Evolutionary Algorithms for Practical Sensor Fault Tolerant Control

Yoshikazu Hirayama

PH.D.

THE UNIVERSITY OF YORK
DEPARTMENT OF ELECTRONICS

September, 2007

Evolutionary Algorithms for Practical Sensor Fault Tolerant Control

Submitted in accordance to the requirements of the University of York for the degree of Doctor of Philosophy.

Yoshikazu Hirayama

September, 2007

THE UNIVERSITY of York
Department of Electronics

Abstract

The Shaky Hand is a multi-input, multi-output laboratory demonstrator which is modelled on a village fête game. In the original, the aim is to guide, by hand, a wire loop along a wire which has been bent to form a meandering track, without touching the loop to the wire. In the original game, touching the hand-held loop against the wire track sets off a loud warning bell and the player loses.

The thesis presents the research work associated with the quest for practical solutions to a generic problem: the correct operation of a fallible system. The work covers three distinct areas: modelling of the demonstrator, design and construction of a physical system, and evolution of algorithms for control of the demonstrator in practice in the presence of sensor faults, using Cartesian Genetic Programming (CGP). The third area forms the core of the thesis. The key challenges in creating the virtual environment to train for generic sensor fault tolerant algorithms are considered and addressed. The evolved algorithms are analysed and then verified using the demonstrator in practice. The practical results showed that sensor fault tolerant control was successfully achieved.

Acknowledgements

There are numerous people to whom I would like to express my grateful thanks. Without their support, the work would not have been possible. Firstly, and mostly, to Tim Clarke, my academic supervisor, for his great advice, guidance, patience and encouragement. He was the source of my research inspiration. To Pete Durkin for his generous technical support and expertise in the student workshop. To Julian Miller, for his expertise and knowledge on cartesian genetic programming. To Martin Robinson for his inspiring ideas. To the technicians at the Department of Electronics, especially Mark Hough for pcb fabrications, and Pete Turner for many useful technical advices.

I have enjoyed working with all the members of the Intelligent Control Systems Group and had pleasure of receiving advices from them; Min Liao who I shared some work on the Shaky Hand, and with whom I have had many useful discussions. Peter Mendham whose wide knowledge and enthusiasm was very inspiring. Andy Pomfort, Lee Netherton, Marc Pearson, Charles Robinson, Andy Sheriff, Jon Aitkin, Yang Liu, Lejun Chen, I enjoyed working with them all.

I would like to thank my family whose invaluable support gave me encouragement throughout this work, and many of my friends who I have spent my time with over the last years for being there for me and shared fun with. Finally, but not least, to the University of York for funding me through a studentship.

Contents

Abstract 3

Acknowledgements 4

1 Introduction 15

1.1 Introduction 15

1.1.1 Sensor Fault tolerance 15

1.1.2 Brief Introduction to Evolutionary Algorithms 17

1.2 Background 19

1.3 Thesis Structure 21

1.4 Contributions 21

1.5 Chapter Bibliography 22

2 Development of the Shaky Hand Hardware Models 25

2.1 Introducing the Shaky Hand 26

2.1.1 The Shaky Hand Game 26

2.1.2 Creating a Game Scenario 27

2.1.3 A Centralised Control Solution 28

2.2 Modelling the Shaky Hand Hardware 30

2.2.1 Hardware Plant Logic 32

2.3 DC Motor Model 32

2.3.1 DC Motor Electrical Part Analysis 33

2.3.2 DC Motor Mechanical Part Analysis 34

2.3.3 DC Motor State Space Equations 38

2.4 Detection Coil Induced Voltage Model 40

2.5 Wire Model 43

2.5.1 Wire Track Used in the Simulink Simulation 47

2.6 Potentiometer Model 49

2.7 Encoder Model 50

2.8 Plant Model Summary 50

2.9 Chapter Bibliography 51

3 The Control System Model Implementation 52
3.1 Introduction . . . 53
3.2 Two Reference Frames and Two Plate Velocities . . . 53
3.3 Control Loops . . . 54
3.3.1 Rotational Offset Loop . . . 55
3.3.2 Lateral Offset Loops . . . 56
3.4 Control System Model . . . 57
3.4.1 Offset Error Sensing Solutions . . . 58
3.4.2 Reverse Potentiometer Model . . . 59
3.4.3 Velocity Error Model . . . 59
3.4.4 Controller Models . . . 60
3.4.5 Transform Model . . . 62
3.5 Simulink Simulation . . . 63
3.6 Introducing I/O Programming . . . 69
3.6.1 I/O software . . . 69
3.6.2 Speed Measurement Program . . . 71
3.6.3 Analogue Input Program . . . 72
3.6.4 Control System Program . . . 72
3.6.5 Digital Output Program . . . 73
3.6.6 Analogue Output Program . . . 74
3.7 Chapter Bibliography . . . 74

4 Design and Constructions of the Physical System 75
4.1 Introduction . . . 76
4.2 Physical Constructions . . . 76
4.2.1 General View . . . 76
4.2.2 Design Concern . . . 79
4.3 Electrical Circuits . . . 81
4.3.1 Coil Sensors and the Pick-Up Assembly . . . 81
4.3.2 Alternative Sensors Considered . . . 89
4.3.3 Optical Encoders . . . 92
4.3.4 Potentiometer . . . 92
4.3.5 Pulse Width Modulation and Comparator . . . 93
4.3.6 H-Bridge . . . 95
4.3.7 Relay Circuit and Micro-switches . . . 96
4.4 I/O Board . . . 97
4.5 Trial Runs . . . 97
4.6 Chapter Bibliography . . . 103

5 Approaching Shaky Hand Problems using Cartesian Genetic Programming 104

5.1 Introduction 106
5.1.1 Shaky Hand Problem 106
5.1.2 Plate Control 106
5.1.3 Brief Introduction to Cartesian Genetic Programming 108
5.2 Cartesian Genetic Programming 110
5.2.1 CGP Structure 110
5.2.2 CGP Evolutionary Process 112
5.3 Cartesian Genetic Programming For The Shaky Hand 113
5.4 Pseudocode for the Shaky Hand CGP 113
5.4.1 Embodiment 114
5.4.2 CGP Structure and Parameters 121
5.5 Shaky Hand Sensor Algorithms 132
5.5.1 Verification Method 133
5.5.2 Evolved Solutions 135
5.6 Evolved Solution Analysis 139
5.6.1 Cost Analysis 139
5.6.2 Evolutionary Convergence Analysis 145
5.6.3 Improvement Over the Original Solutions 155
5.6.4 Further Evaluation of the Evolved Solutions 177
5.7 Adapting The Solutions Into The Shaky Hand 180
5.7.1 Results of Simulink Simulation 180
5.7.2 Results Using Shaky Hand Hardware 186
5.8 Why Do the Evolved Solutions for the One Sensor Failure Cases Work So Well? 187
5.8.1 Analysis of V_α Case 187
5.8.2 Analysis of V_ϕ Case 188
5.9 Chapter Bibliography 191

6 Conclusion and Recommendations for Further Work 193
6.1 Introduction 193
6.2 Recommendation for Further Work 193
6.2.1 Monitoring System 194
6.2.2 Multi-Agent Control for the Shaky Hand 197
6.3 Conclusion 198
6.4 Chapter Bibliography 200

A Layouts for Physical Construction Works 201
A.1 Shaky Hand Metal Works 202
A.2 PCB Design Layouts 205
A.3 Appendix Bibliography 207

B Circuit Diagrams 208
B.1 Electrical Circuits Diagrams 209
B.1.1 Original Sensor System 209
B.1.2 Revised Sensor System 211
B.2 Pictures of the PCB Designs 214
B.3 I/O Pin Assignment 216
B.4 Appendix Bibliography 216

C Nomenclature/Glossary 217

List of Figures

2.1 Outline of the Shaky Hand game 26
2.2 Mechanising the Shaky Hand game 27
2.3 The Shaky Hand game pickup plate 27
2.4 Layout of indexed inductor pickups 28
2.5 Overview of the Shaky Hand model structure 30
2.6 Relationship between the plate and the wire track relative to the base frame 32
2.7 DC motor electrical part modelling 33
2.8 Block diagram of the motor attached to the plate and the pick-up assembly 35
2.9 Block diagram of the X- and Y-motors attached to the carriages 37
2.10 Geometry of the plate/wire interaction 41
2.11 Wire model mechanism 44
2.12 A wire track function example 45
2.13 Coordinates and offset errors 46
2.14 Wire track used in the simulation 48
2.15 Summary of the Hardware Block models 50

3.1 Overview of the Shaky Hand model structure 53
3.2 Two reference frames 54
3.3 A control loop for rotational motion of the plate 55
3.4 Control loops for linear motion of the plate 56
3.5 The control system block of the Shaky Hand model 57
3.6 Centralised solutions: plate behaviour 58
3.7 A PID controller in a loop 60
3.8 Simulink simulation results 64
3.9 Position *1 of Figure 3.8 65
3.10 Position *2 of Figure 3.8 65
3.11 Position *3 of Figure 3.8 66
3.12 The angle of offset, ϕ, in the simulation 66
3.13 The lateral offset, α, in the simulation 67
3.14 Speed of the plate in the simulation 67
3.15 Block diagram of the control process through I/O Boards 69

3.16 Program flow chart 70
3.17 A principle of speed measurement 72

4.1 Top view of the Shaky Hand hardware 77
4.2 Side view of the Shaky Hand hardware 77
4.3 Shaky Hand design concern 79
4.4 Sensor plate 81
4.5 Coil sensor amplification circuit 82
4.6 The wire distribution issue 84
4.7 Lock-in amplifier 85
4.8 Block diagram of the new coil sensor system 86
4.9 Low pass filter for the lock-in amplification [12] 87
4.10 Revised sensor plate 88
4.11 Side view of the optical sensor implementation 90
4.12 Grey scale track 90
4.13 Optical sensor experiment preparation 91
4.14 Optical sensor with transimpedance amplifier [62] 91
4.15 Pull-up resistors on the encoder outputs [55] 92
4.16 A three-turn potentiometer coupled with a DC-motor 93
4.17 Duty ratio of the switch 94
4.18 Functional circuit diagram of LM3524DN [59] 94
4.19 Comparator 95
4.20 Compensated signal 95
4.21 A H-bridge circuit:3952SB IC [45] 96
4.22 Microswitches and a relay circuit 97
4.23 Shaky Hand hardware general view 99
4.24 Shaky Hand hardware top view 100
4.25 Shaky Hand plate 101
4.26 Shaky Hand hardware side view 102

5.1 The Shaky Hand controller process 107
5.2 Four sensors on the Shaky Hand plate and the offset terms 107
5.3 A single CGP functional block representation 110
5.4 An example CGP structure 1 111
5.5 An example CGP structure 2 111
5.6 Plate model to create CGP input sets 115
5.7 y_1 and y_2 data variations 118
5.8 Comparing the ideal and the evolved solutions 119
5.9 y_1 and y_2 data variations with two sliding windows each 120
5.10 Convergence 124

5.11 Crossover 125
5.12 Effect of varying the gene crossover rate on the mean number of generations for various population size 128
5.13 Effect of varying the gene crossover rate on the mean computation time for various population size 128
5.14 Definition of the sensor pairs 132
5.15 Expected α and ϕ obtained from the validation test set 134
5.16 Expected V_α and V_ϕ obtained from the validation test set 134
5.17 Sensor failure patterns 140
5.18 Impossible estimation of the offset errors 141
5.19 Comparison of diagonal and adjacent sensor pair failure solutions for V_α using test sets . . 142
5.20 Comparison of diagonal and adjacent sensor pair failure solutions for V_ϕ using test sets . . 143
5.21 Comparison of opposite sensor pair failure solutions and three sensor failure solutions for V_α using test sets 144
5.22 Comparison of opposite sensor pair failure solutions and three sensor failure solutions for V_ϕ using test sets 144
5.23 V_α cost variation for the adjacent pair failure solution 147
5.24 V_α cost variation for the diagonal pair failure solution 148
5.25 V_α cost variation for the opposite pair failure solution 149
5.26 V_ϕ cost variation for the adjacent pair failure solution 150
5.27 V_ϕ cost variation for the diagonal pair failure solution 151
5.28 V_ϕ cost variation for the opposite pair failure solution 152
5.29 V_α cost variation for the first 5000 generations 153
5.30 V_ϕ cost variation for the first 5000 generations 154
5.31 V_α from the evolved solutions 158
5.32 V_ϕ from the evolved solutions 159
5.33 V_α from the original algorithms 160
5.34 V_ϕ from the original algorithms 161
5.35 Sensor B failure: V_α comparisons between the original and the evolved solutions 163
5.36 Sensor A failure: V_ϕ comparisons between the original and the evolved solutions 164
5.37 Adjacent sensor pair AB failure: V_α comparisons between the original and the evolved solutions 166
5.38 Adjacent sensor pair AB failure: V_ϕ comparisons between the original and the evolved solutions 167
5.39 Diagonal sensor pair BC failure: V_α comparisons between the original and the evolved solutions 169
5.40 Diagonal sensor pair BC failure: V_ϕ comparisons between the original and the evolved solutions 170
5.41 Opposite sensor pair AC failure: V_α comparisons between the original and the evolved solutions 172
5.42 Opposite sensor pair AC failure: V_ϕ comparisons between the original and the evolved solutions 173
5.43 Three sensors ABC failure: V_α comparisons between the original and the evolved solutions 175
5.44 Three sensors ABC failure: V_ϕ comparisons between the original and the evolved solutions . 176
5.45 Different wire coordinates for evaluating the evolved solutions 177
5.46 Simulink simulation: using the original algorithm 182

5.47 Simulink simulation: closer look of Figure 5.46 182
5.48 Simulink simulation: using the sensor C failure case solution 183
5.49 Simulink simulation: using the sensor B&C failure case solution 183
5.50 Simulink simulation: using the sensor A&B failure case solution 184
5.51 Simulink simulation: using the sensor A&C failure case solution 184
5.52 Simulink simulation: using the sensor A&B&D failure case solution 185
5.53 Plate configuration for the V_ϕ solution analysis 188
5.54 V_ϕ against ϕ with different α values using the original solution 190
5.55 V_ϕ against ϕ with different α values using the evolved solution 190

A.1 The base frame 202
A.2 Y-carriage 202
A.3 X-carriage and bearing housing (Y-motor side) 203
A.4 X-carriage and bearing housing (Y-shaft-encoder side) 203
A.5 Horizontal shaft support plate 203
A.6 Horizontal leadscrew support plate 204
A.7 Top view of the supporting layer 204
A.8 Dimensions of the support poles 204
A.9 Dimensions for wire track supporting poles 205
A.10 Layout for PCB boards: relay circuit and power supply lines 205
A.11 Layout for PCB boards: ribbon cable jacks and other connectors 206

B.1 Coil sensor signal amplifier with trimmers [41] 209
B.2 RMS-DC converting circuit [41] 210
B.3 Coil sensors and differential amplifiers with trimmers 211
B.4 Demodulator 212
B.5 Low pass filters 212
B.6 Revised sensor plate PCB layout 213
B.7 PWM, comparator and H-bridge circuit diagrams [41] 213
B.8 Original design of the top plate 214
B.9 Improved design of the top plate 214
B.10 RMS-DC converting circuit on PCB board 215
B.11 PWM, comparator and H-bridge circuits on PCB board 215

List of Tables

2.1 Date sheet for the motors 40

3.1 Expected general behaviour from the plate when offset errors are sensed 59
3.2 PID controller gain values 62
3.3 I/O program functions 71
3.4 Digital I/O port assignment 73

4.1 Specification for each part of the Shaky Hand hardware 78

5.1 Population size experiment 123
5.2 Crossover experiment with population size = 5 127
5.3 Crossover experiment with population size = 10 127
5.4 Crossover experiment with population size = 20 127
5.5 Operators used in the Shaky Hand CGP 130
5.6 Summary of the cost and the standard deviations of the evolved solutions 139
5.7 Comparison of cost/STD between the original V_α algorithm and the evolved V_α algorithms 155
5.8 Comparison of cost/STD between the original V_ϕ algorithm and the evolved V_ϕ algorithms 155
5.9 Evolved solution outputs for different wire positions 177
5.10 Simulink simulation results 180
5.11 Results of the evolved solutions in practice 186

6.1 Second set of evolved solutions comparison 195

B.1 Power supply assignment 211
B.2 Pin assignment for I/O board 1 216
B.3 Pin assignment for I/O board 2 216

List of Accompanying Materials

1. CD-R (Short movies showing Shaky Hand operations)

Folder 1 :	Movie 1	- Using the ideal, no-failure solution.
Folder 2 :	Movie 2	- Using one sensor failure solutions, sensor A failure
	Movie 3	- Using one sensor failure solutions, sensor B failure
	Movie 4	- Using one sensor failure solutions, sensor C failure
	Movie 5	- Using one sensor failure solutions, sensor D failure
Folder 3 :	Movie 6	- Using diagonal sensor pair failure solutions, sensor AD failure
	Movie 7	- Using diagonal sensor pair failure solutions, sensor BC failure
Folder 4 :	Movie 8	- Using adjacent sensor pair failure solutions, sensor AB failure
	Movie 9	- Using adjacent sensor pair failure solutions, sensor CD failure
Folder 5 :	Movie 10	- Using opposite sensor pair failure solutions, sensor AC failure
	Movie 11	- Using opposite sensor pair failure solutions, sensor BD failure
Folder 6 :	Movie 12	- Using three sensor failure solutions, sensor ABC failure
	Movie 13	- Using three sensor failure solutions, sensor ABD failure
	Movie 14	- Using three sensor failure solutions, sensor ACD failure
	Movie 15	- Using three sensor failure solutions, sensor BCD failure

Chapter 1

Introduction

Contents

1.1 Introduction 15
1.1.1 Sensor Fault tolerance 15
1.1.2 Brief Introduction to Evolutionary Algorithms 17
1.2 Background 19
1.3 Thesis Structure 21
1.4 Contributions 21
1.5 Chapter Bibliography 22

1.1 Introduction

The thesis presents the research work associated with the quest for practical solutions to a generic problem: the correct operation of a fallible system. The work has produced a final fall-back system which will provide safe, if degraded, performance of a system when all other fault tolerance mechanisms, based upon multiple redundancy of *sensors*, cease to be available.

1.1.1 Sensor Fault tolerance

A control system capable of tolerating a failure is referred as a fault tolerant control system (FTCS). An overview paper by Patton [29] covers a wide range of literatures, studies and applications on the area of FTCS. However, as Wu [35] pointed out, the overview paper does not discuss much about sensor fault tolerant control. Sensors on a physical system provide the most important information: the status of the system and/or the environment in which the system is situated. However, sensors are subject to failure. A faulty sensor signal may

lead to wrong control system behaviour and bring on an undesirable situation for the system if there is no means for fault tolerance. Wu [35] states that the reason for less literature on sensor fault tolerant control is because the critical role of measured variables in a controlled system makes it necessary to keep sensor reliability high, which is often achieved through the use of direct (hardware) redundancy. This means that multiple sensors are utilised for majority voting for the selection of healthy sensors. Also, a faulty sensor can be replaced, physically, by spare sensors, if they exist. In combination with direct redundancy, or on its own, analytical redundancy can also be used as part of the FTCS design. Analytical redundancy is provided by a model of the system variable of concern, and produces an estimated value in lieu of the faulty sensor. Sensor redundancy may increase reliability but it has some drawbacks. Direct redundancy adds costs for the extra sensors used and it may cause inconvenience in installations if weight, size or maintenance are considered. For example, a physical system may already be built and it may be required to add sensor fault tolerance afterwards. Analytical redundancy is based upon model information of the system variable concerned, so it suffers more from system non-linearity and parameter or model uncertainty [29], compared with direct redundancy.

FTCS involves fault detection scheme such as the fault detection and isolation unit (FDI). It is the same for the sensor FTCS. The emphasis is placed on error detection in the sensors. The FTCS then takes decisions as to whether to continue the operation in degraded mode or to stop the operation. The degree of redundancy and the remaining resources are often considered in the decision-making process. FTCS may either use extra sensors, or reconfigure the control laws. For example, in a control system with a sliding mode observer unit [9][4], faulty sensors are replaced by their estimations.

Here, the author offers an alternative method for tolerating sensor failure in a control system, where multiple sensors are required for measurement, without reconfiguring the control laws, or without having to estimate the correct values from the faulty sensor values. This is achieved by focusing on generating the correct inputs to the controller which are normally calculated based on the working sensor values. A type of evolutionary algorithm called *Cartesian Genetic Programming* (CGP) [25] attempts to evolve algorithms which can generate the appropriate controller input values but using only the remaining, working sensors. The evolved algorithms are tested through systematic analysis, in simulation, and finally in practice using a suitable demonstrator (see Section 2.1). The work assumes a fault detection mechanism exists which simply switches to the novel evolved algorithms in the event of detecting a sensor failure. In this event, degraded, yet acceptable operations are achieved. Using only the remaining

sensors means that spare sensors are not necessary, reducing the cost or inconvenience in the system hardware design. However, they can be used as an adjunct to direct redundancy for higher reliability when all else has failed. The method presented in this thesis is adaptable to a wide range of control systems.

It is worth noting another area of research on sensor fault tolerant system which uses evolvable hardware [13]. (This is a new concept in the development of online adaptive machines [7].) Multiple sensor inputs are connected to a programmable device which employs evolutionary algorithms. The hardware structure can be reconfigured in response to sensor failures, online and autonomously. The system seems to be robust and performs better than the conventional hardware, but is applicable only to systems with slow dynamics. The concept of this area of research is different from the work in the thesis in which the generation of the algorithms by CGP is offline i.e. the evolution takes place at the design stage. Offline applications of evolutionary algorithms have been popular and successful [10] whereas online applications raise difficulties with poor response times that directly affect the overall performance of the system.

1.1.2 Brief Introduction to Evolutionary Algorithms

Evolutionary Algorithms (EAs) are now briefly introduced. However, the emphasis will be on CGP which has been a major research subject in the Intelligent Systems Group at the University of York over several years [12][24][34], and whose application is the main topic of this thesis.

Evolutionary algorithms are a robust search and optimisation methodology and are based on the Darwinian principle of the survival of the fittest. A basic idea is that genetic operators (crossovers/mutations/recombinations) are applied to parental solutions (chromosomes) to produce offspring in a population. Fitter solutions with the environment remain (survive) into subsequent generations. One of main goals of EAs is to simulate the evolutionary process in a computer [6]. Overview papers on EAs for control system engineering can be found at [5] and [10]. Fleming [10] highlights the special properties of EAs which profit control-related topics. He states that EAs cope well with ill-behaved cost landscapes exhibiting such properties as multi-modality, discontinuity, time-variance, randomness, and noise. Each of them can cause severe difficulties for traditional computational search methods. There are four approaches within EAs: *Evolutionary Strategies* (ES) [2], *Evolutionary Programming* (EP) [11], *Genetic Algorithms* (GA) [14] and *Genetic Programming* (GP) [16]. A brief introduction on

them and their applications can be found in a paper by Coello [6]. The basic procedure of an EA is:

1. Initialise the population.
2. Evaluate the individuals in the population.
3. Select the solution(s) according to the fitness.
4. Apply genetic operators and produce offspring.
5. Repeat from procedure 2 until a stopping criterion (e.g. a certain number of generations or a desired response from the offspring) is achieved.

ES were developed as a simple optimization algorithm where a set of random changes are applied to a parent solution. The original form was (1+1)ES, as it consisted of a single parent, which was subject to a random change (mutation) to produce an offspring. The fitter solution of the two is chosen for the next generation. A variety of evolutionary strategies have been proposed over the years. The basic algorithm is similar for EP, but a variety of mutation operators are normally available. The mutation operators are often adaptive such that different adaption rates may be used for each decision variable within an individual [5]. The selection method is probabilistic unlike ES whose selection method is deterministic [6]. EP does not use recombination but, rather, mutation. Thus the behavioural links between parents and offspring (genetical similarity) are emphasized. In GA, the solution representations are encoded, unlike ES and EP. Traditionally, a binary encoding is used [14]. Cost/fitness functions are applied to measure the quality of solutions. GP, first introduced by Cramer [28] in 1985, used GA to evolve tree-structured and string-based computer programs. It is a search technique inspired by biological evolution to evolve computer programs in a population according to a fitness measure defined by the user. Traditionally, in the tree representation, every tree node has an operator function and every terminal node has an operand [16][17]. The evolution process normally involves crossover and mutation. CGP is a different format of GP and was developed by Miller [25][21] for the purpose of evolving digital circuits. It has proven to be highly efficient in learning Boolean functions over standard GP methods. With CGP [25], computer programs in a population are mapped to directed graphs rather than trees and the genotype is represented by a string of integers rather than the high level program statements often favoured by a tree-structure GP. This technique originally only involved mutation. Important benefits of CGP over standard GP methods are that the evolution is bloat free and

has faster evolution through neutral search [12]. Bloat free, because the length of a genotype is fixed and its manipulation does not add extra integers, whereas the code growth or the bloat problem in GP may lead to excessive pressure on system resources and unsatisfactory convergence [3]. Neutrality, refers to the presence of multiple genotypes of the same fitness in a population. This has a potential for changing the future genotype dynamically [26][33]. The genotype in CGP represents connections and functionalities of a rectangular array of primitive functions (see Section 5.2) and a large amount of redundancy is involved. Some of the genotype, including functional block nodes and input nodes, are not used in the graphs and have the potential to add neutrality. The inactive nodes may undergo neutral change and become activated later.

1.2 Background

The author's interest in building an artificial intelligent control system in practice was initially inspired by the research work on Skylon [31][32], a single stage to orbit re-useable launch vehicle, in the Control Systems Laboratory of the University of York [20][27]. Currently in development by Reaction Engines Limited, the vehicle is designed to lower launch costs to around one tenth of the current value and reduce long-term operating costs even further. To reduce both development and operating costs, Skylon is entirely un-manned and un-piloted. From before takeoff until after landing the vehicle should be entirely autonomous, including during all possible abort scenarios. This feature, combined with the extended flight envelope (an extremely wide operating regime) and an unstable airframe (in aerodynamic flight and possibly beyond) presents a challenging control problem. Particularly during ascent and re-entry, all decisions made by the vehicle control systems are safety-critical and hard real-time. There is no potential for standby states to allow further computation in the case of unexpected failure modes. The Control Systems Laboratory has been working on this subject using multi-agent systems [20]. Applying intelligent control to the actual system in practice, however, is not an option. Therefore there has been a need to build a physical platform which is suitable to test aspects of intelligent control in practice. For this purpose, a generic demonstrator, which we have named *Shaky Hand* (see Section 2.1) was chosen.

The Shaky Hand was modelled on a village fête game. In the original game, the aim is to guide, by hand, a wire loop along a meandering wire track from one end to another, without touching the loop to the wire. When the loop touches the wire an electrical circuit is made and an alarm is set off. Using sensors and motors, we produced the automated equivalent

in the form of an x-y plotter. Two rotary DC motors drive the "hand-held loop" by way of a long threaded steel rod in a lead-screw arrangement. The loop can be rotated by a third drive motor to keep the plane of the loop perpendicular to the wire. The Shaky Hand utilises seven sensors in total. Data from the sensors are integrated to give out information about the status of the system (orientation of the loop with respect to the track and the speed at which the loop is traveling along the track). Because of this integration process from data to information, it can be said that the Shaky Hand represents a natural model of a real industrial machine. The Shaky Hand is simple but has a fairly large number of inputs and outputs and also has well established goals. These factors make the Shaky Hand an ideal sensor-agent platform. The Shaky Hand is especially suitable for testing sensor fault tolerant and data fusion techniques due to its multiple sensor environment and also because the the quality of sensor data affects the quality of the output information. One can compensate or enhance the sensor data using these techniques to improve the quality of the output information. As part of a larger research initiative to build a multi-agent control system, the author was tasked to consider the evolution of sensor fault tolerance for the Shaky Hand system using Cartesian Genetic Programming (CGP). Sensors play one of the most important roles in real world machines as a faulty sensor signal may lead to wrong control system behaviour and bring on an undesirable situation for the system if there is no means for fault tolerance as mentioned in Section 1.1.1.

CGP has been a major research subject in Intelligent Systems Group at the University of York as mentioned in Section 1.1.2. CGP has demonstrated its effectiveness in learning Boolean functions over conventional GP [21] and has been applied in variety of applications. The applications include digital circuit designs [23], image filter and its implementation in FPGA [19][30], artificial life [12], bio-inspired developmental models [22], and evolutionary art [1][8]. However, the use of CGP in sensor fault tolerant control application has not been explored before. The CGP based sensor fault tolerant control is novel in the field of the sensor fault tolerant application as well. Since the outcome of CGP can be analysed, the system reliability can be raised, and this can be considered practical. These are the motivations for applying CGP for the sensor fault tolerant application. The work demonstrates that sensor fault control applications can benefit from the use of CGP. The algorithms with sensor fault tolerance are evolved. Parameters and methods used in CGP are explored in order to establish a method to evolve generic solutions.

Most of the initial design and part of the construction of the physical system had already been carried out by Liao [18] when the author started the research. The initial aim of the

research was to revise the design and then complete the Shaky Hand physical system. The author provided the most of the electrical circuit designs, created all the models and wrote all the control software.

1.3 Thesis Structure

The thesis begins by introducing the demonstrator, the Shaky Hand. Development of the Shaky Hand hardware dynamic models is described in Chapter 2. Chapter 3 describes the control system models. All the models developed are implemented using Simulink [15] for the simulation. Practical implementation is also described later. In Chapter 4, design and construction of the physical system, including both mechanical and electrical aspects are detailed. Cartesian Genetic Programming (CGP) [25] is then introduced in Chapter 5 as a method for evolving the algorithms for sensor fault tolerant control of the Shaky Hand. The newly-established work is tested in both simulation and practice. The results are shown and analysed. Finally, the work is concluded in Chapter 6 together with the recommendation for further work.

1.4 Contributions

This section lists the novel contributions provided by this thesis.

- Design and construction of a novel laboratory demonstrator suitable for sensor fault tolerant control/data fusion application.
- Novel exploitation of the potential algorithmic space using CGP a task we consider to be difficult or impossible using a pen and paper approach.
- Establishment of a novel CGP method for producing generic solutions.
- Production of novel fault tolerant solutions.
- Applied in practice: results shows that the Shaky Hand fails gracefully (degradation is kept to minimum).

1.5 Chapter Bibliography

[1] L Ashmore. An investigation into cartesian genetic programming within the field of art. Final year project 2000, Department of Computer Science, University of Birmingham, 2000.

[2] H.G Beyer. *The theory of evolution strategies.* Natural computing series. Springer, 2001.

[3] M Bhattacharya and B Nath. Genetic programming: A review of some concerns. In *Computational Science ICCS 2007, Part 2 Lecture Notes in Computer Science*, volume 2074, pages 1031–40. Springer-Verlag, Berlin, Germany, 2001.

[4] A Chamseddine, H Noura, and M Ouladsine. Sensor fault-tolerant control for active suspension using sliding mode techniques. In *Workshop on networked control systems and fault-tolerant control*, Ajaccio, Corsica, France, October 2005. COSI.

[5] A Chipperfield and P Fleming. An overview of evolutionary algorithms for control system engineering. In J.J Gertler, J.B.Jr Cruz, M Peshkin, R Bitmead, and F.J Kraus, editors, *International federation of automatic control, IFAC, control design 1*, volume C of *proceedings of the 13th world congress*, pages 181–186. Pergamon, Oxford, UK, 1997.

[6] C.A Coello. An introduction to evolutionary algorithms and their applications. In Ramos et al., editor, *Advanced distributed systems, ISSADS, revised selected papers lecture notes in computer science*, volume 3563, pages 425–442. Springer-Verlag, Berlin, Heidelberg, 2005.

[7] D Dhnasekaran, K.B Bagan, and S Ravi. Fault tolerant system design using evolved virtual reconfigurable circuit. *IJCSNS International Journal of Computer Science and Network Security*, 6(5):64–72, May 2006.

[8] S DiPaola. Evolving 'portrait painter programs' using genetic programming (Darwinian evolution) and a portrait of Darwin. website, School of Interactive Arts & Technology, Faculty of Applied Sciences, Simon Fraser University. http://www.dipaola.org/evolve/, August 2007.

[9] C Edwards and C.P Tan. Sensor fault tolerant control using sliding mode observers. *Control Engineering Practice*, 14:897–908, 2006.

[10] P.J Fleming and R.C Purshouse. Evolutionary algorithms in control systems engineering: survey. *Control Engineering Practice*, 10:1223–1241, March 2002.

[11] L.J Fogel. *Intelligence through simulated evolution: forty years of evolutionary programming.* Wiley series on intelligent systems. Wiley, 1999.

[12] S Harding and J.F Miller. Evolution of robot controller using cartesian genetic programming. In *Proceedings of the 6th European Conference on Genetic Programming (EuroGP 2005)*, LNCS 3447, pages 62–72. Springer, 2005.

[13] J.M Hereford. Fault tolerant sensor systems using evolvable hardware. *IEEE Transactions on instrumentation and measurement*, 55(3):846–853, June 2006.

[14] J.H Holland. *Adaptation in natural and artificial systems: an inductory analysis with applications to biology, control, and artificial intelligence.* complex adaptive systems. 1st MIT Press, 1992.

[15] THE MATHWORKS INC. *Simulink, Dynamic System Simulation for Matlab.* THE MATHWORKS INC., Natick, MA, 3 edition, 1990.

[16] J.R Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* The MIT Press, Cambridge, Massachusetts, 1992.

[17] J.R Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs.* The MIT Press, Cambridge, Massachusetts, 1994.

[18] M Liao. Shaky hand. Msc report, Control Systems Laboratory, Intelligent Systems Group, The University of York, 2004.

[19] T Martinek and L Sekanina. An evolvable image filter: experimental evaluation of a complete hardware implementation in fpga. In J.M Moreno, J Madrenas, and J Cosp, editors, *Evolvable Systems: From Biology to Hardware. Proceedings Lecture Notes in Computer Science*, volume 3637 of *6th International Conference, ICES 2005*, pages 76–85, Sitges, Spain, September 2005. Springer-Verlag, Berlin, Germany.

[20] P Mendham. *Multi-agent control for the Skylon spaceplane.* Phd thesis, Department of Electronics, University of York, Department of Electronics, The University of York, York, YO10 5DD, 2006.

[21] J.F Miller. An empirical study of the efficiency of learning boolean functions using cartesian genetic programming approach. volume 2 of *Proc. of GECCO, page 1135-1142.* Morgan Kaufmann, 1999.

[22] J.F Miller. Evolving developmental programs for adaptation, morphogenesis, and self-repair. *Seventh European Conference on Artificial Life*, 2801:256–265, September 2003.

[23] J.F. Miller, D Job, and V.K. Vassilev. Principles in the evolutionary design of digital circuits – part i. *Journal of Genetic Programming and Evolvable Machines*, 1(2):259–288, 2000.

[24] J.F Miller and S.L Smith. Redundancy and computation efficiency in cartesian genetic programming. *IEEE Transactions on Evolutionary Computation*, 10:167–174, 2006.

[25] J.F Miller and P. Thompson. Cartesian genetic programming. volume 1802 of *Proceedings of the 3rd European Conference on Genetic Programming*, pages 121–132. Springer-Verlag, 2000.

[26] J.F Miller and T Yu. Neutrality and evolvability of a boolean function landscape. volume Vol. 2038, pp. 204-217 of *EuroGP2001.* Springer-Verlag, 2001.

[27] L Netherton. *Dynamic Modelling of the SABRE Engine.* PhD thesis, The University of York, Department of Electronics, Department of Electronics, The University of York, York, YO10 5DD, 2007.

[28] N.L.Cramer. A representation for the adaptive generation of simple sequential programs. In J.J.Grefenstette, editor, *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, number 183-187, Carnegie-Mellon University, Pittsburgh, PA, USA, 24-26 Jul 1985.

[29] R Patton. Fault tolerant control systems: the 1997 situation. In *Proceedings of the IFAC Symposium*, Safeprocess, pages 1035 – 1055, 1997.

[30] K Slany and L Sekanina. Fitness landscape analysis and image filter evolution using functional-level cgp. In M Ebner, M O'Neill, A Ekart, L Vanneschi, and A.A Esparcia, editors, *Proceedings Lecture Notes in Computer Science*, volume 4445 of *Genetic Programming. 10th European Conference, EuroGP 2007.*, pages 311–320, Valencia, Spain, April 2007. Springer-Verlag.

[31] R Varvill and A Bond. The skylon spaceplane. In *Proceedings of the 46th Congress of the International Astronautical Federation (IAF 95-V3.07), Oslo, Norway*, 1995.

[32] R Varvill and A Bond. The skylon spaceplane. *The British Interplanetary Society*, 57:22–32, 2004.

[33] V.K. Vassilev and J.F. Miller. The advantages of landscape neutrality in digital circuit evolution. *3rd International Conference on Evolvable Systems*, 1801:252–263, 2000.

[34] J.A Walker and J.F Miller. The automatic acquisition, evolution and reuse of modules in cartesian genetic programming. *IEEE Transactions on Evolutionary Computation*, To be published in 2008.

[35] N.E Wu. Sensor fault masking of a ship propulsion system. *Control Engineering Practice*, 14(11):1337–45, November 2006.

Chapter 2

Development of the Shaky Hand Hardware Models

Contents

2.1 Introducing the Shaky Hand 26
2.1.1 The Shaky Hand Game 26
2.1.2 Creating a Game Scenario 27
2.1.3 A Centralised Control Solution 28
2.2 Modelling the Shaky Hand Hardware 30
2.2.1 Hardware Plant Logic 32
2.3 DC Motor Model 32
2.3.1 DC Motor Electrical Part Analysis 33
2.3.2 DC Motor Mechanical Part Analysis 34
Rotary Mechanical Drive Model 35
Linear Mechanical Drive Model 36
2.3.3 DC Motor State Space Equations 38
2.4 Detection Coil Induced Voltage Model 40
2.5 Wire Model 43
2.5.1 Wire Track Used in the Simulink Simulation 47
2.6 Potentiometer Model 49
2.7 Encoder Model 50
2.8 Plant Model Summary 50
2.9 Chapter Bibliography 51

UNIVERSITY OF YORK LIBRARY

2.1 Introducing the Shaky Hand

In this chapter, the novel laboratory demonstrator which we named *Shaky Hand* is introduced followed by the development of the Shaky Hand Hardware system model.

2.1.1 The Shaky Hand Game

The Shaky Hand game was devised as a generic technology demonstrator and was modelled on a village fête game. In the original game, as shown in Figure 2.1, the aim is to guide, by hand, a wire loop along a meandering wire track from one end to another, without touching the loop to the wire. When the loop touches the wire an electrical circuit is made and an alarm is set off.

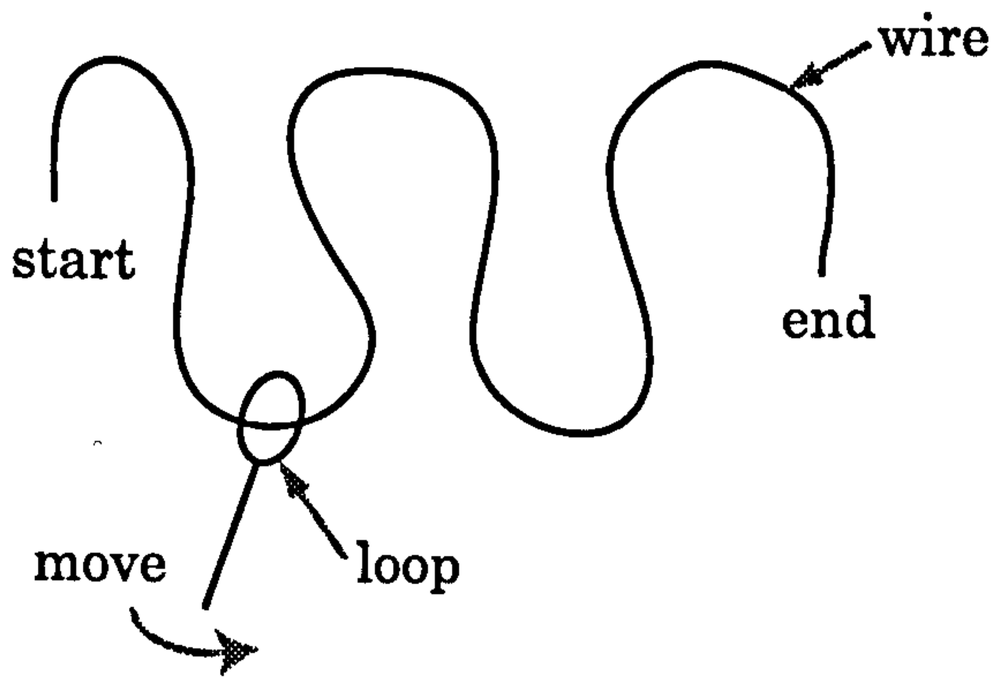


Figure 2.1: Outline of the Shaky Hand game

The Shaky Hand follows this model. However, the loop is guided by a flat bed plotter arrangement with x and y translational drive motors as shown in Figure 2.2. These are rotary DC motors, driving their load by way of a long threaded steel rod in a lead-screw arrangement. The loop can be rotated by a third drive motor to keep the plane of the loop perpendicular to the wire. Again, this uses a DC motor, this time driving the loop rotation through a reduction gearbox. Four inductor coils are provided on a plate just below the wire to allow the proximity and orientation of the wire to be sensed as illustrated in Figure 2.3.

To facilitate this, the wire carries an alternating current of appropriate magnitude and frequency. A basic relationship between the emf coupled into the coil sensor and the proximity is assumed, where the magnitude of the induced emf is inversely proportional to the proximity. The output voltages from the four coil sensors are then amplified and converted into DC

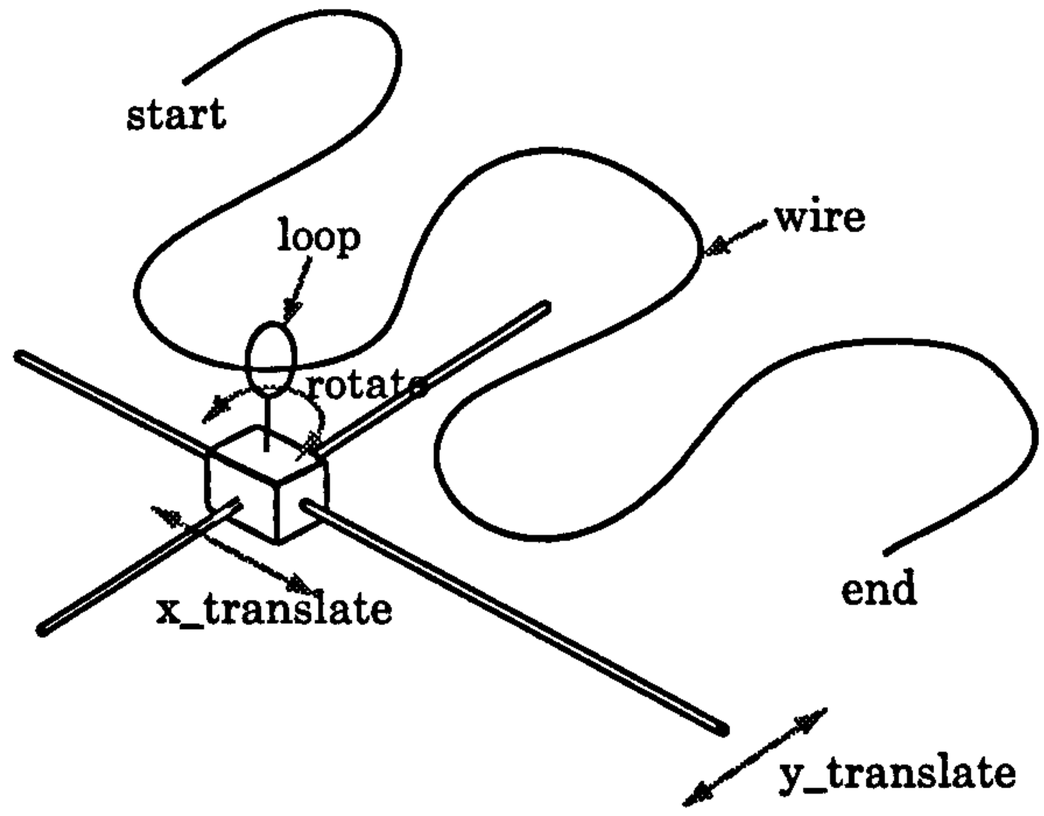


Figure 2.2: Mechanising the Shaky Hand game

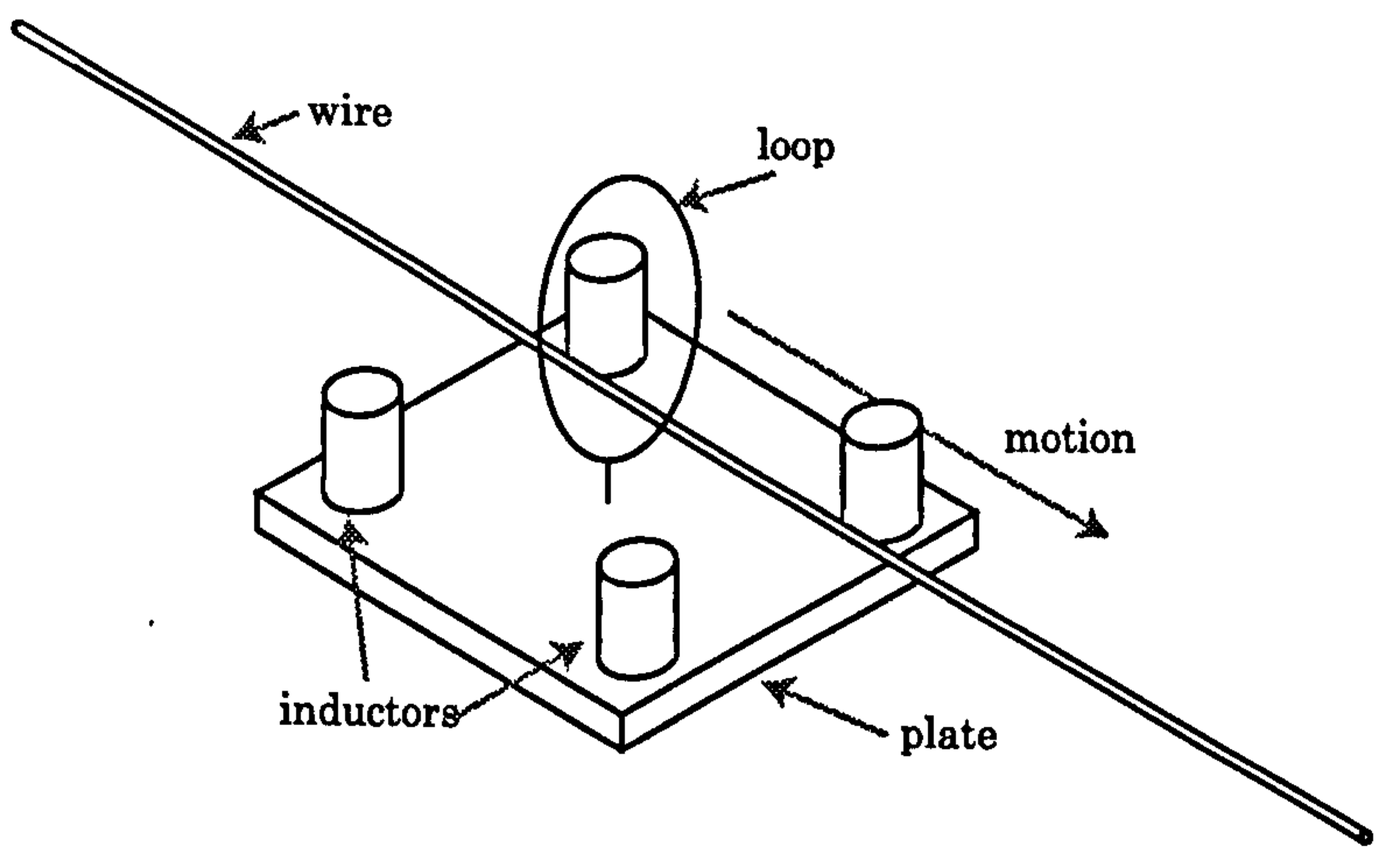


Figure 2.3: The Shaky Hand game pickup plate

signals and presented to a PC based analogue data acquisition card.

2.1.2 Creating a Game Scenario

To make an interesting scenario for development we must define a set of game 'rules'. The main aim does not change from above, i.e. the loop must be guided from one end of the wire to the other. The central constraint also still stands, i.e. that the loop should never touch the wire. The loop size is defined by the sensor positions and so the wire should never touch the sensors. To make this explicit we define this as a failure that constitutes the end of a run and a catastrophic failure. However, the problem does not yet contain any time constraints. The system may stop the movement of the loop as it decides what action to take next. So we introduce a time constraint into the system by defining another catastrophic

failure condition: the speed of the loop along the wire direction shall be kept at a defined level v_S which is non-zero.

2.1.3 A Centralised Control Solution

We will first examine the necessary processes involved in solving the game scenario proposed above using a centralised control system. In order to create a model for the system which allows control independently of wire shape we must split the model into two sections, each working in a different frame of reference. The *fixed* frame has x and y axes aligned with the lead screws. The *plate* frame of reference has axes aligned with the plate with the x axis being parallel to the direction of motion. The angle of the plate in the fixed frame, θ, provides the mapping between the two frames.

The centralised system has four inductor inputs; we will index these as inputs A through D (see Figure 2.4).

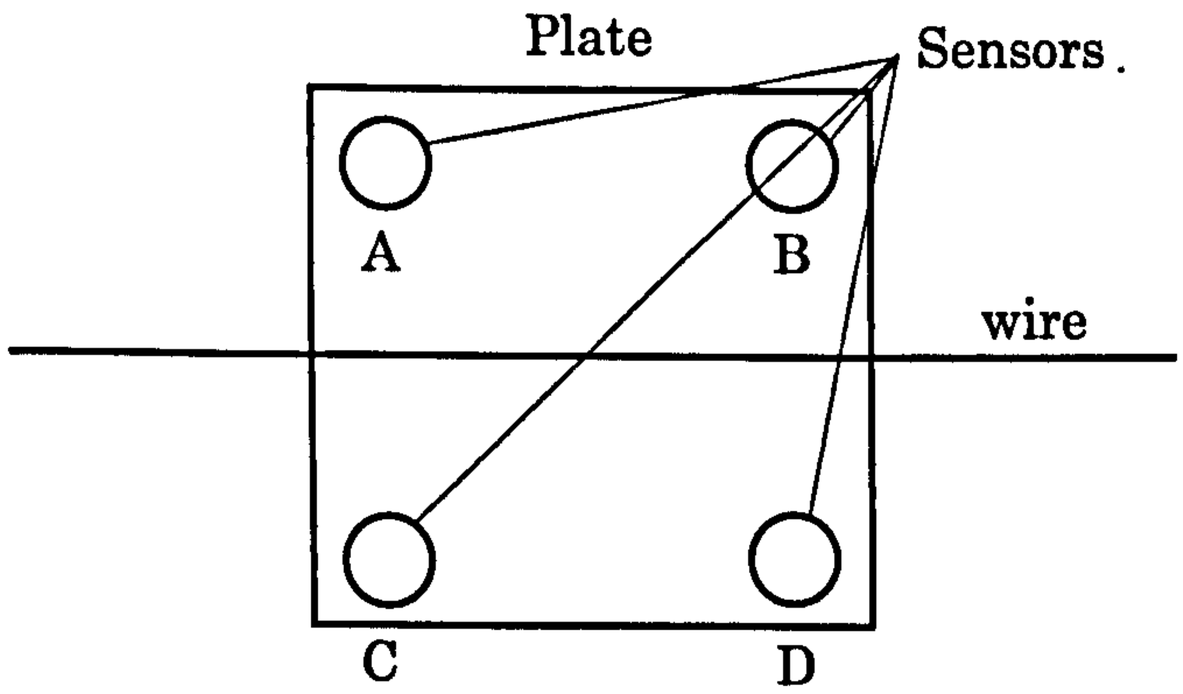


Figure 2.4: Layout of indexed inductor pickups

These will be processed to give two further error outputs, the linear error ratio (e_l):

$$e_l = \frac{A+B}{C+D} - 1 \tag{2.1}$$

and the mean diagonal difference (e_d):

$$e_d = \frac{A+D}{C+B} - 1 \tag{2.2}$$

The existence of e_d implies that the loop is not locally perpendicular to the wire and has an

angular offset, whilst the existence of e_l implies that wire is not passing through the center of the loop, but has a lateral offset.

Using the angle of the plate assembly in the fixed frame, θ, the difference errors may be processed to give commands to the three drive motors: x-axis drive, y-axis drive and loop rotate. The x- and y-axis translational errors (e_x and e_y respectively) can be defined as:

$$e_x = K_t e_l \cos(\theta) \tag{2.3}$$

$$e_y = K_t e_l \sin(\theta) \tag{2.4}$$

where K_t is a scaling constant.

The angular displacement error (e_θ) can be defined as:

$$e_\theta = \tan^{-1}(K_\theta e_d) \tag{2.5}$$

where K_θ is a second scaling constant.

The drives to the translational motors also depend on θ in order to obtain a constant forward speed v:

$$v_x = v \cos(\theta) \tag{2.6}$$

$$v_y = v \sin(\theta) \tag{2.7}$$

Using these relationships we may identify distinct behavioural patterns. The simplest behaviour is forward motion, at a constant speed. For a straight wire, this would drive the loop from one end of the wire track to another. On top of this behaviour we must add wire collision avoidance. This falls into two categories: translational adjustment and rotational adjustment. We can further subdivide translational adjustment into x- and y-axis adjustments.

The behaviors should be layered as follows:

- **Forward motion** Using input from the rotational position sensor to calculate v_x and v_y (Equations 2.6 and 2.7).

- **x-axis adjustment** Using inductor coil inputs to calculate e_x (Equation 2.3).
- **y-axis adjustment** Using inductor coil inputs to calculate e_y (Equation 2.4).
- **Rotation adjustment** Using inductor coil inputs to calculate e_θ (Equation 2.5).

The outputs from each of the behaviours can be summed to give the total outputs for the three drive motors.

2.2 Modelling the Shaky Hand Hardware

The Shaky Hand system model has been created in Simulink [1], a tool for modelling, simulating and analyzing multi-domain dynamic systems, in the Matlab [2] environment. The model can be split into two parts, the Shaky Hand Hardware block and the control system block, as shown in Figure 2.5.

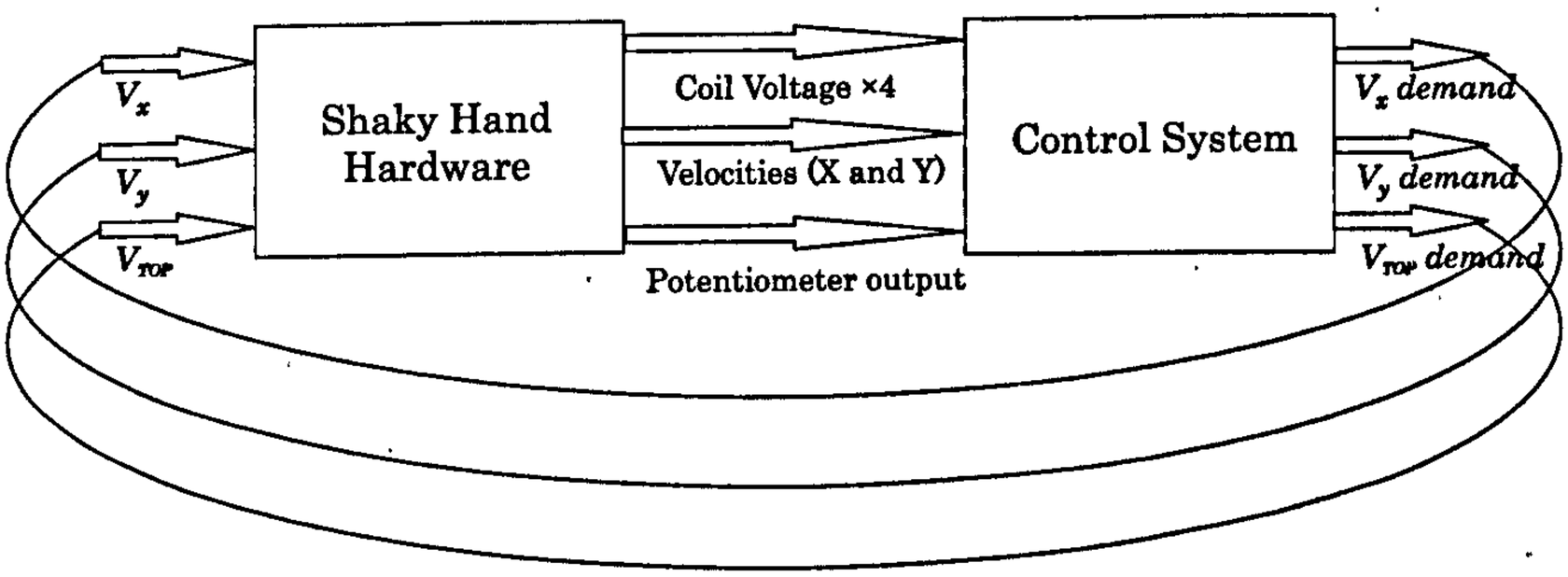


Figure 2.5: Overview of the Shaky Hand model structure

The Hardware block describes all the hardware components used to construct the Shaky Hand, passing essential information to the Control System block and receiving the feedback signals from it. The Control System block describes how the signals from the Hardware components are processed and are converted into appropriate motor signals. Separating the models into hardware and software blocks mimics the Shaky Hand system in practice and also allows the control method to be altered easily without affecting the working function of the hardware component. This chapter concentrates on the work on the Hardware block of the Shaky Hand system model. Essential parts of Shaky Hand Hardware are analysed and

are then developed mathematically. The full dynamic model of the Shaky Hand system may be split into 5 components:

- DC motor model.
- Detection coil induced voltage model.
- Wire model.
- Potentiometer model.
- Encoder model.

In the Shaky Hand, three motors namely, top-, X- and Y-motors, are used to provide rotational and linear motion of the plate. These three motor models are now developed. The top-motor gives rotational motion whereas the X- and Y-motors provide linear motion to the plate. Because of the differences in how each motor is physically implemented in the Shaky Hand system in practice, the mechanical drive models are different for each motor. For example, the load applied to each motor is different. The motor models are detailed in Section 2.3. The detection coil voltage model, described in Section 2.4, shows how an electromotive force (emf) is induced in the coils by the wire track which is carrying current. The wire model used in the simulation is presented in Section 2.5. The motor motions are measured by a potentiometer model and two X- and Y-encoder models to determine the positions of the plate relative to the base frame of the Shaky Hand and the speed at which the plate is travelling. Those models are outlined in Sections 2.6 and 2.7 respectively. The inputs to the Hardware block are the three motor signals and the outputs from the Hardware block are:

- Four coil sensor output voltages.
- Velocities of the plate in both X- and Y-directions measured by the X- and Y-shaft encoders.
- An angular displacement voltage presented by the potentiometer.

The logic of the Hardware block is briefly explained first, followed by the detailed model developments. All the developed models in this chapter are then summarised at the end.

2.2.1 Hardware Plant Logic

It is assumed that the position of the plate, X and Y, the plate angle relative to the base frame, θ, and the velocities of the plate in the X- and Y-directions are known from the outputs of the coil sensors, the shaft encoders and the potentiometer. The wire track function, $y = f(x)$ is user-defined. Figure 2.6 describes the relationship between the plate and the wire track relative to the base frame of the Shaky Hand.

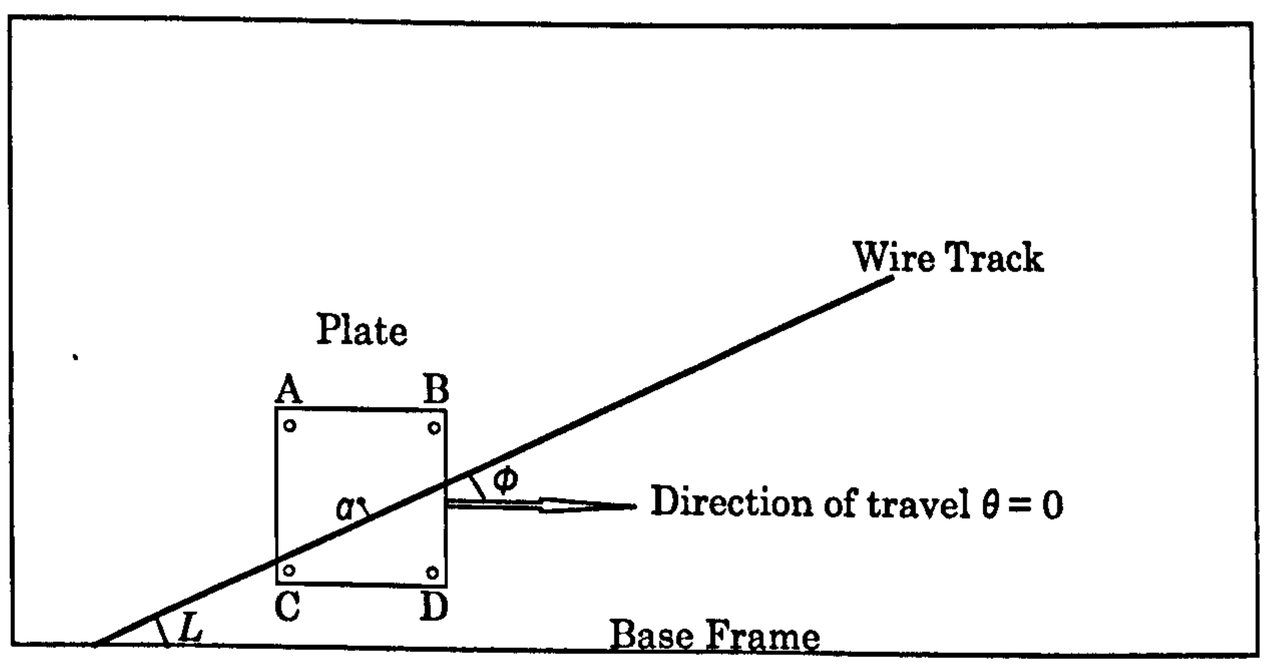


Figure 2.6: Relationship between the plate and the wire track relative to the base frame

Linear and angular velocities of the plate are known from the motor models. The encoder models convert the velocities into the plate displacement. From these models together with the wire model, the lateral offset error, α, and the local slope, L, of the wire track at the current position can be obtained. θ measured by the potentiometer model is used to evaluate the angle of the plate relative to the wire track, ϕ. Using the obtained α and ϕ, the corresponding sensor output voltages are found out.

2.3 DC Motor Model

The DC motors used in the Shaky Hand are A-max-22 Maxon DC motors [4]. The motor models have been constructed by analysing the electrical and the mechanical parts of each motor. From the models, the angular velocity and the X- and the Y-linear velocities of the plate are extracted.

2.3.1 DC Motor Electrical Part Analysis

The electrical part relates the output torque produced by the DC motor to the continuous input voltage applied as shown in Figure 2.7.

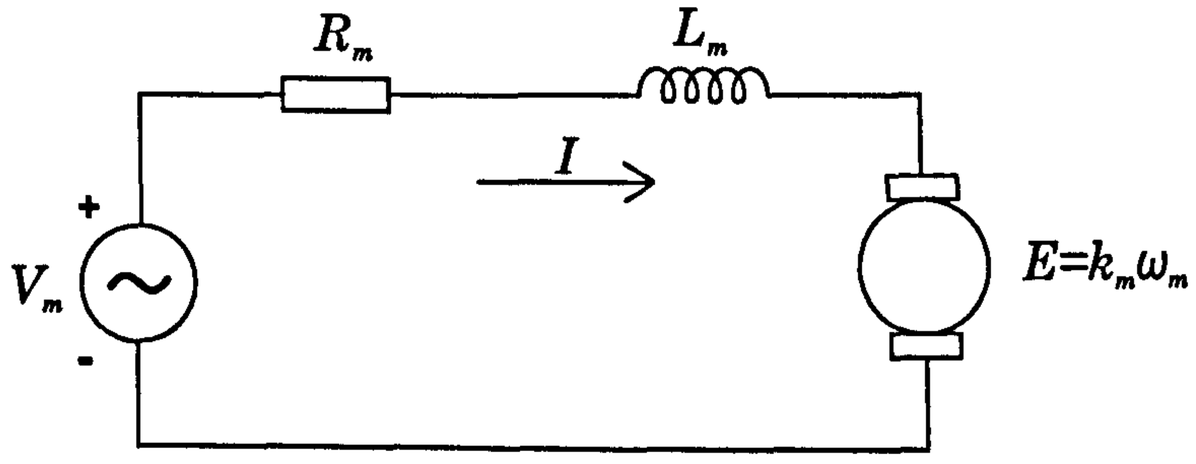


Figure 2.7: DC motor electrical part modelling

where,

V_m = Input voltage applied to the motor

R_m = Resistance of the motor windings

L_m = Inductance of the motor windings

I = Current through the motor

k_m = Electromotive force (emf) constant

E = Back emf produced by the rotor movement

ω_m = Angular velocity of the motor

T_m = Output torque of the motor

When voltage V_m is applied to the motor, this results in current, I, flowing through the resistance (R_m) and the inductance (L_m) of the motor windings. The movement of the rotor results in a back emf, E. This can be expressed as:

$$V_m(t) = R_m I(t) + L_m \frac{dI(t)}{dt} + E \tag{2.8}$$

Taking a Laplace transform;

$$V_m(s) = R_m I(s) + sL_m I(s) + E \tag{2.9}$$

$$V_m - E = R_m I + sL_m I \tag{2.10}$$

The back emf is proportional to the rotational velocity:

$$E \propto \omega_m \qquad or \tag{2.11}$$

$$E = k_m \omega_m \tag{2.12}$$

This constant k_m also relates the current flowing in the motor windings to the output torque of the motor, T_m:

$$T_m = k_m I \tag{2.13}$$

$$I = \frac{T_m}{k_m} \tag{2.14}$$

Substituting Equations 2.12 and 2.14 into Equation 2.10 gives:

$$V_m - k_m \omega_m = \frac{R_m}{k_m} T_m + \frac{L_m}{k_m} s T_m \tag{2.15}$$

$$\frac{L_m}{k_m} s T_m = -\frac{R_m}{k_m} T_m - k_m \omega_m + V_m \tag{2.16}$$

$$\dot{T}_m = -\frac{R_m}{L_m} T_m - \frac{k_m^2}{L_m} \omega_m + \frac{k_m}{L_m} V_m \tag{2.17}$$

2.3.2 DC Motor Mechanical Part Analysis

The mechanical drive between the motors, the rotating plate assembly and the linear motion assemblies (carriages) are now modelled. The effects of the gearing systems used and of the viscous friction must be considered. For simplicity, Coulomb friction, stiction and backlash are initially assumed to be negligible. The assembly structures are also assumed to be perfectly rigid.

There are three motors attached to the Shaky Hand. The top-motor rotationally drives the combined plate and the pick-up assembly through a gearbox. The X- and Y-motors provide the X and Y linear motions respectively. Because of the different mechanical drive structure of the motors, the X- and Y-motors are modelled differently from the top-motor.

The three motors are, for simplicity, modelled as the inertia of the motor armature, I_m, through Newton's second law of motion:

$$T_m = I_m \dot{\omega}_m \tag{2.18}$$

where T_m is the motor torque and $\dot{\omega}_m$ is the motor angular acceleration.

Rotary Mechanical Drive Model

The top-motor can be characterised as shown in Figure 2.8. The plate and the pickup

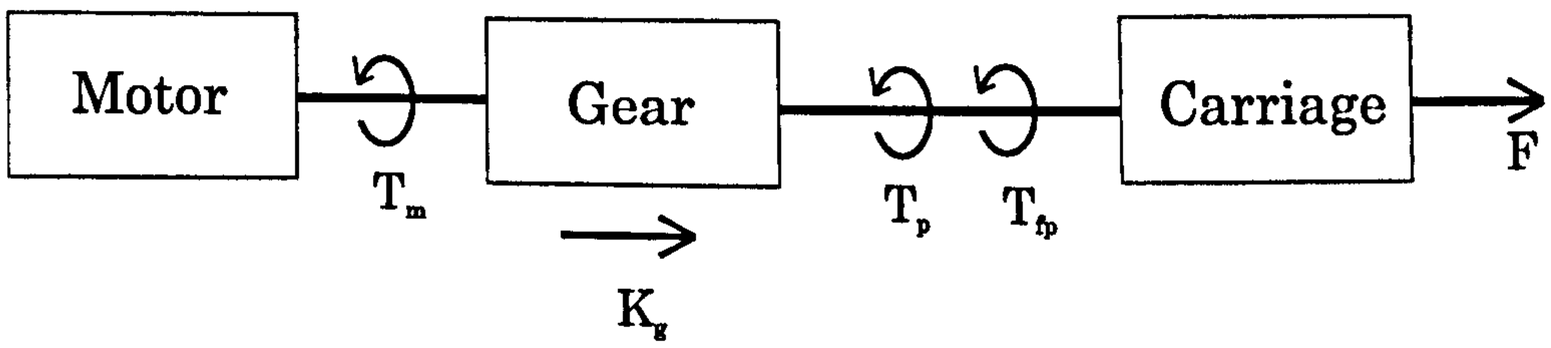


Figure 2.8: Block diagram of the motor attached to the plate and the pick-up assembly

assembly are driven by a single DC motor through a gearbox and produces a plate torque, T_p, with angular velocity, ω_p. The gearing is assumed to be lossless, and backlash-free. Gearing can therefore be represented as a simple gain factor, K_g, relating the plate torque, T_p, to the motor torque, T_m,

$$T_p = K_g T_m \tag{2.19}$$

and also the angular velocity of the plate, ω_p, is derived from the angular velocity of the motor, ω_m, and the gearing.

$$\omega_p = \frac{\omega_m}{K_g} \tag{2.20}$$

The output torque is related to the angular acceleration of the plate through Newton's Second Law:

$$T_p = I_p \dot{\omega}_p \tag{2.21}$$

where I_p is the inertia of the plate and the pickup assembly.

There will be a viscous friction torque, T_{fp}, acting against this driving torque proportional to the angular velocity with a constant, K_{fp}:

$$T_{fp} = K_{fp} \omega_p \tag{2.22}$$

Both the torques, T_p and T_{fp}, which act at the plate are reflected back to the torque produced by the motor, related by the gearing. The total torque at the top-motor is therefore,

$$T_{m_{Top}} = T_m + \frac{T_p}{K_g} + \frac{T_{fp}}{K_g} \tag{2.23}$$

From Equations 2.18, 2.21 and 2.22,

$$T_{m_{Top}} = I_m \dot{\omega}_m + \frac{I_p}{K_g} \dot{\omega}_p + \frac{K_{fp}}{K_g} \omega_p \tag{2.24}$$

This is further expanded using Equation 2.20;

$$T_{m_{Top}} = I_m \dot{\omega}_m + \frac{I_p}{K_g^2} \dot{\omega}_m + \frac{K_{fp}}{K_g^2} \omega_m \tag{2.25}$$

$$= (I_m + \frac{I_p}{K_g^2}) \dot{\omega}_m + \frac{K_{fp}}{K_g^2} \omega_m \tag{2.26}$$

Rearranging Equation 2.26 gives

$$\dot{\omega}_m = \frac{1}{I_m + \frac{I_p}{K_g^2}} T_{m_{Top}} - \frac{\frac{K_{fp}}{K_g^2}}{I_m + \frac{I_p}{K_g^2}} \omega_m \tag{2.27}$$

This is simplified into;

$$\dot{\omega}_m = \frac{K_g^2}{I_m K_g^2 + I_p} T_{m_{Top}} - \frac{K_{fp}}{I_m K_g^2 + I_p} \omega_m \tag{2.28}$$

Linear Mechanical Drive Model

The X- and Y-motors provide linear motions in appropriate directions. The motions are created using backlash-free lead screws converting the motor rotation into the linear motion through a gearbox. The linear mechanical drive model is characterised as shown in Figure 2.9.

In a similar manner to the rotational gearbox, these rods have a conversion factor, K_l, describing the change from the rotational displacement to the linear displacement. Unlike the dimensionless gearbox case, this 'gain' has dimensions of *meters*.

The force on the carriage; F_c, is related to the torque applied by the lead screws, T_m. The

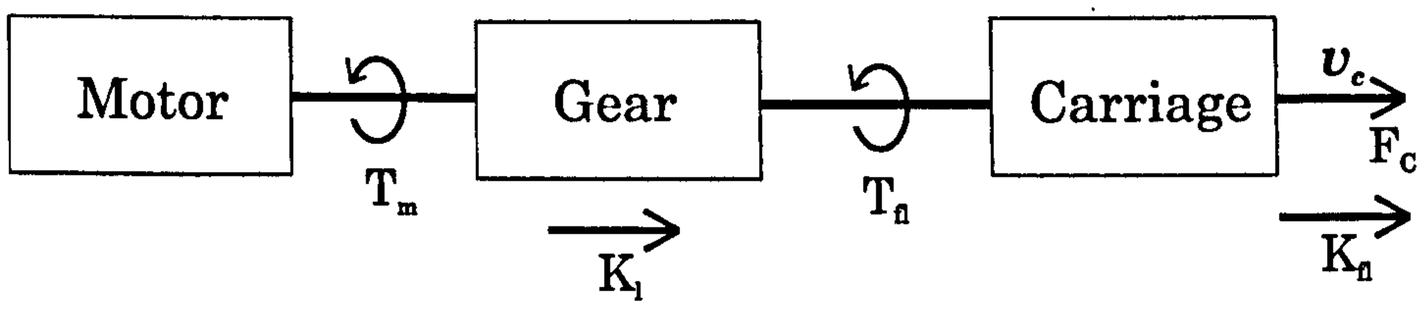


Figure 2.9: Block diagram of the X- and Y-motors attached to the carriages

velocity of the carriage, v_c, is related to its angular velocity, ω_m. Hence,

$$F_c = \frac{T_m}{K_l} \qquad and \tag{2.29}$$

$$v_c = K_l \, \omega_m \tag{2.30}$$

Again, this gearing is assumed to be lossless initially for simplicity. The linear motion of the carriage carrying the plate can be derived using Newton's Second Law;

$$F_c = M_c \, \dot{v}_c \tag{2.31}$$

where M_c is the mass of the carriage assembly.

Both the X-, Y-linear motions and the rotational motion suffer from friction in motion of the carriages and the rotation at the bearing respectively. The viscous friction torque, T_{fl}, acting against the driving torque proportional to the linear velocity, and force acting against the linear motion, F_{fc}, are modelled; all other friction effects are ignored.

$$T_{fl} = K_{fl} \, \omega_m \tag{2.32}$$

$$F_{fc} = K_{fc} \, v_c \tag{2.33}$$

Total force on the carriage is therefore;

$$F_c - F_{fc} = M_c \, \dot{v}_c \tag{2.34}$$

$$F_c - K_{fc} \, v_c = M_c \, \dot{v}_c \tag{2.35}$$

The force is reflected back through the gearbox to the motor. Substituting Equation 2.29

into Equation 2.35 gives;

$$T_m = K_l\, M_c\, \dot{v}_c + K_l\, K_{fc}\, v_c \tag{2.36}$$

and from Equation 2.30,

$$T_m = K_l^2\, M_c\, \dot{\omega}_m + K_l^2\, K_{fc}\, \omega_m \tag{2.37}$$

Torque produced by the motor can be modelled using Newton's second law of motion as before. Adding the effect of the associated vicious friction as stated in Equation 2.32 gives;

$$T_m - K_{fl}\, \omega_m = I_m\, \dot{\omega}_m \tag{2.38}$$

$$T_m = I_m\, \dot{\omega}_m + K_{fl}\, \omega_m \tag{2.39}$$

Thus, the total torque at the X- and the Y-motor, T_{mXY}, is obtained by adding Equations 2.37 and 2.39 together.

$$T_{mXY} = K_l^2\, M_c\, \dot{\omega}_m + K_{fc}\, K_l^2\, \omega_m + I_m\, \dot{\omega}_m + K_{fl}\, \omega_m \tag{2.40}$$

$$= (I_m + K_l^2\, M_c)\, \dot{\omega}_m + (K_{fl} + K_{fc}\, K_l^2)\, \omega_m \tag{2.41}$$

Rearranging Equation 2.41;

$$\dot{\omega}_m = \frac{1}{I_m + K_l^2\, M_c}\, T_{mXY} - \frac{K_{fl} + K_{fc}\, K_l^2}{I_m + K_l^2\, M_c}\, \omega_m \tag{2.42}$$

2.3.3 DC Motor State Space Equations

From the derived equations, state-space equations for each motor can be produced.

For the top-motor model:

$$\begin{bmatrix} \dot{T}_m \\ \dot{\omega}_m \end{bmatrix} = \begin{bmatrix} -\frac{R_m}{L_m} & -\frac{k_m^2}{L_m} \\ \frac{K_g^2}{I_{ptop} + I_m\, K_g^2} & -\frac{K_{fp}}{I_{ptop} + I_m\, K_g^2} \end{bmatrix} \begin{bmatrix} T_m \\ \omega_m \end{bmatrix} + \begin{bmatrix} \frac{k_m}{L_m} \\ 0 \end{bmatrix} V_m$$

The output of the top-motor is the angular velocity of the plate, ω_p.

$$\omega_p = \begin{bmatrix} 0 & \frac{1}{K_g} \end{bmatrix} \begin{bmatrix} T_m \\ \omega_m \end{bmatrix}$$

For the X-motor;

$$\begin{bmatrix} \dot{T}_m \\ \dot{\omega}_m \end{bmatrix} = \begin{bmatrix} -\frac{R_m}{L_m} & -\frac{k_m^2}{L_m} \\ \frac{1}{I_m + K_l^2 M_{cX}} & -\frac{K_{fl} + k_{fc} K_l^2}{I_m + K_l^2 M_{cX}} \end{bmatrix} \begin{bmatrix} T_m \\ \omega_m \end{bmatrix} + \begin{bmatrix} \frac{k_m}{L_m} \\ 0 \end{bmatrix} V_m$$

The output of the X-motor is the linear velocity of the carriage in the X direction. This is represented as;

$$v_{cX} = \begin{bmatrix} 0 & K_l \end{bmatrix} \begin{bmatrix} T_m \\ \omega_m \end{bmatrix}$$

Similarly, for the Y-motor;

$$\begin{bmatrix} \dot{T}_m \\ \dot{\omega}_m \end{bmatrix} = \begin{bmatrix} -\frac{R_m}{L_m} & -\frac{k_m^2}{L_m} \\ \frac{1}{I_m + K_l^2 M_{cY}} & -\frac{K_{fl} + K_{fc} K_l^2}{I_m + K_l^2 M_{cY}} \end{bmatrix} \begin{bmatrix} T_m \\ \omega_m \end{bmatrix} + \begin{bmatrix} \frac{k_m}{L_m} \\ 0 \end{bmatrix} V_m$$

$$v_{cY} = \begin{bmatrix} 0 & K_l \end{bmatrix} \begin{bmatrix} T_m \\ \omega_m \end{bmatrix}$$

Table 2.1 summarises the parameters and their values. They are obtained either from the data sheets or experimental measurements.

Abbreviations	Definition	Value
R_m	Resistance of the motor windings	13.6Ω
L_m	Inductance of the motor windings	0.89mH
k_m	Electromotive Force Constant	17.1mNm/A
K_l	Conversion ratio from angle of rotation to linear displacement	2.143×10^{-4}m
K_g	Gearbox gain	7
K_{fl}	'Gain' in linear motion due to friction in motion of the carriage	0
K_{fc}	'Gain' in rotational motion due to friction in motion of the carriage at the bearing	0
K_{fp}	Viscous friction torque at the bearing of the plate	0
I_m	Rotor inertia of the motor	$3.970\times10^{-7}kgm^2$
$I_{p_{top}}$	The moment of inertia of the plate assembly	$1.6077\times10^{-4}kgm^2$
Mc_X	The mass of the carriage assembly on X-motor	1.6752kg
Mc_Y	The mass of the carriage assembly on Y-motor	0.2924kg

Table 2.1: Date sheet for the motors

2.4 Detection Coil Induced Voltage Model

The electromagnetic coupling between the wire and each of the four inductors is considered in this section. For simplicity it is assumed that the wire is locally straight. This allows us to consider only the proximity of the closest point. A basic relationship is assumed between the emf coupled into the inductor, V_l, and the proximity, δ, and frequency of the a.c. waveform on the wire, ω as shown below. K is a constant term.

$$V_l \propto \frac{\omega}{\delta} \tag{2.43}$$

$$V_l = K\frac{\omega}{\delta} \tag{2.44}$$

Assuming the wire is locally straight, an expression for the proximity of the wire (δ) in terms of the lateral offset between the center of the plate and the wire (α) and its angle of offset (ϕ) both measured in a frame relative to the plate may be developed. The geometry of the plate is illustrated in Figure 2.10.

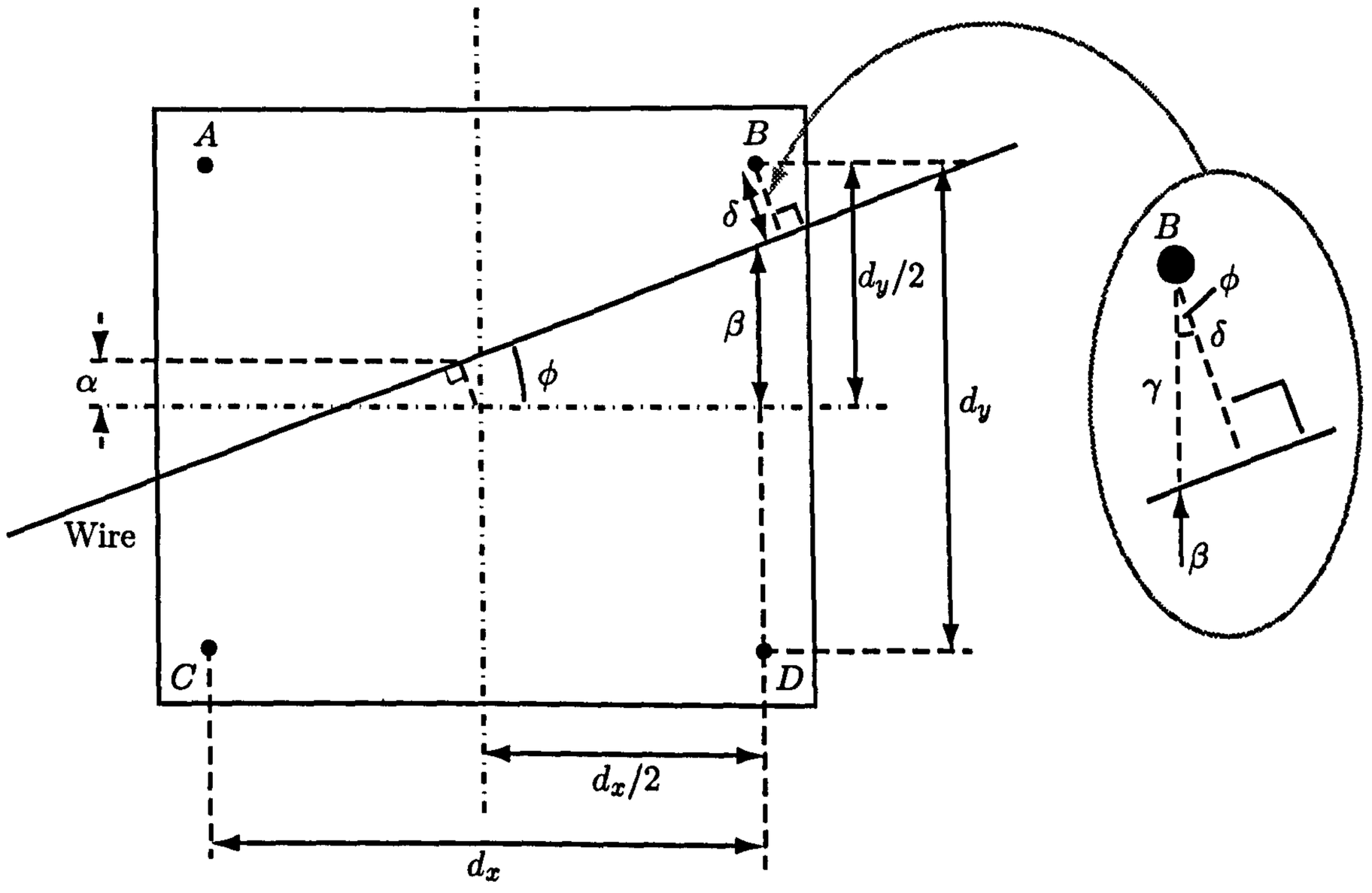


Figure 2.10: Geometry of the plate/wire interaction

Consider only the inductor B in the first instance. An expression for δ_B when there is no lateral offset, α, may be developed as:

$$\beta = \frac{d_x}{2}\tan(\phi) \tag{2.45}$$

Adding the effect of α:

$$\gamma = \frac{d_y}{2} - (\alpha + \beta) \tag{2.46}$$

$$\delta_B = \gamma\cos(\phi) \tag{2.47}$$

$$\therefore \quad \delta_B = \left[\frac{d_y}{2} - \left(\alpha + \frac{d_x}{2}\tan(\phi)\right)\right]\cos(\phi) \tag{2.48}$$

$$= \left(\frac{d_y}{2} - \alpha\right)\cos(\phi) - \frac{d_x}{2}\sin(\phi) \tag{2.49}$$

Similarly, expressions for δ_A, δ_C and δ_D can be developed.

$$\delta_A = \left(\frac{d_y}{2} - \alpha\right)\cos(\phi) + \frac{d_x}{2}\sin(\phi) \tag{2.50}$$

$$\delta_C = \left(\frac{d_y}{2} + \alpha\right)\cos(\phi) - \frac{d_x}{2}\sin(\phi) \tag{2.51}$$

$$\delta_D = \left(\frac{d_y}{2} + \alpha\right)\cos(\phi) + \frac{d_x}{2}\sin(\phi) \tag{2.52}$$

Using small angle approximations, the expressions above can be simplified into:

$$\delta_A = \frac{d_y}{2} + \frac{d_x}{2}\,\phi - \alpha \tag{2.53}$$

$$\delta_B = \frac{d_y}{2} - \frac{d_x}{2}\,\phi - \alpha \tag{2.54}$$

$$\delta_C = \frac{d_y}{2} - \frac{d_x}{2}\,\phi + \alpha \tag{2.55}$$

$$\delta_D = \frac{d_y}{2} + \frac{d_x}{2}\,\phi + \alpha \tag{2.56}$$

Each inductor is connected to an LCR tuned circuit to give a resonant response at the energizing frequency of the signal on the wire ($140KHz$). The resulting induced signal is then amplified and passed through a first order, anti-aliasing filter. The emf generated in the inductor is inversely proportional to the distance δ from the wire. Setting:

$$V_l = \frac{1}{\delta} \tag{2.57}$$

the amount of voltage induced by the emf is therefore

$$V_{coil} = \frac{N\,\mu\,A\,I}{C\,R_m\,2\,\pi}\,V_l \tag{2.58}$$

$$\triangleq K_{coil}\,V_l \tag{2.59}$$

where

N = Number of turns of a coil

μ = Relative permeability

A = Area of the windings

I = Current flowing in the windings

C = Capacitance of the windings

R_m = Resistance of the windings

K_{coil} = A scaling constant

The voltage value, calculated above, will be amplified by a non-inverting amplifier with gain, K_{amp}. The amplified voltage is then converted to an rms value by a factor $K_{rms} = \frac{1}{\sqrt{2}}$. The rms voltage value from each inductor represents the output from this model and this is passed onto the Shaky Hand controller model.

$$\therefore \quad V_{o\,rms} = K_{coil}\, K_{amp}\, K_{rms}\, V_i \tag{2.60}$$

$$= K_{coil}\, K_{amp}\, K_{rms}\, \frac{1}{\delta} \tag{2.61}$$

Chapter 4 considers the physical constructions of the Shaky Hand hardware including all the circuitry designs.

2.5 Wire Model

The wire model enables the estimation of the positions of the center of the plate relative to the wire track. This gives the minimum distance between the center of the plate and the nearest point on the wire where it is perpendicular to the center of the plate. i.e. the lateral offset, α, to be found out. Once the coordinates of the plate and the wire track are known, ϕ can be also estimated using the known local slope of the track and the potentiometer output, θ.

The position search mechanism is based on the equation of a straight line. This was initially developed using Maple [3] and then implemented on Simulink [1] platform. Let us assume the following situation as illustrated in Figure 2.11.

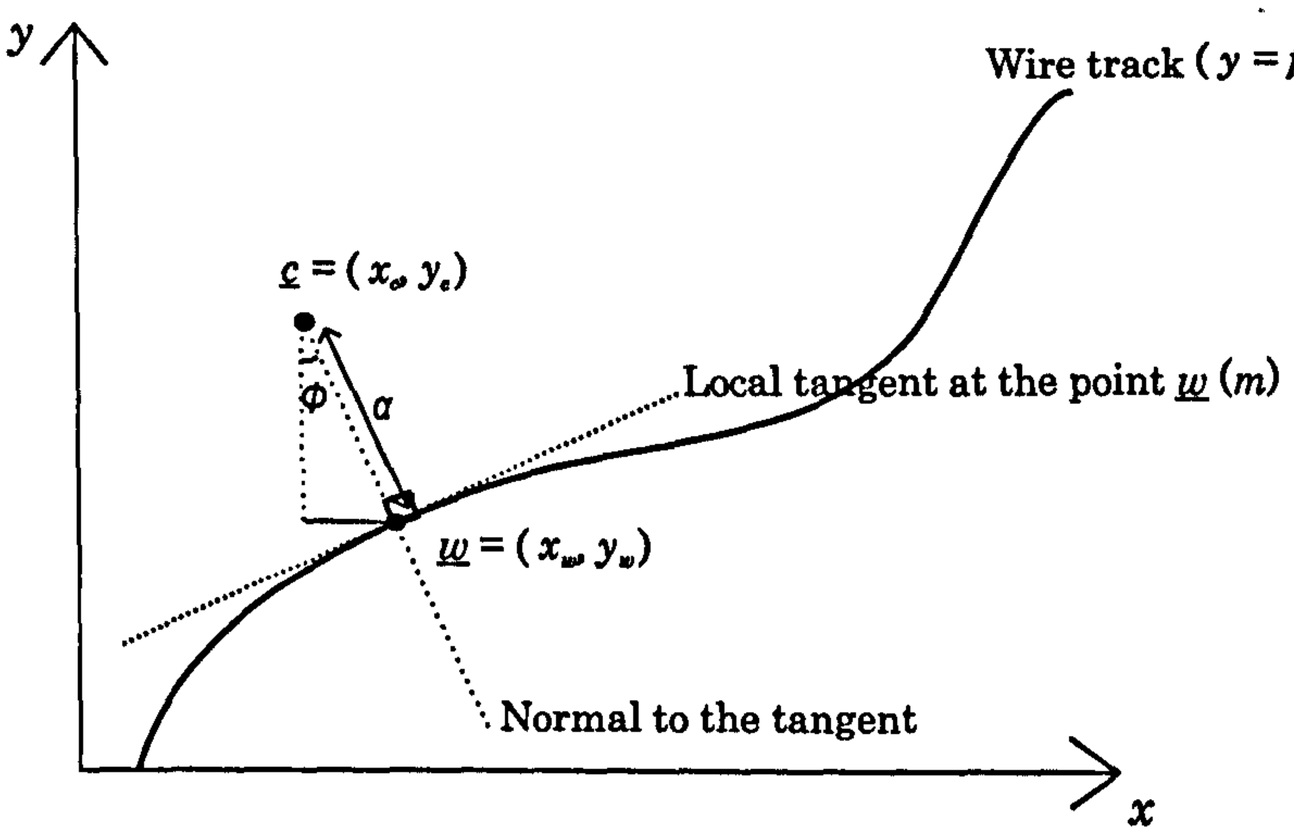


Figure 2.11: Wire model mechanism

The wire track is defined by a known function:

$$y = f(x) \tag{2.62}$$

$\underline{c}$ is the position of the centre of the plate. $\underline{w}$ is the nearest point on the wire track to the center of the plate. (x_c,y_c) and (x_w,y_w) are the coordinates of those points respectively. The coordinate (x_c,y_c) is obtained from the encoders attached to the X- and Y-shafts. (x_w,y_w) must be estimated from this model.

The equation of a straight line is:

$$y - y_1 = m(x - x_1) \tag{2.63}$$

where (y_1, x_1) is an arbitrary coordinate on the wire track and m is a slope of the tangent at that point. In this case,

$$y_1 = y_w \tag{2.64}$$

$$x_1 = x_w \tag{2.65}$$

$$m = \frac{dy_w}{dx_w} \tag{2.66}$$

The equation of the line through $\underline{w}$, normal to the tangent of the point, has a slope m_1 such

that

$$m\,m_1 = -1 \tag{2.67}$$

So the slope of the normal which passes through the point $\underline{c}$ and $\underline{w}$ is

$$m_1 = -\frac{1}{m} \tag{2.68}$$

Therefore the equation of a straight line going through those two points is expressed as

$$y_c - y_w = -\frac{1}{\frac{dy_w}{dx_w}}(x_c - x_w) \tag{2.69}$$

This is best explained using an example as shown in Figure 2.12 whose function is

$$y = \frac{x^2}{2} \tag{2.70}$$

and this is equivalent to:

$$y_w = \frac{x_w^2}{2} \tag{2.71}$$

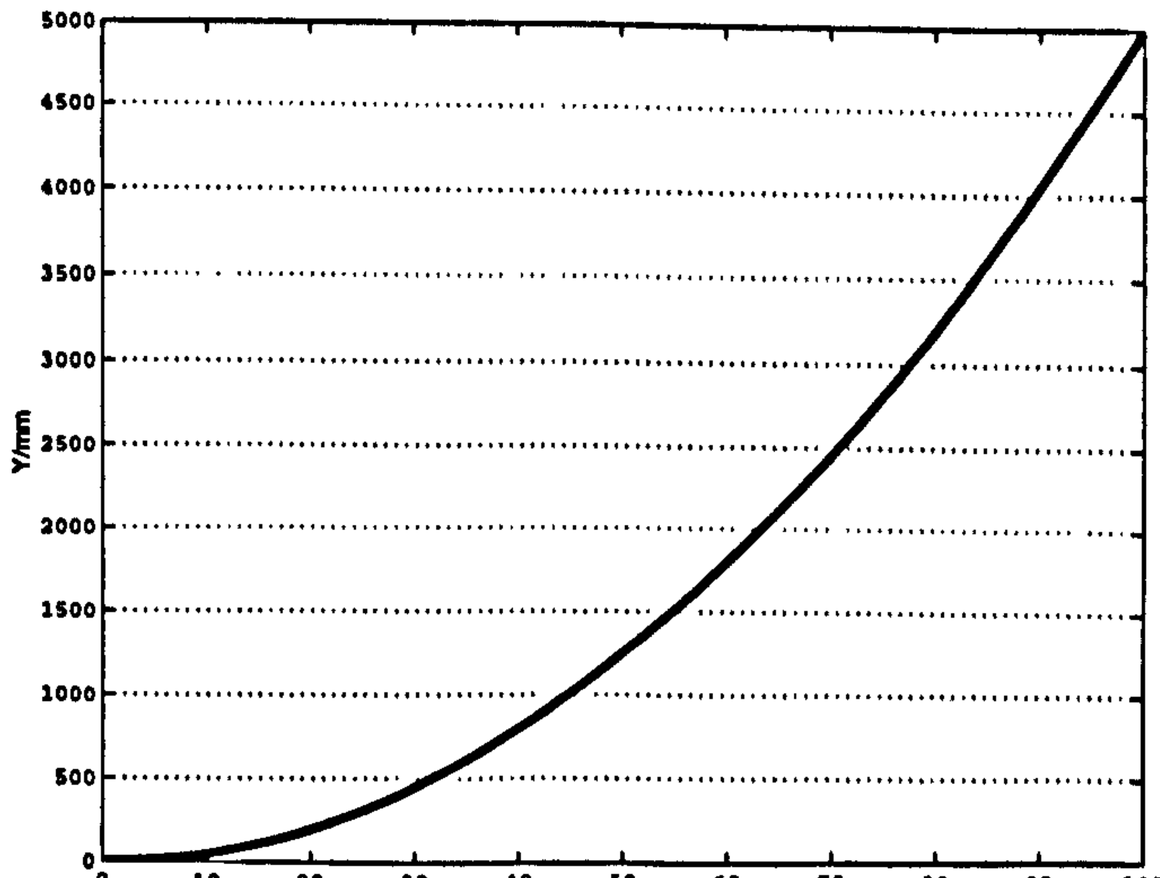


Figure 2.12: A wire track function example

The slope at a point on the curve is

$$m = \frac{dy_w}{dx_w} = x_w \tag{2.72}$$

Therefore the equation of a straight line passing through the center of the plate and the nearest point on the curve is:

$$y_c - y_w = -\frac{1}{x_w}(x_c - x_w) \tag{2.73}$$

Substituting Equation 2.71 into Equation 2.73 gives

$$y_c - \frac{x_w^2}{2} = -\frac{1}{x_w}(x_c - x_w) \tag{2.74}$$

Since the coordinate (y_c, x_c) is known by the system, x_w can be solved for. Substituting the obtained x_w into Equation 2.71 gives the value of y_w. Then the coordinate, (x_w, y_w), is obtained. From this information, α and ϕ can be evaluated. The relationship of the coordinates and the offset errors is illustrated in Figure 2.13.

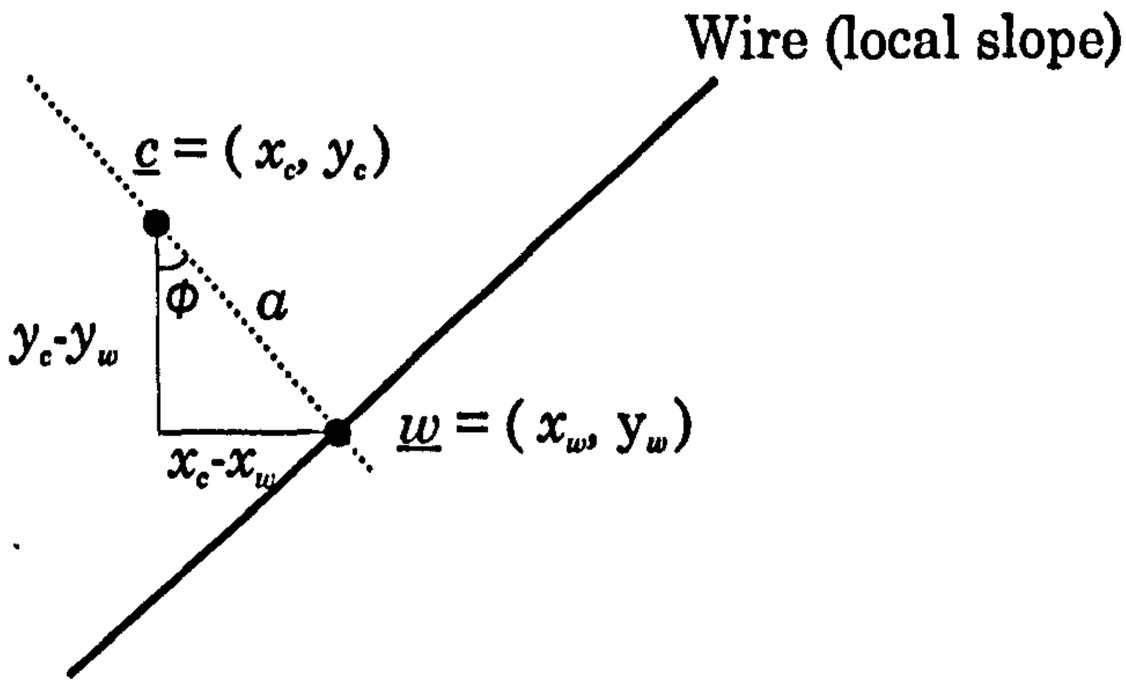


Figure 2.13: Coordinates and offset errors

The distance, α, between the center of the plate and the nearest point on the wire track, can be calculated using Pythagoras theorem below.

$$\alpha = \sqrt{(y_c - y_w)^2 + (x_c - x_w)^2} \tag{2.75}$$

The local slope, L, is obtained by

$$L = \arctan\left(\frac{dy_w}{dx_w}\right) \tag{2.76}$$

The difference between the local slope and the potentiometer output, θ, is the angle offset error ϕ.

$$\phi = L - \theta \tag{2.77}$$

From the wire model, the offset error terms, α and ϕ, are obtained. The wire function must be known to obtain the positions. The position search mechanism was then implemented into the Simulink.

2.5.1 Wire Track Used in the Simulink Simulation

The wire track used in the simulation of the Shaky Hand models is produced by combining three different functions. Since the Shaky Hand assumes the wire is locally straight, tight curves are avoided. The functions are:

1. $y = 0.746\,x$ for x=0:100 mm

2. $y = 74.6$ for x=100:150 mm

3. $y = \frac{1}{1000}\,x^2$ for x=150:600 mm

The virtual track is illustrated in Figure 2.14. Coordinate (x,y) is equivalent to the physical displacement on the base frame and has a unit of *mm*. The functions were chosen to exercise the Shaky Hand across the transitions and to give specific types of movement. The first phase requires simultaneous x- and y-direction movement, the second phase considers the case where one channel has no signal and the third considers a dynamically changing mix of the two. The phase 1 to phase 2 transition is relatively abrupt, whereas the second transition is quite gentle. These wire functions are totaly different from the CGP training sets used for evolving the fault tolerant solutions (Section 5.4.1), therefore, if the solutions work on this model, we can reasonably say that the solutions are generic. Each function is analysed to

obtain the position search algorithm.

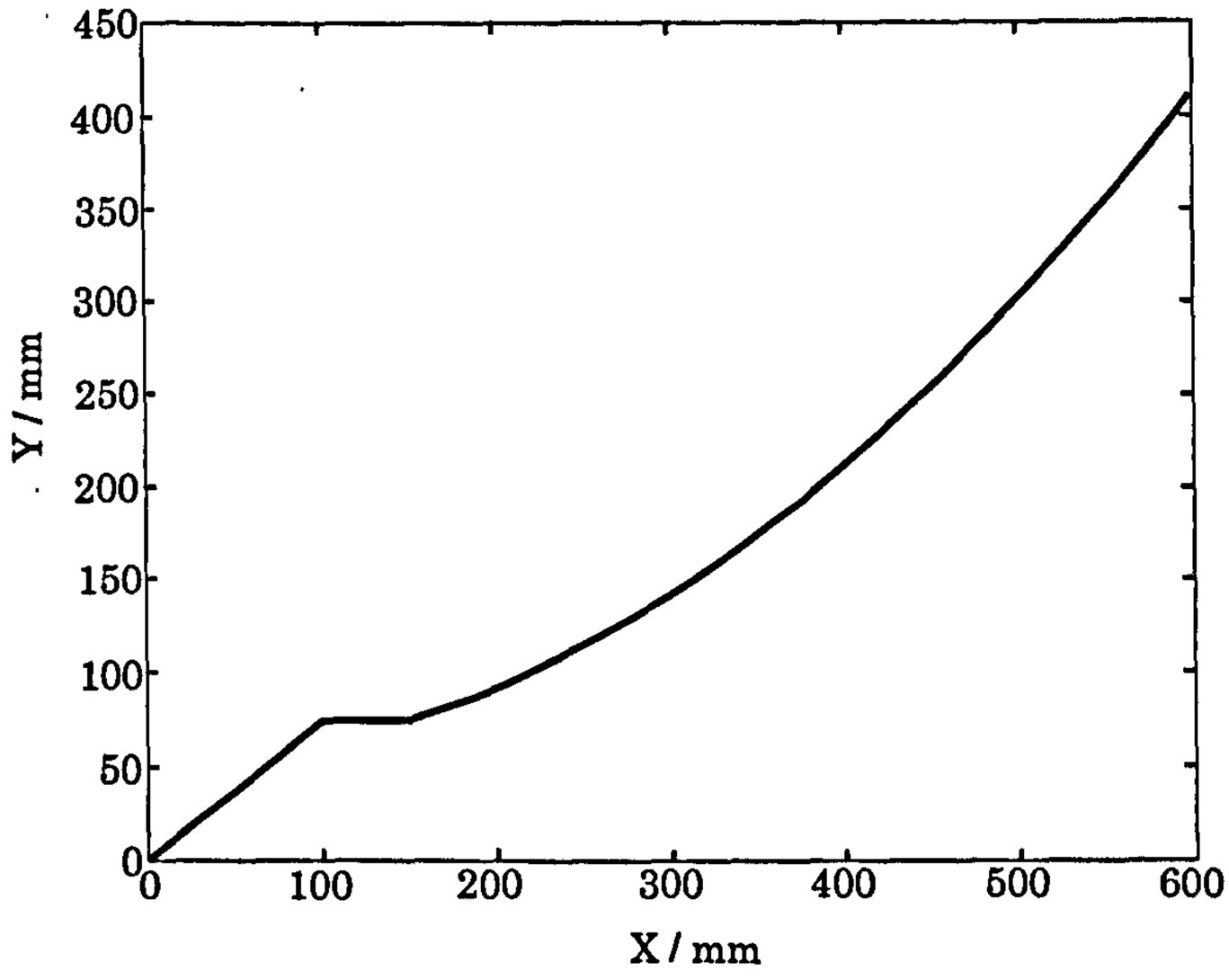


Figure 2.14: Wire track used in the simulation

The first track is a slope with a constant gradient of 0.746. Therefore:

$$y_{w1} = 0.746\, x_{w1} \tag{2.78}$$

$$m = \frac{dy_w}{dx_w} = 0.746 \tag{2.79}$$

Expanding Equation 2.69 and from Equation 2.78

$$y_c - m\, x_{w1} = -\frac{1}{m}\, x_c + \frac{1}{m}\, x_{w1} \tag{2.80}$$

Rearranging Equation 2.80 gives

$$x_{w1} = \frac{y_c}{m + \frac{1}{m}} + \frac{x_c}{m\,(m + \frac{1}{m})} \tag{2.81}$$

Substituting Equation 2.79 into Equation 2.81 produces

$$\therefore\ x_{w1} = 0.4794\, y_c + 0.64246\, x_c \tag{2.82}$$

Substituting the obtained x_{w1} into Equation 2.78 gives out the value of y_{w1}.

The second function is a straight line and has zero gradient. Therefore:

$$y_{w2} = 74.6 \tag{2.83}$$

The third function is a parabola and is solved in similar manner. The gradient is:

$$\frac{dy_w}{dx_w} = \frac{x_w}{500} \tag{2.84}$$

and

$$x_w^3 + (500000 - 1000y_c)x_w - 500000\, x_c = 0 \tag{2.85}$$

Then the real root is evaluated and is substituted into Equation 2.86 for y_w.

$$y_{w3} = \frac{1}{1000}\, x_w^2 \tag{2.86}$$

2.6 Potentiometer Model

A potentiometer converts the angular displacement of the plate into analogue voltages. The one used in the Shaky Hand is a three-turn potentiometer whose helical wiper track allows up to three revolutions of the shaft. One revolution is equivalent to 360 degrees thus three turns enable the potentiometer to measure up to 1080 degrees. The Simulink models use radian representation, therefore, the maximum measurable angle is 6π. The potentiometer is powered by a 5V supply source and is calibrated to give out half the power supply voltage when the plate is aligned with the base frame. Therefore the potentiometer output can be calculated from:

$$V_{pot} = \frac{2.5 * \theta}{6\pi} + 2.5 \tag{2.87}$$

The output corresponds to the plate angle relative to the base frame.

2.7 Encoder Model

The linear velocities in the X- and Y-directions of the fixed frame are measured using two digital optical shaft encoders in practice. In the Simulink model, the linear velocities from the motor model are converted into linear displacement using an integrator.

$$X_c = \frac{v_x}{s} \tag{2.88}$$

$$Y_c = \frac{v_x}{s} \tag{2.89}$$

2.8 Plant Model Summary

All the essential components of the Hardware block of the Shaky Hand system have been analysed and the models have been mathematically developed. Figure 2.15 summarises those models developed in this chapter.

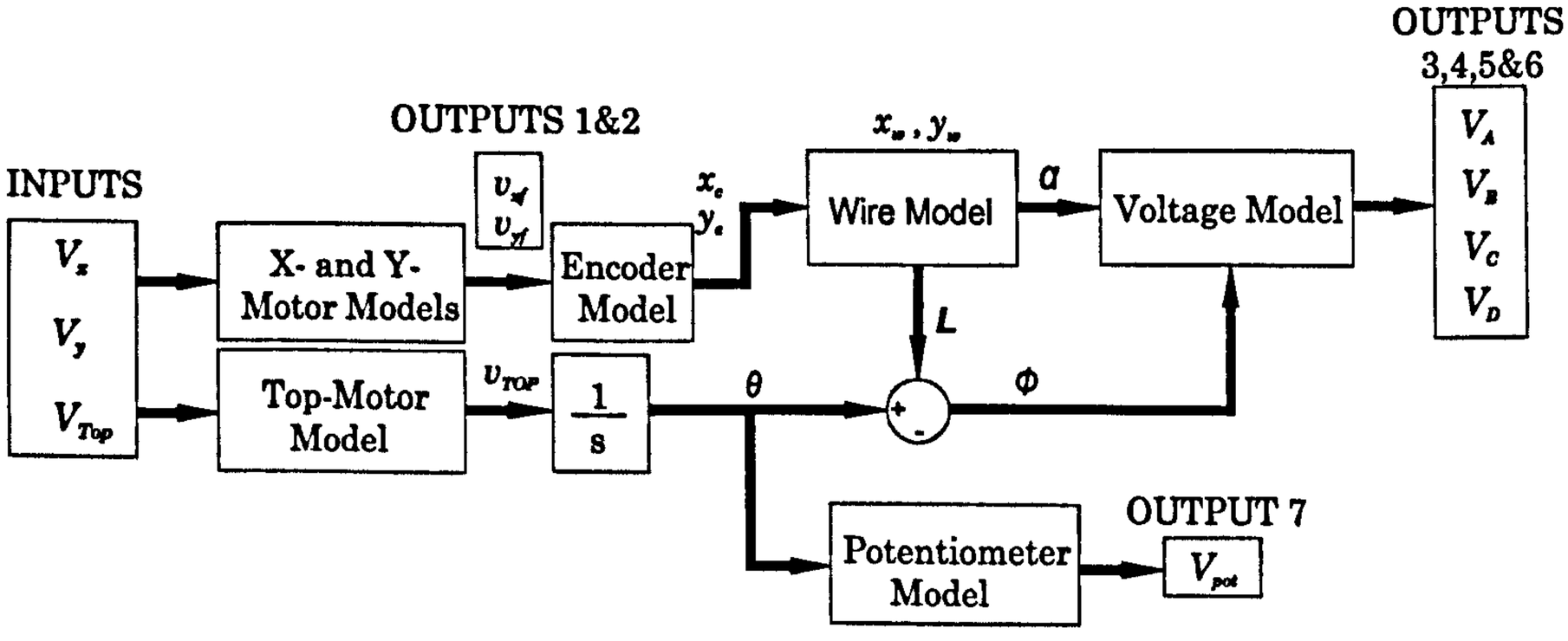


Figure 2.15: Summary of the Hardware Block models

The motor model converts the motor drive voltage demand signals into the velocities. Encoders converts the velocities into linear displacement. The wire model calculates the lateral offset α and the local slope L. The potentiometer model measures θ in terms of voltage, V_{pot}. Then ϕ is obtained. Using the offset errors the appropriate induced voltages in the coil sensors, V_A, V_B, V_C and V_D, are evaluated. The output signals are sent to the Control System block and are processed. The motor drive voltages are fed back to the Hardware block as stated before.

2.9 Chapter Bibliography

[1] THE MATHWORKS INC. *Simulink, Dynamic System Simulation for Matlab.* THE MATHWORKS INC., Natick, MA, 3 edition, 1990.

[2] THE MATHWORKS INC. *Matlab Version 5.3.* THE MATHWORKS INC., Natick, MA, September 1999.

[3] WATERLOO MAPLE INC. *Maple Version 8.0.* WATERLOO MAPLE INC., Maplesoft, Waterloo, Ontario, Canada, April 2002.

[4] MAXON. *Maxon Motor 12VDC, A-max 22.* MAXON. http://www.farnell.com/datasheets/8366.pdf, August 2007.

Chapter 3

The Control System Model Implementation

Contents

3.1 Introduction 53
3.2 Two Reference Frames and Two Plate Velocities 53
3.3 Control Loops 54
3.3.1 Rotational Offset Loop 55
3.3.2 Lateral Offset Loops 56
3.4 Control System Model 57
3.4.1 Offset Error Sensing Solutions 58
3.4.2 Reverse Potentiometer Model 59
3.4.3 Velocity Error Model 59
3.4.4 Controller Models 60
3.4.5 Transform Model 62
3.5 Simulink Simulation 63
3.6 Introducing I/O Programming 69
3.6.1 I/O software 69
3.6.2 Speed Measurement Program 71
3.6.3 Analogue Input Program 72
3.6.4 Control System Program 72
Psedocode for the Control System Program 73
3.6.5 Digital Output Program 73
3.6.6 Analogue Output Program 74
3.7 Chapter Bibliography 74

3.1 Introduction

In this chapter, the Shaky Hand Control System models are developed and described in detail. As described in Chapter 2 and also as shown in Figure 3.1, the Control System block takes in all the outputs from the coil sensors, the optical encoders and the potentiometer and processes them to give appropriate feedback signals for the three dc motors.

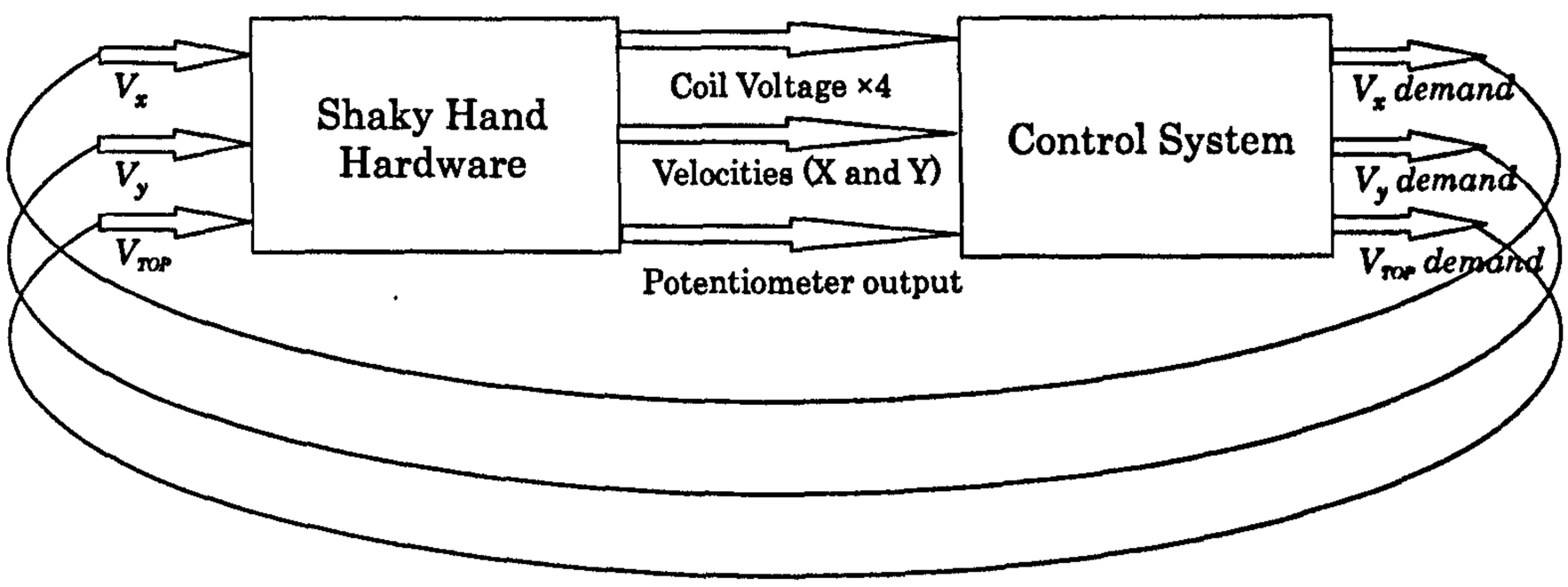


Figure 3.1: Overview of the Shaky Hand model structure

This chapter firstly describes the two frames used in the plate control process and defines some important velocity component terms. The control loops are then discussed. This is followed by the analysis and the development of the Control System models. All the models developed in Chapter 2 and in this chapter are implemented in Simulink. Input-output (I/O) software used for the practical implementation of the control of the Shaky Hand is described at the end of the chapter.

3.2 Two Reference Frames and Two Plate Velocities

The control process is described using two reference frames. One is the plate frame and the other is the base frame, as shown in Figure 3.2. Each frame has an X-Y-axis set, the plate axis set and the base axis set. The plate frame defines the wire track position relative to the plate. The base frame relates the plate position relative to the base frame.

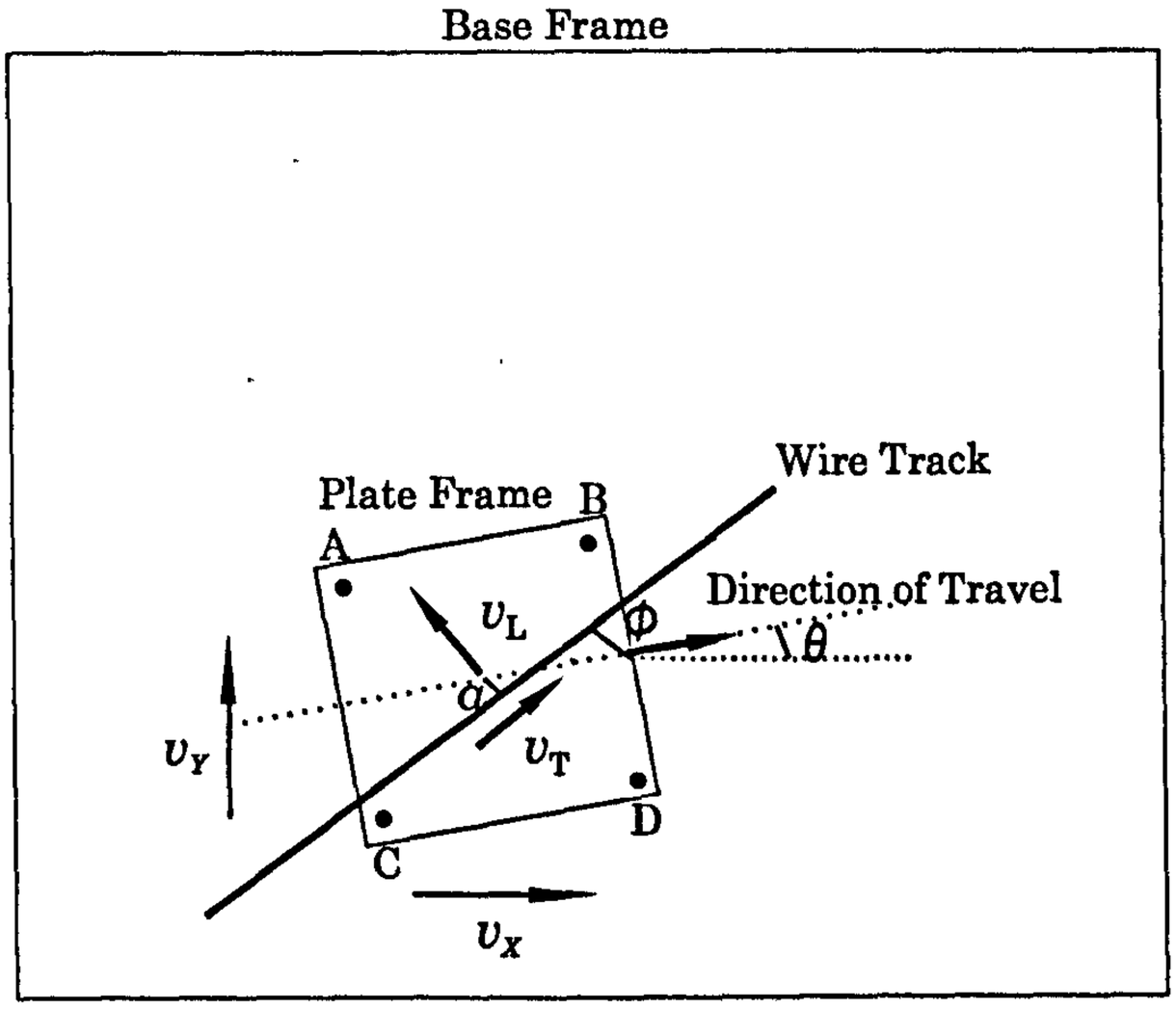


Figure 3.2: Two reference frames

The velocities of the plate in two directions are defined as v_T and v_L. v_T is the tangential velocity along the wire track. v_L is the lateral velocity perpendicular to v_T. It appears only when the plate deviates laterally with respect to the wire. The speed of the plate along the wire track is the resultant of these two velocities and is kept constant, at a set point, during the operation of the Shaky Hand.

In Figure 3.2, the angle error relative to the wire track, ϕ, and the lateral offset, α, exist. Therefore both angular and lateral velocity demands need to be applied, whilst the tangential velocity is adjusted to keep the plate forward motion at the set point value. It is essential to maintain angular alignment (zero ϕ) at all times in order to keep the plate orientation (and therefore, θ,) correct at all times. This means that the angular alignment control process should be faster than the linear motion control of the plate. The assumption of the fast angular motion control loop allows v_T and v_L to be resolved into the X- and Y-directions (v_{TX}, v_{TY}, v_{LX} and v_{LY}) through θ because ϕ is kept zero. θ is the angle offset of the plate relative to the base frame.

3.3 Control Loops

Three controller loops are incorporated in the Shaky Hand system. Each loop is associated with a DC motor attached to the Shaky Hand, controlling the speed, the position and the orientation of the plate. The angular offset loop is associated with the top-motor, providing

a feedback system for plate rotation. The velocity loops are associated with the X- and Y-motors, producing feedback signals for the plate linear motions. These control loops use error signals to produce motor demand voltages.

3.3.1 Rotational Offset Loop

We assume that the plate is always correctly aligned with the wire track, thus the output from the potentiometer is a measure of the local orientation of the wire in the base reference frame. This means that the angular loop must be considerably faster than the others. A loop associated with the angular alignment of the plate can be represented as shown in Figure 3.3. The angle offset error, ϕ, is obtained from the difference between the local slope of the wire track, L, and the angle of the plate relative to the base frame, θ. The presence of ϕ induces rotational motion of the plate. The motor rotates the plate until the angular offset is eliminated.

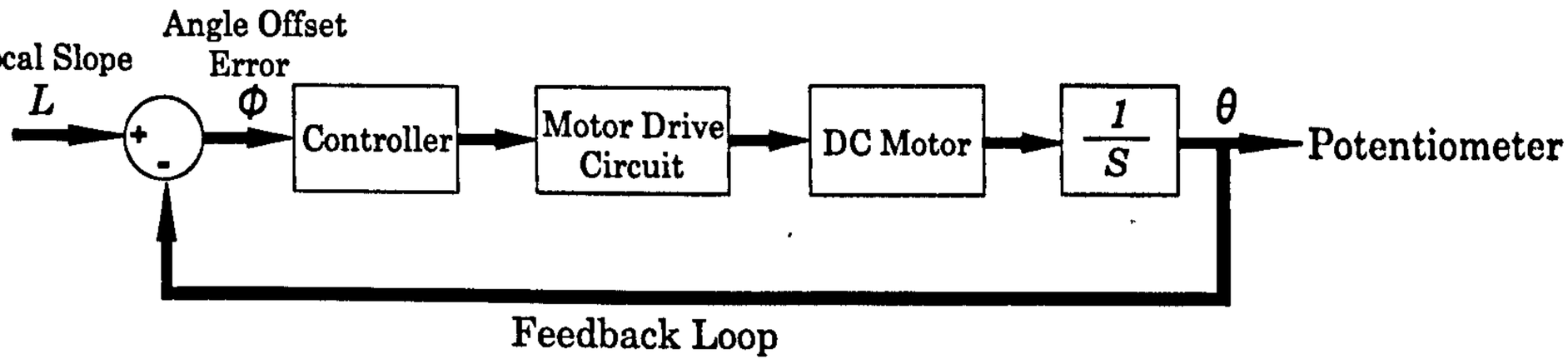


Figure 3.3: A control loop for rotational motion of the plate

In the control system, the presence of ϕ is determined by the angle offset error signal, V_ϕ. This is directly derived from the coil sensor voltages, V_A, V_B, V_C and V_D. The error signal is passed through a Proportional+Integral+Derivative (PID) controller and the top-motor demand voltage is generated. The amount of the plate rotation is measured by a potentiometer in the fixed frame. The error is defined as

$$V_\phi = (V_A + V_D) - (V_B + V_C) \tag{3.1}$$

therefore the V_ϕ is passed to the controller if

$$(V_A + V_D) - (V_B + V_C) \neq 0 \tag{3.2}$$

If there is no error signal then the top-motor stops rotating.

3.3.2 Lateral Offset Loops

The rotational motion of the plate is assumed to be very rapid. Therefore, the local tangent to the wire is assumed to be parallel with the plate Y-axis all the time. However, it is still possible that a lateral offset exists between the wire and the plate. Using two velocity feedback loops for X- and Y-linear motions, as shown in Figure 3.4, the lateral offset can be eliminated maintaining steady linear motion of the plate.

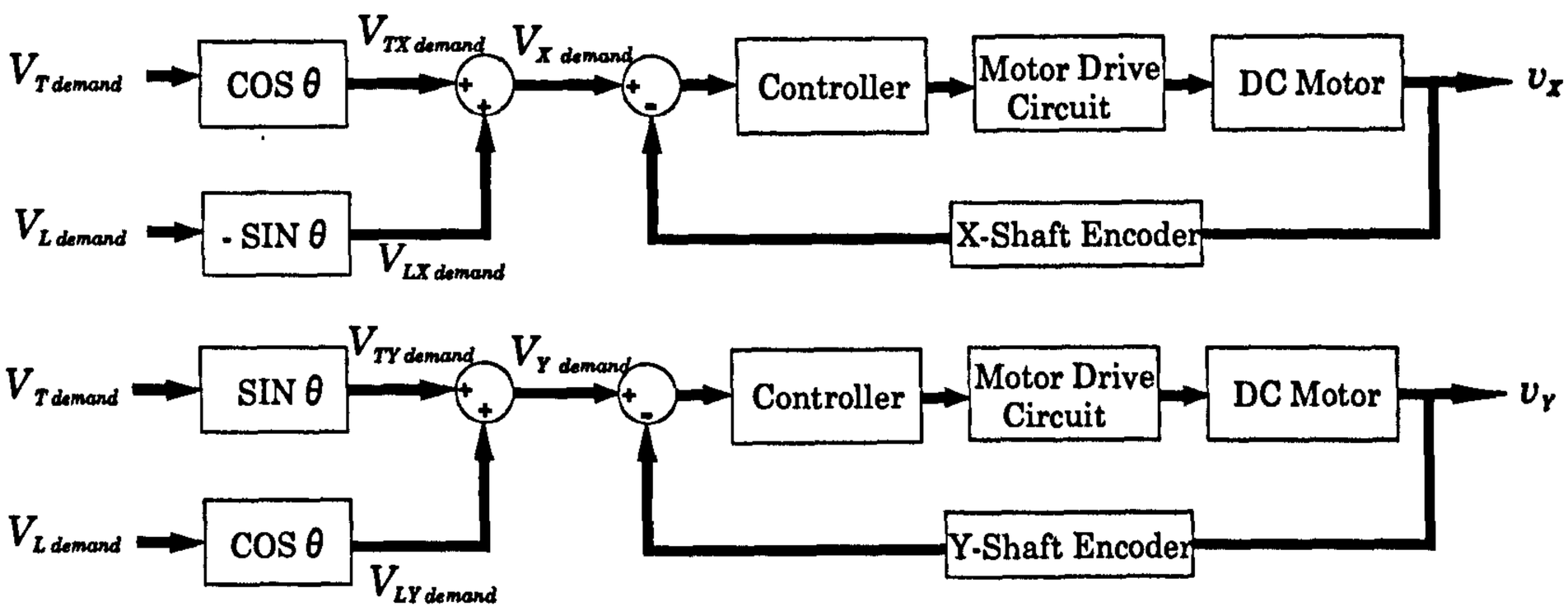


Figure 3.4: Control loops for linear motion of the plate

The lateral offset voltage signal, V_α, is determined from the coil sensors and is defined as

$$V_\alpha = (V_A + V_B) - (V_C + V_D) \tag{3.3}$$

therefore the V_α is passed to the controller if

$$(V_A + V_B) - (V_C + V_D) \neq 0 \tag{3.4}$$

There is another error term associated with the velocity loops: the tangential velocity error, $v_{T\,error}$. It provides the difference between the set point and the current speed of the plate.. To provide appropriate control signals to the X- and Y-motors, these error terms are translated into lateral ($V_{L\,demand}$) and tangential error signals ($V_{T\,demand}$) and then resolved for X- and Y-components. The velocities of the motors are measured by the shaft encoders so the error signals are constantly monitored and appropriate control signals can be applied during the

operation.

3.4 Control System Model

In this section, the Control System block as outlined in Figure 3.1 is described in detail. The block may be split into four parts:

- Offset error sensing model.
- Reverse potentiometer model.
- Tangential velocity error model.
- Controller models.
- Transform model.

Figure 3.5 shows how the Control System block is structured.

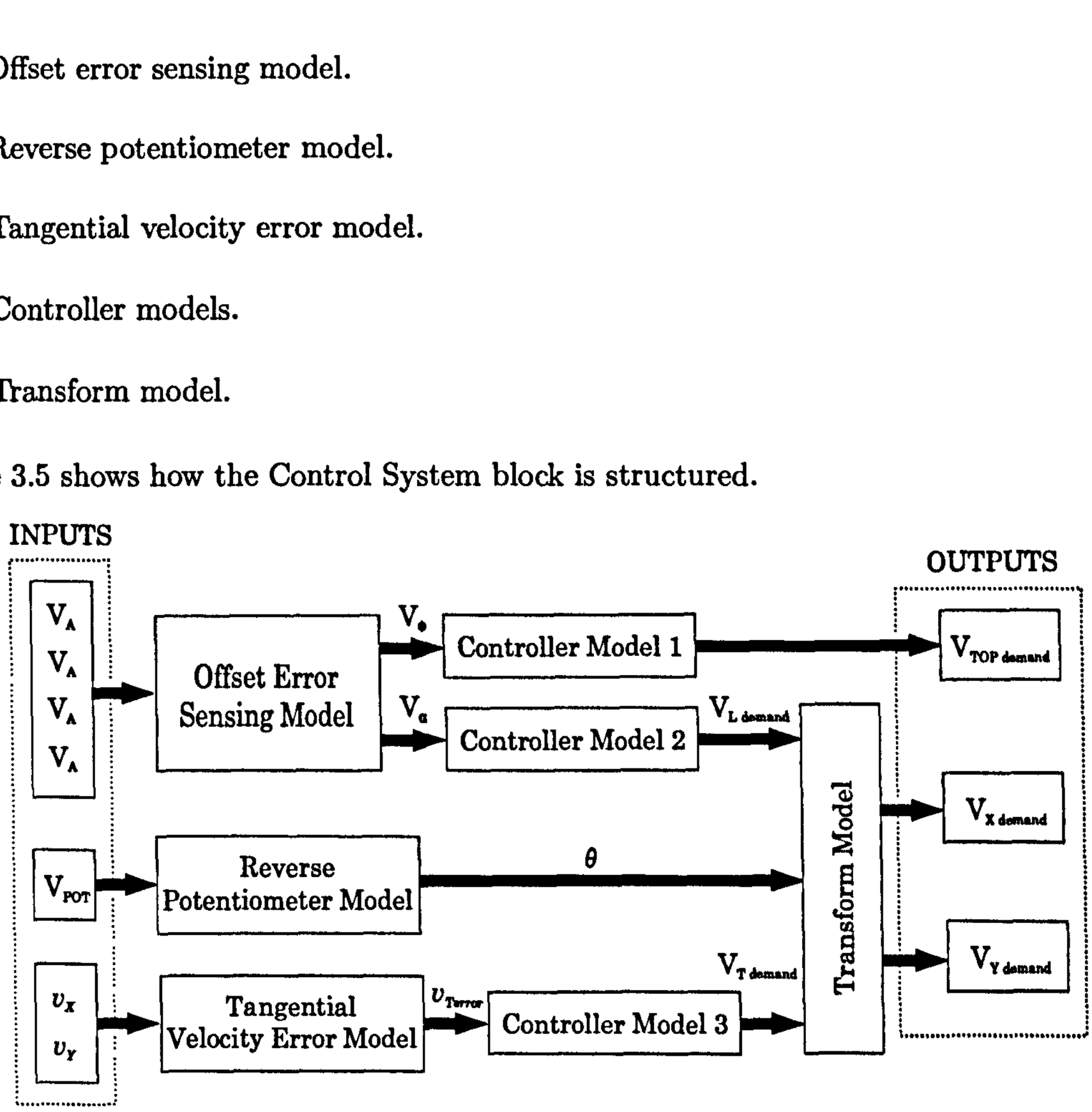


Figure 3.5: The control system block of the Shaky Hand model

The offset error sensing model defines how the error signals V_ϕ and V_α are produced from the four sensor outputs on the plate. The outputs from this model are passed to the controller models to produce appropriate demand signals. The reverse potentiometer model converts the potentiometer output voltage into θ. The tangential velocity model provides the error signal associated with the plate speed. The transform model resolves the lateral and the tangential voltage demands into X- and Y-components to produce X- and Y-motor voltage demands respectively.

3.4.1 Offset Error Sensing Solutions

As described in the previous section, the offset error signals are V_α and V_ϕ and are determined by the four coil sensors A, B, C and D on the plate. V_α is the lateral offset error signal and is defined by:

$$V_\alpha = (V_A + V_B) - (V_C + V_D) \tag{3.5}$$

V_ϕ is the angle offset error signal and is defined by:

$$V_\phi = (V_A + V_D) - (V_B + V_C) \tag{3.6}$$

When the plate is laterally and rotationally misaligned, the appropriate offset error signals are produced through Equations 3.5 and 3.6. The behaviour of the plate can be predicted from the offset error signals. For example, in Figure 3.6, if V_α is positive, the wire track is closer to sensors A and B than sensors C and D. If V_ϕ is negative, the wire track is closer to sensors B and C than sensors A and D.

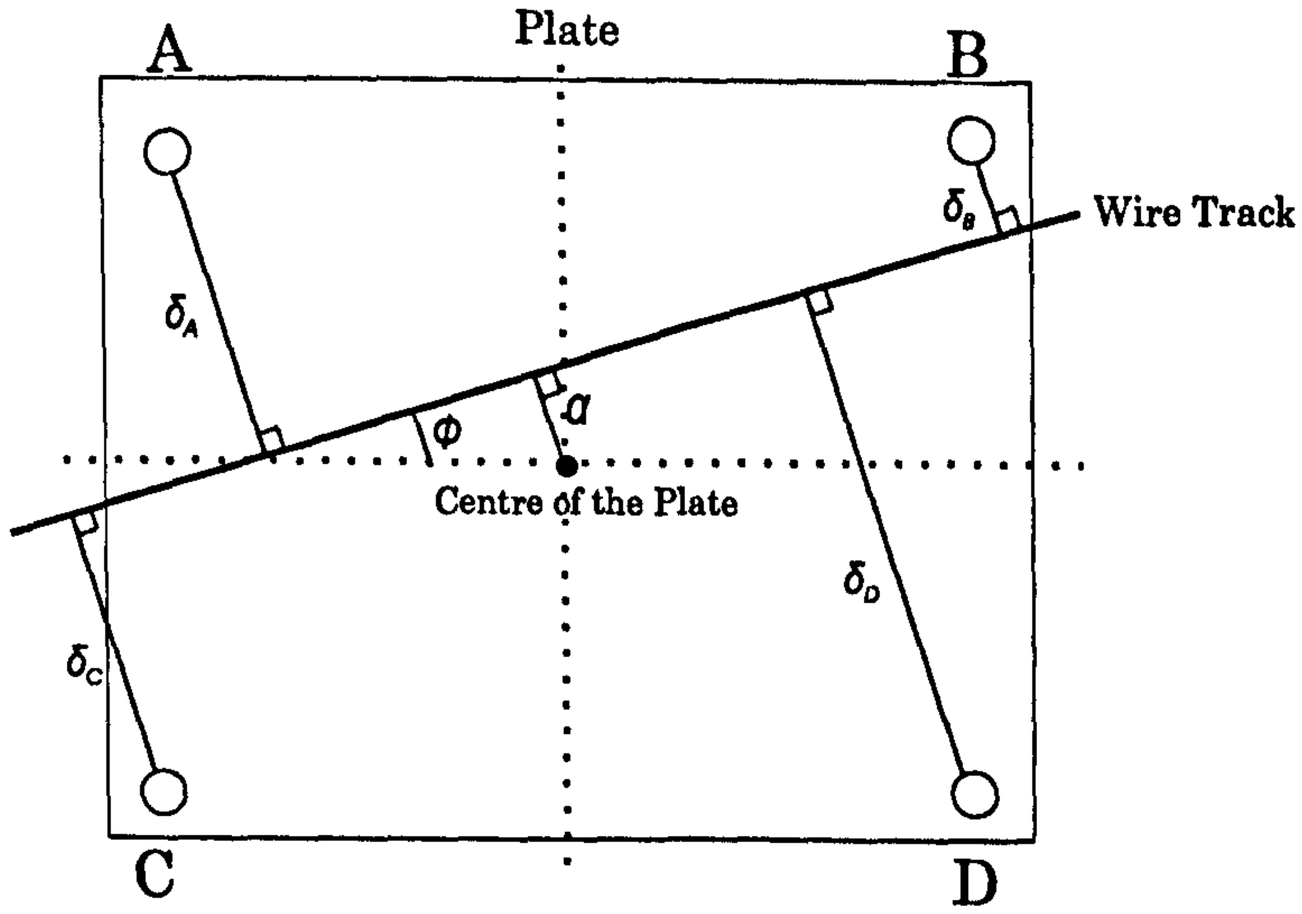


Figure 3.6: Centralised solutions: plate behaviour

In the situation as shown in Figure 3.6, the plate would need to rotate anti-clockwise to align itself with the wire track and then move towards the wire track to diminish the lateral offset. The basic behaviour can be summarised in Table 3.1. The controller is required to produce

appropriate amounts of voltage demand with the correct polarities for precise control.

Condition	Next expected plate movement
(V_A+V_D)-$(V_B+V_C) > 0$	Rotation is clockwise
(V_A+V_D)-$(V_B+V_C) < 0$	Rotation is anti-clockwise
(V_A+V_B)-$(V_C+V_D) > 0$	Movement towards sensors A and B
(V_A+V_B)-$(V_C+V_D) < 0$	Movement towards sensors C and D

Table 3.1: Expected general behaviour from the plate when offset errors are sensed

3.4.2 Reverse Potentiometer Model

The input to this block is the potentiometer output signal. The signal is equivalent to the plate angle, θ, relative to the base frame. The signal is converted into θ by:

$$\theta = \frac{V_{pot} - 2.5}{2.5} * 6\pi \tag{3.7}$$

This is same as Equation 2.87 but inversely applied.

3.4.3 Velocity Error Model

One of the rules of the game of Shaky Hand is that the tangential velocity of the plate along the wire track, v_T, is kept constant during the operation. The velocities in X- and Y-directions, v_x and v_y, are obtained from the encoders in the hardware and they are processed such that:

$$v_T = \sqrt{(v_x^2 + v_y^2)} \tag{3.8}$$

The tangential velocity is compared with the set-point value, v_S, to produce the error signal, $v_{T\,error}$.

$$v_{T\,error} = v_S - v_T \tag{3.9}$$

This is fed into the appropriate controller model and the tangential voltage demand, $V_{T\,demand}$, is produced.

3.4.4 Controller Models

In all three controller models, PID algorithms as shown in Figure 3.7 are used. A PID controller takes in an error signal to provide appropriate output. For the rotational loop, V_ϕ is passed to the rotational PID controller to produce the top-motor voltage demand. For the X- and Y-motor velocity loops, V_α and $v_{T\,error}$ are used as the inputs to the PID controllers to produce the lateral and tangential voltage demands respectively.

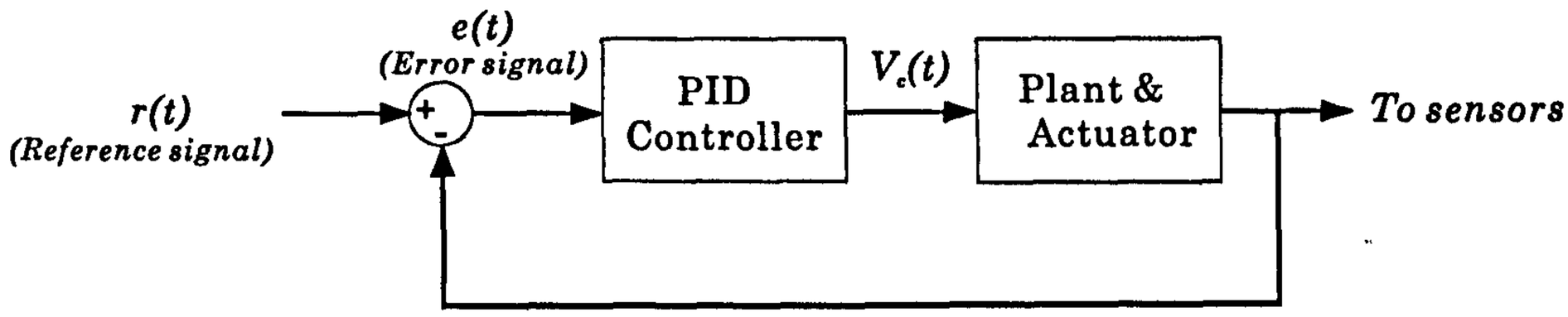


Figure 3.7: A PID controller in a loop

A PID controller applies three combined control actions to an error signal. Proportional action scales the sampled error signal. Integral action ensures zero steady state error by increasing the system type. Derivative action improves transient response (shorter rise time, faster settling time, and good overshoot performance - better damping). The integro-differential equation representing a PID controller is given by

$$V_c(t) = K_P\, e(t) \;+\; K_I \int_0^t e(t)\; d(t) + K_D \frac{de(t)}{dt} \tag{3.10}$$

where,

$V_c(t)$ = PID controller output signal at time t

$e(t)$ = Error signal at time t (Input to the PID controller)

K_P = Proportional control gain

K_I = Integral control gain

K_D = Derivative control gain

In practice, the control system for the Shaky Hand is implemented within the I/O software, so discrete digital PID controllers are used. Assuming a sampling period of T seconds, at n^{th} sampling time the PID control algorithm can be represented as

$$V_c(nT) = K_P\, e(nT) \;+\; K_I \sum_{i=0}^{n} \frac{e(iT) + e([i-1]T)}{2}\, T + K_D \frac{e(nT) - e([n-1]T)}{T} \tag{3.11}$$

In Equation 3.11, the integral term uses a trapezoidal approximation and the derivative term uses a backwards difference approximation. The representation is further simplified to

$$V_c(nT) = K_P\, e(nT) + I([n-1]T) + \frac{K_I\, T}{2}\{e(nT) + e([n-1]T)\} + \frac{K_D}{T}\{e(nT) - e([n-1]T)\} \tag{3.12}$$

where $I([n-1]T)$ is the value of the integral term at the previous sampling time.

A simple algorithm for a digital PID controller can be described as:

1. Sample signal e[nT] and store.
2. Calculate integral component and add I([n-1]T).
3. Calculate derivative component.
4. Calculate controller output.
5. Set e[nT] to e[(n-1)T].
6. Set I(nT) (current integral component) to I([n-1]T).
7. n=n+1, return to step 1 and loop.

The specifications for the PID controllers in the control system are as follows.

- Fast rotational loop: V_ϕ should be quickly eliminated. This loop should be faster than others.
- Zero steady state error in plate-track alignment and in plate speed.
- Minimal overshoot in plate position and in the desired speed while responding to track direction changes.

The PID parameters are tuned to fulfill the specifications and to provide the stable operation of the Shaky Hand throughout the run. The proportional gain is added initially, and the integral gain is added if a steady state error exist, and the derivative gain is added to improve the transient response. These values are tuned to provide the best response. While tuning, the settling time, the steady state error and the overshoot were observed from the model. The tuned PID controllers provide a rotational loop which is approximately twice as fast as the

translational loops. The zero steady state error in plate-track alignment and in plate speed is achieved. The overshoot is kept extremely small. The error plots of the simulation are shown in Section 3.5. The PID controllers have been implemented in Simulink. Table 3.2 presents the parameters of the controllers used in the Simulink model. In practice, the parameters shown here may need to be modified because noise and frictions in the actual hardware are not represented by mathematical models. Noise and frictions can not be measured and they are also likely to change due to wear of the physical system or due to the change of the environment. Using the ideal environment where the system works perfectly, with no noise or frictions on the system, then the fault tolerant solutions can be tested and compared with each other fairly. Frictional effects can readily be added to the system by changing the motor model parameters and also external noise can be incorporated by implementing a summing amplifier in any part of the hardware model if required.

PID Controller	K_P	K_I	K_D
Controller 1: For V_ϕ Offset Error Signal	10	0.05	1
Controller 2: For V_α Offset Error Signal	20	0.5	4
Controller 3: For $v_{T\,error}$ Offset Error Signal	80	0	0

Table 3.2: PID controller gain values

3.4.5 Transform Model

As shown in Section 3.2 Figure 3.2, two reference frames are used to describe the Shaky Hand operation. One is the plate frame which defines the wire track position relative to the plate. The other is the base frame reference which relates the plate position to the base frame. Lateral and tangential velocities appear in the plate frame and are related to the velocities in the base frame, v_X and v_Y by transformation. It is assumed that the rotational dynamics of the plate are very fast compared with the linear dynamics. Therefore, the angular offset error, ϕ, quickly becomes zero, and thus, the local tangent to the wire curvature is assumed parallel to the plate Y-axis at all times. This gives the angle orientation of the plate in the base frame reference.

The motions in the plate frame are defined by the velocities v_L and v_T. v_T is the tangential velocity along the wire driven by the tangential voltage V_T. This velocity is always kept constant. v_L is the lateral velocity, perpendicular to v_T and is driven by the lateral voltage V_L. v_L appears only when the plate deviates laterally with respect to the wire and brings the plate back into correct alignment. Defining θ as the angle of rotation measured relative

to the fixed frame, the conversion process can be described as follows.

The lateral voltage demand consists of x and y components.

$$V_{LX} = -V_L \, sin\theta \tag{3.13}$$

$$V_{LY} = V_L \, cos\theta \tag{3.14}$$

Similarly, the tangential voltage demand can be resolved into x and y components as well.

$$V_{TX} = V_T \, cos\theta \tag{3.15}$$

$$V_{TY} = V_T \, sin\theta \tag{3.16}$$

Therefore, the total voltage demands for X- and Y-motors are:

$$V_X = V_{LX} + V_{TX} \tag{3.17}$$

$$V_Y = V_{LY} + V_{TY} \tag{3.18}$$

From the transform model, the X- and Y-motor demands are obtained.

3.5 Simulink Simulation

The models developed in Chapter 2 and in this chapter are brought together in Simulink to simulate the operation of the Shaky Hand. In simulation, the wire track is represented by the wire model described in Section 2.5.1. The simulation generates a track of the centre of the plate during the run. Thus, we can see whether the plate position corresponds to the wire track. The game rule states that the sensors should never touch the wire track. Therefore the maximum distance between the centre of the plate and the sensor is 22.5mm, half the separation distance between the sensors. The maximum angle of rotation is approximately ± 40 degrees, limited by the presence of the coil sensors.

The initial condition of the plate is set so that the plate sits at coordinate (0,0) position and has angle offset error, ϕ, of +3.4 degrees. The results are displayed in Figure 3.8. It shows the whole base frame with the wire track drawn on it. The solid line represents the trace of the centre of the plate and the dashed line, masked by it, represents the wire track. The base frame has a unit of mm to mimic the Shaky Hand in practice. The graph shows that the centre of the plate overlaps the wire track until the end of the operation.

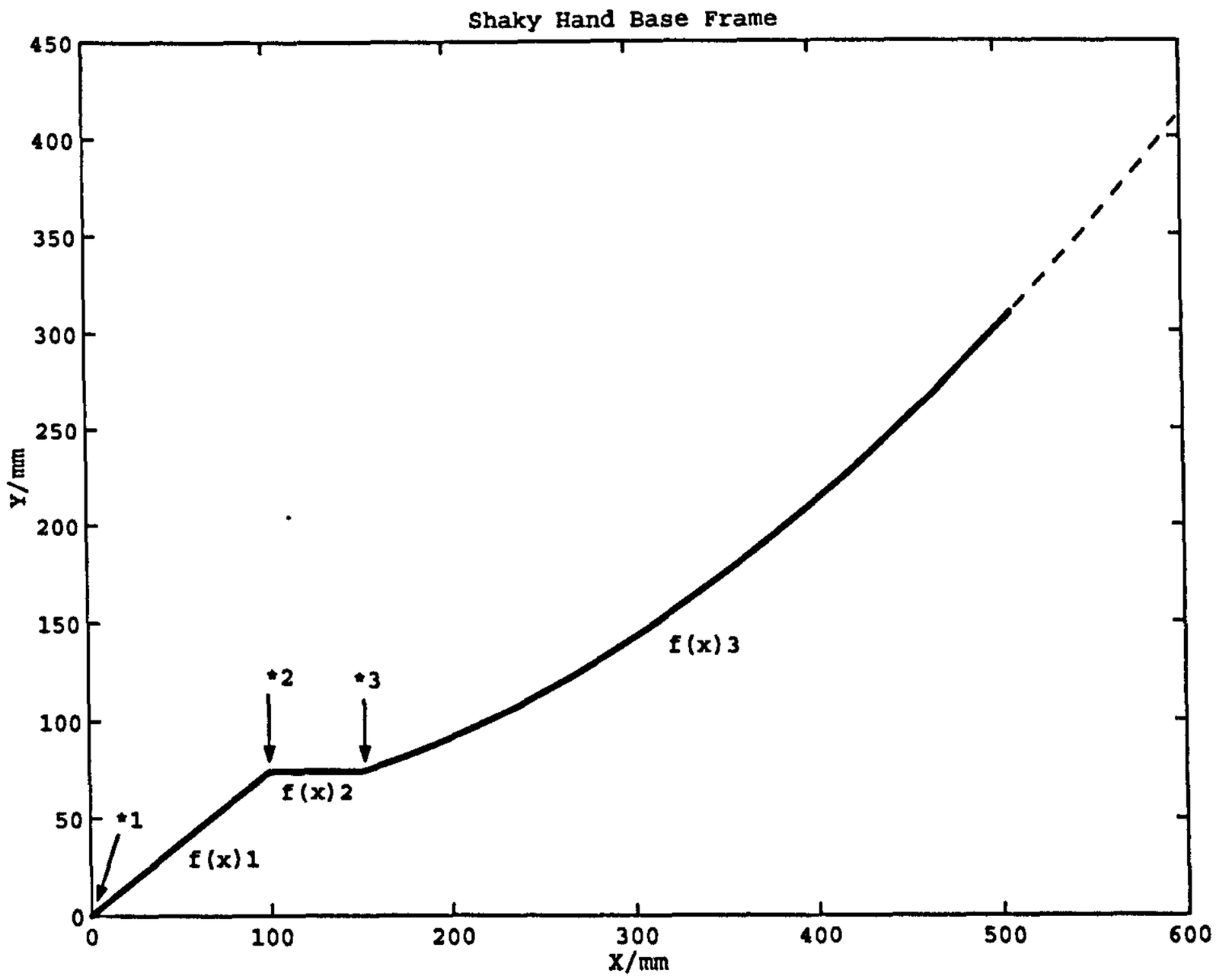


Figure 3.8: Simulink simulation results

Positions marked as *1, *2 and *3 in Figure 3.8 are enlarged and displayed in Figures 3.9, 3.10 and 3.11 respectively, to show what happens when the wire track function changes.

Position *1 (Figure 3.9) looks at the start of the simulation. The plate initially has an angular offset of +3.4 degrees, but the plate aligns with the wire track as soon as the operation starts. The plate follows the track very well from the start and deviations cannot be observed. When the plate comes to point *2 (Figure 3.10), where there is a transition in the wire track functions and a change of wire track directions occurs, a small overshoot, approximately 0.037mm, is observed. The overshoot decays quickly and is small enough to avoid physical collision between the wire track and the sensors on the plate since the maximum separation is 22.5mm. At point *3 (Figure 3.10), the plate seems to cope with the change in the wire track functions smoothly and the overshoot is small. The offset errors, ϕ and α, and the speed of the plate along the wire track are recorded through out the run, and are shown in Figures 3.12, 3.13 and 3.14 respectively.

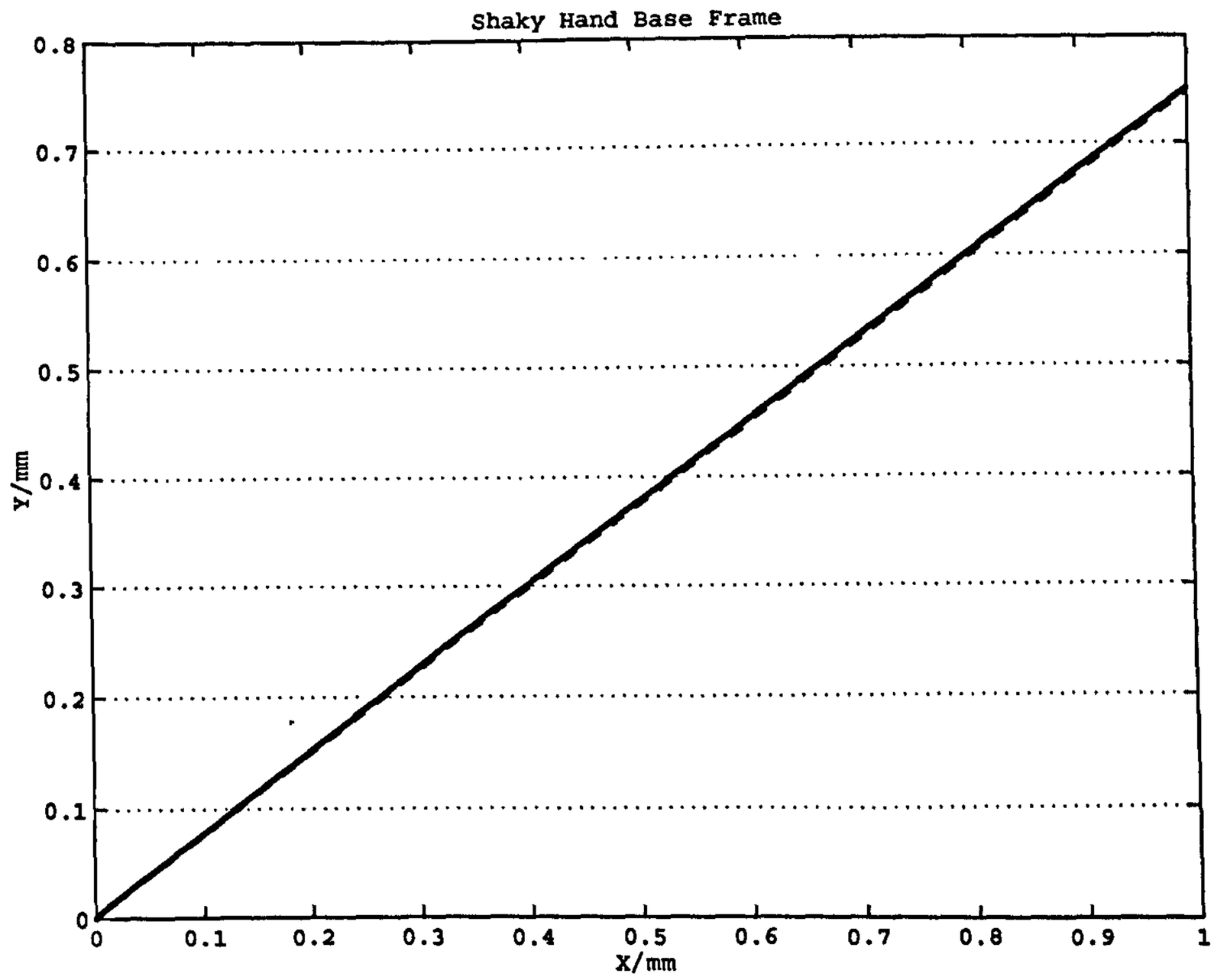


Figure 3.9: Position *1 of Figure 3.8

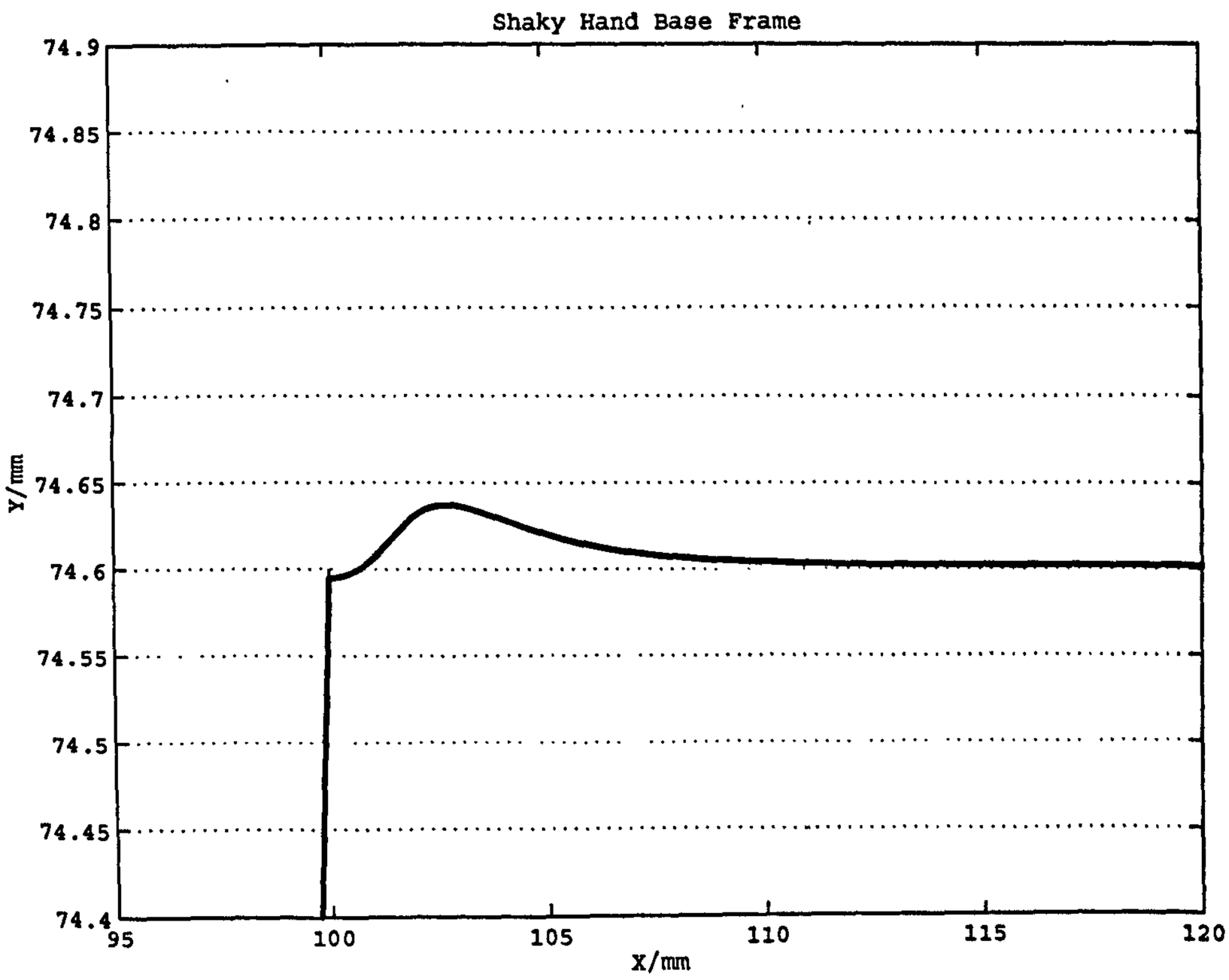


Figure 3.10: Position *2 of Figure 3.8

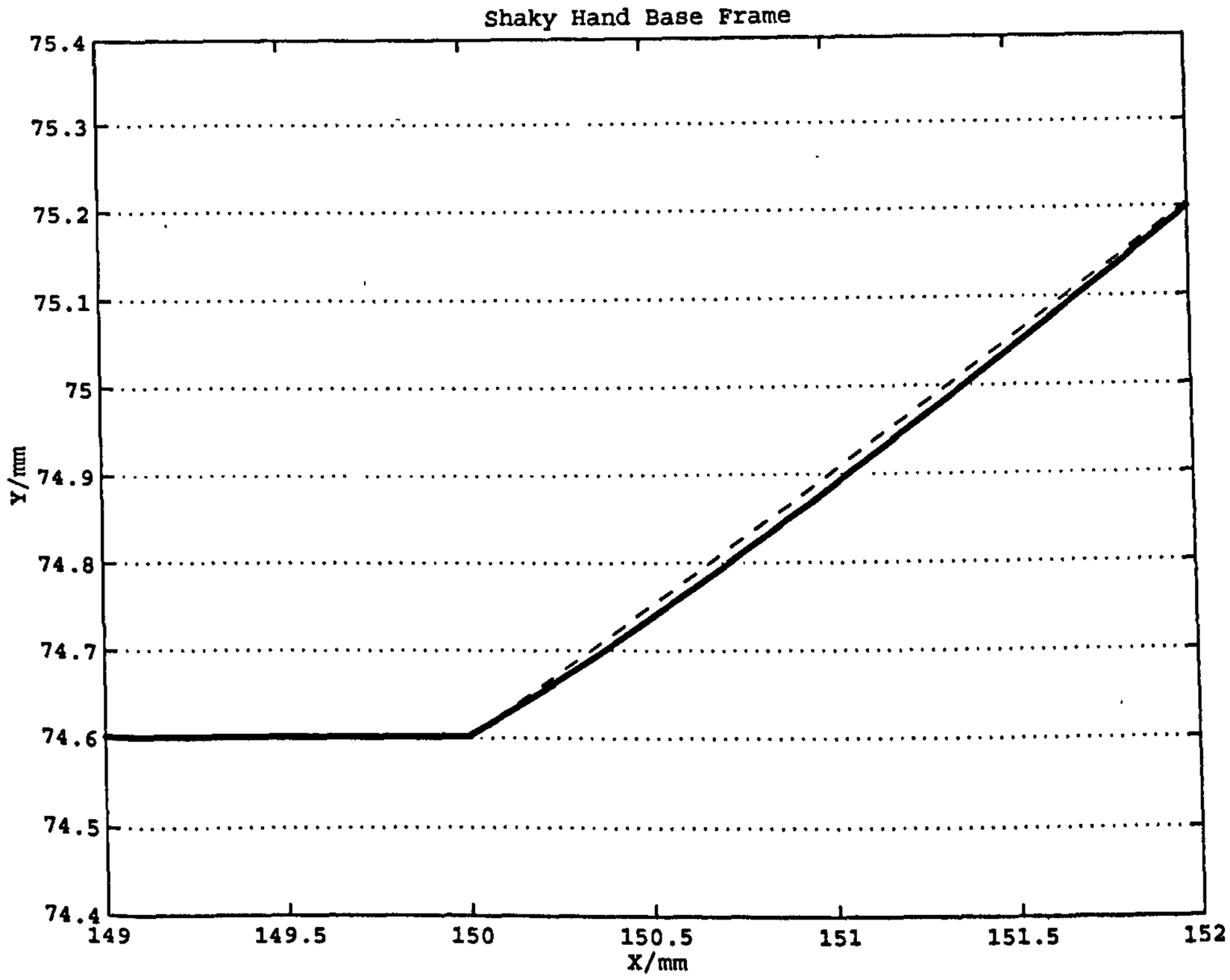


Figure 3.11: Position *3 of Figure 3.8

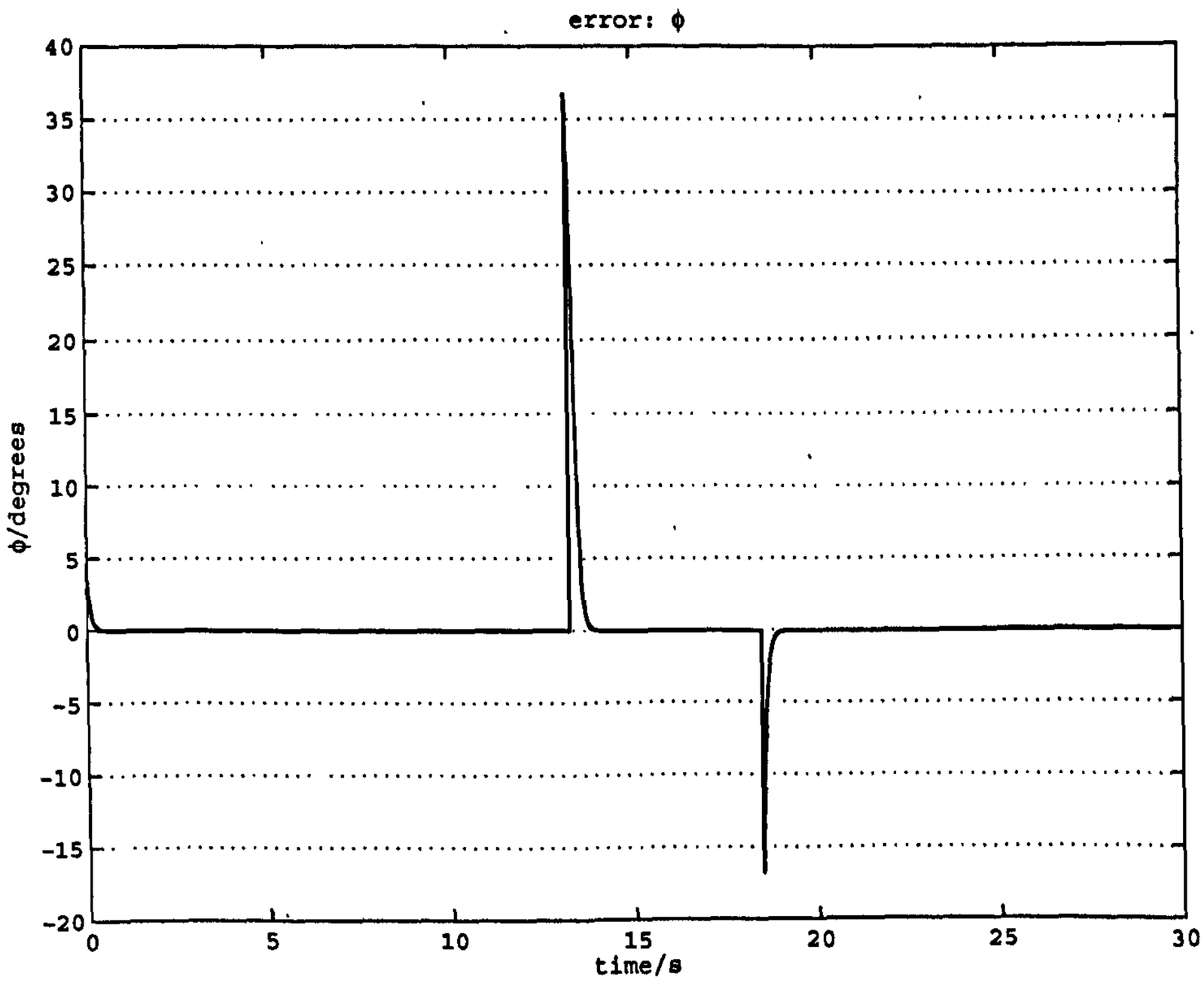


Figure 3.12: The angle of offset, ϕ, in the simulation

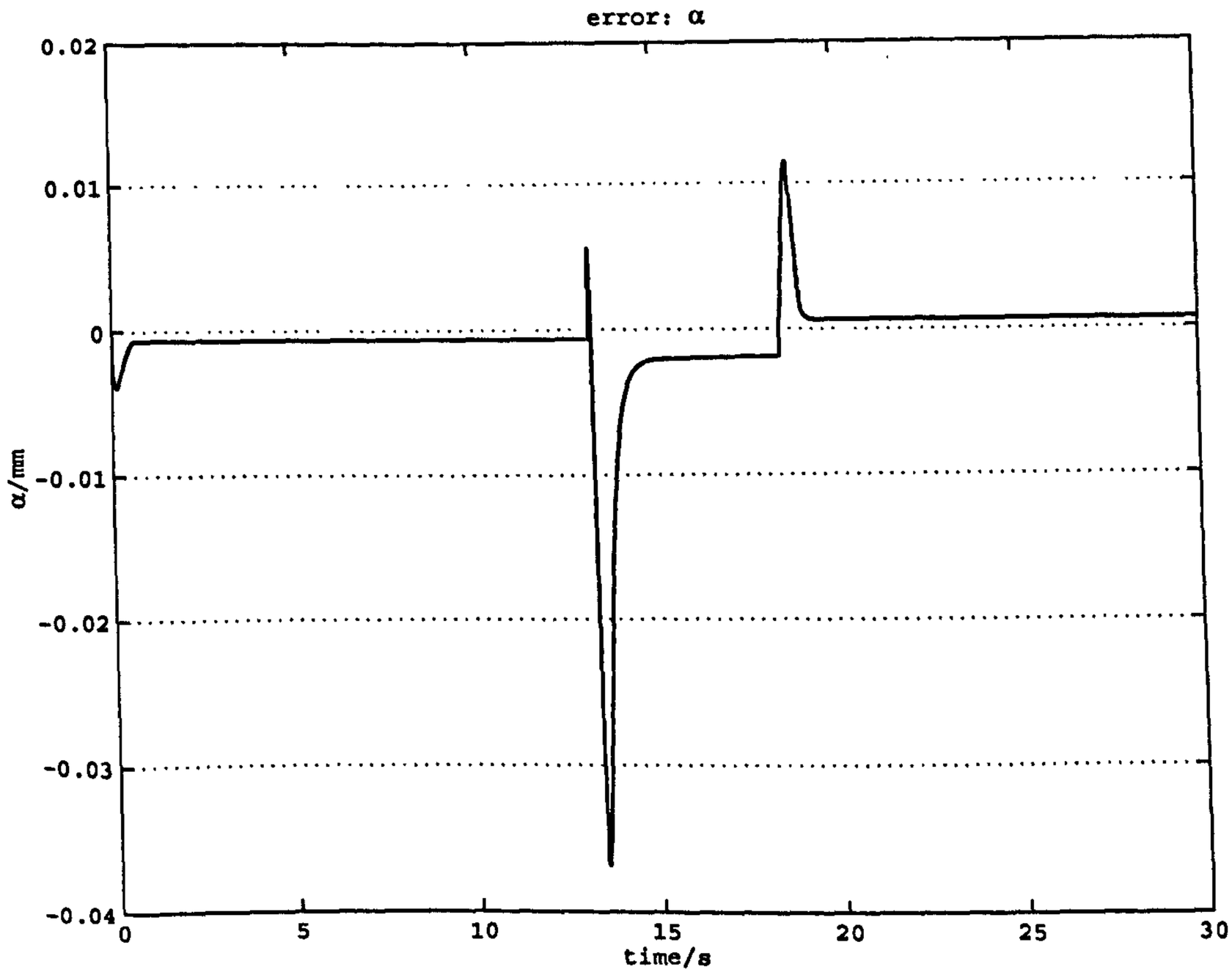


Figure 3.13: The lateral offset, α, in the simulation

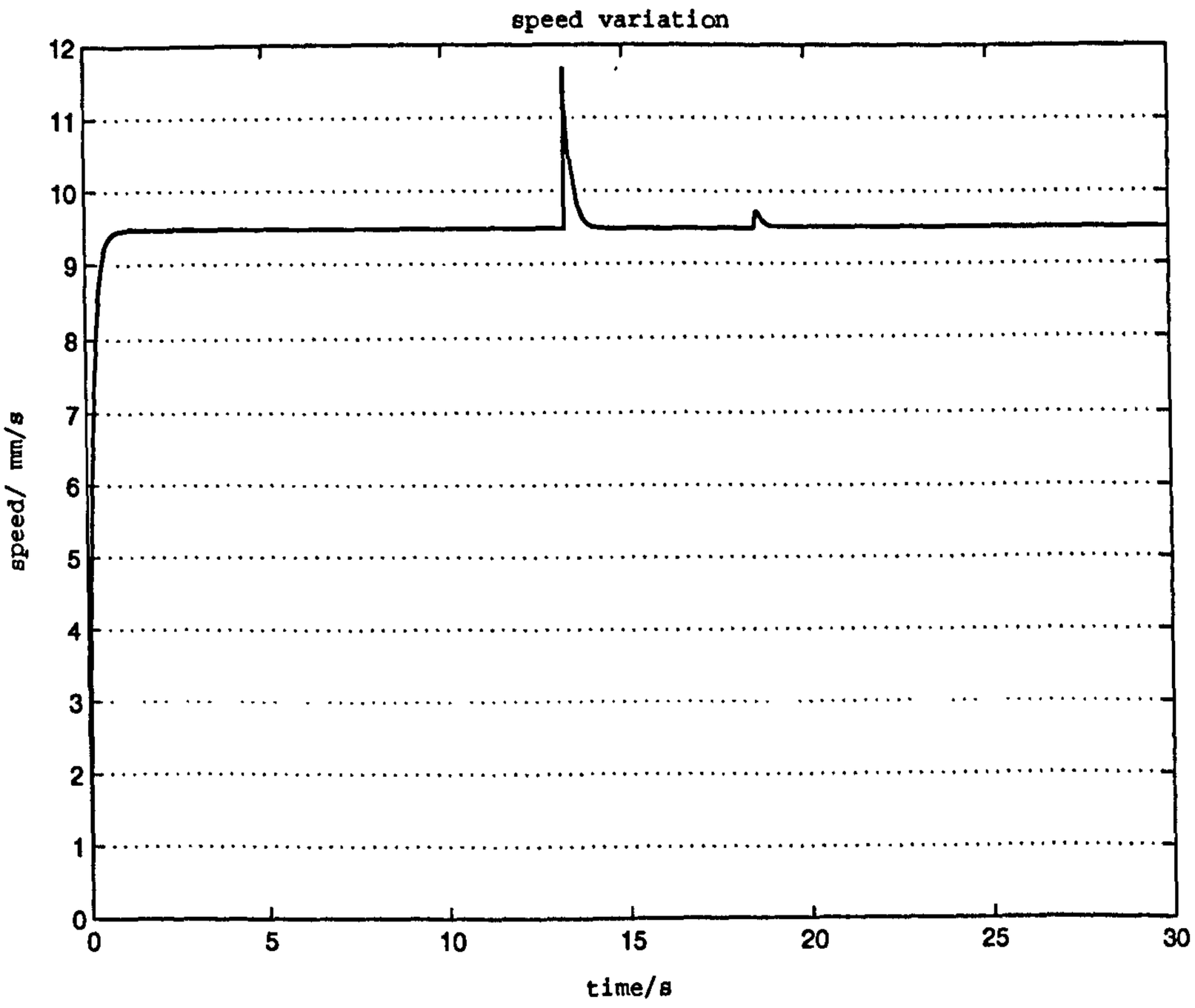


Figure 3.14: Speed of the plate in the simulation

The ϕ variation as shown in Figure 3.12 shows that the initial offset decreases quickly and ϕ is nulled until the wire function is changed at around 13.3 seconds. At this point, ϕ rises rapidly to 36.7 degrees which is within the maximum limit. This offset is quickly eliminated within 1.1 seconds. At around 18.6 seconds, the wire function is, again, changed and a ϕ of -16.7 degrees is detected. This is quickly eliminated within 0.8 seconds.

Figure 3.13 shows the α variation. When the X- and Y-motors start running, a small amount of α (0.0039mm) arises and then disappears. The scale of the α variation is very small such that small deviations around zero can be considered to be negligible and do not affect the total performance of the Shaky Hand. When the wire function changes at around 13.3 seconds, α is increased up to -0.037mm, as can be seen from both Figures 3.10 and 3.13. The α offset quickly disappears within 2 seconds. This α offset settles to 0.002mm which is negligible. When the wire function changes again, a small α (0.011mm) appears but quickly settles down to 0.0006mm, which is negligible. Figure 3.14 shows how the speed of the plate changes during the run. Initially, the plate accelerates within 0.8 second and settles down to 9.5mm/s, which is the set point defined by the author. When the wire function changes, the plate accelerates rapidly to compensate for the α offset. The error plots coincide with the simulation results in Figures from 3.8 to 3.11. From the simulation and the error plots, it can be seen that the Shaky Hand completes the operation without failure.

3.6 Introducing I/O Programming

In practice, analogue sensor signals from the Shaky Hand Hardware, as described in Chapter 2, are passed to a conventional PC for data acquisition and control via an I/O interface card. The control signals are then fed back to the hardware, as described in Section 3.4. In this section, the I/O software used for the data acquisition and control is described; a software overview is shown first followed by program details.

Two standard I/O interface cards are installed in a conventional PC (Windows XP, Pentium 4 CPU 2.8GHz, 512MB RAM). Keithley's KPCI-3100 series data acquisition boards [6] are used for this purpose. The boards come with DriverLINX ([6], [3], [5], [4], [2]), which comprises software drivers for real-time data acquisition under the Microsoft Windows XP environment. This is used as the basic Application Programming Interface (API) for the KPCI-3100 series boards. The control system is embedded into the I/O software using Microsoft Visual C++ [1].

3.6.1 I/O software

The overall control process through the I/O boards is shown in Figure 3.15.

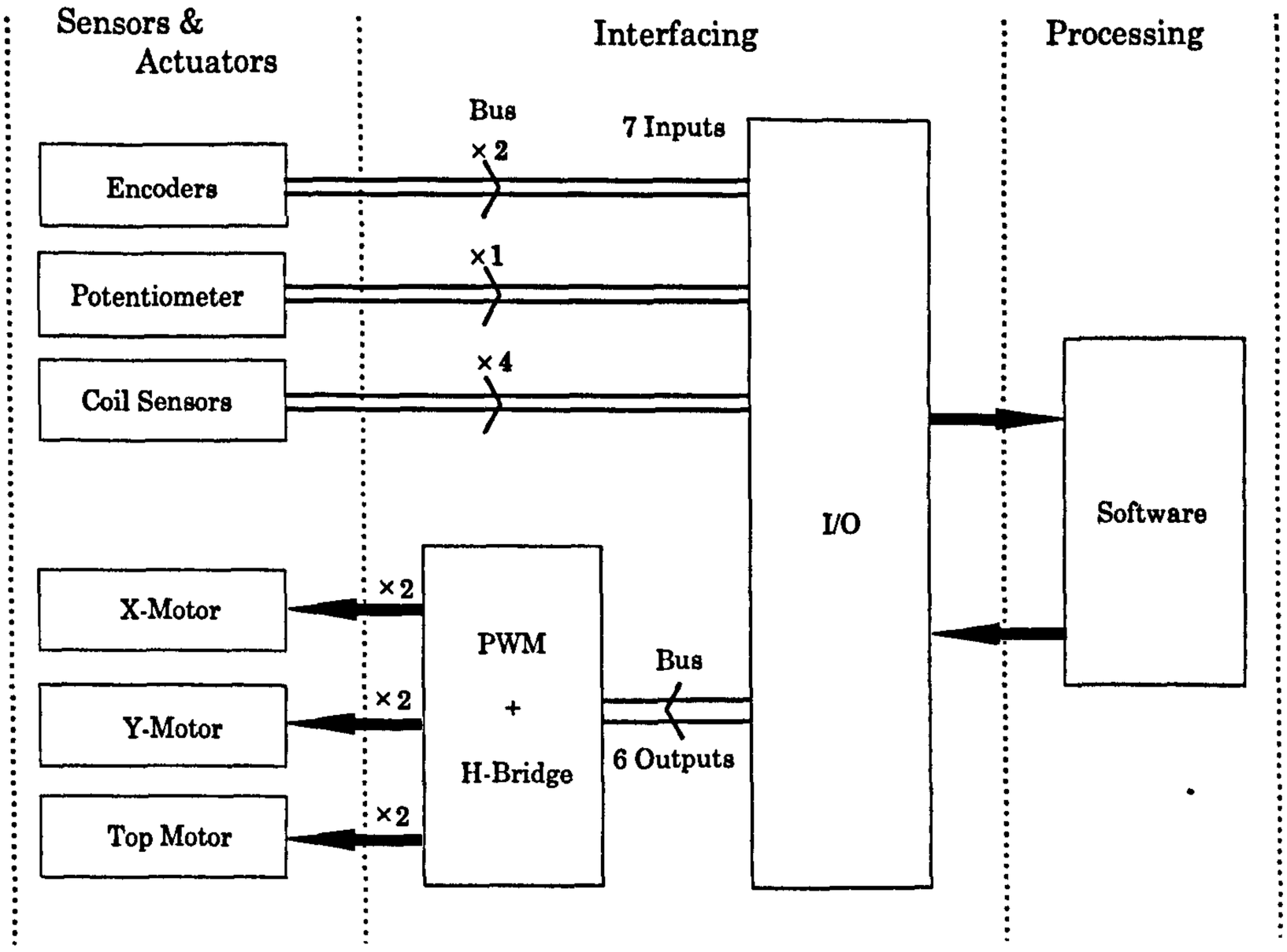


Figure 3.15: Block diagram of the control process through I/O Boards

The inputs to the I/O boards are the seven analogue signals comprising:

- Four coil sensor signals.
- Two optical encoder signals.
- One potentiometer signal.

The I/O boards act as buffers whose stored values can be read and overwritten by the I/O software. These signals are processed and appropriate motor control signals are returned. The outputs from the I/O boards are:

- Three motor demand analogue voltages for the X-, Y-, and top-motors.
- Three direction control digital signals for all three motors.

Each I/O board has only two analogue outputs therefore a second I/O board was required to provide the third analogue output. The I/O board pin assignment is shown in Appendix B.3. Let us look at the detail of the I/O software program. The flow of the program can be seen from Figure 3.16. Terms used in the figure are described in Table 3.3.

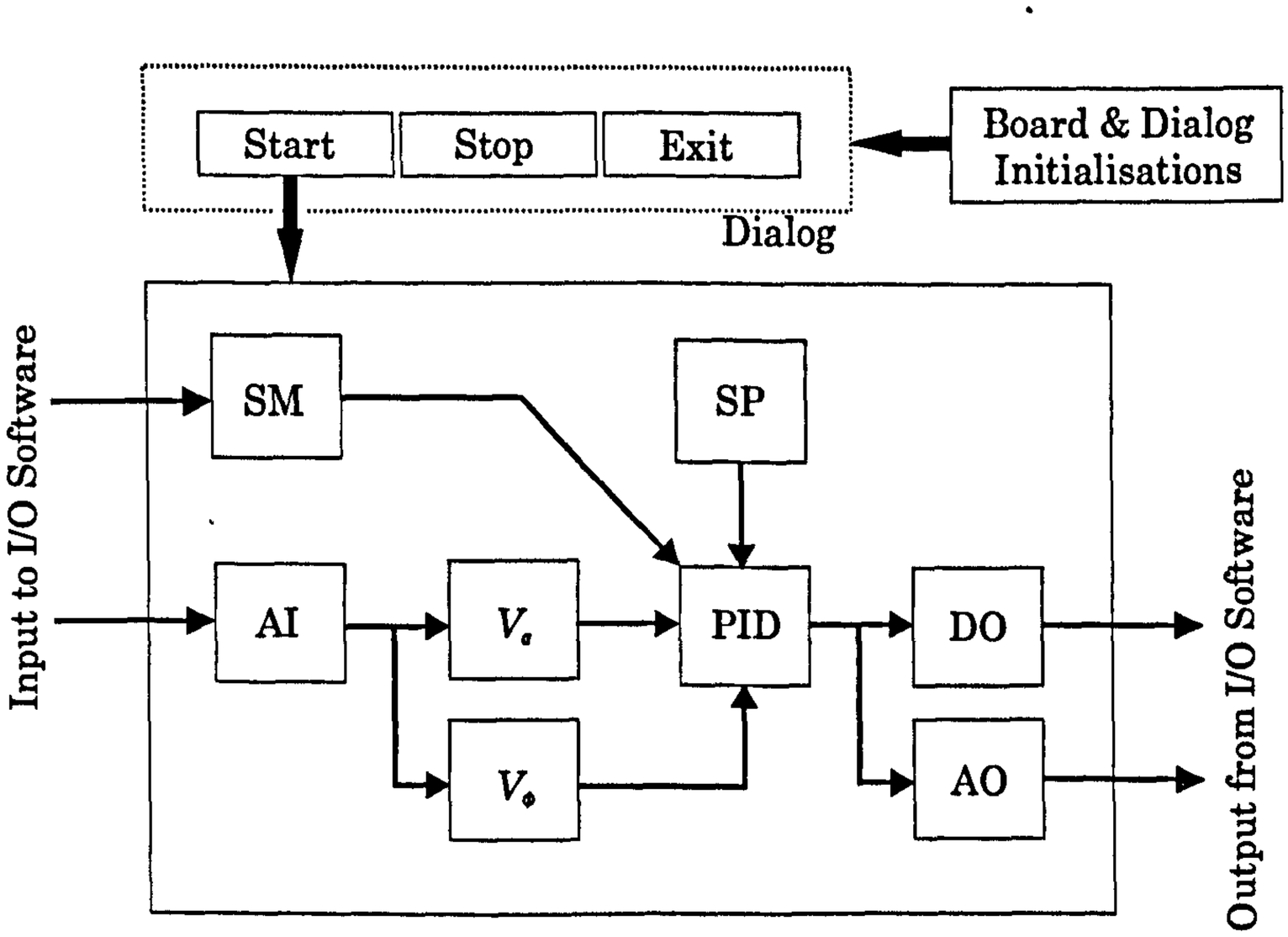


Figure 3.16: Program flow chart

The program is managed by a dialog box. It begins with the initialisation of the board and the opening of the dialog based interface which contains three function buttons for operating

Program Type	Description
Speed Measurement (SM)	This acquires data from the X- and Y-shaft encoders at a user-defined interval.
Analogue Input (AI)	This acquires data from the coil sensors and the potentiometer by selecting the input channels of the board at a user-defined interval.
V_α	This uses data from AI to produce the lateral offset V_α as described in Section 3.4.1.
V_ϕ	This uses data from AI to produce the angle offset V_ϕ as described in Section 3.4.1.
Set Point (SP)	This defines the set point values for the PID controllers. For lateral and angle offset errors, the set point is zero. Set point speed is defined by a user.
PID controllers (PID)	This takes in values from SM, V_α, V_ϕ and SP to produce corresponding motor control signals as described in Section 3.4.4.
Digital Outputs (DO)	This gives out the direction control signals evaluated by the program to appropriate channels of the board.
Analogue Outputs (AO)	This transfers the evaluated motor drive signals to the output channels of the board.

Table 3.3: I/O program functions

the program, *start*, *stop* and *exit*. The interface also displays the input and output values to/from the program. When the start button is pressed, the program begins accepting sensor values and evaluates the outputs at a user-defined sampling interval. The stop button halts the program and the exit button closes it.

Functions within the I/O software may be split into the following parts, as described in Table 3.3. AI and SM programs take in the signals from the seven sensors on the Shaky Hand. These signals are processed as described in Section 3.4. Set points for the PID controllers are defined explicitly by the user. The processed signals are transferred to appropriate ports of the I/O boards by AO and DO programs. The functions are now described in more detail.

3.6.2 Speed Measurement Program

Speed measurement of the X- and Y-motors are achieved by using 16-bit general purpose counters incorporated in the I/O board. X- and Y-shaft encoder outputs are connected to a *CLOCK* input of the counter as shown in Figure 3.17. The gating input of the counter is connected to ground. In this configuration, the counter measures the frequency of the signal at clock input by counting the number of edges received at clock input of the channel for the duration of the Windows timer (1ms) using Equation 3.19 [6]. Measured frequencies are used

to control the speed of the plate.

$$Frequency\ Measurement = \frac{Number\ of\ Events}{Duration\ of\ the\ Windows\ Timer} \tag{3.19}$$

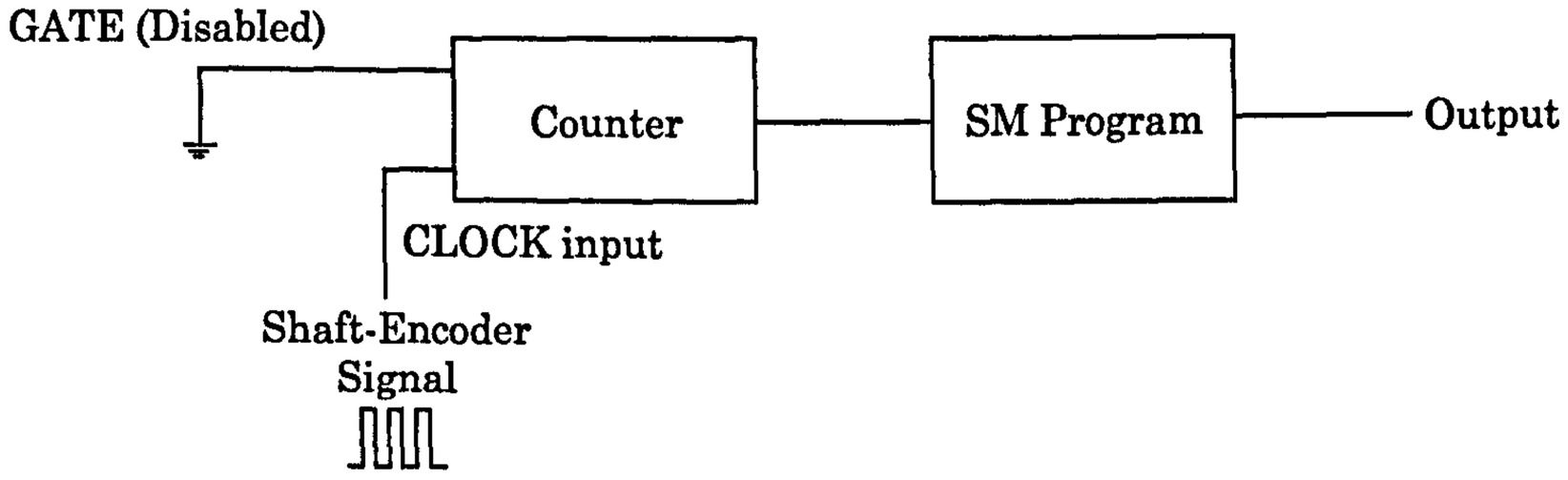


Figure 3.17: A principle of speed measurement

3.6.3 Analogue Input Program

There are several analogue input channels on the I/O boards. AI program specifies which channels to read at user defined intervals. A buffer is used to store a series of different types of data at once, paced with a rate generator which operates using a fixed rate clock. Values from the coil sensors, shaft-encoders and the potentiometer are stored in an array and held by the buffer. The stored values are used to analyse the plate positions.

3.6.4 Control System Program

The plate offset errors (V_ϕ and V_α) are calculated based on the coil sensor values (Section 3.4.1). The velocity error, $v_{T\,error}$ is obtained from the encoder values (Section 3.4.3) These error signals are passed to PID controllers (Section 3.4.4) to provide the top motor voltage, the lateral voltage and the tangential voltage demands. We assume the fast rotational loop so that the latter two voltage demands can be resolved into X- and Y-components using θ obtained from a potentiometer value (Section 3.4.5). Then the X- and Y-motor demands are calculated with appropriate direction signals. Psedocode for the output evaluation process is now described.

Psedocode for the Control System Program

1: Calculate V_ϕ

2: Calculate V_α

3: Calculate $v_{T\,error}$

4: Pass V_ϕ to the first PID controller to evaluate the top motor voltage demand and also the direction for the top-motor.

5: Pass V_α to the second PID controller to evaluate the lateral voltage demand and also the direction for the Y-motor.

6: Pass $v_{T\,error}$ to the third PID controller to evaluate the tangential voltage demand.

7: Calculate θ

8: Resolve the lateral and tangential voltage demands into the X- and Y-components. Calculate the X- and Y-motor voltage demands.

3.6.5 Digital Output Program

The I/O board has configurable digital I/O ports which can be used for either input or output. An array is created containing the data to be sent out. Three ports are used for sending out the direction signals for the X-, Y- and top-motors (represented by *dx*, *dy* and *dtop* respectively). The binary values, which represent the direction of the motors, is fed into Equation 3.20.

$$DO_{value} = dx \times 1 + dy \times 2 + dtop \times 4 \tag{3.20}$$

The DO_{value} integer value represents the status of the three ports as shown in Table 3.4. Port1 is connected to the X-motor drive circuits, Port2 is connected to the Y-motor drive circuits and Port3 is connected to the top-motor drive circuits.

DO_{value}	Port3	Port2	Port1
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Table 3.4: Digital I/O port assignment

3.6.6 Analogue Output Program

The I/O board has digital to analogue converter outputs (DAC Output). An array is created containing motor drive voltage demands calculated in the control system program. The data is transferred from the array to a buffer. The values are fed from the DAC output ports to the motor drive circuits.

In this section, the practical implementation of the control system using I/O software was described. The next chapter describes the design and construction of the actual physical system. The Shaky Hand system is completed when the physical system is connected to the control system as described here.

3.7 Chapter Bibliography

[1] MICROSOFT CORPORATION. *Microsoft Visual C++.* MICROSOFT CORPORATION, Microsoft Corporation, One Microsoft Way, Redmond, WA, 6.0 edition, 1998.

[2] KEITHLEY INSTRUMENTS INC. *Accessing the DriverLINX Tools.* KEITHLEY INSTRUMENTS INC., 1999.

[3] KEITHLEY INSTRUMENTS INC. *Counter/Timer Programming Guide.* KEITHLEY INSTRUMENTS INC., 1999.

[4] KEITHLEY INSTRUMENTS INC. *DriverLINX Programming Tutorial Manual Rev.A.* KEITHLEY INSTRUMENTS INC., 1999.

[5] KEITHLEY INSTRUMENTS INC. *DriverLINX Technical Reference Manual.* KEITHLEY INSTRUMENTS INC., 1999.

[6] KEITHLEY INSTRUMENTS INC. *Using DriverLINX with Your Hardware.* KEITHLEY INSTRUMENTS INC., 1999.

Chapter 4

Design and Constructions of the Physical System

Contents

4.1 Introduction 76
4.2 Physical Constructions 76
4.2.1 General View 76
4.2.2 Design Concern 79
Two Layers 79
Wiring Arrangement 79
Wire Track 80
The Sensor Plate 80
4.3 Electrical Circuits 81
4.3.1 Coil Sensors and the Pick-Up Assembly 81
Original Coil Sensor System 82
Issues raised on the Original Plate Design 83
Revised Coil Sensor System 84
Circuit Designs for the Revised Coil Sensor System 86
Realisation of the Revised Coil Sensor System 87
4.3.2 Alternative Sensors Considered 89
Alternative Sensors 1: Hall Effect Sensors 89
Alternative Sensors 2: Optical Sensors 89
4.3.3 Optical Encoders 92
4.3.4 Potentiometer 92
4.3.5 Pulse Width Modulation and Comparator 93
4.3.6 H-Bridge 95
4.3.7 Relay Circuit and Micro-switches 96
4.4 I/O Board 97
4.5 Trial Runs 97
4.6 Chapter Bibliography 103

4.1 Introduction

In this chapter, we describe how the Shaky Hand Hardware has been constructed. This includes the physical construction, the electrical circuits and all wirings. As a generic technology demonstrator, the final system should be neat and presentable. Design issues are discussed that fulfill this aim. Once completed, the whole system was tested. Successful results were obtained in which the plate followed the track without violating the game rules (Section 2.1.3). Images showing the final Shaky Hand system are also displayed at the end of this chapter. A CD is included which contains a variety of short movies showing Shaky Hand operations.

4.2 Physical Constructions

4.2.1 General View

A top view and a side view of the Shaky Hand hardware system is illustrated in Figures 4.1 and 4.2 respectively. They are the schematic diagrams of the physical system and the images of the actual system are shown in Figures 4.23 to 4.25. It was designed to fit inside a 600mm by 450mm footprint, with a base plate made of aluminium alloy. Mechanical work, involving measuring, cutting, machine-lathing, drilling and tapping, was all done in the Student Workshops of the Department of Electronics.

Criteria for the construction of the Shaky Hand system are as follows:

- **A solid structure**
  High rigidity and structural robustness are necessary to protect the system from vibrations during the operation.
- **Accurate alignments**
  All components must be aligned so that no strain is placed upon moving parts.
- **Lightweight translating components**
  Inertia and friction effects must be minimized to reduce unnecessary drag on the moving parts and to operate the Shaky Hand smoothly.

The aluminium base plate provides rigid support for the Shaky Hand and enables exact alignment of components. Rubber stands are attached on the bottom of the base plate to

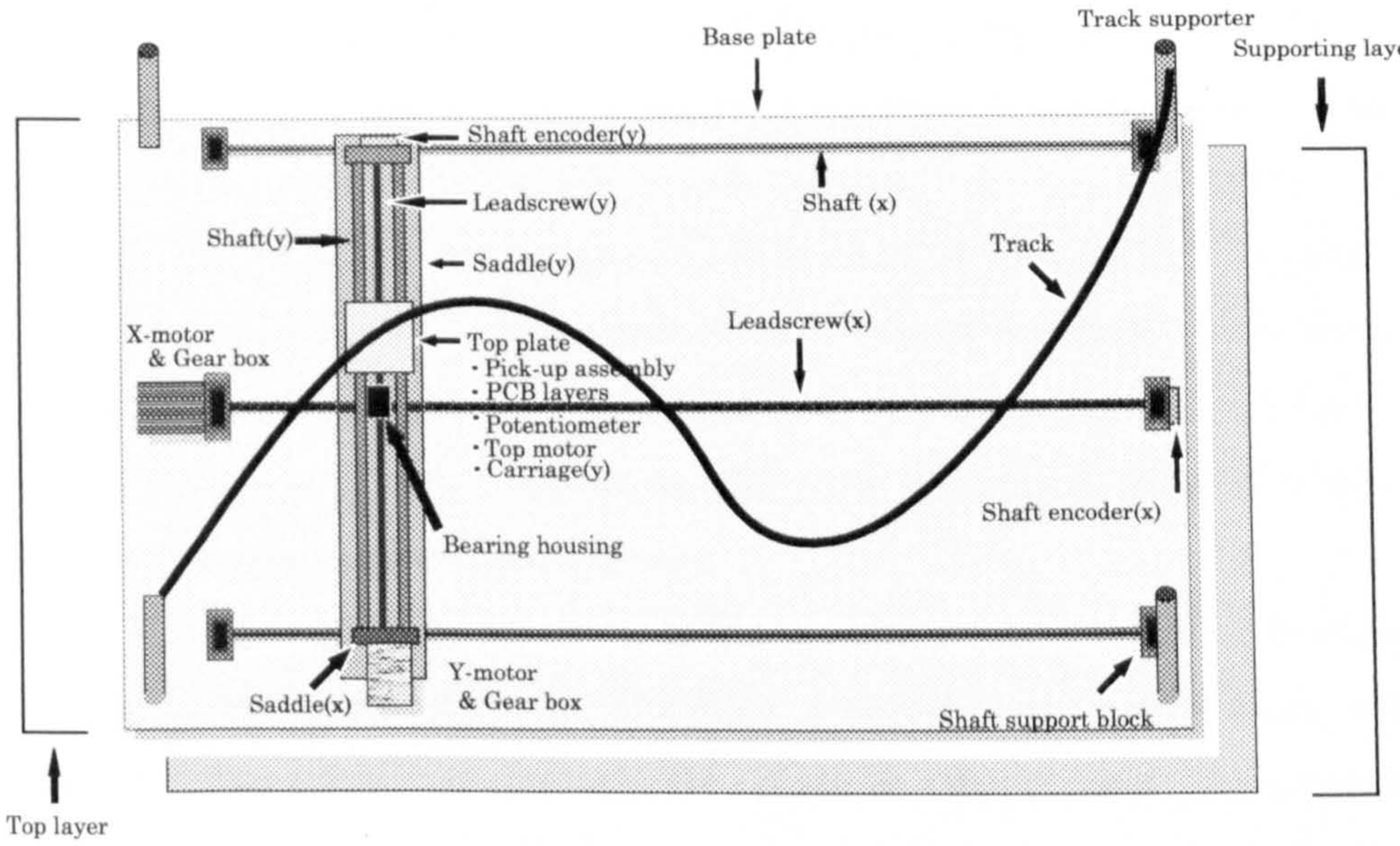


Figure 4.1: Top view of the Shaky Hand hardware

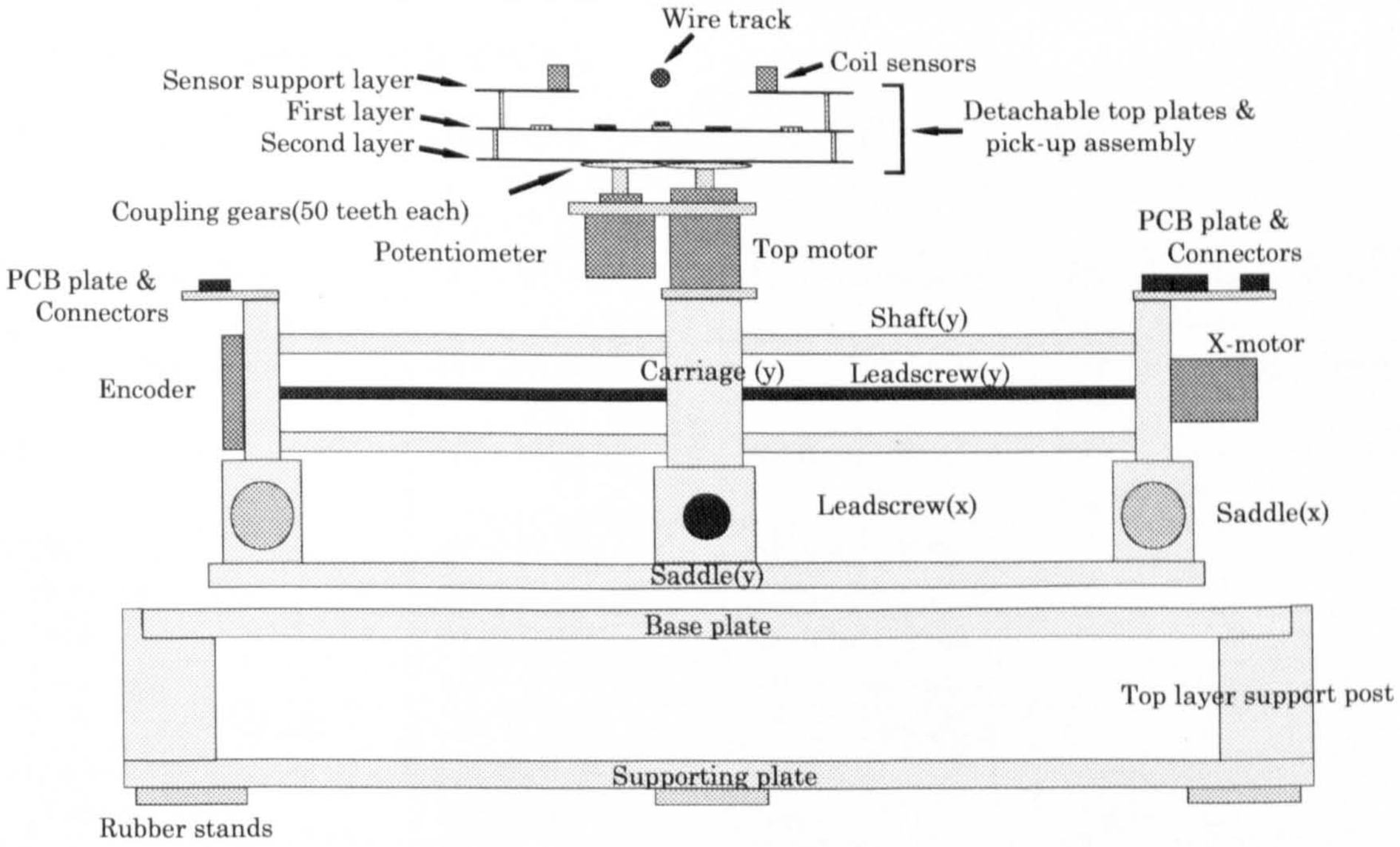


Figure 4.2: Side view of the Shaky Hand hardware

reduce the vibrations on the Shaky Hand. A rotary positioning plate is attached to the X-Y positioning bottom plate, and is driven by a DC-motor (top-motor). The top-motor is fixed upon an aluminium carriage. The plate and the pick-up assembly are placed directly above the top-motor via gear wheels. They are detachable from the top-motor for maintenance purposes (Section 4.2.2). Shafts, carriages and a saddle ensure stable motion of the top plate

in the X- and Y-directions. Linear motions are driven by two DC-motors (X- and Y-motors) coupled to anti-backlash lead-screws via reduction gear boxes. Shaft encoders attached to the motors provide information on the angular positions and velocities of the lead-screw shafts. From this, it is possible to infer the disposition of the top plate in both X- and Y-directions. A potentiometer incorporated within the top-motor allows measurement of the angular orientation of the top plate. The track that the plate is required to follow is made of soft steel, wrapped in a coiled, current carrying wire. PCB plates are used to hold all the connectors for the electrical components used.

The Shaky Hand has two base layers. One is a base frame, holding all the essential parts of the Shaky Hand mentioned above. The other, designed by the author alone, is another rigid plate of similar size that supports the base frame on support posts. This two layer plate arrangement allows the creation of storage space between the layers to hold all the electric circuit boards and other components used to construct the Shaky Hand. The specifications of the metal components are detailed in Table 4.1. The design layouts of the physical parts used are also detailed in Appendix A.1.

Part Name	Dimension (L×W×H/mm or L×D/mm)	Material	Note
Supporting plate	622×469×6	Aluminium alloy	Bottom plate of the Shaky Hand
Supporting posts	25×25×45(39)	Aluminium alloy	Supports the base frame
Bottom plate	600×450×5	Aluminium alloy	Base frame of the Shaky Hand
Shaft(x)	10×532.8	Aluminium alloy	
Shaft(y)	10×380	Aluminium alloy	
Saddle(x)	24×27×25	Aluminium alloy	
Saddle(y)	410×47×5	Aluminium alloy	
Shaft support block	12×27×38	Aluminium alloy	
Lead-screw(x)	532.8×6.35	Lead	1.5mm etching
Lead-screw(y)	380×6.35	Lead	1.5mm etching
Top plate (first layer)	42×20×1.5	PCB	Supports a coil sensor, one for each sensor
Top plate (second layer)	125×125×1.5	PCB	Supports the pick-up assembly
Top plate (third layer)	125×125×1.5	PCB	Supports the first top plate layer
Track	1100×3.5	Soft steel	Core of the wire track
Track supporter	155×18	Aluminium alloy	

Table 4.1: Specification for each part of the Shaky Hand hardware

4.2.2 Design Concern

The Shaky Hand is a generic technology demonstrator. It must look neat and be presentable. The design issues considered by the author are presented here. Figure 4.3 illustrates a summary on the design issue.

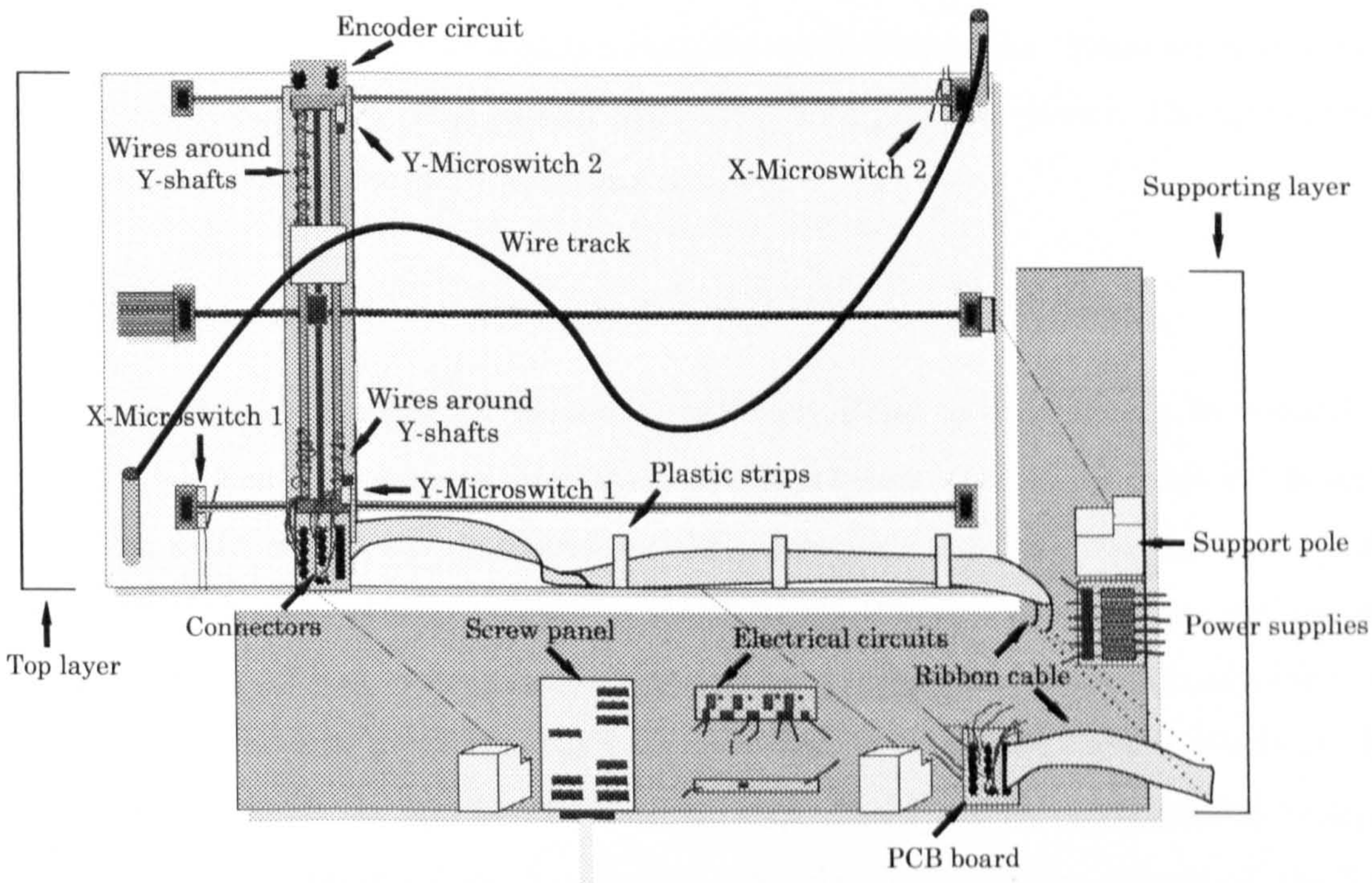


Figure 4.3: Shaky Hand design concern

Two Layers

The two layer base plate system keeps the Shaky Hand looking presentable by allowing all the electrical circuits, connectors, power supplies and I/O board connectors to be hidden between the layers (Figure 4.3). As Figure 4.2 shows, the base frame of the Shaky Hand sits tight on the supporting posts. The posts are not fixed to the base frame but to the supporting layer. Thus, the base frame is detachable. The gap created by the supporting posts is 4.5mm wide.

Wiring Arrangement

The inputs/outputs to/from the I/O boards are connected to appropriate circuits by copper wire. Some of the components move with the plate, thus the wiring materials have to be flexible. Insulated multi-core 1mm copper wire is used. In order to avoid messy wiring, a

ribbon cable is employed together with connector jacks on PCB boards (Figure 4.3). One end of the ribbon cable is capable of moving along the X-saddle and the other end is fixed to the supporting layer via a similar connector on a PCB board. Half of the ribbon cable on the base frame layer is fixed to the base frame by plastic strips to prevent twisting.
The plate must also move in the Y-direction. Therefore any wires from components on the Y-saddle must be able to cope with plate movement in the Y-direction. These wires are coiled around the Y-shafts so that they extend and retract with plate movement. The appropriate PCB design layouts are shown in Appendix A.2.

Wire Track

The wire track shape is designed to enable the Shaky Hand to demonstrate its versatility. It contains two bends in opposite directions and one is gentle whereas the other is relatively tight. The plate model was developed on the assumption that the wire track is locally straight, because of it, abrupt changes in the shape have been avoided. The emf induced into the sensor by current in the wire track is inversely proportional to the distance from the wire as stated in Section 2.4. Thus, the linear relationship between α (lateral offset) and its corresponding voltage, V_α, and also between ϕ (angle offset) and its corresponding voltage, V_ϕ, is only valid for small values of α and ϕ. The physical boundary defined by the four coil sensors on the plate restricts the movement of the wire and means that the values of the offset error can be kept relatively small such that the non-linearity does not pose a problem. The wire track is supported by two track supporter posts located at the corners of the base frame (Figure 4.3).

The Sensor Plate

The top plate supports the tracking sensors and the pick up assembly. A coil sensor is placed on each corner of a square PCB (first layer), equidistance from adjacent sensors as shown in Figure 4.4. Another identical PCB (second layer) is attached underneath this first layer via plastic rods. Signal conditioning circuitry is mounted on this layer. The third layer locks the top plate with the top-motor. Metal rods between the second and the third layers adjust the height of the sensors to match the height of the wire track. The three layer top plate can easily be detached from the top-motor and is replaceable. This allows for the possible installation of alternative sensor systems or plate designs at a later stage. Alternative plate design and the sensors are discussed in Section 4.3.

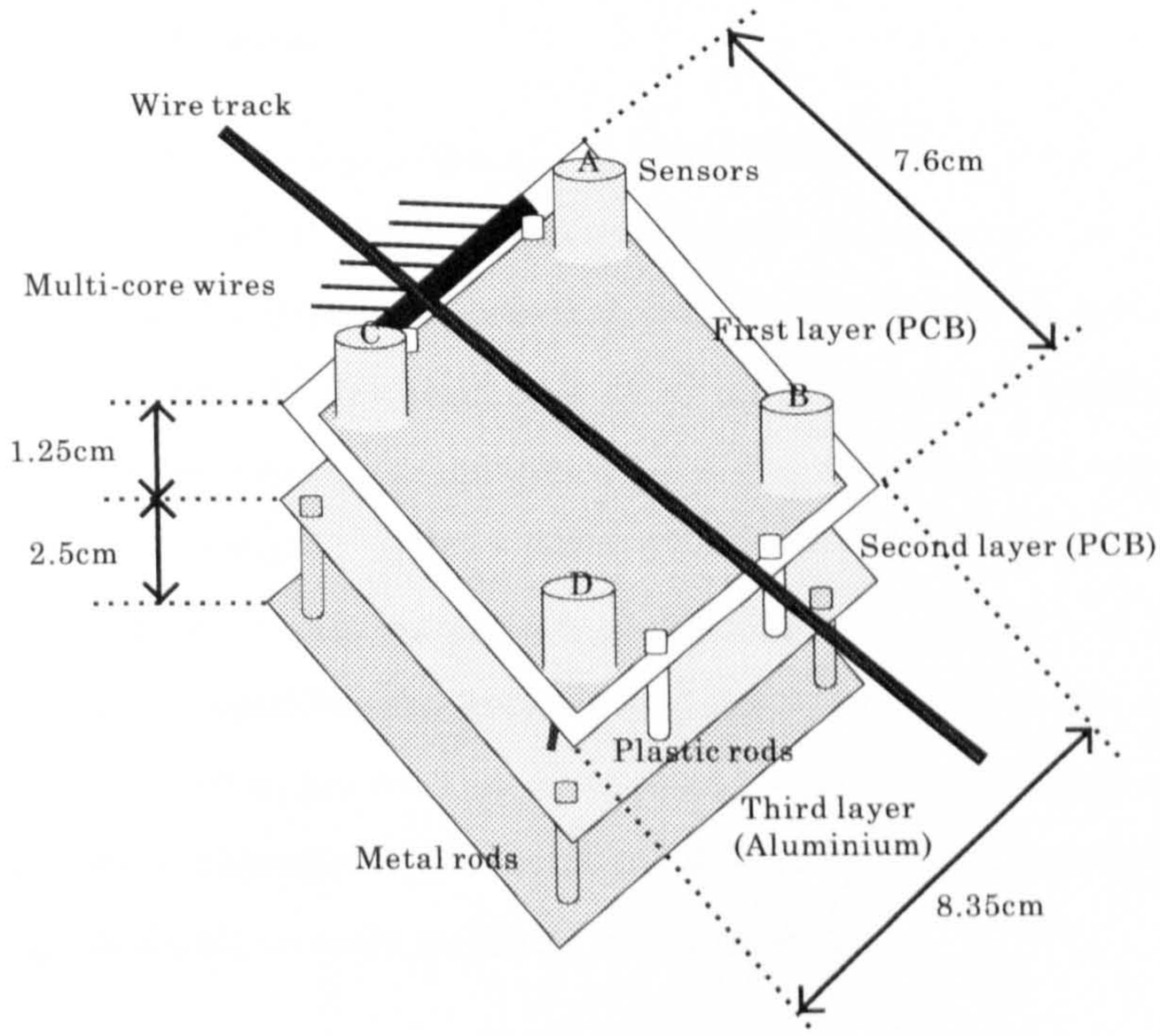


Figure 4.4: Sensor plate

4.3 Electrical Circuits

The specification requires the Shaky Hand plate to move along the wire track at a constant tangential speed. An emf is induced in the coil sensors by the current carrying track wire. The potentiometer measures the angle of the plate relative to the base frame. The optical encoders measure the speed of the plate in the X- and Y-directions. Their signals are sent to a conventional PC via an I/O interface card. The signals are then processed and control signals are fed back to the motors. This process continues until the Shaky Hand plate achieves its positional goal. Micro-switches are provided to stop the operation when the plate reaches the end of the track. All the electrical circuits and sensors involved are described in this section and also in Appendix B.1.

4.3.1 Coil Sensors and the Pick-Up Assembly

In this section, how the coil sensors were implemented initially is first described. When the plate was tested, practical issues arose. A better design of the plate and the circuitry then was made. These are described in the later part of this section.

Original Coil Sensor System

Coil sensors [14] are used as the signal pickup. An emf is induced in each coil by the wire track which carries an a.c. current at 140kHz. The coil sensor outputs must be converted to dc voltage levels before transfer via the I/O interface card and processing on a conventional PC. This RMS-to-d.c. conversion is performed by a single *AD637* IC [3] as shown in Appendix B Figure B.2. This is a high precision monolithic rms-to-d.c. converter that computes the true value of any complex waveforms. *AD637* has a gain-bandwidth limitation in the absolute value circuit. This causes the a.c. bandwidth to drop off as the input level is reduced. To achieve the maximum bandwidth, the induced voltages in the sensors are amplified. This enables the maximum rms signal level to correspond to the rms converter full-scale input level. A quad, precision, high-speed operational amplifier, *OP467* [4], is employed to amplify all four input signals simultaneously as shown in Figure 4.5.

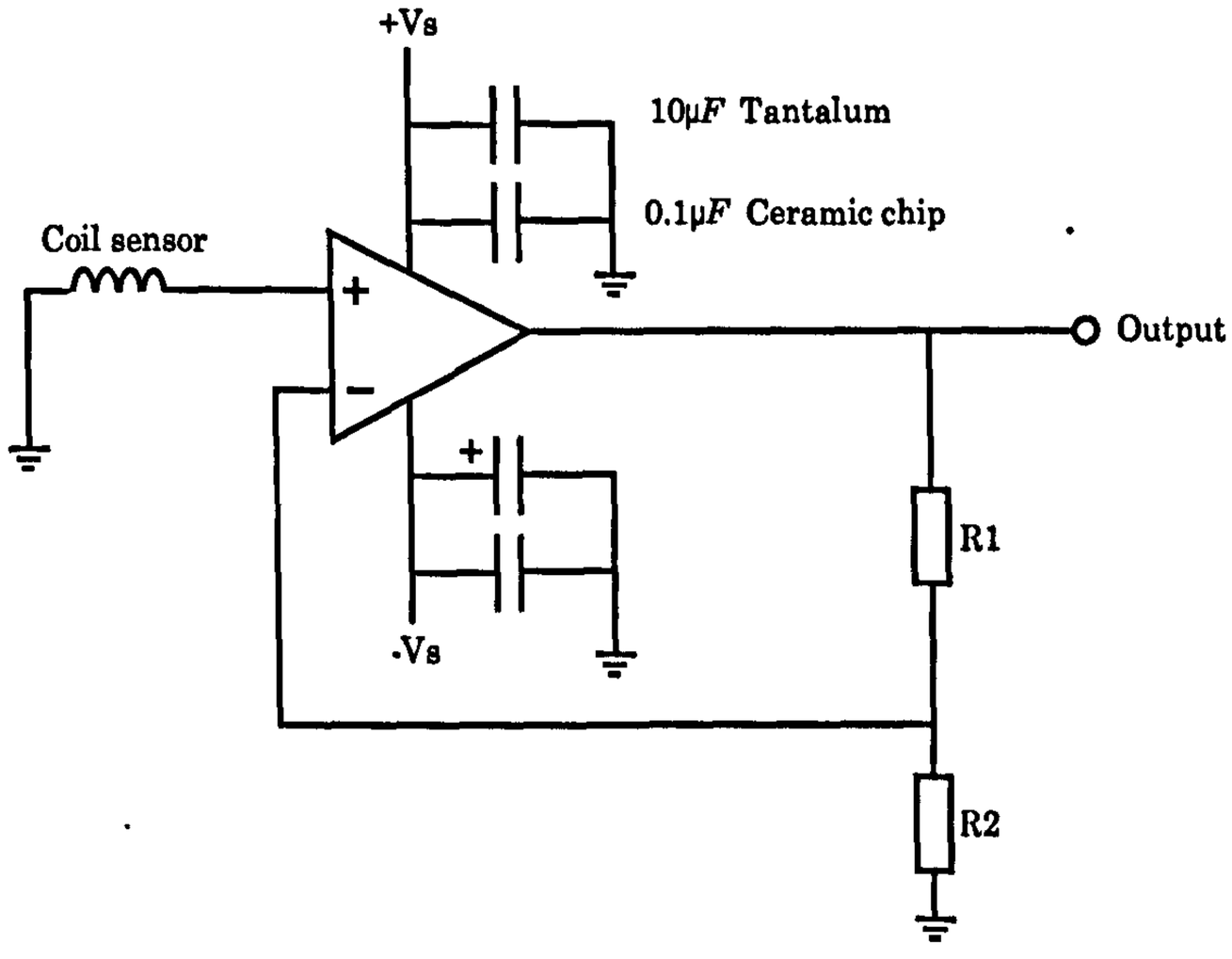


Figure 4.5: Coil sensor amplification circuit

An inductive reactance in series with the circuit is induced by the device lead length due to the relatively high a.c. signal frequency used in the track. This inductance combined with stray capacitance causes gain peaking and additional phase shift, reducing the phase margin of the op-amp which can lead to unstable operation. Placing capacitors across the power supply line and the ground, as close as possible to the op-amp pin, reduces this resonant frequency. Appropriate capacitors have low impedance over the frequency range of the interest. Two

different types of capacitors are combined together for the best results. Tantalum capacitors have high capacitance/size ratio, but their Effective Series Resistance (ESR) increases with frequency, making them less effective. Ceramic chip capacitors have good ESR and also Effective Series Inductance (ESL) performance at high frequencies. The size of the ceramic chips is small placing them very close to the op-amp pins.

Since the four op-amps and sensors may not be strictly identical to each other, the op-amp circuit contains trimmer resistors of the range from 0 to 5kΩ to allow some adjustment. As shown in Figure 4.4, the four coils are placed on the first layer of the top plate and the op-amp and auxiliary resistances and capacitors are mounted on the second layer. Signals from the amplifiers are fed into the rms-dc converter via ribbon cable. The converted signals are passed to the I/O board for processing.

Issues raised on the Original Plate Design

The original design of the plate was constructed and was implemented on the Shaky Hand system for testing. Three implementation issues were found: cross-coupling, interference, and wire distribution.

The first is related to the cross-coupling effect between the coil sensors. In theory, as the wire moves away from a coil sensor, the induced signal strength should decrease inversely with distance. However, in practice, the sensors A and C did not behave this way. The behaviour was normal at close proximity between the wire and the sensor but contradicted with the theory at a distance. The signal became stronger as the wire moved away from the sensor. This arose from a significant amount of cross-coupling between the sensor coils. The second issue was associated with the current induction in the Shaky Hand hardware through the plate. The signals were very sensitive to the proximity of any conducting objects. The third issue is illustrated in Figure 4.6. The seven flexible insulated multi-core 1mm copper wires, used for connecting the outputs and the power supplies with the appropriate circuits placed on one of the X-saddle (Figure 4.3), occasionally interfere with the movement of the plate.

The plate was separated from the rest of the hardware and examined to identify the possible source of the problems. The cross-coupling effect could still be observed on sensors A and C but slightly less significant than before. It suggests the main source of the problem is either in the PCB layout design or the internal structure of the plate. PCB tracks for each sensor circuits may be affecting each other. Closeness of the sensors A and C positions with

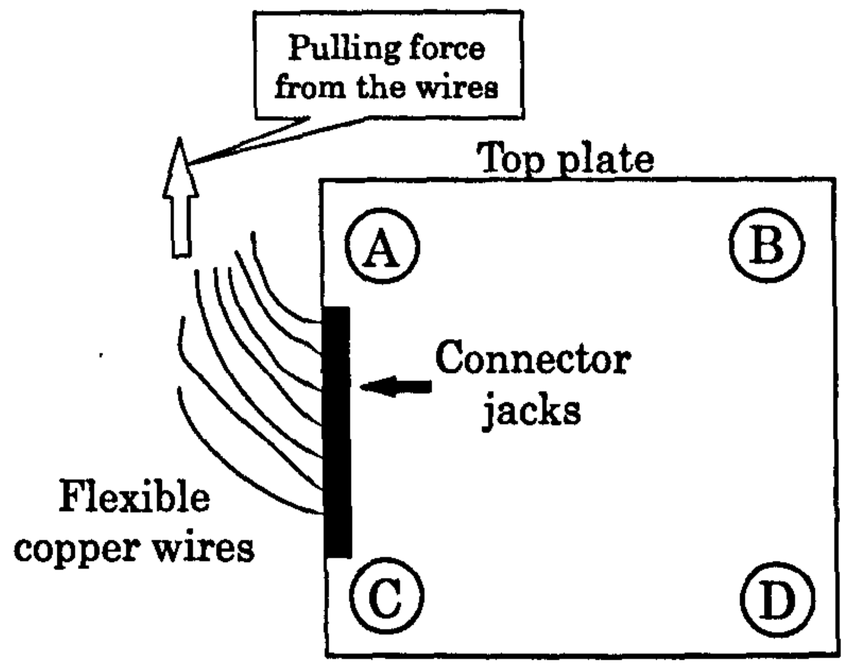


Figure 4.6: The wire distribution issue

the connector jacks may be interfering with the signal outputs. The removal of the external factors for the cross-coupling effect is also required. The signals from the plate are a.c. signals at 140kHz and they are passed to the I/O board via wires and a ribbon cable. The frequency is relatively low but it may interfere with the adjacent wires when the signal strength is high. If the signals from the plate are d.c. signals then the cross-coupling effect can be diminished. Thus, bringing in the d.c. conversion stage on the plate was a possible solution. The interference was largely reduced when the base frame was grounded. The wire physical interference issue can be improved by redistribution of the wires.

Revised Coil Sensor System

Implementation of the rms-to-d.c. converter described previously on the plate was one option. However, a better solution used a lock-in amplifier. This comprises a synchronous demodulator followed by a low-pass filter as shown in Figure 4.7.

In general, a lock-in amplifier utilises two independent differential input stages and a comparator that is used to select the active front end. It works with a low pass filter for rectification. A lock-in amplifier rectifies only the signal of interest while suppressing the effect of noise or interfering components which may accompany the signal. The output is a d.c. signal proportional to the magnitude of an incoming a.c. signal and the phase difference between that incoming signal and a reference signal. Since the frequency of the emf induced into the coil sensors is known and as the reference signal is locked to the incoming signal as shown in Figure 4.7, then, the d.c signal output is proportional to the amplitude of the incoming a.c. signal. This can be verified from the equations below [7].

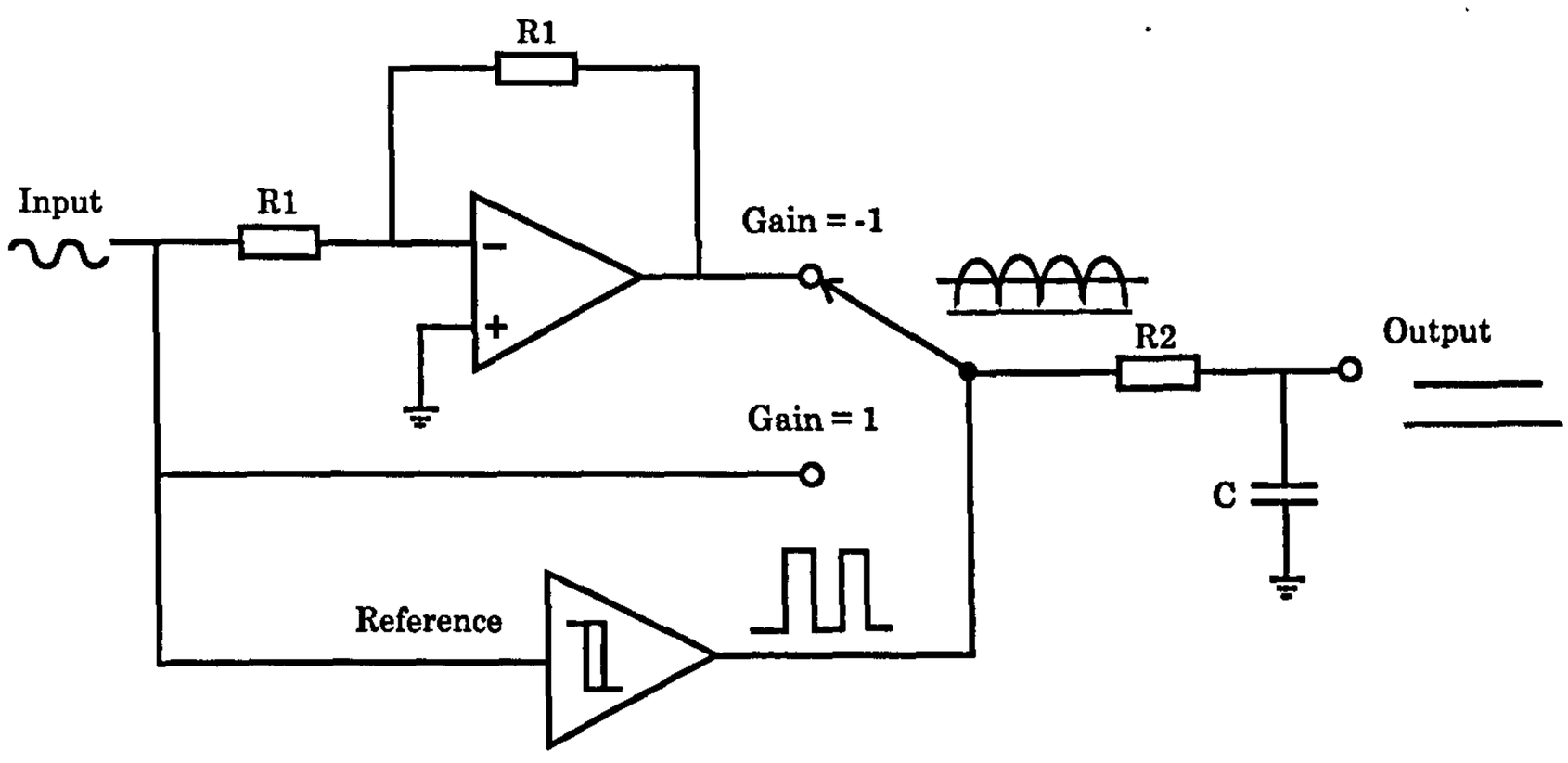


Figure 4.7: Lock-in amplifier

Assume a signal to a demodulator whose reference signal is a square wave with transition at the zero values of $\sin \omega t$.

$$V_{in} = A\cos(\omega t + \phi) \tag{4.1}$$

where A is the signal amplitude, ω is $2\pi \times$ *frequency*, ϕ is the phase difference between the incoming signal and the reference signal.

Let us further assume that we average the output, V_{out} by passing a low pass filter whose time constant is longer than one period, i.e.

$$\tau = RC \gg T = \frac{2\pi}{\omega} \tag{4.2}$$

Then

$$V_{out} = \langle A\cos(\omega t + \phi)\rangle|_0^{\frac{\pi}{\omega}} - \langle A\cos(\omega t + \phi)\rangle|_{\frac{\pi}{\omega}}^{\frac{2\pi}{\omega}} \tag{4.3}$$

where $\langle\rangle$ represents averages, and minus sign comes from the gain reversal over alternate half cycle of reference voltage. Then,

$$V_{out} = -(\frac{2A}{\pi})\sin\phi \tag{4.4}$$

If the input signal and the reference signal are synchronized, the output signal is proportional

only to the amplitude of the incoming signal.

In order to improve the noise performance further, a differential amplifier can be used between each coil and the lock-in amplifier. Figure 4.8 shows the block diagram of the new coil sensor system.

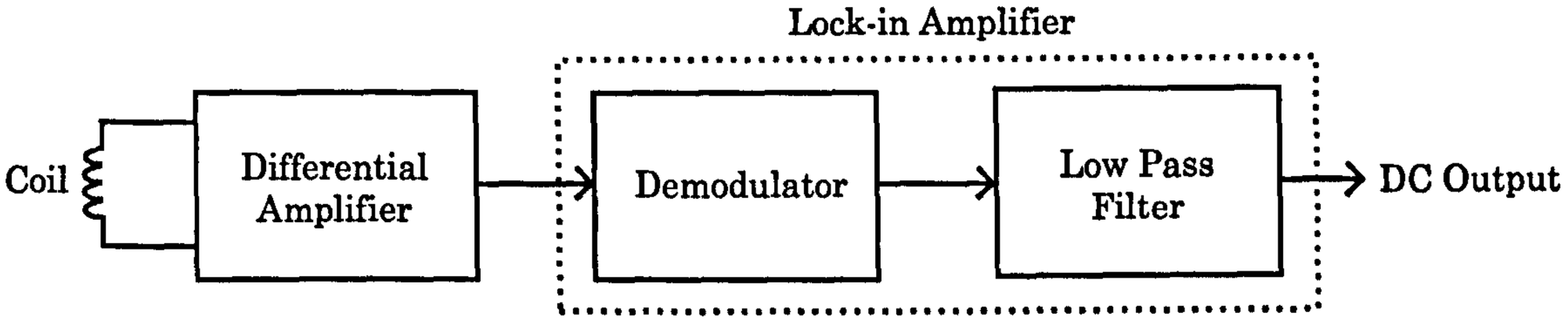


Figure 4.8: Block diagram of the new coil sensor system

Circuit Designs for the Revised Coil Sensor System

Two quad, precision, high-speed operational amplifiers, *OP467*, as used in the previous design were employed in the differential amplifier stage as well as in the low pass filter stage. A differential amplifier can amplify the difference between the two inputs. One of the important features required in the Shaky Hand is the trimming of the sensor signals. This can be effected in the differential amplifier stage by incorporating a variable resistor in the circuit. (See Appendix B Figure B.3 for the circuit configuration). The output from the differential amplifier stage is fed into the lock-in amplifier.

AD630 [2], balanced modulator/demodulator is proposed for the lock-in amplification. It contains two fixed gain stages which can be inserted into the signal path under the control of a voltage sensitive comparator. When the circuit is switched between inverting and non-inverting gain, it provides the basic modulation/demodulation function. It is configured to have gain of unity. The modulation input and the carrier input are connected together to lock each other. (See Appendix B Figure B.4 for the circuit configurations). The output from the demodulator is passed to a low pass filter to rectify the signal as shown in Figure 4.9. The gain is set to 100 to provide appropriate output signals. The time constant of the low pass filter is set longer than one period of the incoming signal to allow the rectification, so $R = 10K\Omega$ and $C = 10nF$. (See Appendix B Figure B.5 for the circuit configuration). Another requirement is that all the circuits use same power supply to keep the number of wire connections to a minimum. The potentiometer, optical encoders and all the ICs on the

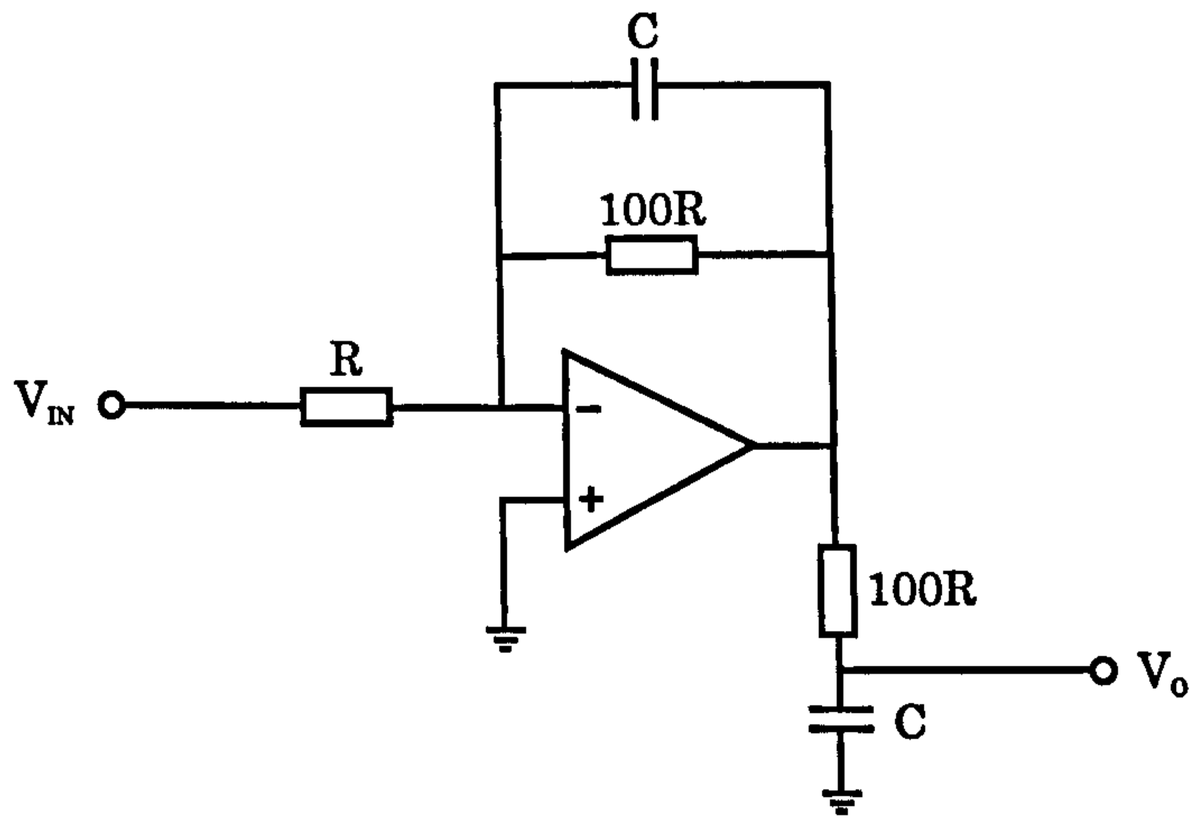


Figure 4.9: Low pass filter for the lock-in amplification [2]

top plate work with ±5V power supply. Decoupling capacitors are used in the same manner as the previous plate design. The test circuits were constructed on a breadboard for testing before implementing onto PCB board. The new coil sensor circuits produces d.c. signals which are proportional to the emf induced in a coil as designed. Four identical coils were selected for implementation.

Realisation of the Revised Coil Sensor System

Basic structure for the top plate is as before. Identical four coil sensors are placed in the first layer, and all the new circuitry described is mounted on a second layer. The third layer locks the plate with the top motor underneath. An image of the new top plate design is shown in Figure 4.10.

Having second layer exposed allows easy adjustment of the trimmers. The sensors are fixed to four separate mounts, and each is attached to the second layer by plastic rods. The pin connector jacks are split onto two opposite sides of the plate, avoiding the wire distribution issue raised before. All the signals from the plate are d.c. signals, diminishing the effect of cross-coupling between adjacent wires or within a ribbon cable.

Upon producing the PCB layout and the plate design, the following points were incorporated to minimise the possible internal cause of cross-coupling effect and the interfering.

- De-couple the power supply.
- Avoid having parallel tracks with the adjacent sensor circuity. Instead, make them orthogonal.

- Provide enough space between the sensors and the pin connector jacks.
- Use twisted pair wiring between the sensors and the circuitry.
- Isolate the rods that connects the layers to avoid current induction to the rest of the hardware.
- Avoid the use of ground plane in the first layer which may not be producing good shielding effect but inducing current.

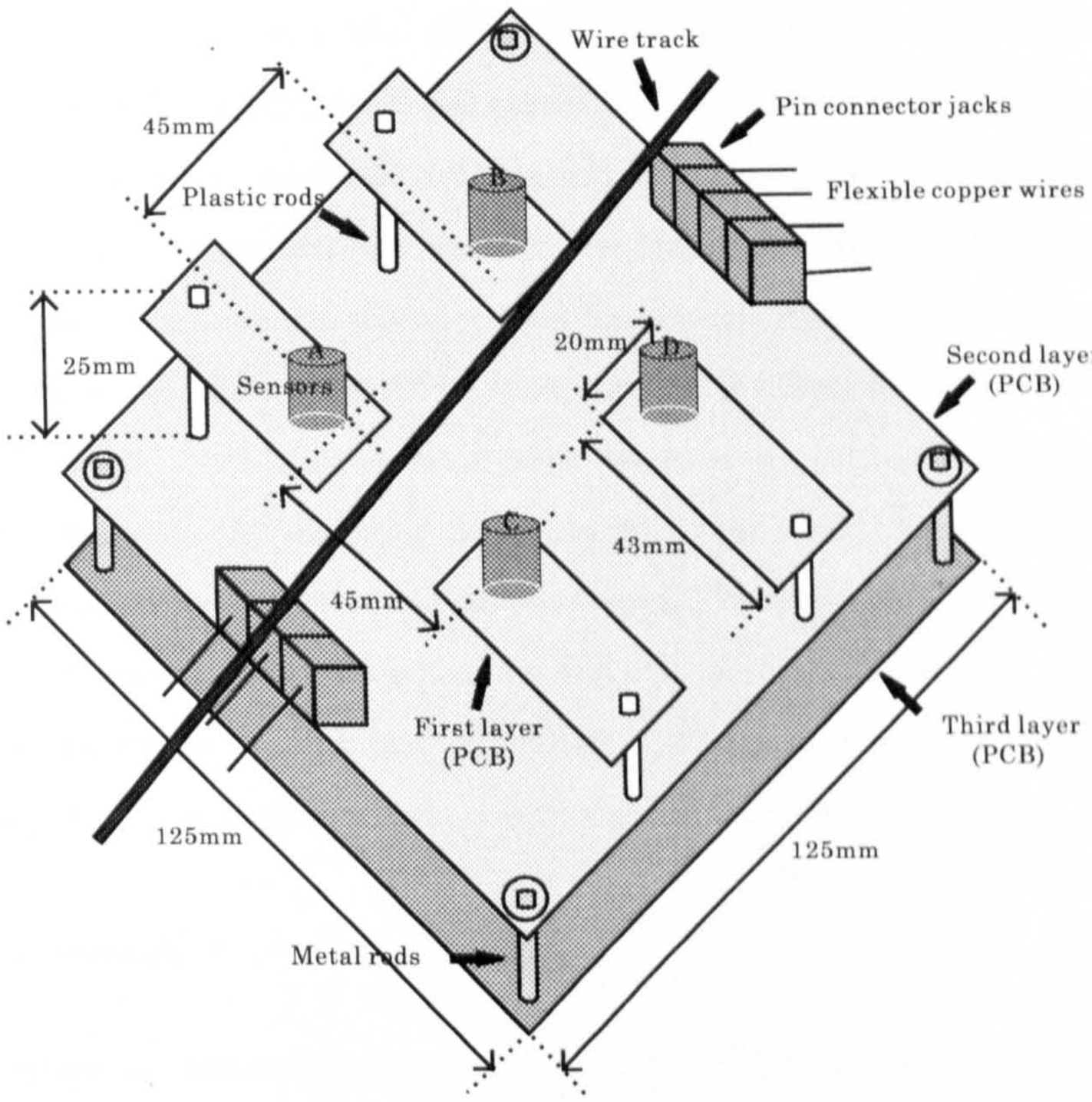


Figure 4.10: Revised sensor plate

Figure B.6 in Chapter B shows the detailed PCB layout of the second layer. It provides hole positions and tracks for all the components. The PCB tracks are on both face and the reverse sides of the layer.

All four sensor outputs provide d.c. outputs proportional to the signal strength picked up by the sensors. The sensors can be trimmed to provide equal signals when the wire is in the central position. Cross-coupling and interference issues have been overcome. The wire distribution issue has been also greatly improved. The outlook of the new and the old plates are displayed in Appendix B.2.

4.3.2 Alternative Sensors Considered

Some possible alternative sensors were also investigated - Hall effect and optical sensors.

Alternative Sensors 1: Hall Effect Sensors

A Hall effect sensor is operated by the magnetic field from a permanent magnet or an electromagnet. The Hall effect sensor, SS495A [6] was investigated, having a ratiometric output voltage, set by the supply voltage. The sensor output varies in proportion to the strength of the magnetic field. The sensor is also slightly smaller than the coil sensors used.
In order to examine the potential for this sensor, four Hall effect sensors were mounted on a breadboard with op-amps and tested with an adhesive flexible magnetic strip as a magnetic track. It was found that the distance between the sensors and the magnetic track had to be kept smaller than the separation distance used for the coil sensors for good detection. Magnetic field strength was then increased by inserting ferrite between magnetic strips for better detection. However, producing a flexible track using ferrite core is challenging in practice. The sensors also detect the magnetic field properly only when the sensors faces magnetic strip head to head. The magnetic field may also easily be disturbed. The Hall effect sensors were then tested with the wire track used for the coil sensors. The sensing accuracy and the reliability was not improved. Practical difficulties and unreliability meant that the Hall effect sensors were not an appropriate alternative to the coil sensors.

Alternative Sensors 2: Optical Sensors

An optical sensor is operated by sensing the reflection of light which it sends out and use it to detect the patterns the sensor is facing. The HEDS-1500 precision optical reflective sensor [12] was investigated. The size of this optical sensor is very similar to the coil sensor used. The photocurrent produced by the sensor is related to the grey-scale level of the sensed surface. In order to exploit this feature a grey-scale track could be implemented just above the sensor, as shown in Figure 4.11. The gap between the lens of the sensor and the grey-scale track should be kept at 4.4mm for best resolution [12].

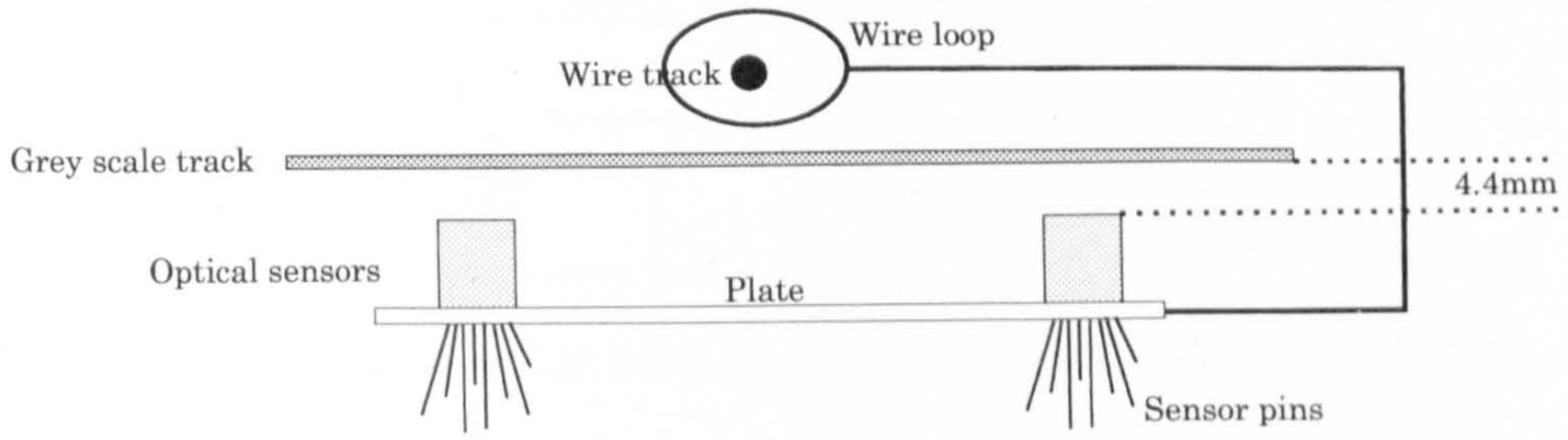
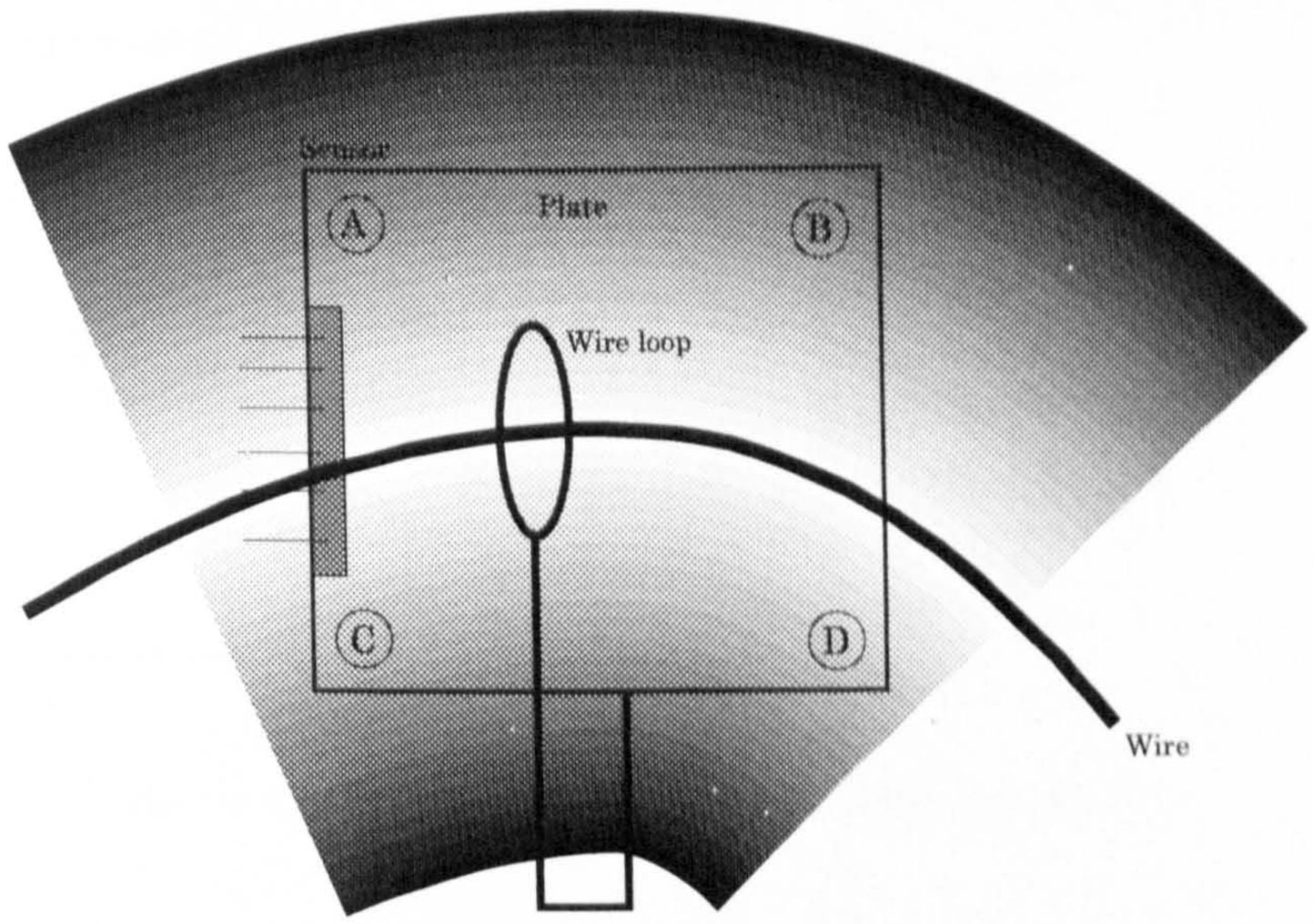


Figure 4.11: Side view of the optical sensor implementation

Figure 4.12 illustrates what the grey scale track would look like. Four sensors would be implemented on the plate just like the coil sensors and the track would cover the whole plate to avoid exposure of the sensors. The grey scale would be printed on white paper and fixed onto a suitable substrate to form the track. With the grey-scale track the relationship between the sensor output and the distance moved by the sensor is linear.



Grey scale track
(Grey scale drawn on the back of plastic board)

Figure 4.12: Grey scale track

An experiment was carried out using a test grey scale track and appropriate components as shown in Figure 4.13. The plate and the grey-scale track were made from PCB. The plate legs were adjustable for best resolution.

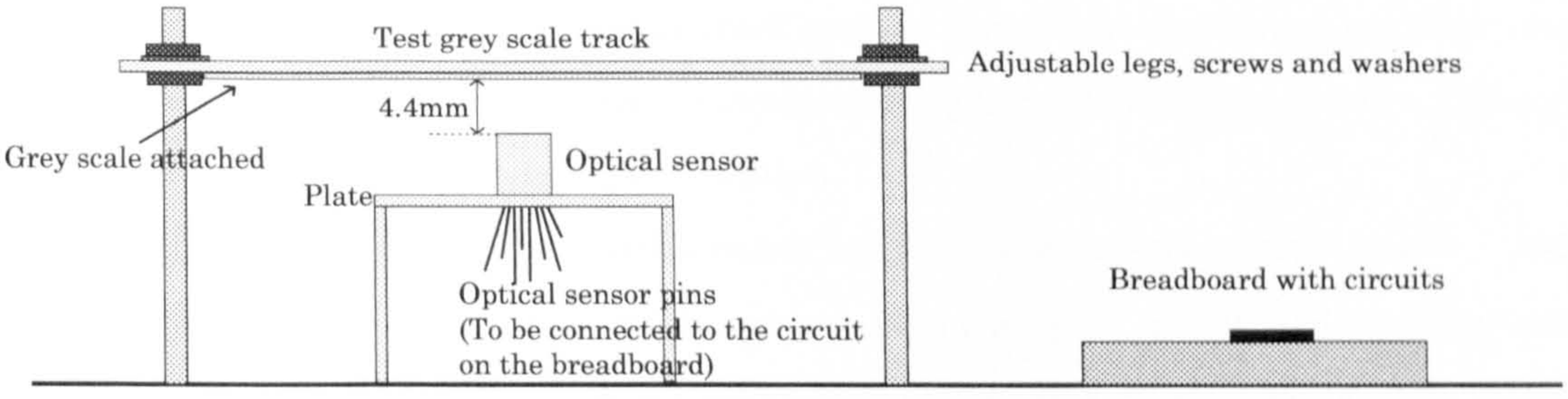


Figure 4.13: Optical sensor experiment preparation

The output from the sensor is the photocurrent (I_{PT}) produced by a photodiode inside the sensor. An output voltage can be generated using a transimpedance amplifier. The test circuit is displayed in Figure 4.14 with appropriate sensor pin numbers for the connections. Forward current of 35mA produces 100nA photocurrent. *MCP601* Single Supply CMOS Op-Amp [13] is used as the op-amp.

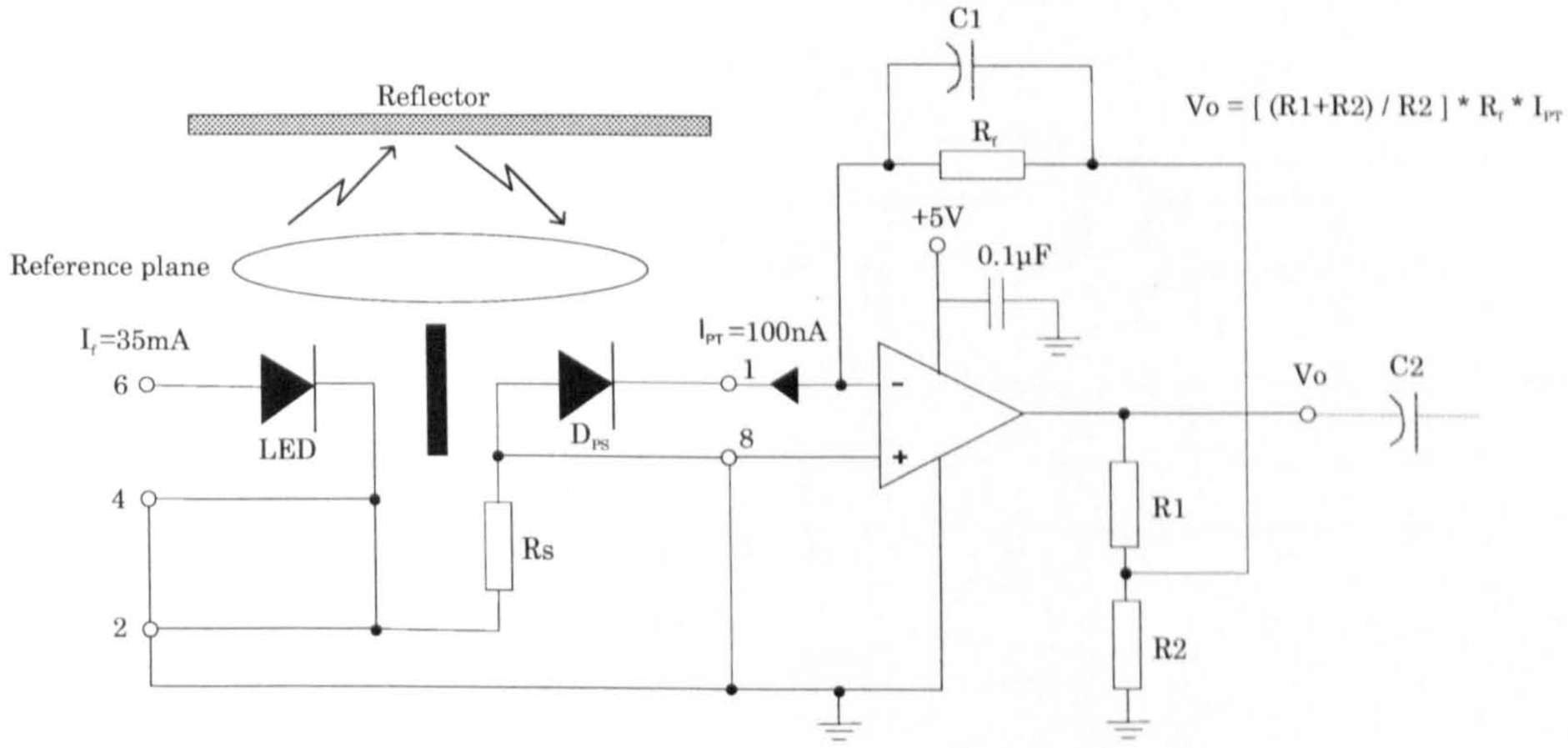


Figure 4.14: Optical sensor with transimpedance amplifier [12]

In the experiment, capacitors C_1 and C_2 were not used. R_f is 100kΩ, R_1 is 100kΩ and R_2 is 150Ω. 35mA Forward current was applied at sensor pin 6. The output voltage value can be calculated from:

$$V_O = \frac{R_1 + R_2}{R2} \times R_f \times I_{PT} \tag{4.5}$$

For a white surface where maximum photocurrent is produced by the sensor, the circuit developed 4.85V. For the black surface this dropped to 1.05V. As the grey-scale colour changed

from white to darker colours, the voltage developed by the circuit changed at constant rate. The track was protected from unnecessary exposure to light for the maximum effect although the result was sufficient under daylight condition.

The optical reflective sensors are possible candidate for the alternative sensor system. However, for visual impact, the optical sensor system is not as good as the coil sensor system because the plate would be hidden underneath the wide grey-scale track made of PCB. Therefore for this research the coil sensors are preferred. The coil sensors worked adequately well.

4.3.3 Optical Encoders

The two optical shaft encoders used were HEDS5540-A11 [10]. These X- and Y-encoders measure the linear velocities of the plate in the X- and Y- directions respectively. It translates the rotary motion of a shaft into either two- or a three-channel digital output. To ensure reliable performance, they employ pull-up resistors as shown in Figure 4.15. PCB is used to place the resisters and also to connect the encoders with the I/O board via interconnection and multi-core wires. They require a 5V supply.

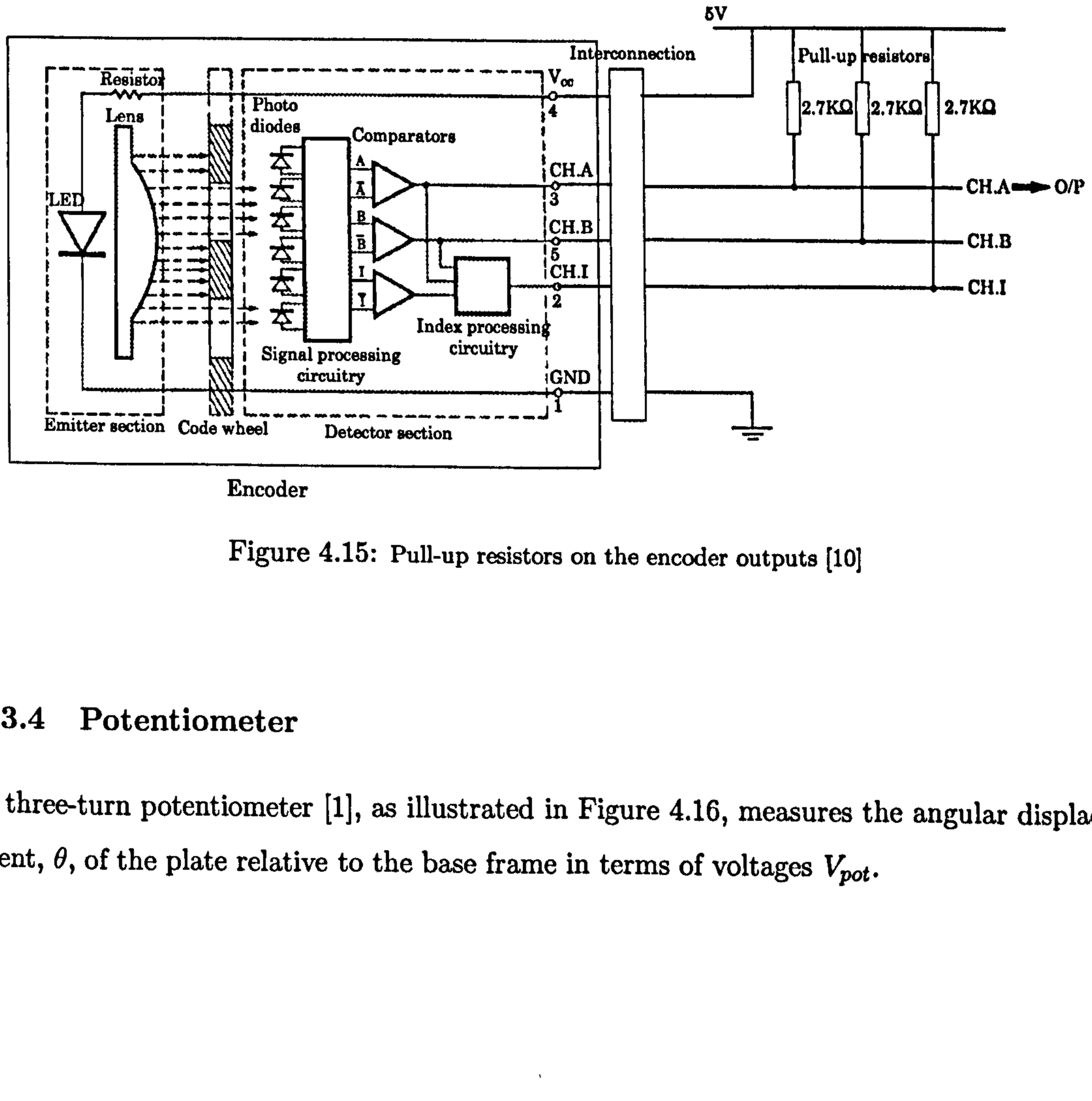


Figure 4.15: Pull-up resistors on the encoder outputs [10]

4.3.4 Potentiometer

A three-turn potentiometer [1], as illustrated in Figure 4.16, measures the angular displacement, θ, of the plate relative to the base frame in terms of voltages V_{pot}.

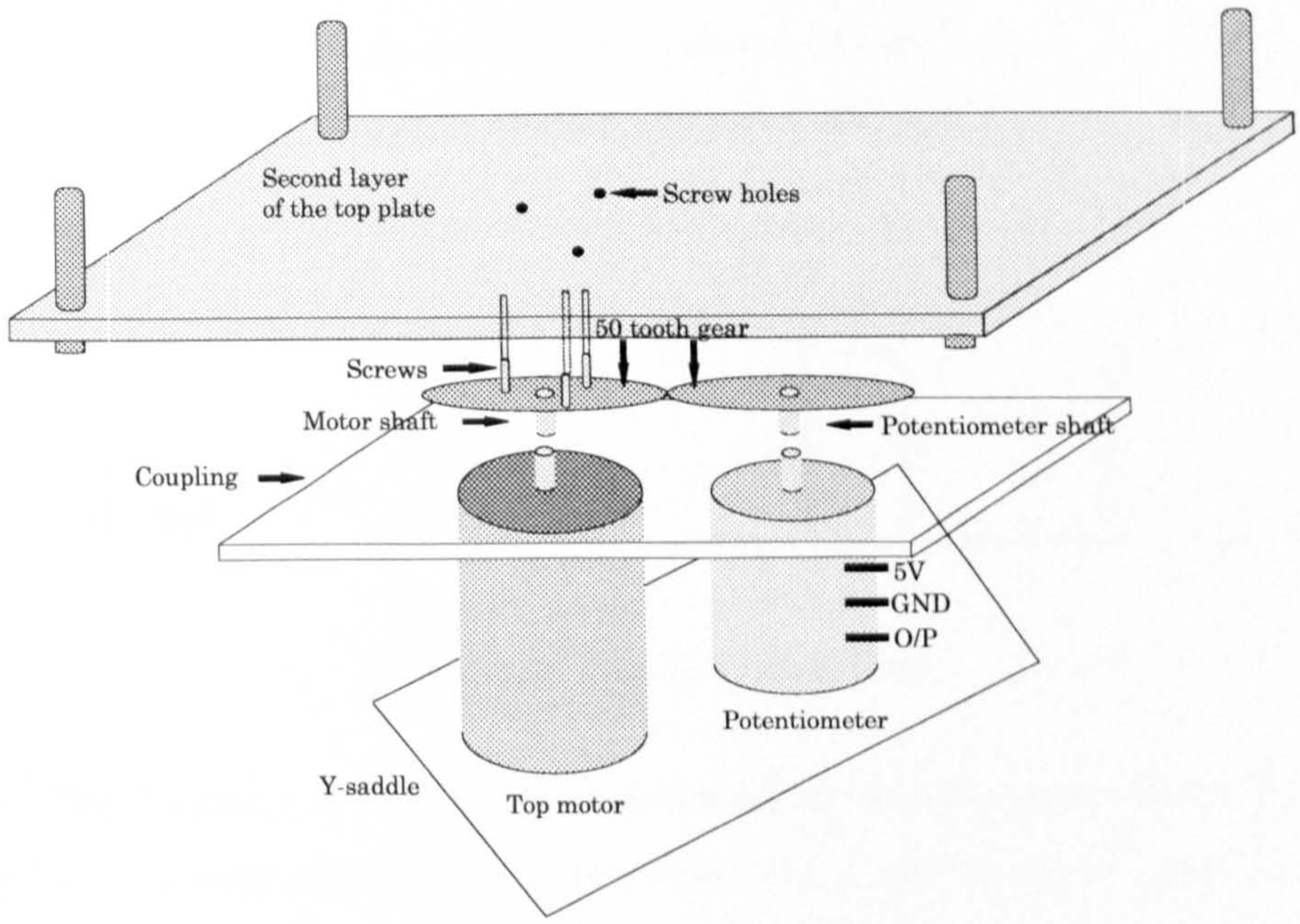


Figure 4.16: A three-turn potentiometer coupled with a DC-motor

It is coupled to the top-motor via two identical 50-tooth gear wheels on a coupling plate. The second layer of the Shaky Hand sensor plate is locked on top. When the top-motor turns, it rotates the plate. The amount of angular displacement is directly transferred to the potentiometer. The potentiometer is powered by a 5V supply and is able to rotate through 1080 degrees in total, although the plate movement is restricted physically by the presence of the sensors on the top plate. The maximum rotation before the sensors touch the wire track is approximately 40 degrees.

4.3.5 Pulse Width Modulation and Comparator

The feedback signals from the I/O board control the speed and the directions of the motors. These signals pass through pulse width modulation (PWM), comparator, and H-bridge circuits, to produce appropriate motor voltages.

The PWM signal drives the motor (Figure 4.17) with the switching frequency of

$$Switching\ Frequency = \frac{1}{T_{ON} + T_{OFF}} \tag{4.6}$$

where

T_{ON} = Duration of motor on time

T_{OFF} = Duration of motor off time

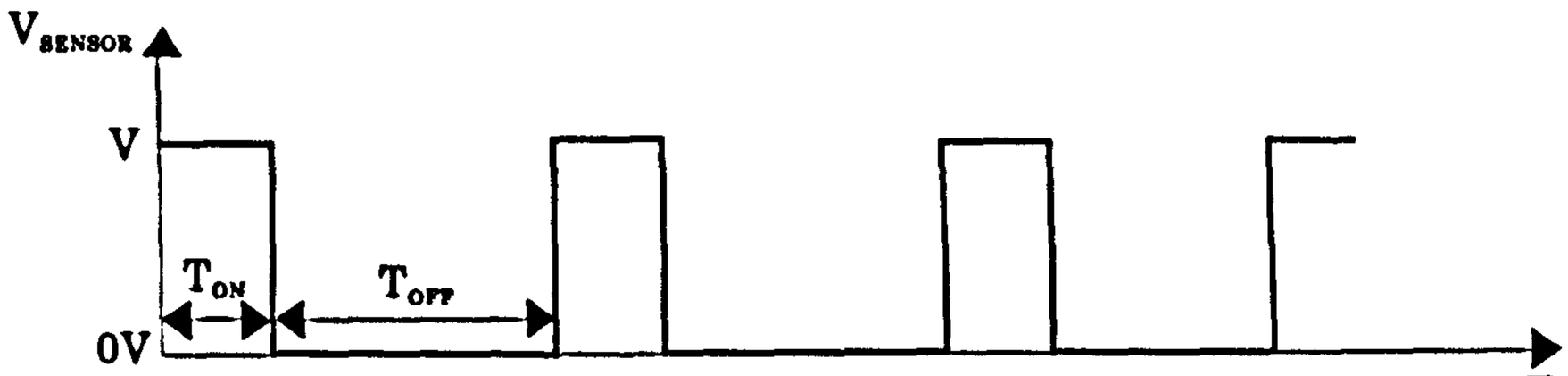


Figure 4.17: Duty ratio of the switch

If the switching frequency is high enough then the motor runs at a steady speed. The average speed, $(V_a)_{avg}$, is regulated by changing the duty cycle of the switch and can be calculated from Equation 4.7. Increasing the duty cycle can increase the motor speed.

$$(V_a)_{avg} = \frac{T_{ON}}{T_{ON} + T_{OFF}} V \tag{4.7}$$

An on-chip oscillator in the LM3524DN IC [11], as shown in Figure 4.18, generates the PWM frequency in a form of a saw-tooth waveform.

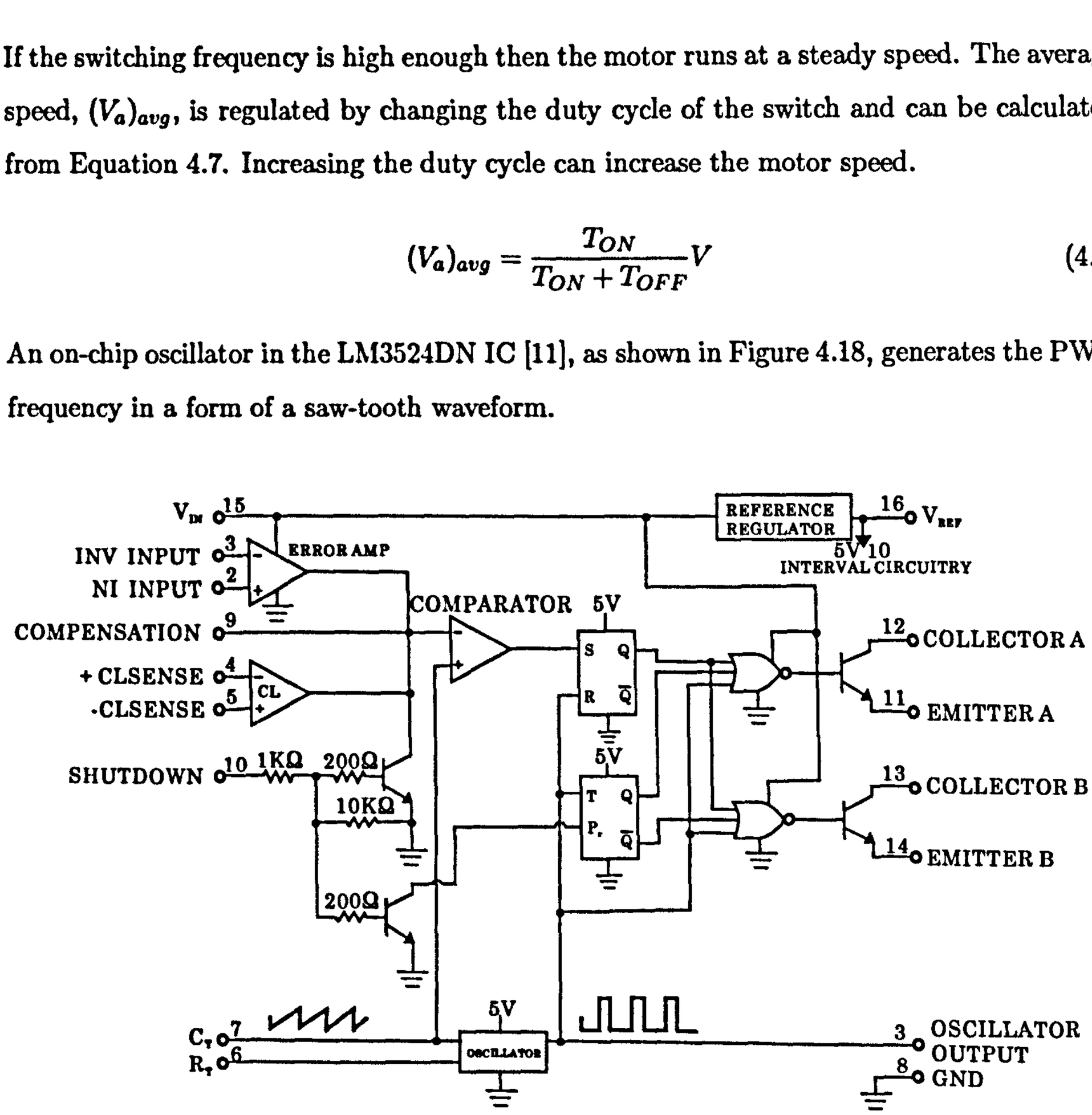


Figure 4.18: Functional circuit diagram of LM3524DN [11]

It is compared with the compensation voltage, which is the speed error voltage, in a comparator amplifier (Figure 4.19).

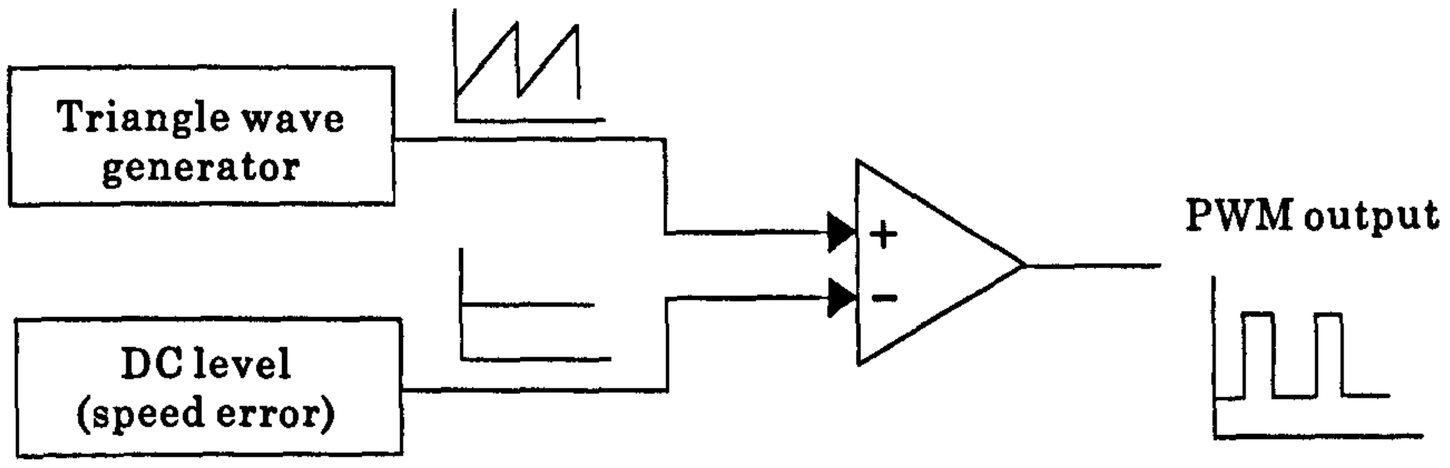


Figure 4.19: Comparator

When the triangular waveform voltage is greater than the reference level, the output of the comparator is high and vice versa. i.e. as the speed error changes, the duty cycle of the PWM signal is varied (Figures 4.20). When the saw-tooth voltage rises above the value of

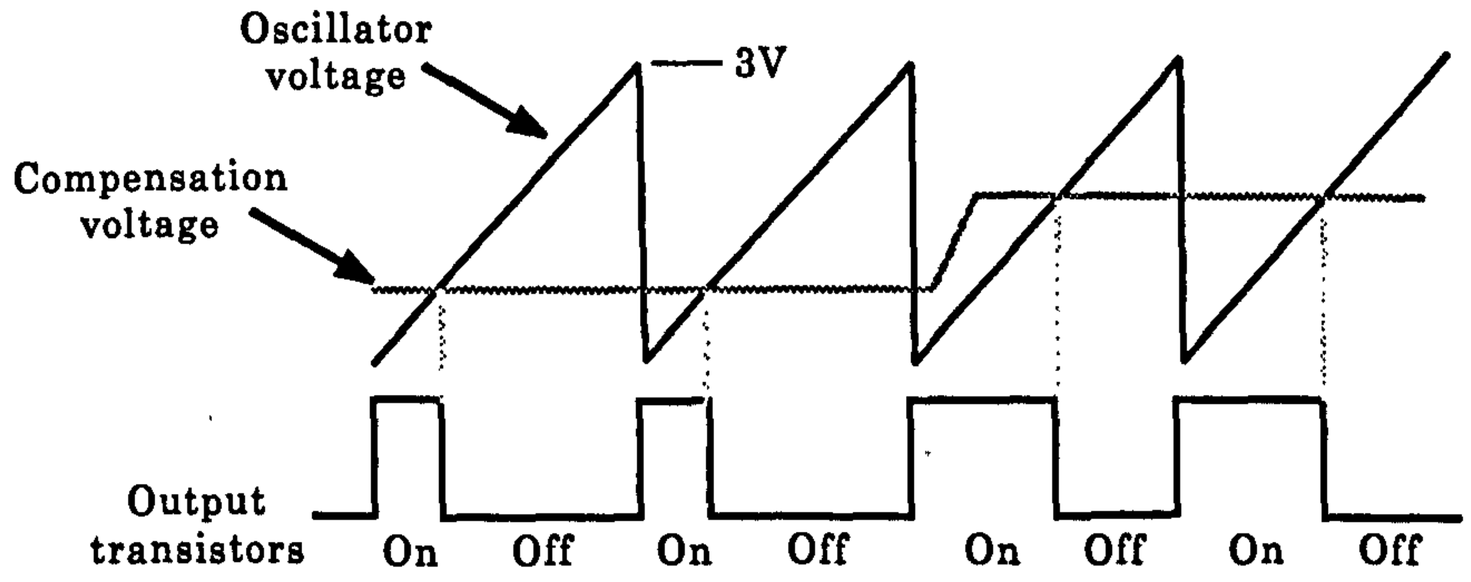


Figure 4.20: Compensated signal

the compensation voltage, the output transistor is turned off until the start of new cycle. By increasing the compensation voltage, the output transistor stays on for longer time which increases the duty cycle. The output from the oscillator triggers internal flip-flops, which pass the PWM information to the outputs.

4.3.6 H-Bridge

A H-bridge circuit is used to control the direction of the motor rotation. A circuit diagram for a motor-reversing circuit using four MOSFETs is shown in Figure 4.21.

When Q_1 and Q_4 are on, the current drives the motor to run clockwise. When Q_2 and Q_3 are on, the motor runs anti-clockwise. The entire operation is operated on 3952SB IC [8].

PWM, comparator and H-bridge circuits are placed together on one PCB board and is shown

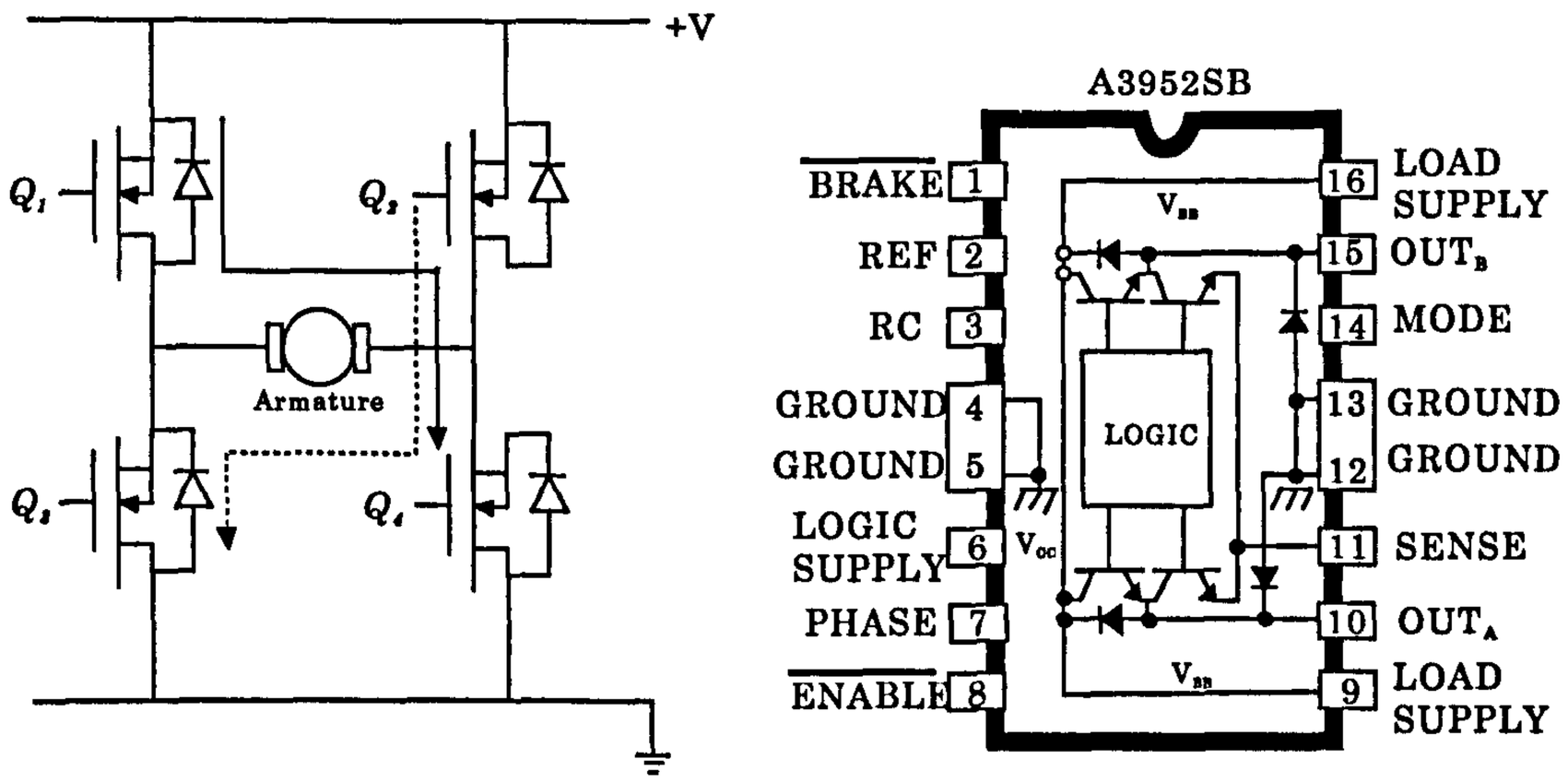


Figure 4.21: A H-bridge circuit:3952SB IC [8]

in Appendix B.2. There are three of them, one for each motor.

4.3.7 Relay Circuit and Micro-switches

The Shaky Hand incorporates four microswitches [9] as previously illustrated in Figure 4.3. They are attached to the shaft support blocks so that they are triggered when the plate reaches the edge of the base plate. When they are triggered the whole operation stops and avoids overrun of the motors. The microswitches in the X-shaft support blocks on each end of the base plate are the positional goals for the plate. The ones in Y-shaft support blocks are installed to abandon the operation when the plate accidentally drifts off too far in Y-direction and touches the Y-shaft.

A relay circuit [5] together with the four microswitches are configured as shown in Figure 4.22. One of the outputs from each PWM circuit is connected to the corresponding DC-motor via the relay circuit and the other output is directly connected. Switching one microswitch can energize the relay and all the motor connections are cut off. A warning light is connected and is turned on when this happens. A buzzer may also be installed.

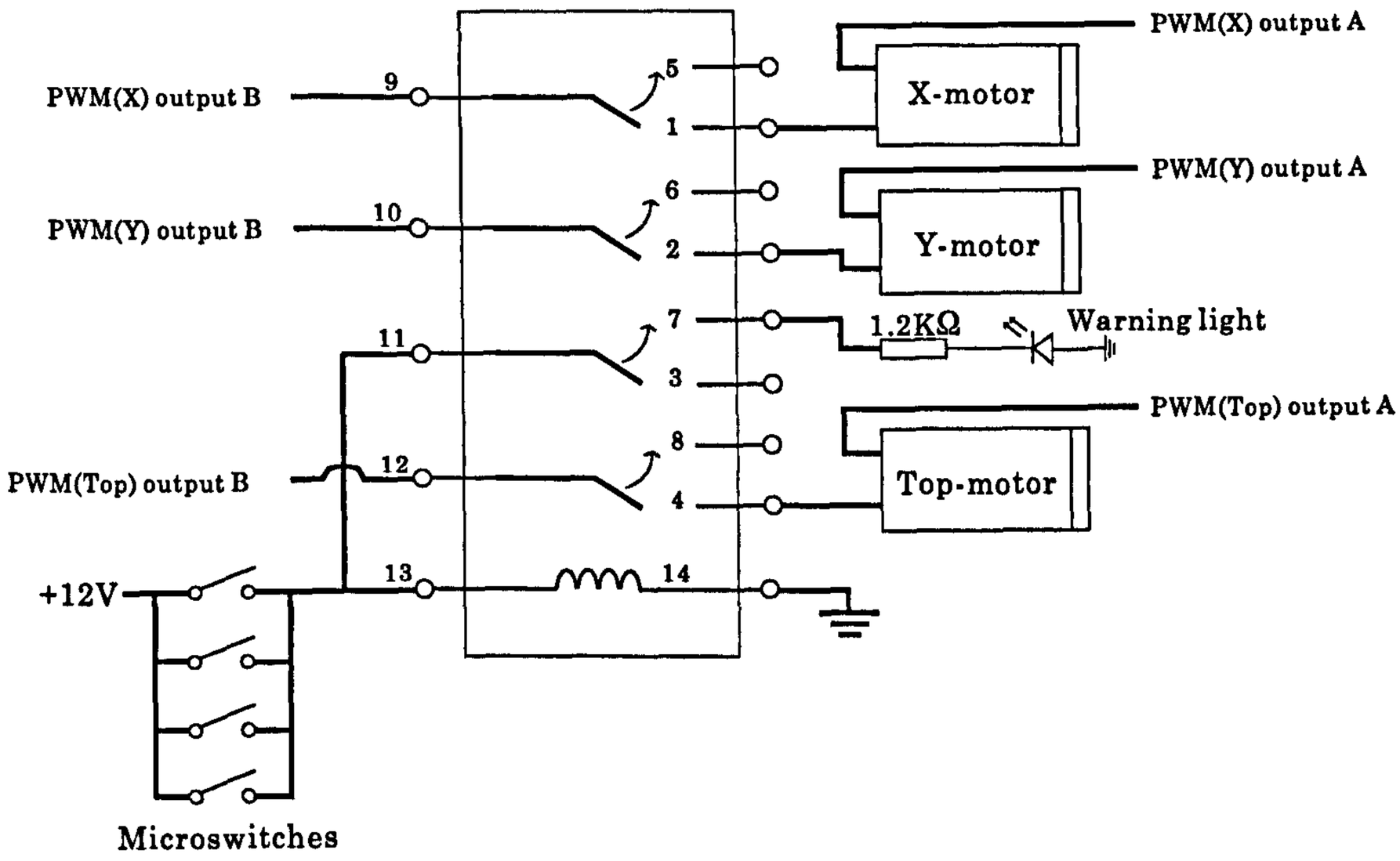


Figure 4.22: Microswitches and a relay circuit

4.4 I/O Board

Two I/O cards are installed in a conventional PC. The seven, continuous sensor outputs are passed to the PC via two I/O interface cards for data acquisition and control signal generation as described in Section 3.6. The returning control signals are passed to the PWM and H-bridge circuits to drive the appropriate motors. The data is transferred between the hardware and the software via bus cables. The I/O pin assignments for the two I/O board are detailed in Appendix B.3.

4.5 Trial Runs

When the whole hardware system was completed, a practical test was carried out to verify the performance of both the hardware and software systems. The followings are investigated.

- Setting up the best operation environment.
- Operation reliability.

Trimmers on the amplifiers on the plate allow the adjustment of the sensor signal strength. Trimming is a necessary procedure before an operation to provide equal sensor signals when

the wire track is in the central position. Sampling time of the signals affects the sensitivity of the plate movement. It was set to 0.1 seconds to enable a fast response. Speed of the motors is adjusted by varying the reference speed and it is increased to 27mm/s from the reference speed used in Simulink simulation (9.5mm/s) for the better performance. The gain parameters of the PID controllers were adjusted to suit the operational condition. The Shaky Hand plate was run, back and forth repeatedly from one end of the base plate to the other, to check the reliability of the control system. Failure is defined as a state in which the plate drifts off completely from the wire track in undesirable directions. On testing, the plate successfully followed the wire track without failure. The operation was automatically stopped once it reaches the end of the track. The speed of the plate was recorded and the mean over 300 seconds was 27.3mm/s, slightly faster than the reference speed.

Pictures of the Shaky Hand system are shown in Figures 4.23 to 4.26. The construction work satisfies the design issues discussed in Section 4.2.2. The wire loop is made of non-conducting material to avoid interfering with the electro-magnetic field created around the wire track.

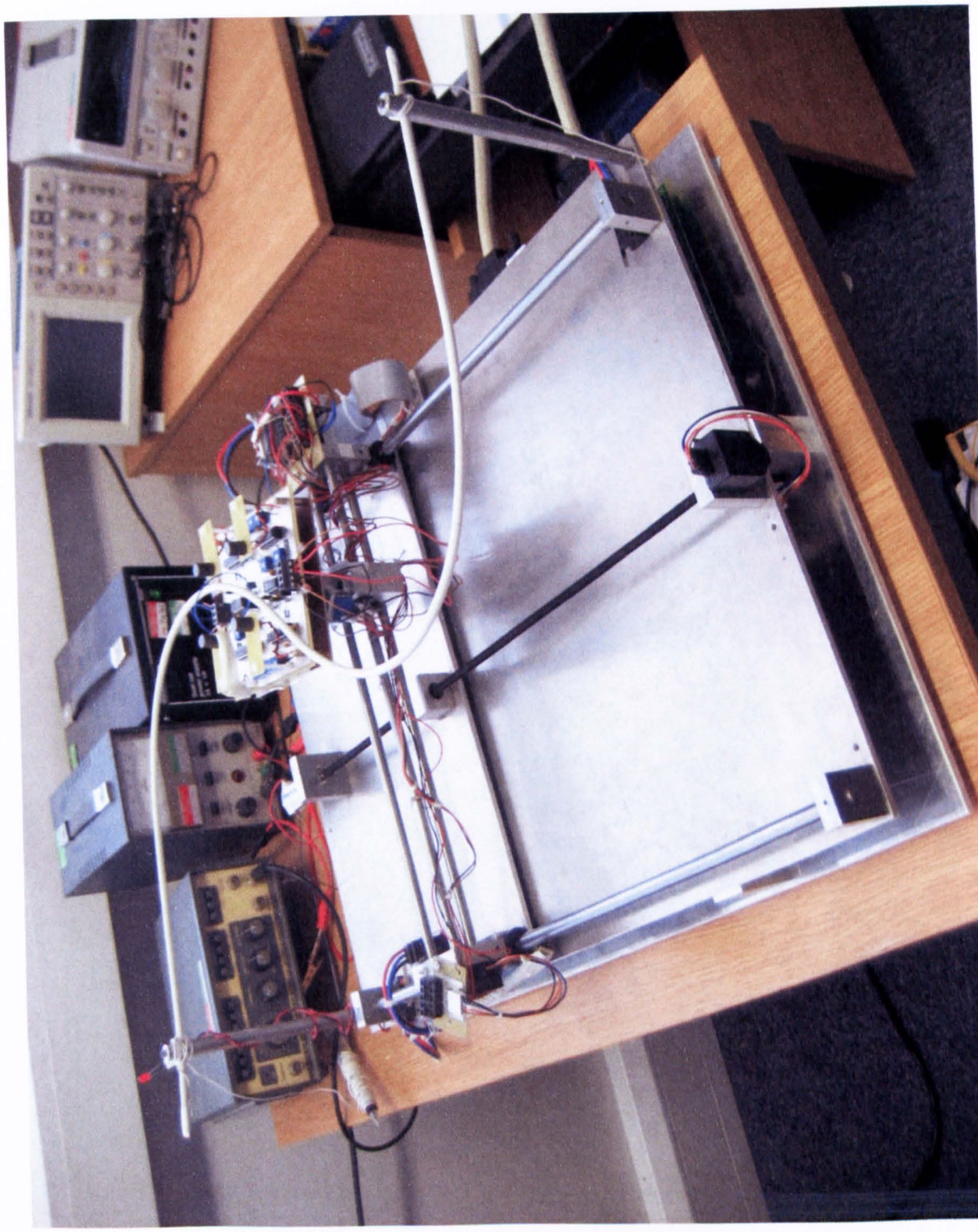


Figure 4.23: Shaky Hand hardware general view

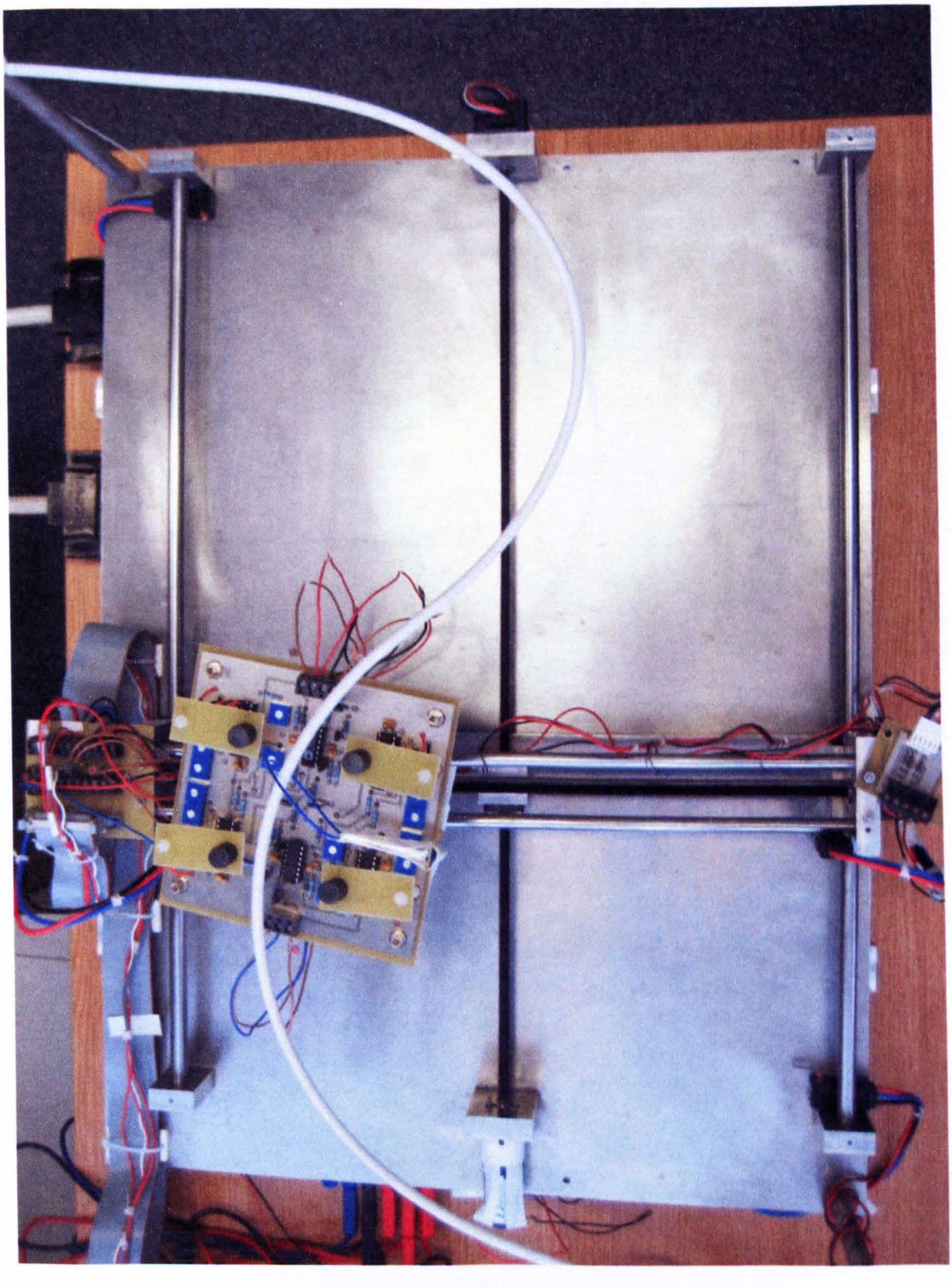


Figure 4.24: Shaky Hand hardware top view

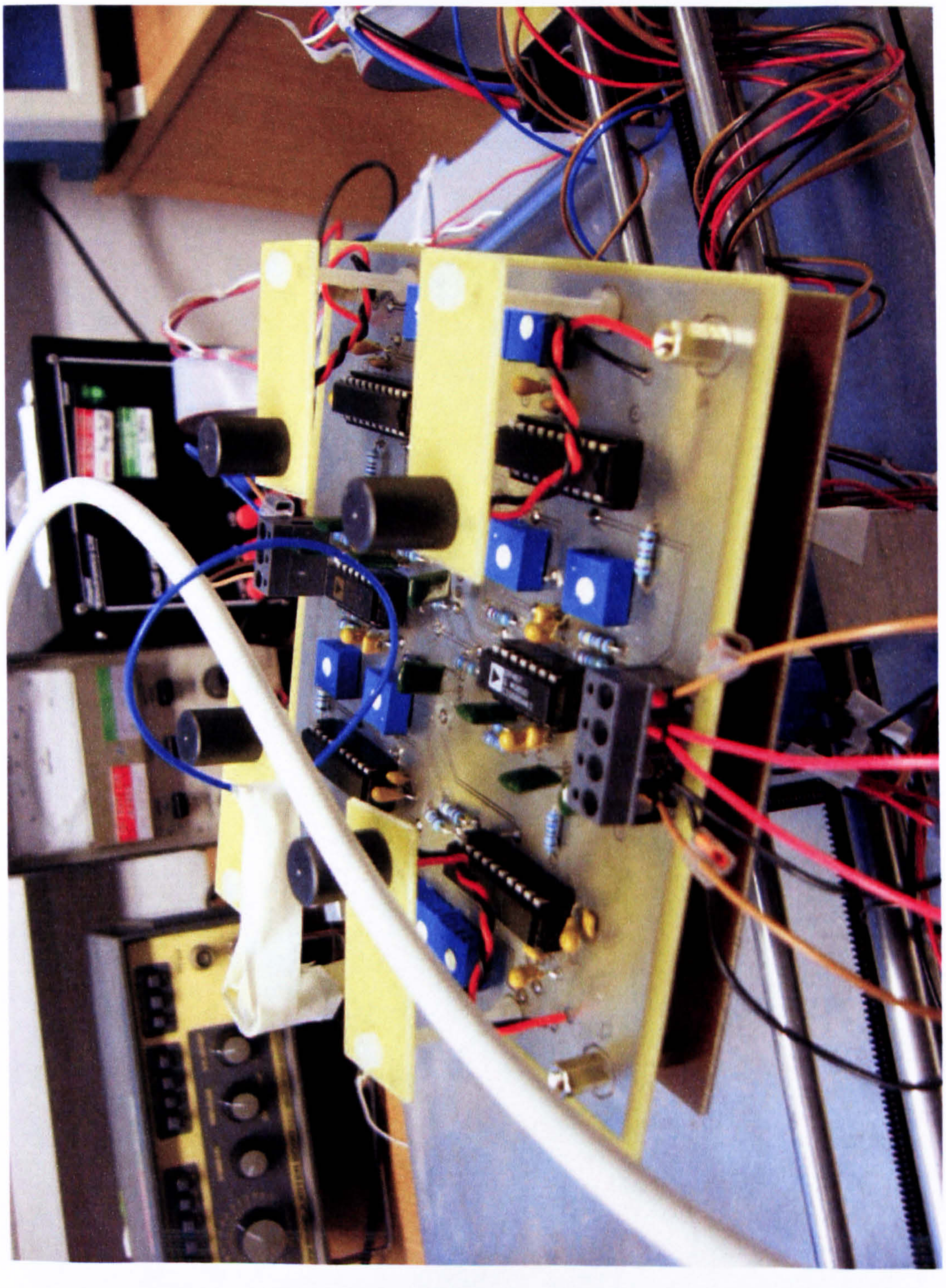


Figure 4.25: Shaky Hand plate

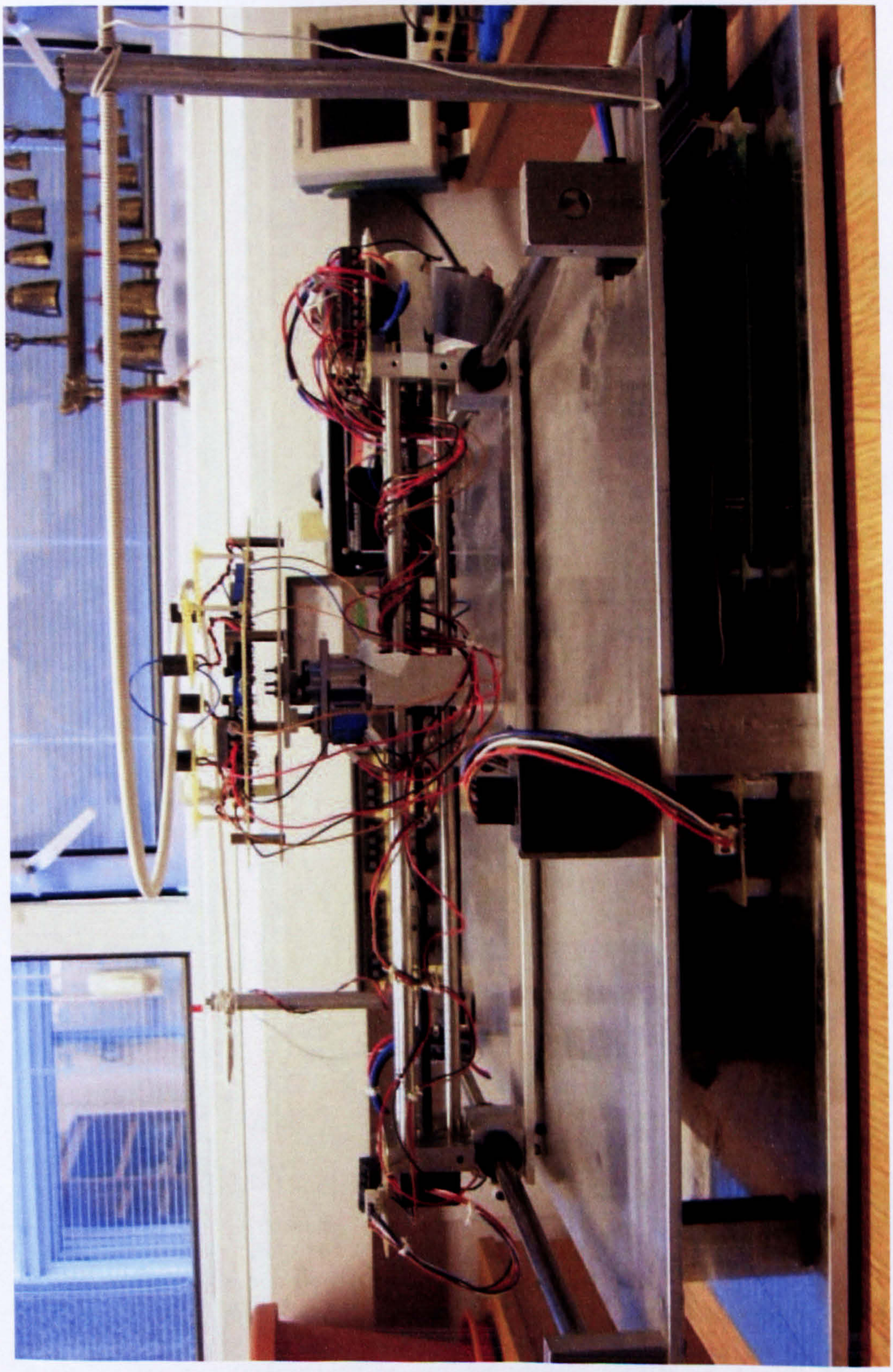


Figure 4.26: Shaky Hand hardware side view

4.6 Chapter Bibliography

[1] BOURNS. *3543/3545 Precision Potentiometer Datasheet.* BOURNS. http://www.bourns.com/pdfs/3543.pdf, August 2007.

[2] ANALOGUE DEVICES. *AD630 Balanced Modulator/Demodulator.* ANALOGUE DEVICES. http://docs-europe.electrocomponents.com/webdocs/077f/0900766b8077fdff.pdf, August 2007.

[3] ANALOGUE DEVICES. *AD637 High Precision, Wide-Band, RMS-DC Converter Data Sheet.* ANALOGUE DEVICES. http://docs-europe.electrocomponents.com/webdocs/0024/0900766b80024b83.pdf, August 2007.

[4] ANALOGUE DEVICES. *OP467 Quad Precision, High Speed Operational Amplifier.* ANALOGUE DEVICES. http://docs-europe.electrocomponents.com/webdocs/077f/0900766b8077ff66.pdf, August 2007.

[5] FINDER. *56 Series - Miniature Power Relays 12A, PCB 4PCO 12VDC.* FINDER. http://www.farnell.com/datasheets/74551.pdf, August 2007.

[6] HONEYWELL. *SS495A Proportional O/P Sensor (Hall Effect Sensor).* HONEYWELL. http://docs-europe.electrocomponents.com/webdocs/002e/0900766b8002e131.pdf, August 2007.

[7] P Horowitz and W Hill. *The art of electronics.* Cambridge University Press, 2nd edition, 1989.

[8] ALLEGRO MICROSYSTEMS. *3952 Full Bridge PWM Motor Driver.* ALLEGRO MICROSYSTEMS, Worcester, Massachusetts. http://people.clarkson.edu/ ortmeyer/ee211/specAugust 2007.

[9] OMRON. *Watertight Miniature Basic Switch.* OMRON. http://www.farnell.com/datasheets/311.pdf, August 2007.

[10] HEWLETT PACKARD. *HEDS5540-A11 Quick Assembly Two and Three Channel Optical Encoders.* HEWLETT PACKARD. http://www.farnell.com/datasheets/7725.pdf, August 2007.

[11] NATIONAL SEMICONDUCTOR. *LM3524 Regulating Pulse-Width-Modulator.* NATIONAL SEMICONDUCTOR, June 1999. http://www.farnell.com/datasheets/9115.pdf, August 2007.

[12] AGILENT TECHNOLOGIES. *HEDS-1500 Precision Optical Reflective Sensor Datasheet.* AGILENT TECHNOLOGIES, www.semiconductor.agilent.com, 1999. http://docs-europe.electrocomponents.com/webdocs/0098/0900766b80098378.pdf, August 2007.

[13] MICROCHIP TECHNOLOGY. *MCP601 Single Supply CMOS Op-Amp.* MICROCHIP TECHNOLOGY. http://docs-europe.electrocomponents.com/webdocs/002f/0900766b8002ffab.pdf, August 2007.

[14] TOKO. *TOKO 10RB124K DATASHEET.* TOKO. http://www.farnell.com/datasheets/76120.pdf, August 2007.

Chapter 5

Approaching Shaky Hand Problems using Cartesian Genetic Programming

Contents

5.1 Introduction 106
5.1.1 Shaky Hand Problem 106
5.1.2 Plate Control 106
5.1.3 Brief Introduction to Cartesian Genetic Programming 108
5.2 Cartesian Genetic Programming 110
5.2.1 CGP Structure 110
5.2.2 CGP Evolutionary Process 112
5.3 Cartesian Genetic Programming For The Shaky Hand 113
5.4 Pseudocode for the Shaky Hand CGP 113
5.4.1 Embodiment 114
Input Data Sets 114
Sliding Windows 117
Evaluation of the Cost Function 118
Multiple Virtual Wire Tracks 119
Fitness 119
5.4.2 CGP Structure and Parameters 121
Selection Method 122
Population Size 122
Number of Functional Blocks 123
Mutation 124
Crossover Effect 125
Operators 129
Number of Generations 130

Program Structure 130
5.5 Shaky Hand Sensor Algorithms 132
5.5.1 Verification Method 133
Test Set 133
5.5.2 Evolved Solutions 135
Case 2: Solutions for One Sensor Failure Cases 135
Case 3: Solutions for Adjacent Sensor Pair Failure Cases 136
Case 4: Solutions for Diagonal Sensor Pair Failure Cases 137
Case 5: Solutions for Opposite Sensor Pair Failure Cases 137
Case 6: Solutions for Three Sensor Failure Cases 138
5.6 Evolved Solution Analysis 139
5.6.1 Cost Analysis 139
5.6.2 Evolutionary Convergence Analysis 145
5.6.3 Improvement Over the Original Solutions 155
One Sensor Failure Case 162
Adjacent Sensor Pair Failure Case 165
Diagonal Sensor Pair Failure Case 168
Opposite Sensor Pair Failure Case 171
Three Sensors Failure Case 174
5.6.4 Further Evaluation of the Evolved Solutions 177
5.7 Adapting The Solutions Into The Shaky Hand 180
5.7.1 Results of Simulink Simulation 180
5.7.2 Results Using Shaky Hand Hardware 186
5.8 Why Do the Evolved Solutions for the One Sensor Failure Cases Work So Well? 187
5.8.1 Analysis of V_α Case 187
5.8.2 Analysis of V_ϕ Case 188
5.9 Chapter Bibliography 191

5.1 Introduction

5.1.1 Shaky Hand Problem

The Shaky Hand must be able to handle unexpected events. What if one or more sensors fails to function correctly and the control solutions used for the Shaky Hand are not valid anymore as a result of the failure? What can the Shaky Hand do except to abandon the operation? These questions can be applied not only to the Shaky Hand but to any hardware of arbitrary scale ranging from a robot in a maze, to a next generation spacecraft. For example, a robot in a maze may be using non-redundant optical sensors to detect the presence of walls in front of it. If one of the sensor fails to function correctly then the robot may simply keep crashing into the walls.

The work presented in this chapter tackles such problems. Using a form of evolutionary programming, Cartesian Genetic Programming (CGP), we evolve generic control solutions with sensor fault tolerance such that the Shaky Hand is able to operate for the case of sensor failures. A novel way of evolving the generic solutions using special training sets is presented as a means of improving evolvability and of avoiding overfitting problems. The evolved solutions were obtained and verified using suitable test sets. They were analysed fully and compared with the original control solutions to show the improvement given by CGP. The solutions were also incorporated into the Simulink [6] simulation before applying the solutions to the real Shaky Hand system.

5.1.2 Plate Control

Figure 5.1 displays the control procedure for the Shaky Hand. Using α and ϕ obtained by the offset error sensing algorithms, together with the potentiometer output and the encoder signals, feedback signals are produced for the three Shaky Hand motors.

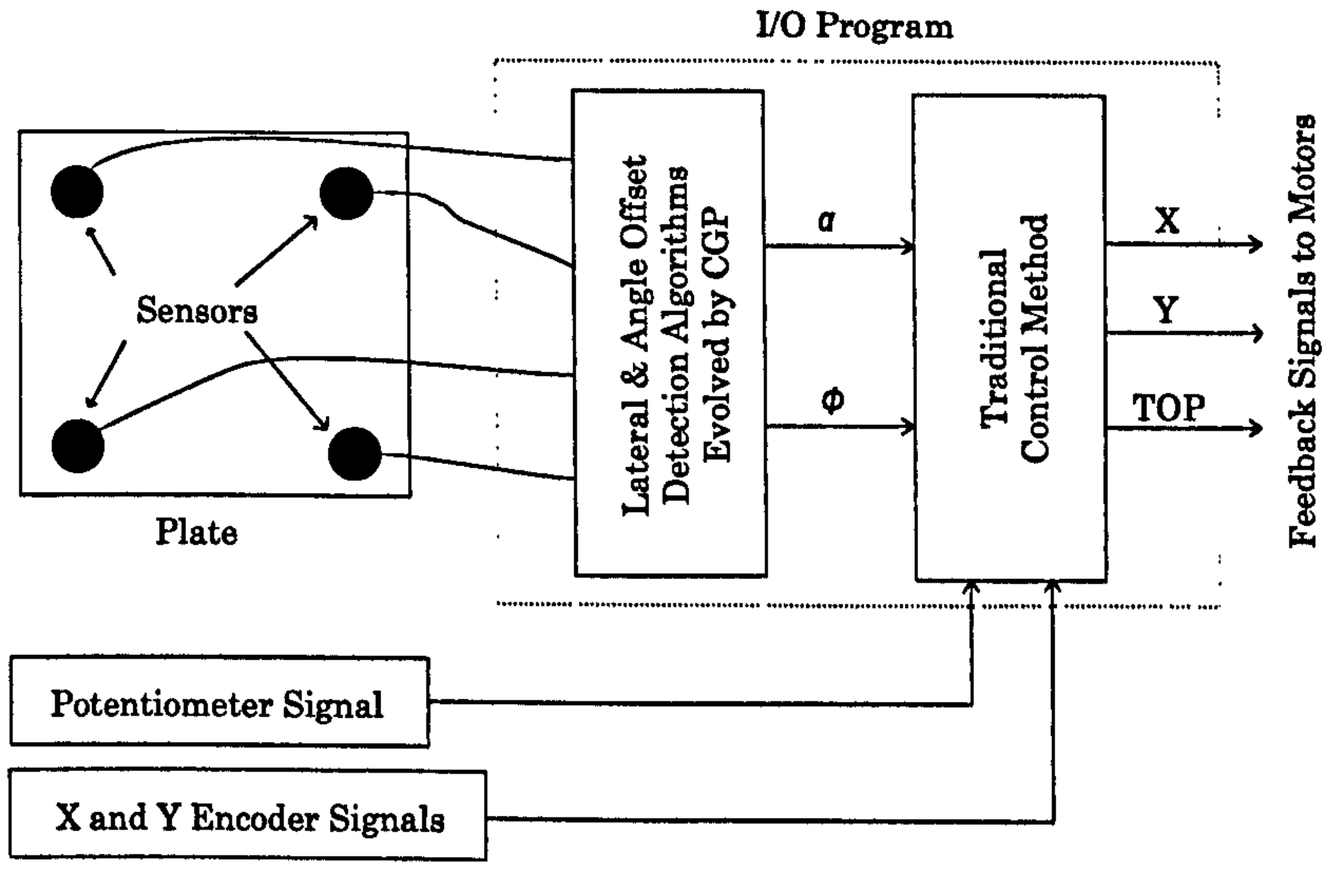


Figure 5.1: The Shaky Hand controller process

The work described in this chapter focuses on the four coil sensors mounted on the Shaky Hand plate. The four sensor outputs are used directly to obtain the lateral offset error, α, and the angle offset error, ϕ, caused by the misalignment between the centre of the plate and the track. These errors are obtained in terms of voltages and are used to control plate movement. The wire is assumed to be locally straight. Figure 5.2 displays the sensor arrangement on the plate. The wire track passes between the top sensors (sensors A and B) and the bottom sensors (sensors C and D).

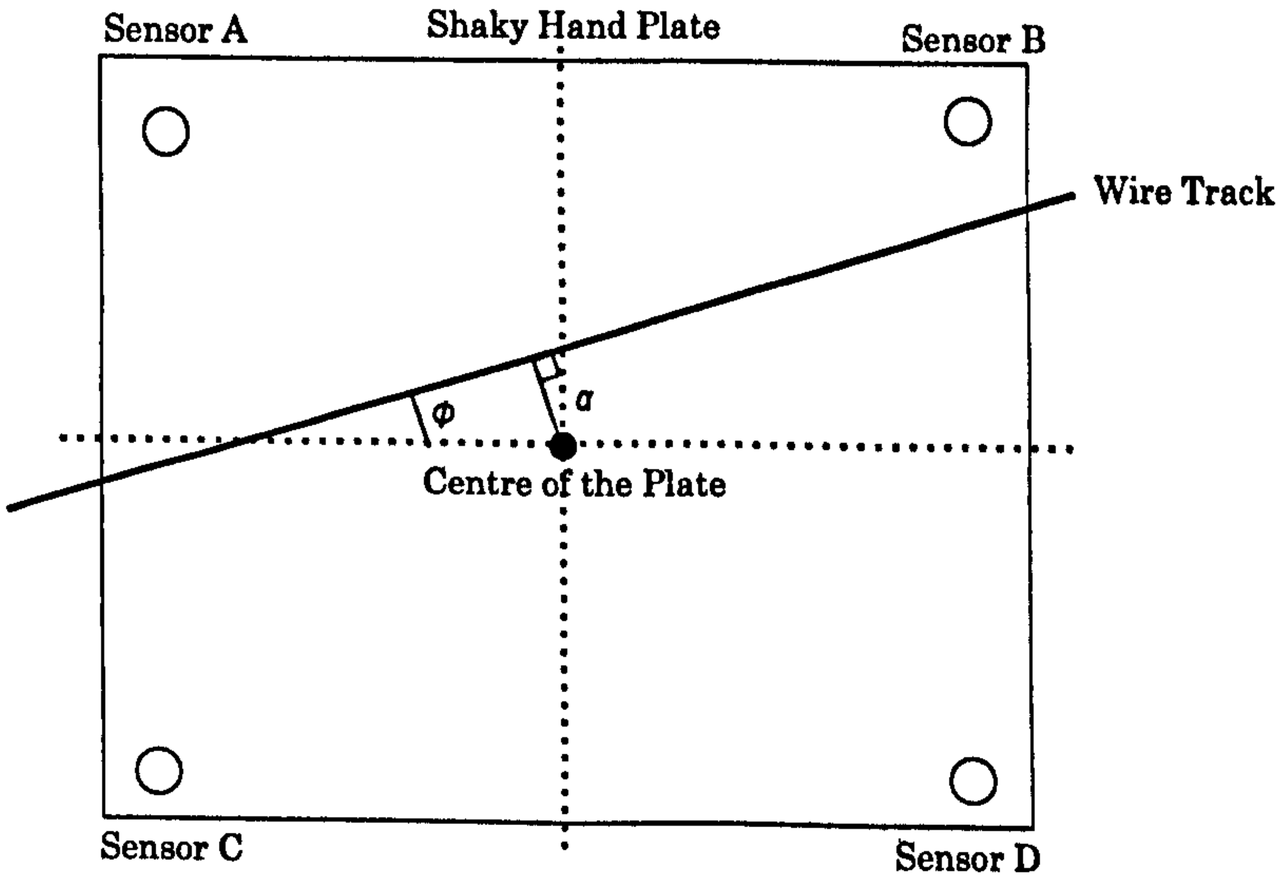


Figure 5.2: Four sensors on the Shaky Hand plate and the offset terms

The lateral offset, α is obtained in terms of the voltage V_α:

$$V_\alpha = (V_A + V_B) - (V_C + V_D) \tag{5.1}$$

and the angle offset, ϕ is obtained in terms of the voltage V_ϕ:

$$V_\phi = (V_A + V_D) - (V_B + V_C) \tag{5.2}$$

The terms V_A, V_B, V_C and V_D used in Equations 5.1 and 5.2 are the output voltages from the sensors A, B, C and D on the plate respectively. The equations assume that all the sensors function correctly and output appropriate signals. However, they become invalid when one or more sensor fails. Thus, it is useful to have not only these equations but other offset error sensing solutions which utilise less than four sensor outputs yet still provide a reasonably accurate estimation of the two offset errors. Assuming, for now, that the Shaky Hand is able to detect the faulty sensor(s) and subsequently select appropriate offset error sensing solutions according to the sensor fault condition, the Shaky Hand would then be able to continue to operate, perhaps with degraded performance. The coil sensor outputs are normally non-zero, positive values so we reasonably assume that under failure conditions, the sensor outputs are reset to zero.

5.1.3 Brief Introduction to Cartesian Genetic Programming

Genetic Programming (GP), first introduced by Cramer [15], used GA to evolve tree-structured and string-based computer programs. GP is a search technique inspired by biological evolution to evolve computer programs in a population according to a fitness measure defined by the user. Traditionally, in the tree representation, every tree node has an operator function and every terminal node has an operand. Koza [7][8] later on applied GP to simple tasks. The evolution process normally involves crossover and mutation. With Cartesian Genetic Programming (CGP) [13], computer programs in a population are mapped to directed graphs rather than trees and the genotype is represented by a string of integers rather than the high level program statements often favoured by a tree-structure GP. CGP was developed by Miller [13][10] for the purpose of evolving digital circuits, and has proven to be highly efficient in learning Boolean functions over standard GP methods. This technique originally only involves mutation. Important benefits of CGP over standard GP methods are that the evolution is bloat free and has faster evolution through neutral search [5]. Bloat free,

because the length of a genotype is fixed and its manipulation does not add extra integers. Neutrality, refers to the presence of multiple genotypes of the same fitness in a population. This has a potential for changing the future genotype dynamically [14][18]. The genotype in CGP represents connections and functionalities of a rectangular array of primitive functions (Section 5.2) and a large amount of redundancy is involved. Some of the genotype, including functional block nodes and input nodes, are not used in the graphs and have the potential to add neutrality. The inactive nodes may undergo neutral change and become activated later. The work demonstrates that control applications can benefit from the use of CGP and that it can be used in practice. The algorithms with sensor fault tolerance are evolved. Parameters and methods used in CGP are explored in order to establish a method to evolve generic solutions. CGP has not been used in the field of sensor fault tolerant control application. Other applications of CGP which have been explored include digital circuit design [12], image filter and its implementation in FPGA [16][9], artificial life [5], bio-inspired developmental models [11], and evolutionary art [1][4].

The structure of this chapter is as follows: Section 5.2 introduces the idea and the general structure of CGP. Section 5.3 describes how CGP is adapted for the Shaky Hand problem. A novel way of evolving the generic solutions using special fitness-based training sets is represented here as a means of improving evolvability and avoiding embodiment problems. The evolved solutions are discussed and fully analysed in Section 5.5. They are then implemented in simulation before applying the solutions to the real Shaky Hand system. This is shown in Section 5.7.

5.2 Cartesian Genetic Programming

5.2.1 CGP Structure

CGP is embodied by linear integer strings, each integer representing program input/output connection nodes and functionalities of a rectangular array of functional blocks. All inputs, whether primary data, node inputs, node outputs, or program outputs, are sequentially indexed by integers. The functional blocks are also separately indexed.

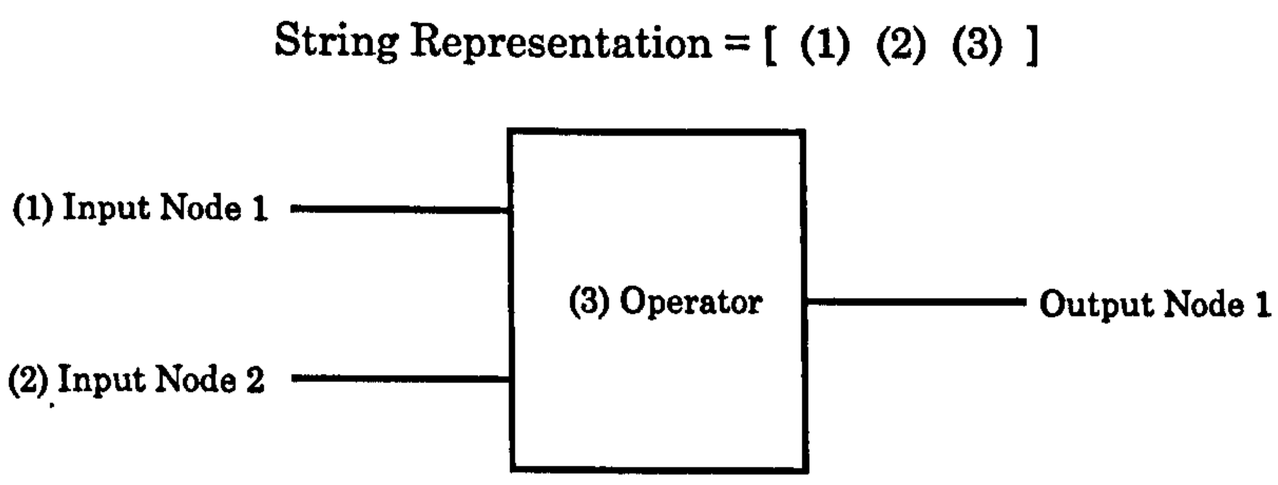
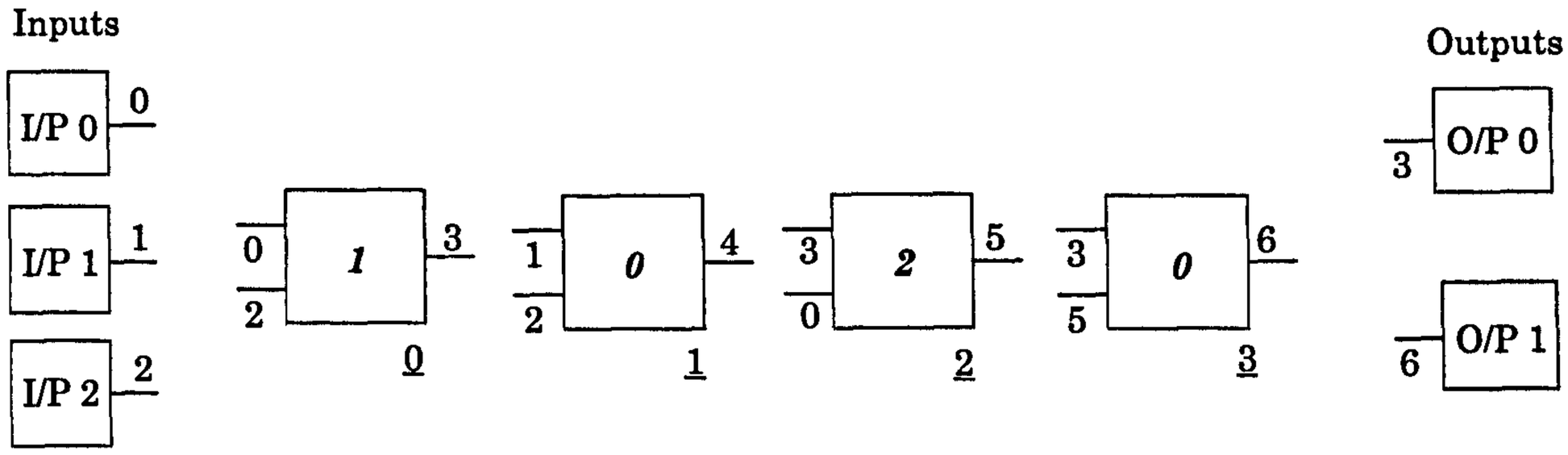


Figure 5.3: A single CGP functional block representation

Figure 5.3 shows a general form of a CGP functional block. In this work, every functional block has two input nodes and one output node, but multiple inputs/outputs nodes are possible in general. In the example shown in Figure 5.3, the first two integers of the string representation are node connections either from other functional block output nodes or from primary input data. The third integer represents an operator array index. The inputs are processed using this selected operator. The genotype of a CGP program is a concatenation of these integers. The following example illustrates. If the genotype for a program is:

Genotype: 021 120 302 350 3 6

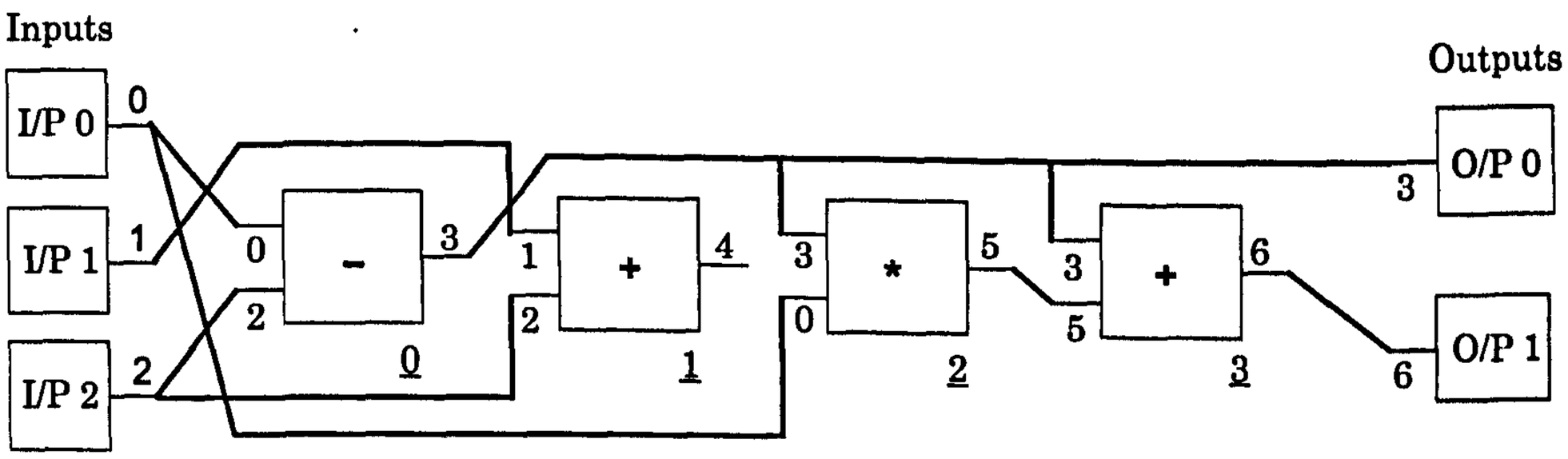
then the program can be mapped onto a directed acyclic graph as shown in Figure 5.4. The program assumes feed-forward connection only. The operators used here are addition, subtraction, multiplication and division and are indexed respectively. There are three external inputs (I/P0, I/P1 and I/P2), four functional blocks (0,1,2 and 3) and two external outputs (O/P0 and O/P1). Each functional block is described by a three integer sequence in the genotype, with the last two genotype integers representing the program outputs.



[*0* - *4*] = Operator [+, -, *, /]
[0 - 3] = Functional Block Indices

Figure 5.4: An example CGP structure 1

The connections are visualised in Figure 5.5. Among the functional blocks, the output from the second functional block is unconnected (node 4). However, this node may be connected in future generations via mutation and the current connected nodes may similarly become unconnected. Only the nodes connected to the program output directly or indirectly, are evaluated for fitness. The evaluated outputs can be described in terms of an equation as shown in Figure 5.5. O/P0 and O/P1 share a common node value (node 3).



O/P0 = 3
= 0 - 2
= I/P0 - I/P2

O/P1 = 6
= 3 + 5
= (I/P0 - I/P2) + (3 * 0)
= (I/P0 - I/P2) + ((I/P0 - I/P2) * I/P0)

Figure 5.5: An example CGP structure 2

5.2.2 CGP Evolutionary Process

The CGP evolutionary process can be summarised as follows:

1. Generate N genotypes at random to form a population.

2. Evaluate the fitness of all associated phenotypes in the population.

3. Determine the best genotype and assign it as *current best.*

4. Mutate the *current best* to generate N-1 *offspring* genotypes. The *current best* and the *offspring* form the new population of N genotypes.

5. Return to 2 and repeat the procedure until a stopping criterion is reached.

Fitness is a measure of how good a produced phenotype is, compared with the given requirement. Phenotype fitness is evaluated for each genotype. Normally, the genotype with the best fitness is promoted as the parent for the next generation and is point mutated in order to produce offspring. Point mutations are the alteration of genes in genotype at random, provided that the values satisfy the restrictions given by the program. When the program output achieves a certain fitness level or reaches a user-defined number of generations, the evolutionary process is stopped. CGP evolution is stochastic, therefore each run can perform differently.

In the process of optimisation, CGP aims to maximise fitness. An alternative equivalent process aims to minimise a *cost function.* In this thesis, the CGP process objective, which is described at length in the following sections, is to minimise a cost function, where the optimal value is zero. In future discussions, where fitness is discussed, it is at a maximum where cost is zero, and is merely a qualitative description of the degree to which the cost function is reduced towards a zero value.

5.3 Cartesian Genetic Programming For The Shaky Hand

A novel evolutionary programming approach to generating offset error voltages in the presence of sensor coil failure is now presented. The pseudo-code of the program is firstly presented, followed by the detail description of the program. This includes the exploitation of the training data sets that avoid over-fitting, which ensures that the resultant algorithmic solutions are generic and therefore will work with any wire track shape. A systematic study of the effect of CGP is also carried out to establish the best method.

5.4 Pseudocode for the Shaky Hand CGP

The pseudo-code of the program which the author developed to produce the novel solutions is;

1: Generate an initial solution of 301 integer long{This is a parent solution}
2: **while** Generation number is less than maximum number & best cost is not 0.01 **do**
3: Fill the population with the copy of the parent solution {one parent and four offsprings}
4: Apply point mutation to the offspring
5: **if** Modulus of (Generation Number, 1000) is 0 **then**
6: **if** This is the first time the program goes through the *while* loop **then**
7: Select two starting data points for y_1 and y_2 each {Input selection}
8: **else**
9: Shift first y_1 data to right by 10{First Sliding Window}
10: Shift second y_1 data to right by 10{Second Sliding Window} {Keep both y_2 data fixed}
11: **end if**
12: Select 100 consecutive data from the starting data and calculate α and ϕ from each input data set selected
13: Calculate the appropriate sensor signals V_A, V_B, V_C and V_D for each input set
14: Calculate the corresponding V_α {They are the expected value if all the sensors are working correctly}
15: **if** There are any failed sensors **then**
16: Set the failed sensor outputs = 0
17: **end if**
18: **end if**

```
19: for all the solutions in the population do
20:     Insert the input data sets
21:     For each data set, evaluate the sum of the cost.
22:     Evaluate the mean cost.
23: end for
24: Select the best solution out of the population and promote it as the parent solution for
        the next generation
25:     Increment the generation number by 1
26: end while
27: Print the best solution obtained
```

5.4.1 Embodiment

A system can be said to be embodied in its environment if it can both sense and affect it. The environment provides feedback to the system, adding a coupling effect of some sort. Brooks [2] had a strong influence on the idea of embodiment. He noted that real intelligence is situated in the world, not in disembodied systems such as expert systems. In a computational system, Stepney [17] states that embodiment is provided by the coupling between the computational system and its environment, and is a rich complex feedback process. Inputs to the system from the environment are through its sensors, and it outputs to the environment through its actuators.

For the Shaky Hand CGP, the solutions are not evolved through the physical environment but a virtual one. The environment for the Shaky Hand case is the wire track shape provided by training data sets. This environment must be sufficiently open to avoid over-adaptation, i.e. its behaviour must not be captured in some predefined description. A closed environment would over-specify the system, shaping its behaviour to that particular environment only, creating solutions that work on a particular wire track only. This case had to be avoided, and so a virtual environment was designed to achieve sufficiently rich interactions between the system and itself. Methods used in the Shaky Hand CGP to achieve such an open and rich environment are discussed below.

Input Data Sets

Input data sets to the program are the signal values from sensors A, B, C and D on the plate. The sensor values should be consistent with the correct operation of the Shaky Hand system.

The signals must be continuous and without repetition, representing realistic input sets that do not present a closed environment. In order to satisfy these criteria, a model relating α and ϕ with the physical dimension of the plate was created, as illustrated in Figure 5.6.

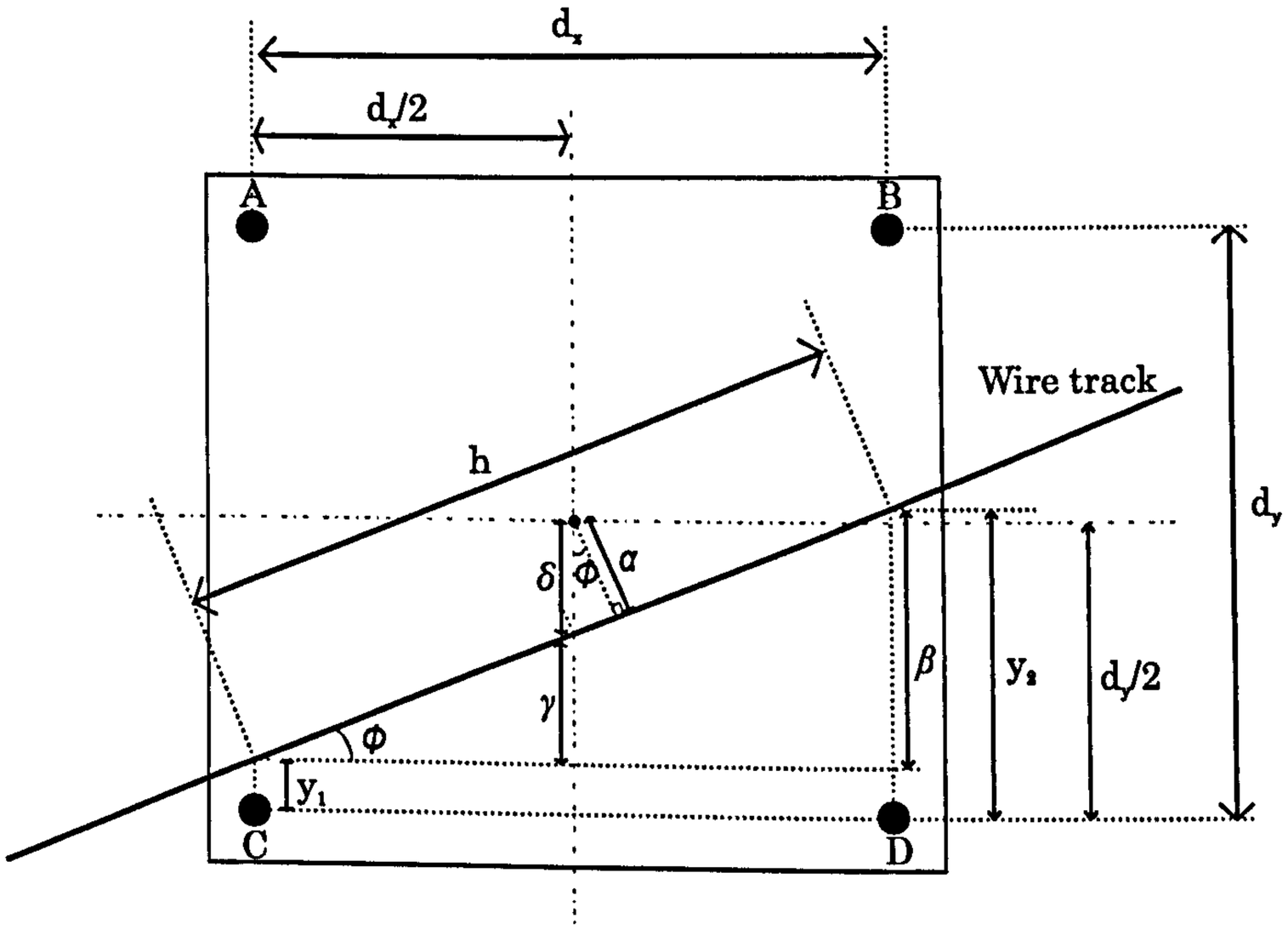


Figure 5.6: Plate model to create CGP input sets

By altering the wire position at the edge of the plate continuously, through y_1 and y_2, and without repetition, α and ϕ are in turn altered continuously and without repetition. The sensor signals are then generated as required.

From Figure 5.6

$$\tan\phi = \frac{\beta}{d_x} \tag{5.3}$$

$$\therefore \phi = \arctan(\frac{\beta}{d_x}) \tag{5.4}$$

Since

$$\beta = y_2 - y_1 \tag{5.5}$$

The angle offset, ϕ, is

$$\phi = \arctan(\frac{y_2 - y_1}{d_x}) \tag{5.6}$$

By scaling,

$$\gamma = \frac{\beta}{2} \tag{5.7}$$

$$= \frac{y_2 - y_1}{2} \tag{5.8}$$

δ is defined as

$$\delta = \frac{d_y}{2} - \gamma - y_1 \tag{5.9}$$

Therefore,

$$\delta = \frac{d_y}{2} - \frac{y_2 - y_1}{2} - y_1 \tag{5.10}$$

$$= \frac{d_y - y_2 - y_1}{2} \tag{5.11}$$

and also,

$$\cos\phi = \frac{\alpha}{\delta} \tag{5.12}$$

$$\alpha = \delta \cos\phi \tag{5.13}$$

α in Figure 5.6 has a negative sign by convention and from Equations 5.11 and 5.13,

$$\alpha = -\frac{d_y - y_2 - y_1}{2} \cos\phi \tag{5.14}$$

Equation 5.14 can be further simplified.

$$h = \sqrt{d_x^2 + \beta^2} \tag{5.15}$$

$$= \sqrt{d_x^2 + (y_2 - y_1)^2} \tag{5.16}$$

$$\cos\phi = \frac{d_x}{h} \tag{5.17}$$

$$= \frac{d_x}{\sqrt{d_x^2 + (y_2 - y_1)^2}} \tag{5.18}$$

Substituting Equation 5.18 into Equation 5.14 gives

$$\alpha = -\frac{d_y - y_2 - y_1}{2} \frac{d_x}{\sqrt{d_x^2 + (y_2 - y_1)^2}} \tag{5.19}$$

$$\therefore \alpha = -\frac{d_x(d_y - y_2 - y_1)}{2\sqrt{d_x^2 + (y_2 - y_1)^2}} \tag{5.20}$$

The continuous and non-repeating y_1 and y_2 data are obtained using the Matlab *polyfit* function. This generates a polynomial of predefined order that fits data points provided by an user. In this case, two 9th order polynomials were generated to express y_1 and y_2 variations as shown in Figure 5.7. Data points were chosen so that the polynomials varied within the range constrained by the physical size of the plate. The order of the polynomials was chosen to provide reasonable smoothness of the curves. The term, Sliding Window, on this figure is explained later. Using the Matlab *polyval* function, 5000 data points each were extracted from the y_1 and y_2 polynomials. They become the *data bank* for the input sets of the Shaky Hand CGP. Equations 5.6 and 5.20 are then used to convert the wire positions provided by the data bank into the offset errors. The offset errors are then converted into the sensor voltages as described by Equation 2.61. Continuous sequence of sensor voltages become the input sets for the Shaky Hand CGP.

Sliding Windows

The Shaky Hand CGP takes in 100 consecutive input data points per generation from each of the y_1 and y_2 data banks. The program selects the starting data points at random for both y_1 and y_2 data displayed as A_0 and B_0 in Figure 5.7. 100 consecutive data points from the starting data points, enclosed by the Sliding Window are selected and then converted into the sensor signals V_A, V_B, V_C, and V_D as shown previously using α and ϕ. The combination of randomly and independently selecting the start points A_0 and B_0 over these two long data sets means that the training data sets are realistic yet, to all intents and purposes, unrepeated over very many experiments. The Sliding Window on y_1 data bank is shifted by 10 data points to the right every 1000 generations, and the window on y_2 data bank is fixed. (see the pseudo-code lines 7 to 14 in Section 5.4). In other words, the input sets to the CGP are kept the same for 1000 generations and are then modified over 10% of their range. Using the modified input sets, the solution is evolved again. This gradual rather than an abrupt change in input sets, helps the evolution to migrate towards generic solutions. The window on y_2 data bank is fixed, yet, the variation in input sets is still enormous.

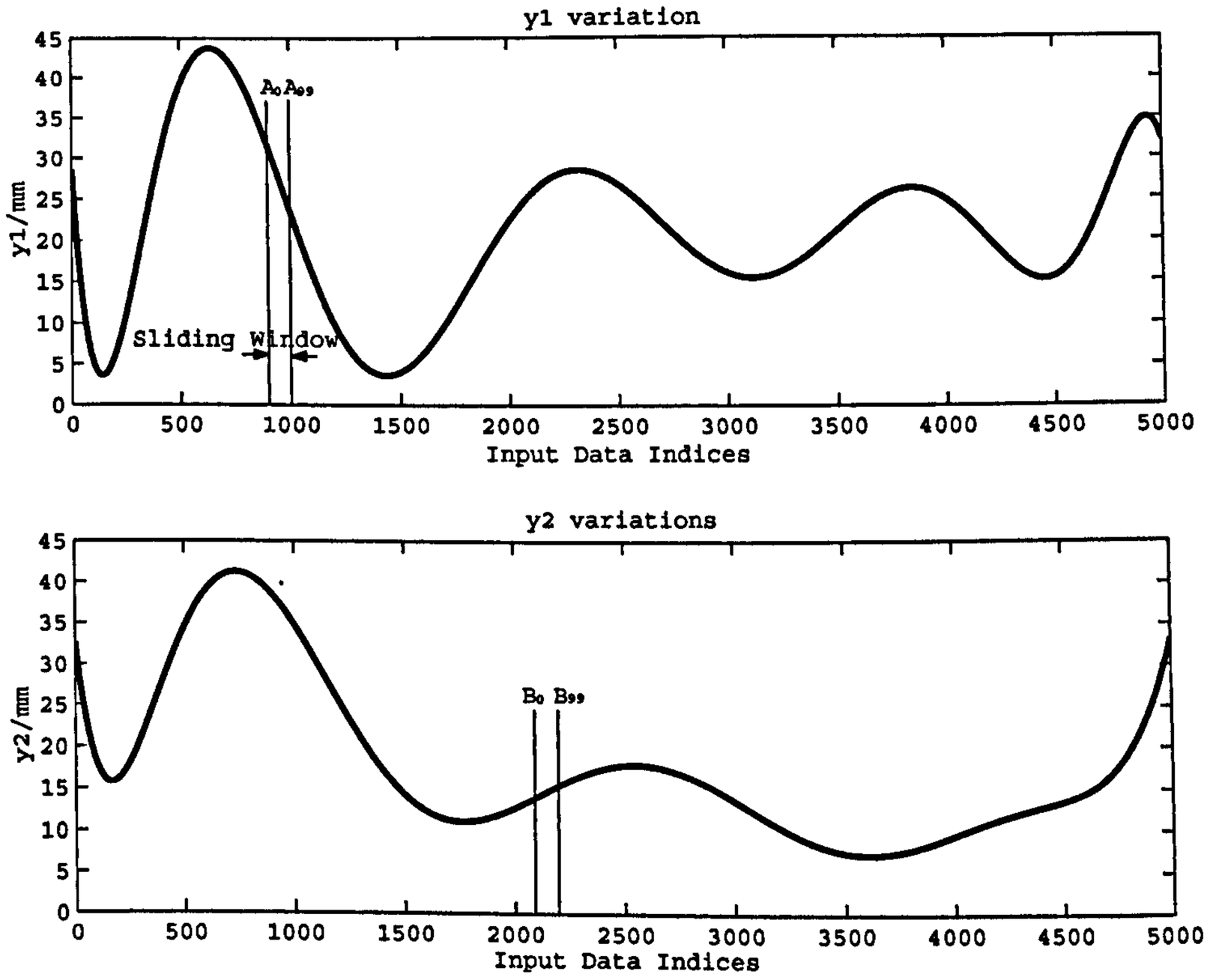


Figure 5.7: y_1 and y_2 data variations

Evaluation of the Cost Function

Figure 5.8 illustrates how the cost is evaluated using the evolved and the original algorithm outputs. The output sets from the original algorithms are defined as ideal output sets in this figure.

To simulate a sensor failure, one member of the input sets is forced to be zero. It is reasonably assumed that the Shaky Hand has a fault detection system such that when a failure is detected the signals from the failed sensors are nulled. Using these input sets, solutions are evolved. (the pseudo-code lines 15 to 24). For each generation, the output sets from all the solutions in the population are compared with the ideal output sets to determine the best one. The generation of the new population ends when a stopping criterion set by the user is met.

In theory, the best possible cost value is 0, which would mean the successful evolution of identical output sets to the fault-free originals. However, the sensor failures are expected to cause deteriorations and 0% error may not be achieved. Therefore the criterion for the convergence was initially set to 0.01. (the pseudo-code line 2). If the cost is less than 0.01, the solution is considered to have the equivalent response to the original algorithm. i.e. the outputs produced by the current best solution in the population for 100 incoming data have

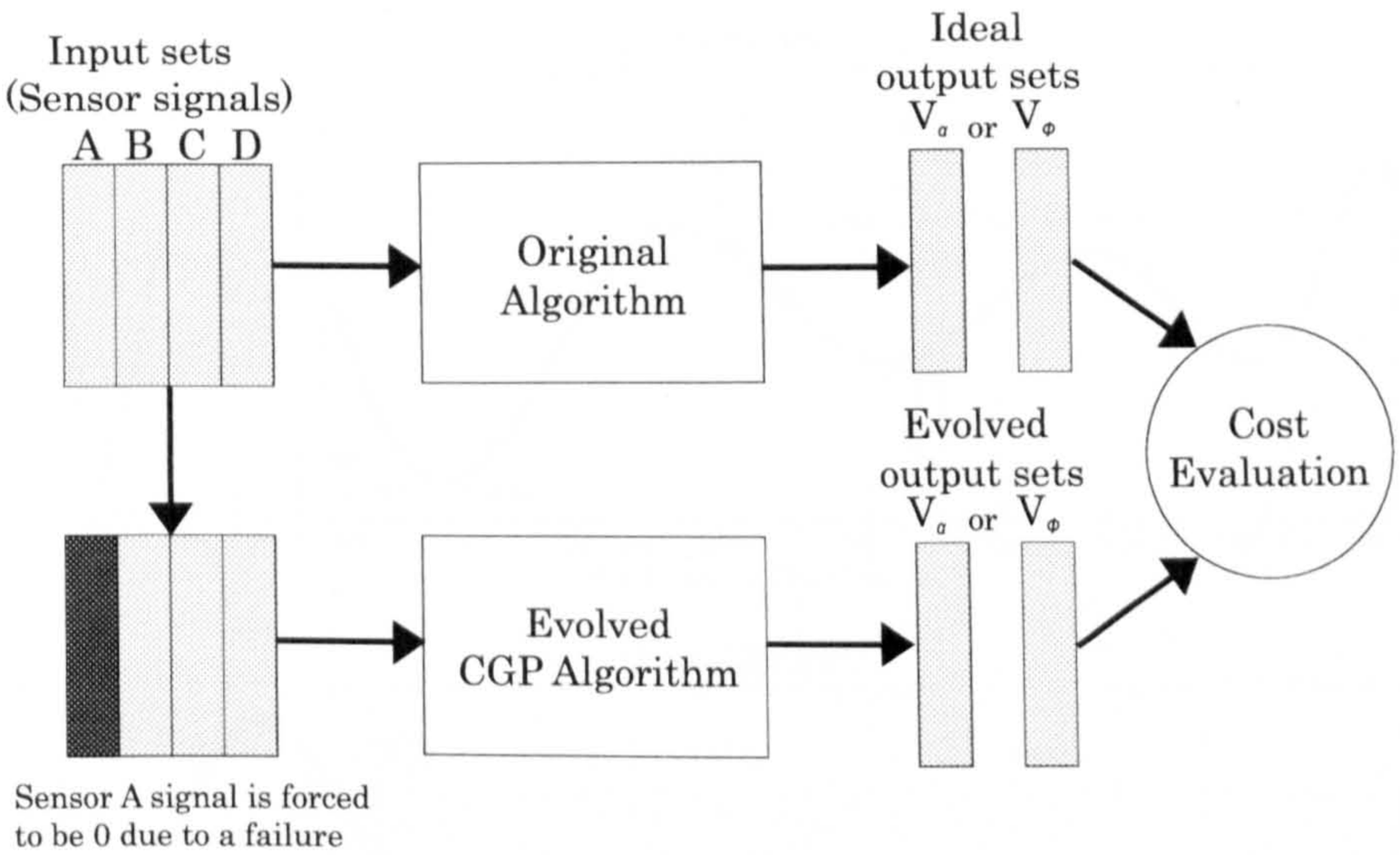


Figure 5.8: Comparing the ideal and the evolved solutions

less than 1% error. A 1% error in measurement would have negligible effect on the tracking along the wire by the Shaky Hand plate. The negative feedback control would act on this error as if it was a disturbance. The disturbance rejection properties of negative feedback are well known and documented in all good elementary control engineering text books.

Multiple Virtual Wire Tracks

In a further effort to ensure the genericity of CGP solutions, training sets were further modified. A generic solution means it works on any given input set. Therefore, two different input sets were selected and applied to the evolutionary processes. i.e. the cost of a solution is evaluated using two totally different virtual wire tracks. The two starting data points are chosen from each of the y_1 and y_2 data bank as shown in Figure 5.9. (the pseudo-code line 7).

Fitness

The two original equations used to calculate the different offset errors given by the Shaky Hand plate were described in Equations 5.1 and 5.2. They are valid only when all the sensors are functioning correctly, as they utilise all the sensor outputs. The outputs of the evolved solutions that follow are required to generate values as close as possible to the original but in the presence of zero value coil voltages. The degree of closeness is measured by comparing the output values from both solutions using same input sets. Equations 5.21 and 5.22 show

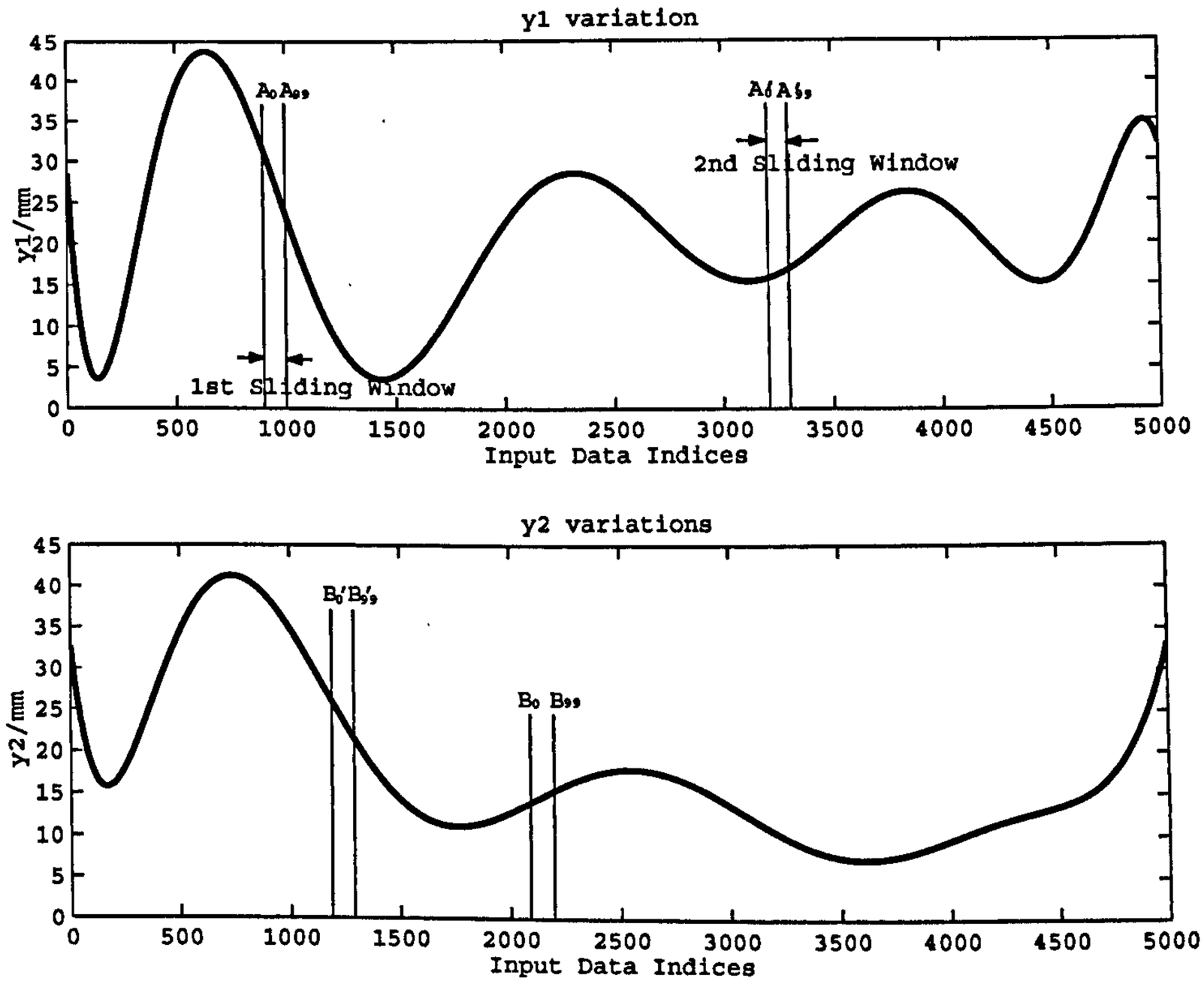


Figure 5.9: y_1 and y_2 data variations with two sliding windows each

how the fitness of each offset is obtained, by minimising a cost, J. The best possible fitness therefore incurs zero cost. The cost functions are:

$$J_{V_\alpha} = \left| \frac{V_{\alpha_{ideal}} - V_{\alpha_{evolved}}}{V_{\alpha_{ideal}}} \right| \tag{5.21}$$

$$J_{V_\phi} = \left| \frac{V_{\phi_{ideal}} - V_{\phi_{evolved}}}{V_{\phi_{ideal}}} \right| . \tag{5.22}$$

If ($V_{\alpha_{ideal}} = 0$) or ($V_{\phi_{ideal}} = 0$), then the equations are modified to:

$$J_{V_\alpha} = | V_{\alpha_{ideal}} - V_{\alpha_{evolved}} | \tag{5.23}$$

and similarly,

$$J_{V_\phi} = | V_{\phi_{ideal}} - V_{\phi_{evolved}} | \tag{5.24}$$

This normalised cost function was used because of its property of forcing the evolution of

good solution algorithms when the actual offsets, α and ϕ, are very small. Since the loop closure of the control system will tend to drive the offsets towards zero (good wire tracking), it is important that the sensor failure algorithms should work best under tight tracking.
There are two outputs from the Shaky Hand CGP. One is the lateral offset error voltage V_α and the other is the angular offset error voltage V_ϕ as mentioned. The solutions for obtaining V_α and V_ϕ were evolved using separate programs in order to simplify the problem. The reasoning behind this is that if both solutions were evolved using a single program, there may be a situation where one of the solutions is good but the other is not, with the condition masked by a combined total fitness. The evolution would then continue on both solutions, with a real risk of the good component being lost. To avoid this, a multi-objective evolution using Pareto optimisation technique could be employed. However, a single objective evolution was preferred for its simplicity and to allow faster evolution.

5.4.2 CGP Structure and Parameters

The program method can be described as follows:
The program takes in 100 input data sets per generation. For each input set, an output (either V_α or V_ϕ) is obtained and the cost is evaluated using the method mentioned before. The mean cost for the 100 input sets is then obtained for all the solutions in the population. The best solution is selected as the parent for the next generation. If the cost is less than 0.01 the solution is considered to have the equivalent response to the original algorithm as stated in Section 5.4.1. This process is repeated until the best cost is less than 0.01 or when the user-defined number of generations is reached. The whole process is then repeated 60 times for each V_α and V_ϕ solutions to produce 60 possible candidates for the generic solution.
The other parameters considered in CGP are

- Selection method.
- Population size.
- Number of functional blocks and their input/output complements.
- Mutation rate.
- Crossover effect.
- Number of operators and their types.

- Number of generations.

In order to select suitable parameters for the Shaky Hand CGP, several experiments were carried out.

Selection Method

On the grounds of simplicity, the chosen selection method is elitism with neutral walk [18]. This encourages genetic drift. Here, the best of the population is selected as a parent for the next generation. If an alternative genotype of equivalent fitness/cost exists, but with a different genetic complement to a parent, this is selected in preference. A consequence of this is that subsequent generations may see dramatic change for the better in the evolved solutions.

Population Size

A series of experiments was carried out to find out a suitable population size. The results are displayed in Table 5.1. POP size refers to the population size and STD stands for the standard deviation. A program to search for the original V_α error detection algorithm (Equation 5.1) was made for different population sizes assuming all four sensors working. The mean number of generations to complete the task and their standard deviations were compared for population sizes of 5, 10 and 20. The corresponding actual mean computation time for 100,000 generations was also measured. The average was taken over eight runs, since more structurally complex solutions took a longer time to evaluate their costs. To calculate the mean number of generations and their success rates, 25 runs were performed for each population size. The program ran on a conventional PC (Windows XP, Pentium 4 CPU 2.8GHz, 512MB RAM). The other parameters for this experiment were chosen to be as follows:

- Number of functional Blocks = 30 (this determines the length of the integer representation (genotype).)
- Number of input sets per generation = 100
- Mutation rate = 1.1% (1 gene out of 91 in the offspring genotype is mutated for each generation.)
- Number of operators = 4:[addition, subtraction, multiplication, division]

- Number of generations stops at 100,000 if convergence is not achieved.
- Selection method = Elitism
- Number of runs = 25

POP size	% success rate	Mean no.of generations	STD	Mean computation time/s
5	92	33386	26014	43
10	96	29559	24414	78
20	100	26118	24401	141

Table 5.1: Population size experiment

The results indicate that the larger the population size, the fewer generations are required and the success rate shows that a larger population size can evolve the solution in fewer generations. However, this test is unfair to the small population sizes because the total number of evolution in the population within the given maximum permitted generation number for smaller population size is less. Comparing the population size of 5 with 20, if the generation limit for population size of 5 was four times larger, the success rate is likely to increase. The standard deviation values show the number of generations required to evolve a particular known solution can be very small for any population size. Moreover, the mean computation time increases significantly with population size compared with the reduction in mean number of generations. In this test experiment, only four operators and thirty functional blocks were used. However, in the Shaky Hand CGP, the number of operators and functional blocks as well as the maximum number of generations are increased adding considerably to the computation time. Taking into account the total number of evolution in the given population size, the standard deviation, and the computation burden, the population size of 5 is the most effective size. Therefore, 4+1 CGP (Four offspring and one parent) was employed in the Shaky Hand CGP.

Number of Functional Blocks

Two input, single output functional blocks of the type depicted in Figure 5.3 were arranged into an array structure. The number of rows and columns within array define the unconnected topology of the CGP. It was considered that a (1×100) array presents the most natural topology. It supplies enough blocks to afford the CGP with adequate complexity and, using only feed-forward connections, a rich variety of multi-sensor inputs/single offset voltage output algorithms could be evolved. The number of functional blocks determines the length of

the integer representation. Each block contains three genes representing two incoming node connections and one operator type, combining this with the output. There are 301 genes in total. This genotype is sufficient to produce relatively complex solutions.

Mutation

The mutation operation generates random changes in the genotype. The mutation rate is chosen to be small, 1%. In any generations, this rate corresponds to the mutation of three genes out of the 301 genes which comprise an evolved solution.

The mutation rate is chosen to balance premature convergence against disruption of successful convergence as shown in Figure 5.10. Premature convergence gives a globally suboptimal result and occurs when a mediocre suboptimal solution happens to have particularly good fitness relative to the other solutions in the population at the time. Mutation prevents this convergence [7]. The genetic drift effect [18][14] can also help the dynamic change in the population as mentioned before. There is no way of knowing when the solution has converged since the correct solution is unknown. Therefore a small mutation rate was chosen to allow gradual change in the offspring rather than rapid diversion. Rapid diversion may also give rise to disrupted convergence.

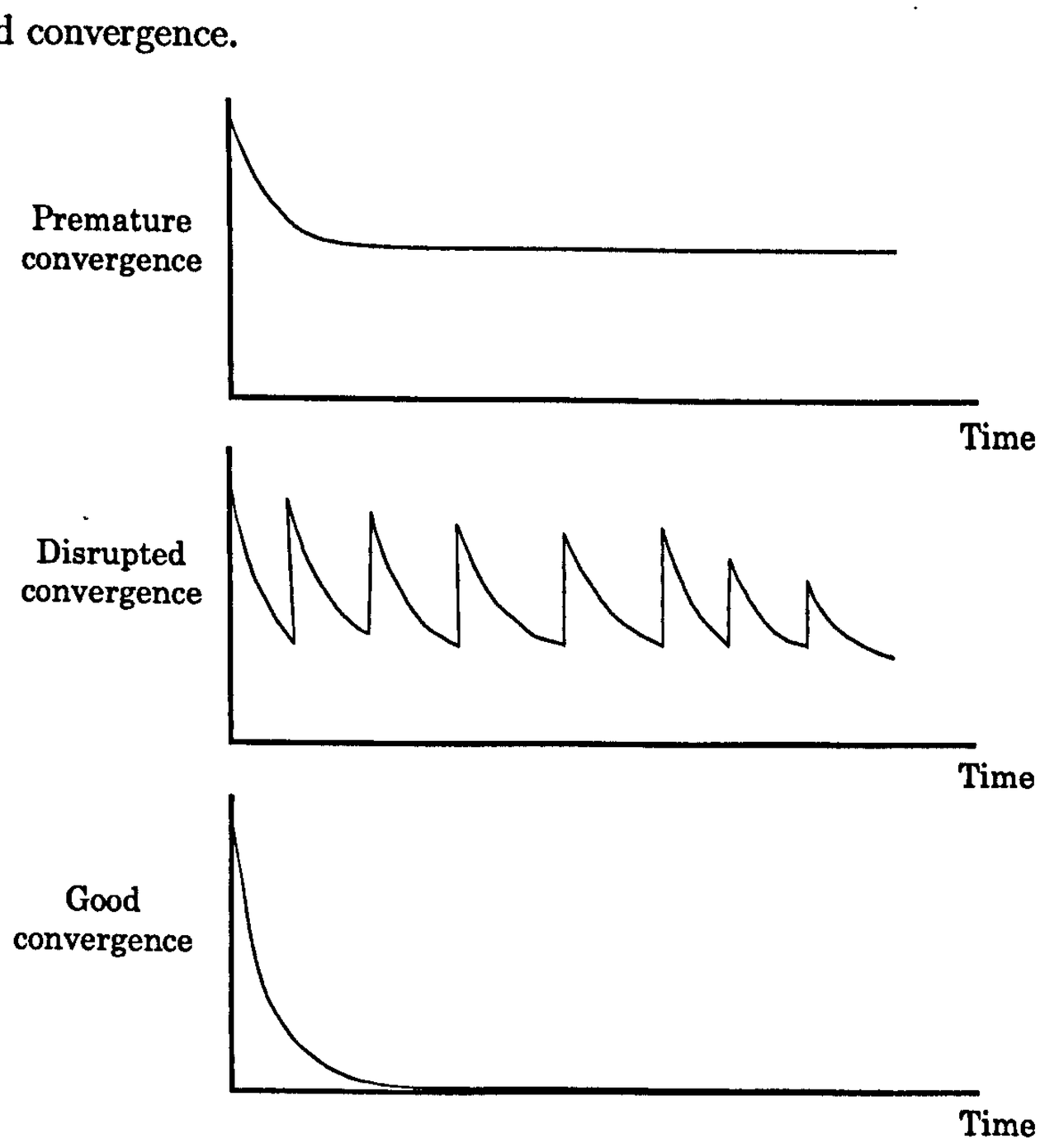


Figure 5.10: Convergence

Crossover Effect

Crossover is a technique in which elements of the genotype of parents are swapped as part of the evolutionary process. For example, as shown in Figure 5.11 one-point crossover would select a single point on both parent genotypes and swap all the genes beyond that point to produce offspring. Two-point crossover select two points on the parent genotypes and all genes between the two points are swapped for the reproduction.

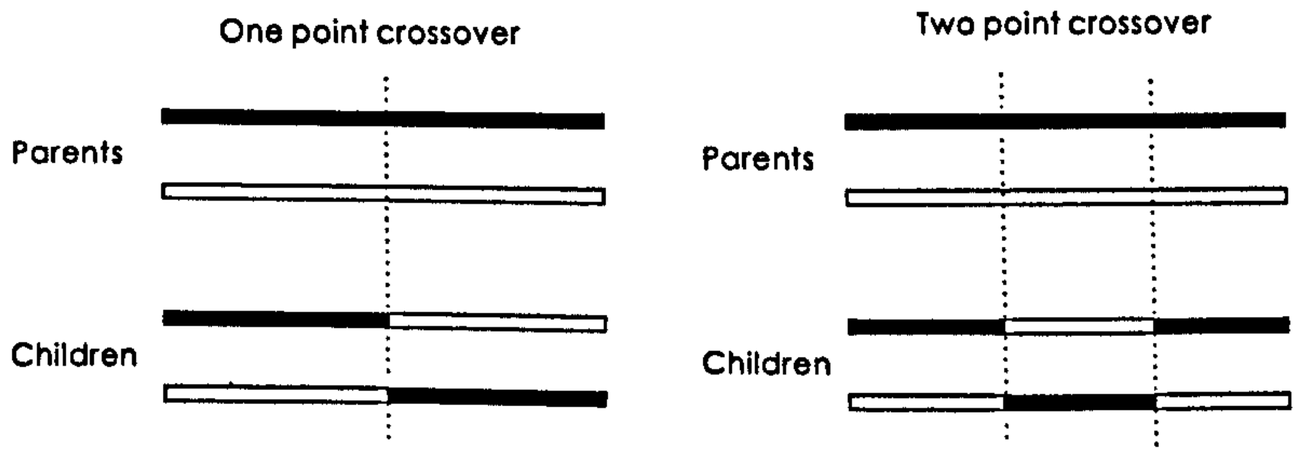


Figure 5.11: Crossover

In general, CGP is implemented using mutation only [13][10] and Clegg et al [3] has also shown that the use of traditional crossover techniques on CGP hinders performance rather than improves it. However, she has introduced a new crossover technique, inspired by the real-valued crossover operator found in real-valued GAs, which can be incorporated into CGP. This new crossover technique requires a modification to the CGP representation. A new level of encoding is introduced into the CGP genotype, which represents the directed graph as a fixed length of real-valued numbers rather than a string of integers. Each real-valued number represents a single gene in the genotype and has a range between 0 and 1. The genotype still works in the same way as before such that first two real-valued gene encodes the inputs of the functional block and the third one encodes the operator types. Let us look into an example shown earlier in Section 5.2.1. Again, the program is visualised in Figure 5.4. The program has three inputs and two outputs and has four operator types. The real-valued genotype is expressed as

Real-valued genotype: 0.1 0.75 **0.33** 0.4 0.58 **0.05** 0.7 0.15 **0.6** 0.61 0.9**0.22** 0.48 0.84

This can be decoded into the integer-based genotype as

Integer-based genotype: 021 120 30**2** 350 3 6

The decoding process is achieved using Equations 5.25 and 5.26.

$$integer\ based\ gene\ (input\ node\ number) = floor\ (gene_i * output\ node\ number) \quad (5.25)$$

$$integer\ based\ gene\ (operator\ types) = floor\ (gene_i * total\ operator\ number) \quad (5.26)$$

The output node number of the first functional block is 4 and the total operator number is also 4. Using the real-valued representation, the crossover can be applied such that the genes of the offspring at selected positions have real values anywhere between the value of the first parental gene and the second parental gene at the same positions. Clegg et al applied this crossover to simple regression problems and the results indicated that it can improve the performance of CGP by speeding up its convergence, especially at the early generations. The author investigated the effect of using this crossover technique to see if this would help in the Shaky Hand CGP.

An experiment to evolve a solution (Equation 5.1) was carried out to see the effect of using the new crossover technique with various gene crossover rates when combined with mutation on various population sizes. The gene crossover rate shows the proportion of genes in the two parental solutions which are applied this crossover. So, if the gene crossover rate is 10%, then 30 gene positions within the 301 genes, which makes up one genotype, are randomly selected, and the crossover is applied to the two parental solutions to create offspring which may then be mutated. In this experiment, the best two candidates in the population pool are promoted as the parental solutions for the next generation. All the experimental parameters are the same as the experiments for determining the population size as described previously in Section 5.4.2. The mutation rate is fixed (1%) but the gene crossover rate varies. The program was modified to adapt the new crossover technique. Runs were repeated 25 times for each gene crossover rate.

The results are shown in Tables 5.2, 5.3 and 5.4. GCOR stands for gene crossover rate. Success rate is defined as the number of times the program produced the expected solution over 25 runs. The mean number of generations to reach the solution was measured as well as the standard deviation (STD). The equivalent mean computation time was also calculated as before. These data are presented graphically in Figures 5.12 and 5.13.

With Population Size = 5:

GCOR/%	Success rate/%	Mean no.of gens	STD	Mean computation time/s
0	92	33386	26014	43
10	96	31319	28558	213
25	92	26496	25192	180
50	96	27578	27237	188
75	96	32367	29462	220

Table 5.2: Crossover experiment with population size = 5

With Population Size = 10:

GCOR/%	Success rate/%	Mean no.of gens	STD	Mean computation time/s
0	96	29559	24414	78
10	96	23550	25770	310
25	92	21137	18167	279
50	92	26434	25536	349
75	92	28797	23320	380

Table 5.3: Crossover experiment with population size = 10

GCOR/%	Success rate/%	Mean no.of gens	STD	Mean computation time/s
0	100	26118	24414	141
10	100	20348	18455	639
25	100	11285	14230	354
50	100	12003	11780	377
75	100	22542	22607	707

Table 5.4: Crossover experiment with population size = 20

From the experiments, the followings can be observed:

- The mean number of generations required is less for the larger population size, thus, giving rise to higher success rate.
- Crossover can reduce the mean number of generations for any population size.
- Using a crossover rate of 25%, the mean number of generations is the smallest for all the population sizes.
- The mean computation time is greatly reduced with mutation only.

Similarly to the experiment for determining the suitable population size, the mean number of generations required to produce the known solutions is less and the success rate is higher with the increase in the population size. However, this experiment is unfair to the smaller population size as the number of evolutions per generation is less for the smaller population

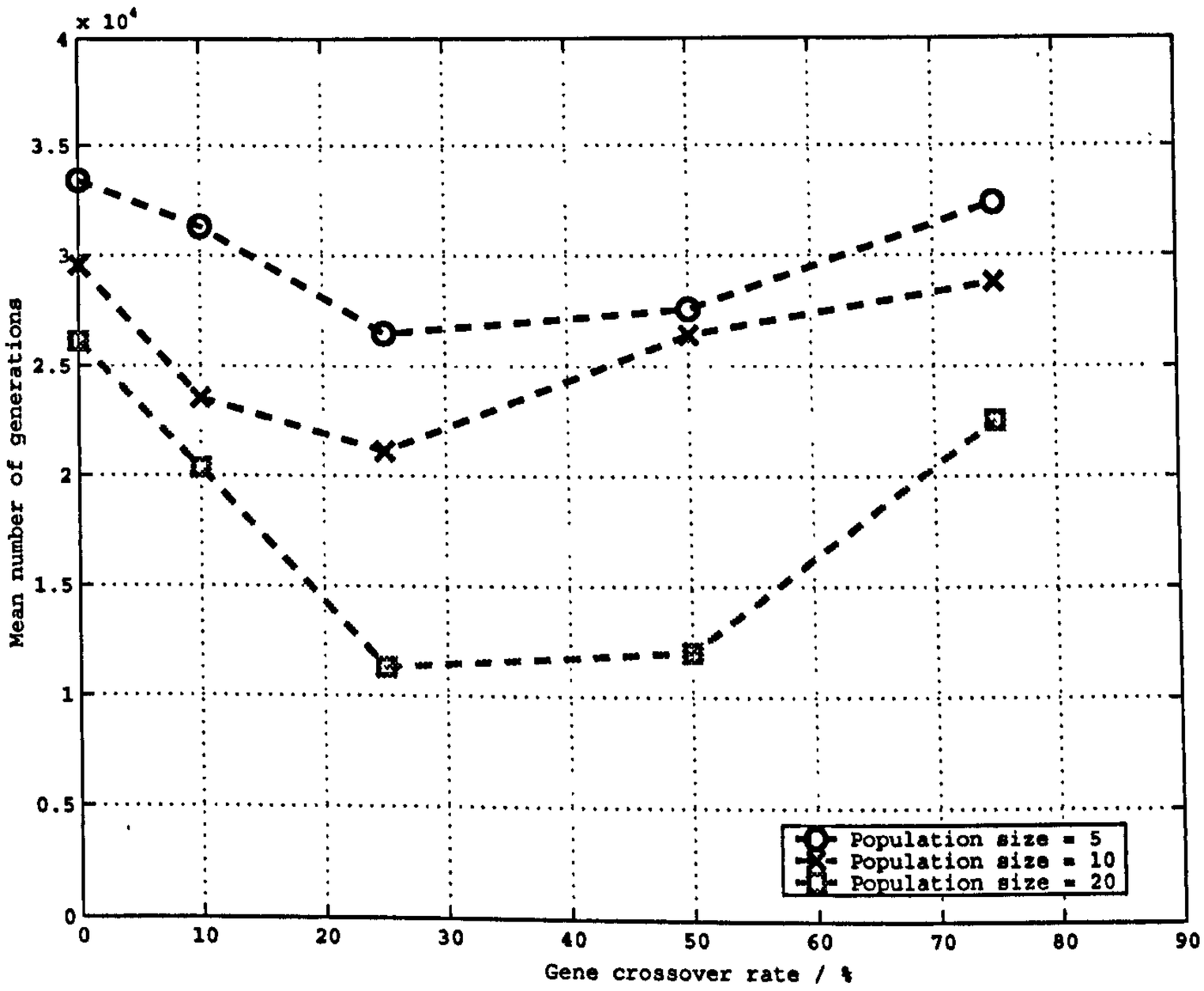


Figure 5.12: Effect of varying the gene crossover rate on the mean number of generations for various population size

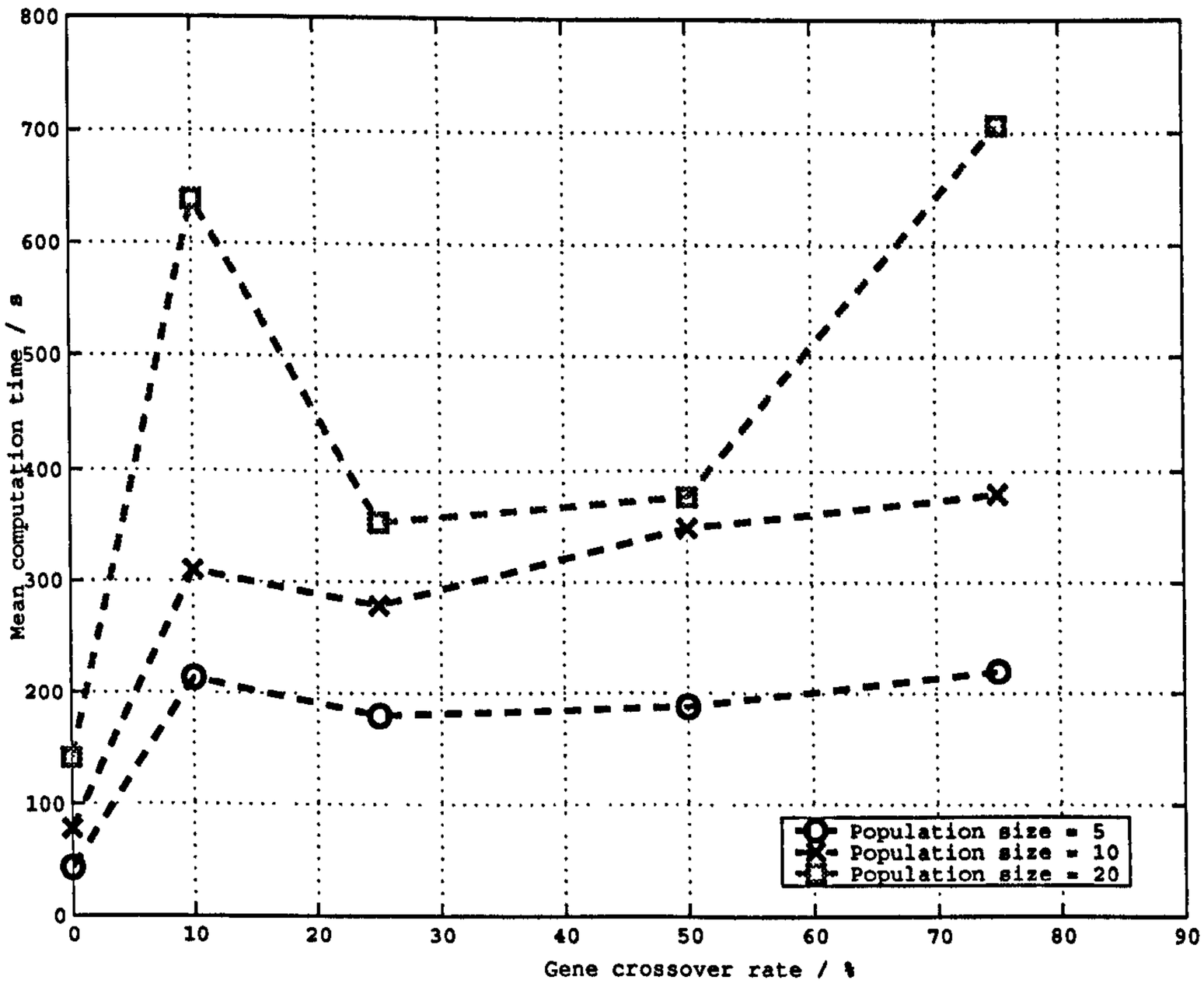


Figure 5.13: Effect of varying the gene crossover rate on the mean computation time for various population size

size. Therefore, it is likely that the success rate for the population size of 5 and 10 would have been higher or even achieved 100% if the maximum permitted generation number was set higher. With the gene crossover rate of 25%, the efficiency of the program in terms of the mean number of generations and the mean computation time seems to increase for all population size. The mean generation number is the fewest with this gene crossover rate and the population size of 20. With larger population size (more than 20), and a crossover rate of 25%, the computation time may even become shorter than the ones for smaller population size. However, the most efficient convergence in terms of the mean computation time uses mutation only with a population size of 5. The huge difference in the computation time between the method using mutation only and the method using mutation and crossover is because of the structural difference in the program. In order to adapt the new crossover technique the program was modified, which slowed down the computation time. The crossover works better with larger population size but the population size experiment showed the population size of 5 is the most suitable for the Shaky Hand CGP. Moreover, having crossover with this population size does not provide significant improvement on the mean number of generations but adds significant amount of computation burden. Table 5.2 shows that the increase in the computation time from the mutation only method to 25% crossover is greater than the reduction rate of the number of generations. Therefore, crossover is not implemented in subsequent Shaky Hand CGP work.

Operators

A wide range of operators is provided including primitive and conditional operators so the evolutions would have variety of choice of the operators. Seventeen operators are used as described in Table 5.5. X_1 indicates the input node 1 of one functional block, X_2 is the input node 2. O is the output node of a functional block. For the evaluation of the output of the solution, exception handling is incorporated into some operators such as a divide by zero. These protected functions are also described in Table 5.5. Conditional operators (operators numbered 10 and 11) allow more complex solutions to be evolved. The conditional operators provide solution choices according to the values assigned to the input nodes. Depending on the situations, a solution can therefore have totally different values and can better adapt to dynamic changes in the environment.

Operator indices	Operator types	Protected functions
1	Addition (O=X_1+X_2)	
2	Subtraction (O=X_1-X_2)	
3	Multiplication (O=$X_1 \times X_2$)	
4	Division (O=$\frac{X_1}{X_2}$)	if X_2=0, O=0
5	Square (O=X_1^2)	
6	Square root (O=$\sqrt{\lvert X_1 \rvert}$)	Use absolute value
7	Reciprocal (O=$\frac{1}{X_1}$)	if X_1=0, O=0
8	Natural log (O=lnX_1)	if $X_1 \leq 0$, O=0
9	$\log_{10}$ (O=$\log_{10} X_1$)	if $X_1 \leq 0$, O=0
10	Max (O=max(X_1,X_2))	
11	Min (O=min(X_1,X_2))	
12	Absolute value (O=$\lvert X_1 \rvert$)	
13	Sine (O=sinX_1)	
14	Cosine (O=cosX_1)	
15	Tangent (O=tanX_1)	if X_1=(n+$\frac{1}{2}\pi$) n=(0,1,2...), O=0
16	Power (O=$X_1^{X_2}$)	if X_1=0, O=0
17	Sign change (O=-X_1)	

Table 5.5: Operators used in the Shaky Hand CGP

Number of Generations

The number of generations was limited to 100,000 initially, with each run being halted earlier if a solution is found. Detailed analysis of the results, which are discussed in Section 5.6.2, shows that the maximum number of generations could be reduced to 50,000.

Program Structure

The program structure for the Shaky Hand CGP is summarised as follows.

1. Generate 5 sets of initial solutions to form population.
2. Select two different starting input data sets for y_1 and y_2 at random, to define multiple virtual tracks.
3. Evaluate the cost of all the solutions in the population.
4. Determine the best solution and assign it as the *current best.*
5. Mutate the *current best* to generate 4 sets of offspring to form new population.
6. Return to step 3 for 1000 generations.

7. Shift both the input data sets by 10 points and then return to step 3 until either 50,000 generations are completed or convergence is achieved.

One solution for obtaining V_α is achieved at the end of this process. This is then repeated 60 times to produce 60 possible candidates for the generic solution. The whole process is then repeated to obtain 60 candidates for the V_ϕ solutions.

5.5 Shaky Hand Sensor Algorithms

Solutions for V_α and V_ϕ were obtained for the following cases (See Figure 5.14 for definitions):

Case 1 : All four sensors working correctly

Case 2 : One sensor failure (e.g. sensor A)

Case 3 : Two sensor failures 1 - Adjacent sensor pair failure (e.g. sensors A and B)

Case 4 : Two sensor failures 2 - Diagonal sensor pair failure (e.g. sensors A and D)

Case 5 : Two sensor failures 3 - Opposite sensor pair failure (e.g. sensors A and C)

Case 6 : Three sensor failures (e.g. sensors A, B and C)

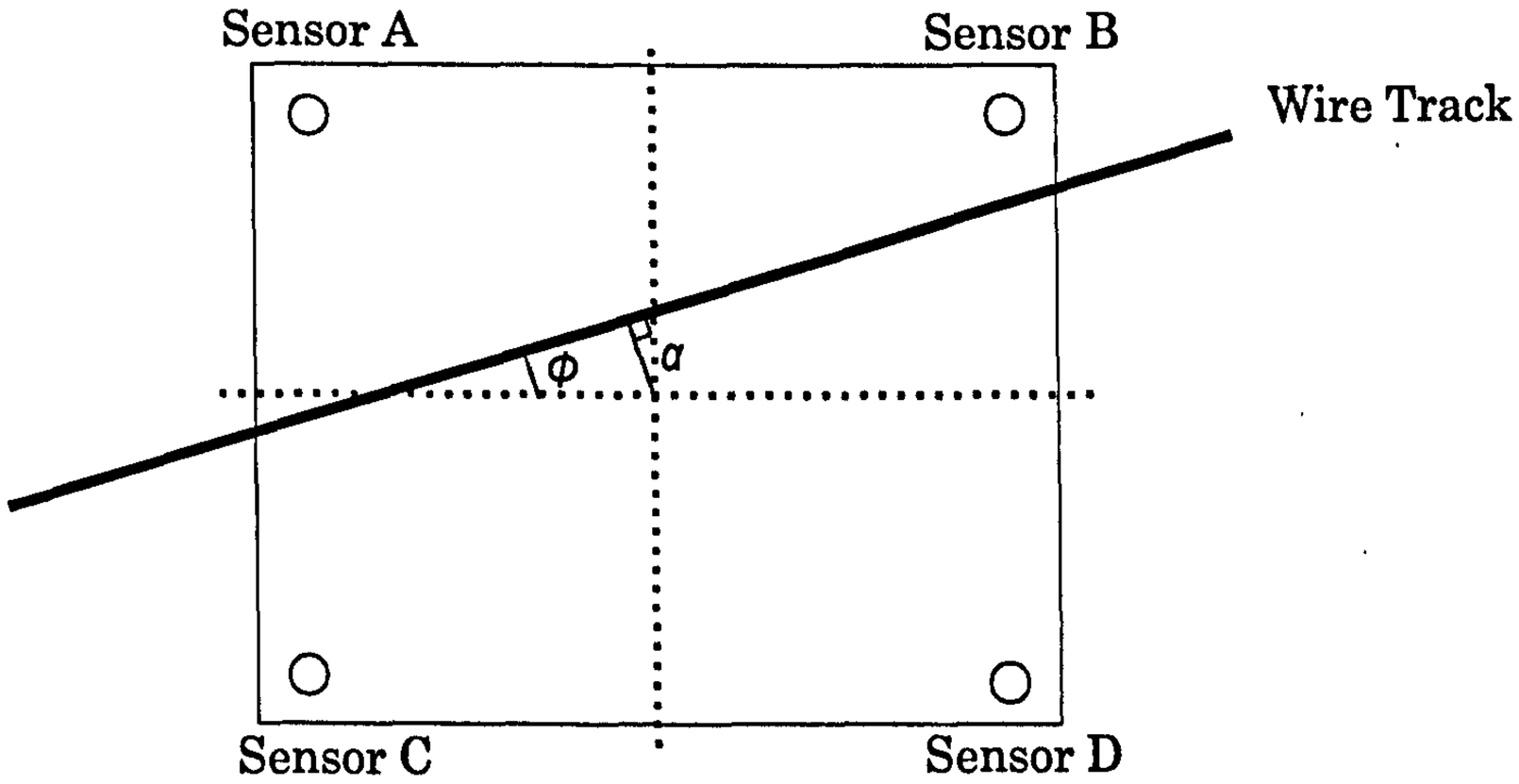


Figure 5.14: Definition of the sensor pairs

Sixty runs were carried out for each V_α and V_ϕ case and all the solutions were verified using a process which is described in Section 5.5.1.

Case 1 was investigated to prove that CGP achieves a correct normal solution. Out of 60 runs, 60 successful solutions were obtained. A successful run is such that the output from the evolved solution is equal to the output from the original algorithm, thus giving a cost of zero (maximum fitness). All the obtained solutions were the same as the original algorithms (Equations 5.1 and 5.2).

5.5.1 Verification Method

The stochastic nature of CGP meant that the obtained solutions for the failure cases could be different from each other. The requirement was for generic solutions, so a genericity test was devised as follows.

Test Set

The test input sets were characterised from y_1 and y_2 data in the data bank described in Section 5.4.1. Each bank consisted of 5000 data points and, in this case, the order of y_1 data was reversed. All of the reversed y_1 and non-reversed y_2 data were used as the test sets. Because of the reversing, the test sets would look different from the training sets. The 5000 test sets were applied to each solution and the mean costs were analysed and compared with each other. Figure 5.15 shows the actual value of α in mm and ϕ in degrees corresponding to the test set. Figure 5.16 provides the corresponding offset error voltages, V_α and V_ϕ. The graphs display a wide range of V_α and V_ϕ which are smooth and continuous, thus giving a good test set range. A solution with the best mean cost out of 60 obtained solutions was selected as the best solution. The best solutions for each failure cases are described next.

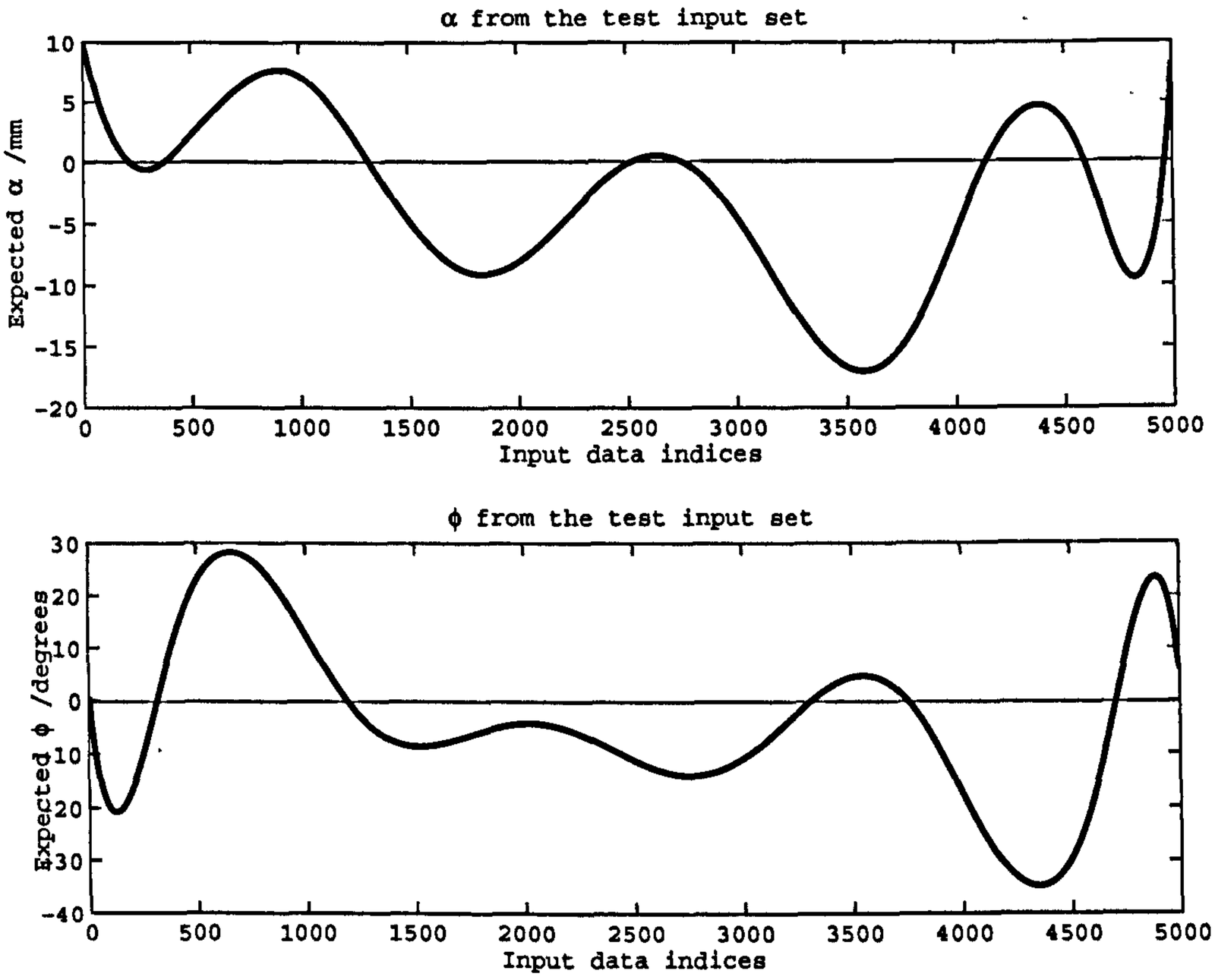


Figure 5.15: Expected α and ϕ obtained from the validation test set

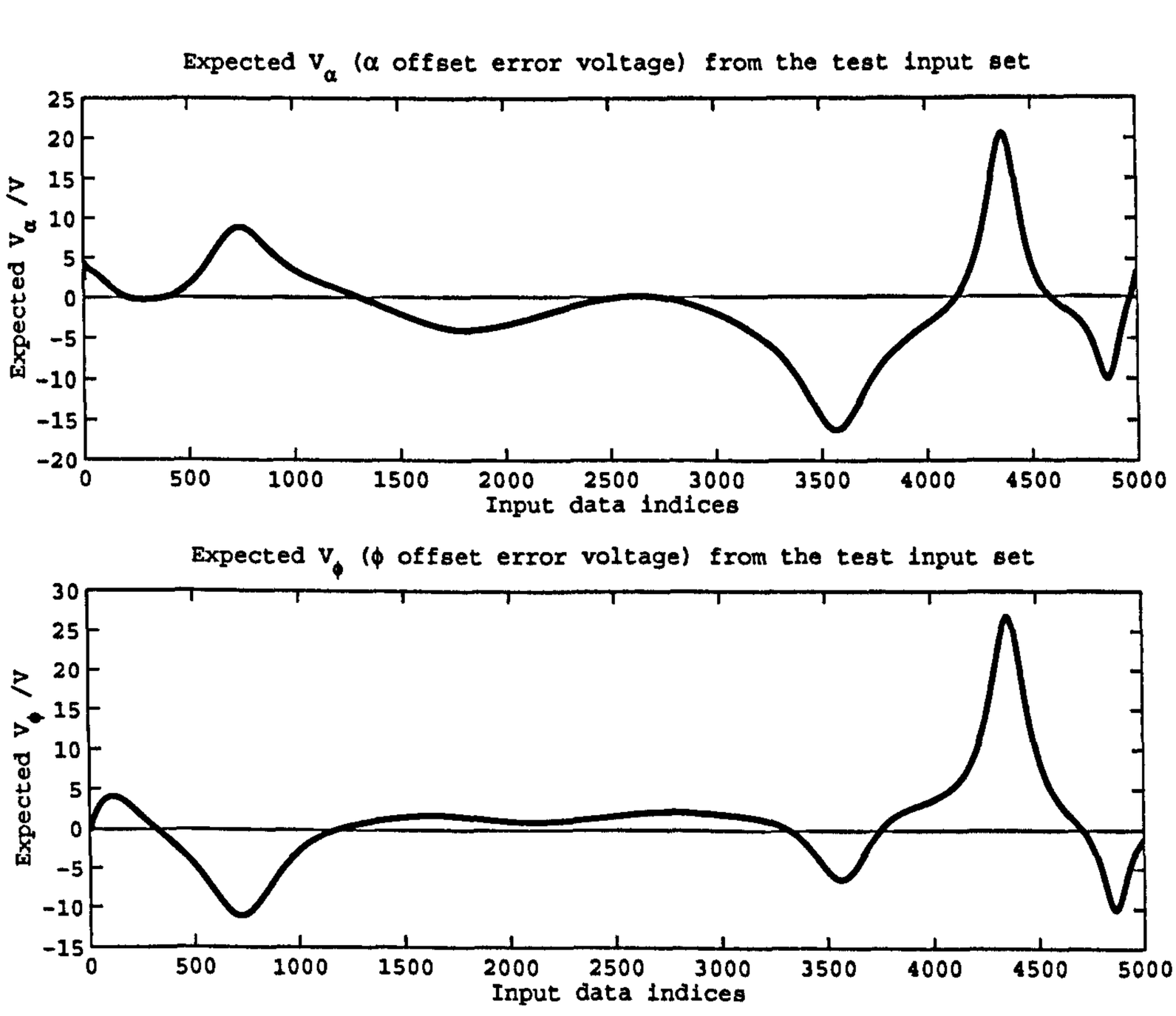


Figure 5.16: Expected V_α and V_ϕ obtained from the validation test set

5.5.2 Evolved Solutions

The symmetry of the plate means that if one good solution is obtained then that solution can be modified to fit other equivalent, complementary sensor failure cases. For example, assuming that a good solution is obtained for the case of a diagonal sensor pair, sensors A and D failure, then by swapping sensors A and D in the solutions with sensors B and C, and making the appropriate sign changes, a good solution for the other diagonal pair failure case follows. This technique has been used in realising the evolved solutions.

Recalling Equations 5.1 and 5.2, the solutions for detecting the offset errors when all the sensors are functioning correctly are:

$$V_\alpha = (V_A + V_B) - (V_C + V_D) \tag{5.27}$$

$$V_\phi = (V_A + V_D) - (V_B + V_C) \tag{5.28}$$

The Shaky Hand CGP has produced solutions for different failure cases as mentioned earlier. The results are summarised below.

Case 2: Solutions for One Sensor Failure Cases

1. Sensor A Failure

$$V_\alpha = \ln((\frac{V_B}{V_C})^2) \tag{5.29}$$

$$V_\phi = \frac{V_D}{V_C} - \frac{V_C}{V_D} \tag{5.30}$$

2. Sensor B Failure

$$V_\alpha = \ln((\frac{V_A}{V_D})^2) \tag{5.31}$$

$$V_\phi = \frac{V_D}{V_C} - \frac{V_C}{V_D} \tag{5.32}$$

3. Sensor C Failure

$$V_\alpha = \ln((\frac{V_A}{V_D})^2) \tag{5.33}$$

$$V_\phi = \frac{V_A}{V_B} - \frac{V_B}{V_A} \tag{5.34}$$

4. Sensor D Failure

$$V_\alpha = \ln((\frac{V_B}{V_C})^2) \tag{5.35}$$

$$V_\phi = \frac{V_A}{V_B} - \frac{V_B}{V_A} \tag{5.36}$$

Each V_α and V_ϕ solutions utilise only two sensor signals rather than three. However, all the three sensor signals are required to obtain full information for the position of the center of the plate relative to the wire track.

The two sensor pairs used are diagonal pairs for V_α solutions and adjacent pairs for V_ϕ solutions. Development of the solutions for the two sensor pair failure cases was then investigated, but it was found that the solutions for one sensor failure cases shown above are also the best solutions for the relative two sensor pair failure cases.

Case 3: Solutions for Adjacent Sensor Pair Failure Cases

1. Adjacent Pair AB Failure,

$$V_\alpha = -(\log_{10}(\ln({V_C}^{V_D} V_D) + V_C) + min[\frac{\ln({V_C}^{V_D} V_D)}{min[(V_C V_D)^{2({V_C}^{V_D})}, V_C]}, V_C]) \tag{5.37}$$

$$V_\phi = \frac{V_D}{V_C} - \frac{V_C}{V_D} \tag{5.38}$$

2. Adjacent Pair CD Failure,

$$V_\alpha = \log_{10}(\ln({V_B}^{V_A} V_A) + V_B) + min[\frac{\ln({V_B}^{V_A} V_A)}{min[(V_A V_B)^{2({V_B}^{V_A})}, V_B]}, V_B] \tag{5.39}$$

$$V_\phi = \frac{V_A}{V_B} - \frac{V_B}{V_A} \tag{5.40}$$

Case 4: Solutions for Diagonal Sensor Pair Failure Cases

1. Diagonal Pair AD Failure,

$$V_\alpha = \ln((\frac{V_B}{V_C})^2) \tag{5.41}$$

$$\begin{aligned} V_\phi = &- \ln(V_B V_C) + |\sin(\ln(V_C))|^{|max[(V_B V_C)^2\,,\,\ln(V_C)]|} \\ &- \log_{10}(\,max[\log_{10}(\sin(V_B + V_C))\,,\,max[(V_B V_C)^2\,,\,\ln(V_C)]]\,) \end{aligned} \tag{5.42}$$

2. Diagonal Pair BC Failure,

$$V_\alpha = \ln((\frac{V_A}{V_D})^2) \tag{5.43}$$

$$\begin{aligned} V_\phi = &- (\,-\ln(V_A V_D) + |\sin(\ln(V_D))|^{|max[(V_A V_D)^2\,,\,\ln(V_D)]|} \\ &- \log_{10}(\,max[\log_{10}(\sin(V_A + V_D))\,,\,max[(V_A V_D)^2\,,\,\ln(V_D)]]\,)\,) \end{aligned} \tag{5.44}$$

Case 5: Solutions for Opposite Sensor Pair Failure Cases

1. Opposite Pair AC Failure,

$$\begin{aligned} V_\alpha = &\ln(V_D^{V_B})\{\,\sqrt{|\ln(V_D^{V_B}) - max[V_D^{V_B}\,,\,V_B]|} + \ln(V_D^{V_B}) - max[V_D^{V_B}\,,\,V_B] \\ &+ \log_{10}(max[\frac{1}{max[V_D\,,\,V_B] + V_D - max[V_D^{V_B}\,,\,\sin(V_B^2)]}\,,\,V_B])\,\} \end{aligned} \tag{5.45}$$

$$V_\phi = |min[\ln(\sqrt{V_B})\,,\,V_D]| \tag{5.46}$$

2. Opposite Pair BD Failure,

$$\begin{aligned} V_\alpha = &- \ln(V_A^{V_C})\{\sqrt{|\ln(V_A^{V_C}) - max[V_A^{V_C}\,,\,V_C]|} + \ln(V_A^{V_C}) - max[V_A^{V_C}\,,\,V_C] \\ &+ \log_{10}(max[\frac{1}{max[V_A\,,\,V_C] + V_A - max[V_A^{V_C}\,,\,\sin(V_C^2)]}\,,\,V_C])\} \end{aligned} \tag{5.47}$$

$$V_\phi = |min[\ln(\sqrt{V_A})\,,\,V_C]| \tag{5.48}$$

Case 6: Solutions for Three Sensor Failure Cases

1. BCD Sensor Failure,

$$V_\alpha = \log_{10}(\sqrt{|\sin(\sqrt{V_A})|}) \tag{5.49}$$

$$V_\phi = -\ln(min[\sin(\sqrt{V_A}), V_A]) \tag{5.50}$$

2. ACD Sensor Failure,

$$V_\alpha = \log_{10}(\sqrt{|\sin(\sqrt{V_B})|}) \tag{5.51}$$

$$V_\phi = -\ln(min[\sin(\sqrt{V_B}), V_B]) \tag{5.52}$$

3. ABD Sensor Failure,

$$V_\alpha = \log_{10}(\sqrt{|\sin(\sqrt{V_C})|}) \tag{5.53}$$

$$V_\phi = -\ln(min[\sin(\sqrt{V_C}), V_C]) \tag{5.54}$$

4. ABC Sensor Failure,

$$V_\alpha = \log_{10}(\sqrt{|\sin(\sqrt{V_D})|}) \tag{5.55}$$

$$V_\phi = -\ln(min[\sin(\sqrt{V_D}), V_D]) \tag{5.56}$$

The evolved V_α solutions made use of all operator types given except *cos* and *tan*. The V_ϕ evolved solutions made use of all the operator types given except *min*, *reciprocal*, *cos* and *tan*.

5.6 Evolved Solution Analysis

5.6.1 Cost Analysis

Using the validation data sets, the mean costs, J_{V_α} and J_{V_ϕ}, over the test input set (5000 input sets) for the best evolved V_α and V_ϕ solutions and their standard deviations are summarised in Table 5.6.

Number of sensor failures	Failure case	J_{V_α}(STD)	J_{V_ϕ}(STD)
1	Sensor A failure	0.312(0.222)	0.206(0.215)
	Sensor B failure	0.241(0.256)	0.206(0.215)
	Sensor C failure	0.241(0.256)	0.299(0.297)
	Sensor D failure	0.312(0.222)	0.299(0.297)
2	Adjacent pair AB failure	0.792(13.700)	0.206(0.215)
	Adjacent pair CD failure	0.655(13.707)	0.299(0.297)
	Diagonal pair AD failure	0.312(0.222)	0.473(2.669)
	Diagonal pair BC failure	0.241(0.256)	0.504(2.671)
	Opposite pair AC failure	1.039(3.471)	1.084(2.338)
	Opposite pair BD failure	1.023(3.442)	1.082(2.332)
3	Sensor BCD failure	1.161(6.552)	1.237(5.094)
	Sensor ACD failure	1.241(6.134)	1.134(5.103)
	Sensor ABD failure	1.259(6.133)	1.091(4.265)
	Sensor ABC failure	1.180(6.551)	1.178(4.261)

Table 5.6: Summary of the cost and the standard deviations of the evolved solutions

The one sensor failure cases have mean costs of less than 0.5. This indicates that for every input set used, an error of less than 50% is made on the estimation of the wire position on the plate relative to the sensor positions. The opposite sensor pair and the three sensor failure cases have mean costs above 1 and the operation is more likely to fail in these cases as the solution outputs become unreliable.

Looking into the physical size of the plate (45mm by 45mm), if the cost is relatively low, for example less than 0.5, the centre of the plate will be close to the wire track, e.g. if α is 5mm, then the error would be approximately 2.5mm. An error-driven control algorithm will drive the plate to reduce this error, provided that the sense of the error is in the correct direction. So, as long as the motors move correctly and α and ϕ are small, then the Shaky Hand should operate correctly but with degraded tracking performance. If the offsets are large and/or of the wrong sense, combined with high cost then there would be a serious problem.

From Table 5.6, the following observations can be made.

- Adjacent sensor pair failure V_ϕ solutions and diagonal sensor pair failure V_α solutions produce good fitness with small standard deviations. The combined solutions are used for the one sensor failure cases as mentioned in Section 5.5.2.

- Adjacent sensor pair failure V_α solutions do not have as good fitness as those for V_ϕ or diagonal sensor pair failure V_α solutions. The standard deviations are also high.

- Diagonal sensor pair failure V_ϕ solutions have relatively good fitness but have high standard deviations.

- Opposite sensor pair failure solutions have similar fitness to those given by the three sensor failure solutions.

- A huge difference can be observed in the fitness between 'One sensor failures, adjacent sensor pair failures, diagonal sensor pair failures' and 'opposite sensor pair failures, three sensor failures'.

Figure 5.17 displays how the Shaky Hand would look at the plate upon failures.

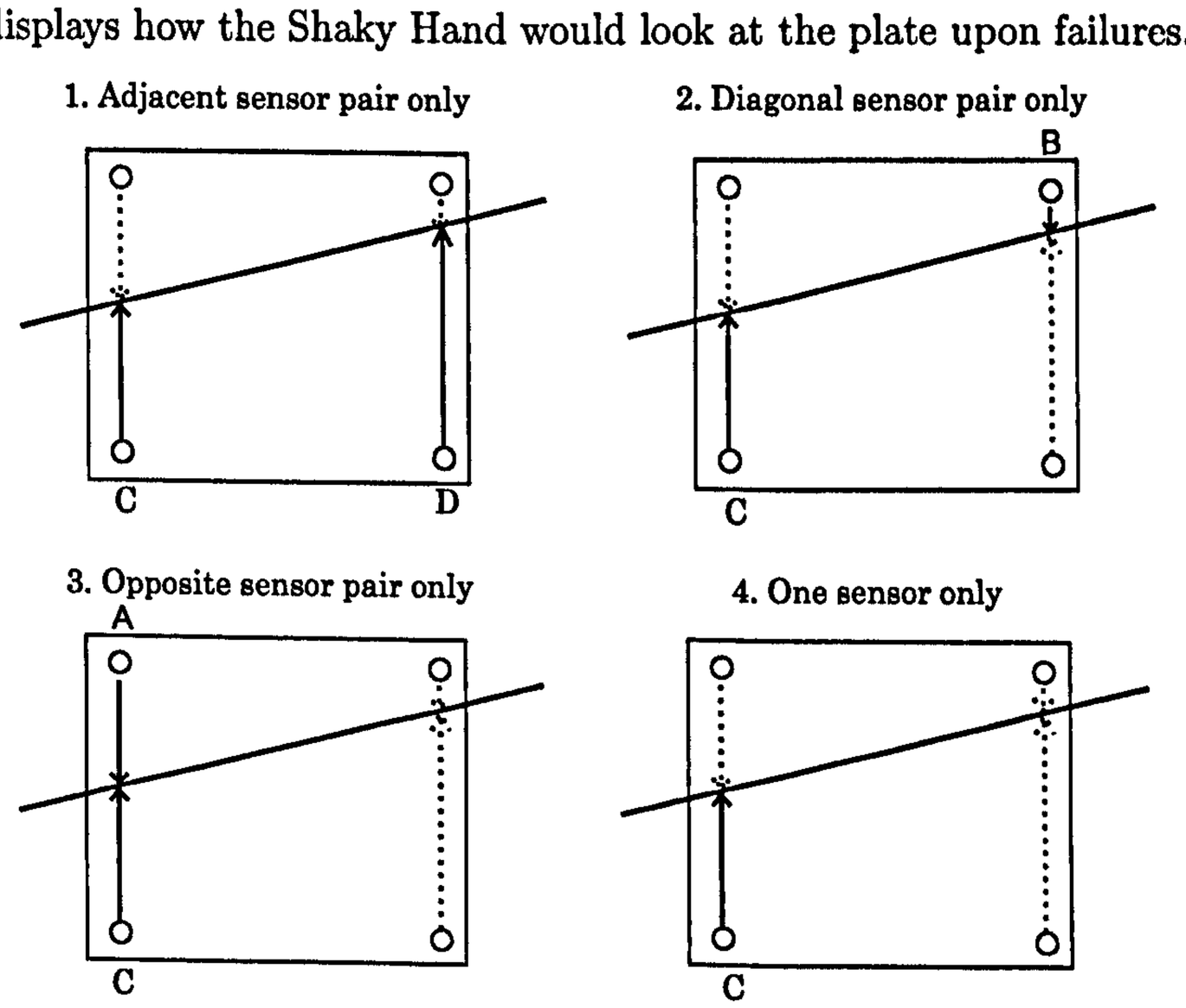


Figure 5.17: Sensor failure patterns

If there is only one adjacent pair sensor (Figure 5.17, No.1), it is still possible to obtain the approximate angle information using the relative sensor voltages. However, the sensors on

the other side of the plate do not provide any information regarding the distance to the other side of the plate, so the exact lateral offset would be more difficult to obtain than the angle. This explains the very good V_ϕ fitness and poor V_α fitness.

If only the diagonal sensor pairs are active (Figure 5.17, No.2), then a reference on each side (top and bottom) of the plate is provided, giving more accurate estimation for the lateral offset. Thus V_α fitness is good. The difference in the diagonal sensor pair output values can provide correct polarity of the angle, and the proportion of the difference can be used to estimate the rough value of V_ϕ.

The one sensor failure cases use the same V_ϕ solutions for the adjacent sensor pair failure cases and the V_α solutions for the diagonal sensor pair failure cases. Both fitness are very good. These cases use only two sensor outputs to estimate each V_α and V_ϕ but require the use of all three remaining sensors.

For the opposite sensor pair failure case (Figure 5.17, No.3), the sensors would not be able to provide any information about the angle. Figure 5.18 explains the situation graphically.

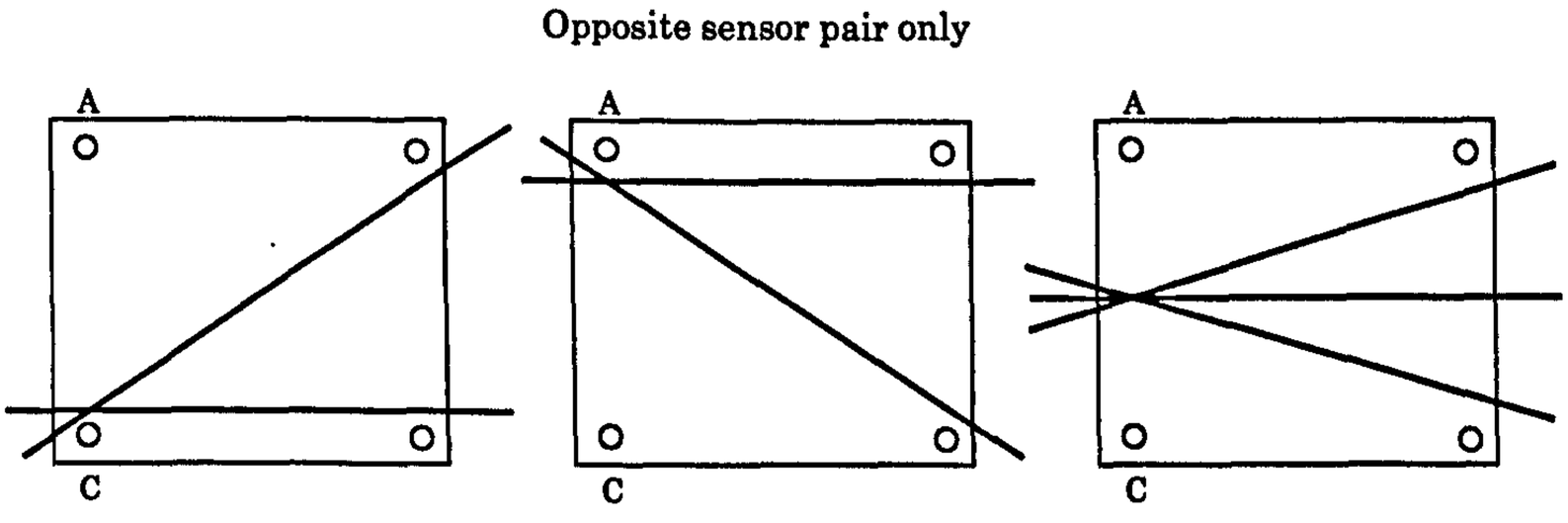


Figure 5.18: Impossible estimation of the offset errors

The three plate diagrams on this figure, each contain two or three different wire positions whose α and ϕ are different. However, the working opposite pair sensors (in this case, sensors A and C) would produce the same signals for these different wire positions in each case. The Shaky Hand can obtain information about the position of the wire at the edge of the plate, but not a position relative to the centre of the plate. Therefore the fitness for both V_α and V_ϕ are bad. For this reason, if there are only opposite sensor pairs on the plate, the Shaky Hand operation fails.

For the three sensor failure cases (Figure 5.17, No.4), the remaining one sensor does not provide much information about the wire positions to the Shaky Hand as well. There is no reference and the Shaky Hand cannot obtain any information about the angle or lateral offset. It would be wise to abandon the operation.

There is a reason for the high standard deviation for V_α solution obtained for adjacent sensor pair failure case. Costs for the test sets were obtained from the evolved solutions using Equations 5.21 and 5.22. If the denominator of the equations is small, i.e. if the ideal values of V_α or V_ϕ, are small, the outcome of the equations tends to be large. The difference in the ideal non-failure case values and the obtained values from the evolved solutions therefore becomes more critical for the small ideal values. The high standard deviations for the diagonal sensor pair failure cases are due to the discontinuity seen when evaluating the costs. This can be observed from Figures 5.16 and 5.19. Figure 5.19 compares the cost of one of the V_α diagonal sensor pair failure solutions (J_{V_α}=0.241, STD=0.256) and the adjacent pair failure solution (J_{V_α}=0.655, STD=13.707) for all the test input sets.

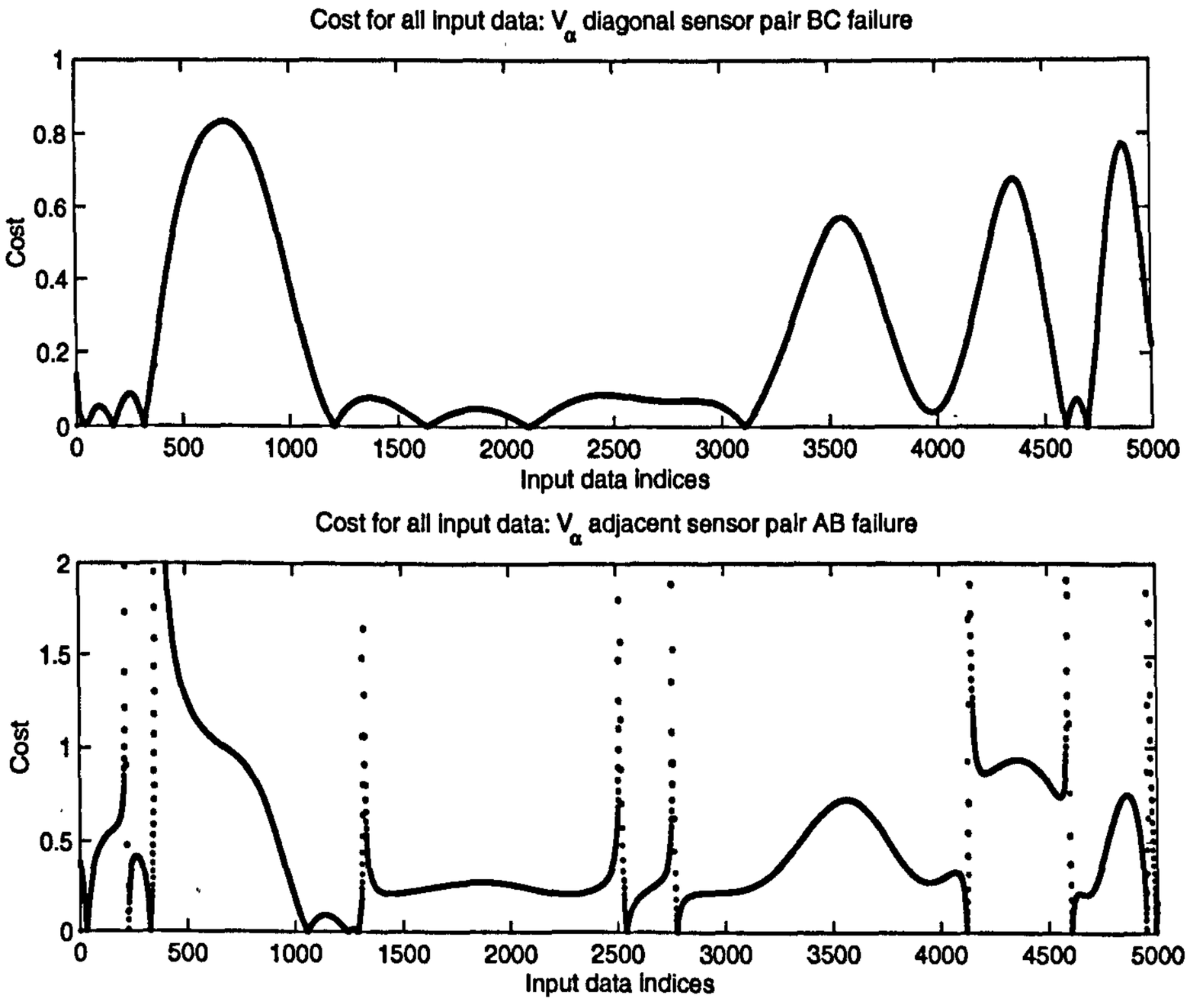


Figure 5.19: Comparison of diagonal and adjacent sensor pair failure solutions for V_α using test sets

Cost variation for the V_α diagonal sensor pair failure solution shows that the cost is kept below 1 (good) for all the input sets. For the adjacent sensor pair failure solution, the cost is kept below 1 for most of the input sets applied but exceeds 1 for some occasions. This is exactly the number of times that V_α data in Figure 5.16 crosses the *zero* line (transition from positive to negative, negative to positive). Unless a very good solution whose outcome is very similar to the original algorithm outcome is evolved, the standard deviation will be

affected by this evaluation method. It is important to note that the cost from the adjacent sensor pair failure solution is kept low for the majority of the input sets which suggests the high quality of the obtained solution.

The same analysis was carried out for other solutions. The adjacent sensor pair failure V_ϕ solution produced similar graph (Figure 5.20, bottom) to the diagonal sensor pair failure V_α solution. The diagonal sensor pair V_ϕ solution produced similar graph (Figure 5.20, top) to the adjacent sensor pair failure V_α solution.

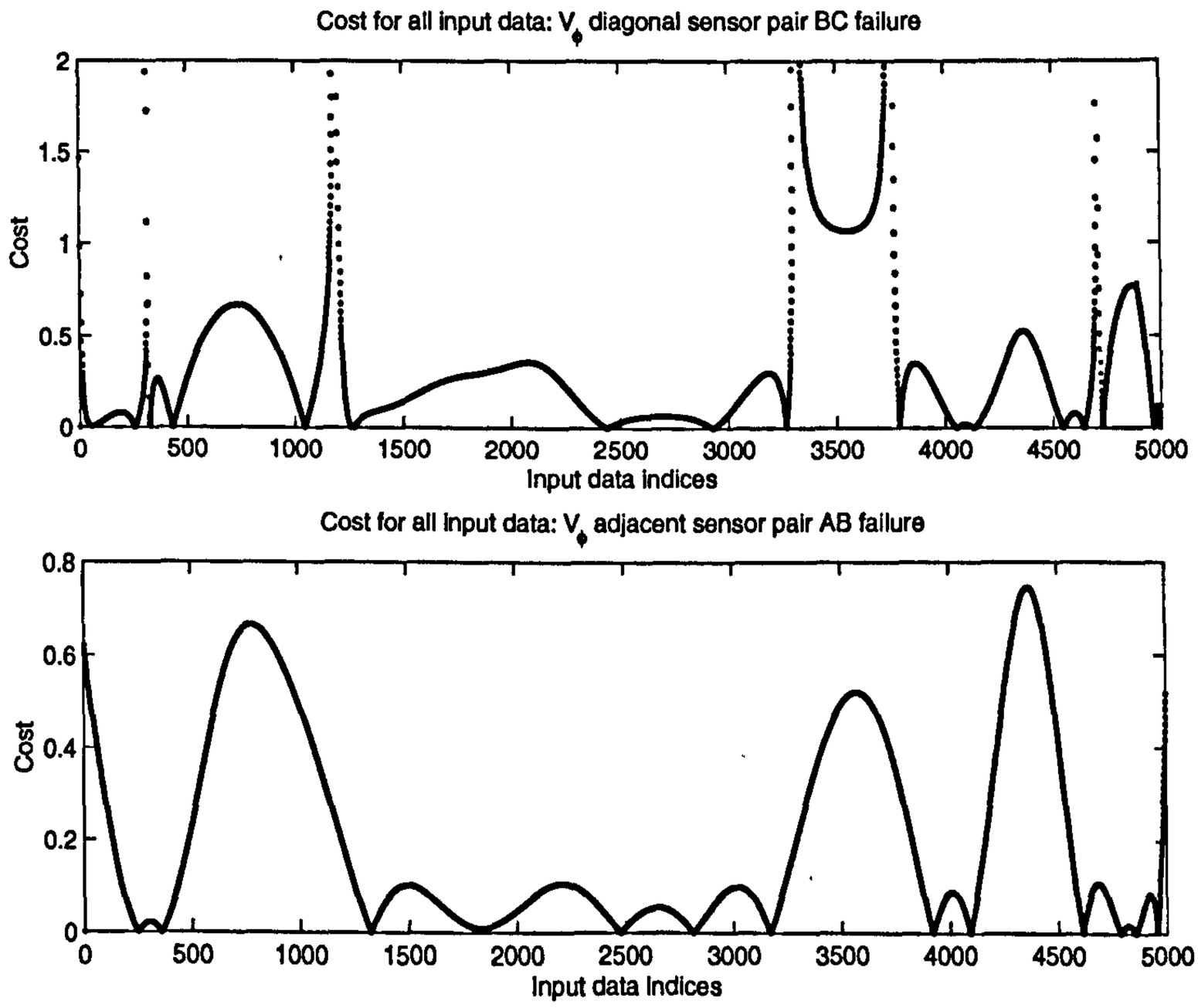


Figure 5.20: Comparison of diagonal and adjacent sensor pair failure solutions for V_ϕ using test sets

For both the V_α and V_ϕ opposite sensor pair failure cases and three sensor failure cases as shown in Figures 5.21 and 5.22, their fitness is kept to near unity for the majority of input sets. Their cost values have extreme values when the expected values are zero.

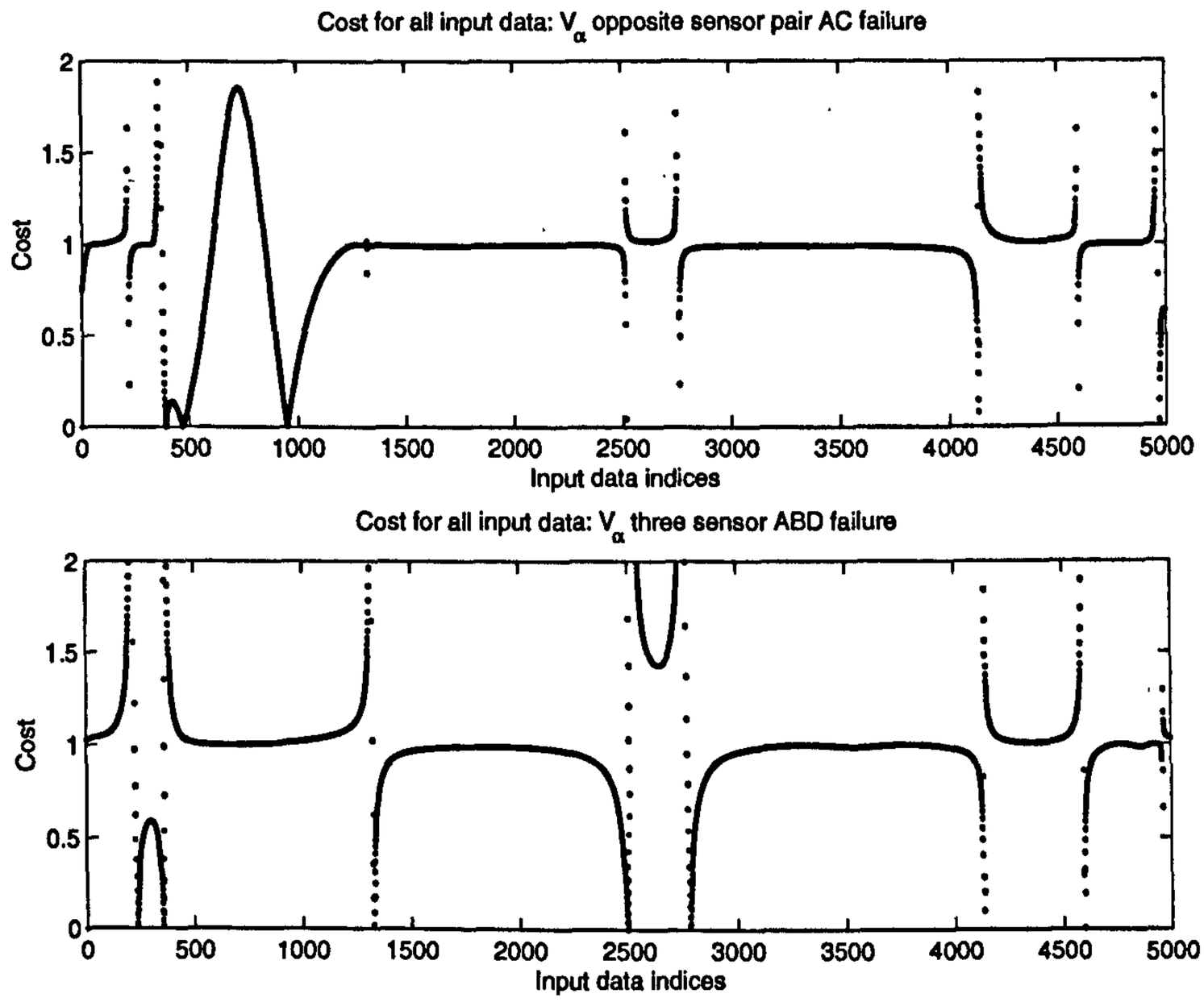


Figure 5.21: Comparison of opposite sensor pair failure solutions and three sensor failure solutions for V_α using test sets

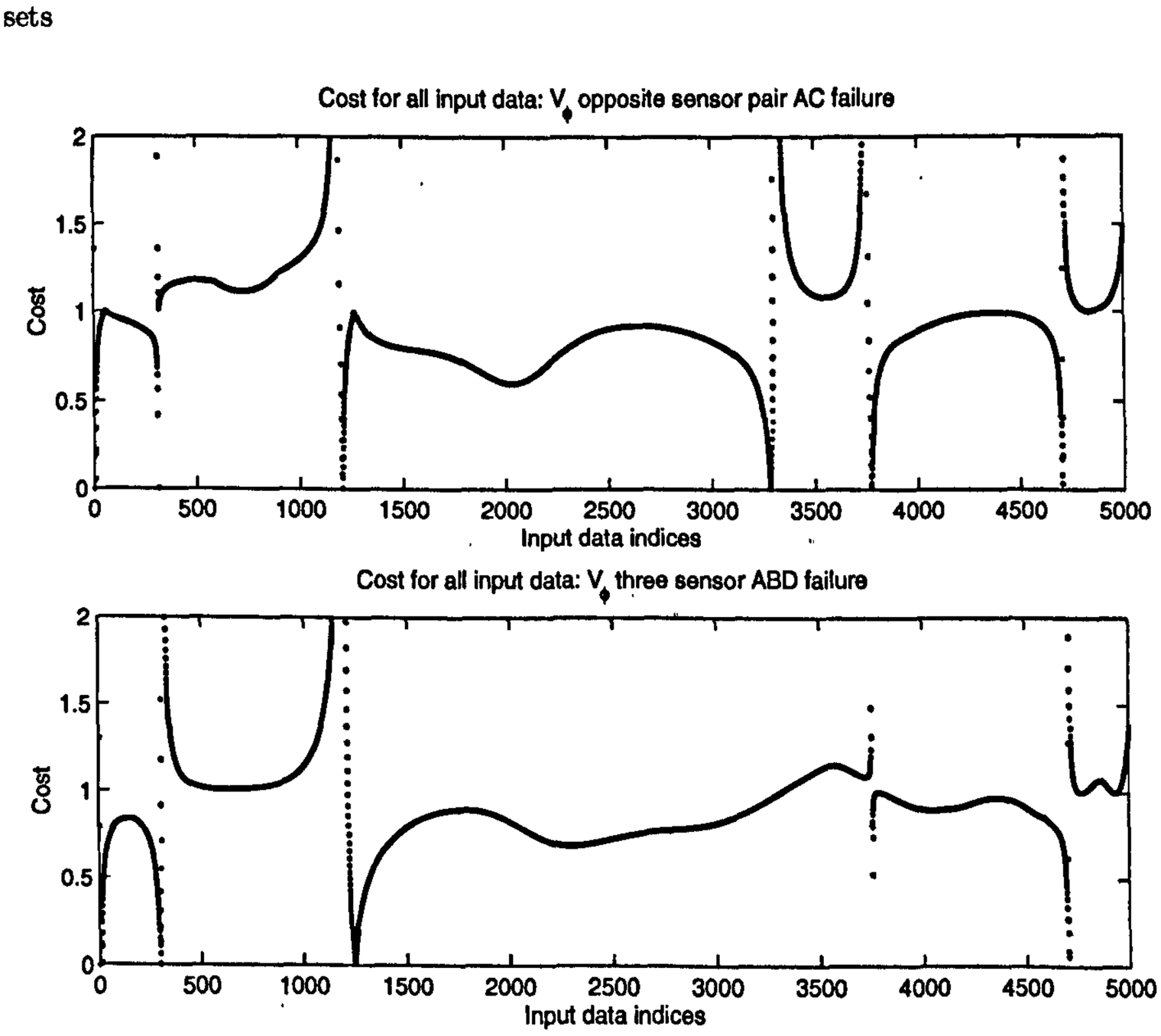


Figure 5.22: Comparison of opposite sensor pair failure solutions and three sensor failure solutions for V_ϕ using test sets

5.6.2 Evolutionary Convergence Analysis

Costs from the best obtained evolved solutions were recorded every 200 generations during the evolution process. This allows us to observe how the evolutions produce generic solutions. There may be similar patterns to all successfully obtained generic solutions. The results are shown in Figures 5.23 to 5.28. The first 5000 generations of the evolutions are also featured in Figures 5.29 and 5.30. The input training set remains constant for 1000 generations. As the CGP evolution progresses over these 1000 generations, reduction in cost (increased fitness) is to be expected. After each 1000 generations, the training set is changed, so a sharp increase in cost (reduced fitness) is to be expected. However, the convergence process should recommence. This general trend can be observed in all the figures (Figures 5.23 to 5.30). It is not always the case that the fitness gets worse but it is clear that the fitness is affected by this change. These figures show occasional sudden increase in costs and this coincides with a change in the input set, but generally drops off rapidly as the solution evolves further.

The V_α solution for diagonal sensor pair failure case and the V_ϕ solution for adjacent pair failure case produce good fitness, as shown in Table 5.6. Their cost evolution graphs (Figures 5.24 and 5.26 respectively) show gradual convergence towards zero at later generations. The V_α solution for adjacent sensor pair failure case and the V_ϕ solution for diagonal pair failure case produce reasonably good fitness. Their cost evolution graphs (Figures 5.23 and 5.27 respectively) are quite similar to Figures 5.24 and 5.26 that their solutions converge towards zero at later generations. For the opposite sensor pair failure solutions (Figures 5.25 and 5.28), which produce mean cost over 1, the graphs also show the tendency to converge, however, the costs get poorer at the later generations.

As mentioned before, the initial maximum generation number was limited to 100,000. All the solutions were evolved with this maximum generation number. However, we found that the graphs of the V_α solutions for adjacent pair and the opposite pair failures seem to contain two large *waves*. Their cost suddenly gets higher than the initial cost value after 50,000 generations and starts to settle down again gradually. For this reason, the V_α diagonal sensor pair failure solution was re-evolved using 50,000 generations to observe the outcome. Exactly the same solution was obtained despite CGP being a random process. So, all the solutions were re-evolved with 50,000 generations and proved to be successful except for the opposite sensor pair failure solutions. Evolutions using fewer generations (10,000 and 30,000 generations) to obtain the solutions were then attempted but a generic solution could not be obtained for any of the failure conditions. This is due to the under-training of the solution

that it did not exploit enough generations to converge towards a generic solution. Evolutions using more generations (300,000, 500,000 and 1,000,000) were attempted for the opposite sensor pair failure solutions, but generic solutions still could not be obtained. This is because of the difficulty in estimating the α and ϕ offsets as explained in Section 5.6.1.

In some parts of graphs, for example, the costs of the solution being evolved at generation number of between 15,000 and 18,000 in Figure 5.23 show that the solution did not converge after 1000 generations of fixed input set, so, the cost was constantly increased when the input set was modified. Increasing the sliding window size might help this part of the evolution to converge, but on the whole, these evolutions converged well and provided the good solutions. Therefore no further change was effected.

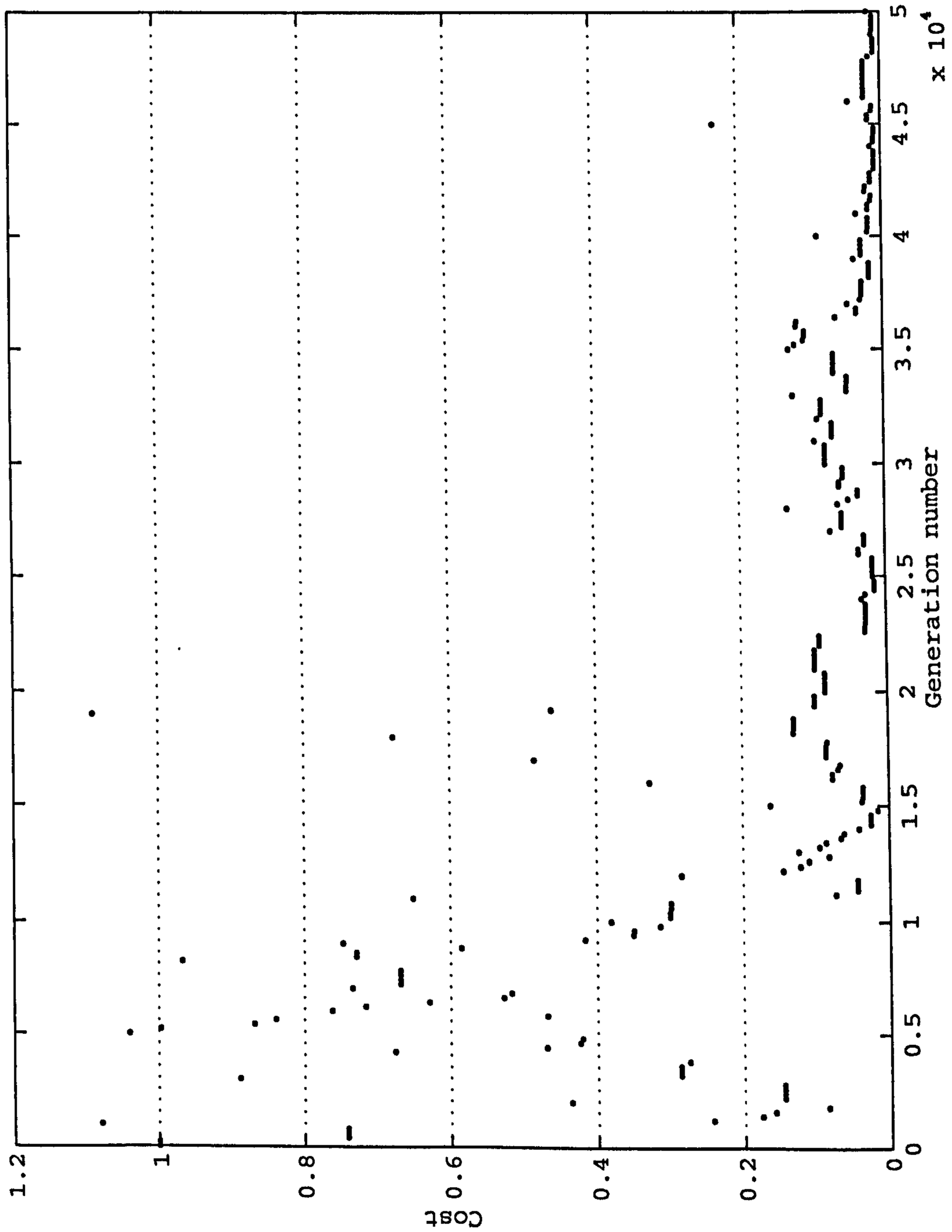


Figure 5.23: V_α cost variation for the adjacent pair failure solution

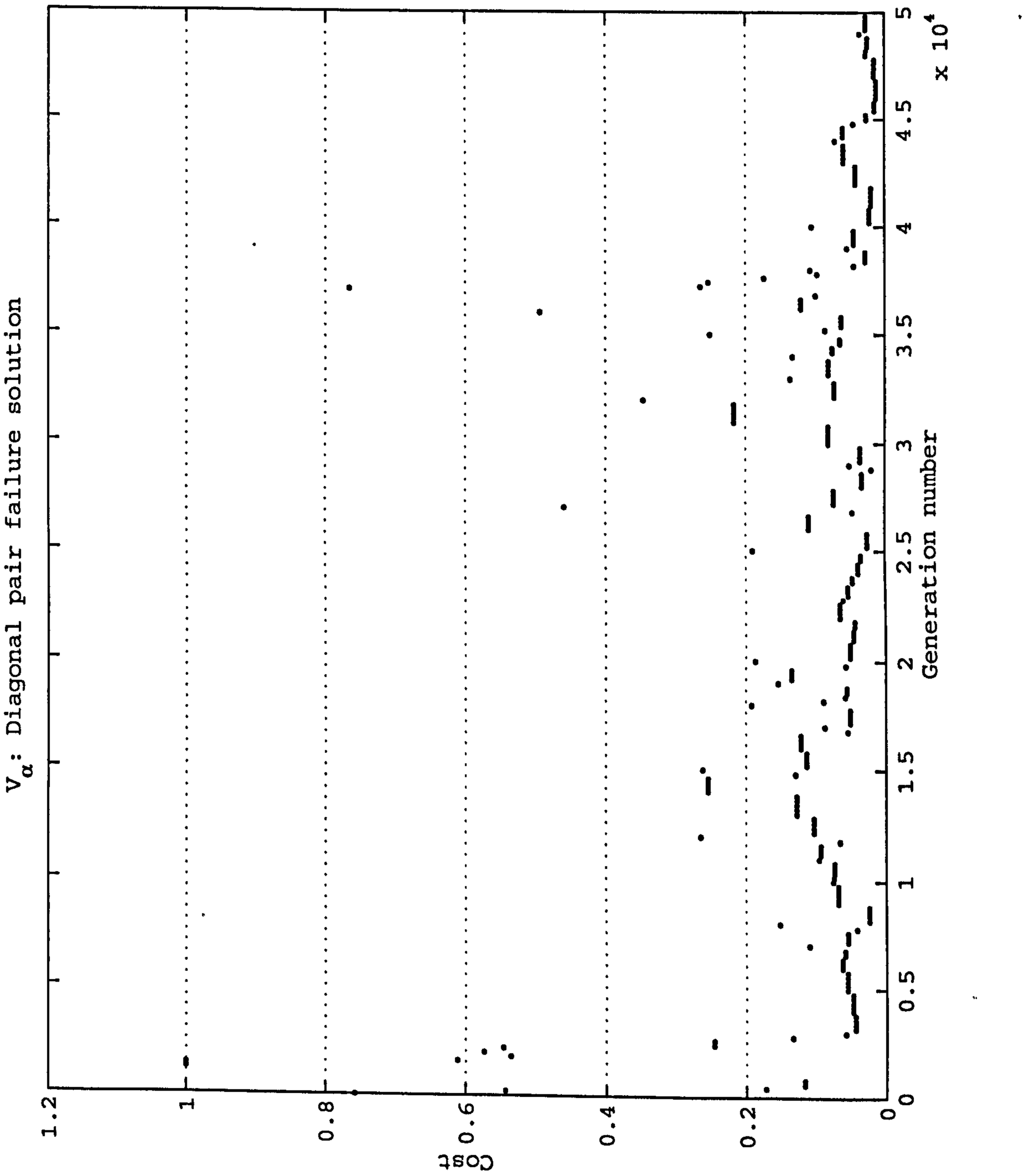


Figure 5.24: V_α cost variation for the diagonal pair failure solution

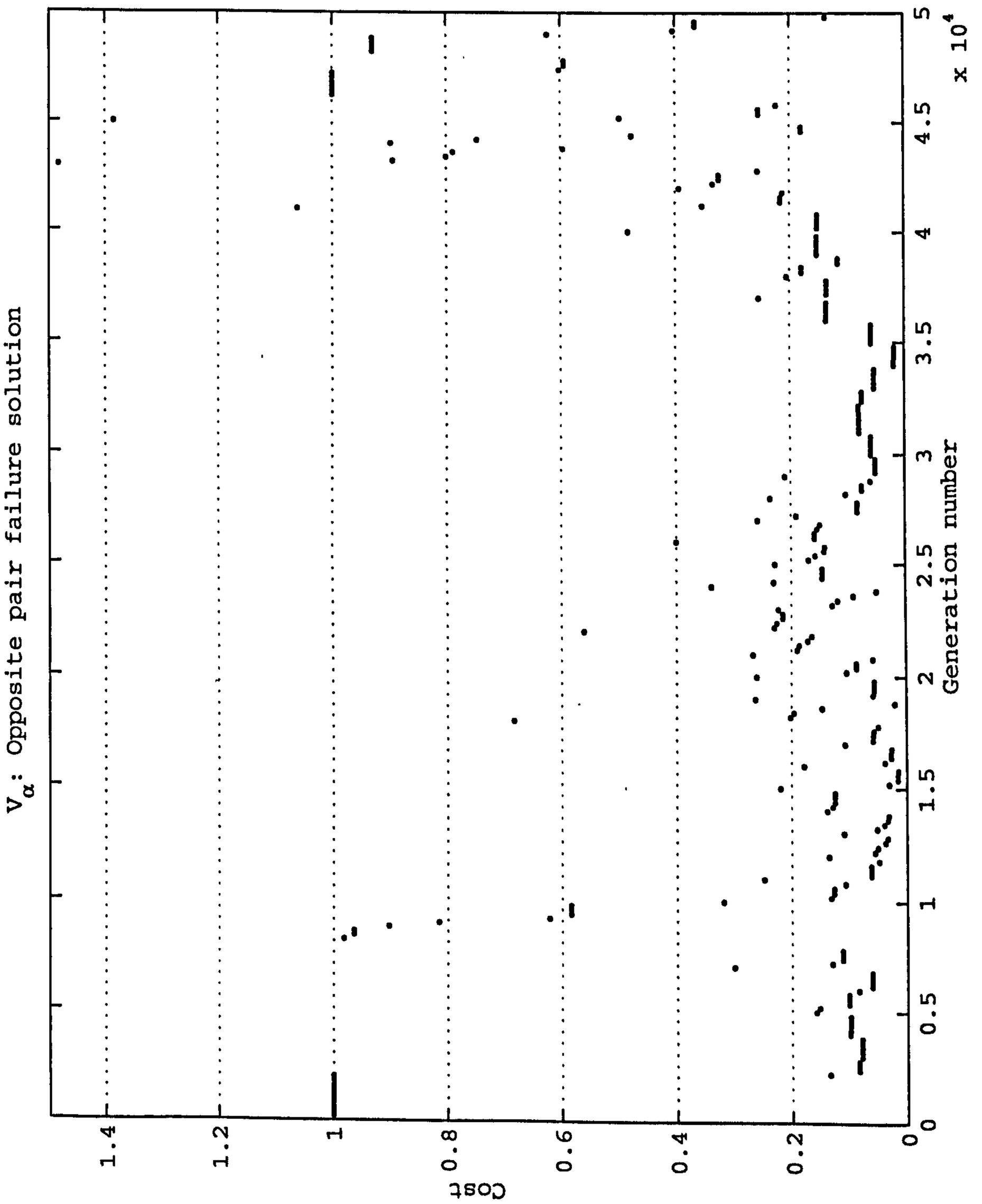


Figure 5.25: V_α cost variation for the opposite pair failure solution

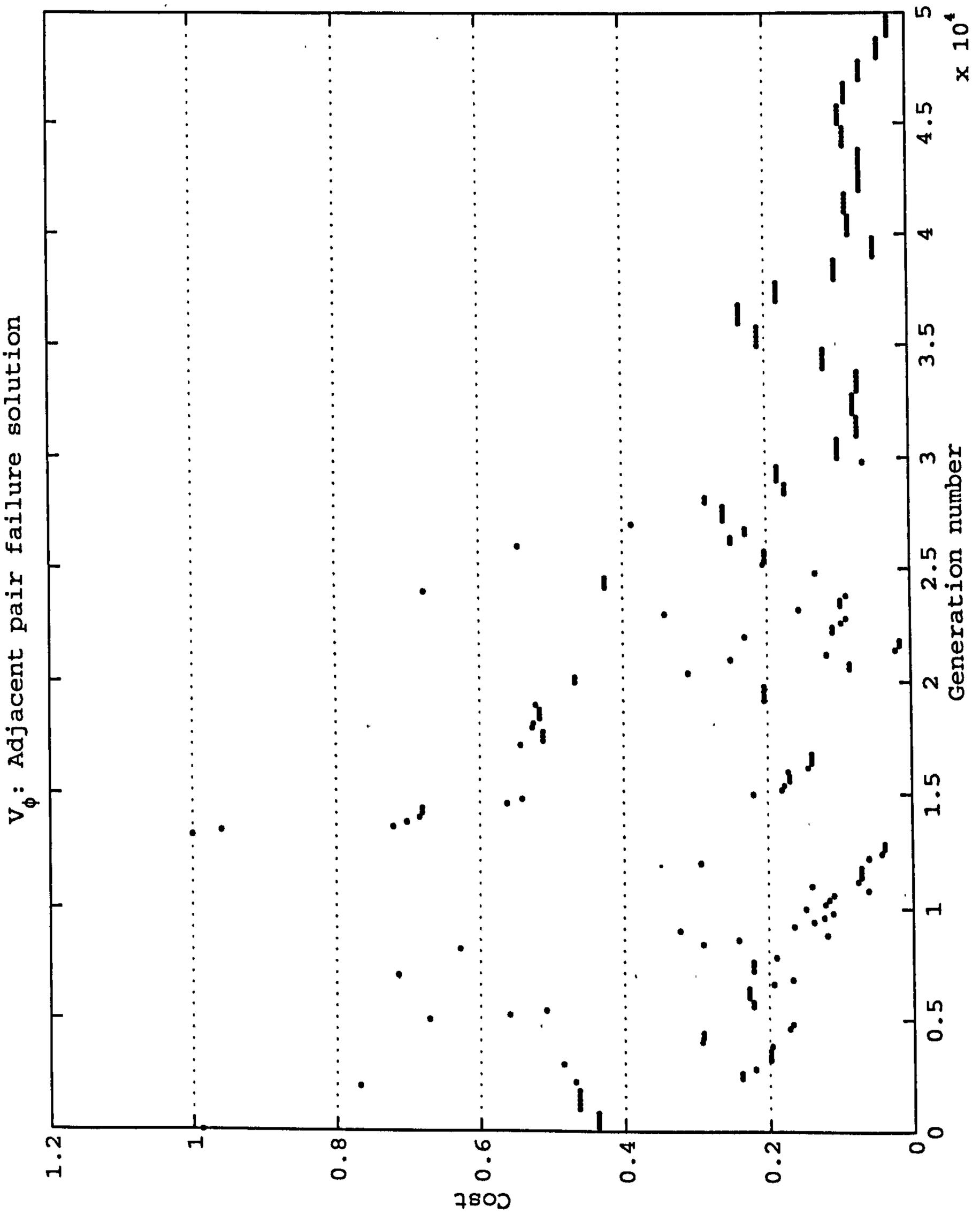


Figure 5.26: V_ϕ cost variation for the adjacent pair failure solution

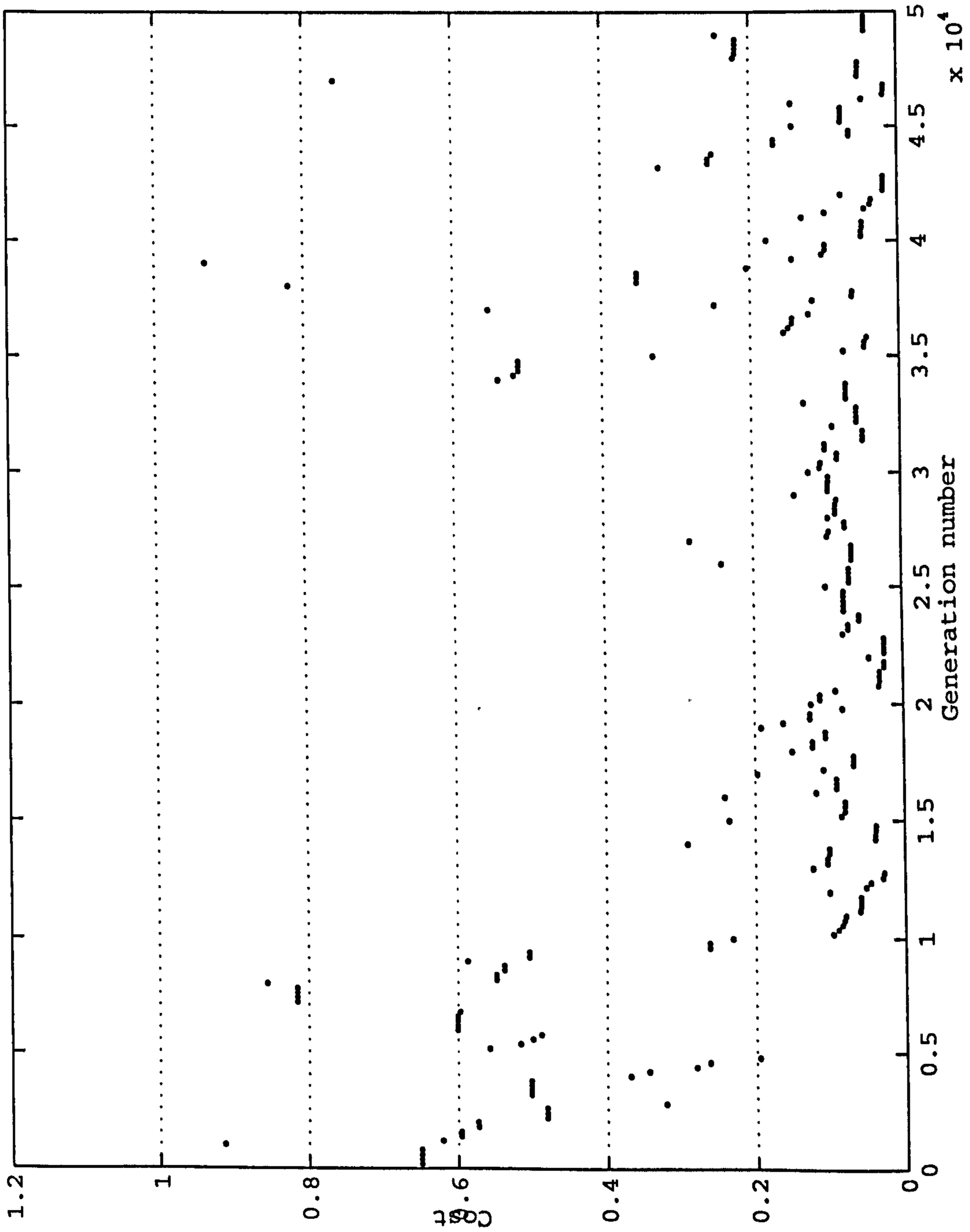


Figure 5.27: V_ϕ cost variation for the diagonal pair failure solution

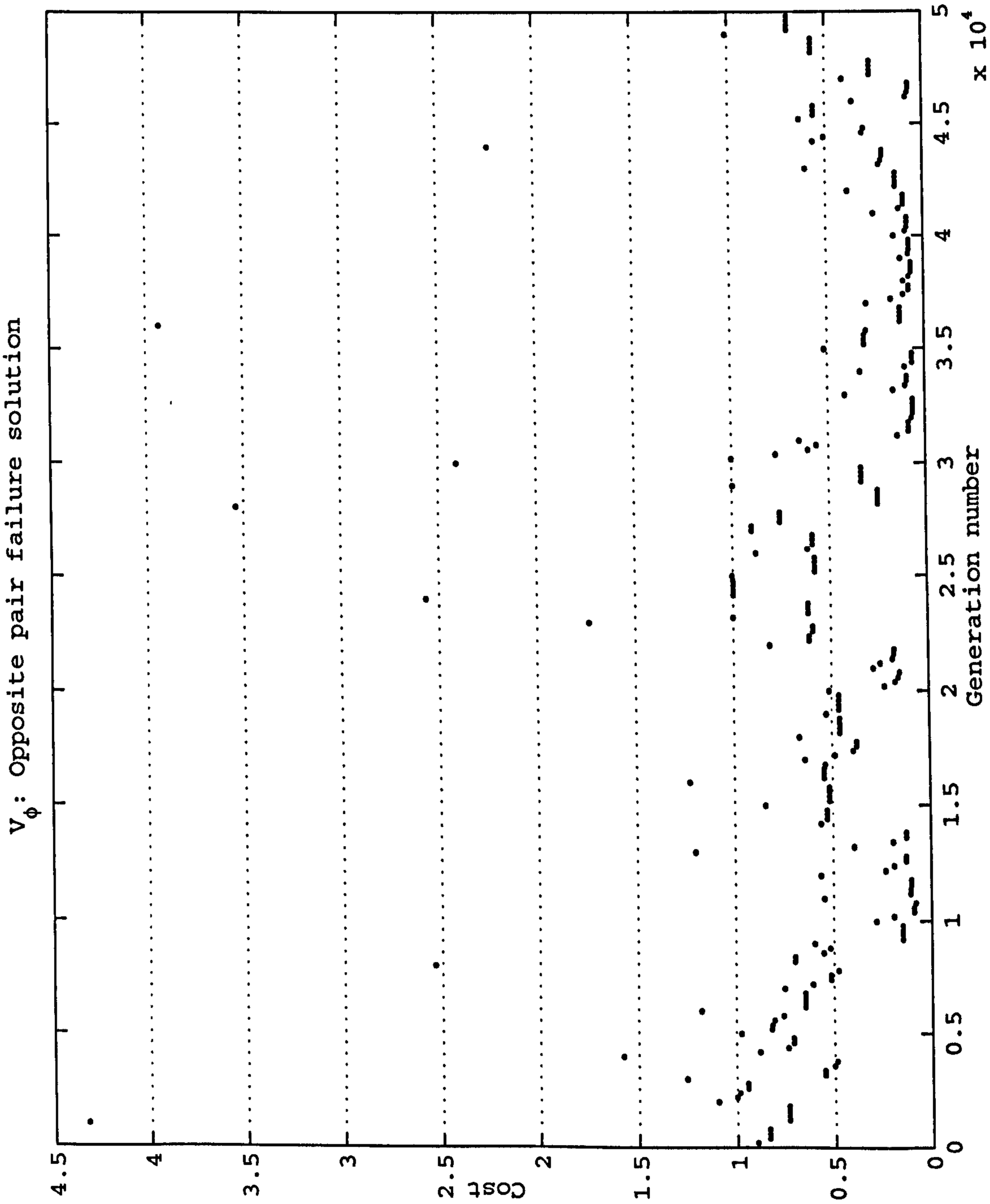


Figure 5.28: V_ϕ cost variation for the opposite pair failure solution

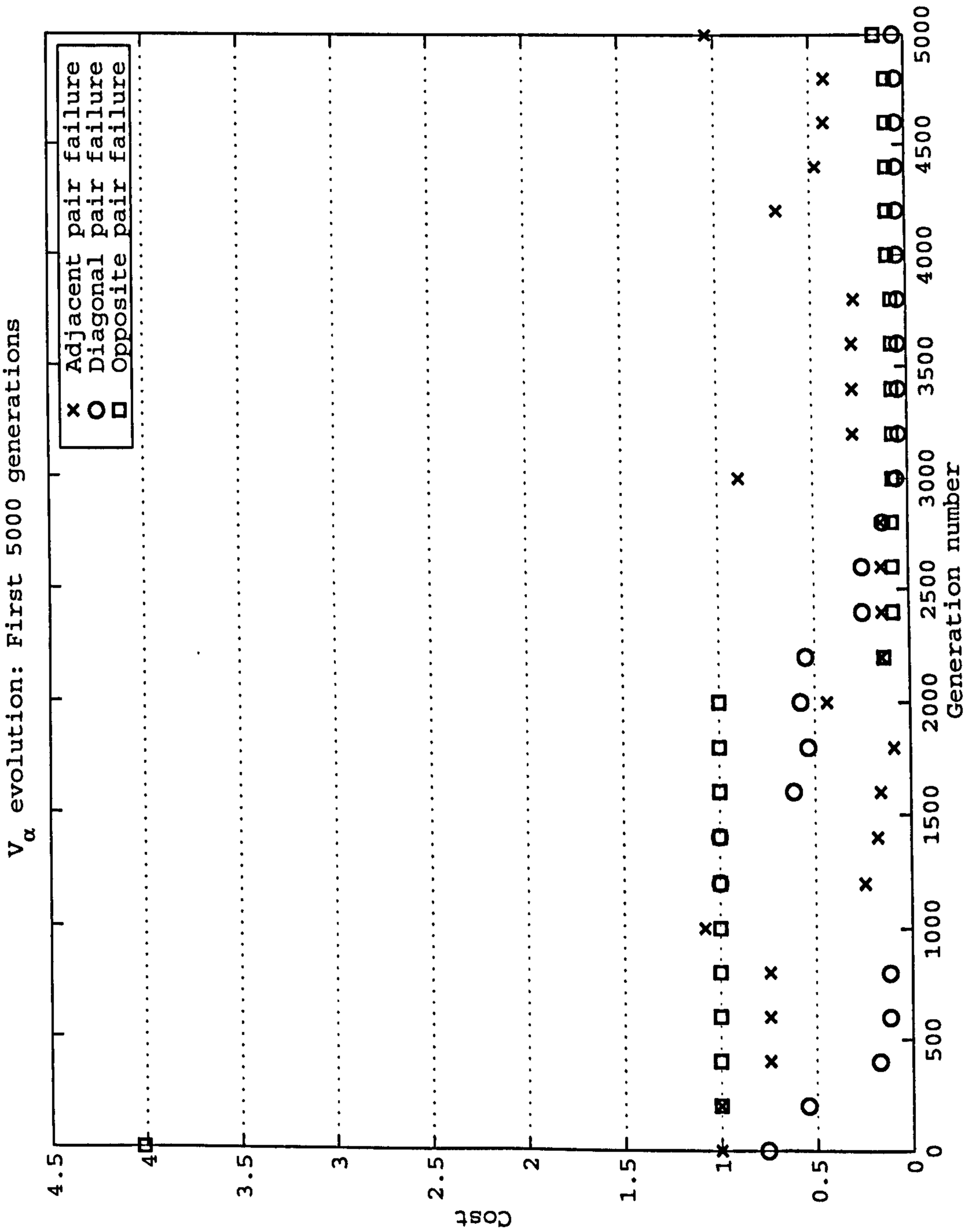


Figure 5.29: V_α cost variation for the first 5000 generations

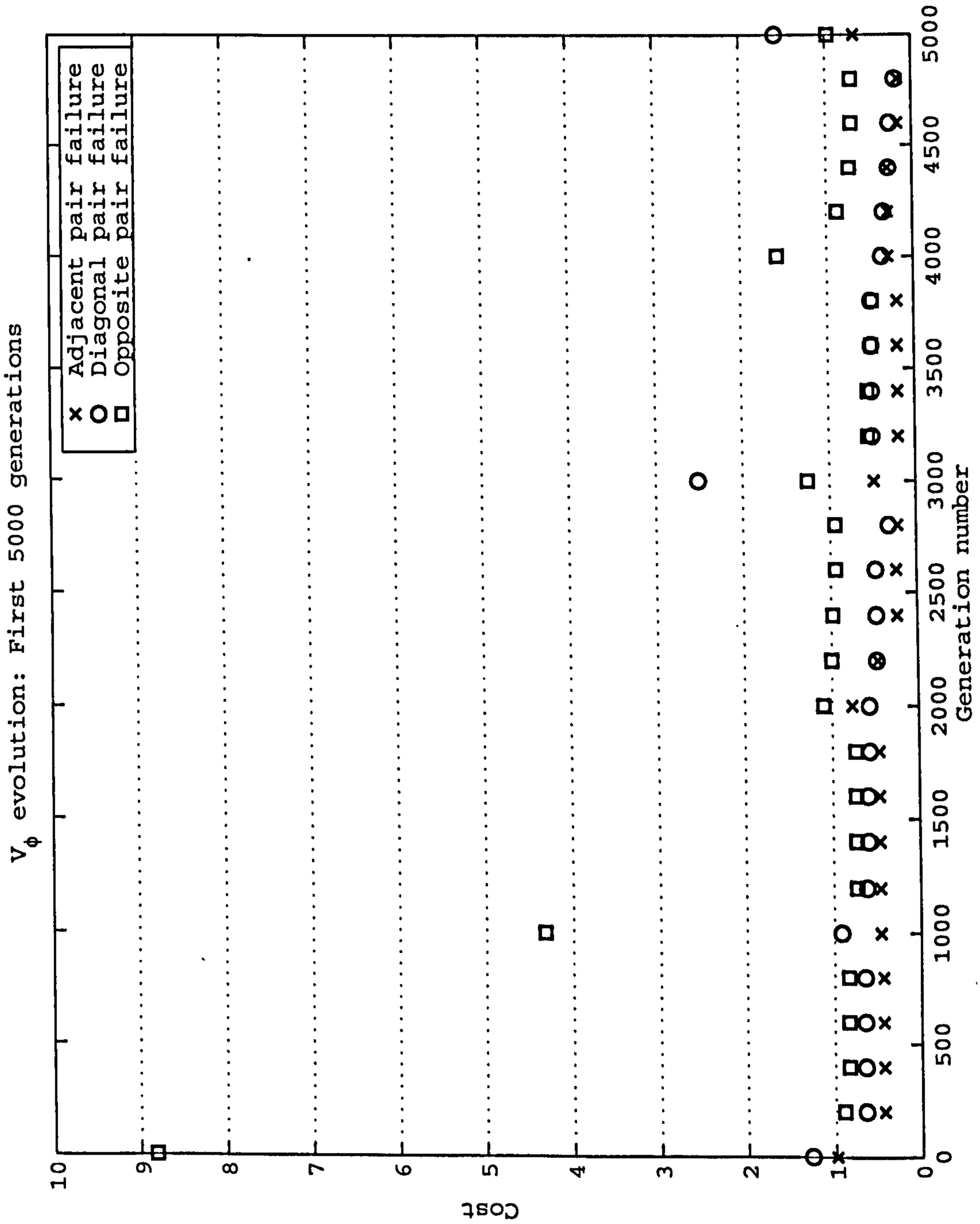


Figure 5.30: V_ϕ cost variation for the first 5000 generations

5.6.3 Improvement Over the Original Solutions

For completeness, the performance of the original 4-sensor algorithms (Equations 5.27 and 5.28) were assessed when they were used under failure conditions, with zeroed faulty coil voltages applied as appropriate. The standard test sets were used as in previous experiments to validate evolved algorithms. The following data was collected.

- Mean costs and the standard deviations corresponding to the test set (5000 data points).
- V_α and V_ϕ values for all the failure cases.

Cost and the standard deviation obtained from the original algorithms were compared with those obtained from the evolved solution. The results are tabulated in Table 5.7 for V_α solutions and in Table 5.8 for V_ϕ solutions.

Failure Cases	Cost/STD From Original algorithms	Cost/STD From Evolved Algorithm
One Sensor (B Failure)	8.798/179.945	0.241/0.256
Adjacent Pair (AB Failure)	19.870/353.160	0.792/13.700
Diagonal Pair (BC Failure)	0.441/0.245	0.241/0.256
Opposite Pair (AC Failure)	4.312/66.834	1.039/3.471
Three Sensors (ABC Failure)	11.150/179.504	1.180/6.551

Table 5.7: Comparison of cost/STD between the original V_α algorithm and the evolved V_α algorithms

Failure Cases	Cost/Std From Original Algorithm	Cost/Std From Evolved Algorithm
One Sensor (A Failure)	2.593/24.492	0.206/0.215
Adjacent Pair (AB Failure)	0.411/0.262	0.206/0.215
Diagonal Pair (BC Failure)	6.133/51.626	0.504/2.671
Opposite Pair (AC Failure)	1.982/16.461	1.084/2.338
Three Sensors (ABC Failure)	3.668/29.479	1.178/4.261

Table 5.8: Comparison of cost/STD between the original V_ϕ algorithm and the evolved V_ϕ algorithms

All the mean cost obtained using evolved solutions are much better than the originals for the same failure situations. Standard deviations also generally improved. The evolved solutions surpassed the originals.

Cost/STD values from the original V_α algorithm for the case of diagonal sensor pair failure and from the original V_ϕ algorithm for the case of adjacent sensor pair failure case are relatively good and suggest that they may be still useable for one sensor failure cases. However, there is a contradiction with the fitness for the one sensor failure case obtained from the original

algorithm. With only one sensor failure, the fitness is expected to be better than two sensor failure cases, but actually, a deterioration is observed for both V_α and V_ϕ solutions. This may be because the original algorithms become *unbalanced* as there is one more sensor on one side of the plate than on the other. Thus, Equation 5.57 becomes Equation 5.58 for the sensor B failure as the output from the sensor B output is forced to be zero. For diagonal sensor pair BC failure case, Equation 5.57 becomes Equation 5.59, providing information about the polarity of α (whether it is in the top half of the plate or in the bottom half). Thus, the fitness of the diagonal sensor pair failure case solution is better than the one sensor failure case.

$$V_\alpha = (V_A + V_B) - (V_C + V_D) \tag{5.57}$$

If sensor B fails,

$$V_\alpha = V_A - (V_C + V_D) \tag{5.58}$$

Furthermore, if sensor C fails,

$$V_\alpha = V_A - V_D \tag{5.59}$$

Similarly for V_ϕ, Equation 5.60 becomes Equation 5.61 for the case of sensor A failure and becomes Equation 5.62 for the adjacent sensor pair AB failure. Equation 5.62 provides information about the polarity of ϕ thus the fitness of the adjacent sensor pair failure case solution is better than one sensor failure case.

$$V_\phi = (V_A + V_D) - (V_B + V_C) \tag{5.60}$$

If sensor A fails,

$$V_\phi = V_D - (V_B + V_C) \tag{5.61}$$

Furthermore, if sensor B fails,

$$V_\phi = V_D - V_C \tag{5.62}$$

The actual outputs, V_α and V_ϕ, of the evolved solutions under failure conditions are compared with the expected values when all the sensors are working correctly and are displayed in Figures 5.31 and 5.32. For the evolved solutions, one sensor failure solutions have the same responses as the diagonal sensor pair failure solution for V_α and adjacent solution for V_ϕ. Thus, responses from the one sensor failure case solutions are not included. For completeness, Figures 5.33 and 5.34 show the results of failures (zeroed sensor outputs) when the original algorithms are used.

The evolved solutions provide better estimations of the offset error voltages than the original algorithms. Both the V_α and V_ϕ given by the solutions for one sensor, adjacent sensor pair and diagonal sensor pair failure cases, track the nominal, ideal values reasonably well. The other solutions do not track well at all. The graphs coincide with the associated costs (Table 5.6). As for the original solutions, although some solution output patterns look similar to the ideal pattern, their polarity is not necessarily the same. In conclusion, the evolved solutions do show a definite improvement over the original algorithms.

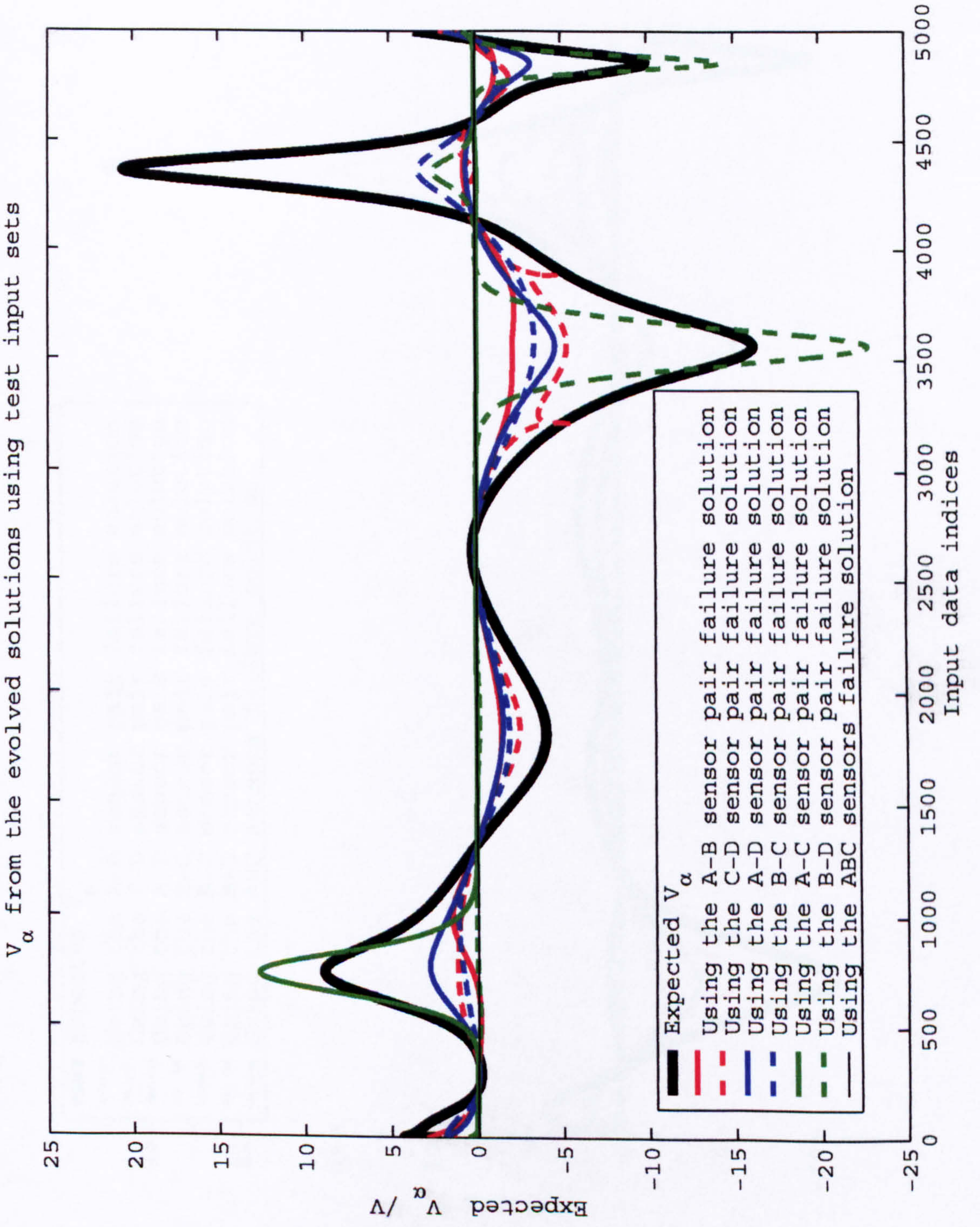


Figure 5.31: V_α from the evolved solutions

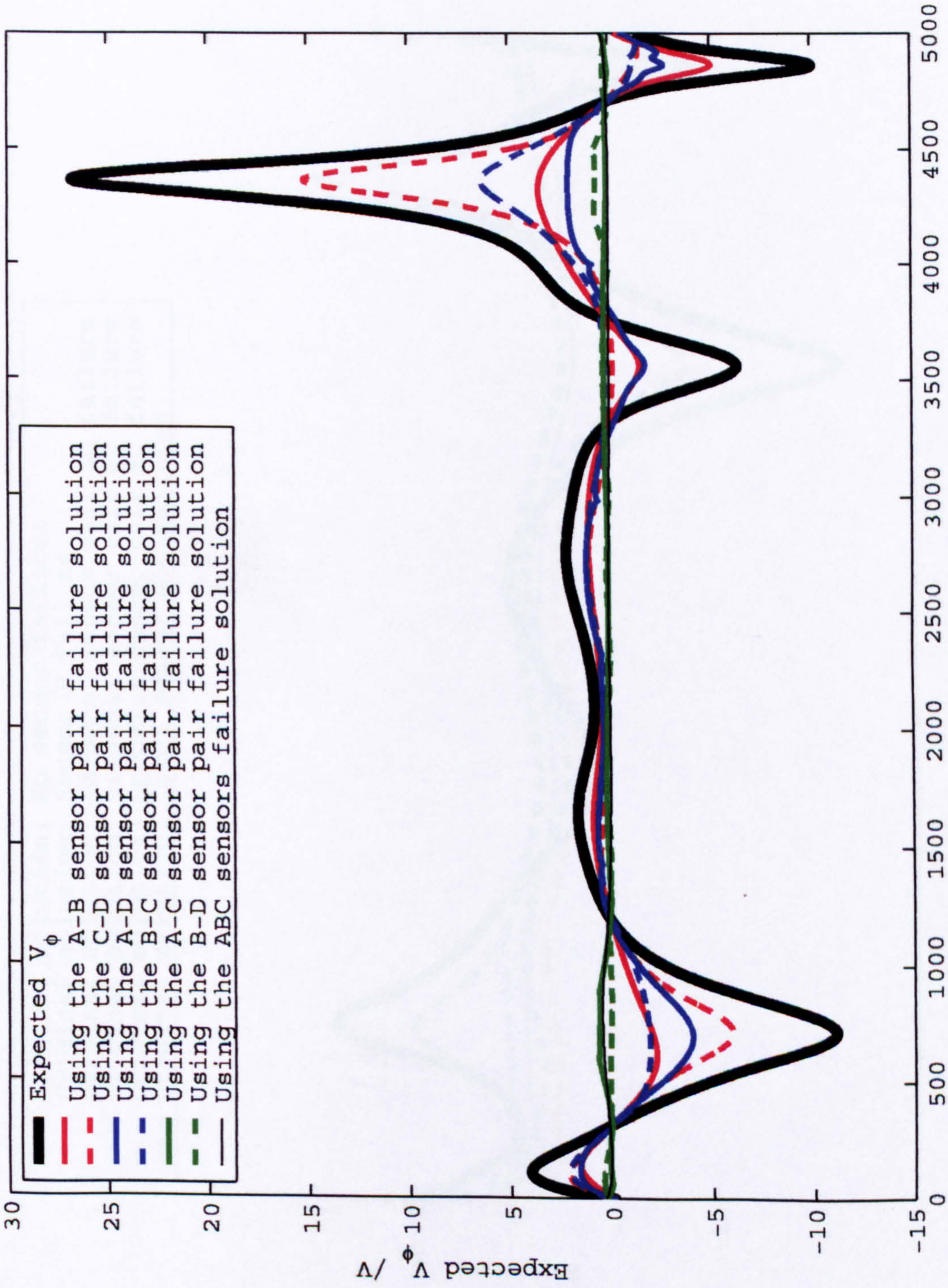


Figure 5.32: V_ϕ from the evolved solutions

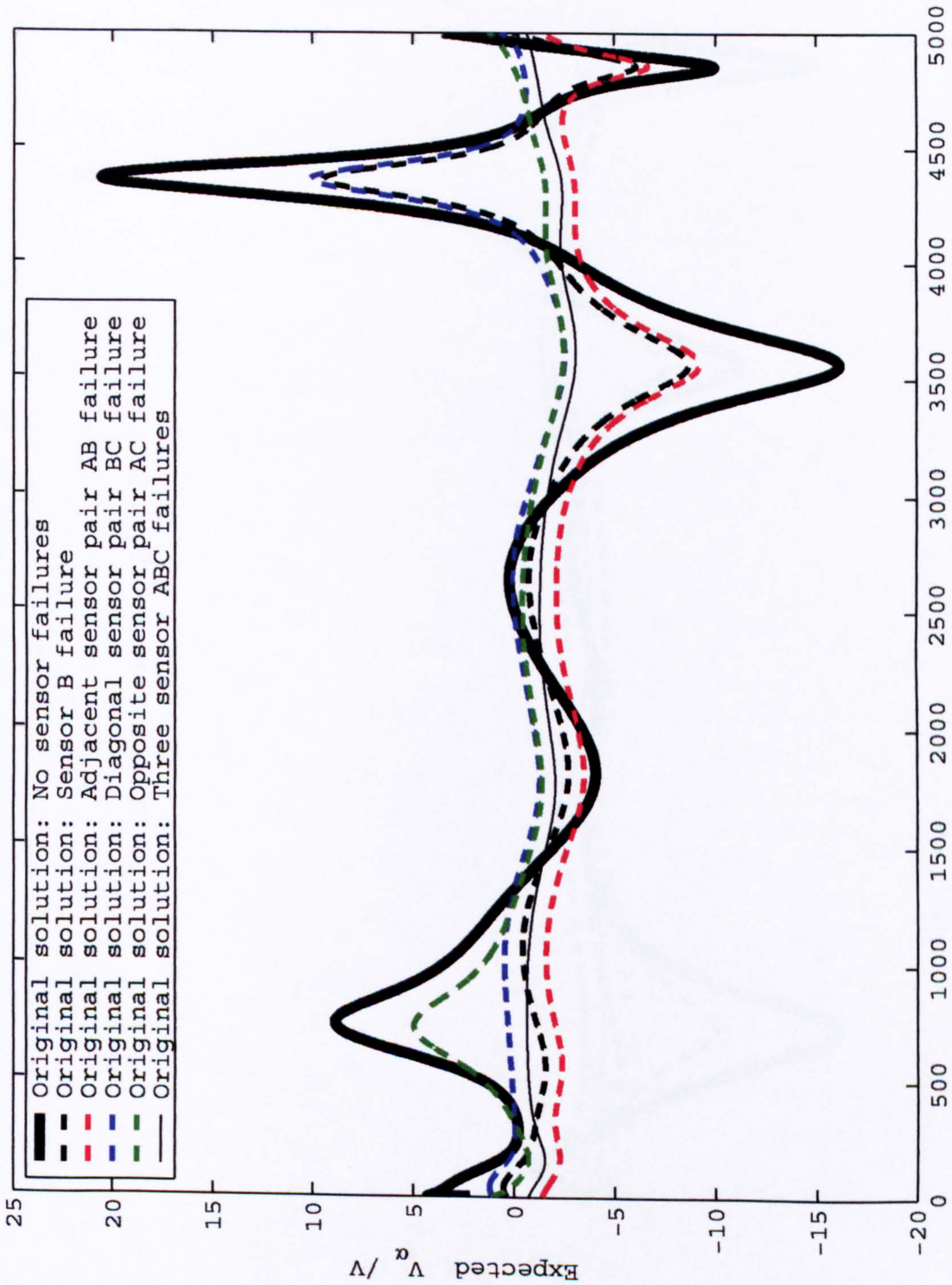


Figure 5.33: V_α from the original algorithms

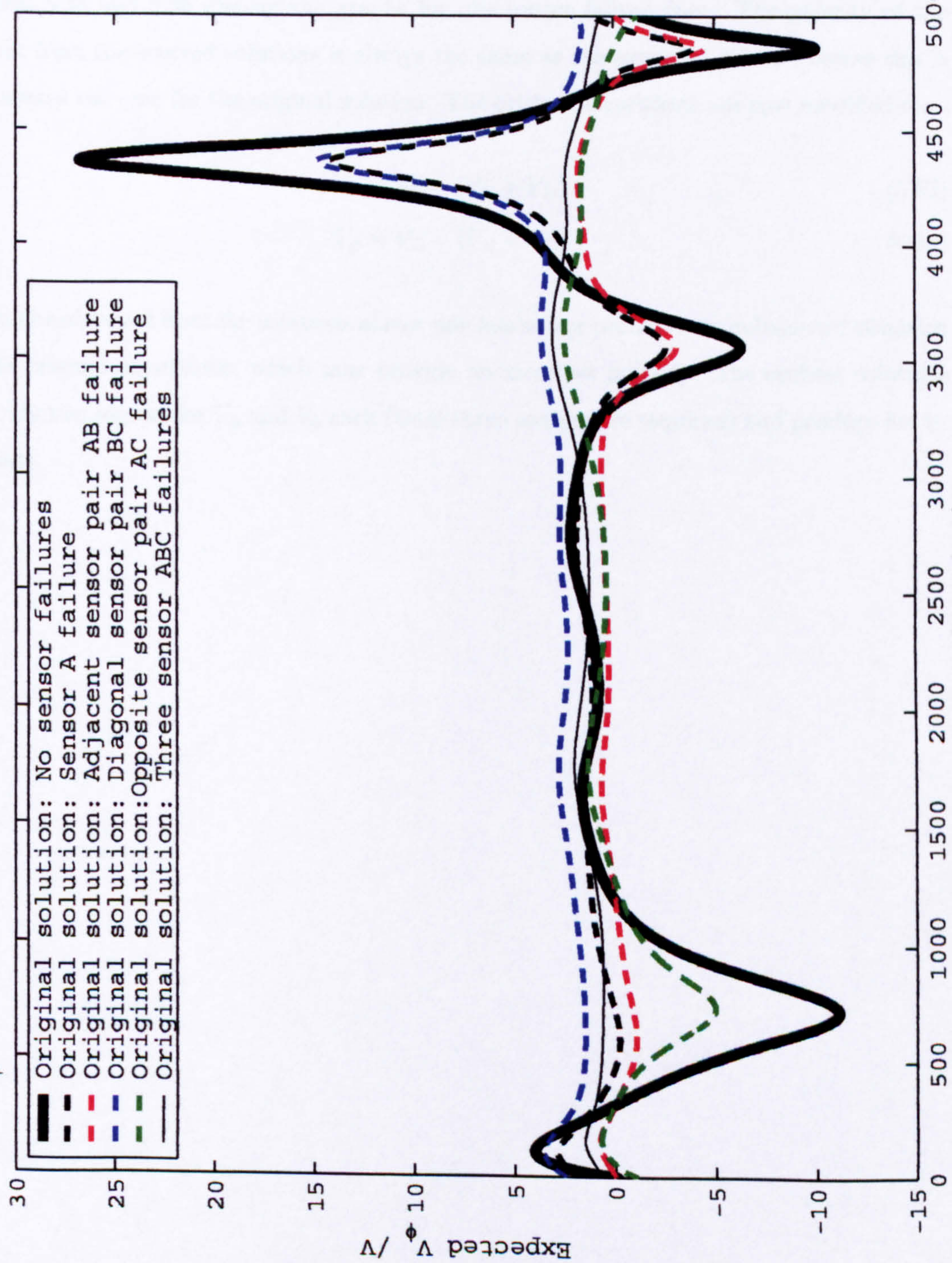


Figure 5.34: V_ϕ from the original algorithms

One Sensor Failure Case

Figures 5.35 and 5.36 display the graphs for one sensor failure case. The polarity of the output from the evolved solutions is always the same as the expected value, whereas this is not always the case for the original solution. The original algorithms are now modified to:

$$V_\alpha = V_B - (V_C + V_D) \tag{5.63}$$

$$V_\phi = V_D - (V_B + V_C) \tag{5.64}$$

As can be observed from the solutions above, one less sensor produces an unbalanced situation in the original algorithms, which may provide an incorrect polarity. The evolved solutions use only two sensors for V_α and V_ϕ each (total three sensors are required) and produce better outputs.

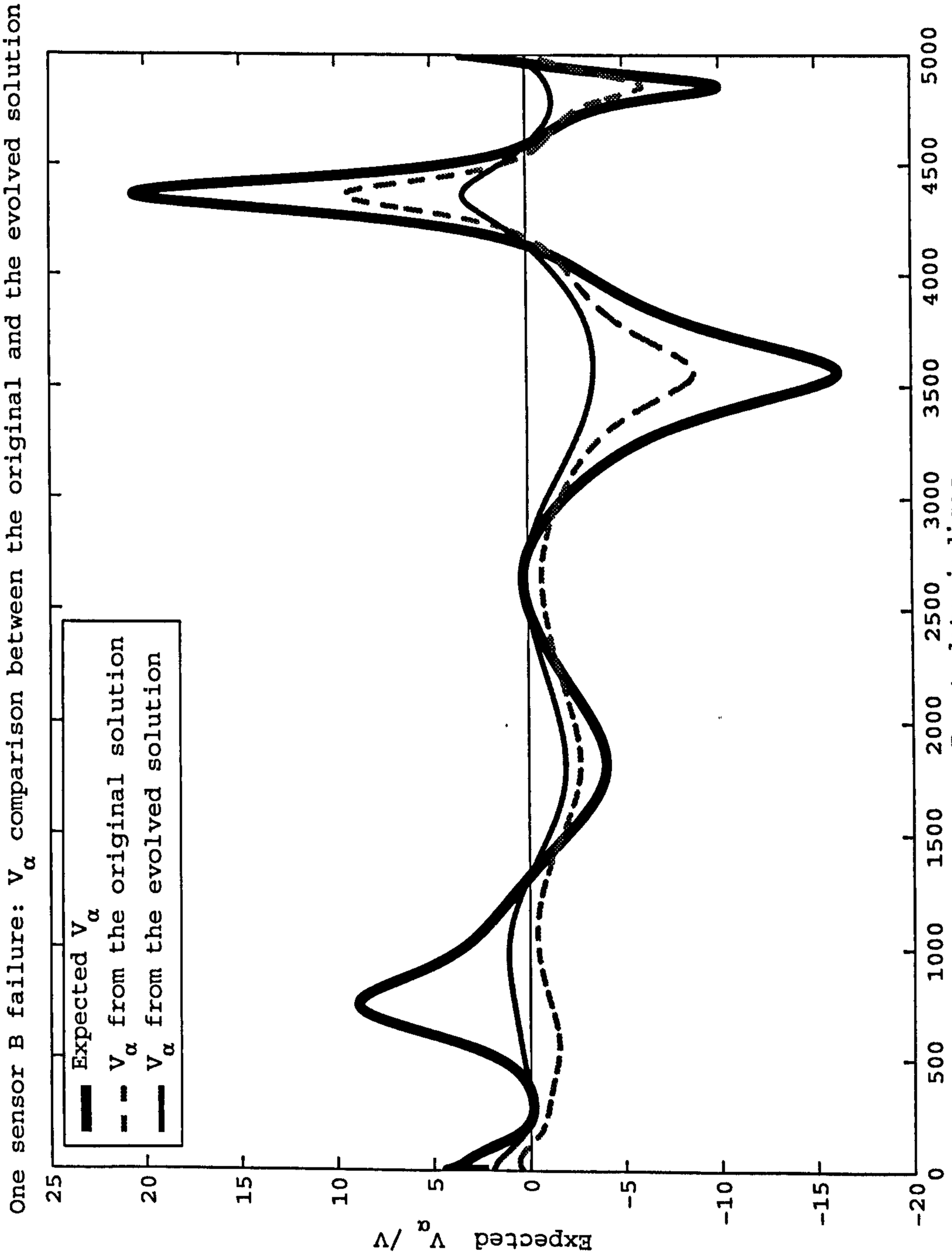


Figure 5.35: Sensor B failure: V_α comparisons between the original and the evolved solutions

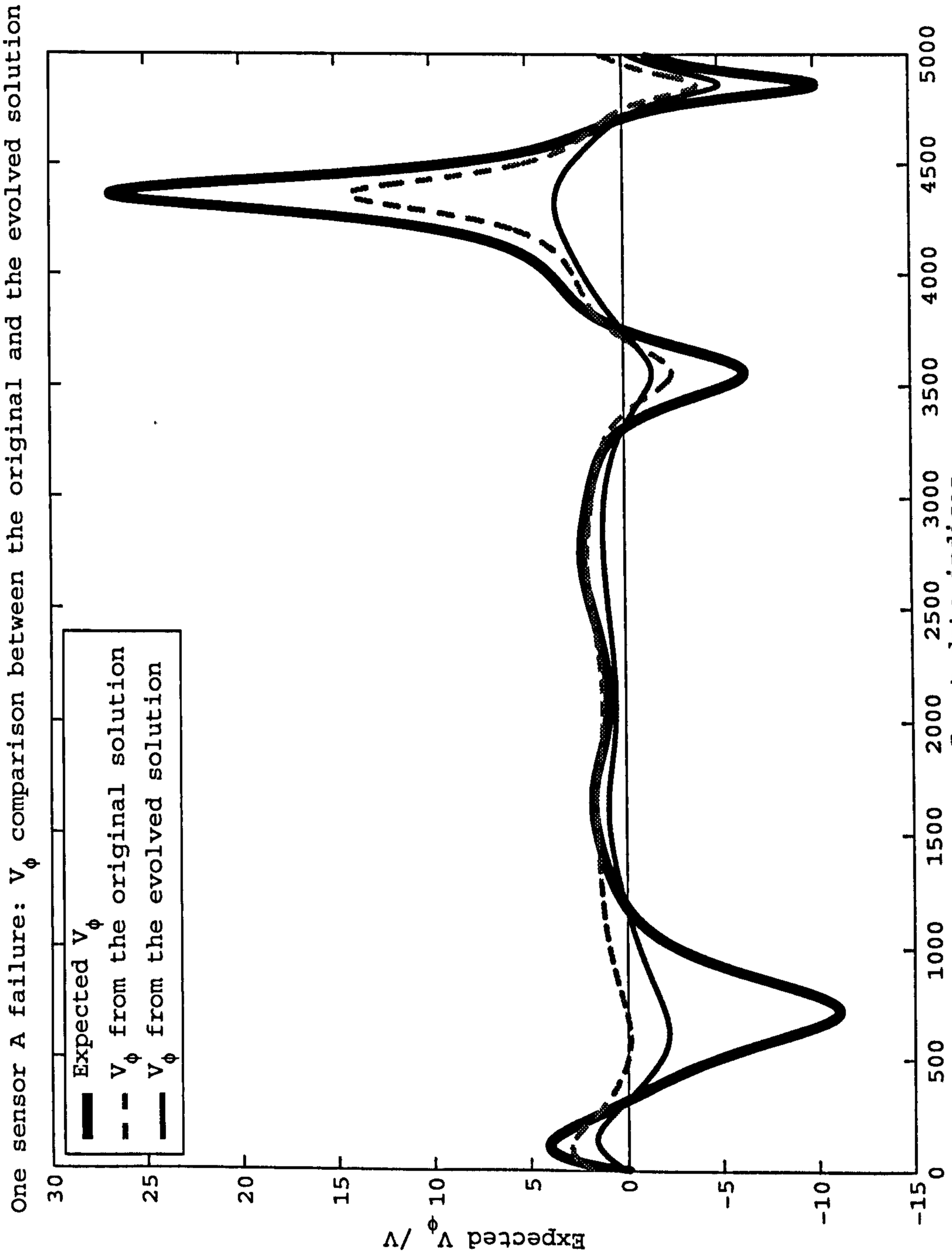


Figure 5.36: Sensor A failure: V_ϕ comparisons between the original and the evolved solutions

Adjacent Sensor Pair Failure Case

Solution outputs are displayed in Figures 5.37 and 5.38. They show the evolved solutions have better response than the original ones. The original algorithms with sensors A and B failure cause them to be modified into:

$$V_\alpha = -(V_C + V_D) \tag{5.65}$$

$$V_\phi = V_D - V_C \tag{5.66}$$

The difference between sensors C and D output voltages indicates the existence of V_ϕ. Thus the polarity of the ϕ can be implied even using the original algorithm. Therefore, the original algorithm for V_ϕ would still allow reasonably good error estimation. This is also indicated in Table 5.8. However, V_α is always negative as Equation 5.65 states, so not even the polarity can be implied.

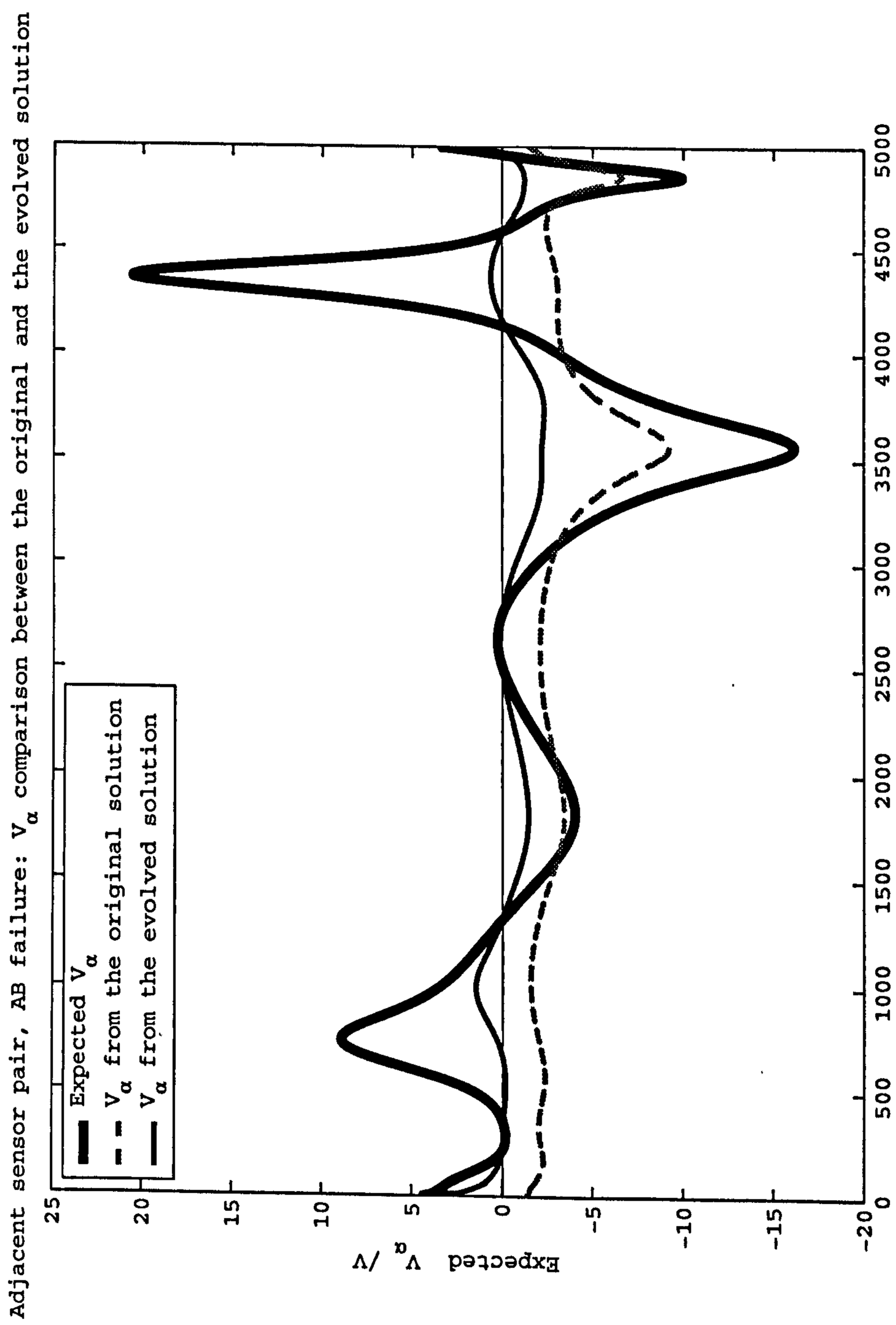


Figure 5.37: Adjacent sensor pair AB failure: V_α comparisons between the original and the evolved solutions

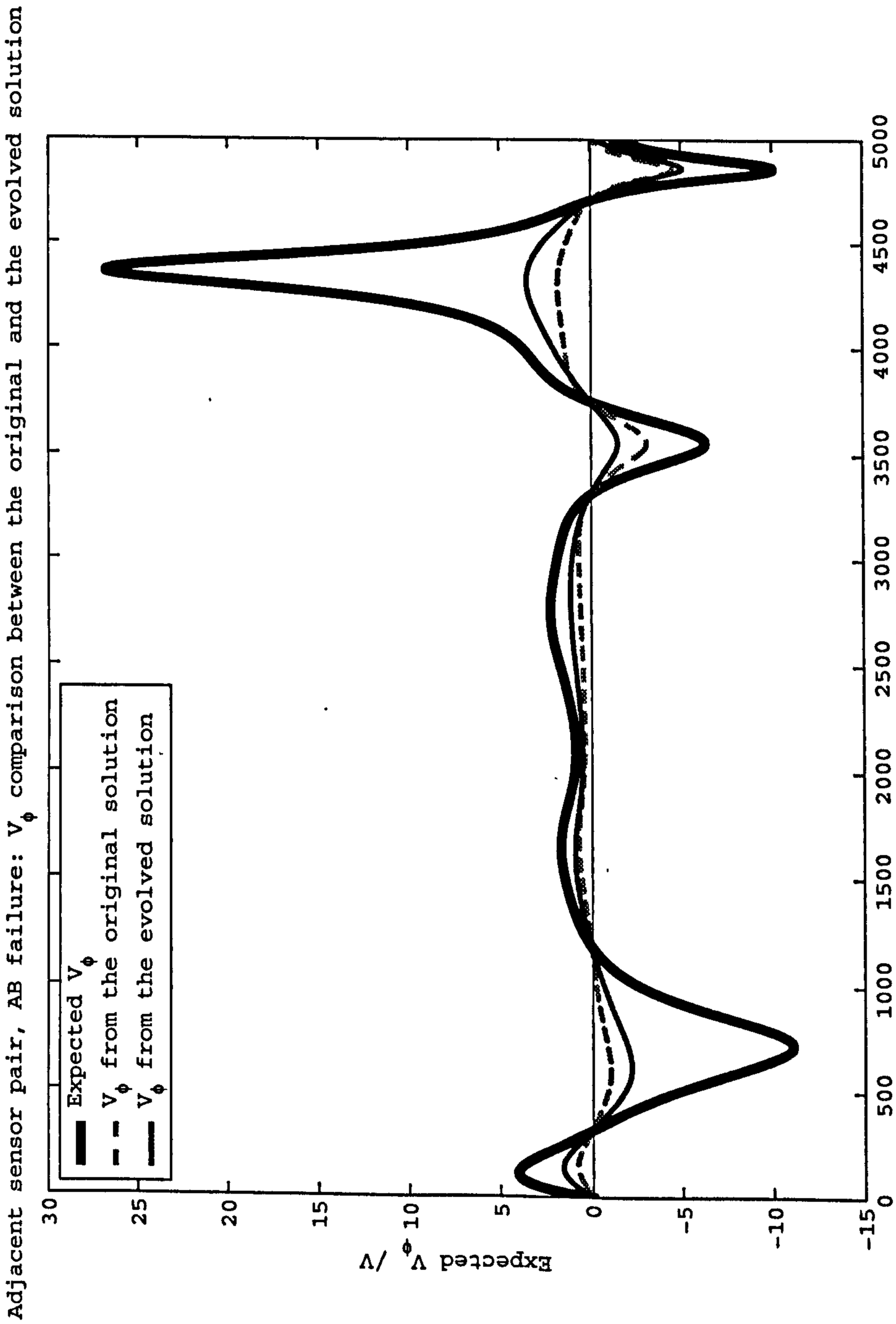


Figure 5.38: Adjacent sensor pair AB failure: V_ϕ comparisons between the original and the evolved solutions

Diagonal Sensor Pair Failure Case

Similar to the adjacent sensor pair failure case, the evolved solution responses to V_α and V_ϕ are better than the original algorithms as can be seen in Figures 5.39 and 5.40. The original algorithms are modified into Equations 5.67 and 5.68.

$$V_\alpha = V_A - V_D \tag{5.67}$$

$$V_\phi = V_A + V_D \tag{5.68}$$

The polarity of V_α is correct for the whole duration, so the cost is reasonably good as can be seen from Table 5.7. However, the polarity of V_ϕ will always be positive which can cause a catastrophic failure.

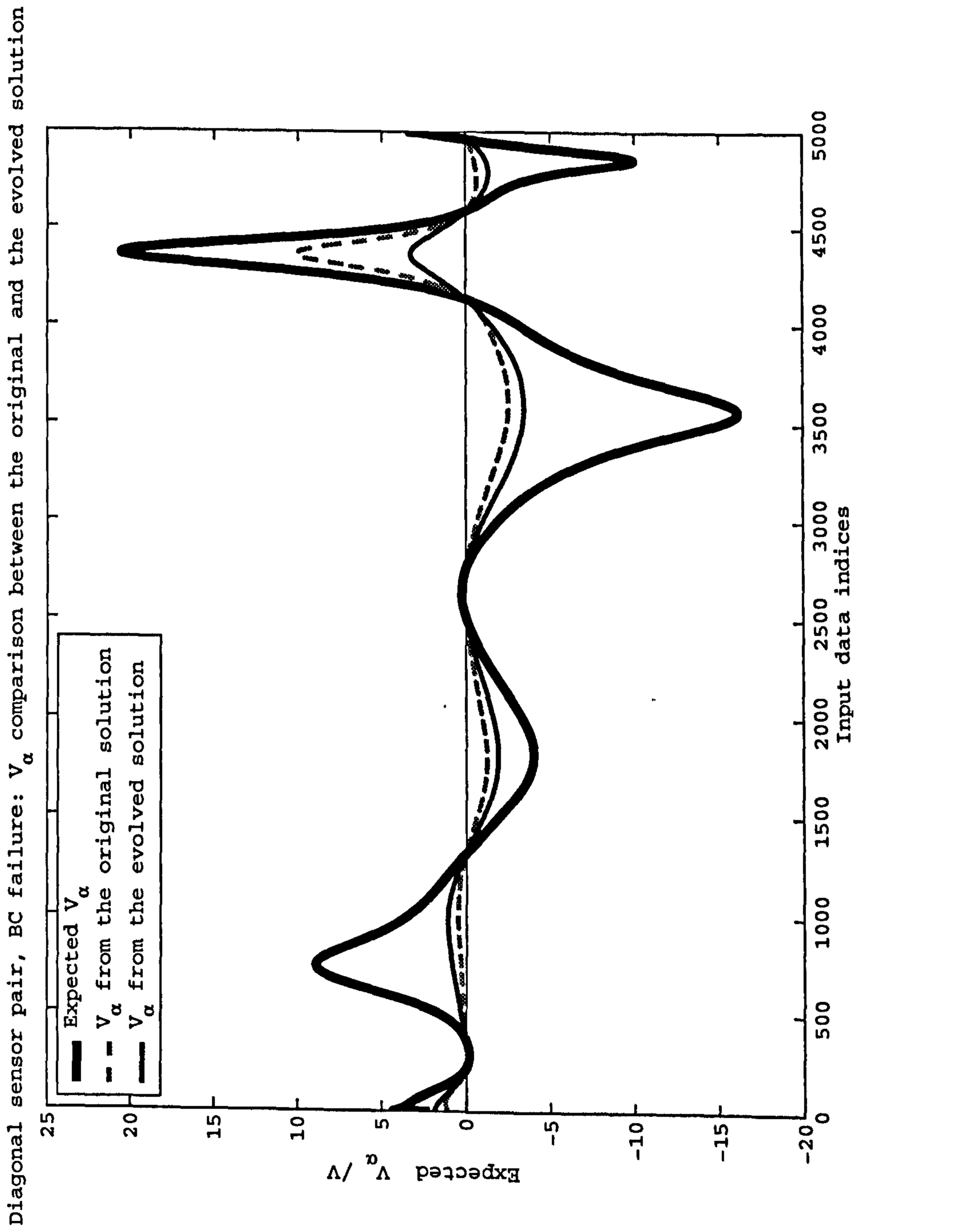


Figure 5.39: Diagonal sensor pair BC failure: V_α comparisons between the original and the evolved solutions

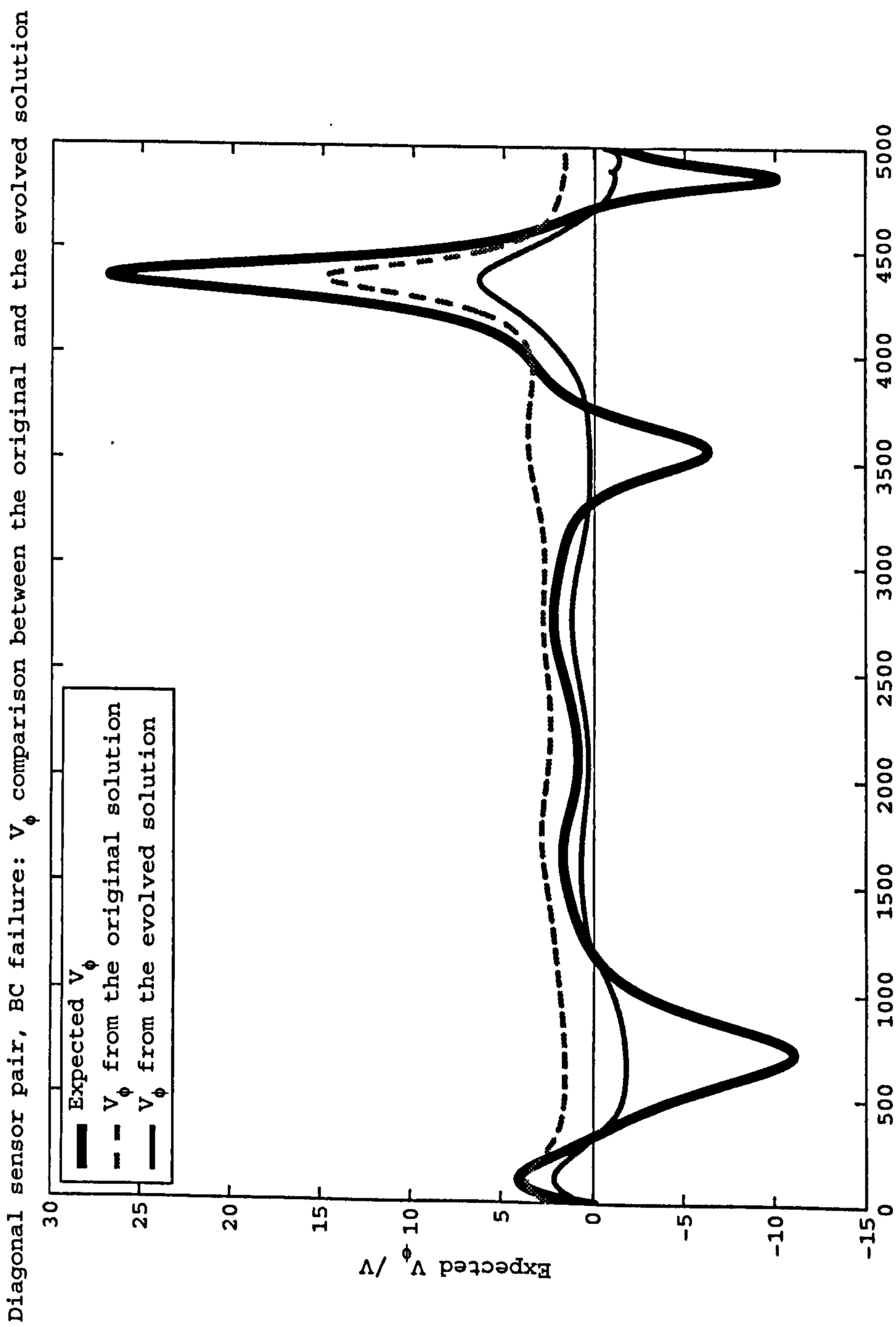


Figure 5.40: Diagonal sensor pair BC failure: V_ϕ comparisons between the original and the evolved solutions

Opposite Sensor Pair Failure Case

As explained using Figure 5.18 previously, the opposite sensor pair failures cause an impossible situation for the Shaky Hand to estimate the correct wire positions. Equations 5.69 and 5.70 cannot provide any information regarding α or ϕ. Figures 5.41 and 5.42 support this theory. This deficiency is also highlighted in the evolved solutions. Fitness from the evolved solution is better than that from the original algorithm because the solution is kept at close to zero most of the time. This way, the relative difference between the expected values and the evolved solution are better balanced than the original case. The Shaky Hand CGP has evolved solutions to satisfy only the best fitness. The solution selected was the one which gave the most balanced outputs.

$$V_\alpha = V_B - V_D \tag{5.69}$$

$$V_\phi = V_D - V_B \tag{5.70}$$

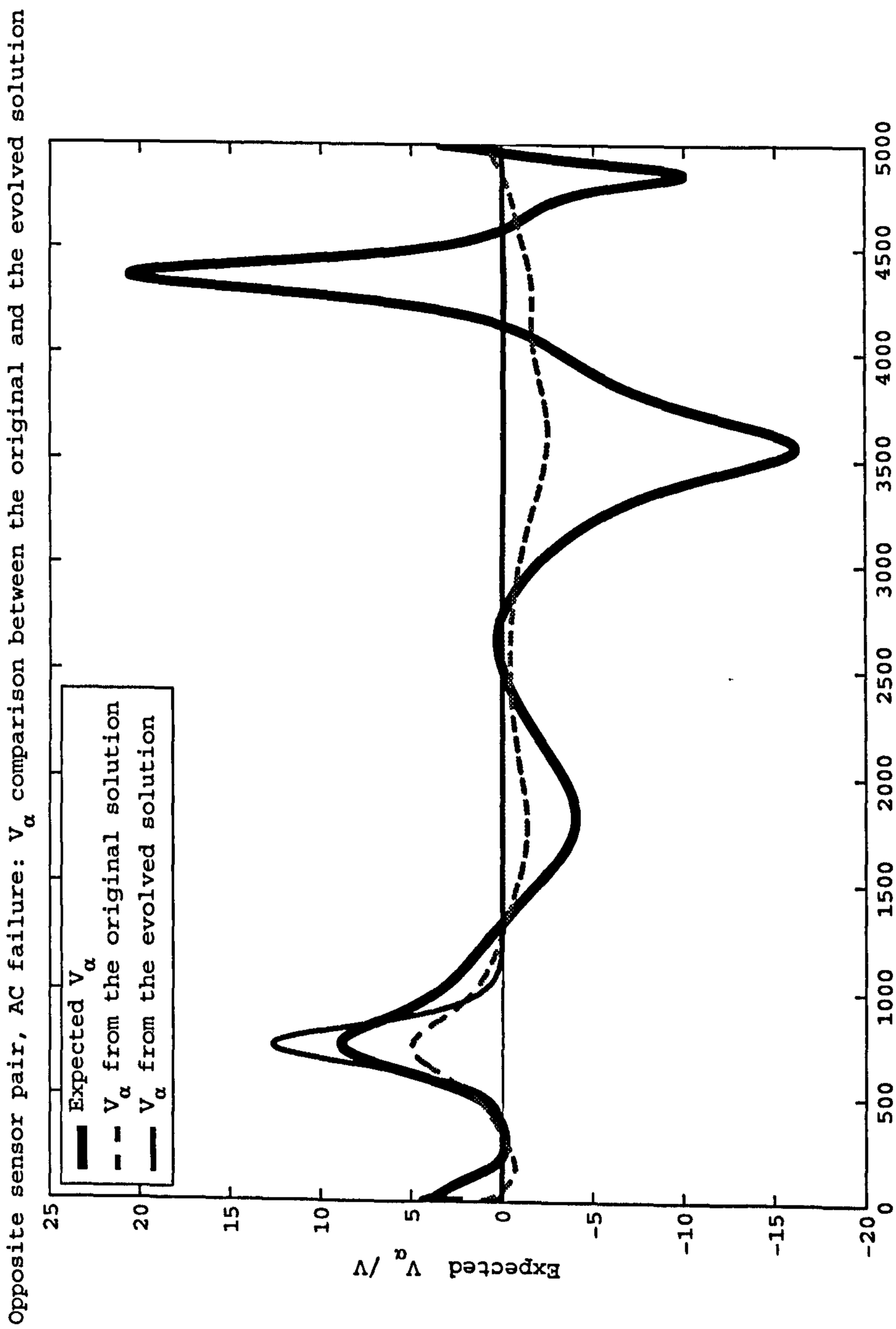


Figure 5.41: Opposite sensor pair AC failure: V_α comparisons between the original and the evolved solutions

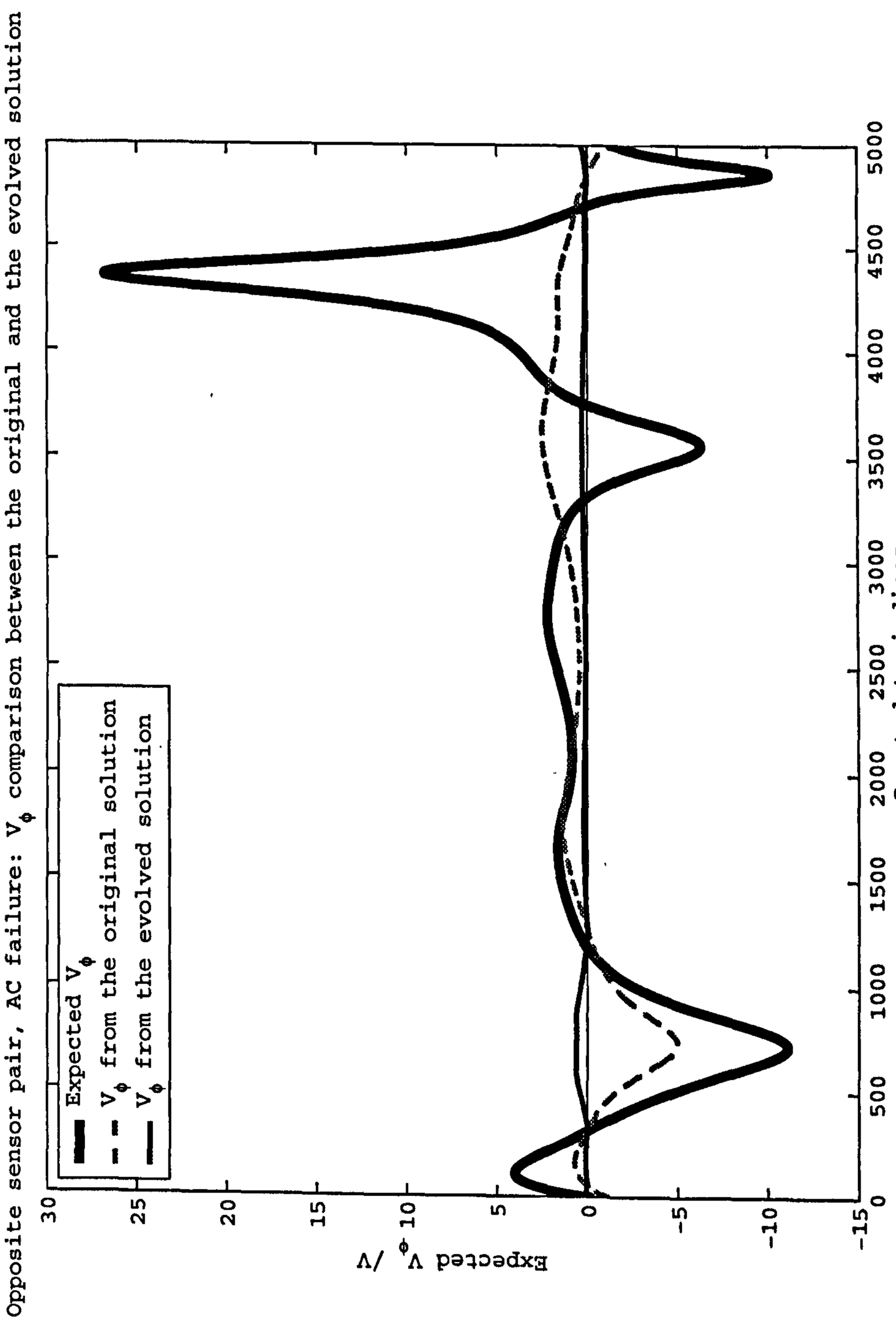


Figure 5.42: Opposite sensor pair AC failure: V_ϕ comparisons between the original and the evolved solutions

Three Sensors Failure Case

The outputs from the original algorithms are kept either always positive or always negative as Equations 5.71 and 5.72 show. Similar to the opposite sensor pair failure case, estimation of V_α and V_ϕ is impossible using only one sensor. The outputs from the evolved solutions are also kept near to zero for the majority of the profile.

$$V_\alpha = -V_D \tag{5.71}$$

$$V_\phi = V_D \tag{5.72}$$

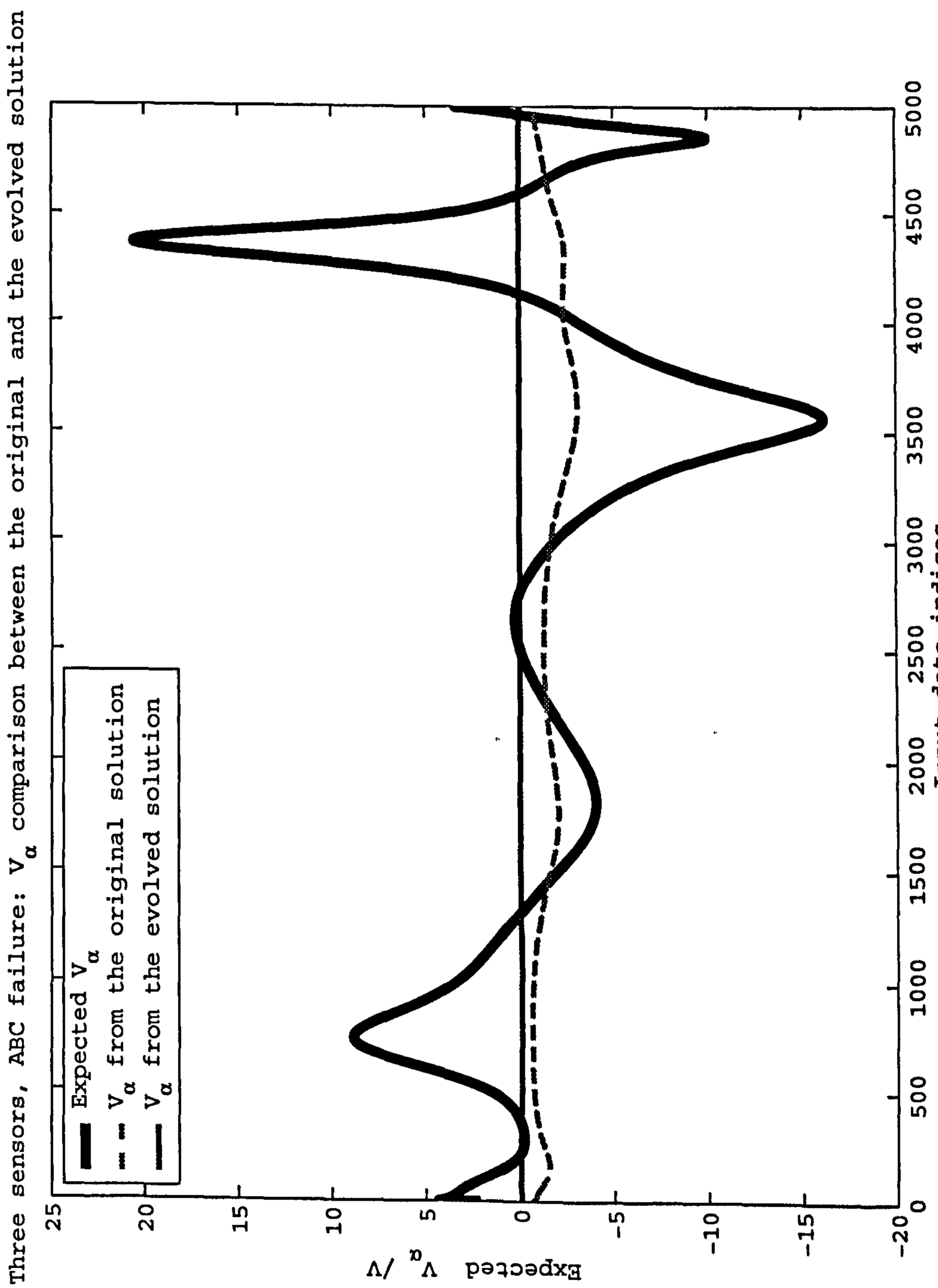


Figure 5.43: Three sensors ABC failure: V_α comparisons between the original and the evolved solutions

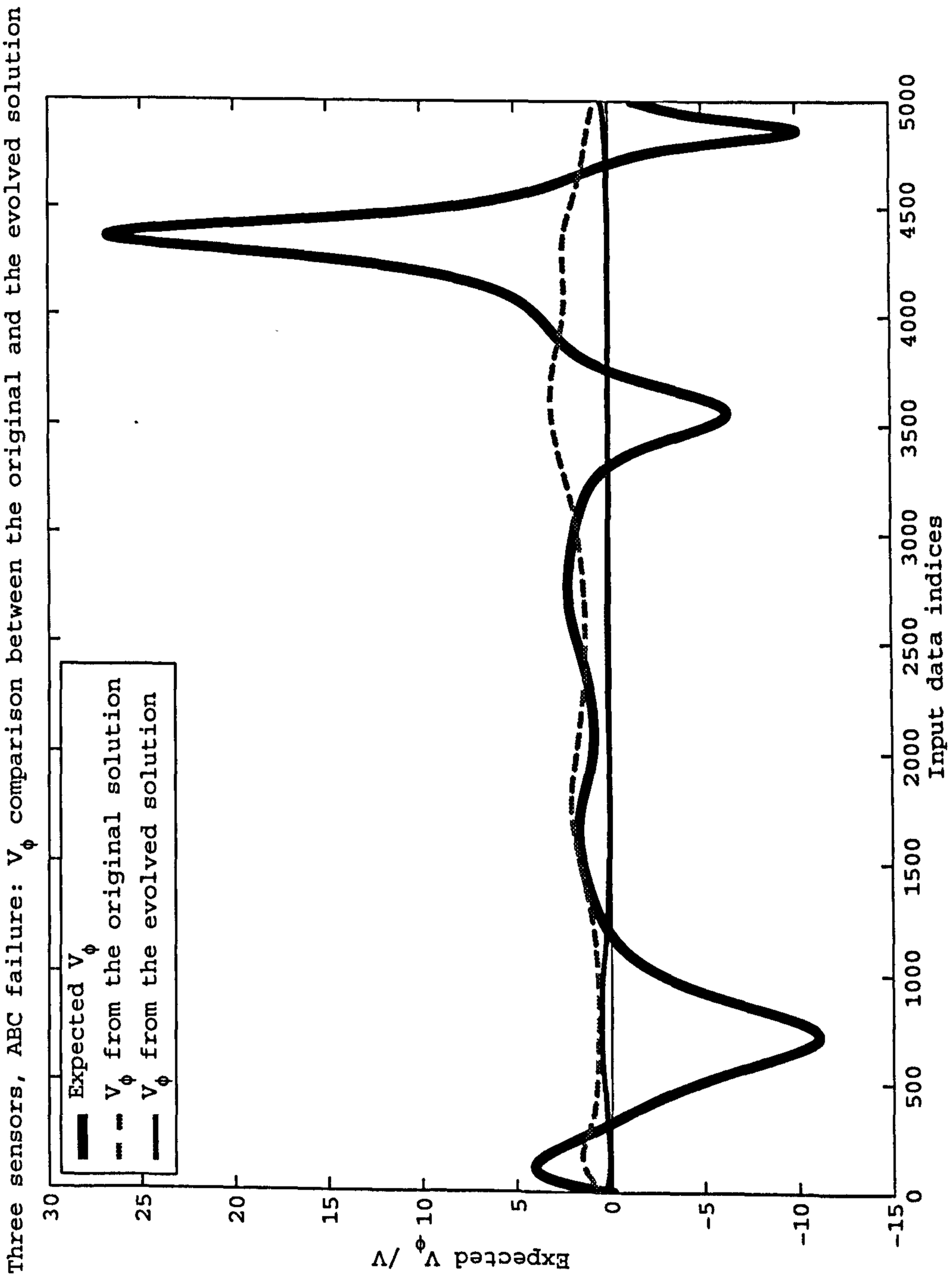


Figure 5.44: Three sensors ABC failure: V_ϕ comparisons between the original and the evolved solutions

5.6.4 Further Evaluation of the Evolved Solutions

In order to look into the features of the evolved solutions, V_α and V_ϕ from the evolved solutions are evaluated for the wire positions as shown in Figure 5.45. The corresponding solutions, V_α and V_ϕ, are summarised in Table 5.9.

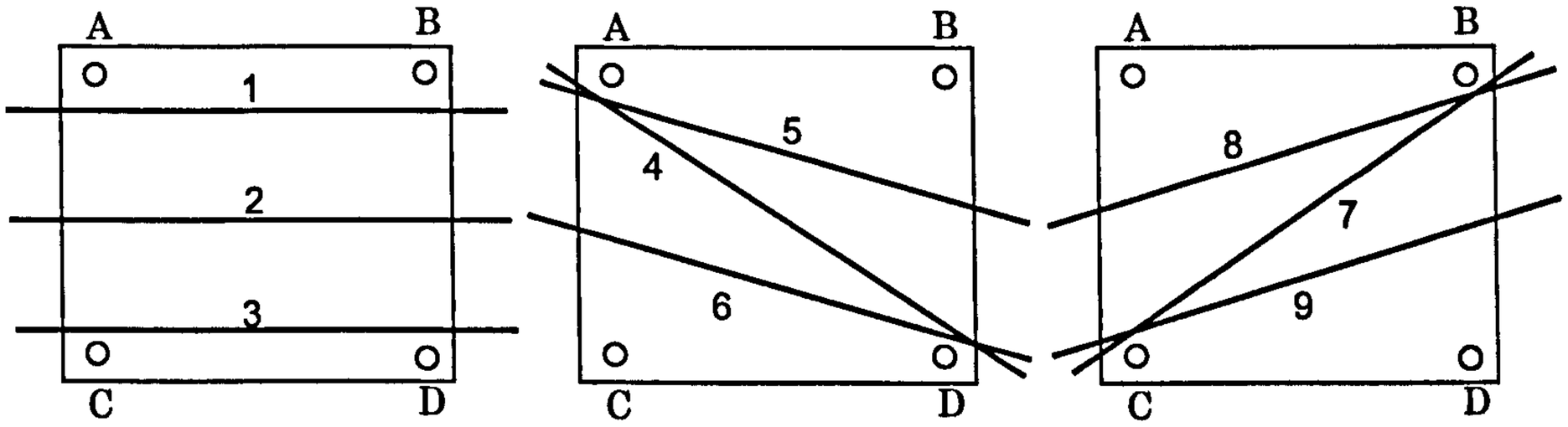


Figure 5.45: Different wire coordinates for evaluating the evolved solutions

wire position	V_A V_B V_C V_D	$V_{\alpha Ideal}$	$V_{\phi Ideal}$	$V_{\alpha AB}$	$V_{\phi AB}$	$V_{\alpha BC}$	$V_{\phi BC}$	$V_{\alpha AC}$	$V_{\phi AC}$	$V_{\alpha ABC}$	$V_{\phi ABC}$
1	5.00 5.00 0.56 0.56	8.88	0	5.30	0	4.38	1.92	12.73	0.56	-0.08	0.58
2	1.00 1.00 1.00 1.00	0	0	0	0	0	0	0	0	-0.04	0.17
3	0.56 0.56 5.00 5.00	-8.88	0	-3.10	0	-4.38	0.93	-0.51	0.29	-0.05	0.24
4	5.00 0.56 0.56 5.00	0	8.88	2.44	8.82	0	5.39	-0.51	0.29	-0.05	0.24
5	5.00 1.00 0.56 1.00	4.44	4.44	2.81	1.23	3.22	3.01	0	0	-0.04	0.17
6	1.00 0.56 1.00 5.00	-4.44	4.44	-1.42	4.80	-3.22	2.03	-0.51	0.29	-0.05	0.24
7	0.56 5.00 5.00 0.56	0	-8.88	-0.79	-8.82	0	-3.11	12.73	0.56	-0.08	0.58
8	1.00 5.00 1.00 0.56	4.44	-4.44	2.23	-1.23	1.16	-1.91	12.73	0.56	-0.08	0.58
9	0.56 1.00 5.00 1.00	-4.44	-4.44	-1.14	-4.80	-1.16	-1.08	0	0	-0.04	0.17

Table 5.9: Evolved solution outputs for different wire positions

In Table 5.9, all the subscript letters beside V_α and V_ϕ represent which evolved solutions are used. For example, $V_{\alpha Ideal}$ is the V_α value obtained from the ideal non-sensor failure case, and the original solutions are used. Similarly, $V_{\alpha AB}$ is the V_α value obtained from the sensors A and B failure case and the appropriate evolved solutions are used.

The one sensor failure case solutions are excluded from the table because they share the solutions with the diagonal sensor pair failure solution for V_α, and the adjacent sensor pair failure solution for V_ϕ.

A maximum voltage of 5V, and a minimum voltage of 0.56V are assumed from the sensors. In the middle, 1V is generated.

From Table 5.9, followings can be observed:

1. For All the Solutions:
   - Estimation of the offset errors are correct when the wire sits at the ideal zero offset position (position 1) for all the solutions except for the three sensor failure case.
2. From Adjacent Sensor Pair AB Failure Solution (4th column):
   - A good estimate of V_ϕ is obtained. The polarity of V_ϕ is always correct.
   - V_ϕ solution is accurate when the ideal V_ϕ is zero.
   - Estimate of V_ϕ at position 6 is better than at position 5. Estimate at position 9 is better than at position 8. Estimate at position 4 and 7 are equal. This implies that the estimation of the V_ϕ error signals is more accurate when the wire position is nearer to the working sensor pair C and D.
   - Estimate of $V\alpha$ become unreliable at extreme ϕ (position 4 and 7), otherwise the polarity is always correct.
   - Estimate of $V\alpha$ at position 1 is better than position 3. Estimate at position 5 is better than at position 6. Estimate at position 8 is better than at position 9. Therefore, quite unexpectedly, for V_α adjacent solution, a better estimation of the offset is observed away from the working sensors. Since the solution only utilises the two working sensor outputs, the solution can be said to work better with the smaller values of the working sensors.
3. From Diagonal Sensor Pair BC Failure Solution (5th column):
   - A good estimate of V_α is obtained. The polarity of V_α is always correct.
   - V_α solution is accurate when the ideal V_α is zero.
   - Estimate of V_α at position 5 is better than at position 8. Estimate at position 6 is better than at position 9. It suggests the estimation of the lateral offset is more accurate when the wire is nearer to the working sensor pair A and D.

- Estimate of V_ϕ at position 4 is better than at position 7. Estimate at position 5 is better than at position 8. Estimate at position 6 is better than at position 9. It suggests that the estimation of the V_ϕ error signals is also more accurate when the wire position is nearer to the working sensor pair A and D.
- Estimate of $V\phi$ become inaccurate when the wire is parallel to the plate except at the ideal zero offset wire position. Otherwise the polarity is always correct.

4. From Opposite Sensor Pair AC Failure Solution (6th column):
   - Estimations are generally unreliable except at position 2.
   - The polarity of V_ϕ is always positive.
   - The remaining sensors can detect the position of the wire only when at the edge of the plate. Only three different positions are available in this case, as shown in Figure 5.18. Therefore the solutions are unreliable.
5. From Three Sensor ABC Failure Solution (7th column):
   - Estimations are unreliable.
   - Polarity of the outputs are either kept positive or negative and their values are always small, giving a flat response to any input sets. Both V_α and V_ϕ solutions are unreliable.

The Shaky Hand CGP demonstrates the ability to search for the optimum solution for this problem. With one sensor failure, the solutions have very good fitness. The diagonal sensor pair failure and the adjacent sensor pair failure solutions also have reasonably good fitness. However, if a solution does not exist, or is impossible to solve, then it produces a solution to cause least errors for the given training sets such as the opposite sensor pair failure solutions and the three sensor failure solutions.

5.7 Adapting The Solutions Into The Shaky Hand

The solutions are generic as suggested in Section 5.5.1. These generic solutions were then implemented in Simulink. The original offset error sensing algorithms inside the Simulink models were replaced by the evolved solutions. Failed sensors were forced to produce zero voltage as discussed. All the evolved solutions were then implemented in hardware to test them in practice.

5.7.1 Results of Simulink Simulation

The wire track model has been developed for the simulation purpose as described in Section 2.5.1. The virtual track consists of three different functions. Two straight lines with different gradient, and a curved line. The initial condition of the plate is set so that the plate sits at coordinate (0,0) and has an angular offset, ϕ, of +3.4 degrees. Coordinate (x,y) is equivalent to the physical displacement on the base frame and has a unit of *mm*. When the simulation starts, the position of the centre of the plate is recorded over time, then a locus of the plate is plotted. If the plate tracks the wire from the beginning to the end of the simulation then the simulation result is stated as successful, else, the simulation is deemed as unsuccessful. Another game rule states that the sensors should never touch the wire track, so the maximum separation distance between the sensors and the wire is 22.5mm. The simulation results are summarised in Table 5.10. A tick, ✓, indicates a successful simulation result, and a cross, × indicates an unsuccessful simulation result.

Number of sensor failures	Failure Case	Simulation Result
1	Sensor A failure	✓
	Sensor B failure	✓
	Sensor C failure	✓
	Sensor D failure	✓
2	Adjacent pair AB failure	✓
	Adjacent pair CD failure	✓
	Diagonal pair AD failure	✓
	Diagonal pair BC failure	✓
	Opposite pair AC failure	×
	Opposite pair BD failure	×
3	Sensors BCD failure	×
	Sensors ACD failure	×
	Sensors ABD failure	×
	Sensors ABC failure	×

Table 5.10: Simulink simulation results

Figures 5.46 to 5.52 display the results of the simulation in graphical forms. Figure 5.46 indicates the outcome of the simulation when all the sensors are functioning correctly. The dashed line on the graph is the wire track and the solid line represents the position of the centre of the plate. The lines overlap from the start to the end for the successful simulation results. Figure 5.47 shows a zoomed view in the vicinity of a track change from the first function to the second function. Figures 5.48, 5.49 and 5.50 represent the one sensor failure case, diagonal sensor pair failure case and adjacent sensor pair failure case respectively. These simulations are shown to be successful. Figures 5.48 and 5.49 are quite similar to each other and also to the ideal non-failure case. This suggests that the generic solutions for the one sensor failure cases and also the diagonal case can provide a performance which is virtually indistinguishable from the ideal non-failure case. Out of the successful two sensor failure case solutions, the diagonal sensor pair failure solutions provide better performance. The overshoot in the graph is smaller and is better damped. This follows the pattern of the mean cost as shown in Table 5.6. The diagonal sensor pair failure solutions have better fitness overall than the adjacent sensor pair failure solutions. The other graphs indicate that the plate position is totally off the wire track. The solutions for opposite sensor pair failure cases and three sensor failure cases would fail.

For completeness, Simulink simulation using the original algorithms were also carried out assuming there are failed sensors. All the simulation results were unsuccessful as predicted.

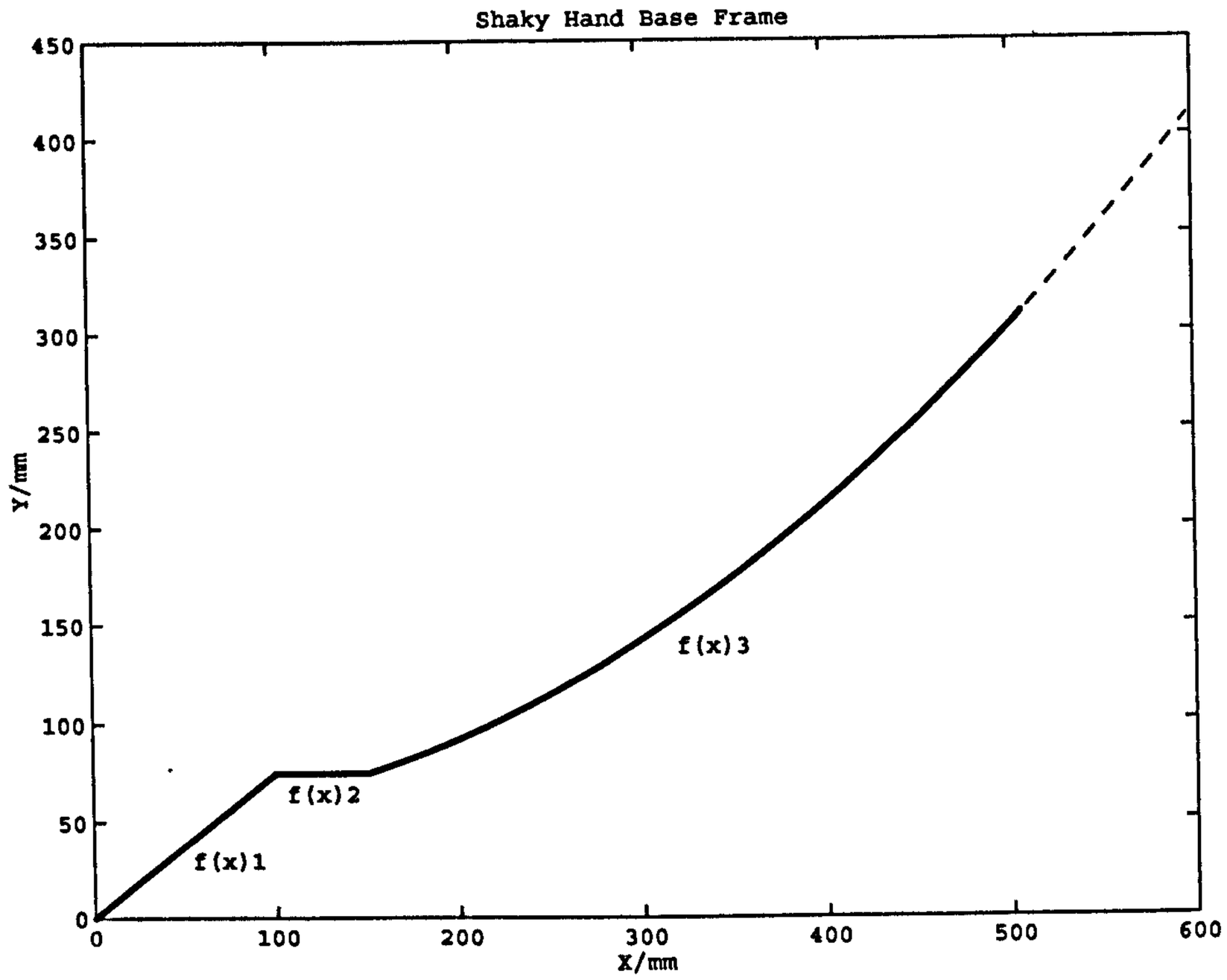


Figure 5.46: Simulink simulation: using the original algorithm

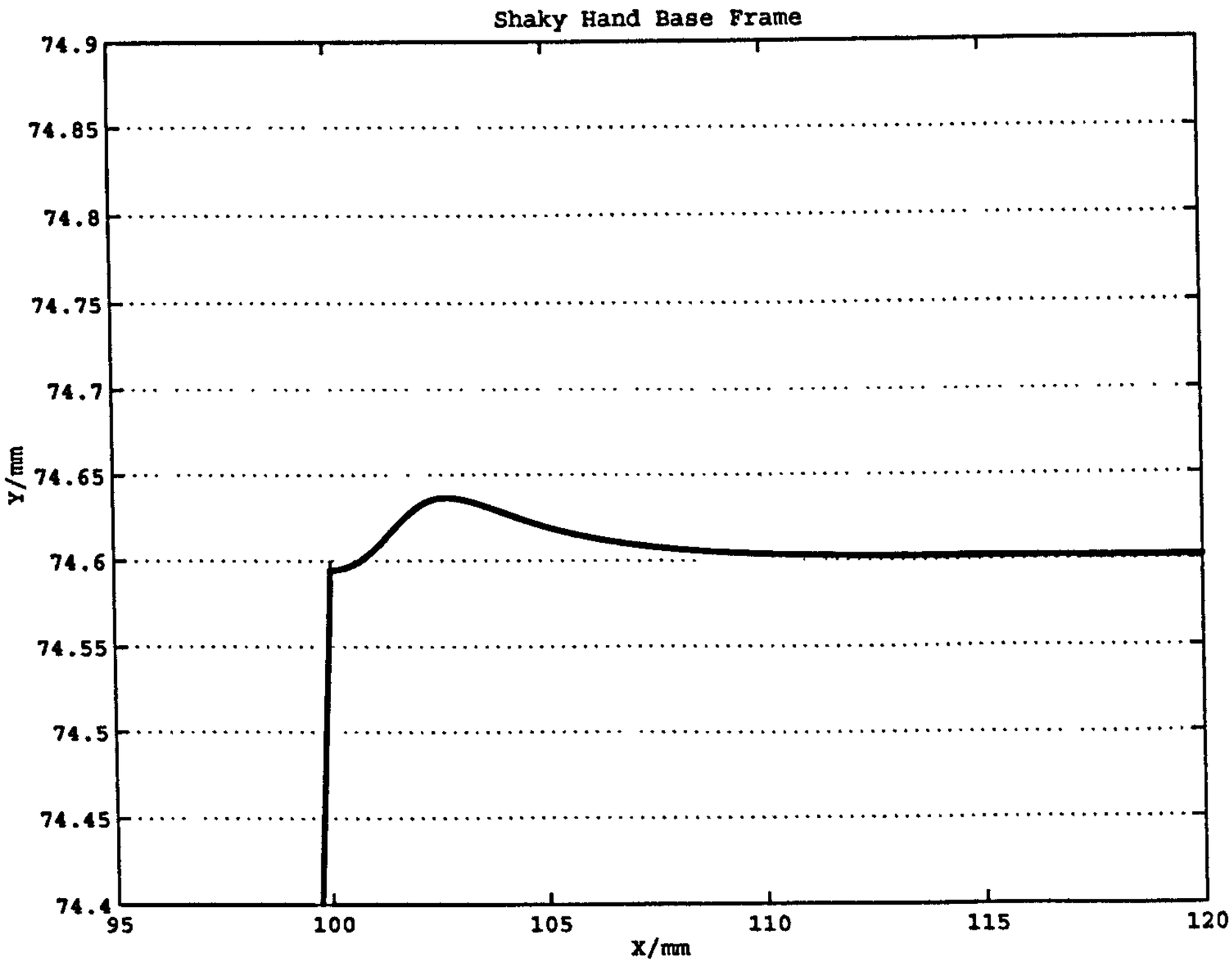


Figure 5.47: Simulink simulation: closer look of Figure 5.46

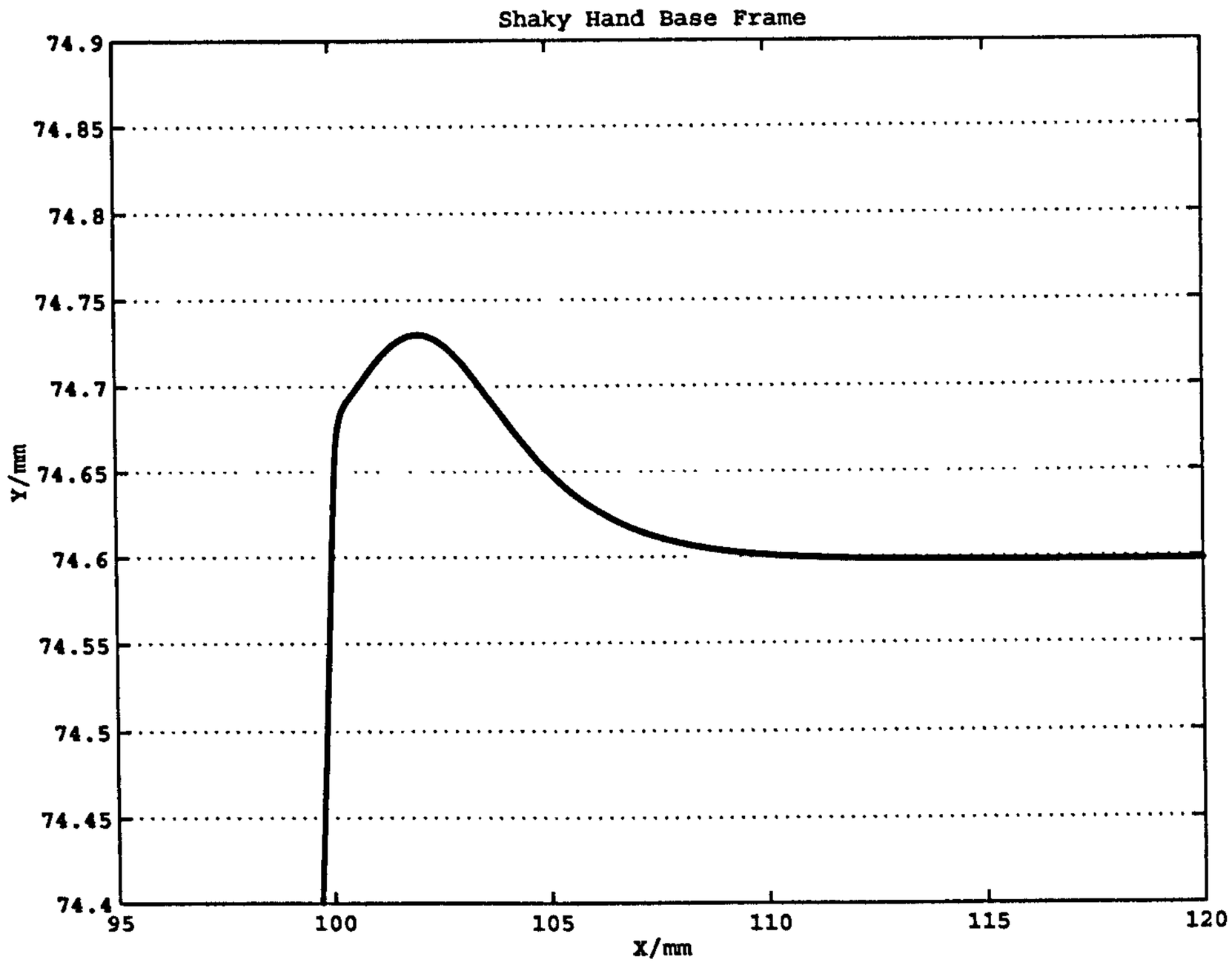


Figure 5.48: Simulink simulation: using the sensor C failure case solution

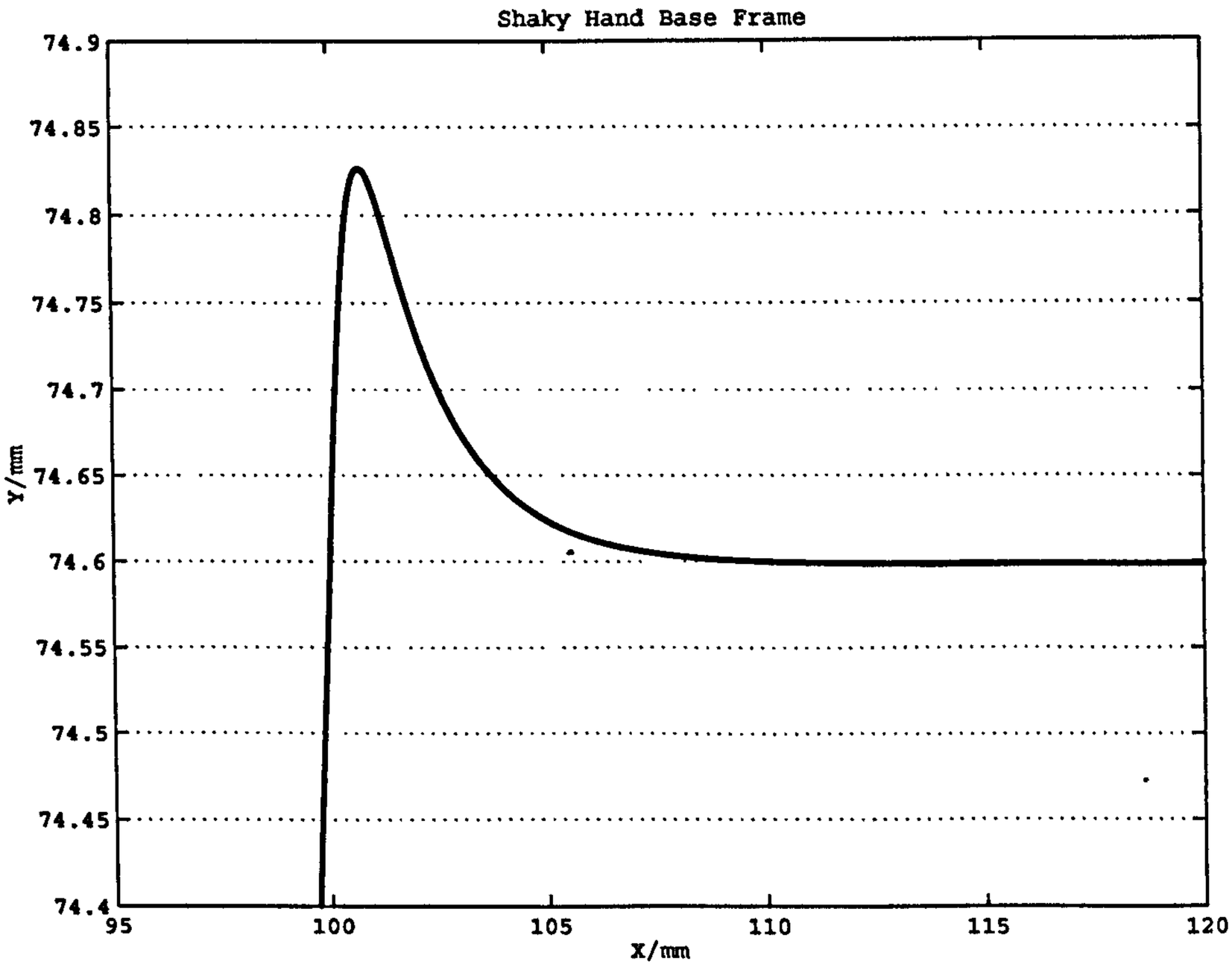


Figure 5.49: Simulink simulation: using the sensor B&C failure case solution

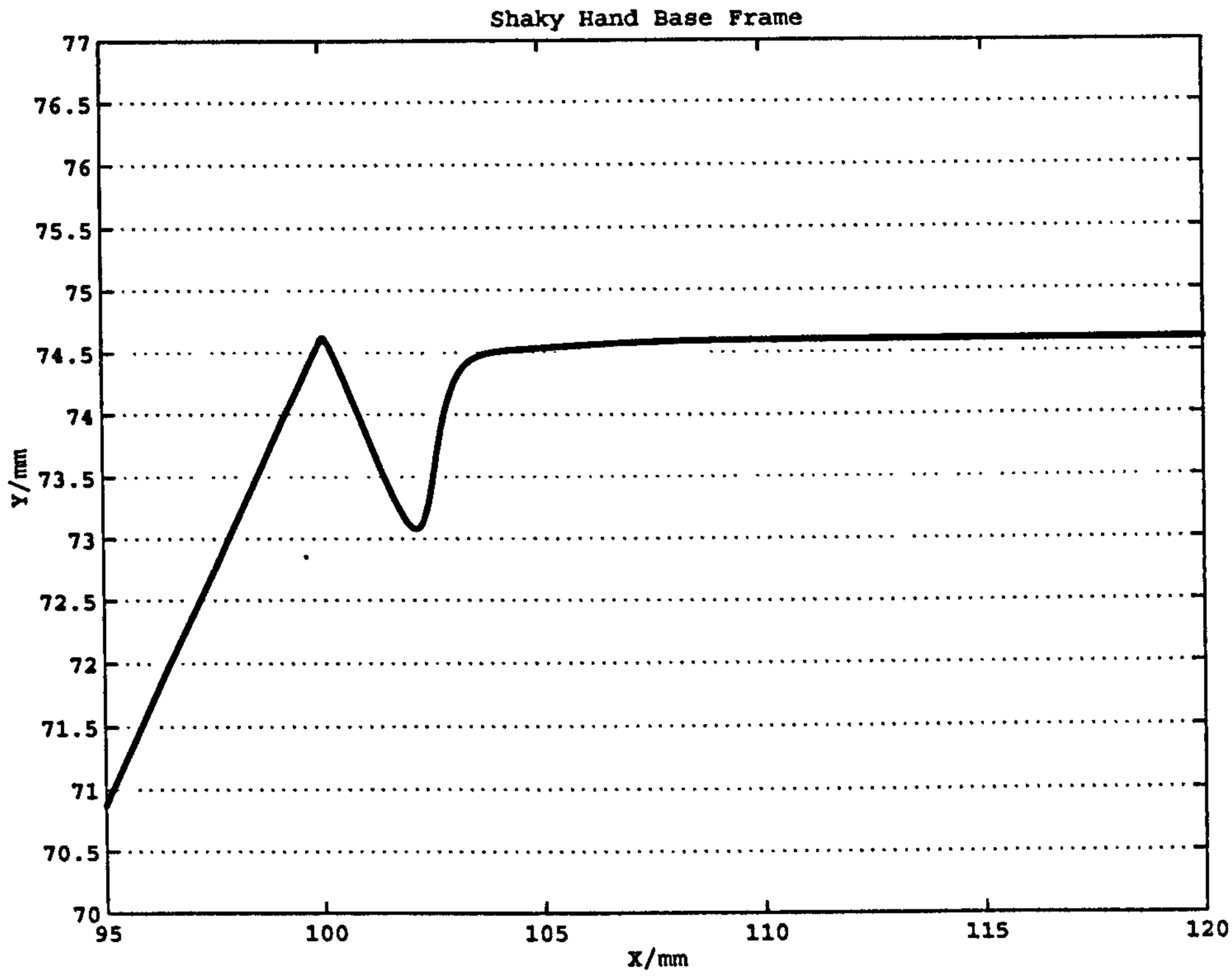


Figure 5.50: Simulink simulation: using the sensor A&B failure case solution

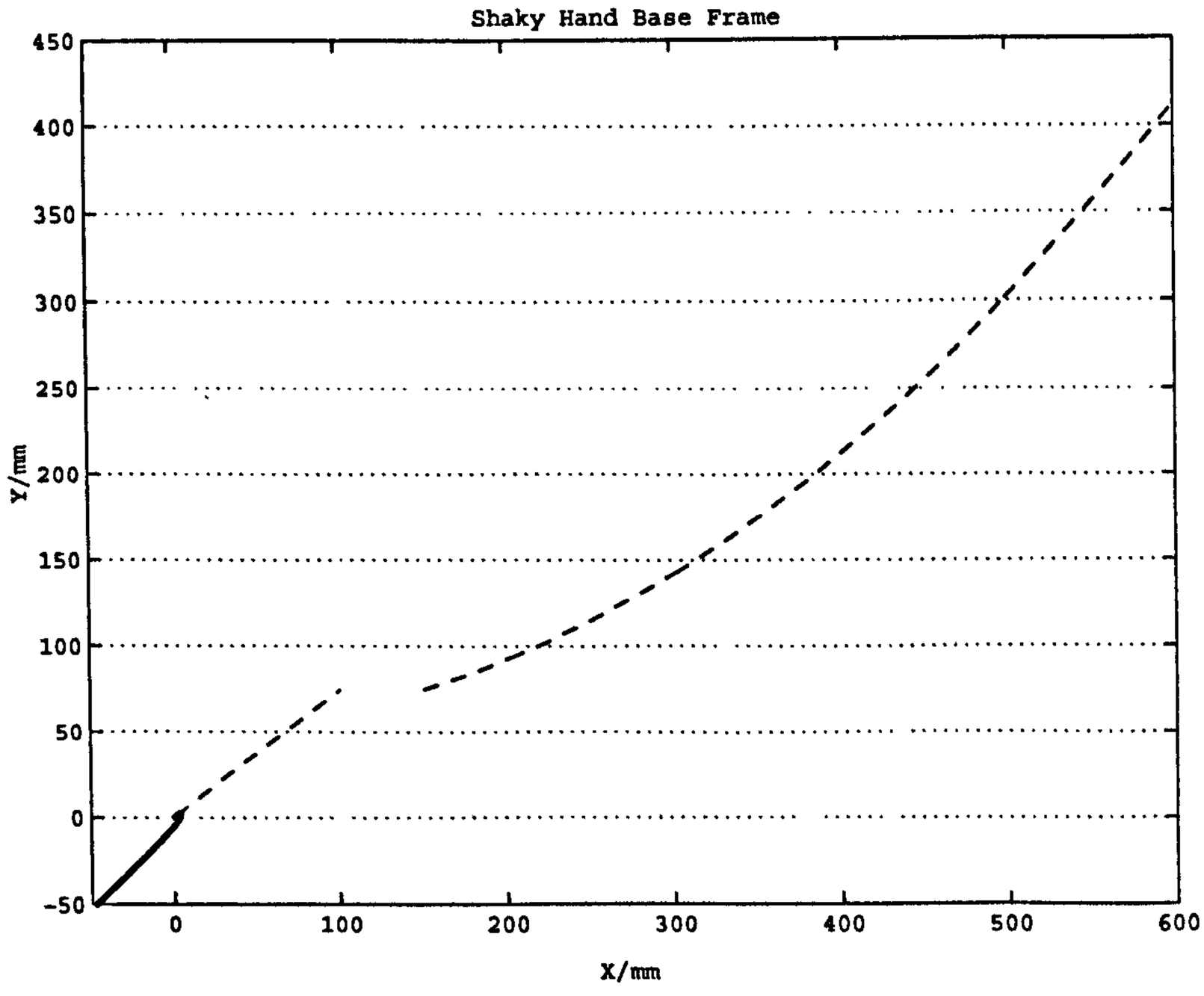


Figure 5.51: Simulink simulation: using the sensor A&C failure case solution

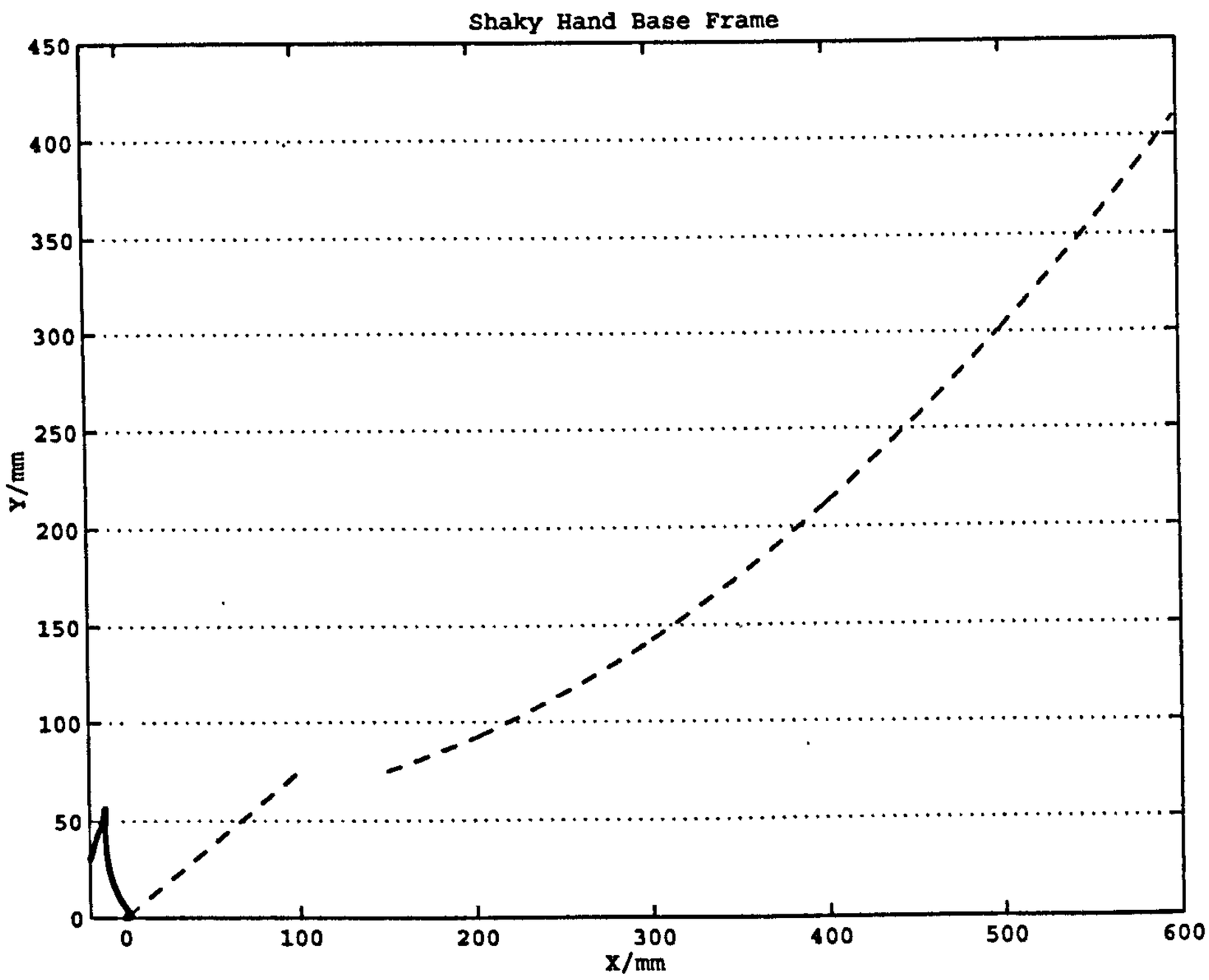


Figure 5.52: Simulink simulation: using the sensor A&B&D failure case solution

5.7.2 Results Using Shaky Hand Hardware

Following on from the Simulink simulation, these solutions were implemented using the Shaky Hand hardware. All the solutions were tested and videos were taken. The results of the practical experiments are shown in Table 5.11. A tick, ✓, indicates that the plate followed the track from the starting point to the end point of the wire track without drifting off. A cross, ×, indicates that the plate drifted off, ignoring the presence of the wire track, and the run had to be aborted. The original algorithms were first used in practice so that coil sensors on the plate could be approximately calibrated.

Number of sensor failures	Failure Case	Practical Result
1	Sensor A failure	✓
	Sensor B failure	✓
	Sensor C failure	✓
	Sensor D failure	✓
2	Adjacent pair AB failure	✓
	Adjacent pair CD failure	✓
	Diagonal pair AD failure	✓
	Diagonal pair BC failure	✓
	Opposite pair AC failure	×
	Opposite pair BD failure	×
3	Sensors BCD failure	×
	Sensors ACD failure	×
	Sensors ABD failure	×
	Sensors ABC failure	×

Table 5.11: Results of the evolved solutions in practice

The practical results reflect the Simulink simulation results and follows the pattern analysed in Section 5.6.4. The generic solutions for one sensor, diagonal sensor pair and adjacent sensor pair failure cases were able to compensate for the sensor failures. For the one sensor failure case, the operation could not be distinguished from the ideal case. For the diagonal sensor pair failure case, the plate movement is close to the ideal case, but the plate becomes less sensitive to the α and ϕ offsets when the wire track is away from the working sensors. Further degradation could be observed for the adjacent sensor pair failure case. The wire track settles very close to the non-working sensor for most of the time. The solution works for the smaller value of the working sensors as predicted in Section 5.6.4. The operation fails for the other failure case solutions.

Evolution of the solutions by CGP proved to be effective. The behaviour of the solutions could be predicted by evaluating their costs and analysing the solutions. The evolved solutions are

generic and can be applied successfully not only to the Simulink models but also in practice. The Shaky Hand is able to cope with some sensor failures as shown in Table 5.11 and complete the operation.

5.8 Why Do the Evolved Solutions for the One Sensor Failure Cases Work So Well?

The solutions for the one sensor failure cases, as recalled by Equations 5.73 to 5.76, are elegant. In this section, analytical reasoning behind these solutions are discussed.

1. V_α Solution for One Sensor Failure

$$V_\alpha = \ln((\frac{V_B}{V_C})^2) \quad and \tag{5.73}$$

$$V_\alpha = \ln((\frac{V_A}{V_D})^2) \tag{5.74}$$

2. V_ϕ Solution for One Sensor Failure

$$V_\phi = \frac{V_D}{V_C} - \frac{V_C}{V_D} \quad and \tag{5.75}$$

$$V_\phi = \frac{V_A}{V_B} - \frac{V_B}{V_A} \tag{5.76}$$

5.8.1 Analysis of V_α Case

The V_α solutions utilise diagonal sensor pair outputs. Let us look into Equation 5.73, where sensor A (or D) has failed. If the wire track is situated in the centre of the plate then V_B and V_C are equal, giving $V_\alpha = 0$. If the wire is closer to sensor B than to sensor C, then V_B is larger than V_C. So, V_B/V_C is greater than 1. Taking natural logarithmic of the value provides positive value. If V_C is larger than V_B, then, the solution provides a negative value as V_B/V_C is a fraction. So, the natural logarithmic term gives the correct polarity of α. A square term in the equation gives an amplification effect, providing greater penalty in the presence of α, which drives the plate back into the correct position quickly through control action.

During the evolution of the solution, the program identified the natural logarithmic function rather than the log_{10} function which could also provide the correct polarity. However, the natural logarithmic function generates a stronger penalty effect in the presence of α.

5.8.2 Analysis of V_ϕ Case

Let us look into Equation 5.75 (Sensor A or B failure case) for the V_ϕ solution. It uses two adjacent sensors to evaluate the output. V_D is normalised to V_C by the term V_D/V_C and V_C is normalised to V_D by the term V_C/V_D. The difference is taken as V_ϕ. It gives correct polarity at all times. So, how does the evolved solution differ from the original solution? Figure 5.53 illustrates the plate configuration.

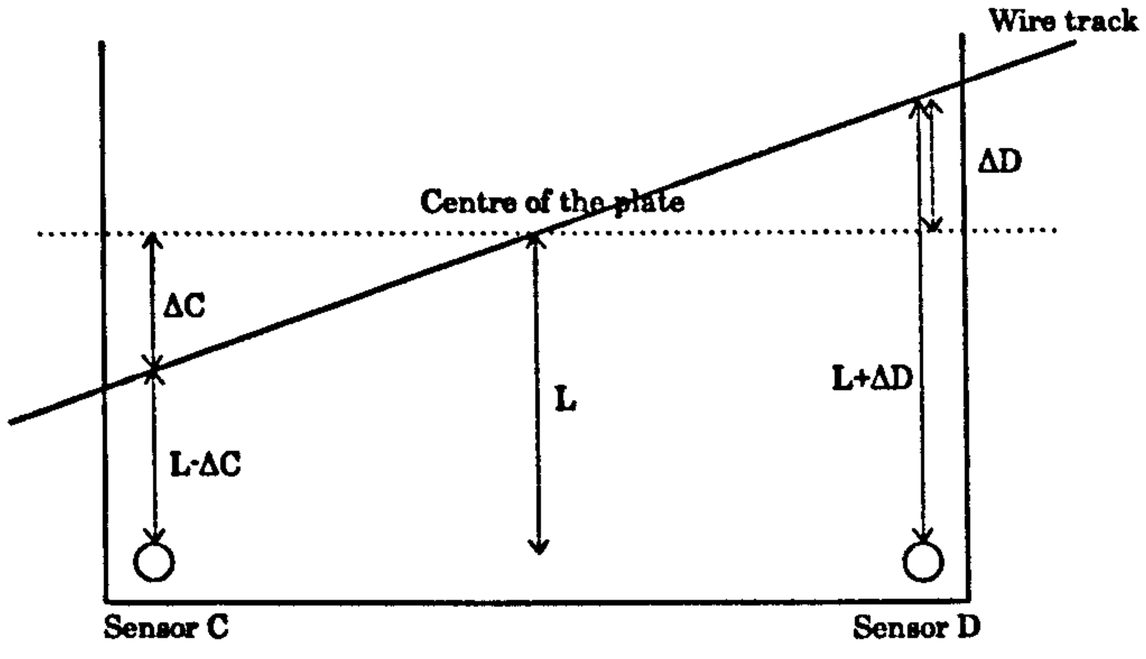


Figure 5.53: Plate configuration for the V_ϕ solution analysis

Initially, we assume the presence of ϕ, with α=0. So,

$$\Delta C = \Delta D \tag{5.77}$$

The voltage induced in the coil is inversely proportional to its proximity to the wire track and has a constant proportionality which we denote K. So,

$$V_C = \frac{K}{L - \Delta C} = \frac{K}{L - \Delta D} \tag{5.78}$$

$$V_D = \frac{K}{L + \Delta D} \tag{5.79}$$

The evolved solution (Equation 5.75) can be represented as

$$V_\phi = \frac{K}{L + \Delta D}\frac{L - \Delta D}{K} - \frac{K}{L - \Delta D}\frac{L + \Delta D}{K} \tag{5.80}$$

$$= \frac{L - \Delta D}{L + \Delta D} - \frac{L + \Delta D}{L - \Delta D} \tag{5.81}$$

$$= -\frac{4L\Delta D}{L^2 - \Delta D^2} \tag{5.82}$$

Let us now add an offset α, so, L→L+α,

$$V_\phi = -\frac{4(L+\alpha)\triangle D}{(L+\alpha)^2 - \triangle D^2} \tag{5.83}$$

The evolved solution (Equation 5.83) is now compared with the original solution:

$$V_\phi = (V_A + V_D) - (V_B + V_C) \tag{5.84}$$

In Figure 5.53, V_A and V_B are the same as V_D and V_C respectively. Therefore, using substitution and simplification, the original solution (Equation 5.84) becomes

$$V_\phi = -\frac{4K\triangle D}{L^2 - \triangle D^2} \tag{5.85}$$

Equation 5.85 is very similar to Equation 5.82 which is the evolved solution. In fact, fortuitously, K and L do have the same value. Therefore these solutions are exactly the same when there is no lateral offset. When α is present, Equation 5.85 becomes

$$V_\phi = -\frac{4K\triangle D}{(L+\alpha)^2 - \triangle D^2} \tag{5.86}$$

So the term (L+α) in Equation 5.83 no longer matches K in Equation 5.86, which does not change in the presence of α.

The graphs plotted based on Equations 5.86 and 5.83 are shown in Figures 5.54 and 5.55 respectively. The graphs plot the curves of V_ϕ over a range of ϕ under the influence of α. The dashed line in each plot represent the V_ϕ values when α=0.

Graphs for the original and the evolved solutions show similar pattern. However, the magnitude of V_ϕ is different. The magnitude of V_ϕ from the evolved solution is smaller when α is negative, and is larger when α is positive. The fitness is higher (i.e. the difference between the original and the evolved solution becomes less) when α becomes smaller. The evolved solution can compute exactly the same V_ϕ as the original solution when α=0. Therefore, as long as α is kept small, V_ϕ from the evolved solution is kept close to the ideal value. So, the evolved solution works best under tight tracking which is achieved through the control system which drives the offsets towards zero. This behaviour coincides with the intention of the use of the relative fitness evaluation method (Equations 5.21 to 5.24) during the CGP evolution. In conclusion, the evolved solution as shown in Equation 5.75 may possibly be found manually, but, the CGP evolution found this solution without any information about

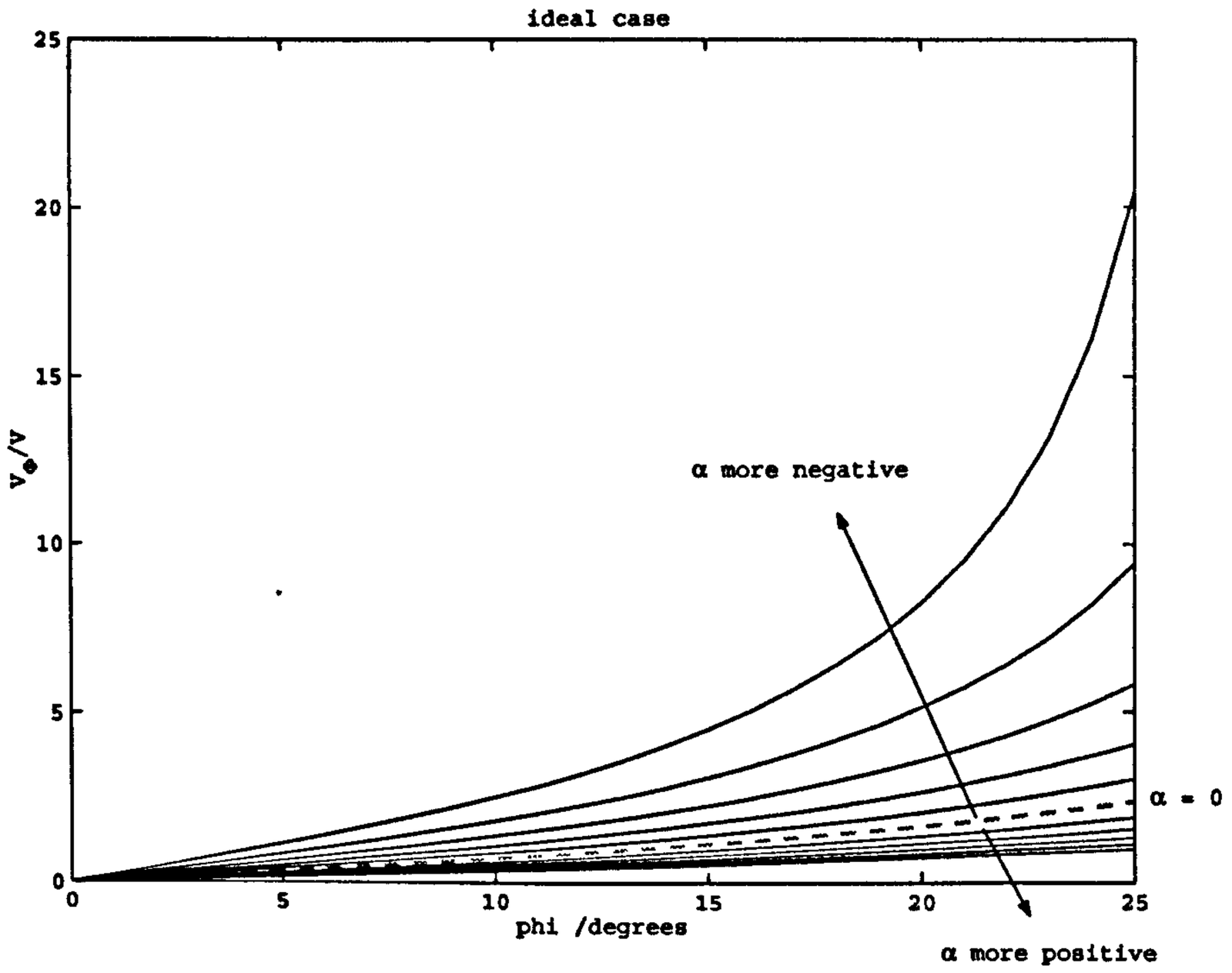


Figure 5.54: V_ϕ against ϕ with different α values using the original solution

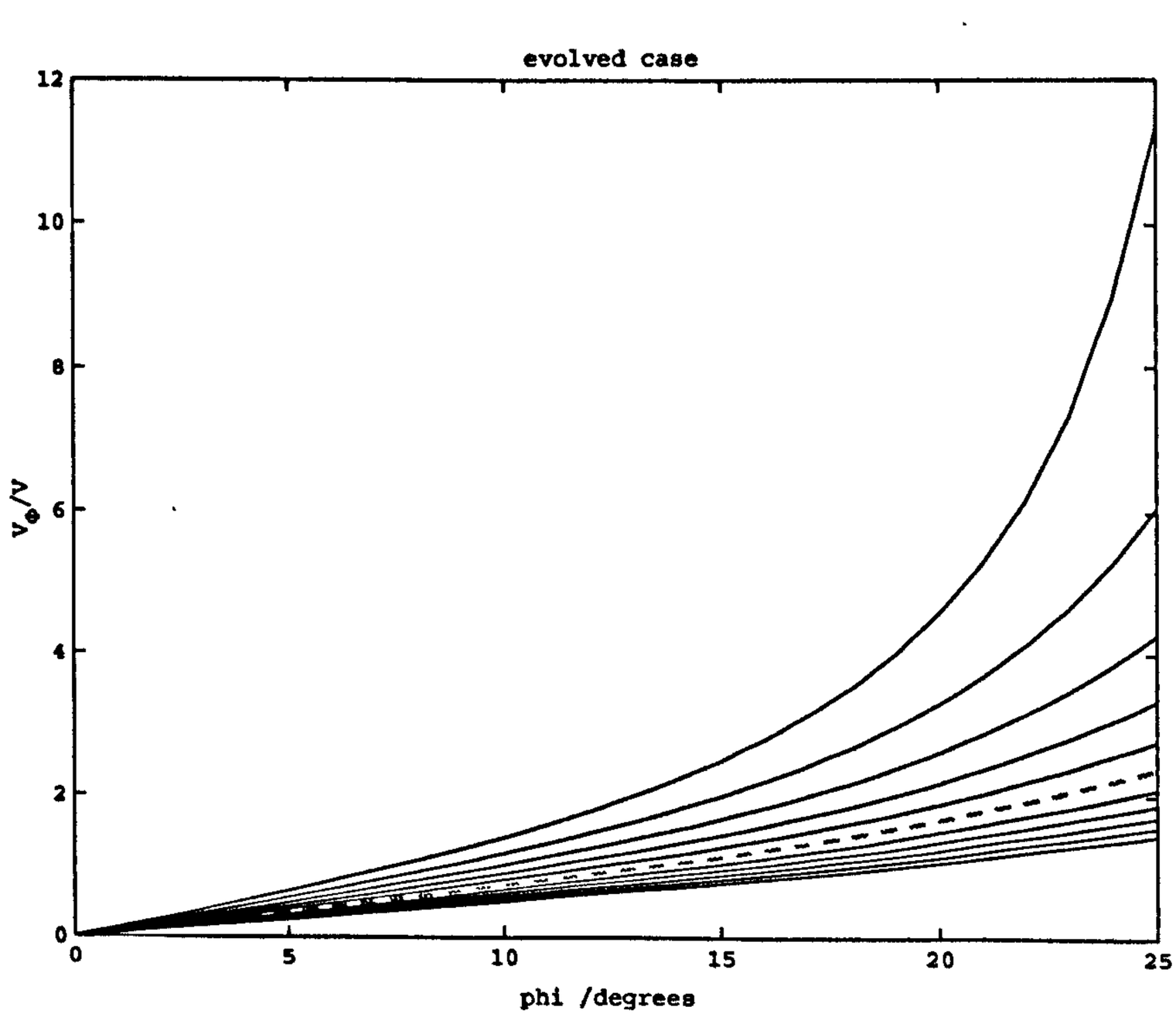


Figure 5.55: V_ϕ against ϕ with different α values using the evolved solution

the plate geometry, and only the four sensor voltage values.

5.9 Chapter Bibliography

[1] L Ashmore. An investigation into cartesian genetic programming within the field of art. Final year project 2000, Department of Computer Science, University of Birmingham, 2000.

[2] R.A Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2(1):14–23, 1986.

[3] J Clegg, J.A Walker, and J.F Miller. A new crossover technique for cartesian genetic programming. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1580–1587, London, England, 2007. ACM Press. GECCO 2007.

[4] S DiPaola. Evolving 'portrait painter programs' using genetic programming (Darwinian evolution) and a portrait of Darwin. website, School of Interactive Arts & Technology, Faculty of Applied Sciences, Simon Fraser University. http://www.dipaola.org/evolve/, August 2007.

[5] S Harding and J.F Miller. Evolution of robot controller using cartesian genetic programming. In *Proceedings of the 6th European Conference on Genetic Programming (EuroGP 2005)*, LNCS 3447, pages 62–72. Springer, 2005.

[6] THE MATHWORKS INC. *Simulink, Dynamic System Simulation for Matlab.* THE MATHWORKS INC., Natick, MA, 3 edition, 1990.

[7] J.R Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* The MIT Press, Cambridge, Massachusetts, 1992.

[8] J.R Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs.* The MIT Press, Cambridge, Massachusetts, 1994.

[9] T Martinek and L Sekanina. An evolvable image filter: experimental evaluation of a complete hardware implementation in fpga. In J.M Moreno, J Madrenas, and J Cosp, editors, *Evolvable Systems: From Biology to Hardware. Proceedings Lecture Notes in Computer Science*, volume 3637 of *6th International Conference, ICES 2005*, pages 76–85, Sitges, Spain, September 2005. Springer-Verlag, Berlin, Germany.

[10] J.F Miller. An empirical study of the efficiency of learning boolean functions using cartesian genetic programming approach. volume 2 of *Proc. of GECCO, page 1135-1142.* Morgan Kaufmann, 1999.

[11] J.F Miller. Evolving developmental programs for adaptation, morphogenesis, and self-repair. *Seventh European Conference on Artificial Life*, 2801:256–265, September 2003.

[12] J.F. Miller, D Job, and V.K. Vassilev. Principles in the evolutionary design of digital circuits – part i. *Journal of Genetic Programming and Evolvable Machines*, 1(2):259–288, 2000.

[13] J.F Miller and P. Thompson. Cartesian genetic programming. volume 1802 of *Proceedings of the 3rd European Conference on Genetic Programming*, pages 121–132. Springer-Verlag, 2000.

[14] J.F Miller and T Yu. Neutrality and evolvability of a boolean function landscape. volume Vol. 2038, pp. 204-217 of *EuroGP2001*. Springer-Verlag, 2001.

[15] N.L.Cramer. A representation for the adaptive generation of simple sequential programs. In J.J.Grefenstette, editor, *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, number 183-187, Carnegie-Mellon University, Pittsburgh, PA, USA, 24-26 Jul 1985.

[16] K Slany and L Sekanina. Fitness landscape analysis and image filter evolution using functional-level cgp. In M Ebner, M O'Neill, A Ekart, L Vanneschi, and A.A Esparcia, editors, *Proceedings Lecture Notes in Computer Science*, volume 4445 of *Genetic Programming. 10th European Conference, EuroGP 2007.*, pages 311–320, Valencia, Spain, April 2007. Springer-Verlag.

[17] S Stepney. Embodiment. In J Timmis, editor, *In Silico Immunology*. Springer, 2007.

[18] V.K. Vassilev and J.F. Miller. The advantages of landscape neutrality in digital circuit evolution. *3rd International Conference on Evolvable Systems*, 1801:252–263, 2000.

Chapter 6

Conclusion and Recommendations for Further Work

Contents

6.1 Introduction 193
6.2 Recommendation for Further Work 193
6.2.1 Monitoring System 194
Fault Detection and Identification Using the Evolved Solutions . . . 194
Monitoring System Using Agent 196
6.2.2 Multi-Agent Control for the Shaky Hand 197
Sensor and Data Fusion 198
6.3 Conclusion 198
6.4 Chapter Bibliography 200

6.1 Introduction

This chapter gives recommendations for further work, followed by a summary of the work presented in this thesis.

6.2 Recommendation for Further Work

The author recommends the following further work, applicable to the Shaky Hand.

- Development of a monitoring system.
- Multi-agent control of the Shaky Hand.

6.2.1 Monitoring System

Throughout, this work has assumed that the Shaky Hand is able to detect and identify failed sensor(s). The Shaky Hand would then select appropriate algorithms to enable continued operation or it would abandon further operation.

This section proposes an implementation for a monitoring system for the Shaky Hand. Firstly, we describe how the failure detection and identification for the one sensor failure case can be achieved using the evolved solutions obtained in Chapter 5. The idea of constructing a monitoring system using intelligent agents in conjunction with social attentive monitoring (SAM) [5] is then introduced.

Fault Detection and Identification Using the Evolved Solutions

A method for detecting a failure would be to employ a diversity system, where multiple different implementations of the same specification are provided and the system chooses the correct output on the basis of a quorum to cope with errors. For the Shaky Hand, the method would be to compare the original algorithm outputs (Equation 6.1, 6.2) with the corresponding evolved solutions for the one sensor failure cases (Equation 6.3, 6.4, 6.5 and 6.6).

$$V_{\alpha\, ideal} = (V_A + V_B) - (V_C + V_D) \tag{6.1}$$

$$V_{\phi\, ideal} = (V_A + V_D) - (V_B + V_C) \tag{6.2}$$

$$V_{\alpha\, A\, or/and\, D\, failure} = \ln((\frac{V_B}{V_C})^2) \tag{6.3}$$

$$V_{\alpha\, B\, or/and\, C\, failure} = \ln((\frac{V_A}{V_D})^2) \tag{6.4}$$

$$V_{\phi\, A\, or/and\, B\, failure} = \frac{V_D}{V_C} - \frac{V_C}{V_D} \tag{6.5}$$

$$V_{\phi\, C\, or/and\, D\, failure} = \frac{V_A}{V_B} - \frac{V_B}{V_A} \tag{6.6}$$

Their solution fitnesses are quite high and are quite similar to each other (Table 5.6), so, we assume that the outputs are *approximately* the same at all times. However, if sensor B fails,

then the values from all the solutions would no longer be (approximately) equal. i.e.

$$V_{\phi\, ideal} \neq V_{\phi\, A\, or/and\, B\, failure} \neq V_{\phi\, C\, or/and\, D\, failure} \tag{6.7}$$

$$V_{\alpha\, ideal} \neq V_{\alpha\, A\, or/and\, D\, failure} \neq V_{\alpha\, B\, or/and\, C\, failure} \tag{6.8}$$

Since the values are likely to be different, the failure can be detected. The failed sensor B may produce following outputs.

1. Invalid values: In this case, any solutions utilising a value from the sensor B become invalid. Therefore, the V_ϕ solution for the sensor pair AB failure case (Equation 6.5) and the V_α solution for the sensor pair BC failure case (Equation 6.4) are the only ones which produce valid outputs. So, in the event of a failure, the appropriate three-sensor algorithm is automatically selected.

2. Extreme values: Similar to the above, only Equation 6.5 and 6.4 produce valid outputs.

3. Constant values which are valid: All the solutions may produce valid outputs.

For the first two cases, it is easy to infer that sensor B has failed because all solutions utilising a value from the sensor B produce an invalid output. However, for the third case, another comparison is required to identify the failed sensor. Since a failure is already detected, the ideal solutions can no longer be used. A second comparison set, as shown in Table 6.1, must be used.

$V_{\phi\, A\, or/and\, B\, failure}$	solution with	$V_{\phi\, A\, or/and\, D\, failure}$	solution
$V_{\phi\, A\, or/and\, B\, failure}$	solution with	$V_{\phi\, B\, or/and\, C\, failure}$	solution
$V_{\phi\, C\, or/and\, D\, failure}$	solution with	$V_{\phi\, A\, or/and\, D\, failure}$	solution
$V_{\phi\, C\, or/and\, D\, failure}$	solution with	$V_{\phi\, B\, or/and\, C\, failure}$	solution
$V_{\alpha\, A\, or/and\, D\, failure}$	solution with	$V_{\alpha\, A\, or/and\, B\, failure}$	solution
$V_{\alpha\, A\, or/and\, D\, failure}$	solution with	$V_{\alpha\, C\, or/and\, D\, failure}$	solution
$V_{\alpha\, B\, or/and\, C\, failure}$	solution with	$V_{\alpha\, A\, or/and\, B\, failure}$	solution
$V_{\alpha\, B\, or/and\, C\, failure}$	solution with	$V_{\alpha\, C\, or/and\, D\, failure}$	solution

Table 6.1: Second set of evolved solutions comparison

If sensor B has failed, then only one solution pair produces similar outputs for each V_ϕ and V_α. By using this second comparison, it is possible to infer which sensor has failed. This method relies on the actual solution outputs. This is a simple solution which would work with a reasonably high success probability.

Monitoring System Using Agent

A more reliable way of failure detection and fault identification, is by monitoring the consistency of signals in the control system. For example, if the top-motor rotates then we expect a change in the sensor signals. If the change is not seen or it is inconsistent with the direction of the rotation, a failure is detected. Also, if the signal on sensor A increases then the signal from sensor C, whose position is opposite of sensor A, is expected to decrease. By monitoring these signals, it is possible to infer the failure and identify the faulty sensors. The monitoring system may be made even more robust if it incorporates a learning feature. For instance, if the system can learn about the relationship between the motor signal strength and the rate of change of the sensor signals on the plate or the rate of change of velocities in X- and Y-directions then the monitoring system is effectively utilising more information.

To implement such a monitoring system, we suggest the use of agents. The field of multi-agent systems is well established and includes many diverse applications. For example, Deep Space One, a NASA spacecraft, utilised an autonomous software for its control as part of the remote agent experiment [1]. Agents can be incorporated in a network system for real time tracking of a vehicle [3]. Agents have also been used for automating supply chain negotiations [8].

To avoid confusion, the use of the word 'agent' shall be clarified. When referring to 'agent' it is referring to a software entity with one or more independent threads of execution which is entirely responsible for its own input to and output from the environment in which it is situated. It is assumed that the agents have well defined objectives or goals and exercise problem solving behaviour in pursuit of these goals, reacting in a timely fashion while interacting with each other; it is this social behaviour which we refer to as *intelligence* [11].

Kaminka and Tambe [5][4][6] have discussed an agent based monitoring system known as social attentive monitoring (SAM) which focuses on monitoring failures in the social relationships between agents. The idea of SAM was inspired by Social Comparison Theory [2]. The key idea is that agents compare their own behaviour, beliefs, goals, and plans to those of other agents, reason about the differences in belief and behaviour, and draw useful conclusions regarding the correctness of their own actions or that of their peers [5]. SAM has three stages, detection of failures, diagnosis of failures and failure recovery through social monitoring. The Shaky Hand only requires the first and possibly the second stages of this. The self-recovery is not required because the appropriate control solutions are already prepared. Removing the third stage may also reduce the cost in risk (of violating the game rule) as the additional computation cost which would come with the recovery system is reduced. A

monitoring system such as SAM can be used to monitor the consistency of the coil sensor signals through social interaction of agents associated with each sensor.

The monitoring system introduced here only focused on the coil sensors on the top plate to satisfy the assumption made for the offset error detection. The monitoring area can be expanded to include the other sensors, namely the two shaft-encoder signals and the potentiometer signal. Social monitoring is one of the possible technique to observe the behaviour of these sensors and perform the failure detection and diagnosis.

6.2.2 Multi-Agent Control for the Shaky Hand

In this section, a way of providing dependable control of the Shaky Hand using intelligence to react to the failure of subsystems is suggested. A multi-agent system is, again, proposed as a strategy for such a system. This is because the Shaky Hand plant is simple but has fairly large number of inputs and outputs and also has well established goals. These factors make the Shaky Hand an ideal sensor-agent platform. The monitoring system mentioned in the previous section provides reliability for the current control system via agents. The idea of multi-agent control here changes the current control system completely. Agents associated with sensors, controllers, and monitors interact and exchange information via communication of some sort to achieve their own goals. If multi-agents were to be incorporated successfully into the Shaky Hand, the demonstrator could be capable of the following:

- deciding on the most efficient way of following the track, e.g. low power consumption, minimum travelling distance.
- using all available resources, such as sensors, to *navigate* the plate along the track provided.
- adjusting necessary parameters in a timely fashion to cope with the changes in the plant.
- identifying failures and diagnosing faults, whose information affects the further navigation in real-time.
- presenting the decision-making process for human interpretation for maintenance purpose.
- coping with failure and degradation due to wear and tear.

Since the plant depends on the sensor inputs, a way to maximise the use of sensors is suggested, which is a sensor-fusion technique. This may be the initial step in structuring the multi-agent control.

Sensor and Data Fusion

Multi-sensor fusion is a method of integrating signals from multiple sources [9], which are mainly sensors or other devices that allow for measurement of changing environment. The idea behind this is that fusion of complementary information available from different sensors will yield more accurate results for information processing problems. Varshney [10] states some of the advantages on having multi-sensor fusion system:

- *Improved system reliability and robustness*: It has an inherent redundancy. When one or more sensors fail, the system can continue to operate at a reduced performance level. *graceful degradation*

- *Extended coverage*: Multiple sensors can observe a region larger than the one observed by a single sensor.

- *Increased confidence*: Sensors can confirm each other's inferences, thereby increasing confidence in the final system inference.

- *Shorter response time*: More data is collected by multiple sensors, therefore a goal is achieved in shorter time.

- *Improved resolution*:Use of sensors with different resolutions can result in an inference with better resolution than any single sensor.

These points are the driving force for suggesting this technique to apply to the Shaky Hand. Using the data fusion technique, more accurate and reliable performance may be obtained.

6.3 Conclusion

The basic physical system was designed and built at the beginning of the research. The Shaky Hand is a novel laboratory demonstrator and an ideal platform for applying intelligent systems for testing sensor fault tolerant, data fusion as well as multi-agent techniques because of the multiple inputs/outputs configuration and its simplicity.

The author developed the model of the Shaky Hand by studying the physics of the system. Models of the control system of the demonstrator were represented by I/O software written by the author in practice. The control system receives sensor signals from the hardware and feeds back the appropriate control signals via I/O cards installed in a conventional PC. After the initial testing of the whole system, the design of the electrical circuitries and the physical system were revised in order to overcome the practical implementation issues arising from the practical test of the demonstrator.

Once the demonstrator was completed, refined and tested, development of the sensor fault tolerant control solutions for the Shaky Hand using CGP proceeded. A novel way to evolve a fault tolerant *generic* solution was established using special training sets. Creation of the virtual environment which achieves suitably complex dynamics to allow rich interactions between the Shaky Hand and the environment was carefully constructed. A systematic study of the effects of CGP was carried out to establish the best method. The best solutions were selected using verification test sets. All the fitnesses and the standard deviations obtained from the evolved solutions were analysed to discover how the fitness changes throughout the evolution process, and to find any pattern in the successful solution. A good solution seems to gradually emerge as the evolution proceeds. The evolved solutions were also compared with the original algorithms to show their improvement. The tendency given by the evolved solutions suggests that the estimation of errors is generally better when the wire track is closer to the working sensors which vindicates the choice of normalised cost function for the CGP process. The novel evolved solutions were then implemented in Simulink simulations before practical application.

The practical results coincide with the simulation results. The development of the generic sensor fault tolerant algorithms were successful for all the one sensor failure cases, two diagonal sensor pair failure cases and two adjacent sensor pair failure cases. They are reliable and the Shaky Hand system is able to work with failed sensors and on any arbitrary wire track shape. Some of the successful solutions involve conditional operators which would be very difficult or impossible to develop manually, without the help of CGP-based evolution. It is clear that the established CGP method can be applied for sensor fault tolerant applications in practice. The work shows that the use of intelligent techniques can provide a reliable practical system and that CGP is a powerful tool to satisfy the control problems raised in this thesis. The objectives of the work were verified not only in simulation but also in practice.

In final conclusion, this work has produced a final fall-back system which will provide safe, if degraded, performance of a system when all other fault tolerance mechanisms, based upon

multiple redundancy of sensors, cease to be available. CGP has been used to explore the potential algorithmic solution space - a task which we consider to be difficult or impossible using a manual pen and paper approach. Both the method of generating the solutions and the solutions themselves are completely novel. This work opens the door to a different and complementary scheme to achieve sensor fault tolerance.

6.4 Chapter Bibliography

[1] D Bernard, G Dorais, E Gamble, B Kanefsky, J Kurien, G.K Man, W Millar, N Muscettola, P Nayak, K Rajan, N Rouquette, B Smith, W Taylor, and Y Tung. Spacecraft automy flight experience: The ds1 remote agent experiment. In *In Proceedings of the Space Technology Conference and Exposition*, AIAA-99, Albuquerque, NM, USA, 1999.

[2] L Festinger. A theory of social comparison processes. *Human Relations*, 7:117–140, 1954.

[3] B Horling, R Vinvent, R Mailler, J Shen, R Becker, K Rawlins, and V Lesser. Distributed sensor network for real time tracking. In *International Conference on Autonomous Agents, Proceeding of the fifth international conference on Autonomous agents*, pages 417–424, Montreal, Quebec, Canada, 2001. ACM Press. ISBN:1-58113-326-X.

[4] G.A Kaminka and M Tambe. Social comparison for failure detection and recovery. In Munindar P. Singh, Anand Rao, and Michael J. Wooldridge, editors, *Intelligent Agents IV: ATAL'97 Proceedings*, volume 1365 of *Lecture Notes in Artificial Intelligence*, pages 127–141. Springer-Verlag, 1998.

[5] G.A Kaminka and M Tambe. What is wrong with us? improving robustness through social diagnosis. In *AAAI'98 Proceedings, Madison, WI, USA*, 1998.

[6] G.A Kaminka and M Tambe. I'm OK, you're OK, we're OK: Experiments in distributed and centralized socially attentive monitoring. In *Proceedings of the Third International Conference on Autonomous Agents (Agents'99), Seattle, WA, USA*, 1999.

[7] M Liao. Shaky hand. Msc report, Control Systems Laboratory, Intelligent Systems Group, The University of York, 2004.

[8] S Putten, V Robu, H.L Poutre, A Jorritsma, and M Gal. Automating supply chain negotiations using autonomous agents: a case study in transportation logistics. In *International Conference on Autonomous Agents, Proceedings of the fifth international joint conference on autonomous agents and multiagent systems*, pages 1506 – 1513, Hakodate, Japan, 2006. ACM Press, New York, NY, USA.

[9] J.Z Sasiadek. Sensor fusion. *Annual Review in Control*, (26):203–228, 2002.

[10] P.K Varshney. Multisensor data fusion. *IEE Electronics & Communication Engineering Journal*, 9:245–253, 1997.

[11] M Wooldridge. *An Introduction to MultiAgent Systems*. Wiley, 2002.

Appendix A

Layouts for Physical Construction Works

Contents

A.1 Shaky Hand Metal Works 202
A.2 PCB Design Layouts 205
A.3 Appendix Bibliography 207

A.1 Shaky Hand Metal Works

Design layouts for the Shaky Hand metal work are shown. All the work has been done in the Student Workshop, Department of Electronics, the University of York. Figures from A.1 to A.6 have been pre-designed and constructed by Liao [1].

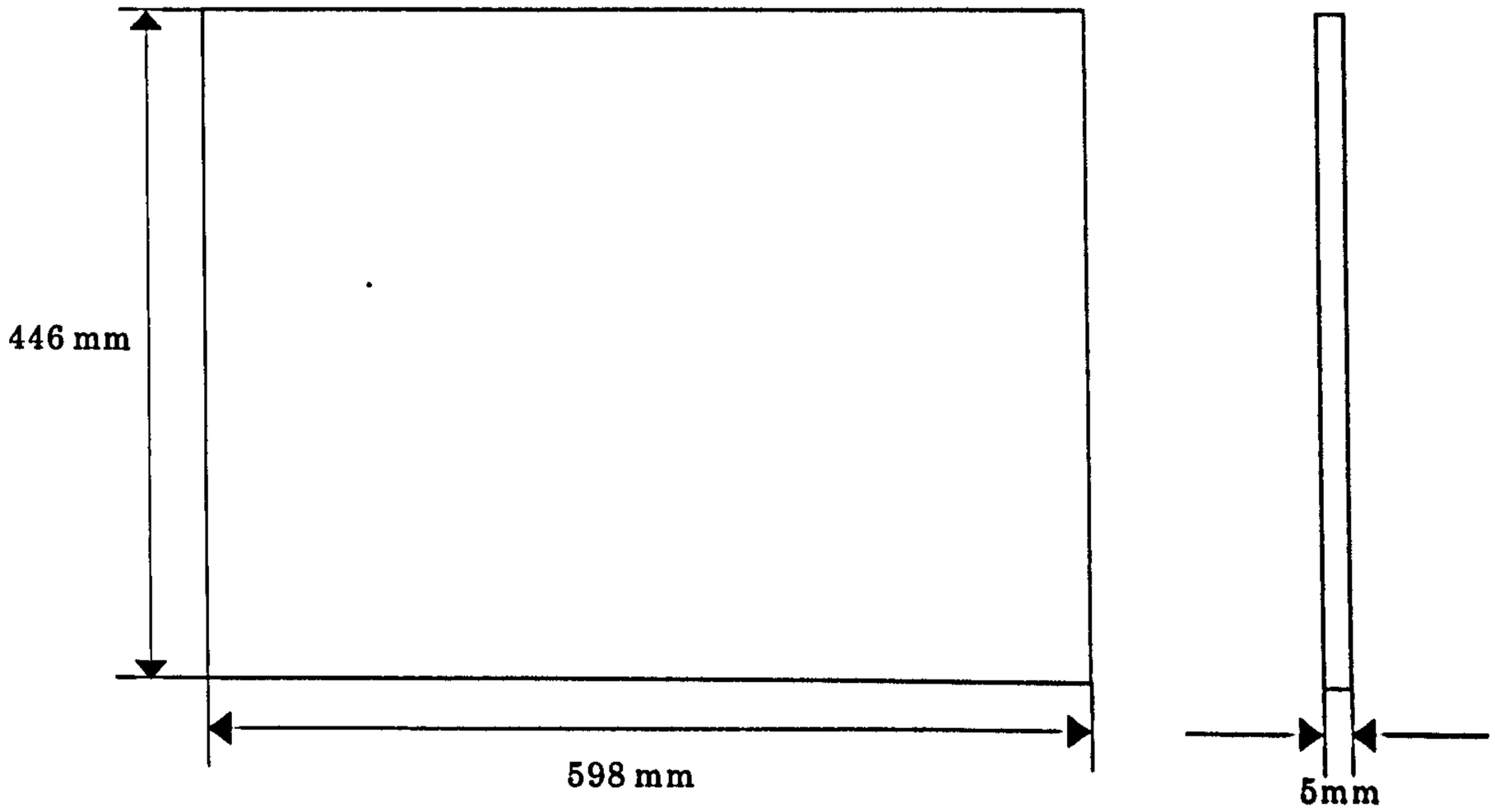


Figure A.1: The base frame

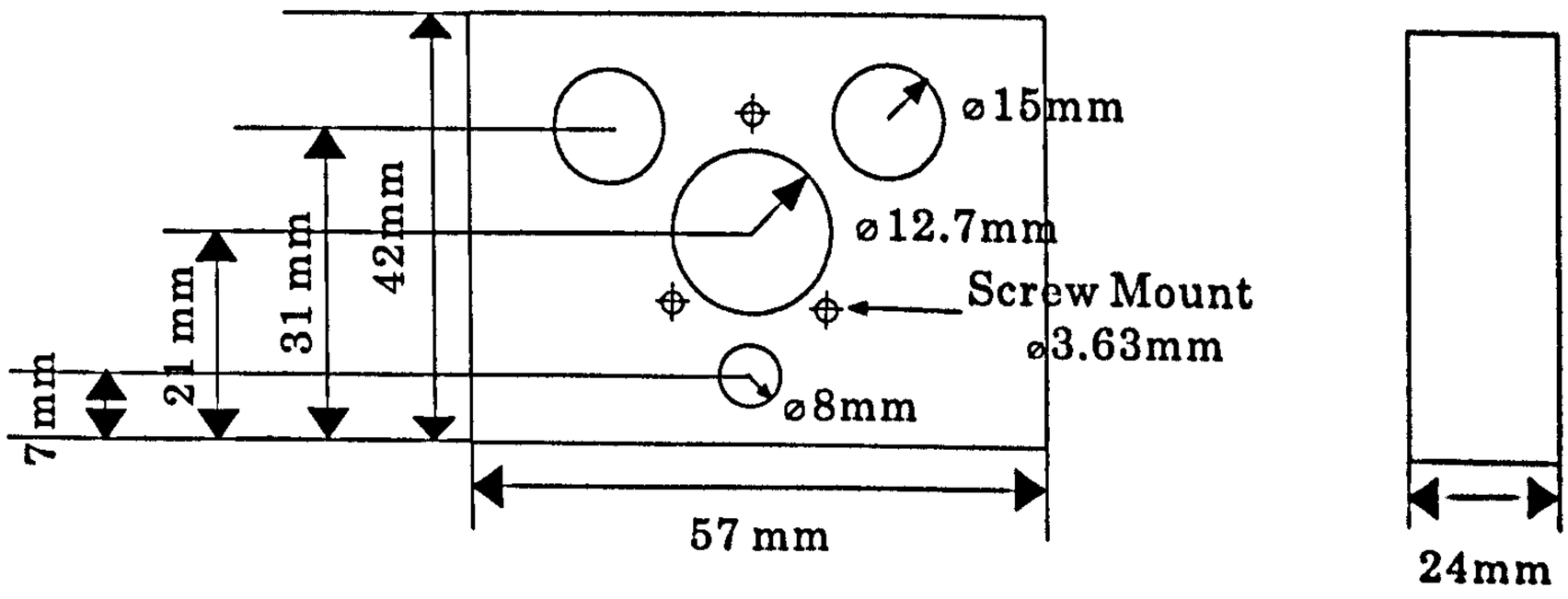


Figure A.2: Y-carriage

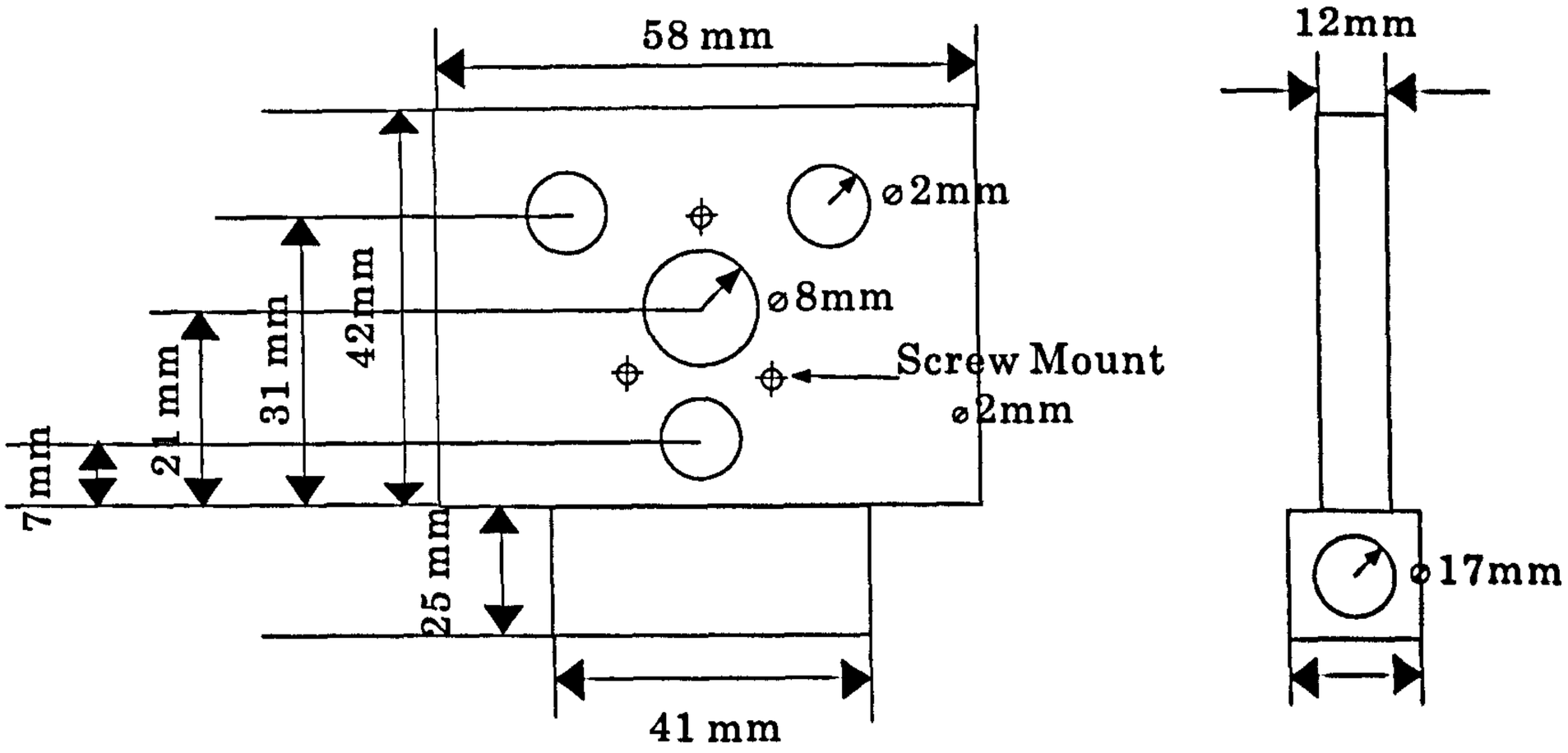


Figure A.3: X-carriage and bearing housing (Y-motor side)

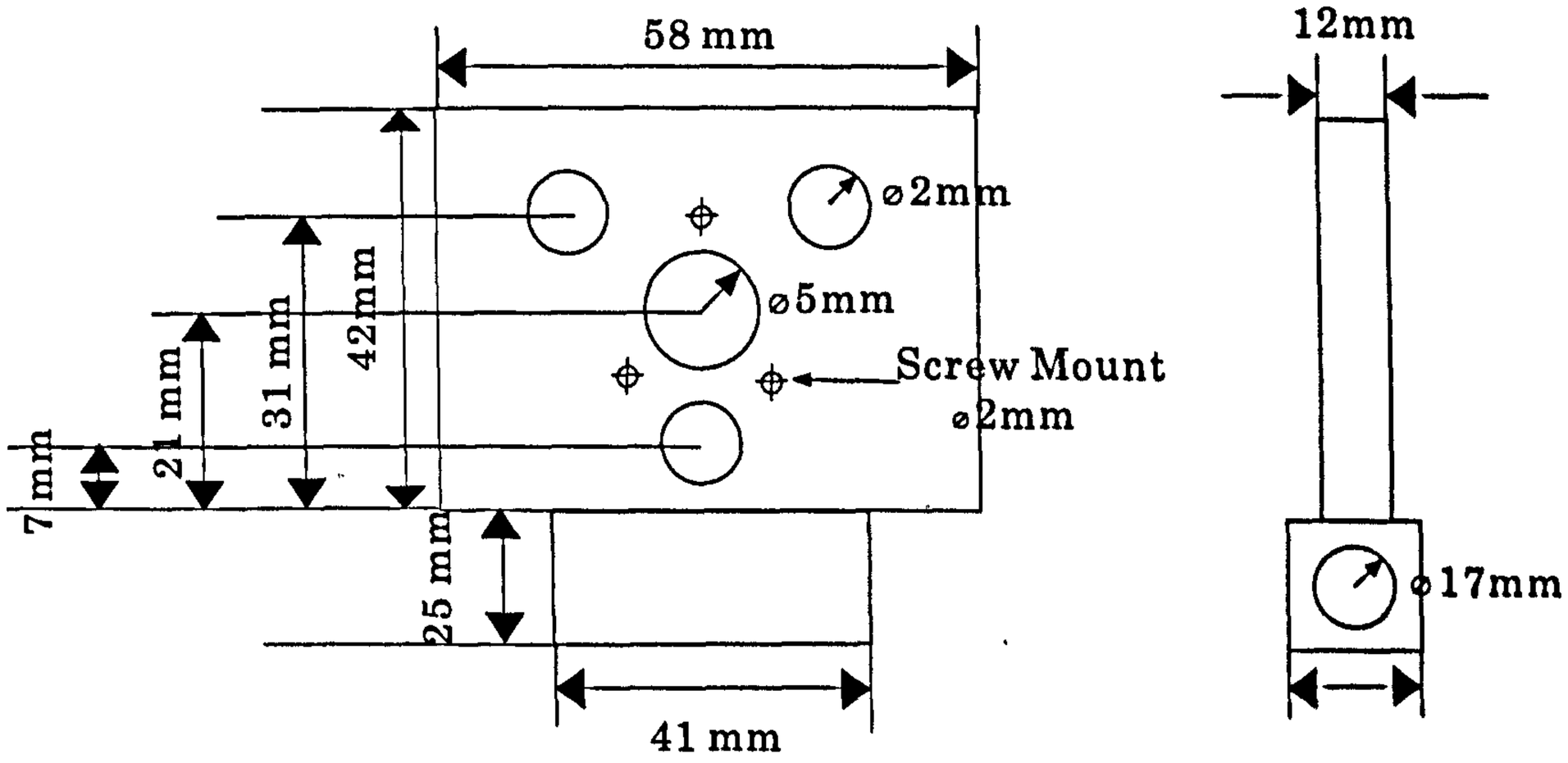


Figure A.4: X-carriage and bearing housing (Y-shaft-encoder side)

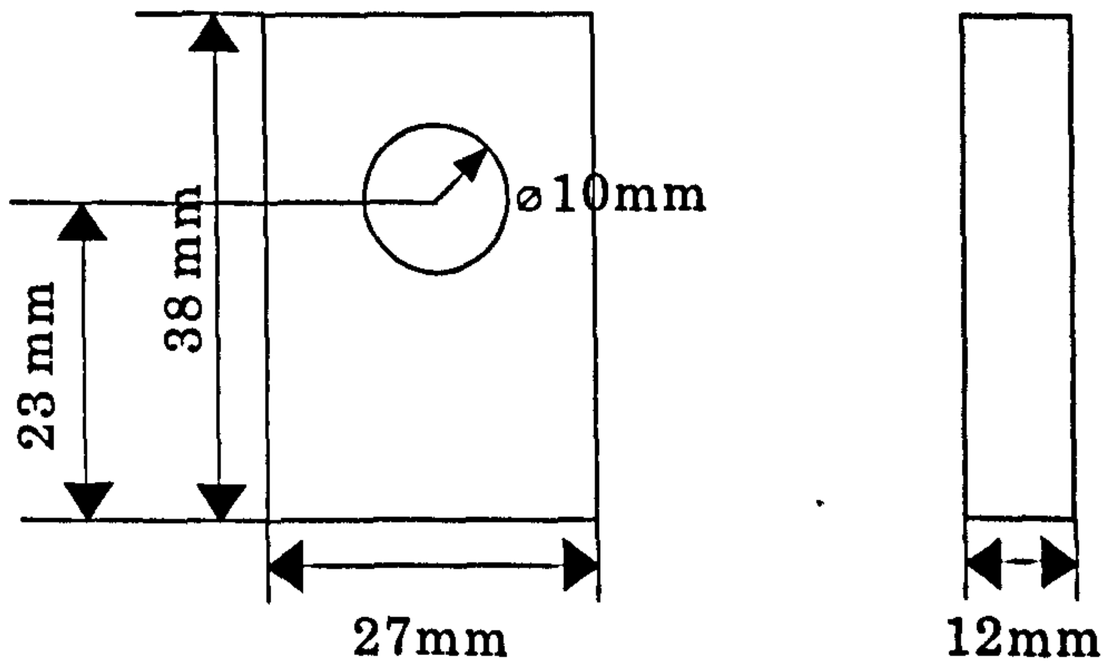


Figure A.5: Horizontal shaft support plate

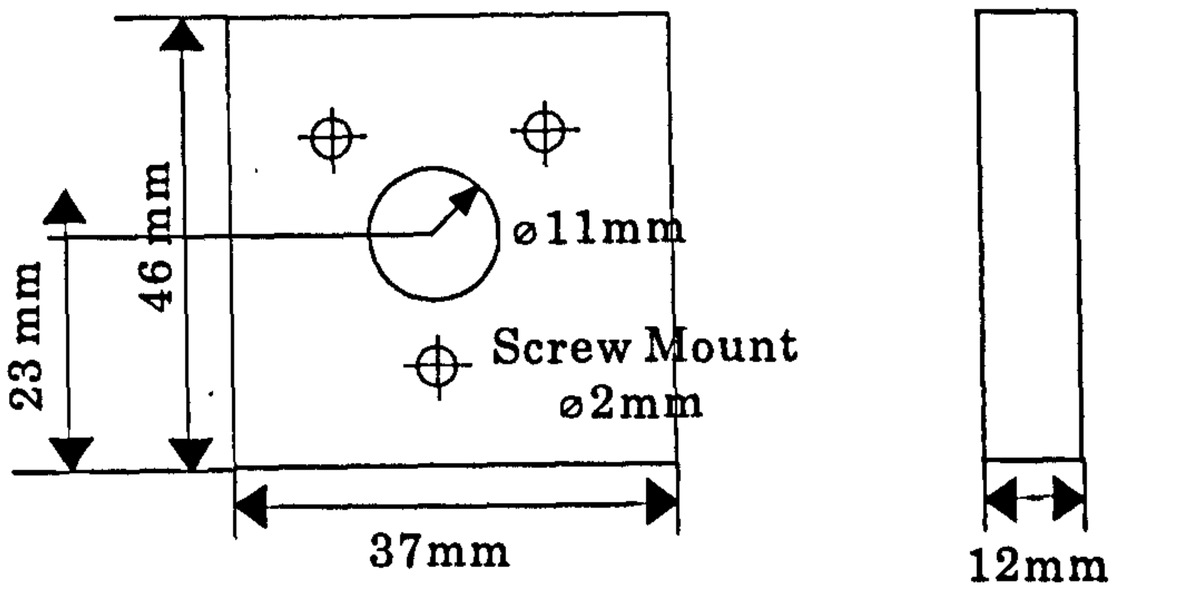


Figure A.6: Horizontal leadscrew support plate

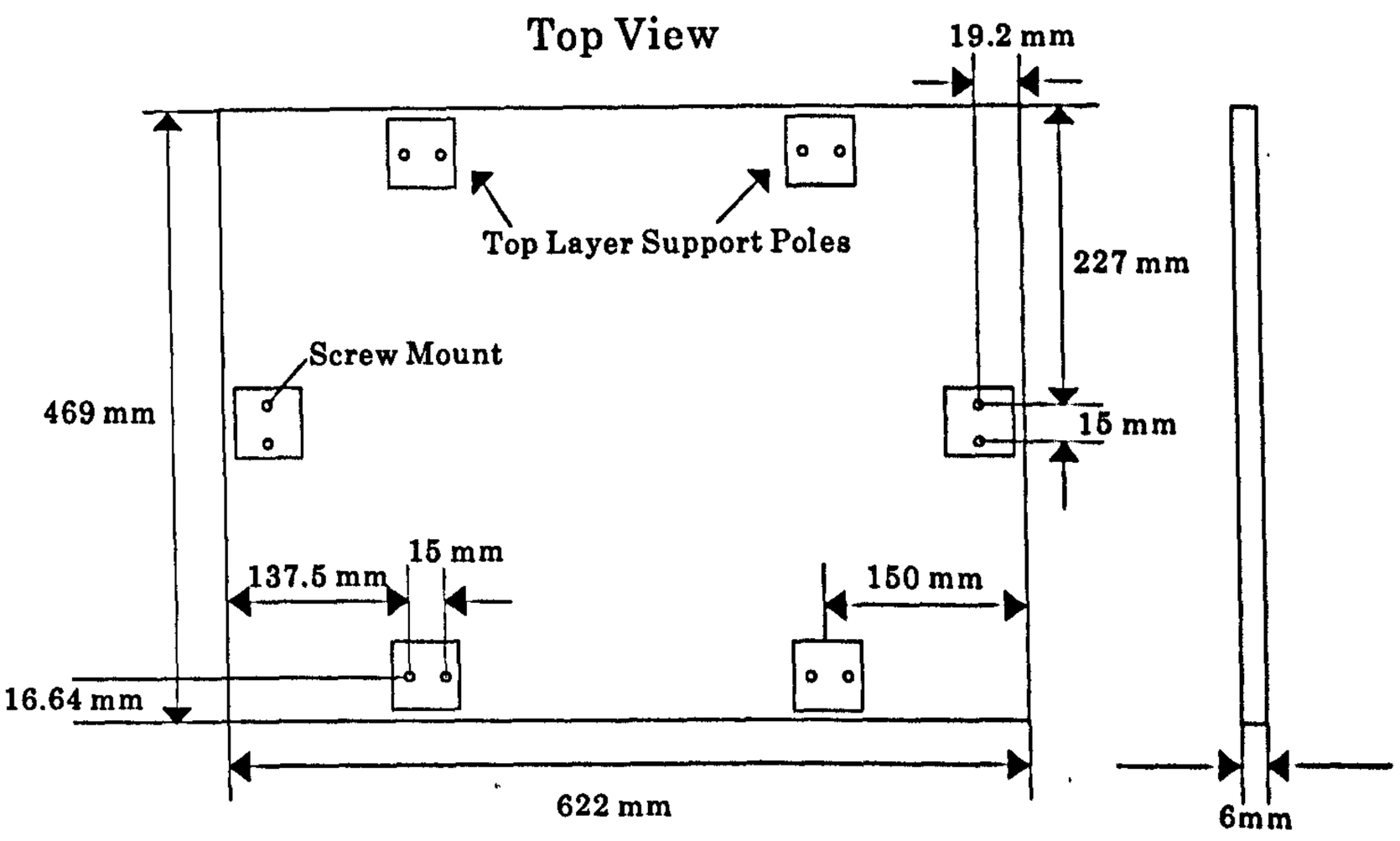


Figure A.7: Top view of the supporting layer

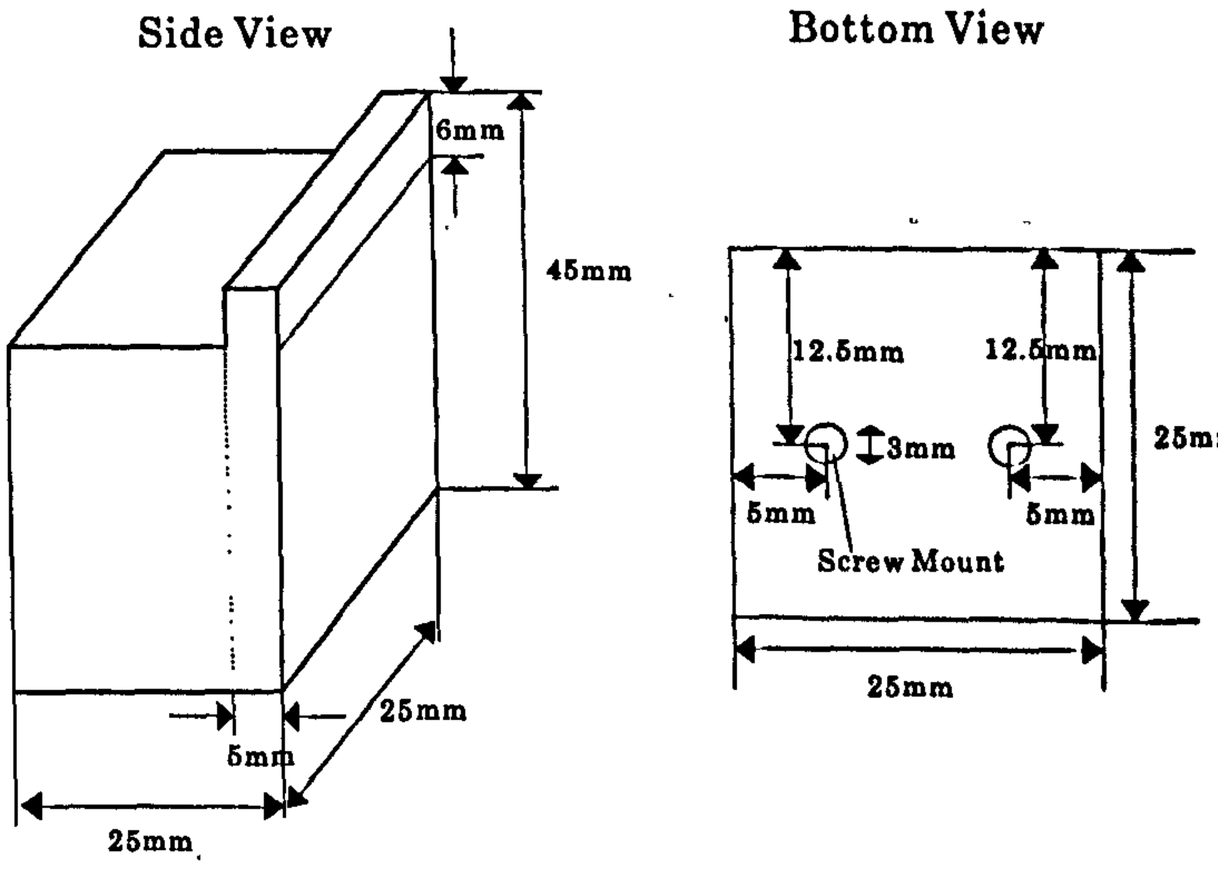


Figure A.8: Dimensions of the support poles

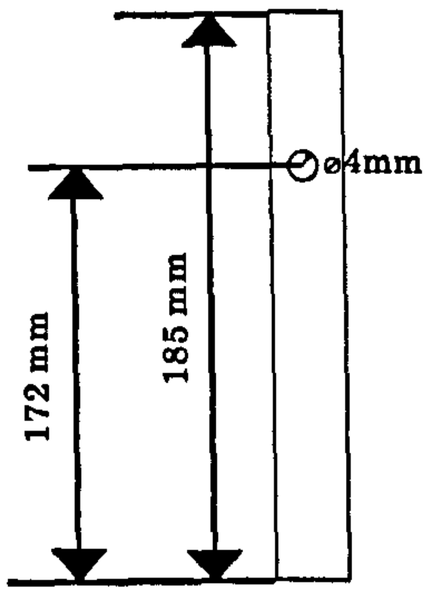


Figure A.9: Dimensions for wire track supporting poles

A.2 PCB Design Layouts

Many PCB boards are used to help the construction of the Shaky Hand.

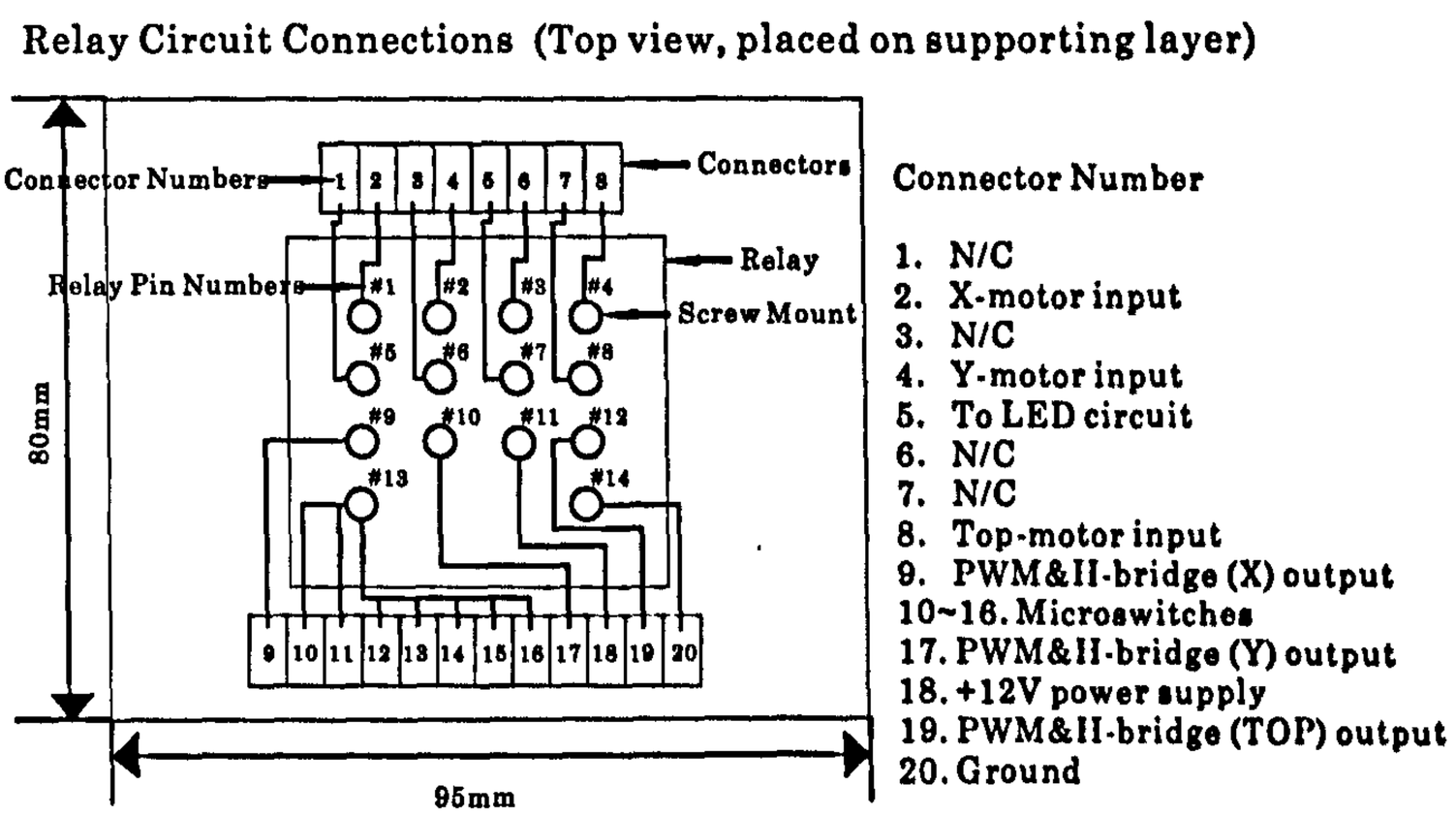
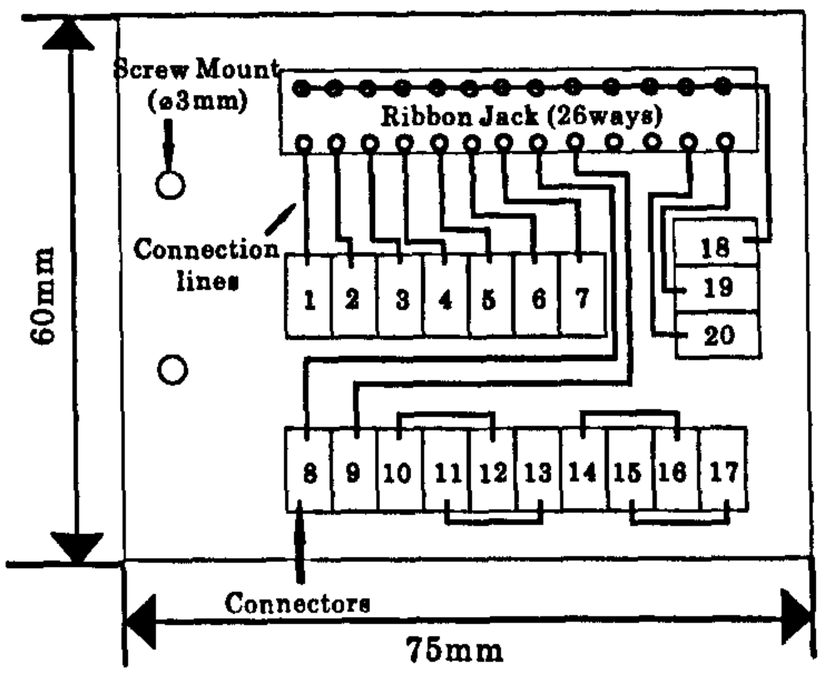


Figure A.10: Layout for PCB boards: relay circuit and power supply lines

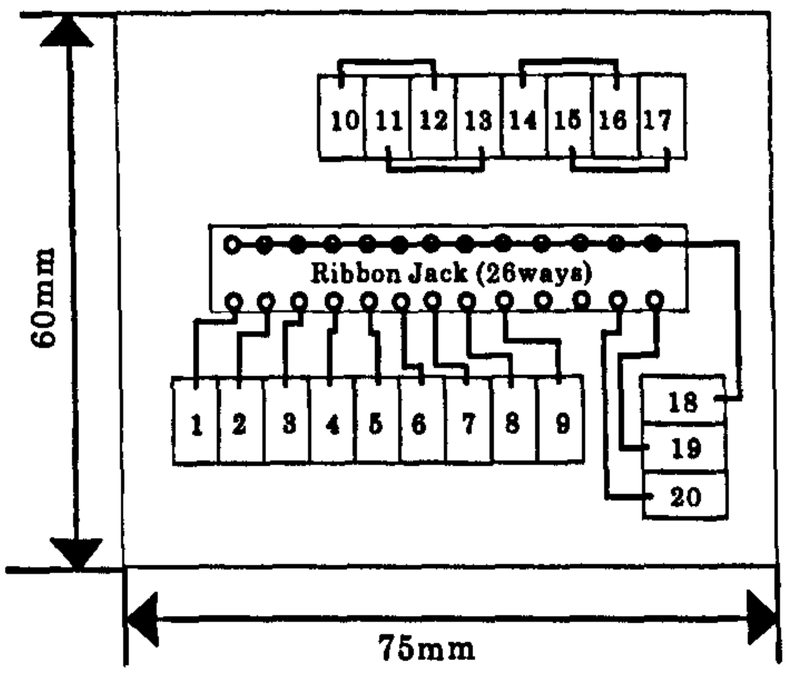
Ribbon Cable Connector (Top view, on X-carriage)



Connector Number

1. Signal from Sensor A
2. Signal from Sensor B
3. Signal from Sensor C
4. Signal from Sensor D
5. Signal from Potentiometer
6. Signal from Encoder(y)
7. Microswitch Relay
8. Microswitch Relay
9. Microswitch Relay
10&12. Input to Top Motor (from PWM(Top))
11&13. Input to Top Motor(from PWM(Top))
14&16. Input to Y-Motor (from PWM(Y))
15&17. Input to Y-Motor (from PWM(Y))
18. Ground
19. - 5V Power Supply
20. +5V Power Supply

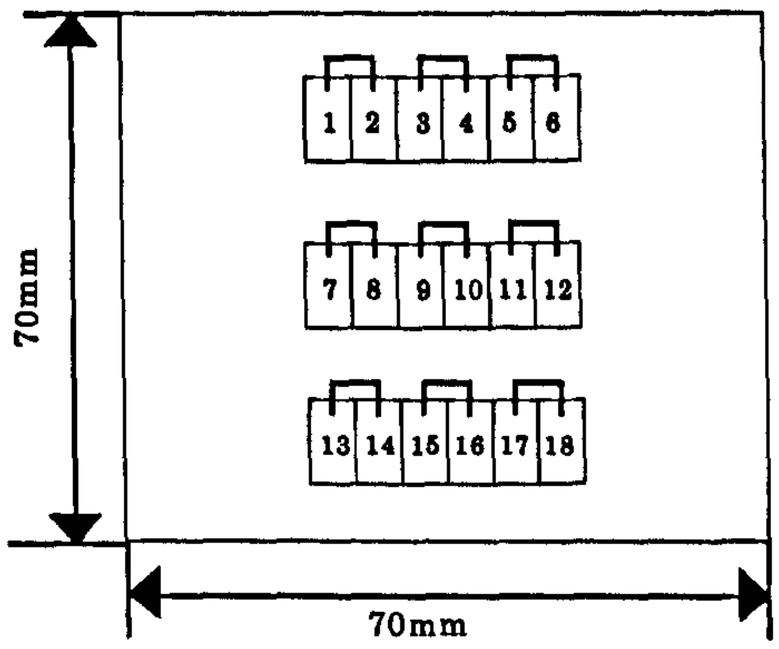
Ribbon Cable Connector (Top view, on supporting plate)



Connector Number

1. Signal from Sensor A
2. Signal from Sensor B
3. Signal from Sensor C
4. Signal from Sensor D
5. Signal from Potentiometer
6. Signal from Encoder(y)
7. Microswitch Relay Terminal
8. N/C
9. N/C
10&12. Input to Top Motor (from PWM(Top))
11&13. Input to Top Motor(from PWM(Top))
14&16. Input to Y-Motor (from PWM(Y))
15&17. Input to Y-Motor (from PWM(Y))
18. Ground
19. - 5V Power Supply
20. +5V Power Supply

Relay Connector Jacks (Top view, on supporting plate)



Connector Number

1. Ground
2. Microswitch Relay (to PCV1, Connector 8)
3. +12V Power Supply
4. Microswitch Relay (to Microswitch 1)
5. Microswitch Relay (from Microswitch 2)
6. Microswitch Relay (to Microswitch 3)
7. N/C
8. N/C
9. N/C
10. N/C
11. N/C
12. N/C
13. N/C
14. N/C
15. N/C
16. N/C
17. N/C

Figure A.11: Layout for PCB boards: ribbon cable jacks and other connectors

A.3 Appendix Bibliography

[1] M Liao. Shaky hand. Msc report, Control Systems Laboratory, Intelligent Systems Group, The University of York, 2004.

Appendix B

Circuit Diagrams

Contents

B.1 Electrical Circuits Diagrams 209
B.1.1 Original Sensor System 209
B.1.2 Revised Sensor System 211
B.2 Pictures of the PCB Designs 214
B.3 I/O Pin Assignment 216
B.4 Appendix Bibliography 216

B.1 Electrical Circuits Diagrams

Circuit diagrams used in the Shaky Hand construction stage are shown.

B.1.1 Original Sensor System

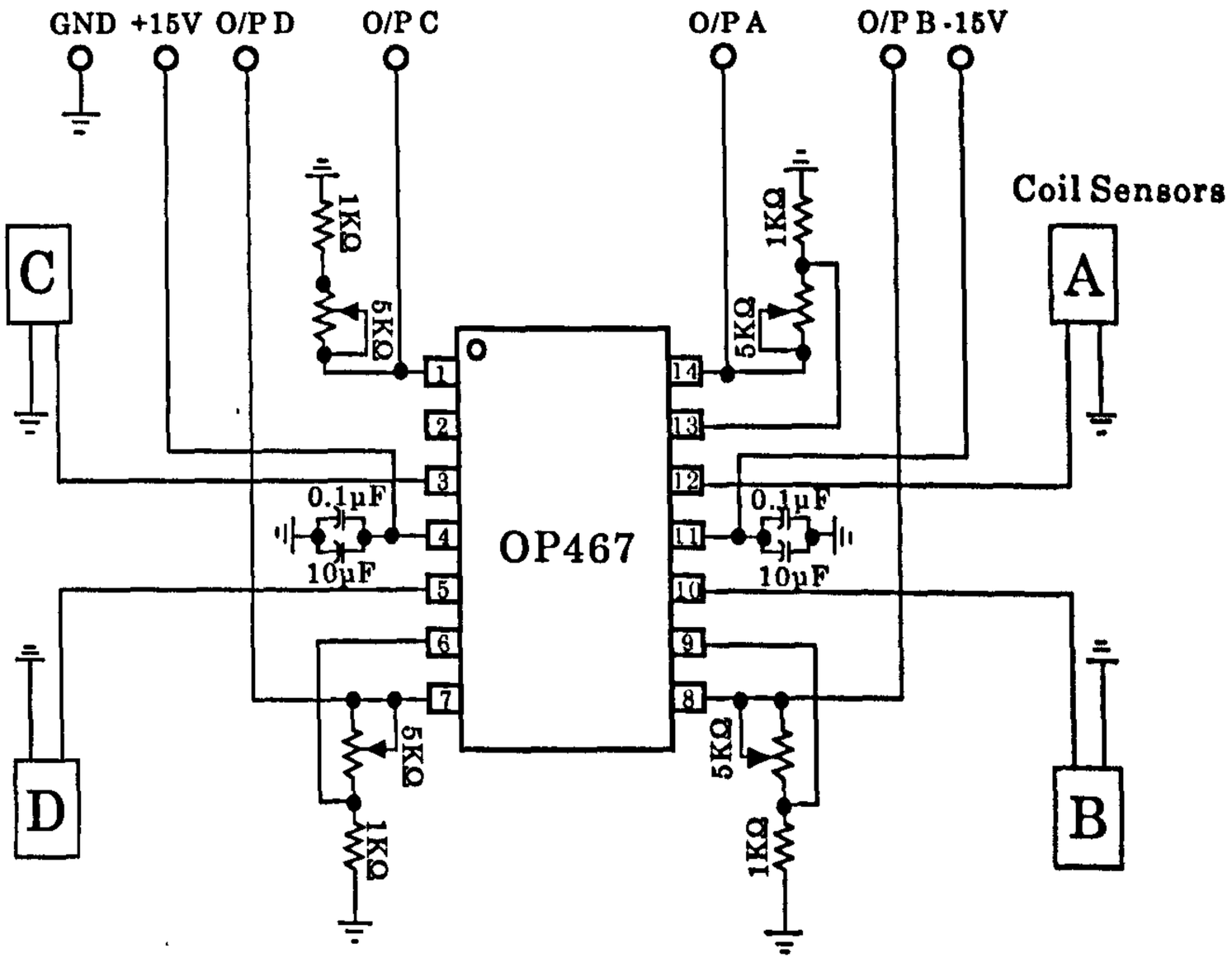


Figure B.1: Coil sensor signal amplifier with trimmers [1]

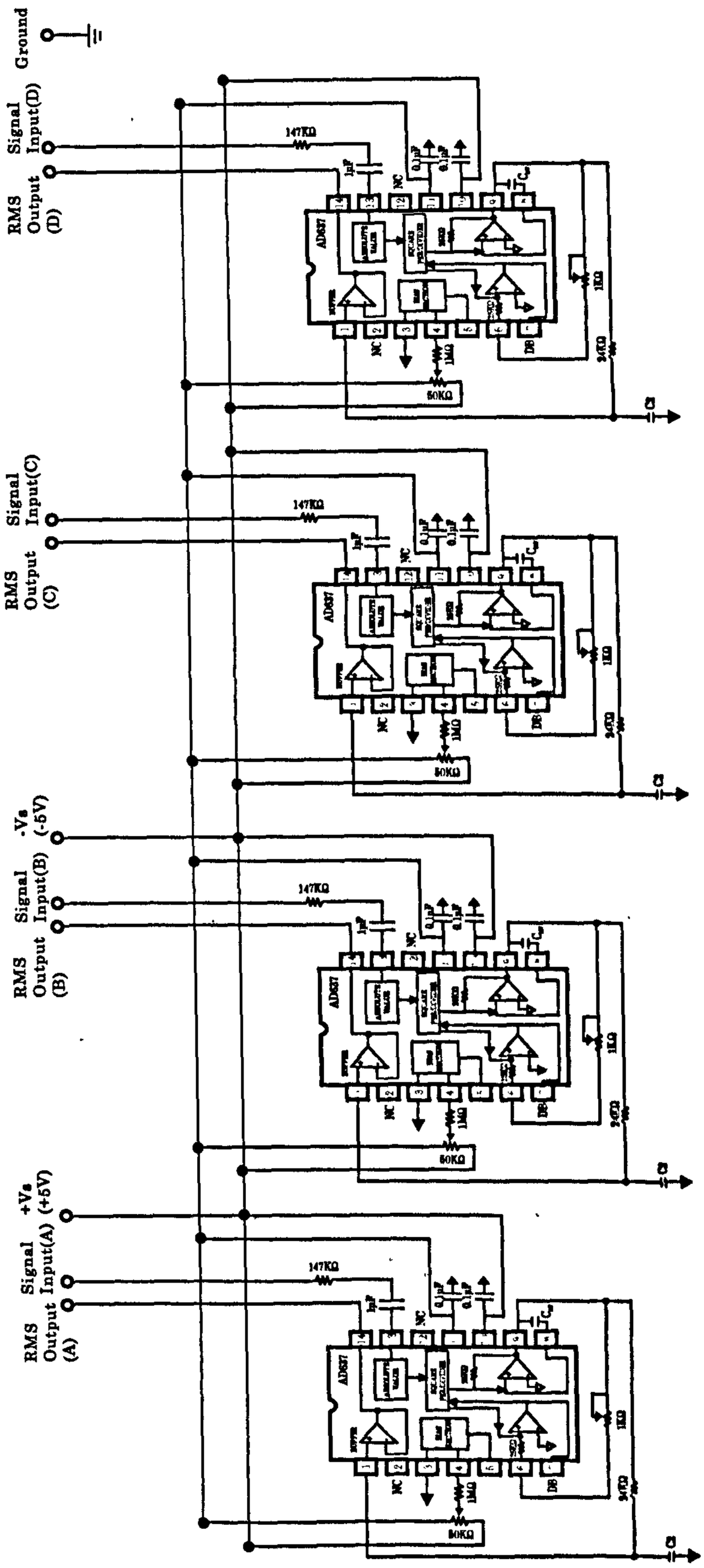


Figure B.2: RMS-DC converting circuit [1]

B.1.2 Revised Sensor System

Power Supply	Circuits
±5V	Differential amplifiers (OP467), demodulator (AD630), optical encoders, potentiometer
+15V	PWM (LM3524)
+20V	H-bridge (3952SB)
GND	All

Table B.1: Power supply assignment

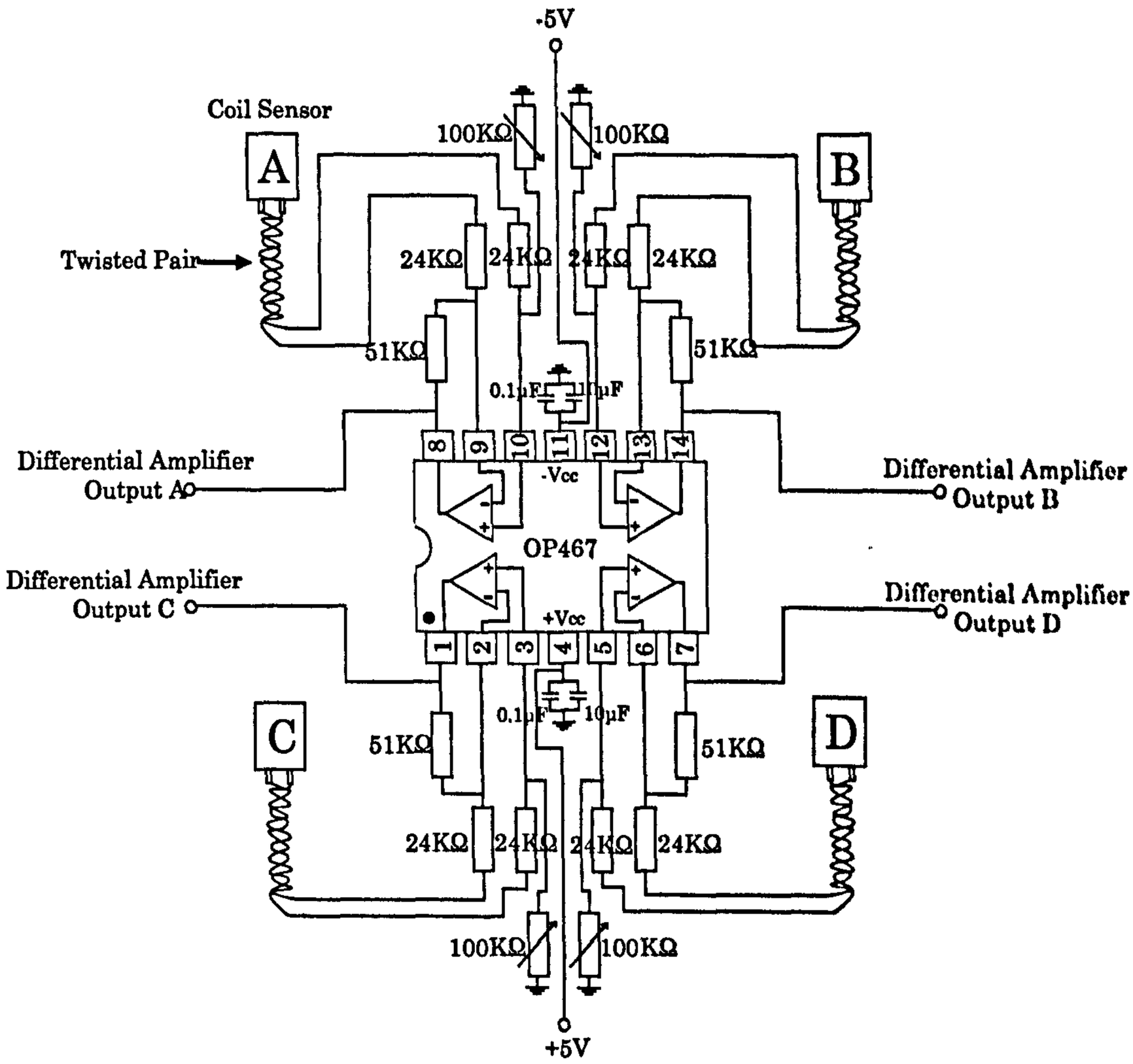


Figure B.3: Coil sensors and differential amplifiers with trimmers

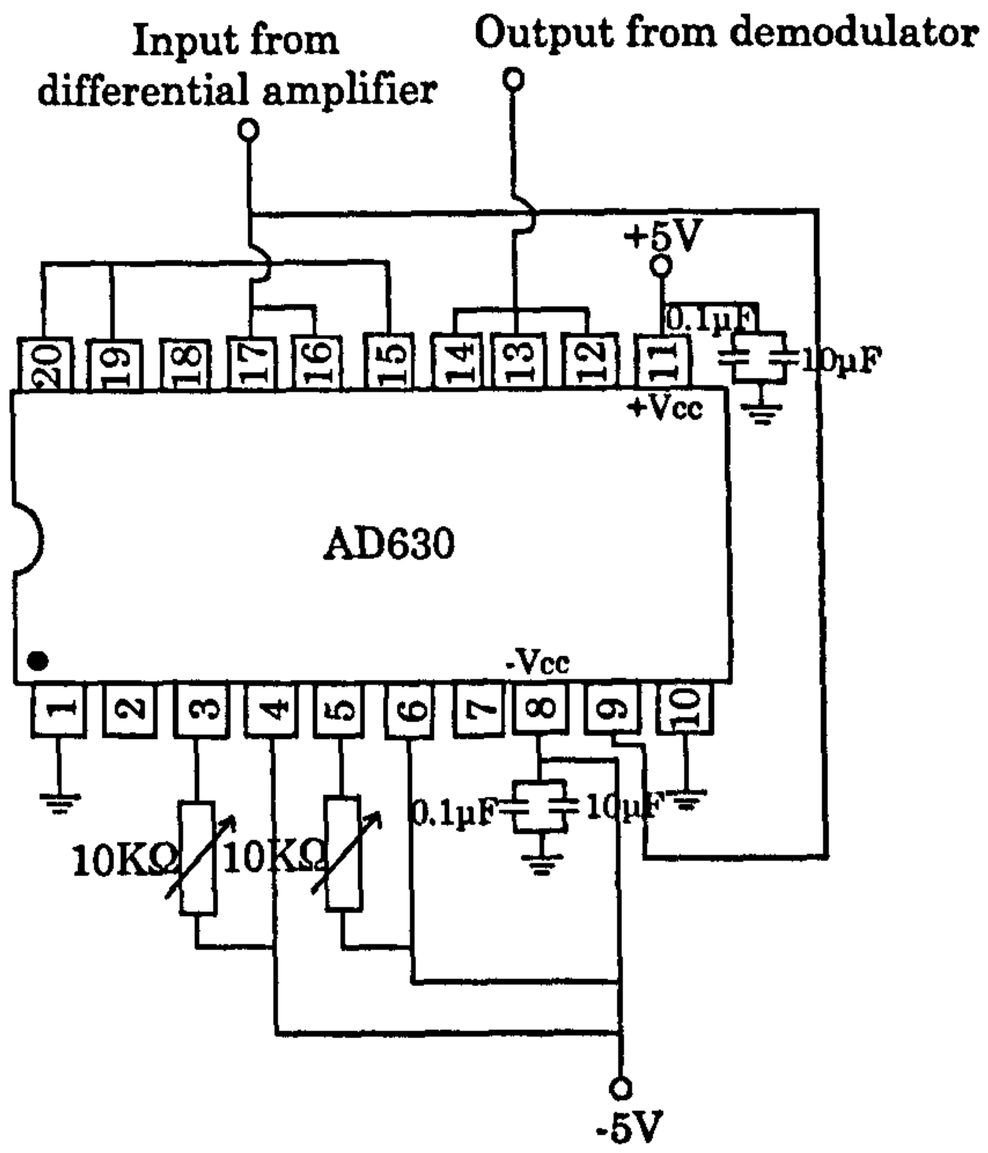


Figure B.4: Demodulator

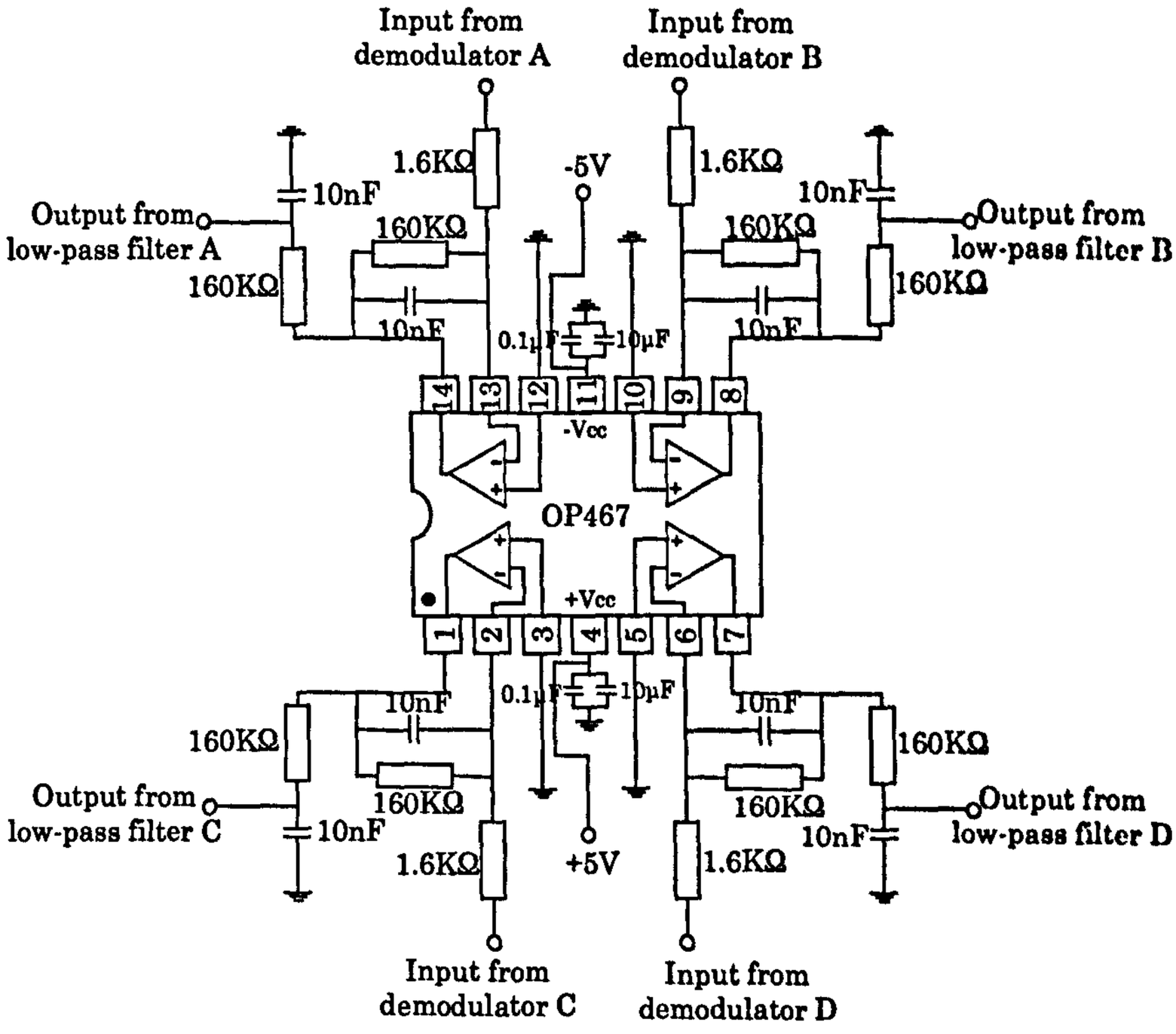


Figure B.5: Low pass filters

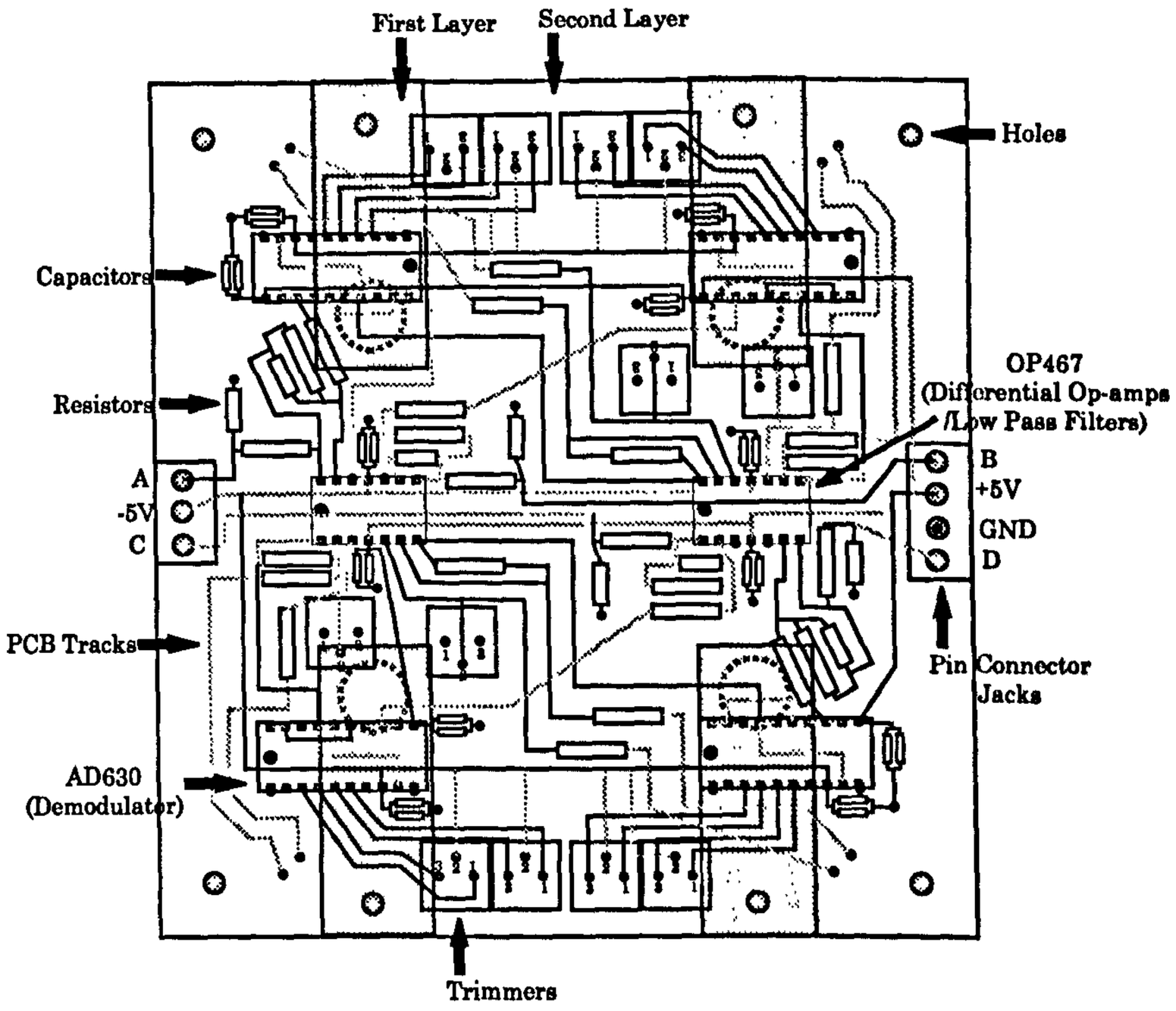


Figure B.6: Revised sensor plate PCB layout

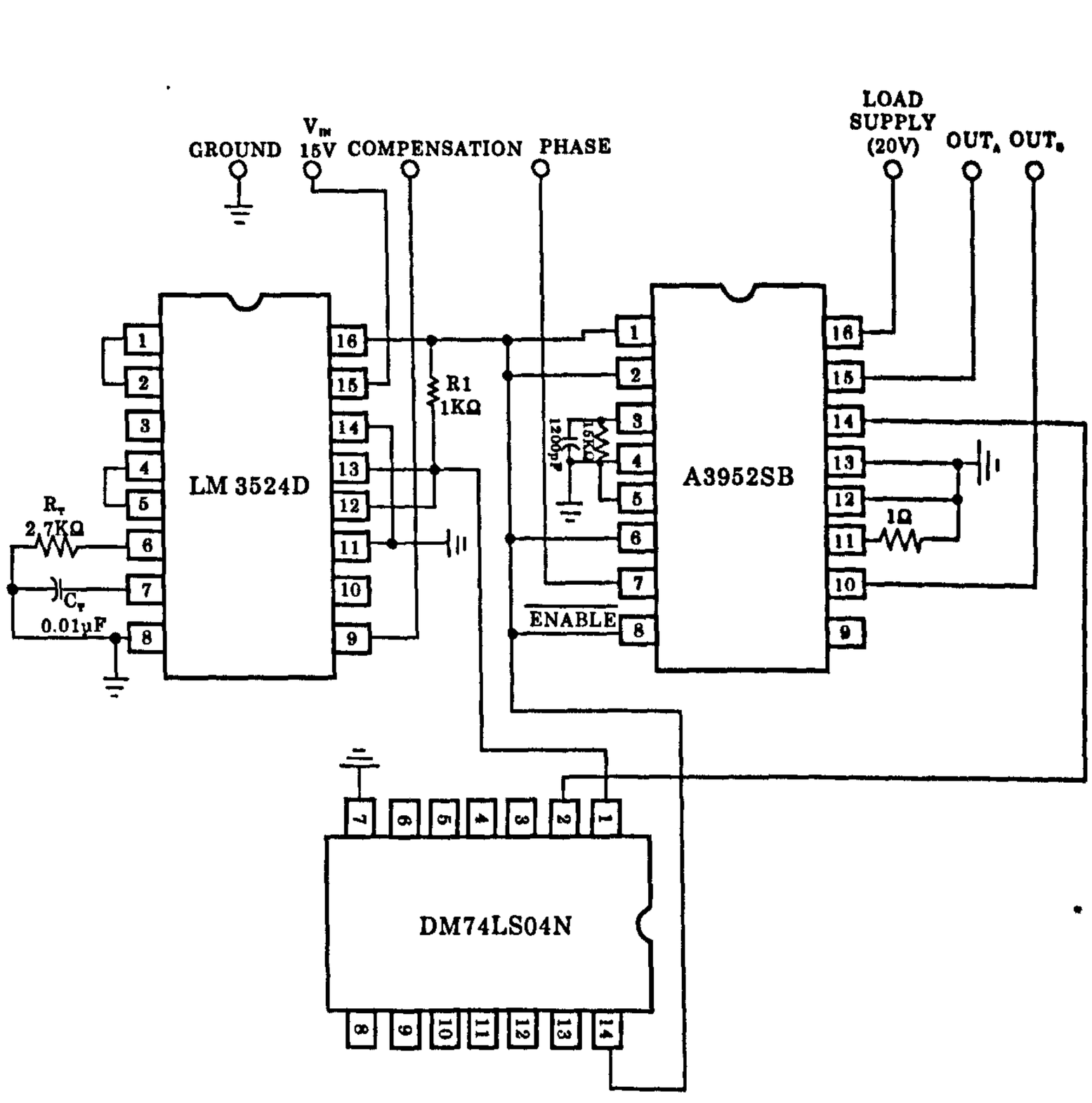


Figure B.7: PWM, comparator and H-bridge circuit diagrams [1]

B.2 Pictures of the PCB Designs

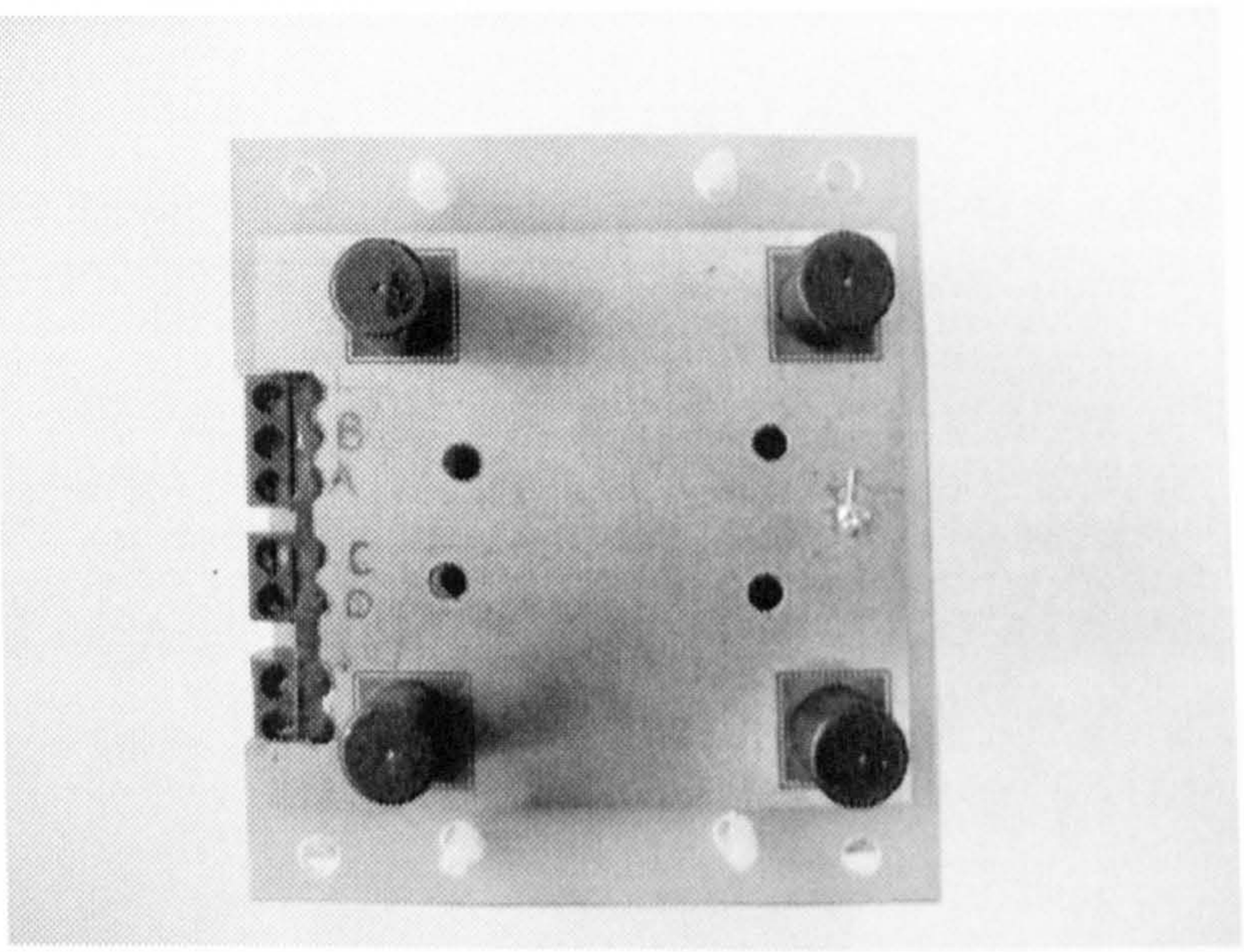


Figure B.8: Original design of the top plate

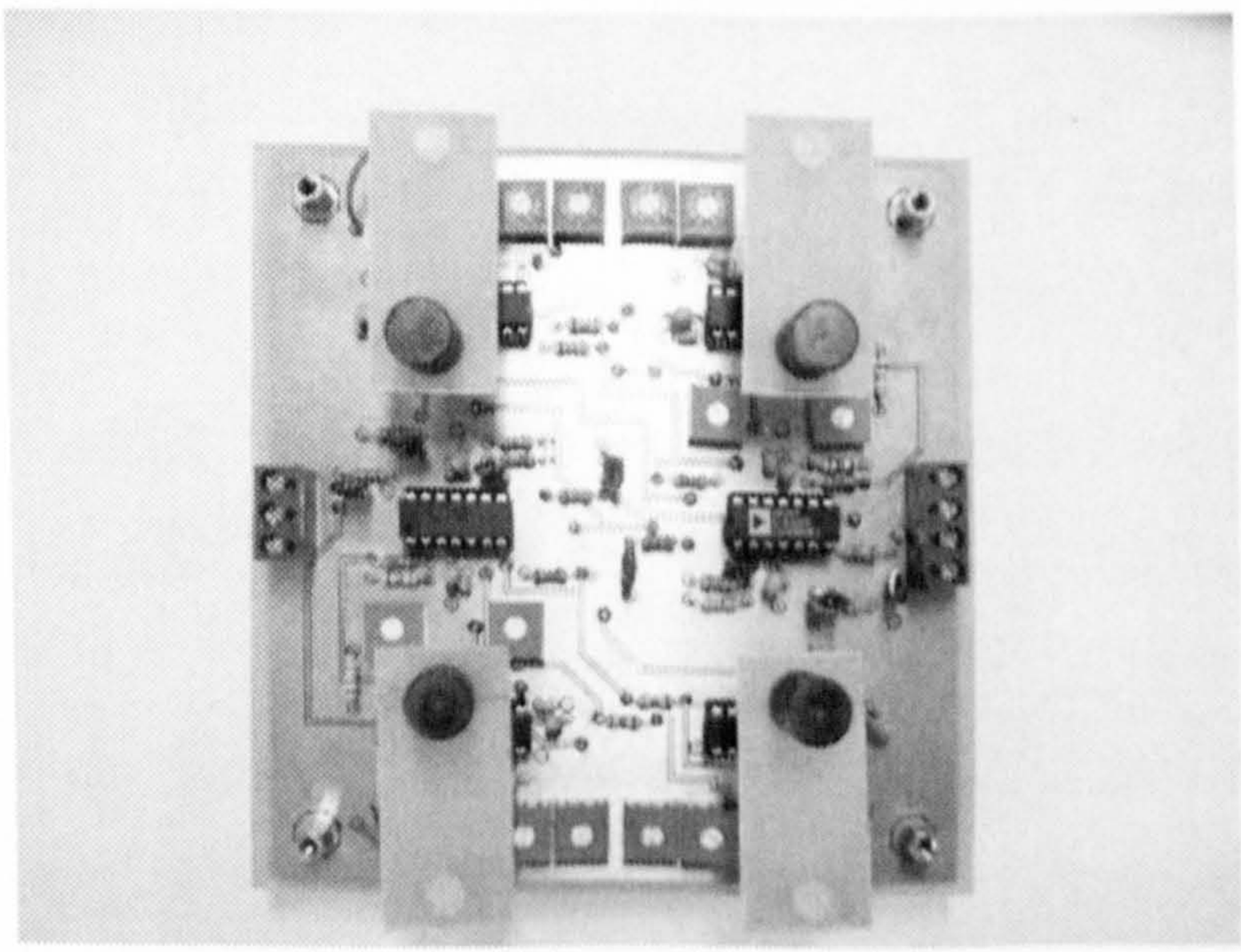


Figure B.9: Improved design of the top plate

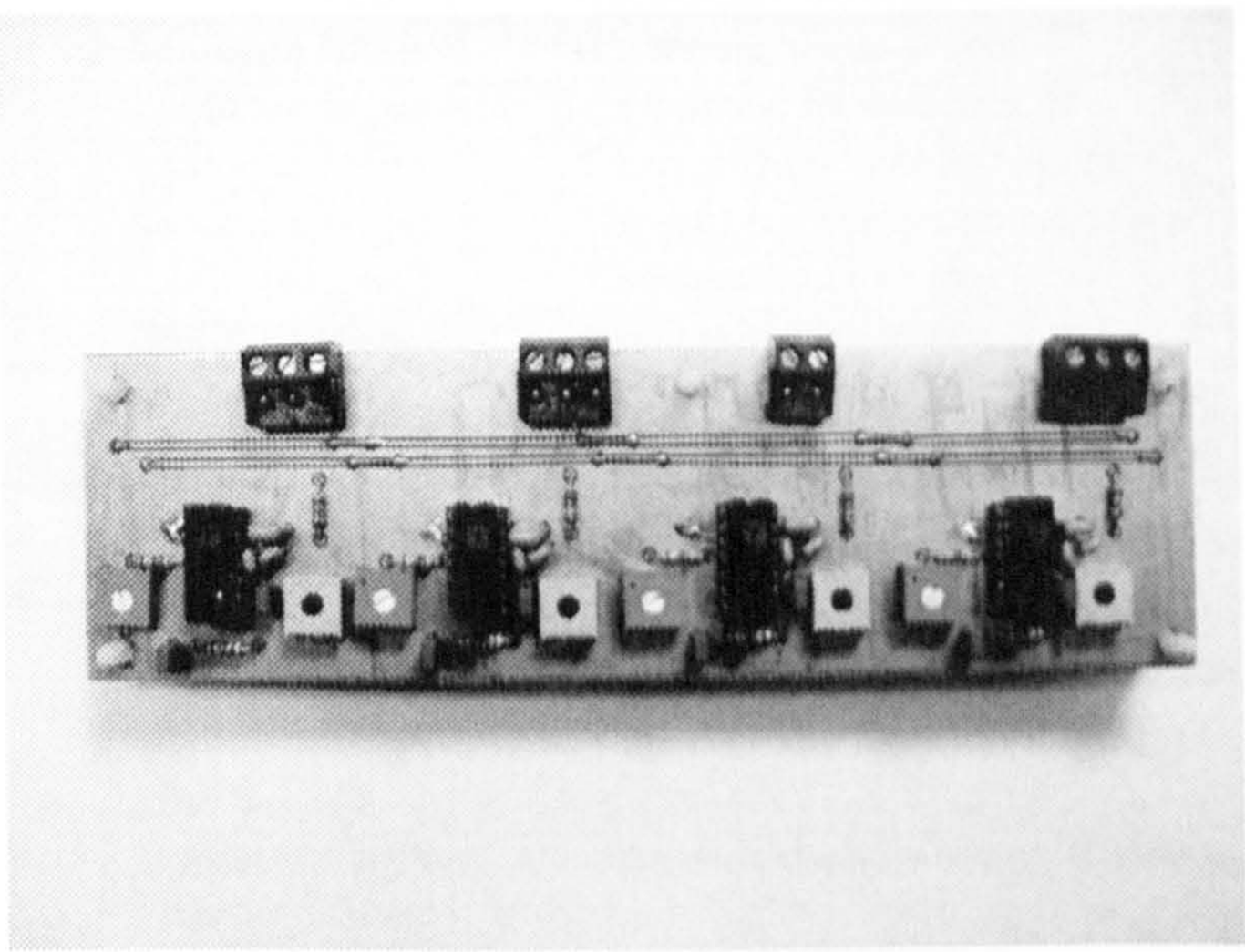


Figure B.10: RMS-DC converting circuit on PCB board

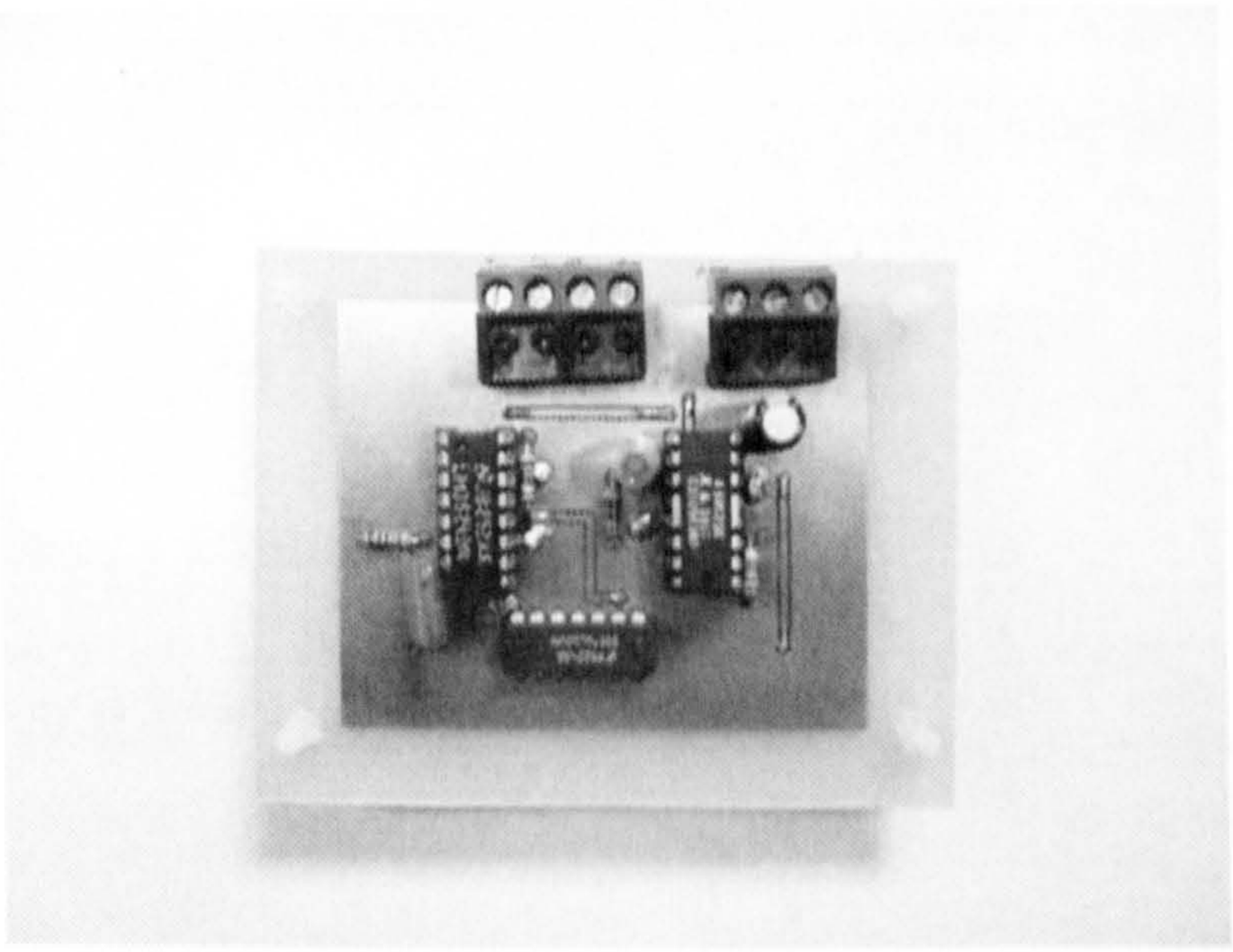


Figure B.11: PWM, comparator and H-bridge circuits on PCB board

B.3 I/O Pin Assignment

Pin Number	Pin Description	Pin Connections from/to I/O Board
1	Analogue Input 0	d.c output from sensor A
3	Analogue Input 1	d.c output from sensor B
5	Analogue Input 2	d.c output from sensor C
7	Analogue Input 3	d.c output from sensor D
9	Analogue Input 4	Output from the potentiometer
18	Analogue Ground	Ground for the power supplies
19	DAC0 Output	*Compensation* of the Y-motor PWM circuit
23	DAC1 Output	*Compensation* of the X-motor PWM circuit
25	Digital Ground	Ground for the Y-shaft encoder
26	User Clock Input 0	Clock from the Y-shaft encoder
29	Digital Ground	Ground for the X-shaft encoder
30	User Clock Input 1	Clock from the X-shaft encoder
49	Digital I/O Port A Line 0	*Phase* of the Y-motor H-bridge
50	Digital I/O Port A Line 1	*Phase* of the X-motor H-bridge
51	Digital I/O Port A Line 2	*Phase* of the top-motor H-bridge

Table B.2: Pin assignment for I/O board 1

Pin Number	Pin Description	Pin Connections from/to I/O Board
18	Analogue Ground	
19	DAC0 Output	*Compensation* of top-motor PWM circuit

Table B.3: Pin assignment for I/O board 2

B.4 Appendix Bibliography

[1] M Liao. Shaky hand. Msc report, Control Systems Laboratory, Intelligent Systems Group, The University of York, 2004.

Appendix C

Nomenclature/Glossary

$\underline{c}$	Position of the centre of the plate
d_x	Width of the top plate (distance between sensors A and B)
d_y	Length of the top plate (distance between sensors A and C)
emf	Electromotive force
$e(t)$	Error signal at time t
e_d	Diagonal difference, one of the error outputs resulted from four sensors
e_l	Linear error ratio, one of the error outputs resulted from four sensors
e_x	X-axis translational error
e_y	Y-axis translational error
e_θ	Angular displacement error
m	a slope of the tangent at point $\underline{w}$
$r(t)$	Reference signal at time t
rpm	Revolutions per minute
v_c	Velocity of the carriage
$\dot{v}_c$	Acceleration of the carriage
v_{cX}	Velocity of the X-carriage
v_{cY}	Velocity of the Y-carriage
v_x	Linear velocity along horizontal direction
v_y	Linear velocity along vertical direction
v_L	Lateral velocity perpendicular to the track

v_{LX}	Lateral velocity resolved in horizontal direction
v_{LY}	Lateral velocity resolved in vertical direction
v_S	Set-point velocity of the plate
v_T	Tangential velocity along the track
$v_{T\,error}$	Velocity error along the wire track
v_{TX}	Tangential velocity resolved in horizontal direction
v_{TY}	Tangential velocity resolved in vertical direction
v_X	Velocity of the plate in X-direction
v_Y	Velocity of the plate in Y-direction
$\underline{w}$	Position of the nearest point on the wire track from the centre of the plate
x_c	X coordinate of the centre of the plate
x_w	X coordinate of the nearest point on the wire track from the centre of the plate
y_c	Y coordinate of the centre of the plate
y_w	Y coordinate of the nearest point on the wire track from the centre of the plate
A	Area of windings of a coil
AI	Analogue input program of author's I/O software
AO	Analogue output program of author's I/O software
C	Capacitance of the windings of a coil
CGP	Cartesian genetic programming
CS	Control solutions program of author's I/O software
DO	Digital output program of author's I/O software
DO_{value}	An integer value
E	Back electromotive force produced by the rotor movement
ESL	Effective series inductance
ESR	Effective series resistance
F_c	Force on the carriage
F_{fc}	Frictional force due to the rotation of the carriage at the bearing
GP	Genetic programming
I	Electric current in a circuit
I_m	Inertia of the motor armature

I_p	The moment of inertia of the plate and pickup assembly
J_{V_α}	fitness of α signal solution
J_{V_ϕ}	fitness of ϕ signal solution
K	An arbitrary constant term
K_{amp}	A scaling constant
K_{coil}	A scaling constant
K_{fc}	Gain in rotational motion due to friction in motion of the carriage at the bearing
K_{fl}	Gain in linear motion due to friction in motion of the carriage
K_{fp}	Viscous friction torque at the bearing of the plate
K_g	Gear box gain
K_l	Conversion ratio from angle of rotation to linear displacement
K_m	Electromotive force constant
K_{rms}	A scaling constant
K_t	A scaling constant
K_D	Derivative control gain
K_I	Integral control gain
K_P	Proportional control gain
K_θ	A scaling constant
L	Local slope of the wire track
L_m	Inductance of the motor windings
MAS	Multi-agent system
M_c	Mass of the carriage assembly
Mc_X	The mass of the carriage assembly on the X-motor
Mc_Y	The mass of the carriage assembly on the Y-motor
N	Number of turns of a coil
Original Algorithms	$V_\alpha = (V_A + V_B) - (V_C + V_D)$
	$V_\phi = (V_A + V_D) - (V_B + V_C)$
PID	Proportional+Integral+Derivative
PWM	Pulse width modulation
R_m	Resistance of the motor windings

SAM	Social attentive monitoring
SM	Speed measurement program of author's I/O software
STD	Standard deviation
T	Sampling period
T_{fl}	Viscous friction torque acting against the motion of the carriage
T_{fp}	Viscous friction torque acting against T_p
T_m	Output torque of a motor
$T_{m\,Top}$	Total torque for the theta motor
$T_{m\,XY}$	Total torque for the X- and Y-motors
T_p	Output torque of the plate
T_{OFF}	Duration of a motor 'off' time
T_{ON}	Duration of a motor 'on' time
$(V_a)_{aug}$	Voltage at which makes a motor run at average speed
$V_c(t)$	Output signal from a PID controller at time t
V_{coil}	Voltage induced in an inductor
V_i	emf coupled into the inductor
V_m	Input voltage applied to the motor
V_{pot}	Voltage from a potentiometer
$V_{top\ demand}$	Top-motor voltage demand
V_A	Output voltage from sensor A
V_B	Output voltage from sensor B
V_C	Output voltage from sensor C
V_D	Output voltage from sensor D
$V_{L\,demand}$	Lateral voltage demand
$V_{O\,rms}$	rms voltage from an inductor
$V_{T\,demand}$	Tangential voltage demand
$V_{X\,demand}$	X-motor voltage demand
$V_{Y\,demand}$	Y-motor voltage demand
V_α	Lateral offset error signal
V_ϕ	Angle offset error signal

α	Vertical offset of the wire measured in a frame relative to the plate
β	The plate geometry representation
γ	The plate geometry representation
δ	Proximity of the a.c. waveform on the wire
δ_A	The distance between the sensor A and the wire
δ_B	The distance between the sensor B and the wire
δ_C	The distance between the sensor C and the wire
δ_D	The distance between the sensor D and the wire
θ	Angle of the plate assembly in the fixed frame
μ	Relative permeability of a coil
ϕ	Angle of rotation measured in a frame relative to the plate
ω	Frequency of the a.c. waveform on the wire
ω_m	Angular velocity of the motor
$\dot{\omega}_m$	Angular acceleration of the motor
ω_p	Angular velocity of the plate
$\dot{\omega}_p$	Angular acceleration of the plate