

University of Natal

**FEASIBILITY STUDY
OF A
NEURAL NETWORK CURRENT CONTROLLER
FOR A
BOOST RECTIFIER**


by

Cedric Alwyn Worthmann

Submitted in fulfilment of the academic requirements for the degree of Masters of Science in Engineering in the Department of Electrical Engineering of the University of Natal in South Africa.

Date: December 2000

I hereby declare that the material incorporated into this thesis is my own original and unaided work except where specific reference is made by name or in the form of a numbered reference. The work contained herein has not been submitted for a degree at any other university.

Signed:  _____
Cedric Worthmann

With all my love, to my *Mom*, *Dad*, and *Liesel*,
for all their inspiration, support, sacrifices
and encouragement.

Abstract

During the past two decades, Quality of Supply has become a serious problem for Variable Speed Drives in the industrial and commercial sectors. Quality of Supply problems can trip Variable Speed Drives, which results in loss of production, which is a significant problem in the paper and pulp industry. Researchers have proposed that Quality of Supply problems can be minimised in-house, using controlled front end rectifiers (boost rectifier), to maintain a regulated DC link voltage in the Variable Speed Drive configuration, as most faults are created by a varying supply voltage.

This thesis extends the work performed on boost rectifiers by investigating the feasibility of replacing the classical controllers with a Continual Online Trained Artificial Neural Network current controller. The approach adopted in this thesis was to evaluate and extend the work previously performed on conventional boost rectifier current controllers and Continual Online Trained Artificial Neural Network current controlled inverter, at the University of Natal. During this evaluation, the respective controller shortcomings were identified and addressed. Thereafter the Continual Online Trained Artificial Neural Network current controller was modified, according to the control requirements of the boost rectifier, and used as a replacement for the conventional current controller in the boost rectifier system. Finally, the Continual Online Trained Artificial Neural Network current controller was evaluated to assess its viability as a current controller for a boost rectifier.

The concept of implementing the real-time Continual Online Trained Artificial Neural Network current controller using a DSP (Digital Signal Processor) was described, along with the main features and practical limitations of existing commercial DSP's. It is shown that at the time of writing of this thesis, the commercially available DSP's are not powerful enough to implement the Continual Online Trained Artificial Neural Network current controller. However this thesis also shows that it is feasible to implement the real-time controller on the newly released TMS320C67 DSP card.

Acknowledgments

The work presented in this thesis was carried out under the supervision of Mr G Diana of the Department of Electrical Engineering, University of Natal, Durban. I wish to thank Mr Diana for his advice, guidance and support.

I also wish to thank:

- My family and friends for their support throughout my University career;
- My colleges Messrs M.L. Walker, A. Stylo, B. van Blerk, T. Rae, B. Burton, M. Pillay, Ganesh K. Venayagamoorthy , R. Hariparsad;
- The technical staff, particularly Messrs A. Roos, A. G. M. Munnick, G. Loubser and A. Stengel for their assistance;
- The Foundation for Research and development (FRD) in South Africa and the University of Natal, for providing financial support.
- Mrs Fiona Higginson, Dr Derrick Hoch and Professor Broadhurst for providing courage during a trying time.
- Eskom TESP.
- And last, my colleagues from Eskom Electro-technology Test and Demonstration Center, Westmead, for all their support and encouragement.

Table of Contents

Abstract	i
Acknowledgments	ii
Table of Contents	iii
List of Figures and Tables	viii
List of Symbols and Abbreviations	xv

CHAPTER 1 INTRODUCTION

1.1	General	1.1
1.2	Thesis Layout	1.4
1.3	New Formulations	1.6
1.4	Publications	1.6

CHAPTER 2 OVERVIEW OF PREVIOUS WORK

2.1	Introduction	2.1
2.2	Quality of Supply	2.4
2.3	Conventional Rectifiers and their Shortcomings	2.5
2.3.1	Phase-controlled Thyristor Rectifier	2.5
2.3.2	Three-phase Diode Rectifier	2.6
2.4	Improved Rectifier Topologies and Control Schemes	2.7
2.5	Chathury's Conventional Boost Rectifier Controller	2.11
2.6	COT ANN based IM Current Controller	2.15
2.6.1	The Basic Principles of System Identification using ANNs	2.15

2.6.2	Burton's adaptive current control law	2.16
2.7	Summary	2.20

**CHAPTER 3 SIMULATION OF A CONTINUOUSLY ONLINE TRAINED
ARTIFICIAL NEURAL NETWORK CONTROLLED VOLTAGE-
SOURCE INVERTER FED SCIM DRIVE**

3.1	Introduction	3.1
3.2	CASED Simulation of Power Conversion Systems	3.2
3.3	Investigation of a basic COT ANN Controlled VSI fed SCIM Variable Speed Drive	3.5
3.4	Investigation of the performance of a COT ANN Applied to a VSI fed SCIM Variable Speed Drive	3.13
3.5	Summary	3.38

**CHAPTER 4 INVESTIGATION OF A CONTINUOUSLY ONLINE TRAINED
ARTIFICIAL NEURAL NETWORK CURRENT CONTROLLER FOR
A BOOST RECTIFIER SYSTEM**

4.1	Introduction	4.1
4.2	System Outline	4.2
4.2.1	COT ANN Conventions	4.2
4.2.2	Mains Synchronization	4.7
4.2.3	DC link Voltage Controller	4.9
4.3	Investigation of a COT ANN Current Controller for a Boost Rectifier	4.12
4.3.1	ANN Learning Rate	4.20
4.3.2	ANN Voltage Constant	4.22
4.3.3	PWM Switching Frequency	4.25

4.3.4	ANN Sampling Frequency	4.26
4.3.5	Alpha/Beta Current and Voltage Feedback Terms	4.29
4.3.6	Proportional Gain	4.31
4.4	Summary	4.40

CHAPTER 5 FEASIBILITY OF USING A DSP-BASED COT ANN CURRENT CONTROLLER

5.1	Introduction	5.1
5.2	Processor Requirements	5.2
5.2.1	Transputer versus ADC64 DSP card	5.2
5.2.2	Controller Computational Requirements	5.3
5.2.2	Proposed DSP solution	5.11
5.3	Summary	5.14

CHAPTER 6 CONCLUSION AND RECOMMENDATIONS

6.1	General	6.1
6.2	Suggestions for Further Work	6.3

APPENDIX A DERIVATION OF EQUATIONS

A.1	Equations for the Boost Rectifier Power Conversion System in DQ Coordinates	A.1
A.1.1	Three-phase a-b-c reference frame mathematical model	A.1
A.2	Linearisation of the dynamic power balance equation during rectification	A.5
A.3	NARMAX model of the Boost Rectifier	A.6
A.3.1	Continuous Time Electrical Model of the Induction Motor	A.7

A.3.2	Derivation of the Electrical NARMAX Model of the Boost rectifier	A.13
A.3.3	Root-locus Angle and Magnitude Conditions	A.18

APPENDIX B SIMULATION CODE LISTING

B.1	Artificial Neural Network model for SCIM drive	B.1
B.2	Simplified Induction motor model	B.13
B.3	Artificial Neural Network model	B.17
B.4	PWM Hanning Model	B.29
B.5	Induction motor model	B.38
B.6	Three-Phase Sinusoidal Source Model	B.43
B.7	User Model for Neural Network Current Controller for a Boost Rectifier	B.46

APPENDIX C HYPERSIGNAL REAL-TIME BLOCK FUNCTION CODE LISTING

C.1	PWM Control Block Function	C.1
C.1.1	PWM Control Block Include code	C.1
C.1.2	PWM Control Block C-source code	C.5
C.2	ANN Current Controller Block Function	C.13
C.2.1	ANN Current Controller Block Include Code	C.13
C.2.2	ANN Current Controller Block C-source code	C.19

APPENDIX D BOOST RECTIFIER HARDWARE PROFILE

D.1	Overview	D.1
D.2	Practical System Structure	D.2
D.3	The Power Circuit	D.3
D.3.1	The Insulated Gate Bipolar Transistor Boost Rectifier	D.4

D.3.2	Electrical Load	D.5
D.3.3	Three-phase Supply	D.6
D.4	The Control Hardware	D.6
D.4.1	ADC64 Digital Signal Processing Card	D.6
D.4.2	Hanning PWM Interface Card	D.12
D.4.3	Optical Fibre Interface Circuitry	D.18
D.4.4	Signal Sensing and Conditioning Circuitry	D.20

REFERENCES	R.1
-------------------	-----

List of Figures and Tables

Fig. 1.1 Supply voltage dip or outage	1.2
Fig. 1.2 Voltage-Sourced Inverter	1.2
Fig. 2.1 Conventional controller configuration used to control a boost rectifier	2.2
Fig. 2.2 COT ANN controlled VSI fed SCIM drive	2.3
Fig. 2.3 Proposed boost rectifier control loops	2.3
Fig. 2.4 Simplified power network	2.4
Fig. 2.5 Three-phase thyristor rectifier	2.5
Fig. 2.6 Typical phase A operating voltage and current at $\alpha=15^\circ$	2.6
Fig. 2.7 Three-phase diode rectifier	2.7
Fig. 2.8 Boost rectifier configuration	2.11
Fig. 2.9 Chathury's boost rectifier current and voltage controller	2.13
Fig. 2.10 Block diagram of the current control law Eqn. (2.11)	2.19
Fig. 3.1 Structure of a typical industrial power conversion system	3.2
Fig. 3.2 Structure of CASED	3.3
Fig. 3.3 Basic COT ANN controlled power conversion system	3.6
Fig. 3.4 Flow diagram of COT ANN CASED module	3.7
Fig. 3.5 Flow diagram of CASED induction motor module	3.9
Fig. 3.6 Phase A motor supply current (I_{a_supply}) and COT ANN alpha reference current (I_{α_ref}), with a sampling rate of 500 Hz, learning rate of 0.1 pu, c_v of 1.5 pu, and no pretraining at stator frequencies of (a) 1 Hz (CASED and MATLAB results), (b) 50 Hz (CASED) and (c) 100 Hz (CASED) respectively.	3.11
Fig. 3.7 (a) Phase A motor current (I_{a_supply}) and COT ANN alpha reference current (I_{α_ref}), with a sampling frequency of 10 kHz and learning rate of 0.1 pu, (b) Current tracking error	3.12

Fig. 3.8 Block diagram of a COT ANN controlled SCIM drive	3.13
Fig. 3.9 Flow diagram of COT ANN CASED module for control of a VSI fed SCIM VSD	3.15
Fig. 3.10 Flow diagram of PWM CASED module	3.17
Fig. 3.11 Phase A motor supply current (I_{a_supply}) and COT ANN alpha reference current (I_{α_ref}), with sampling frequency of 10 kHz, learning rate of 0.5 pu and C_v of 1.5 pu, at motor frequencies of (a) 1 Hz and (b) 50 Hz respectively	3.19
Fig 3.12 (a) Phase A motor supply current (I_{a_supply}) and COT ANN alpha reference current (I_{α_ref}), with $K = 0.1$ pu, sampling frequency of 1 kHz, learning rate of 0.01 pu, stator frequency of 50 Hz and stator current magnitude of 1 pu, (b) Current tracking error	3.20
Fig 3.13 (a) Phase A motor supply current (I_{a_supply}) and COT ANN alpha reference current (I_{α_ref}), with $K = 0.1$ pu, sampling frequency of 5 kHz, learning rate of 0.01 pu, stator frequency of 50 Hz and stator current magnitude of 1 pu, (b) Current tracking error	3.21
Fig 3.14 (a) Phase A motor supply current (I_{a_supply}) and COT ANN alpha reference current (I_{α_ref}), with $K = 0.1$ pu, sampling frequency of 10 kHz, learning rate of 0.01 pu, stator frequency of 50Hz and stator current magnitude of 1 pu, (b) Current tracking error	3.22
Fig 3.15 (a) Phase A motor supply current (I_{a_supply}) and COT ANN alpha reference current (I_{α_ref}), with $K = 0.6$ pu, sampling frequency of 1 kHz, learning rate of 0.01 pu, stator frequency of 50 Hz and stator current magnitude of 1 pu, (b) Current tracking error	3.24
Fig 3.16 (a) Phase A motor supply current (I_{a_supply}) and COT ANN alpha reference current (I_{α_ref}), with $K = 0.6$ pu, sampling frequency of 5 kHz, learning rate of 0.01 pu, stator frequency of 50Hz and stator current magnitude of 1 pu, (b) Current tracking error	3.25

Fig 3.17 (a) Phase A motor supply current (I_{a_supply}) and COT ANN alpha reference current (I_{α_ref}), with $K = 0.6$ pu, sampling frequency of 10 kHz, learning rate of 0.01 pu, stator frequency of 50Hz and stator current magnitude of 1 pu,	
(b) Current tracking error	3.26
Fig 3.18 (a) Phase A motor supply current (I_{a_supply}) and COT ANN alpha reference current (I_{α_ref}), with $K = 0.9$ pu, sampling frequency of 1 kHz, learning rate of 0.01 pu, stator frequency of 50 Hz and stator current magnitude of 1 pu,	
(b) Current tracking error	3.28
Fig 3.19 (a) Phase A motor supply current (I_{a_supply}) and COT ANN alpha reference current (I_{α_ref}), with $K = 0.9$ pu, sampling frequency of 5 kHz, learning rate of 0.01 pu, stator frequency of 50 Hz and stator current magnitude of 1 pu,	
(b) Current tracking error	3.29
Fig 3.20 (a) Phase A motor supply current (I_{a_supply}) and COT ANN alpha reference current (I_{α_ref}), with $K = 0.9$ pu, sampling frequency of 10 kHz, learning rate of 0.01 pu, stator frequency of 50 Hz and stator current magnitude of 1 pu,	
(b) Current tracking error	3.30
Fig. 3.21 Current dependence, $T_s=10$ kHz, $B=0.01$ pu, $C_v=0.6$ pu, and motor frequency $f = 50$ Hz	3.32
Fig. 3.22 Motor frequency dependence, $T_s=10$ kHz, $B=0.01$ pu, $C_v=0.6$ pu, and motor current $I=0.5$ pu	3.33
Fig. 3.23 ANN learning rate dependence, $T_s=10$ kHz, $C_v=0.6$ pu, $I=0.5$ pu and $f=50$ Hz	3.34
Fig. 3.24 ANN voltage constant dependence, $T_s=10$ kHz, $B=0.01$ pu, $I=0.5$ pu and $f=50$ Hz	3.35
Fig. 3.25 ANN sampling frequency dependence, $C_v=0.6$ pu, $B=0.01$ pu, $f=50$ Hz and $I=0.5$ pu	3.36
Fig. 3.26 (a) Phase A motor supply current (I_{a_supply}) and COT ANN alpha reference current (I_{α_ref}), with $K = 0.6$ pu, sampling frequency of 8 kHz, learning rate of 0.01 pu, stator frequency of 50 Hz, and stator current magnitude of 1 pu, (b) Current	

tracking error	3.37
Fig. 4.1 Conventional control loops	4.2
Fig. 4.2 Phase conventions for a three phase inverter	4.3
Fig. 4.3 Alpha/Beta reference frame	4.3
Fig. 4.4 Boost rectifier configuration	4.4
Fig. 4.5 Continual Online training for adaptive system identification	4.6
Fig. 4.6 Converting the conventional current controller to a COT ANN current controller	4.7
Fig. 4.7 Simplified DC voltage control block diagram	4.9
Fig. 4.8 Root locus of stabilised voltage control system	4.11
Fig. 4.9 ANN controlled Boost rectifier block diagram	4.12
Fig. 4.10 Flow diagram of COT ANN CASED module for boost rectifier with voltage controller.	4.14
Fig. 4.11 Flow diagram of PWM CASED module	4.16
Fig. 4.12 Three-phase source flow diagram	4.16
Fig. 4.13(a) Phase A supply current (I_{sa}) and COT ANN alpha reference current (I_{α_ref}) (internal reference variable), (b) Current tracking error	4.17
Fig. 4.14(a) Phase A supply current (I_{sa}) and COT ANN alpha reference current (I_{α_ref}), (b) DC link Voltage	4.18
Fig 4.15 A (I_{sa}), B (I_{sb}), and C (I_{sc}) phase currents	4.19
Fig. 4.16 (a) Phase A supply current (I_{sa}) and COT ANN alpha reference current (I_{α_ref}), with $C_v = 2.0$ pu, $f_s = 4$ kHz, $f_{han} = 4$ kHz, and Learning Rate of 0.01 pu, (b) Current tracking error	4.20
Fig. 4.17 (a) Phase A supply current (I_{sa}) and COT ANN alpha reference current (I_{α_ref}), with $C_v = 2.0$ pu, $f_s = 4$ kHz, $f_{han} = 4$ kHz, and Learning Rate of 0.1 pu, (b) Current tracking error	4.21
Fig. 4.18 (a) Phase A supply current (I_{sa}) and COT ANN alpha reference current (I_{α_ref}), with $C_v = 10.0$ pu, $f_s = 4$ kHz, $f_{han} = 4$ kHz, and Learning Rate of 0.05 pu,	

(b) Current tracking error 4.22

Fig. 4.19 (a) Phase A supply current (I_{sa}) and COT ANN alpha reference current (I_{α_ref}),
with $C_v = 1.0$ pu, $f_s = 4$ kHz, $f_{han} = 4$ kHz, and Learning rate of 0.05 pu,

(b) Current tracking error 4.23

Fig. 4.20 (a) Phase A supply current (I_{sa}) and COT ANN alpha reference current (I_{α_ref}),
with $C_v = 1.1$ pu, $f_s = 4$ kHz, $f_{han} = 4$ kHz, and Learning rate of 0.12 up, (b) Current
tracking error 4.24

Fig. 4.21 (a) Phase A supply current (I_{sa}) and COT ANN alpha reference current (I_{α_ref}),
with $C_v = 1.1$ pu, $f_s = 4$ kHz, $f_{han} = 8$ kHz, and Learning rate of 0.12 pu, (b) Current
tracking error 4.25

Fig. 4.22 (a) Phase A supply current (I_{sa}) and COT ANN alpha reference current (I_{α_ref}),
with $C_v = 1.1$ pu, $f_s = 10$ kHz, $f_{han} = 10$ kHz, and Learning Rate of 0.12 pu,
(b) Current tracking error 4.26

Fig. 4.23 (a) Phase A supply current (I_{sa}) and COT ANN alpha reference current (I_{α_ref}),
with $C_v = 1.1$ pu, $f_s = 15$ kHz, $f_{han} = 15$ kHz, and Learning rate of 0.12 pu,
(b) Current tracking error 4.27

Fig. 4.24 (a) Phase A supply current (I_{sa}) and COT ANN alpha reference current (I_{α_ref}),
with $C_v = 1.1$ pu, $f_s = 4$ kHz, $f_{han} = 2$ kHz, and Learning rate of 0.12 pu, (b) Current
tracking error 4.28

Fig. 4.25 (a) Phase A supply current (I_{sa}) and COT ANN alpha reference current (I_{α_ref}),
with $C_v = 1.1$ pu, $f_s = 4$ kHz, $f_{han} = 2$ kHz, and Learning rate of 0.12 pu, no $I(k-1)$
terms, (b) Current tracking error 4.29

Fig. 4.26 (a) Phase A supply current (I_{sa}) and COT ANN alpha reference current (I_{α_ref}),
with $C_v = 0.1$ pu, $f_s = 4$ kHz, $f_{han} = 2$ kHz, and Learning Rate of 0.12 pu, no $I(k-1)$
terms, (b) Current tracking error 4.30

Fig. 4.27 (a) Phase A supply current (I_{sa}) and COT ANN alpha reference current (I_{α_ref}),
with $C_v = 1.1$ pu, $f_s = 4$ kHz, $f_{han} = 4$ kHz, and $B = 0.12$ pu, $K_p = 0.01$ pu, (b) Current
tracking error 4.31

Fig. 4.28 (a) DC link voltage, (b) DC link tracking error, with $C_v = 1.1$ pu, $f_s = 4$ kHz, $f_{han} = 2$ kHz, $B = 0.12$ pu, $K_p = 0.01$ pu	4.32
Fig. 4.29 (a) Phase A supply current (I_{sa}) and COT ANN alpha reference current (I_{α_ref}), with $C_v = 1.1$ pu, $f_s = 4$ kHz, $f_{han} = 2$ kHz, $B = 0.12$ pu and $K_p = 0.15$ pu, (b) Current tracking error	4.33
Fig. 4.30 (a) DC link voltage, (b) DC link tracking error, with $C_v = 1.1$ pu, $f_s = 4$ kHz, $f_{han} = 2$ kHz, $B = 0.12$ pu, $K_p = 0.15$ pu	4.34
Fig. 4.31 (a) Phase A supply current (I_{sa}) and COT ANN alpha reference current (I_{α_ref}), with $C_v = 1.1$ pu, $f_s = 4$ kHz, $f_{han} = 2$ kHz, $B = 0.12$ pu and $K_p = 1.0$ pu, (b) Current tracking error	4.35
Fig. 4.32 (a) DC link voltage, (b) DC link tracking error, with $C_v = 1.1$ pu, $f_s = 4$ kHz, $f_{han} = 2$ kHz, $B = 0.12$ pu, $K_p = 1.0$ pu	4.36
Fig. 4.33 (a) Phase A supply current (I_{sa}) and COT ANN alpha reference current (I_{α_ref}), with $C_v = 1.1$ pu, $f_s = 4$ kHz, $f_{han} = 2$ kHz, $B = 0.12$ pu and $K_p = 0.15$ pu, (b) Current tracking error	4.37
Fig. 4.34 (a) DC link voltage, (b) DC link tracking error, with $C_v = 1.1$ pu, $f_s = 4$ kHz, $f_{han} = 2$ kHz, $B = 0.12$ pu, $K_p = 0.15$ pu	4.38
Fig. 5.1 Closed loop COT ANN current control scheme for a boost rectifier	5.4
Fig. 5.2 Flow diagram of the Hypersignal real-time COT ANN controller, including voltage control, for a boost rectifier.	5.5
Fig. 5.3 Flow diagram of real-time PWM block	5.7
Fig. 5.4 Calculation cycle of the combined PWM/COT ANN control software for the boost rectifier	5.8
Fig. 5.5 COT ANN matrix and calculation time cycle	5.9
Fig. 5.6 (a) Phase A supply current (I_{sa}) and ANN alpha reference current (I_{α_ref}), with $C_v = 1.1$, $T_s = 1.5$ kHz, $T_{han} = 0.75$ kHz, $B = 0.12$ and $K_p = 0.15$, (b) Current tracking error	5.10
Fig. 5.7 Practical results: Phase A supply current (I_{sa}) and ANN alpha reference current	

(I_{α_ref}), with $C_v = 1.1$, $T_s = 1.5$ kHz, $T_{han} = 0.75$ kHz, $B = 0.12$ and $K_p = 0.15$ 5.11

Fig. 5.8 Calculation cycle of the combined PWM/COT ANN control software for the
boost rectifier 5.13

Fig. A.1 FCR-based power conversion system A.1

Fig. A.2 The dq axis reference frame A.7

Fig. A.3 Control system A.18

Fig. D.1 Structure of practical system D.2

Fig. D.2 IGBT power module D.4

Fig. D.3 IGBT boost rectifier and associated driver modules D.5

Fig. D.4 ADC64 Block Diagram D.7

Fig. D.5 ADC64 PCI bus DSP card layout D.8

Fig. D.6 Analog input circuitry D.10

Fig. D.7 PWM circuit diagram D.16

Fig. D.8 PWM Hanning ISA bus card D.17

Fig. D.9 Cross section of an emitter diode D.19

Fig. D.10 Optic receiver board D.20

Fig. D.11 Current sensing circuit diagram D.21

Fig. D.12 Voltage sensing circuit D.22

Fig. D.13 Current and voltage sensing and conditioning boards D.23

List of Symbols and Abbreviations

ABBREVIATIONS

AC	Alternating Current
ANN	Artificial Neural Network
CASED	Computer-aided Analysis and Simulation of Electrical Drives
COT	Continuous Online Trained
DC	Direct Current
DSP	Digital Signal Processor(ing)
DLL	Dynamic Linker Library
FCR	Forced-Commutated Rectifier
PCC	Point of Common Coupling
PI	Proportional/Integral
PID	Proportional/Integral/Differential
PWM	Pulse-width Modulation
QoS	Quality of Supply
SCIM	Squirrel Cage Induction Motor
VSD	Variable Speed Drive
VSI	Voltage-Source Inverter

SYMBOLS

L_s	Line Inductance (H)
R_s	Line Resistance (Ω)
R_0	Load Resistance (Ω)
$R_d(s)$	Transfer Function of d axis current controller
$R_q(s)$	Transfer Function of q axis current controller
C	Capacitance (μF)
$\underline{f}_{cl}(k)$	Vector Function defined by motor parameters
ζ	Damping Factor
MagUs	Magnitude of Alpha/Beta Voltage (V)
ω_e	Line Frequency in Radians Seconds
$\omega_r(k)$	per unit value of the shaft speed at time (k)
$\lambda_{ds}, \lambda_{qs}$	are the d and q axis stator flux linkages (in weber-turns),
$\lambda_{dr}, \lambda_{qr}$	are the d and q axis rotor flux linkages (in weber-turns),
ω_{dq}	is the arbitrary angular speed rotation (in rad/s) of the d,q axes,
R_1, R_2	are the stator and rotor resistance (in ohms), and
p	is the differential operator d/dt
L_m	denotes the stator to rotor mutual inductance (in henry's)
L_{11}, L_{22}	are the stator to rotor self inductances (in henry's).
B	COT ANN Learning Rate
C_v	COT ANN Voltage Constant
T_s	COT ANN Sampling Frequency
K	COT ANN Scaling Factor
U_{sa}	Phase A supply voltage (V)
U_{sb}	Phase B supply voltage (V)

U_{sc}	Phase C supply voltage (V)
U_{ua}	Phase A voltage at input to converter (V)
U_{ub}	Phase B voltage at input to converter (V)
U_{uc}	Phase C voltage at input to converter (V)
u_{ud}	Instantaneous d-axis voltage at input to converter (V)
u_{sd}	Instantaneous d-axis supply voltage (V)
u_{uq}	Instantaneous q-axis voltage at input to converter (V)
u_{sq}	Instantaneous q-axis supply voltage (V)
u_{sz}	Instantaneous zero voltage (V)
U_{dc}	DC Link Voltage (V)
U_0	Zero Voltage (V)
U_1	RMS phase voltage (V)
U_α	Alpha Supply Voltage (V)
U_β	Beta Supply Voltage (V)
$U_{u\alpha}$	Alpha Voltage at input to converter (V)
$U_{u\beta}$	Beta Voltage at input to converter (V)
$\underline{v}(k)$	Per unit value of the alpha/beta stator voltage applied at time (k)
V_{base}	Voltage Base Values (V)
U_m	Maximum Supply voltage (V)
v_{ds}, v_{qs}	are the d and q axis stator voltages (V),
v_{dr}, v_{qr}	are the d and q axis rotor voltages (V),
I_{α_ref}	Alpha Reference Current (pu)
I_{β_ref}	Alpha Reference Current (pu)
I_{ds_ref}	d-axis Reference Current (pu)
I_{qs_ref}	q-axis Reference Current (pu)
i_{sa}	Phase A Supply Current (pu)
i_{sb}	Phase B Supply Current (pu)

i_{sc}	Phase C Supply Current (pu)
I_{amot}	Motor Stator α Current (pu)
I_{bmot}	Motor Stator β Current (pu)
i_{α}	Instantaneous alpha current (pu)
i_{β}	Instantaneous beta current (pu)
$i(k)$	per unit value of alpha/beta stator current vector at time (k)
i_{ds}, i_{qs}	are the d and q axis stator currents (in amperes),
i_{dr}, i_{qr}	are the d and q axis rotor currents (in amperes),

CHAPTER 1

INTRODUCTION

1.1 General

In South Africa most of the Electrical Energy generation stations are situated near coal mines located in remote areas in Mpumalanga and Gauteng (in the Johannesburg area), necessitating long transmission and distribution power lines for supply to the coastal regions (Durban, Cape Town, Richards Bay etc.) [ELECTRON1]. Over the past two decades it has been noted, by Eskom (South Africa's Energy Utility) that the long power lines are susceptible to changes from tropical weather conditions, run away fires, network switching, harmonic interference and many other factors which may cause faults or disturbances on the lines (i.e., Quality of Supply problems) [AFRICAN EN1].

Initially, poor Quality of supply (QoS) was not classified as a severe problem in industry, because there was sufficient spare generating capacity to clear faults without affecting Eskom's consumers [ELEKTRON1]. Unfortunately over the past two decades, South Africa has reached the limit of its existing generating capacity, and QoS has become a noticeable and costly problem for the industrial and commercial sectors respectively [ELECTRON2].

An example is the paper and pulp industry, where a voltage dip may cause paper winders and unwinders to become unsynchronized causing the paper to tear, resulting in a loss of production. This example shows that Variable Speed Drives (VSDs) are susceptible to QoS phenomenon [ENSLIN1] such as voltage dips, trips and outages (Fig. 1.1), caused by the faults and disturbances on the supply lines [VANZYL1]. The reason for this is that voltage dips cause the VSD's DC link voltage to decrease, thereby reducing the inverter output voltage, hence the motor torque (motor torque is proportional to the square of the voltage in certain configurations). The voltage and torque reduction affects the control capabilities of VSDs [NASAR1].

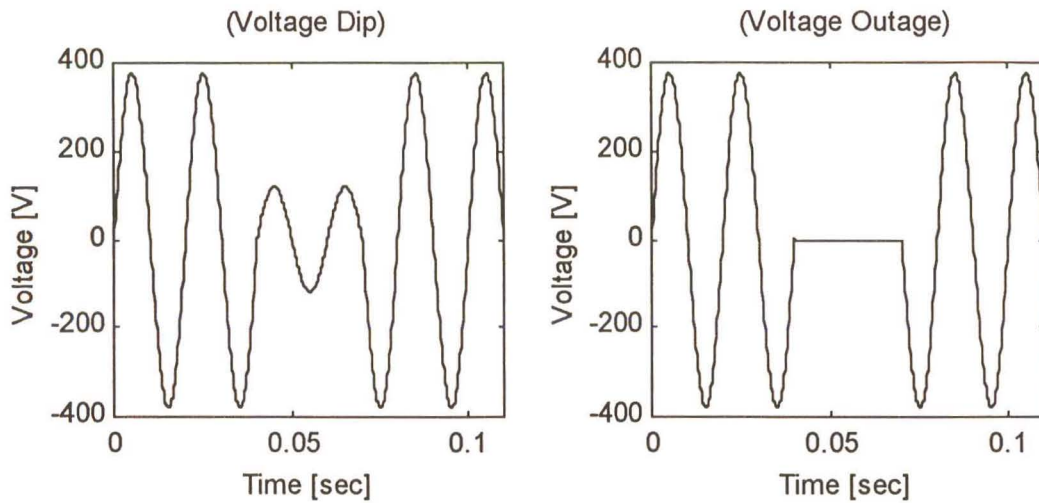


Fig. 1.1 Supply voltage dip or outage

Modern AC Variable Speed Drives (VSDs) utilise power electronic converters to control the flow of electrical energy between the source and load with the most commonly used topology being the cascaded VSI rectifier-inverter configuration shown in Fig. 1.2.

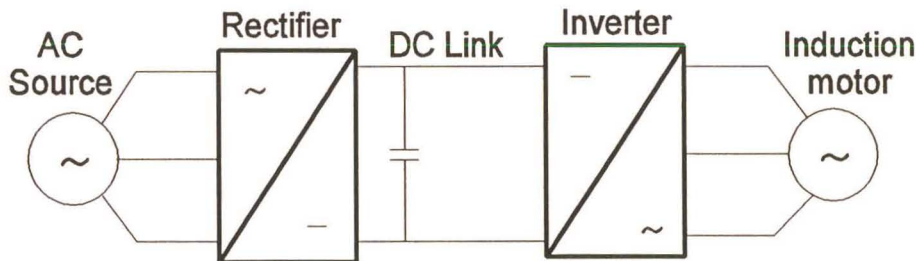


Fig. 1.2 Voltage-Sourced Inverter

In the above configuration, the rectifier normally consists of either a six-pulse diode rectifier, or a phase controlled thyristor rectifier, both of which cause non-sinusoidal supply currents, resulting in harmonic distortion and non-unity power factor operation [CHATHURY1-3], [GREEN1], [WU1], [HABETLER1], which can create QoS problems upstream for other energy consumers.

In the above topology voltage dips in the source generally reflect themselves in the VSD, as the VSD contains an unregulated DC link voltage, which translates the problem to the motor controller [BROD1], as the DC link voltage appears as a gain in the current control loop. There are basically three strategies that can be used to overcome the systems dynamic problem, namely:

- 1) place a robust controller at the inverter end of the VSD, or
- 2) try to maintain the DC link constant via the use of a controlled boost rectifier, or
- 3) use a combination of 1 and 2 .

The decreasing cost, and increasing controllability of power electronic devices [ELEKTRON3], [BOSE1] has to lead the realisation of a new generation of Voltage-Source Forced-Commutated Rectifiers (FCR's) Such FCR's are capable of not only reducing the harmonic distortion and power factor problems inherent in the diode and thyristor rectifiers [GREEN1-2], but can also be used to reduce the effects of QoS. FCR's are controlled using conventional PI and PID controllers, which are tuned using classical design methods (such as the pole placement method) [CHATHURY1-3] which suggests that the controller will provide the desired control, but only in the localised region in which the controller was tuned. This means that the controller can not be interchanged from VSD to VSD, without first re-tuning the operating points. Ideally a self-adapting controller is required, so that a single controller can be used in multiple systems, with little or no re-tuning/re-programming.

In light of this, the School of Electrical Engineering at the University of Natal, proposed the topic of a Continual Online Trained (COT) Artificial Neural Network (ANN) current controlled boost rectifier for this thesis, as an extension of previous research performed by Chathury [CHATHURY1-3] Wishart [WISHART1-2] and Burton [BURTON1-3]. The COT ANN was chosen, as it would provide a system capable of being applied to a variety of topologies, with little or no re-tuning, due to its self commissioning and continual online training properties. Furthermore the boost rectifier was chosen for its ability to boost its DC link voltage above the rectifiers nominal operating voltages for limited time periods. This enables the system to achieve point 2 listed above, if any QoS problems are experienced.

The approach adopted in this thesis was to combine, evaluate and extend the work by Chathury [CHATHURY1-3], and Burton [BURTON1-2]. During this evaluation, the respective controller [BURTON1], [CHATHURY1] shortcomings were identified and addressed. Thereafter the COT ANN current controller was changed, according to the control requirements of the boost rectifier, and used as a replacement for the conventional current controller in the boost rectifier system

developed by Chathury [CHATHURY1-3]. Finally the COT ANN current controller is investigated, to assess its use as a current controller for a boost rectifier. The comparison of the conventional and COT ANN current controller will not be covered in this thesis as it is beyond the scope of the thesis.

1.2 Thesis Layout

The basic outline of this thesis is as follows:

Chapter 2

Chapter 2 provides a brief overview of the pertinent theory and results of the work performed by Chathury [CHATHURY1-3] and Burton [BURTON1-2] on Voltage Source boost rectifier's and COT ANN current controllers respectively. Furthermore, it describes the basic structure, inputs and outputs that are required by the Conventional current controller and COT ANN current controller respectively, for use in the development of the COT ANN current controller for the boost rectifier in Chapter 4.

Chapter 3

Chapter 3 presents the results of two simulation studies performed using the existing COT ANN current controlled Voltage-Source Inverter (VSI) fed Squirrel cage induction motor (SCIM) model, developed by Burton [BURTON1-2]. The first set of simulations, replicate the simulations performed by Burton [BURTON1] (in this model the inverter and PWM controller are omitted), in Computer-aided Analysis and Simulation of Electrical Drives (CASES), in order to become familiar with the new simulation package [CASES1], [KLEINHANS1-4]. The second set of comprehensive simulation study uses Burton's COT ANN controlled VSI fed SCIM drive model, but includes the inverter and PWM switching block in the model, to gain an understanding of the nature and form of the COT ANN while operating under the non-linear effects of the PWM plus the switching of the inverter. The simulations are discussed and analysed to obtain familiarity with the COT ANN current controller developed by Burton and to evaluate the conditions under which the COT ANN can operate and achieve current convergence (while the non-linear characteristics of the inverter are included in the model).

Chapter 4

In this chapter the structure of the COT ANN current controller developed by Burton [BURTON1] is described, along with the changes required, to modify the COT ANN current controller to control the boost rectifier, making use of the mathematical model of a boost rectifier discussed in Chapter 2. The COT ANN boost rectifier is then simulated to verify its effectiveness in controlling a boost rectifier.

Chapter 5

This chapter investigates the feasibility of implementing the real-time COT ANN current controller using a single processor based DSP. Furthermore, it shows why commercial DSP's, available at the time of writing this thesis, are not powerful enough to implement the COT ANN current controller.

Chapter 6

The work presented in this thesis is summarised, and further work is proposed.

1.3 New Formulations

This thesis investigates and utilises the work of Wishart, Burton, Chathury, Harashima and Pinheiro, performed on conventional current controllers [CHATHURY1-3] for boost rectifiers, and on COT ANN current controllers [BURTON1-3], [WISHART1-2], [PINHEIRO1], [HARASHIMA1] for inverters, in order to develop COT ANN current controller for a boost rectifier.

New formulations are as follows:

- 1) The existing COT ANN current controller for the SCIM drive developed by Burton and Wishart is simulated, including all the non-linear effects of the PWM and inverter in the model. The COT ANN current controllers inputs and outputs are then modified, the controller is tuned to obtain efficient and reliable system convergence, up to motor frequencies of 350 Hz.
- 2) A COT ANN current controller is then developed for application to a boost rectifier.

1.4 Publications

During the research period, a number of papers were published. [WORTHMANN1-2] describe and analyse the results of the evaluation performed for the COT ANN current controlled SCIM drive system developed by Burton, detailing the shortcomings of the practical system. [WORTHMANN2] discusses the modification process used to realise the new COT ANN current controller for a boost rectifier, and analyses simulation results to demonstrate the transient response and tracking capability of the controller.

CHAPTER 2

OVERVIEW OF PREVIOUS WORK

2.1 Introduction

The objective of this research, is to investigate the feasibility of using a COT ANN current controller in a conventional boost rectifier configuration, as an extension of the research performed by Chathury [CHATHURY1-3] and Burton [BURTON1-3]. The approach adopted is to first review the research performed by both Chathury [CHATHURY1-3] (on traditional current controlled boost rectifiers) and by Burton[BURTON1-2] (on COT ANN current controlled inverters), highlighting their respective shortcomings. Thereafter the COT ANN controller is used as a direct replacement for the conventional current controller in Chathury's system. Finally the COT ANN current controller is tuned for the boost rectifier.

The research performed by Chathury [CHATHURY1-3] provides an understanding of the boost rectifier's structure and its voltage and current control requirements. Chathury developed a microprocessor based voltage and current controller (Fig. 2.1), with the following characteristics:

- a. unity power factor operation,
- b. a constant DC link voltage, and
- c. sinusoidal supply currents.

The information provided by Chathury on PWM switching techniques, boost rectifiers, and current controllers is also reviewed in this chapter. Furthermore, Quality of Supply (QoS) problems, and their possible causes are reviewed in this chapter, whereafter conventional rectifier structures are reviewed, identifying the non-sinusoidal line current and non-unity power factor problems, which generates QoS problems for other electrical users.

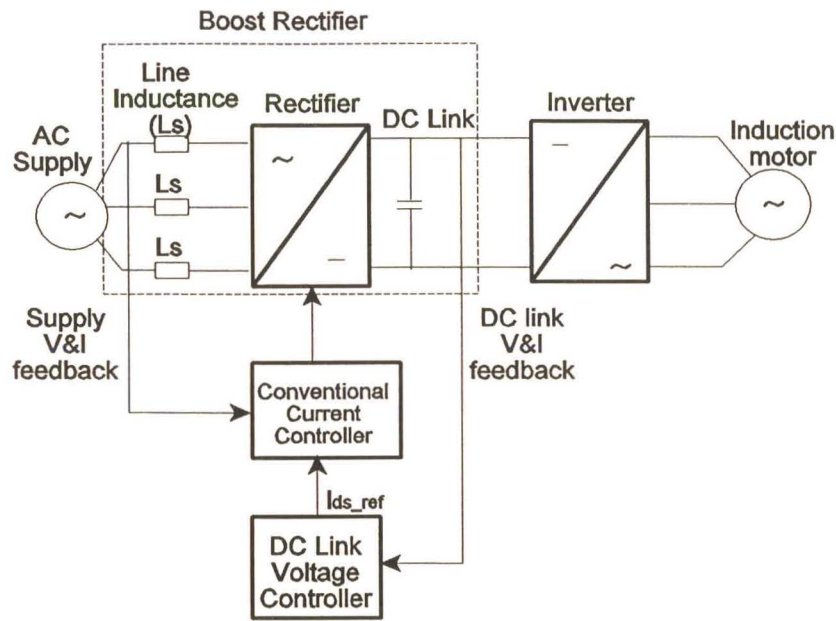


Fig. 2.1 Conventional controller configuration used to control a boost rectifier (Chathury's system)

The COT ANN current controller, developed by Burton [BURTON1-2] to control a voltage-source inverter fed SCIM drive (Fig. 2.2), is then analysed to address and overcome the following:

- 1) a practical implementation of the COT ANN current controller, at a sampling rate of 500 Hz, could not track stator currents of frequencies above 3 Hz, and
- 2) the COT ANN current controller simulation.

The aforementioned research is then extended, to realise an adaptive COT ANN current controller for use with boost rectifiers. This is done by replacing the boost rectifier current controller, shown in Fig. 2.1, with the COT ANN current controller developed by Burton [BURTON1-2] as shown in Fig. 2.3. It should be noted that in Fig. 2.3 the COT ANN current controller only replaces the conventional current controller, and not the voltage controller in Chathury's boost rectifier system. In the past a variety of current controllers [CHATHURY1], [GREEN1], [PINHEIRO1], have been investigated for boost rectifier control, but none other than Burton [BURTON3-4], have yet considered a COT ANN current controller.

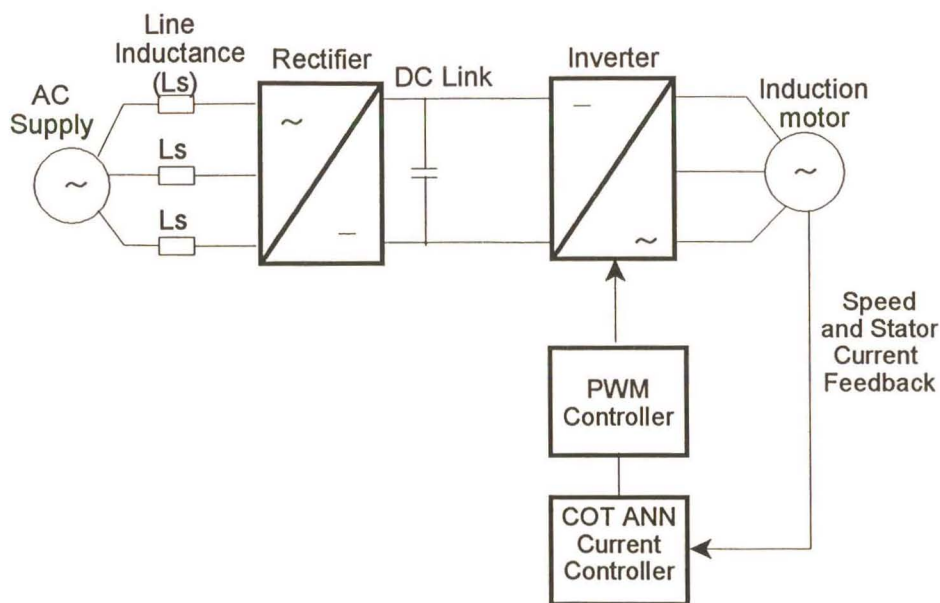


Fig. 2.2 COT ANN controlled VSI fed SCIM drive (Burton's system)

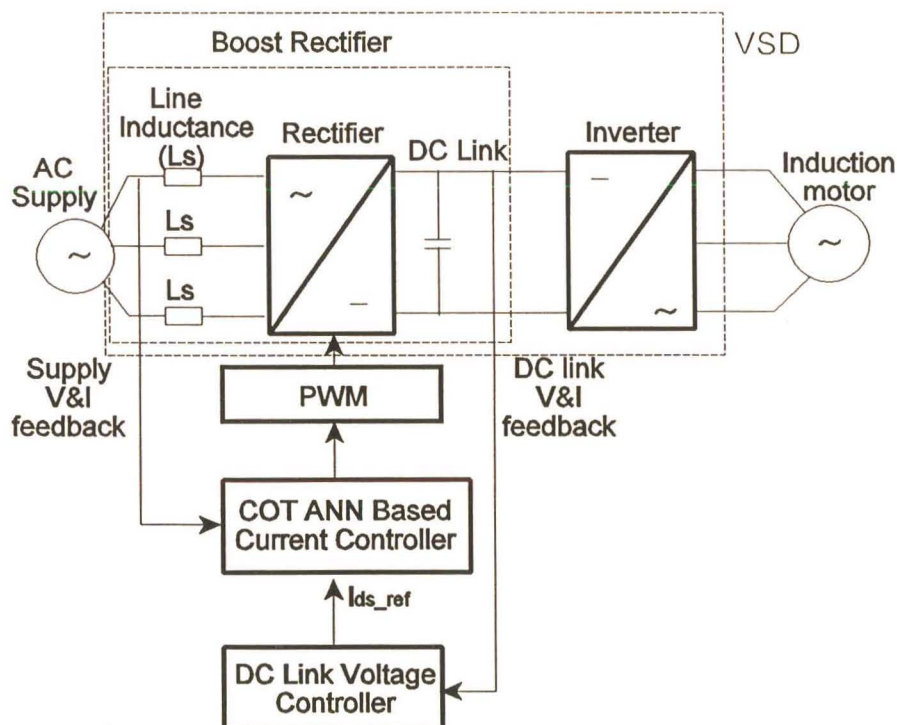


Fig. 2.3 Proposed boost rectifier control loops (Candidates system)

The next section presents literature on QoS problems, where emphasis is placed on non-sinusoidal line currents, voltage distortion and non-unity power factor.

2.2 Quality of Supply

When the supply voltages and currents become distorted (or non-sinusoidal) [ELEKTRON2], inefficient operation and malfunctioning of the electrical equipment can result. For example, induction machines exhibit reduced efficiencies and computer and relaying systems may be prone to errors when fed with distorted supply voltages [CHATHURY2].

In order to address the problem of AC supply voltage distortions, it is necessary to identify the origin of the distortion. In the simplified network shown in Fig. 2.4, voltage U_s is an idealised voltage source, and Z_s represents the total impedance of the generator, supply cabling and supply transformers [ELEKTRON1], [CHATHURY1].

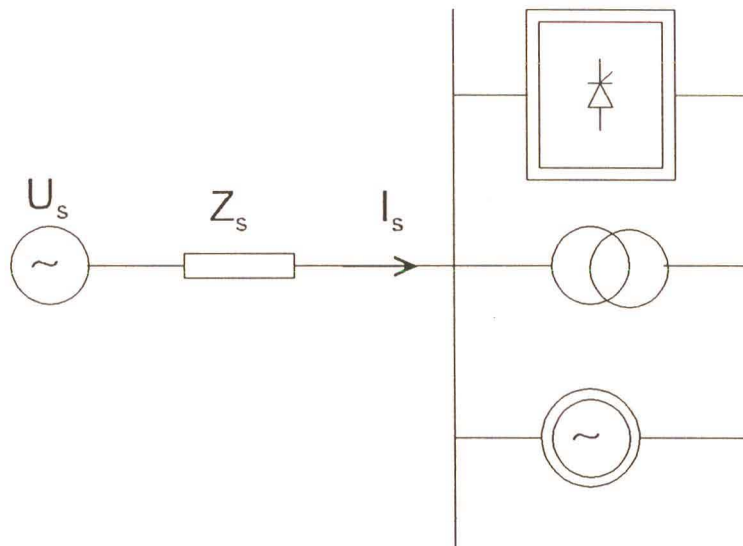


Fig. 2.4 Simplified power network [CHATHURY1]

The Common bus voltage is dependent on the supply current, I_s , and can be calculated as the difference between U_s and the voltage drop across Z_s . It thus makes sense that if the total current I_s drawn by different loads is non-sinusoidal, the voltage at the common bus will be distorted [CHATHURY2]. According to Chathury, these loads which draw non-sinusoidal currents need to be investigated and improved in order to draw sinusoidal currents [CHATHURY1-2].

The next section discusses power electronic equipment, and their contribution to the AC supply voltage distortion problem in power networks.

2.3 Conventional Rectifiers and their Shortcomings

2.3.1 Phase-controlled Thyristor Rectifier

Three-phase phase-controlled thyristor rectifiers are extensively used in a variety of applications in industry [ELEKTRON2], [CHATHURY2]. A basic three-phase phase-controlled thyristor rectifier connected to a DC motor load is shown in Fig. 2.5.

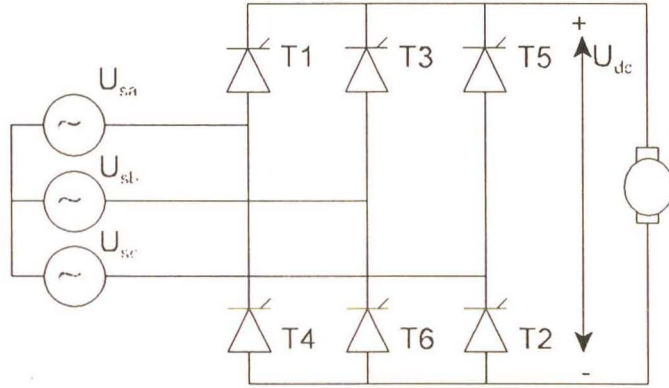


Fig. 2.5 Three-phase thyristor rectifier

The input phase voltages are given in Eq. (2.1), assuming that the armature inductance of the DC motor is large enough to make the armature ripple current negligible [CHATHURY1], [SAY1].

$$\begin{aligned}
 u_{sa} &= \sqrt{2}U_1 \sin(\omega t) \\
 u_{sb} &= \sqrt{2}U_1 \sin(\omega t - \frac{2\pi}{3}) \\
 u_{sc} &= \sqrt{2}U_1 \sin(\omega t + \frac{2\pi}{3})
 \end{aligned} \tag{2.1}$$

Say [SAY1] showed that the average output voltage is calculated as:

$$U_{dc} = \frac{3\sqrt{6}U_1}{\pi} \cos(\alpha) \tag{2.2}$$

where: U_1 : is the RMS phase voltage
 α : is the firing angle, where α is varied from 0 to $[180^\circ - \delta]$ (δ is the angle required for the commutation of the thyristors) to vary the output voltage.

However, as the firing angle is varied, a phase shift is generated between the current and the voltage (current will lag the voltage), resulting in a displacement factor $\cos(\alpha)$. This causes the overall power factor to reduce with an increase in the firing angle [NASAR1].

In Fig 2.6 Chathury shows that there is a phase displacement between the line current i_{sa} and the phase voltage u_{sa} [CHATHURY1] when a firing angle of $\alpha = 15^\circ$ is used. Furthermore, it shows the non-sinusoidal nature of the line current.

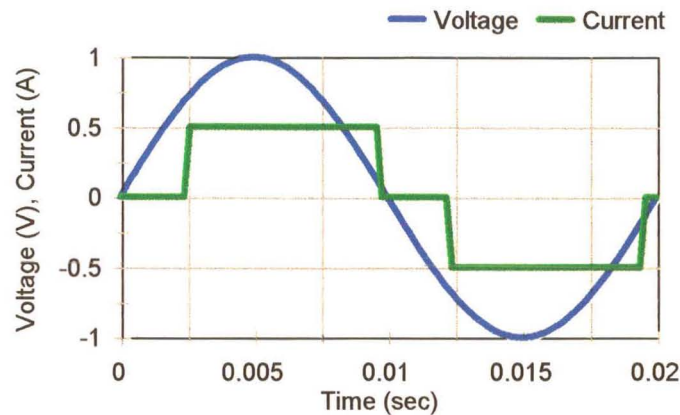


Fig. 2.6 Typical phase A operating voltage and current at $\alpha = 15^\circ$ [CHATHURY1] (pp 1.7)

Therefore it can be deduced that the phase controlled rectifier contributes to poor power factor and voltage distortion problems, since it draws non-sinusoidal line currents, which lag the voltage.

2.3.2 Three-phase diode rectifier

The three-phase diode rectifier (as shown in Fig. 2.7) provides a fixed DC output voltage, which cannot be varied, because the displacement factor remains at unity at all times due to the fixed firing angle. With a diode rectifier, the level of harmonic distortion in the line currents are similar to those of the thyristor rectifier with $\alpha = 0$.

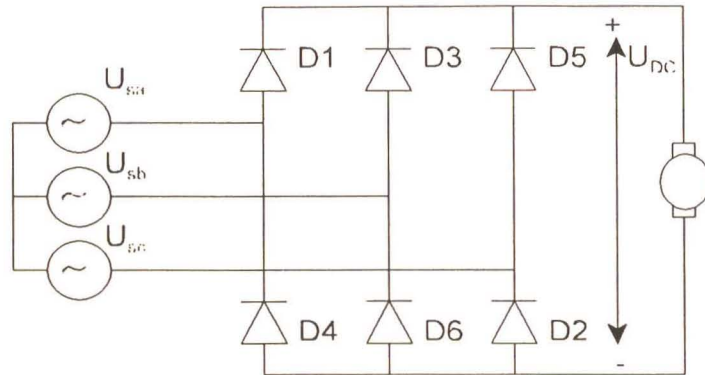


Fig. 2.7 Three-phase diode rectifier [CHATHURY1]

When these rectifiers are faced with QoS problems (any transient voltage or current faults), they generally have a dip in DC link voltage, which affects the load. In the rectifier/inverter topology, voltage dips and transients in the source generally reflect themselves in the Variable Speed Drive (VSD), because the VSD contains an unregulated DC link voltage. This in turn translates the problem to the motor controller [BROD1], as the DC link voltage appears as a gain factor in the current control loop. Ideally a system is needed that can keep the DC link voltage constant throughout a dip, thus the suggestion of using a boost rectifier.

The next section summarises research [WU2], [OOI1-2], [CHATHURY1-2] performed to address the dynamic performance and shortcomings of these rectifiers.

2.4 Improved Rectifier Topologies and Control Schemes

It is found that the field of rectifier control has been well researched and that a number of schemes presently exist, realising rectifiers with near sinusoidal line currents at unity power factor [CHATHURY1-2]. Chathury, in Chapter 1 pp 1.9 - 1.14, reviewed a number of advanced converter topologies and switching schemes [CHATHURY1]. This section briefly summarises these new topologies and their switching schemes.

The operating principles of the inverter in the variable frequency AC drive (discussed briefly in Chapter 1) during a reversal in the direction of power flow through the inverter, could be applied

to realise a PWM-switched rectifier coupling a *fixed* frequency AC supply to an electrical load [WU2], [OOI1-2], [DIXON2], [BOSE2]. Using the correct control, the above mentioned rectifier can be made bidirectional, with unity power factor, and sinusoidal line currents. Many unity power factor rectifiers have been realised, with sinusoidal line current bidirectional, which are similar in topology and PWM operation to the inverters used in variable frequency drives. These rectifier control algorithms are modified to meet the different performance requirements and characteristics of the new topologies [CHATHURY1-2]; for example in a rectifier the DC voltage is a controlled quantity whereas in the inverter it is fixed.

The above research has identified two categories of rectifiers, namely the voltage-source rectifier and the current-source rectifier, also known as boost and buck rectifiers respectively. Voltage-source rectifiers have a filter capacitor on the DC side and filter inductors on the AC side (which provides the voltage boost characteristic); while the current-source rectifiers have an inductor on the DC side and filter capacitors on the AC side [ZARGARI1], [GREEN3].

These rectifiers are then controlled according to an appropriate control algorithm, using three-phase PWM switching to control the rectifiers AC line currents [BISWAS1]. Chathury showed that the PWM schemes belong to one of two categories, the Direct Current Control (DCC) and the Indirect Current Control (ICC) PWM schemes. Typical examples of both these PWM schemes are briefly described in this section.

Green [GREEN1], Dixon [DIXON1] and Ooi [OOI1] discussed the control of voltage-source rectifiers using a Direct Current Control PWM scheme known as Hysteresis Current Control (HCC). In this control scheme, the switching devices are controlled so that the AC line currents are forced to track sinusoidal reference waveforms generated, in phase with the supply voltages, by the control circuitry. Furthermore, hysteresis bands get imposed around the current reference signals in order to limit the switching frequency of the switching devices [DIXON1]. The difficulties associated with this circuit, is that the random switching of the switching devices, makes it difficult to design protection circuitry for the system. Furthermore, the infinite gain of the hysteresis comparators cause the HCC to have an un-acceptable fault tolerance [DIXON1].

Dixon [DIXON3] presents a three-phase voltage-source rectifier switched using a ICC PWM scheme known as Sinusoidal Pulse-width Modulation (SPWM) [BOYS1]. In this topology the AC line currents are controlled by controlling the magnitude and phase of the voltages across the three line inductances, which are part of the rectifier topology. The magnitude and phase of the PWM control voltages are controlled, to vary the voltage drop across the line inductors, in order to control the rectifiers AC line current. Chathury stated that the advantage of the SPWM scheme over the HCC scheme is the fixed switching frequency, although this is obtained at the expense of poorer dynamic performance and a lower degree of robustness [CHATHURY2].

The third harmonic injection PWM scheme is a modification of the sinusoidal PWM scheme, where third harmonics are added to the modulating signals, to make them appear flat-topped [BOOST1]. This provides a greater degree of controllability of the rectifiers line current. These circuits may be either naturally or regularly sampled [BOWES1], [CHATHURY1].

A further widely used PWM method is Space Vector Modulation (SVM), which is used to control force-commutated voltage-source rectifiers. The SVM scheme uses concepts developed by Broeck [BROECK1], where the space vector is a mathematical quantity describing the phase voltage or current waveform. The voltage space vector used to describe the three-phase voltage-source converter input PWM voltages is discrete in nature because of the switch-mode operation of the rectifier. The space vector modulation technique involves creating a reference voltage vector for the rectifiers input voltages and generating it by switching between its immediately adjacent (realisable voltage vectors) and the two zero voltage vectors so that optimal harmonic performance is achieved [CHATHURY1].

The final control method is pre-programmed PWM which pre-calculates the switching instants using numerical techniques to solve sets of equations defined according to the rectifier's performance criteria, in order to optimise either efficiency, total harmonic current distortion or selective harmonic elimination. These calculated switching instants are stored in lookup tables for different values of the fundamental converter input voltage [BOWES1], and used to control the rectifier.

Rectifier control schemes that use pre-programmed PWM schemes often require low switching frequencies and are characterised by good steady-state, but poor dynamic performance.

In summary, Chathury showed that two types of rectifier schemes may be employed, namely the voltage-source and current-source converter [CHATHURY1]. In these schemes a PWM switching technique is used to control the magnitude and frequency spectrum of the AC line currents according to a control algorithm, using a mathematical model of the power conversion system. However, the most commonly used technique is indirect current control. In this technique, the AC line currents are controlled by controlling the voltage drops across inductors placed on the AC side of the converter. The actual control is achieved by using PWM voltages of variable magnitude and phase produced at the input of the converter [CHATHURY2], [DIXON3].

Chathury [CHATHURY1-2] chose to use the voltage-source rectifier (referred to as a boost rectifier from hereon) as the most appropriate topology since it allowed for bidirectional power flow, and provided a regulated DC output voltage with sinusoidal line currents at unity power factor. Furthermore, Chathury used a regularly-sampled third-harmonic injected PWM technique to control the boost rectifier [CHATHURY1], [DIXON3], [BOOST1].

The structure of the controller designed by Chathury is similar to that presented by Kolar [KOLAR1].

As mentioned in the introduction, this thesis extends the research performed by Chathury [CHATHURY1-3]. Thus the structure of the boost rectifier, and the linearised current controller developed by Chathury [CHATHURY1, 3] is discussed in the next section. This structure is then used in subsequent Chapters for the modification of the inputs, outputs and sign conventions of the existing COT ANN current controller [BURTON1-2] for use with the boost rectifier.

2.5 Chathury's Conventional Boost Rectifier Controller

This section presents the mathematical model of the boost rectifier developed by Chathury, to gain an understanding of its structure, sign conventions, and the controller design process [CHATHURY1]. This was done to allow Burton's COT ANN to be modified for the boost rectifier [BURTON1]. Fig 2.8 shows the basic structure of a boost rectifier.

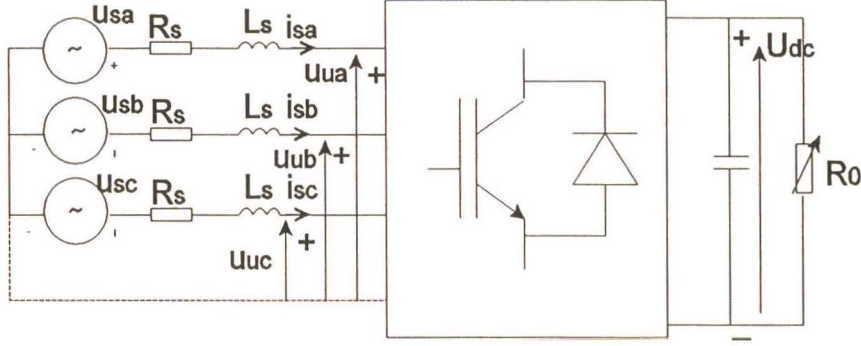


Fig. 2.8 Boost rectifier configuration

A mathematical model of a boost rectifier is derived by setting equations (APPENDIX A) to represent the AC side dynamics of the boost rectifier, and equating the instantaneous AC power into the boost rectifier to the instantaneous DC output power in the d, q reference frame (synchronous reference frame) [CHATHURY1]. These dynamic representative expressions are provided in Eq. (2.3).

$$\begin{aligned} u_{ud} &= u_{sd} - L_s \frac{di_{sd}}{dt} - R_s i_{sd} + \omega \cdot L_s i_{sq} \\ u_{uq} &= u_{sq} - L_s \frac{di_{sq}}{dt} - R_s i_{sq} - \omega \cdot L_s i_{sd} \end{aligned} \quad (2.3)$$

where: R_s is the incoming line resistance (Fig. 2.8),

L_s is the incoming line inductance (Fig. 2.8),

ω is the line frequency in radians per second,

u_{ud} , u_{uq} are the instantaneous d and q axis voltages at the converter input respectively, and

u_{sd} , u_{sq} are the instantaneous d and q axis supply voltages respectively.

The expression for the power balance is shown in Eq. (2.4).

$$\begin{aligned} \text{AC input Power} &= \text{DC output Power} \\ \frac{3}{2}(u_{sd} \cdot i_{sd} - \frac{1}{2} \cdot L_s \cdot \frac{d(i_{sd}^2 + i_{sq}^2)}{dt} - (i_{sd}^2 + i_{sq}^2) \cdot R_s + u_{sq} \cdot i_{sq}) &= \frac{1}{2} \cdot C \cdot \frac{du_{dc}^2}{dt} + \frac{u_{dc}}{R_o} \end{aligned} \quad (2.4)$$

Chathury then linearised the dynamic power balance equation about the operating point (APPENDIX A), to enable the design of the DC voltage regulator [CHATHURY1]. The new linearised power balance equation is given as:

$$\begin{aligned} &\frac{3}{2} \left(U_{sdo} \Delta i_{sd} - (2 \Delta i_{sd} I_{sdo} + \Delta i_{sd}^2 + \Delta i_{sq}^2) R_s - \frac{1}{2} L_s \frac{d(2 \Delta i_{sd} I_{sdo} + \Delta i_{sd}^2 + \Delta i_{sq}^2)}{dt} \right) \\ &= \frac{1}{2} C \frac{d(2 \Delta u_{dc} U_{dco} + \Delta u_{dc}^2)}{dt} + \frac{2 \Delta u_{dc} U_{dco} + \Delta u_{dc}^2}{R_o} \end{aligned} \quad (2.5)$$

The mathematical model, shown by Eq. (2.5), is then used to develop the necessary control structure for the boost rectifier, so that a block diagram may be realised for the boost rectifier controller as shown in Fig. 2.9 [CHATHURY1, 3].

Control of the AC line currents is achieved according to the ICC method by controlling U_{ua} , U_{ub} , and U_{uc} in Fig. 2.9. In d-q coordinates, this implies that i_{sd} and i_{sq} are regulated by controlling u_{ud} and u_{uq} . Closed loop current controllers set up reference values u_{ud}^ and u_{uq}^* , for the d and q axis components of the fundamental component of the desired rectifier voltage space vector \underline{U}_u^* . u_{ud}^* and u_{uq}^* are then transformed into the α - β reference frame (this is required for the PWM ASIC [HANNING1]) to yield $u_{u\beta}^*$ and $u_{u\alpha}^*$, the $\alpha\beta$ components of \underline{U}_u^* . These signals are then normalised and applied to the pulse-width modulator. The pulse-width modulator then switches the rectifier according to the third-harmonic injected PWM technique discussed in the previous section, to ensure that $u_{ud} = u_{ud}^*$ and $u_{uq} = u_{uq}^*$ so that i_{sd} and i_{sq} are regulated to their reference values [CHATHURY1].*

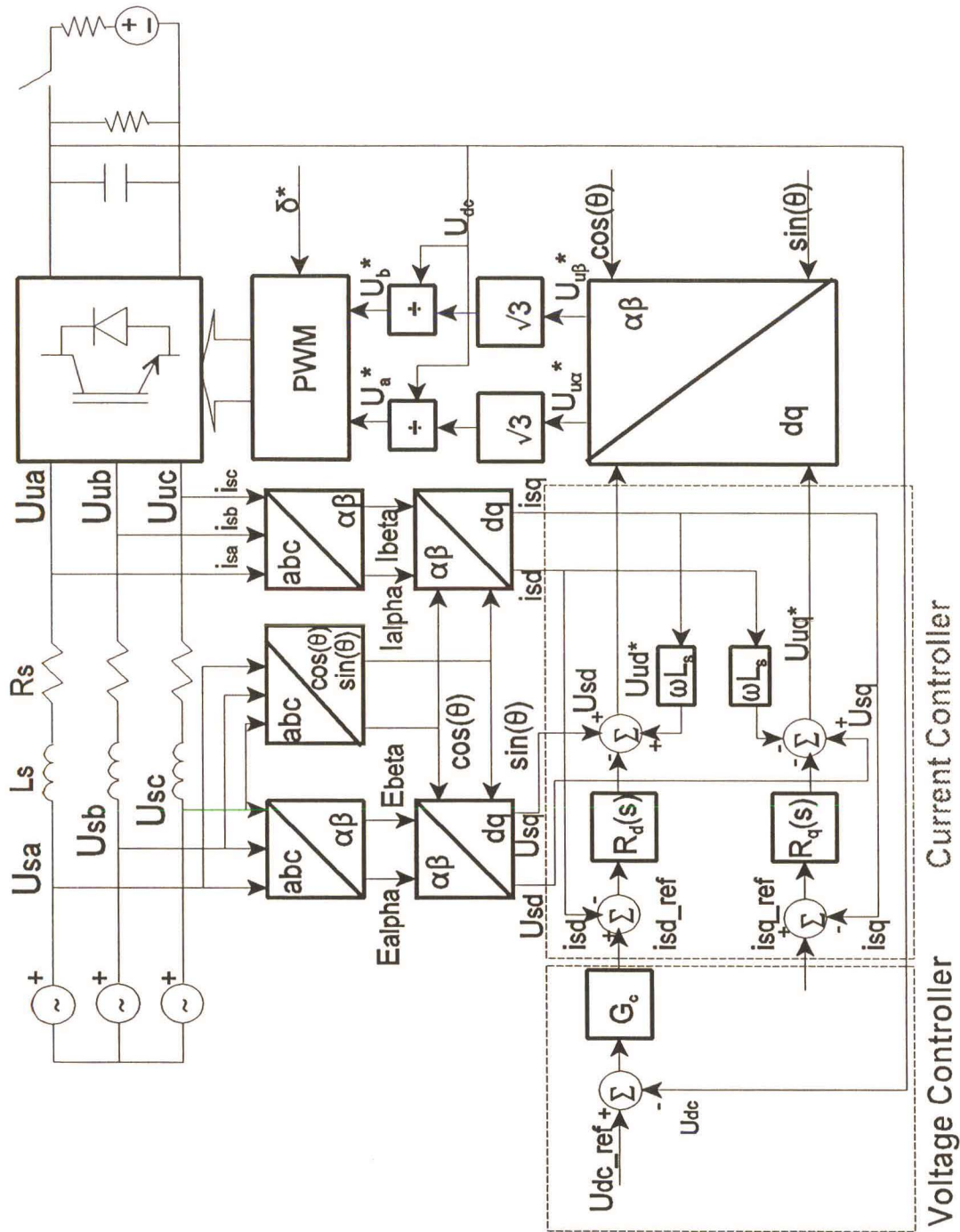


Fig. 2.9 Chathury's boost rectifier current and voltage controller [CHATHURY1]

In Fig. 2.9

G_c represents the transfer function of the voltage controller, and $R_d(s)$, and $R_q(s)$ are the transfer functions of the d and q current controllers respectively.

Chathury [CHATHURY1,3] showed that this structure could control the boost rectifier, in the motoring mode of operation, but had a shortcoming, in that the controller was designed about a specific operating point. This implies that the controller may only be able to operate correctly in a limited range of system variations (i.e. impedance changes and voltage dips).

It was thus proposed to use the above mentioned control structure to modify the structure, inputs, outputs and sign conventions of the COT ANN current controller developed by Burton [BURTON1-2]. The modified COT ANN would then be used to replace the current controller in Fig. 2.9, in order to obtain a controller which is stable [MORSE1] under all operating conditions.

The next section provides an overview of the structure of the COT ANN current controller developed during earlier research by Burton [BURTON1-2].

2.6 COT ANN Based IM Current Controller

Previous research investigated the use of a back-propagated COT ANN current controller on the inverter side of a SCIM Variable Speed drive [BURTON1-2], [WISHART1-2], [BUHL1] as shown in Fig. 2.2, to obtain IM current control. Burton [BURTON1] investigated the practical implementation of an identification and control method that exploits the inherent self learning capability of ANNs to infer an intelligent, adaptive, nonlinear control strategy. This strategy is self commissioning and adaptive and thus does not rely on any specific prior or continually updated knowledge of machine parameters [BURTON2] [NARENDRA2].

The next section briefly describes the principles used to tune and train a COT ANN.

2.6.1 The Basic Principles of System Identification using ANNs

Burton stated that a three layer ANN, with sufficient hidden neurons, can learn to approximate any continuous vector mapping function to an arbitrary degree of accuracy over a given range [BURTON1-2]. This is the basic property of ANNs that is exploited for system identification.

Vector Function Approximation

The objective is to supply the ANN with input values $\underline{x}(k)$ and the resulting output values $\underline{y}(k)$ from the function $\underline{f}(\cdot)$ which has to be identified by the ANN. The ANN uses this information to modify its weight states, using the backpropagation algorithm, until the feed forward function $\underline{\hat{f}}(\cdot)$ approximates the function to the desired level. The function $\underline{f}(\cdot)$ may be identified by either offline batch, offline recursive or continual online training techniques.

Offline Batch Training

A representative set of \underline{x} and \underline{y} values for function $\underline{f}(\cdot)$ are constructed over a predetermined range (set of \underline{x} values) for training purposes. The data is arranged into pairs and presented to the ANN, until the desired accuracy of identification is achieved. The training is said to be convergent once the approximation error drops below a predefined minimum or ceases to decrease after a number of training periods. Once convergence is achieved, the ANN is used exclusively in the feedforward mode to replace the function $\underline{f}(\cdot)$ or to perform some other purpose.

Offline Recursive Training

In this technique, the training data is obtained by taking samples of the desired functions input and output values. Special input ranges are chosen, so that the ANN has adequate data on the function to learn the function over the whole range of interest. This training technique is usually used where it is difficult to generate a batch of data to adequately represent the function in the required range. Recursive training is normally used in situations where it is difficult to generate a batch of data that adequately represents the function or system to be identified.

Continual Online training

Continual online training is performed using a series of data samples from the function under normal operating conditions. This data does not provide a clear representation of the system over the entire range of operation, but it does provide sufficient information for adaptive identification of the time varying system by means of either an adaptive global or local minimum [BURTON1].

The next section summarises the Narmax model of the induction motor developed by Burton [BURTON1-2] and Wishart [WISHART1-2]. It also provides a summary of the derivation of the COT ANN based current control law from Burton.

2.6.2 Burton's adaptive current control law

Wishart [WISHART1-2] derived the stator current control law which can be applied from a knowledge of the outputs of the discrete time NARMAX (Nonlinear Auto-regressive Moving Average with eXogenous inputs) model (also derived by [WISHART1]) of the electrodynamics of an induction motor. Wishart showed that the time discrete NARMAX model of an induction motors electrodynamics (in the stationary axis, or $\alpha\beta$ reference frame), is expressed by Eq. (2.6) or in simplified form by Eq. (2.7) [WISHART1].

$$\underline{i}(k) = \underline{f}_{cl}^{(k-1)}(\underline{i}(k-1), \underline{i}(k-2), \omega(k-1), \omega(k-2), \underline{v}(k-2)) + c_v(k-1)\underline{v}(k-1) \quad (2.6)$$

[WISHART1]

where: $\underline{i}(k)$ is the per unit value of the $\alpha\beta$ stator current vector $[i_\omega i_\beta]^T$ at time (k)

$\underline{i}(k-1)$ is the per unit value of the $\alpha\beta$ stator current vector $[i_\omega i_\beta]^T$ at time (k-1)

$\underline{i}(k-2)$ is the per unit value of the $\alpha\beta$ stator current vector $[i_\omega i_\beta]^T$ at time (k-2)

$\omega(k-1)$ is the per unit value of the shaft speed at time (k-1)

$\omega(k-2)$ is the per unit value of the shaft speed at time (k-2)

$\underline{v}(k-1)$ is the per unit value of the $\alpha\beta$ stator voltage applied at time (k-1)

$\underline{v}(k-2)$ is the per unit value of the $\alpha\beta$ stator voltage applied at time (k-2)

c_v is the per unit value of the voltage constant defined by the motor parameters.

$\underline{\$}(k) = (\underline{i}(k-1), \underline{i}(k-2), \omega(k-1), \omega(k-2), \underline{v}(k-2))$

$$\underline{i}(k) = \underline{f}_{el}^{(k-1)}(\underline{\$(k-1)}) + c_v(k-1)\underline{v}(k-1) \quad (2.7)$$

[WISHART1]

From this equation, Burton and Wishart made an informed decision, that continual online training must be used to adaptively identify the motor, since the form of the chosen identification model is essentially time varying [BURTON1], [WISHART1]. This section summarises the derivation of the COT ANN based current control law, from the ANN identification of Eq. (2.7) [BURTON1].

Since the current control input variable at time (k) is the stator voltage $\underline{v}(k)$, the first step in deriving the control law from Eq. 2.7 is to rewrite it in terms of $\underline{v}(k)$ [BURTON1]. Burton [BURTON1] performed this derivation and produced the current control law shown in Eq. (2.8).

$$\underline{v}^*(k) = \frac{\underline{i}^*(k+1) - \underline{f}_{el}^{(k)}(\underline{\$(k))}}{c_v(k)} \quad (2.8)$$

where the * modifier is used to identify the desired value of voltage or current vector.

From Burton, the ANN identification equation is given as [BURTON1]:

$$\hat{\underline{i}}(k) \approx \hat{\underline{f}}_{el}^{(k-1)}(\hat{\underline{\$(k-1)}) + \hat{c}_v(k-1)\hat{\underline{v}}(k-1) \quad (2.9)$$

and the ANN approximation equation is given as [BURTON1]:

$$\hat{\underline{i}}(k+1) = \hat{\underline{f}}_a^{(k-1)}(\underline{\$}(k)) + \hat{c}_v(k-1)\underline{v}(k) \quad (2.10)$$

This control law yields the control vector $\underline{v}^*(k)$, which is required to make the desired value of one step ahead current $\underline{i}^*(k+1)$ to flow in the motor. This can be directly implemented, using the information made available by the ANN identification and the ANN approximation equations, see ([BURTON1] pp 3.11 (Eq. (3.8)) and pp 3.27 (Eq. (3.12)) respectively (shown in Eq. (2.9) and (2.10) above for convenience), as:

$$\underline{v}^*(k) = \frac{\underline{i}^*(k+1) - \hat{\underline{f}}_a^{(k-1)}(\underline{\$}(k))}{\hat{c}_v(k-1)} \quad (2.11)$$

Burton [BURTON1] states that with the implementation of control Eq. (2.11), the ANN estimated current vector $\hat{\underline{i}}(k+1)$ of Eq. (2.10) is approximately equal to the desired current vector $\underline{i}^*(k+1)$. Thus the ANN identification error can be approximated as:

$$\underline{e}_v(k) = \underline{i}(k) - \underline{i}^*(k) \quad (2.12)$$

Burton used Eq. (2.12) to evaluate the ANN training error, in order to incorporate any training error in the approximation of Eq. (2.11) into the COT ANN [BURTON1], [WISHART1].

Burton stated that it can be assumed that the use of Eq. (2.12) will provide the ANN with the sufficient information to adaptively compensate for the approximation of Eq. (2.10) (on which control Eq. (2.11) is based) to track the desired currents more closely [BURTON1]. By using Eq. (2.12), the ANN identification error used for the continual online training is equal to the negative of the control error. This means that the ANN current identifier will reduce the control error magnitude and force the actual motor current $\underline{i}(k)$ to track the desired current reference $\underline{i}^*(k)$. It should be noted that convergence is defined as the state when the control variable tracks the reference variable to within a certain limit (local minima is achieved by tracking function). Whereas divergence is defined as the state when the control variable cannot track the reference variable at all.

Burton's main real-time issues and practical effects which contribute to the control bandwidth limitations of the prototype practical implementation of the ANN current loop were [BURTON1-2]:

- 1) a delay in the output of $\underline{v}^*(k)$ to the PWM ASIC,
- 2) the sampling rate of the practical system and
- 3) the limitations of the commercial IGBT inverter.

A detailed study is performed in the next chapter, duplicating Burton's Matlab simulations of the COT ANN current controlled VSI fed SCIM Drive [BURTON1] in CASE3 [KLEINHANS1], to determine and correct its shortcomings. The COT ANN current controller developed by Burton is then modified, refined, and substituted for the current controller in the boost rectifier model developed by Chathury, whereafter the COT ANN current controlled boost rectifier system was simulated to determine the feasibility of using the COT ANN current controller with a boost rectifier.

2.7 Summary

This chapter provided an overview of previous work, and presented the necessary theory and models for the classical controller developed by Chathury [CHATHURY1] and the COT ANN current controller developed by Burton [BURTON1]. Emphasis was placed on the derivations of the controller structure. A brief overview of QoS issues, conventional rectifiers, and improved rectifiers and control schemes was also provided. Thereafter the COT ANN based control law developed by Burton [BURTON1], [BURTON2] was presented in a summarised format. Finally, the possible research extension, using the COT ANN current controller as a direct replacement for the conventional current controller in the Boost rectifier configuration, was discussed.

The next chapter repeats the simulation work performed by Burton [BURTON1] to determine and overcome the problems that he experienced in the practical implementation of the COT ANN current controller.

CHAPTER 3

SIMULATION OF A CONTINUOUSLY ONLINE TRAINED ARTIFICIAL NEURAL NETWORK CONTROLLED VOLTAGE-SOURCE INVERTER FED SCIM DRIVE

3.1 Introduction

This thesis investigates the use of the COT ANN current controller developed by Burton [BURTON1-2], to overcome the shortcomings of the current controller used in Chathury's [CHATHURY1-3] research. Chapter 1 introduced the shortcomings experienced in the current controller developed by Chathury, and the COT ANN current controller developed by Burton, describing what processes should be followed to minimise these effects. Chapter 2 provided an overview of the work performed by Chathury and Burton, describing how the research will be extended to realise a COT ANN current controller for the control of a boost rectifier.

This chapter provides an overview of the CASED (Computer-aided Analysis and Simulation of Electrical Drives) simulation package [CASED1], which will be used to reproduce Burton's simulations. Burton's work is reproduced in this chapter, to gain an understanding of the nature and form of the COT ANN as well as its operation under the nonlinear effects of a VSI fed SCIM drive. Burton's simulations, which were implemented in Matlab, omitted the PWM and inverter from the VSI fed SCIM drive model, resulting in inaccuracies between the simulated and practical results (the practical system became unstable at stator frequencies above 2 to 3 Hz), due to the non-linearities introduced by the PWM controller and inverter. The COT ANN current controller, developed by Burton for an SCIM drive, is evaluated in this chapter and re-tuned to overcome these shortcomings (as noted by Burton).

The next section describes the simulation approach used by CASED, along with its capabilities.

3.2 CASED Simulation of Power Conversion Systems

Power electronic conversion systems are typically modular in their basic structure, and consist of interfacing subsystems such as an electrical supply, switch-mode converters, electro-mechanical (or electrical) loads, and digital or analogue signal sensing and control hardware, as shown in Fig. 3.1.

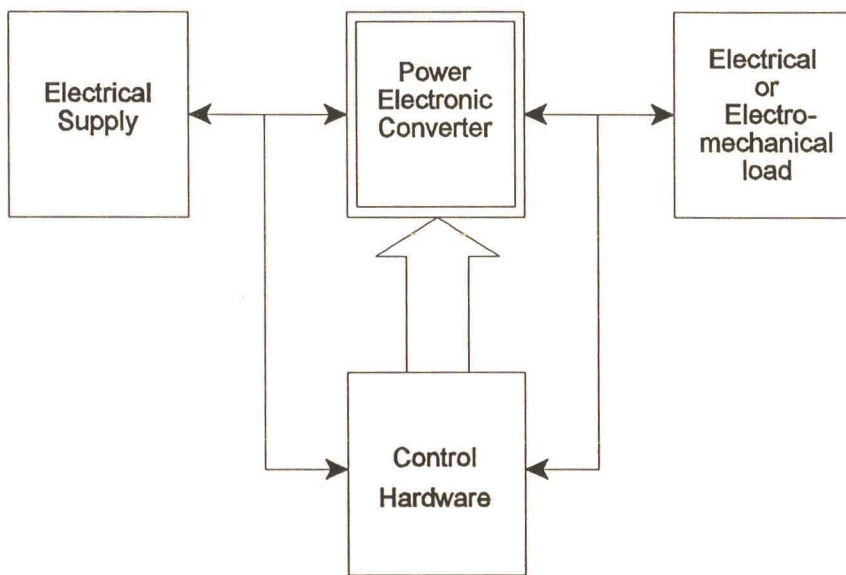


Fig. 3.1 Structure of a typical industrial power conversion system

The internal structure of the individual subsystems shown in Fig. 3.1, differ significantly from the different existing power conversion systems. For example, a number of converter topologies are possible, and the load subsystems usually differ greatly between applications. Possible load types range from simple electrical loads such as resistor banks, to complex electro-mechanical loads such as induction motors coupled to conveyer belt systems. This, together with the complex interactions between the different subsystems, means that the simulation of power conversion systems poses a number of unique challenges. Furthermore, the simulation packages used require a great deal of flexibility in defining the different subsystems to accommodate the many different converter topologies, load types and control schemes [CHATHURY1], [CASED1].

It was found that the most versatile and flexible approach in simulating power conversion systems was to use a modular simulation environment, in which each subsystem is simulated as an independent module by virtue of its functionality. A complete simulation model of a power conversion system is then built up as an interconnected system of modules, some of which may be common between different conversion systems. In such a modular simulation environment, the effects of different load types on the performance of a particular power conversion system may be easily evaluated by changing only the load module while maintaining the rest of the system. CASED [CASED1], [KLEINHANS1-4] uses the modular approach of simulation, and provides a completely general and flexible simulation environment specially designed for the simulation needs of the modern power conversion system.

The subsystems comprising a power conversion system may be classified into one of the four basic simulation modules available in CASED [CASED1]. These are the generalised converter, analogue model, analogue controller and digital controller modules as shown in Fig. 3.2 [CHATHURY1]. Each of these modules are written in C.

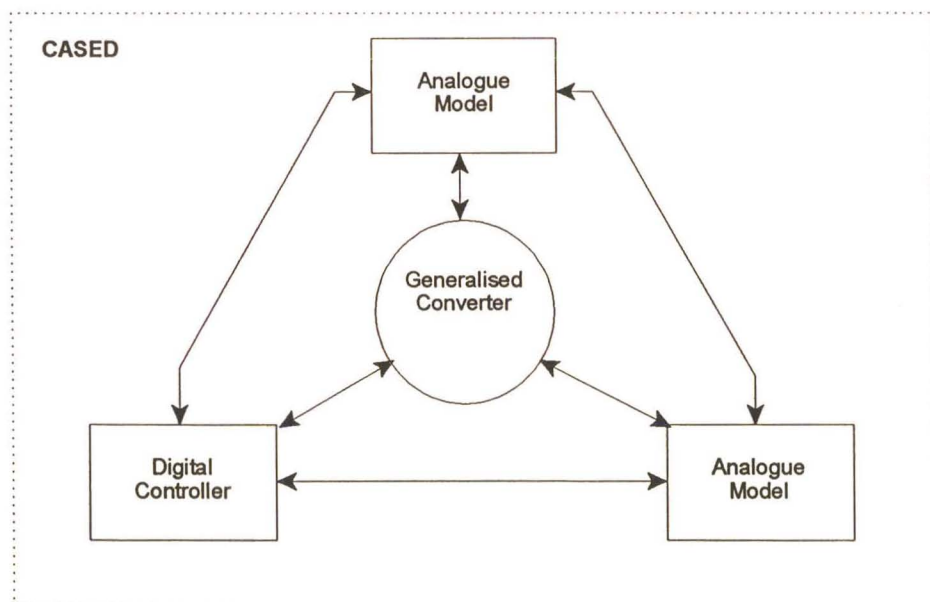


Fig. 3.2 Structure of CASED

Each module is briefly described in the following paragraphs.

Analogue Model and Analogue Controller Modules

Power conversion systems usually consist of a mixture of analogue (or continuous time) subsystems and digital (or discrete time) subsystems. Analogue subsystems (e.g. an electrical load) are modeled in CASED in their state-space forms as analogue model modules. Input and Output variables specific to an analogue model are typically defined, and the corresponding state-space equations written in a C format. The module is then interfaced to the rest of the system via its I/O variables.

Digital Controller Modules

Digital Controller modules are used to simulate discrete processes occurring at regular time intervals within the power conversion system. Such discrete processes may include the execution of the converter control algorithm, and the generation of PWM switching signals. Digital controller modules have full access to all system I/O variables, as do analogue model modules, and they may update certain system I/O variables. Digital controller modules may also be used to simulate digital control processes such as PI control, since they make provision for the definition of digital states. Digital control tasks and processes are coded as user-programmed C format subroutines, which are compiled and linked into the main body of CASED.

Interfacing between Modules in CASED

Modules are interfaced to each other via their defined I/O variables. These variables are usually voltages and currents when interfacing an analogue model describing an electrical model (such as an electrical supply) to another (such as an electrical load); and torque, speed and angular position when interfacing an electro-mechanical model (such as an induction machine) to a mechanical model (such as a conveyor belt load). For other general user-defined models, the interface variables are user-defined. The interconnections between the different modules are user-specified when different modules comprising a single system are linked together. The system linking program (called create) repeatedly prompts the user to specify the input to which a specific output connects until all the module outputs have been connected to their respective inputs in the other modules. Module outputs which do not connect to other module inputs are referred to as dummy outputs [CHATHURY1].

Generalised Converter Module

CASED provides a graphical environment (Netgen) for specifying the topology of a power electronic converter. The various switching devices available may be interconnected in any manner to realise a desired converter topology. It is sometimes desirable to define the electrical supply and load subsystems as independent modules to be interfaced to the generalised converter.

The next section reproduces the Matlab simulations performed by Burton, in CASED.

3.2 Investigation of a basic COT ANN Controlled VSI fed SCIM Variable Speed Drive

A first step toward understanding Burton's work was to duplicate his Matlab simulations in CASED [KLEINHANS1-4], [CASED1], to become familiar with the concept of COT ANN current controllers and to identify any shortcomings. The second step is to simulate the COT ANN system that Burton implemented on the transputer [BURTON1], [BURTON2]; this is discussed in greater detail in section 3.3.

Burton's Matlab simulations omitted the inverter and its PWM controller from the model. Therefore, the system configuration used in the CASED simulation, initially omits the inverter and its PWM controller from the power conversion system (this configuration is identical to that used by Burton in Matlab), as shown in Fig. 3.3. In these simulations, the COT ANN controller feeds the alpha and beta controlling voltages directly to the induction motor to achieve the desired system response. These initial simulations are performed using the same COT ANN learning rate and momentum values used by Burton in his Matlab simulation model. Furthermore no pre-training is used, and the initial weight states are randomly generated to duplicate the conditions simulated by Burton.

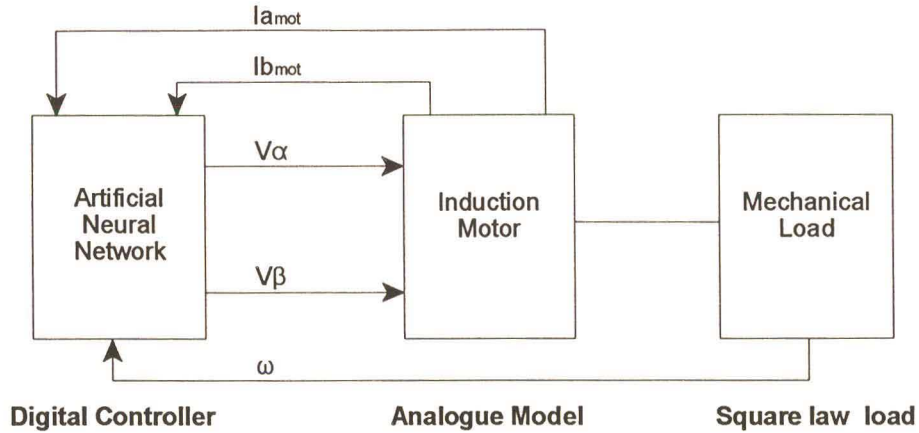


Fig. 3.3 Basic COT ANN controlled power conversion system

The model shown in Fig. 3.3 consists of three user-defined modules linked together in the CASED work space to form a closed loop system, namely:

- 1) a COT ANN stator current controller module developed by Burton [BURTON1] (C code provided in APPENDIX B.1). Fig. 3.4 shows the flow diagram of the COT ANN current controller module source code. The following main steps are sequentially executed in the program:
 - Step 1: All the simulation parameters and variables (e.g. sampling period), identification ANN parameters and variables (e.g. ANN size, gains, initial weights, inputs), induction motor parameters and variables (e.g. impedances, initial conditions etc.) are initialised.
 - Step 2: The main loop starts when an enable or interrupt signal is received.
 - Step 3: Steps 4 to 9 are omitted if the loop counter value is less than the corresponding start time T_{start} .
 - Step 4: v_α and v_β are written to the predefined port memory address for processing by the induction motor model.
 - Step 5: The digital values of the A and C phase motor line currents and shaft encoder positions are then read from the predefined dual port memory addresses and converted to floating point numbers representing the sampled $\alpha\beta$ motor currents $i(k)$ (shown as $I_{a_{mot}}$ and $I_{b_{mot}}$ in Fig 3.3) and shaft speed $\omega(k)$.

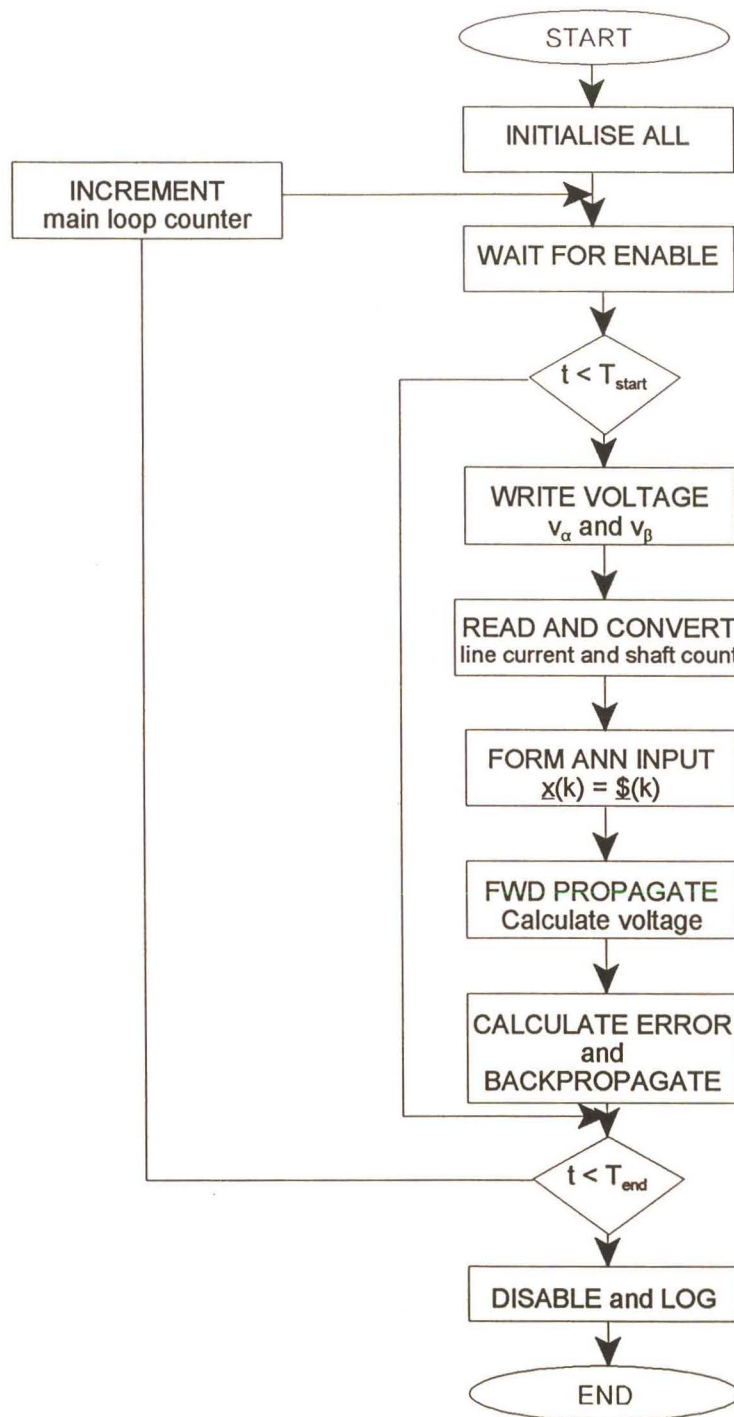


Fig. 3.4 Flow diagram of COT ANN CASED module

- Step 6: The ANN input $\underline{x}(k+1)=\underline{\$}(k)=[i(k), i(k-1), \omega(k), \omega(k-1), v(k-1)]$ is formed.
- Step 7: $\underline{x}(k+1)$ is forward propagated through the ANN to obtain $\underline{y}(k)=\hat{f}_{el}^{(k-1)}(.)$.
- Step 8: $\underline{i}^*(k+1)$ is calculated, either with or without the use of the reference model [BURTON1], and used with $\hat{f}_{el}^{(k-1)}(.)$ and constant c_v to calculate $\underline{v}^*(k)$ using Eq. (2.11) from Chapter 2.
- Step 9: The ANN output error $e_y(k) = i(k) - i^*(k)$ is calculated and backpropagated through the ANN to calculate $W(k+1)$ and $V(k+1)$, which represents $\hat{f}_{el}^{(k-1)}(.)$, for the next sampling period.
- Step 10: Step 11 is omitted if the specified loop count corresponding with T_{end} has been reached.
- Step 11: The main loop counter is updated and the program returns to step 2.

2) a induction motor module (code provided in APPENDIX B.2). Fig 3.5 shows the flow diagram of the induction motor modules source code. The following main steps are sequentially executed in the program:

- Step 1: All the induction motor parameters and variables are initialised and set.
- Step 2: The main loop starts when an enable signal or interrupt is received from the COT ANN controller.
- Step 3: Steps 4 to 6 are omitted if the loop counter value is less than the corresponding start time T_{start} .
- Step 4: The induction motor input voltages v_α and v_β are read into the motor module.
- Step 5: The induction motor module uses the describing function of an induction motor [NASAR1], to calculate the $\alpha\beta$ motor stator currents I_{amot} and I_{bmot} , and the machine torque T_{em} .
- Step 6: The module then outputs the $\alpha\beta$ motor stator currents I_{amot} and I_{bmot} , and the machine torque T_{em} .
- Step 7: Step 8 is omitted if the specified loop count corresponding with T_{end} has been reached.
- Step 8: The main loop counter is updated and the program returns to step 2.

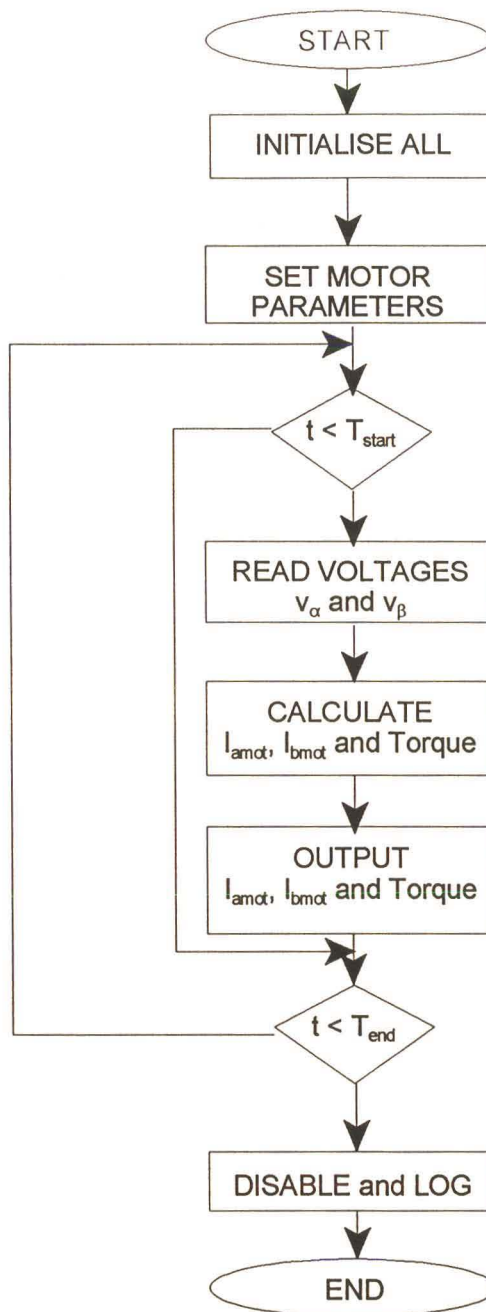


Fig. 3.5 Flow diagram of CASED induction motor module

- 3) a mechanical load (a simple inertial load) module (CASED library model).

The COT ANN used for these simulations is identical to the one used in Burton's simulations. The COT ANN consists of eight inputs, 12 hidden nodes and 2 output nodes. Fig 3.6 (a), (b) and (c) are simulation results obtained for the simplified COT ANN current controlled SCIM drive (shown in Fig 3.3), with a COT ANN sampling frequency of 500 Hz (this is the sampling frequency used in Burton's practical system), for motor supply frequencies of 1 Hz, 50 Hz and 100 Hz respectively. These CASED simulation results show that the existing COT ANN current controller forces the motor supply current (shown as I_{a_supply} in all diagrams hereon) to converge to the reference current (shown as I_{a_ref} in all diagrams hereon) at low frequencies (Fig. 3.6 (a) and (b)). But at higher motor supply current frequencies, the motor current diverges from the reference current (Fig. 3.6 (c)), which verifies Burton's simulations at a COT ANN sampling frequency of 500 Hz (which is the operating frequency of Burton's practical COT ANN controller) as shown on page 3.67 in [BURTON1]. Burton achieved current convergence at motor supply frequencies of 200 Hz and above when simulating the COT ANN at sampling frequencies above 500 Hz (see [BURTON1] pp 3.67). Fig. 3.6 (a) shows a comparison between Burton's Matlab simulation results, and the CASED results obtained in this section, for a stator frequency of 1 Hz. Here it is seen that the results obtained in Matlab have a lower initial overshoot, and achieve current convergence in a shorter time span. This is due to the fact that the CASED simulation package is designed specifically to simulate inverters and converters, whereas Matlab is focused on generic system simulations.

Burton stated that one of the reasons for the discrepancies between the practical and simulation results was due to the omission of the inverter and PWM controller from his simulation model. Thus to accurately model Burtons practical system, the inverter and PWM controller must be included in the simulation model. Before performing these simulations, the COT ANN current controller sampling and learning rates are modified, to try to achieve convergence above 50 Hz. Fig 3.7 provides an example of the simulation results obtained for the COT ANN current controlled system with modified parameters (learning rate, momentum gain etc.), with a motor frequency of 250 Hz and a COT ANN sampling frequency of 10 kHz. A Comparison between Fig. 3.6 (b) and Fig 3.7 (a) shows that the COT ANN controller requires 40 milliseconds of training, whereas the Burtons COT ANN current controller required 180 milliseconds training, before it tracks the 50 Hz stator current waveform. This displays the effect of COT ANN

sampling frequency on the performance of the controller. This subject will be investigated in further detail in section 3.3.

At this point the COT ANN current controller's internal parameters and variables have been changed to achieve current convergence with the simplified SCIM drive system (Fig. 3.3). The next step was to include the inverter and PWM controller in the SCIM drive simulation model, thereby simulating Burton's practical system more accurately.

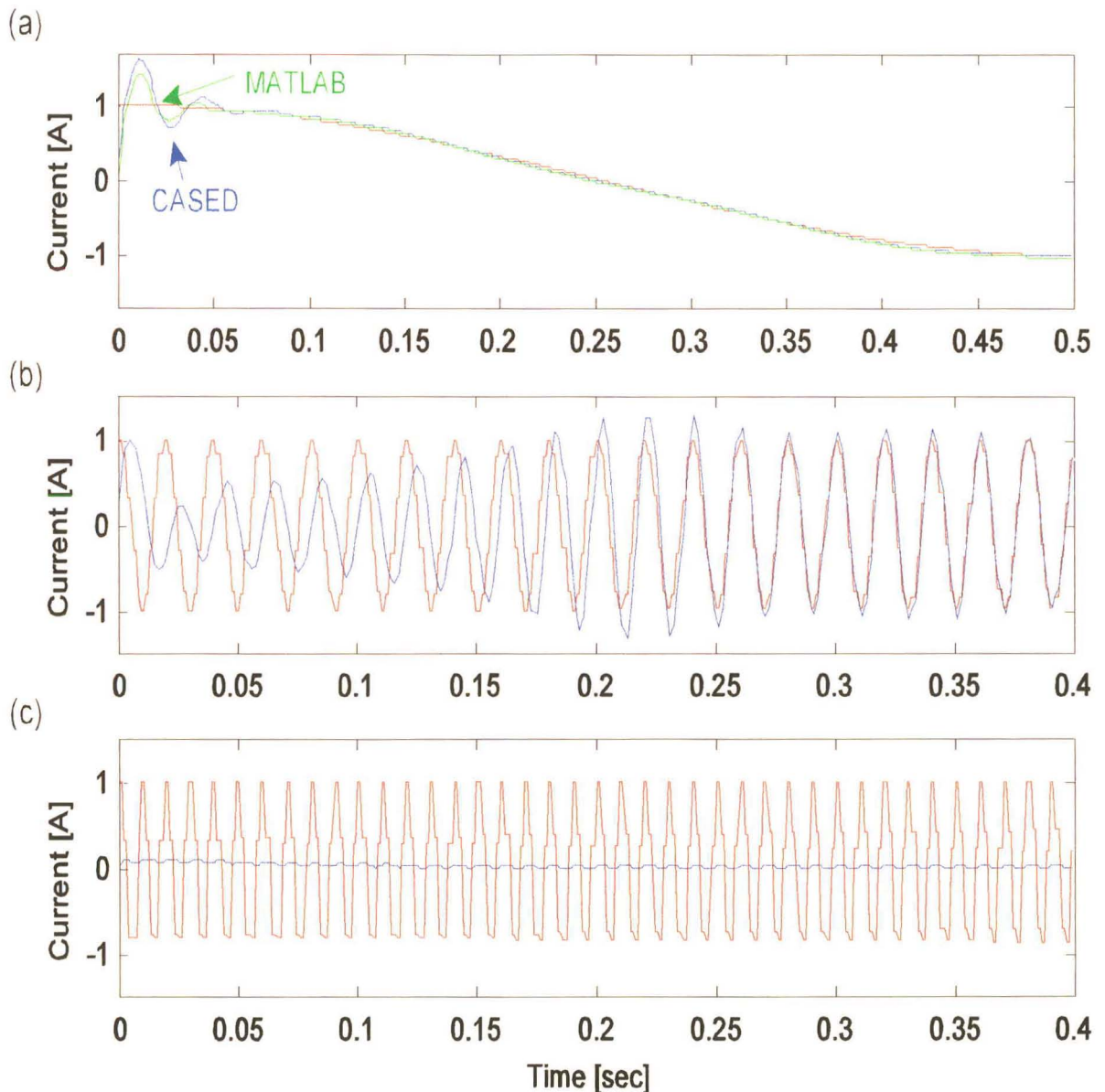


Fig. 3.6 Phase A motor supply current (I_{a_supply}) and COT ANN alpha reference current (I_{α_ref}), with a sampling rate of 500 Hz, learning rate of 0.1 pu, c_v of 1.5 pu, and no pretraining at stator frequencies of (a) 1 Hz (CASED and MATLAB results), (b) 50 Hz (CASED) and (c) 100 Hz (CASED) respectively.

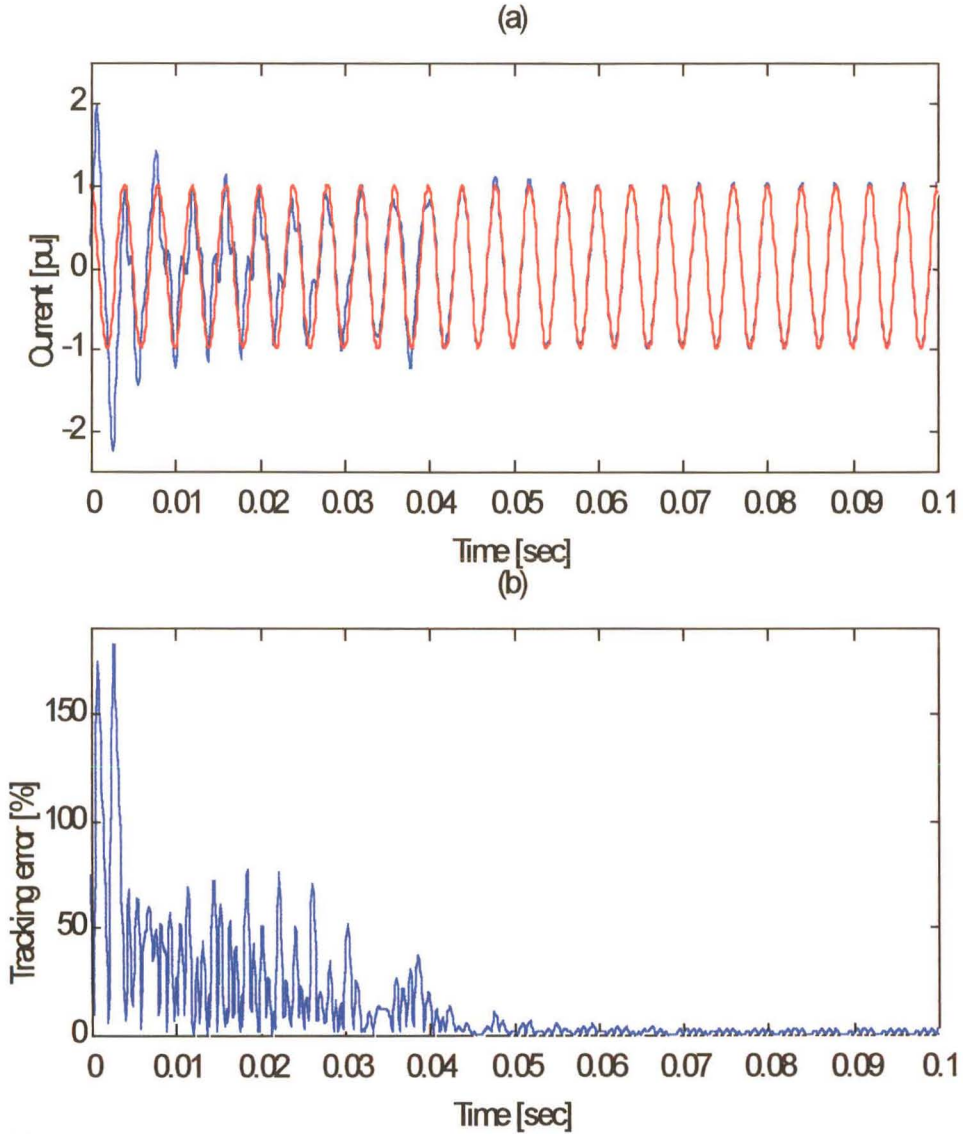


Fig. 3.7 (a) Phase A motor current (I_{a_supply}) and COT ANN alpha reference current (I_{a_ref}), with a sampling frequency of 10 kHz and learning rate of 0.1 pu, (b) Current tracking error

The next section considers the addition of the inverter and PWM controller into the simulation model, to investigate their effects on the operation of the system.

3.3 Investigation of the performance of a COT ANN Applied to a VSI fed SCIM Variable Speed Drive

Having successfully duplicated Burton's Matlab simulations in CASES, it is now extended by including an inverter and PWM controller at the input of the induction motor. The COT ANN current controller (code provided in APPENDIX B.3) now feeds the alpha and beta controlling voltages to the PWM controller (code provided in APPENDIX B.4), which in turn controls the inverter, as shown in Fig. 3.8, to achieve the desired system response. The system shown in Fig. 3.8 is simulated to investigate the nonlinear effects of the inverter on the COT ANN current controller. These simulations are performed to accurately simulate Burton's practical system, to be able to quantify the discrepancies noted between Burton's practical and simulation results [BURTON1].

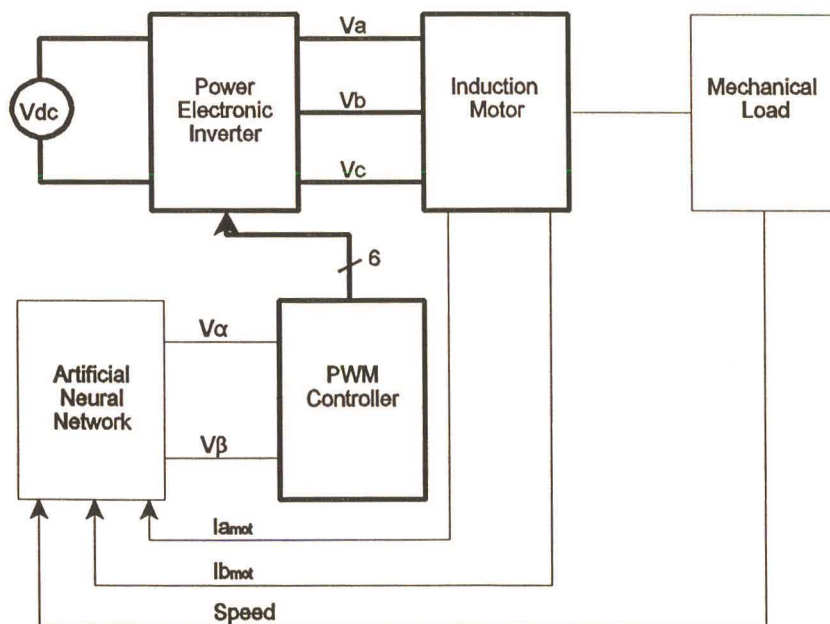


Fig. 3.8 Block diagram of a COT ANN controlled SCIM drive

The model shown in Fig. 3.8 consists of five user-defined modules linked together in the CASES work space to form a closed loop system, namely:

- 1) a COT ANN stator current controller module developed by Burton [BURTON1] C code provided in APPENDIX B.3). Fig. 3.9 shows the flow diagram of the COT ANN current controller module source code. The following main steps are sequentially executed in the program:
 - Step 1: All the simulation parameters and variables (e.g. sampling period), identification ANN parameters and variables (e.g. ANN size, gains, initial weights, inputs), induction motor parameters and variables (e.g. impedances, initial conditions etc.) are initialised.
 - Step 2: The main loop starts when an enable or interrupt signal is received.
 - Step 3: Steps 4 to 9 are omitted if the loop counter value is less than the corresponding start time T_{start} .
 - Step 4: v_α and v_β are written to the predefined port memory address for processing by the PWM controller model.
 - Step 5: The digital values of the A and C phase motor line currents and shaft encoder position are then read in from the predefined dual port memory addresses and converted to floating point numbers representing the sampled $\alpha\beta$ motor currents $\underline{i}(k)$ (shown as I_{amot} and I_{bmot} in Fig 3.3) and shaft speed $\omega(k)$.
 - Step 6: The ANN input $\underline{x}(k+1)=\underline{\hat{x}}(k)=[\underline{i}(k), \underline{i}(k-1), \omega(k), \omega(k-1), \underline{v}(k-1)]$ is formed.
 - Step 7: $\underline{x}(k+1)$ is forward propagated through the ANN to obtain $\underline{y}(k)=\underline{\hat{f}}_{cl}^{(k-1)}(.)$.
 - Step 8: $\underline{i}^*(k+1)$ is calculated, either with or without the use of the reference model [BURTON1], and used with $\underline{\hat{f}}_{cl}^{(k-1)}(.)$ and constant c_v to calculate $\underline{v}^*(k)$ using Eq. (2.11) from Chapter 2.
 - Step 9: The ANN output error $e_y(k) = \underline{i}(k) - \underline{i}^*(k)$ is calculated and backpropagated through the ANN to calculate $W(k+1)$ and $V(k+1)$, which represents $\underline{\hat{f}}_{cl}^{(k-1)}(.)$, for the next sampling period.
 - Step 10: Step 11 is omitted if the specified loop count corresponding with T_{end} has been reached.
 - Step 11: The main loop counter is updated and the program returns to step 2.

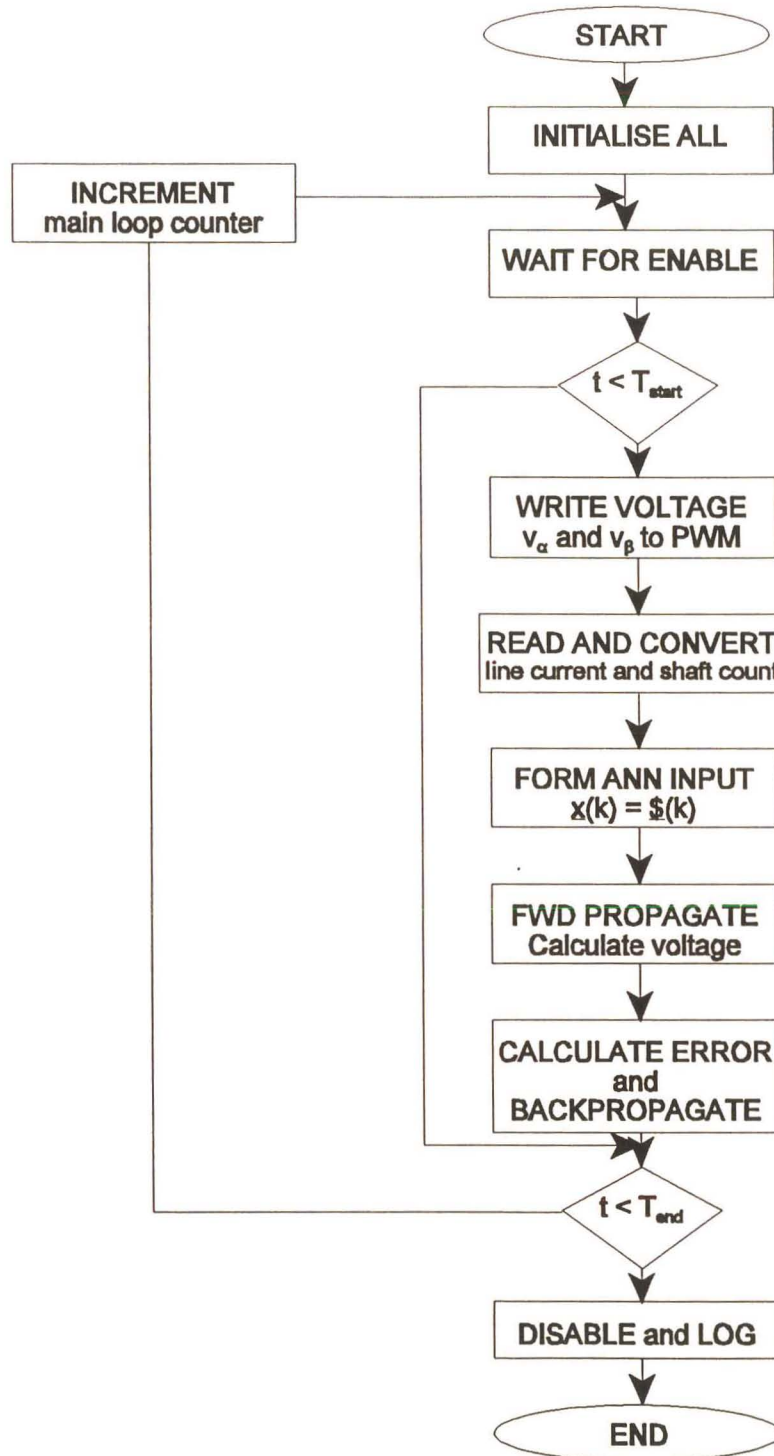


Fig. 3.9 Flow diagram of COT ANN CASED module for control of a VSI fed SCIM VSD

-
- 2) a PWM controller module developed by Chathury [CHATHURY1] C code provided in APPENDIX B.4). Fig. 3.10 shows the flow diagram of the COT ANN current controller module source code. The following main steps are sequentially executed in the program:
- Step 1: The module waits for a start signal.
 - Step 2: The module reads in the v_α and v_β values from the predefined port memory address, where they were saved by the COT ANN controller.
 - Step 3: The module calculates modulating signals, by simulating the Hanning PBM 1/87 ASIC's [HANNING1] operation (this ASIC is a PWM generator). This ASIC is used by Burton [BURTON1-3] and Chathury [CHATHURY1-3] in their respective research.
 - Step 4: The switching times are calculated.
 - Step 5: The module then generates CASED switching events, which are written to predefined port address to control the power electronic inverter.
- 3) a power electronic inverter module (CASED library model).
- 4) a induction motor module (code provided in APPENDIX B.2), as described in section 3.2 above. Fig 3.5 shows the flow diagram of the induction motor modules source code.
- 5) a mechanical load (a simple inertial load) module (CASED library model).

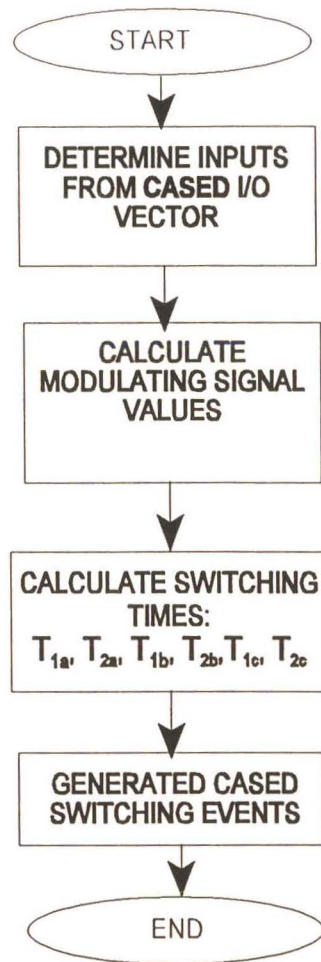


Fig. 3.10 Flow diagram of PWM CASED module

The COT ANN current controller in section 3.2, using the identical initial conditions, is also used to control the SCIM model in these simulations, except that it now feeds the controlling voltages to the PWM module (as shown in Fig. 3.9). The initial simulations are performed at a number of stator frequencies, for comparison with the results obtained in section 3.2, and the results appear in Fig 3.11. These results show that the COT ANN current controlled SCIM drive, tracks a 1 Hz current, but loses convergence at 50 Hz. In comparison, the simulation results in Fig. 3.6 (a) and Fig. 3.11 (a) show that, in both instances, the controller takes approximately 60 milliseconds to get the motor supply current to converge to the reference current 1 Hz. In both cases, the initial overshoot is limited to between 0.7 and 0.9 pu. This shows that the COT ANN

current controller performance is identical in both models at the lower frequencies (1 Hz), but that there is a discrepancy at higher frequencies (50 Hz), as shown in Fig. 3.6 (b) and Fig. 3.11 (b). In Fig. 3.6 (b), it is seen that the controller takes 250 milliseconds to get the motor supply current to converge to the 50 Hz reference current, whereas in Fig. 3.11 (b) initial tracking is achieved after 70 milliseconds, whereafter the system becomes divergent. These results demonstrate the problem that Burton experienced when implementing the COT ANN current controller, at motor frequencies above 3 Hz [BURTON1]. Thus the remainder of this section attempts to determine the possibility of using a COT ANN current controller with VSDs over a wide frequency range, under realistic conditions (i.e. including the nonlinear effects of the practical system). This is done to investigate the feasibility of applying the COT ANN to the Boost rectifier, which operates at 50 Hz.

The initial step is to change the COT ANN's learning rate and sampling rate, as described in section 3.2, to try to obtain convergent system operation [BURTON1]. Unfortunately this process does not provide convergent operation, which means that all the COT ANN current controller's internal constants and variables have to be investigated. This investigation is done using systematic simulations, to determine the link between the COT ANN current controllers tracking capability, and the COT ANN's internal constants and variables. Previously Burton [BURTON1] recommended that the voltage constant c_v be optimised as part of future research, in order to optimise the COT ANN current controller performance. Fig. 3.12 to 3.20 are sample plots which show how the COT ANN tracking capability varies, as the COT ANN sampling rate, and voltage constant c_v , which is dependent on K as shown in Eq. (3.1) [BURTON1], are varied, with a fixed motor frequency.

$$C_v = \left(\frac{1}{K} \right) \cdot \left(\frac{T_s}{L_{11} \cdot \text{Sigma}} \right) \cdot \left(\frac{V_{base}}{I_{base}} \right) \quad (3.1)$$

where: T_s is the inverse of the sampling frequency, known as the sampling rate,
 K is a scaling factor,
 L_{11} is the induction motor impedance,
 Sigma is a constant dependent on the motor parameters, and
 V_{base} and I_{base} are the voltage and current base values.

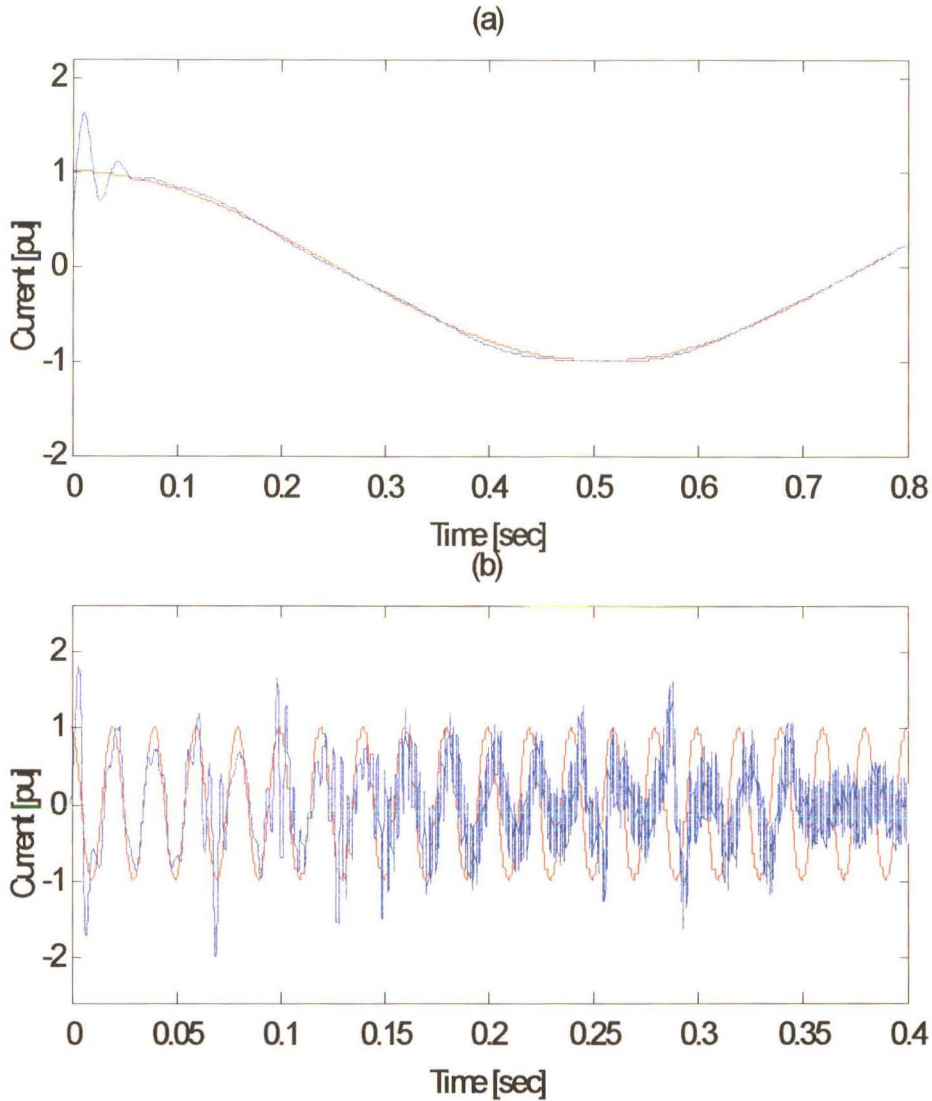


Fig. 3.11 Phase A motor supply current (I_{a_supply}) and COT ANN alpha reference current (I_{α_ref}), with sampling frequency of 10 kHz, learning rate of 0.5 pu and C_v of 1.5 pu, at motor frequencies of (a) 1 Hz and (b) 50 Hz respectively

Figs. 3.12 to 3.20 are sample plots taken from the optimisation process, to show the reader how the controllers performance varies as the internal parameters are varied. These parameter dependencies are then plotted to determine the best operating point for the COT ANN current controller.

Figs. 3.12, 3.13 and 3.14 are obtained using sampling frequencies of 1 kHz, 5 kHz and 10 kHz respectively, with a K of 0.1 pu, while fixing the rest of the variables and constants. At low sampling frequencies (1 kHz), the COT ANN current controller can not get the motor supply current to track the reference current, as shown in Fig. 3.12. As the COT ANN sampling frequency is increased to 5 kHz (as shown in Fig. 3.13) the system begins to track the reference current, with a low initial overshoot after approximately 200 milliseconds of training. As the sampling frequency is increased to 10 kHz (Fig. 3.14), the system begins to track the reference current after only 25 milliseconds, but has an initial current overshoot of approximately 1 pu. It can be deduced that there is an optimal operating point for the COT ANN current controller.

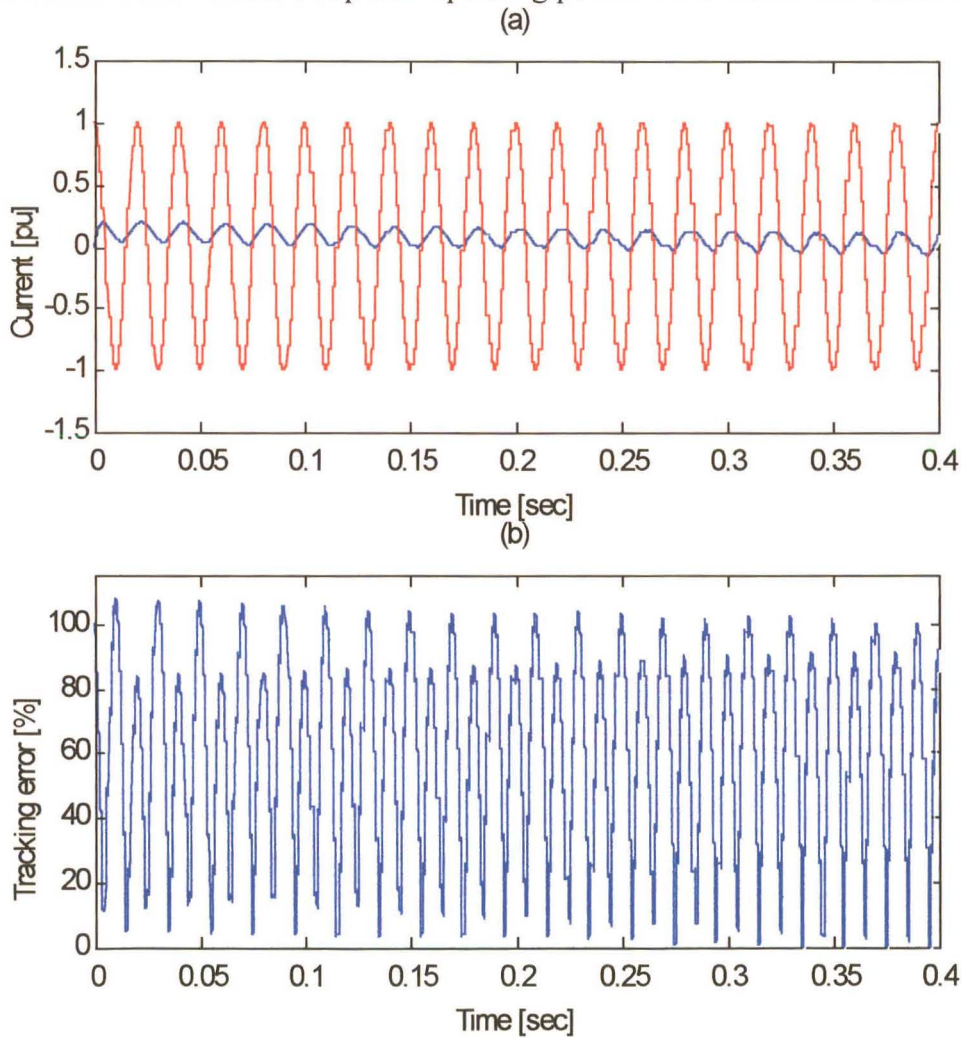


Fig 3.12 (a) Phase A motor supply current (I_{a_supply}) and COT ANN alpha reference current (I_{α_ref}), with $K = 0.1$ pu, sampling frequency of 1 kHz, learning rate of 0.01 pu, stator frequency of 50 Hz and stator current magnitude of 1 pu, (b) Current tracking error

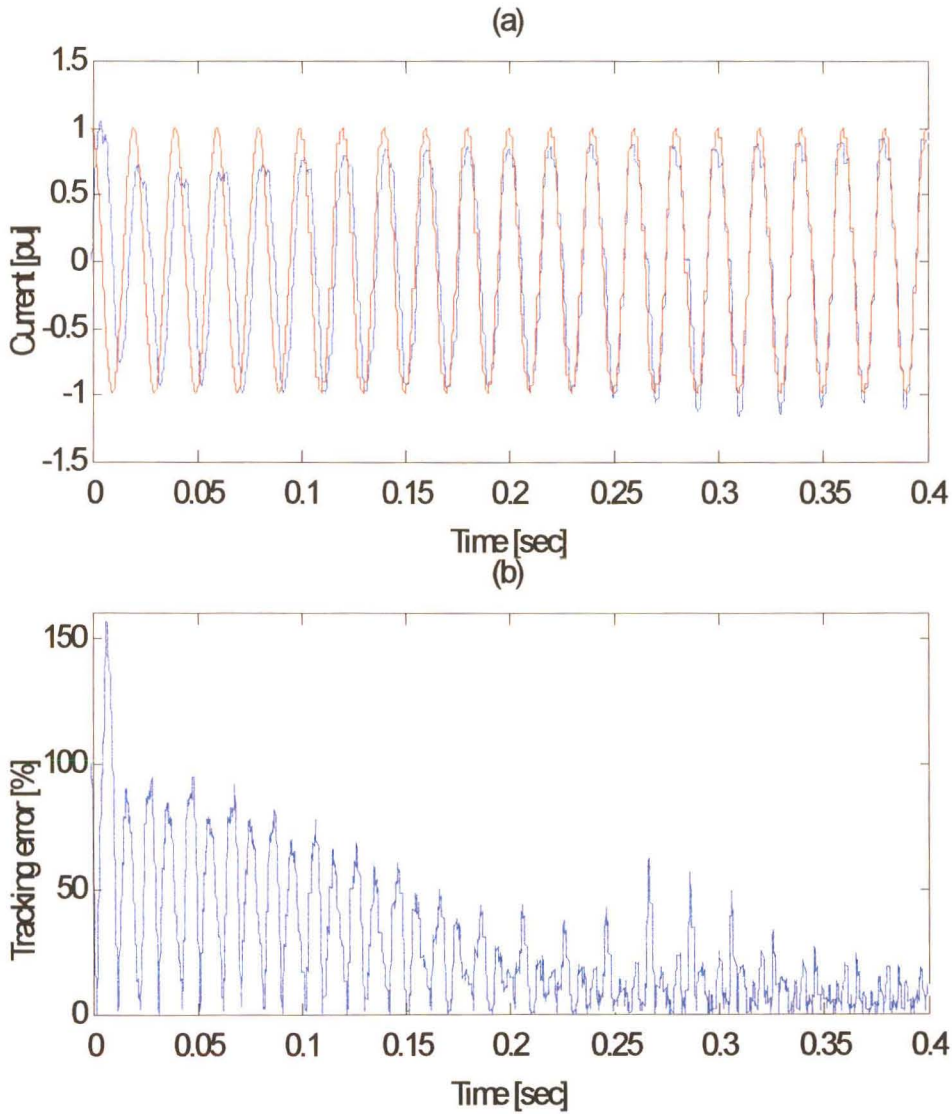


Fig 3.13 (a) Phase A motor supply current (I_{a_supply}) and COT ANN alpha reference current (I_{α_ref}), with $K = 0.1$ pu, sampling frequency of 5 kHz, learning rate of 0.01 pu, stator frequency of 50 Hz and stator current magnitude of 1 pu, (b) Current tracking error

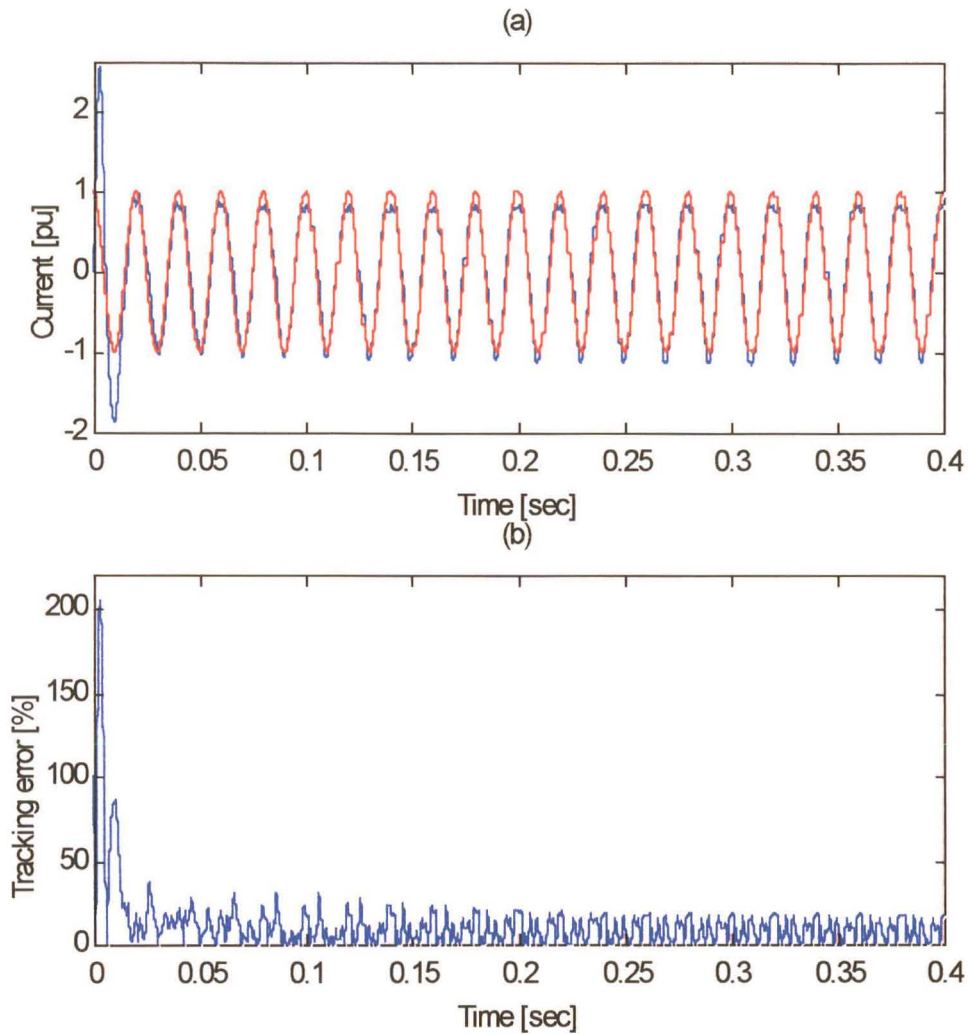


Fig 3.14 (a) Phase A motor supply current (I_{a_supply}) and COT ANN alpha reference current (I_{α_ref}), with $K = 0.1$ pu, sampling frequency of 10 kHz, learning rate of 0.01 pu, stator frequency of 50Hz and stator current magnitude of 1 pu, (b) Current tracking error

Figs. 3.15, 3.16 and 3.17 are obtained using sampling frequencies of 1 kHz, 5 kHz and 10 kHz respectively, with a constant K of 0.6 pu, while fixing the rest of the variables and constants.

When comparing Figs. 3.12 and 3.15, it can be seen, that the COT ANN controller's tracking capability has improved at lower sampling frequencies (1 kHz). As the sampling frequency is increased to 5 kHz, the system begins to track the reference current after approximately 30 milliseconds of training, with an overshoot of four times the rated current (Fig. 3.15). Whereas with a K of 0.1 pu (Fig. 3.13) at a 5 kHz sampling frequency, the system begins to track the reference current, with a low initial overshoot after approximately 200 milliseconds of training. From this it can be seen that a compromise has to be made between initial overshoot, and convergence time.

Furthermore, as the sampling frequency is increased to 10 kHz, the system begins to track the reference current after approximately 20 milliseconds of training, with an overshoot of approximately 3 pu (Fig. 3.17). After two further cycles the system loses its convergence, and takes approximately a further 100 milliseconds to re-achieve convergence. Whereas with a K of 0.1 pu (Fig. 3.14) at a 10 kHz sampling frequency, the system begins to track the reference current after approximately 25 milliseconds of training with a 1 pu initial overshoot. This again highlights that there is an optimal operating point for the COT ANN controller. In summary it appears that the COT ANN current controller has superior tracking capability at the medium sampling frequency (5 kHz).

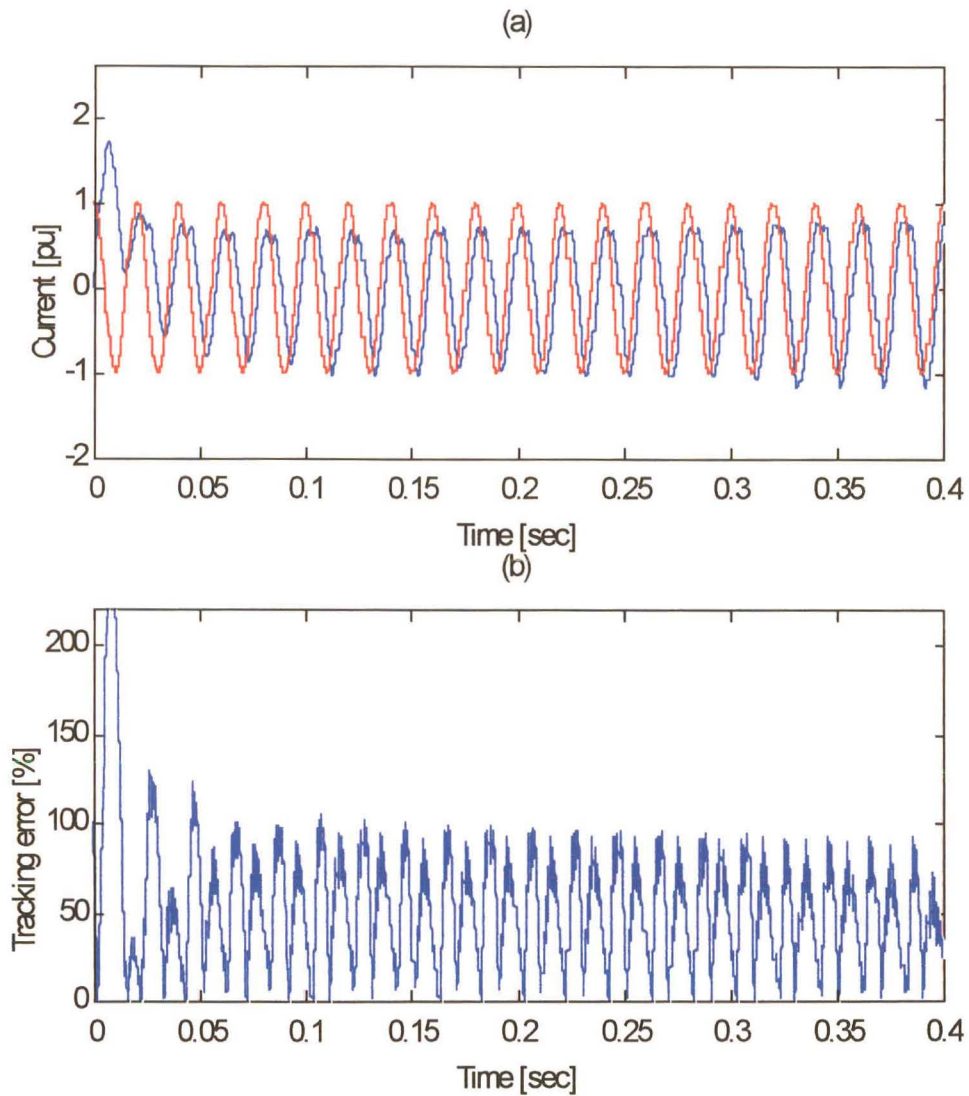


Fig 3.15 (a) Phase A motor supply current (I_{a_supply}) and COT ANN alpha reference current (I_{α_ref}), with $K = 0.6$ pu, sampling frequency of 1 kHz, learning rate of 0.01 pu, stator frequency of 50 Hz and stator current magnitude of 1 pu, (b) Current tracking error

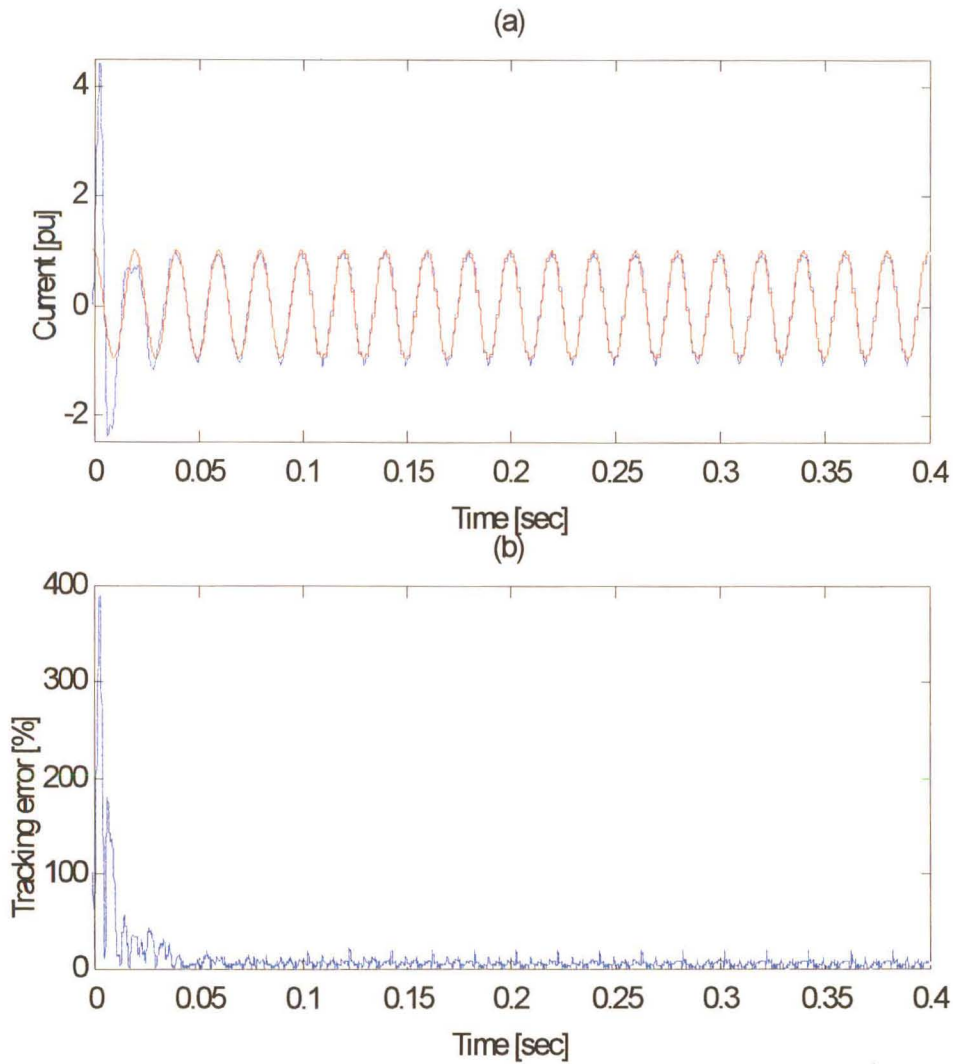


Fig 3.16 (a) Phase A motor supply current (I_{a_supply}) and COT ANN alpha reference current (I_{α_ref}),, with $K = 0.6$ pu, sampling frequency of 5 kHz, learning rate of 0.01 pu, stator frequency of 50Hz and stator current magnitude of 1 pu, (b) Current tracking error

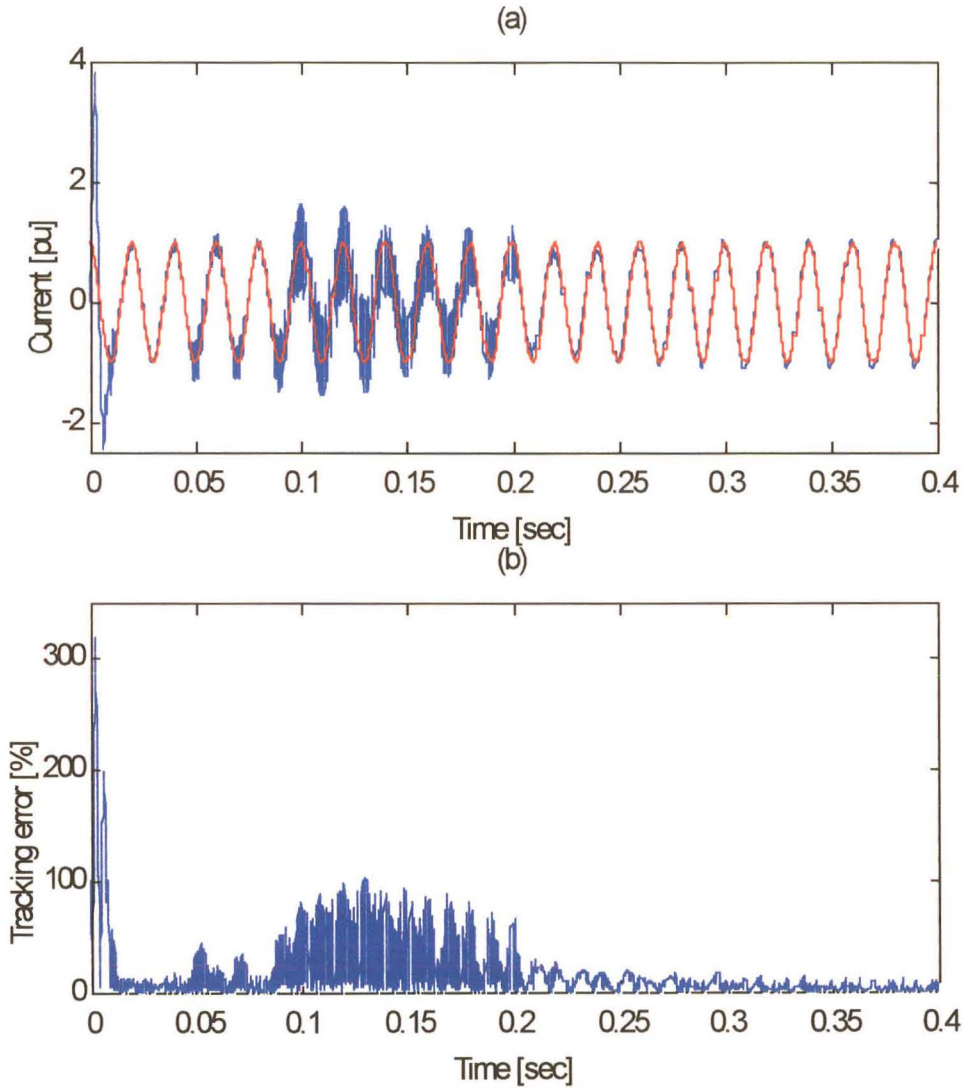


Fig 3.17 (a) Phase A motor supply current (I_{a_supply}) and COT ANN alpha reference current (I_{a_ref}), with $K = 0.6$ pu, sampling frequency of 10 kHz, learning rate of 0.01 pu, stator frequency of 50Hz and stator current magnitude of 1 pu, (b) Current tracking error

Figs. 3.18, 3.19 and 3.20 are obtained using sampling frequencies of 1 kHz, 5 kHz and 10 kHz respectively, with a K of 0.9, while fixing the rest of the variables and constants.

When comparing Figs. 3.12, 3.15 and 3.18, it can be seen, that the COT ANN controller's tracking capability has improved at lower sampling frequencies (1 kHz), but does not achieve complete current convergence at the low COT ANN sampling frequencies. As the sampling frequency is increased to 5 kHz, the system begins to track the reference current after approximately 18 milliseconds of training, with an initial overshoot of four times the rated current (Fig. 3.19). Whereas with a K of 0.1 pu (Fig. 3.13) and K of 0.6 pu (Fig. 3.16) at a 5 kHz sampling frequency, the system begins to track the reference current, after 200 milliseconds and 30 milliseconds of training respectively. However in the case with K of 0.9, the steady-state tracking error is higher, which suggests that the optimal value of K for best operation has been passed.

Furthermore, as the sampling frequency is increased to 10 kHz, the system can not track the reference current (i.e. is divergent) (Fig. 3.20). Whereas with a K of 0.1 pu (Fig. 3.14) and K of 0.6 pu, at a 10 kHz sampling frequency, the system begins to track the reference current after approximately 25 milliseconds and 20 milliseconds of training respectively. This again highlights that there is an optimal operating point for the COT ANN controller.

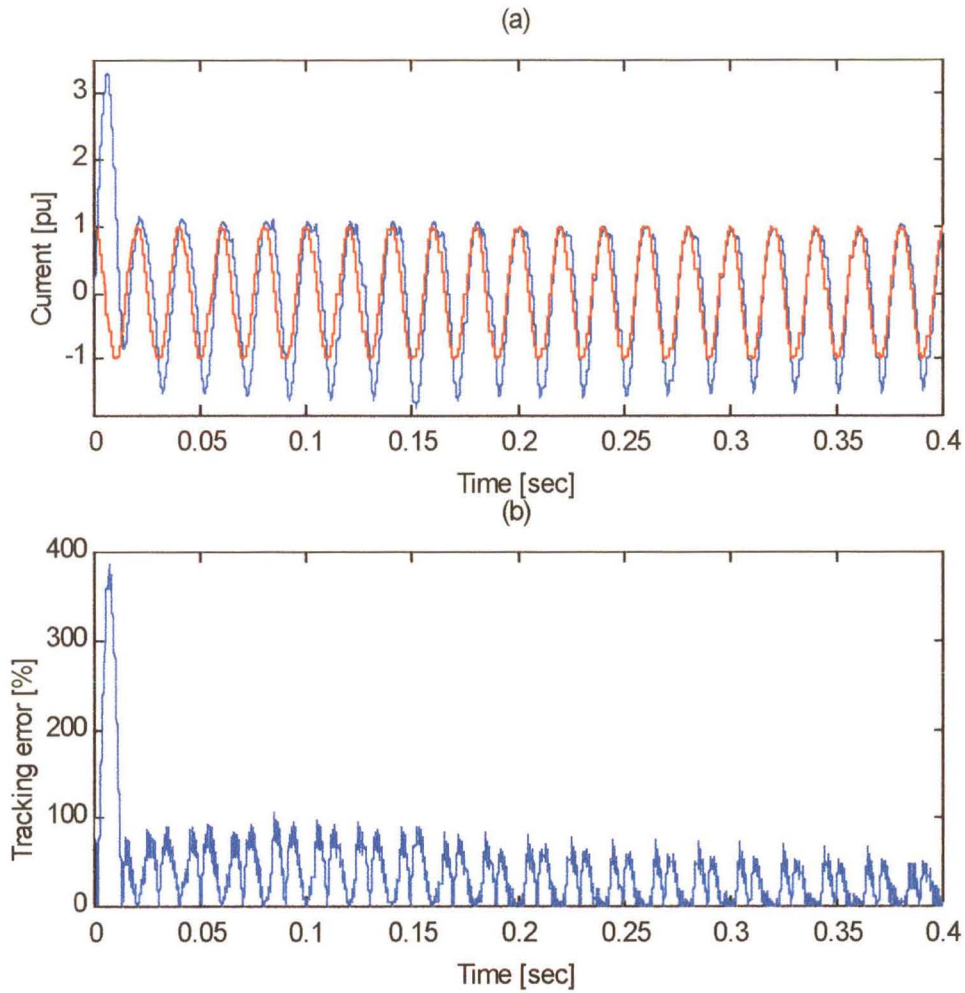


Fig 3.18 (a) Phase A motor supply current (I_{a_supply}) and COT ANN alpha reference current (I_{α_ref}), with $K = 0.9$ pu, sampling frequency of 1 kHz, learning rate of 0.01 pu, stator frequency of 50 Hz and stator current magnitude of 1 pu, (b) Current tracking error

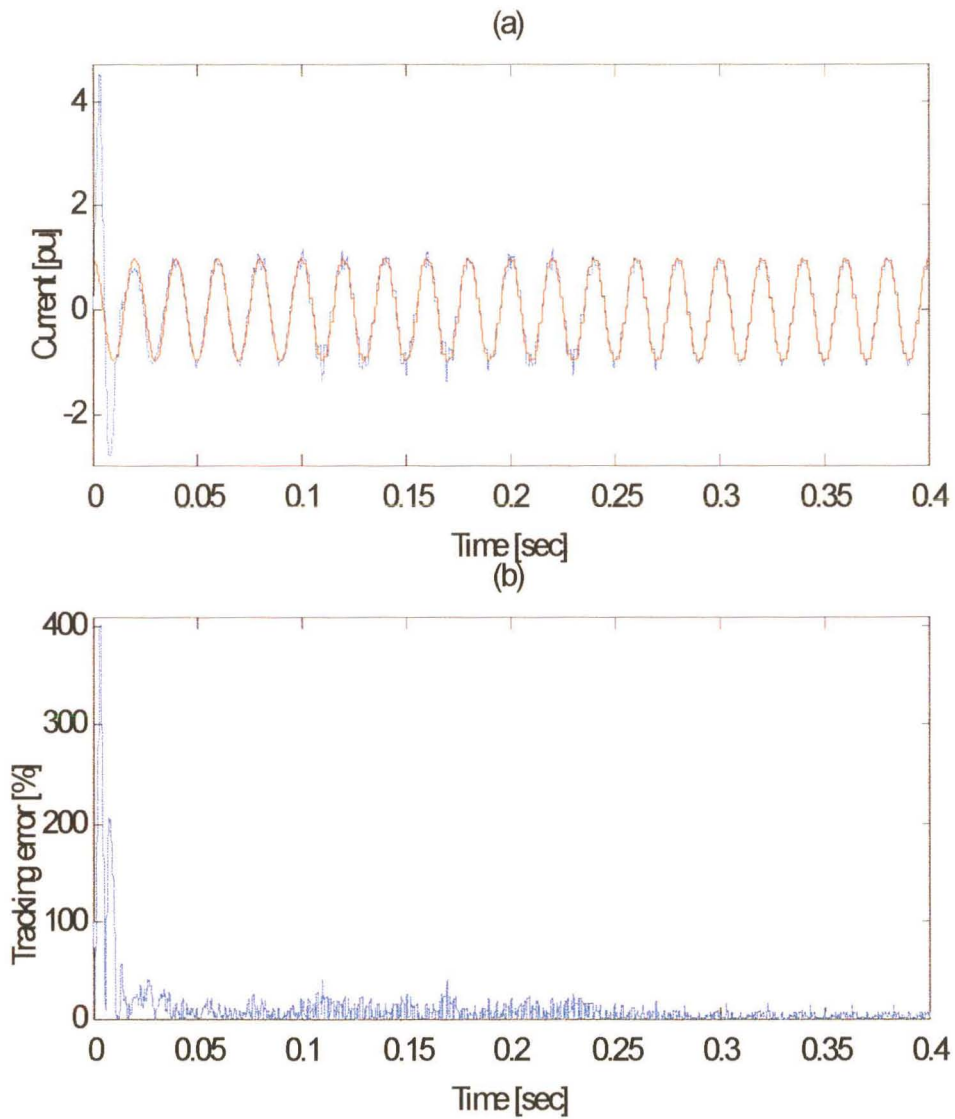


Fig 3.19 (a) Phase A motor supply current (I_{a_supply}) and COT ANN alpha reference current (I_{α_ref}), with $K = 0.9$ pu, sampling frequency of 5 kHz, learning rate of 0.01 pu, stator frequency of 50 Hz and stator current magnitude of 1 pu, (b) Current tracking error

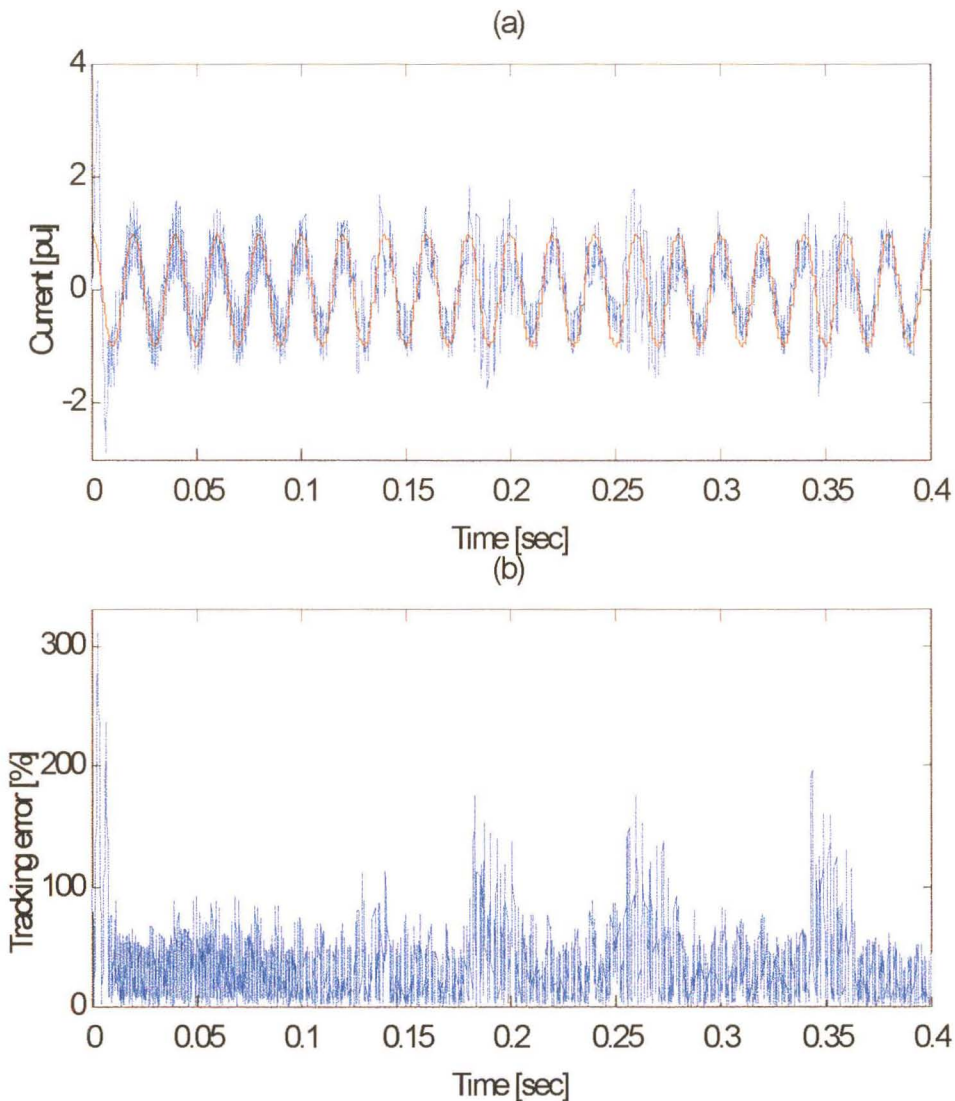


Fig 3.20 (a) Phase A motor supply current (I_{a_supply}) and COT ANN alpha reference current (I_{a_ref}), with $K = 0.9$ pu, sampling frequency of 10 kHz, learning rate of 0.01 pu, stator frequency of 50 Hz and stator current magnitude of 1 pu, (b) Current tracking error

When comparing the three sets of results shown above, it can be seen that there is a relationship between the COT ANN sampling frequency and the COT ANN voltage constant, and an optimal point for favorable COT ANN performance, which agrees with Burton's findings [BURTON1]. Further simulations are performed, and analysed to ascertain what conditions are required for the COT ANN controller to operate at high motor frequencies, while delivering the required response. Analysis of the simulation results reveals that the transient response and tracking capability of the COT ANN are dependent on many factors (not only on the sampling rate and voltage constant), namely:

- 1) steady-state motor Current magnitude,
- 2) motor frequency,
- 3) ANN learning rate (B),
- 4) ANN voltage constant (C_v)/ constant K,
- 5) ANN sampling frequency (T_s),
- 6) ANN momentum term, and
- 7) ANN hidden nodes.

These dependencies are investigated and discussed in greater detail. The momentum parameter has very little effect on the performance of the COT ANN controller, so it is not included in this investigation. Furthermore an investigation into optimising the number of hidden nodes is beyond the scope of this thesis. For this discussion the assumption is made, that the COT ANN current controller should have reached steady state operation after a single 50 Hz cycle (i.e. in 20 milliseconds). Using this assumption two plots of each dependence can be provided, namely:

- 1) before steady-state is achieved (from start-up to the first 20 milliseconds of operation), and
- 2) after steady-state is achieved.

The COT ANN error plotted on these curves, is calculated as the maximum peak error point that occurred during the predetermined time spans discussed above. For example, before steady-state is reached, the maximum peak error that is achieved between time $t = 0$ seconds and time $t = 20$ milliseconds is used for the optimisation curves.

Motor Current Magnitude Dependence

At very low steady-state motor currents (0.1 pu or lower), the ANN tracking error is very large, but as the motor current magnitude is increased, it is seen that the ANN tracking error becomes smaller, and it can be assumed that the ANN current controller may exhibit stability problems at low motor currents, due to the non-linearities of the inverter.

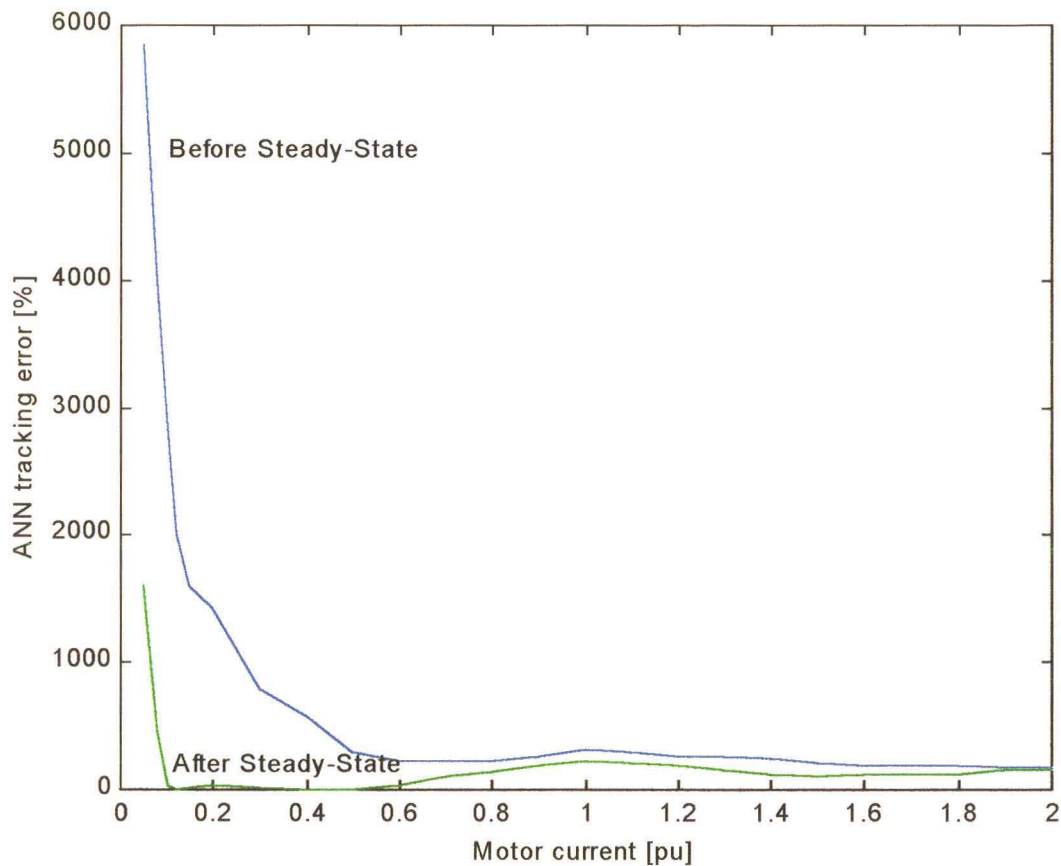


Fig. 3.21 Current dependence, $T_s=10$ kHz, $B=0.01$ pu, $C_v=0.6$ pu, and motor frequency $f=50$ Hz

Motor Frequency Dependence

The COT ANN displays large tracking errors at motor frequencies higher than 3 Hz, this suggests that the COT ANN could become unstable at large motor frequencies. This poses a problem for variable speed drives, as they can operate up to frequencies of 400 Hz or higher.

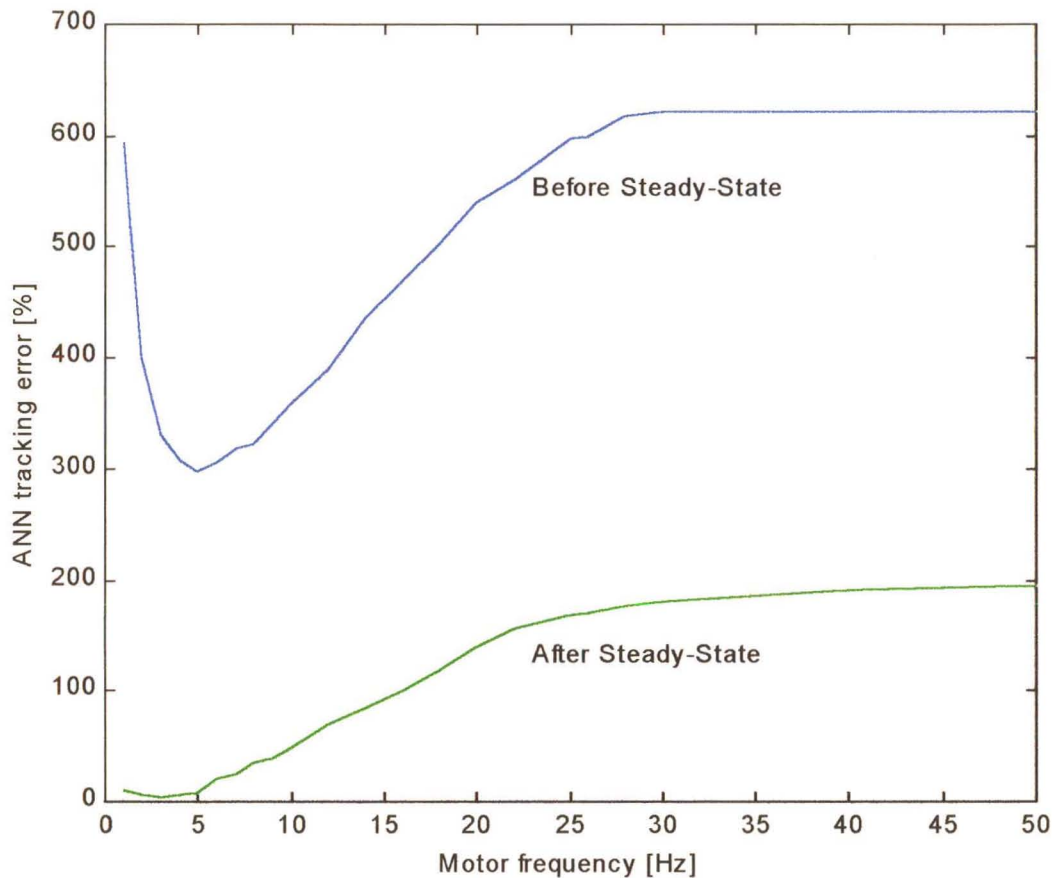


Fig. 3.22 Motor frequency dependence, $T_s=10$ kHz, $B=0.01$ pu, $C_v=0.6$ pu, and motor current $I=0.5$ pu

ANN learning rate dependence

To operate the ANN at high motor stator frequencies, it is found that at startup, the ANN requires a high learning rate (0.1), but during steady-state operation, the ANN needs a low learning rate (0.01) for stability.

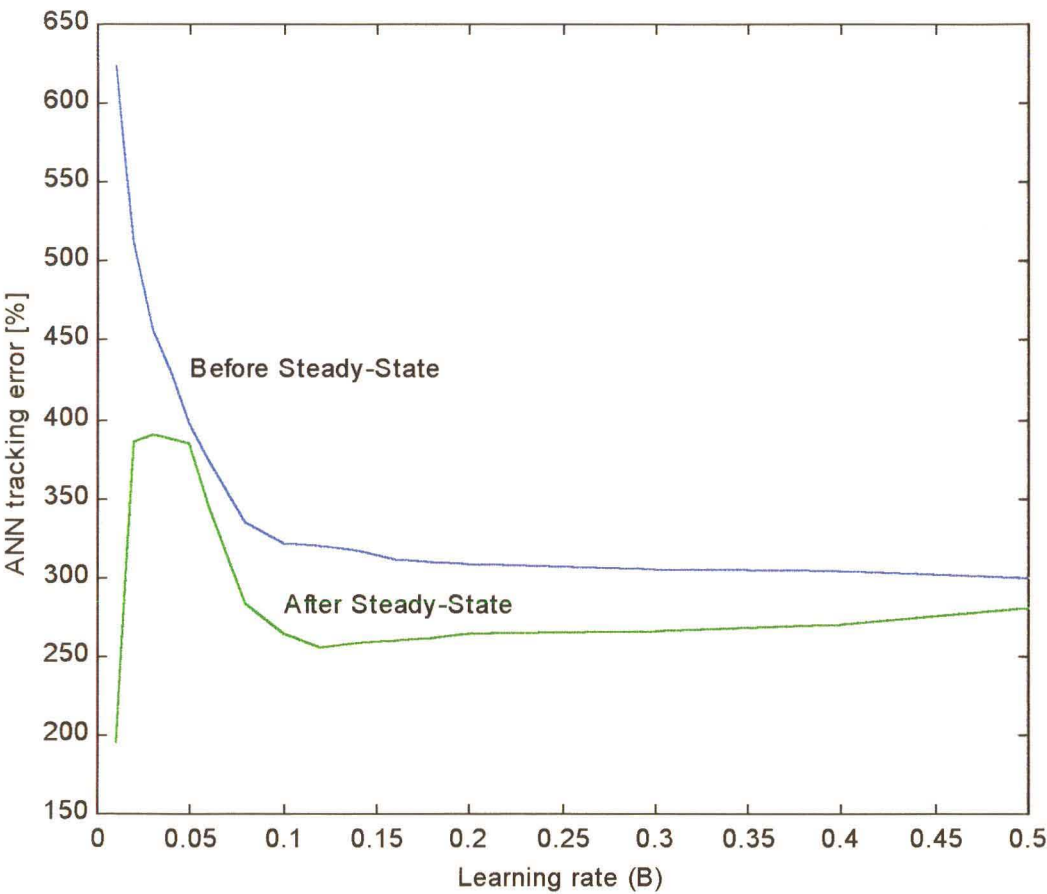


Fig. 3.23 ANN learning rate dependence, $T_s=10$ kHz, $C_v=0.6$ pu, $I=0.5$ pu and $f=50$ Hz

ANN Voltage Constant (C_v) dependence

The voltage constant, C_v , is a scaling factor for the ANN voltage feedback loop, which influences the ANN's transient tracking response. At start-up a large C_v is required to reduce the ANN tracking error, but once steady-state is achieved, it is noticed that a smaller C_v is required for stable ANN operation.

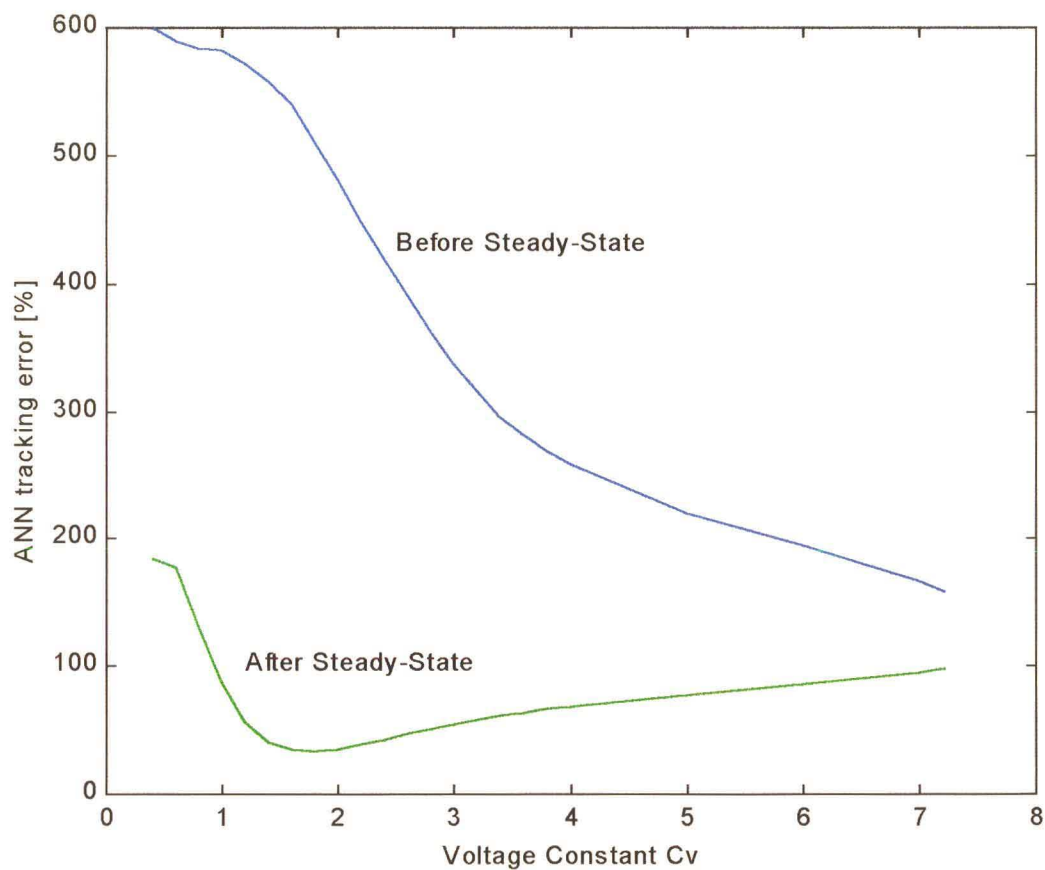


Fig. 3.24 ANN voltage constant dependence, $T_s=10$ kHz, $B=0.01$ pu, $I=0.5$ pu and $f=50$ Hz

ANN Sampling Frequency Dependence

A low sampling frequency (1 kHz) is required at start-up to reduce the ANN tracking error, but during steady-state operation, a higher sampling rate is required for ANN stability. This is due to the fact that at low sampling frequencies, the ANN training errors are lower, which allows it to approximate the function easier and attain lower initial overshooting. Fig. 3.25 shows that the most suitable area of stable operation of the ANN, after start-up, is between sampling frequencies of 5 to 8 kHz.

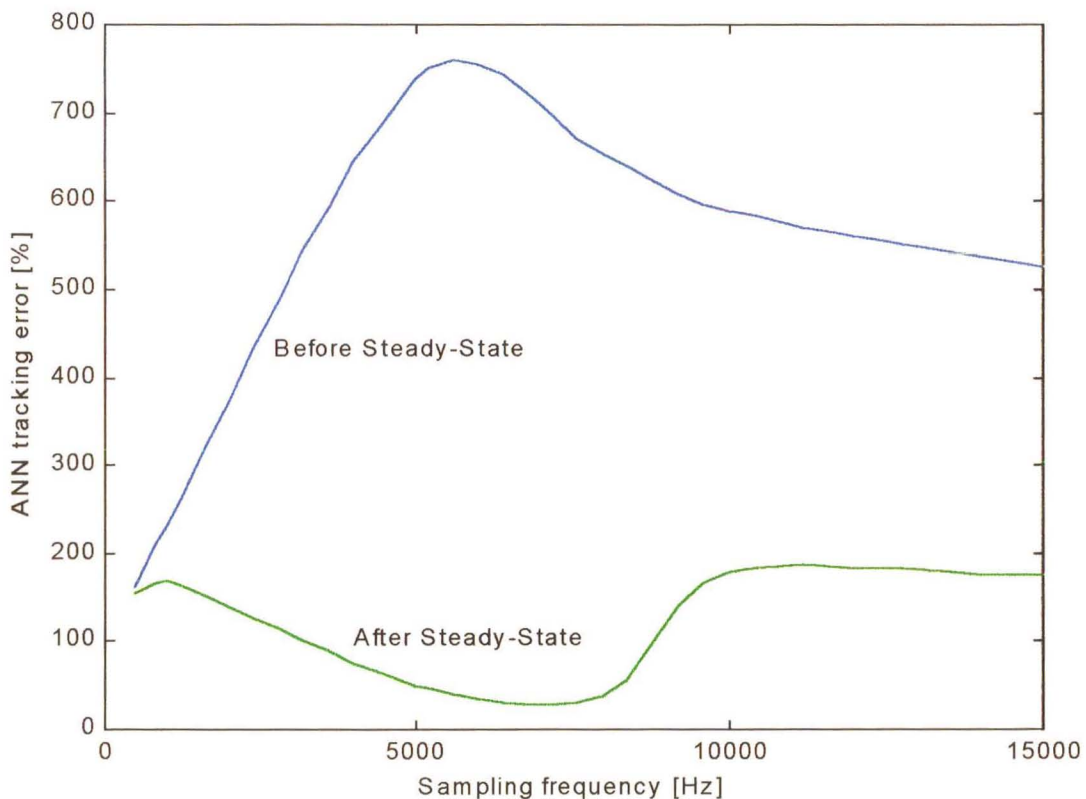


Fig. 3.25 ANN sampling frequency dependence, $C_v=0.6$ pu, $B=0.01$ pu, $f=50$ Hz and $I=0.5$ pu

These simulations show that the shortcomings experienced in Burton's practical system could be overcome by optimising the COT ANN current controller's internal variables and constants, only if the hardware platform can provide the necessary computational speed (Burton's transputer could not operate at sampling frequencies above 450 Hz) [BURTON1]. Under these conditions, it is possible for the COT ANN current controller to control the SCIM drive to stator frequencies of 50 Hz, as shown in Fig. 3.26.

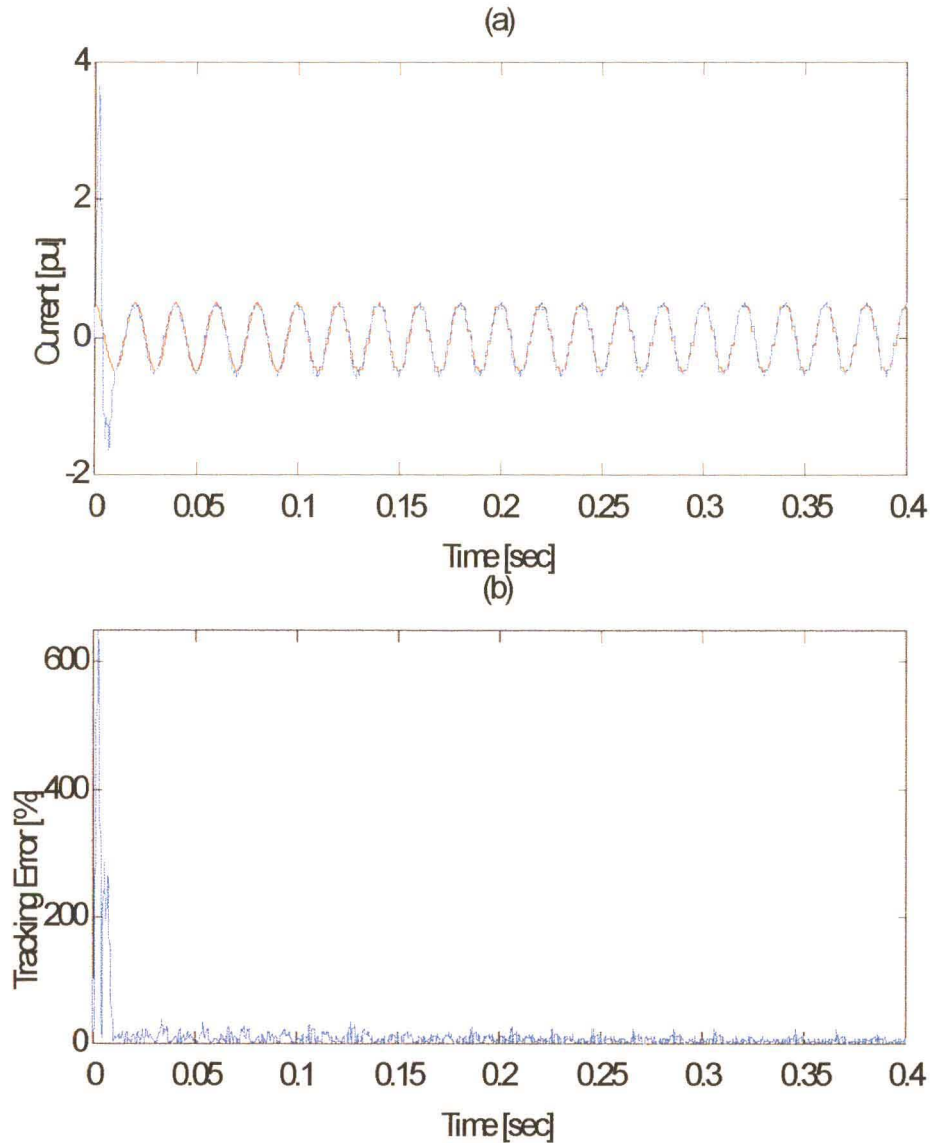


Fig. 3.26 (a) Phase A motor supply current (I_{a_supply}) and COT ANN alpha reference current (I_{α_ref}), with $K = 0.6$ pu, sampling frequency of 8 kHz, learning rate of 0.01 pu, stator frequency of 50 Hz, and stator current magnitude of 1 pu, (b) Current tracking error

Fig. 3.26 shows that the COT ANN controller forces the motor supply current to track the reference current after approximately 20 milliseconds, with an initial overshoot of 3,5 pu. In conclusion, it can be stated that it is important that the COT ANN variable dependencies be addressed when designing a COT ANN current controller, to obtain convergence, and an efficient transient response. It should be noted that the commissioning or initial overshoot is relatively

unimportant when designing a COT ANN (as the ANN starts at random weight states), it is more important to achieve a low steady-state transient error. This allows the ANN to easily adapt to any changing parameters in a system with minimal transient errors.

3.4 Summary

This chapter is used to gain familiarity with the concept of COT ANN current controllers, and to evaluate the performance of the COT ANN current controller developed by Burton. The analysis shows that the nonlinear effects introduced by the inverter and PWM interface in the SCIM drive model, affects the ANN current controller's dynamic performance, and causes its tracking capability (convergence) to be dependent on a number of factors, namely:

- 1) inverter AC side current magnitude,
- 2) inverter AC side frequency,
- 3) COT ANN sampling frequency,
- 4) COT ANN learning rate,
- 5) COT ANN voltage constant, C_v ,
- 6) COT ANN momentum term, and
- 7) number of inputs and outputs.

These dependencies are analysed and discussed, and it can be concluded that they need to be addressed when designing a ANN current controller to achieve efficient system control. This chapter also provided an overview of the CASED simulation package.

The next section describes the derivation of a COT ANN current controller for the control of a boost rectifier. Simulations are performed to determine the feasibility of using the ANN current controller for a boost rectifier to overcome the shortcomings of linear controllers.

CHAPTER 4

INVESTIGATION OF A CONTINUOUSLY ONLINE TRAINED ARTIFICIAL NEURAL NETWORK CURRENT CONTROLLER FOR A BOOST RECTIFIER SYSTEM

4.1 Introduction

Chapter 1 and 2 provided an overview of Chathury's work on boost rectifiers, and Burton's work on COT ANN current controllers for SCIM drives, and highlighted some of their shortcomings. As a result it was proposed to extend this research to apply a COT ANN current controller to a boost rectifier. In Chapter 3 a simulation study of Burton's COT ANN current controller was performed. The simulation work of Burton [BURTON1-2] was also extended by adding the effects of the PWM controller and the inverter into the simulations (see Chapter 3 section 3.3).

In this chapter, the feasibility of applying a COT ANN current controller to a boost Rectifier is investigated. The initial simulation study uses Burton's COT ANN current controller (developed for the control of an inverter, as discussed in Chapter 3) as a direct replacement for Chathury's current controller in the boost rectifier system.

Burton's Narmax model (mathematical representative method), which was used to derive an ANN structure for the SCIM, is used to derive a COT ANN model for the boost rectifier. This is then used to determine how the inputs, outputs and structure of Burton's COT ANN current controller may be adapted and applied as a current controller for a boost rectifier. The COT ANN current controller sign conventions, inputs and outputs are then reviewed and changed for the control of the boost rectifier.

The system configuration to be used throughout the remainder of this thesis is presented in the next section.

4.2 System Outline

As mentioned in Chapter 2, the objective is to investigate the use of a COT ANN current controller as a replacement for Chathury's current controller in the boost rectifier shown in Fig. 4.1; proportional or proportional/integral control is used to control the DC link voltage, and the COT ANN is used to control the rectifier current.

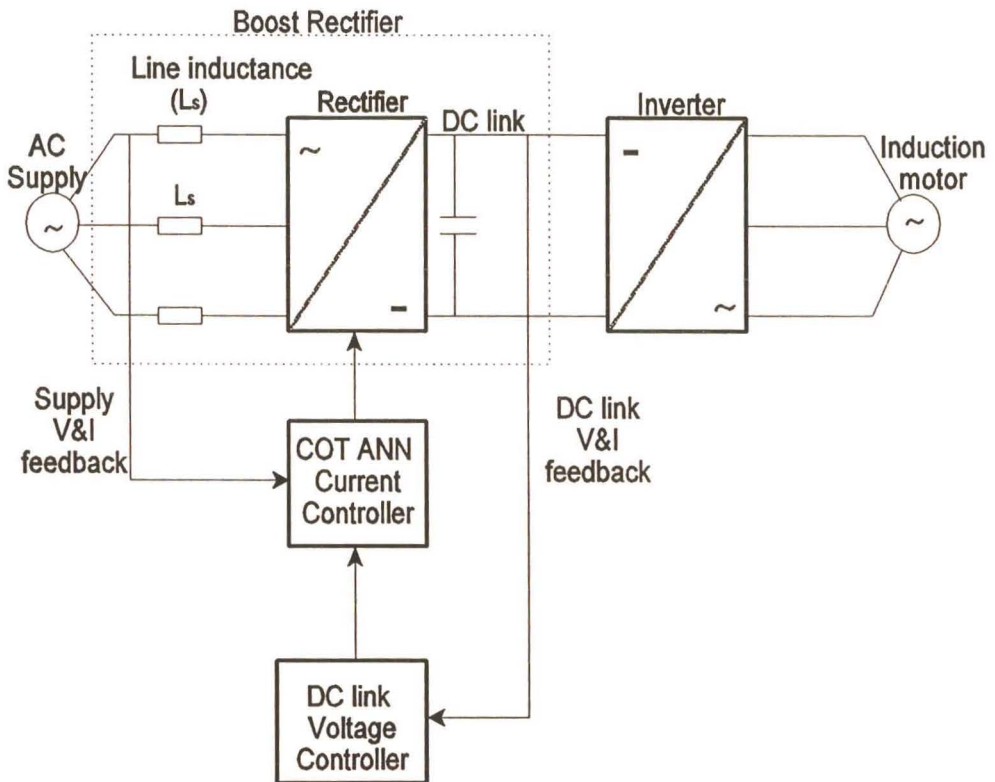


Fig. 4.1 Conventional control loops

4.2.1 COT ANN conventions

The COT ANN developed by Burton, as described in Chapter 2, for an SCIM may be applied to the boost rectifier, with minor changes to the COT ANN inputs, outputs and sign conventions. Fig. 4.2 shows the sign conventions for motoring and regeneration. In the case of Burton, the motoring block was adopted as the positive convention for the control of the SCIM (slip is positive). Therefore when controlling a boost rectifier, the regenerative block (Fig. 4.2) is adopted as the positive convention (here the slip is negative, as discussed in APPENDIX A [SAY1]). Thus to adapt Burton's COT ANN current controller to control a boost rectifier, the COT ANN's I/O and structure must be altered to use the regenerative convention as the positive norm. When changing sign conventions, the $\alpha\beta$ reference frame must be altered to achieve the desired system representation, as shown in Fig. 4.3 [CHATHURY1].

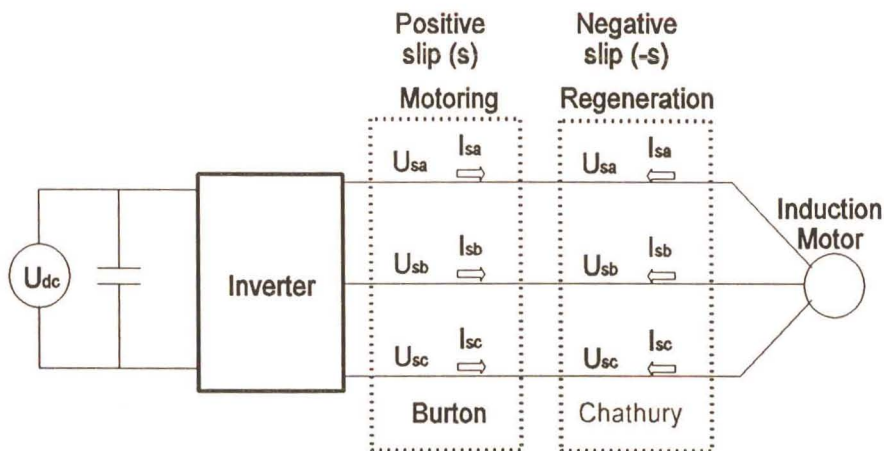


Fig. 4.2 Phase conventions for a three phase inverter

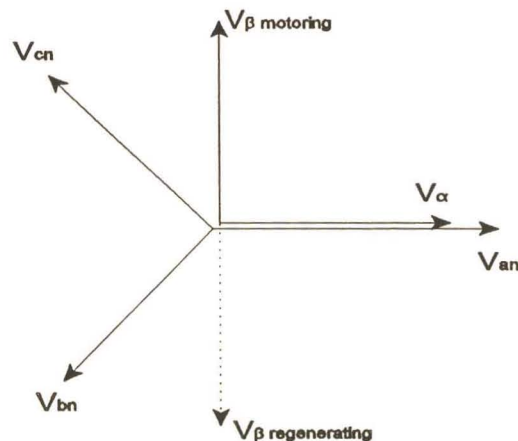


Fig. 4.3 Alpha/Beta reference frame

In this section a sign convention is chosen for the boost rectifier current controller, as described above, for use in the derivation of the Narmax model of the boost rectifier in the alpha beta reference frame [WISHART1]. The Narmax model is used to determine the structure, inputs and outputs for the COT ANN current controller for the boost rectifier [BURTON1], [WISHART1]. The mathematical model of the VSI-based boost rectifier (shown in Fig. 4.4) is derived assuming a balanced three-phase system. First the equations for the system are written in the three-phase a-b-c reference frame using Kirchoff's Voltage Law, and then transformed into the α - β reference frame by using the direct Park's transformation.

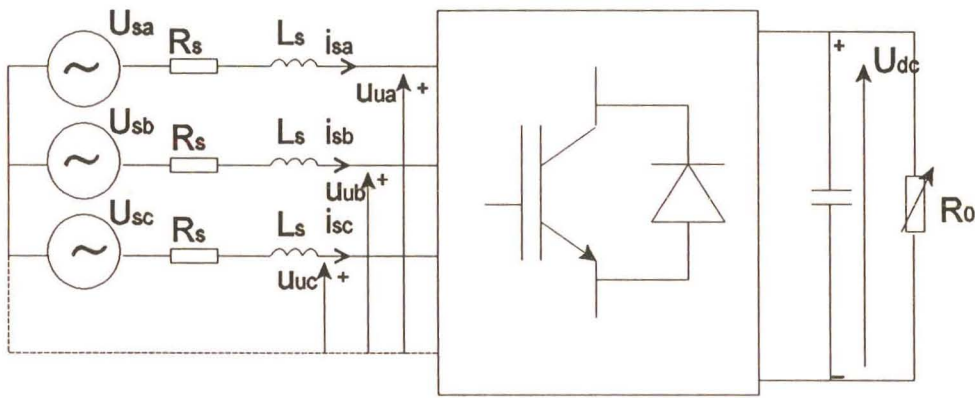


Fig. 4.4 Boost rectifier configuration

The equation describing the dynamics on the AC side of the boost rectifier may be written as follows:

$$\begin{aligned}
 L_s \frac{di_{sa}}{dt} + R_s i_{sa} &= U_{sa} - U_{ua} \\
 L_s \frac{di_{sb}}{dt} + R_s i_{sb} &= U_{sb} - U_{ub} \\
 L_s \frac{di_{sc}}{dt} + R_s i_{sc} &= U_{sc} - U_{uc}
 \end{aligned} \tag{4.1}$$

Eqn. (4.1) may be transformed, using Eqn. (4.2), into the α - β reference frame [CHATHURY1] to yield Eqn. (4.3).

$$\begin{bmatrix} U_\alpha \\ U_\beta \\ U_0 \end{bmatrix} = \sqrt{\frac{2}{3}} \cdot \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \cdot \begin{bmatrix} U_a \\ U_b \\ U_c \end{bmatrix} \quad (4.2)$$

$$\begin{aligned} L_s \cdot \frac{di_\alpha}{dt} &= -R_s \cdot i_\alpha + U_{s\alpha} - U_{u\alpha} \\ L_s \cdot \frac{di_\beta}{dt} &= -R_s \cdot i_\beta + U_{s\beta} - U_{u\beta} \end{aligned} \quad (4.3)$$

Converting Eqn. (4.3) into state-space yields:

$$\begin{bmatrix} \dot{i}_\alpha \\ \dot{i}_\beta \end{bmatrix} = -\frac{R_s}{L_s} \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} + \frac{1}{L_s} \cdot \begin{bmatrix} U_{s\alpha} \\ U_{s\beta} \end{bmatrix} - \frac{1}{L_s} \cdot \begin{bmatrix} U_{u\alpha} \\ U_{u\beta} \end{bmatrix} \quad (4.4)$$

The state-space matrix may then be used to derive a Narmax model [WISHART1] of the boost rectifier system (APPENDIX A). The relevant equation appears in Eqn. (4.5)

$$\begin{bmatrix} i_{\alpha(k+1)} \\ i_{\beta(k+1)} \end{bmatrix} = \begin{bmatrix} -\frac{R_s \cdot T_s}{L_s} + 1 & 0 \\ 0 & -\frac{R_s \cdot T_s}{L_s} + 1 \end{bmatrix} \cdot \begin{bmatrix} i_{\alpha(k)} \\ i_{\beta(k)} \end{bmatrix} + \frac{T_s}{L_s} \cdot \begin{bmatrix} U_{s\alpha(k)} \\ U_{s\beta(k)} \end{bmatrix} - \frac{T_s}{L_s} \cdot \begin{bmatrix} U_{u\alpha(k)} \\ U_{u\beta(k)} \end{bmatrix} \quad (4.5)$$

where: T_s is the ANN sampling rate,
 L_s is the system line impedance, and
 R_s is the system line resistance.

From Eqn. (4.5), it can be seen that the neural network identifier requires the source current, source voltage ($U_{s\alpha}$ and $U_{s\beta}$), and the boost rectifier input voltage ($U_{u\alpha}$, and $U_{u\beta}$), as inputs. It should be noted that speed is no longer an input into this model, because the source frequency of a boost rectifier is approximated as a constant of 50 Hz (this is entered as a constant into the ANN). If Eqn. (4.5) is compared to the structure of the COT ANN developed by Burton, as given by Eqn. (4.6) (as discussed in Chapter 2) it can be seen that the same control structure may

be applied to the boost rectifier; the only changes required are that the speed input is removed and the number of input nodes should be increased for improved bandwidth (this was determined during deliberation with Burton) [BURTON1]. These new input nodes are then used to input the delayed alpha and beta controlling voltages and currents, as shown in Fig. 4.5, to allow the COT ANN current controller improved controllability of the system. Using the alpha and beta delayed controlling voltages and currents as inputs to the ANN, allows the ANN to adapt to the system faster, thereby achieving faster convergence.

$$\underline{i}(k+1) = f_{el}(\underline{i}(k), \underline{i}(k-1), \omega(k), \omega(k-1), \underline{v}(k)) + c_v \cdot \underline{v}(k) \quad (4.6)$$

Where: $\underline{i}(k)$ is the per unit value of the $\alpha\beta$ stator current vector at time (k)

$\omega(k)$ is the per unit value of the shaft speed at time (k)

$\underline{v}(k)$ is the per unit value of the $\alpha\beta$ stator voltage applied at time (k)

c_v is the per unit value of the voltage constant defined by the motor parameters

$f_{el}(k)$ is the vector function defined by the motor parameters

The basic structure used for the training of the ANN under these conditions is shown in Fig.4.5.

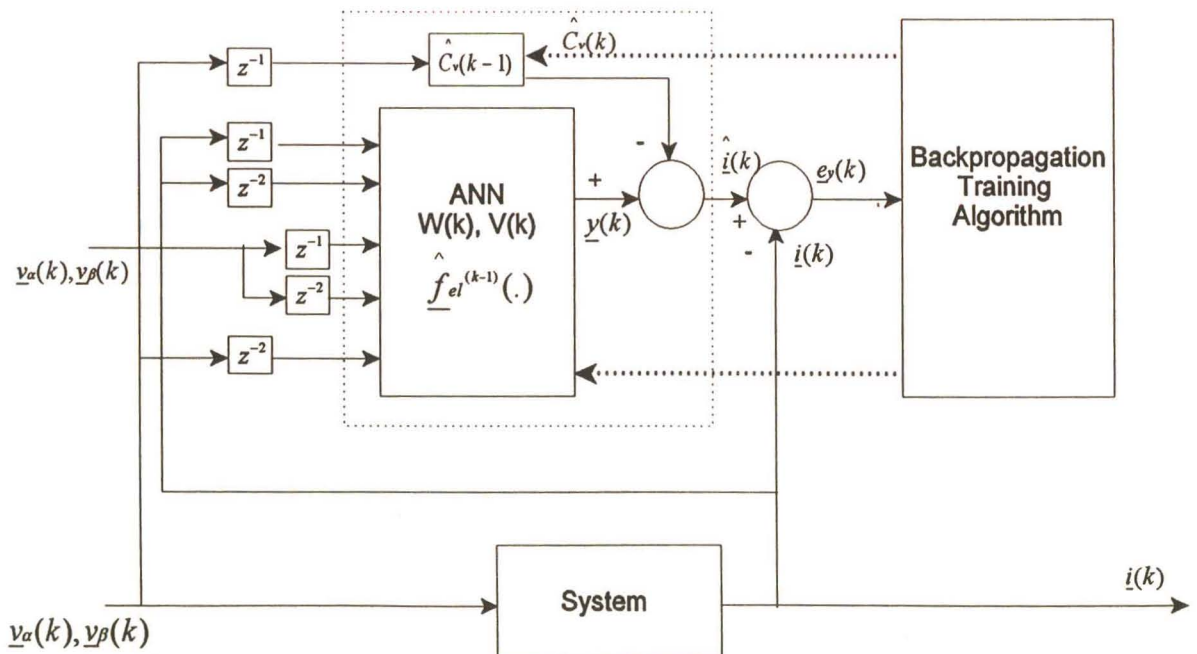


Fig. 4.5 Continual Online training for adaptive system identification

The next section describes the technique utilised to achieve unity power factor operation of the boost rectifier system.

4.2.2 Mains Synchronization

To achieve unity power factor operation, the COT ANN current controller must be synchronised to the mains supply by using the phase to neutral voltages as inputs to the COT ANN, as shown in Fig. 4.6. Fig. 4.6 shows the block diagram of Chathury's controller, indicating which section of the controller is to be replaced by the COT ANN current controller.

Wishart developed the COT ANN current controller to operate in the Alpha/Beta reference frame, as it allowed the COT ANN to learn online more effectively [WISHART1]. Thus, the sampled voltages and currents must be transformed to their Alpha/Beta components.

Voltage Equations:

$$V_{\alpha}(k) = (U_{sa} - \frac{1}{2} \cdot U_{sb} - \frac{1}{2} \cdot U_{sc}) / \sqrt{\frac{2}{3}} \quad (4.7)$$

$$V_{\beta}(k) = (U_{sb} - U_{sc}) / \sqrt{2} \quad (4.8)$$

and Current equations:

$$I_{\alpha}(k) = (I_{sa} - \frac{1}{2} \cdot I_{sb} - \frac{1}{2} \cdot I_{sc}) / \sqrt{\frac{2}{3}} \quad (4.9)$$

$$I_{\beta}(k) = (I_{sb} - I_{sc}) / \sqrt{2} \quad (4.10)$$

The Alpha/Beta voltages are input to the COT ANN, and used to calculate the per unit main's Sine and Cosine components as shown in Eq. (4.11) and (4.12).

$$\cos(\theta) = V_{\alpha}(k) / M_{agUs} \quad (4.11)$$

$$\sin(\theta) = V_{\beta}(k) / M_{agUs} \quad (4.12)$$

where: $M_{agUs} = \sqrt{V_{\alpha}^2 + V_{\beta}^2}$

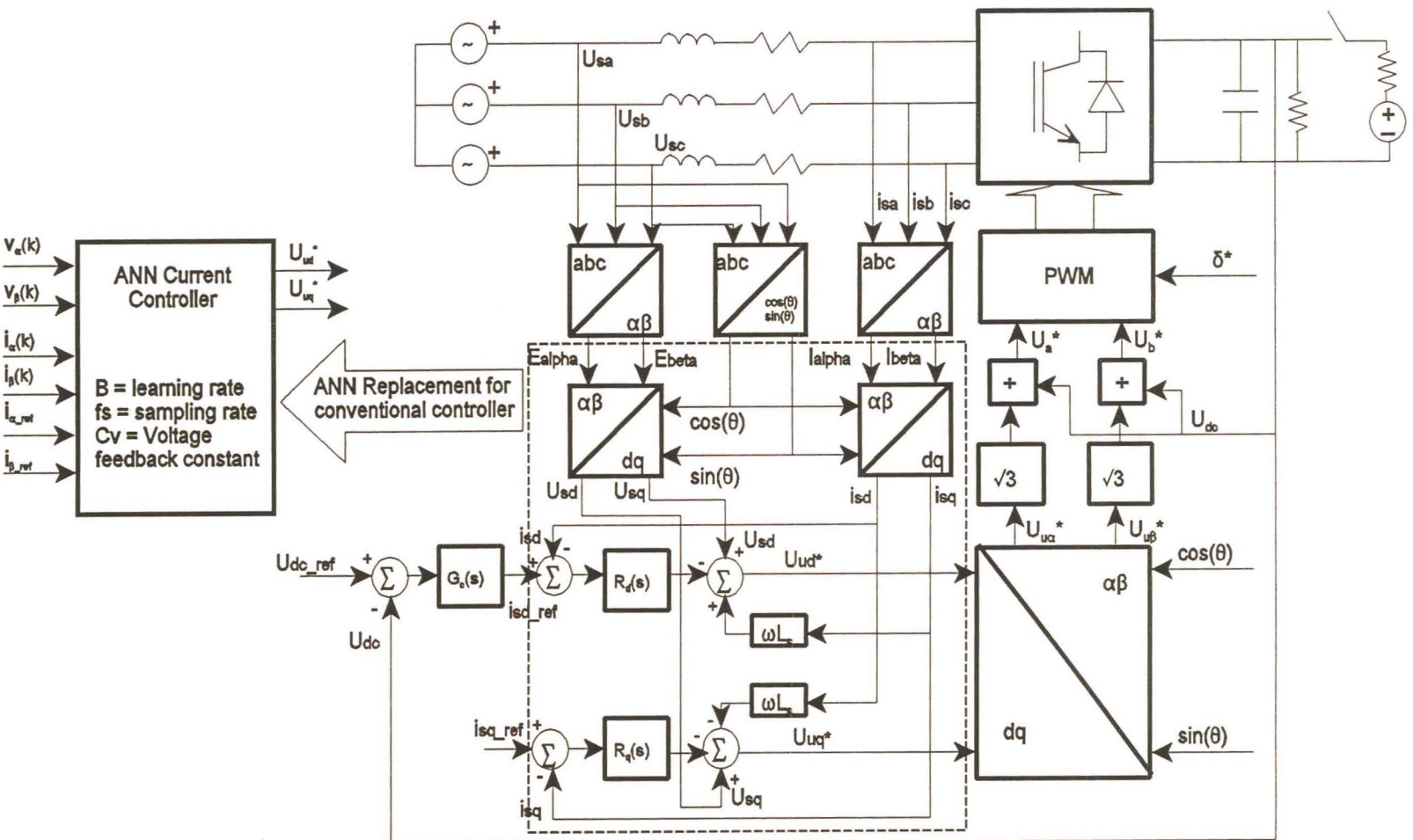


Fig. 4.6 Converting the conventional current controller to a COT ANN current controller

The Sine and Cosine components are then used to calculate the reference line currents, $I_{\alpha_{ref}}$ and $I_{\beta_{ref}}$ (see Fig. 4.6), in phase with the phase to neutral supply voltages. Chathury's controller structure uses these reference currents for current control, and to obtain synchronous operation, i.e. unity power factor.

$$I_{\alpha_{ref}} = I_{ds_{ref}} \cdot \cos(\theta) - I_{qs_{ref}} \cdot \sin(\theta) \quad (4.13)$$

$$I_{\beta_{ref}} = I_{ds_{ref}} \cdot \sin(\theta) + I_{qs_{ref}} \cdot \cos(\theta) \quad (4.14)$$

Where $I_{ds_{ref}}$ is the required current level, and

$I_{qs_{ref}}$ is set to zero, to achieve unity power factor operation.

In the next section, the design of the voltage control loop, developed by Chathury is discussed.

4.2.3 DC link Voltage Controller

The voltage controller is designed using the same assumptions as Chathury, namely in that the dynamics of the current control loops may be neglected. Under this assumption, the COT ANN current controller can be treated as the simple gain K shown in Fig. 4.7.

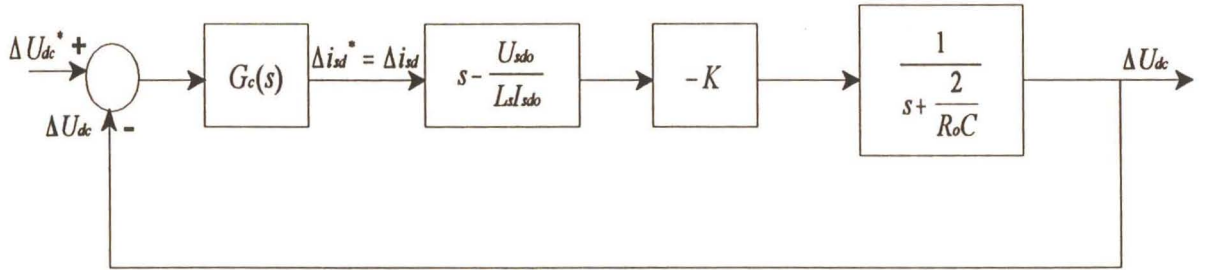


Fig. 4.7 Simplified DC voltage control block diagram

Chathury then performed a root locus analysis of the system to determine the stable range of operation. The root locus of the closed loop system (Fig. 4.8) is used by Chathury, to show that the uncontrolled boost rectifier system is naturally stable during the inverter mode of operation, but needs to be stabilised during the rectifier mode of operation [CHATHURY1-3]. Thus a

voltage controller is added to the system to ensure a stable closed loop response [CHATHURY1]. Chathury discovered that the addition of a standard PI regulator did not alleviate the instability problem, thus a controller of the form shown below is used by Chathury [CHATHURY1]:

$$G_c(s) = \frac{K_p(s + z)}{s(s + p)} \quad (4.15)$$

The regulator gain is calculated on the basis of second order transient response specifications according to d-axis current controllers closed loop transfer function shown in Eqn. (3.55) pp 3.27 [CHATHURY1] (provided as Eqn. (4.16) below for readers convenience), by placing the dominant closed loop poles and zeros at desired locations in the left half plane.

$$G(s) = \frac{\frac{K_{pi}}{L_s} \left(s + \frac{K_{ii}}{K_{pi}} \right)}{s^2 + \left(\frac{R_s}{L_s} + \frac{K_{pi}}{L_s} \right) s + \frac{K_{ii}}{L_s}} \quad (4.16)$$

The gain and value of p corresponding to the pole position are calculated, thereby yielding a stable closed loop system. From Chathury, the settling time for the dominant second order poles is set to ten times that assumed for the current controller, in order that the assumption of negligible current dynamics becomes valid. Hence a settling time of $t_s=225$ ms and a damping factor $\zeta=0.75$ (from Eqn(4.17), with a natural frequency of 23.7 rad/sec) ($M_0=2.7\%$) is obtained. The desired dominant closed loop poles are then calculated as $S_{1,2} = -\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2} = -17.78 \pm j15.67$.

$$\xi = 4 / \omega_n t_s \quad (4.17)$$

The open loop zero of the voltage regulator is arbitrarily positioned at $s=-200$ and the values of K_p and p are calculated using the angle and magnitude conditions [OGATA1] (described in APPENDIX A, pp A.18) as $K_p=0.173$ and $p=50$. Thus the final form of the DC voltage regulator is given as follows:

$$G_c(s) = \frac{0.173(s + 200)}{s(s + 50)} \quad (4.18)$$

Giving the following system root-locus:

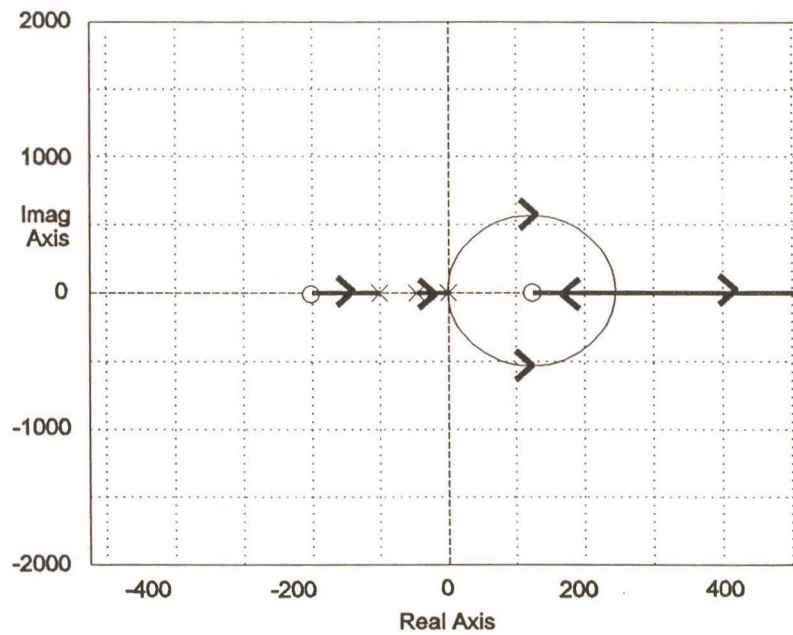


Fig. 4.8 Root locus of stabilised voltage control system

The controller, derived above, is used for all the simulations in this chapter.

In the next section the COT ANN current controlled boost rectifier system is simulated.

4.3 Investigation of a COT ANN Current Controller for a Boost Rectifier

The feasibility of using a COT ANN current controller for the boost rectifier is investigated in this section. As a first step, Burton's COT ANN, described in Chapter 3, is directly substituted for Chathury's current controller in the boost rectifier system described in Chapter 2, as shown in Fig. 4.9. These simulations, performed in CASE2, are used to determine whether Burton's COT ANN is applicable to both the boost rectifier and the SCIM. It should be noted here that the COT ANN parameters, inputs and outputs, are identical to those presented at the end of Chapter 3.

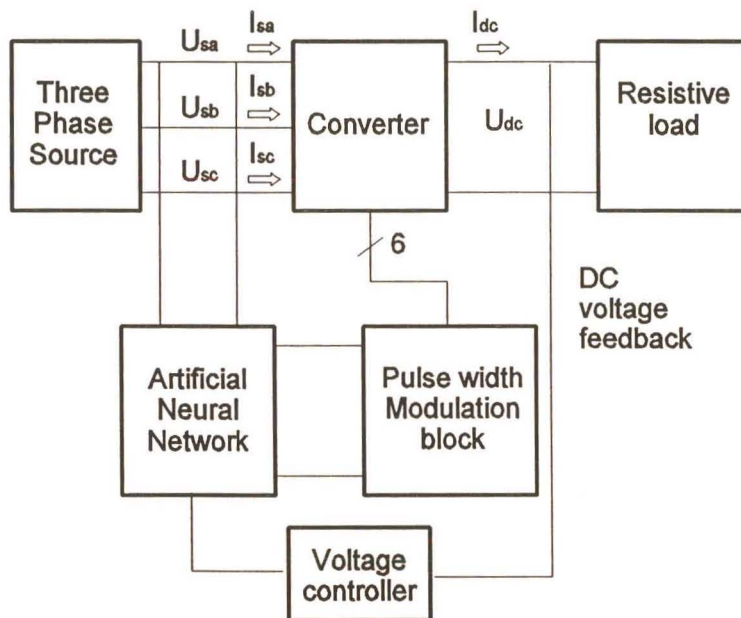


Fig. 4.9 ANN controlled Boost Rectifier block diagram

The model shown in Fig. 4.9 consists of five user-defined modules linked together in the CASE2 work space to form a closed loop system, namely:

- 1) a COT ANN stator current controller and voltage controller module developed by Burton [BURTON1] (code provided in APPENDIX B.7). Fig. 4.10 shows the flow diagram of the COT ANN current controller module source code. The following main steps are sequentially executed in the program:

- Step 1: All the simulation parameters and variables (e.g. sampling period), identification ANN parameters and variables (e.g. ANN size, gains, initial weights, inputs), induction motor parameters and variables (e.g. impedances, initial conditions etc.) are initialised.
- Step 2: The main loop starts when an enable or interrupt signal is received.
- Step 3: Steps 4 to 9 are omitted if the loop counter value is less than the corresponding start time T_{start} .
- Step 4: v_α and v_β are written to the predefined port memory address for processing by the PWM controller model.
- Step 5: The digital values of the A B and C phase source voltages and currents are then read in from the predefined dual port memory addresses and converted to floating point numbers.
- Step 6: The voltage PI loop output is calculated.
- Step 7: The ANN input $\underline{x}(k+1)=\underline{\hat{x}}(k)$ is formed.
- Step 8: $\underline{x}(k+1)$ is forward propagated through the ANN to obtain $\underline{y}(k)=\underline{\hat{f}}_{el}^{(k-1)}(.)$.
- Step 9: $\underline{i}^*(k+1)$ is calculated, either with or without the use of the reference model [BURTON1], and used with $\underline{\hat{f}}_{el}^{(k-1)}(.)$ and constant c_v to calculate $\underline{v}^*(k)$ using Eq. (2.11) from Chapter 2.
- Step 10: The ANN output error $e_y(k) = \underline{i}(k) - \underline{i}^*(k)$ is calculated and backpropagated through the ANN to calculate $W(k+1)$ and $V(k+1)$, which represents $\underline{\hat{f}}_{el}^{(k-1)}(.)$, for the next sampling period.
- Step 11: Step 11 is omitted if the specified loop count corresponding with T_{end} has been reached.
- Step 12: The main loop counter is updated and the program returns to step 2.

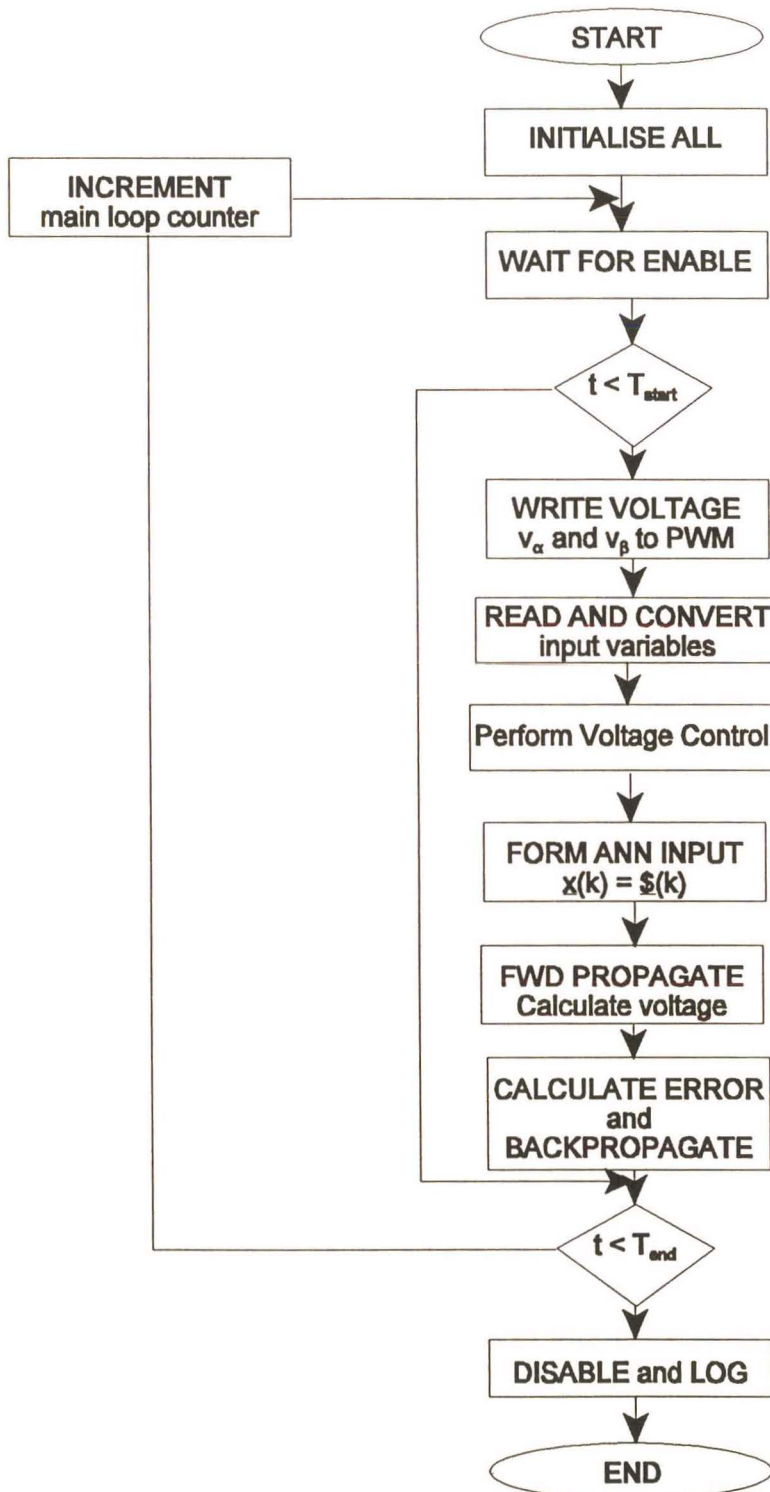


Fig. 4.10 Flow diagram of COT ANN CASED module for boost rectifier with voltage controller.

- 2) a PWM controller module developed by Chathury [CHATHURY1] (code provided in APPENDIX B.4). Fig. 4.11 shows the flow diagram of the PWM control module source code. The following main steps are sequentially executed in the program:
 - Step 1: The module waits for a start signal.
 - Step 2: The module reads in the v_α and v_β values from the predefined port memory address, where they were saved by the COT ANN controller.
 - Step 3: The module calculates modulating signals, by simulating the Hanning PBM 1/87 ASIC's [HANNING1] operation (this ASIC is a PWM generator). This ASIC is used by Burton [BURTON1-3] and Chathury [CHATHURY1-3] in their respective researches.
 - Step 4: The switching times are calculated.
 - Step 5: The module then generates CAsED switching events, which are written to a predefined port address to control the power electronic inverter.

- 3) a user-defined three-phase source module (code provided in APPENDIX B.6). Fig. 4.12 shows the flow diagram of the three-phase module source code. The following main steps are sequentially executed in the program:
 - Step 1: The module waits for a start signal.
 - Step 2: The simulation parameters and inputs are set.
 - Step 3: The A, B and C phase to neutral currents are calculated.
 - Step 4: The A, B and C line voltages are calculated.
 - Step 5: The voltage values are written to predefined port addresses.

- 4) a general converter module (CAsED library model).

- 5) a resistive load module (CAsED library model).

Fig. 4.13 presents an example of the simulation results obtained under these conditions. It is seen that the COT ANN current controller becomes divergent, which suggests that the COT ANN current controller structure must be changed to achieve convergent operation with the boost rectifier.

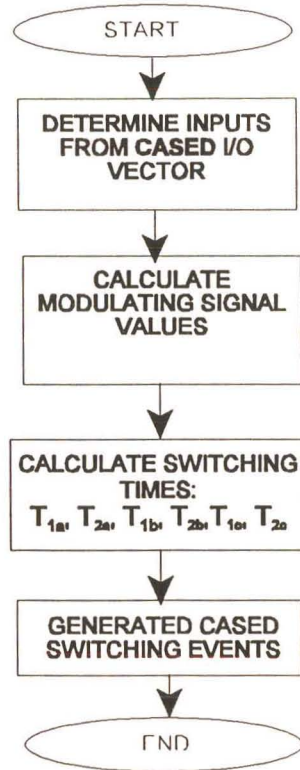


Fig. 4.11 Flow diagram of PWM CASED module

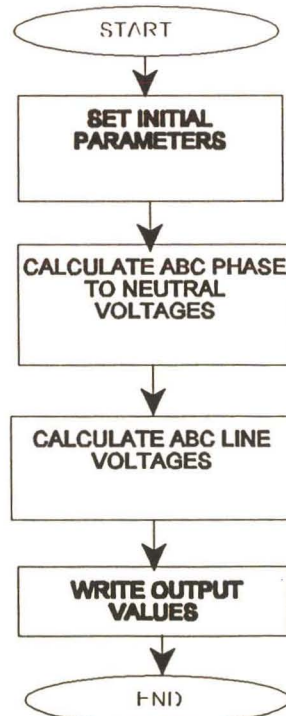


Fig. 4.12 Three-phase source flow diagram

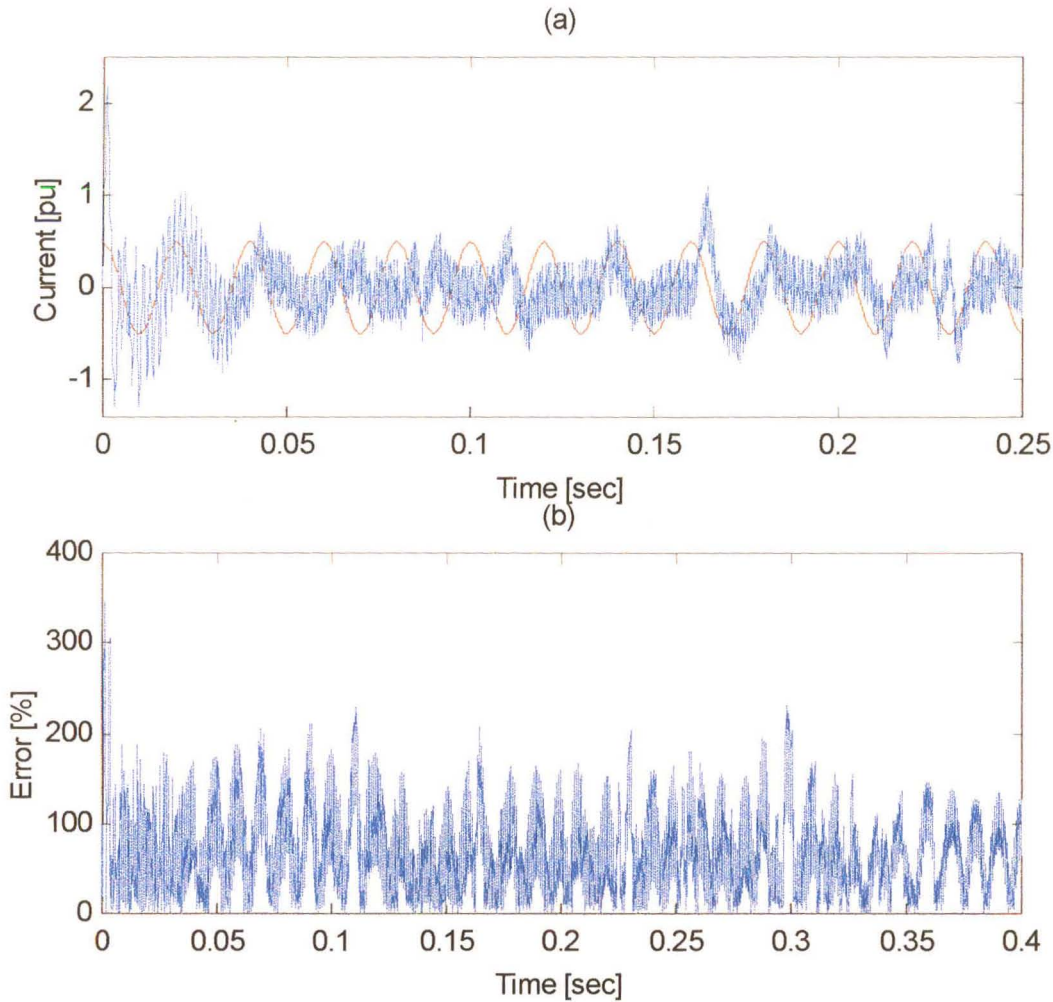


Fig. 4.13(a) Phase A supply current (I_{sa}) and COT ANN alpha reference current (I_{α_ref}) (internal reference variable), (b) Current tracking error

The COT ANN current controller's inputs and outputs are then changed to comply with the Narmax model of the boost rectifier, as described in section 4.2.1. The restructured COT ANN current controller is then used with the boost rectifier for further evaluation (with the same internal variable and constant settings used in Chapter 3), the DC link capacitor is pre-charged to its nominal voltage as it would be in a normal drive. It should be noted that the results presented in this section, are obtained using a simple proportional controller as the DC voltage control loop (as shown in Fig. 4.9), for the motoring mode of the boost rectifier only, unless otherwise stated.

Fig. 4.14 presents an example of divergent operation of the COT ANN current controller, obtained while determining the incoming line impedance range for convergent controller operation. In this example, the system line currents become divergent as the incoming line impedance becomes too large (50mH). In this case the COT ANN controller demands large currents in order to compensate for the line impedance voltage drop. Unfortunately the boost rectifier cannot source these large currents, dropping the DC link voltage to zero causing the COT ANN controller to become divergent. Fig. 4.15 shows the corresponding three phase line currents, for the above example. Here it can be seen that all three line currents become divergent after approximately 100 milliseconds.

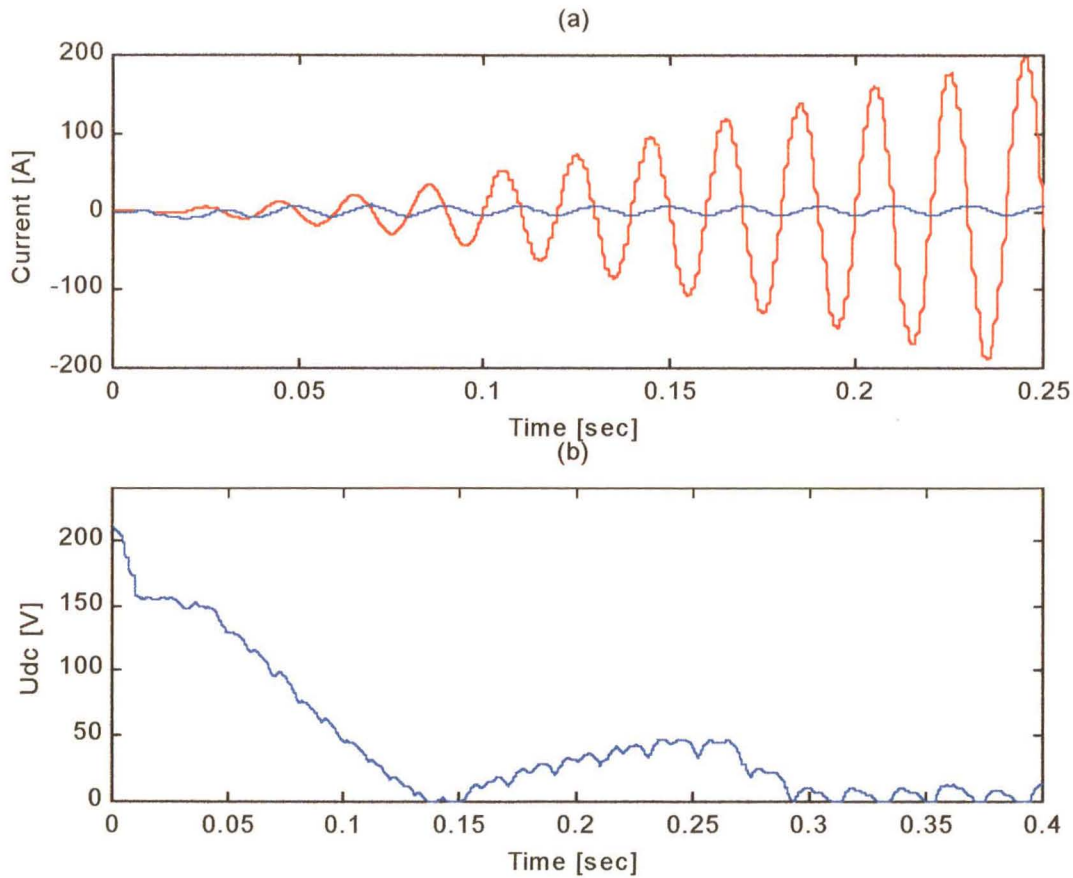


Fig. 4.14(a) Phase A supply current (I_{sa}) and COT ANN alpha reference current (I_{α_ref}),
(b) DC link Voltage

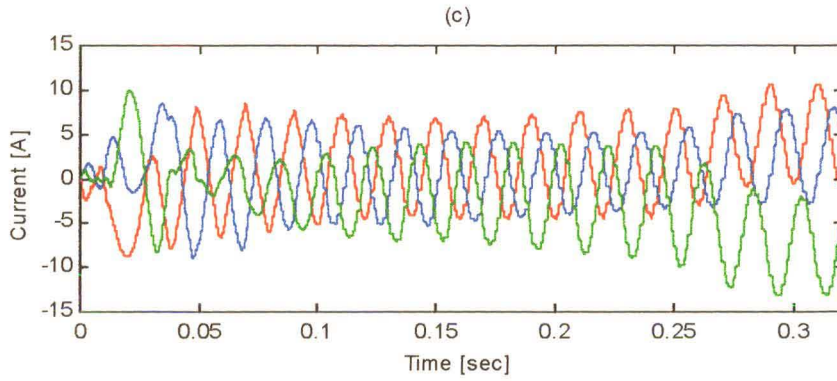


Fig 4.15 A (I_{sa}), B (I_{sb}), and C (I_{sc}) phase currents

Successive tests are then performed, in the same manner as described in Chapter 3, in order to determine the COT ANNs dependencies on its internal constant and variable set points. These dependencies are then used to tune the COT ANN current controller to an operating point that achieves efficient, convergent system control. It should be noted that the desired COT ANN current controller operating point is chosen relative to the response obtained by Chathury's [CHATHURY1-3] research. Furthermore, this section investigates whether the structural size of the COT ANN can be reduced, minimizing the computational power required by the COT ANN current controller.

The following variables are investigated during the tests, to determine the desired operating point for the ANN (it should be noted that the momentum rate is omitted from this investigation):

- 1) ANN learning rate (B),
- 2) ANN voltage constant (C_v),
- 3) PWM switching frequency (f_{han}),
- 4) ANN sampling rate (f_s),
- 5) α/β current and voltage feedback terms, and
- 6) The DC link voltage controller proportional gain (K_p)

The subsequent subsections analyse the simulation results, and show the method used to determine the desired operating point for the ANN current controller.

4.3.1 ANN Learning Rate

Successive tests are performed using a variable learning rate, while fixing the other variables, to determine the effect of the learning rate on the performance of the COT ANN current controller.

The first series of results shown in Fig. 4.16 are taken with a low learning rate (B), i.e. the ANN is allowed a long time to train, and to track the desired response.

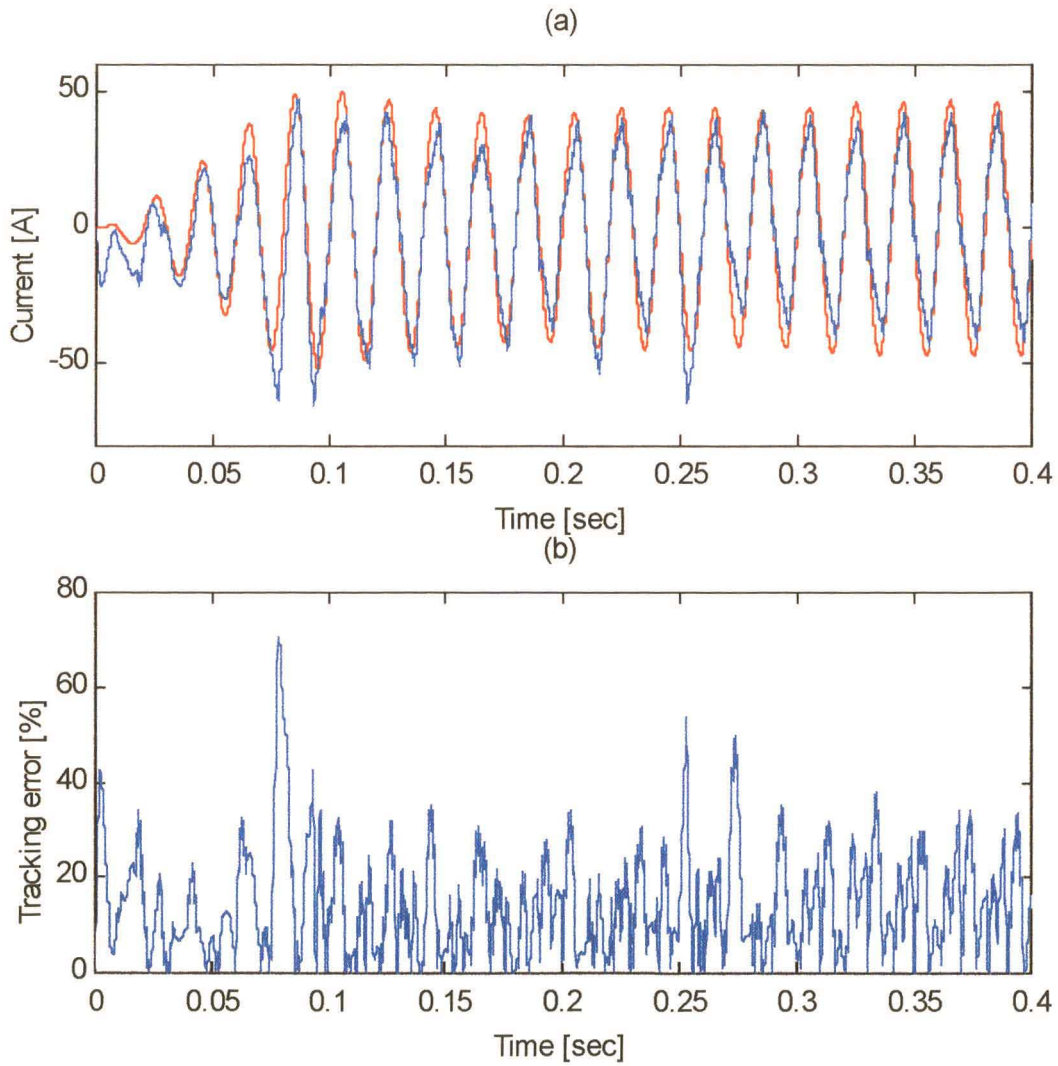


Fig. 4.16 (a) Phase A supply current (I_{sa}) and COT ANN alpha reference current (I_{α_ref}), with $C_v = 2.0$ pu, $f_s = 4$ kHz, $f_{han} = 4$ kHz, and Learning Rate of 0.01 pu, (b) Current tracking error

The second series of results shown in Fig 4.17 are obtained with a higher learning rate (0.1).

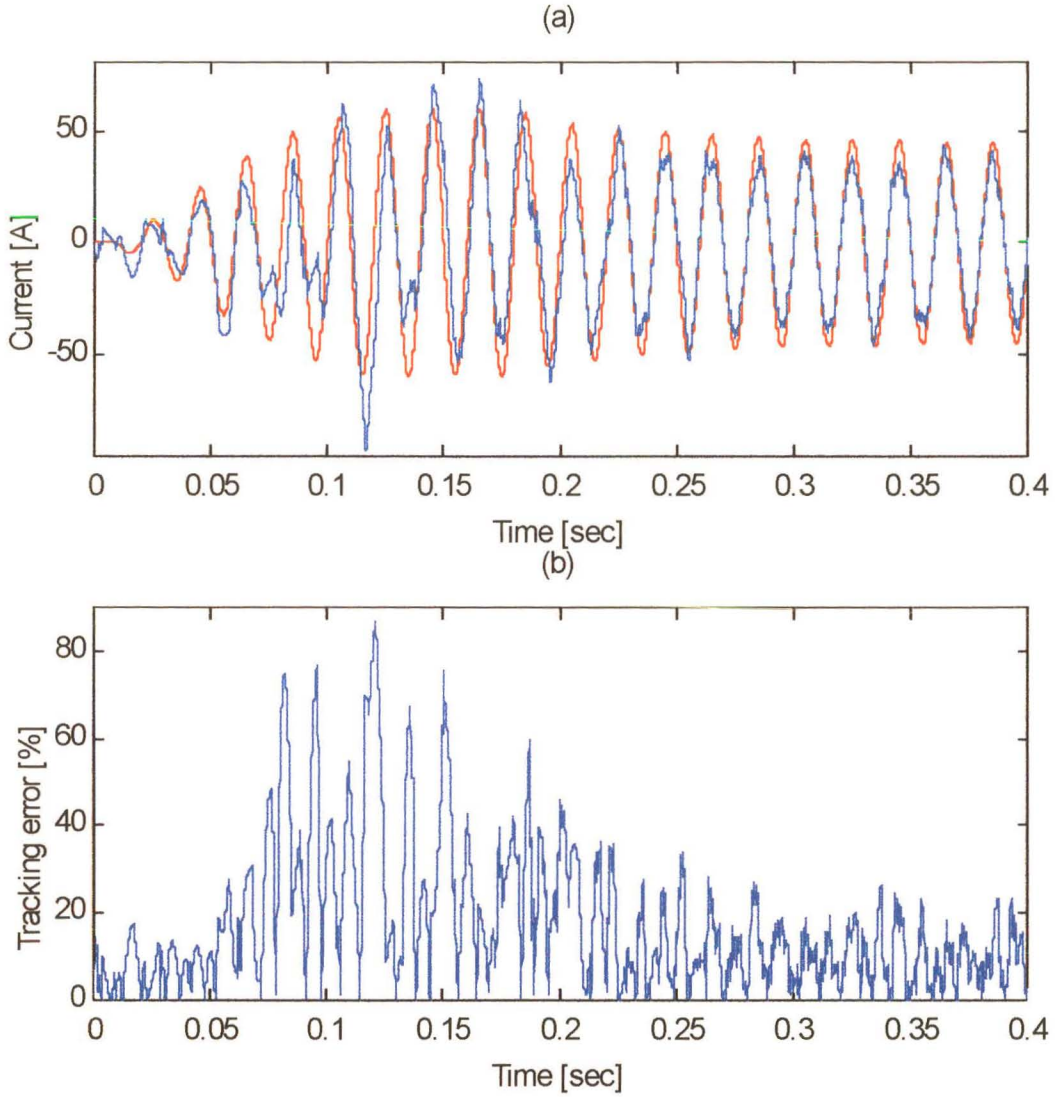


Fig. 4.17 (a) Phase A supply current (I_{sa}) and COT ANN alpha reference current (I_{α_ref}), with $C_v = 2.0$ pu, $f_s = 4$ kHz, $f_{han} = 4$ kHz, and Learning Rate of 0.1 pu, (b) Current tracking error

The results show that the ANN has an improved tracking capability (lower tracking error) at high learning rates, thus the ANN will be operated with a high learning rate ($B = 0.05 \Rightarrow B = 0.1$), to achieve the desired system response.

4.3.2 ANN Voltage Constant

Successive tests are performed with a variable voltage constant, C_v , while fixing the other variables, to determine the effect of the voltage constant on the performance of the ANN current controller. The first series of results shown in Fig. 4.18 are taken with a large value of C_v (10 pu).

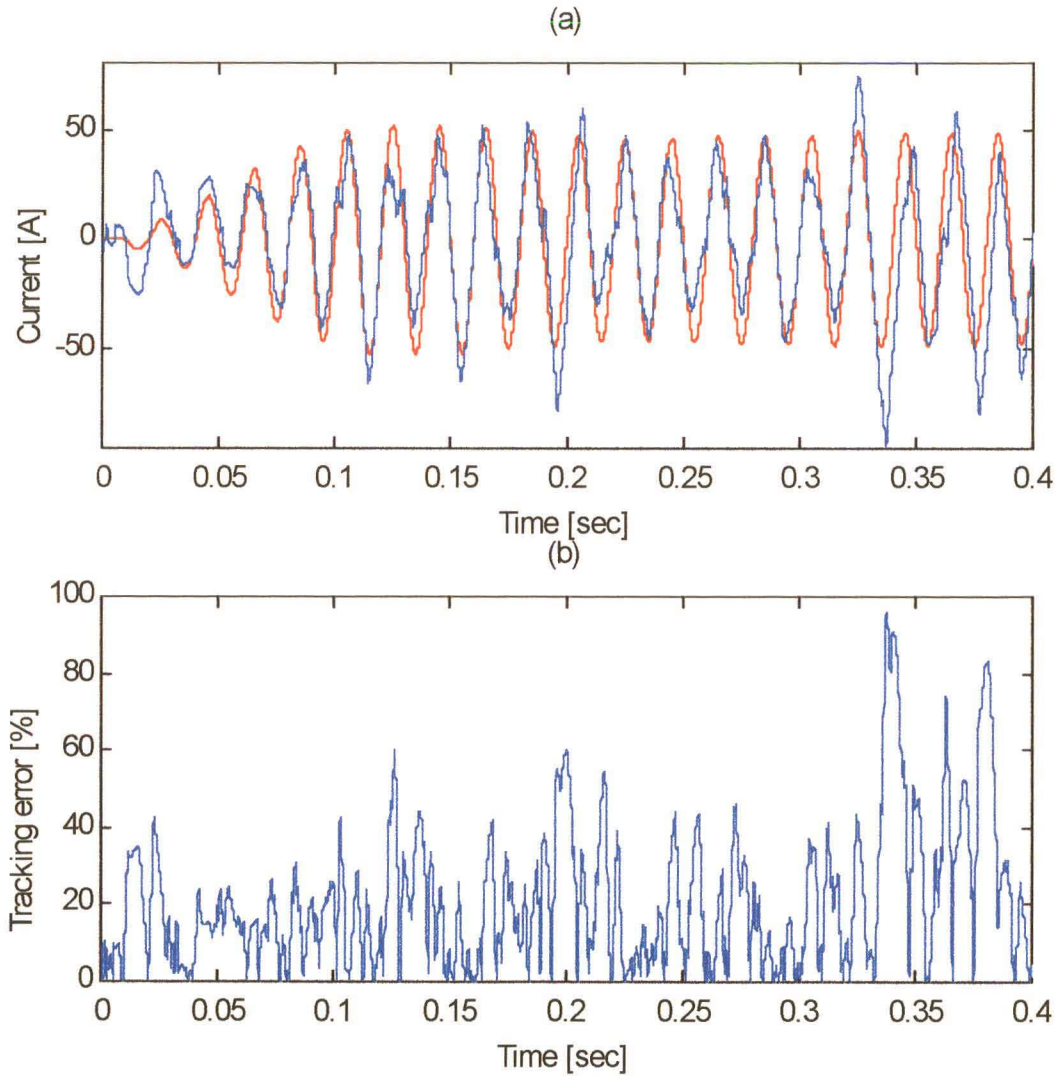


Fig. 4.18 (a) Phase A supply current (I_{sa}) and COT ANN alpha reference current (I_{α_ref}), with $C_v = 10.0$ pu, $f_s = 4$ kHz, $f_{han} = 4$ kHz, and Learning Rate of 0.05 pu, (b) Current tracking error

The second series of results shown in Fig 4.19 are obtained with a small voltage constant (1.0).
(a)

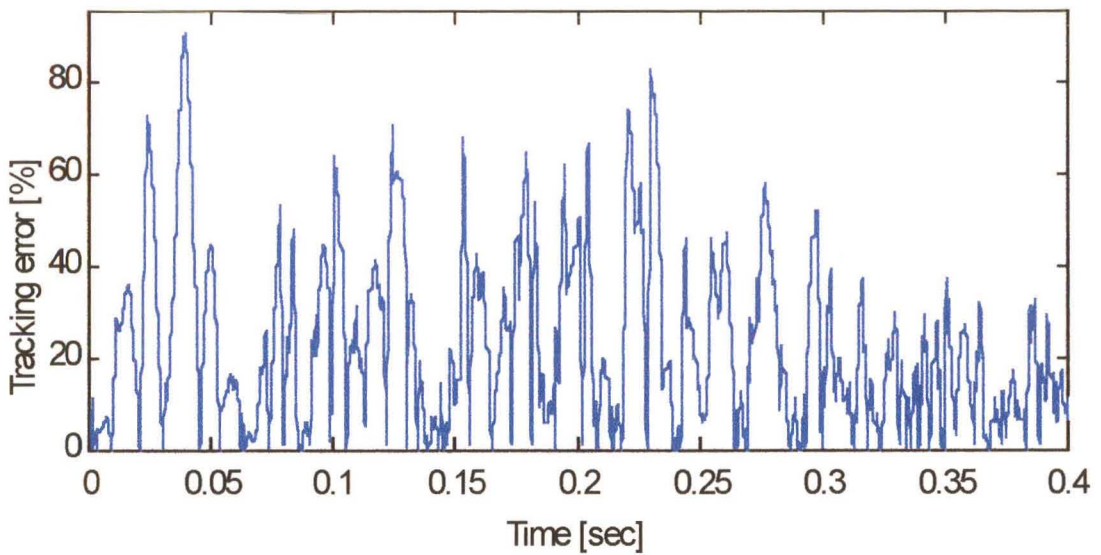
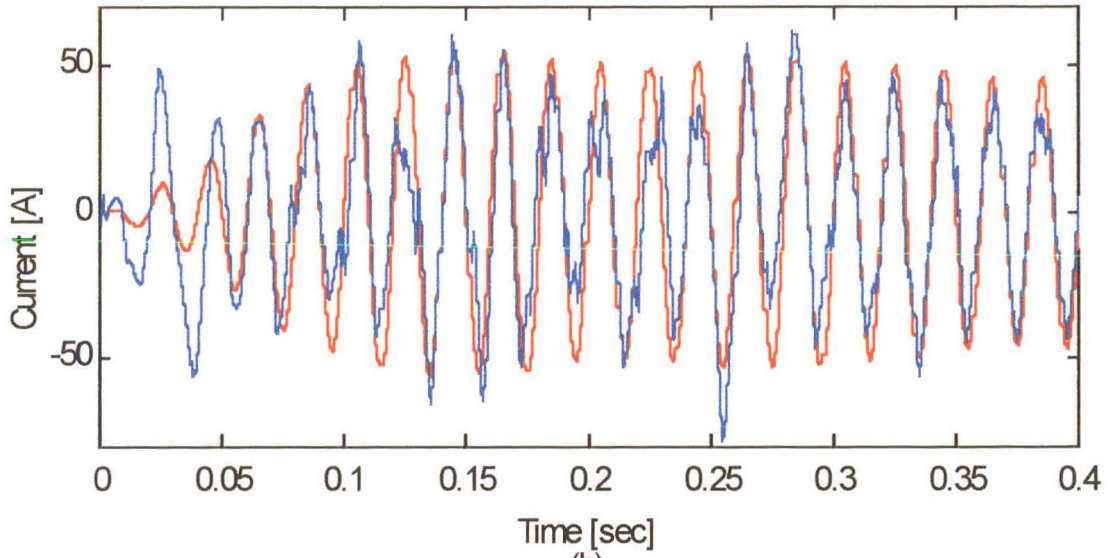


Fig. 4.19 (a) Phase A supply current (I_{sa}) and COT ANN alpha reference current (I_{α_ref}), with $C_v = 1.0$ pu, $f_s = 4$ kHz, $f_{han} = 4$ kHz, and Learning rate of 0.05 pu,
(b) Current tracking error

From these results and many other trials, for different values of C_v , it is concluded that the voltage constant should be between 0.8 and 1.2, to achieve the desired transient response.

The final set of tests, shown in Fig. 4.20, shows that the desired operating point (in comparison to Chathury's system response) was with a learning rate between 0.1 and 0.15, and a voltage constant between 1.0 and 1.1. At this point, the controller is achieving current convergence after 150 milliseconds, with a low initial overshoot (less than 40 %), as shown in Fig. 4.20 above.

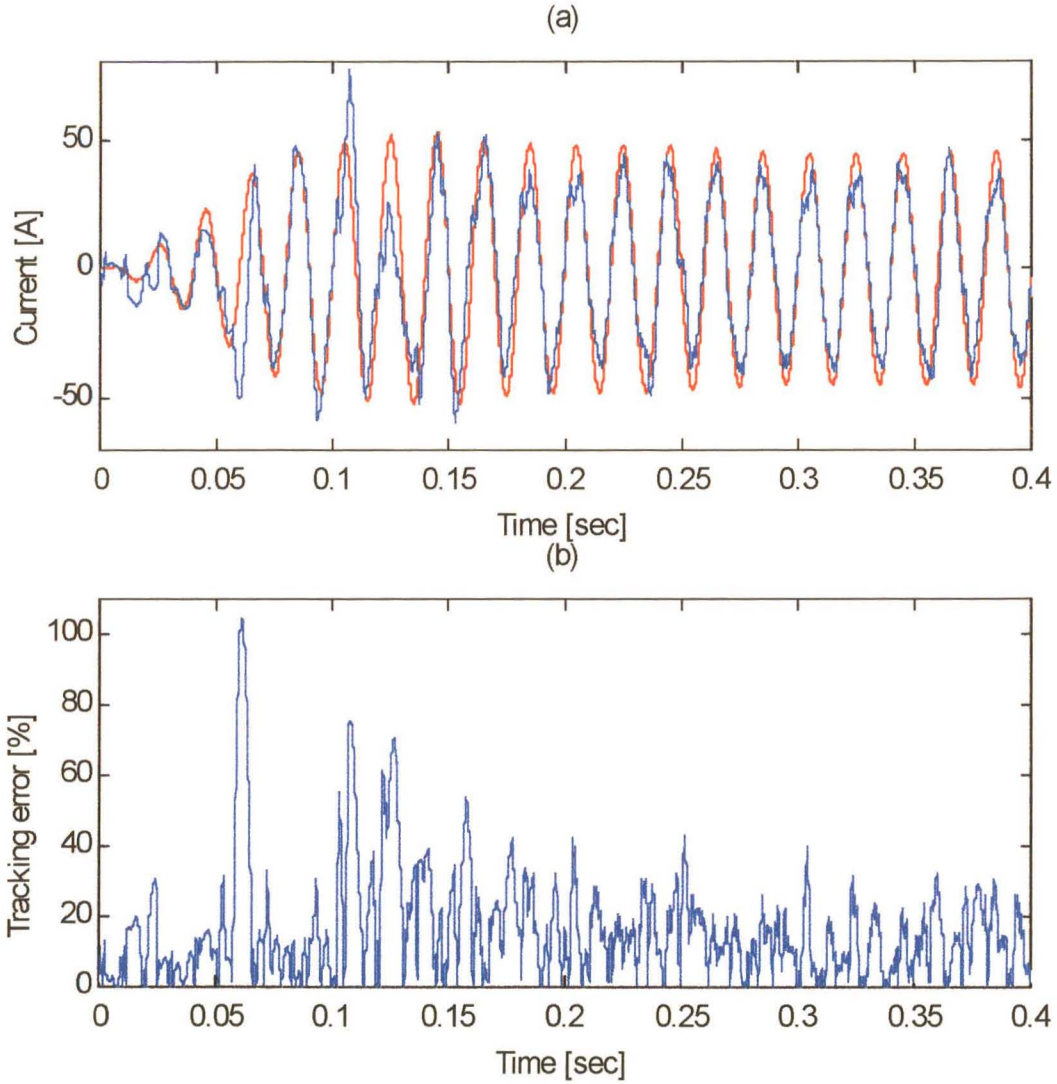


Fig. 4.20 (a) Phase A supply current (I_{sa}) and COT ANN alpha reference current (I_{α_ref}), with $C_v = 1.1$ pu, $f_s = 4$ kHz, $f_{han} = 4$ kHz, and Learning rate of 0.12 up, (b) Current tracking error

4.3.3 PWM Switching Frequency (f_{han})

While performing the simulations, the PWM switching frequency (f_{han}) is maintained equal to the ANN sampling frequency, as a default setting. An investigation of the PWM switching frequency, shows the system is convergent with a PWM switching frequency greater than the ANN sampling frequency (Fig. 4.21), but is divergent with a PWM switching frequency lower than the sampling frequency, thus it is suggested that the PWM switching frequency be set equal to the ANN sampling frequency to achieve the best transient response, with a low initial overshoot (less than 20 %).

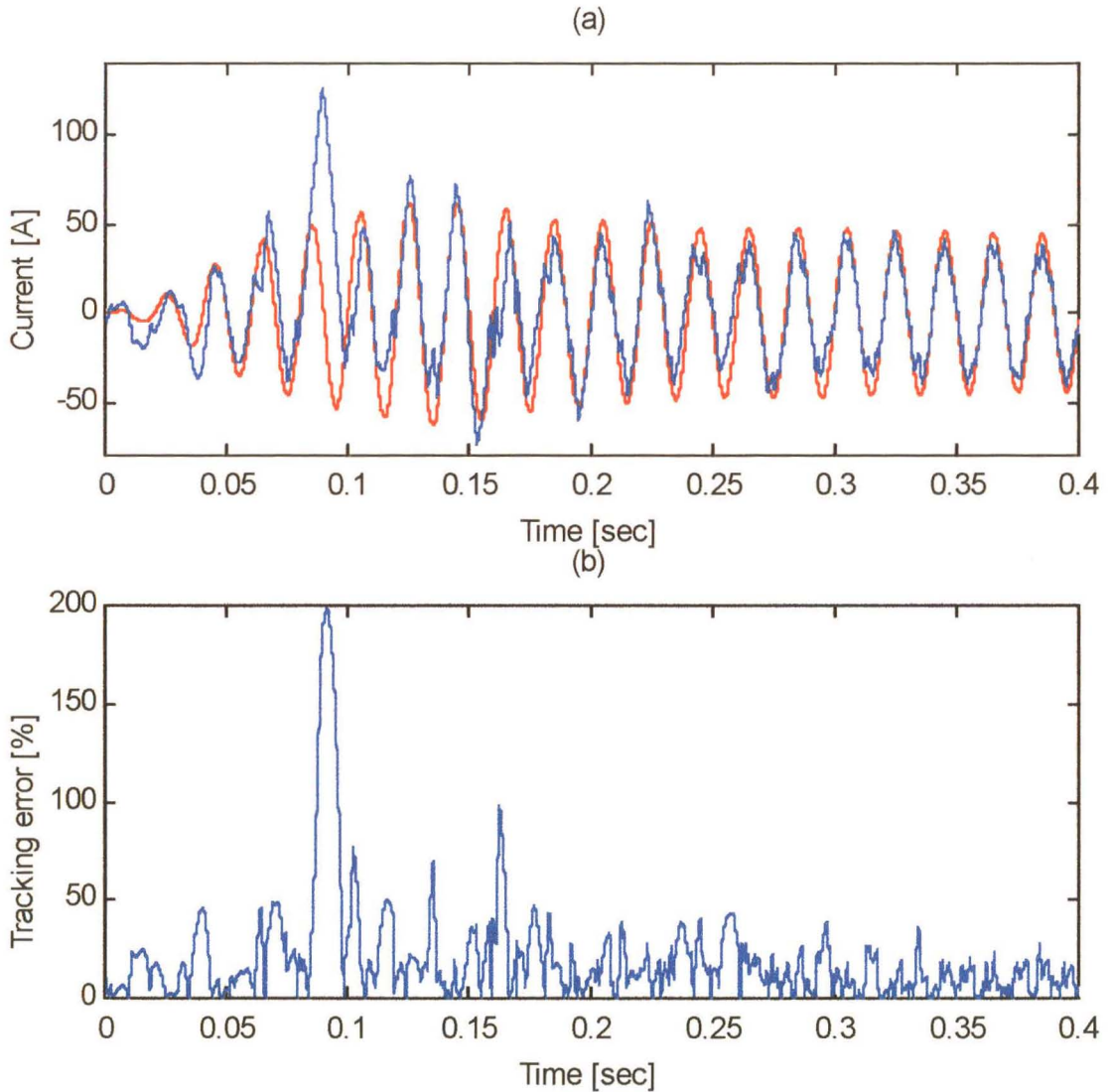


Fig. 4.21 (a) Phase A supply current (I_{sa}) and COT ANN alpha reference current ($I_{\alpha_{ref}}$), with $C_v = 1.1$ pu, $f_s = 4$ kHz, $f_{\text{han}} = 8$ kHz, and Learning rate of 0.12 pu, (b) Current tracking error

4.3.4 ANN Sampling Frequency (f_s)

The ANN sampling frequency (f_s) is varied, while fixing the other variables, to determine the ANN's most efficient point of operation. Fig. 4.22 shows results taken with a medium sampling rate (10 kHz).

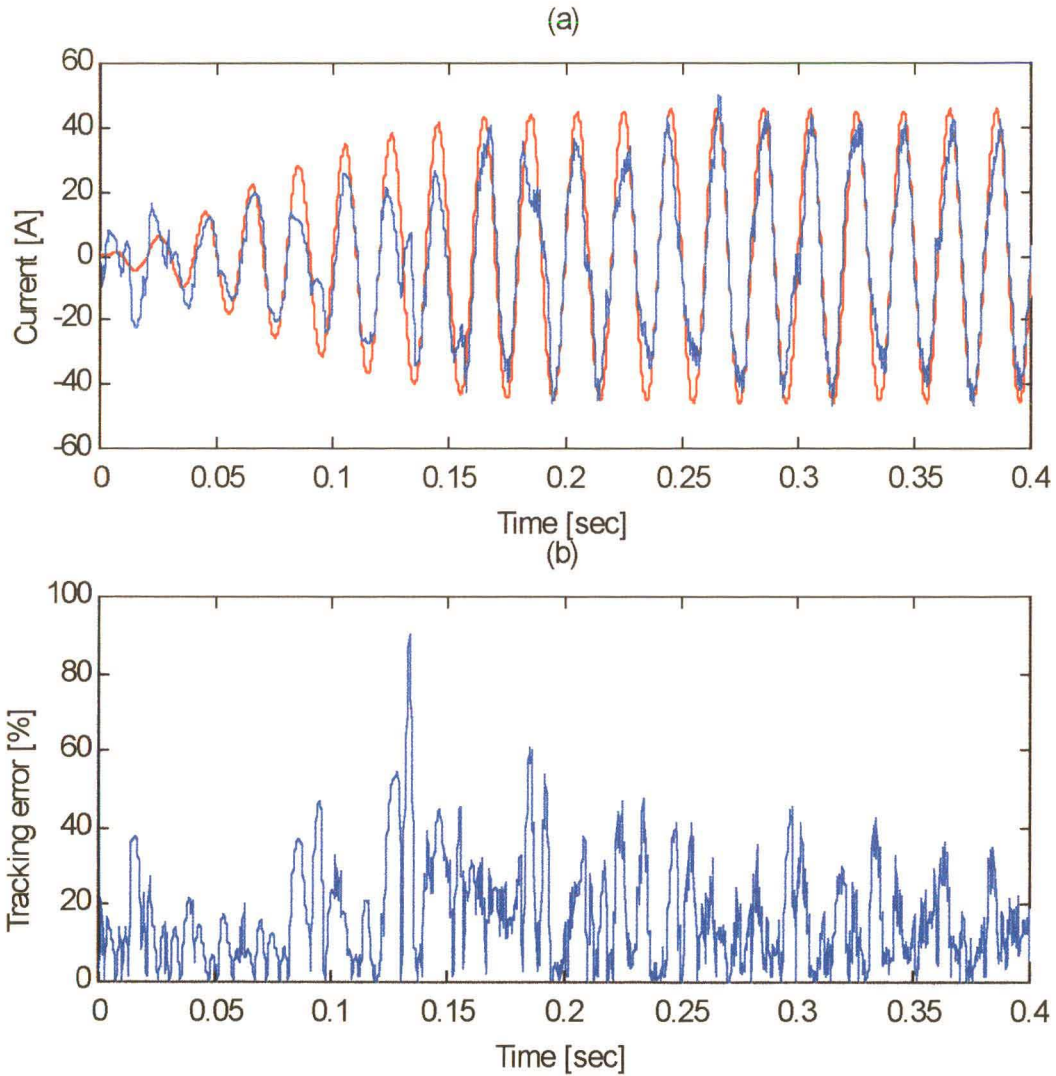


Fig. 4.22 (a) Phase A supply current (I_{sa}) and COT ANN alpha reference current (I_{α_ref}), with $C_v = 1.1$ pu, $f_s = 10$ kHz, $f_{han} = 10$ kHz, and Learning Rate of 0.12 pu, (b) Current tracking error

The second set of results (Fig. 4.23) illustrates the ANN's response at a sampling frequency of 15 kHz. In Fig. 4.23, the COT ANN current controller cannot achieve convergence between the phase A source current (I_{sa}) and ANN reference current (I_{α_ref}), hence the reference current increases as the voltage controller error increases in magnitude (system becomes unstable).

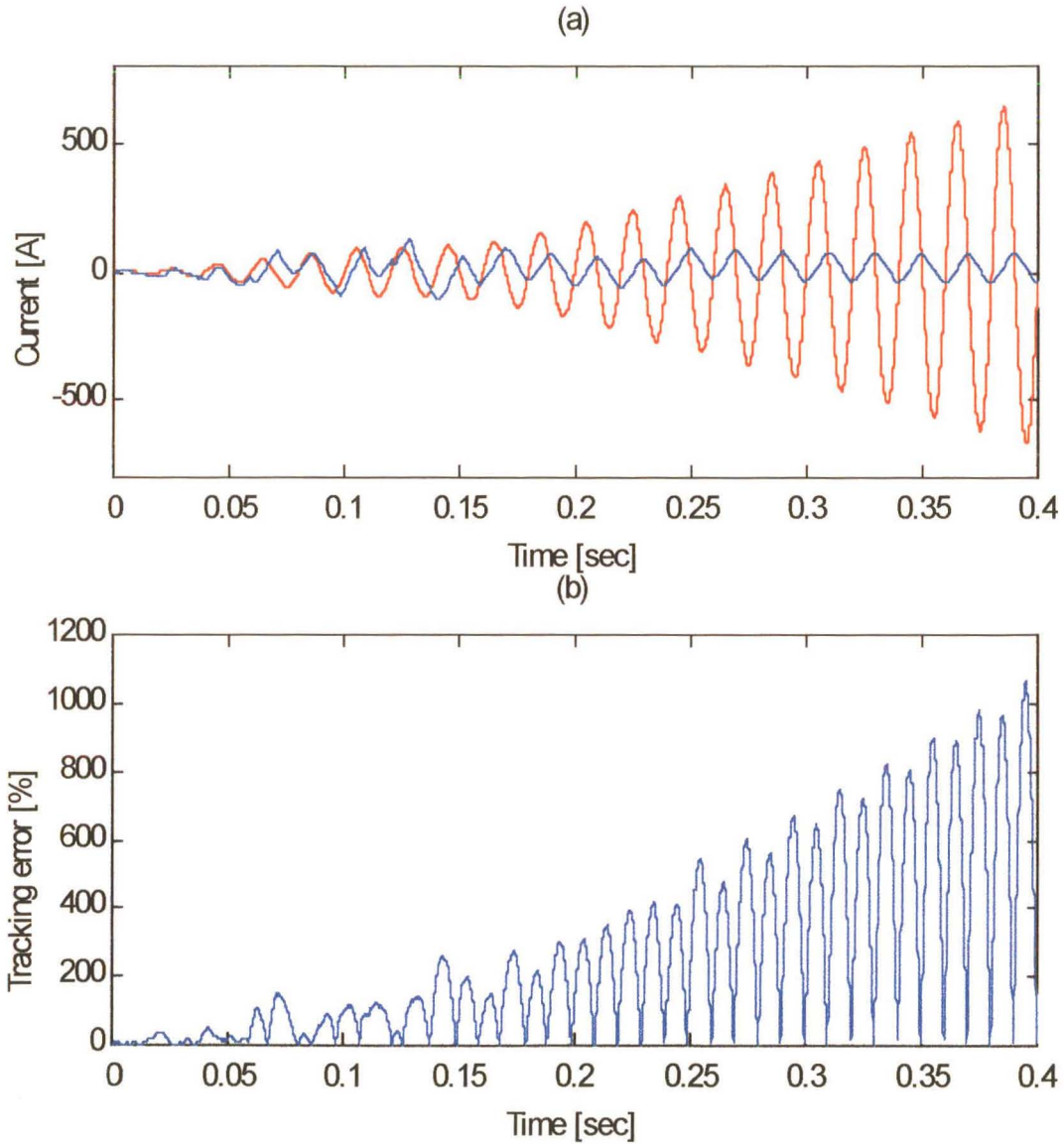


Fig. 4.23 (a) Phase A supply current (I_{sa}) and COT ANN alpha reference current (I_{α_ref}), with $C_v = 1.1$ pu, $f_s = 15$ kHz, $f_{han} = 15$ kHz, and Learning rate of 0.12 pu, (b) Current tracking error

Fig. 4.24 shows that the overall current tracking error is reduced at a sampling frequency of 4 kHz, it can be seen that the best transient tracking response is achieved with a sampling frequency of between 4 and 10 kHz.

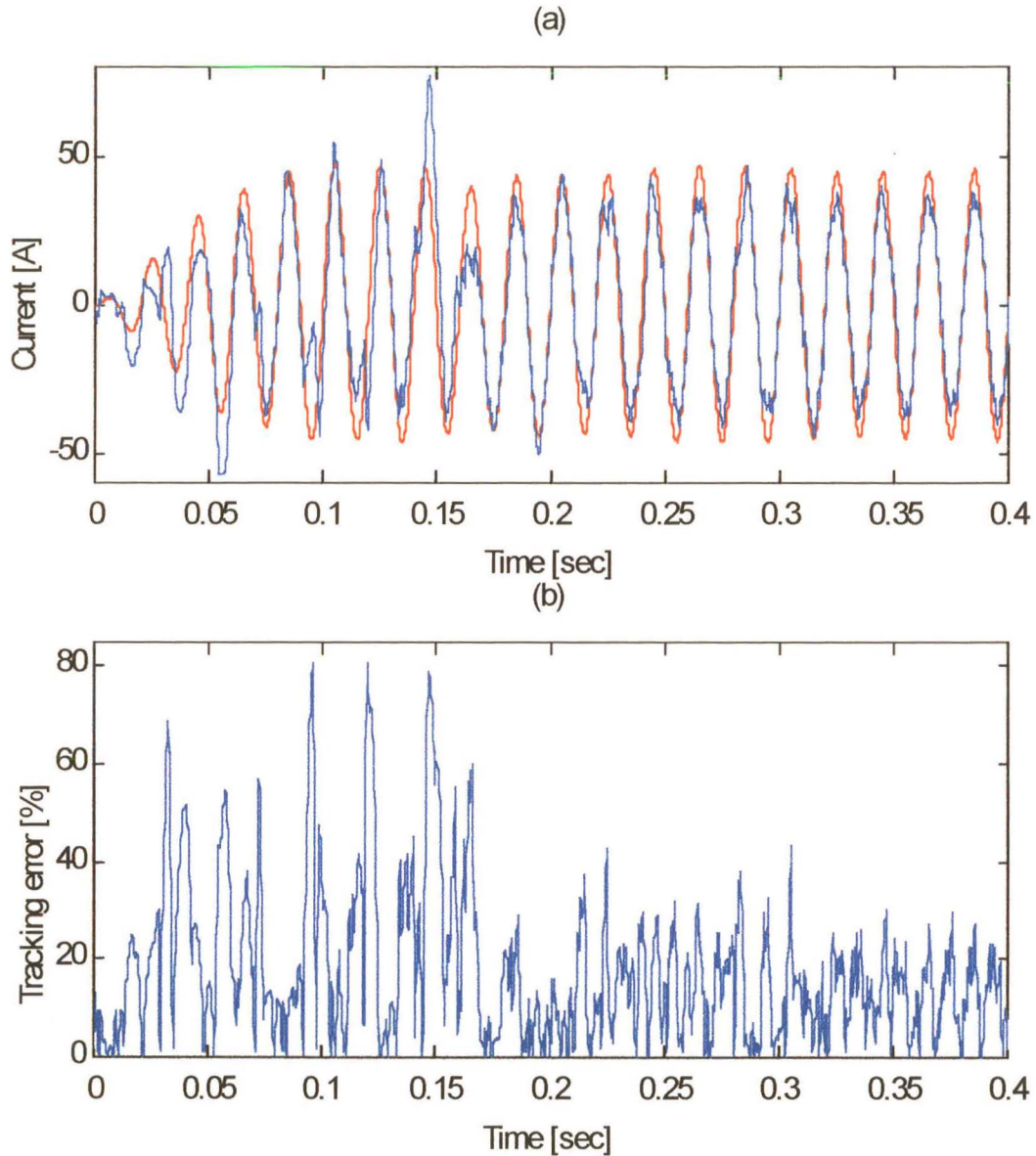


Fig. 4.24 (a) Phase A supply current (I_{sa}) and COT ANN alpha reference current (I_{α_ref}), with $C_v = 1.1$ pu, $f_s = 4$ kHz, $f_{han} = 2$ kHz, and Learning rate of 0.12 pu, (b) Current tracking error

4.3.5 Alpha/Beta Current and Voltage Feedback Terms

The number of inputs are reduced by omitting the delayed alpha/beta current and voltage terms (the $(k-1)$ terms, shown in Fig. 4.5) from the ANN input, thus simplifying the ANN structure, reducing its computation cycle. The results shown in Fig 4.25 show that the ANN's tracking capability is reduced under these circumstances, due to the reduction in available training information to the ANN.

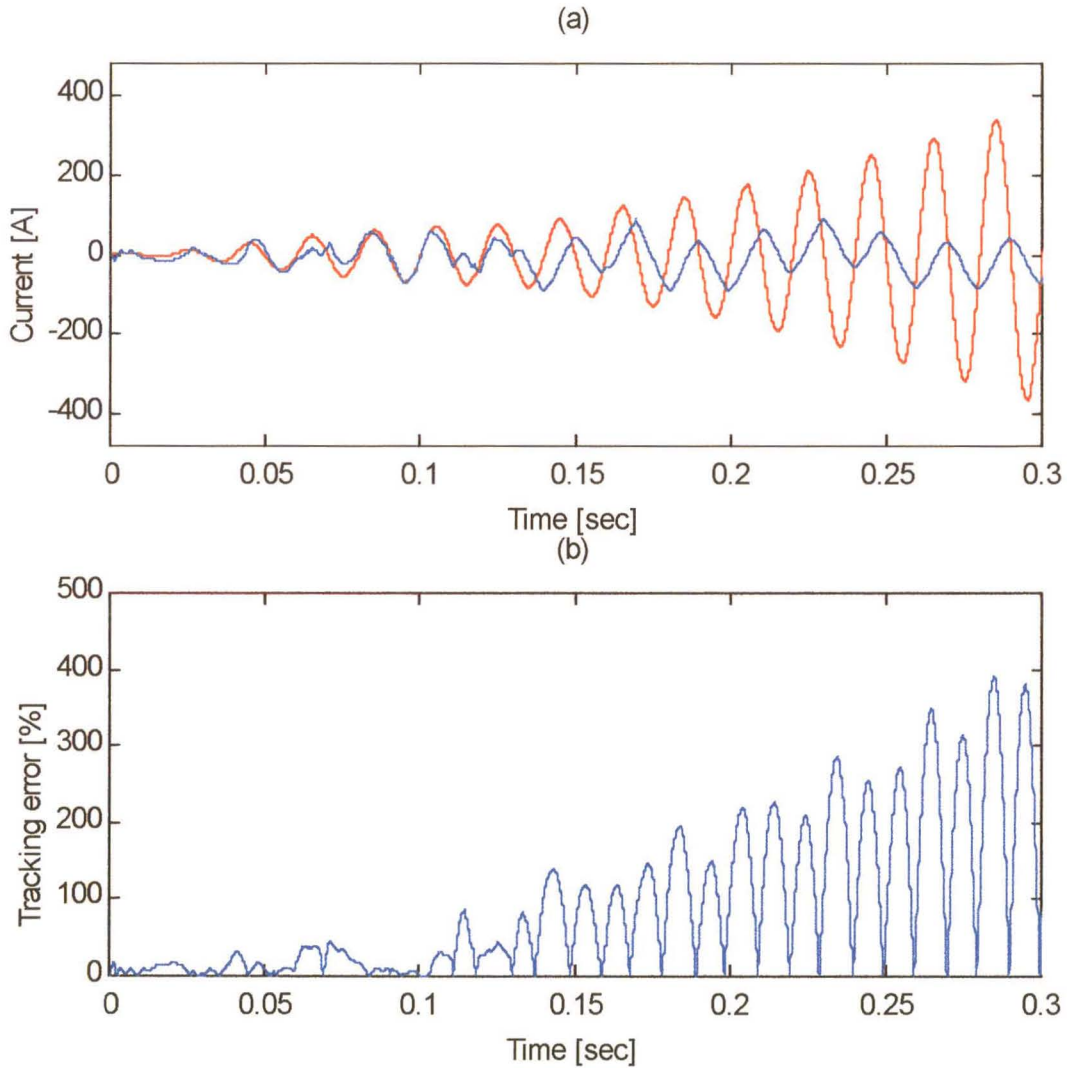


Fig. 4.25 (a) Phase A supply current (I_{sa}) and COT ANN alpha reference current (I_{α_ref}), with $C_v = 1.1$ pu, $f_s = 4$ kHz, $f_{han} = 2$ kHz, and Learning rate of 0.12 pu, no $I(k-1)$ terms, (b) Current tracking error

Further simulations are performed with the delayed current terms in place, while omitting the delayed voltage terms, thus allowing the ANN additional information, in order to train. Fig. 4.26 depicts the system response under these conditions.

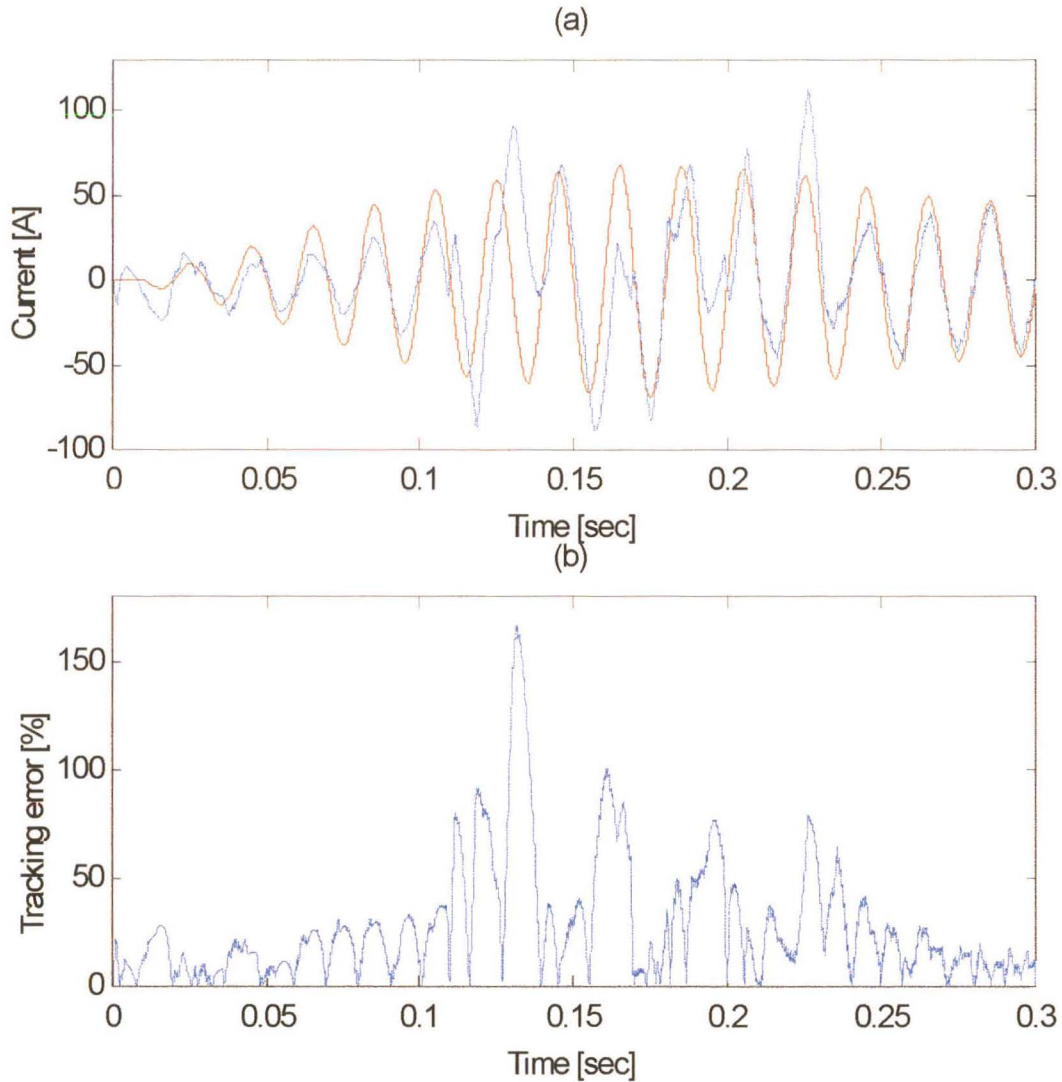


Fig. 4.26 (a) Phase A supply current (I_{sa}) and COT ANN alpha reference current (I_{α_ref}), with $C_v = 0.1$ pu, $f_s = 4$ kHz, $f_{han} = 2$ kHz, and Learning Rate of 0.12 pu, no $I(k-1)$ terms, (b) Current tracking error

Thus the COT ANN may be operated while omitting the Alpha\Beta delayed voltage terms, which means that the ANN's structure may be simplified, by reducing the number of input and hidden nodes, thereby reducing the ANN's calculation cycle.

4.3.6 Proportional Gain K_p

The aim of this section is to tune the voltage controller, to achieve a regulated DC link voltage. Successive tests are performed to obtain the desired proportional gain for efficient and stable voltage regulation. Sample plots are shown below to demonstrate the effect of the proportional gain on the voltage regulation. Figs. 4.27, and 4.28 are taken for a low proportional gain ($K_p=0.01$), while regulating the DC link to 212 volts.

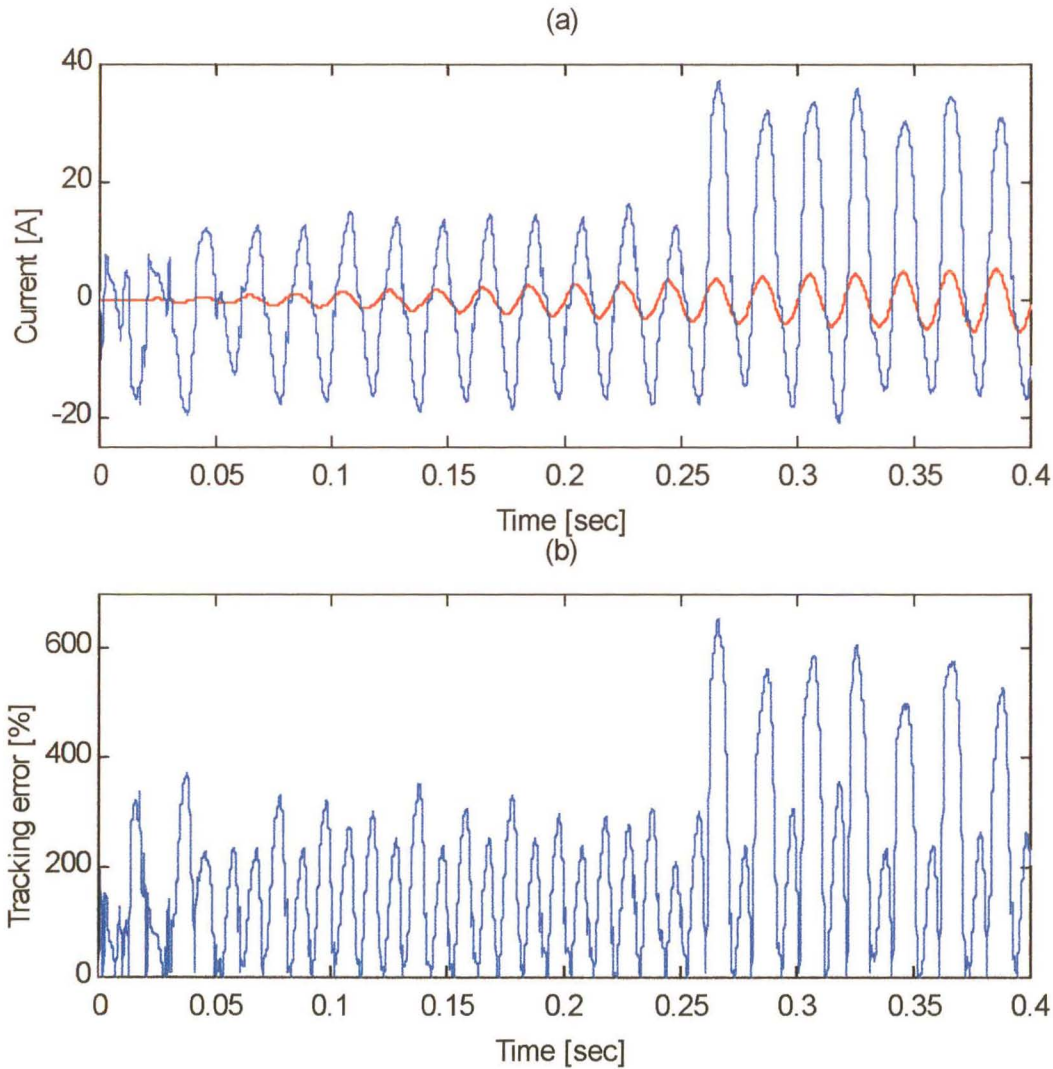


Fig. 4.27 (a) Phase A supply current (I_{sa}) and COT ANN alpha reference current ($I_{\alpha_{ref}}$), with $C_v = 1.1$ pu, $f_s = 4$ kHz, $f_{han} = 4$ kHz, and $B = 0.12$ pu, $K_p = 0.01$ pu, (b) Current tracking error

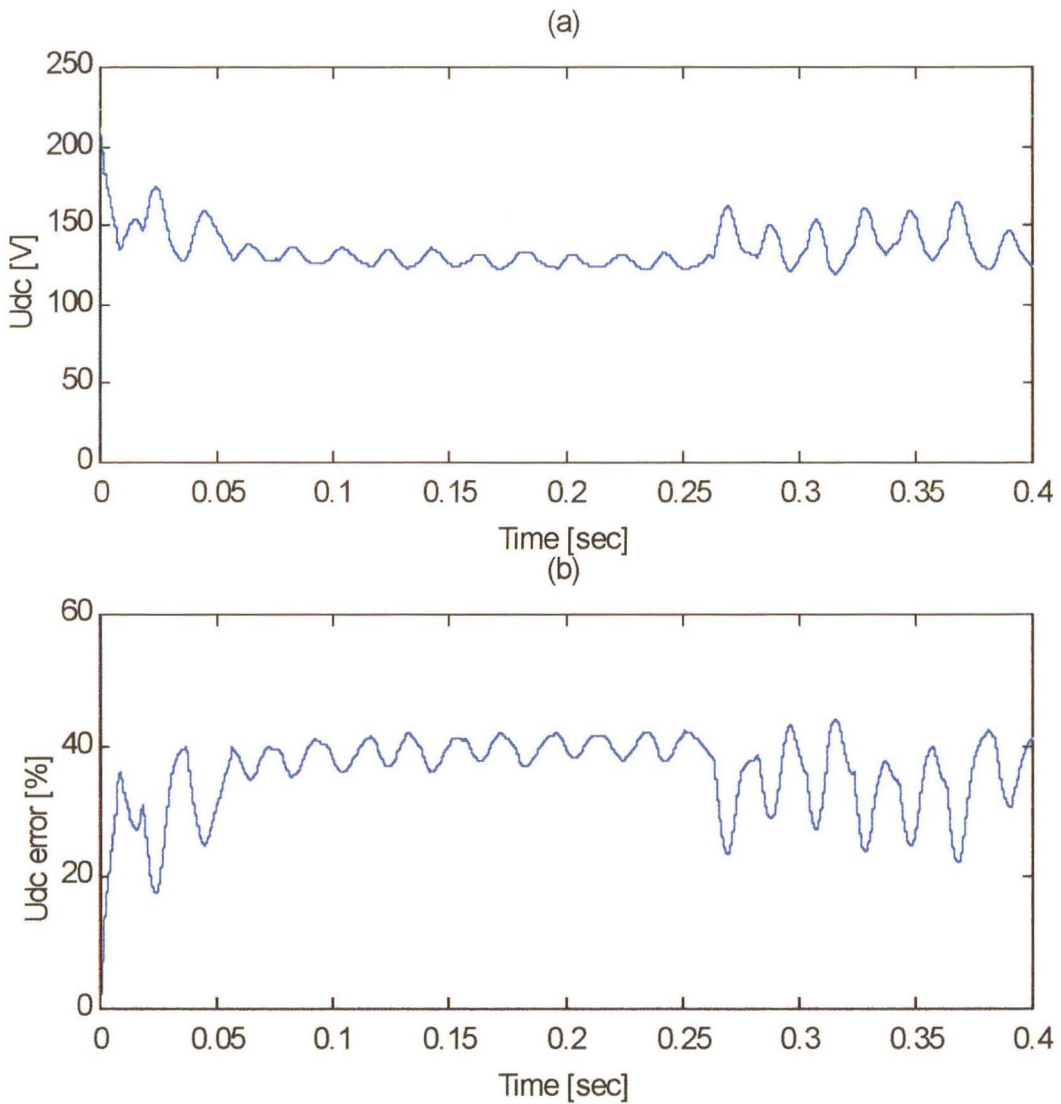


Fig. 4.28 (a) DC link voltage, (b) DC link tracking error, with $C_v = 1.1$ pu, $f_s = 4$ kHz, $f_{han} = 2$ kHz, $B = 0.12$ pu, $K_p = 0.01$ pu

The next series of results (Figs. 4.29 and 4.30) show the system response with a proportional gain of 0.14.

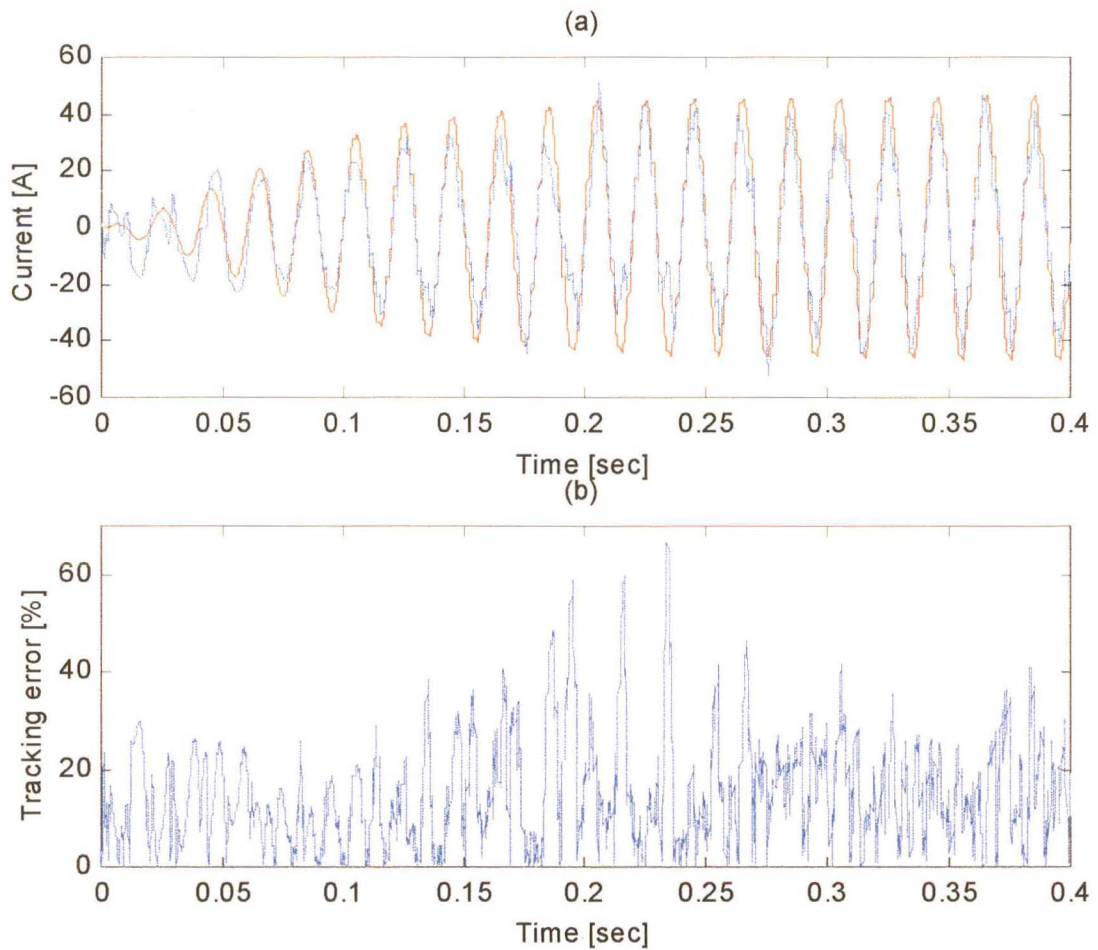


Fig. 4.29 (a) Phase A supply current (I_{sa}) and COT ANN alpha reference current (I_{α_ref}), with $C_v = 1.1$ pu, $f_s = 4\text{kHz}$, $f_{han} = 2\text{kHz}$, $B = 0.12$ pu and $K_p = 0.15$ pu, (b) Current tracking error

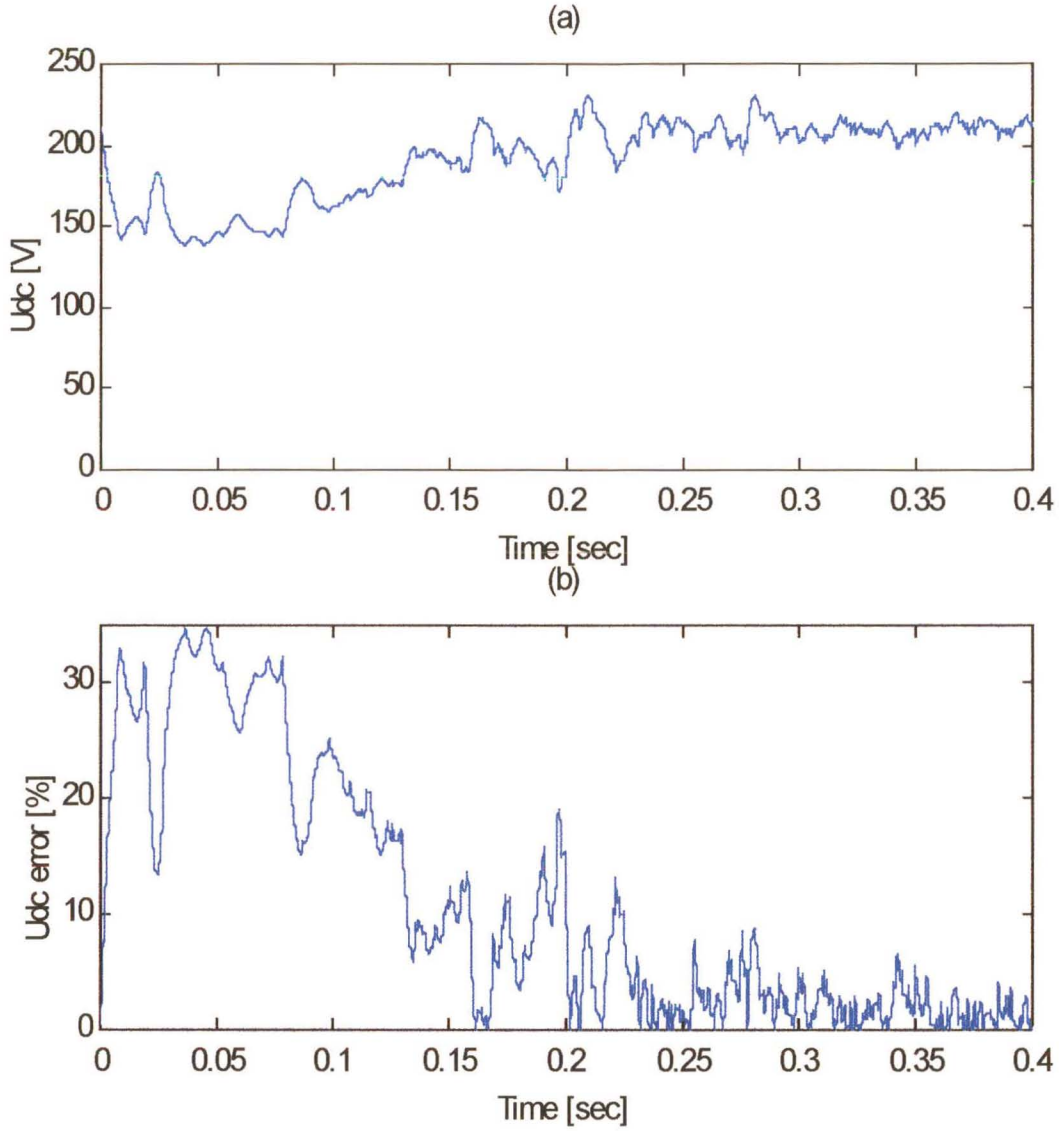


Fig. 4.30 (a) DC link voltage, (b) DC link tracking error, with $C_v = 1.1$ pu, $f_s = 4$ kHz, $f_{han} = 2$ kHz, $B = 0.12$ pu, $K_p = 0.15$ pu

The final series of results taken (Fig. 4.31 and 4.32) are for a high proportional gain (1.0).

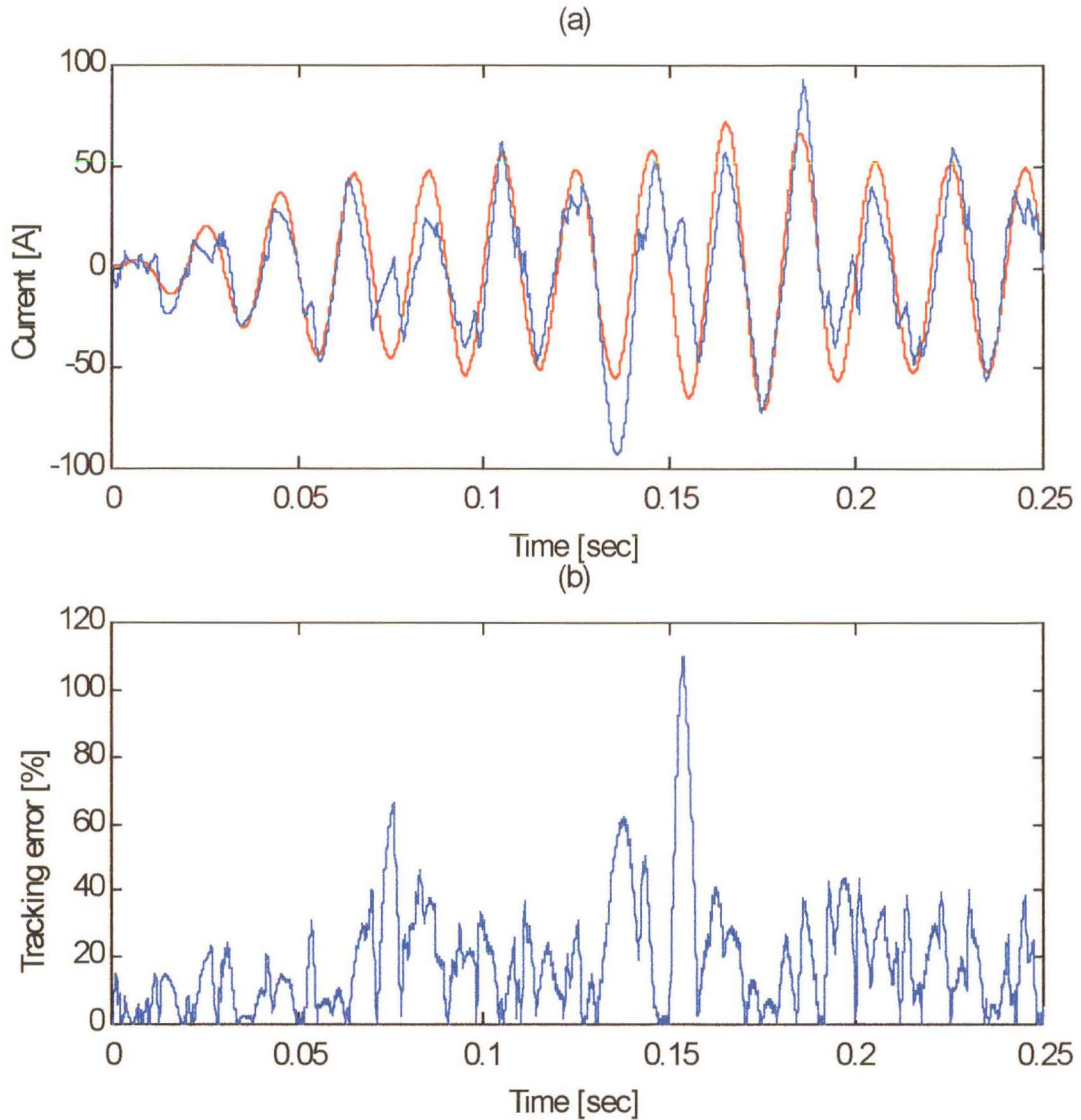


Fig. 4.31 (a) Phase A supply current (I_{sa}) and COT ANN alpha reference current (I_{α_ref}), with $C_v = 1.1$ pu, $f_s = 4$ kHz, $f_{han} = 2$ kHz, $B = 0.12$ pu and $K_p = 1.0$ pu, (b) Current tracking error

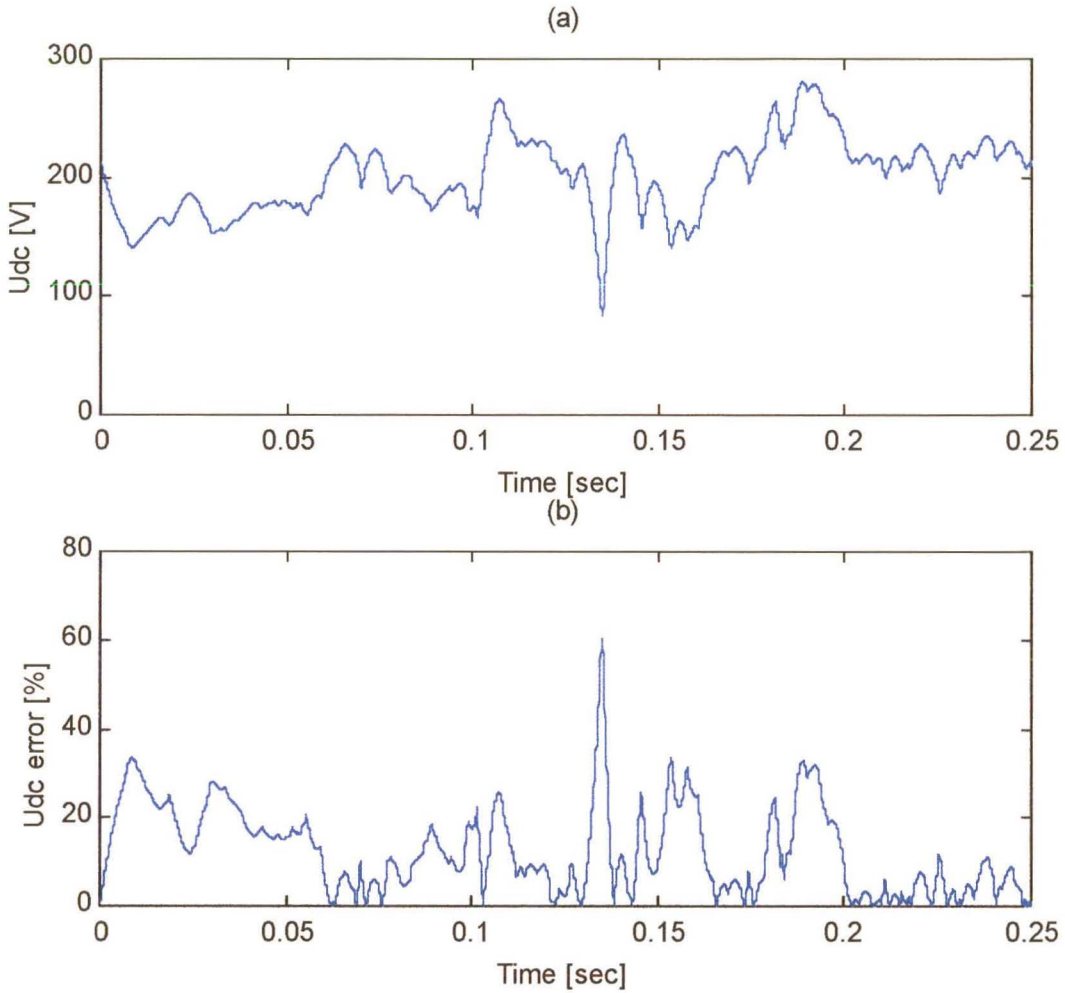


Fig. 4.32 (a) DC link voltage, (b) DC link tracking error, with $C_v = 1.1$ pu, $f_s = 4$ kHz, $f_{han} = 2$ kHz, $B = 0.12$ pu, $K_p = 1.0$ pu

It can be concluded, that the voltage regulator should be operated with a proportional gain between 0.15 and 0.2, to achieve the desired voltage regulation. The ANN current controller variables are adjusted to the desired operating point as described above, and re-simulated, providing Figs. 4.33 and 4.34.

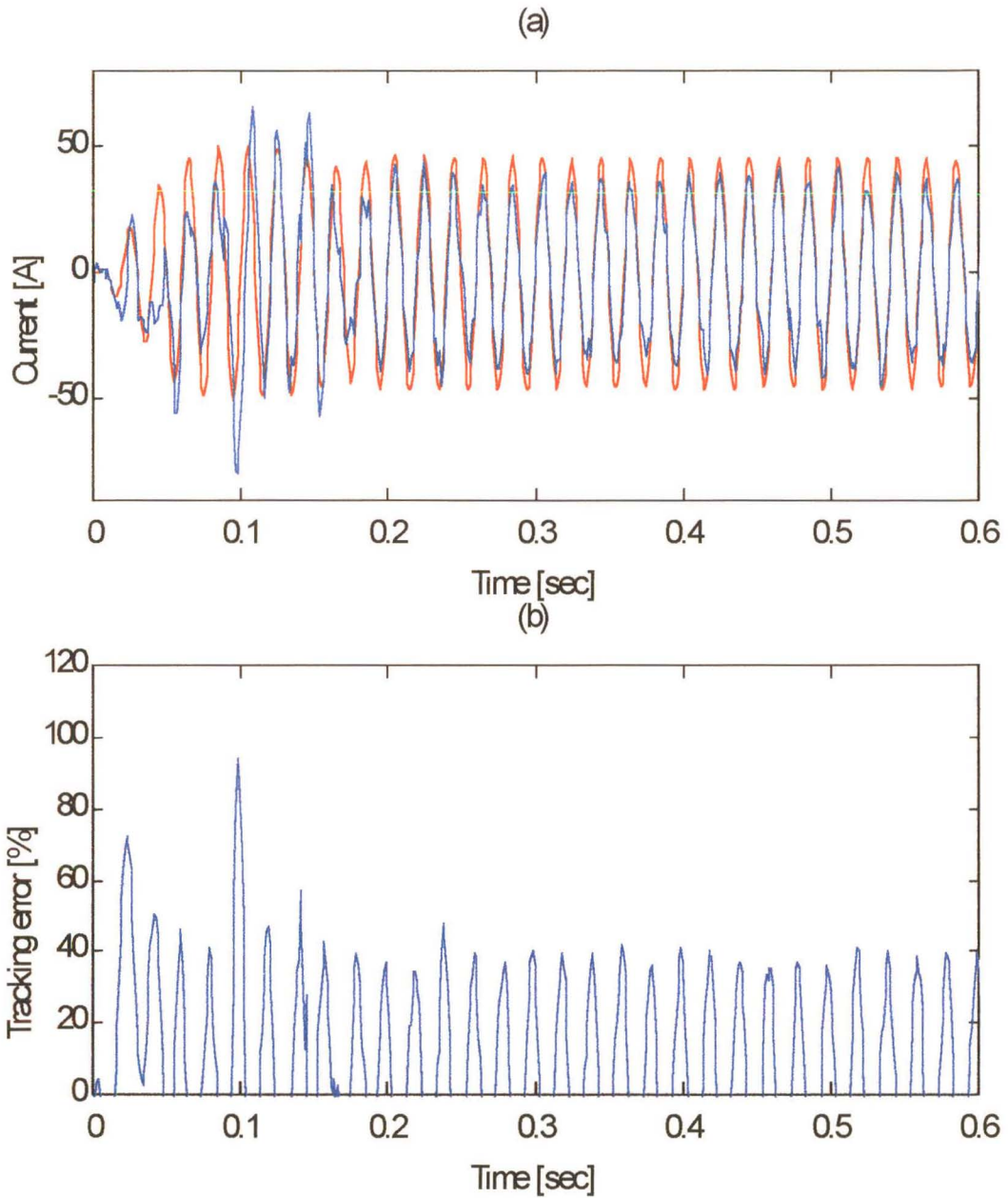


Fig. 4.33 (a) Phase A supply current (I_{sa}) and COT ANN alpha reference current (I_{α_ref}), with $C_v = 1.1$ pu, $f_s = 4$ kHz, $f_{han} = 2$ kHz, $B = 0.12$ pu and $K_p = 0.15$ pu, (b) Current tracking error

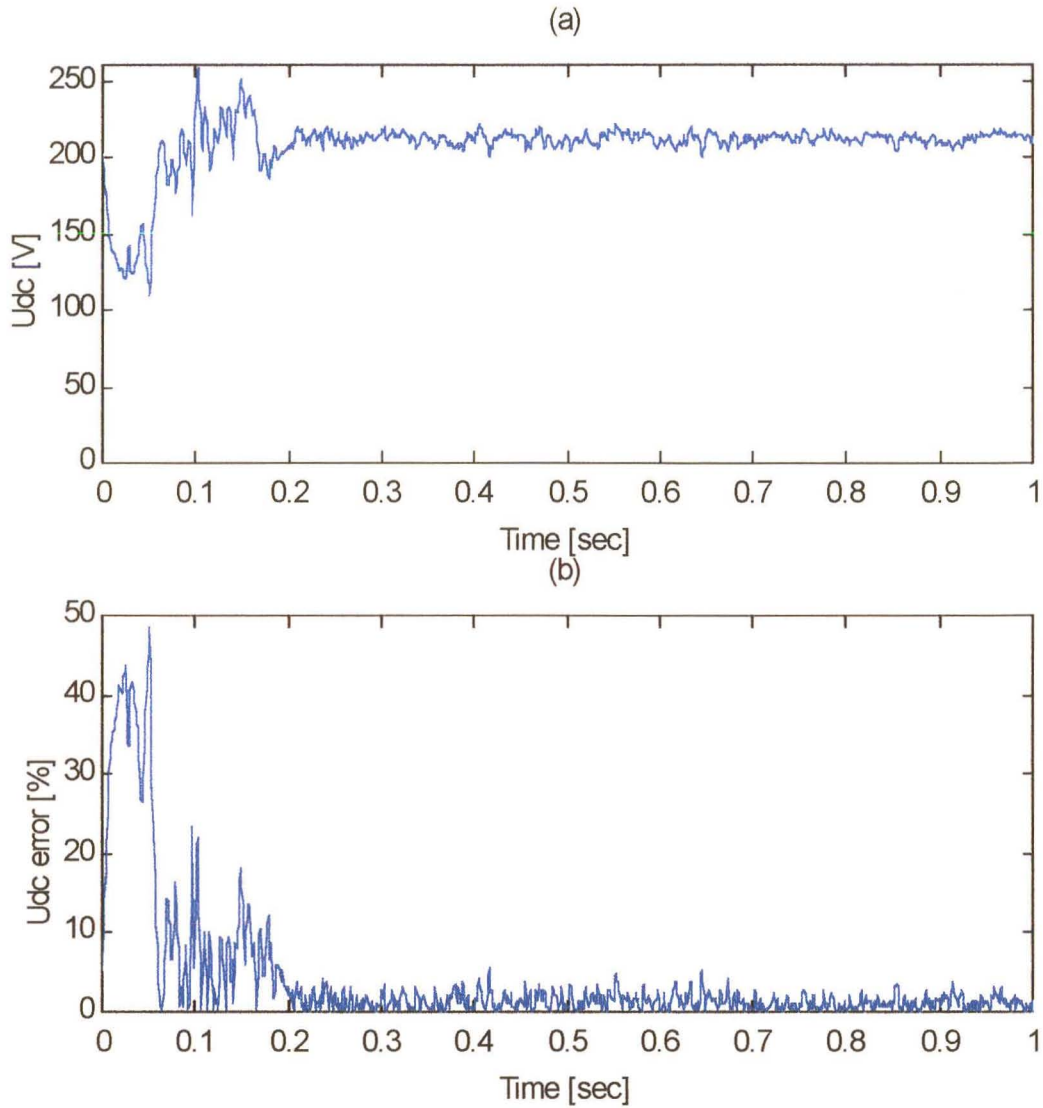


Fig. 4.34 (a) DC link voltage, (b) DC link tracking error, with $C_v = 1.1$ pu, $f_s = 4$ kHz, $f_{han} = 2$ kHz, $B = 0.12$ pu, $K_p = 0.15$ pu

The simulation results show that the desirable operating conditions for the COT ANN current controller can be achieved at the following operating points:

- | | | | |
|-----|-------------------------|---|-------------------|
| 7) | ANN Learning rate | : | $B = 0.12$ pu |
| 8) | ANN Voltage constant | : | $C_v = 1.1$ pu |
| 9) | PWM Switching frequency | : | $f_{han} = 2$ kHz |
| 10) | ANN Sampling frequency | : | $f_s = 4$ kHz, |
| 11) | Proportional gain | : | $K_p = 0.15$ pu |

At these operating points, the best current tracking capability is achieved, and the DC link voltage is regulated to the desired value of 212 volts as shown in Fig. 4.34. Furthermore at these operating points, the voltage steady-state tracking error is in the order of 5 %, which is comparable to the approximately 5 % error achieved by Chathury (see [CHATHURY1] Fig. 7.5 pp 7.9). Unfortunately the current steady-state tracking error is in the order of 40 to 45 %, whereas Chathury achieved an error of approximately 10 to 15 % (see [CHATHURY1] Fig. 7.5 pp 7.9). It should be noted that Chathury's research does not document an exact value for the steady-state tracking error, therefore the errors stated above are approximations from his results, as an exact value cannot be achieved from the figures in Chathury's thesis.

In conclusion, the simulations have shown that the steady-state current error achieved with the COT ANN is larger than that achieved by Chathury. However, it is theoretically feasible to use the COT ANN current controller to control a boost rectifier.

4.4 Summary

This chapter analyses the feasibility of replacing Chathury's current controller with a COT ANN current controller, to control a boost rectifier, by means of simulation. The investigation revealed that Burton's COT ANN current controller for the SCIM could not be directly applied to the boost rectifier. The COT ANN inputs, outputs and internal variables had to be changed according to the procedure set out by Wishart [WISHART1], to control the boost rectifier. The changed COT ANN current controller was then adjusted to achieve satisfactory behavior.

This chapter shows that the COT ANN current controller must be finely "tuned" otherwise it cannot achieve current convergence. The "tuning process" used to achieve convergent operation of the COT ANN current controller, while achieving a reliable transient response under fault conditions, was discussed, and the simulation results indicate that a COT ANN current controller could be used with a boost rectifier. However, it is shown that the Chathury's current controller has superior current tracking capabilities.

The next chapter describes the feasibility study that was performed, to ascertain whether the COT ANN could be implemented using a DSP.

CHAPTER 5

FEASIBILITY OF USING A DSP-BASED COT ANN CURRENT CONTROLLER

5.1 Introduction

Chapters 1 and 2 provided an overview of the research performed by Chathury, and Burton on boost rectifiers and COT ANN current controlled SCIM drives respectively. In Chapter 2 it was proposed to extend Chathury's by using a COT ANN current controller for the boost rectifier [CHATHURY1]. In Chapter 3 Burtons simulations were extended to include the nonlinear effects of the PWM and inverter, and the COT ANN parameters were modified which enabled frequencies up to 100 Hz to be achieved. In Chapter 4 a Narmax model [WISHART1] of the boost rectifier was developed to determine the inputs, outputs and structures of the COT ANN current controller for the boost rectifier. The COT ANN current controller system was then implemented and simulated in the boost rectifier, the results showed that it was a viable controller.

This chapter conducts an investigation, to determine whether any of the currently available DSP's are suited for real-time implementation of a COT ANN current controller. The aim of this investigation is twofold, firstly, to ascertain whether a single processor-based DSP card could provide superior performance to that of the Transputer, and secondly, whether it will comply with the requirements of a minimum COT ANN sampling frequency of 4 kHz, as established in Chapter 4.

5.2 Processor Requirements

As stated in Chapter 2, Burton's system was limited to a maximum COT ANN sampling frequency of 500 Hz due to the processing speed of the Transputer. However, the feasibility study performed in Chapters 3 and 4 showed that a sampling frequency of between 4 and 10 kHz was required to achieve convergent COT ANN operation. At the time of writing this thesis, the fastest and most cost effective commercially available single processor solution was the TMS320C32 DSP processor, which was available as a PC-BASED card (ADC64 card), from Innovative Integration [II1]. This chapter investigates the cycle time required for the operation of the controller, in order to determine the feasibility of using this DSP to implement the controller. Chapters 3 and 4 showed that a minimum sampling frequency of 4 kHz is necessary for the COT ANN to operate convergently. Thus a processor is required which can operate at approximately eight (8) times the speed of the Transputer.

The next subsection compares the transputer's performance to the ADC64 DSP card.

5.2.1 Transputer versus ADC64 DSP card

The T800 Transputer-based EPHSC used by Burton [BURTON1-2] is a 32-bit CMOS microcomputer, which is generally used for real time processing functions. The Transputer benchmark is its performance of 10 million instructions per second (MIPS) [INMOS1].

The ADC 64 card, which utilises a TMS320C32 DSP, is an enhanced 32-bit floating point processor operating at 60 MHz (now available at 150 MHz), with a performance of 30 MIPS [TI1]. The superior performance statistics of the ADC64 DSP card, suggests that it should be able to achieve a COT ANN sampling frequency of three (3) times that achieved by Burton on the Transputer.

The next section performs a real-time investigation using the ADC64 DSP card, to determine the attainable COT ANN sampling frequency, as discussed above.

5.2.2 Controller Computational Requirements

This section will ascertain the processing time required by using Burton's hand optimised C code of the COT ANN controller to perform its necessary control functions, on the DSP. It should be noted that the COT ANN requires double precision floating point operation to perform all its necessary control functions. The Hyperception RIDE40 real-time workspace is utilised as the operating platform for this investigation.

Hyperception's RIDE40 software is based on a revolutionary concept in Digital Signal Processing (DSP) and related fields, that address scientific and engineering problems through the use of individual software applications running under the Windows environment [HYPER SIGNAL1]. Hypersignal also provides support for a number of optional plug in DSP/acquisition cards, such as the Innovative Integration PC32 and ADC64 DSP cards [II1], to allow associated Real-time Signal processing.

The boost rectifier used for this investigation is identical in all respects to the one used by Chathury [CHATHURY1-3] in his research. The hardware profile is described in detail in APPENDIX D. The real-time model shown in Fig. 5.1 consists of two user-defined software blocks, linked together in the Hypersignal workspace, to form a closed loop system, namely:

- 1) a COT ANN stator current controller and voltage controller block (code provided in APPENDIX C.2). Fig. 5.2 shows the flow diagram of the COT ANN current controller module source code. The following main steps are sequentially executed in the program:

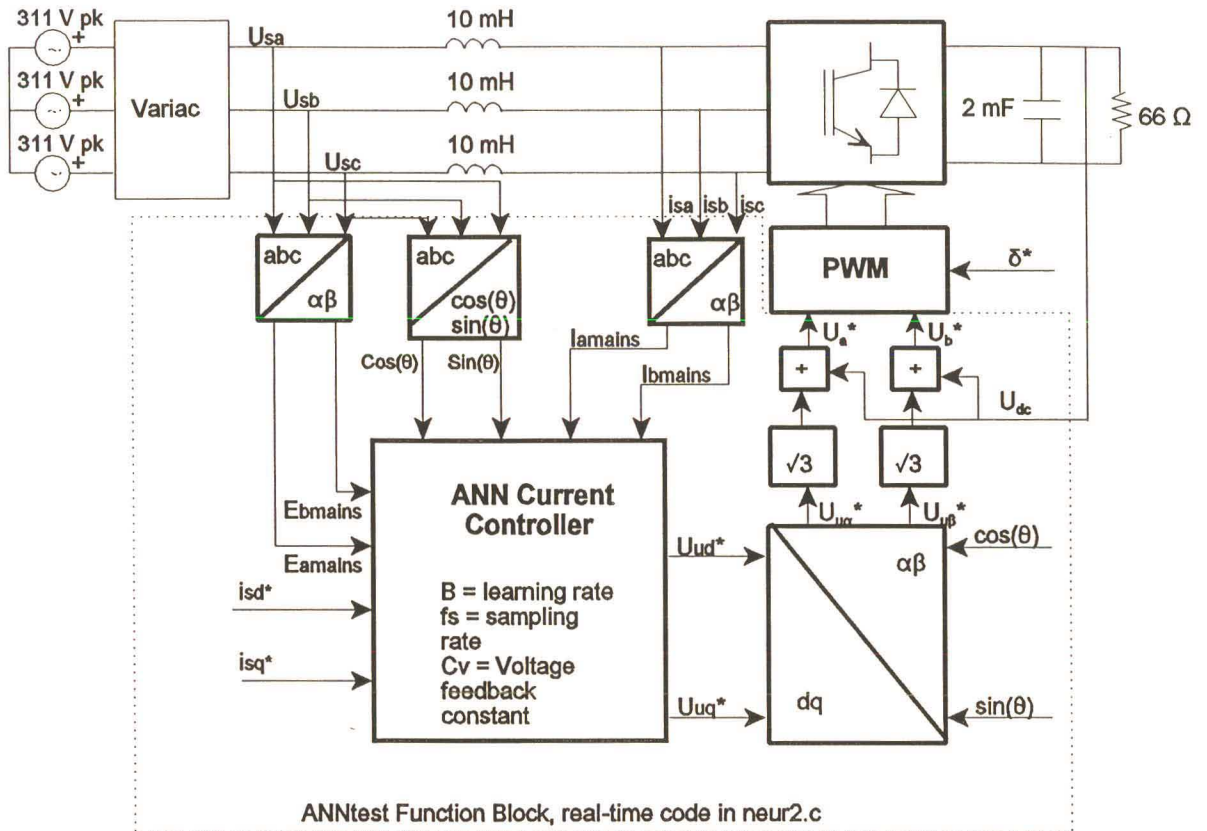


Fig. 5.1 Closed loop COT ANN current control scheme for a boost rectifier

- Step 1: All the simulation parameters and variables (e.g. sampling period), identification ANN parameters and variables (e.g. ANN size, gains, initial weights, inputs), induction motor parameters and variables (e.g. impedances, initial conditions etc.) are initialised.
- Step 2: The main loop starts when an enable or interrupt signal is received.
- Step 3: Steps 4 to 9 are omitted if the loop counter value is less than the corresponding start time T_{start} .
- Step 4: v_α and v_β are written to PWM control block inputs.
- Step 5: The digital values of the A B and C phase source voltages and currents are then read in from the ADC64 cards Analog to digital converter and converted to floating point numbers. The synchronisation process is then performed, to attain a unity power factor.

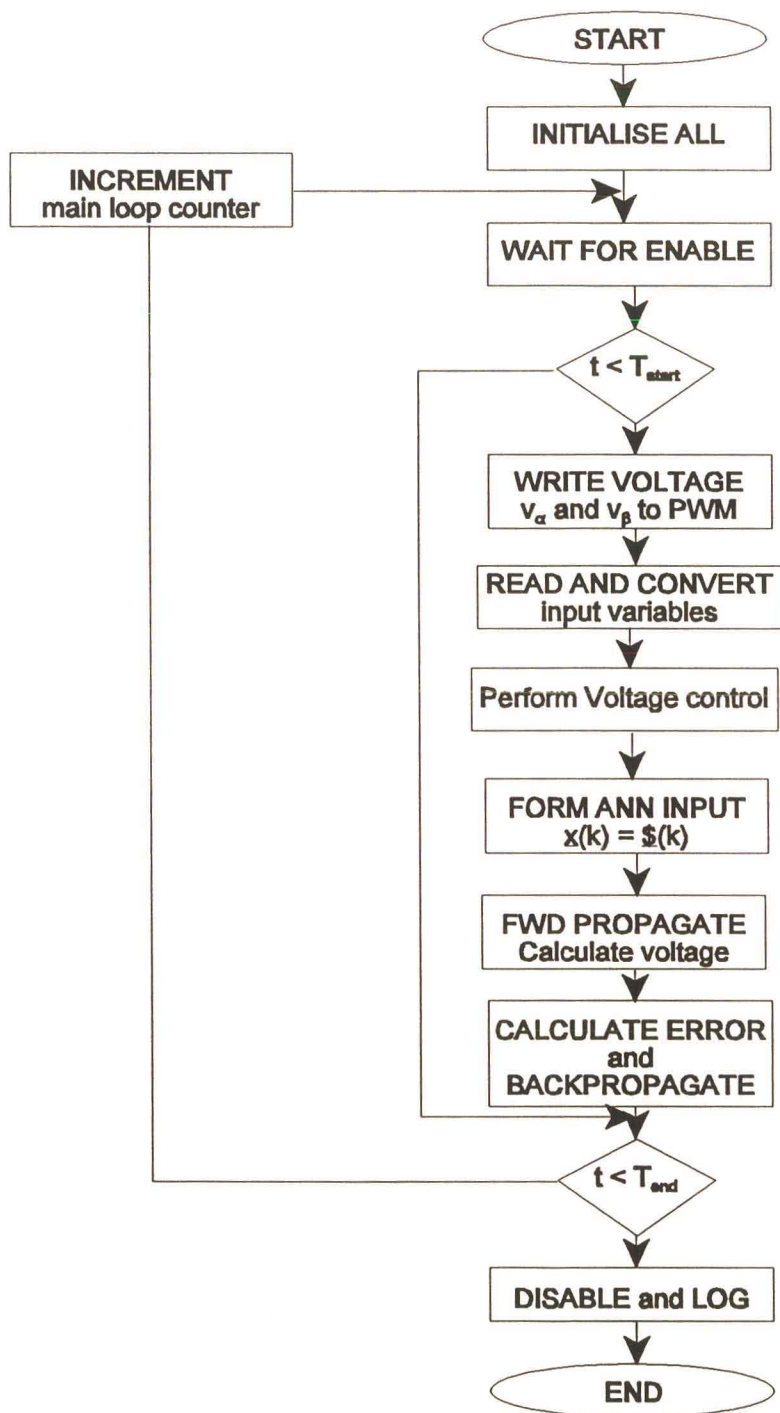


Fig. 5.2 Flow diagram of the Hypersignal real-time COT ANN controller, including voltage control, for a boost rectifier.

- Step 6: The voltage PI loop output is calculated.
- Step 7: The ANN input $\underline{x}(k+1)=\underline{\hat{x}}(k)$ is formed.
- Step 8: $\underline{x}(k+1)$ is forward propagated through the ANN to obtain $\underline{y}(k)=\underline{\hat{f}}_{el}^{(k-1)}(.)$.
- Step 9: $\underline{i}^*(k+1)$ is calculated, either with or without the use of the reference model [BURTON1], and used with $\underline{\hat{f}}_{el}^{(k-1)}(.)$ and constant c_v to calculate $\underline{y}^*(k)$ using Eq. (2.11) from Chapter 2.
- Step 10: The ANN output error $e_y(k) = \underline{i}(k) - \underline{i}^*(k)$ is calculated and backpropagated through the ANN to calculate $W(k+1)$ and $V(k+1)$, which represents $\underline{\hat{f}}_{el}^{(k-1)}(.)$, for the next sampling period.
- Step 11: Step 11 is omitted if the specified loop count corresponding with T_{end} has been reached.
- Step 12: The main loop counter is updated and the program returns to step 2.

- 2) a PWM control block (C code provided in APPENDIX C.1). Fig. 5.3 shows the flow diagram of the PWM control block source code. The following main steps are sequentially executed in the program:

- Step 1: The module waits for a start signal.
- Step 2: The module reads in the v_a and v_b values from the ANN controller outputs.
- Step 3: The module calculates modulating signals, by simulating the Hanning PBM 1/87 ASIC's [HANNING1] operation (this ASIC is a PWM generator).
- Step 4: The switching times are calculated.
- Step 5: The module then generates switching events, which are written to the Inverter switching devices.

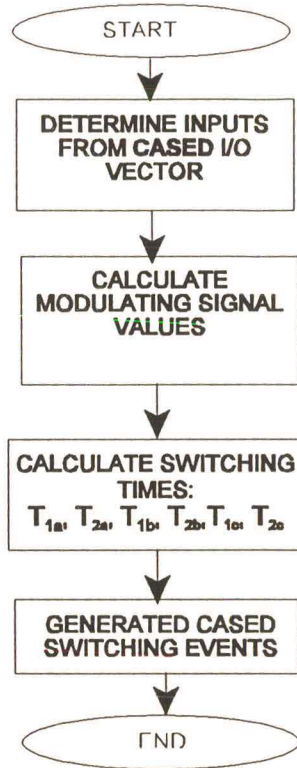


Fig. 5.3 Flow diagram of real-time PWM block

These blocks are configured in such a way, that a bit is set high at the beginning of the controller calculation cycle, and reset when the control variables are written out. Fig. 5.4 shows the processing pulse obtained from the combined PWM/COT ANN controller (this is total processing time of the COT ANN back and forward propagation, and the PWM), while running in real-time on the DSP card.

It should be noted that Burton's hand optimised C code structure is used here to minimise the calculation cycle. Each block is tested individually to check its operation, by monitoring the system I/O and the internal calculation values and cycles. Finally the PWM block is tested in the commissioning of a voltage/frequency drive.

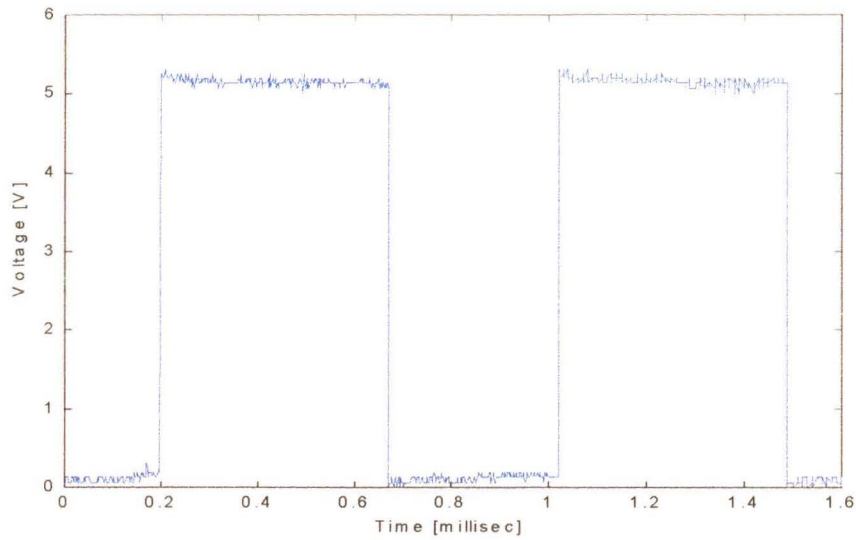


Fig. 5.4 Calculation cycle of the combined PWM/COT ANN control software for the boost rectifier

The processing cycle, obtained using a COT ANN sampling rate of 1.2 kHz, and a PWM switching frequency of 600 Hz, is shown in Fig. 5.4. From Fig. 5.4, it can be seen that the controller requires approximately 480 microseconds to perform the necessary functions. It is also noted that the calculation cycle time increases occasionally, due to calculations being performed by the COT ANN. Under these conditions, the COT ANN requires 460 microseconds to perform its matrix calculations and manipulations, as shown in Fig. 5.5; the remainder of the time is used by the A/D read and PWM write functions. Thus it is important to minimise the COT ANN calculation cycle, or to obtain a platform that can provide sufficient processing speed to sustain the COT ANN calculation cycle at the desired operating point. It should be noted that the code could be further hand optimised to obtain further speed increases, but it would still be insufficient for the requirements of this thesis.

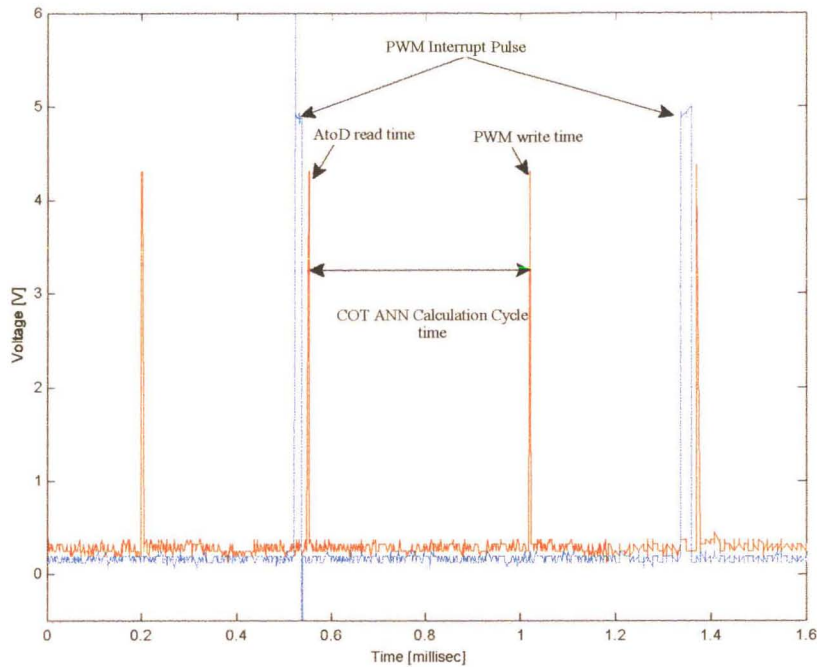


Fig. 5.5 COT ANN matrix and calculation time cycle

If these values, shown in Fig 5.4, are extrapolated, it is seen that the TMS320C32 DSP processor can only execute the controller at a maximum sampling frequency of 1.5 kHz (approximately three times that achieved with the Transputer). This result agrees with the performance evaluation performed in the previous section. Under these conditions the COT ANN is divergent, as shown in Fig. 5.7, which agrees with the simulation results obtained in Chapter 3 and 4. Fig. 5.6 and 5.7 present an example of the simulation and practical results obtained using a COT ANN sampling frequency of 1.5 kHz, respectively. Here it is seen that the practical and simulated system exhibits similar responses, except that the actual phase currents are larger. This is due to system parameter differences between the simulation and practical systems, and the non-ideal resistance's and inductances, which can not be accounted for in CASED. Furthermore, the noise in the practical AC supply phase A current in Fig. 5.7 is attributed to the impedance of the three-phase Variac used in the practical system, across which the voltage spikes and transients appear, due to the switching action of the boost rectifier.

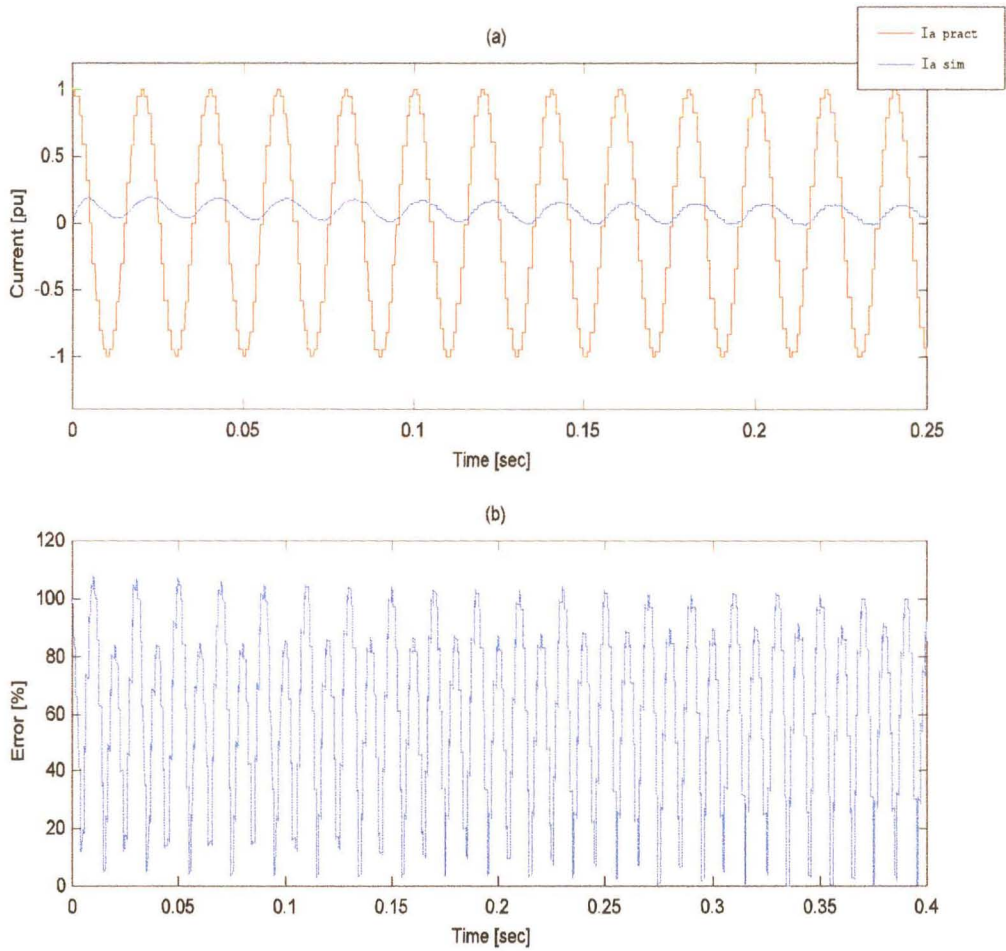


Fig. 5.6 (a) Phase A supply current (I_{sa}) and ANN alpha reference current ($I_{\alpha_{ref}}$), with $C_v = 1.1$, $T_s = 1.5$ kHz, $T_{han} = 0.75$ kHz, $B = 0.12$ and $K_p = 0.15$, (b) Current tracking error

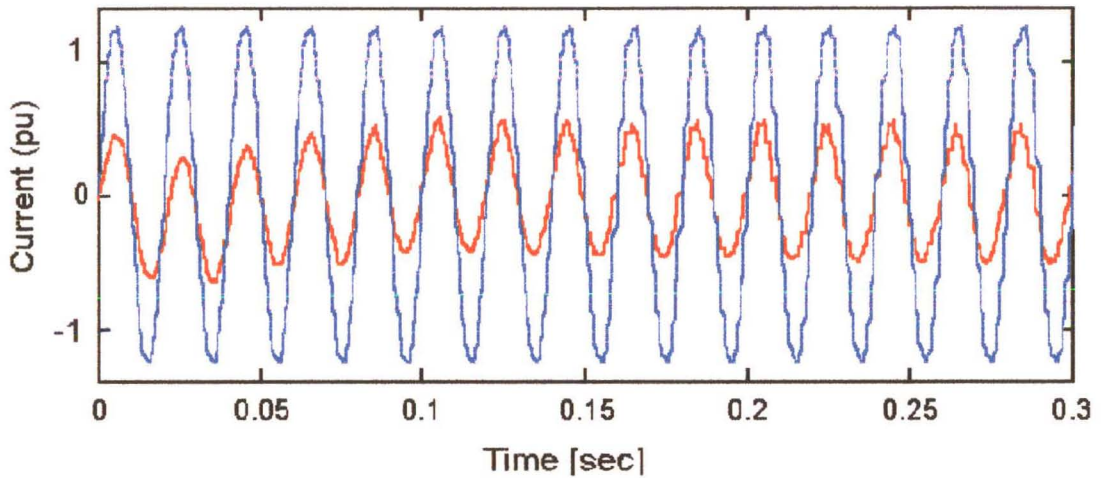


Fig. 5.7 Practical results: Phase A supply current (I_{sa}) and ANN alpha reference current (I_{α_ref}), with $C_v = 1.1$, $T_s = 1.5$ kHz, $T_{han} = 0.75$ kHz, $B = 0.12$ and $K_p = 0.15$

According to the simulations, the COT ANN should be operated at sampling frequencies above 4 kHz, before convergent operation is achievable. It can thus be concluded that the TMS320C32 DSP's (ADC64 card) processing speed is insufficient to implement the real-time controller. Therefore, it is not feasible to implement the COT ANN current controller for the boost rectifier using the TMS320C32 DSP.

5.2.3 Proposed DSP solution

A single processor, which does not contain multiple processing paths, is desirable as it eliminates the need for code scheduling, which is required by a multiple processor platform. This allows for easy controller implementation, and results in a more cost effective solution. At the onset of the research presented in this thesis, a single processor with a sufficiently high processing speed had not yet been released. During the latter part of the writing of this thesis, the TMS320C67 DSP was released. This section compares the TMS320C32 and TMS32C67 DSP's, in order to determine whether the TMS320C67 can provide sufficient processing speed to implement the COT ANN with a sampling frequency of 4 kHz.

The TMS320C32 DSP has a performance of 30 MIPS, and 60 MFLOPS [TI1], whereas the TMS320C67 has a performance of 1336 MIPS and 250 MFLOPS [TI2] at 167 MHz for double precision operations. Thus the TMS320C67 DSP floating point operation speed is 5 times faster than that of the TMS320C32 DSP, and 40 times faster when compared on MIPS. The TMS320C67 DSP should achieve a COT ANN sampling frequency of at least 5 times higher than that achieved by the TMS320C32. In the previous section, it was shown that the TMS320C32 DSP's processor could only achieve a COT ANN sampling frequency of 1.5 kHz, which suggests that the TMS320C67 DSP should be able to achieve a minimum COT ANN sampling frequency of 20 kHz.

As mentioned in the previous section, the COT ANN requires 460 microseconds to perform all its calculations and matrix functions, on the TMS320C32 DSP, which has a single ALU (Arithmetic Logic Unit). However, the TMS320C67 has 4 ALU's which will speed up the calculation process by allowing the software to be split and run as parallel processes, provided the code is optimised to use multiple ALU's for matrix and vector multiplication.

As mentioned previously, the TMS320C67 DSP was released at the latter part of writing this thesis. Therefore it was possible to obtain a TMS320C67 DSP at the completion of the writeup, to perform the calculation cycle test for comparison with that obtained for the COT ANN operating on the TMS320C32 DSP card.

The processing cycle was obtained utilising the identical code and I/O structure described for the TMS320C32 processor above. The processing cycle, obtained using a COT ANN sampling rate of 10 kHz, and a PWM switching frequency of 5 kHz, is shown in Fig. 5.8. From Fig. 5.8, it is seen that the controller requires approximately 74 microseconds to perform the matrix calculations and manipulations.

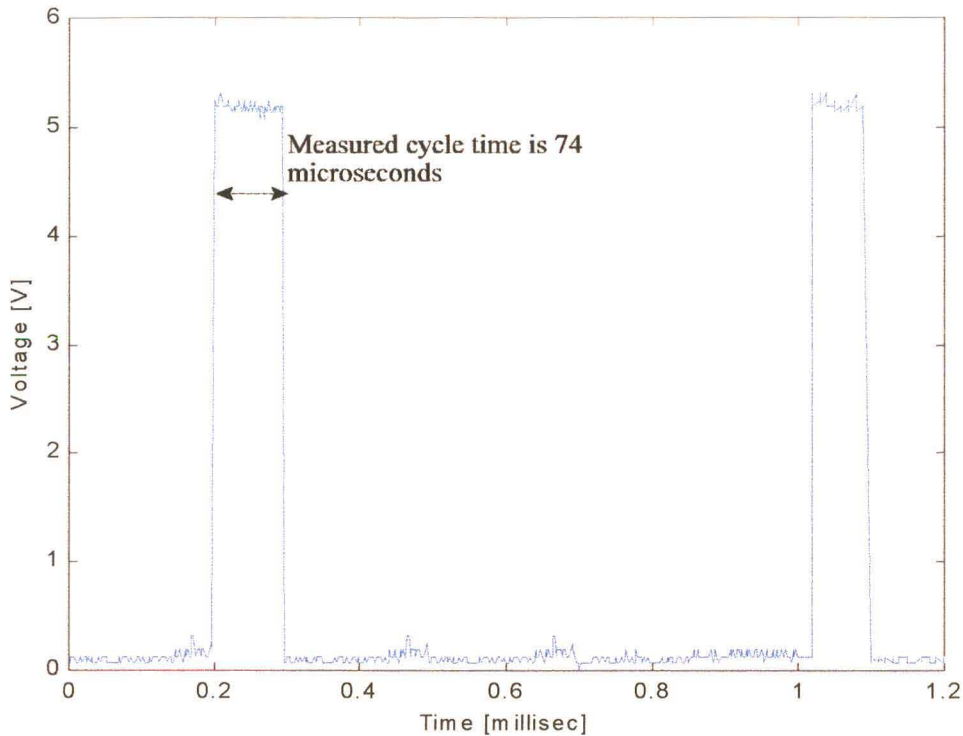


Fig. 5.8 Calculation cycle of the combined PWM/COT ANN control software for the boost rectifier on the TMS320C67 DSP

From this it is shown that the TMS320C67 DSP has sufficient processing power to implement a real-time COT ANN current controller for a boost rectifier. It is proposed that further research be performed, using the TMS320C67 DSP, to implement the COT ANN current controller for a boost rectifier. The code implemented on the TMS320C32 DSP, as described in section 5.2.2, requires slight changes, before it can be implemented on the TMS320C67 DSP, as they have a similar structure. However the hardware requires major modifications, such as the redesign of the PWM card, to enable it to operate with the TMS320C67 DSP. These changes require an extra years worth of work, which is beyond the scope of this thesis.

In summary it has been shown that the TMS320C67 DSP can be used to implement a COT ANN current controller for a boost rectifier. However the final implementation of the system will be part of further research.

5.3 Summary

The TMS320C32 DSP is reviewed as a possible operating platform for the implementation of the COT ANN current controller. However when the controller is run on the DSP, it is found that the DSP can only support COT ANN sampling frequencies up to 1.5 kHz. At this operating point, the controller becomes divergent, which agrees with the simulation results obtained in the feasibility study, performed in Chapter 3 and 4.

The TMS320C67 DSP provides an operating platform, which can sustain the sampling speeds required by the COT ANN current controller. It can be concluded that the TMS320C32 DSP is too slow to implement the COT ANN current controller, but the TMS320C67 provides sufficient processing capability to implement the COT ANN.

CHAPTER 6

CONCLUSION AND RECOMMENDATIONS

6.1 General

In South Africa QoS has become a noticeable and costly problem for the industrial and commercial sectors, as it affects both machinery and controls. Furthermore, Industry uses a large number of VSD's, where the rectifier consists of either a six-pulse diode rectifier, or a phase controlled thyristor rectifier, both of which cause non-sinusoidal supply currents. These currents result in harmonic distortion and non-unity power factor operation, which can create Quality of Supply problems further down the line for other energy consumers. The research described in this thesis was to determine the feasibility of using a COT ANN current controlled boost rectifier, by combining the works of Chathury [CHATHURY1-3] and Burton [BURTON1-2]. This strategy is used, because the COT ANN is self commissioning and adaptive, and therefore does not rely on any specific prior or continually updated knowledge of system parameters. Whereas the conventional controller has to be tuned using prior knowledge of the system parameters. The COT ANN allows the system to be able to overcome the transient effects of QoS on the rectifier system, due to its adaptive properties.

The question of whether or not a real-time adaptively controlled rectifier can be used to overcome QoS problems will form the basis of future research. This is due to the unavailability of a single processor-based DSP with sufficient processing power to implement the COT ANN current controller, at the time of writing the thesis. The TMS320C32 and TMS320C67 DSP's were evaluated as possible platforms for the implementation of the practical COT ANN current controller for the boost rectifier. It was identified that the TMS320C32 processor had insufficient processing speed to implement the controller, whereas the TMS320C67 had sufficient processing speed. Unfortunately a TMS320C67 processor was only made available at the University, at the time when

the thesis writing was complete.

Chapter 2 provided an overview of the pertinent theory and results of the work performed by Chathury [CHATHURY1] and Burton [BURTON1] on Voltage Source boost rectifiers and COT ANN current controllers respectively. Furthermore, it described the basic structure, inputs and outputs required by the conventional current controller of Chathury and the COT ANN current controller of Burton, for use in the development of the COT ANN current controller for the boost rectifier in Chapter 4.

Chapter 3 presented the results of two CASES simulation studies performed, using the existing COT ANN current controlled VSI fed squirrel cage induction motor model, developed by Burton [BURTON1]. The initial set of simulations, performed in CASES, duplicated the simulations performed by Burton [BURTON1] (in this model the inverter was omitted) in order to become familiar with a new simulation package (CASES) and to determine its accuracy in comparison with Burton's simulations. This was followed by a comprehensive simulation study using CASES, applying the COT ANN current controller to a Voltage-Source Inverter (VSI) fed Squirrel Cage Induction Motor (SCIM) drive model. This model included the inverter and PWM switching block in order to gain an understanding of the nature and form of the COT ANN while operating under the non-linear effects of the Voltage-Source Inverter (VSI) SCIM drive. The simulations were then discussed and analysed to gain familiarity with the COT ANN current controller developed by Burton, and to evaluate the conditions under which the COT ANN could operate and achieve convergence under the non-linearities of the inverter and PWM.

In Chapter 4, the structure of the COT ANN current controller developed by Burton [BURTON1] was described, along with the variable and I/O changes required in order to realise a COT ANN current controller that could control a boost rectifier. This makes use of the mathematical model of a boost rectifier discussed in Chapters 2, and 4. The tuned COT ANN current controller was then simulated in CASES, to verify its effectiveness in controlling a boost rectifier. The findings of this chapter, was that in theory, a COT ANN current controller could be used to control a boost rectifier.

Chapter 5 presented the concept of implementing the real-time COT ANN current controller using a single processor-based DSP. It was proven that none of the currently available single processor-based DSP's are fast enough to implement the real time COT ANN current controller, with a sampling frequency of 4 kHz or higher. The chapter evaluated and compared the TMS320C32 and TMS320C67 DSP's, outlining why the existing single processor-based DSP's are not powerful enough to implement the COT ANN current controller.

In conclusion, the simulations showed that a COT ANN current controller can be used to control a boost rectifier.

6.2 Suggestions for Further Work

This thesis established the initial step toward developing a COT ANN current controlled boost rectifier for the use in variable speed drives, and many other power conversion applications. The research concentrated on the development of a current controller to control a boost rectifier.

Specific suggestions for future work are listed below:

- a. The new C67 DSP should be investigated as the operating platform for the real-time COT ANN current controller. This would form a good extension to the feasibility study discussed in this thesis.
- b. Once the real-time COT ANN current controller has been implemented, a feasibility study should be performed to determine whether the real-time system is in agreement with the simulation results obtained in this thesis.
- c. An investigation should be performed to evaluate and compare the COT ANN current controller to conventional current controllers.
- d. Further research should be performed with the COT ANN current-controlled converter to identify to what extent the system can be used to overcome voltage dips

and other QoS problems in networks. It is suggested that a dip generator be developed, where the magnitude and duration of the voltage dip/fluctuation can be varied. This piece of equipment should be used with the converter to identify the class of dip that the system can withstand (i.e., under what conditions it will maintain a constant DC link voltage) in comparison to conventional systems.

- e. Finally, it is suggested that research should be performed to investigate the effects of voltage dips or fluctuations on a cascaded converter inverter system, as shown in Chapter 1 Fig. 1.2. This research could be used in the Paper and Pulp industry where voltage dips cause system synchronisation loss (i.e., paper winder and un-winder VSD's lose synchronisation), which in turn causes paper breaks and system down time. The overall objective of the investigation would be to develop a cascaded controller that could ensure that two inverters, for example in a paper winder / un-winder system, would remain in synchronism even under voltage dip conditions.

REFERENCES

R.1 References

- [ASTROM1] Astrom KJ, Wittenmark B, "Adaptive Control", Addison-Westley, 1989, ISBN 0-201-09720-6.
- [AFRICAN EN1] African Energy, "Revealing Africa's Energy Development", February-March 1999, Volume 1, No. 1, pp 37 - 39, Resource Publications.
- [BISWAS1] Biswas SK, Mahesh MS, Iyengar BSR, "Simple new PWM patterns for thyristor three-phase AC/DC converters", IEE Proceedings, Vol. 133, Pt. B, No. 6, November 1986, pp 354 - 358.
- [BOSE1] Bose BK, "Evaluation of Modern Power Semiconductor Devices and Future Trends of Converters", IEEE Transactions on Industry Applications, Vol. 28, No. 2, March/April 1992, pp 403 - 413.
- [BOSE2] Bose BK (Editor), "Modern Power Electronics", IEEE Press, Piscataway, New Jersey, 1991, ISBN 0-87942-282-3.
- [BOOST1] Boost MA, Ziogas PD, "State of the Art Carrier PWM Techniques: a Critical Evaluation", IEEE Trans. Ind. Applicat., Vol. 24, No. 2, March/April 1988, pp 271-280.

-
- [BOWES1] Bowes SR, Midoun A, "New PWM switching strategy for microprocessor controlled inverter drives", IEE Proceedings, Vol. 133, Pt. B, No. 4, July 1986, pp 237 - 254.
- [BOYS1] Boys JT, Walton SJ, "A loss minimised sinusoidal PWM inverter" IEE Proceedings, Vol. 132, Pt. B, No. 5, September 1985, pp 260 - 268.
- [BROD1] Brod DM, Novotny DW, "Current Control of VSI-PWM Inverters", IEEE Industry Applications Conference Record, 1984, pp 418 - 425.
- [BROECK1] Van Der Broeck HW, Skudelny HC, Stanke GV, "Analysis and Realisation of a Pulsewidth Modulator Based on Voltage Space Vectors", IEEE Transactions on Industry Applications, Vol. 24, No. 1, January/February 1988, pp 142 - 150.
- [BUHL1] Buhl MR, Lorenz RD, "Design and Implementation of Neural Networks for Digital Current Regulation of Inverter Drives", IEEE Industry Applications Society Annual Meeting, Vol. 1, October 1991, pp 415 - 421.
- [BURTON1] Burton B, "Analysis and Practical Implementation of a Continually Online Trained Artificial Neural Network to Identify and Control Voltage Source Inverter Fed Induction Motor Stator Currents", M.Sc. Eng. Thesis, Dep. Elect.Eng., Univ. Natal, Durban, South Africa, July 1995.
- [BURTON2] Burton B, Kamran F, Harley RG, Habetler TG, Brooke M and Poddar R, "Identification and control of induction motor stator currents using fast on-line random training of a neural network", Conference Record of 1995 IEEE Industry Applications Society, 30th IAS Annual Meeting, Vol 2, October 1995, ISBN 0-7803-3008-0, pp 1781 - 1787.
-

-
- [BURTON3] Burton B, Kamran F, Harley RG, Habetler TG, Brooke M, "A Fast On-line Neural Network Training Algorithm for a Rectifier Regulator", IEEE 21st International Conference on Industrial Electronics, Control and Instrumentation, Vol. 2, ISBN 0-7803-3026-9, pp 1462 - 1467.
- [BURTON4] Burton B, Harley RG, Habetler TG, "The Application and Hardware Implementation of Continually Online Trained Feedforward Neural Networks for Fast Adaptive Control and Prediction in Power Electronic Systems", South African Universities Power Engineering Conference 1999, Potchefstroom, pp 252 - 260.
- [CASED1] McCulloch MD, "Computer Analysis and Simulation of Electric Drives", Users manual, Machines Research Group, Department of Electrical Engineering, University of Witwatersrand, Johannesburg, South Africa, 1990.
- [CHATHURY1] Chathury AS, "Two Quadrant Digitally Controlled Force-Commutated Rectifier with Near Sinusoidal Line Currents at Unity Power Factor", M.Sc. Eng. Thesis, Dep. Elect. Eng, Univ Natal, Durban, South Africa, May 1995.
- [CHATHURY2] Chathury AS, Diana G, Harley RG, "AC to DC Converter with Unity Power Factor and Minimal Harmonic Distortion", South African Universities Power Engineering Conference 1994, Capetown, pp 22 - 26.
- [CHATHURY3] Chathury AS, Diana G, Harley RG, Woodward DR, "Two Quadrant Fully Digitally Controlled Unity Power Factor Converter", Power Electronics, Motion Control Conference Proceedings, September 1994, pp 257 - 262.

-
- [DIXON1] Dixon JW, Kulkarni AB, Nishimoto M, Ooi BT, "Characteristics of a Controlled Current PWM Rectifier-Inverter Link", IEEE Ind. Application Soc. Conf. Rec., 1986, pp. 685-691.
- [DIXON2] Dixon JW, Kulkarni AB, Nishimoto M, Ooi BT, "Characteristics of a Controlled-Current PWM Rectifier-Inverter Link", IEEE Transactions on Industry Applications, Vol. IA-23, No. 6, November/December 1987, pp 1022 - 1028.
- [DIXON3] Dixon JW, Ooi BT, "Indirect Current Control of a Unity Power Factor Sinusoidal Current Boost Rectifier Type Three-Phase Rectifier", IEEE Transactions on Industrial Electronics, Vol. 35, No. 4, November 1988, pp 508 - 515.
- [ELECTRON1] I Smit, "Eskom National Power Quality Measurement Project: Voltage Depressions (Dips/Sags).", MSAIEEE Eskom Power Quality Technology, Elektron Journal, 1996, pp 27 - 30, EE Publishers c.c., Muldersdrift, South Africa.
- [ELEKTRON2] Tom Shaughnessy, "Quality of Supply: Harmonics", Power CET Corporation (Dranetz Group) represented by Spescom MeasureGraph, Elektron Journal, 1996, pp 41 - 43, EE Publishers c.c., Muldersdrift, South Africa.
- [ELEKTRON3] "Semiconductor Growth Continuing", Information from Electronica '98 Trade Fair, Munich, Elektron Journal, June 1998, pp 10, EE Publishers c.c., Muldersdrift, South Africa.

-
- [ENSLIN1] Enslin J, "Unified Approach to Power Quality Mitigation", IEEE International Symposium on Industrial Electronics Proceedings, Pretoria, South Africa, 1998, pp 8 - 20.
- [GREEN1] Green AW, Boys JT, "Hysteresis Current-Forced Three Phase Voltage-Sourced Reversible Rectifier", IEE Proceedings, Vol. 136, Pt. B, No. 3, May 1989, pp 113-120.
- [GREEN2] Green AW, Boy JT, Gates GF, "Three Phase Voltage-Sourced Reversible Rectifier", IEE Proceedings, Vol. 135, Pt. B, No. 6, Nov. 1988, pp 362-370.
- [GREEN3] Green AW, Boys JT, "Current-forced single-phase reversible rectifier" IEE Proceedings, Vol. 136, Pt. B, No. 5, September 1989, pp 205 - 211.
- [HABETLER1] Habetler TG, "A Space Vector-Based Rectifier Regulator for AC/DC/AC Converters", IEEE Transactions on Power Electronics, Vol. 8, No. 1, January 1993, pp 30 - 36.
- [HANNING1] Hanning Electro-werke GmbH & Co., "Pulsewidth Modulator PBM 1/87 PBM 1/89 Data Sheet", Revision 2.0, October, 1993, Oerlinghausen, Germany.
- [HARASHIMA1] Harashima F, Demizu Y, Kondo S, Hashimoto H, "Application of Neural Networks to Power Converter Control", Conference record of the 1989 IEEE Industry Applications Society Annual Meeting, Part 1, 1989, pp 1086 - 1091.
- [HYPER SIGNAL1] Hyperception, "Hypersignal Block Diagram/Ride User's Manual", Hyperception Inc., 1996, United States.
-

-
- [II1] Innovative Integration, "ADC64/cADC64 Hardware Manual", Innovative Integration Inc., 1994, United States.
- [INMOS1] Inmos, "TMS T800 transputer", Inmos group of companies, Danziger Strasse 2, 8057 Eching, Munich, West Germany.
- [KLEINHANS1] Kleinhans CE, "Simulation and Practical Implementation of Field Orientated Control on the Current Source Inverter-Fed Induction Motor", M.Sc. Eng. Thesis, Dep. Elect. Eng., Univ. Natal, Durban, South Africa, May 1995.
- [KLEINHANS2] Kleinhans CE, Harley RG, Diana G, McCulloch MD, Randelhoff MC, "A Drive Development Platform for Direct Portation of Control Algorithm Code between Simulations and Hardware Environment", Southern African Universities Power Engineering Conference, 1995, Pretoria, pp 77 - 80.
- [KLEINHANS3] Kleinhans CE, Harley RG, Diana G, McCulloch MD, "The Application of CASED as a Simulation Tool for Design and Analysis of Variable Speed Drives", Conference Record of the IEEE Industry Applications Society Annual Meeting, IAS, Oct. 1994, Denver, Colorado, ISBN 0-7803-1993-1, pp 750 - 757.
- [KLEINHANS4] Kleinhans CE, Diana G, Harley RG, McCulloch MD, "Analysing a CSI-Fed Field Orientated Controlled Induction Motor using a New Simulation Package CASED", South African Universities Power Engineering Conference, 1994, Capetown, pp 295 - 300.
- [KOCHER1] Kocher MJ, Steigerwald RL, "An AC-to-DC Converter with High Quality Input Waveforms", IEEE Transactions Industry Applications, Vol. IA-19, No. 4, July/August 1983, pp 586 - 599.
-

-
- [KOLAR1] Kolar JW, Ertl H, Edelmöser K, Zach FC, "Analysis of the Control Behaviour of a Bidirectional Three-Phase PWM Rectifier System", 4th European Conference on Power Electronics and Applications (EPE) Proceedings, Vol. 2, September 1991, pp 95 - 100.
- [MORSE1] Morse AS, "Global Stability of Parameter-Adaptive Control Systems", IEEE Transactions on Automatic Control, Vol. AC-25, No. 3, June 1980, pp 433 - 439.
- [NARENDRA1] Narendra KS, Lin YH, Valavani LS, "Stable Adaptive Controller Design, Part II: Proof of Stability", IEEE Transactions on Automatic Control, Vol. AC-25, No. 3, June 1980, pp 440 - 448.
- [NARENDRA2] Narendra KS, Kumpati S, Parthasarthy K, "Identification and Control of Dynamical Systems using Neural Networks", IEEE Transactions on Neural Networks, Vol. 1, No. 1, March 1990, pp 4 - 27.
- [NASAR1] Syed A. Nasar, "Theory and Problems of Electrical Machines and Electromechanics", Schaum's outline series, McGraw-Hill, pp 132, ISBN 0-07-045886-3.
- [OGATA1] Ogata K, "Discrete-Time Control Systems", Prentice-Hall, Englewood Cliffs, New Jersey, 1987, ISBN 0-13-216102-8.
- [O'KELLY1] O'Kelly D, Simmons S, "Introduction to Generalised Machine Theory", McGraw-Hill, 1986.
- [OOI1] Ooi BT, Salmon JC, Dixon JW, Kulkarni AB, "A 3-Phase Controlled Current PWM Converter with Leading Power Factor", Conference Records IEEE Industry Applications Society, 1985, pp 1008 - 1014.
-

-
- [OOI2] Ooi BT, Salmon JC, Dixon JW, Kulkarni AB, "A Three-Phase Controlled-Current PWM Converter with Leading Power Factor", IEEE Transactions on Industry Applications, Vol. IA-23, No. 1, January/February 1987, pp 78 - 84.
- [PINHEIRO1] Pinheiro H, Joos G, Khorasani K, "Neural Network-Based Controllers for Voltage Source PWM Front End Rectifiers", IEEE 21st International conference on Industrial Electronics, Control and Instrumentation, Orlando, 1995, pp 488 - 493.
- [RUMMELHART1] Rummelhart DE, Hinton GE, Williams RJ, "Learning Internal Representations by Error Propagation", Parallel Distributed Processing, 1986, (editors Rummelhart and McClelland JL), MIT Press, Massachusetts, 1986.
- [SAY1] Say MG, Taylor EO, "Direct Current Machines 2nd Edition", Pitman, London, 1986, ISBN 0-273-02457-4.
- [SEMIKRON1] Semikron, "Semidriver: Hybrid Double IGBT and Mosfet Driver SKHI21, SKHI22 Preliminary Data Sheet", Seminar Record, University of Natal, 1993, pp 61-68.
- [SIEMENS1] Siemens, "Plastic Fibre Components (PFC): A Cost Effective Solution for Optical Signal Transmission", Application Note 40, pp 14-121 to 14-127, Germany.
- [TI1] Texas Instruments, "TMS320C3x Users Guide", Texas Instruments Incorporated, 1997.
- [TI2] Texas Instruments, "TMS320C6711, Floating Point Digital Signal Processor", <http://www.ti.com/sc/docs/products/dsp/tms320c6711.html>
-

-
- [VANZYL1] Van Zyl A, Enslin JHR, Spee R, " Converter-Based Solutions to Power Quality Problems on Radial Distribution Lines", IEEE Transactions on Industry Applications, Vol. 32, No. 6, November/December 1996, pp 1323 - 1330.
- [WALKER1] Walker ML, Diana G, " Motion Control Card for the Implementation of Variable Speed Drives and Associated Controllers", Eighth South African Universities Power Engineering Conference, Pg 46-51, Potchefstroom, South Africa, January 1999.
- [WERNEKINCK1] Wernekinck E, Kawamura A, Hoft R, "A High Frequency AC/DC Converter with Unity Power Factor and Minimum Harmonic Distortion", IEEE Transactions on Power Electronics, Vol. 6, No. 3, July 1991, pp 364 - 370.
- [WISHART1] Wishart MT, "Identification and Control of Induction Machines Using Artificial Neural Networks", M.Sc. Eng. Thesis, Dep. Elect. Eng., Univ. Natal, Durban, South Africa, 1993.
- [WISHART2] Wishart MT, Harley RG, "Identification and Control of Induction Machines using Artificial Neural Networks", Conference Record of the 1993 IEEE Industry Applications Society Annual Meeting, Part 1, October 1993, pp 703 - 709.
- [WOOD1] Wood P, "Switching Power Converters", Van Nostrand Reinhold, New York, 1981, ISBN 0-442-24333-2.
- [WORTHMANN1] Worthmann CA, Diana G, "Neural Network Current Controlled Boost Rectifier", Proceedings of the Seventh Southern African Universities Power Engineering Conference 1998, Stellenbosch, pp 247 - 250.
-

-
- [WORTHMANN2] Worthmann CA, Diana G, "Neural Network Current Controller for a Boost Rectifier", IEEE International Symposium on Industrial Electronics Proceedings, Pretoria, South Africa, 1998, pp 234 - 239.
- [WU1] Wu R, Dewan SB, Slemon GR, "Analysis of a PWM AC to DC Voltage Source Converter under the Predicted Current Control with a Fixed Switching Frequency", IEEE Transactions on Industry Applications, Vol. 27, No. 4, July/August 1991, pp 756 - 764.
- [WU2] Wu R, Dewan SB, Slemon GR, "A PWM AC-to-DC Converter with Fixed Switching Frequency", IEEE Transactions on Industry Applications, Vol. 26, No. 5, September/October 1990, pp 880 - 885.
- [ZARGARI1] Zargari NR, Joos G, "An On-Line Operated Unity Power Factor PWM Rectifier for AC Drive Applications", Industry Applications Society 1994 29th Annual Meeting, Vol. 1, Denver, ISBN 0-7803-1993-1, pp 673 - 678.

APPENDIX A

DERIVATION OF EQUATIONS

A.1 Equations for the Boost Rectifier Power Conversion System in DQ Coordinates

The mathematical model of the FCR based power conversion system shown in Fig. A.1 can be derived by using Kirchoff's Voltage Law (KVL) and a power balance equation. The equations for the system were formulated in the three-phase a-b-c coordinate system using KVL. The direct Park's transformation, as discussed in chapter 3 (Eq. 3.12), is used to transform the equations into the d-q coordinate system.

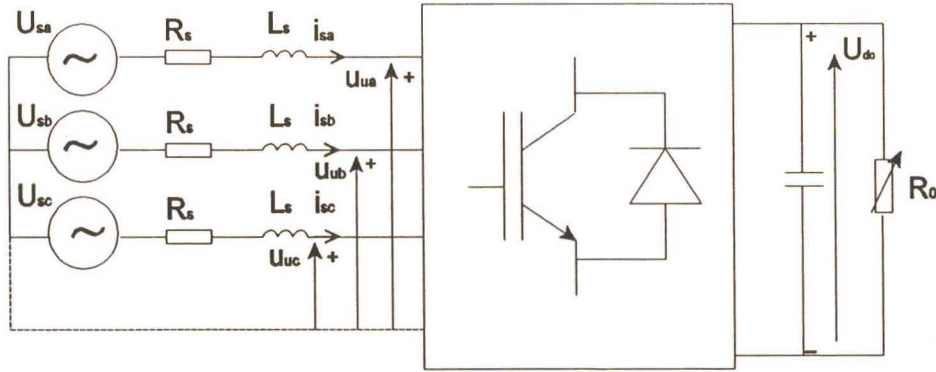


Fig. A.1 FCR-based power conversion system

A.1.1 Three-phase a-b-c reference frame mathematical model

The following equations describing the dynamics of the AC side of Fig. 3.14, may be written by applying Kirchoff's Voltage Law:

$$\begin{aligned}
L_s \frac{di_{sa}}{dt} + R_s i_{sa} &= u_{sa} - u_{ua} \\
L_s \frac{di_{sb}}{dt} + R_s i_{sb} &= u_{sb} - u_{ub} \\
L_s \frac{di_{sc}}{dt} + R_s i_{sc} &= u_{sc} - u_{uc}
\end{aligned} \tag{A.1}$$

The supply voltages are assumed to be:

$$\begin{aligned}
u_{sa} &= U_m \cos(\omega t) \\
u_{sb} &= U_m \cos(\omega t - \frac{2\pi}{3}) \\
u_{sc} &= U_m \cos(\omega t + \frac{2\pi}{3})
\end{aligned} \tag{A.2}$$

Eq. (A.2) may be re-written in vector-matrix form as follows:

$$\underline{Z} \cdot \dot{\underline{x}} = \underline{A} \cdot \underline{x} + \underline{B} \cdot \underline{u}_s + \underline{C} \cdot \underline{u}_u \tag{A.3}$$

where:

$$\begin{aligned}
\underline{x} &= [i_{sa} \quad i_{sb} \quad i_{sc}]^T, \underline{u}_s = [u_{sa} \quad u_{sb} \quad u_{sc}]^T \\
\underline{u}_u &= [u_{ua} \quad u_{ub} \quad u_{uc}]^T
\end{aligned} \tag{A.4}$$

and:

$$\begin{aligned}
\underline{A} &= \begin{bmatrix} -R_s & 0 & 0 \\ 0 & -R_s & 0 \\ 0 & 0 & -R_s \end{bmatrix}, \quad \underline{B} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\
\underline{C} &= \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}, \quad \underline{Z} = \begin{bmatrix} L_s & 0 & 0 \\ 0 & L_s & 0 \\ 0 & 0 & L_s \end{bmatrix}
\end{aligned} \tag{A.5}$$

Assuming this to be a lossless converter, a further equation describing the dynamics of the system may be written by equating the instantaneous AC power into the FCR to the instantaneous DC output power to obtain:

$$u_{ua}i_{sa} + u_{ub}i_{sb} + u_{uc}i_{sc} = \frac{1}{2}C \frac{du_{dc}^2}{dt} + \frac{u_{dc}^2}{R_0} \quad (\text{A.6})$$

Eqs. (A.1) and (A.6), which provide a complete mathematical description of the dynamics of the system in the a-b-c reference frame, are transformed into the d-q reference frame, using the Park's transformation.

Transformation of equations from a-b-c to d-q reference frame

Eq. (A.1) is transformed into the synchronously rotating d-q reference frame by defining:

$$\begin{aligned} \underline{x}_r &= \underline{P} \cdot \underline{x}, \\ \underline{u} &= \underline{P} \cdot \underline{u}_s, \\ \underline{u}_{ur} &= \underline{P} \cdot \underline{u}_u \end{aligned} \quad (\text{A.7})$$

where:

$$\begin{aligned} \underline{x}_r &= \begin{bmatrix} i_{sd} & i_{sq} & i_{sz} \end{bmatrix}^T, \underline{u}_{sr} = \begin{bmatrix} u_{sd} & u_{sq} & u_{sz} \end{bmatrix}^T \\ \underline{u}_{ur} &= \begin{bmatrix} u_{ud} & u_{uq} & u_{uz} \end{bmatrix}^T \end{aligned} \quad (\text{A.8})$$

and \underline{P} is the Park's transformation matrix defined in Chapter 3 section 3.6.2. Substituting Eq. (A.7) into Eq. (A.3) and rearranging yields:

$$\underline{P} \cdot \underline{Z} \cdot \underline{P}^{-1} \cdot \dot{\underline{x}}_r = (\underline{P} \cdot \underline{A} \cdot \underline{P}^{-1} - \underline{P} \cdot \underline{Z} \cdot \underline{P}^{-1}) \cdot \underline{x}_r + \underline{P} \cdot \underline{B} \cdot \underline{P}^{-1} \cdot \underline{u}_{sr} + \underline{P} \cdot \underline{C} \cdot \underline{P}^{-1} \cdot \underline{u}_{ur} \quad (\text{A.9})$$

which, when simplified, yields:

$$\begin{bmatrix} L_s & 0 & 0 \\ 0 & L_s & 0 \\ 0 & 0 & L_s \end{bmatrix} \begin{bmatrix} \dot{i}_{sd} \\ \dot{i}_{sq} \\ \dot{i}_{sz} \end{bmatrix} = \begin{bmatrix} -R_s & \omega L_s & 0 \\ -\omega L_s & -R_s & 0 \\ 0 & 0 & -R_s \end{bmatrix} \begin{bmatrix} i_{sd} \\ i_{sq} \\ i_{sz} \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_{sd} \\ u_{sq} \\ u_{sz} \end{bmatrix} + \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} u_{ud} \\ u_{uq} \\ u_{uz} \end{bmatrix} \quad (\text{A.10})$$

as the mathematical model of the FCR-based power conversion system in Fig. A.1, in d-q coordinates.

Expanding and rearranging Eq. (A.10) yields:

$$\begin{aligned} u_{ud} &= u_{sd} - L_s \frac{di_{sd}}{dt} - R_s i_{sd} + \omega L_s i_{sq} \\ u_{uq} &= u_{sq} - L_s \frac{di_{sq}}{dt} - R_s i_{sq} - \omega L_s i_{sd} \end{aligned} \quad (\text{A.11})$$

which under steady-state conditions become:

$$\begin{aligned} u_{ud} &= u_{sd} - R_s i_{sd} + \omega L_s i_{sq} \\ u_{uq} &= u_{sq} - R_s i_{sq} - \omega L_s i_{sd} \end{aligned} \quad (\text{A.12})$$

Eq. (A.6) is transformed into the d-q reference frame by applying the Park's transformation to the left hand side of the equation and simplifying, to yield the following expression for power balance in the d-q reference frame during rectification:

$$\frac{3}{2} (u_{ud} \cdot i_{sd} + u_{uq} \cdot i_{sq}) = \frac{1}{2} C \frac{du_{dc}^2}{dt} + \frac{u_{dc}^2}{R_0} \quad (\text{A.13})$$

Eq. (A.13) may be expressed in terms of the AC system parameters R_s , L_s , u_{sd} , and u_{sq} by substituting the expression for u_{ud} , and u_{uq} in Eq. (A.11) into Eq. (A.13) and simplifying to yield:

$$\frac{3}{2}(u_{sd} \cdot i_{sd} - \frac{1}{2} L_s \frac{d(i_{sd}^2 + i_{sq}^2)}{dt} - (i_{sd}^2 + i_{sq}^2) R_s + u_{sq} \cdot i_{sq}) = \frac{1}{2} C \frac{du_{dc}^2}{dt} + \frac{u_{dc}^2}{R_0} \quad (\text{A.14})$$

The dynamic model of the power conversion system can be modeled fully by Eq. (A.11) and (A.14).

A.2 Linearisation of the dynamic power balance equation during rectification

Eq. (A.14) is a non-linear relationship between i_{sd} , i_{sq} and u_{dc} (as discussed in Chapter 2). Since the design of the voltage controller requires a linear relationship between these quantities, Eq. (A.14) is linearised around the operating point (I_{sdo} , I_{sqo} , U_{dco}) by defining [OGATA1], [CHATHURY1], [CHATHURY2]:

$$\begin{aligned} i_{sd} &= I_{sdo} + \Delta i_{sd} \\ i_{sq} &= I_{sqo} + \Delta i_{sq} \\ u_{dc} &= U_{dco} + \Delta u_{dc} \\ u_{sd} &= U_{sdo} \\ u_{sq} &= U_{sqo} \end{aligned} \quad (\text{A.15})$$

where Δi_{sd} , Δi_{sq} and Δu_{dc} are the perturbations of i_{sd} , i_{sq} and u_{dc} respectively about the operating point and the operating point quantities are determined under steady-state conditions. For the steady-state unity power factor condition, $U_{sqo}=0$ and $I_{sqo}=0$, in addition, steady-state power balance implies that:

$$\frac{3}{2}(U_{sdo} I_{sdo} - I_{sdo}^2 R_s) = \frac{U_{dco}^2}{R_0} \quad (\text{A.16})$$

Therefore, substituting Eqs. (A.15) and (A.16) into Eq. (A.14) and simplifying yields:

$$\begin{aligned} & \frac{3}{2} \left(U_{sdo} \Delta i_{sd} - (2\Delta i_{sd} I_{sdo} + \Delta i_{sd}^2 + \Delta i_{sq}^2) R_s - \frac{1}{2} L_s \frac{d(2\Delta i_{sd} I_{sdo} + \Delta i_{sd}^2 + \Delta i_{sq}^2)}{dt} \right) \\ &= \frac{1}{2} C \frac{d(2\Delta u_{dc} U_{dco} + \Delta u_{dc}^2)}{dt} + \frac{2\Delta u_{dc} U_{dco} + \Delta u_{dc}^2}{R_0} \end{aligned} \quad (A.17)$$

For small perturbations about the operating point, Eq. (A.17) approximates to:

$$\frac{3}{2} \left(U_{sdo} \Delta i_{sd} - 2\Delta i_{sd} I_{sdo} R_s - L_s I_{sdo} \frac{d\Delta i_{sd}}{dt} \right) = C U_{dco} \frac{d\Delta u_{dc}}{dt} + \frac{2\Delta u_{dc} U_{dco}}{R_0} \quad (A.18)$$

which is a linearised version of the dynamic power balance equation about the operating point (I_{sdo} , I_{sqo} , U_{dco}).

A.3 NARMAX model of the Boost Rectifier

The NARMAX (Nonlinear AutoRegressive Moving Average with eXogeneous inputs) model of the boost rectifier was derived, assuming that the rectifier may be modeled as an induction motor operating in the regenerative mode, as discussed in [SAY1] (pp 250 - 265). This will be quantified in the following subsection. This section begins by presenting the mathematical equations of the hybrid two axis continuous time model of the electrical dynamics of the induction motor. The electrical dynamics of the generator and the mechanical dynamics of its shaft and load are coupled by the electromagnetic torque equation; the torque equation is also presented here, together with the equations describing a simple mechanical load model. In Wishart [WISHART1], the equations of the continuous time electrical model are used to derive the discrete time NARMAX input/output model of the electrical dynamics of the induction machine, for identification using ANN's.

A.3.1 Continuous Time Electrical Model of the Induction Motor

From Chapter 4, the two axis continuous time model of the electrical dynamics of an n-pole, distributed winding, three-phase induction motor is derived [O'KELLY1] by simplifying its structure to that of an equivalent two-pole motor with three symmetrically arranged, balanced and, concentrated windings in each of the stator and rotor circuits (assuming no saturation). The analysis can be further simplified by resolving the three phase electrical variables into two orthogonal components by projecting them onto a set of orthogonal axes, namely the d (direct) and q (quadrature) axes, which rotate at an arbitrary speed ω_{dq} with respect to the stator windings of the motor, as shown in Fig. A.2.

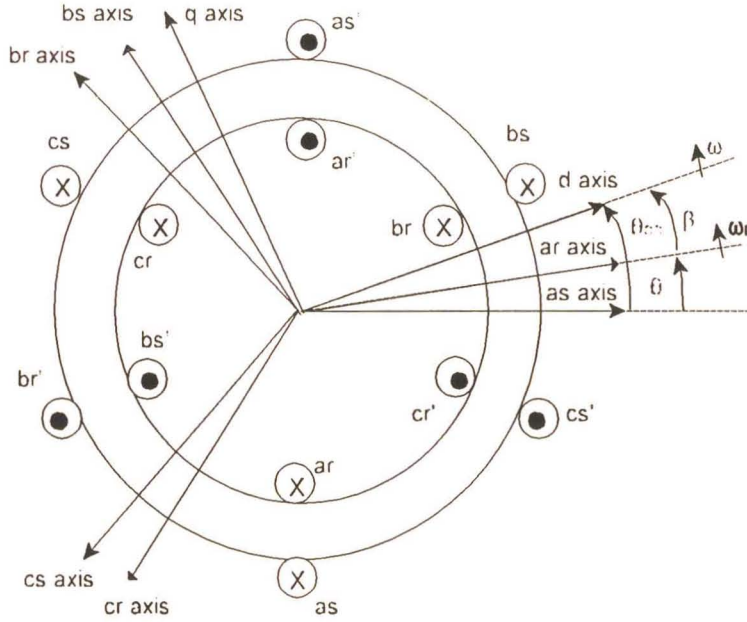


Fig. A.2 The dq axis reference frame

The motor can be modeled in this reference frame by Eq. (A19).

$$\begin{aligned}
 v_{ds} &= R_1 i_{ds} + p \lambda_{ds} - \omega_e \lambda_{qs} \\
 v_{qs} &= R_1 i_{qs} + p \lambda_{qs} + \omega_e \lambda_{ds} \\
 v_{dr} &= R_2 i_{dr} + p \lambda_{dr} - s \omega_e \lambda_{qr} = 0 \\
 v_{qr} &= R_2 i_{qr} + p \lambda_{qr} + s \omega_r \lambda_{dr} = 0
 \end{aligned} \tag{A.19}$$

The factor difference between the stator field speed and rotor speed, as shown in Eq. (A.20), is known as slip.

$$S = \frac{\omega_e - \omega_r}{\omega_e} \quad (\text{A.20})$$

In the motoring reference frame, slip is taken as positive, therefore, a regenerating motor in the identical reference frame (motoring reference frame), the slip would be negative [SAY1]. Nasar [NASAR1] shows that an inverter is modeled as a motor in the motoring reference frame, and that a converter can be modeled as an inverter in regeneration (i.e. a motor in regeneration). Thus a boost rectifier can be modeled as a motor in regenerative mode, in the motoring reference frame [NASAR1], [SAY1]. Therefore, the boost rectifier may be modeled in the motoring reference frame, using the motor model, but with a slip of -s.

Thus in the dq reference frame, the voltage equations of the boost rectifier may be written as [O'KELLY1]:

$$\begin{aligned} v_{ds} &= R_1 i_{ds} + p \lambda_{ds} - \omega_e \lambda_{qs} \\ v_{qs} &= R_1 i_{qs} + p \lambda_{qs} + \omega_e \lambda_{ds} \\ v_{dr} &= R_2 i_{dr} + p \lambda_{dr} + s \omega_e \lambda_{qr} = 0 \\ v_{qr} &= R_2 i_{qr} + p \lambda_{qr} - s \omega_e \lambda_{dr} = 0 \end{aligned} \quad (\text{A.21})$$

If using negative s in Eq. (A.20), and substituting into Eq. (A.21), the model becomes:

$$\begin{aligned} v_{ds} &= R_1 i_{ds} + p \lambda_{ds} - \omega_e \lambda_{qs} \\ v_{qs} &= R_1 i_{qs} + p \lambda_{qs} + \omega_e \lambda_{ds} \\ v_{dr} &= R_2 i_{dr} + p \lambda_{dr} + s(\omega_e - \omega_r) \lambda_{qr} = 0 \\ v_{qr} &= R_2 i_{qr} + p \lambda_{qr} - s(\omega_e - \omega_r) \lambda_{dr} = 0 \end{aligned} \quad (\text{A.22})$$

where: v_{ds}, v_{qs} are the d and q axis stator voltages (in Volts),
 v_{dr}, v_{qr} are the d and q axis rotor voltages (in Volts),
 i_{ds}, i_{qs} are the d and q axis stator currents (in Amperes),

i_{dr}, i_{qr}	are the d and q axis rotor currents (in Amperes),
$\lambda_{ds}, \lambda_{qs}$	are the d and q axis stator flux linkages (in Weber-turns),
$\lambda_{dr}, \lambda_{qr}$	are the d and q axis rotor flux linkages (in Weber-turns),
ω	is the arbitrary angular speed rotation (in rad/s) of the d,q axes,
ω_r	is the rotor speed (in rad/s),
R_s, R_r	are the stator and rotor resistance (in Ohms), and
p	is the differential operator d/dt

The currents and flux linkages of the four voltage equations in Eq. (B.19) (i.e. $i_{ds}, i_{qs}, i_{dr}, i_{qr}$ and $\lambda_{ds}, \lambda_{qs}, \lambda_{dr}, \lambda_{qr}$) represent eight electrical states, of which only four are independent, since these currents and flux linkages are related by:

$$\begin{bmatrix} \lambda_{ds} \\ \lambda_{qs} \\ \lambda_{dr} \\ \lambda_{qr} \end{bmatrix} = \begin{bmatrix} L_{11} & 0 & L_m & 0 \\ 0 & L_{11} & 0 & L_m \\ L_m & 0 & L_{22} & 0 \\ 0 & L_m & 0 & L_{22} \end{bmatrix} \begin{bmatrix} i_{ds} \\ i_{qs} \\ i_{dr} \\ i_{qr} \end{bmatrix} \quad (\text{A.23})$$

where: L_m denotes the stator to rotor mutual inductance (in Henry),

L_{11}, L_{22} are the stator to rotor self inductances (in Henry), and

$$L_{11} = L_l + L_m.$$

Thus the two-axis electrical state space model of the induction motor can be derived in terms of any set of four independent states. In drive applications, it is often most convenient to choose the state vector as $[i_{ds}, i_{qs}, \lambda_{dr}, \lambda_{qr}]^T$. Due to the presence of both currents and flux linkages in this state vector, the resulting two-axis electrical state space model is referred to as the hybrid dq model of the induction motor and the state vector is accordingly denoted by \underline{h} .

Now substituting Eq. (A.23) into Eq. (A.22), to eliminate the dependent states (i.e. $i_{dr}, i_{qr}, \lambda_{ds}, \lambda_{qs}$) and rearranging in terms of $p\underline{h}$ gives:

$$p\underline{h} = -L^{-1}(R - \omega F - \omega_r G)\underline{h} + L^{-1}\underline{v} \quad (\text{A.24})$$

where

$$L^{-1} = \begin{bmatrix} \frac{1}{\sigma L_{11}} & 0 & -\frac{L_m}{\sigma L_{11} L_{22}} & 0 \\ 0 & \frac{1}{\sigma L_{11}} & 0 & -\frac{L_m}{\sigma L_{11} L_{22}} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad F = \begin{bmatrix} 0 & -\sigma L_{11} & 0 & -\frac{L_m}{L_{22}} \\ \sigma L_{11} & 0 & \frac{L_m}{L_{22}} & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{bmatrix},$$

$$R = \begin{bmatrix} R_1 & 0 & 0 & 0 \\ 0 & R_1 & 0 & 0 \\ -\frac{R_2 L_m}{L_{22}} & 0 & \frac{R_2}{L_{22}} & 0 \\ 0 & -\frac{R_2 L_m}{L_{22}} & 0 & \frac{R_2}{L_{22}} \end{bmatrix}, \quad G = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix},$$

$$\sigma = 1 - \frac{L_m^2}{L_{11} L_{22}}, \quad \underline{h} = [i_{ds} \quad i_{qs} \quad \lambda_{dr} \quad \lambda_{qr}]^T$$

$$\underline{v} = [v_{ds} \quad v_{qs} \quad 0 \quad 0]^T$$

Rewriting Eq. (A.24) in compact form yields:

$$p \underline{h} = A \underline{h} + B \underline{v} \quad (\text{A.25})$$

where

$$A = -L^{-1}(R - \omega F - \omega_r G)$$

$$B = L^{-1}$$

Evaluating matrices A and B gives:

$$p\mathbf{h} = \begin{bmatrix} -\frac{1}{\tau_s} & \omega & \frac{\xi}{\tau_r} & \xi\omega_r \\ -\omega & -\frac{1}{\tau_s} & -\xi\omega_r & -\frac{1}{\tau_r} \\ R_{22} & 0 & -\frac{1}{\tau_r} & (\omega_r - \omega) \\ 0 & R_{22} & -(\omega_r - \omega) & -\frac{1}{\tau_r} \end{bmatrix} \mathbf{h} + \frac{1}{\sigma L_{11}} \mathbf{v} \quad (\text{A.26})$$

where

$$\tau_s = \left[\frac{R_1}{\sigma L_{11}} + \frac{R_2 L_m^2}{\sigma L_{11} L_{22}^2} \right]^{-1}, \quad \tau_r = \frac{L_{22}}{R_2}, \quad R_{22} = \frac{L_m R_2}{L_{22}}, \quad \xi = \frac{L_m}{\sigma L_{11} L_{22}}$$

Up to this point the dq axes have been assumed to rotate at an arbitrary speed ω with respect to the stator windings of the motor. However, the two most common choices for ω are the speed of rotation of the abc voltage space vector ω_e and zero (i.e. $\omega = \omega_e$ or $\omega = 0$). For these two values of ω , the two-axis reference frame is known as the synchronous and stationary (or $\alpha\beta$) reference frame respectively. In the main text of the thesis, the synchronous reference frame is simply referred to as the dq reference frame, and the stationary reference frame is referred to as the $\alpha\beta$ reference frame.

The synchronous or dq reference frame

Substituting ω_e into Eq. (A.26) for ω gives:

$$p\mathbf{h}^{dq} = \begin{bmatrix} -\frac{1}{\tau_s} & \omega_e & \frac{\xi}{\tau_r} & \xi\omega_r \\ -\omega_e & -\frac{1}{\tau_s} & \xi\omega_r & -\frac{1}{\tau_r} \\ R_{22} & 0 & -\frac{1}{\tau_r} & (\omega_r - \omega_e) \\ 0 & R_{22} & -(\omega_r - \omega_e) & -\frac{1}{\tau_r} \end{bmatrix} \mathbf{h}^{dq} + \frac{1}{\sigma L_{11}} \mathbf{v}^{dq} \quad (\text{A.27})$$

A consequence of reducing the structure of an n -pole machine (where $n \in \{4, 6, 8 \dots\}$) to that of an equivalent two-pole machine is that the rotor speed is no longer equal to the mechanical speed of rotation of the motor shaft ω_{mech} but is given instead by $\omega = n \omega_{mech}/2$. For this reason, ω is often referred to as the electrical rotor speed.

Stationary or $\alpha\beta$ reference frame

Substituting 0 into Eq. (A.26) for ω gives:

$$p\mathbf{h}^{\alpha\beta} = \begin{bmatrix} -\frac{1}{\tau_s} & 0 & \frac{\xi}{\tau_r} & \xi\omega_r \\ 0 & -\frac{1}{\tau_s} & -\xi\omega_r & -\frac{1}{\tau_r} \\ R_{22} & 0 & -\frac{1}{\tau_r} & \omega_r \\ 0 & R_{22} & -\omega_r & -\frac{1}{\tau_r} \end{bmatrix} \mathbf{h}^{\alpha\beta} + \frac{1}{\sigma L_{11}} \mathbf{v}^{\alpha\beta} \quad (\text{A.28})$$

The $\alpha\beta$ reference frame model does not contain the nonlinear terms which arise due to the multiplication of \mathbf{h} and ω_e . The rotor speed ω_r , however is common to both the dq and $\alpha\beta$ representations of the hybrid two-axis model and provides the coupling between the electrical and mechanical dynamics of the generator.

A.3.2 Derivation of the Electrical NARMAX Model of the Boost rectifier

The first step of the derivation of [WISHART1] of the discrete time NARMAX model of the electrical dynamics of the induction motor, is the discretisation of the two-axis hybrid model of Eq. (A.27). Wishart assumes a high enough sampling rate so that discretisation of Eq. (A.27) by means of Eulers forward difference method [ASTROM1] produces accurate results. Discretising Eq. (A.27) in this way gives:

$$\underline{h}(k+1) = \phi(k)\underline{h}(k) + \Gamma \underline{v}(k) \quad (\text{A.29})$$

where

$$\phi(k) = \begin{bmatrix} 1 - \frac{T}{\tau_s} & T\omega_e(k) & \frac{T\xi}{\tau_r} & T\xi\omega_r(k) \\ -T\omega_e(k) & 1 - \frac{T}{\tau_s} & -T\xi\omega_r(k) & \frac{T\xi}{\tau_r} \\ TR_{22} & 0 & 1 - \frac{T}{\tau_r} & -T(\omega_e(k) - \omega_r(k)) \\ 0 & TR_{22} & T(\omega_e(k) - \omega_r(k)) & 1 - \frac{T}{\tau_r} \end{bmatrix}, \quad \Gamma = \frac{T}{\sigma L_{11}}$$

and T is the sampling interval.

The electromagnetic torque equation is as follows:

$$T_{em} = -\frac{L_m}{L_{22}}(i_{qs}(k)\lambda_{dr}(k) - i_{ds}(k)\lambda_{qr}(k)) \quad (\text{A.30})$$

Now substituting $\omega_e = 0$ into Eq. (A.29) gives the stationary or $\alpha\beta$ reference frame representation as:

$$\underline{h}^{a\beta}(k) = \begin{bmatrix} 1 - \frac{T}{\tau_s} & 0 & \frac{T\xi}{\tau_r} & T\xi\omega_r(k) \\ 0 & 1 - \frac{T}{\tau_s} & -T\xi\omega_r(k) & \frac{T\xi}{\tau_r} \\ TR_{22} & 0 & 1 - \frac{T}{\tau_r} & T\omega_r(k) \\ 0 & TR_{22} & -T\omega_r(k) & 1 - \frac{T}{\tau_r} \end{bmatrix} \underline{h}^{a\beta}(k) + \frac{T}{\sigma L_{11}} \underline{v}^{a\beta}(k) \quad (\text{A.31})$$

Expanding Eq. (A.31) into individual scalar equations gives:

$$i_{as}(k+1) = \alpha_1 i_{as}(k) + \alpha_2 \lambda_{ar}(k) + \alpha_3 \omega_r(k) \lambda_{\beta r}(k) + \alpha_6 v_{ar}(k) \quad (\text{A.32})$$

$$i_{\beta s}(k+1) = \alpha_1 i_{\beta s}(k) + \alpha_2 \lambda_{\beta r}(k) - \alpha_3 \omega_r(k) \lambda_{ar}(k) + \alpha_6 v_{\beta r}(k) \quad (\text{A.33})$$

$$i_{ar}(k+1) = \alpha_4 \lambda_{ar}(k) + \alpha_5 i_{ar}(k) + T\omega_r(k) \lambda_{\beta r}(k) \quad (\text{A.34})$$

$$i_{\beta r}(k+1) = \alpha_4 \lambda_{\beta r}(k) + \alpha_5 i_{\beta r}(k) + T\omega_r(k) \lambda_{ar}(k) \quad (\text{A.35})$$

where

$$\alpha_1 = 1 - \frac{T}{\tau_s}$$

$$\alpha_2 = T \frac{\xi}{\tau_r}$$

$$\alpha_3 = T\xi$$

$$\alpha_4 = 1 - \frac{T}{\tau_r}$$

$$\alpha_5 = TR_{22}$$

$$\alpha_6 = \frac{T}{\sigma L_{11}}$$

Solving Eq. (A.32) for $\lambda_{\beta r}(k)$ gives:

$$\lambda_{\beta r}(k) = \frac{i_{\alpha r}(k+1) - \alpha_1 i_{\alpha s}(k) - \alpha_2 \lambda_{\alpha r}(k) - \alpha_6 v_{\alpha s}(k)}{\alpha_3 \omega_r(k)} \quad (\text{A.36})$$

Similarly, solving Eq. (A.33) for $\lambda_{\alpha r}(k)$ gives:

$$\lambda_{\alpha r}(k) = \frac{-i_{\beta r}(k+1) + \alpha_1 i_{\beta s}(k) + \alpha_2 \lambda_{\beta r}(k) + \alpha_6 v_{\beta s}(k)}{\alpha_3 \omega_r(k)} \quad (\text{A.37})$$

Now, substituting Eq. (A.37) into Eq. (A.36) and factorising gives:

$$\lambda_{\beta r}(k) = \frac{\alpha_3 \omega_r(k)(i_{\alpha s}(k+1) - \alpha_1 i_{\alpha s}(k) - \alpha_6 v_{\alpha s}(k)) + \alpha_2(i_{\beta s}(k+1) - \alpha_1 i_{\beta s}(k) - \alpha_6 v_{\beta s}(k))}{(\alpha_3 \omega_r(k))^2 + \alpha_2^2} \quad (\text{A.38})$$

Similarly, substituting Eq. (A.36) into Eq. (A.37) and factorising gives:

$$\lambda_{\alpha r}(k) = \frac{\alpha_3 \omega_r(k)(i_{\beta s}(k+1) - \alpha_1 i_{\beta s}(k) - \alpha_6 v_{\beta s}(k)) + \alpha_2(i_{\alpha s}(k+1) - \alpha_1 i_{\alpha s}(k) - \alpha_6 v_{\alpha s}(k))}{(\alpha_3 \omega_r(k))^2 + \alpha_2^2} \quad (\text{A.39})$$

Now, substituting Eq. (A.38) and (A.39) into Eq. (A.34) gives:

$$\begin{aligned} \lambda_{\alpha r}(k+1) = & \alpha_4 \frac{\alpha_3 \omega_r(k)(-i_{\beta s}(k+1) + \alpha_1 i_{\beta s}(k) + \alpha_6 v_{\beta s}(k)) + \alpha_2(i_{\alpha s}(k+1) - \alpha_1 i_{\alpha s}(k) - \alpha_6 v_{\alpha s}(k))}{(\alpha_3 \omega_r(k))^2 + \alpha_2^2} \\ & + T \omega_r(k) \frac{\alpha_3 \omega_r(k)(i_{\alpha s}(k+1) - \alpha_1 i_{\alpha s}(k) - \alpha_6 v_{\alpha s}(k)) + \alpha_2(i_{\beta s}(k+1) - \alpha_1 i_{\beta s}(k) - \alpha_6 v_{\beta s}(k))}{(\alpha_3 \omega_r(k))^2 + \alpha_2^2} \\ & + \alpha_5 i_{\alpha s}(k) \end{aligned} \quad (\text{A.40})$$

Similarly, substituting Eqs. (A.38) and (A.39) into Eq. (A.35) gives:

$$\begin{aligned} \lambda_{ar}(k+1) = & \alpha_4 \frac{\alpha_3 \omega_r(k)(i_{as}(k+1) - \alpha_1 i_{as}(k) - \alpha_6 v_{as}(k)) + \alpha_2(i_{\beta s}(k+1) - \alpha_1 i_{\beta s}(k) - \alpha_6 v_{\beta s}(k))}{(\alpha_3 \omega_r(k))^2 + \alpha_2^2} \\ & + T \omega_r(k) \frac{\alpha_3 \omega_r(k)(-i_{\beta s}(k+1) + \alpha_1 i_{\beta s}(k) + \alpha_6 v_{\beta s}(k)) + \alpha_2(i_{as}(k+1) - \alpha_1 i_{as}(k) - \alpha_6 v_{as}(k))}{(\alpha_3 \omega_r(k))^2 + \alpha_2^2} \\ & + \alpha_1 i_{\beta s}(k) \end{aligned} \quad (\text{A.41})$$

The alpha stator current (Eq. (A.32)) was then shifted forward by one sampling period, to move it into the $(k+1)$ time period, as shown in Eq. (A.42), to allow for direct substitution of Eq. (A.40) and (A.41) into Eq. (A.32).

$$i_{as}(k+2) = \alpha_1 i_{as}(k+1) + \alpha_2 \lambda_{ar}(k+1) + \alpha_3 \omega_r(k+1) \lambda_{\beta r}(k+1) + \alpha_6 v_{as}(k+1) \quad (\text{A.42})$$

Eqs. (A.40) and (A.41) were then substituted into (A.42), and delaying by one sampling period to return to the $(k+1)$ time period to give:

$$\begin{aligned} i_{as}(k+1) = & \alpha_1 i_{as}(k) + \alpha_6 v_{as}(k) + \alpha_2 \alpha_1 i_{as}(k-1) + \alpha_3 \omega_r(k) \alpha_1 i_{\beta s}(k-1) \\ & - \alpha_2 \alpha_4 \frac{\alpha_3 \omega_r(k-1)(-i_{\beta s}(k) + \alpha_1 i_{\beta s}(k-1) + \alpha_6 v_{\beta s}(k-1)) + \alpha_2(i_{as}(k) - \alpha_1 i_{as}(k-1) - \alpha_6 v_{as}(k-1))}{(\alpha_3 \omega_r(k-1))^2 + \alpha_2^2} \\ & + T \alpha_2 \omega_r(k-1) \frac{\alpha_3 \omega_r(k-1)(i_{as}(k) - \alpha_1 i_{as}(k-1) - \alpha_6 v_{as}(k-1)) + \alpha_2(i_{\beta s}(k) - \alpha_1 i_{\beta s}(k-1) - \alpha_6 v_{\beta s}(k-1))}{(\alpha_3 \omega_r(k-1))^2 + \alpha_2^2} \\ & - \alpha_3 \omega_r(k) \alpha_4 \frac{\alpha_3 \omega_r(k-1)(i_{as}(k) - \alpha_1 i_{as}(k-1) - \alpha_6 v_{as}(k-1)) + \alpha_2(i_{\beta s}(k) - \alpha_1 i_{\beta s}(k-1) - \alpha_6 v_{\beta s}(k-1))}{(\alpha_3 \omega_r(k-1))^2 + \alpha_2^2} \\ & - T \alpha_3 \omega_r(k) \omega_r(k-1) \frac{\alpha_3 \omega_r(k-1)(-i_{\beta s}(k) + \alpha_1 i_{\beta s}(k-1) + \alpha_6 v_{\beta s}(k-1)) + \alpha_2(i_{as}(k) - \alpha_1 i_{as}(k-1) - \alpha_6 v_{as}(k-1))}{(\alpha_3 \omega_r(k-1))^2 + \alpha_2^2} \end{aligned} \quad (\text{A.43})$$

Similarly, forward shifting the beta stator current Eq. (A.33) by one sampling period gives:

$$i_{\beta s}(k+2) = \alpha_1 i_{\beta s}(k+1) + \alpha_2 \lambda_{\beta r}(k+1) + \alpha_3 \omega_r(k+1) \lambda_{ar}(k+1) + \alpha_6 v_{\beta s}(k+1) \quad (\text{A.44})$$

Substituting Eqs. (A.40) and (A.41) into (A.43), and delaying by one sampling period gives:

$$\begin{aligned}
 i_{\beta s}(k+1) = & \alpha v_{\beta s}(k) + \alpha_2 \alpha s i_{\beta s}(k-1) + \alpha_3 \omega_r(k) \alpha s i_{\alpha s}(k-1) \\
 & - \alpha_2 \alpha_4 \frac{\alpha_3 \omega_r(k-1)(i_{\alpha s}(k) - \alpha v_{\alpha s}(k-1) - \alpha_6 v_{\alpha s}(k-1)) + \alpha_2(i_{\beta s}(k) - \alpha v_{\beta s}(k-1) - \alpha_6 v_{\beta s}(k-1))}{(\alpha_3 \omega_r(k-1))^2 + \alpha_2^2} \\
 & + T \alpha_2 \omega_r(k-1) \frac{\alpha_3 \omega_r(k-1)(-i_{\beta s}(k) + \alpha v_{\beta s}(k-1) + \alpha_6 v_{\beta s}(k-1)) + \alpha_2(i_{\alpha s}(k) - \alpha v_{\alpha s}(k-1) - \alpha_6 v_{\alpha s}(k-1))}{(\alpha_3 \omega_r(k-1))^2 + \alpha_2^2} \\
 & - \alpha_3 \omega_r(k) \alpha_4 \frac{\alpha_3 \omega_r(k-1)(-i_{\beta s}(k) + \alpha v_{\beta s}(k-1) + \alpha_6 v_{\beta s}(k-1)) + \alpha_2(i_{\alpha s}(k) - \alpha v_{\alpha s}(k-1) - \alpha_6 v_{\alpha s}(k-1))}{(\alpha_3 \omega_r(k-1))^2 + \alpha_2^2} \\
 & - T \alpha_3 \omega_r(k) \omega_r(k-1) \frac{\alpha_3 \omega_r(k-1)(i_{\alpha s}(k) - \alpha v_{\alpha s}(k-1) - \alpha_6 v_{\alpha s}(k-1)) + \alpha_2(i_{\beta s}(k) - \alpha v_{\beta s}(k-1) - \alpha_6 v_{\beta s}(k-1))}{(\alpha_3 \omega_r(k-1))^2 + \alpha_2^2}
 \end{aligned} \tag{A.45}$$

Finally, Eqs. (A.43) and (A.45) may be written in compact vector form as:

$$\underline{i}(k+1) = \underline{f}_{el}(\underline{i}(k), \underline{i}(k-1), -\omega(k), -\omega(k-1), \underline{v}(k-1)) + c_v \underline{v}(k) \tag{A.46}$$

where

$\underline{i}(k)$ is the $\alpha\beta$ stator current vector $[i_\alpha i_\beta]^T$ at the time (k),

$\omega_r(k)$ is the shaft speed at time (k),

$\underline{v}(k)$ is the $\alpha\beta$ stator voltage applied at time (k),

c_v is the voltage constant defined by the motor parameters and is equal to α_6 , and

$\underline{f}_{el}(\cdot)$ is the vector function $[f_{el\alpha} f_{el\beta}]^T$ defined by the motor parameters as shown in Eqs. (A.43) and (A.45) respectively.

In order for Eq. (A.46) to be identified by an ANN, it is necessary to normalise the range of each variable.

A.3.3 Root-Locus Angle and Magnitude Conditions

The derivation of the voltage controller in Chapter 4 requires the use of the angle and magnitude conditions for a closed loop transfer function. Consider the system shown in Fig. A.3.

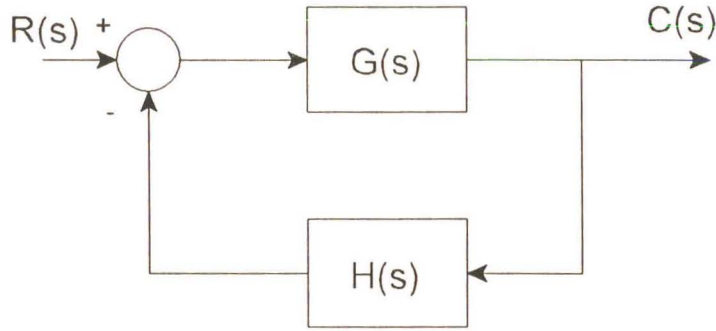


Fig. A.3 Control system

The closed-loop transfer function is:

$$\frac{C(s)}{R(s)} = \frac{G(s)}{1 + G(s)H(s)} \quad (\text{A.47})$$

The characteristic equation for this closed-loop system is obtained by setting the denominator of the right side of Eqn. (A.47) equal to zero. That is:

$$1 + G(s)H(s) = 0$$

or

$$G(s)H(s) = -1 \quad (\text{A.48})$$

Here it is assumed that $G(s)H(s)$ is a ratio of polynomials in 's'. Since $G(s)H(s)$ is a complex quantity, Eqn. (A.48) can be split into two equations by equating the angles and magnitudes of both sides to obtain:

Angle condition:

$$\angle G(s)H(s) = \pm 180^\circ (2k + 1) \quad (\text{A.49})$$

Magnitude condition:

$$|G(s)H(s)| = 1 \quad (\text{A.50})$$

APPENDIX B

SIMULATION CODE LISTING

This Appendix contains the program listings of the CASED simulation module programs used to produce the simulation results in Chapter 3 and Chapter 4. The Listings are arranged in the order in which they are cited in these chapters. All the programs listed were written as part of this thesis or is a modification of previous researchers code as discussed in the relevant chapters.

B.1 Artificial Neural Network model

This program is a copy of the COT ANN simulation program written by Burton [BURTON1-2]. This model is used for the simulation of the basic COT ANN controlled power conversion system, as shown in Fig. 3.3 Chapter 3, section 3.2. A flow diagram is provided in section 3.2, Fig. 3.4.

```

/*****
/* Title   : SIM Discrete Control Model :                               */
/* Routine : DC_UserEvent                                           */
/* 0.00 Edit : 20/05/92 : Create Blank                               */
/* 0.01 Edit : 15/07/93 : Define Input as a function                */
/* Author   : C.A. Worthmann                                         */
/* Calls    :                                                         */
/* EventFcn : 3000 (Event function number in simdisc.c)             */
/*-----*/
/*      : Description: This is a model of a continuously online trained artificial neural */
/*      : network for the control of a simplified power conversion system */

```

```

/*          where the inverter has been omitted          */
/*          */
/*      : Input : 0) Ialpha_ref      (alpha reference current)      */
/*      :      : 1) Ibeta_ref      (Beta reference current)      */
/*      :      : 2)  $\omega$       (Motor Speed)      */
/*      : Output: 0) Valpha      (Alpha motor controlling voltage)      */
/*      :      : 1) Vbeta      (Beta motor controlling voltage)      */
/*****/

#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include "simtype.h"
#include "simtevt.h"
#include "screen.h"

/*-----*/
/*Define all constants and setup variable names      */
/*-----*/

#define FT double      /* Float type (float or double)      */
#define EXP exp      /* exp for FT double, expf for FT float      */
#define Sqrt sqrt      /* sqrt for FT double, sqrtf for FT float      */

#define n (int)8      /* number of inputs      */
#define m (int)12      /* number of hidden nodes      */
#define r (int)2      /* number of outputs      */
#define B (FT)0.1      /* learning rate      */
#define M (FT)0.05
#define xmin (FT)-1.0

```

```

#define xmax (FT)1.0
#define wmin (FT)-0.7
#define wmax (FT)0.7
#define xscale ((FT)xmax-(FT)xmin)
#define xshift (FT)xmin
#define wscale ((FT)wmax-(FT)wmin)
#define wshift (FT)wmin

#define Vbase ((FT)311.0)          /* Vbase constant          */
#define Ibase ((FT)14.0)          /* Ibase constant          */
#define Wrbase ((FT)314.0)        /* base rotor speed        */
#define L1 ((FT).0031)
#define Lm ((FT).1032)
#define L2 ((FT).0032)
#define L11 ((FT).1063)
#define Sigma ((FT).0578)

FT W[(m*n)];                      /* input weight matrix     */
FT V[(r*m)];                      /* output weight matrix    */
FT dW[(m*n)];                    /* change in input weights */
FT dV[(r*m)];                    /* change in output weights */
FT x[n];                         /* input data in use       */
FT x_old[n];                     /* input data in use       */
FT y[r];                         /* actual output vector    */
FT e[r];                         /* output error vector     */
FT d[m];                         /* descision vector        */
FT d_old[m];                     /* old descision vector    */

FT Cv;                           /* Cv=Ts/Sigma/L11*Vbase/Ibase */

```

```

FT K=(FT)0.5;
FT i_s_albe_des[2];          /* PU desired albe stator currents      */
FT i_s_albe[2];              /* PU actual albe stator currents        */
FT i_s_albe_hat[2];          /* PU predicted albe stator currents     */
FT v_s_albe[2];              /* PU albe output voltage to PWM        */
FT v_s_albe_out[2];          /* RealU albe output voltage to PWM     */
FT w_r;                      /* PU rotor speed                        */
FT nncurcon_outputs[2];      /* output vector from this program->    */

void init_data(void);         /* initialise weight,input and output data */
void nnop_calc(void);         /* feedforward output calculation        */
void error_calc(void);        /* error vector calculation               */
void backprop(void);          /* error backpropagation weight update    */
void input(FT*);
void v_s_albe_out_calc(void);
void i_s_albe_hat_calc(void);

/*-----*/
/*Set up main program          */
/*-----*/

int Neural_Net (Q,E, Time, State, In )
    Que *Q;
    Event *E;
    Times Time;
    float *State;
    float *In;
{
    extern Models Model;

```

```

float   *Const, *InputIndex,*Output,*DState ;
double  Ts, Valpha, Vbeta, Ialpha, Ibeta, Ialpha_ref, Ibeta_ref;
double  Ids_ref, Iqs_ref, nncurcon_inputs[5];
static  float theta_e;
static  int nnn=0;

#define Sample (Q -> Period)           /* Sampling Period of the controller */
#define NumConst (int)(E->Val[0])
#define NumInput (int)(E->Val[1])
#define ModelNum (int)(E->Val[2])
#define NumDState (int)(E->Val[3])

Const = E-> Val + 4;
InputIndex = E-> Val + 4+ NumConst;
Output = Model[ModelNum] . ModNL->NLConsts;
DState = E-> Val + 4 + NumConst + NumInput ;
#define Input(a) In[(int)InputIndex[a]]

/*-----*/
/*Define and label all inputs */
/*-----*/

#define dum1 (Input(0))
#define dum2 (Input(1))
#define dum3 (Input(2)/314.0)

Ts = Sample;

if(nnn==0)
{

```

```

srand(0);
init_data();
Cv=(1/K)*(Ts/L11/Sigma)*(Vbase/Ibase); /*Calculate the voltage constant */
}

/* ----- */
/* Determine Inputs */
/* ----- */

Ids_ref = 1.0;
Iqs_ref = 2.0;
theta_e = 314.0*Time; /*Calculate the phase angle */
Ialpha_ref = 1.0*cos(theta_e); /*Determine the alpha reference current */
Ibeta_ref = 1.0*sin(theta_e); /*Determine the beta reference current */

nncurcon_inputs[0] = Ialpha_ref; /*pass the inputs to the ANN */
nncurcon_inputs[1] = Ibeta_ref;
nncurcon_inputs[2] = dum3;
nncurcon_inputs[3] = dum1;
nncurcon_inputs[4] = dum2;

/* ----- */
/* Determine States */
/* ----- */

input(nncurcon_inputs);
nnop_calc();

if(nnn != 0)
{
    error_calc();

```

```

        backprop();
    }
    v_s_albe_out_calc();

/* ----- */
/* Write Outputs                                     */
/* ----- */
    Output[0] = nncurcon_outputs[0]/Vbase; /* Pass alpha controlling voltage out */
    Output[1] = nncurcon_outputs[1]/Vbase; /* Pass beta controlling voltage out */
    Output[2] = Ialpha_ref;
    Output[3] = Ibeta_ref;

    nnn=nnn+1;

/* ----- */
/* Cleanup and return                               */
/* ----- */
    return (0);
}                                     /* End DC_Blank */

/* ----- */
/*Perform all sub-routine functions                 */
/* ----- */
/*Organises all ANN inputs into correct variables */
/* ----- */

void input(temp)
float temp[];
{
    int i;

```

```
FT *px=x-1, *px_old=x_old-1, *pi_s_albe=i_s_albe, *pv_s_albe=v_s_albe;
```

```
FT *pi_s_albe_des=i_s_albe_des;
```

```

*(pi_s_albe_des)= temp[0];          /* -> i_s_albe_des[0]          */
*(pi_s_albe_des+1)=temp[1];        /* -> i_s_albe_des[1]          */
w_r=temp[2];                        /* -> w_r                      */
*(pi_s_albe)=temp[3];               /* -> i_s_albe[0]             */
*(pi_s_albe+1)=temp[4];             /* -> i_s_albe[1]             */

```

```
for(i=0;i<n;i++)
```

```
    *(px_old+=1)=*(px+=1);
```

```
px=x;
```

```
*(px+7)=*(pv_s_albe+1);
```

```
*(px+6)=*(pv_s_albe);
```

```
*(px+5)=*(px+2);
```

```
*(px+4)=*(px+1);
```

```
*(px+3)=*(px);
```

```
*(px+2)=w_r;
```

```
*(px+1)=*(pi_s_albe+1);
```

```
*(px)=*(pi_s_albe);
```

```
}
```

```

/*-----*/
/*Calculate all outputs for ANN          */
/*-----*/

```

```
void v_s_albe_out_calc()
```

```
{
```

```
    FT *pv_s_albe=v_s_albe, *py=y, *pi_s_albe_des=i_s_albe_des;
```

```

FT *pv_s_albe_out=v_s_albe_out;
static int nn=0;

*pv_s_albe=((*pi_s_albe_des)-(*py))/Cv;
*(pv_s_albe+1)=((*pi_s_albe_des+1))-(*py+1))/Cv;

if(*(pv_s_albe)>1) *pv_s_albe=1;
if(*(pv_s_albe+1)>1) *(pv_s_albe+1)=1;
if(*(pv_s_albe)<-1) *pv_s_albe=-1;
if(*(pv_s_albe+1)<-1) *(pv_s_albe+1)=-1;

ScreenXY(2,18);
printf("Cv %f n %d ",Cv,nn);
nn=nn+1;

*(pv_s_albe_out)=*(pv_s_albe)*Vbase;
*(pv_s_albe_out+1)=*(pv_s_albe+1)*Vbase;
nncurcon_outputs[0]=*(pv_s_albe_out); /* v_s_albe_out[1] -> */
nncurcon_outputs[1]=*(pv_s_albe_out+1); /* v_s_albe_out[2] -> */
}

/*-----*/
/*Optimise I/O for ANN */
/*-----*/

void nnop_calc()
{
int row, col, opnum;
FT z[1];
FT *pz=z,*pd=d-1,*pd_old=d_old-1,*pV=V-1,*pW=W-1,*px=x-1,*py=y-1;

```

```

for(row=0;row<m;row++){
    px=x_old-1;
    py=y-1;
    *pz=0.0;

    for(col=0;col<n;col++){
        *pz+=*(pW+=1)*(*(px+=1));
    }
    *(pd_old+=1)=*(pd+=1);
    *(pd)=1.0/(1.0+EXP(-(*pz)));
    for(opnum=0;opnum<r;opnum++){
        if(row==0)
            *(py+1)=(FT)0.0;
        *(py+=1)+=*(pV+=1)*(*pd);
    }
}

/*-----*/
/*Calculate error values for use in training      */
/*-----*/

void error_calc()
{
    FT *pi_s_albe=i_s_albe, *pi_s_albe_hat=i_s_albe_hat, *pe=e;
    FT *pi_s_albe_des=i_s_albe_des;

    *pe=(*pi_s_albe)-(*pi_s_albe_hat);
    *(pe+1)=(*(pi_s_albe+1))-(*(pi_s_albe_hat+1));

```

```

*pi_s_albe_hat=*pi_s_albe_des;          /* for constant Cv          */
*(pi_s_albe_hat+1)=*(pi_s_albe_des+1);
}

/*-----*/
/*Perform back propagation routine          */
/*-----*/
void backprop()
{
    int row, col, opnum;
    FT *pW=W-1,*pV=V-1,*pdW=dW-1,*pdV=dV-1,*pd=d_old-1,*px,temp,*pe;

    for(row=0;row<m;row++){
        pd+=1;
        px=x-1;
        pe=e-1;
        temp=0.0;
        for(opnum=0;opnum<r;opnum++){
            temp+=(*pe+=1))*(*pV+=1));
        }
        temp*=B*(*pd)*(1.0-(*pd));
        for(col=0;col<n;col++){
            pdW+=1;
            *pdW=M*(*pdW))+temp*(*px+=1));
            *(pW+=1)+=*pdW;
        }
    }

    pd=d_old-1;

```

```

pV=V-1;
for(row=0;row<m;row++){
    temp=B*(*pd+=1));
    pe=e-1;
    for(opnum=0;opnum<r;opnum++){
        pdV+=1;
        *pdV=M*(*pdV)+temp*(*pe+=1));
        *(pV+=1)+=*pdV;
    }
}
}

/*-----*/
/*Setup initial weight states                                     */
/*-----*/

void init_data()
{
    int row,col;
    FT *p;

    p=W-1;                                     /* initialise W                                     */
    for(row=0;row<m;row++){
        for(col=0;col<n;col++)
            *(p+=1)=wscale*(FT)rand()/((FT)RAND_MAX+(FT)1.0)+wshift;
    }

    p=V-1;                                     /* initialise V                                     */
    for(row=0;row<m;row++){
        for(col=0;col<r;col++)

```

```

*(p+=1)=wscale*(FT)rand()/((FT)RAND_MAX+(FT)1.0)+wshift;
}
}

```

B.2 Simplified Induction motor model

This program is a CASED module, written to model a Squirrel Cage Induction motor. This model is used for the simulation of the basic COT ANN controlled power conversion system, as shown in Fig. 3.3 Chapter 3, section 3.2. A flow diagram is provided in section 3.2, Fig. 3.5.

```

/* *****/
/* Blank File : NC_BLANK.C */
/* Title : SIM Non Linear Model : */
/* Last Edit : 17/05/97 */
/* Author : G. Diana and C.A. Worthmann */
/* Routines : SCIM MOTOR MODEL */
/* Model ID : 4002 in SIMNLIN.C */
/*-----*/
/* : User Model Definition */
/* Setup and define the interface between the Electrical Supply */
/* and the Mechanical load */
/* : Input : 0)  $V_\alpha$  COT ANN alpha control voltage */
/* : : 1)  $V_\beta$  COT ANN beta control voltage */
/* : Output:0)  $I_{amot}$  Phase A Stator Current */
/* : : 1)  $I_{bmot}$  Phase B Stator Current */
/* : : 1)  $T_{em}$  Torque */
/* : Const : 0) R1 Stator Resistance (ohms) */
/* : : 1) R2 Rotor Resistance (ohms) */
/* : : 2) X1 Stator Leakage Reactance (ohms) */

```

```

/*      :      : 3) X2   Rotor Leakage Reactance (ohms)      */
/*      :      : 4) Xm   Mutual Reactance (ohms)             */
/*      :      : 5) Fno   Nominal Frequency for reactances    */
/*      :      : 6) N     Number of poles                      */
/*      :      : 7) Ref   Reference Frame                      */
/* ***** */

#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include "simord.h"
#include "simtype.h"
#include "screen.h"

/*-----*/
/*Define all constants and setup variable names      */
/*-----*/
#define Const Model-> ModNL -> NLConsts

/*-----*/
/*Define main program                                */
/*-----*/
int ACMOTOR (Model, State, In, Deriv, Time, Out, Command)

ModelDefn *Model;
float   *State;
float   *In;
float   *Deriv;
float   Time;

```

```

float    *Out;
int      Command;                                /* 0=Update IO, 1= Update Derivatives */

{
    float V $\alpha$ , V $\beta$ , Tem;
    float Iamot, Ibmot, W, L11, L22, Lm, sig;
    const float R1=0.0217, R2=0.0417, X1=0.0624, X2=0.0636, Xm=2.064, Fno=50.0, Wno
=314.14;

    /*-----*/
    /*Update Inputs and States                                */
    /*-----*/

    #define Uu $\alpha$    In[0]                                /* Phase A Input from Voltage Source */
    #define Uu $\beta$    In[1]                                /* Phase B Input from Voltage Source */
    #define Ids      State[0]                             /* D-Axis Stator Current              */
    #define Iqs      State[1]                             /* Q-Axis Stator Current              */
    #define Ldr      State[2]                             /* D-Axis Rotor Flux Linkages        */
    #define Lqr      State[3]                             /* Q-Axis Rotor Flux Linkages        */

    L11 = (X1 + Xm)/Wno;
    L22 = (X2 + Xm)/Wno;
    Lm  = Xm/Wno;
    sig =(1.0 - (Lm*Lm/(L11*L22)));
    W = 0.0;

    if (!Command )
    {

```

```

/* ----- */
/* Update Outputs                                     */
/* ----- */

Tem = 2.0*Lm*Wno*(Ldr*Ibmot - Lqr*Iamot)/(L22*3.0);

Out[0] = Iamot;
Out[1] = Ibmot;
Out[2] = Tem;
}
else
{
/* ----- */
/* Update Derivatives                               */
/* ----- */

Deriv[0] = (V $\alpha$  - (R1 + R2*Lm*Lm/(L22*L22))*Iamot + W*sig*L11*Ibmot +
            R2*Lm*Ldr/(L22*L22) + Wr*Lm*Lqr/L22)/(sig*L11);
Deriv[1] = (V $\beta$  - (R1 + R2*Lm*Lm/(L22*L22))*Ibmot - W*sig*L11*Iamot +
            R2*Lm*Lqr/(L22*L22) - Wr*Lm*Ldr/L22)/(sig*L11);
Deriv[2] = R2*Lm/L22*Iamot + (W - Wr)*Lqr - R2/L22*Ldr;
Deriv[3] = R2*Lm/L22*Ibmot - (W - Wr)*Ldr - R2/L22*Lqr;

};

return(0);
}
/* End ACMOTOR.C */

```

B.3 Artificial Neural Network model

This module is a model of Burton's COT ANN current controller, where the internal variables and I/O has been changed to achieve current convergence at stator frequencies above 50 Hz. This model is used for the simulation of the COT ANN controlled SCIM drive, as shown in Fig. 3.8 Chapter 3, section 3.3. A flow diagram is provided in section 3.3, Fig. 3.9.

```

/* *****/
/* Title   : SIM Discrete Control Model :                               */
/* Routine  : DC_UserEvent                                                */
/* EventFcn : 3000 (Event function number in simdisc.c)                  */
/*-----:-----*/
/*      : Description: This is a model of a continuously online trained artificial neural */
/*      :               network for the control of a squirrel cage induction motor      */
/*      :               variable speed drive system                                  */
/*      :               */
/*      : Input : 0) Ialpha_ref      (alpha reference current)              */
/*      :       : 1) Ibeta_ref       (Beta reference current)              */
/*      :       : 2)  $\omega$          (Motor Speed)                        */
/*      :               */
/*      : Output: 0) Valpha          (Alpha motor controlling voltage)      */
/*      :       : 1) Vbeta          (Beta motor controlling voltage)      */
/* *****/

#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include "simtype.h"
#include "simtevnt.h"

```

```

#include "screen.h"

/*-----*/
/*Define all constants and setup variable names      */
/*-----*/

#define FT double          /*Float type (float or double)      */
#define EXP exp           /*exp for FT double, expf for FT float */
#define Sqrt sqrt         /*sqrt for FT double, sqrtf for FT float */

#define n (int)8          /*number of inputs                */
#define m (int)12         /*number of hidden nodes          */
#define r (int)2          /*number of outputs               */
#define B (FT)0.01        /*learning rate, previously 0.1   */
#define M (FT)0.05        /*momentum rate                   */
#define xmin (FT)-1.0     /*set up scaling factors for the   */
#define xmax (FT)1.0      /*propagation process             */
#define wmin (FT)-0.7
#define wmax (FT)0.7
#define xscale ((FT)xmax-(FT)xmin)
#define xshift (FT)xmin
#define wscale ((FT)wmax-(FT)wmin)
#define wshift (FT)wmin

#define Vbase ((FT)311.0) /* Vbase constant                 */
#define Ibase ((FT)14.0)  /* Ibase constant                  */
#define Wrbase ((FT)314.0) /* base rotor speed               */
#define L1 ((FT).0031)    /* Motor constants                */
#define Lm ((FT).1032)
#define L2 ((FT).0032)

```

```

#define L11 ((FT).1063)
#define Sigma ((FT).0578)          /* Constant used to calculate voltage const */

FT W[(m*n)];                      /* input weight matrix */
FT V[r*m];                        /* output weight matrix */
FT dW[(m*n)];                     /* change in input weights */
FT dV[r*m];                       /* change in output weights */
FT x[n];                          /* input data in use */
FT x_old[n];                      /* input data in use */
FT y[r];                          /* actual output vector */
FT e[r];                          /* output error vector */
FT d[m];                          /* decision vector */
FT d_old[m];                      /* old decision vector */

FT Cv;                           /* Cv=Ts/Sigma/L11*Vbase/Ibase */
FT K=(FT)0.6;                    /* Scaling factor for the voltage const */
FT i_s_albe_des[2];              /* PU desired albe stator currents */
FT i_s_albe[2];                  /* PU actual albe stator currents */
FT i_s_albe_hat[2];              /* PU predicted albe stator currents */
FT v_s_albe[2];                  /* PU albe output voltage to PWM */
FT v_s_albe_out[2];              /* RealU albe output voltage to PWM */
FT w_r;                          /* PU rotor speed */
FT nncurcon_outputs[2];          /* output vector from the program->

/*-----*/

void init_data(void);             /* initialise weight, input and output data */
void nnop_calc(void);             /* feedforward output calculation */
void error_calc(void);            /* error vector calculation */
void backprop(void);              /* error backpropagation weight update */

```

```

void input(FT*);
void v_s_albe_out_calc(void);
void i_s_albe_hat_calc(void);

/*-----*/
/*Initialise main program                                */
/*-----*/

int Neural_Net (Q,E, Time, State, In )
    Que *Q;
    Event *E;
    Times Time;
    float *State;
    float *In;
{
    extern Models Model;

    float  *Const, *InputIndex, *Output, *DState ;
    double  Ts, Valpha, Vbeta, Ialpha, Ibeta, Ialpha_ref, Ibeta_ref;
    double  Ids_ref, Iqs_ref, nncurcon_inputs[5];
    static  float theta_e;
    static  int nnn=0;

    #define Sample (Q -> Period)                                /* Sampling Period of the controller */
    #define NumConst (int)(E->Val[0])
    #define NumInput (int)(E->Val[1])
    #define ModelNum (int)(E->Val[2])
    #define NumDState (int)(E->Val[3])

    Const = E-> Val + 4;

```

```

InputIndex = E-> Val + 4+ NumConst;
Output = Model[ModelNum] . ModNL->NLConsts;
DState = E-> Val + 4 + NumConst + NumInput ;
#define Input(a) In[(int)InputIndex[a]]

/*-----*/
/*Update all inputs                                     */
/*-----*/

#define dum1 (Input(0))                                /*Read in the inputs to the ANN */
#define dum2 (Input(1))
#define dum3 (Input(2)/314.0)

Ts = Sample;

if(nnn==0)
{
    srand(0);
    init_data();
    Cv=(1/K)*(Ts/L11/Sigma)*(Vbase/Ibase); /*Calculate the voltage constant */

    ScreenXY(20,16);
    printf("nnn= %d Cv = %f Ts = %f",nnn,Cv,Ts);
}

```

```

/*-----*/
/* Calculate Inputs                                     */
/*-----*/

Ids_ref = 1.0;          /* d-axis reference current */
Iqs_ref = 2.0;          /* q-axis reference current */
theta_e = 314.0*Time;   /* calculate the phase angle for sync */
Ialpha_ref = 0.5*cos(theta_e); /* calculate alpha reference current */
Ibeta_ref = 0.5*sin(theta_e); /* calculate beta reference current */

nncurcon_inputs[0] = Ialpha_ref; /*write values to the input of the ANN */
nncurcon_inputs[1] = Ibeta_ref;
nncurcon_inputs[2] = dum3;
nncurcon_inputs[3] = dum1;
nncurcon_inputs[4] = dum2;

/*-----*/
/* Determine States                                     */
/*-----*/

input(nncurcon_inputs);
nnop_calc();

if(nnn != 0)
{
    error_calc();
    backprop();
}
v_s_albe_out_calc();

```

```

/*-----*/
/* Determine Outputs */
/*-----*/

    Output[0] = nncurcon_outputs[0]/Vbase; /* Pass alpha controlling voltage out */
    Output[1] = nncurcon_outputs[1]/Vbase; /* Pass beta controlling voltage out */
    Output[2] = Ialpha_ref;
    Output[3] = Ibeta_ref;

    nnn=nnn+1;

/*-----*/
/* Cleanup and return */
/*-----*/

    return (0);
}                                     /* End DC_Blank */

/*-----*/
/*Perform all sub-routine functions */
/*-----*/
/*Organises all ANN inputs into correct variables */
/*-----*/

void input(temp)

float temp[];

{
    int i;
    FT *px=x-1, *px_old=x_old-1, *pi_s_albe=i_s_albe, *pv_s_albe=v_s_albe;
    FT *pi_s_albe_des=i_s_albe_des;

```

```

*(pi_s_albe_des)= temp[0];          /* -> i_s_albe_des[0]          */
*(pi_s_albe_des+1)=temp[1];        /* -> i_s_albe_des[1]          */
w_r=temp[2];                       /* -> w_r                      */
*(pi_s_albe)=temp[3];              /* -> i_s_albe[0]             */
*(pi_s_albe+1)=temp[4];            /* -> i_s_albe[1]             */

for(i=0;i<n;i++)
    *(px_old+=1)=*(px+=1);

px=x;
*(px+7)=*(pv_s_albe+1);
*(px+6)=*(pv_s_albe);
*(px+5)=*(px+2);
*(px+4)=*(px+1);
*(px+3)=*(px);
*(px+2)=w_r;
*(px+1)=*(pi_s_albe+1);
*(px)=*(pi_s_albe);
}

/*-----*/
/*Calculate all outputs for ANN          */
/*-----*/

void v_s_albe_out_calc()
{
    FT *pv_s_albe=v_s_albe, *py=y, *pi_s_albe_des=i_s_albe_des;
    FT *pv_s_albe_out=v_s_albe_out;
    static int nn=0;

```

```

*pv_s_albe=((*pi_s_albe_des)-(*py))/Cv;
*(pv_s_albe+1)=((*pi_s_albe_des+1)-(*py+1))/Cv;

if(*(pv_s_albe)>1) *pv_s_albe=1;
if(*(pv_s_albe+1)>1) *(pv_s_albe+1)=1;
if(*(pv_s_albe)<-1) *pv_s_albe=-1;
if(*(pv_s_albe+1)<-1) *(pv_s_albe+1)=-1;

*(pv_s_albe_out)=*(pv_s_albe))*Vbase;
*(pv_s_albe_out+1)=*(pv_s_albe+1))*Vbase;
nncurcon_outputs[0]=*(pv_s_albe_out);    /* v_s_albe_out[1] ->          */
nncurcon_outputs[1]=*(pv_s_albe_out+1); /* v_s_albe_out[2] ->          */
}

/*-----*/
/*Optimise I/O for ANN                                     */
/*-----*/

void nnop_calc()
{
    int row, col, opnum;
    FT z[1];
    FT *pz=z,*pd=d-1,*pd_old=d_old-1,*pV=V-1,*pW=W-1,*px=x-1,*py=y-1;

    for(row=0;row<m;row++){
        px=x_old-1;
        py=y-1;
        *pz=0.0;

        for(col=0;col<n;col++){

```

```

    *pz+=*(pW+=1)*(*px+=1));
}
*(pd_old+=1)=*(pd+=1);
*(pd)=1.0/(1.0+EXP(-(*pz)));
for(opnum=0;opnum<r;opnum++){
    if(row==0)
        *(py+1)=(FT)0.0;
        *(py+=1)+=*(pV+=1)*(*pd);
    }
}
}

/*-----*/
/*Calculate error for back propagation                                     */
/*-----*/
void error_calc()
{
    FT *pi_s_albe=i_s_albe, *pi_s_albe_hat=i_s_albe_hat, *pe=e;
    FT *pi_s_albe_des=i_s_albe_des;

    *pe=(*pi_s_albe)-(*pi_s_albe_hat);
    *(pe+1)=(*pi_s_albe+1)-(*pi_s_albe_hat+1));

    *pi_s_albe_hat=*pi_s_albe_des;          /* for constant Cv          */
    *(pi_s_albe_hat+1)=*(pi_s_albe_des+1);
}

```

```

/*-----*/
/*Perform COT ANN Backpropagation function                                     */
/*-----*/

void backprop()
{
    int row, col, opnum;
    FT *pW=W-1,*pV=V-1,*pdW=dW-1,*pdV=dV-1,*pd=d_old-1,*px,temp,*pe;

    for(row=0;row<m;row++){
        pd+=1;
        px=x-1;
        pe=e-1;
        temp=0.0;
        for(opnum=0;opnum<r;opnum++){
            temp+=>(*pe+=1)*(*pV+=1);
        }
        temp*=B>(*pd)*(1.0-(*pd));
        for(col=0;col<n;col++){
            pdW+=1;
            *pdW=M>(*pdW)+temp*(*(px+=1));
            *(pW+=1)+=*pdW;
        }
    }

    pd=d_old-1;
    pV=V-1;
    for(row=0;row<m;row++){
        temp=B>(*pd+=1);
        pe=e-1;

```

```

for(opnum=0;opnum<r;opnum++){
    pdV+=1;
    *pdV=M*(*pdV)+temp*(*pe+=1));
    *(pV+=1)+=*pdV;
}
}
}

/*-----*/
/*Setup initial COT ANN weight states */
/*-----*/

void init_data()
{
    int row,col;
    FT *p;

    p=W-1; /* initialise W */
    for(row=0;row<m;row++){
        for(col=0;col<n;col++)
            *(p+=1)=wscale*(FT)rand()/((FT)RAND_MAX+(FT)1.0)+wshift;
    }

    p=V-1; /* initialise V */
    for(row=0;row<m;row++){
        for(col=0;col<r;col++)
            *(p+=1)=wscale*(FT)rand()/((FT)RAND_MAX+(FT)1.0)+wshift;
    }
}

```

B.4 PWM Hanning Model

This module is a model of the PBM 1/87 Pulsewidth modulation ASIC, including switching delays and time lags. This model is used for the control of the inverter in the simulation of the COT ANN controlled SCIM drive, as shown in Fig. 3.8 Chapter 3, section 3.3. A flow diagram is provided in section 3.3, Fig. 3.10.

```

/*****
/* Title   : SIM Discrete Control Model : Hanning PBM 1/87 ASIC Controller      */
/* Routine : DC_UserEvent                                                         */
/* EventFcn : 4000 (Event function number in simdisc.c)                         */
/*-----:-----*/
/*       : Description: Hanning Chip; PWM Controller : 3rd harmonic              */
/*       :               injection triangular modulation PWM                     */
/*       :                                                       */
/*       : Inputs : 0) UaP, Normalised d-axis voltage                          */
/*       :         : 1) UbP, Normalised q-axis voltage                          */
/*       :         : 2) RHOModP, modified d-axis angle                         */
/*       : Outputs : 0) U1, A phase modulating signal                          */
/*       :         : 1) U2, B phase modulating signal                          */
/*       :         : 2) U3, C phase modulating signal                          */
/*       :         : 3) PHI, Hanning Chip INTERNAL angle phi                   */
*****/

#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include "simtype.h"
#include "simtevent.h"
#include "screen.h"

```

```

/*-----*/
/*Initialise main program                                */
/*-----*/

int HANNING (Q,E, Time, State, In )
    Que *Q;
    Event *E;
    Times Time;
    float *State;
    float *In;
{
    extern Models Model;

    float  *Const, *InputIndex,*Output,*DState ;

    float  U1mod, U2mod, U3mod;
    double U, Umag;
    float  To1, To2, To3, T11, T12, T13, Delay, MinOnTop, MaxOnTop;
    float  TurnOn0, TurnOff0, TurnOn1, TurnOff1, TurnOn2, TurnOff2;
    float  TurnOn3, TurnOff3, TurnOn4, TurnOff4, TurnOn5, TurnOff5;
    float  ang, phi, PHI,U1,U2,U3,RHOModP;
    float  Ua, Ub, RHOMod;
    static float  UaOLD = 0;
    static float  UbOLD = 0;
    static float  RHOModOLD = 90*pi/180;

```



```

/*-----*/
/*Setup all definitions and variables                                     */
/*-----*/

#define Sample (Q -> Period)          /* Sampling Period of the controller */
#define NumConst (int)(E->Val[0])
#define NumInput (int)(E->Val[1])
#define ModelNum (int)(E->Val[2])
#define NumDState (int)(E->Val[3])

Const = E-> Val + 4;
InputIndex = E-> Val + 4 + NumConst;
Output = Model[ModelNum] . ModNL->NLConsts;
DState = E-> Val + 4 + NumConst + NumInput ;
#define Input(a) In[(int)InputIndex[a]]

/*-----*/
/* Deteremine Inputs                                                  */
/*-----*/

#define UaP (Input(0))          /* Present voltage and angle values */
#define UbP (Input(1))          /* enter Voltages in PU angle in rad */
RHOModP=90*pi/180;              /* #define RHOModP (Input(2)) */

/*-----*/
/* Determine outputs                                                  */
/*-----*/

#define U1 Output[0]
#define U2 Output[1]
#define U3 Output[2]
#define PHI Output[3]

```

```

/*-----Constants-----*/

Delay = 0.0;
MinOnTop = 0.001;          /* Relative Minimum On Time          */
MaxOnTop = 1-MinOnTop;     /* Corresponds to Min On Time of lower switch */

/*-----*/
/*Simulate PBM 1/87 Hanning ASIC function and calculate switching sequences */
/*-----*/

Ua = UaOLD;                 /* Hanning Chip introduces one sample delay */
Ub = UbOLD;
RHOMod = RHOModOLD;

UaOLD = UaP;
UbOLD = UbP;
RHOModOLD = RHOModP;

U = sqrt(Ua*Ua+Ub*Ub);
if (U > 1.0)
    U = 1.0;

ang = atan2(Ub,Ua);
if(ang<0.0)ang = ang+2.0*pi;

phi = RHOMod+ang;
if (phi < 0.0)
    phi = phi+2.0*pi;
if (phi > 2.0*pi)
    phi = phi-2.0*pi;

```

```

PHI = phi;

if ((0.0) <= phi && phi <= (pi/3.0)) {           /* 1 */
    U1mod = U*(2.0*sin(phi+pi/6.0)-1.0);
    U2mod = -U;
    U3mod = U*(2.0*sin(phi-pi/6.0+2.0*pi/3.0)-1.0);
}

if ((pi/3.0) < phi && phi <= (2.0*pi/3.0)) {       /* 2 */
    U1mod = U;
    U2mod = U*(2.0*sin(phi-pi/6.0-2.0*pi/3.0)+1.0);
    U3mod = U*(2.0*sin(phi+pi/6.0+2.0*pi/3.0)+1.0);
}

if ((2.0*pi/3.0) < phi && phi <= pi) {             /* 3 */
    U1mod = U*(2.0*sin(phi-pi/6.0)-1.0);
    U2mod = U*(2.0*sin(phi+pi/6.0-2.0*pi/3.0)-1.0);
    U3mod = -U;
}

if (pi < phi && phi <= (4.0*pi/3.0)) {             /* 4 */
    U1mod = U*(2.0*sin(phi+pi/6.0)+1.0);
    U2mod = U;
    U3mod = U*(2.0*sin(phi-pi/6.0+2.0*pi/3.0)+1);
}

if ((4.0*pi/3.0) < phi && phi <= (5.0*pi/3.0)) {   /* 5 */
    U1mod = -U;
    U2mod = U*(2.0*sin(phi-pi/6.0-2.0*pi/3.0)-1.0);
    U3mod = U*(2.0*sin(phi+pi/6.0+2.0*pi/3.0)-1.0);
}

if ((5.0*pi/3.0) < phi && phi <= (2.0*pi)) {       /* 6 */
    U1mod = U*(2.0*sin(phi-pi/6.0)+1.0);

```

```

        U2mod = U*(2.0*sin(phi+pi/6.0-2.0*pi/3.0)+1.0);
        U3mod = U;
    }

    U1 = U1mod;
    U2 = U2mod;
    U3 = U3mod;

    /*-----*/
    /*Simulate the triangular modulation function                                */
    /*-----*/

    To1 = (1.0 + U1mod)/2.0;          /* Total Relative On Times in one Sample */
    To2 = (1.0 + U2mod)/2.0;
    To3 = (1.0 + U3mod)/2.0;

    T11 = (1.0 - U1mod)/4.0;          /* Relative Switch On Times              */
    T12 = (1.0 - U2mod)/4.0;
    T13 = (1.0 - U3mod)/4.0;

    if (To1 < MinOnTop)
        To1 = MinOnTop;

    if (To2 < MinOnTop)
        To2 = MinOnTop;

    if (To3 < MinOnTop)
        To3 = MinOnTop;

    if (To1 > MaxOnTop)

```

To1 = MaxOnTop;

if (To2 > MaxOnTop)

To2 = MaxOnTop;

if (To3 > MaxOnTop)

To3 = MaxOnTop;

TurnOn0 = T11 + Delay/Sample;

TurnOff0 = T11 + To1;

TurnOn1 = T11 + To1 + Delay/Sample;

TurnOff1 = T11;

TurnOn2 = T12 + Delay/Sample;

TurnOff2 = T12 + To2;

TurnOn3 = T12 + To2 + Delay/Sample;

TurnOff3 = T12;

TurnOn4 = T13 + Delay/Sample;

TurnOff4 = T13 + To3;

TurnOn5 = T13 + To3 + Delay/Sample;

TurnOff5 = T13;

```

/*-----Inverter Switch Configuration-----*/
/*
                                0 2 4
/*
                                1 3 5
/*-----*/
/*Generate Switching Events
/*-----*/

#define DSPL
#ifdef DIPL
ScreenXY(20,15);
printf("TON0 %f TOFF0 %f TON1 %f TOFF1 %f",TurnOn0,TurnOff0,TurnOn1,TurnOff1);
ScreenXY(20,16);
printf("TON2 %f TOFF2 %f TON3 %f TOFF3 %f",TurnOn2,TurnOff2,TurnOn3,TurnOff3);
ScreenXY(20,17);
printf("TON4 %f TOFF4 %f TON5 %f TOFF5 %f",TurnOn4,TurnOff4,TurnOn5,TurnOff5);
ScreenXY(20,18);
printf("T11 %f T12 %f T13 %f",T11,T12,T13);
ScreenXY(20,19);
printf("To1 %f To2 %f To3 %f",To1,To2,To3);
#endif

    ClearEvt (Q -> FstEvent);

/*-----*/
/*Centred PWM
/*-----*/

/* ---- A ---- */

    On(TurnOn0, 0, Q, 1);
    Off(TurnOff0, 0, Q, 1);
    On(TurnOn1, 1, Q, 1);
    Off(TurnOff1, 1, Q, 1);

```

```
/* ----- B ----- */
```

```
On(TurnOn2, 2, Q, 1);  
Off(TurnOff2, 2, Q, 1);  
On(TurnOn3, 3, Q, 1);  
Off(TurnOff3, 3, Q, 1);
```

```
/* ----- C ----- */
```

```
On(TurnOn4, 4, Q, 1);  
Off(TurnOff4, 4, Q, 1);  
On(TurnOn5, 5, Q, 1);  
Off(TurnOff5, 5, Q, 1);
```

```
/*-----*/
```

```
/* Cleanup and return */
```

```
/*-----*/
```

```
return (0);
```

```
}
```

```
/* End DC_Blank */
```

B.5 Induction motor model

This program is a CASED module, written to model a Squirrel Cage Induction motor. This model is used in the simulation of the COT ANN controlled SCIM drive, as shown in Fig. 3.8 Chapter 3, section 3.3. A flow diagram is provided in section 3.2, Fig. 3.5.

```

/*****/
/* Routines : IM MOTOR MODEL */
/* Model ID : 4002 in SIMNLIN.C */
/*-----*/
/* : User Model Definition : AC Squirrel cage induction motor model */
/* : Setup and define the interface between the Electrical */
/* : Supply and the Mechanical load */
/* : */
/* : States : 0) Ids D-Axis Stator Current */
/* : : 1) Ids Q-Axis Stator Current */
/* : : 2) Ldr D-Axis Rotor Flux Linkages */
/* : : 3) Lqr Q-Axis Rotor Flux Linkages */
/* : */
/* : Input : 0) Va Phase A Stator Voltage */
/* : : 1) Vb Phase B Stator Voltage */
/* : : 2) Vc Phase C Stator Voltage */
/* : */
/* : Output: 0) Tem Electromagnetic Torque */
/* : : 1) Iamot Phase A Stator Current */
/* : : 2) Ibmot Phase B Stator Current */
/* : : 3) Wr Speed from mechanical load */
/* : : 2) Thetar mechanical load position */
/* : Constants : 0) R1 Stator Resistance (ohms) */

```

```

/*      :      : 1) R2  Rotor Resistance (ohms)                                */
/*      :      : 2) X1  Stator Leakage Reactance(ohms)                        */
/*      :      : 3) X2  Rotor Leakage Reactance (ohms)                        */
/*      :      : 4) Xm  Mutual Reactance (ohms)                              */
/*      :      : 5) Fno Nominal Freq for reactances                           */
/*      :      : 6) N   Number of poles                                       */
/*      :      : 7) Ref Reference Frame                                       */
/*****/

```

```

#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include "simord.h"
#include "simtype.h"
#include "screen.h"

```

```

#define Const Model-> ModNL -> NLConsts

```

```

/*-----*/
/*Initialise main program                                */
/*-----*/

int ACMOTOR (Model, State, In, Deriv, Time, Out, Command)

```

```

ModelDefn *Model;
float      *State;
float      *In;
float      *Deriv;
float      Time;
float      *Out;

```

```

int    Command;                                /* 0=Update IO, 1= Update Derivatives */

{
float  Uds, Uqs, Tem;
float  Iamot, Ibmot, W, L11, L22, Lm, sig;
const float R1=0.0217, R2=0.0417, X1=0.0624, X2=0.0636, Xm=2.064, Fno=50.0,
Wno=314.14;

#define Va    In[0]                            /*Phase A Input current from inverter to motor*/
#define Vb    In[1]                            /*Phase B Input current from inverter to motor*/
#define Vc    In[2]                            /*Phase C Input current from inverter to motor*/
#define Wr    In[3]                            /* Speed Input from the Mechanical Load */
#define Thetar In[4]                            /* Position Input from the Mechanical Load */
#define Ids    State[0]                        /* D-Axis Stator Current */
#define Iqs    State[1]                        /* Q-Axis Stator Current */
#define Ldr    State[2]                        /* D-Axis Rotor Flux Linkages */
#define Lqr    State[3]                        /* Q-Axis Rotor Flux Linkages */

L11 = (X1 + Xm)/Wno;
L22 = (X2 + Xm)/Wno;
Lm  = Xm/Wno;
sig =(1.0 - (Lm*Lm/(L11*L22)));
W = 0.0;

Vds = 0.8165*(Va -0.5*Vb -0.5*Vc);
Vqs = 0.8165*0.866*(Vb - Vc);

ScreenXY(2,20);
printf("Va %f Vb %f Vc %f\n ", Va, Vb, Vc);

```

```
printf("Vds %f Vqs %fn ", Vds, Vqs);
printf("D0 %f D1 %f D2 %f D3 %fn ",Deriv[0],Deriv[1],Deriv[2],Deriv[3]);

if (!Command )
{

/*-----*/
/* Update Outputs                                     */
/*-----*/

Ias   = 0.8165*Ids;
Ibs   = 0.8165*(-0.5*Ids + 0.866*Iqs);
Ics   = 0.8165*(-0.5*Ids - 0.866*Iqs);

/*Theta*Time;*/

Tem = 2.0*Lm*Wno*(Ldr*Iqs - Lqr*Ids)/(L22*3.0);

Out[0] = Ids;
Out[1] = Iqs;
Out[2] = Ias;
Out[3] = Ibs;
Out[4] = Ics;
Out[5] = Tem;
}
else
{
/*-----*/
/* Update Derivatives                                     */
/*-----*/
```

```

Deriv[0] = (Vds - (R1 + R2*Lm*Lm/(L22*L22))*Ids + W*sig*L11*Iqs +
            R2*Lm*Ldr/(L22*L22) + Wr*Lm*Lqr/L22)/(sig*L11);
Deriv[1] = (Vqs - (R1 + R2*Lm*Lm/(L22*L22))*Iqs - W*sig*L11*Ids +
            R2*Lm*Lqr/(L22*L22) - Wr*Lm*Ldr/L22)/(sig*L11);
Deriv[2] = R2*Lm/L22*Ids + (W - Wr)*Lqr - R2/L22*Ldr;
Deriv[3] = R2*Lm/L22*Iqs - (W - Wr)*Ldr - R2/L22*Lqr;
};

return(0);

}                                     /* End ACMOTOR.C */

```


B.6 Three-Phase Sinusoidal Source Model

This CASED module was written to simulate a three-phase sinusoidal source. This model is used in the simulation of the COT ANN controlled SCIM drive, as shown in Fig. 4.9 Chapter 4, section 4.3. A flow diagram is provided in section 4.3, Fig. 4.12.

```

/*****/
/* Title   : SIM Non Linear Model : Three phase Supply          */
/* Model   : 3100                                              */
/* Author   : CA Worthmann                                    */
/*-----:-----*/
/*       : Input : 0) Iadum                                     */
/*       :       : 1) Ib dum                                     */
/*       :       : 2) Icdum                                     */
/*       :                                     */
/*       : Output: 0) Uab           A to B phase voltage        */
/*       :       : 1) Ubc           B to C phase voltage        */
/*       :       : 2) Uca           C to A phase voltage        */
/*       :       : 3) Uan           A to neutral voltage        */
/*       :       : 4) Ubn           B to neutral voltage        */
/*       :       : 5) Ucn           C to neutral voltage        */
/*****/

#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include "simord.h"
#include "simtype.h"
#include "screen.h"

```

```

#define Const Model-> ModNL -> NLConsts

float    Uan,Ubn,Ucn,Uab,Ubc,Uca,U;

/*-----*/
/* Update Outputs                                     */
/*-----*/

int NL_3sin (Model, State, In, Deriv, Time, Out, Command)
    ModelDefn *Model;
    float    *State;
    float    *In;
    float    *Deriv;
    float    Time;
    float    *Out;
    int      Command;                                /* 0 Update IO, 1 Derivatives */
{

/*-----*/
/* Define and update Inputs                           */
/*-----*/

#define Iadum (In[0])                                /* Define three phase currents */
#define Ibdum (In[1])
#define Icdum (In[2])
#define w    314.0                                  /* Define system frequency at 50 Hz */

if (!Command )
{

```

```

/*-----*/
/* Update Outputs */
/*-----*/

U = 150/sqrt(3.0);          /* Calculate the phase to neutral voltage mag. */
    Uan = U*sin(w*Time);    /* Calculate the sinusoidal A,B,and C phase to*/
    Ubn = U*sin(w*Time-120*pi/180); /* neutral voltages */
    Ucn = U*sin(w*Time+120*pi/180);

    Uab = Uan-Ubn;          /*Calculate the A, B, and C phase line */
    Ubc = Ubn-Ucn;          /* voltages */
    Uca = Ucn-Uan;

/*-----*/
/* Write Outputs */
/*-----*/

Out[0] = Uab;              /* Output all variables to the workspace */
Out[1] = Ubc;
Out[2] = Uca;
Out[3] = Uan;
Out[4] = Ubn;
Out[5] = Ucn;

}
else
{
};

return(0);
}                                /* nlm_blnk.c */

```

B.7 User Model for Neural Network Current Controller for a Boost Rectifier

This CASED module is a model of the COT ANN current controller, which has had its internal variables, constants and I/O changed to enable it to operate with a boost rectifier. This model is used for the simulation of the COT ANN controlled boost rectifier, as shown in Fig. 4.9 Chapter 4, section 4.3. A flow diagram is provided in section 4.3, Fig. 4.10.

```

/*****/
/* Title   : SIM Discrete Control Model :                               */
/* Routine : DC_UserEvent                                                    */
/* Author  : C.A. Worthmann                                                  */
/* EventFcn : 3000 (Event function number in simdisc.c)                     */
/*-----*/
/*      : Description: ANN current controller model                        */
/*      : Derived to control a Boost Rectifier to reduce mains voltage dips */
/*      :                                                    */
/*      : Input   : 0) Usa  Phase A phase to neutral voltage              */
/*      :          : 1) Usb  Phase B phase to neutral voltage              */
/*      :          : 2) Usc  Phase C phase to neutral voltage              */
/*      :          : 3) Isa   Phase A supply current                      */
/*      :          : 4) Isb   Phase B supply current                      */
/*      :          : 5) Isc   Phase C supply current                      */
/*      :          : 6) Udc   DC link voltage                             */
/*      :                                                    */
/*      : Output  : 0) Valpha                                              */
/*      :          : 1) Vbeta                                              */
/*      :          : 2) Ialpha reference                                    */
/*      :          : 3) Ibeta reference                                    */
/*      :                                                    */

```

```

/* *****/
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include "simtype.h"
#include "simtevnt.h"
#include "screen.h"

/*-----*/
/*Define all constants and setup variable names          */
/*-----*/

#define FT double          /* Float type (float or double)          */
#define EXP exp           /* exp for FT double, expf for FT float */
#define Sqrt sqrt         /* sqrt for FT double, sqrtf for FT float */

#define n (int)10          /* number of inputs                      */
#define m (int)12          /* number of hidden nodes                */
#define r (int)2           /* number of outputs                     */
#define B (FT)0.12         /* learning rate                         */
#define M (FT)0.05         /* momentum rate                         */
#define xmin (FT)-1.0
#define xmax (FT)1.0
#define wmin (FT)-0.7
#define wmax (FT)0.7
#define xscale ((FT)xmax-(FT)xmin)
#define xshift (FT)xmin
#define wscale ((FT)wmax-(FT)wmin)
#define wshift (FT)wmin

```

```

#define Vbase ((FT)1.225*150.0/sqrt(3.0)) /* Vbase constant */
#define Ibase ((FT)1.225*15) /* Ibase constant */
#define Wrbase ((FT)314.0) /* base rotor speed */
#define Ls ((FT).00866) /* line inductance */

FT W[(m*n)]; /* input weight matrix */
FT V[(r*m)]; /* output weight matrix */
FT dW[(m*n)]; /* change in input weights */
FT dV[(r*m)]; /* change in output weights */
FT x[n]; /* input data in use */
FT x_old[n]; /* input data in use */
FT y[r]; /* actual output vector */
FT e[r]; /* output error vector */
FT d[m]; /* decision vector */
FT d_old[m]; /* old decision vector */

FT Cv; /* Cv=Ts/Ls*Ibase/Vbase */
FT K=(FT)55; /* 27.5 for 5kHz, dependent on sampling rate */
FT i_s_albe_des[2]; /* PU desired albe stator currents */
FT i_s_albe[2]; /* PU actual albe stator currents */
FT i_s_albe_hat[2]; /* PU predicted albe stator currents */
FT v_s_albe[2]; /* PU albe output voltage to PWM */
FT v_s_albe_out[2]; /* RealU albe output voltage to PWM */
FT v_s_albe_in[2]; /* alpha beta mains voltages */
FT nncurcon_outputs[2]; /* OUTPUT VECTOR FROM THIS PROGRAM -> */

/*-----*/
void init_data(void); /* initialise weight, input and output data */
void nnop_calc(void); /* feedforward output calculation */

```

```

void error_calc(void);           /* error vector calculation      */
void backprop(void);            /* error backpropagation weight */
void input(FT*);
void v_s_albe_out_calc(void);
void i_s_albe_hat_calc(void);

float  Eamains, Ebmain, Iamains, Ibmain;
float  Eapu, EBpu;
float  Cos, Sin;
float  MagUs;
float  Ud, Uq, Id, Iq;
float  UdcStar;
float  UdcError;
float  IvOutk1, IvOut;
float  CvOutk1, CvOut;

/*-----*/
/*Initialise main program      */
/*-----*/

int Neural_Net (Q,E, Time, State, In )
    Que *Q;
    Event *E;
    Times Time;
    float *State;
    float *In;
{
    extern Models Model;

    float  *Const, *InputIndex, *Output, *DState;

```

```

double   Ts, Valpha, Vbeta, Ialpha, Ibeta, Ialpha_ref, Ibeta_ref;
double   Ids_ref, Iqs_ref, nncurcon_inputs[6];
static   float theta_e;
static   int nnn=0;

#define Sample (Q -> Period)           /* Sampling Period of the controller */
#define NumConst (int)(E->Val[0])
#define NumInput (int)(E->Val[1])
#define ModelNum (int)(E->Val[2])
#define NumDState (int)(E->Val[3])

Const = E-> Val + 4;
InputIndex = E-> Val + 4+ NumConst;
Output = Model[ModelNum] . ModNL->NLConsts;
DState = E-> Val + 4 + NumConst + NumInput ;
#define Input(a) In[(int)InputIndex[a]]

/*-----*/
/*Update and read all inputs from system */
/*-----*/

#define Usa In[3]           /* read in phase A phase to neutral voltage */
#define Usb In[4]           /* read in phase B phase to neutral voltage */
#define Usc In[5]           /* read in phase C phase to neutral voltage */
#define Isa In[0]           /* read in phase A line current */
#define Isb In[1]           /* read in phase B line current */
#define Isc In[2]           /* read in phase C line current */
#define Udc In[7]           /* read in DC link voltage */

```

```

/*-----*/
/*Mains synchronisation block, see Chapter 4 pp 4.7 Eqn. (4.7) through (4.12) */
/*-----*/
Eamains = (Usa-0.5*Usb-0.5*Usc)*sqrt(2.0/3.0); /* calculate alpha mains voltage comp. */
Ebmain = (Usb - Usc)/(sqrt(2.0)); /*calculate beta mains voltage component */

MagUs = sqrt(Eamains*Eamains+Ebmain*Ebmain); /* calculate voltage magnitude */

Cos = Eamains/MagUs; /* calculate mains cosine value */
Sin = Ebmain/MagUs; /* calculate mains sine value */

EApu = Eamains/Vbase; /* alpha voltage in per unit */
EBpu = Ebmain/Vbase; /* beta voltage in per unit */

Iamains = (Isa-0.5*Isb-0.5*Is c)*sqrt(2.0/3.0); /* calculate alpha mains current comp. */
Ibmains = (Isb - Is c)/(sqrt(2.0)); /* calculate beta mains current comp. */

/*-----*/
/*Input synchronised values into ANN */
/*-----*/
#define dum1 Iamains/Ibase /* define values to input variables */
#define dum2 Ibmains/Ibase
#define dum3 EApu
#define dum4 EBpu

/*-----*/
Ts = Sample;

if(nnn==0)

```

```

{
    srand(0);
    init_data();
    Cv=(K)*(Ts/Ls)*(Ibase/Vbase);

    ScreenXY(22,21);
    printf("Cv = %f Ts = %f",Cv,Ts);
}

/*-----*/
/* Voltage PI loop, see Chapter 4, pp 4.11, Eqn. (4.18) */
/*-----*/

    UdcStar = 212.0;                /* desired DC link voltage level */
    UdcError = UdcStar - Udc;       /* calculate DC voltage error */
    IvOutk1 = IvOut;               /* save old control values */
    CvOutk1 = CvOut;

    IvOut = (0.3/2)*Sample*UdcError + IvOutk1; /* DC link PI controller */
    CvOut = ((1.0+200.0*Sample)/(1.0+50.0*Sample))*IvOut + (CvOutk1-
        IvOutk1)/(1.0+50.0*Sample);

/*-----*/
/*Define all reference currents, see Chapter 4, pp 4.9, Eqns. (4.13) and (4.14) */
/*-----*/

    Ids_ref = IvOut;                /* d-axis reference current */
    Iqs_ref = 0.0;                  /* q-axis reference current */
    Ialpha_ref = Ids_ref*EApu-Iqs_ref*EBpu; /* alpha reference current */
    Ibeta_ref = Ids_ref*EBpu+Iqs_ref*EApu; /* beta reference current */

```

```

/*-----*/
/*Pass values to ANN Backpropagation algorithm                                     */
/*-----*/

    nncurcon_inputs[0] = Ialpha_ref;
    nncurcon_inputs[1] = Ibeta_ref;
    nncurcon_inputs[2] = dum3;
    nncurcon_inputs[3] = dum4;
    nncurcon_inputs[4] = dum1;
    nncurcon_inputs[5] = dum2;

/*-----*/
/* Determine States                                                                */
/*-----*/

    input(nncurcon_inputs);
    nnop_calc();

    if(nnn != 0)
    {
        error_calc();
        backprop();
    }
    v_s_albe_out_calc();

/*-----*/
/* Determine Outputs                                                              */
/*-----*/

    Output[0] = nncurcon_outputs[0]/Vbase; /* Valpha volts                      */
    Output[1] = nncurcon_outputs[1]/Vbase; /* Vbeta volts                          */
    Output[2] = Ialpha_ref*Ibase;

```

```

Output[3] = Ibeta_ref*Ibase;

nnn=nnn+1;

/*-----*/
/* Cleanup and return */
/*-----*/

return (0);

}                                /* End DC_Blank */

/*-----*/
/*Perform all sub-routine functions */
/*-----*/
/*Organises all ANN inputs into correct variables */
/*-----*/

void input(temp)
float temp[];
{
    int i;
    FT *px=x-1, *px_old=x_old-1, *pi_s_albe=i_s_albe, *pv_s_albe=v_s_albe;
    FT *pi_s_albe_des=i_s_albe_des, *pv_s_albe_in=v_s_albe_in;

    *(pi_s_albe_des)= temp[0];          /* -> i_s_albe_des[0] */
    *(pi_s_albe_des+1)=temp[1];         /* -> i_s_albe_des[1] */
    *(pv_s_albe_in)=temp[2];
    *(pv_s_albe_in+1)=temp[3];
    *(pi_s_albe)=temp[4];               /* -> i_s_albe[0] */
    *(pi_s_albe+1)=temp[5];             /* -> i_s_albe[1] */

```

```

for(i=0;i<n;i++)
    *(px_old+=1)=*(px+=1);

px=x;
*(px+9)=*(pv_s_albe+1);
*(px+8)=*(pv_s_albe);
*(px+7)=*(px+1);
*(px+6)=*(px);
*(px+5)=*(px+3);
*(px+4)=*(px+2);
*(px+3)=*(pv_s_albe_in+1);
*(px+2)=*(pv_s_albe_in);
*(px+1)=*(pi_s_albe+1);
*(px)=*(pi_s_albe);
}

/*-----*/
/*Calculate all outputs for ANN                                     */
/*-----*/

void v_s_albe_out_calc()
{
    FT *pv_s_albe=v_s_albe, *py=y, *pi_s_albe_des=i_s_albe_des;
    FT *pv_s_albe_out=v_s_albe_out;
    static int nn=0;

    *pv_s_albe=((*pi_s_albe_des)-(*py))/Cv;
    *(pv_s_albe+1)=((*pi_s_albe_des+1)-(*py+1))/Cv;

    if(*(pv_s_albe)>1) *pv_s_albe=1;

```

```

if(*(pv_s_albe+1)>1) *(pv_s_albe+1)=1;
if(*(pv_s_albe)<-1) *pv_s_albe=-1;
if(*(pv_s_albe+1)<-1) *(pv_s_albe+1)=-1;

*(pv_s_albe_out)=(-(pv_s_albe))+EApu)*Vbase;
*(pv_s_albe_out+1)=(-(pv_s_albe+1))+EBpu)*Vbase;

nncurcon_outputs[0]=*(pv_s_albe_out); /* v_s_albe_out[1] -> */
nncurcon_outputs[1]=*(pv_s_albe_out+1); /* v_s_albe_out[2] -> */
}

/*-----*/
/*Optimise I/O for ANN */
/*-----*/

void nnop_calc()
{
int row, col, opnum;
FT z[1];
FT *pz=z,*pd=d-1,*pd_old=d_old-1,*pV=V-1,*pW=W-1,*px=x-1,*py=y-1;

for(row=0;row<m;row++){
px=x_old-1;
py=y-1;
*pz=0.0;

for(col=0;col<n;col++){
*pz+=*(pW+=1)*(*(px+=1));
}
*(pd_old+=1)=*(pd+=1);

```

```

*(pd)=1.0/(1.0+EXP(-(*pz)));
for(opnum=0;opnum<r;opnum++){
    if(row==0)
        *(py+1)=(FT)0.0;
        *(py+=1)+=*(pV+=1)*(*pd);
    }
}
}

/*-----*/
/*Calculate error for Backpropagation */
/*-----*/
void error_calc()
{
    FT *pi_s_albe=i_s_albe, *pi_s_albe_hat=i_s_albe_hat, *pe=e;
    FT *pi_s_albe_des=i_s_albe_des;

    *pe=(*pi_s_albe)-(*pi_s_albe_hat);
    *(pe+1)=(*pi_s_albe+1)-(*pi_s_albe_hat+1));

    *pi_s_albe_hat=*pi_s_albe_des;          /* for constant Cv */
    *(pi_s_albe_hat+1)=*(pi_s_albe_des+1);
}

/*-----*/
/*Perform Backpropagation function */
/*-----*/
void backprop()
{

```

```

int row, col, opnum;
FT *pW=W-1,*pV=V-1,*pdW=dW-1,*pdV=dV-1,*pd=d_old-1,*px,temp,*pe;

for(row=0;row<m;row++){
    pd+=1;
    px=x-1;
    pe=e-1;
    temp=0.0;
    for(opnum=0;opnum<r;opnum++){
        temp+=(*pe+=1)*(*pV+=1);
    }
    temp*=B*(*pd)*(1.0-(*pd));
    for(col=0;col<n;col++){
        pdW+=1;
        *pdW=M*(*pdW)+temp*(*px+=1);
        *(pW+=1)+=*pdW;
    }
}

pd=d_old-1;
pV=V-1;
for(row=0;row<m;row++){
    temp=B*(*pd+=1);
    pe=e-1;
    for(opnum=0;opnum<r;opnum++){
        pdV+=1;
        *pdV=M*(*pdV)+temp*(*pe+=1);
        *(pV+=1)+=*pdV;
    }
}

```

```
}  
}  
  
/*-----*/  
/*Setup initial weight states*/  
/*-----*/  
void init_data()  
{  
    int row,col;  
    FT *p;  
  
    p=W-1; /* initialise W */  
    for(row=0;row<m;row++){  
        for(col=0;col<n;col++){  
            *(p+=1)=wscale*(FT)rand()/((FT)RAND_MAX+(FT)1.0)+wshift;  
        }  
  
        p=V-1; /* initialise V */  
        for(row=0;row<m;row++){  
            for(col=0;col<r;col++){  
                *(p+=1)=wscale*(FT)rand()/((FT)RAND_MAX+(FT)1.0)+wshift;  
            }  
        }  
    }  
}
```

APPENDIX C

HYPER SIGNAL REAL-TIME BLOCK FUNCTION CODE LISTING

C.1 PWM Control Block Function

The Block Wizard generates code for the Dynamic Linker Library files (DLL) in Hypersignal and a template for the real-time DSP code. This DSP code template is generated as two separate files, one being the main code body, and the other being the include file where all the constants and subroutines are initialised.

This Appendix contains the program listings of the Hypersignal module programs used to perform the timing measurements in Chapter 5. The Listings are arranged in the order in which they are cited in the chapter. All the programs listed were written as part of this thesis.

C.1.1 PWM Control Block Include code

This is the include file for the PBM 1/87 Hanning ASIC control code, which is listed below in C.1.2.

```

/*=====//
//      FILE NAME : _pwmcont.h                      //
//      BLOCK NAME: PWM control                      //
//      GROUP NAME: Motion Control                  //
//      PURPOSE   : Provides the real-time block's DSP header file. //
//      Hypersignal Block Wizard Version 4.00.14 Auto-Generated Block //
//                                                    //
//      Number of Inputs : 3                        //
//      Number of Outputs: 0                        //

```

```

//                                                                    //
//      Creation Date: Thu - 06 November 1997                        //
//      Creation Time: 08:15 AM                                       //
//=====*/

/*****/
/*                                DEFINITIONS                                */
/*****/

/* boolean macros */
typedef unsigned int BOOL;
#define FALSE  0
#define TRUE   1

#define Decodes0 0                /* Define the DECODES0 pin      */
#define Decodes1 1                /* Define the DECODES1 pin      */

#define dec0 (int) 0x818800        /* Define the memory location of DECODES0 */
#define dec1 (int) 0x819000        /* Define the memory location of DECODES1 */

/*****/
/*                                Data Pointers / Parameter Structure                                */
/*****/

typedef struct
{
/*****/
/*                                Synchronization Addresses                                */
/*****/

    unsigned int *SyncIn;          /* ptr to input sync value      */
    unsigned int *SyncOut;         /* ptr to output sync value     */

```

```

/*****
/*          Input Data Pointers / Framesizes          */
*****/

float      *InPtr0;          /* input 0 data pointer */
float      *InPtr1;          /* input 1 data pointer */
float      *InPtr2;          /* input 2 data pointer */
unsigned int FramesizeIn0;    /* input 0 framesize */
unsigned int FramesizeIn1;    /* input 1 framesize */
unsigned int FramesizeIn2;    /* input 2 framesize */

/*****
/*          Block Parameters          */
*****/

int      _Enable;          /* Define parameters for the function blocks */
int      _Frequencyphase;  /* dialog box */
int      _Decoding;
int      _BoardNo;
int      *Data_word;
int      *Status_word;

/*****
/*          Internal Persistent Variables          */
*****/

/*      NOTE: If your block requires persistent variables, then they must be put here.      */
/*      DO NOT USE global variables or statics in the C source file unless you really      */
/*      want them to be shared by all instances of this block.  If you do add variables,      */
/*      update the "#define INTVAR_SIZE" in the PC's header (.h) file to reflect the          */
/*      proper internal variable storage.          */
*****/
```



```
} PARAMS;
```

```

/*****
/*
/*
/*****
void pwmcontr_INT (PARAMS *pParam); /* optional block interrupt routine */

void pwmcontr_INIT (PARAMS *pParam); /* optional block initialization routine */

void pwmcontr_STOP (PARAMS *pParam); /* optional block stop routine */

void pwmcontr_RESTART (PARAMS *pParam); /* optional block restart routine */

void pwmcontr(PARAMS *pParam); /* main block routine */

void pollpwm(int *S_word); /* polls the pwm chip until ready */

```

C.1.2 PWM Control Block C-source code

This program is the control code for the PBM 1/87 Hanning ASIC. This code is used to calculate the COT ANN current controller's sampling times when controlling a boost rectifier, as shown in Fig. 5.1 Chapter 5, section 5.1. A flow diagram is provided in section 5.1, Fig. 5.3.

```

/*=====//
//      FILE NAME : _pwmcont.c                                //
//      BLOCK NAME: PWM control                                //
//      GROUP NAME: Motion Control                             //
//      PURPOSE   : Provides the real-time block's DSP C source code for the control of //
//                  PWM Hanning control card                    //
/      Hypersignal Block Wizard Version 4.00.14 Auto-Generated Block //
//                                                                //
//  Number of Inputs : 3                                        //
//  Number of Outputs: 0                                        //
//                                                                //
//  Creation Date: Thu - 06 November 1997                      //
//  Creation Time: 08:15 AM                                     //
//=====*/

#include "_pwmcont.h"                /* Include necessary files */

#define TAUS ((int)0)                /* Define the turn-off time */
#define TTOT ((int)0)                /* Define the dead-time */
#define TMIN ((int)0)                /* Define the turn-on time */
#define VORTL ((int)2)               /* Define switching frequency scaling factor */
#define TSTART ((int)(512-(322/(VORTL+1)))) /* Calculate the start time */

```

```

/*****
/*          Optional real-time block interrupt routine          */
/*****
/* If this routine is activated, it will be called in response to a selected DSP interrupt. */
/* If this routine is not activated, the main block routine will be called in response to a */
/* selected DSP interrupt.                                     */
/*****
*/

void pwmcontr_INT(PARAMS *pParam)
{

}

*/

/*****
/*          Optional real-time block initialization routine      */
/*****
/* If this routine is activated, it will be called one time before the main application begins. */
/* This allows for any required software or hardware initialization to be performed before */
/* the block executes.                                          */
/*****
*/

void pwmcontr_INIT(PARAMS *pParam)
{ int *temp;

    if(pParam->_Decoding == Decodes0)    /* check which decoding pin is being used */
    {
        (pParam->Data_word) = (int*)(dec0 + ((pParam->_BoardNo)*256));
        (pParam->Status_word) = (int*)(dec0 + 1 + ((pParam->_BoardNo)*256));
    }
}

```

```

}
else
{
    (pParam->Status_word) = (int*)(dec1 + 1+((pParam->_BoardNo)*256));
    (pParam->Data_word) = (int*)(dec1 + ((pParam->_BoardNo)*256));
}

*(pParam->Status_word) = 128;          /* set up 16 bit addressing mode      */
*(pParam->Status_word) = 128;          /* set addres to zero                  */

pollpwm((pParam->Status_word));        /* wait for pwm chip to be ready      */
*(pParam->Data_word) = ((int)0);        /* write Ua to PWM chip               */
pollpwm((pParam->Status_word));
*(pParam->Data_word) = ((int)0);        /* write Ub to PWM chip               */
pollpwm((pParam->Status_word));
*(pParam->Data_word) = ((int)0);        /* write phi1 to PWM chip             */
pollpwm((pParam->Status_word));
*(pParam->Data_word) = ((int)0);        /* write dphi1 to PWM chip            */
pollpwm((pParam->Status_word));
*(pParam->Data_word) = ((int)0);        /*write phi0 to PWM chip              */
pollpwm((pParam->Status_word));
*(pParam->Data_word) = ((int)0);        /*write dphi0 to PWM chip             */
pollpwm((pParam->Status_word));
*(pParam->Data_word) = ((int)0);        /*write phiadd to PWM chip            */
pollpwm((pParam->Status_word));
*(pParam->Data_word) = ((int)0);        /* unused                             */
pollpwm((pParam->Status_word));

*(pParam->Data_word) = TAUS;            /*write turn-off time to PWM chip     */
pollpwm((pParam->Status_word));

```



```

*(pParam->Data_word) = TTOT;          /*write dead band to PWM chip      */
pollpwm((pParam->Status_word));
*(pParam->Data_word) = TMIN;           /*write turn-on time to PWM chip    */
pollpwm((pParam->Status_word));
*(pParam->Data_word) = VORTL;         /*write switch freq. scale value to PWM chip*/
pollpwm((pParam->Status_word));
*(pParam->Data_word) = TSTART;        /*write start time to PWM chip      */
}

/*****
/*          Optional real-time block stop routine          */
*****/
/* If this routine is activated, it will be called whenever the block diagram worksheet's
/* execution is stopped. Blocks that deal with hardware may need this routine to stop the
/* hardware's execution.                                     */
*****/

void pwmcontr_STOP(PARAMS *pParam)
{
    *(pParam->Status_word) = 0;        /* disable PWM chip                */
    *(pParam->Status_word) = 0;        /* disable PWM chip                */
}

/*****
/*          Optional real-time block restart routine          */
*****/
/* If this routine is activated, it will be called whenever a block diagram worksheet is
/* executed after being stopped. Blocks that deal with hardware may need this routine
/* to restart the hardware's execution.                     */
*****/

```

```

void pwmcontr_RESTART(PARAMS *pParam)
{
    if(pParam->_Decoding == Decodes0)    /* re-initialises the chip on reset    */
    { (pParam->Status_word) = (int*)(dec0 + 1+((pParam->_BoardNo)*256) );
      (pParam->Data_word)  = (int*)(dec0 + ((pParam->_BoardNo)*256) );
    }
    else
    { (pParam->Status_word) = (int*)(dec1 + 1+((pParam->_BoardNo)*256) );
      (pParam->Data_word)  = (int*)(dec1 + ((pParam->_BoardNo)*256) );
    }

    *(pParam->Status_word) = 128;          /* set up 16 bit addressing mode    */
    *(pParam->Status_word) = 128;          /* set addres to zero              */

    pollpwm((pParam->Status_word));        /* wait for pwm chip              */
    *(pParam->Data_word) = ((int)0);        /* Ua                              */
    pollpwm((pParam->Status_word));
    *(pParam->Data_word) = ((int)0);        /* Ub                              */
    pollpwm((pParam->Status_word));
    *(pParam->Data_word) = ((int)0);        /* phi1                            */

    pollpwm((pParam->Status_word));
    *(pParam->Data_word) = ((int)0);        /* dphi1                           */
    pollpwm((pParam->Status_word));
    *(pParam->Data_word) = ((int)0);        /* phi0                            */
    pollpwm((pParam->Status_word));
    *(pParam->Data_word) = ((int)0);        /* dphi0                           */
    pollpwm((pParam->Status_word));
    *(pParam->Data_word) = ((int)0);        /* phiadd                          */

```

```

pollpwm((pParam->Status_word));
*(pParam->Data_word) = ((int)0);          /* unused          */
pollpwm((pParam->Status_word));
*(pParam->Data_word) = TAUS;              /* turn off time    */
pollpwm((pParam->Status_word));
*(pParam->Data_word) = TTOT;              /* dead band        */
pollpwm((pParam->Status_word));
*(pParam->Data_word) = TMIN;              /* turn on time     */
pollpwm((pParam->Status_word));
*(pParam->Data_word) = VORTL;             /* switching frequency scale value */
pollpwm((pParam->Status_word));
*(pParam->Data_word) = TSTART;            /* start of processing cycle */
}

/*****
/*                      Real-time block routine                      */
*****/

/* This is the main block routine. It is called during each loop of the main application. */
/* If an interrupt is selected, and the interrupt routine above is not activated, this routine */
/* will be called in response to the selected */
/* interrupt instead of during the main application loop. */
*****/

void pwmcontr(PARAMS *pParam)
{
    unsigned int index;                    /* index for frame processing loop */
    float Ua,Ub,freq ;                    /* variable used */

    Ua = *(pParam->InPtr0);                /* accessing input data 0 */
    Ub = *(pParam->InPtr1);                /* accessing input data 1 */

```

```

freq = *(pParam->InPtr2);          /* accessing input data 2          */

if(pParam->_Enable)                  /* check if enabled in dialog box    */
{
    *(pParam->Status_word) = 129;    /* set address to zero               */
    pollpwm((pParam->Status_word));
    *(pParam->Data_word) = (int)Ua;   /* write out Ua voltage to PWM chip  */
    pollpwm((pParam->Status_word));
    *(pParam->Data_word) = (int)Ub;   /* write out Ub voltage PWM chip     */

if (pParam->_Frequencyphase)         /* check mode of operation of PWM chip */
{
    *(pParam->Status_word) = 897;    /* go frequency memory address for   */
    pollpwm((pParam->Status_word));
    *(pParam->Data_word) = (int)freq; /* write frequency to PWM chip        */
}
else
{
    pollpwm((pParam->Status_word));
    *(pParam->Data_word) = (int)freq; /* write phase angle to PWM chip      */
}
}
else                                 /* if not enabled in dialog box       */
{
    *(pParam->Status_word) = 0;       /* disable chip                       */
}
}

void pollpwm(int *S_word)            /* Subroutine to poll the PWM chip    */

```

```
{
    int check;

    check = *(S_word);
    while ((check & 0x1) != 0)
    {
        check = *(S_word);
    }
}
```

C.2 ANN Current Controller Block Function

C.2.1 ANN Current Controller Block Include Code

This is the include file for the COT ANN current controller, which is listed below in C.2.1.

```

/*=====//
//      FILE NAME : _neur2.h                                //
//      BLOCK NAME: ANNtest                                  //
//      GROUP NAME: neural network                          //
//      PURPOSE   : Provides the real-time block's DSP header file. //
//      Hypersignal Block Wizard Version 4.00.14 Auto-Generated Block //
//                                                         //
//      Number of Inputs : 7                                //
//      Number of Outputs: 4                                //
//                                                         //
//      Creation Date: Thu - 28 May 1998                    //
//      Creation Time: 03:01 PM                              //
//=====*/

/*****/
/*                                DEFINITIONS                                */
/*****/

/* boolean macros */
typedef unsigned int BOOL;
#define FALSE  0
#define TRUE   1

#define FT double                /* Float type (float or double) */

```

```

#define EXP exp          /* exp for FT double, expf for FT float */
#define SQRT sqrt        /* sqrt for FT double, sqrtf for FT float */

#define n (int)10        /* number of inputs */
#define m (int)12        /* number of hidden nodes */
#define r (int)2         /* number of outputs */
#define B (FT)0.12       /* learning rate */
#define M (FT)0.05       /* Momentum rate */
#define xmin (FT)-1.0    /* setting up dimensions for ANN matrices */
#define xmax (FT)1.0
#define wmin (FT)-0.7
#define wmax (FT)0.7
#define xscale ((FT)xmax-(FT)xmin)
#define xshift (FT)xmin
#define wscale ((FT)wmax-(FT)wmin) /* Scaling factor for weight functions */
#define wshift (FT)wmin

FT W[(m*n)];            /* input weight matrix */
FT V[(r*m)];            /* output weight matrix */
FT dW[(m*n)];          /* change in input weights */
FT dV[(r*m)];          /* change in output weights */
FT x[n];                /* input data in use */
FT x_old[n];            /* input data in use */
FT y[r];                /* actual output vector */
FT e[r];                /* output error vector */
FT d[m];                /* descision vector */
FT d_old[m];            /* old descision vector

FT Cv;                  /* Cv=Ts/Ls*Ibase/Vbase */

```

```

FT K=(FT)40.0;          /* for 5kHz Sampling rate          */
FT i_s_albe_des[2];     /* PU desired albe stator currents          */
FT i_s_albe[2];         /* PU actual albe stator currents          */
FT i_s_albe_hat[2];     /* PU predicted albe stator currents        */
FT v_s_albe[2];         /* PU albe output voltage to PWM           */
FT v_s_albe_out[2];     /* RealU albe output voltage to PWM        */
FT v_s_albe_in[2];      /* alpha beta mains voltages              */
FT nncurcon_outputs[2]; /* Output vector from this program ->      */

float Eamains,Ebmains,Iamains,Ibmains; /*alpha-beta mains voltage and current */
float Eamainsn, Ebmainsn, Iamainsn, Ibmainsn; /*phase adjusted alpha-beta mains V and I */
float Usa, Usb, Usc, Isa, Isb, Isc, Udc; /* sampled voltages and currents          */
float Ls, a;             /* Line inductance and phase adjustment    */
float EApu,EBpu;         /* alpha-beta per unit voltages            */
float Cos,Sin;           /* mains cosine and sine components        */
float MagUs, Imag;       /* magnitude of mains V and I space vectors */
float Voltlim;           /* Voltage limit                          */
float Ud,Uq,Id,Iq;       /* d-q voltage and currents                */
float UdcStar;           /* required voltage level                  */
float UdcError, UdcErrorp; /* DC error states                        */
float IvOutk1,IvOut;     /* Voltage integrator output              */
float CvOutk1,CvOut;     /* Voltage controller output              */
float Vbase,Ibase,Ls1,Samplingrate,Vscale,Udcscale,Iscale,Udc_ref;
float Idmax;             /*Maximum d-axis line current             */

/*****/
/*                               Data Pointers / Parameter Structure                               */
/*****/

typedef struct

```

```

{
/*****/

/*          Synchronization Addresses          */
/*****/

unsigned int *SyncIn;          /* ptr to input sync value          */
unsigned int *SyncOut;        /* ptr to output sync value        */


/*****/

/*          Input Data Pointers / Framesizes          */
/*****/

float    *InPtr0;              /* input 0 data pointer            */
float    *InPtr1;              /* input 1 data pointer            */
float    *InPtr2;              /* input 2 data pointer            */
float    *InPtr3;              /* input 3 data pointer            */
float    *InPtr4;              /* input 4 data pointer            */
float    *InPtr5;              /* input 5 data pointer            */
float    *InPtr6;              /* input 6 data pointer            */


unsigned int FramesizeIn0;     /* input 0 framesize              */
unsigned int FramesizeIn1;     /* input 1 framesize              */
unsigned int FramesizeIn2;     /* input 2 framesize              */
unsigned int FramesizeIn3;     /* input 3 framesize              */
unsigned int FramesizeIn4;     /* input 4 framesize              */
unsigned int FramesizeIn5;     /* input 5 framesize              */
unsigned int FramesizeIn6;     /* input 6 framesize              */


/*****/

/*          Output Data Pointers / Framesize          */
/*****/

```

```

float    *OutPtr0;           /* output 0 data pointer    */
float    *OutPtr1;           /* output 1 data pointer    */
float    *OutPtr2;           /* output 2 data pointer    */
float    *OutPtr3;           /* output 3 data pointer    */

unsigned int FramesizeOut0;   /* output 0 framesize      */
unsigned int FramesizeOut1;   /* output 1 framesize      */
unsigned int FramesizeOut2;   /* output 2 framesize      */
unsigned int FramesizeOut3;   /* output 3 framesize      */

/*****
/*                               Block Parameters                               */
*****/

float    _Vbase;              /* define scaling factors   */
float    _Ibase;
float    _Ls;
float    _Samplingrate;
float    _Vscale;
float    _Udcscale;
float    _Iscale;
float    _Udc_ref;

```

```

/*****/
/*          Internal Persistent Variables          */
/*****/

/* NOTE: If your block requires persistent variables, then they must be put here.          */
/* DO NOT USE global variables or statics in the C source file unless you really want          */
/* them to be shared by all instances of this block. If you do add variables, update the          */
/* "#define INTVAR_SIZE" in the PC's header (.h) file to reflect the proper internal          */
/* variable storage.                               */
/*****/

} PARAMS;

/*****/
/*          FUNCTION PROTOTYPES          */
/*****/

void neur2_INT (PARAMS *pParam);    /* optional block interrupt routine          */
void neur2_INIT (PARAMS *pParam);   /* optional block initialization routine          */
void neur2_STOP (PARAMS *pParam);   /* optional block stop routine          */
void neur2_RESTART (PARAMS *pParam); /* optional block restart routine          */
void neur2(PARAMS *pParam);         /* main block routine          */
void init_data(void);               /* initialise weight,input and output data          */
void nnop_calc(void);               /* feedforward output calculation          */
void error_calc(void);              /* error vector calculation          */
void backprop(void);                /* error backpropagation weight update          */
void input(double temp[]);
void v_s_albe_out_calc(void);
void i_s_albe_hat_calc(void);

```

C.2.2 ANN Current Controller Block C-source code

This program is the COT ANN current controller for a boost rectifier. This code is used to calculate the COT ANN current controller's sampling times when controlling a boost rectifier, as shown in Fig. 5.1 Chapter 5, section 5.1. A flow diagram is provided in section 5.1, Fig. 5.2.

```

/*=====//
//      FILE NAME : _neur2.c                                //
//      BLOCK NAME: ANNtest                                  //
//      GROUP NAME: neural network                          //
//      PURPOSE   : Provides the real-time block's DSP C source code for an ANN //
//                  current controller for a Boost Rectifier. //
//      Hypersignal Block Wizard Version 4.00.14 Auto-Generated Block //
// //
//      Number of Inputs : 7                                //
//      Number of Outputs: 4                                //
// //
//      Creation Date: Thu - 28 May 1998                    //
//      Creation Time: 03:01 PM                              //
//=====*/
#include "_neur2.h"                /* Include necessary files */
#include <stdlib.h>
#include <math.h>

/*****
/*          Optional real-time block interrupt routine          */
/*****
/* If this routine is activated, it will be called in response to a selected DSP interrupt. */

```

```

/* If this routine is not activated, the main block routine will be called in response to a */
/* selected DSP interrupt.                                                                */
/*****
/*
void neur2_INT(PARAMS *pParam)
{
}

*/

/*****
/*          Optional real-time block initialization routine                            */
/*****
/* If this routine is activated, it will be called one time before the main application begins. */
/* This allows for any required software or hardware initialization to be performed before */
/* the block executes.                                                                */
/*****
/*
void neur2_INIT(PARAMS *pParam)
{

}

*/

/*****
/*          Optional real-time block stop routine                                    */
/*****
/* If this routine is activated, it will be called whenever the block diagram worksheet's */

```

```

/* execution is stopped. Blocks that deal with hardware may need this routine to stop */
/* the hardware's execution. */
/*****/
/*
void neur2_STOP(PARAMS *pParam)
{

}
*/

/*****/
/* Optional real-time block restart routine */
/*****/
/* If this routine is activated, it will be called whenever a block diagram worksheet is */
/* executed after being stopped. Blocks that deal with hardware may need this routine to */
/* restart the hardware's execution. */
/*****/
/*
void neur2_RESTART(PARAMS *pParam)
{

}*/

/*****/
/* Real-time block routine */
/*****/
/* This is the main block routine. It is called during each loop of the main application. */
/* If an interrupt is selected, and the interrupt routine above is not activated, this routine */
/* will be called in response to the selected interrupt instead of during the main */

```

```

/* application loop. */
/*****/
void neur2(PARAMS *pParam)
{
    unsigned int index;          /* index for frame processing loop */
    double  Ts, Valpha, Vbeta, Ialpha, Ibeta, Ialpha_ref, Ibeta_ref;
    double  Ids_ref, Iqs_ref, nncurcon_inputs[6];
    static  float theta_e;
    static  int nnn=0;
    a= (53.0/360.0)*2.0*3.1415;    /* define phase adjustment angle */

    Vbase = (pParam->_Vbase);      /* read in scaling factors from dialog box */
    Ibase = (pParam->_Ibase);
    Ls1 = (pParam->_Ls);
    Samplingrate = (pParam->_Samplingrate);
    Vscale = (pParam->_Vscale);
    Udcscale = (pParam->_Udcscale);
    Iscale = (pParam->_Iscale);
    Udc_ref = (pParam->_Udc_ref);

    /*****/
    /*Update and read all inputs from system */
    /*****/

    Usa = *(pParam->InPtr0);        /* Read in phase A phase voltage */
    Usa = (7.4*Vscale*(Usa+80.0))/32767.0; /* Scale phase A voltage */
    Usb = *(pParam->InPtr1);        /* Read in phase B phase voltage */
    Usb = (7.5*Vscale*(Usb+50.0))/32767.0; /* Scale phase B voltage */
    Usc = *(pParam->InPtr2);        /* Read in phase C phase voltage */
    Usc = (7.9*Vscale*(Usc))/32767.0; /* Scale phase C voltage */

```

```

Isa = *(pParam->InPtr3);          /* Read in phase A line current */
Isa = (10.4*Iscale*(Isa-40.0))/32767.0; /* Scale phase A current */
Isb = *(pParam->InPtr4);          /* Read in phase B line current */
Isb = (10.94*Iscale*(Isb+137.0))/32767.0; /* Scale phase B current */
Isb = *(pParam->InPtr5);          /* Read in phase C line current */
Isb = (10.4*Iscale*(Isb+10.0))/32767.0; /* Scale phase C current */
Udc = *(pParam->InPtr6);          /* Read in DC link voltage */
Udc = (11.0*Udcscale*(Udc-22.0))/32767.0; /* Scale DC link voltage */

/*****
/*Mains synchronisation block, see Chapter 4 pp 4.7 Eqn. (4.7) through (4.12)
*****/

Eamains = (Usa-0.5*Usb-0.5*Usc)*sqrt(2.0/3.0); /* calculate the alpha/beta comp. */
Ebmain = (Usb - Usc)/(sqrt(2.0)); /* of the mains space vector */

Eamainsn= Eamains*cos(a) - Ebmain*sin(a); /*phase adjust the alpha/beta voltages*/
Ebmainn=Eamains*sin(a)+Ebmain*cos(a);

MagUs = sqrt(Eamainsn*Eamainsn+Ebmainn*Ebmainn); /* calculate mains voltage */
/* space vector */

EApu = Eamainsn/MagUs; /* calculate per unit alpha/beta voltages */
EBpu = Ebmainn/MagUs;

Iamains = (Isa-0.5*Isb-0.5*Isb)*sqrt(2.0/3.0); /*calculate alpha/beta current comp. */
Ibmain = (Isb - Isc)/(sqrt(2.0));
Imag=sqrt(Iamains*Iamains+Ibmain*Ibmain); /*calculate current magnitude */

```

```

/*****/
/*Input synchronised values into ANN */
/*****/

#define dum1 Iamains/Imag          /* prepare inputs to ANN */
#define dum2 Ibmain/Imag
#define dum3 EApu
#define dum4 EBpu

Ts = Samplingrate;

if(nnn==0)
{
    srand(0);
    init_data();
    Cv=(K)*(Ts/Ls1)*(13.5/107.0);    /*calculate voltage constant` */
}

/*****/
/* Voltage PI loop, see Chapter 4, pp 4.11, Eqn. (4.18) */
/*****/

UdcError = Udc_ref - Udc;          /*calculate DC link error */
IvOutk1 = IvOut;                   /* store previous integrator value */
CvOutk1 = CvOut;                   /* store previous voltage controller output */
Idmax = 1.0;                       /* set d-axis current limit */

IvOut = ((0.273*(UdcError)*Samplingrate)+IvOutk1);    /* perform integration */
CvOut=((1.0+200.0*Samplingrate)/(1.0+50.0*Samplingrate))*(IvOut) +
(CvOutk1-IvOutk1)/(1.0+50.0*Samplingrate);    /* perform PI control */

```

```

/*****/
/*Define all reference currents, see Chapter 4, pp 4.9, Eqns. (4.13) and (4.14) */
/*****/

Ids_ref= CvOut;          /* set d-axis current reference */
if (Ids_ref > Idmax)      /* check that reference is below the max limit*/
    {Ids_ref = Idmax;}
if (Ids_ref < -Idmax)
    {Ids_ref = -Idmax;}

Iqs_ref = 0.0;           /* set q-axis current to zero to achieve unity pf*/

Ialpha_ref = Ids_ref*EApu-Iqs_ref*EBpu;    /* calc alpha reference current */
Ibeta_ref = -(Ids_ref*EBpu+Iqs_ref*EApu);  /* calc beta reference current */

/*****/
/*Pass values to ANN Backpropagation algorithm */
/*****/

nncurcon_inputs[0] = Ialpha_ref;
nncurcon_inputs[1] = Ibeta_ref;
nncurcon_inputs[2] = dum3;
nncurcon_inputs[3] = dum4;
nncurcon_inputs[4] = dum1;
nncurcon_inputs[5] = dum2;

/*****/
/*                               Determine States                               */
/*****/

input(nncurcon_inputs);

```

```

nnop_calc();

if(nnn != 0)
{
    error_calc();
    backprop();
}

v_s_albe_out_calc();

/*****
/*
Determine Outputs
*/
*****/

*(pParam->OutPtr0) = (32767.0*(nncurcon_outputs[0])/Vbase); /* Valpha volts */
*(pParam->OutPtr1) = (32767.0*(nncurcon_outputs[1])/Vbase); /* Vbeta volts */
*(pParam->OutPtr2) = Ialpha_ref*32000.0;
*(pParam->OutPtr3) = Ibeta_ref*32000.0;
    nnn=nnn+1;
}

/*****
/*Perform all sub-routine functions
*/
*****/
/*Organises all ANN inputs into correct variables
*/
*****/

void input(double temp[])
{
    int i;

```

```
FT *px=x-1, *px_old=x_old-1, *pi_s_albe=i_s_albe, *pv_s_albe=v_s_albe;
```

```
FT *pi_s_albe_des=i_s_albe_des, *pv_s_albe_in=v_s_albe_in;
```

```
*(pi_s_albe_des)= temp[0];                /* -> i_s_albe_des[0]                */
```

```
*(pi_s_albe_des+1)=temp[1];                /* -> i_s_albe_des[1]                */
```

```
*(pv_s_albe_in)=temp[2];
```

```
*(pv_s_albe_in+1)=temp[3];
```

```
*(pi_s_albe)=temp[4];                      /* -> i_s_albe[0]                      */
```

```
*(pi_s_albe+1)=temp[5];                    /* -> i_s_albe[1]                      */
```

```
for(i=0;i<n;i++)
```

```
    *(px_old+=1)=*(px+=1);
```

```
px=x;
```

```
*(px+9)=*(pv_s_albe+1);
```

```
*(px+8)=*(pv_s_albe);
```

```
*(px+7)=*(px+1);
```

```
*(px+6)=*(px);
```

```
*(px+5)=*(px+3);
```

```
*(px+4)=*(px+2);
```

```
*(px+3)=*(pv_s_albe_in+1);
```

```
*(px+2)=*(pv_s_albe_in);
```

```
*(px+1)=*(pi_s_albe+1);
```

```
*(px)=*(pi_s_albe);
```

```
}
```



```

/*****/
/*Calculate all outputs for ANN */
/*****/

void v_s_albe_out_calc()
{
    FT *pv_s_albe=v_s_albe, *py=y, *pi_s_albe_des=i_s_albe_des;
    FT *pv_s_albe_out=v_s_albe_out;
    static int nn=0;

    *pv_s_albe=(((pi_s_albe_des)-(*py))/(Cv));      /* calc alpha/beta voltages */
    *(pv_s_albe+1)=(((pi_s_albe_des+1)-(*py+1)))/(Cv);

    if(*(pv_s_albe)>1) *pv_s_albe=1;      /* scale alpha/beta voltages < 1 per unit */
    if(*(pv_s_albe+1)>1) *(pv_s_albe+1)=1;
    if(*(pv_s_albe)<-1) *pv_s_albe=-1;
    if(*(pv_s_albe+1)<-1) *(pv_s_albe+1)=-1;

    *(pv_s_albe_out)=(-(pv_s_albe)+(EApu))*Vbase/2.0;
    *(pv_s_albe_out+1)=(-(pv_s_albe+1)+(EBpu))*Vbase/2.0;

    nncurcon_outputs[0]=*(pv_s_albe_out); /* v_s_albe_out[1] -> */
    nncurcon_outputs[1]=*(pv_s_albe_out+1); /* v_s_albe_out[2] -> */

    if((nncurcon_outputs[0])>Vbase) nncurcon_outputs[0]=Vbase; /* scale voltages */
    if((nncurcon_outputs[1])>Vbase) nncurcon_outputs[1]=Vbase;
    if((nncurcon_outputs[0])<-Vbase) nncurcon_outputs[0]=-Vbase;
    if((nncurcon_outputs[1])<-Vbase) nncurcon_outputs[1]=-Vbase;
}

```

```

/*****
/*Optimise I/O for ANN
/*****
void nnop_calc()
{
    int row, col, opnum;
    FT z[1];
    FT *pz=z,*pd=d-1,*pd_old=d_old-1,*pV=V-1,*pW=W-1,*px=x-1,*py=y-1;

    for(row=0;row<m;row++){
        px=x_old-1;
        py=y-1;
        *pz=0.0;

        for(col=0;col<n;col++){
            *pz+=*(pW+=1)*(*(px+=1));
        }
        *(pd_old+=1)=*(pd+=1);
        *(pd)=1.0/(1.0+EXP(-(*pz)));
        for(opnum=0;opnum<r;opnum++){
            if(row==0)
                *(py+1)=(FT)0.0;
            *(py+=1)+=*(pV+=1)*(*pd);
        }
    }
}

```

```

/*****
/*Calculate error for Backpropagation                                     */
/*****

void error_calc()
{
    FT *pi_s_albe=i_s_albe, *pi_s_albe_hat=i_s_albe_hat, *pe=e;
    FT *pi_s_albe_des=i_s_albe_des;

    *pe= (*pi_s_albe)-(*pi_s_albe_hat);
    *(pe+1)= (*pi_s_albe+1)-(*pi_s_albe_hat+1));

    *pi_s_albe_hat=*pi_s_albe_des;          /* for constant Cv          */
    *(pi_s_albe_hat+1)=*(pi_s_albe_des+1);
}

/*****
/*Perform Backpropagation function                                     */
/*****

void backprop()
{
    int row, col, opnum;
    FT *pW=W-1, *pV=V-1, *pdW=dW-1, *pdV=dV-1, *pd=d_old-1, *px,temp, *pe;

    for(row=0;row<m;row++){
        pd+=1;
        px=x-1;
        pe=e-1;
        temp=0.0;
        for(opnum=0;opnum<r;opnum++){

```

```

    temp+=(*pe+=1))*(*pV+=1));
}
temp*=B>(*pd)*(1.0-(*pd));
for(col=0;col<n;col++){
    pdW+=1;
    *pdW=M>(*pdW)+temp>(*px+=1));
    *(pW+=1)+=*pdW;
}
}

```

```

pd=d_old-1;
pV=V-1;
for(row=0;row<m;row++){
    temp=B>(*pd+=1));
    pe=e-1;
    for(opnum=0;opnum<r;opnum++){
        pdV+=1;
        *pdV=M(*pdV)+temp>(*pe+=1));
        *(pV+=1)+=*pdV;
    }
}
}

```

```

/*****
/*Setup initial weight states                                     */
/*****

```

```

void init_data()
{
    int row,col;

```



```
FT *p;

p=W-1;                                /* initialise W                                */
for(row=0;row<m;row++){
    for(col=0;col<n;col++)
        *(p+=1)=wscale*(FT)rand()/((FT)RAND_MAX+(FT)1.0)+wshift;
}

p=V-1;                                /* initialise V                                */
for(row=0;row<m;row++){
    for(col=0;col<r;col++)
        *(p+=1)=wscale*(FT)rand()/((FT)RAND_MAX+(FT)1.0)+wshift;
}
}
```

APPENDIX D

BOOST RECTIFIER HARDWARE PROFILE

D.1 Overview

This Appendix provides an overview of the construction of the IGBT Voltage-Sourced Inverter topology, used to realise the practical boost rectifier system for the investigation in Chapter 5. The signal sensing and conditioning circuitry for the current, and voltage feedback is discussed. The design and operation of the PWM Hanning controller card will be described, along with the optical fibre link to the boost rectifier power circuitry.

For the practical implementation and verification of the design of the ANN current controller, and the system simulations performed in chapter 4, a hardware and software platform is required that can realise the necessary real time control strategies. Innovative Integrations ADC64 Digital Signal Processing (DSP) card was chosen as the hardware platform, as it allows for fast data acquisition and control. Hyperceptions Hypersignal Real time Integrated Design Environment (RIDE) [Hypersignal1] was chosen as the software platform, as it supports both real-time and simulation DSP analysis, in addition to providing support for user-written and user-defined functions or algorithms for unique situations. The software not only allows for limitless simulation, analysis and design, but also provides support for a number of plug-in DSP hardware cards (such as the ADC64 card) to allow associated real-time signal processing.

D.2 Practical System Structure

Fig. D.1 shows the practical system in block diagram form.

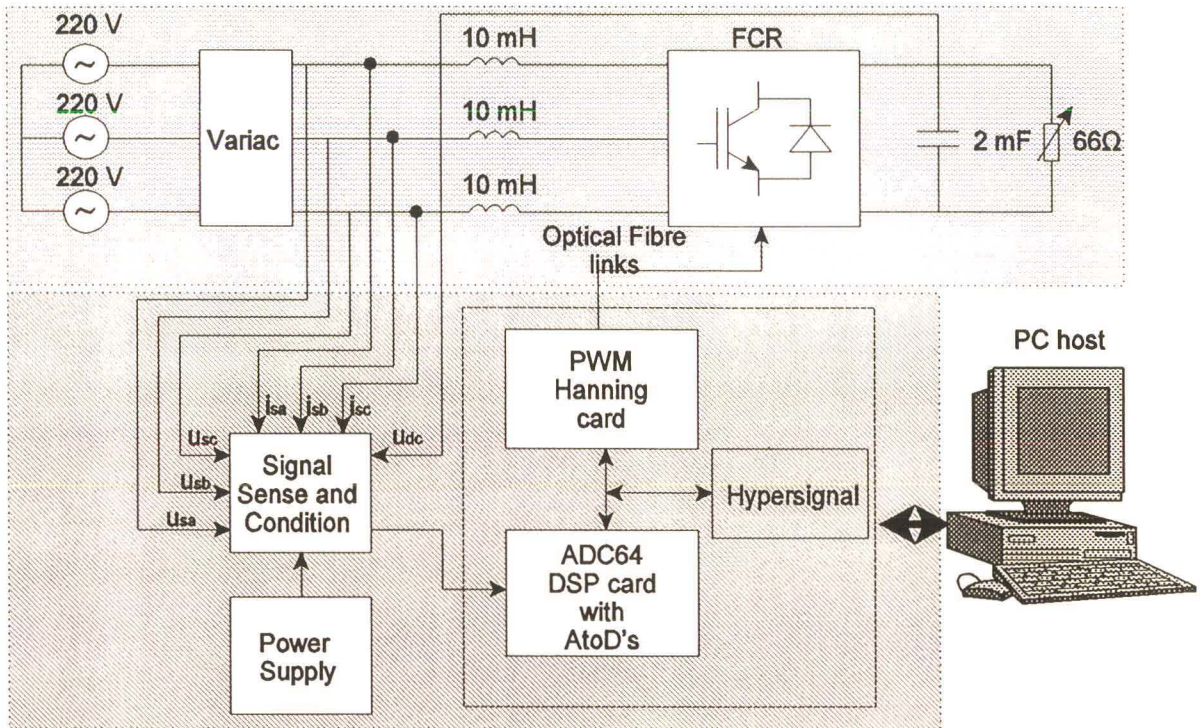


Fig. D.1 Structure of practical system

The components of the hardware system, are the IGBT Forced-commutated rectifier, the analogue signal sensing and conditioning circuitry, the PWM Hanning card, the ADC64 DSP card and a personal computer to host the cards and to interface between the software and hardware platforms.

The signal sensing and conditioning circuitry senses the boost rectifier's line currents and the system phase voltages, and scales them to lie in a range ± 10 V. The scaled voltage signals are then applied to the Analog to Digital (A/D) converters onboard the ADC64 DSP card, for conversion into digital form for use by the on-board TMS320C32 DSP chip. The DSP uses the digital representation of the system currents and voltages to implement the user-defined ANN current controller. The required pulsewidth modulator inputs are calculated by the real-time

ANN current controller software, then outputted via the ADC64 cards 3x bus, to the PWM hardware which generates the boost rectifier switching signals according to the third-harmonic injection technique discussed in chapter 2, to achieve sinusoidal three-phase line currents, with a regulated DC link voltage.

The following sections describe the hardware subsections in Fig. D.1 in greater detail. The power circuit, consisting of the three-phase AC voltage supply, the IGBT boost rectifier and the electrical load is described, after which the control hardware and software is discussed.

D.3 The Power Circuit

The detailed design of the boost rectifier power circuit in Fig. D.1 involves the selection of the AC and DC side reactive component sizes, and the selection of the power rating (or current and voltage ratings) of the power switches used for the design. These both depend on the maximum permissible PWM switching frequency, which in turn is determined by a number of other factors, such as the maximum permissible switching frequency that the IGBT's are capable of, the capabilities of the driver modules, the computational speed of the microcontroller or DSP, and the physical layout of the boost rectifier power circuit and driver modules. In particular, the physical layout of the boost rectifier power circuit must be designed to ensure that stray inductances are minimised, in order to make possible the use of high PWM switching frequencies. Stray inductance is a severe problem, and can cause high frequency switching schemes that work well in theory, to fail in practice.

Minimisation of the reactive energy storage units, such as the line inductors and DC link capacitor, require a detailed analysis of the system, using tools such as the Fourier Series. This is considered beyond the scope of this thesis. Rather, the reactive components are selected according to availability within the department. CASED simulations are performed to validate the use of the available energy storage components, to achieve the desired system response.

D.3.1 The Insulated Gate Bipolar Transistor Boost Rectifier

The boost rectifier power conversion system consists of three power modules, each comprising two Insulated Gate Bipolar Transistors (IGBT's), and two power diodes as shown in Fig. D.2.

The IGBT possesses a number of beneficial characteristics that have made it a popular choice of power switch in low and medium power inverter and converter applications. The IGBT combines the high input gate impedance of the MOSFET, BJT like conduction characteristics and a number of other beneficial characteristics, which together result in the advantages such as simpler gate drive circuitry, easy device protection, snubberless operation, and the ability to be switched at high frequencies.

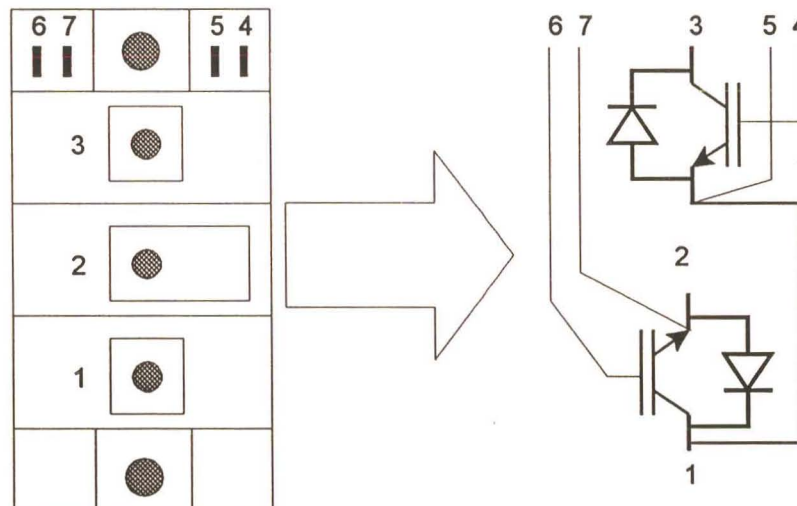


Fig. D.2 IGBT power module

The IGBT's are driven using commercially available IGBT driver modules [Semikron1]. Each SKHI 21 driver module drives two IGBT's, and provides functions such as short circuit protection, galvanic isolation between the power circuit and the control electronics (pulse transformers), narrow pulse suppression, and switching dead-time generation with a minimum dead-time of 2.7 μsec . The driver inputs are CMOS compatible, and they possess Schmitt trigger characteristics to suppress false trigger pulses:- turn-on and off pulses shorter than 0.5 μsec are not transmitted. The driver modules are also capable of switching frequencies up to 100 kHz,

which is beneficial when using PWM switching schemes.

The IGBT boost rectifier with its associated driver modules are displayed in Fig. D.3.

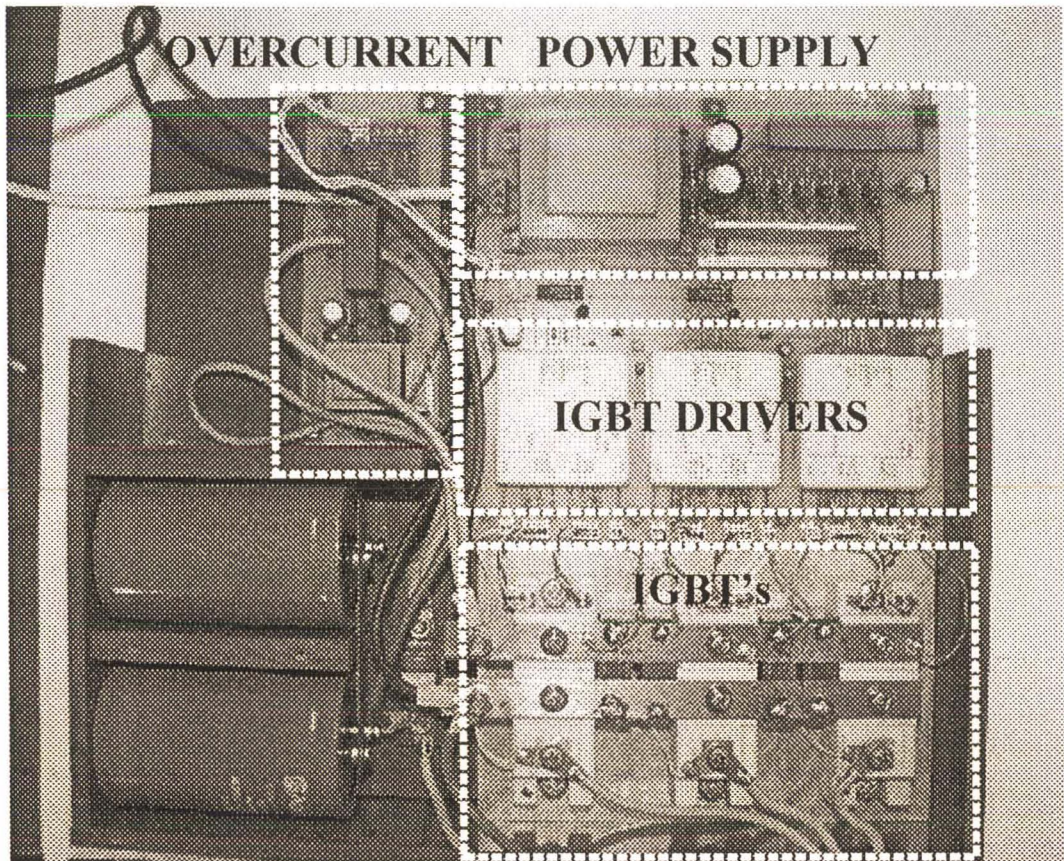


Fig. D.3 IGBT boost rectifier and associated driver modules

D.3.2 Electrical load

The variable resistance load bank is realised as a series connection of three individual resistance units, each with a maximum DC voltage rating of 250 V, and a resistance stepped range of 5 to 80 Ω , giving the resistor bank a maximum DC voltage rating of 750 V assuming equal resistances in each unit.

D.3.3 Three-phase Supply

A three-phase Variac is used to reduce the peak AC phase voltages, so as to reduce the peak DC link voltages, this allows a safety margin with respect to the resistor bank voltage rating.

D.4 The Control Hardware

This section describes the DSP based ANN current controller used for the control of the boost rectifier power circuit. Basically the control hardware consists of an ADC64 PCI bus DSP card, a PWM Hanning ISA bus card, an optical fibre interface between the controller and the boost rectifier driver circuitry, and signal sensing and conditioning circuitry. The boost rectifier a-b-c three-phase voltages and line currents are sensed and converted to voltages in the range ± 10 V. The ADC64 cards onboard A/D's sample the voltage and current values from the conditioning circuit, and then utilises the values for real-time mains synchronisation process. Thereafter the PWM Hanning alpha/beta controlling voltages are updated via the ADC64's 3x bus expansion header, to achieve the desired system response. The controlling signals are then transferred to the boost rectifier driver circuitry via an optical fibre link, which is used to maintain isolation between the DSP based controller and boost rectifier power circuit.

D.4.1 ADC64 Digital Signal Processing Card

The ADC64 card is a complete data acquisition system, it offers a very high performance DSP-based computing engine with determinant timing for fast data acquisition and control to complement the host platform. Using the ADC64 card allows the host to be free from time-critical events. The card can operate under windows and other operating systems where real-time performance is an after-thought. General applications of the DSP card are:

- 1) Data acquisition,
- 2) Data analysis,

- 3) Intelligent digital control systems,
- 4) Acoustic processing, and
- 5) Co-processing for the PC.

Fig. D.4 shows a block diagram of the ADC64 cards architecture [III].

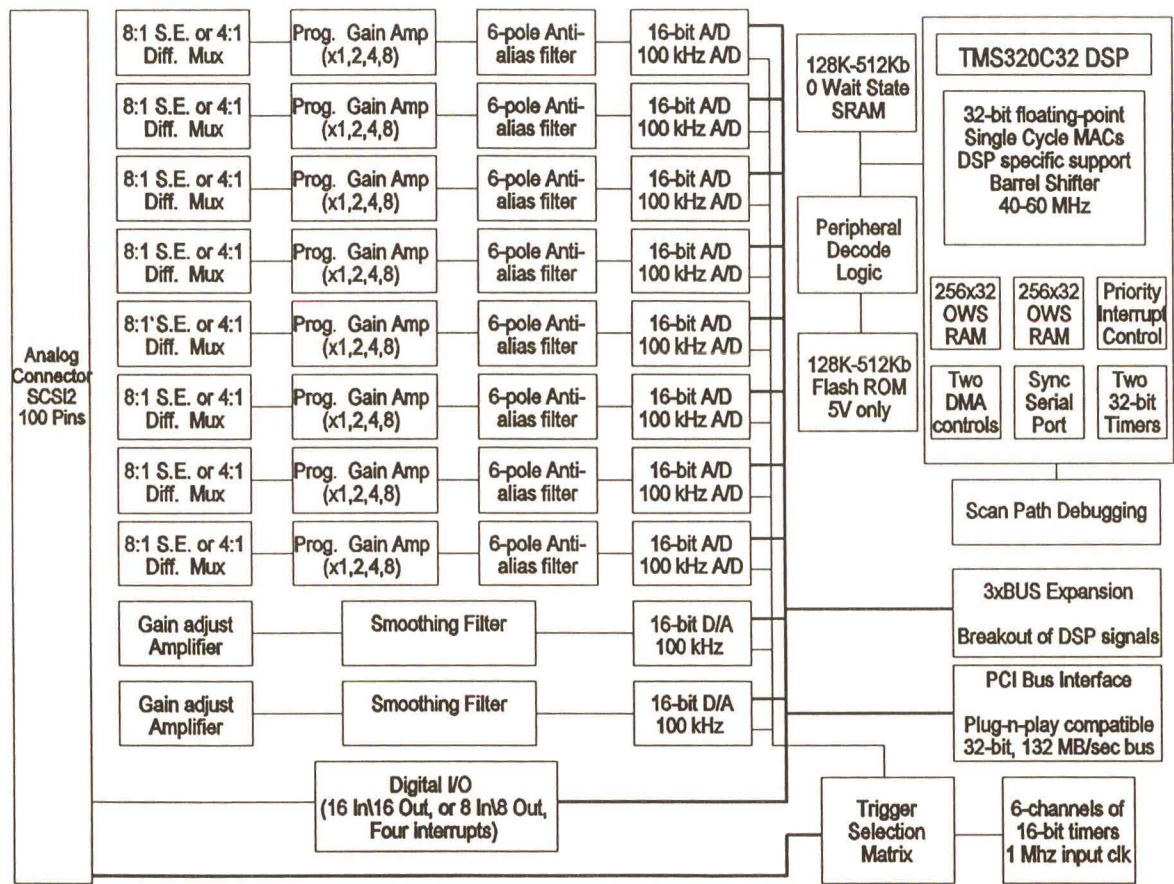


Fig. D.4 ADC64 Block Diagram

Fig. D.5 provides the board layout for the Innovative Integration PCI bus ADC64 DSP card.

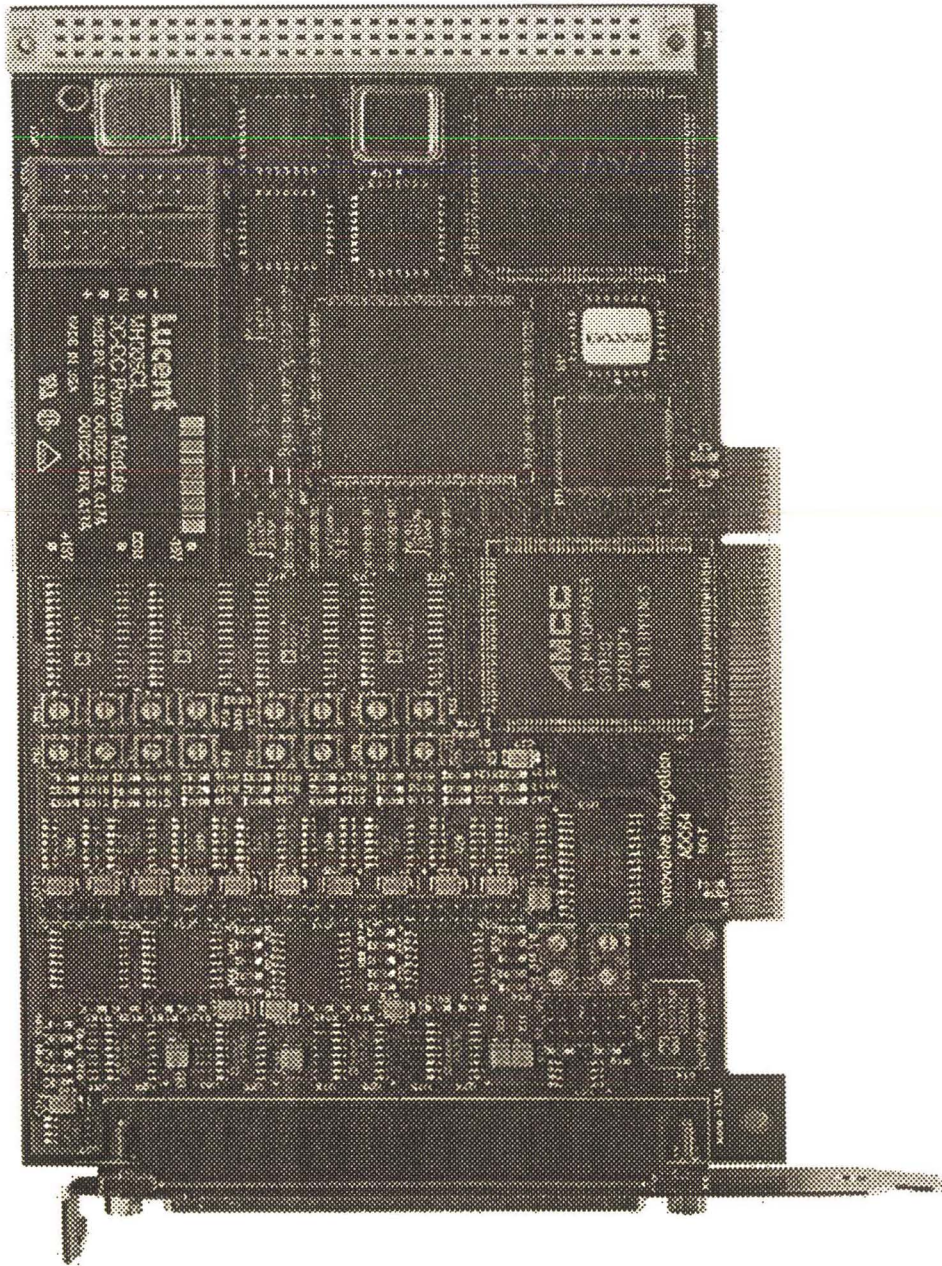


Fig. D.5 ADC64 PCI bus DSP card layout

TMS320C32 Processor Core and interrupts

The TMS320C32 processor and peripherals represent the bulk of the onboard hardware implemented on the ADC64 DSP card. Fig. A.4 shows that the card is designed for a bank of 128K x 32 SRAM, 64 channels of instrumentation grade 16-bit analog I/O, two 16-bit D/A channels, and 16 bits of bidirectional digital I/O. The ADC64 card also provides the option to generate interrupts to the onboard TMS320C32 processor from on- and off-board sources. The interrupt sources are user selectable via an onboard jumper header, as shown in Table D.1.

Interrupt Input	Source	JP17 Setting
EI0	PCI write complete	1-2
EI1	PCI read composer	3-4
EI2	A/D interrupt	5-6
EI3	PC interrupt to DSP	7-8

Table D.1 Interrupt source settings

The TMS320C32 DSP default is to have edge sensitive interrupts, as they tend to be the easiest to work with, and typically eliminate the possibility of multiple interrupt triggers from ill-conditioned interrupt sources. The various interrupt sources for the ADC64 card are as follows:

- 1) EI0, EI1, EI2, and EI3: These are the processor interrupt input pins. They can be used to signal an outside event requiring the ADC64's attention, to invoke either an interrupt routine or initiate a DMA transfer,
- 2) PC_INT: This is an output from the host PC,
- 3) PCI write complete: A write to the PCI FIFO has been transferred and FIFO is ready for new data,
- 4) PCI read complete: The PCI FIFO is empty, no new data is available, and
- 5) A/D status: one of the A/D's needs service or a D/A needs new data loaded.

Memory Map

The ADC64's memory is mapped to three areas of functionality, based around the three external address decoding strobes: STRB1, STRB0, and IOSTROBE.

The STRB1 space implements the fast external SRAM bank, from which the users software is run. STRB0 is space used for the FLASH boot ROM on the ADC64 card. The IOSTROBE region is used to map all external onboard peripherals including the A/D's, D/A's, digital I/O, and the bus interface features. Two user definable strobes, DECODES0 and DECODES1, are included in the IOSTROBE space and presented for external interface purposes on a 96-pin 3X BUS expansion header. This allows for relatively simple expansion of external hardware without need for additional decoding.

Analog I/O

The analog input section of the ADC64 card, features as many as 64 16-bit channels, each configured with a high input impedance instrumentation amplifier, to permit direct connection to sensors and signal sources. There are 8 A/D's on the ADC64 card, which may be sampled simultaneously. The A/D channels are instrumentation grade and have full calibration for accurate DC results, with an input range between ± 10 V. Fig. D.6 shows a simple block diagram of a single channel analog input system.

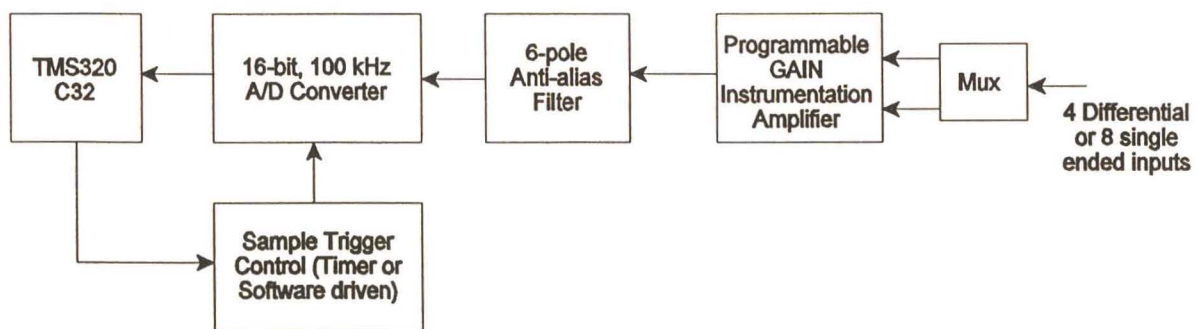


Fig. D.6 Analog input circuitry

The A/D's are memory mapped read-only peripherals in the TMS320C32 IOSTROBE space, which are configured as pairs, and read back as 32-bit numbers. For the 64 channel cards, the

first channel is mapped to data bits D₀ ... D₁₅, and the second channel to the upper half of the data bus as bits D₁₆ ... D₃₁. Each A/D device is provided with independent 6-pole anti-alias filters with offset adjustment. These filter circuits invert the incoming analog signal, thus correction must be made when interpreting the inputs to obtain the desired polarity.

Each A/D also has an independent programmable gain, the magnitude of which is selected by writing a number to the programmable gain amplifier. The A/D's are multiplexed either 8:1 single-ended or 4:1 differentially inputted, thus providing the ADC64 card with 64 or 32 A/D inputs respectively. The analog values are inputted to the ADC64 card via the SCSI2 end connector [II1].

Analog Output

The ADC64 DSP card incorporates two independent 200 kHz, 16-bit instrumentation grade digital to analog converters, producing outputs between + 10 V and - 10 V. This allows signals to be passed from the ADC64 card to external offboard peripherals for control or monitoring purposes. The D/A channels are amplified and filtered with high speed, low offset op-amps using an inverting topology. The D/A output value is outputted via the ADC64 cards SCSI2 end connector.

The next section will describe the generation of PWM control signals for the control of a boost rectifier using the Hanning PBM 1/87 ASIC. A brief description will be provided for the design and implementation of the PWM Hanning card.

D.4.2 Hanning PWM Interface Card

The Hanning-PBM is a slave peripheral used for the purpose of generating three-phase pulsewidth modulation signals, suitable for use in VSI AC drives and boost rectifier control applications. The PWM ASIC generates the entire PWM pulse pattern independently, thereby relieving the host processor from intensive processing and time critical calculations [HANNING1]. The Hanning-PBM implements the third-harmonic injection PWM scheme described in earlier chapters. The pulsewidth modulator provides a maximum PWM switching frequency of 20 kHz, and is configured to interface to either a 8- or 16-bit processor.

U_a , U_b , δ , T_{off} , T_{dead} , T_{min} , VORTL and T_{start} are required as inputs to the pulsewidth modulator, for the purposes of the Boost rectifier Controller. T_{off} , T_{dead} , and T_{min} are used to determine the minimum PWM output pulse off-time, PWM pulse dead-time and the minimum PWM pulse on-time respectively; VORTL and T_{start} are used to set the PWM switching frequency. U_a , U_b , and δ are used to calculate the modulating signal amplitude, U , and phase angle, ϕ , as described previously. The PWM ASIC uses the input information, to control the boost rectifier based power conversion system, to achieve the desired response, while achieving sinusoidal line currents at a unity power factor.

Bus Operating Modes

The PWM module is software configurable to interface to either an 8- or 16-bit bidirectional bus (DB₀ ... DB₁₅). In 8-bit mode the A0 address bit controls the multiplexing of the lower (DB₀ ... DB₇) and upper (DB₈ ... DB₁₅) bytes of the data word, by toggling between 0 and 1 respectively. Both bytes may be connected directly to the 8 data lines of the host processor, because when a byte is placed on the bus, the remaining byte is automatically placed in high impedance mode for the duration of that data transfer. Table D.2 summarises the host bus pin functions in the 8-bit mode.

/RD	/WR	/CE	A0	A1	
X	X	1	X	X	high-Z, write disabled
1	1	0	X	X	high-Z, write disabled
0	1	0	0	0	DB0...DB7=data byte,DB8...DB15=high-Z
0	1	0	1	0	DB0...DB7=high-Z, DB8...DB15=data byte
0	1	0	0	1	DB0...DB7=status byte, DB8...DB15=high-Z
0	1	0	1	1	DB0...DB7=high-Z, DB8...DB15=status byte
1	0	0	0	0	write DB0...DB7 to data register
1	0	0	1	0	write DB8...DB15 to data register
1	0	0	0	1	write DB0...DB7 to control register
1	0	0	1	1	write DB8...DB15 to control register
0	0	0	X	X	not possible

Table D.2 8-bit bus mode pin functions

The 16-bit bus mode is selected when bit 7 of the status register (or BUS16 bit) is set high (1). Since the 8-bit bus is the default mode, address bit A0 must be set to zero (0) in order to write to the BUS16 bit. The A0 address bit has no function in the 16-bit mode; the A1 address bit is now used to distinguish between the control and data registers. During the 16-bit bus mode, all 16 data bits, DB0 ... DB15, are written or read simultaneously. Table D.3 lists the bus pin functions for the 16-bit mode of operation.

/RD	/WR	/CE	A0	A1	
X	X	1	X	X	high-Z, write disabled
1	1	0	X	X	high-Z, write disabled
0	1	0	X	0	DB0...DB15=data word
0	1	0	X	1	DB0...DB15=status word
1	0	0	X	0	write DB0...DB15 to data register
1	0	0	X	1	write DB0...DB15 to control register
0	0	0	X	X	not possible

Table D.3 16-bit bus mode pin functions

Internal Registers

The PWM chip possesses two internal 16-bit registers, namely, the CONTROL register for configuring and controlling the chips functions, and the DATA register for writing and reading internal data values. The DATA register can also be used to interface between the PWM module and an external processor. The CONTROL register bits have different functions when read or written.

Processing of Data Words

A number of data inputs are required by the PWM module for the generation of the PWM output pulses. Before writing a data value to the PWM module, the write address WAD0-WAD3 in the CONTROL register must be set to their desired values. Data may be written into the DATA register using one of two possible modes - the polled mode or interrupt mode. In the polled mode, the external processor updates the WAD bits of the CONTROL register, the flag bit, WRFLAG, of the CONTROL register is then polled to determine whether data values may be written to the PWM module. When the flag bit, WRFLAG, is set to zero (0), the data is written into the DATA register. The WRFLAG is then re-polled and the subsequent data values are written when the bit is equal to zero again.

In order to read the PWM modules internal data, the read address bits RAD0-1 must first be set to the pre-defined values [HANNING1], depending on the data value to be read. An internal read cycle is then initiated by the external processor, writing a '1' into the RDSTART bit of the CONTROL register. The RDFLAG bit is then polled until RDFLAG=0, at which point the data value may be read from the DATA register.

PWM Pulse Generation

The PWM switching signal transition times are calculated separately in each switching half-period, from the PWM module inputs at the start of the switching half-period, and then transferred to the internal pulse generation unit at the start of the next switching half-period for the generation of the PWM output pulses. The duration of the switching period is $N_{scale} * 1024$ clock cycles, where N_{scale} is a whole number in the range 1 to 32. N_{scale} is determined from the lower 5 bits of the VORTL input as $N_{scale} = VORTL + 1$. The PWM switching frequency is calculated by Eqn. (D.1):

$$\begin{aligned}
 F_{switch} &= \frac{F_{clock}}{N_{scale} * 1024} \\
 &= \frac{F_{clock}}{(VORTL + 1) * 1024}
 \end{aligned}
 \tag{D.1}$$

Pulse generation is enabled by setting the EIN bit in the CONTROL register, once all the internal data has been written to its respective registers.

When the EIN bit is set, the first processing cycle begins, and the following events take place:

- A) The next half-period switching times are calculated,
- B) At the end of the half-period, the calculated switching times are transferred to the pulse generation unit, and the pulses are initialised,
- C) During the next half-period, at the instant of switching, the corresponding signals are set to '1'.

The PWM output pulse generation can be disabled by setting the EIN bit, in the CONTROL register, to '0'.

PWM Hanning ASIC circuit

A computer based ISA bus PWM card is designed and constructed using the PWM ASIC, and is interfaced to the ADC64 DSP card via its 3x bus expansion header [WALKER1]. The PWM chip is set to operate in 16-bit mode by hardwiring the A0 address bit to zero. The PWM ASIC is interfaced to the DSP card by connecting the D0 ... D15 data bus lines of the ADC64 card to the DB0 ... DB15 PWM ASIC pins. The control inputs, A1, /RD, /WR, and /CE, are connected to DSP card via some decoding circuitry, as shown in Fig. D.7 below

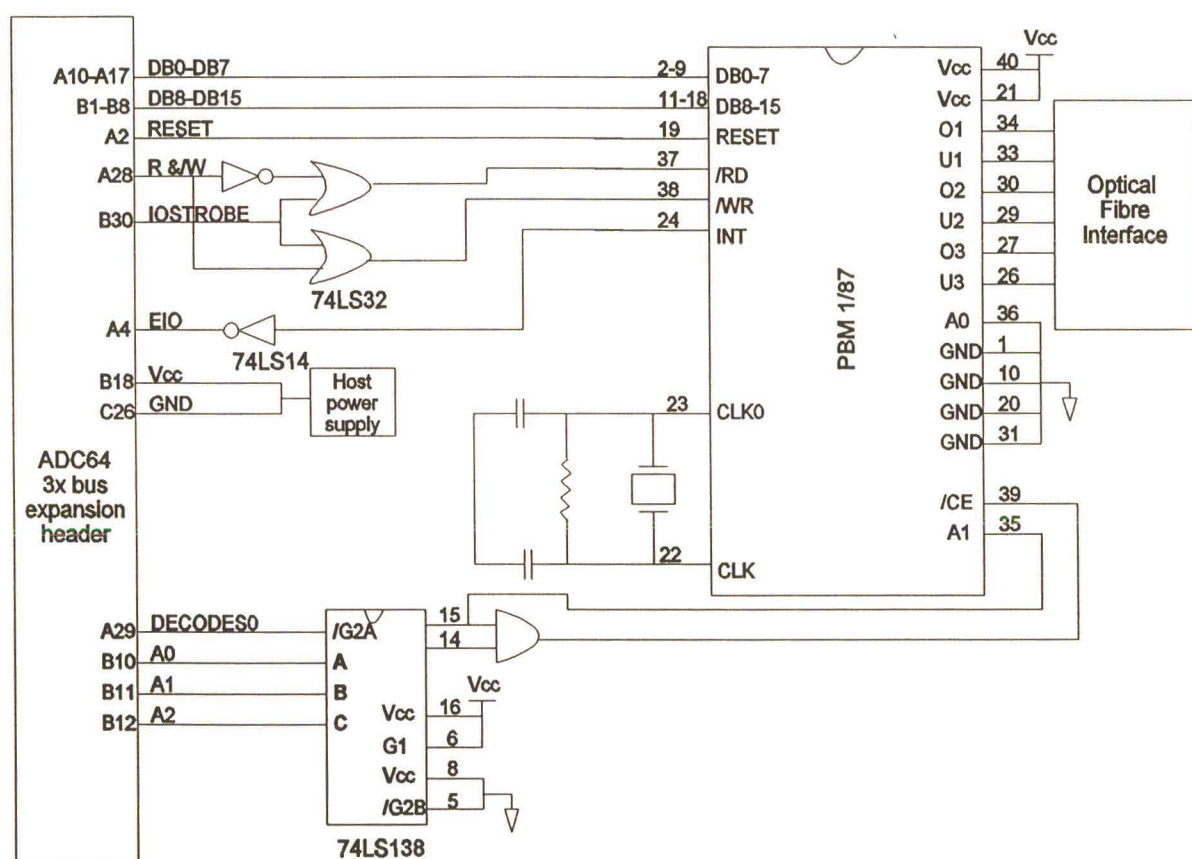


Fig. D.7 PWM circuit diagram

The board layout is shown in Fig. D.8.

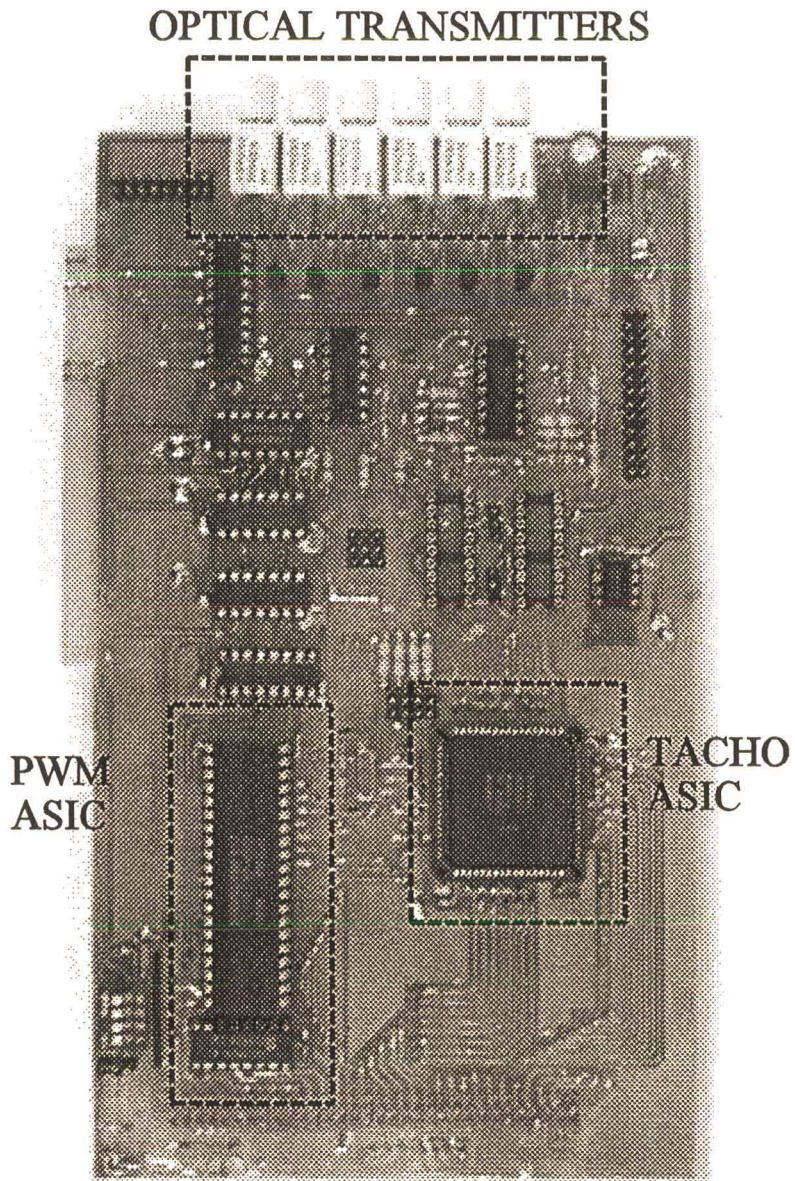


Fig. D.8 PWM Hanning ISA bus card

The next section describes the optical interface between the PWM Hanning card and the boost rectifier power driver circuitry.

D.4.3 Optical Fibre Interface Circuitry

The drivers used to trigger rectifier power circuits are hampered by switching noise, due to the commutation of the power electronic components. Furthermore, the possibility always exists that a spike may be returned to the controller via the interface, thus an electrically uncoupled interface is required between the external control circuitry and the driver circuitry. This can be achieved by using optical fibre interfacing, which allows electrical isolation between the driver circuitry and the controller. The advantage of the fibre optic interface over other electrical decoupling systems is as follows:

- 1) Insensitive to electromagnetic interference,
- 2) No cross talk between lines,
- 3) No arc formation at fibre ends or breaks,
- 4) No ground loops, and
- 5) Reduced weight and volume.

The use of glass fibre optic is restricted due to their relatively high cost, but this disadvantage has been overcome by the introduction of Siemens plastic fibre optic. The plastic fibre optic is simple to use, and is competitively priced. Depending on the optical emitter power and the transmission rate, distances up to 100 metres can be covered [SIEMENS1], with no loss of data.

Plastic Fibre Optic Emitter diodes and Photo-diode Detectors

The optical emitters and detectors are specially suited for the insertion of commercially available plastic fibres, due to two important constructional (Fig. D.9) features:

- 1) The package surface incorporates a cylindrical hole, for insertion of the fibre, and
- 2) an optical lens is placed at the bottom of the hole

Previously the fibre had to be stripped before it could be inserted into the transmitter/receiver, but with the new package design, the fibre has the same outer diameter as the aperture in the

transmitter/receiver. Thus no special tools or accessories are required for the use of the plastic optic fibre. The insertion of the lense in the casing increases the efficiency of the emitter and detector.

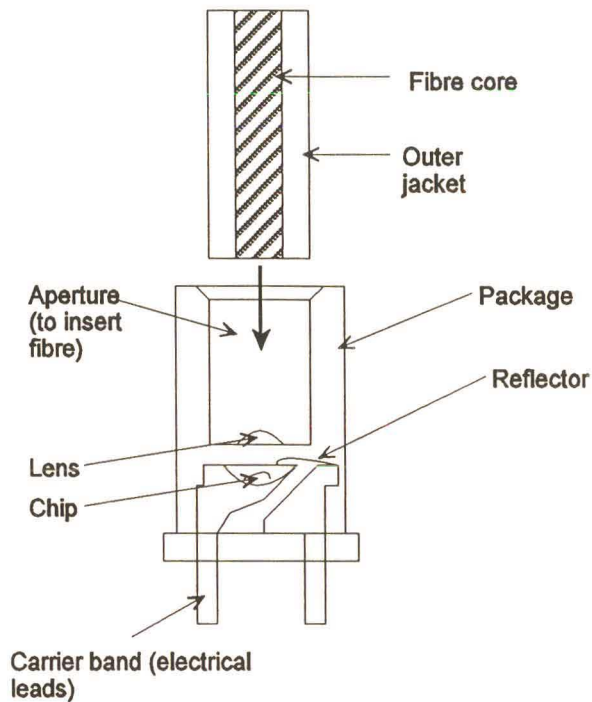


Fig. D.9 Cross section of an emitter diode

The Siemens SFH250V receiver and SFH451V emitter are used in the optical interface for this thesis.

The optic fibre transmitters are mounted on the output stage of the PWM Hanning card, as shown in the layout in Fig. D.8. The physical layout of the optic receiver board is provided in Fig. D.10 below.

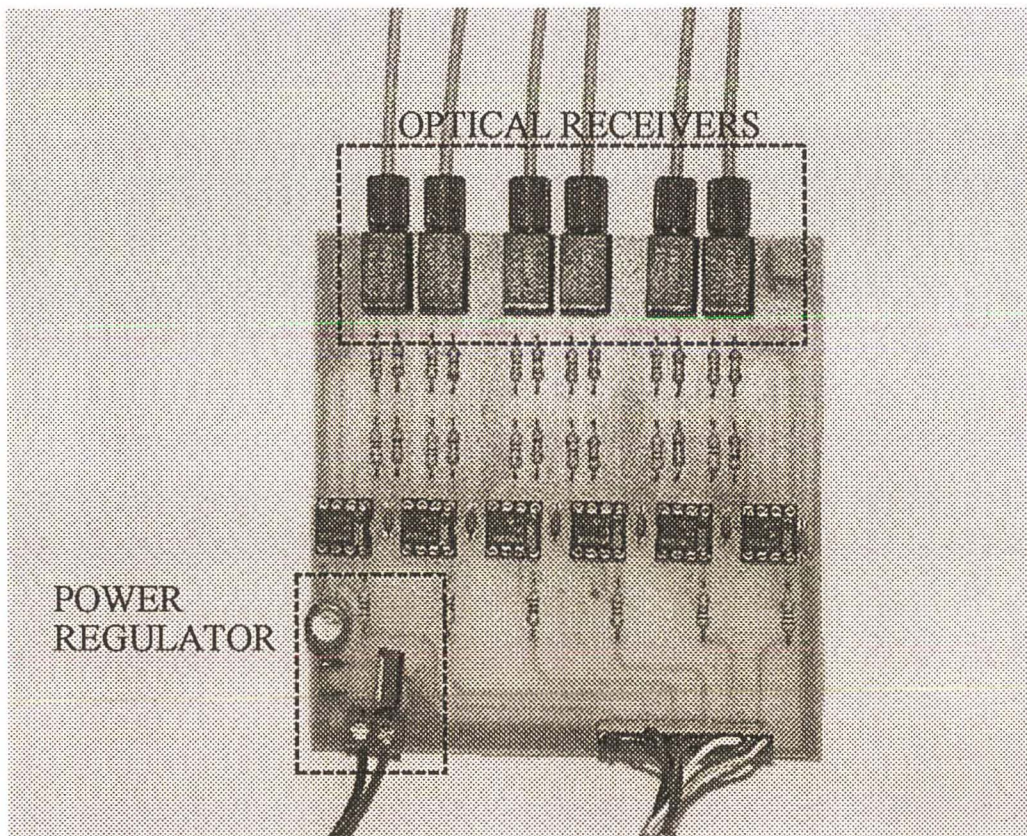


Fig. D.10 Optic receiver board

D.4.4 Signal Sensing and Conditioning Hardware

The system variables to be sensed for the boost rectifier control purposes are the boost rectifier line currents, i_{sa} , i_{sb} , and i_{sc} ; the three-phase AC boost rectifier supply phase voltages, u_{sa} , u_{sb} , and u_{sc} , and the DC link voltage, u_{dc} . The signals are sensed using LEM current and voltage transducers, and then scaled to ± 10 V before being applied to the ADC64 onboard A/D analogue inputs.

Line currents

The boost rectifier line currents are sensed using LEM LA 55-P current transducers, each LEM module provides an output current proportional to the current flowing through the conductor, passing through a hole in the middle of the device. The primary (high power) and secondary (electronic) circuits are galvanically isolated, thus isolating the control circuitry from the power circuit. The secondary current is transferred to the conditioning board (the signal is transferred as a current, because in this state it is less prone to noise), where it is passed through a burden resistor which converts it to an equivalent voltage signal. The LEM modules are rated to provide output currents of 50 mA for primary RMS currents of 50 A. The LEM module is supplied with ± 15 V, as shown in Fig. D.11.

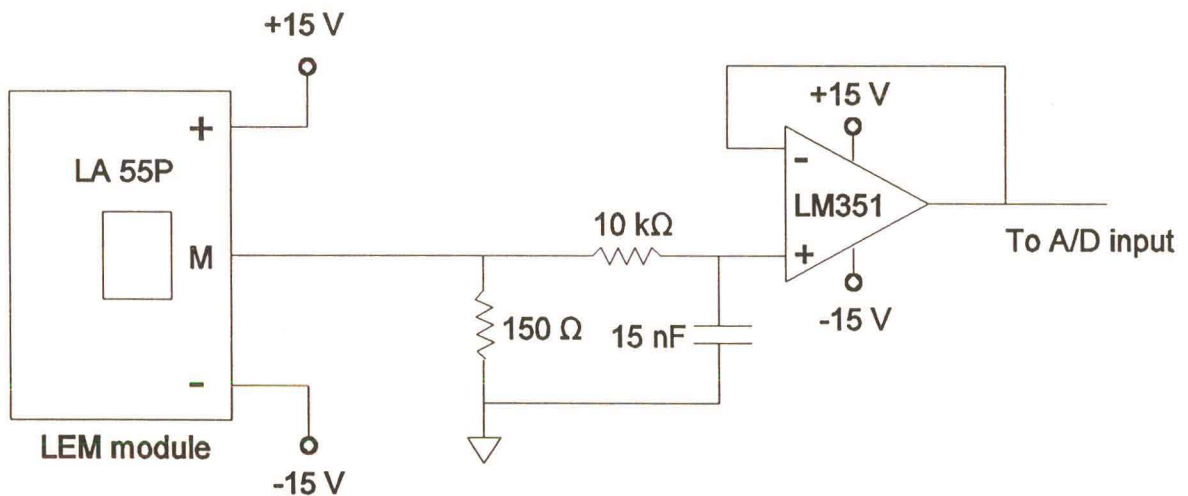


Fig. D.11 Current sensing circuit diagram

The voltage generated across the burden resistor is passed through a unity gain buffer, and then read by the A/D analogue input onboard the ADC64 card.

Phase voltages and DC Link Voltage

The system phase voltages and DC link voltage are sensed using a Hall Effect Device, better known as LEM voltage transducers, which provides galvanic isolation between the power circuit and the control electronics side of the system. Fig. D.12 shows the voltage sensing circuit used for a single phase; all the other voltage sensor circuits are configured in the same manner as this.

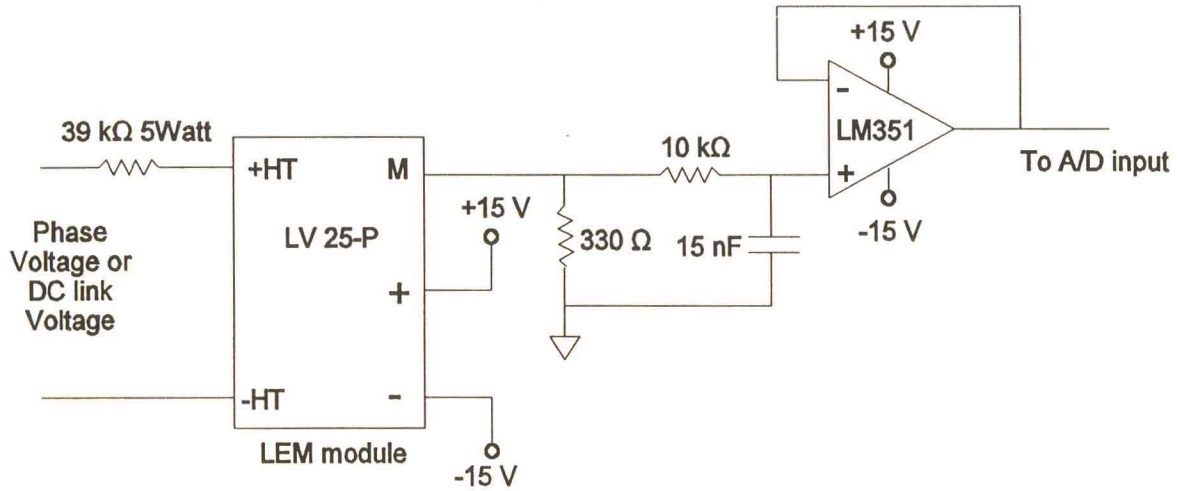


Fig. D.12 Voltage sensing circuit

The maximum primary current is 10 mA, thus a primary resistor is used to ensure that the nominal voltage to be measured corresponds to a primary current of 10 mA. The voltage LEM transducer produces a current on the secondary which is proportional to the primary measured voltage. This current is transferred to the conditioning board where it is passed through a burden resistor, which converts it to an equivalent voltage signal, and then buffered before being read by the A/D analogue input. Fig. D.13 displays the physical layout of the current and voltage sensing board, and the relevant conditioning circuitry.

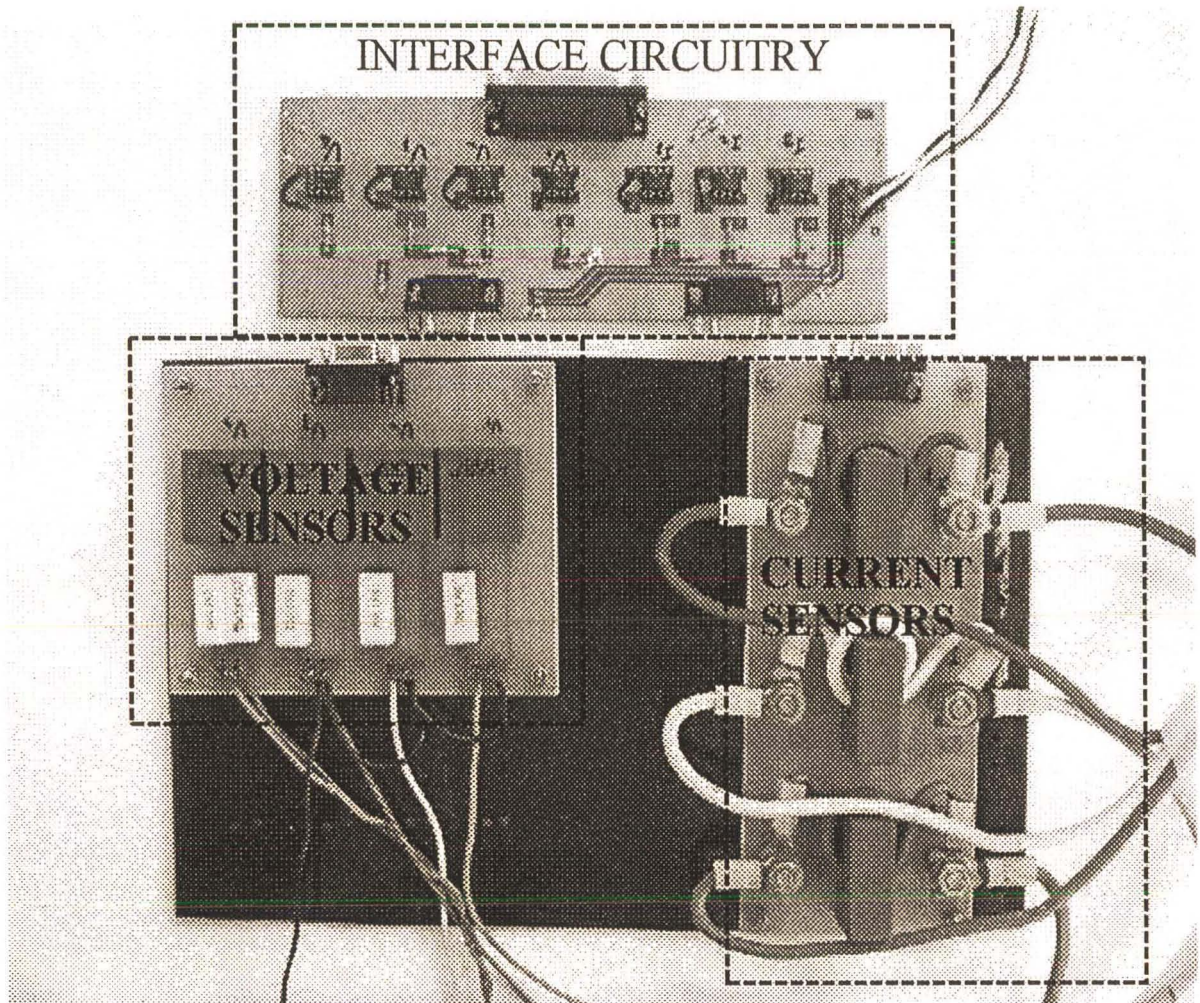


Fig. D.13 Current and voltage sensing and conditioning boards