flows via DC1, point C goes to Vdc, zero state starts
6. CU CLOSES iC continues to flow via DC1, point C remains at Vdc, zero state remains
7. CU OPENS iC continues to flow via DC1, point C remains at Vdc, zero state remains
8. CL CLOSES iC flows via CL, point C goes to 0v, state B starts
9. BU OPENS iB continues to flow via DB1, point B remains at Vdc, state B remains
10. BL CLOSES iB flows via BL, point B goes to 0v, state A starts
11. AU OPENS iA flows via DA2, point A goes to 0v, zero state starts
12. AL CLOSES iA continues to flow via DA2, point A remains at 0v, zero state continues

Case Two:-



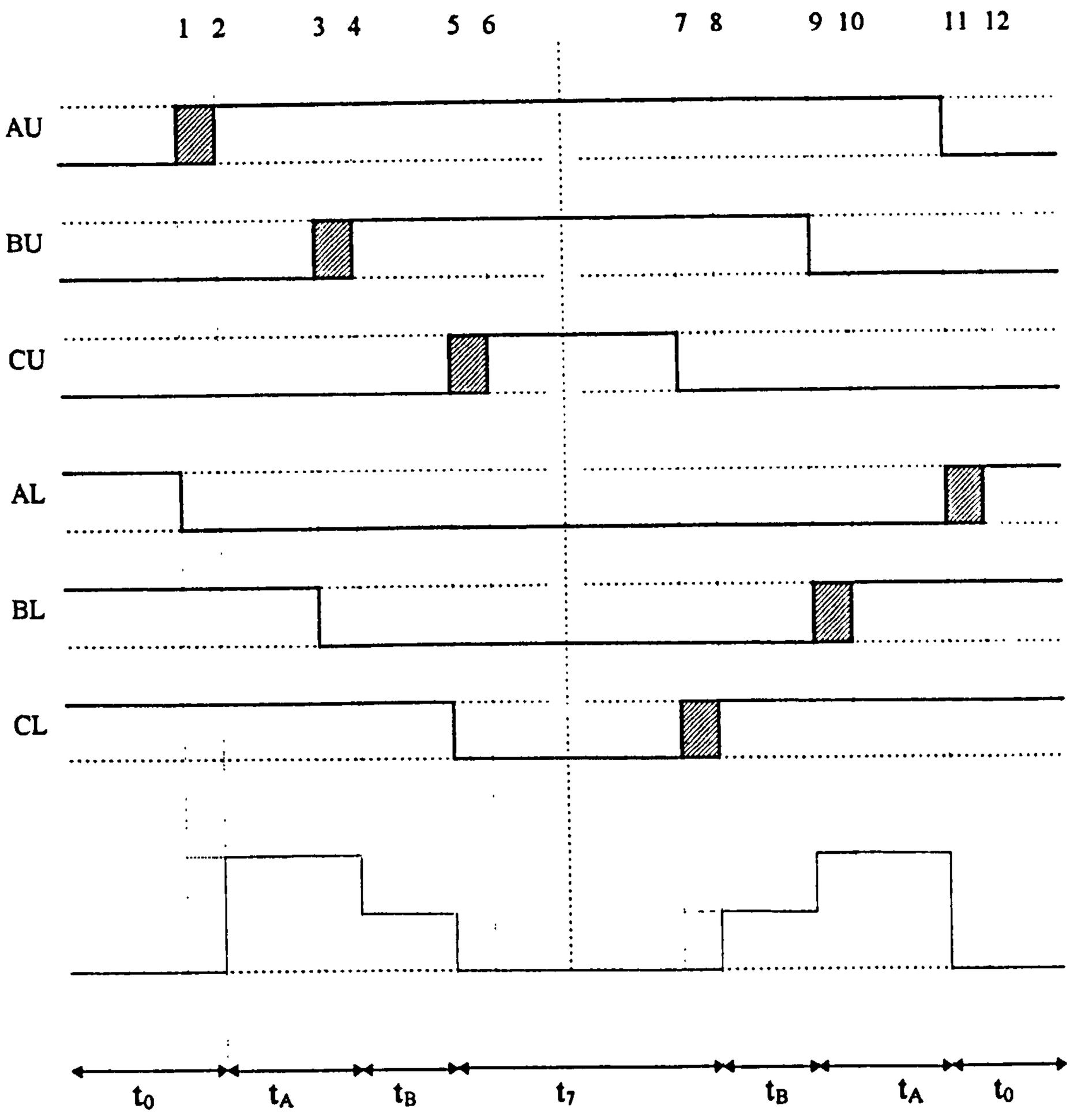


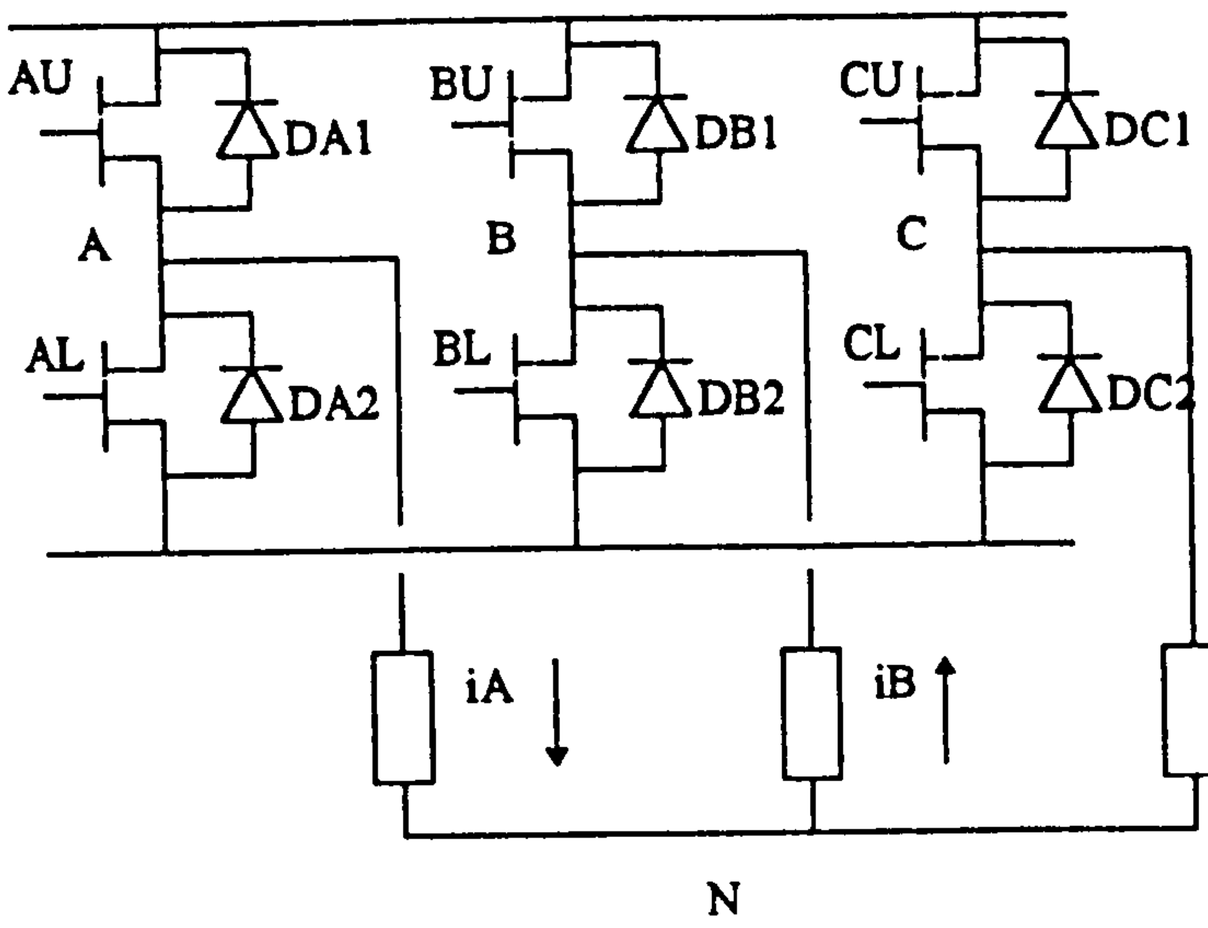
Figure C.2: DeadTime Case Two

To start all lower devices ON.

DA2,DB2,CL conducting:-

1. AL OPENS iA continues to flow via DA2, point A remains at 0v, zero state continues
2. AU CLOSES iA flows via AU, point A goes to Vdc, state A starts
3. BL OPENS iB continues to flow via DB2, point B remains at 0v, state A continues
4. BU CLOSES iB flows via BU, point B goes to Vdc, state B starts
5. CL OPENS iC flows via DC1, point C goes to Vdc, zero state starts
6. CU CLOSES iC continues to flow via DC1, point C remains at Vdc, zero state remains
7. CU OPENS iC continues to flow via DC1, point C remains at Vdc, zero state remains
8. CL CLOSES iC flows via CL, point C goes to 0v, state B starts
9. BU OPENS iB flows via DB2, point B goes to 0v, state A starts
10. BL CLOSES iB continues to flow via DB2, point B remains at 0v, state A continues
11. AU OPENS iA flows via DA2, point A goes to 0v, zero state starts
12. AL CLOSES iA continues to flow via DA2, point A remains at 0v, zero state continues

Case Three:-



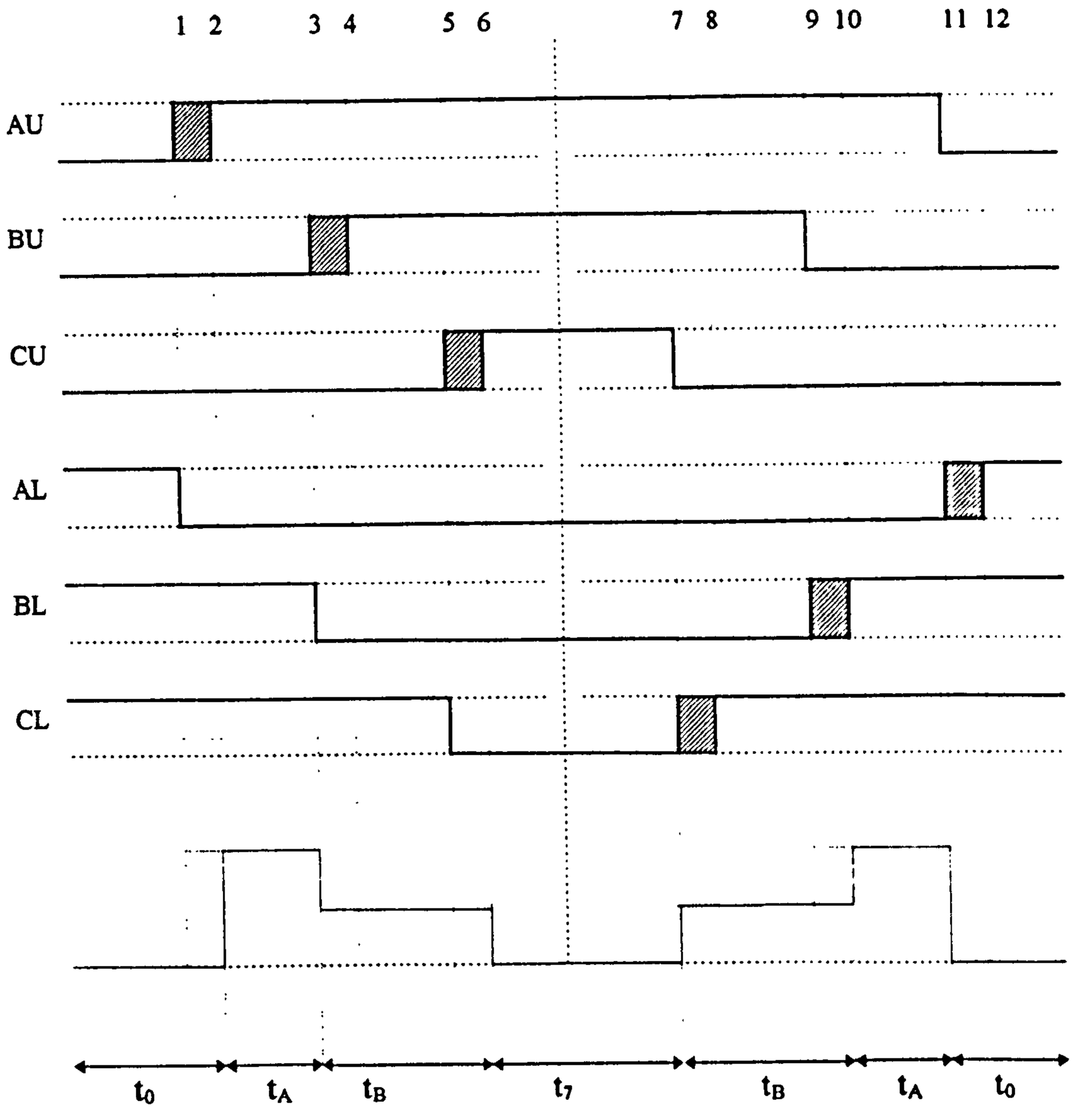


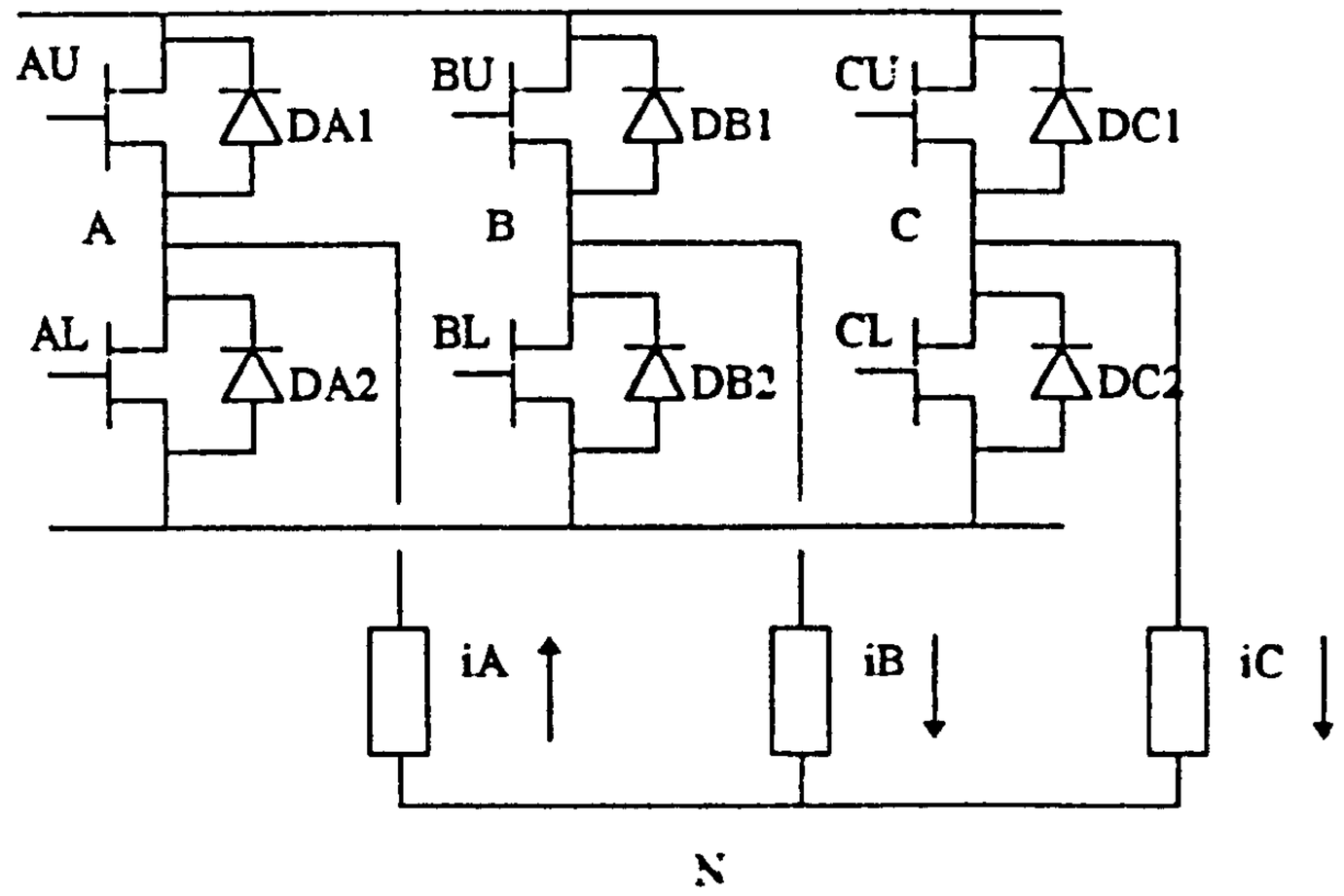
Figure C.3: DeadTime Case Three

To start all lower devices ON.

DA2,BL,DC2 conducting:-

1. AL OPENS iA continues to flow via DA2, point A remains at 0V, zero state continues
2. AU CLOSES iA flows via AU, point A goes to Vdc, state A starts
3. BL OPENS iB flows via DB1, point B goes to Vdc, state B starts
4. BU CLOSES iB continues to flow via DB1, point B remains at Vdc, state B continues
5. CL OPENS iC continues to flow via DC2, point C remains at 0v, state B continues
6. CU CLOSES iC flows via CU, point C goes to Vdc, zero state starts
7. CU OPENS iC flows via DC2, point C goes to 0v, state B starts
8. CL CLOSES iC continues to flow via DC2, point C remains at 0v, State B continues
9. BU OPENS iB continues to flow via DB1, point B remains at Vdc, state B continues
10. BL CLOSES iB flows via BL, point B goes to 0v, state A starts
11. AU OPENS iA flows via DA2, point A goes to 0v, zero state starts
12. AL CLOSES iA continues to flow via DA2, point A remains at 0v, zero state continues

Case Four:-



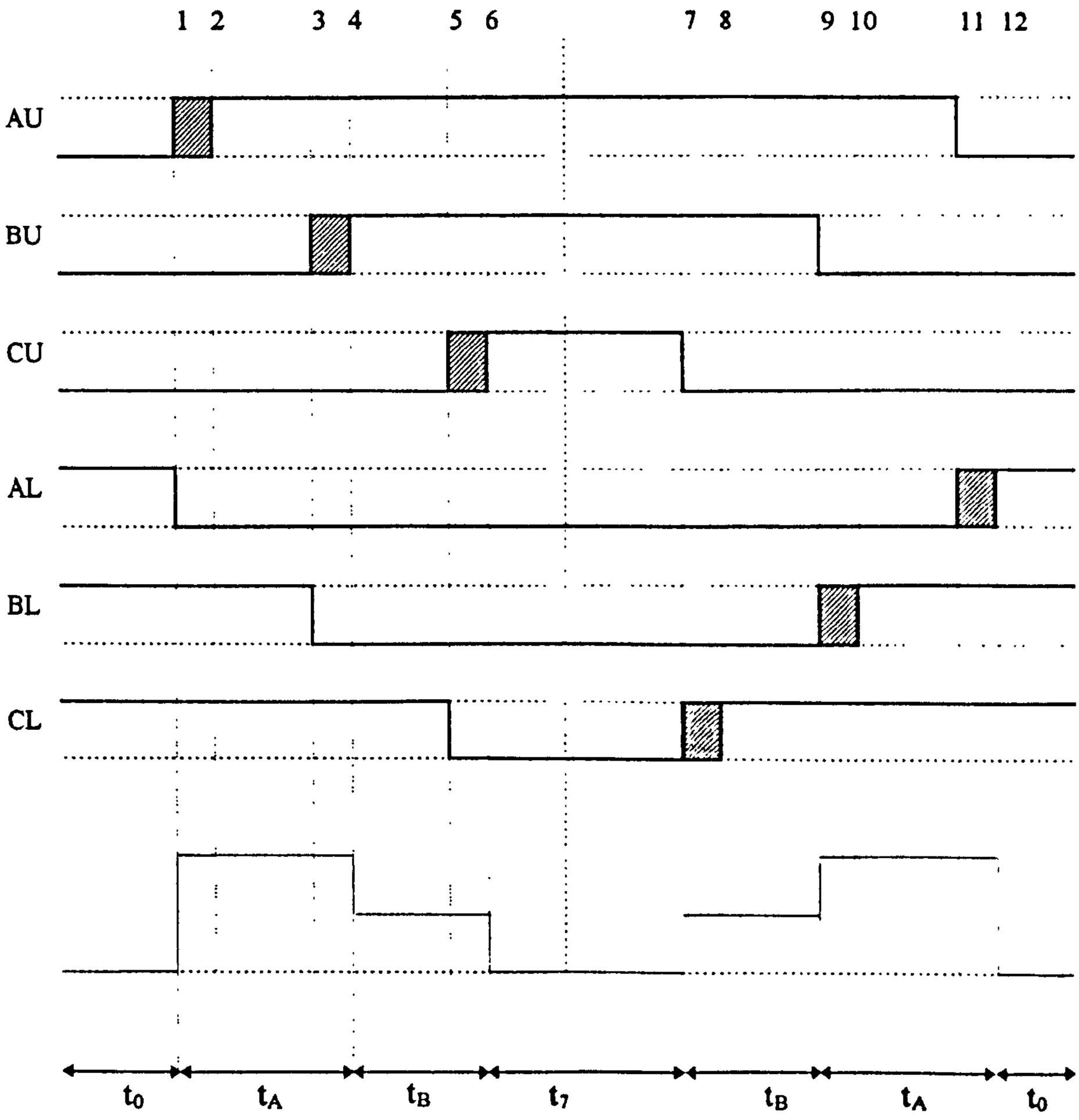


Figure C.4: DeadTime Case Four

To start all lower devices ON.

AL,DB2,DC2 conducting:-

1. AL OPENS iA flows via DA1, point A goes to Vdc, state A starts
2. AU CLOSES iA continues to flow via DA1, point A remains at Vdc, state A continues
3. BL OPENS iB continues to flow via DB2, point B remains at 0v, state A continues
4. BU CLOSES iB flows via BU, point B goes to Vdc, state B starts
5. CL OPENS iC continues to flow via DC2, point C remains at 0v, state B continues
6. CU CLOSES iC flows via CU, point C goes to Vdc, zero state starts
7. CU OPENS iC flows via DC2, point C goes to 0v, state B starts
8. CL CLOSES iC continues to flow via DC2, point C remains at 0v, State B continues
9. BU OPENS iB flows via DB2, point B goes to 0v, state A starts
10. BL CLOSES iB continues to flow via DB2, point B remains at 0v, state A continues
11. AU OPENS iA flows via DA1, point A remains at Vdc, state A continues
12. AL CLOSES iA flows via AL, point A goes to 0v, zero state starts

Case Five:-

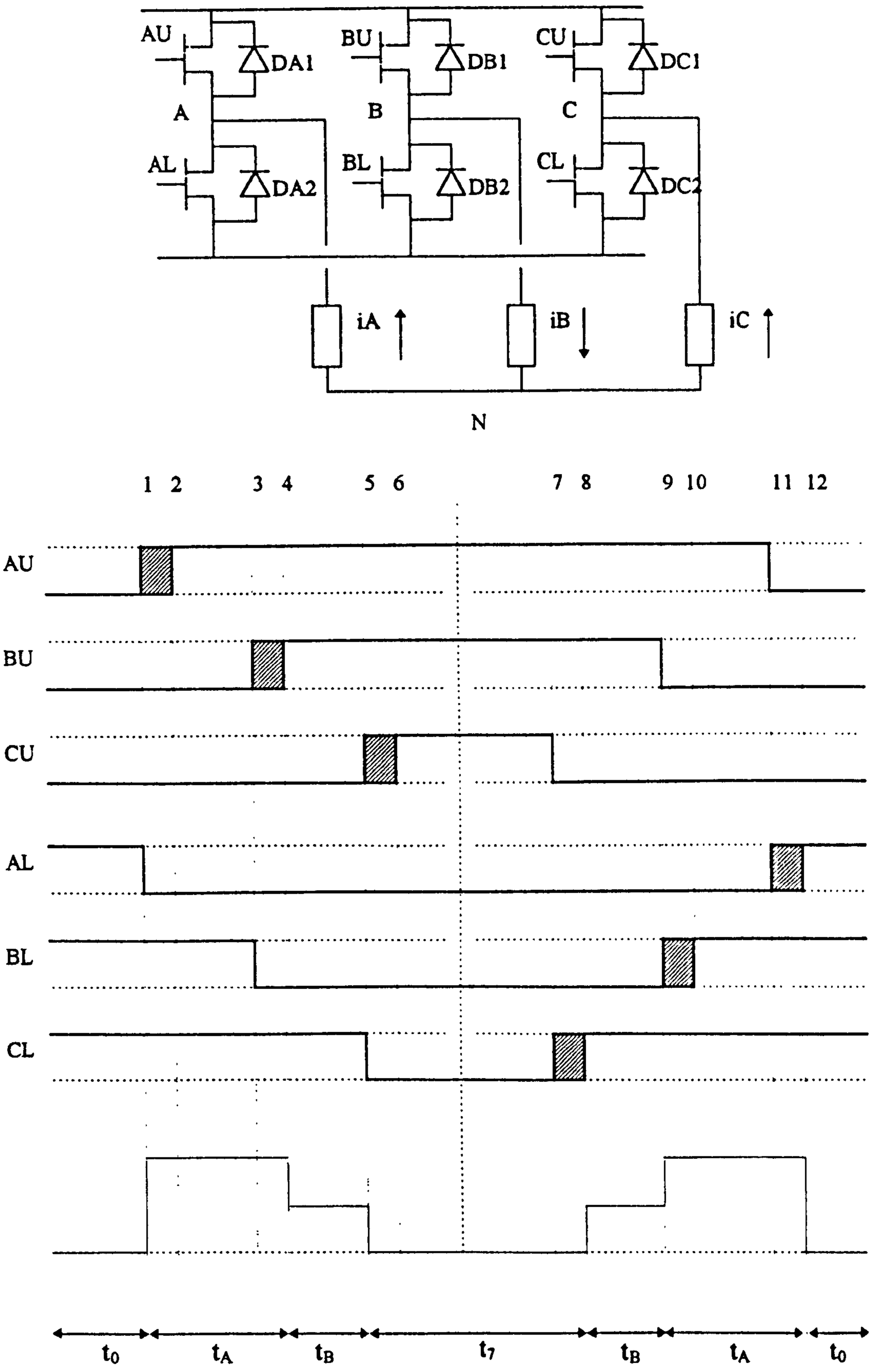


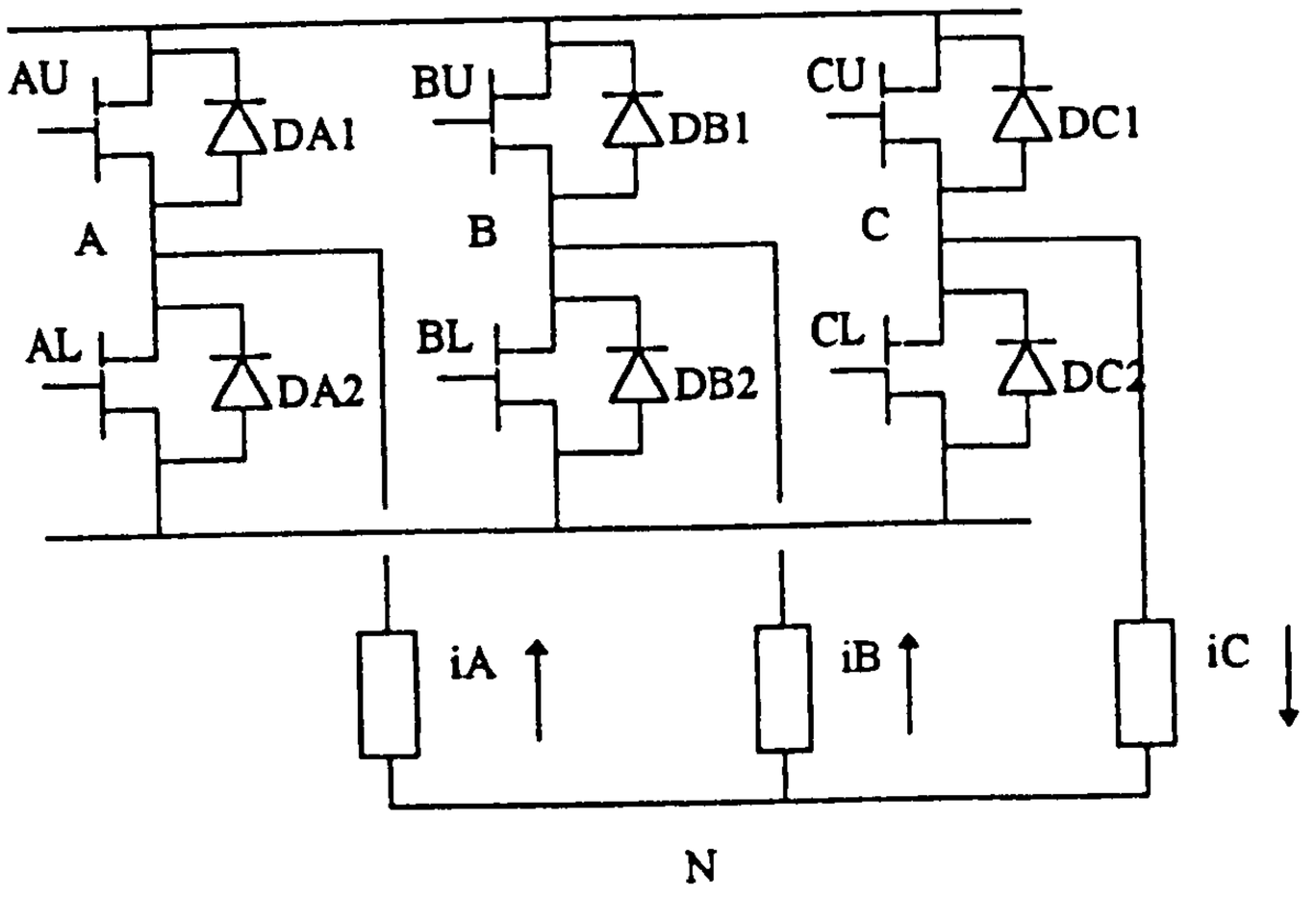
Figure C.5: DeadTime Case Five

To start all lower devices ON.

AL,DB2,CL conducting:-

1. AL OPENS iA flows via DA1, point A goes to Vdc, state A starts
2. AU CLOSES iA continues to flow via DA1, point A remains at Vdc, state A continues
3. BL OPENS iB continues to flow via DB2, point B remains at 0v, state A continues
4. BU CLOSES iB flows via BU, point B goes to Vdc, state B starts
5. CL OPENS iC flows via DC1, point C goes to Vdc, zero state starts
6. CU CLOSES iC continues to flow via DC1, point C remains at Vdc, zero state remains
7. CU OPENS iC continues to flow via DC1, point C remains at Vdc, zero state remains
8. CL CLOSES iC flows via CL, point C goes to 0v, state B starts
9. BU OPENS iB flows via DB2, point B goes to 0v, state A starts
10. BL CLOSES iB continues to flow via DB2, point B remains at 0v, state A continues
11. AU OPENS iA flows via DA1, point A remains at Vdc, state A continues
12. AL CLOSES iA flows via AL, point A goes to 0v, zero state starts

Case Six:-



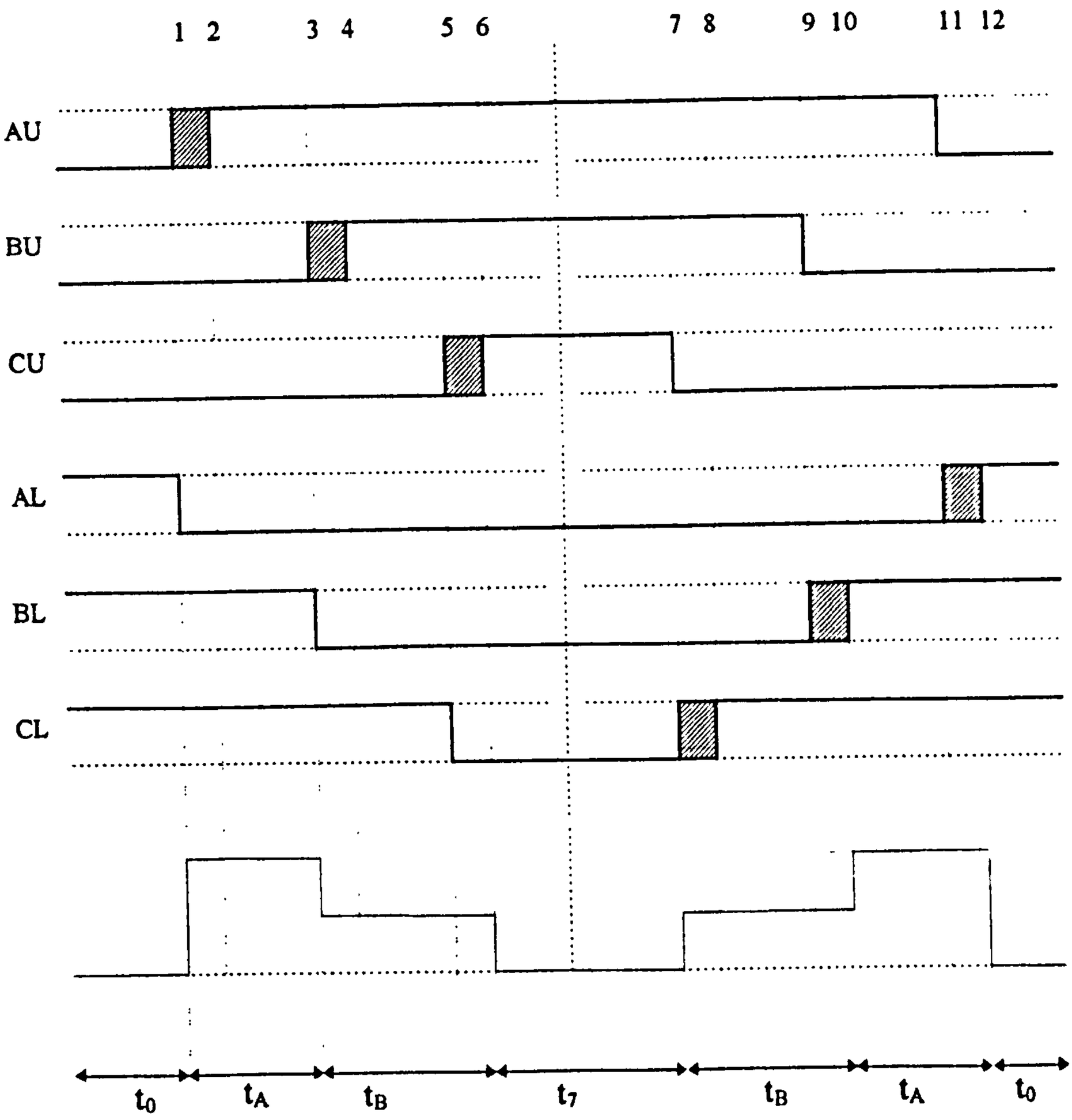


Figure C.6: DeadTime Case Six

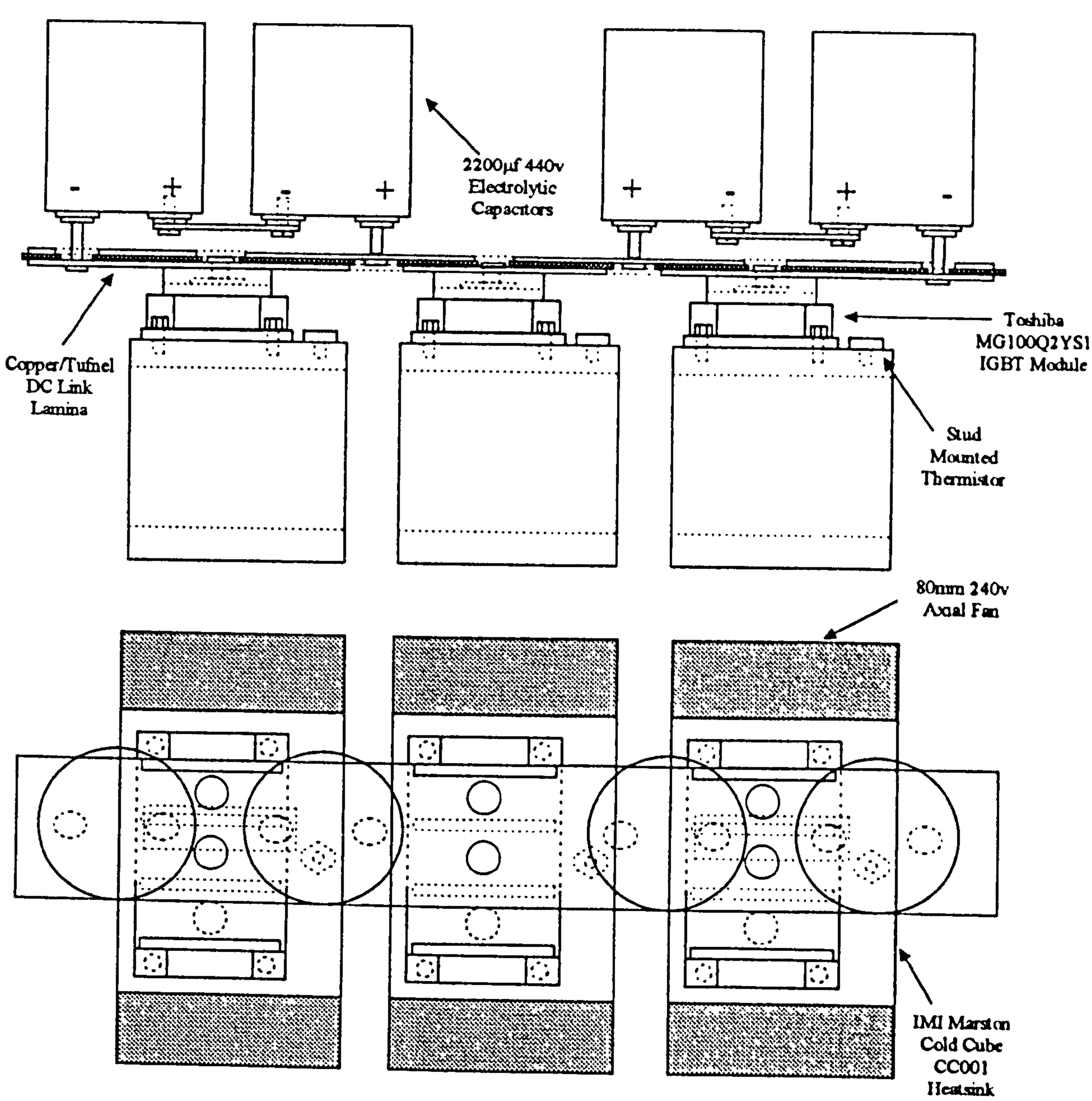
To start all lower devices ON.

AL,BL,DC2 conducting:-

1. AL OPENS iA flows via DA1, point A goes to Vdc, state A starts
2. AU CLOSES iA continues to flow via DA1, point A remains at Vdc, state A continues
3. BL OPENS iB flows via DB1, point B goes to Vdc, state B starts
4. BU CLOSES iB continues to flow via DB1, point B remains at Vdc, state B continues
5. CL OPENS iC continues to flow via DC2, point C remains at 0v, state B continues
6. CU CLOSES iC flows via CU, point C goes to Vdc, zero state starts
7. CU OPENS iC flows via DC2, point C goes to 0v, state B starts
8. CL CLOSES iC continues to flow via DC2, point C remains at 0v, State B continues
9. BU OPENS iB continues to flow via DB1, point B remains at Vdc, state B continues
10. BL CLOSES iB flows via BL, point B goes to 0v, state A starts
11. AU OPENS iA flows via DA1, point A remains at Vdc, state A continues
12. AL CLOSES iA flows via AL, point A goes to 0v, zero state starts

Appendix D

Drive Schematic: IGBT and DC Link Layout



NB:- NOT TO SCALE