

CODEN: LUTFD2/(TFRT-5374)/1-33/(1987)

# Towards an experimental set-up for robot learning

Pär Kvist

Department of Automatic Control  
Lund Institute of Technology  
November 1987

<b>Department of Automatic Control</b> <b>Lund Institute of Technology</b> P.O. Box 118 S-221 00 Lund Sweden		<i>Document name</i> Master thesis
		<i>Date of issue</i> November 1987
		<i>Document Number</i> CODEN: LUTFD2/(TFRT-5374)/1-33/(1987)
<i>Author(s)</i> Pär Kvist		<i>Supervisor</i> Lars Nielsen
		<i>Sponsoring organisation</i>
<i>Title and subtitle</i> An experimental set-up for robot learning		
<i>Abstract</i> <p>Usually robot tasks are well defined and preprogrammed. An approach to make the robot perform ill defined tasks is to let it have a learning behaviour. Here a system for investigation of different aspects of different kinds of robot learning is developed. The system is based on a simplified version of table tennis, where the ball is rolling on a table. Supervised learning is treated and experiments are done.</p>		
<i>Key words</i> Real-time system, Robot control, Supervised learning		
<i>Classification system and/or index terms (if any)</i>		
<i>Supplementary bibliographical information</i>		
<i>ISSN and key title</i>		<i>ISBN</i>
<i>Language</i> English	<i>Number of pages</i> 33	<i>Recipient's notes</i>
<i>Security classification</i>		

The report may be ordered from the Department of Automatic Control or borrowed through the University Library 2, Box 1010, S-221 03 Lund, Sweden, Telex: 33248 lubbis lund.

# Contents

<b>Preface</b> . . . . .	2
<b>1. Introduction</b> . . . . .	3
<b>2. The Game</b> . . . . .	4
2.1 The Arrangement . . . . .	4
2.2 Possible Rules of the Game . . . . .	5
2.3 The Actual System . . . . .	5
<b>3. Physical Equipment</b> . . . . .	6
3.1 The Robot . . . . .	6
3.2 The Computers . . . . .	7
3.3 The Software . . . . .	8
<b>4. System Description</b> . . . . .	9
4.1 The Game Computer . . . . .	9
4.2 The Control Computer . . . . .	14
4.3 The Outside World . . . . .	17
<b>5. Robot Control</b> . . . . .	22
5.1 Kinematics . . . . .	22
5.2 Inverse Kinematics . . . . .	23
5.3 PID-Regulators . . . . .	25
<b>6. Principle of Learning</b> . . . . .	27
6.1 Algorithm of Learning . . . . .	27
6.2 One typical experiment . . . . .	29
<b>7. Conclusions</b> . . . . .	32
<b>8. References</b> . . . . .	33

# Preface

This Master thesis was done during the very rainy summer of 1987 at the Department of Automatic Control, Lund Institute of Technology. Due to unforeseen circumstances, the aim of the work was changed. I had a great deal of freedom of selecting this new aim. Under Lars Nielsen's supervision, the work has been interesting and stimulating. Lars Nielsen's criticism and proposals for improvements have been very valuable.

I wish to make a collective thank to all the personnel at the Department of Automatic Control, specially the supportive help from Rolf Braun. I am also indebted to the Department of Industrial Automation for being able to use their robot, and to the helpful assistance of Per Göran Nilsson. A special thank is due to Lars Nielsen.

# 1. Introduction

There are two extremes of learning behaviour. Biological systems, e.g. humans or animals, learn from external stimuli, partially autonomous and partially supervised by e.g. parents. Industrial robots, on the other hand, are very rigid in behaviour and need well defined (preprogrammed) tasks. An interesting topic of study is partially learning systems. Here we think of robots where some functions are well defined (as usual), but with some added capability of learning ill defined tasks. Learning systems have, with varying intensity, been studied over the past decades, but the interest has currently been reinforced due to new hardware. There are now commercially available VLSI chips implementing neural networks, and there will probably in the near future be plug in expander boards to personal computers implementing the same or similar functions.

The purpose of the present work is to develop a system such that a robot could play a simplified version of table tennis, where the ball is rolling on the table. Many interesting aspects of different kinds of learning could be studied. We have concentrated the work on basic building of the system and made it in a way such that a future extension should be straightforward. The system consists of a man-robot interface for flexible interaction, robot control, simulation of the outside world, a monitoring capability for animation of the ball and the robot hand, ball prediction, and supervised learning of basic shots. We use three computers, a Control Computer, a Simulation/Monitoring Computer, and a Game Computer, implementing these features. The Control Computer does the robot control and communicates with the operator via the man-robot interface. The simulation of the ball and the monitoring are done in the Simulation/Monitoring Computer. In the Game Computer, the algorithm of supervised learning and ball prediction are performed.

In the next chapter, Chapter 2, the game will be described. Chapter 3 is devoted to the physical equipment, that is, the robot and the computers. A system description is done in Chapter 4, where each computer's software is presented. How the interface to the user is working is also treated here. Two fundamental problems in robotics, the problem of kinematics and the problem of inverse kinematics, are penetrated in Chapter 5. A brief discussion about PID regulators concludes the chapter. The algorithms and formulas which are used to implement the specific type of learning used here, are treated in Chapter 6. A typical experiment is also presented in this chapter. A final discussion and some conclusions are done in the last chapter, Chapter 7.

## 2. The Game

The physical set-up, the rules of the game, and the actual system will be presented.

### 2.1 The Arrangement

The arrangement consists of a black rectangular table with borders at the long sides, a white ball rolling on the table, a TV camera hanging above the table, a Computer Vision unit that, given an image from the TV camera, gives the ball position  $(x, y)$  in a coordinate system attached to the table, two Game Computers containing prediction algorithms and learning facilities, two Control Computers doing coordinate transformations from Cartesian space to joint angle space and performing the controlling of the robots, two Robot Interfaces mapping control signals from the Control Computers to control signals to the robots and gives measured signals from robot joint angles, and two robots with air blowers as their tools, see Figure 2.1.

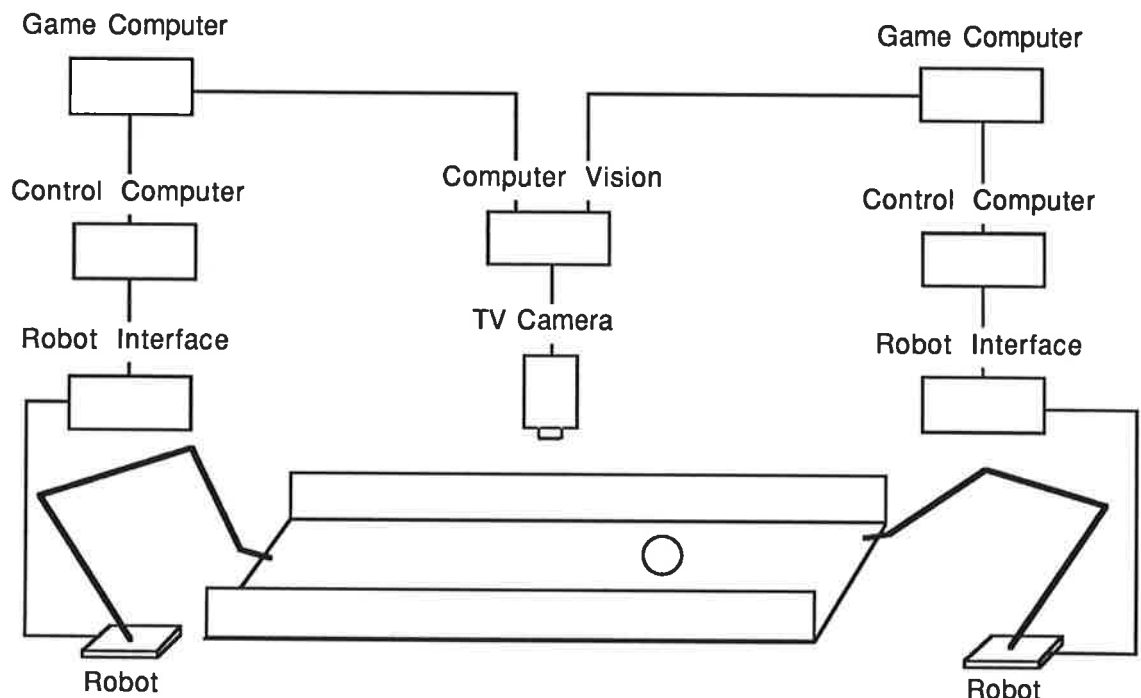


Figure 2.1 The scenario

Each player consists of a Game Computer, a Control Computer, a Robot Interface, and a robot. When the ball arrives at one short side, the player positions the robot tool at the point where the ball is predicted to hit the short side. The air blower blows at the ball, and, if the air pushes the ball in the right direction, the ball leaves the short side and rolls against the other player, possible after one or several bounces at the borders. The shooting at the ball affects the direction of the ball velocity in a complicated manner,

and therefor this relation is suitable for learning instead of explicit formula programming.

## 2.2 Possible Rules of the Game

A match is played best of five sets with a short pause between the sets for consultation with the trainer. The player who first wins ten balls wins the set. The players serve five times each in a set.

This game gives us possibilities to study several aspects of learning. Some properties, such as prediction of the ball position, could be preprogrammed. Other properties, such as selection of an appropriate game strategy, could be modified during the match. Before the match supervised learning could be used to learn basic shots. Between the sets the trainers could have the possibility to make basic changes in the tactic- and strategy learning capabilities.

## 2.3 The Actual System

In this system basic shot learning is studied. In principle it is possible to model the shots by study of the aerodynamic interaction between the blower and the ball. The idea here is instead to simplify the modelling by training the robot so that after each shot it learns more about the effect it causes on the ball. The ball, the TV camera, and the Computer Vision unit are simulated in the Simulation/Monitoring Computer.

### 3. Physical Equipment

The physical properties of the robot and the interface to the outside world are discussed in the first section. A description of the actual computers, the connection between these and the Robot Interface are treated in the second section. The third section is devoted to the software.

#### 3.1 The Robot

A five linked robot with five joint angles is used, see Figure 3.1. The first link is at the robot base and has length 0. The robot is built at the Department of Industrial Automation, Lund Institute of Technology. An interface to the outer world is available. The measured signals are measured from the interface. The joint angles  $\theta_1, \theta_2, \theta_3, \theta_4,$  and  $\theta_5$  (in radians) vary in the following intervals

$$\begin{aligned} -0.88 &\leq \theta_1 \leq 0.88 \\ 1.21 &\leq \theta_2 \leq 2.05 \\ -2.0 &\leq \theta_3 \leq -1.13 \\ -1.7 &\leq \theta_4 \leq 1.7 \\ -3.23 &\leq \theta_5 \leq 3.23 \end{aligned}$$

These angles are mapped into measured signals in the interval -10 V to +10 V.

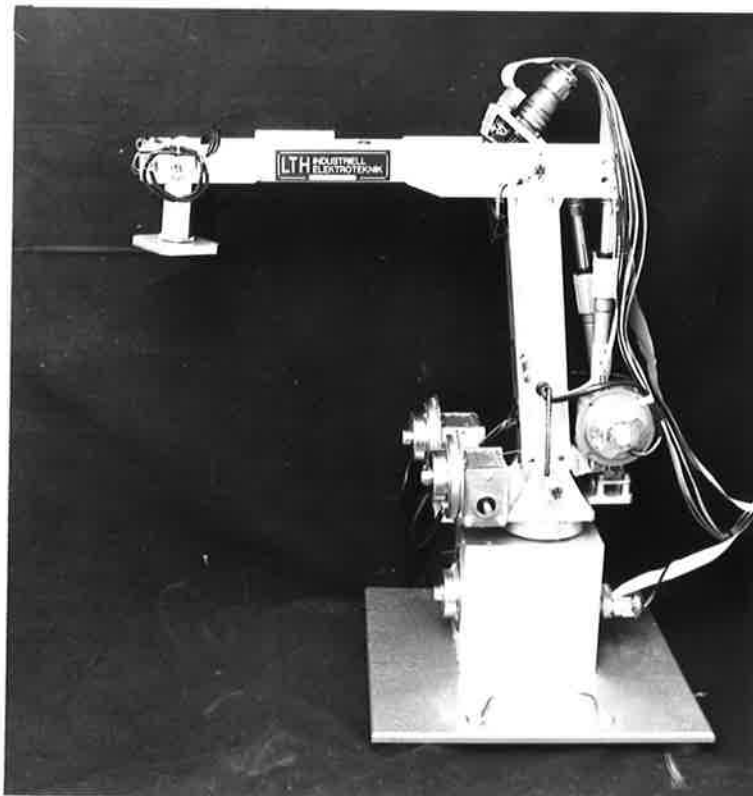


Figure 3.1 The robot



The geometric description of the robot is based on a Cartesian coordinate system with origin at the robot base, with  $z$ -axis in a direction orthogonal against the floor and  $x$ -axis in the same direction as the robot arm, when the robot is at the initial position, see Figure 3.1. When the first joint angle is changed the robot rotates around the  $z$ -axis. Except the joint angle at the hand, this joint angle is the only one which is rotational around the  $z$ -axis. When we want to reach a point with the tool, we first direct this joint angle in the right direction. When this is done, the problem is a planar one. When the second joint angle is changed, the second link is rotated around its base relative to the  $xy$ -plane. The third and fourth joint angle is rotational around the intersection between the actual link and the previous one, relative the  $xy$ -plane. Notice that when the second, third, or fourth joint angle is changed, the robot tool is changed in one plane.

The actual link lengths  $\lambda_1$ ,  $\lambda_2$ , and  $\lambda_3$  (in meters) are

$$\lambda_1 = 0.38$$

$$\lambda_2 = 0.40$$

$$\lambda_3 = 0.10$$

### 3.2 The Computers

Three IBM PC-AT compatible computers, Tandon, equipped with AD/DA plug in boards, are used. Communication with other computers and with the Robot Interface is achieved using these. The voltage range is -10 V to +10 V. The connections between the different computers and the Robot Interface are listed below. The following notations will be used: *SC*, *GC*, *CC*, *RI* meaning the Simulation/Monitoring Computer, the Game Computer, the Control Computer and the Robot Interface respectively. Further, we will write, for example,

$$SCO_i \rightarrow CCI_j$$

meaning channel number  $i$  in the Simulation/Monitoring Computer's output section is connected to channel number  $j$  in the Control Computer's input section. The connections due to the Simulation/Monitoring Computer are

$$SCO0 \rightarrow GCI0$$

$$SCO1 \rightarrow GCI1$$

$$SCO4 \rightarrow GCI4$$

$$SCO5 \rightarrow GCI5$$

$$SCI0 \leftarrow CCO0$$

$$SCI1 \leftarrow RIO3$$

$$SCI2 \leftarrow RIO2$$

$$SCI3 \leftarrow RIO1$$

$$SCI4 \leftarrow RIO4$$

$$SCI5 \leftarrow RIO5$$

Next the connections seen from the Game Computer are listed.

*GCO0* → *CCI0*

*GCO4* → *CCI6*

*GCO5* → *CCI7*

*GCI0* ← *SCO0*

*GCI1* ← *SCO1*

*GCI4* ← *SCO4*

*GCI5* ← *SCO5*

Finally, the Control Computer's connections are listed.

*CCO0* → *SCI0*

*CCO1* → *RII3*

*CCO2* → *RII2*

*CCO3* → *RII1*

*CCO4* → *RII4*

*CCO5* → *RII5*

*CCI0* ← *GCO0*

*CCI1* ← *RIO3*

*CCI2* ← *RIO2*

*CCI3* ← *RIO1*

*CCI4* ← *RIO4*

*CCI5* ← *RIO5*

*CCI6* ← *GCO4*

*CCI7* ← *GCO5*

Note that some information is redundant, but we have listed it for convenience.

### 3.3 The Software

The programs are written in Modula-2 (Astor, 1986). There are several advantages of using this kind of language, one is that the program can be divided into modules. Large programs can be easily read, if data and the operations on data are structured together in modules. It is also desirable that the rest of the program only can operate on data via the operations. If the operations change the data in a consistent way and if data only can be accessed via these operations, the data can never be brought to an inconsistent state. This fact can be used when we want to isolate errors in large programs.

A real-time kernel, developed at the Department, is used to implement parallel processes. Real-time programming concepts such as processes and monitors are used (Young, 1987; Peterson and Silberschatz, 1985). Modules containing primitives for communication via AD/DA, for mathematical functions, for conversion between numbers and strings, and primitives for graphical presentation is also used.

## 4. System Description

The system will be described by discussing the following sections, the Game Computer, the Control Computer, and the Outside World respectively. The description of the software is built on process graphs. We will use the following convention here : active parts, parallel processes, will be illustrated as ellipses and passive parts, monitors or passive procedures, will be viewed as rectangles. The arrows can be seen as communication links. Very often several processes communicate with each other via a monitor. The reason for this is to guarantee mutual exclusion. In Figure 4.1 we can see the interaction between the different units and the robot.

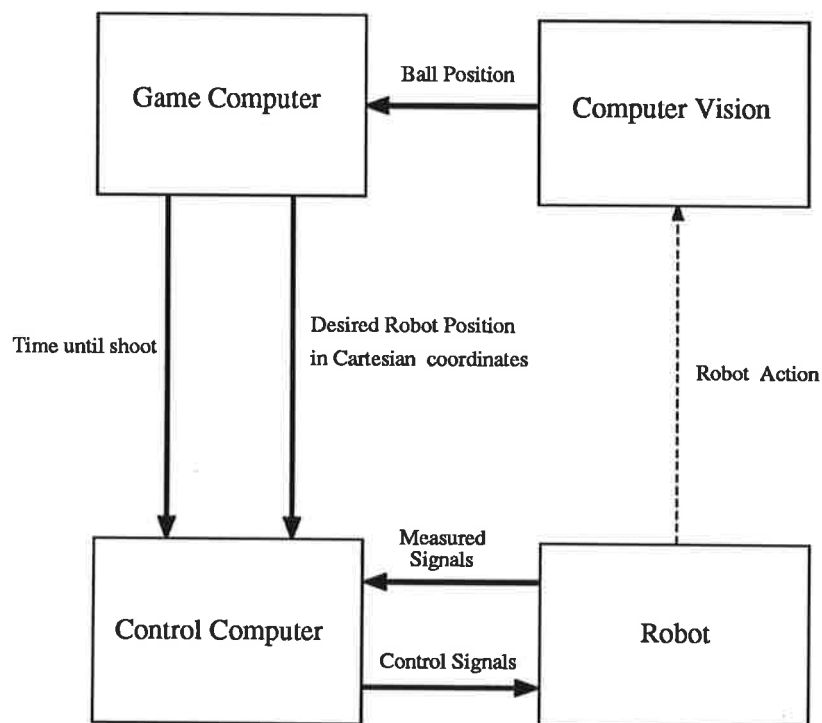


Figure 4.1 A schematic description of the system

The simulation of the Computer Vision is done in a computer, the Simulation/Monitoring Computer. We can see the simulated system in Figure 4.2. In Section 4.4, where we will discuss the Outside World, we will also discuss the Simulation/Monitoring Computer and the models we use there.

### 4.1 The Game Computer

The Game Computer can, in its current state of development, be trained to predict where the ball will hit the short side.

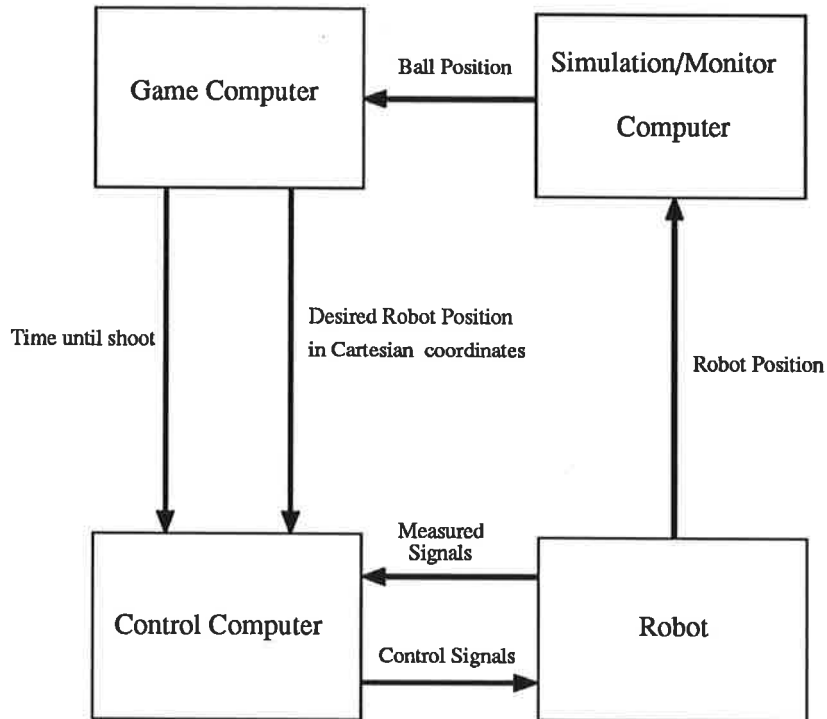


Figure 4.2 A schematic description of the simulated system

### Process Description

From the outer world the ball position is gotten into the computer. The process CalcBallPosVel calculates the ball velocity  $v_x$  and  $v_y$ , see Figure 4.3. The monitor BallPosVelMonitor is the storing place for the ball position and velocity. With this information the process CalcRobotRef can calculate where the ball will hit the short side. The fact that the ball can bounce against a border must also be taken care of, and is so. CalcRobotRef sends the desired position of the robot, in Cartesian space, to the Control Computer.

During the learning phase the angle of the air blower is chosen at random. The distribution of this stochastic variable is a rectangle distribution i.e. all angles in an interval have the same probability. In this context we mean by stochastic the pseudo stochastic mechanism which can be supported by a computer.

The time until the ball will arrive at the short side is also sent to the Control Computer. The Control Computer needs this information for knowing when to shoot. The reference values for the robot is also stored in RobotMonitor together with some properties about the ball. The purpose of storing this information in a monitor is that the processes which handles the learning will have to know these parameters.

The operator can communicate with the system via the process OpCom. The estimated value of the ball position and velocity, the robot reference value in Cartesian space, and the time until the ball will arrive at the short side can be viewed on the screen.

A process, Learner, has been implemented for the purpose of learning the position where the ball will hit the opponent short side, in relation to the position and angle of the air blower and ball position and velocity. This relationship is a scalar valued function of several variables. The value is the point in which the ball will hit the opponent short side and the variables

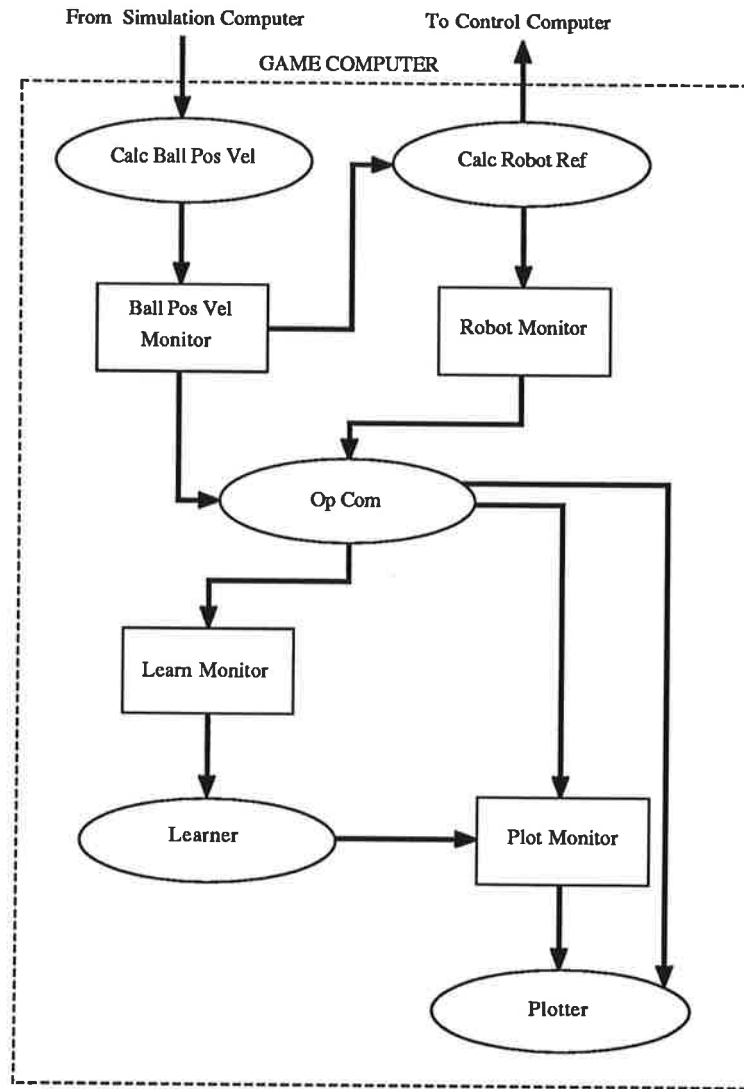


Figure 4.3 A description of how the processes in the Game Computer communicate with each other and with external units

are parameters in RobotMonitor. When the ball is arriving at our short side all the variables are known. The only variable which we have the ability to affect is the angle of the air blower. In Learner we are using the algorithm of supervised learning, which is discussed in Chapter 6. When the system is running in Learn mode, the operator must decide which of the robot hits is good enough for using as parameters to Learner. These parameters are stored in a buffer, LearnMonitor, where Learner can get them. The operator has also the possibility of seeing how the parameters are updated and how and if these parameters converges. It is for this purpose that the process Plotter has been written. The operator has the possibility of selecting different scales and different parameters for plotting. In PlotMonitor, information of in which Plot mode the Plotter is to be runned in and the parameters of learning, are stored.



Figure 4.4 The basic menu

### The communication with the operator

In the basic menu, which is presented when the system is started, the operator has three different choices : Press one of the mouse buttons when the cursor is inside the **Exit** box, or when the cursor is inside the **Show** box or inside the **Learn** box. A selection of the **Exit** box will cause the system to shut down.

If the **Show** box is selected some parameters are viewed, see Figure 4.5. A second mouse button press in the **Show** box will hide the viewed parameters and the operator will be returned to the basic menu.

The selection of the **Learn** box will cause the system to go into **Learn** mode. This menu can be seen in Figure 4.6. If the operator presses the mouse button inside the **Learn** box a second time the system will go out of the **Learn** mode and back to the basic menu.

Three different collections of learn parameters are used, **Straight**, **Down**, and **Up**. **Straight** is chosen when the ball, on it's way to the opponent short side, does not bounce, **Up** is chosen if the ball bounces against the upper border and **Down** if it bounces against the lower border. When the operator confirms a stroke, that is, when the robot has done a hit, which the operator is pleased with, the operator just press one of the mouse buttons inside the **Confirm** box. The operator cannot select the **Confirm** box before the ball has reached the other short side. The reason for this is that when the operator confirms some parameters, of which the point where the ball hits the opponent short side is one, are stored in a monitor. However, the point where the ball will hit the other side is not available until the ball has reached it. The box next to the **Confirm** box, that one which is marked by **Straight** in Figure 4.6, must be in a proper state when the **Confirm** is chosen. By pressing the mouse button inside the box, the state can be changed to **Up** and **Down**.

Sometimes it is desirable to have the parameters of learning viewed in a diagram, to see if and in what way they converge. The last row of boxes are

Exit	Show	Learn	BallPar
			x =
			y =
			vx =
			vy =
			RobotPos
			y =
			th =

Figure 4.5 The menu when Show is selected

Exit	Show	Learn			
Confirm	Straight				
ShowPlot	y	Straight	All	1.000	




Figure 4.6 The menu written when Learn is chosen

plotting command boxes, that is, if the state of one of these boxes is changed only the plot is affected, not the parameters of learning. The boxes can be set in different modes independently of each other. As will be shown in Chapter

6, we have an estimation function with terms of the form

$$\alpha_{ij} x_i x_j \quad i, j = 0, \dots, 5$$

where  $\alpha_{ij}$  is an unknown constant and  $x_i$  is a known quantity. The y box can be set in the following states : y, 0, 1, 2, 3, 4, 5, 12, 13, 14, 15, 23, 24, 25, 34, 35, 45, 11, 22, 33, 44, or 55. The different numbers refer to the constants in the different terms in the estimation function. For example, number 2 refers to  $\alpha_2 = \alpha_{02}$  and number 14 refers to  $\alpha_{14}$ . If this box is set in the first state, y, the estimated function, the estimation function and the error between these, are plotted. In this state the box marked by All can be used. If this box is set in the initial state all the parameters are plotted. By selecting the All box, only one of the plots are shown. When the y box is selected and set into another state, the constants belonging to different terms in the estimation function will be plotted. The box marked by Straight has the same meaning as the Straight box above, except that this affects the plotting. The last box is a plotting scale box. By pressing the mouse button inside it, the operator can change to different scales. When pressing, different numbers are viewed. The actual plot is that number multiplied with the real value. When the plot is outside the plot window a scale less than one is desirable to chose, and if the plot is almost undiscernible from the x-axis a scale larger than one is to be chosen. Notice that the selection of different modes does not affect the purpose until the Confirm or the ShowPlot box is selected.

## 4.2 The Control Computer

The Control Computer's task is to, given reference signals in Cartesian space, calculate reference signals in joint angle space and control the robot with PID-regulators.

### Process Description

The problem of inverse kinematics, i.e. the problem of calculating the joint angles from the desired position in Cartesian coordinates, is taken care of in the process InvKin. The formulas are derived in Chapter 5. When InvKin has done the calculations the result is stored in RobotRefMonitor. OpCom is a process which handles the communication with the operator. It is possible to change the PID-parameters in OpCom. The PID-parameters are then stored in PIDMonitor, where they are available for the process Regul. When the system is started OpCom reads the PID-parameters via FileHandler from disk, and when the system is shut downed the parameters are written back onto the disk via FileHandler. The process Regul does the basic controlling, i.e. implements the PID-regulators. These formulas will be discussed in Chapter 5. The PID-parameters and the reference values are available in monitors. Command of shooting is coming from the Game Computer directly. The measured angles and the reference angles are stored in AngleMonitor continuously, where OpCom can get them. The angles which are stored in AngleMonitor can be viewed via OpCom and is done so with help of the procedures in AngleWriter.



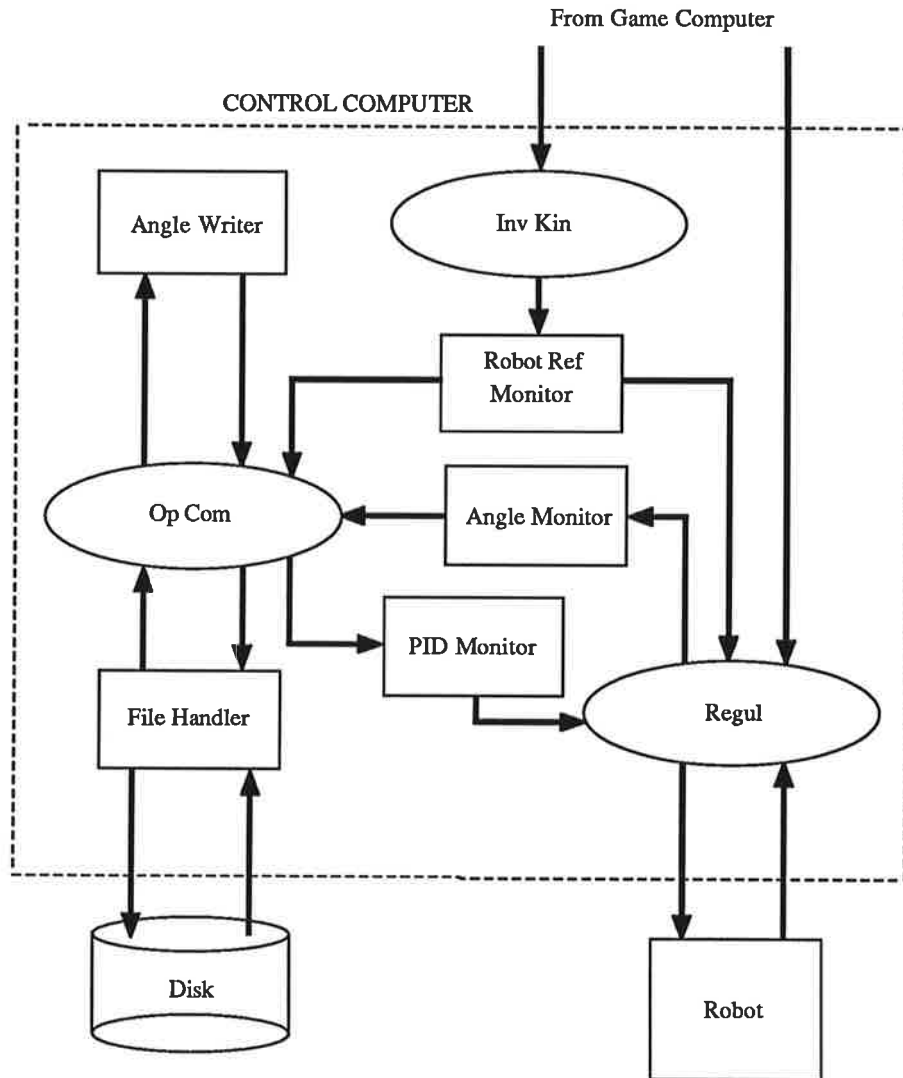


Figure 4.7 A description of how the processes in the Control Computer communicate with each other and with external units

### The communication with the operator

When the Control Computer is started the basic menu, see Figure 4.8, is shown. If the operator presses one mouse button when the cursor is inside the **Exit** box the system will save the actual PID-parameters on disk and shut down. When the system is started the latest parameters are read from disk. On the screen the five parameters of each of the five regulators,  $K$ ,  $T_d$ ,  $T_i$ ,  $b$ , and  $N$  can be seen. On the upper right corner of the screen two boxes marked by  $H$  and  $T_r$  can be seen. The meaning of these parameters and in what way a change of them affect the control are discussed in (Åström, 1987). The operator has the ability to change these parameters during controlling. A change of a certain parameter is done in the following way : Press a mouse button inside the box, which is to be changed. When this is done it is just to enter the new value in the box. The new parameter is sent to **PIDMonitor** when the operator is selecting the **Store** box by pressing one mouse button inside it. If the new performance of the regulator is bad, we can restore the old parameters by selecting the **Old Param** box.

Store	Angle	Exit	Old Param	H =	Tr =
-------	-------	------	-----------	-----	------

CONTROLLER 1 <-> ROTATION

K =	Td =	Ti =	b =	N =
-----	------	------	-----	-----

CONTROLLER 2 <-> LOWER ARM

K =	Td =	Ti =	b =	N =
-----	------	------	-----	-----

CONTROLLER 3 <-> UPPER ARM

K =	Td =	Ti =	b =	N =
-----	------	------	-----	-----

CONTROLLER 4 <-> HAND

K =	Td =	Ti =	b =	N =
-----	------	------	-----	-----

CONTROLLER 5 <-> TOOL

K =	Td =	Ti =	b =	N =
-----	------	------	-----	-----

Figure 4.8 The basic menu

Angle	Exit
-------	------

theta1ref	theta1
r1 =	t1 =
theta2ref	theta2
r2 =	t2 =
theta3ref	theta3
r3 =	t3 =
theta4ref	theta4
r4 =	t4 =
theta5ref	theta5
r5 =	t5 =

Figure 4.9 The menu when Angle is selected. Angle and Exit boxes in same position on the screen as in the main menu

A selection of the Angle box will cause the system to show the menu in Figure 4.9. The reference angles and the measured angles are continuously written in the different boxes during operation. To return to the basic menu, select the Angle box.

### 4.3 The Outside World

The outside world is simulated in Simulation/Monitoring Computer and, as the name say, some monitoring is done here to.

#### Process Description

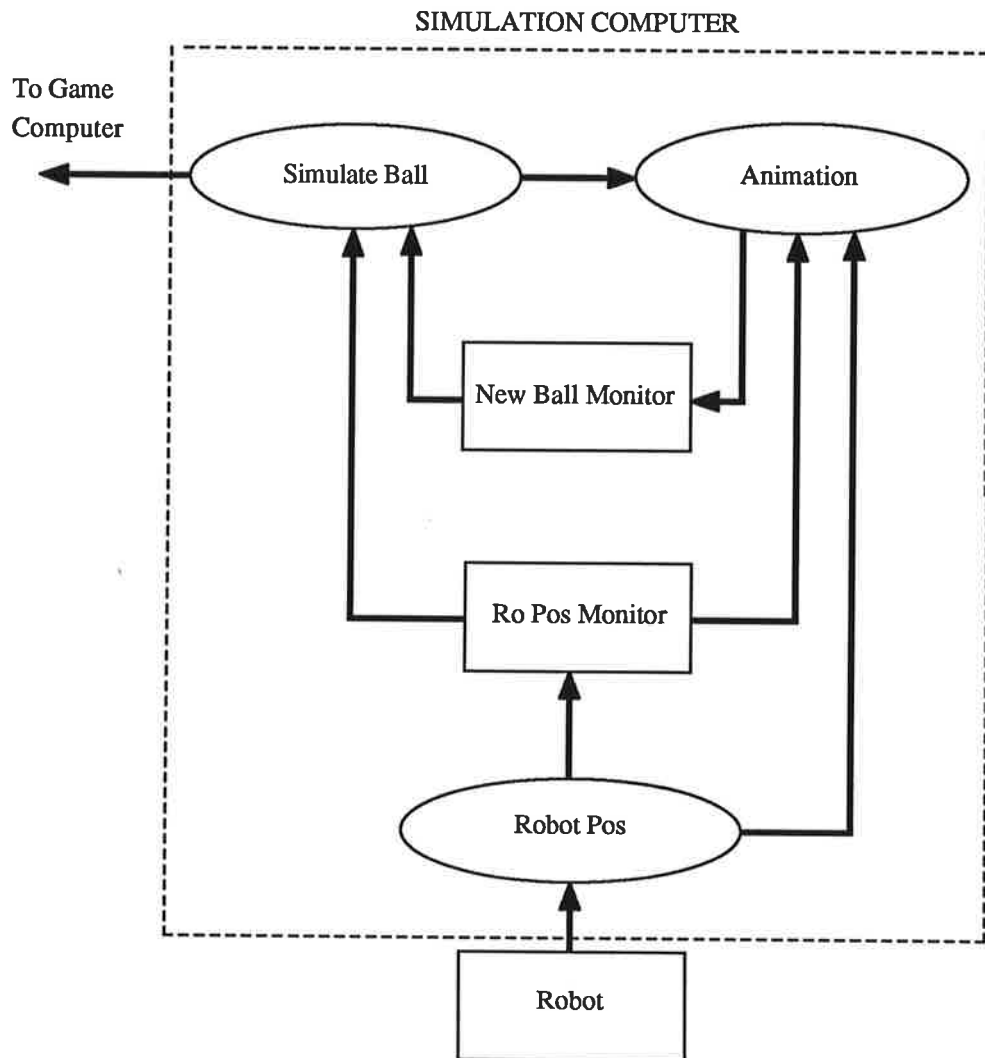


Figure 4.10 A description of how the processes in the Simulation/Monitoring Computer communicates with each other and with external units

The basic process is Animation. The table, the robot end-effector, and the ball is animated on the screen. It is possible for the operator to select different initial positions, different initial angles, and different initial velocities for the ball. It is also possible for the operator to have some basic parameters viewed. The basic parameters are the ball position, the ball velocity, the robot position, and the angle of the air blower. The robot position is fetched from RoPosMonitor, where the process RobotPos is storing it. RobotPos calculates the position in Cartesian space given the measured joint angles. This problem, the problem of kinematics, is treated in Chapter 5, where the formulas which are used is derived. RobotPos is also communicating with Animation directly,

when the commands of shooting is transferred. The fact that the robot is shooting can be seen on the screen as the robot turns white. The simulation of the ball is done in SimulateBall. The ball position is transferred to the Game Computer and to Animation for monitoring. To decide whether the robot really hits the ball SimulateBall needs information about the robot position and if it is shooting. This information can be fetched in RoPosMonitor. When the operator wants to shoot another ball against the robot the process Animation stores this information in NewBallMonitor, where SimulateBall can get it.

### The communication with the operator

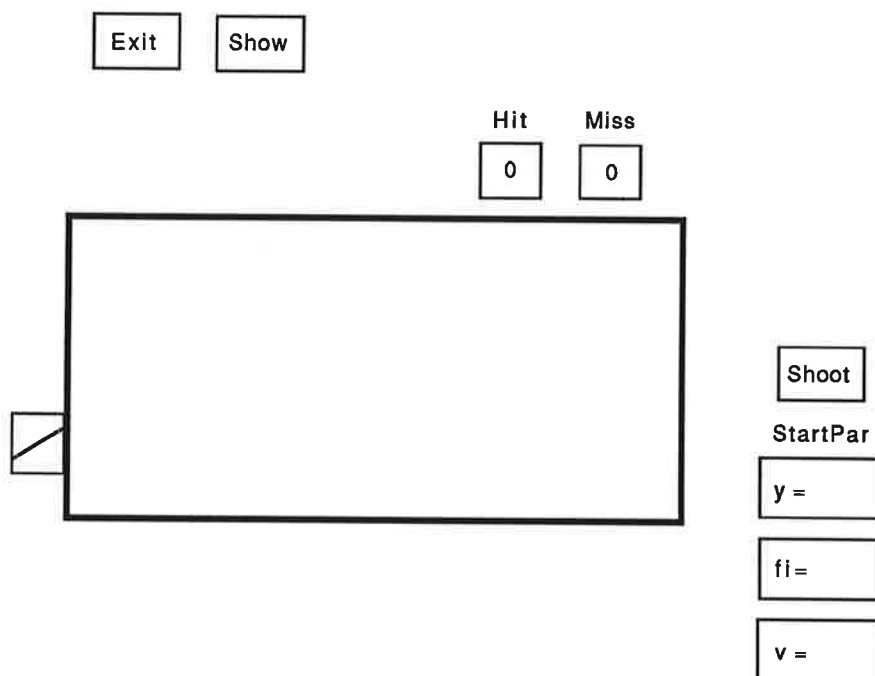


Figure 4.11 The basic menu

The menu written on the screen as the system is started can be seen in Figure 4.11. In the upper left corner two boxes, **Exit** and **Show**, can be seen. When the operator presses one mouse button when the cursor is inside the **Exit** box the system will be shut down. If the operator selects the **Show** box, the ball position, the ball velocity, the robot position, and the angle of the air blower, will be viewed. The screen when **Show** is selected is shown in Figure 4.12. A second selection of **Show** will make the system hide the viewed parameters and go back to the normal mode.

In the centre of the screen the table can be seen. A brown rectangle with a red line on it is at the table's left short side. This rectangle describe the robot hand. The red line is the air blower. The angle of the air blower can be seen as the angle of the red line relative the normal to the short side of table. A coordinate system is associated with the table. The lower left corner has the coordinate  $(0,0)$  and the upper right has the coordinate  $(1,0.7)$ . When the operator has changed the initial position,  $y$ , the initial angle,  $\varphi$ , and the initial velocity,  $v$ , by pressing one mouse button when the cursor is inside one

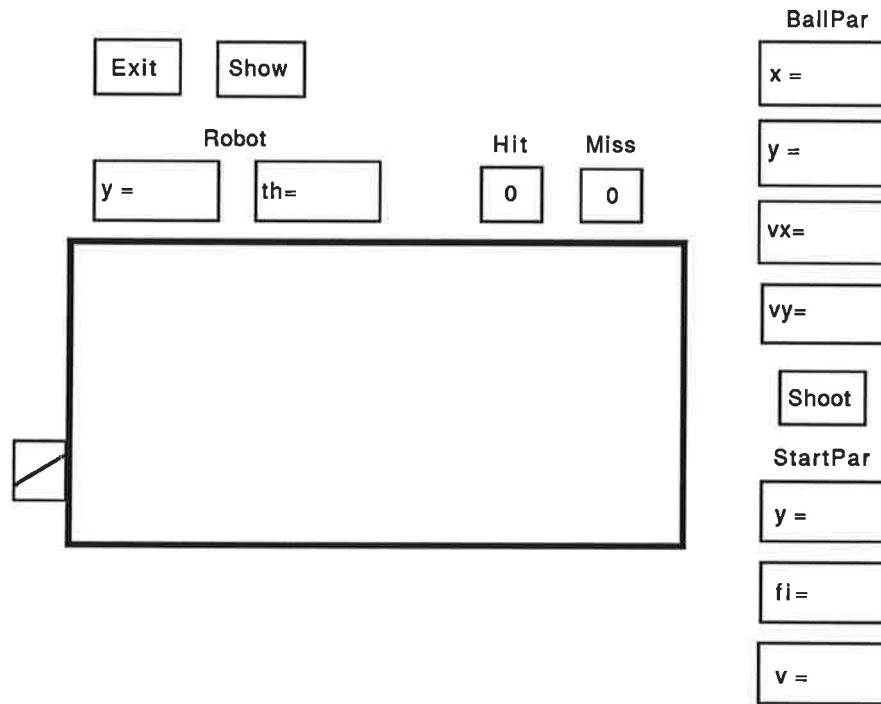


Figure 4.12 The menu when Show is selected

of the three boxes marked by  $y$ ,  $fi$ , or  $v$ , and the correct value has been written, the operator can shoot a ball with the specified initial position, initial angle, and initial velocity, by pressing a mouse button when the cursor is inside the Shoot box. Above the upper right corner of the table two boxes can be seen, Hit and Miss. In the Hit box the number of hits the robot has done is shown. The number of misses can be seen in the Miss box.

### The simulation model

In Figure 4.13 we can see the robot tool, centre in point  $A$ , and the ball, centre in point  $E$ , as the robot is shooting. The angle of the air blower is  $\theta$ . We define the distance between the ball centre and the robot centre to be  $\eta$  in the  $y$ -direction and  $\xi$  in the  $x$ -direction. The radius of the ball is  $\rho$ . In the following we will calculate  $\bar{v}_{out}$ , that is, the velocity of the ball when it leaves the short side. We assume that when the air blower blows, the ball gets a velocity component in the direction against the centre of the ball. We see from Figure 4.13 that if the angle of the air blower is  $\theta$  the ball will get a velocity component in the direction  $\varphi$ . Thus the angle relative the  $x$ -axis will be  $\psi = \theta + \varphi$ . We assume that the ball will get a velocity of the magnitude  $k \cos \varphi$ , where  $k$  is constant, from the air blower. From Figure 4.13 we can write down the following two equations

$$\zeta = \xi \tan \theta$$

$$\lambda = \eta - \zeta = \eta - \xi \tan \theta$$

The two triangles  $ACD$  and  $EBC$  have one right angle and one angle in common and hence their third angle must be the same. The conclusion is that the angle  $BEC$  is  $\theta$ . The distance  $R$  can now be calculated

$$R = \lambda \cos \theta = \eta \cos \theta - \xi \sin \theta$$



The point in which the ball will hit the opponent short side is

$$y = y_b + x_b \frac{k\sqrt{1 - \left(\frac{R}{\rho}\right)^2} \sin\left(\theta + \arcsin \frac{R}{\rho}\right) + v_{yin}}{k\sqrt{1 - \left(\frac{R}{\rho}\right)^2} \cos\left(\theta + \arcsin \frac{R}{\rho}\right) + v_{xin}} \quad (4.1)$$

Let the robot  $y$ -coordinate be  $r$ . Then it follows that

$$\eta = y_b - r$$

and

$$R = (y_b - r) \cos \theta - \xi \sin \theta$$

We assume that  $\xi$  is constant, i.e. the distance, in the  $x$ -direction, between the robot hand and the ball, at the moment of shooting, does not change much. We can now apprehend  $y$  as a function of the ball velocity  $v_{xin}$ ,  $v_{yin}$ , the robot position  $r$ , the ball position  $y_b$ , and the angle of the air blower  $\theta$ . We will return to this function in Chapter 6, except that the velocity will be written in polar coordinates.

## 5. Robot Control

The problem of how to control a robot will be treated in this chapter. In the first section we will derive the equations of kinematics for the robot. This is done in a straightforward manner by projecting links on coordinate axis. We will use the notation  $\mathcal{P}_{\bar{e}}^{\lambda}$ , meaning the projection of  $\lambda$  on the vector  $\bar{e}$ . In our calculations  $\lambda$  will be the robot links and  $\bar{e}$  will be unity vectors in a Cartesian coordinate system i.e.  $\bar{e}_x$ ,  $\bar{e}_y$ , and  $\bar{e}_z$ . In robot control we are more interested in the equations of inverse kinematics. We will derive these equations in Section 5.2. For a detailed discussion about robotics see (Craig, 1986). A brief discussion of PID-regulators will be done in Section 5.3.

### 5.1 Kinematics

If the link lengths and the joint angles are known, we can calculate the position of the tool in a Cartesian coordinate system, with the origin at the robot base, simply by using a "bottom-up" approach. We start from the robot base and work ourselves up. The method will be illustrated in the following where we will calculate a point  $(x, y, z)$  as a function of the link lengths  $\lambda$  and the joint angles  $\theta$ . It is custom to have a link at the robot base with length 0. We will also use this convention here.

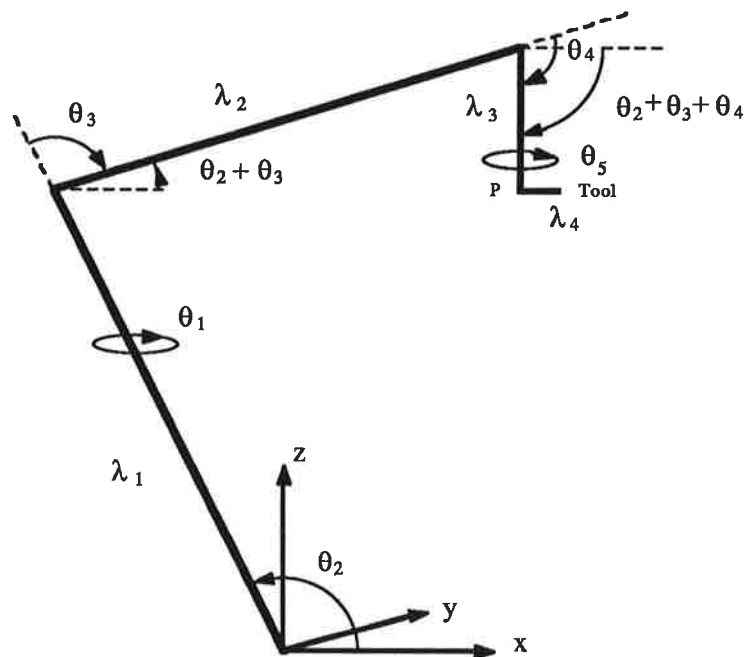


Figure 5.1 The configuration with five links and five joint angles

Let us calculate the coordinates of point  $P$ , see Figure 5.1. If we project the second link on the  $x$ -axis we obtain

$$\mathcal{P}_{\bar{e}_x}^{\lambda_1} = \lambda_1 \cos \theta_2 \cos \theta_1 \bar{e}_x$$



the projection of the third link on the  $x$ -axis will be

$$\mathcal{P}_{\bar{e}_x}^{\lambda_2} = \lambda_2 \cos(\theta_2 + \theta_3) \cos \theta_1 \bar{e}_x$$

and finally the fourth link projection will be

$$\mathcal{P}_{\bar{e}_x}^{\lambda_3} = \lambda_3 \cos(\theta_2 + \theta_3 + \theta_4) \cos \theta_1 \bar{e}_x$$

To obtain the points  $x$ -coordinate we only have to sum the three links projective on the  $x$ -axis. If we instead project on the  $y$ -axis we obtain

$$\mathcal{P}_{\bar{e}_y}^{\lambda_1} = \lambda_1 \cos \theta_2 \sin \theta_1 \bar{e}_y$$

$$\mathcal{P}_{\bar{e}_y}^{\lambda_2} = \lambda_2 \cos(\theta_2 + \theta_3) \sin \theta_1 \bar{e}_y$$

$$\mathcal{P}_{\bar{e}_y}^{\lambda_3} = \lambda_3 \cos(\theta_2 + \theta_3 + \theta_4) \sin \theta_1 \bar{e}_y$$

In the same way we obtain the  $z$ -coordinate by projecting on the  $z$ -axis.

$$\mathcal{P}_{\bar{e}_z}^{\lambda_1} = \lambda_1 \sin \theta_2 \bar{e}_z$$

$$\mathcal{P}_{\bar{e}_z}^{\lambda_2} = \lambda_2 \sin(\theta_2 + \theta_3) \bar{e}_z$$

$$\mathcal{P}_{\bar{e}_z}^{\lambda_3} = \lambda_3 \sin(\theta_2 + \theta_3 + \theta_4) \bar{e}_z$$

Finally we can calculate the point  $P$ 's coordinate as

$$\begin{cases} x_p = (\lambda_1 \cos \theta_2 + \lambda_2 \cos(\theta_2 + \theta_3) + \lambda_3 \cos(\theta_2 + \theta_3 + \theta_4)) \cos \theta_1 \\ y_p = (\lambda_1 \cos \theta_2 + \lambda_2 \cos(\theta_2 + \theta_3) + \lambda_3 \cos(\theta_2 + \theta_3 + \theta_4)) \sin \theta_1 \\ z_p = \lambda_1 \sin \theta_2 + \lambda_2 \sin(\theta_2 + \theta_3) + \lambda_3 \sin(\theta_2 + \theta_3 + \theta_4) \end{cases} \quad (5.1)$$

The fifth link projective on the  $x$ -, and  $y$ -axis will be  $\lambda_4 \cos(\theta_1 + \theta_5)$  and  $\lambda_4 \sin(\theta_1 + \theta_5)$  respectively. To obtain the tool coordinates we have to add  $\lambda_4 \cos(\theta_1 + \theta_5)$  to  $P$ 's  $x$ -coordinate and  $\lambda_4 \sin(\theta_1 + \theta_5)$  to the  $y$ -coordinate. If  $P$ 's coordinates are  $(x_p, y_p, z_p)$ , the tool coordinates  $(x_t, y_t, z_t)$  will be

$$\begin{cases} x_t = x_p + \lambda_4 \cos(\theta_1 + \theta_5) \\ y_t = y_p + \lambda_4 \sin(\theta_1 + \theta_5) \\ z_t = z_p \end{cases} \quad (5.2)$$

## 5.2 Inverse Kinematics

In this section we will treat the inverse kinematic problem, that is, given a point  $(x, y, z)$  in a Cartesian coordinate system with the origin at the robot base and with axis as in Figure 5.1, find the joint angles such that the tool is positioned in  $(x, y, z)$ . This problem can be solved by solving the equations of kinematics (5.1), (5.2) in the previous section. We will use a approach here, basically built on the theorem of cosine. We can make things easier if we observe the fact that the fourth link,  $\lambda_3$ , always will be parallel with the  $z$ -axis. Thus, we can consider the point,  $Q$ , where the third and fourth link intersect. We define  $r$  as the distant from the origin to  $Q$

$$r = \sqrt{x^2 + y^2 + z^2}$$

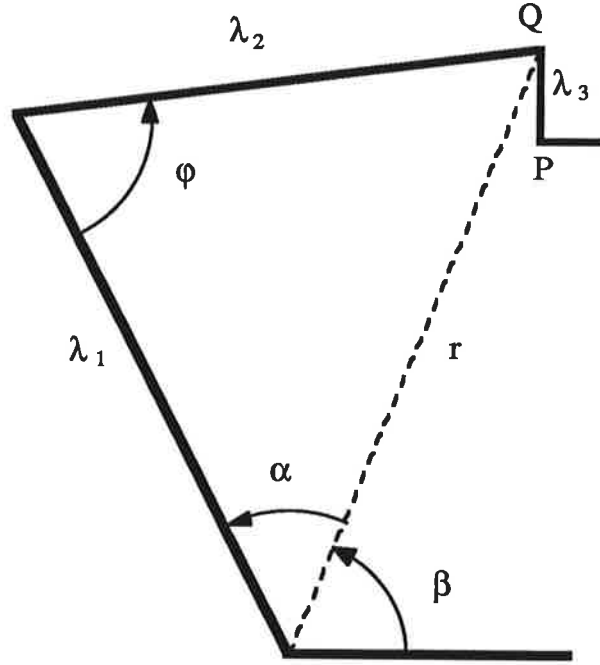


Figure 5.2 Robot with auxiliary angles

We can calculate the joint angle  $\theta_1$ , see Figure 5.1, by projecting the point  $(x, y, z)$  to the  $xy$ -plane i.e.  $(x, y, 0)$ . More precisely we calculate  $\theta_1$  as

$$\theta_1 = \arctan \frac{y}{x}$$

The angle  $\beta$  in Figure 5.2 is calculated as

$$\beta = \arctan \frac{z}{\sqrt{x^2 + y^2}}$$

If we apply the theorem of cosine to the angle  $\alpha$  we obtain

$$\cos \alpha = \frac{\lambda_1^2 + r^2 - \lambda_2^2}{2\lambda_1 r}$$

We see that the joint angle  $\theta_2$  is simply the sum of  $\alpha$  and  $\beta$  i.e.

$$\theta_2 = \alpha + \beta = \arctan \frac{z}{\sqrt{x^2 + y^2}} + \arccos \frac{\lambda_1^2 + r^2 - \lambda_2^2}{2\lambda_1 r}$$

To calculate  $\theta_3$  we apply the theorem of cosine to the angle  $\varphi$  in Figure 5.2

$$\cos \varphi = \frac{\lambda_1^2 + \lambda_2^2 - r^2}{2\lambda_1 \lambda_2}$$

From Figure 5.1 and 5.2 we see that  $\varphi - \theta_3 = \pi$  and hence

$$\theta_3 = \varphi - \pi = \arccos \frac{\lambda_1^2 + \lambda_2^2 - r^2}{2\lambda_1 \lambda_2} - \pi$$

We can calculate  $\theta_4$ , if we observe the fact that the three angles  $\theta_2$ ,  $\theta_3$ , and  $\theta_4$  lies in one plane. In the way we have defined the joint angles we can simply add them and obtain the final direction of the fourth link relative the positive  $x$ -axis. If we want the direction of link 3 to be  $-\pi/2$  we obtain

$$\theta_2 + \theta_3 + \theta_4 = -\pi/2 \Leftrightarrow \theta_4 = -\pi/2 - \theta_3 - \theta_2$$

The angle  $\theta_5$  can be changed independently of the other angles. If we position the robot so that the point  $P$ 's coordinate will be  $(x, y, z)$ , we can position the tool to any point which lie on a circle with origin in  $(x, y, z)$  and with radius  $\lambda_4$ , by changing  $\theta_5$ .

### 5.3 PID-Regulators

Each joint angle was controlled by a PID-regulator implemented in a computer. A continuous time PID-regulator can be written as

$$u(t) = K \left( e(t) + T_d \frac{de}{dt} + \frac{1}{T_i} \int_{-\infty}^t e(\tau) d\tau \right)$$

where  $e(t)$  is the difference between the reference signal,  $r(t)$ , and the measured signal,  $y(t)$

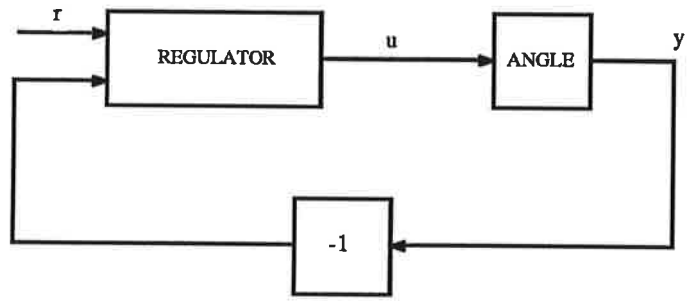
$$e(t) = r(t) - y(t)$$

We see that the control signal,  $u(t)$ , is a sum of three terms. The first term is simply proportional to the error, the second is proportional to the derivative of the error, and the third is proportional to the integral of the error. An intuitive way of understanding this regulator is to reason as follows : The derivative can be associated by a prediction of the error in future. The integral can be seen as bearer of information about the past. Thus the control signal is composed of information about the past, the present, and the future. To implement this regulator in a computer, we have to modify the structure a bit. A first modification is to substitute the continuous time regulator by a discrete time one. However, this is not enough. To obtain a well functional regulator some further changes will be needed. How these modifications should be done is discussed in (Åström, 1987). We will just summarize the equations we have used.

$$u(t_k) = P(t_k) + I(t_k) + D(t_k)$$

where

$$\begin{aligned} P(t_k) &= K (b r(t_k) - y(t_k)) \\ I(t_{k+1}) &= I(t_k) + \frac{K h}{T_i} e(t_k) + \frac{h}{T_r} (u(t_k) - v(t_k)) \\ D(t_k) &= \frac{T_d}{T_d + N h} D(t_{k-1}) - \frac{K T_d N}{T_d + N h} (y(t_k) - y(t_{k-1})) \end{aligned}$$



**Figure 5.3** Block diagram of the control system

## 6. Principle of Learning

In this chapter we will consider the problem of learning and training basic shots. The purpose of learning the robot is thus to make it learn how to change the angle of the air blower, so that when the robot is shooting at the ball, the ball will hit the opponent short side at a given position.

### 6.1 Algorithm of Learning

In Figure 6.1 we can see the table, the ball, and the robot hand. Consider the situation when the ball arrives at the robot short side. We define the angle of the air blower to  $\theta$ , the angle of the ball to  $\varphi$ , the ball velocity to  $v$ , the distance from the lower border to the ball to  $y_b$ , and the distance from the lower border to the robot hand to  $r$ . We define the position where the ball hits the opponent short side to be  $y$ . Introduce  $\eta = y_b - r$ .

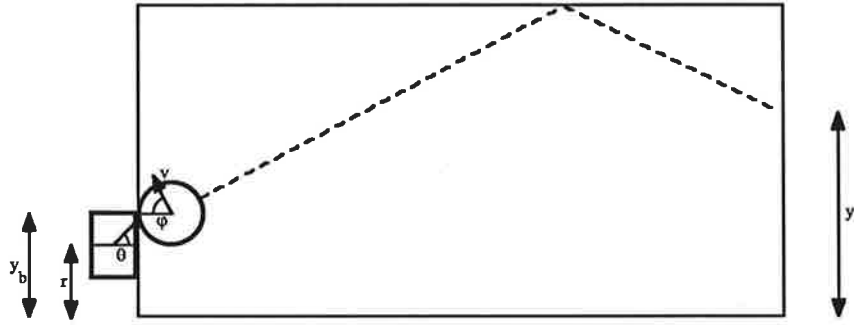


Figure 6.1 The table with the robot and the ball

We grasp  $y$  as a scalar valued function of five variables i.e.

$$y = y(\theta, \varphi, v, \eta, y_b)$$

Assume that we have an estimation function

$$\hat{y}(\theta, \varphi, v, \eta, y_b) = \bar{\alpha}^T \bar{\Phi}(\theta, \varphi, v, \eta, y_b)$$

where

$$\bar{\alpha} = \begin{pmatrix} \alpha_0 \\ \vdots \\ \alpha_n \end{pmatrix}$$

is a vector with  $n + 1$  unknown components, and

$$\bar{\Phi}(\theta, \varphi, v, \eta, y_b) = \begin{pmatrix} \Phi_0(\theta, \varphi, v, \eta, y_b) \\ \vdots \\ \Phi_n(\theta, \varphi, v, \eta, y_b) \end{pmatrix}$$

is a  $n + 1$  dimensional vector valued function of five variables, i.e.

$$\bar{\Phi}: \mathbb{R}^5 \rightarrow \mathbb{R}^{n+1}$$

When we are training the robot, we shoot balls with different initial positions, different initial angles and different initial velocities. Thus the ball will arrive at the robot short side with different velocities, in different positions and with different angles, depending on the initial conditions. By varying the initial conditions, we can influence  $\varphi$ ,  $v$ , and  $y_b$ . The robot position,  $r$ , is also affected because the robot hand must be positioned where the ball arrives. The angle of the air blower,  $\theta$ , is chosen at random. Hence, by training the robot  $N$  times we observe

$$\begin{aligned} &\theta_1, \varphi_1, v_1, \eta_1, y_{b1}, y_1 \\ &\quad \vdots \\ &\theta_N, \varphi_N, v_N, \eta_N, y_{bN}, y_N \end{aligned}$$

where

$$y_k = y(\theta_k, \varphi_k, v_k, \eta_k, y_{bk}), \quad k = 1, \dots, N$$

The idea is to approximate the rather complicated function  $y(\theta, \varphi, v, \eta, y_b)$ , see equation (4.1), with the simpler  $\hat{y}(\theta, \varphi, v, \eta, y_b)$ . Let the notations  $\hat{y}_k$  and  $\bar{\Phi}_k$  mean

$$\begin{aligned} \hat{y}_k &= \hat{y}(\theta_k, \varphi_k, v_k, \eta_k, y_{bk}) \\ \bar{\Phi}_k &= \bar{\Phi}(\theta_k, \varphi_k, v_k, \eta_k, y_{bk}) \end{aligned}$$

We can now state the problem formulation, (Tsyppin, 1971). Find the vector  $\bar{\alpha}$  such that the loss function

$$J(\bar{\alpha}) = \frac{1}{2} \sum_{k=1}^N (y_k - \hat{y}_k)^2$$

is minimized. This is the well known least square method. In the least square terminology we have the regression vector  $\bar{\Phi}$ , and the parameter vector  $\bar{\alpha}$ . It is possible to formulate the solution to this problem in a recursive manner (Söderström, 1984; Åström and Wittenmark, 1984). The solution is

$$\begin{aligned} \varepsilon_k &= y_k - \hat{y}_k = y_k - \bar{\Phi}_k^T \bar{\alpha}_{k-1} \\ \bar{K}_k &= \frac{P_{k-1} \bar{\Phi}_k}{1 + \bar{\Phi}_k^T P_{k-1} \bar{\Phi}_k} \\ \bar{\alpha}_k &= \bar{\alpha}_{k-1} + \bar{K}_k \varepsilon_k \\ P_k &= P_{k-1} - \frac{P_{k-1} \bar{\Phi}_k \bar{\Phi}_k^T P_{k-1}}{1 + \bar{\Phi}_k^T P_{k-1} \bar{\Phi}_k} \end{aligned}$$

$\varepsilon_k$  is a scalar saying how much the real position  $y_k$  differs from the estimated one  $\hat{y}_k$ . The components in the vector  $\bar{K}_k$  can be interpreted as weights saying how much of the error,  $\varepsilon_k$ , should affect the new estimate of the components in the vector  $\bar{\alpha}_k$ . An interpretation of the matrix  $P_k$  is, except for a factor, the covariance of  $\bar{\alpha}_k$ .

We will now have to choose the regression vector,  $\bar{\Phi}$ . Assuming that we have none, or very little, a priori information about  $y$ , we choose the regressors as

$$\bar{\Phi}(\theta, \varphi, v, \eta, y_b) = \left( 1 \quad \theta \quad \dots \quad y_b \quad \theta\varphi \quad \dots \quad \eta y_b \quad \theta^2 \quad \dots \quad y_b^2 \right)^T$$

$\bar{\alpha}^T \bar{\Phi}$  will now be a polynomial of degree two in five variables with unknown coefficients. The choice of regressors above is not as innocent as it may seem. We try to estimate a function with a second degree polynomial. If we, for simplicity, consider the one dimensional case, we try to estimate a function with a parabola. If we only was interesting in good estimation near a point, we could, according to Taylors formula, hope to get reasonable results. However, our intention is to find a parabola that fits well with the unknown function in an interval. In the worst case, where the unknown function vary much, the result will be bad.

## 6.2 One typical experiment

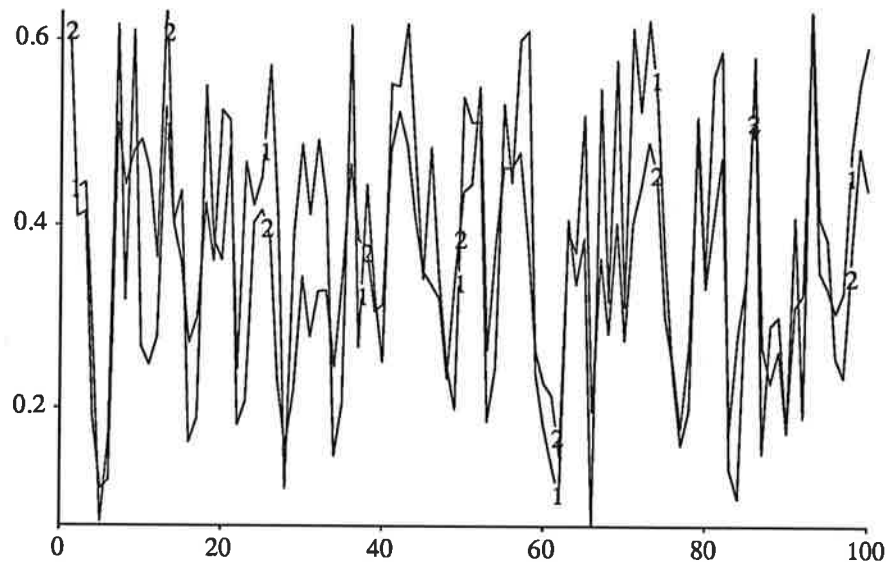


Figure 6.2 A plot over  $y$ , 1, and  $\hat{y}$ , 2.

The results of a typical experiment of supervised learning is presented in Figures 6.2 - 6.6. The training session consisted of 250 shots. The point in which the ball hits the opponent short side,  $y$ , and our estimation of  $y$ ,  $\hat{y}$ , are plotted in Figure 6.2. We have selected the last 100 shots. Some of the parameters are plotted in Figure 6.3 - 6.6. These plots have been chosen so that different behaviour are illustrated. Figure 6.3 shows a parameter with relative fast rate of convergence. A parameter with slow rate of convergence is plotted in Figure 6.6. In these plots we have omitted the transient behaviour, that is, the first 50 shots. The reason for this is that the transients were large and caused large scales on the plots. From the figures we can draw several conclusions. First, the accordance between  $y$  and  $\hat{y}$  is acceptable, see Figure 6.2. Secondly, although we did a training session of 250 shots with the ball, some of the parameters have not yet converged as indicated in the figures. We know, from equation (4.1), that  $y$  is proportional to  $y_b$ . Thus all but one of

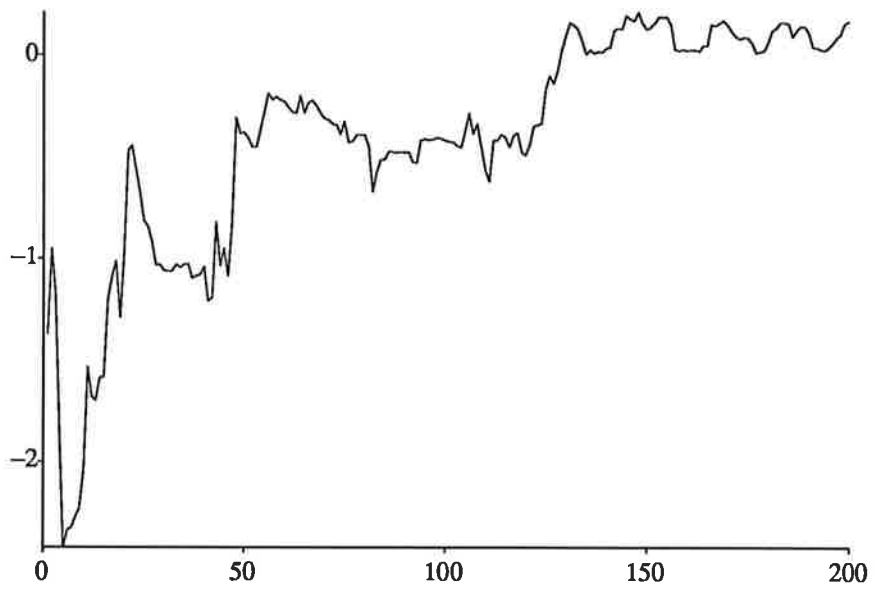


Figure 6.3 A plot over the  $v$  coefficient

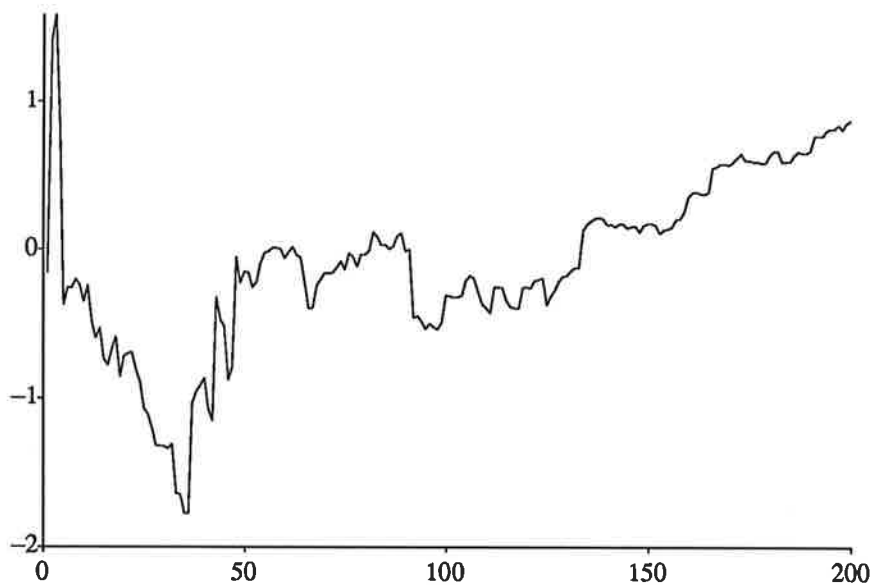


Figure 6.4 A plot over the  $y_b$  coefficient

the different terms in the estimation function containing  $y_b$  should be zero. We can use this fact to investigate the reasonableness of the parameter values. Two plots containing  $y_b$  are shown, Figure 6.4 and 6.5. The curve in Figure 6.4 tends to 1 and the curve in Figure 6.5 tends to 0. In fact, those of the plots omitted containing  $y_b$  also tends to 0. Figure 6.3 shows that the  $v$  coefficient tends to 0.



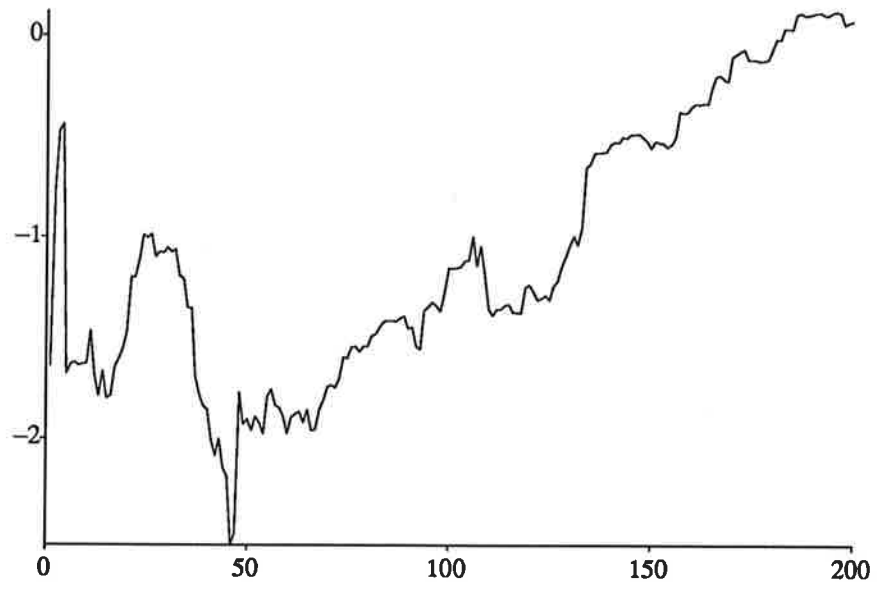


Figure 6.5 A plot over the  $\varphi y_b$  coefficient

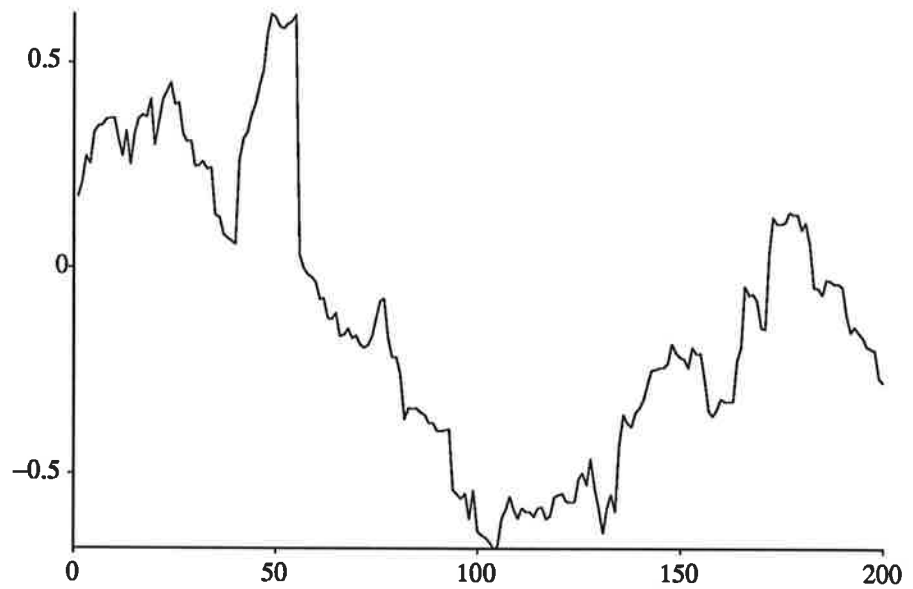


Figure 6.6 A plot over the  $\eta^2$  coefficient