National Library of Canada
Acquisitions and Bibliographic Services Branch
395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale du Canada
Direction des acquisitions et des services bibliographiques
395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Canada

A NEW LEARNING METHOD UTILIZING A ZONE APPROACH FOR PERFORMANCE ENHANCEMENT OF JOINT CONTROLLED MANIPULATORS

by

Gilles P. Doucet

Thesis submitted to the School of Graduate Studies

in partial fulfilment of the requirements for the degree of

Master of Applied Science

in

Mechanical Engineering

Ottawa-Carleton Institute for

Mechanical and Aerospace Engineering

University of Ottawa

ISBN 0-315-82575-8

Canada

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

ABSTRACT

The conventional design strategy of robot manipulator fixed gain joint controllers is to critically damp the system for the most extreme configuration condition to prevent overshoot. This results in a highly overdamped system for all other configuration conditions which leads to slow response and poor high speed performance. A "zone learning method" has been developed which entails partitioning the robot work volume into zones based on the positions of the links. The link velocity feedback gain is then modified by multiplier values associated with each zone. The values of the multipliers are determined by a learning process based on the performance criteria of fastest possible response without overshoot. This method can also be implemented using link velocity zones. The method results in significant improvement in the performance of individual joint controlled manipulators and can easily and inexpensively be added to the hardware of existing robots using similar controllers.

IN APPRECIATION

My sincere appreciation is offered to my wife and children who have persevered along with me through this very busy period and whose understanding and sacrifice was essential to the completion of this work.

A large debt is owed to my thesis supervisor Dr. Atef Fahim for his ideas, guidance, friendship and dedication to this project. I am indebted to my employer the Department of National Defence for the generous allocation of education leave granted for this research work. I am especially grateful to my immediate superiors Mr. Terry Rollins and Dr. Al Robitaille for their unwavering support and to my DND colleagues as well for their encouragement and friendship.

My colleagues in B-203 were a constant source of inspiration and support. I would like to thank them for making my stay at the university a very rewarding experience. I am particularly thankful for the assistance and friendship of Hyun Choi, Sam Tanary and Rudy Tolkamp and I wish them great success. Numerous others have contributed in different ways. I am grateful for the blessings of brothers, sisters and dear friends.

This thesis is dedicated to my parents, Charlotte and Rodolphe, who gave me the ambition to endeavour and to my children, Stephanie and Andre, who give me the strength to persevere.

TABLE OF CONTENTS

List of Figures

List of Tables

NOMENCLATURE

Robot Model

m_1 = total mass of link 1

m_{21} = mass of link 2 component 1

m_{22} = mass of link 2 component 2

m_3 = mass of link 3

l_{12} = length of link 1

l_{21} = length of link 2 component 1

l_{22} = length of link 2 component 2

l_3 = length of link 3

r_{21} = radius of link 2 component 1

r_{22} = radius of link 2 component 2

r_3 = radius of link 3

K_i = kinetic energy of link i

U_i = potential energy of link i

q_i = generalized coordinate for link i (angle or distance)

Q_i = generalized driving force for link i (torque or force)

I_{ij} = inertia matrix (link and actuator)

C_{ijk} = coefficients of the centrifugal and coriolis forces

g_i = gravity torque acting on link i

B_i = viscous damping coefficient for link i

T_{f_i} = friction disturbance force for link i

T_{v_i} = disturbance force acting on link i due to multiple link motion

$u(t)$ = control force input

$\theta_d(t)$ = desired link position at time (t)

$\theta(t)$ = actual link position at time (t)

F_m = motor back e.m.f. constant

B_m = motor viscous damping coefficient

K_p = link controller proportional gain

K_v = link controller velocity feedback gain

T_f = disturbance torque due to friction

T_d = viscous damping disturbance torque

T_v = disturbance torque due to motion of multiple links

J_m = motor rotor mass moment of inertia

J_l = link mass moment of inertia

K_m = motor constant

V_m = voltage applied to the motor

N = gear ratio

ω_n = system natural frequency

ξ = damping ratio

Zone Learning Method

$x_j y_j z_j$ = the cartesian coordinates of the actual end effector position (point j)

x_j^d, y_j^d, z_j^d = the cartesian coordinates of the desired end effector position (point j)

$\cap$ = set "union" operator

$\bigcap$ = logical "and" operator

$\cup$ = set "intersection" operator

$\bigcup$ = logical "or" operator

t_j = time at trajectory reference point j

λ_i = feedback velocity gain multiplier for configuration zone i

Λ_i = feedback velocity gain limit for configuration zone i

γ_l = feedback velocity gain multiplier for velocity zone l

Γ_l = feedback velocity gain limit for velocity zone l

β = trajectory position error vector

CHAPTER 1

INTRODUCTION

The field of robotics motion control has seen many advances in recent years as researchers move away from the classical approaches to more advanced approaches that are more complex but better performing. The most basic approach entails using individual joint fixed gain controllers designed with the assumption that each link of the manipulator is a linear time-invariant system. This represents an idealized model which ignores two important factors.

(1) The individual links cannot be accurately represented by a time-invariant model since the mass moment of inertia, measured at a link's axis of rotation, usually depends on the positions of all the successive links (i.e. manipulator configuration). Therefore the manipulator configuration will affect the load demand on a given link.

(2) The link's own motion, and those of the other links, induce centrifugal and coriolis forces which act as disturbances on a given link.

The conventional solution to this difficulty is to design the controller to be critically damped for the most extreme configuration condition to prevent overshoot, and then manually fine tune the controller parameters. This results in a highly over-damped, slow system but which has good position accuracy at slow speeds. However, this type of control scheme generally

results in poor trajectory following capabilities and poor position accuracy at high speeds.

Many new approaches have been introduced to improve manipulator performance. These include: inverse dynamics methods, centralized or hierarchical control, adaptive techniques and learning methods. Some of these will be briefly discussed in Chapter 2 (Literature Review). All the new methods possess certain advantages over the classical approach. However, some of the new methods introduce a much higher degree of complexity to the controller. Many of these techniques cannot easily be directly applied to existing industrial robots. The result is that there are currently very few robots in industrial use which use any of the above methods. There is, therefore, still a need to improve the performance of the conventionally controlled fixed gain manipulators with inexpensive "add-on" software or hardware modifications.

This thesis presents a new learning method which improves the trajectory position accuracy of robot manipulators equipped with individual joint controllers. The method was developed for a robot which uses proportional control with velocity feedback for the control of individual links. One of the major criteria driving the study was the need to improve the performance of robots currently used in industrial applications, with no hardware modifications and only minor software modifications. This would allow for an improvement in robot performance at very low cost.

The proposed learning method improves the manipulator's position and tracking accuracy by adapting the feedback velocity gain of the individual joint controllers to the robot time varying

state parameters. The modification of the feedback velocity gain is accomplished by the use of multipliers which are stored in a data base and referenced according to the manipulator link positions and velocities.

The learning method, called the "zone learning method", is implemented in two parts. The first part of the implementation consists of partitioning the robot workspace into zones based on the positions of the links ("configuration zones"). Each of these configuration zones is assigned a variable ("multiplier") which will be used to modify the velocity gain value whenever the manipulator configuration coincides with the configuration zone. A series of learning trials are then used to determine the optimum multiplier values within each configuration zone. This learning process is based on the performance criteria of fastest possible response without overshoot.

The second part of the implementation aims to reduce the effects of link motion on the position errors of the manipulator. "Configuration-velocity zones" are defined as combinations of link velocity ranges within each configuration zone. As in the case for the configuration zones, multipliers are assigned to these zones. Once the optimum values for the configuration zones have been established, a learning process is executed to determine the optimum values of the multipliers within the configuration-velocity zones. The controller derivative gain is then adapted to the specific configuration and velocity conditions by the combined action of both the configuration and the configuration-velocity multipliers.

The zone leaning method is simple in both concept and application. It results in significant improvement in the performance of individual joint controlled manipulators and can easily and inexpensively be added to the hardware of existing robots which use individual link proportional controllers with velocity feedback. The implementation of the method is flexible in that after a "training" period, the learning system may be disabled and the zone multiplier data base will remain constant from that point on. However, the learning process may continue "on-line" during the execution of industrial tasks and continue to improve the performance until a plateau is reached. After having been disabled, the learning system can later be reinitiated in response to changing conditions (e.g. environmental conditions or manipulator aging). In this case the resulting multiplier data values will reflect those changed conditions and result in a manipulator better suited to those new conditions.

Chapter 2 of this thesis presents a brief overview of the directions which are currently being pursued in the field of learning applied to robot motion control. Chapter 3 outlines the classical control system design upon which the proposed learning method is overlaid. A detailed description of the proposed zone learning method and the structure of the required data bases is presented in Chapter 4. Chapter 5 describes the performance indices that will be used to assess the performance of the proposed learning method. Results of computer simulations which show the performance of the learning method when applied to a test manipulator are presented in Chapter 6. A conclusion highlighting the learning method, its performance, as well as future exploration and developments constitutes Chapter 7. Mathematical details of methods used in the robot simulations are contained in four appendices at the end of the main body of the thesis.

CHAPTER 2

LITERATURE REVIEW

The "zone learning" method proposed in this thesis is an innovative technique aimed at improving the performance of robot manipulators equipped with fixed gain individual link proportional controllers with velocity feedback. Since this is not a variation of an existing method, there is no literature dealing explicitly with this structure for robot learning. However, there has been considerable work in the field of learning as a means to improve the motion control of robot manipulators. This section briefly explores some of the major thrusts in this area and references some important contributions.

2.1 Iterative Learning Control

Iterative learning consists of learning by repetition. When the input torque is a periodic function (repeating identical trajectories) algorithms can be developed such that the errors of previous trials are used to improve the results of future trials until eventually convergence of the desired and actual trajectories occur. Reference [1] presents a good basic introduction to the principle of feedforward torque learning and proposes several learning functions (laws). There are two basic approaches, one which assumes no knowledge of the manipulator dynamics and one which uses a model of the manipulator.

References [2] [3] [4] [5] [6] and [7] all explore iterative learning without using a model and assume no knowledge of the manipulator dynamics. In this basic technique, the input of the present trial is modified by the error profile of the previous trial (modified by an appropriate "learning gain" factor). The error profile, which consists of the desired output minus the previous response, of an iteration is stored and used to modify the control input of the next iteration. This process is illustrated in Figure 2.1 [2]. [2] and [5] use the velocity error pattern, [3] uses the position error pattern, while [4] uses a combined learning function of both the position and velocity error profiles. The authors showed that this type of algorithm is convergent given certain reasonable conditions.

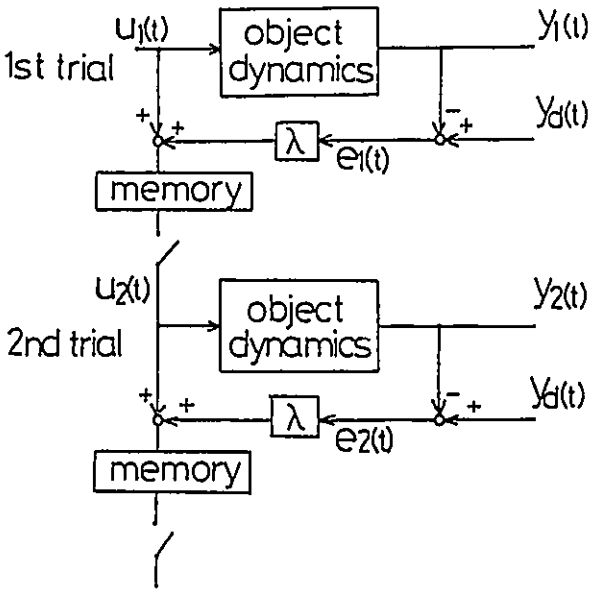


Figure 2.1: Learning Control

References [6] and [7] pursue the idea even further by presenting a method of transferring knowledge from one trajectory to another. According to [6] and [7], if a number of input patterns have already been iteratively learned for different desired outputs, these sets can be decomposed and recombined to fit a new trajectory without the need for iterative learning of the new trajectory. This method however, relies on linear approximations of manipulator dynamics (i.e. high ratio reduction gears, and high gain feedback). This has been shown to be effective given that the linearity approximations are realistic and has the merit of requiring little a priori

knowledge of the manipulator dynamics. However, the method presented in [6] and [7] of transferring knowledge from one trajectory to another can only be applied under the restrictive condition of linear manipulator dynamics. Also a large memory capacity may be required to store the results of many iterative trials in the case of [6] and [7].

2.2 Repetitive Control

Another approach which has been successfully explored is repetitive control. In repetitive robot tasks, the input torque profile required for the manipulator to describe a certain trajectory and the disturbance torques encountered, are periodic. From this periodicity, a controller can learn to compensate for the trajectory tracking errors of previous cycles.

This approach is closely related to iterative learning control with the exception that, while "the control input is only updated once each cycle in learning control, it is continuously adjusted in repetitive control" [8]. The repetitive control procedure generally consists of determining a feedforward torque profile, through repetition, which will minimize the trajectory errors. Numerous specific algorithms have been published, but only a few examples will be presented here. The repetitive controllers are normally used in conjunction with standard link controllers (PD or computed torque control laws). Reference [8] describes a basic digital implementation of repetitive control (Figure 2.2). Reference [9] demonstrates the feasibility of a "plug in" repetitive controller for an industrial robot. This raises the possibility of improving the performance of currently existing industrial robots by retrofit.

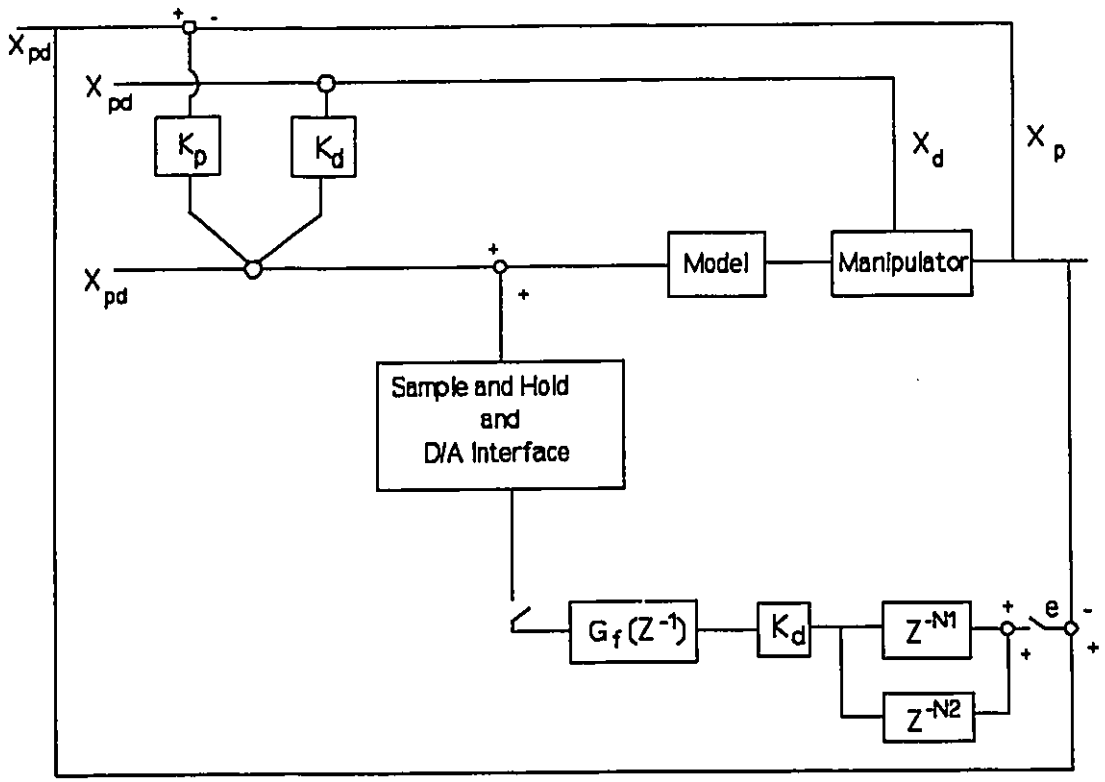


Figure 2.2: Repetitive Control Applied to a Computed Torque Controlled Robot

Reference [10] uses feedforward command torques, along with standard feedback link controllers, to drive the robot motion. The learning process, whose goal is to eliminate the feedforward command error, uses the full robot dynamic model. In the limit, when the learning is perfect, the feedforward torques will be solely responsible for driving the robot motion and the feedback controllers will provide no output. This is because there will be no trajectory error. This particular method has the advantage of working with many different types of controllers and plants. However, the exact path must be repeated in order to learn and an accurate model of the manipulator is required.

The requirement for an accurate model is dropped in the methods proposed in [11], [12]

and [13] which consist of learning the required feedforward torque profile for the desired trajectory by curve fitting a linear combination of appropriately selected periodic functions, such as the fourier series or a polynomial. The coefficients of these functions are updated from the trajectory errors using a parameter estimation algorithm. This method also reduces the amount of memory space required to store the learned feedforward profile as the profile is stored as a series of curves rather than as a set of numbers (as in many other methods).

2.3 Neural Network Learning

A neural network is a system of computational units that are interconnected in a structured manner. Each of the computational units, or processing elements, can accept many inputs but generally yield only one output. The layered structure of neural networks dictates that the outputs from the computational units of level $i-1$ become the inputs to all the units of level i. Figure 2.3 depicts a simple neural network. [14] The large circles represent the computational units with many inputs and one output each.

The processing elements each have a number of linear internal parameters called weights. Changing the values of the weights within an element changes the output of that element which then alters the network input/output relationships. In order to be useful a neural network has to be trained. This involves a learning process by which the weights within the processing elements are adjusted in order for the network to match the input/output relationships of the real process.

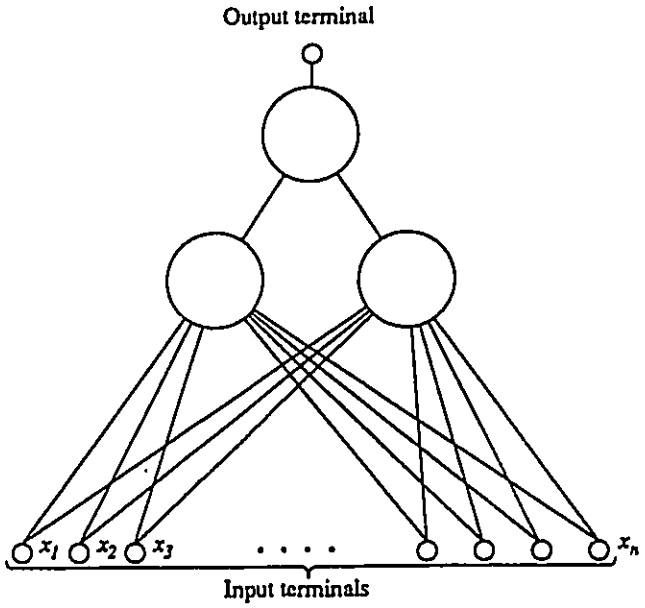


Figure 2.3: A three-unit two-level neural network with n inputs

When the relationships are identical, or nearly identical, then the neural network has become a model of the actual process. The network can then be used in decision making, prediction or control applications.

The neural network approach is being investigated by researchers for application to robot motion control. This approach is promising due to the potential of neural networks to be trained to reproduce the input/output relationships of a complex nonlinear system without extensive knowledge of the system itself. In this particular neural network application, the network is trained to become a model of the manipulator dynamics. The objective of the process is to enable the network to predict the appropriate feedforward torques required in all conditions.

Reference [15] proposes the use of neural networks in highly nonlinear control

problems. References [16] and [17] present some practical applications of neural networks to the control of robot manipulators. One of the main advantages of neural learning is that no initial knowledge of the manipulator dynamics is required. It is also a method in which non-repetitive learning can be applied since "the objective is to learn how to control the robot, rather than to learn how to perform a specific movement" [15].

The "zone learning" method proposed in this thesis has several similarities to neural network learning. Both methods begin with a structured framework containing many variables (or weighting factors) for which values have not been determined. The learning process consists of determining the best possible values for all the parameters. Neither method requires knowledge of the manipulator dynamics, however, the manipulator dynamics are used in the learning process. The learning process is generally trajectory independent, and the amount of memory required for implementation depends on the complexity of the learning structure (numbers of inputs and outputs and processing elements for the neural network, number of "zones" for the proposed method). There are significant differences as well. The neural network begins the learning process with no knowledge while the zone learning method begins with conservative knowledge (controller parameters derived from conventional fixed gain tuning procedures). The neural network approach is more complex and seeks to replace the controller by creating an exact model of manipulator behaviour to be used for predictive control. This requires a large memory and a lengthy learning process as the complexity of the system increases. The zone learning method, on the other hand, simply seeks to augment the existing feedback controller by using a piece-wise linear approach to a highly non-linear system. This

provides the capability for the simple feedback controller to adapt to varying manipulator configurations and velocities..

2.4 Self Tuning Controllers

Numerous self tuning algorithms based on PID and PD controllers have been developed and successfully implemented. However, most of these are applicable to time invariant or very slowly varying plants. The block diagram in Figure 2.4 depicts a simple architecture for such types of automatic tuning. Examples of such algorithms can be found in [18], [19], and [20]. The restrictive conditions of time invariant or even slowly time variant plants render such approaches unusable for robot manipulators since the state parameters of robots are highly time variant.

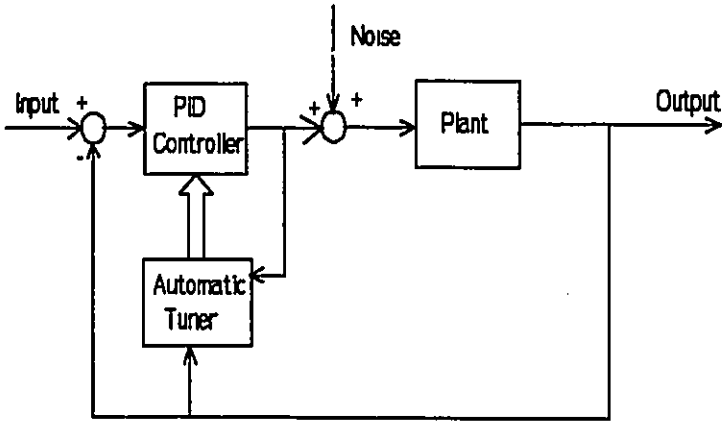


Figure 2.4: Simple Automatic Tuning Architecture

Two different approaches for self tuning of robot manipulators will be discussed. The first method, [21] and [22], uses a transfer function approach with a simplified robot model and parameter estimation from error feedback, to vary the individual joint PID controller parameters as the conditions change along the manipulator's trajectory. This is essentially an

adaptive control technique, not a learning method. While this results in improved performance, no knowledge is gained about the manipulator. Therefore, at the start of every new trajectory, the performance may initially be poor until the adaptation process can have an effect.

The second approach is based on the repetitive nature of many robot tasks. Reference [23] proposes a computer controlled automatic tuning procedure (Autotune) which will iteratively run a robot through a specified trajectory while adjusting the joint controller parameters. The goal of this process is to minimize a specified objective error function along the entire path of the trajectory. This tuning process finds the best possible combination of fixed gains for one particular trajectory. However, the algorithm lacks an explicit overshoot constraint. The learning is trajectory specific and can therefore not be applied to different trajectories. Also the inability to vary the controller parameters, within the trajectory, does not allow for the best possible result from the tuning process.

CHAPTER 3

ROBOT MOTION CONTROL

3.1 Dynamics

3.1.1 Determining Equations of Motion

The two most commonly used methods for determining the equations of motion of an n-link rigid body manipulator are the Newton-Euler and Lagrange methods. The Newton-Euler method is a force based approach in which the links are considered individually. There are two main steps to its application: [24]

(1) the kinematics of the individual links are calculated recursively from link 1 and the Newton-Euler (force/torque) equations are applied to each link.

(2) The forces and torques of interaction and joint actuator torques are then computed recursively from link n to link 1.

The Lagrangian method is energy based and relies on the relationship between a system's rate of energy change and the forces acting on it. Unlike the Newton-Euler method, the robotic manipulator is analyzed as an entire dynamic system, not individual links. The Lagrangian L

is defined as follows:

$$L = \sum_{i=1}^{n} (K_i - U_i) \tag{3.1}$$

The equations of motion for a manipulator with n links are derived by applying the Lagrange equation (3.2) for each coordinate q_i.

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = Q_i \qquad i=1,2,...n \tag{3.2}$$

The Lagrangian formulation results in equations from which the system can be assessed as a whole. The disadvantage of the Lagrange formulation is that the total forces on a link can be found but not necessarily the cause of the force. The equations are computationally complex unless recursive expressions are used, in which case the derivatives of the generalized variables do not appear explicitly. This results in a confused structure. The Newton-Euler formulation, on the other hand, better describes the dynamical forces on each link as a result of the adjacent links. [25]

3.1.2 Robot Model for Simulation and Testing

By application of the Lagrange equation for all links, the idealized equations of motion of an n-link robotic manipulator can be developed. The equations are expressed in terms of the generalized coordinates q_i for each link i. The system dynamics are characterized by n equations of the following form:

$$\sum_{j=1}^{n} I_{ij}\ddot{q}_j + \sum_{j=1}^{n}\sum_{k=1}^{n} C_{ijk}\dot{q}_j\dot{q}_k + B_i\dot{q}_i + g_i + T_{f_i} = Q_i \quad i=1,2,...n \tag{3.3}$$

The robot model which was used in this study is a three axis revolute type depicted in Figure 3.1. It is an idealized model in which the links are represented by slender cylindrical rods and the motors by solid cylinders. Gravity, friction and viscous damping are included in the model, but the effects of backlash are not.

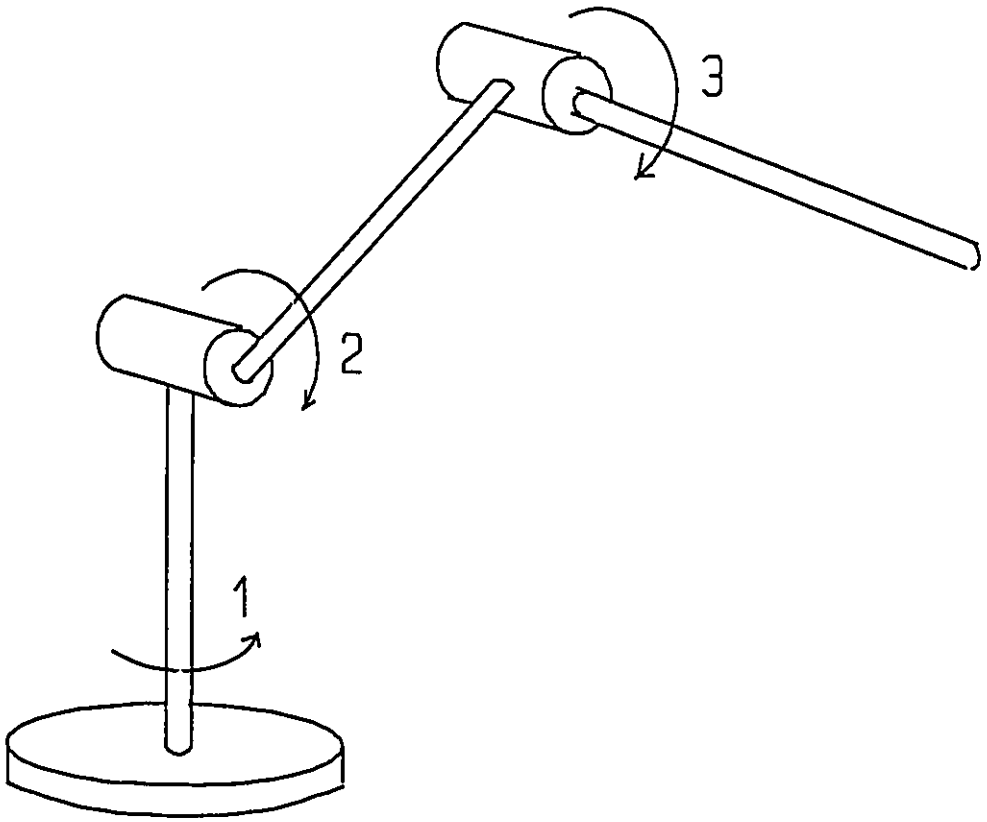


Figure 3.1: Three Link Revolute Robot

Since the velocity disturbance torques are complex to calculate, the effects of $\sum_{j=1}^{n}\sum_{k=1}^{n} C_{ijk}\dot{q}_j\dot{q}_k$ are usually ignored in simple models. However, even with a good model of the

friction T_{f_i} and damping torques $B_i\dot{q}_i$ (which are very difficult to determine), the representation is complex since the inertia varies with the link angles. In reality a fast robot is desirable to increase efficiency and therefore a method of compensation for the forces which result from multiple link motion should be developed. To illustrate the effects of multiple link motion, two components of the disturbance forces were calculated from the complete dynamic equations of motion of the a three link revolute robot representing the two major types of forces:

(1) As an example of the disturbance forces which are functions of the square of other link velocities, the torque induced on link 1 by the motion of link 2 is expressed as follows.

$$T^1{}_{v_1} = \dot{\theta}_2^2 \left[\frac{1}{2}\left(\frac{1}{2}m_{21}+m_{22}\right)l_{21}l_{22}C_2 + m_3\frac{1}{2}l_3\left(\frac{1}{2}l_{13}+\frac{1}{2}l_{22}\right)C_{23} + m_3 l_{21}\left(\frac{1}{2}l_{13}+\frac{1}{2}l_{22}\right)C_2 \right] \tag{3.4}$$

(2) To illustrate the coriolis disturbance force, developed from the product of two different link velocities, the torque induced on link 1 as a result of the motion of both links 1 and 2 is expressed as follows

$$\begin{aligned} T^2{}_{v_1} = -\dot{\theta}_1\dot{\theta}_2 \Big[& \left(\frac{1}{2}m_{21}+2m_{22}\right)l_{21}^2C_2S_1 + (m_{21}+2m_{22})l_{21}l_{13}C_1S_1S_2 \\ & + m_3l_3l_{21}(S_1^2S_{23}C_1+S_1S_2C_1C_{23}+C_1^2(C_2S_{23}+S_2C_{23})) \\ & + m_3(l_3^2C_{23}S_{23}(S_1C_1+C_1^2) + 2l_{21}^2S_2C_2) \Big] \end{aligned} \tag{3.5}$$

The complete derivation of the equations of motion of the three link revolute robot simulator used in this work is presented in Appendix A. The robot was statically balanced before

simulations were run. The static balancing is described in Appendix B.

3.2 Robot Control

The control of an industrial robot generally involves two very different levels: the system/task level and the link motion level. For both the software/hardware combinations are often referred to as "controllers". To differentiate between these two in this work, the system/task controller will be referred to as the "robot controller" and the link motion controller will be referred to as the "link controller".

3.2.1 Robot Controller

The robot controller is responsible for the high functions which can vary depending on the sophistication of the particular manipulator. However, the functions can include task planning, coordinate transformations and coordination of the individual link motions. There is only one robot controller. Figure 3.2 illustrates the basic architecture of the robot controller. Generally, there is no feedback to this controller.

The trajectory for an industrial manipulator usually consists of a set of positions which are functions of time. At the task the trajectory can be defined as a continuous time function or a discrete set of points. Either Cartesian or joint coordinates can be used to define the trajectory. Cartesian coordinates are usually simpler for the user. The manipulator, however,

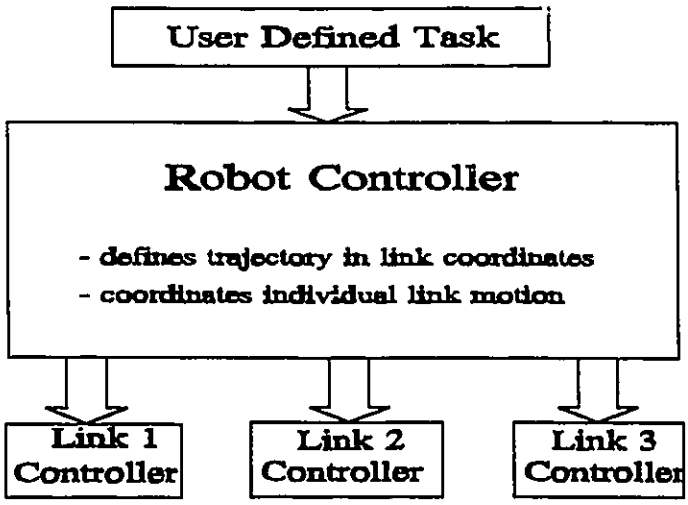


Figure 3.2: Basic Robot Control Architecture

operates in joint coordinates. Therefore any trajectory defined in Cartesian coordinates by the user must be translated to joint coordinates, using inverse kinematics, before execution. This is one of the functions of the robot controller. The inverse kinematics procedures are different for each type of robot. Table 3.1 presents an example of a trajectory in Cartesian coordinates. This trajectory has been translated to joint coordinates for a three link revolute manipulator and is presented in Table 3.2.

3.2.2 Link Controller

Each link of the manipulator has its own feedback controller. The operation of each of these low level controllers is independent of the other link controllers. It drives the link according to an input signal received from the robot controller and measured feedback signals. This relationship is illustrated in Figure 3.3.

X(cm)	Y(cm)	Z(cm)	V(cm/s)
.200	.200	.100	.300
.050	.200	.200	.300
-.100	.200	.300	.300
-.250	.200	.400	.300
-.400	.200	.500	.300
-.400	.050	.500	.300
-.400	-.100	.500	.300
-.400	-.250	.500	.300
-.400	-.400	.500	.300
-.250	-.400	.425	.300
-.100	-.400	.350	.300
.050	-.400	.275	.300
.200	-.400	.200	.300
.200	-.250	.175	.300
.200	-.100	.150	.300
.200	.050	.125	.300
.200	.200	.100	.300

Table 3.1: Cartesian Coordinate Trajectory

Θ_1	Θ_2	Θ_3	time(s)
0.42	1.63	-1.97	0.00
0.87	1.73	-2.44	0.58
1.57	1.23	-2.46	1.16
2.15	0.69	-2.02	1.74
2.45	0.21	-1.29	2.33
2.77	0.28	-1.50	2.81
3.14	0.27	-1.46	3.29
-2.80	0.17	-1.16	3.78
-2.53	0.18	-0.33	4.26
-2.34	0.39	-1.30	4.80
-2.06	0.71	-1.67	5.34
-1.70	0.92	-1.72	5.88
-1.33	0.92	-1.46	6.43
-1.21	1.36	-1.98	6.92
-0.93	1.78	-2.28	7.41
-0.26	1.91	-2.28	7.90
0.42	1.63	-1.97	8.39

Table 3.2: Joint Coordinate Trajectory

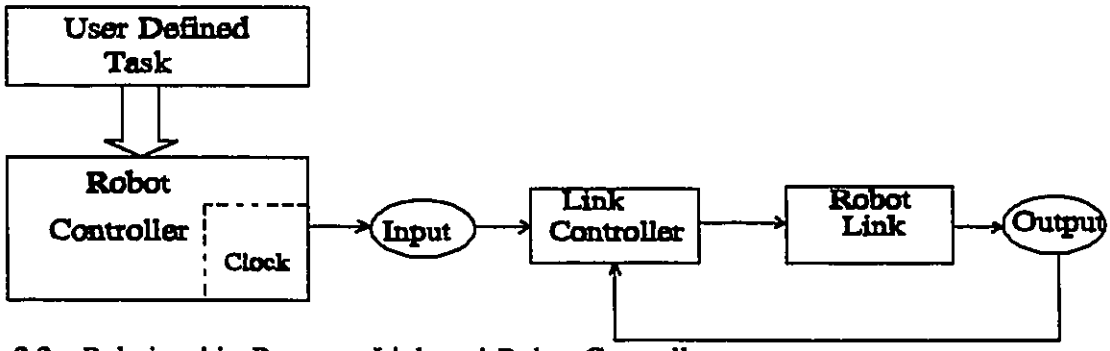


Figure 3.3: Relationship Between Link and Robot Controllers

A very common type of link controller used in industrial robots is the fixed gain proportional controller with velocity feedback. The controller equation for a rotational link of a manipulator is

$$u(t) = K_p[\theta(t)-\theta_d(t)] - K_v\frac{d}{dt}\theta(t) \tag{3.6}$$

Figure 3.4 illustrates, in block diagram form, the principal of proportional control with velocity feedback of a single robot link based on a very simplified robot model.

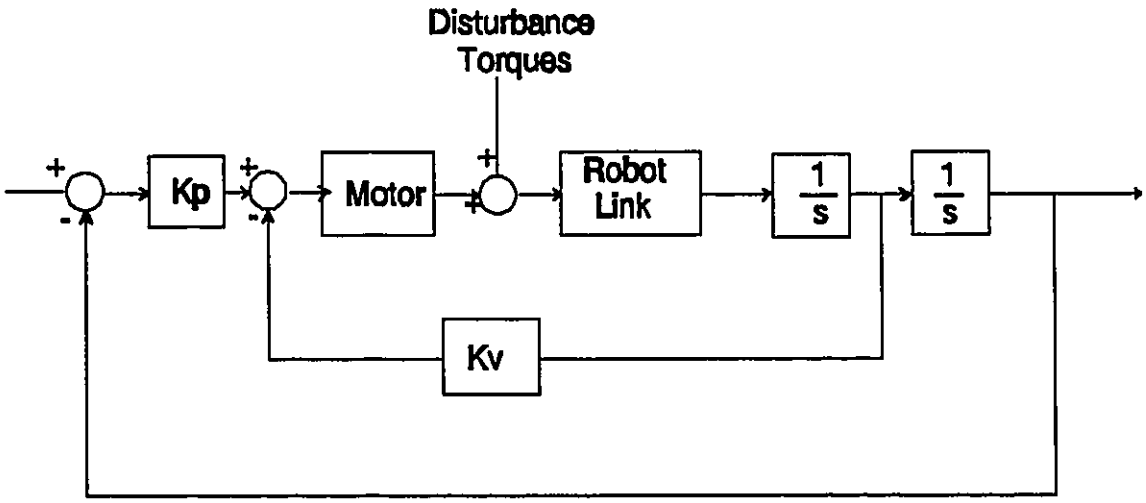


Figure 3.4: Single Link Proportional Controller with Velocity Feedback

The proportional controller for a robotic link has only one input: the position error. For this reason the trajectory cannot include velocity directly and only a position path should be specified for the robot. This normally consists of a list of points through which the robot manipulator must pass.

Time is not a controller variable, therefore the proportional controller, by itself, cannot follow a specified trajectory. The controller can only deal with one position input at any given time. Thus the link controller requires a higher controller (robot controller) to pass the desired trajectory information to it in real time. The robot controller permits the manipulator to follow a trajectory by coordinating (in time) the desired inputs to the link controllers.

Even though there is no velocity control for this type of link controller, a rudimentary control of the velocity is established by specifying a time at each point in the path. The time is usually calculated based upon a desired Cartesian speed of the robot end effector. The set of step inputs are time coordinated by the robot controller. At each time point specified in the trajectory, the robot controller passes the next desired position to the link controller. This results in the specified average velocity over each interval although the velocity varies during each interval.

The time column of the trajectory (Table 3.2) is essential to the motion of the manipulator as it dictates the "speed" of the manipulator. At time t=0.0, the link controllers receive the desired joint positions as step inputs. The links move independently until the next time reference point is reached. At that time the next set of joint positions is passed from the robot controller to the link controllers. If a joint has reached its desired position (step input) before the next time point is given, the link will simply stop and wait for the next input, provided that it is sufficiently damped. If the joint has not yet reached its desired position when the next step input is given, it will ignore the first position and move towards the next point. In that case there is a position error at that target point. The entire trajectory is executed in such a manner (series of step

inputs) until the final position is given. The manipulator will eventually come to a stop at that final position.

The number of points defined in the trajectory is a function of the task and how precise the path control must be. It may be that only the end points of the motion are specified. In this case the manipulator is free to move in any manner in between the specified end points. If a tighter control is required on the path that the manipulator must take in between the end points, then numerous intermediate points must be defined. The manipulator will therefore be forced to pass through or near those intermediate points. However, the path control can never be absolute since there is never any control in between any two specified points of the trajectory.

3.3 Dynamic Modelling of a Single Robotic Link

To design a link controller, the link should be modelled as accurately as possible. The components of a single link of a multiple link manipulator, and its proportional controller with velocity feedback, are presented, in block diagram form, in Figure 3.5.

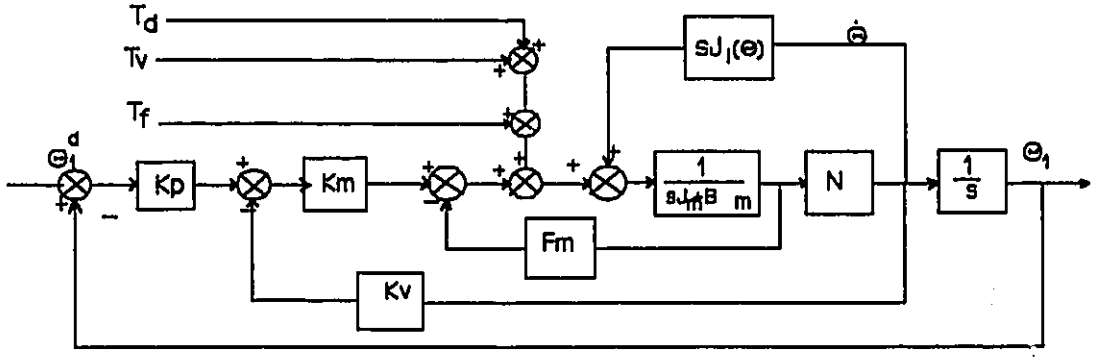


Figure 3.5 Block Diagram of Proportional Control of a Single Link with Velocity Feedback

Some of the link parameters are very difficult to determine accurately for a variety of reasons. The most notable are:

- Link inertia, which depends on mass distribution, is very difficult to calculate or measure. Furthermore, it varies with the link positions.

- Dry friction effects are extremely nonlinear and can only be determined experimentally. Also they may vary with temperature and load.

- Viscous drag can only be determined experimentally. The process, however, is very complicated and subject to ambient variance.

Despite all the deficiencies of the model, it remains the starting point for the controller design.

The dynamic model of the robot link is developed as follows. An electric DC motor is used as a prime mover for the robot links in this work. The dynamic behaviour of an idealized electric motor is given as: [26]

$$J_m\ddot{\theta} = K_m V_m - (F_m+B_m)\dot{\theta} \tag{3.7}$$

From equation 3.7 the Laplace domain transfer function is developed for a single link controlled by a proportional controller with velocity feedback:

$$G(s) = \frac{\theta(s)}{\theta_d(s)} = \frac{K_p K_m}{(J_m+J_l \cdot N^2)s^2 + (B_m+F_m+K_v K_m)s + K_p K_m} \tag{3.8}$$

The disturbance forces T_v, T_d and T_f are not included in this equation. All the variables are functions of the physical system with the exception of the controller parameters K_p and K_v. The

design of the controller consists of choosing the values for those two parameters such that the best possible performance of the robot link is achieved.

3.4 Link Controller Design

The key to the design of an effective link controller is the choice of the fixed gains K_p and K_v. For a robot manipulator some of the most desired system characteristics are: [27]

(1) accurate static and dynamic positioning

(2) smooth small motions

(3) rapid gross motions (fast response)

(4) stability

(5) no overshoot

The choice of gains will inevitably lead to compromises among these objectives. The controller proportional gain K_p is related to the speed of the response, and in some respects, the position error. The velocity feedback gain K_v is instrumental in controlling the damping of the system, and hence the overshoot.

For simple time invariant, linear systems, determining the best values for the controller is a difficult task. The highest possible proportional gain is chosen which will not result in an unstable system. Then the value of K_v is calculated to achieve a desired damping. In the case

of a robot, where overshoot is not tolerated, critical damping is chosen (damping coefficient = 1.0). Both values can be determined from the transfer function. To implement this on an individual link, the following procedure is used.

(1) The simplified model is used.

- Estimations are used for T_f and T_d

- The effect of T_v is generally ignored

(2) The maximum possible value of I is used.

(3) The controller is designed for the fastest stable response (highest possible K_p) with a damping coefficient of 1.0 at maximum I.

A set of four criteria is presented in [28] which serve as guidelines for the design of each link controller.

1. For safety reasons, the manipulator must never overshoot its target. Therefore the system must be overdamped ($\xi > 1$) or at least critically damped ($\xi = 1$). Since damping slows the response of the system a critically damped system will give the best performance while adhering to the no overshoot criteria.

2. The proportional controller cannot completely eliminate the final position error along the gravity gradient direction. This is because, in the steady state, an error signal is only generated when a position error is detected. In order to reduce the steady state error, a very large

proportional gain is desired. This criteria is not a factor in this work since an automatic gravity compensation scheme has been implemented in the controller for links 2 and 3 which eliminates the steady state error due to gravity. This scheme is described in Appendix C. Despite this, the recommendation remains valid since a higher K_p decreases the response time of the system which is desirable.

3. To prevent resonant oscillations of the whole structure of the robot manipulator itself, it must be ensured that the servo-system characteristic frequency (ω_n) is significantly less than the lowest robot structural frequency (ω_o). From experience it has been found that when $\omega_n < .5\ \omega_o$, the structural frequency cannot be excited and undesired oscillations cannot appear. The structural frequency can only be found experimentally, and is very difficult to determine. For the simulation, an estimate is used based on results obtained from other similar robots. This criteria limits the value of the proportional gain, since the characteristic frequency of the servo-system is dependent on K_p.

4. Electrical noise in the feedback signals are also amplified by the gains K_p and K_v. Therefore if the gains are too large, this amplified noise could become significant. However, since the amplitude of the noise is usually much lower than that of the actual signal, any value of K_p which satisfies requirement 3 will also usually satisfy this requirement.

The task, therefore, is to choose a K_p as large as possible, which also satisfies requirements 3 and 4. K_v is then chosen to critically damp the servo-system according to condition 1. [5]

The transfer function of the single link robot model given by equation 3.8 is a second order one and can be cast in the form

$$G(s) = \frac{\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2} \tag{3.9}$$

The system natural frequency is therefore

$$\omega_n^2 = \frac{K_p K_m}{(J_m + J_l N^2)} \tag{3.10}$$

and the damping ratio is

$$\xi = \frac{(B_m + F_m + K_v K_m)}{2\omega_n (J_m + J_l N^2)} \tag{3.11}$$

From condition 3 ($\omega_n < .5\omega_o$), the maximum proportional gain can be calculated as

$$K_p = \frac{\omega_o^2}{4_m}(J_m + J_l N^2) \tag{3.12}$$

For a critically damped system, K_v can be calculated

$$K_v = \frac{2}{K_m}[\sqrt{K_m K_p (J_m + J_l N^2)} - F_m - B_m] \tag{3.13}$$

This analysis is necessary for each link i.

3.5 Tuning

The link controller design method presented in section 3.4 was based on several assumptions and approximations. The choice of gains is therefore unlikely to be optimum for actual robot operations, where the assumptions are not all valid and the approximations not precise. It is therefore usually required to further refine the controller by modifying the gains as a result of tests of actual robot motion.

3.5.1 Considerations for a Robot Manipulator

Tuning the manipulator consists of choosing the values for the link controller gains K_p and K_v which will give the best possible performance while respecting the given constraints. The objective of the tuning is to balance the following desired attributes for the manipulator: (1) provide the manipulator with the fastest possible response, (2) reduce position errors to a minimum, (3) avoid, at all costs, overshoot of target points.

Several effects, which cannot be ignored, make the tuning of a robotic manipulator a difficult task.

(1) the actuator dynamics are an important component of the system transfer function and

must be included.

(2) the robot inertia is variable and depends on the positions of the other links of the manipulator,

(3) there are several types of disturbance forces which are difficult or impossible to determine or very complex to calculate (friction, viscous damping, coriolis and centrifugal forces).

The inclusion of the actuator model results in a more complex set of equations which can be analytically solved. However, further difficulties arise from the fact that the inertia terms (I_{ij}) are not constant, but functions of the generalized coordinates (q_i). This makes a good design for a fixed gain controller over all values of q_i impossible to achieve. The problem is also compounded by the fact that accurate estimates of friction and viscous damping are difficult or impossible to achieve, and that the centrifugal and coriolis forces are nonlinear and difficult to calculate.

3.5.2 Ziegler-Nichols Rules

Ziegler and Nichols developed a set of empirical rules for the optimum gains values of PID controllers [29]. Two methods were proposed: one for systems which can provide an open loop process response, and one for a closed loop response (with the controller in the loop). The robot manipulator has no stable open loop response, therefore the second method (known

as the ultimate cycle method) would normally apply.

However, this standard method is not suited to tuning robot manipulators. Firstly, because a robot manipulator is a time varying system, which is beyond the scope of this method. Secondly, since the torque provided by the link motors is limited, increasing the gain beyond a certain value does not alter the response of the system. Furthermore, the criteria are specified for P, PI and PID controllers but not for proportional controllers with velocity feedback and only for continuous systems with analogue controllers.

3.5.3 Manual Fine Tuning

Even when a fairly extensive process is used to design the link controller, it is usually necessary to manually tune the robot. The tuning process consists of slightly modifying the designed gains to achieve the best possible performance of the robot. This process involves commanding the actual robot to move over trial trajectories. Examples of problems which may be encountered when using the actual robot, and which may be alleviated by tuning are listed here: [4]

- slow response (K_v too high)
- inaccurate positioning (K_p too low)
- instability (K_p too high)
- underdamped response (K_v too low)

- inadequate stiffness (K_v too low)

3.5.4 Automatic Tuning

It is possible to construct automatic methods for the purpose of fine tuning robot manipulators. Several such methods have been reported in the Literature Review in Chapter 2. This thesis will present a new method for a learning procedure that uses zone dependant controller gains and tunes continuously as the robot operates.

3.6 Shortcomings of Fixed Gain Link Controllers

One of the major drawbacks of the standard link controller, when applied to a robot manipulator, is that the gains K_p and K_v are fixed. Fixed gains are not ideal since a robot manipulator is a time variant system. The performance of the system could be greatly improved if the controller gains were variable and allowed to change in response to changing conditions. Some of the conditions which make the system time variant are discussed below.

3.6.1. Changes in Inertia

The standard design method (Section 3.4) selects the controller velocity gain to achieve critical damping for maximum inertia. The inertia, which is the major component of a rotational link's actuator load for link acceleration, is dependant on the manipulator configuration

(Appendix D). Each of the manipulator's individual link controllers is unaware of the positions of all the other links of the manipulator. However, the positions of the other links can have a major effect on the control of a given link.

It is theoretically possible to vary K_v as the inertia varies to maintain critical damping. This should result in improved performance while maintaining the no overshoot criteria. This would make it possible for the manipulator to be near critically damped in all configurations, rather than only in the most extreme. However, some drawbacks persist:

(1) The inertia is difficult to calculate exactly. Therefore only an approximation of the inertia could realistically be used.

(2) The computation time, for a multiple link robot, may be quite lengthy, thereby limiting the controller sampling time.

(3) Numerous other factors affecting the performance of the robot are ignored such as friction, viscous damping, and the forces generated from the movement of multiple links.

(4) The damping would not necessarily be critical at every point since the calculation of K_v imposed no limits on the input force (torque). In practice, the torque available from an actuator (such as an electric motor) is limited to a certain maximum value. It would thus be practically impossible to verify if, under all conditions, the system is not underdamped.

The performance of a robot manipulator could be significantly improved, while maintaining the no overshoot constraint, if the controller could compensate for the configuration

dependence of the link inertias.

3.6.2 Changes in Load

The load being carried by the robot manipulator can greatly affect the performance or position errors. The load inertia will add to the problem of the varying inertia. Also the load will affect the gravity loading on the links. For an unknown load, this makes gravity compensation very difficult. A new method of gravity compensation under uncertainty has been developed and applied to a robot simulator for this research work. This method is described in Appendix C.

3.6.3 Effects of Link Motion

The link controller is generally designed by considering the links independently. As such the nonlinear centrifugal and coriolis forces cannot directly influence the controller design. They are therefore considered as disturbances to the system and throughout the remainder of this work these will be referred to as disturbance forces. At low operating speeds, the velocity disturbance forces are small and thus may be ignored without serious effects. However, at higher operating velocities, these forces are larger and can cause position errors.

It is possible to try and calculate all the forces acting on each link and use this result, in addition to the calculated inertia value, to modify the velocity gain. However, the calculation

of all these parameters is very complex and, in most cases, very lengthy. The time required for the calculations make this very difficult to implement in real time. Also the result is unlikely to be precise since an imperfect model is used. Such a sophisticated procedure may be justified for continuously controlled trajectories, but much simpler methods are available for point to point trajectory applications.

3.6.4 Summary

The characteristic of inertia variation in a robotic manipulator arm makes for a configuration dependant dynamic system. Therefore classical control methods are inadequate for the design of good control systems. In addition, the disturbance torques induced by the motion of multiple links are difficult to model and compensate. The problem is further compounded by the uncertainties of the values of some link parameters such as inertia, friction and damping.

Despite all the uncertainties, proportional fixed gain link controllers with velocity feedback are commonly used in industrial robots due to their simplicity and stability. The fixed gains are chosen to create an overdamped system. This will ensure that any errors caused by the modelled uncertainties will not result in overshoot of the manipulator. Although this is a relatively simple system to design and implement, it results in poor performance.

CHAPTER 4

PROPOSED LEARNING METHOD

4.1 Improvements to Individual Link Controllers For Robots

Despite the difficulties discussed in Section 3.6 individual link controllers are commonly used in industrial robots due to their simplicity and stability. However, fixed gains are not ideal since a robot manipulator is a time variant system. Although many control methods have been developed which offer improved performance over proportional controllers with velocity feedback, their complexity, instabilities or other drawbacks make them often unattractive in industrial applications. Proportional controllers with velocity feedback currently make up a large percentage of controllers used on industrial robots. Therefore any method which can be implemented on existing hardware should be explored. The newer, more elaborate, control methods may be incorporated into the design of future generation robots.

Some major considerations influencing the choice of improvement methods are:

1. Improvements to the link controllers should be as general as possible. This is to ensure that the method can be implemented on many robots with different configurations and can be applied to different types of trajectories. If a learning method is to be used, it is preferable that the learning be general, and not trajectory specific.

2. A new method should not have, as an essential requirement, detailed knowledge of the robot system's dynamics. Such model based methods are difficult to implement since a completely accurate model is nearly impossible to produce. Even approximate models can require very complex calculations.

3. The new method should not necessitate complex calculations nor require a large memory capacity. Complex calculations can limit the controller sampling frequency and many industrial robots do not have large memory capacities.

4. An important characteristic of any new method is compatibility with existing hardware configurations. This makes upgrades inexpensive and therefore attractive to industry. Figure 4.1 presents, in block diagram form, the simplified architecture of a single robot link proportional controller with velocity feedback. Any new method should by compatible with this structure.

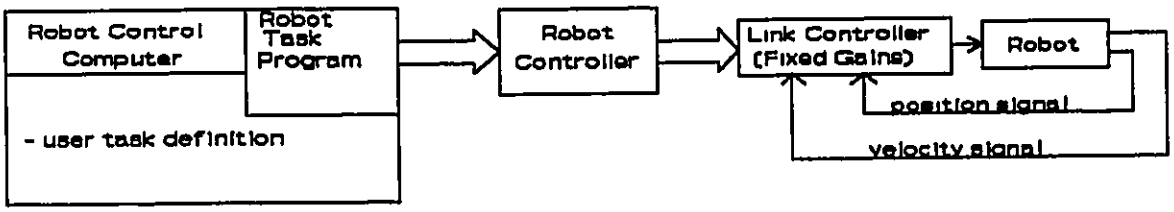


Figure 4.1: Basic Robot/Link Controller Architecture

4.2 Zone Learning Approach

As described in Section 3.6, changes in certain operational parameters of the manipulator (such as configuration, velocities, and load) can affect system performance. The performance could be improved if the link controller gains were tailored to each of the different possible values of the operational parameters. An analytic solution to link parameter variability is possible but is not without difficulties (Section 3.6).

This thesis proposes that a learning process be used to determine the best possible link controller gains for each set of specific operational parameter values. This can be done by grouping the operational parameter values into ranges of values or "zones". The types of zones which can be defined include configuration zones which are based on the positions of the manipulator links, and configuration-velocity zones which are based on the velocities of all the links within the configuration zones. The learning process seeks to determine the best possible values of the link controller gains within each of the defined zones for use during execution of manipulator trajectories.

4.3 Configuration Zones Learning

4.3.1 Basic Concept of Configuration Zone Learning Method

The objective of the proposed configuration zone learning method is to determine through

a learning process the most appropriate level of damping for each portion of the robot's work volume. Ideally, the link gains should be continuously changing in response to the changing positions of the links. However since this is very difficult to implement zones are defined within which the link gains are constant.

The manipulator's work volume can be partitioned into zones based on its configuration. Within each of these zones, a data set can be defined and their values determined through a learning process based on executing numerous different trajectories. This data set will contain multipliers λ_i which will be used in the control law to modify the velocity feedback gain and hence the damping of each link. In this manner, the damping becomes a learned value which will be configuration dependant. The control law for each link then becomes

$$u(t) = K_p[\theta_d(t) - \theta(t)] - K_v\dot{\theta}(t) \cdot \lambda_i \qquad (4.1)$$

The number of data sets is dependent on the number of partitions of the robot work volume which also dictates the physical size of each of the configuration zones. The multiplier values required to achieve critical damping will vary somewhat within a zone unless the zones are made infinitesimally small. Therefore the value of each λ_i will converge towards the worst case configuration within each zone. This is similar to the normal method of tuning the fixed gain controllers for the worst case condition. However, in this method, the volume within which the worst case is found is greatly restricted. This should result in much better performance than is possible with fixed gains (i.e. a single zone).

Although the partition of the work volume into zones will be based on manipulator configuration the method will give different results from those based on calculated variations of inertia. This is because the system will be learning on the actual robot, not on a model whose values for inertia may not be exact. Also, since the learning system cannot distinguish between effects cause by inertia variation or those caused by other disturbances (e.g. friction) the learning process includes, by necessity, the effect of friction and other repeatable disturbances. Additionally, it is an important feature of this learning system that knowledge acquired during one trajectory can be applied to different trajectories, provided that they traverse the same region of robot work volume. The block diagram of the link/controller system modified to include effects of the configuration zone data base is shown in Figure 4.2.

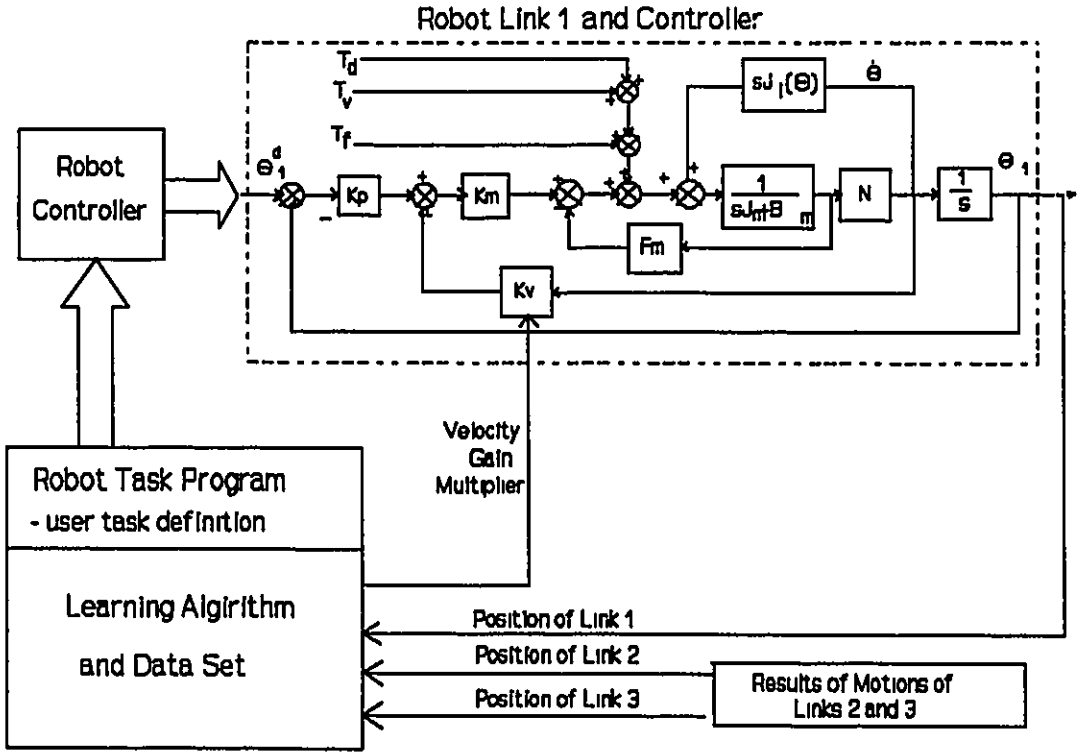


Figure 4.2: Use of Configuration Zone Data Base to Modify Velocity Gain

4.3.2 Description of the Method

The structure and operation of the learning system is described using set relations. The set theory and boolean algebra notations follow those specified in [30]. Given a single link, there are an infinite number of positions which that link can occupy within the bounds of its degree of freedom (ex. $(0,2\pi)$). A set E, part of the universal set of real numbers U, can be defined which contains all the possible positions (θ) of a given link.

$$E = \{\ \theta \in U \mid 0 \leq \theta < 2\pi\ \} \tag{4.2}$$

The set E can be divided into a finite number of subsets, each corresponding to a portion of the link position domain. These subsets will each contain an infinite number of members (link positions). Figure 4.3 illustrates the domain of the link position $(0,2\pi)$ partitioned into n angular sectors each corresponding to a subset of E. The i^{th} subset will be called S_i and is defined as follows.

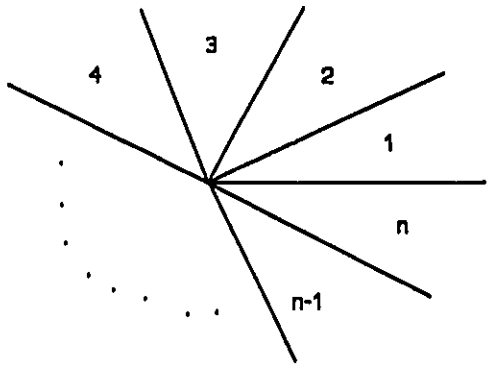


Figure 4.3: Single Link Angular Sectors

$$S_i = \left\{\ \theta \in E \mid \frac{2\pi}{n}(i-1) \leq \theta < \frac{2\pi}{n} i\ \right\} \tag{4.3}$$

From this definition any given link position will be a member of exactly one of the n subsets

(S_i) of E and therefore the following two relations hold true.

$$\{ S_1 \cup S_2 \cup ....S_{n-1} \cup S_n \} = E \tag{4.4}$$

$$\{ S_1 \cap S_2 \cap ... S_{n-1} \cap S_n \} = \varnothing \tag{4.5}$$

An alternate subset definition can be used in which the intersection of two successive zones is not an empty set. In this case the i^{th} subset S_i is defined as

$$S_i = \left\{ \theta \in E \mid \frac{2\pi}{n}(i-1) - \delta \leq \theta < \frac{2\pi}{n} i + \delta \right\} \tag{4.6}$$

where δ is the size of the hysteresis zone.

This formulation can be useful in providing a smoother parameter transition between adjacent zones, especially for trajectories which follow close to zone boundaries or have frequent zone changes. In this work, however, the non-overlapping zones formulation will be used.

During the execution of a trajectory the link moves and θ changes value. The membership of the link position will change between the subsets as the link enters and exits the angular sectors defining their boundaries. The presence of the link in zone i is confirmed by the following logical relation.

$$A \bigcap B = 1 \tag{4.7}$$

where A and B are boolean objects defined as follows:

$$A = \theta \geq \frac{2\pi}{n}(i-1)$$
$$B = \theta < \frac{2\pi}{n}i$$

Since the concept of the proposed learning system is to associate gain multiplier values with each zone of the robot work volume, two additional members are added to each S_i subset. These members form a subset of their own within the subset of link positions. They are different in that operations will be allowed on these two members only. This subset, called the "data" subset, is labelled D and is defined to contain two members.

$$D_i = \{ \lambda_i, \Lambda_i \}, \qquad D_i \subset S_i \tag{4.8}$$

The two members of the subset have following functions:

λ_i is called the multiplier. The value of this element will be used to multiply the feedback velocity gain in the control equation.

Λ_i is called the limit value. This limit value is initialized when an overshoot occurs and records the lowest possible value which can be assigned to λ_i during the learning process. The detailed operations of these data values will be discussed later.

The desired trajectory for the robot manipulator is defined as a set of time referenced link positions that can be represented as a finite set T composed of the time/position pairs.

$$T = \{ (t_0, \theta_d^0), (t_1, \theta_d^1), \ldots (t_m, \theta_d^m) \} \tag{4.9}$$

with θ_d^j being the desired link position at time t_j.

It should be noted, however, that the actual trajectory that would be executed contains an infinite number of points. This trajectory can be divided into m segments, each bounded by two successive reference times (i.e. the bounds of segment j would be (t_{j-1}, t_j). The infinite number of actual time referenced link positions ($\theta(t)$) in a given segment j of the trajectory belongs to the set (P_j) which is defined as follows.

$$P_j = \{ \ \theta \in E \mid \theta = \theta(t) \qquad for \ \ t_{j-1} < t \leq t_j \ \} \tag{4.10}$$

For each trajectory segment j, it is necessary to determine the "active" set of all configuration subsets (S_i) for which learning is possible. This active set A_j is defined as the set of all "zones" or sectors which were traversed by the link during the execution of that segment.

$$A_j = \{ \ S_i \in E \mid P_j \cap S_i \neq \varnothing \ \} \qquad i=1...n \tag{4.11}$$

The system performance is measured at the reference time points as the error vector length from the desired to the actual link positions. The objective of the learning process is to reduce the damping in each angular sector so as to increase the speed of the response and, at the same time, reduce the cumulative error at the reference points. However, overshoot must be avoided.

The boolean object O_i^t records the presence of overshoot occurring within zone i at time t. The value of O_i^t is determined by the following logical relation.

$$O_i^t = \left[(\theta(t)>\theta_d^j) \bigcap (\theta(t_{j-1}<\theta_d^j)\right] \bigcup \left[(\theta(t)<\theta_d^j) \bigcap (\theta(t_{j-1})>\theta_d^j)\right] \quad (4.12)$$

Since there are an infinite number of times t, there are an infinite number of O_i^t objects. The boolean object O_i represents the occurrence of overshoot at any point within zone i and its value is determined by the following logical relation.

$$O_i = O_i^{t_a} \bigcup O_i^{t_a+\delta t} \bigcup O_i^{t_a+2\delta t} \bigcup \ldots \bigcup O_i^{t_b-\delta t} \bigcup O_i^{t_b} \quad (4.13)$$

where δt is an infinitesimally small interval of time.

t_a is the time at which the manipulator entered the zone

t_b is the time at which the manipulator exited the zone

At time t_j the following learning laws are applied to the data subsets of all $S_i \in A_j$.

$$\lambda_i = \begin{cases} 1-\Delta_1 & \textit{for } O_i \textit{ false} \\ 1+\Delta_2 & \textit{for } O_i \textit{ true} \end{cases} \quad (4.14)$$

$$\Lambda_i = \lambda_i \quad \textit{for } O_i \textit{ true}$$

Both Δ_1, the learning rate, and Δ_2, the step back, can be set by the user. The learning laws have the effect of:

(1) reducing the system damping in all the zones whose corresponding position subsets S_i are members of the active set A_j in the case of no overshoot, or

(2) increasing the system damping and setting the limit value in case of an overshoot.

The multiplier λ_i of the data subset $D_i = \{ \lambda_i , \Lambda_i \}$ is initially defined to be equal to

1.0. This is the starting point for the learning process where the control equation will not be modified by the learning system. The robot operation will be the same as that of a fixed gain robot system where it is tuned for the worst case conditions. As the manipulator executes trajectories and learns, the value of λ_i will decrease. It should be noted that λ_i will not exceed 1.0 since the initial fixed velocity gain of the link has been tuned for the most extreme set of conditions which will be encountered during operations.

The limit Λ_i indicates the minimum possible value which the multiplier can take without an overshoot condition arising. This value is initially undetermined and it is therefore set to zero. As the learning process progresses, the λ_i will gradually decrease until, at some point, an overshoot condition will occur. At that point, the value of both Λ_i will be set to be equal to that of λ_i. The membership of subset D_i will then decrease to one as λ_i will absorb Λ_i. From then on, λ_i will not be permitted to decrease, but may increase (and will likely do so) as different overshoot conditions are encountered in other trajectories.

After many trials the entire robot work volume (all the zones) will have been explored several times. The variables in all the data subsets D_i will have reached their ultimate values and the learning will be complete. A graphical representation of this learning algorithm is presented in the Figure 4.4.

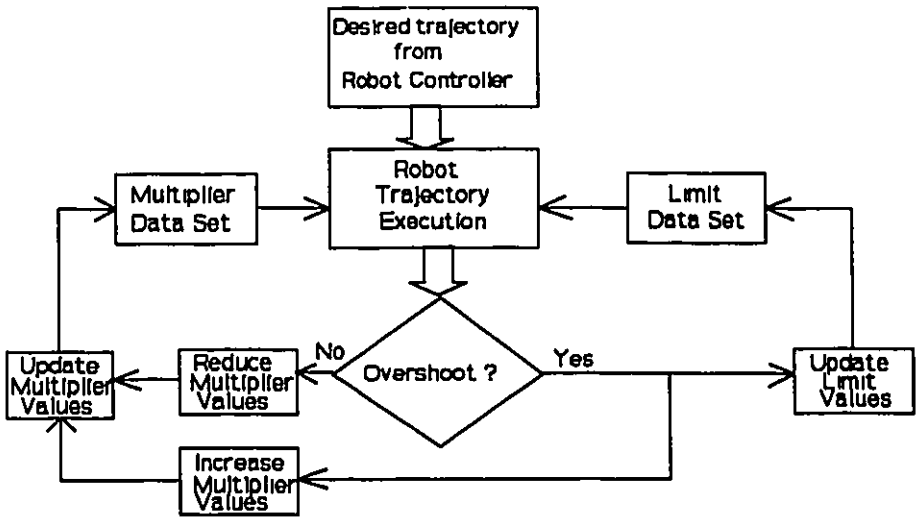


Figure 4.4: Learning Algorithm Block Diagram

4.3.3 Zone Space Definition for Links 2 and 3

The mass moment of inertia about the axis of link 2 is a function of the position of a single link; namely link 3 (Appendix D). Therefore the work volume of link 2 is partitioned into n equal angular sectors of link 3 position. This results in n subsets S_i as described in the previous section and represents the configuration zones for link 2. Although the mass moment of inertia about the axis of rotation of link 3, does not vary, it was decided to implement the learning system on that link as well, in the same manner as for link two. This will permit the learning system to compensate for other disturbances which may exist. Its configuration zone space, defined by the subsets S_i, is identical to that of link 2.

4.3.4 Zone Space Definition for Link 1

The mass moment of inertia about the axis of rotation of link 1 varies with the positions of both links 2 and 3 (Appendix D). It is therefore necessary to define a more complex configuration zone space for link 1 than is required for the other two links. For the purposes of the proposed learning method the configuration space for link 1 is defined by combinations of angular positions of links 2 and 3 and represented by the infinite set of all possible combinations of the angular orientations of links 2 and 3 (i.e. θ_2, θ_3 pairs) as follows:

$$E = \{ \ (\theta_2 , \theta_3) \in U \mid (0 \le \theta_2 < 2\pi \ , 0 \le \theta_3 < 2\pi) \ \} \tag{4.15}$$

E can be divided into a finite number of subsets based on angular sectors for links 2 and 3. If both links contain n angular sectors, then the total number of subsets will be n^2. The subset S_{ij}, which contains all the members of zone $(i - 1)n + j$ is defined as

$$S_{ij} = \left\{ (\theta_2, \theta_3) \in E \mid (\tfrac{2\pi}{n}(i-1) \le \theta_2 < \tfrac{2\pi}{n} i \ , \ \tfrac{2\pi}{n}(j-1) \le \theta_3 < \tfrac{2\pi}{n} j) \right\} \tag{4.16}$$

The presence of link 1 of the manipulator in zone $(i - 1)n + j$ can be determined using the following logical relation

$$A \bigcap B \bigcap C \bigcap D = 1 \tag{4.17}$$

where A, B, C and D are boolean objects defined as follows:

$$A = \theta_2 \ge \tfrac{2\pi}{n}(i-1)$$
$$B = \theta_2 < \tfrac{2\pi}{n} i$$
$$C = \theta_3 \ge \tfrac{2\pi}{n}(j-1)$$
$$D = \theta_3 < \tfrac{2\pi}{n} j$$

4.3.5 Practical Realization of the Configuration Zones Learning Method

The proposed configuration zone learning method can be implemented on existing fixed gain proportional controllers with velocity feedback only when two requirements are met.

1. All the link positions and velocities are known at all times.
2. The link controller K_v gain or the velocity feedback signal can be modified.

The first requirement is almost certainly met since the link positions and velocities are part of the standard control loops of manipulator systems. The second requirement may be satisfied by either of two methods.

- If the link controller gains are alterable by software, the robot controller can modify these values by direct application of the appropriate multipliers from the data bases. The gain modification equation is thus written as

$$K_{v\ (mod)} = K_v\ \lambda_i \tag{4.18}$$

and the controller equation as

$$u(t) = K_p[\theta_d(t)-\theta(t)] - K_{v\ (mod)}\ \dot{\theta}(t) \tag{4.19}$$

- If the controller unit is sealed and the gains are not alterable the velocity feedback signal input to the controller can be amplified or attenuated by an amount proportional to the multiplier values. Figure 4.5 shows a realization of this approach. With reference to the figure the velocity signal is intercepted and read by the robot controller. The appropriate multipliers from the

learning algorithm data base augmented to the robot controller are then used to modify the value of the feedback signal as follows:

$$\dot{\theta}_{(mod)} = \dot{\theta}\ \lambda_i \tag{4.20}$$

The modified velocity is then converted to an analogue signal and fed to the link controller unit. Thus, the objective of changing the damping is achieved with no modification to the link controller. In this case the control equation becomes

$$u(t) = K_p[\theta_d(t) - \theta(t)] - K_v\ \dot{\theta}(t)_{(mod)} \tag{4.21}$$

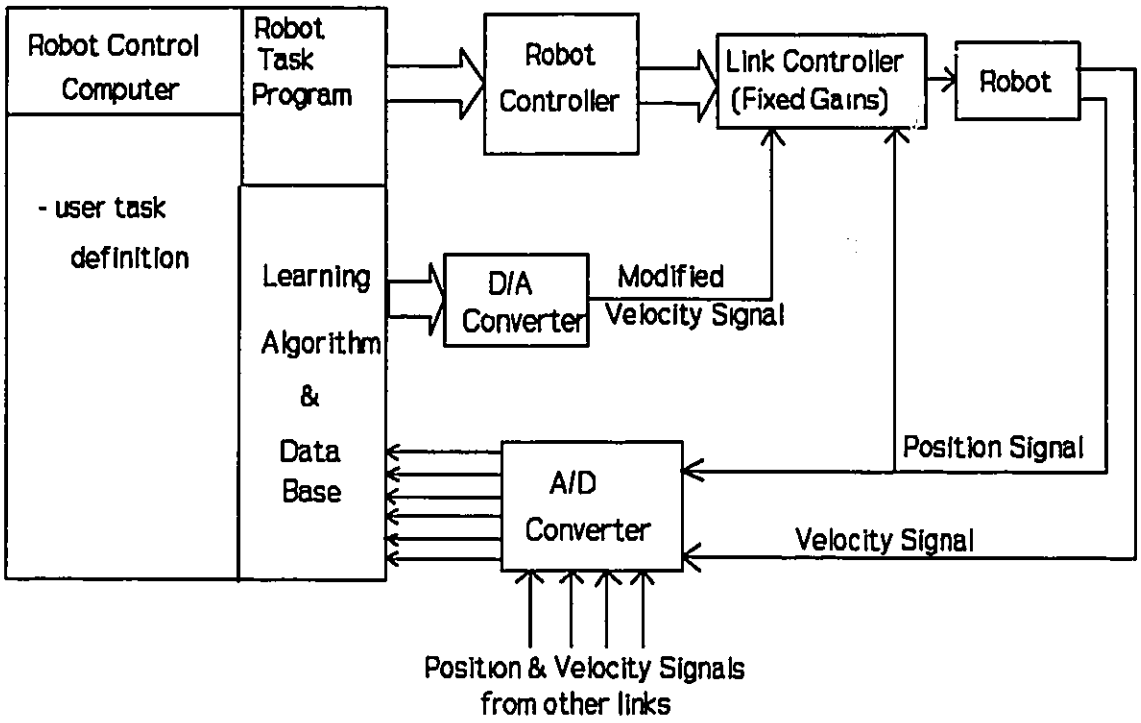


Figure 4.5: Block Diagram of the Learning System Implementation

4.4 Configuration-Velocity Zone Learning

4.4.1 Basic Concept of the Configuration-Velocity Zone Learning Method

In a specific zone, the inertia values are consistent from one motion to another. However, the disturbance forces caused by multiple link motion can vary from one trajectory to another. Therefore, over many iterations, the values present in the limit data base represent the worst set of conditions encountered in that particular zone. This is because whenever a condition causes a link to overshoot its target point, a limit value is set. The damping is increased slightly and will not be allowed to be reduced below that limit.

To improve the learning process, it is necessary to include as many of the operational parameters that affect the performance of the manipulator as possible. The most important of these parameters are the link velocities which directly lead to the centrifugal and coriolis forces. These forces act as disturbances on the links and can cause position errors (Section 3.6.3). Once the configuration zone learning is complete, a second learning process can be initiated to compensate for the effects of link motion.

4.4.2 Description of the Method

Configuration-velocity zones can be defined by partitioning the set of all possible link velocities within each configuration zone into ranges of velocities defined in set notation as:

$$W = \{ \dot{\theta} \in U \mid a < \dot{\theta} \leq b \} \tag{4.22}$$

a and b are the lower and upper bounds of the possible link velocities. These values are functions of the manipulator's physical characteristics or limits set by the robot manufacturer or by the user. W contains an infinite number of members. However, it can be subdivided into a finite number of subsets, each with an infinite number of members. The l^{th} of m subsets for a certain link is defined as follows.

$$V_l = \{ \dot{\theta} \in W \mid a_l < \dot{\theta} \leq b_l \} \tag{4.23}$$

where a_l and b_l are the lower and upper bounds of the range of link velocities in subset V_l. If the subsets are defined such that $a_l = b_{l-1}$ and $b_l = a_{l+1}$, then

$$\{ V_1 \cup V_2 \ldots \cup V_n \} = W \tag{4.24}$$

$$\{ V_1 \cap V_2 \ldots \cap V_n \} = \varnothing \tag{4.25}$$

This group of subsets is similar to that of the S_l subsets of the configuration zone approach. In an alternate formulation the subsets are defined such that $a_l < b_{l-1}$ and $b_l > a_{l+1}$ creating overlapping zones. The hysteresis created can be useful in developing smoother transitions between zones. However, in this work the non-overlapping zones formulation is used.

During the execution of a trajectory segment, the velocity of the link varies greatly. Typically the link first accelerates, cruises at a speed for a while, and then decelerates towards the desired position. The link velocity can therefore have membership in many of the subsets during each single segment. However, since this acceleration and deceleration periods are

generally small in comparison to the constant speed periods, the average velocity of the link during the segment will be used. Therefore the subsets V_i are redefined as the subsets of all the possible average link velocities during the trajectory segments.

To account for all the possible disturbance torques induced by link motion, the velocities of all the links must be considered. The average velocity subsets (m for each link) must then be combined with each other to generate m^3 combination velocity subsets for a 3 degree of freedom robot. From the following velocity subsets for each link

V_i = subset i of link 1 average velocities

V_j = subset j of link 2 average velocities

V_k = subset k of link 3 average velocities

the combination subsets can be defined.

$$V_{ijk} = \left\{ (\dot{\theta}_1,\dot{\theta}_2,\dot{\theta}_3) \in W \mid (\dot{\theta}_1 \in V_i , \dot{\theta}_2 \in V_j , \dot{\theta}_3 \in V_k) \right\} \tag{4.26}$$

For each trajectory segment j, an active set A_j of velocity subsets is defined by the expression

$$A_j = \left\{ V_{ijk} \in W \mid (\dot{\theta}_1 \in V_i) \cap (\dot{\theta}_2 \in V_j) \cap (\dot{\theta}_3 \in v_k) \right\} \tag{4.27}$$

Since the subsets V_{ijk} have been defined using the average link velocities within a segment, A_j will contain only one V_{ijk} subset.

The members of the subsets V_{ijk} are combinations of groups of link velocities. In a

parallel structure to that of the configuration subsets, each of the velocity subsets contains a data subset with two members: a multiplier γ and a limit Γ.

$$D_{ijk} = \{ \gamma_l, \Gamma_l \}, \quad D \subset V_{ijk}, \qquad l=(i-1)m^2+(j-1)m+k \tag{4.28}$$

These data serve the same role as those within the configuration subsets. The multiplier γ_l is used in the control law to modify the velocity feedback gain of the link while the limit Γ_l sets a lower bound on γ_l in order to prevent an overshoot condition from arising.

At the end of each trajectory segment, the existence of overshoot at any point within the segment is determined in the same manner as in the configuration zone learning process. The learning control laws applied to the data velocity subset which is a member of the active set for the segment are similar to those used in the configuration data subsets.

$$\gamma_l = \begin{cases} 1-\nabla_1 & \textit{for } O_l \textit{ false} \\ 1+\nabla_2 & \textit{for } O_l \textit{ true} \end{cases} \tag{4.29}$$

$$\Gamma_l = \gamma_l \quad \textit{for } O_l \textit{ true}$$

The new controller equation which includes multipliers for both configuration and configuration-velocity zones is given as follows.

$$u(t) = K_p[\theta_d(t)-\theta(t)] - K_v\dot{\theta}(t)\,\lambda_l\,\gamma_l \tag{4.30}$$

4.4.3 Practical Realization of the Configuration-Velocity Zone Learning Method

The implementation of the configuration-velocity zone approach on existing hardware parallels the implementation of the configuration zones approach. However, the application is more complex. The use of global velocity zones consists of partitioning the velocities independent of link position. This means that the value in a specific data base velocity zone will be used for all configurations of the robot manipulator provided that the combination of velocities are the same. This method is simple to implement but cannot be used since the forces induced by the motion of the multiple links are functions of link positions as well as link velocities (Equations 3.4 and 3.5). This dictates that there should exist one complete set of V_{ijk} subsets within each configuration zone S_i subset. This greatly increases the number of subsets and elements in the learning data set. The learning is therefore much slower since many more trials are required to explore all the possible combinations which make up the configuration-velocity zone space.

CHAPTER 5

ASSESSMENT OF THE PROPOSED LEARNING METHOD

Computer simulations were used to assess the performance of the proposed learning system for the robot joint controllers. The simulated robot has three revolute links. The equations of motion and their derivation for the simulated robot are given in Appendix A.

The effectiveness of the proposed learning method was evaluated using two basic methods. One method is to compare the system's step input response before and after the implementation of the learning procedure. The principal method of evaluation, however, consisted of measuring the cumulative position error during the execution of several test trajectories. Several different strategies for the learning method were evaluated as was the effect of manipulator velocity on the position errors and the learning process.

5.1 Single Point Step Input

One of the most basic performance measures of a controlled dynamic system is its step response. The step response measure is important in this work since robot trajectories are composed of series of step inputs. The procedure for these tests is the following:

- The response to a step input of the original conventionally tuned fixed gain manipulator system is measured and recorded.

- A learning procedure is implemented which modifies the values in the configuration zones data sets (λ_i and Λ_i) based on the results of a series of trials consisting of randomly generated paths.
- The response of the manipulator, whose control algorithm has been modified to include the learned values of the data set, is again recorded for the same step input.
- The two responses are then plotted versus the same time scale to highlight any differences.

Four case were tested: two for the motion of link 1 and two for the motion of link 2. All cases represent motion of a link in a less than maximum inertia manipulator configuration. In these cases the original fixed gain system is overdamped. However, the system using the learned gain multiplier values (λ_i) should be able to maintain near critical damping and respond faster.

5.2.1 Trajectory Description

A continuous trajectory cannot be defined for a robot whose joints are controlled proportionally with velocity feedback. Typically, a rudimentary trajectory following system is implemented by creating an input file of time driven step inputs (Table 3.2). The file contains a list of positions for each link and times at which these positions are fed to the link controllers for execution. The process is controlled by the robot controller.

For the purpose of this work, the end point (or stop point) in a trajectory is treated as a special case. At such a point the effective damping of the joint is increased to ensure that no overshoot occurs. The damping for all the intermediate points for which the manipulator is not

required to stop can be maintained low provided that the link does not overshoot a point before the start of the next corresponding time segment. For a given manipulator operating velocity, this approach will allow a faster response and a reduction in position error for the intermediate points.

5.2.2 Trajectory Error Assessment

In order to evaluate the performance of the learning system, an objective evaluation criteria has been established. At each time entry in the series defining the test trajectory the Cartesian error is determined as the vector length between the actual position of the end effector reference point and the desired position specified in the given trajectory list. The cumulative error vector length β is defined as the sum of the position errors (in cm) over the entire test trajectory.

$$\beta = \sum_{j=1}^{m} \sqrt{(x_j - x_j^d)^2 + (y_j - y_j^d)^2 + (z_j - z_j^d)^2} \qquad (5.1)$$

5.2.3 Test Paths

The performance of the proposed learning system is evaluated using realistic trajectories for the motion of the manipulator. Five test paths have been defined throughout the robot's work volume and arbitrarily labelled 1 to 5. These are used to assess the learning process. Test paths 1 through 4 are Cartesian straight lines. However, the joint motions have some stop points and

changes of direction. Test path five consists of an oblique rectangle. This path forces the manipulator arm to move through a wide range of positions and velocities. The joint angles are plotted versus time for the five test paths in Figures 5.1 through 5.5. These are used to generate trajectories at different velocities for all the different applications of the zone learning method (repetitive learning, random learning using configuration zones, and random learning using configuration-velocity zones) and for all the sizes of data sets.

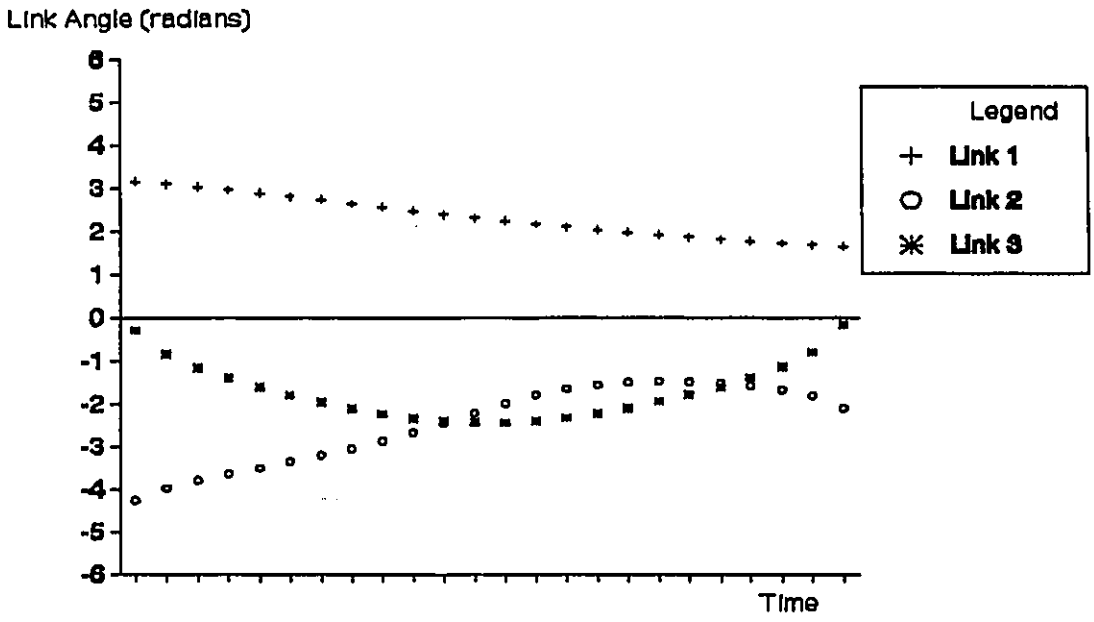


Figure 5.1 Test Path 1

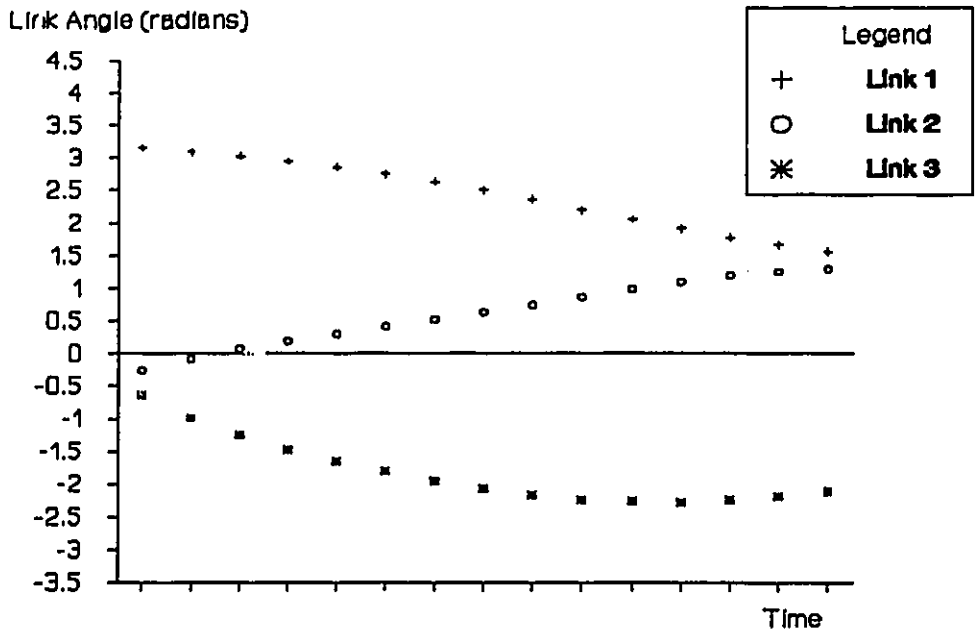


Figure 5.2 Test Path 2

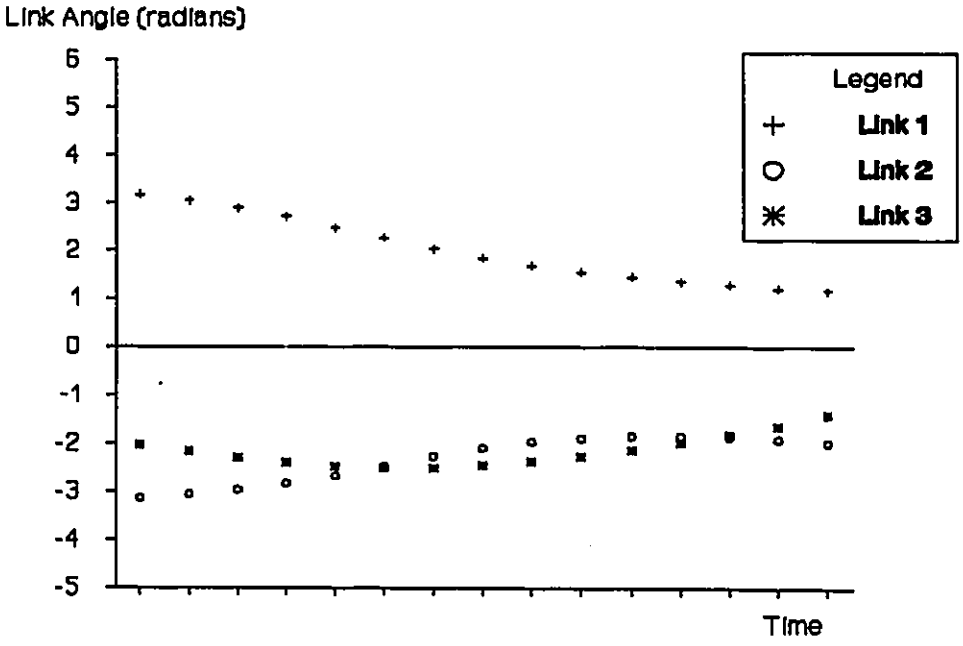


Figure 5.3 Test Path 3

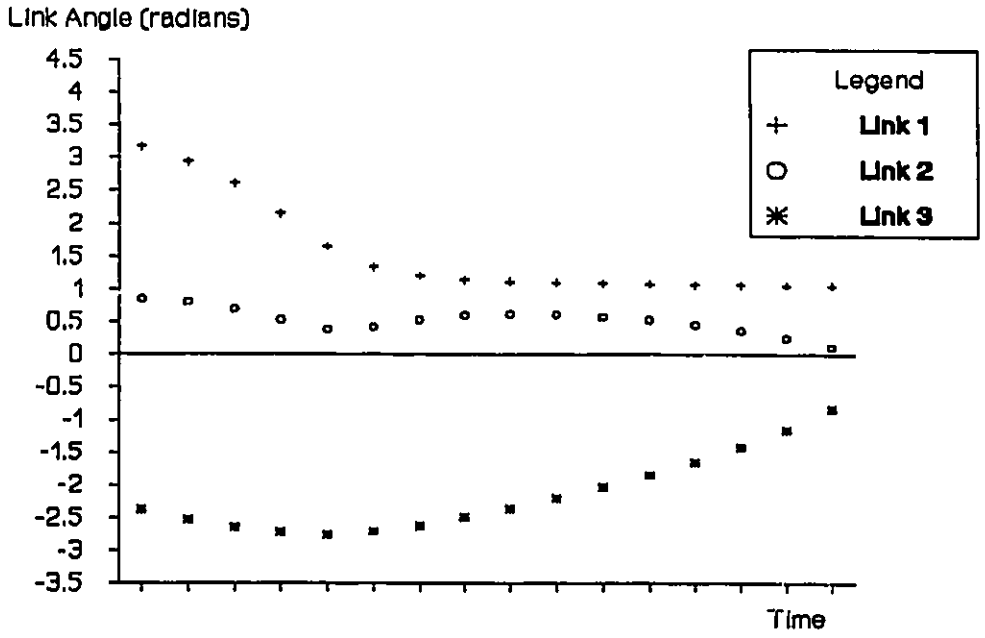


Figure 5.4 Test Path 4

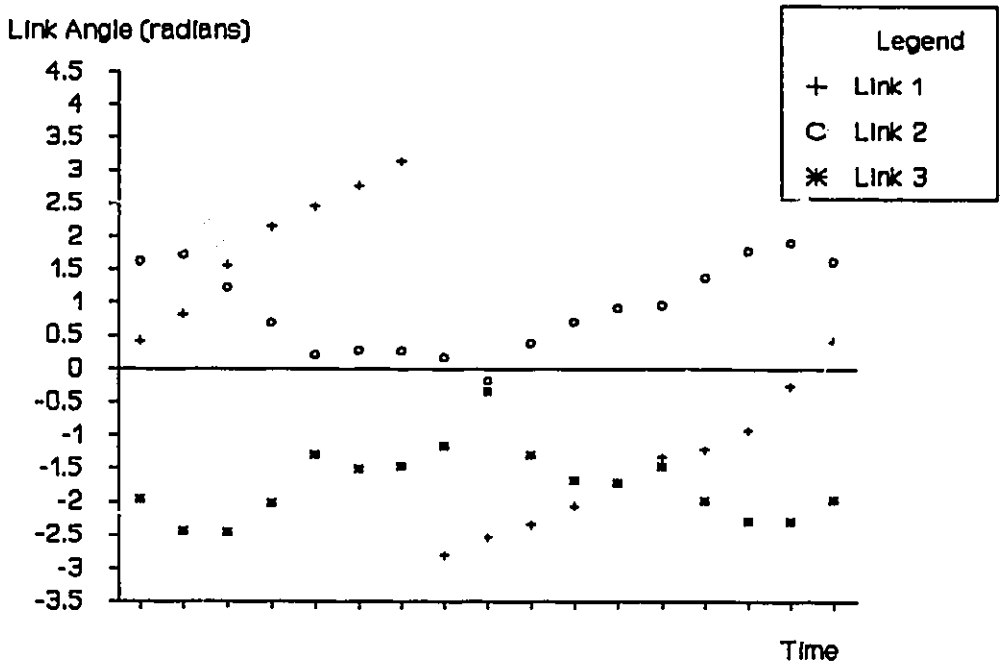


Figure 5.5 Test Path 5

5.3 Effects of Velocity

The "operating velocity" of the manipulator generally refers to the Cartesian velocity of the end effector. This velocity has a major effect on the position error over the course of a given path. This is because, depending on the specified operating velocity, a link may not have enough time to reach the desired position before the next reference position is passed to its controller by the robot controller. To investigate the effect of velocity on the position error, a series of trajectories were executed over test path 5 with fixed controller gains. The results, plotted in Figure 5.6, show the effect of increased operating velocity on the value of the objective function β. The figure shows that in general good position accuracy can be achieved for trajectories defined with a slow Cartesian velocity but errors increase as the velocity increases. The evaluation tests of all the different strategies were conducted at two operating velocities: 60 cm/s and 90 cm/s.

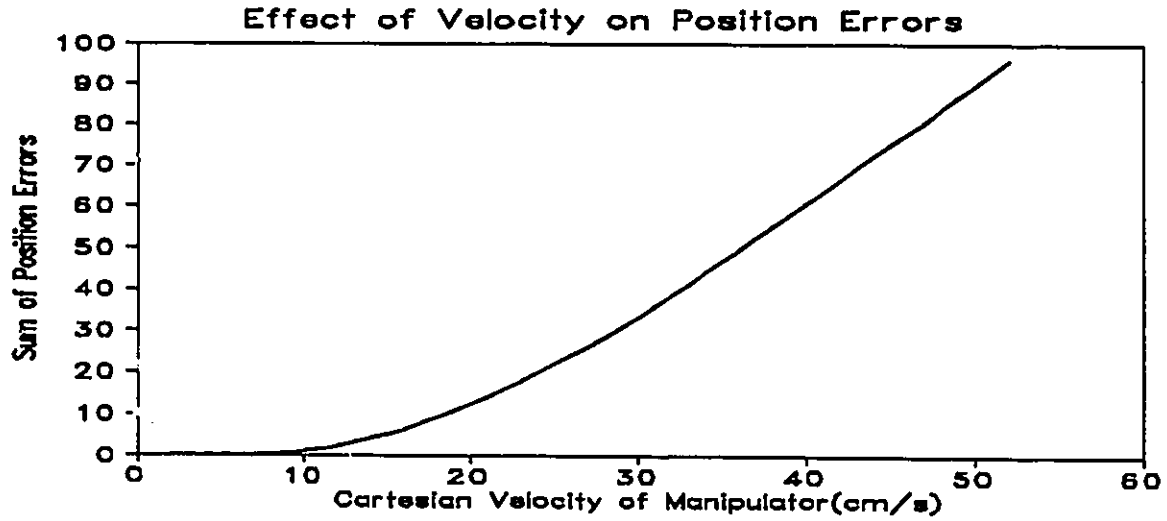


Figure 5.6 Effect of Velocity on Position Errors

5.4 Different Strategies for the Zone Learning Method

Several strategies of learning were used with the zone learning method. These strategies are described below. The results obtained using these strategies will be presented in Chapter 6.

5.4.1 Repetitive Learning

Repetitive learning is achieved by continuously repeating a specific trajectory at the same operating velocity thereby recreating almost identical conditions for the manipulator. At every iteration, parameters are adjusted based on the results of the previous trials in an attempt to improve the performance. Several ways of applying this approach to robot motion control applications have already been reported and briefly discussed in Chapter 2.

For the proposed zone learning method the configuration zone data set values (λ_i and Λ_i) are modified with every repeat of the test trajectory until optimum values have been reached which respect the no overshoot constraint. This procedure updates only the data values corresponding to those zones through which the test trajectory passes. This learning process is therefore specific to each particular test trajectory. If the goal is to optimize one specific trajectory, this strategy will converge quickly since the same conditions (positions, velocities, etc) are encountered at every iteration. However, a new learning process is required whenever a different trajectory is to be executed.

5.4.2 Random Configuration Learning

This learning is nonrepetitive and tests the ability of the learning method to transfer knowledge from one trajectory to another. To conduct random learning, a computer program was written to randomly generate trajectories, given a specified Cartesian operating velocity. The trajectories contain from 14 to 24 points (no more than 10 cm apart) through which the manipulator end effector reference point must pass. This Cartesian trajectory is then translated into joint coordinates and is presented to the robot simulator as a series of time coordinated step inputs for each joint. The following procedure was employed to evaluate the performance of the random configuration learning method.

1 - One initial trial of the test trajectory was executed using the simulator in the fixed gain mode. The value of the objective function is recorded.

2 - A number of randomly generated trajectories are executed during which learning occurs. This learning consists of modifying the multiplier λ_i and limit Λ_i values in the configuration zone data sets according to the procedures outlined in Chapter 4.

3 - The test trajectory is executed using the modified values of λ_i and Λ_i in the zone data sets to modify the velocity feedback gain whenever the test trajectory ventures into the corresponding zones. No knowledge is obtained from the test trajectory. Again the value of the objective function is recorded.

4 - Steps 2 and 3 are continually repeated until all the zones have been explored several times and convergence of the objective function occurs.

5.4.3 Configuration-Velocity Learning (Direction)

The application of the configuration-velocity zone learning process (refered to as simply "velocity learning" for brevity) occurs only after the completion of the random configuration learning process. This is because the multiplier values from the configuration data sets λ_i are used to modify the feedback velocity gain along with the multiplier values from the configuration-velocity data sets γ_i. The velocity learning approach uses the random (non-repetitive) strategy similar to that described for the random configuration learning in Section 5.4.2 with the exception that the configuration-velocity set data values, multiplier γ_i and limit Γ_i, are modified and not those pertaining to the configuration data sets.

In the velocity direction application of the learning method, each of the link velocities is partitioned into only two zones: a positive direction zone and a negative direction zone. This means that each configuration zone must contain a set of 2^i configuration-velocity zones where i is the number of links. In this work i=3, thus 8 configuration-velocity zones are required for each configuration zone. If the robot work volume is partitioned into a large number of configuration zones (n) then a large number of configuration-velocity zones will be required. Link 1 requires $8n^2$ configuration-velocity zones while links 2 and 3 only require $8n$ zones.

5.4.4 Configuration-Velocity Learning (4 Zones)

This approach ("velocity learning") is basically similar to the velocity direction learning approach except that the link velocities have been partitioned into 4 zones rather than just 2. Since 4^n zones are required for a full set of configuration-velocity zone combinations, the number of data sets required for this approach can be very large if n is large.

5.4.5 Velocity Learning (Quasi-Static Base)

The learning approaches thus far presented are all conducted at a specific manipulator operating velocity. This is true for both the configuration and configuration-velocity zone learning. This results in data set values which are specific to a particular manipulator operating velocity. Therefore a complete data set must be built and stored for each of the normal manipulator operating velocities. This may require a lengthy training period and large memory capacity.

An alternative approach in which one complete data set may be applied to a wide range of manipulator operating velocities is the "quasi-static" approach. This approach has two steps to its application. The initial learning is based on configuration zones and conducted at a very low manipulator Cartesian velocity. A speed of 10 cm/s was used in this work. The values for the configuration data sets are compiled at this slow velocity. The position errors will be small at this velocity and the no overshoot constraint is respected since the manipulator almost stops

at each reference point. This learning is the base upon which the velocity learning is to be conducted.

Once the configuration learning is complete, the manipulator is nearly critically damped in all zones. However, for higher desired Cartesian operating velocities, the response will be too slow and errors will be large. The second step, velocity overlay learning, provides additional multipliers to adjust the velocity gains based on the joint velocities. The learning process consists of executing random test trajectories at different operating velocities so that a wide range of joint velocities are incurred. In this work the learning was conducted by generating the random trajectories at alternatively 30 cm/s, 60 cm/s and 90 cm/s. To evaluate the process both operating velocities of 60 cm/s and 90 cm/s were tested using the unique set of configuration-velocity multipliers which were compiled by this learning strategy. The results are shown in Chapter 6.

5.4.6 Random Configuration Learning (with Stop Point Isolation)

For all the above methods, the end points of the trajectories were treated as special cases allowing sufficient damping to be applied to prevent overshoot. However, even in a straight Cartesian line trajectory, one or more of the joints may have to stop or change directions. These intermediate stop points were not isolated. They were considered in a similar manner as those for which the link must pass through. However, this may result in difficulties, especially for certain trajectories. Therefore, the configuration learning method was amended to isolate all the

joint stop points and impose sufficient damping to avoid overshoot. In addition, the learning process was disabled during those segments as to not detrimentally influence the values of the data set variables.

5.4.7 Configuration-Velocity Learning (4 zones with Stop Point Isolation)

This method consists of applying the stop point isolation technique (Section 5.3.6) to the velocity learning with 4 zones. The results are presented in Chapter 6.

CHAPTER 6

TEST RESULTS OF THE LEARNING PROCEDURE

This chapter will detail the results of the evaluation tests carried out using the simulation of the 3 DOF revolute robot.

6.1 Step Input Results

As outlined in Chapter 5 several means were used to measure the effectiveness of the learning method. The most basic test consisted of recording the step response of the robot links using first the conventional fixed gain controller and then using the configuration zone data base. Four test cases were used, involving rotations of links 1 and 2. The test cases represented motions where the moment of inertia of the link about the pivot point is less than its maximum possible value.

Case 1: Link 1 was commanded to rotate from 0 to 90 degrees while links 2 and 3 remained stationary at respective angles of 0 degrees and -90 degrees. (Figure 6.1)

Case 2: Link 1 was again commanded to rotate from 0 to 90 degrees while the positions of links 2 and 3 remained fixed, both at -45 degrees. (Figure 6.2)

Case 3: Link 2 was commanded to rotate from +45 to -45 degrees while link 3 remained fixed at 90 degrees. The position of link 1 has no bearing on the result provided that it is constant. (Figure 6.3)

Case 4: Link 2 was commanded to rotate from -45 to +45 degrees while link 3 remained fixed at -135 degrees. The position of link 1 has no bearing on the result. (Figure 6.4)

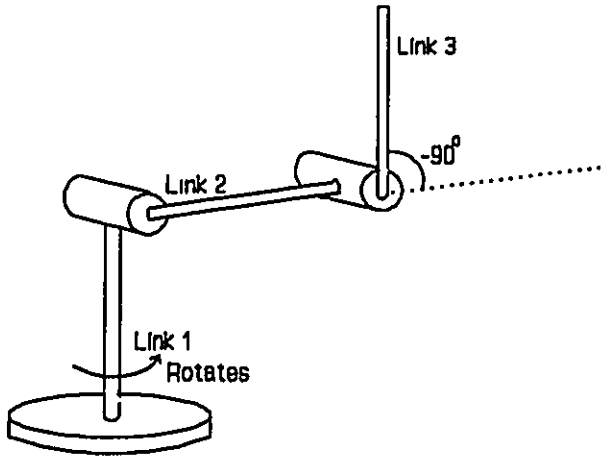


Figure 6.1 Unit Step Test 1

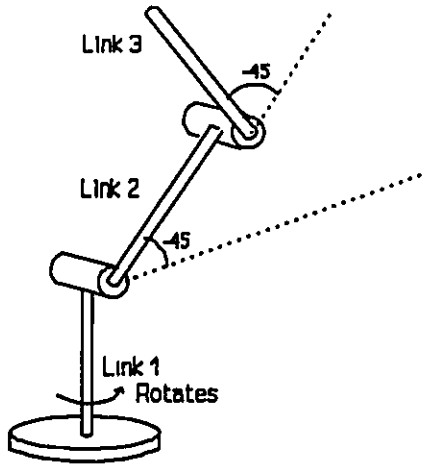


Figure 6.2 Unit Step Test 2

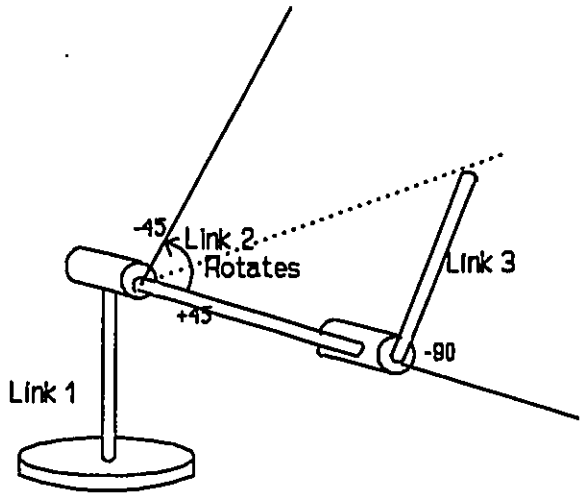


Figure 6.3 Unit Step Test 3

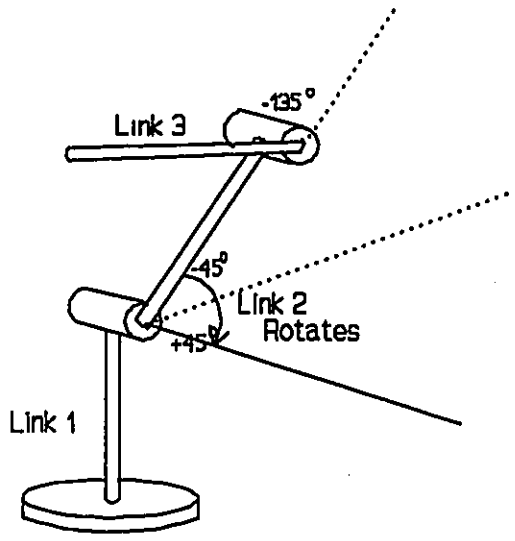


Figure 6.4 Unit Step Test 4

The responses of both the conventionally controlled system and the system whose controller was modified using the configuration zone multipliers λ_i are plotted versus time in Figures 6.5 to 6.8. A configuration zone size of 30 was used for these tests.

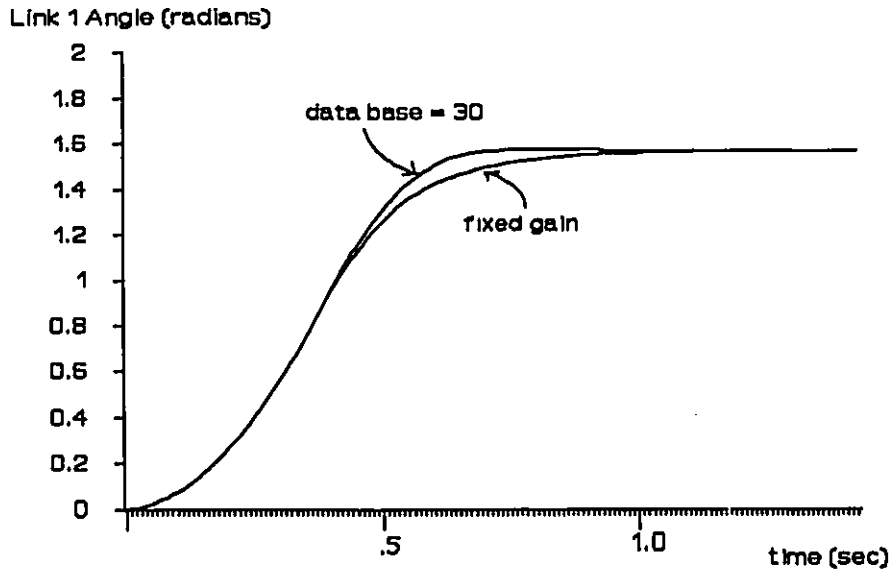


Figure 6.5 Link 1 Unit Step Response (Case 1)

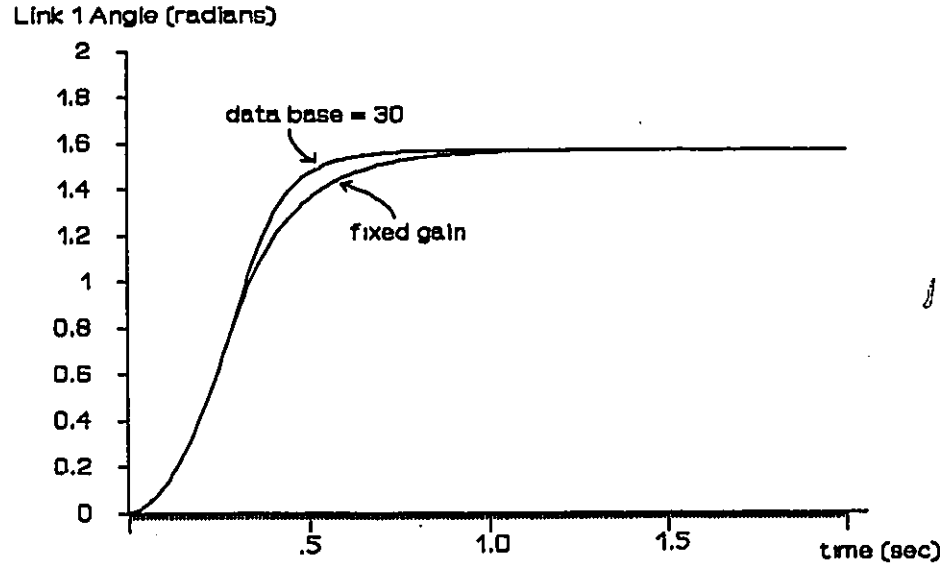


Figure 6.6 Link 1 Unit Step Response (Case 2)

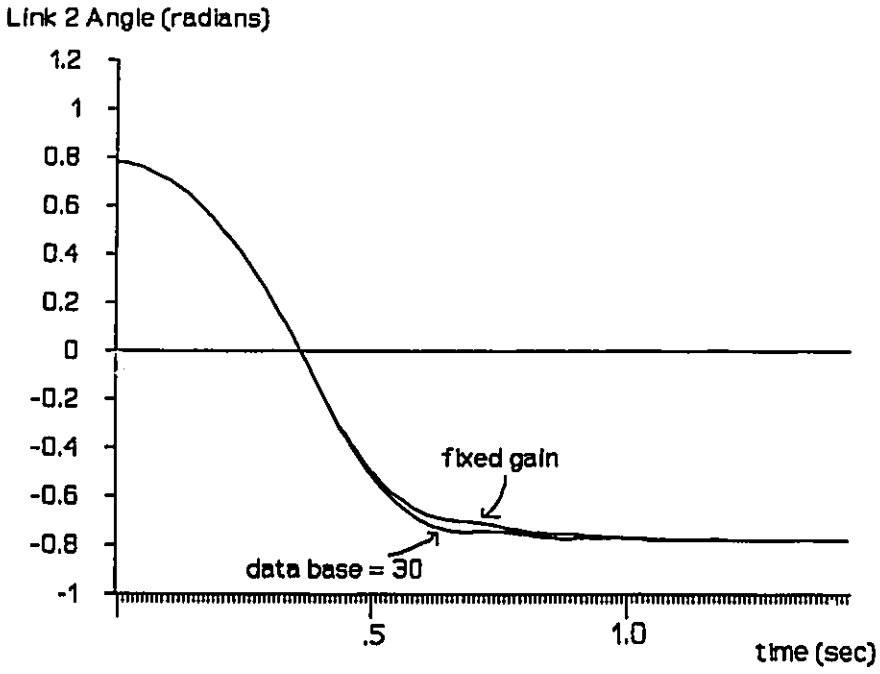


Figure 6.7 Link 2 Unit Step Response (Case 3)

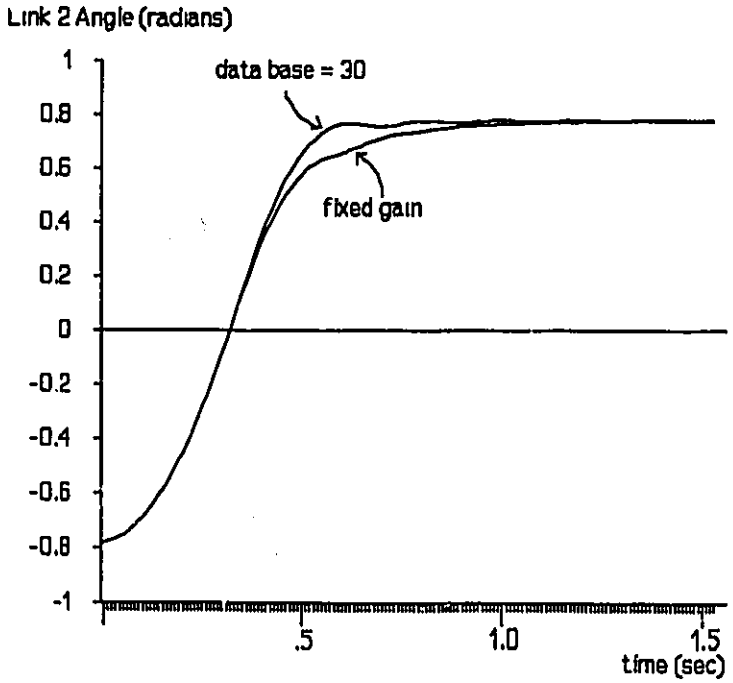


Figure 6.8 Link 2 Unit Step Response (Case 4)

Table 6.1 summarizes the results of the step input tests by comparing the time required for the link to reach the desired output for both the conventional controller and the modified controller. In all cases the table shows that the system with the modified controller responded significantly faster than the conventionally controlled system without any over-shoot occurring.

	Conventional Controller	Learning Based Controller	Percentage Improvement
Case 1	1.12 sec	0.72 sec	36 %
Case 2	1.30 sec	0.92 sec	29 %
Case 3	1.02 sec	0.84 sec	18 %
Case 4	1.32 sec	0.96 sec	27 %

Table 6.1 Results of Step Response Tests

6.2 Effect of Subdividing Zones on Multiplier Values

The design of conventional fixed gain link controllers for robot manipulators is based on providing critical damping for the worst condition that will be encountered. This usually corresponds to maximum inertia configurations of the manipulator. For all other configurations the system will be overdamped and thus have a slower response than would be possible if it were

critically damped. However, the damping could be reduced in many parts of the work volume without causing overshoot.

The learning method based on configuration zones seeks to determine the feedback velocity gain which will critically damp the system for the worst case within a certain part of the work volume. This work volume is called the configuration space. The modification of the velocity feedback value is accomplished by the multiplier values λ_i. If the manipulator work volume is partitioned into only one zone, the result is the same as the conventional fixed gain controller. When the volume is partitioned into many zones, the performance of the manipulator will improve since the damping can be reduced in most configuration zones where the inertia is less than maximum.

The multiplier values are a measure of the amount of damping which is required for each configuration zone. For a one zone partition, the value of λ_i should be 1.0. This is because the standard fixed gain is required to cope with the maximum inertia configuration. When the work volume is partitioned into several zones the majority of the zones will have multiplier values less than 1.0. At the limit of the learning process at least one of subzone multipliers will have the value of one. The others will have values less than 1.0.

In the general case if a zone is subdivided and a learning process performed, the majority of the subzone multiplier values will be less than the multiplier for the parent zone. In the limit at least one of the subzone multiplier values will be identical to that of the parent zone. The

other subzone multiplier values will be less than the parent zone multiplier value. It is these lower subzone multiplier values which result in improved manipulator performance.

This effect is demonstrated in Figure 6.9. Random configuration learning was conducted at 10 cm/s for different numbers of partitions of the robot s work volume (1, 5, 10, and 20 partitions). The configuration zone multiplier values for link 2 were recorded for each size of partition of the work volume. A pyramid arrangement is used to display the zone multiplier values (1 zone on top, going through 5, 10, and 20 zones on the bottom). For each level of the pyramid of multiplier values, the number of zones is indicated on the far right.

For a single zone the multiplier value is 1.0. All the multiplier values of the five zone data base are less than one, although two of them are .99. In the limit of learning, at least one of these should be 1.0. Each multiplier pair of the 10 zone data base corresponds to only one multiplier value of the 5 zone data base. At the limit of the learning process, one of the two will have the same value as that of the parent zone. The other multiplier value will generally be less. The same holds true for the relationship between the 10 zone and 20 zone data bases. Since the limit of learning has not yet been reached for the data bases presented in Figure 6.9, the corresponding parent zone/subzone multiplier values are not identical as was predicted. However, the values are close enough to demonstrate the effect.

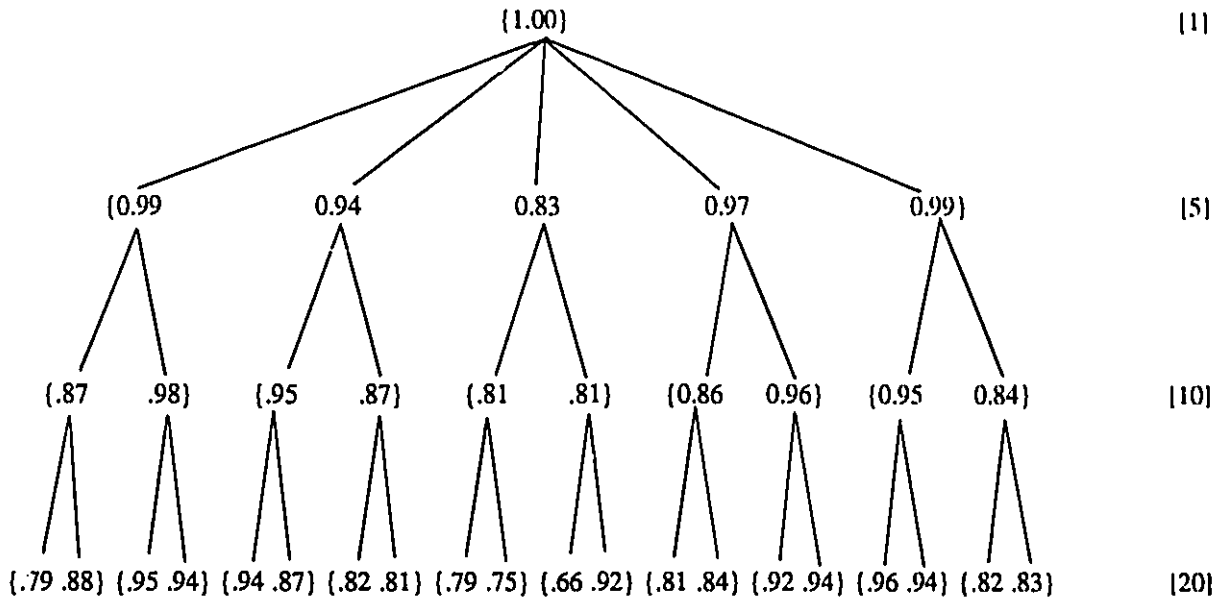


Figure 6.9 Multiplier Values for Different Numbers of Volume Partitions

6.3 Test Results of the Various Learning Strategies

Various strategies of learning were tested using the five test trajectories. All the strategies are variations of the zone learning method. The strategies for which results are presented are the following:

- repetitive learning
- random configuration learning
- velocity direction learning
- 4 zone velocity learning
- velocity learning (on a quasi-static base)

- random configuration learning with stop point isolation

- 4 zone velocity learning with stop point isolation

These approaches were all described in Section 5.4. The short form "velocity learning" is used instead of the complete "configuration-velocity learning".

The results of all the learning trials are presented in two forms: a tabular form and a graphical form. The graphical presentation of the results (Figures 6.10 to 6.84) show the trajectory error (objective function value β) plotted versus "series of learning trials". This presents a complete picture of the learning curve as well as the final error values. The figures present the results of all the different learning strategies which have been tested. A separate figure is required for each test trajectory at each operating velocity for each learning strategy. As a result 10 figures (5 trajectories, 2 velocities) are presented for most of the learning strategies. The exception is for random configuration learning where an additional 5 figures are required due to the extra velocity (10 cm/s) tested. The learning curves for each of the 5 sizes of data bases (n = 5,10,15,20,25,30) are presented together on each figure for most cases.

Five tables (Tables 6.1 to 6.5) have been constructed to display the value of the trajectory position error objective function β after a lengthy learning process for each of the learning strategies. Next to the value of the absolute error, in each cell of the table, the percentage improvement from the original fixed gain system is shown in parentheses "( )". If the result of the learning strategy was worse than the original system the percentage value is shown in curly brackets "{ }". Each table is specific to one of the five test paths. In addition, the tables contain the results for different data base sizes and different manipulator operating velocities.

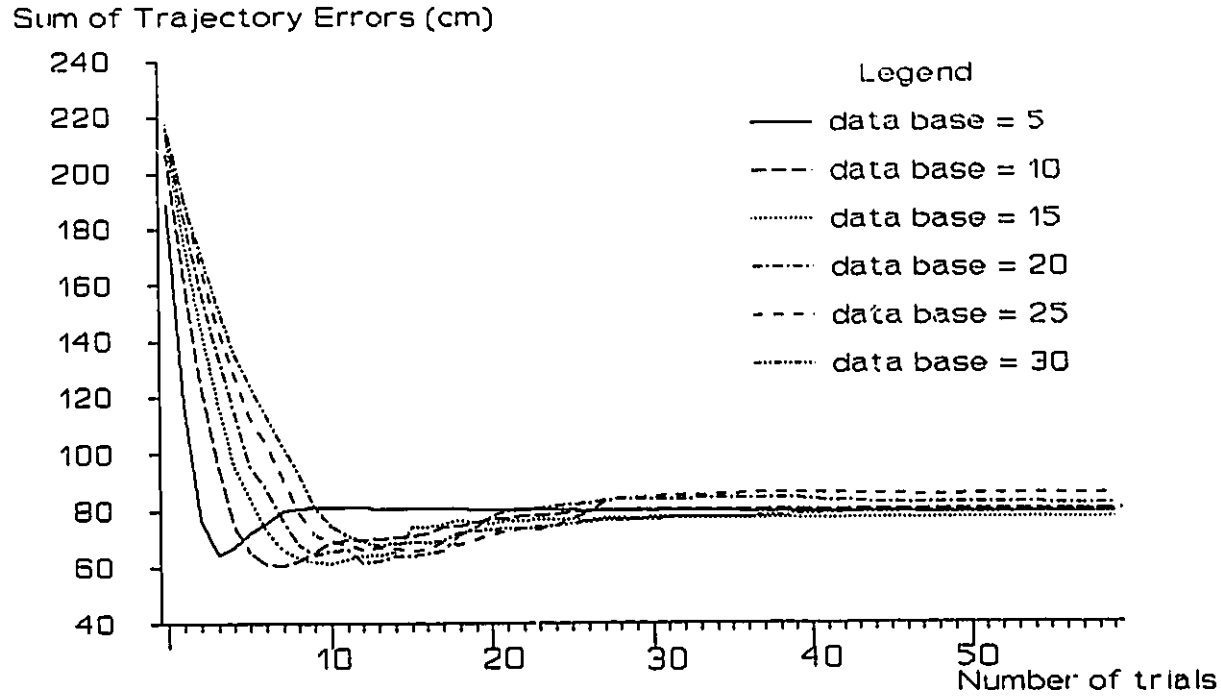


Figure 6.10 Repetitive Learning at 60 cm/s (Test Trajectory 1)

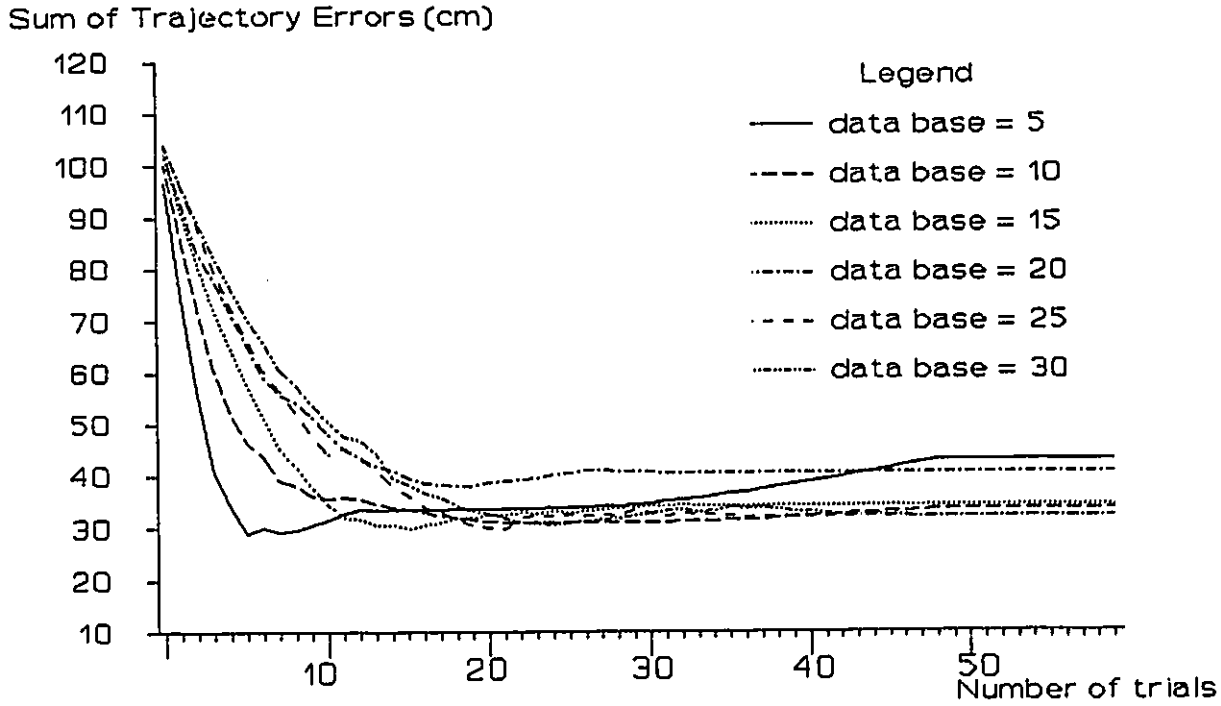


Figure 6.11 Repetitive Learning at 60 cm/sec (Test Trajectory 2)

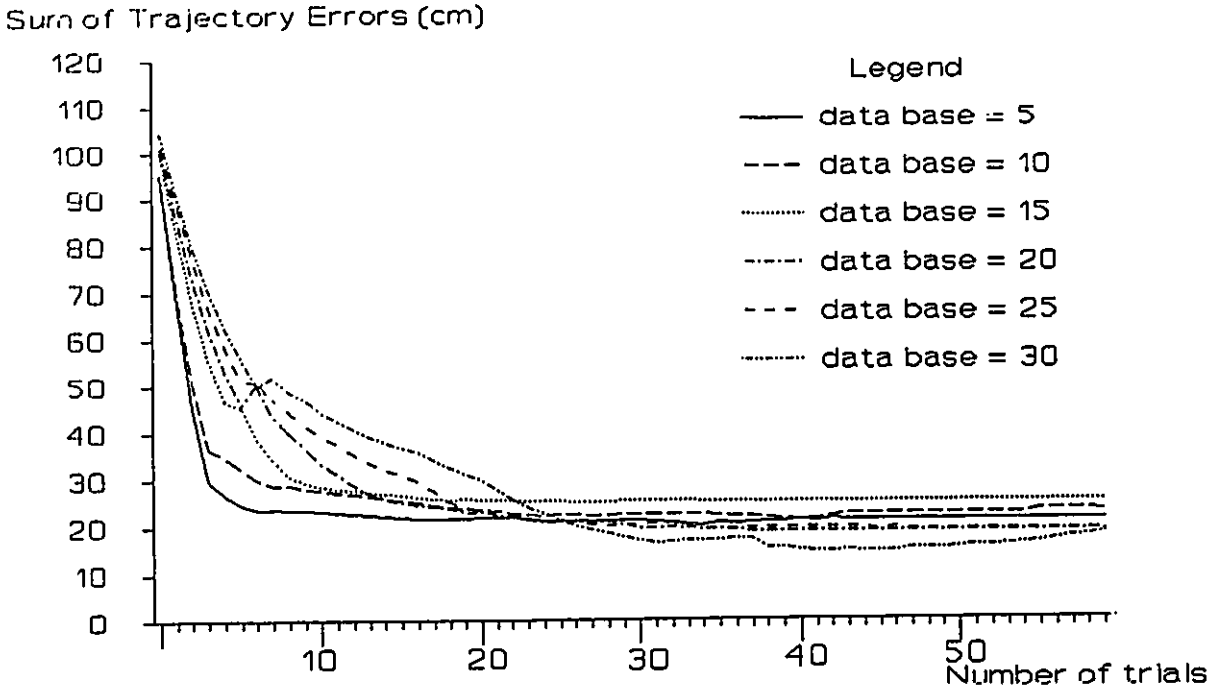


Figure 6.12 Repetitive Learning at 60 cm/s (Test Trajectory 3)

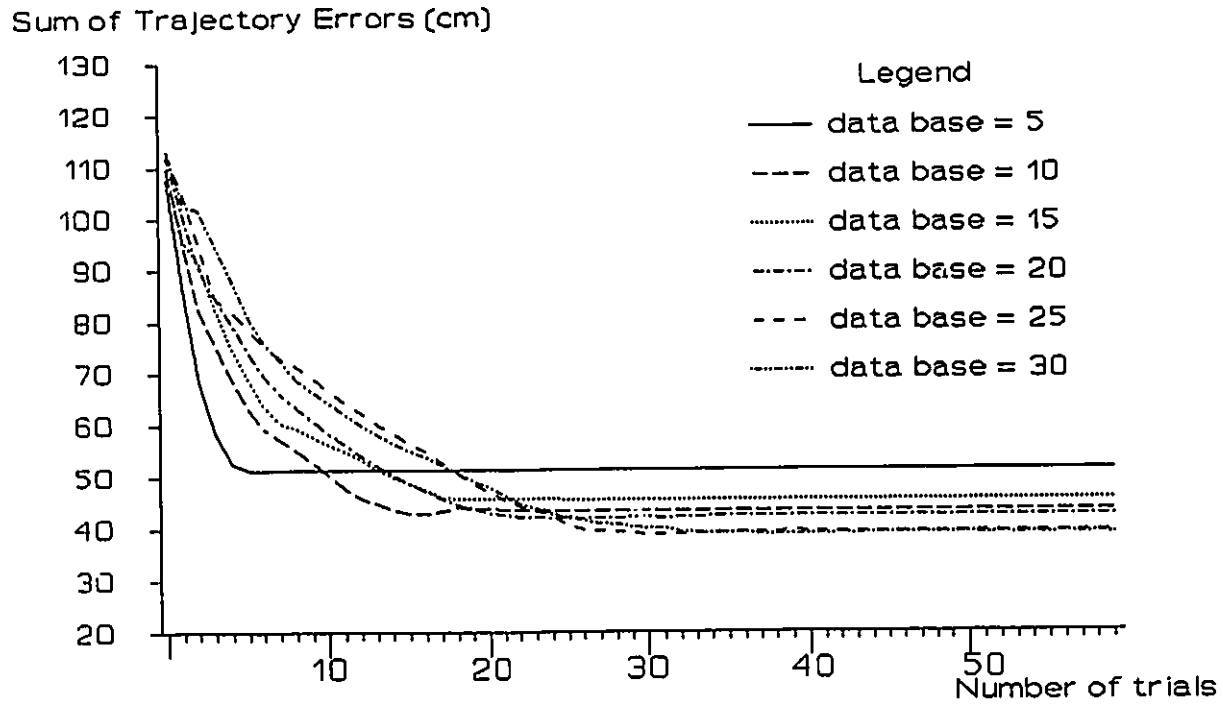


Figure 6.13 Repetitive Learning at 60 cm/s (Test Trajectory 4)

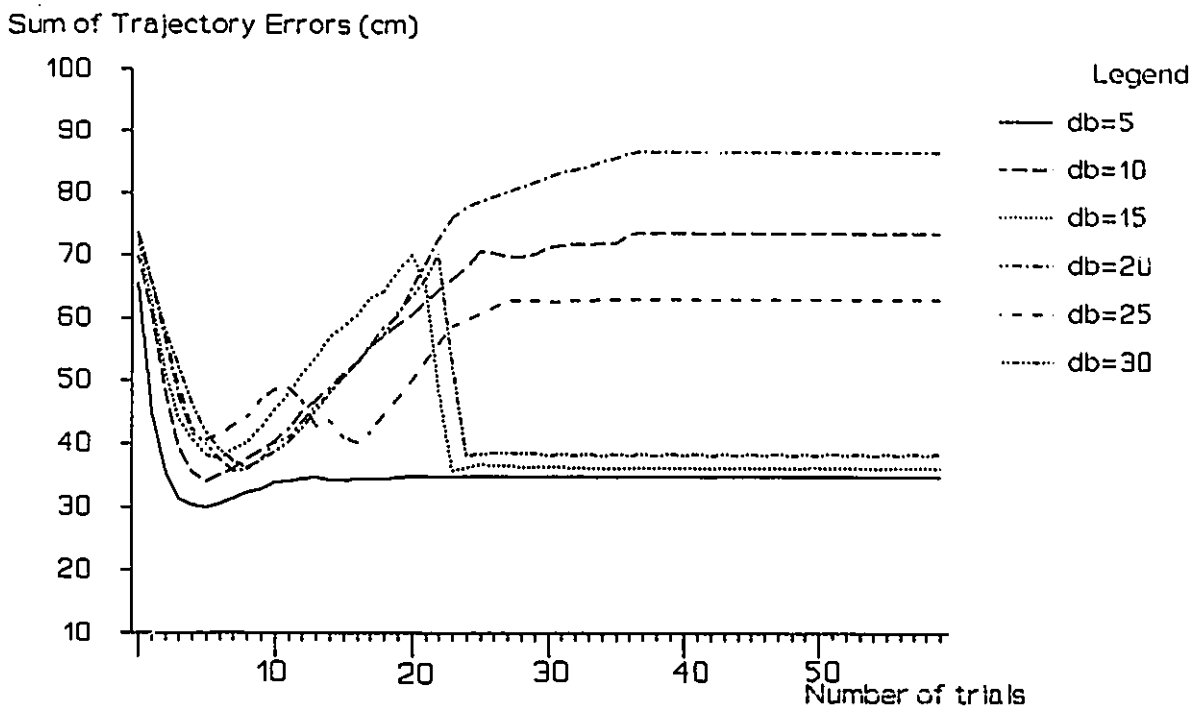


Figure 6.14 Repetitive Learning at 60 cm/s (Test Trajectory 5)

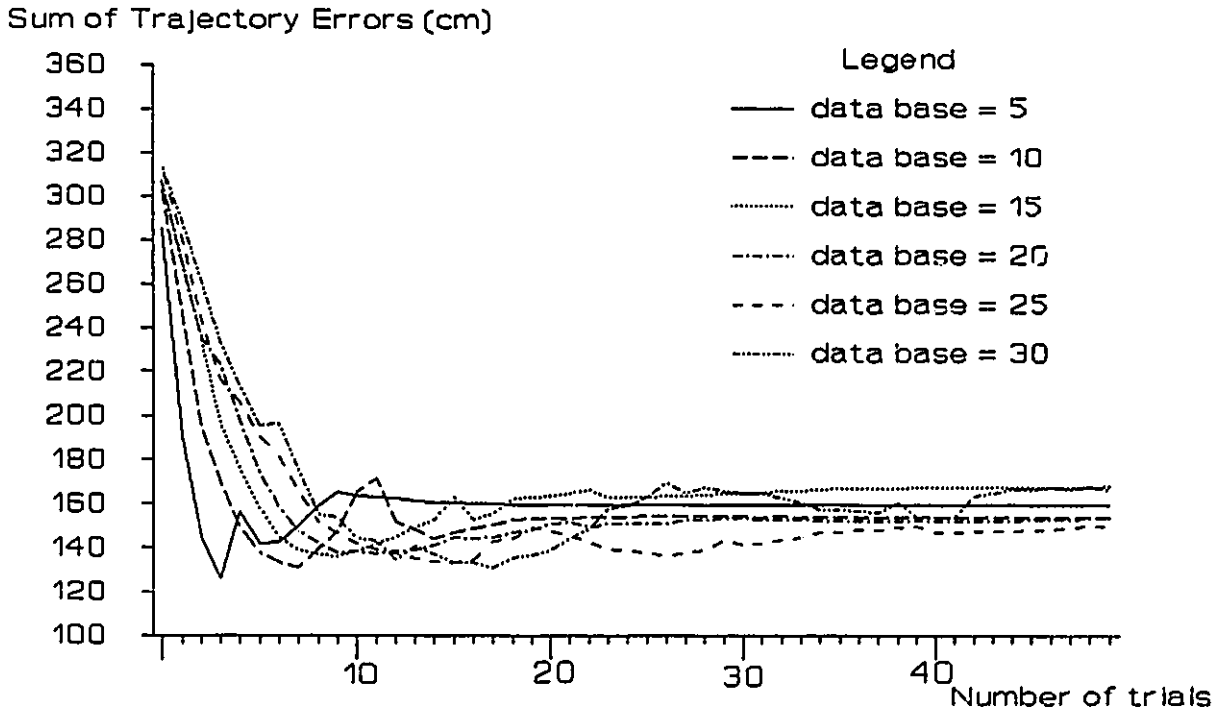


Figure 6.15 Repetitive Learning at 90 cm/s (Test Trajectory 1)

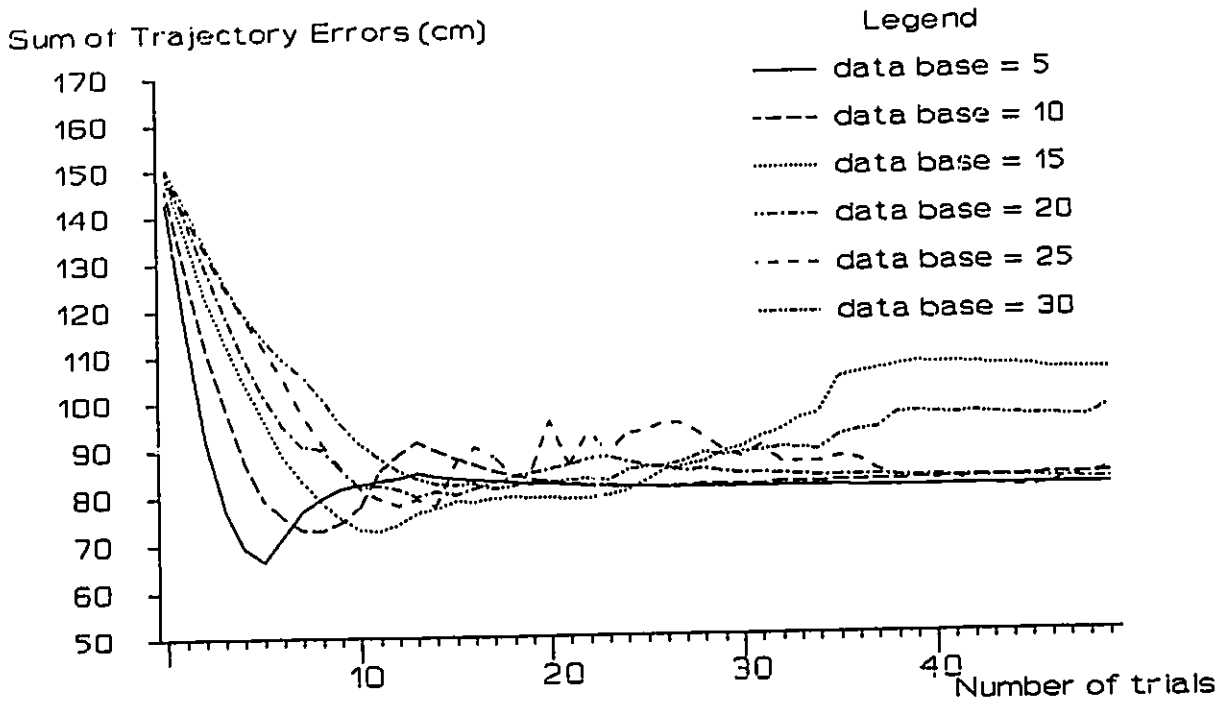


Figure 6.16 Repetitive Learning at 90 cm/sec (Test Trajectory 2)

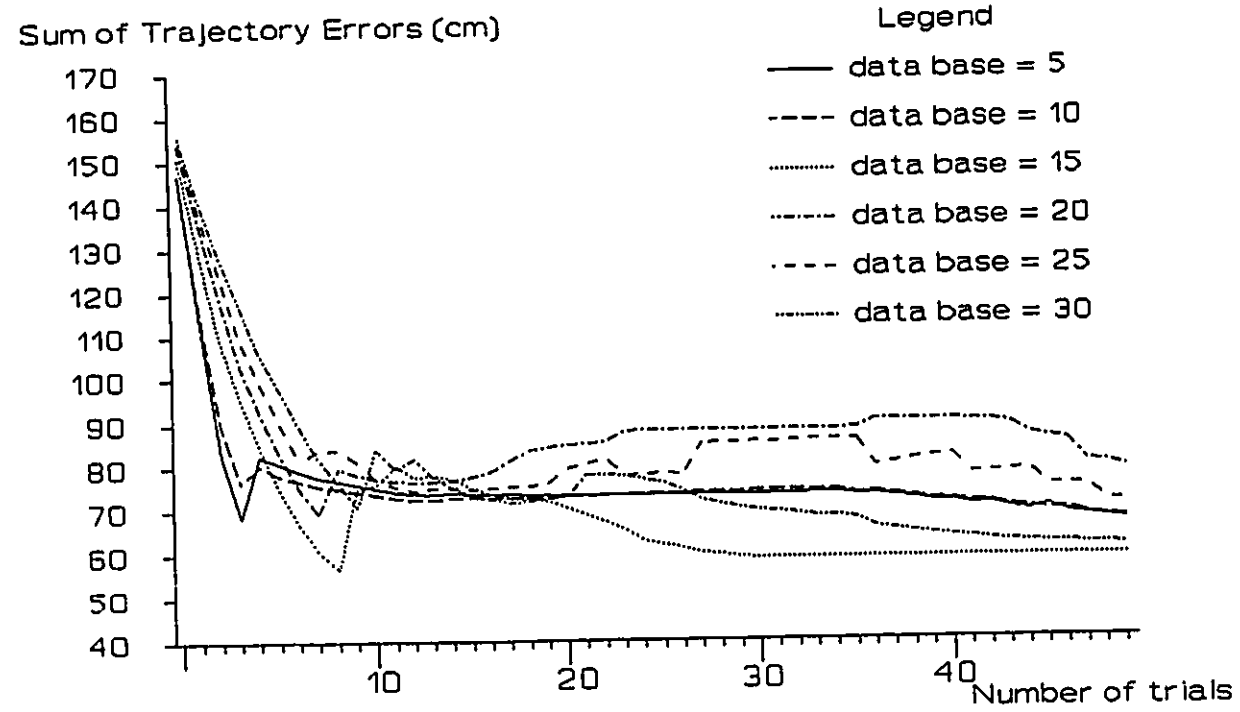


Figure 6.17 Repetitive Learning at 90 cm/s (Test Trajectory 3)

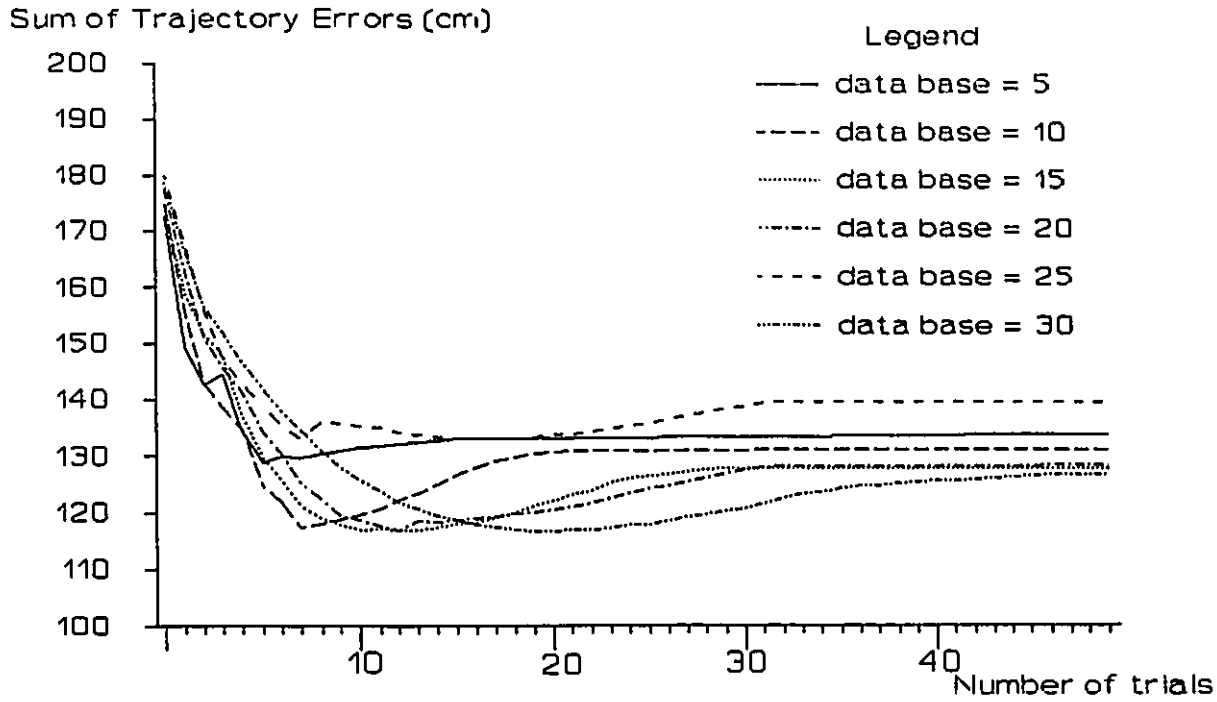


Figure 6.18 Repetitive Learning at 90 cm/s (Test Trajectory 4)

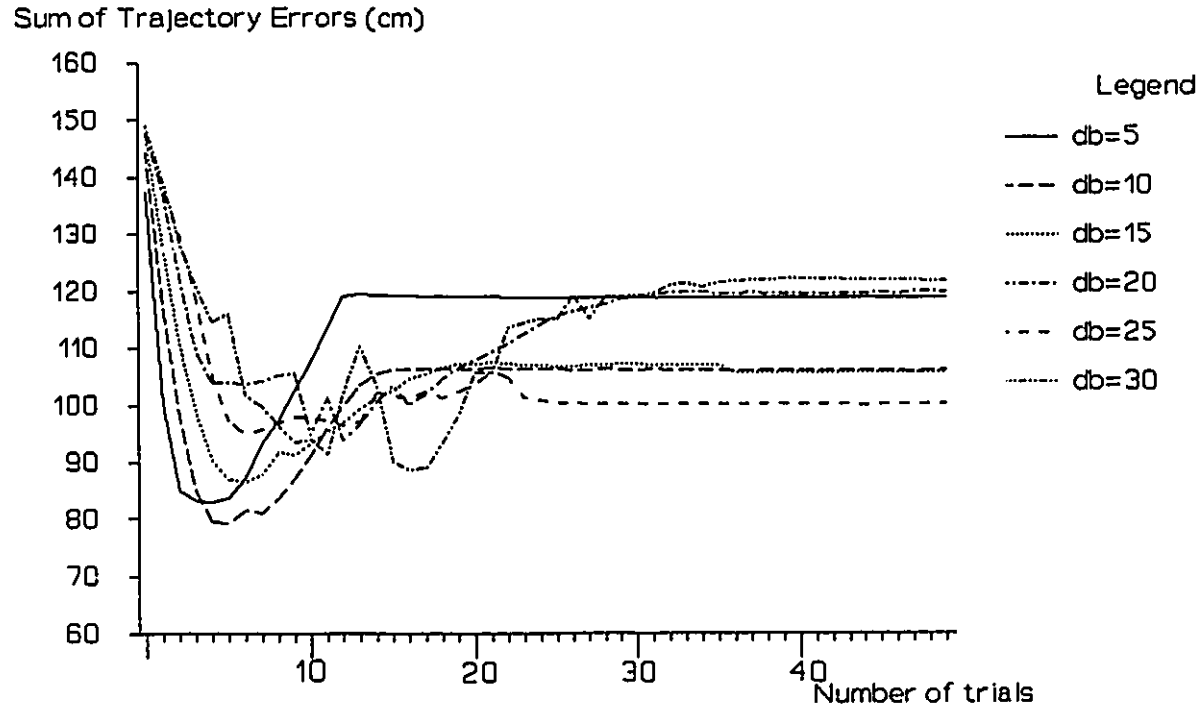


Figure 6.19 Repetitive Learning at 90 cm/s (Test Trajectory 5)

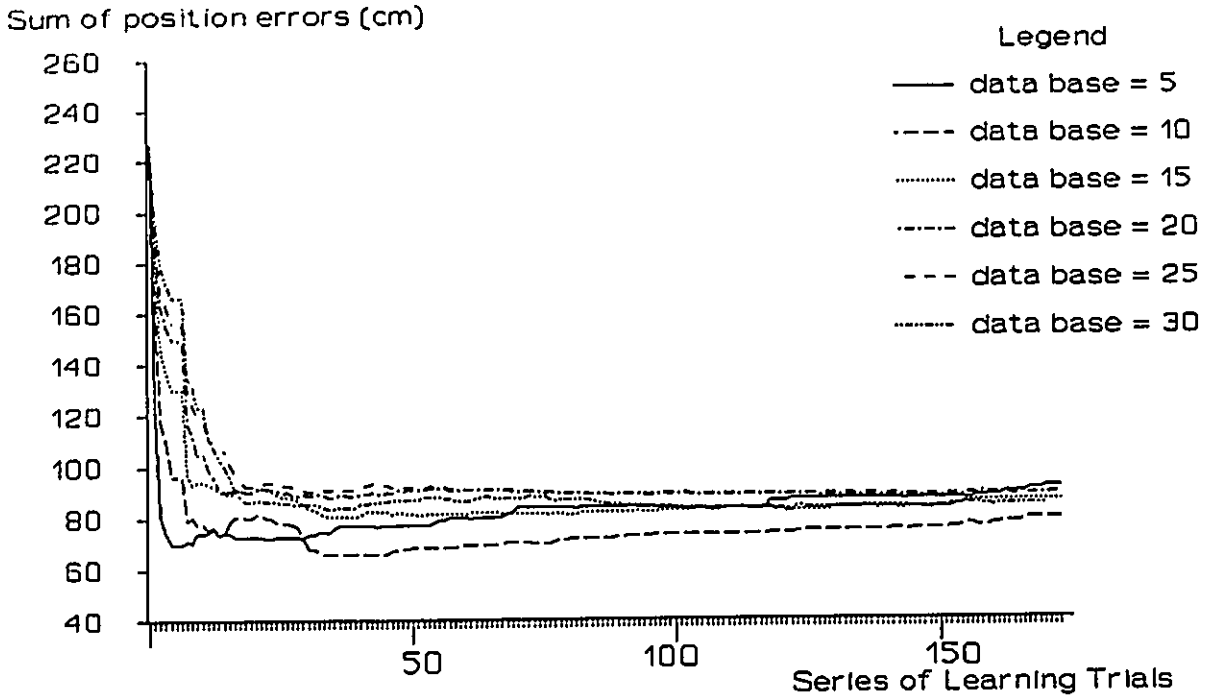


Figure 6.20 Random Configuration Learning at 60 cm/s (Test Trajectory 1)

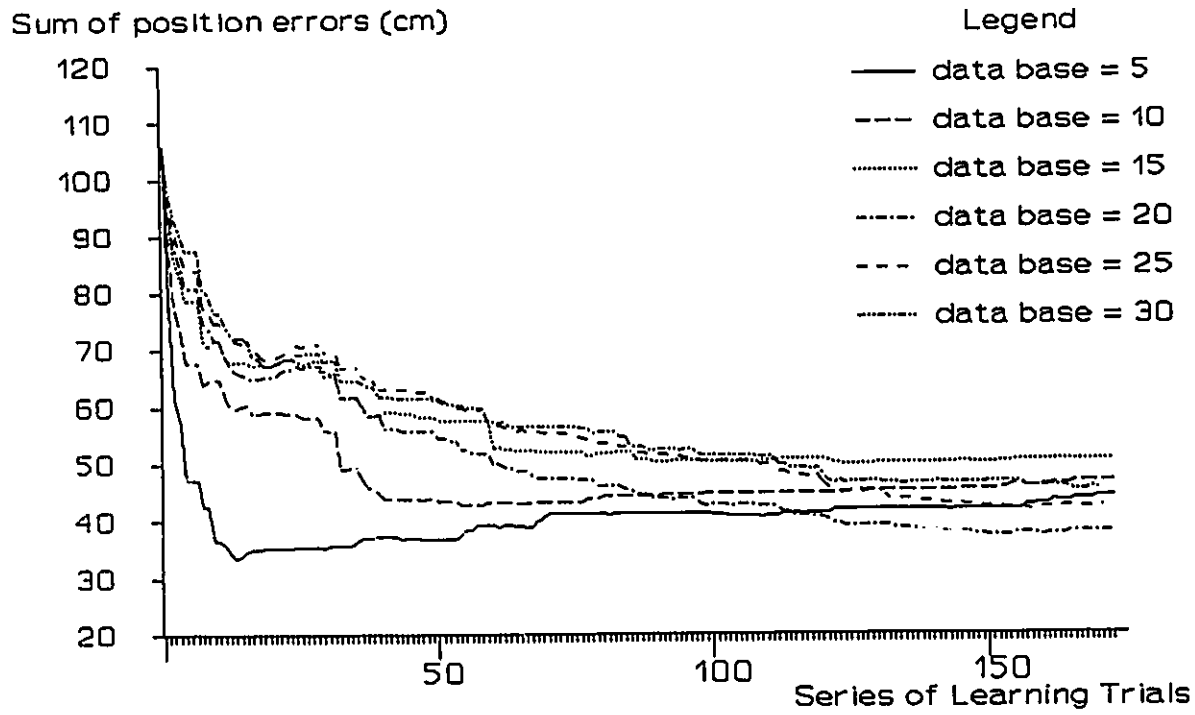


Figure 6.21 Random Configuration Learning at 60 cm/s (Test Trajectory 2)

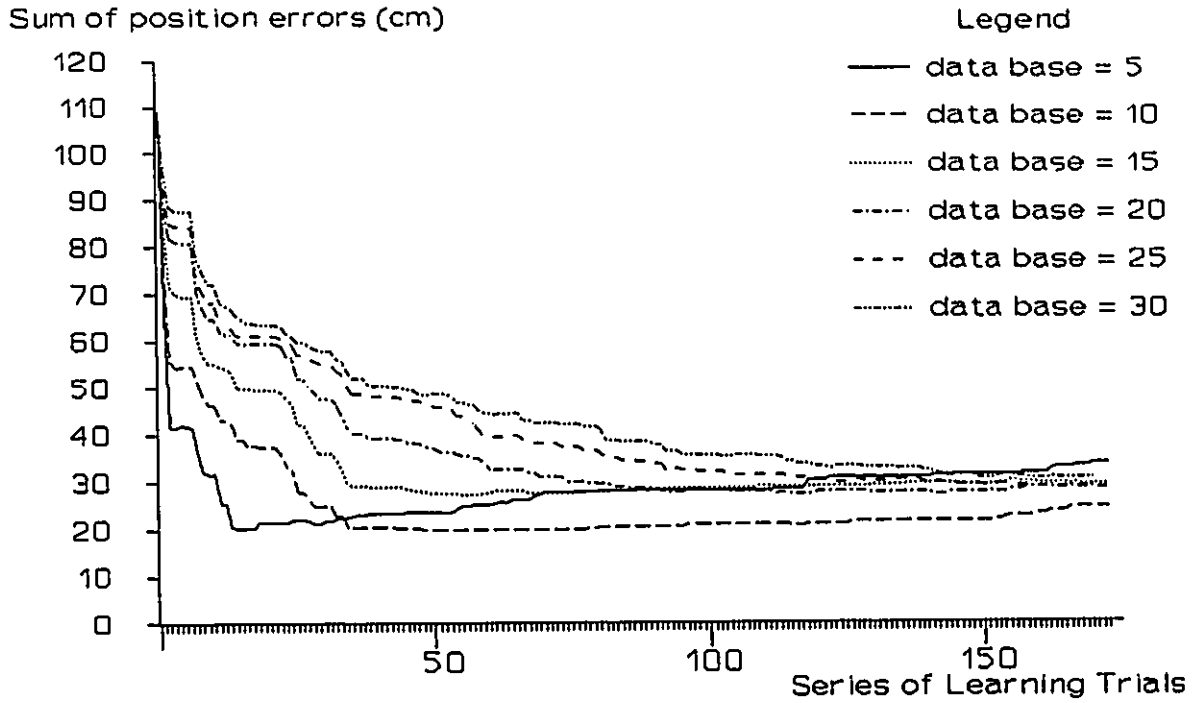


Figure 6.22 Random Configuration Learning at 60 cm/s (Test Trajectory 3)

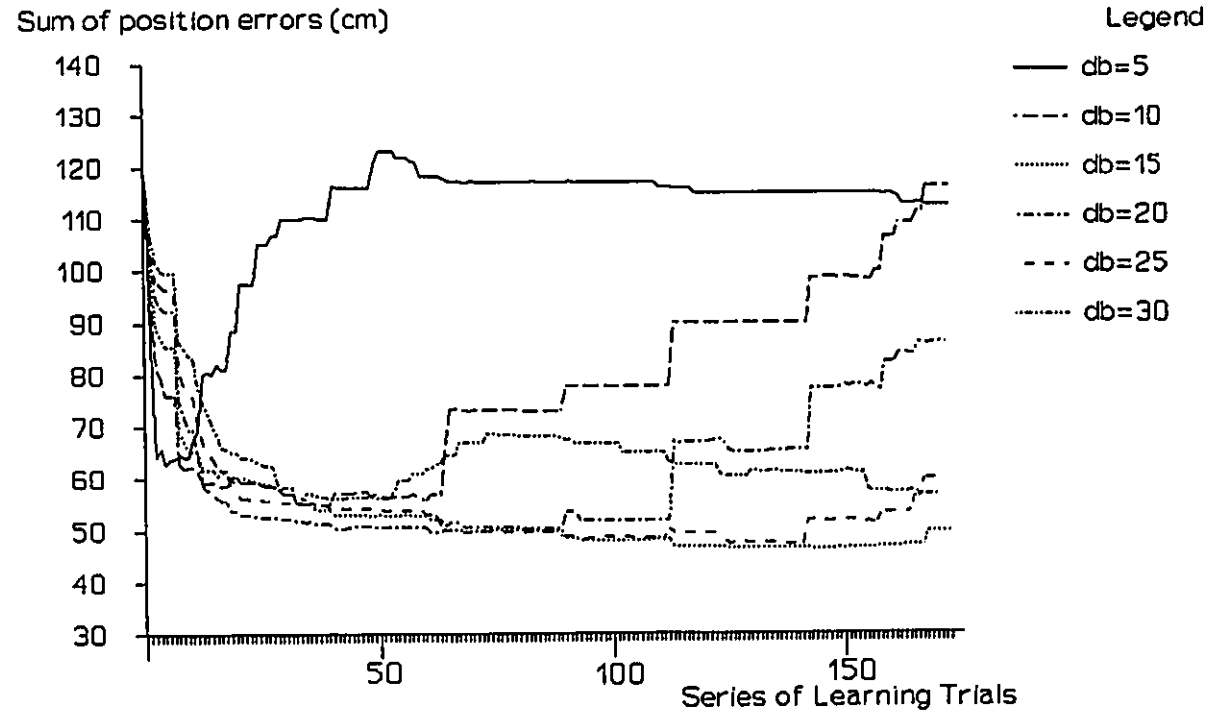


Figure 6.23 Random Configuration Learning at 60 cm/s (Test Trajectory 4)

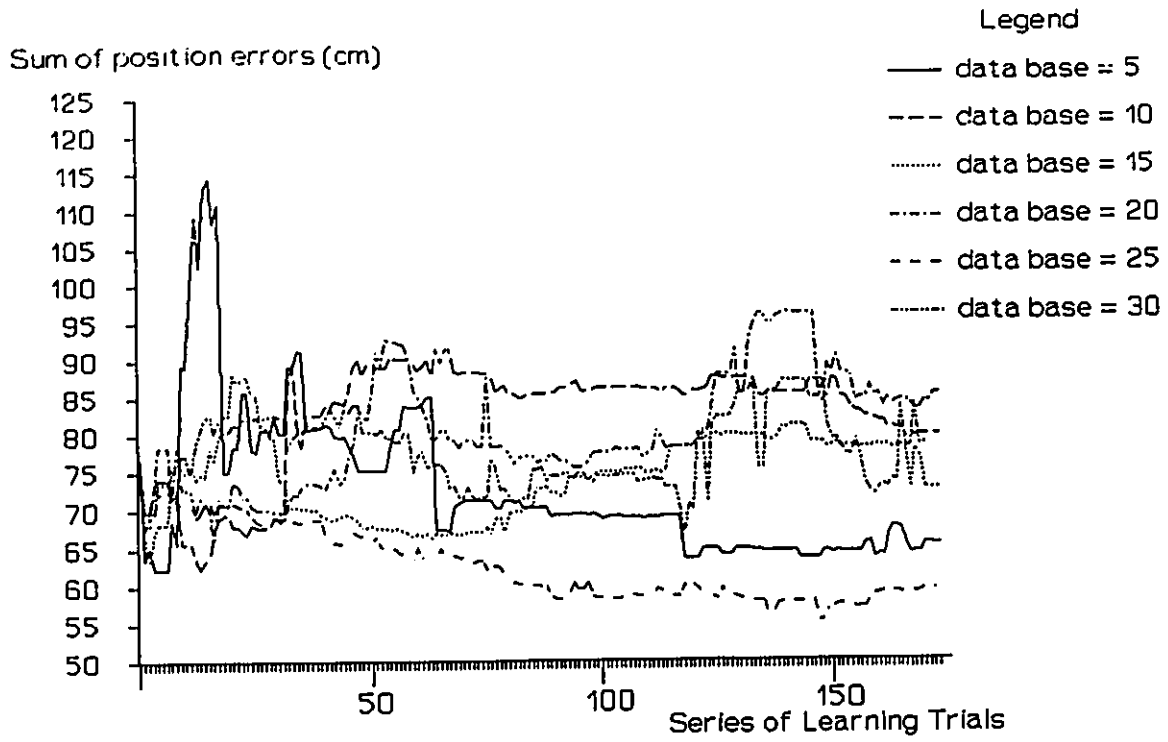


Figure 6.24: Random Configuration Learning at 60 cm/s (Test Trajectory 5)

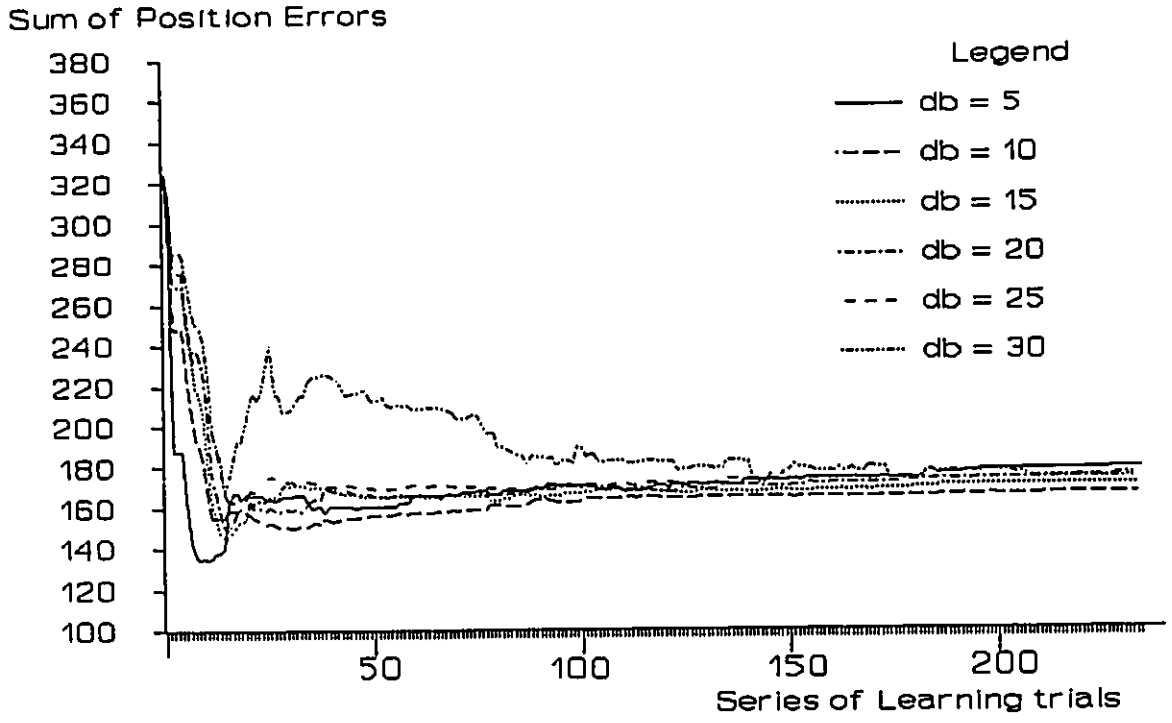


Figure 6.25 Random Configuration Learning at 90 cm/s (Test Trajectory 1)

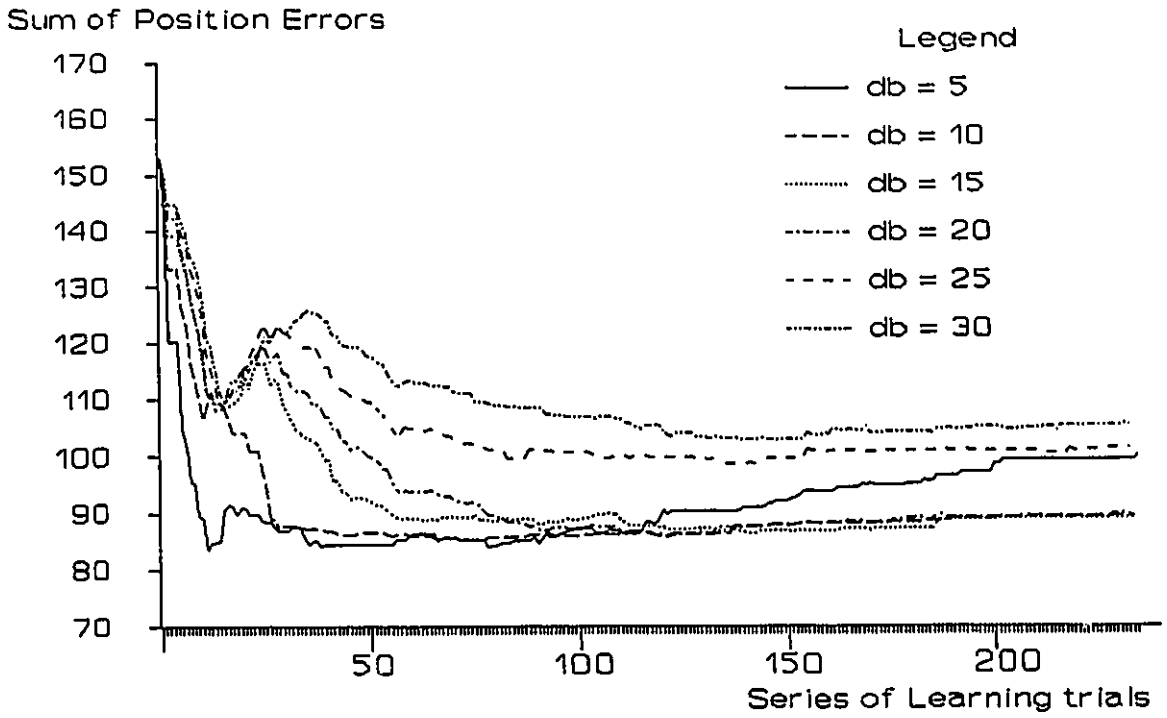


Figure 6.26 Random Configuration Learning at 90 cm/s (Test Trajectory 2)

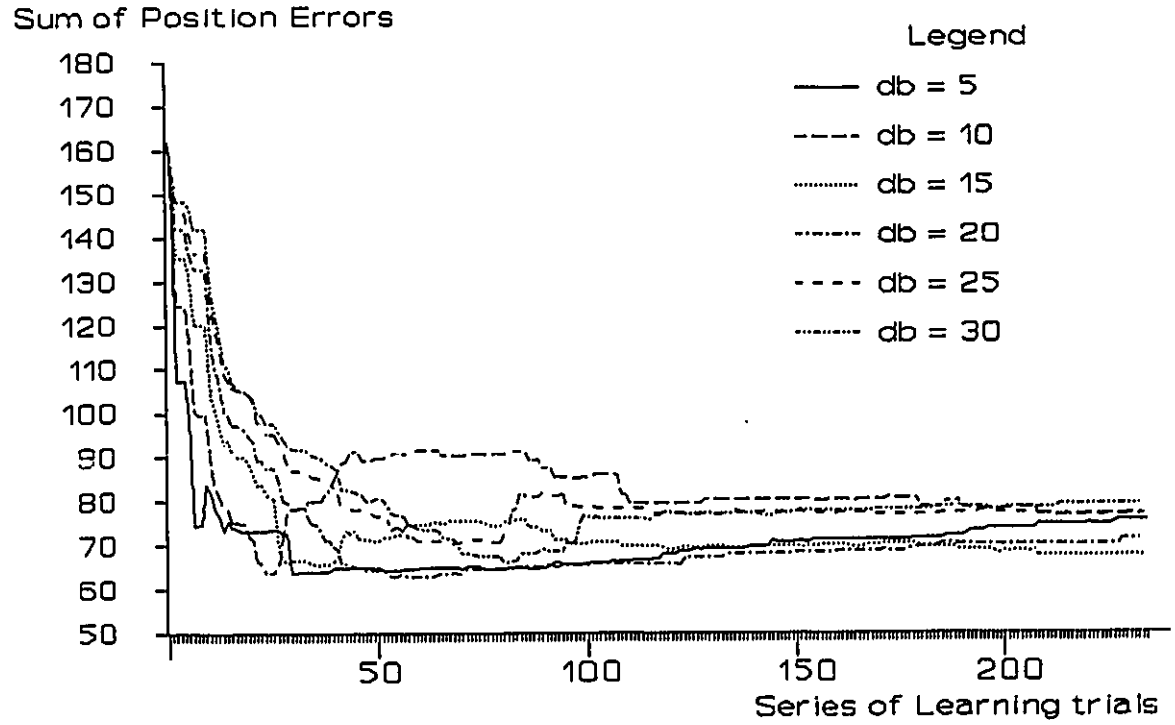


Figure 6.27 Random Configuration Learning at 90 cm/s (Test Trajectory 3)

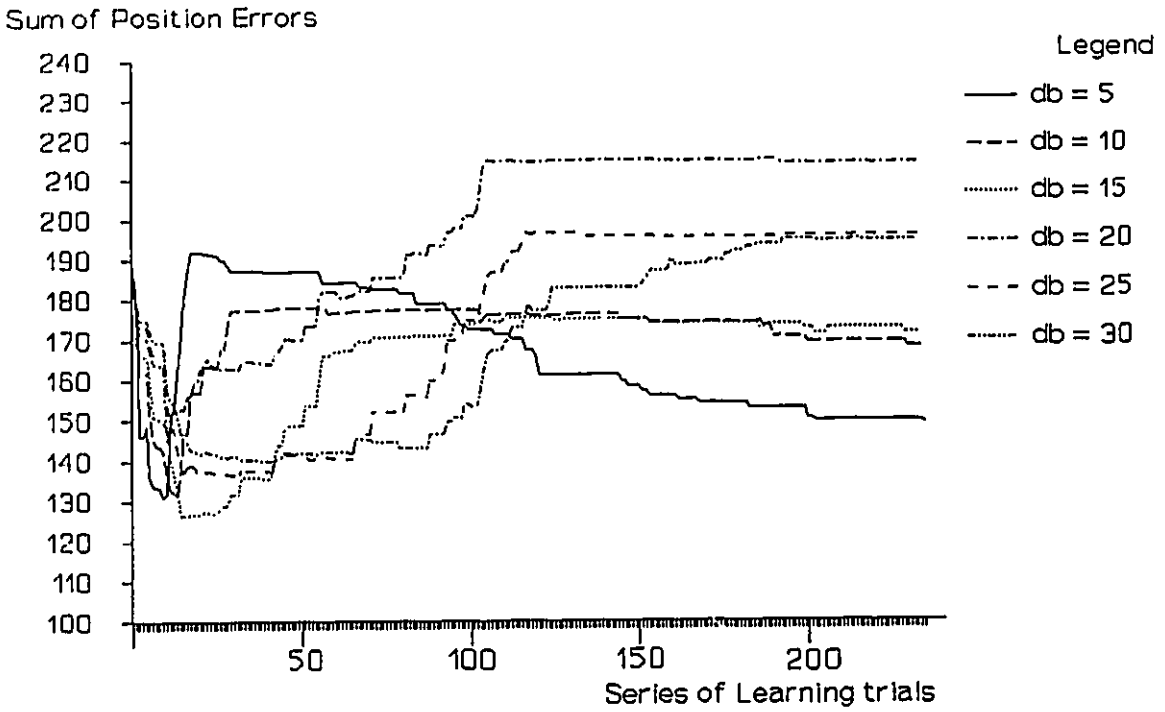


Figure 6.28 Random Configuration Learning at 90 cm/s (Test Trajectory 4)

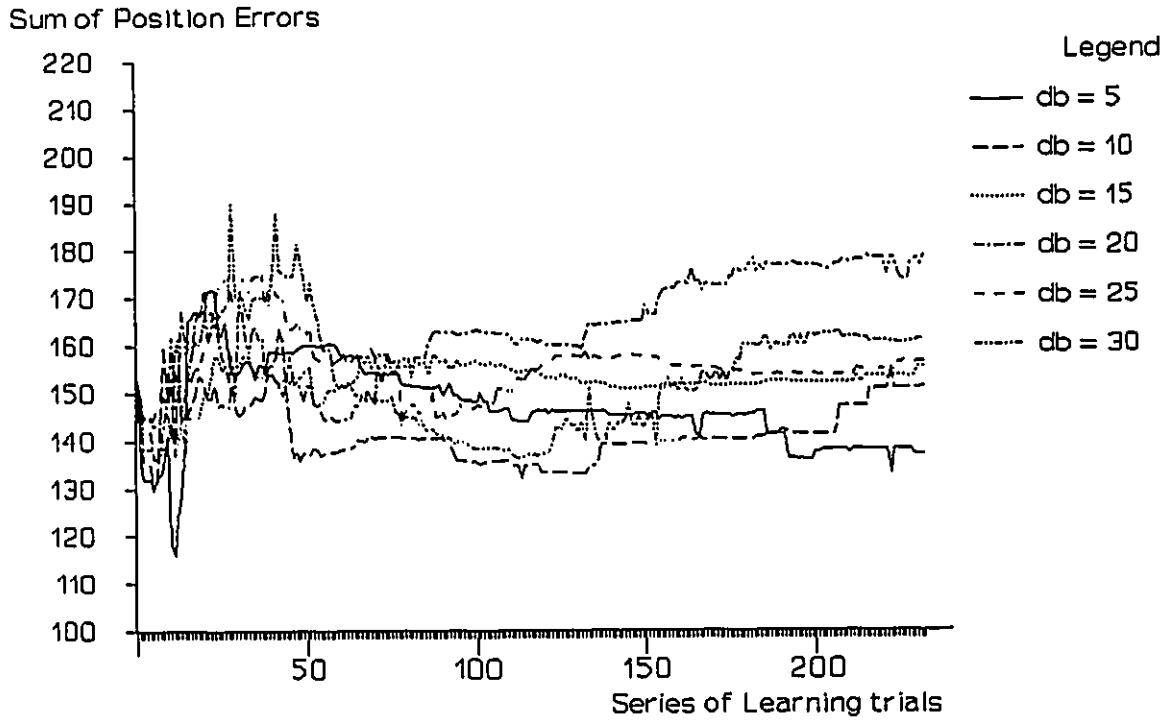


Figure 6.29 Random Configuration Learning at 90 cm/s (Test Trajectory 5)

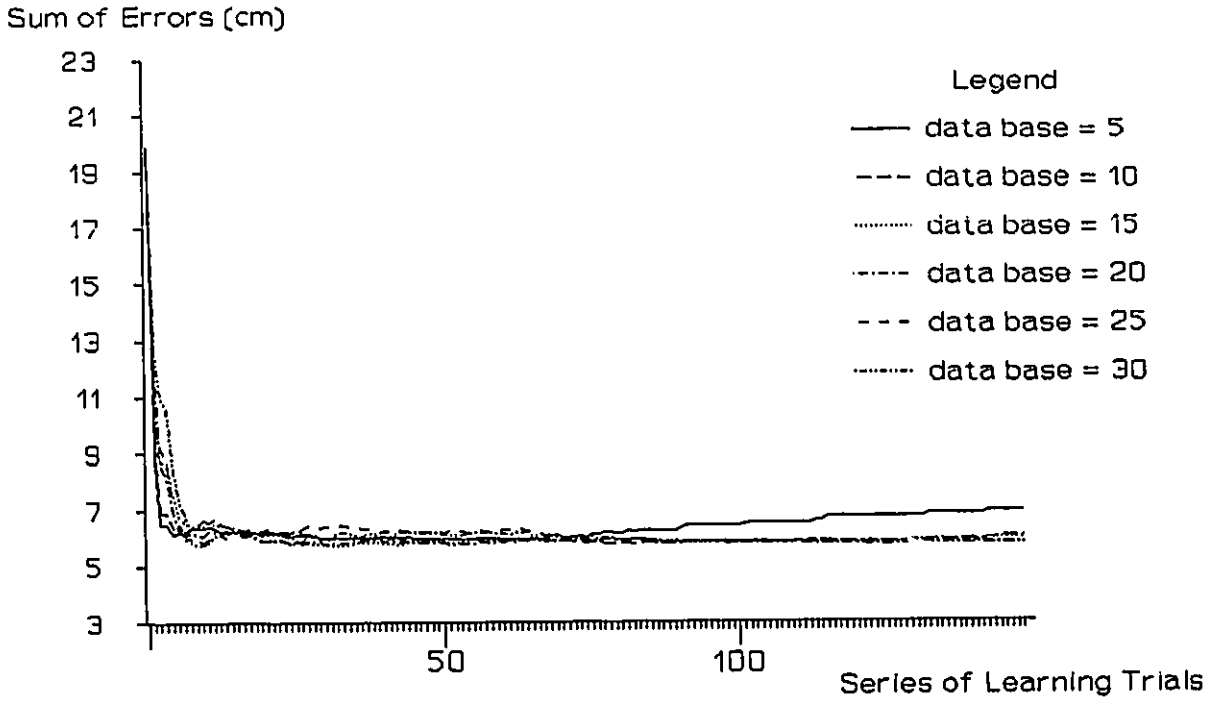


Figure 6.30 Random Configuration Learning 10 cm/s (Test Trajectory 1)

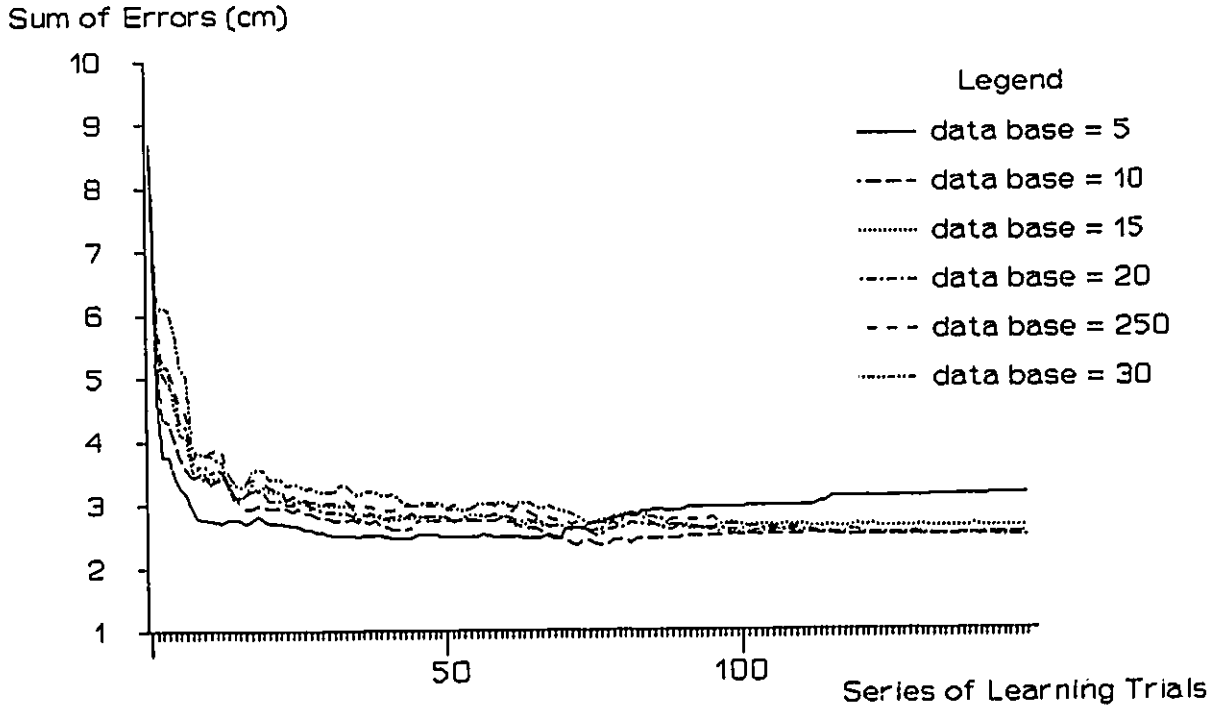


Figure 6.31 Random Configuration Learning 10 cm/s (Test Trajectory 2)

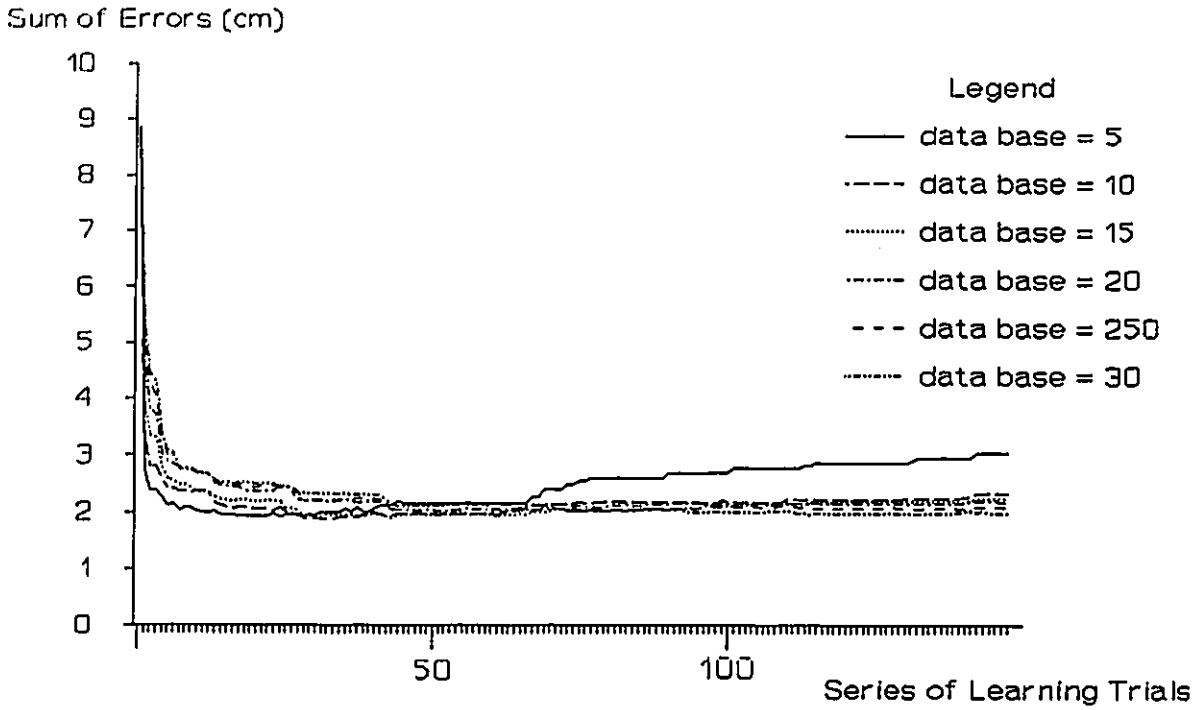


Figure 6.32 Random Configuration Learning 10 cm/s (Test Trajectory 3)

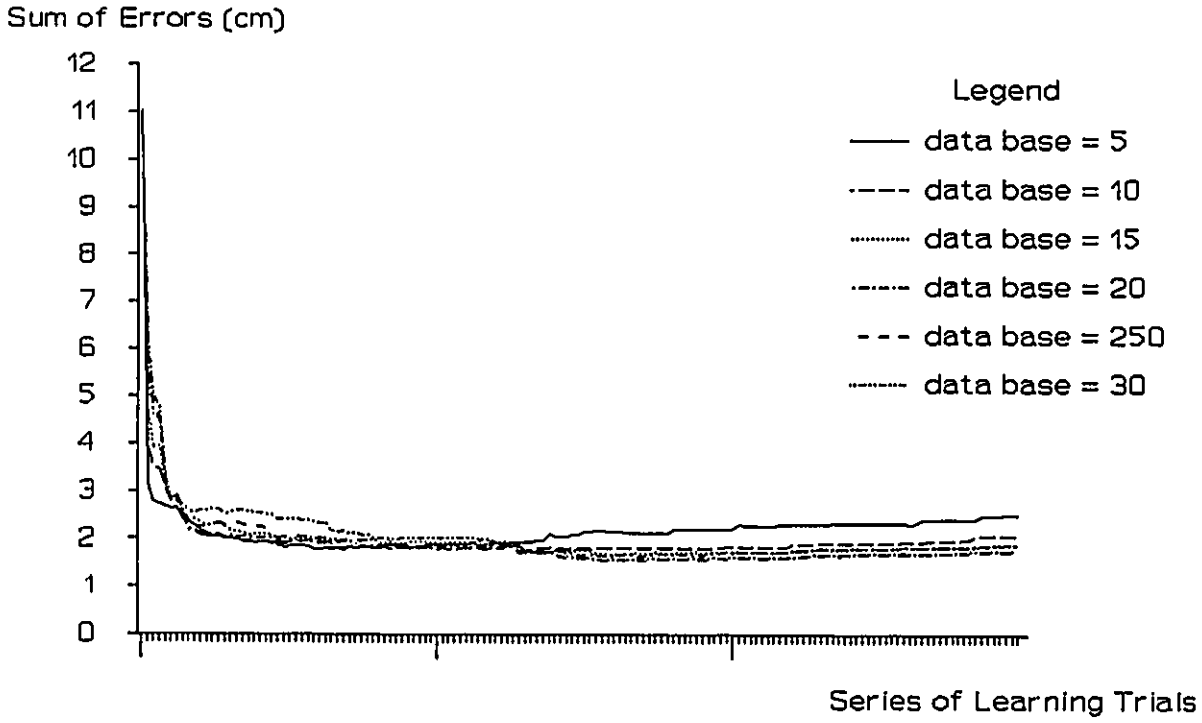


Figure 6.33 Random Configuration Learning at 10 cm/s (Trajectory 4)

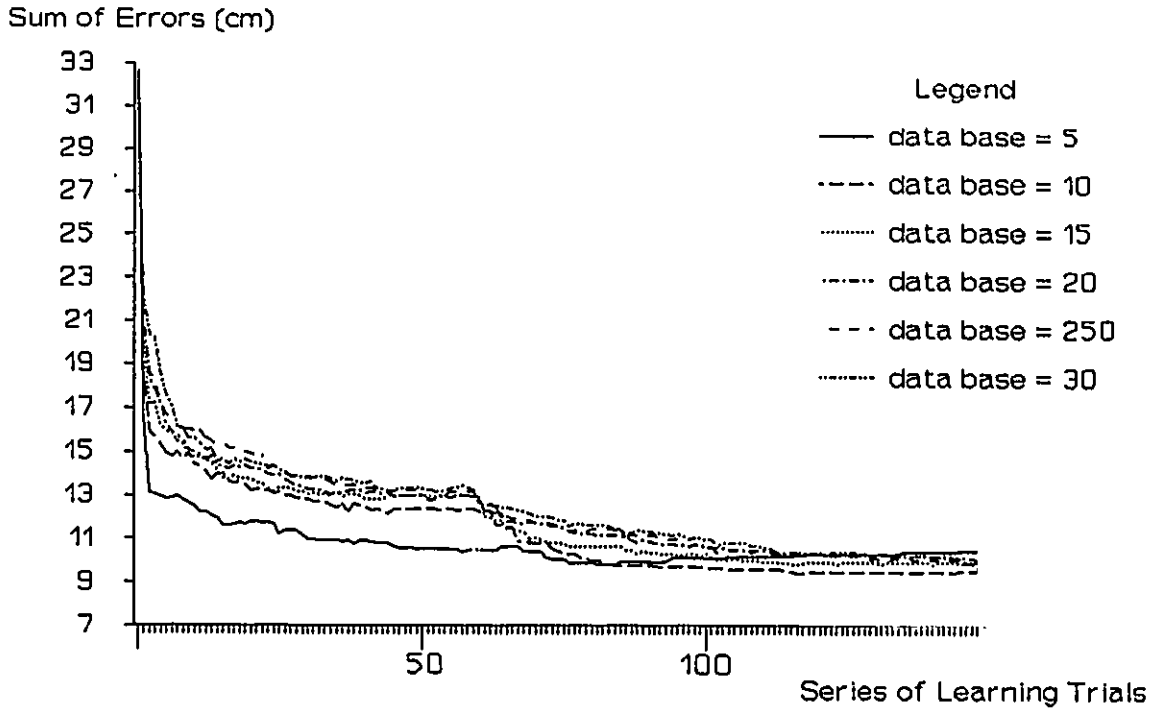


Figure 6.34 Random Configuration Learning at 10 cm/s (Trajectory 5)

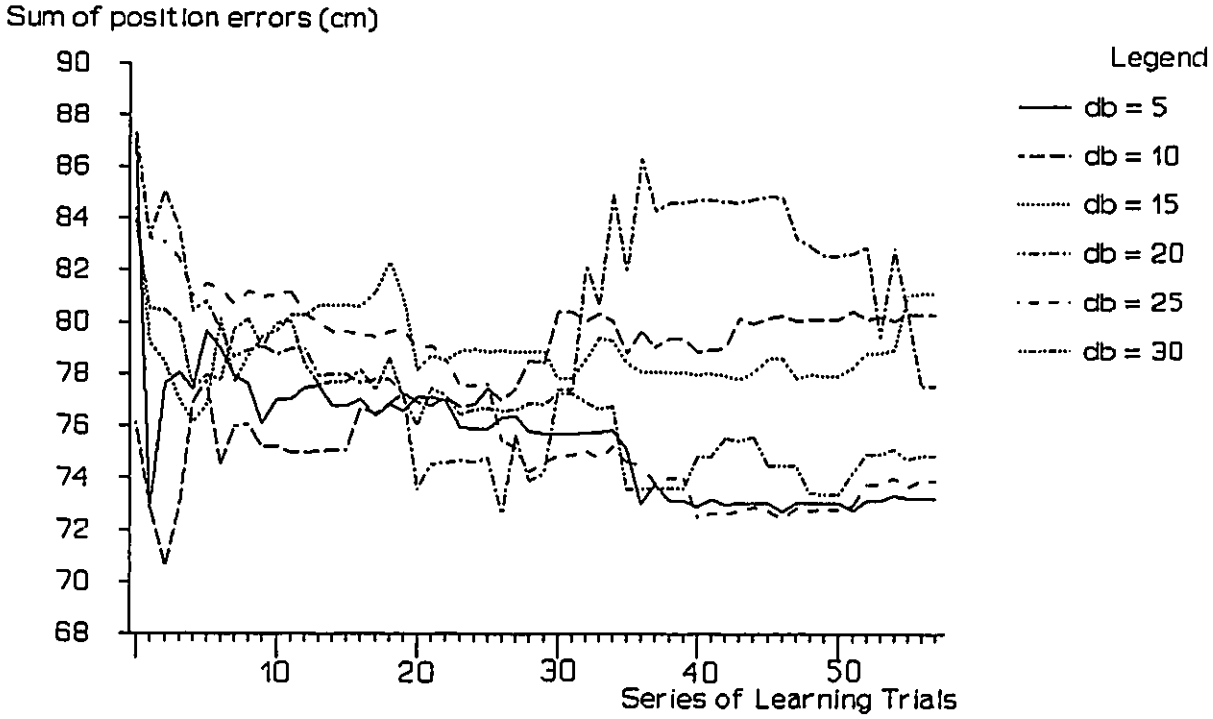


Figure 6.35 Velocity Direction Learning at 60 cm/s (Test Trajectory 1)

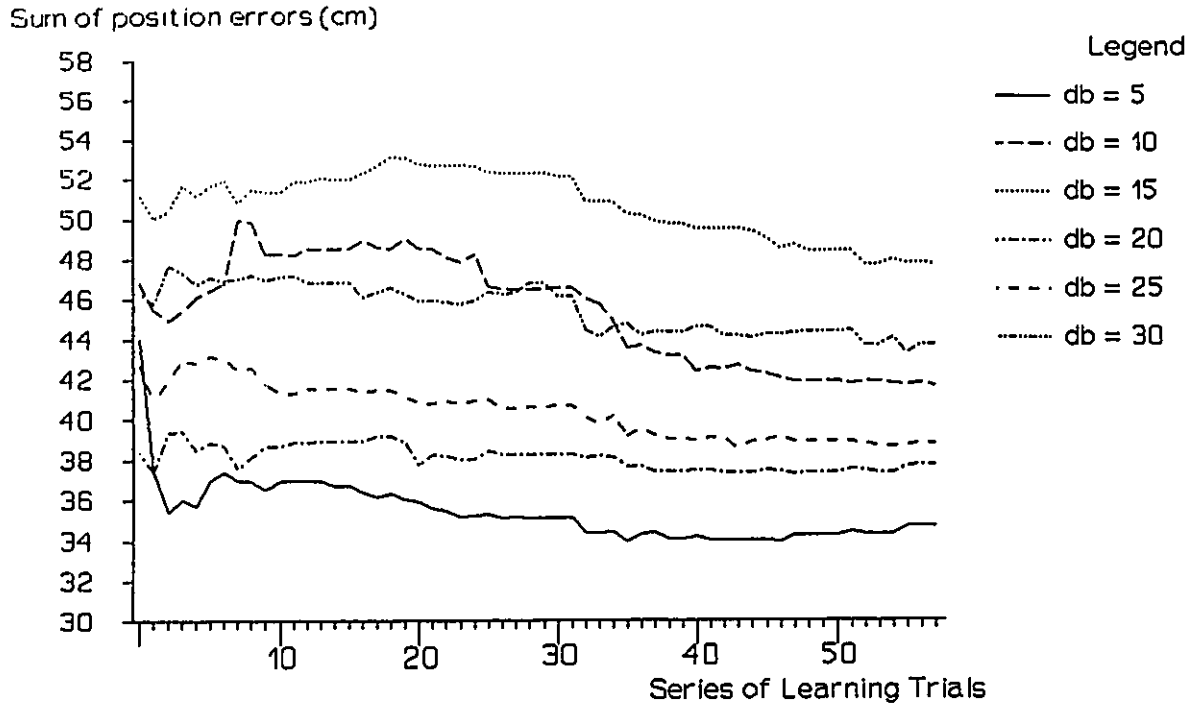


Figure 6.36 Velocity Direction Learning at 60 cm/s (Test Trajectory 2)

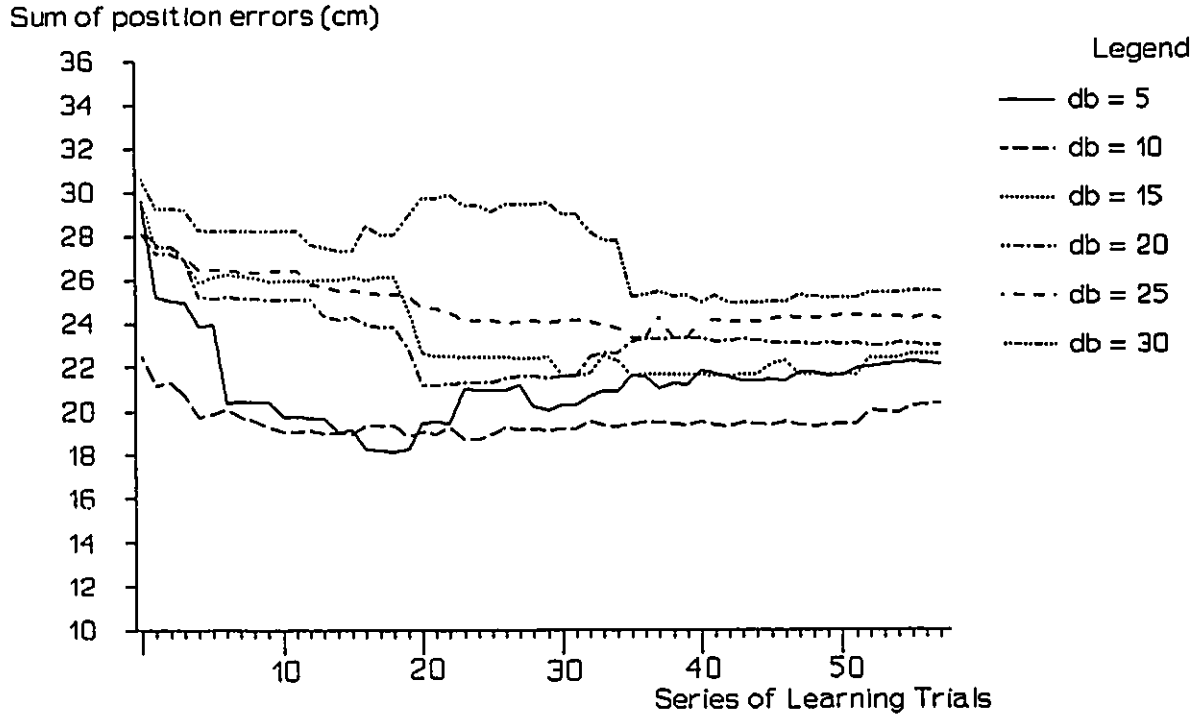


Figure 6.37 Velocity Direction Learning at 60 cm/s (Test Trajectory 3)

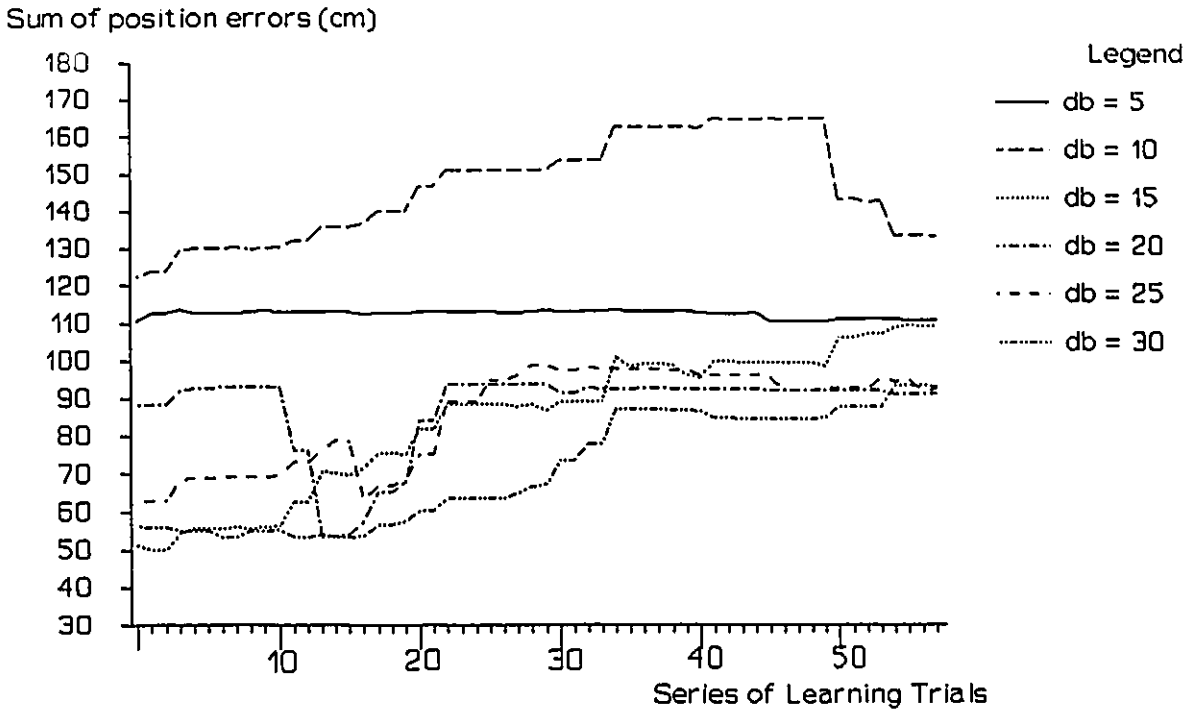


Figure 6.38 Velocity Direction Learning at 60 cm/s (Test Trajectory 4)

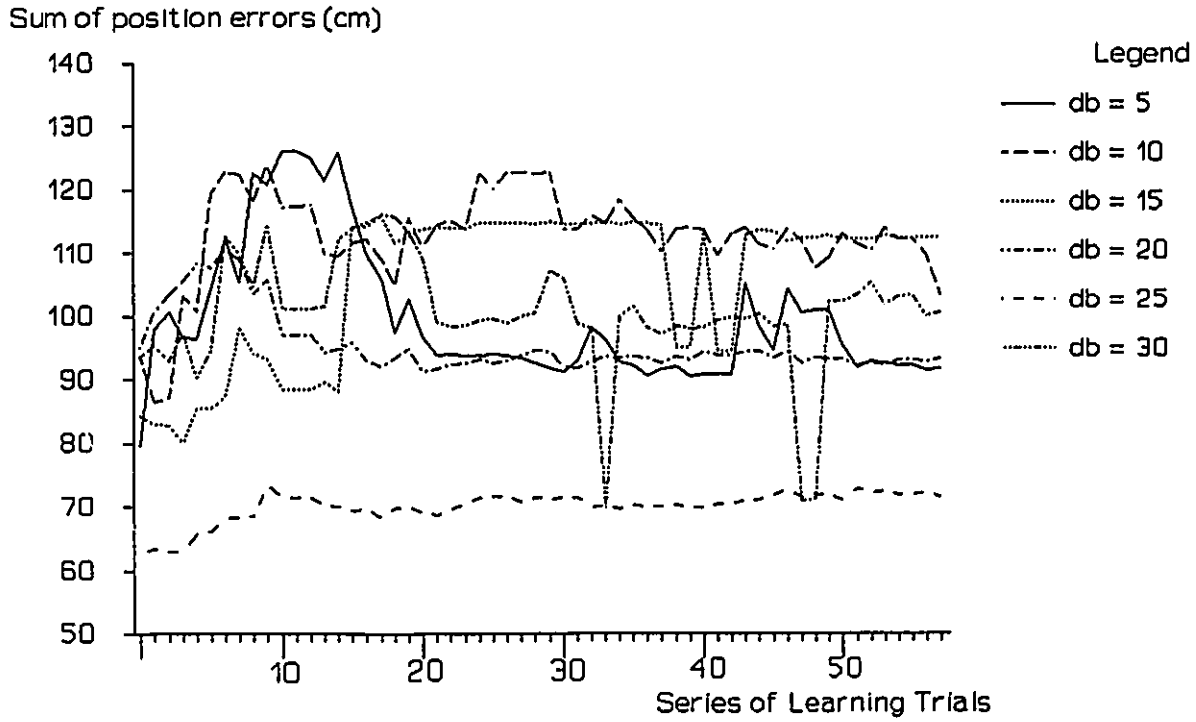


Figure 6.39 Velocity Direction Learning at 60 cm/s (Test Trajectory 5)

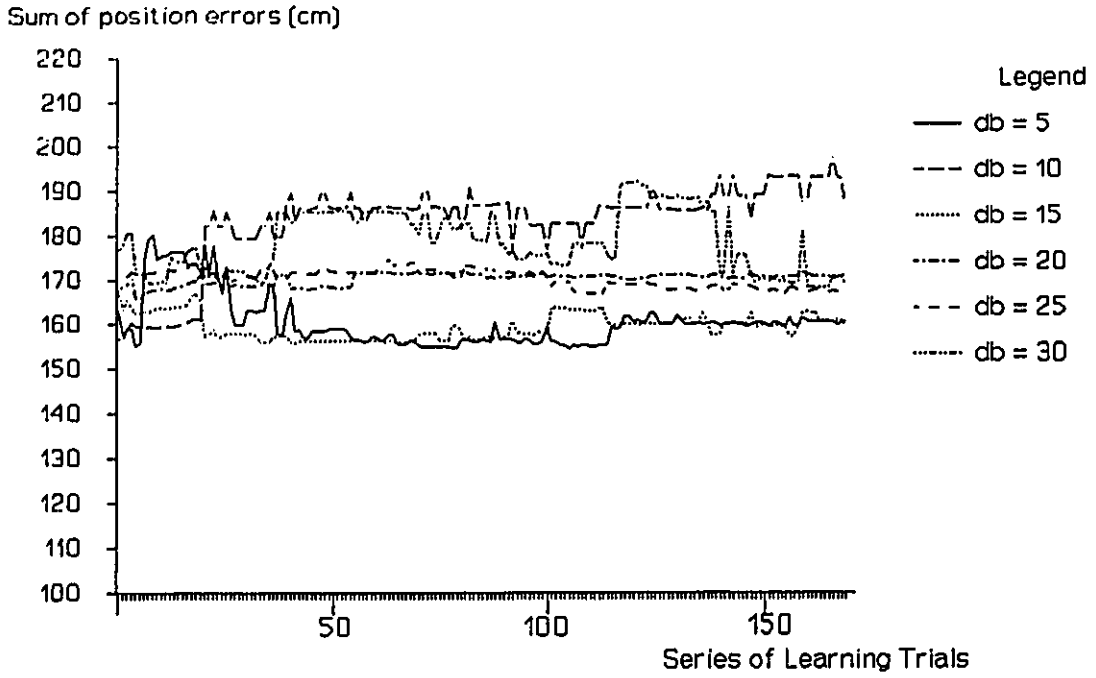


Figure 6.40 Velocity Direction Learning at 90 cm/s (Trajectory 1)

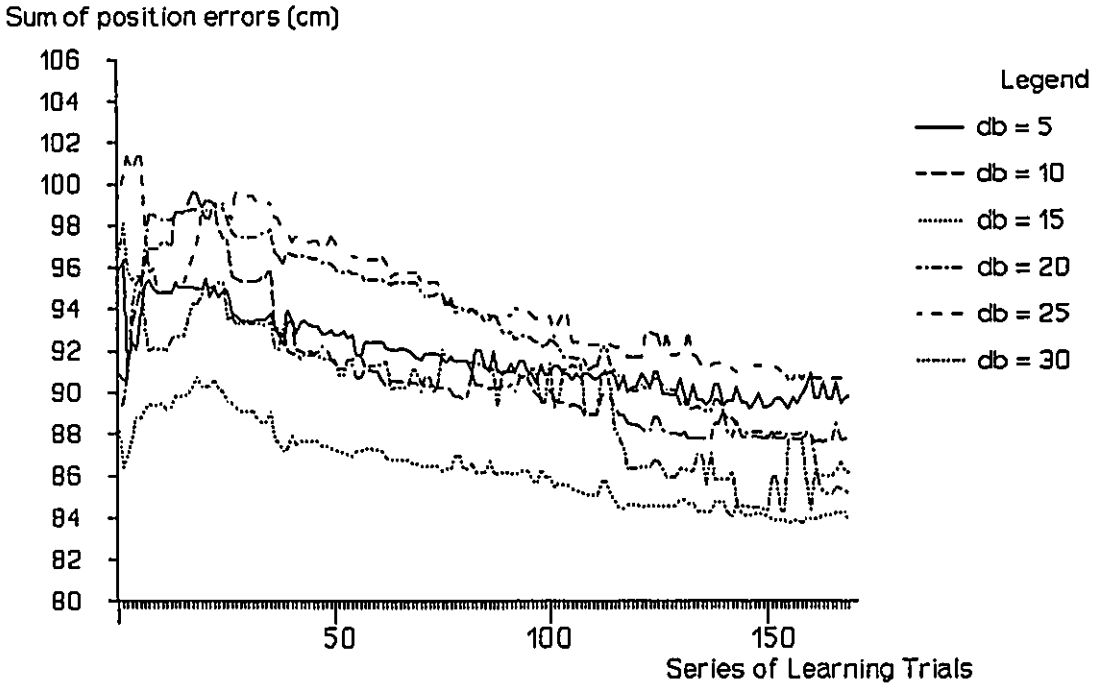


Figure 6.41 Velocity Direction Learning at 90 cm/s (Trajectory 2)

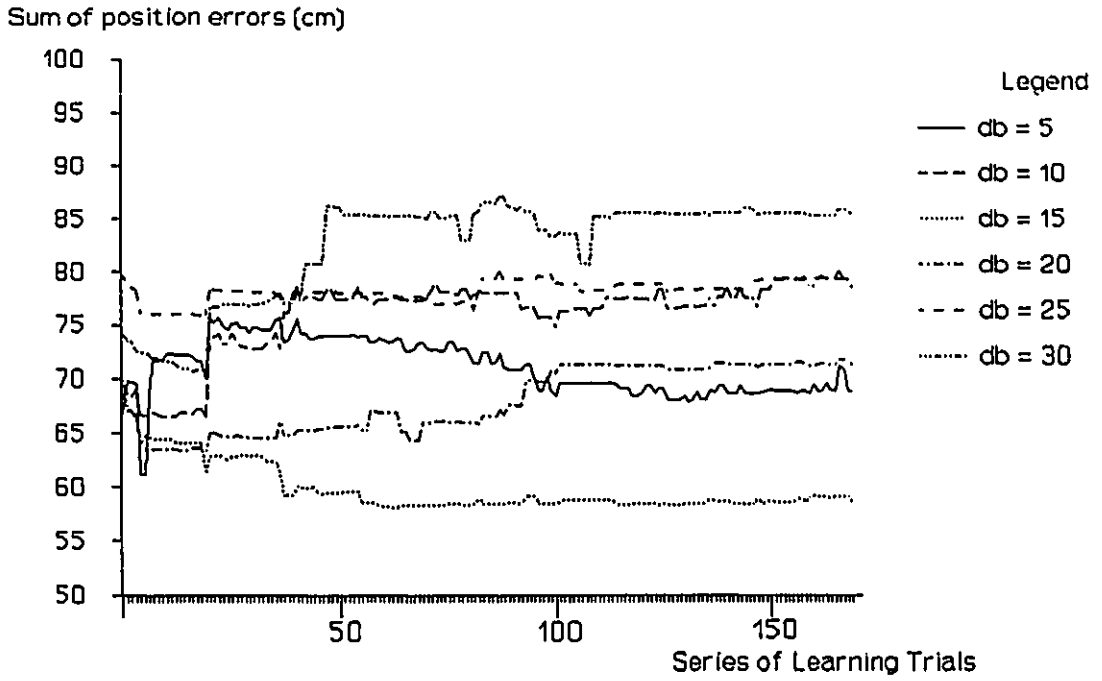


Figure 6.42 Velocity Direction Learning at 90 cm/s (Trajectory 3)

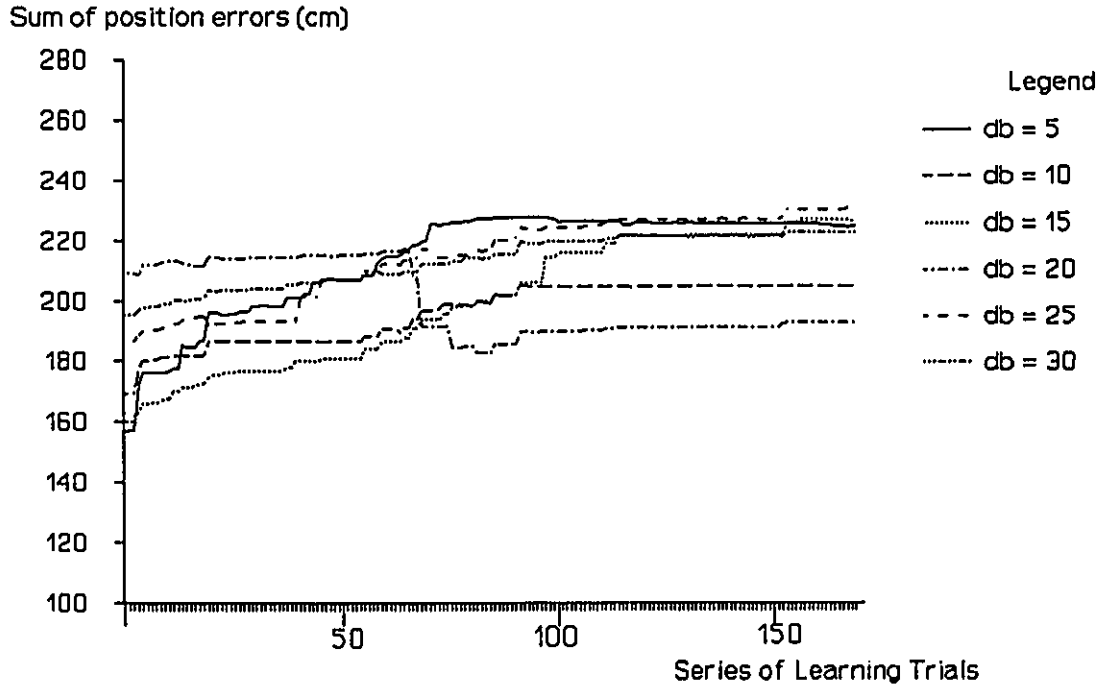


Figure 6.43 Velocity Direction Learning at 90 cm/s (Trajectory 4)

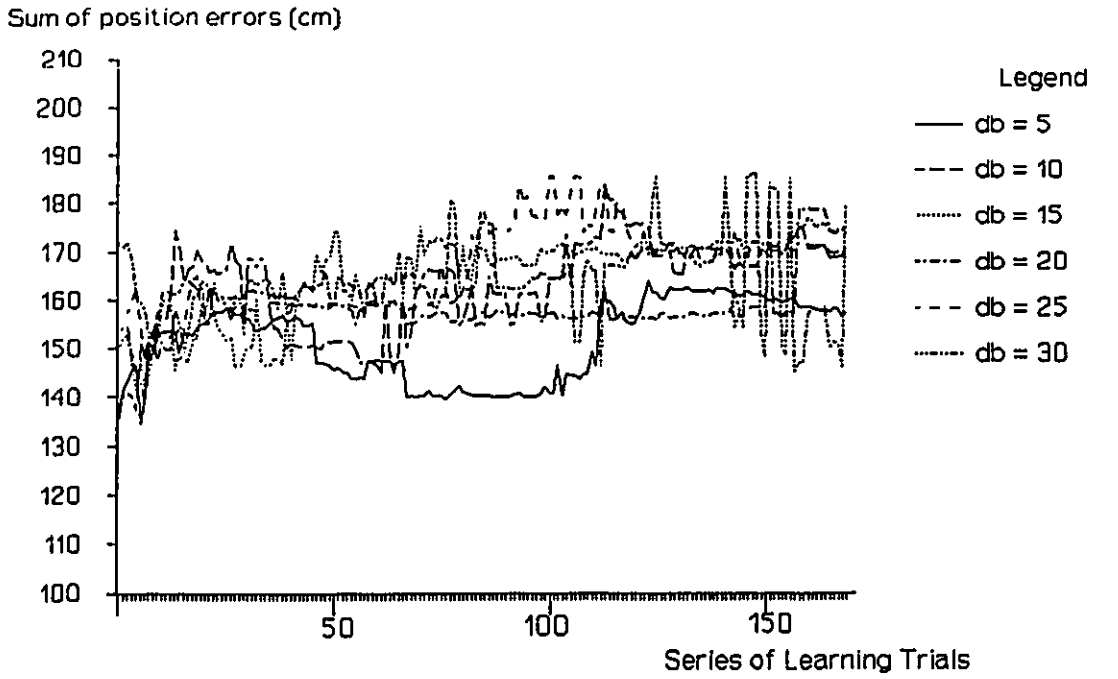


Figure 6.44 Velocity Direction Learning a 90 cm/s (Trajectory 5)

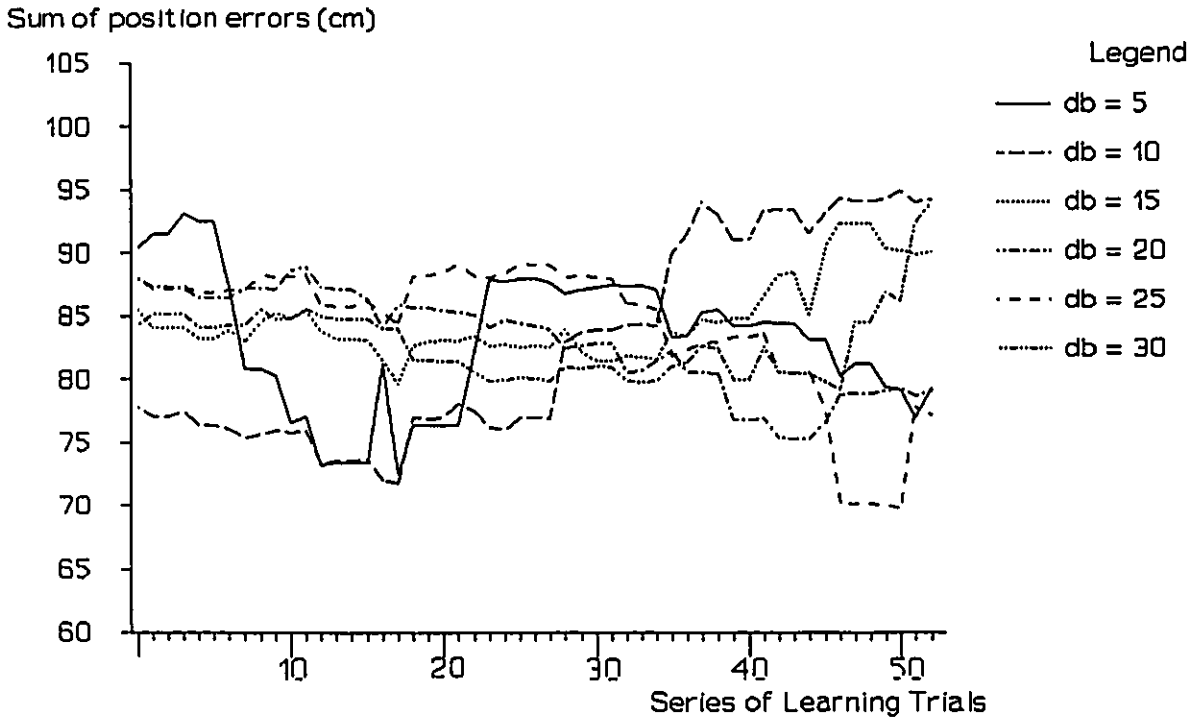


Figure 6.45 4 Zone Velocity Learning at 60 cm/s (Test Trajectory 1)

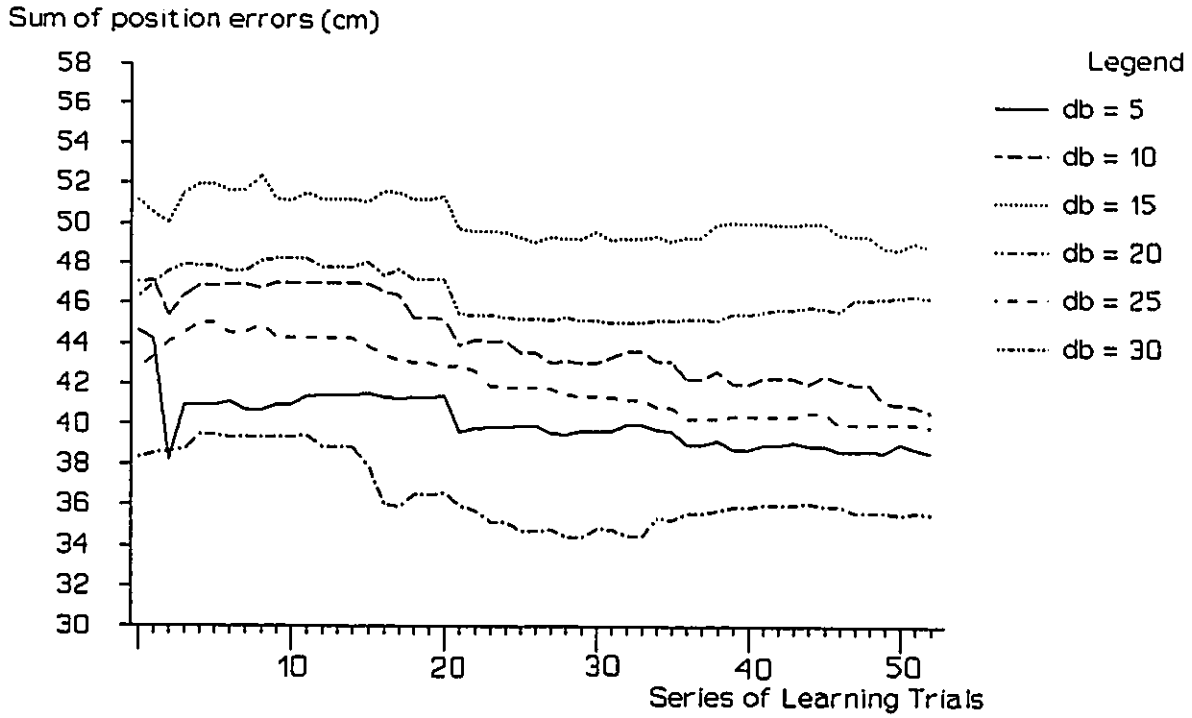


Figure 6.46 4 Zone Velocity Learning at 60 cm/s (Test Trajectory 2)

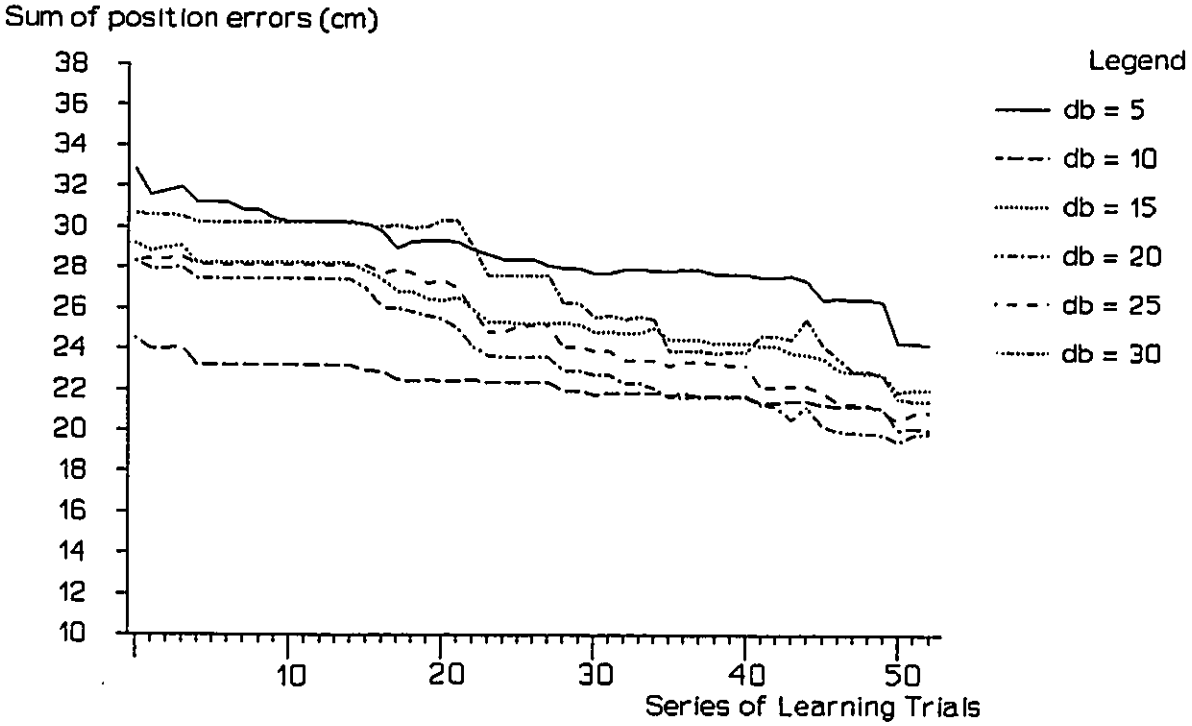


Figure 6.47 4 Zone Velocity Learning at 60 cm/s (Test Trajectory 3)

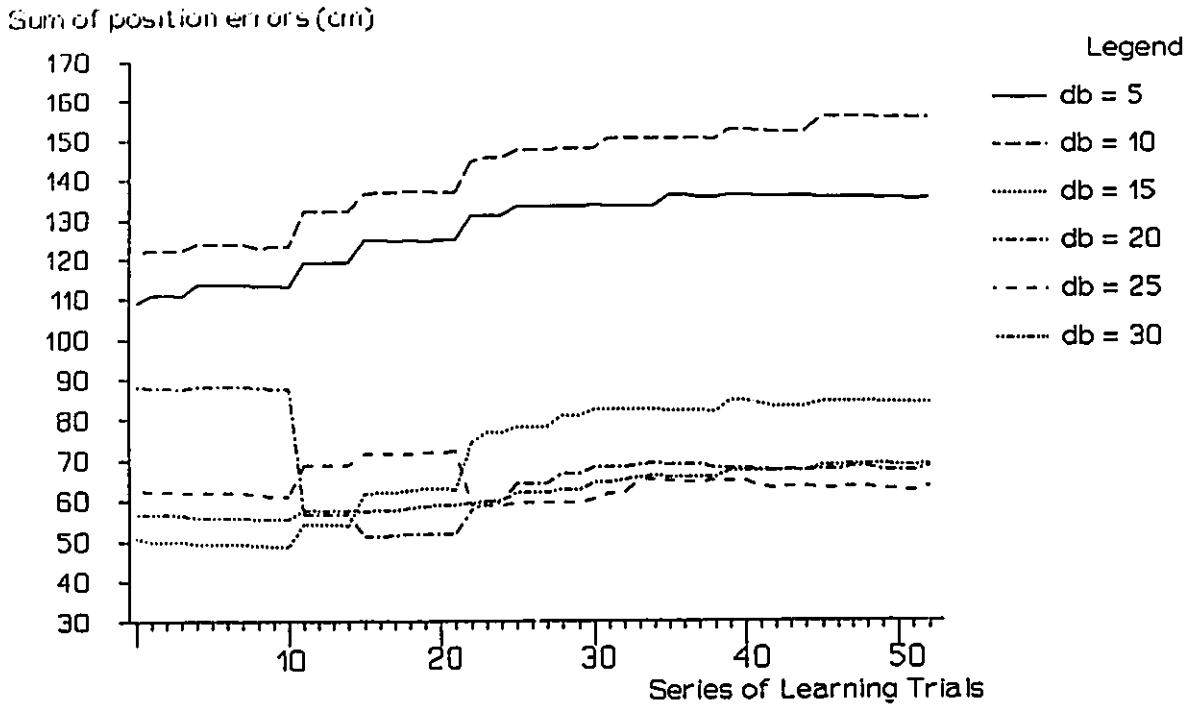


Figure 6.48 4 Zone Velocity Learning at 60 cm/s (Test Trajectory 4)

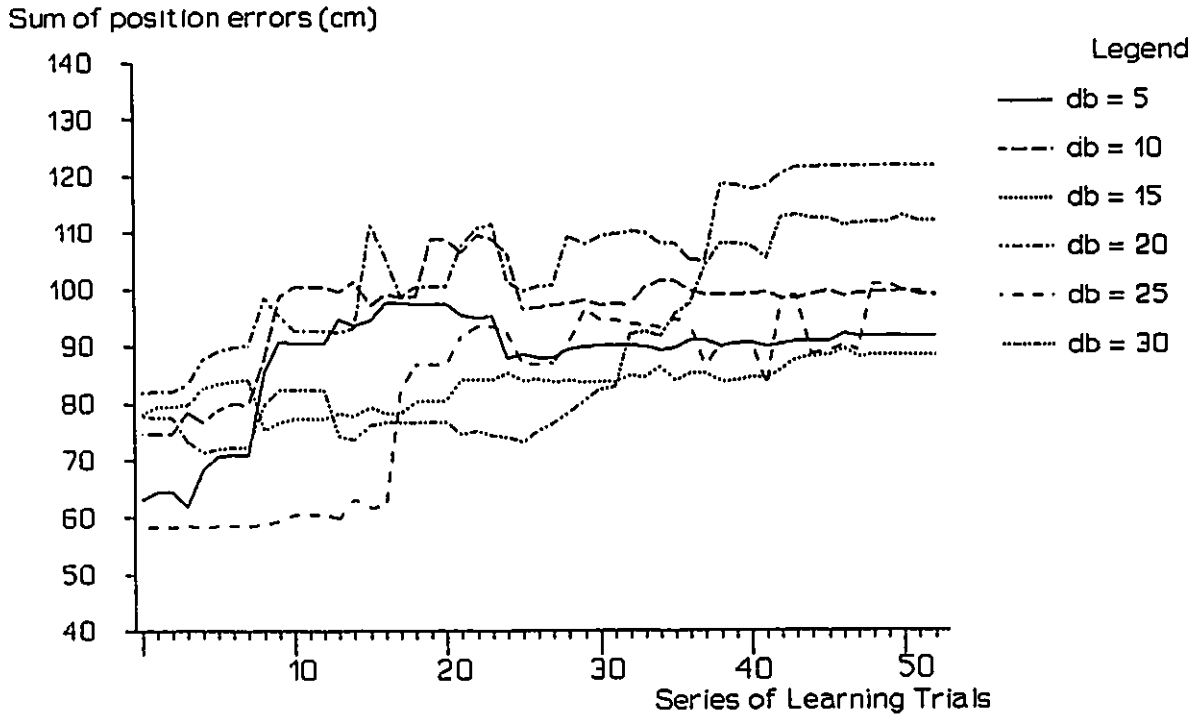


Figure 6.49 4 Zone Velocity Learning at 60 cm/s (Test Trajectory 5)

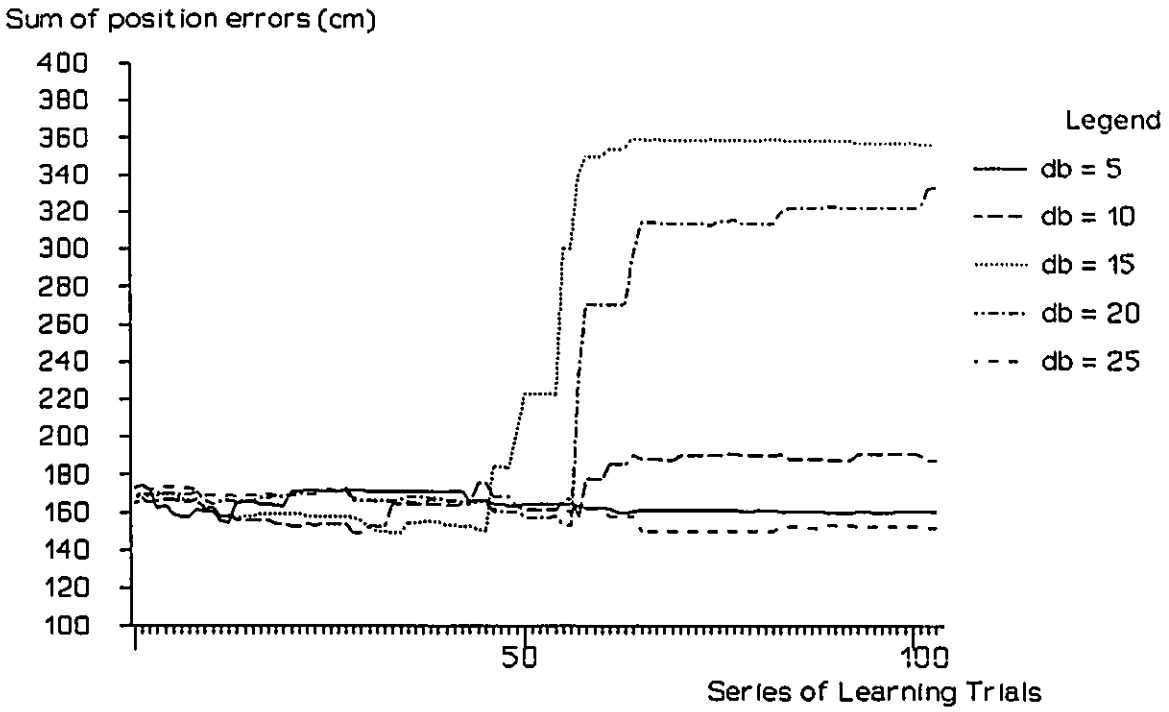


Figure 6.50 4 Zone Velocity Learning at 90 cm/s (Test Trajectory 1)

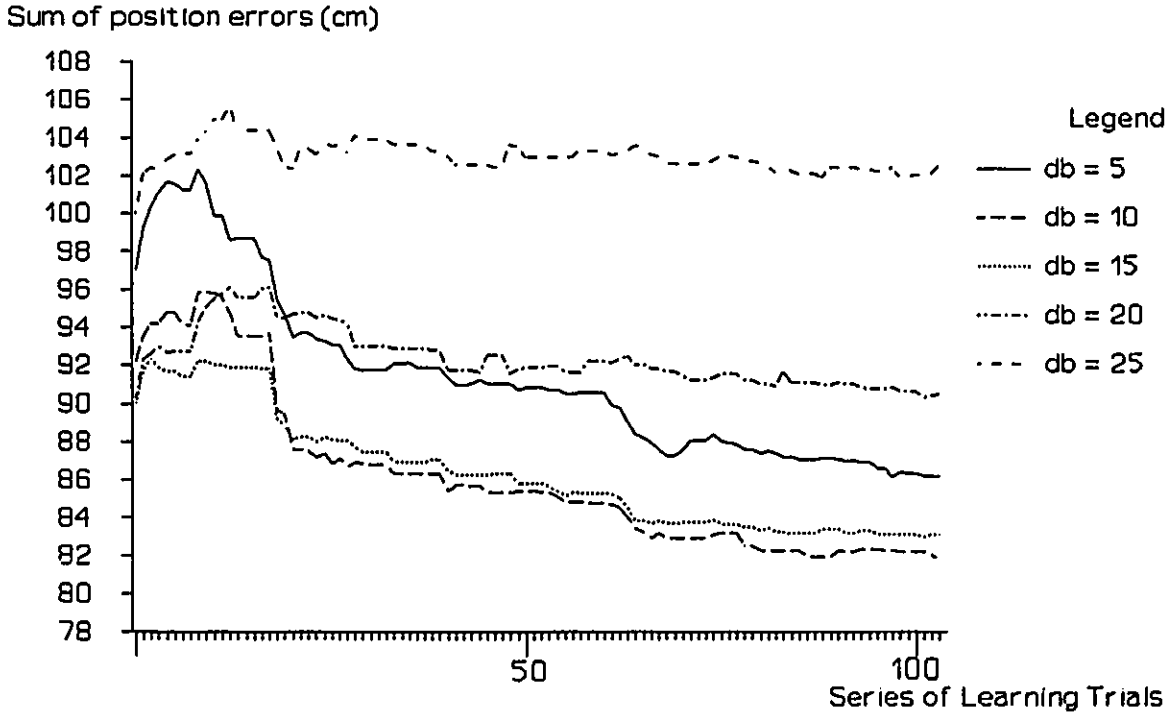


Figure 6.51 4 Zone Velocity Learning at 90 cm/s (Test Trajectory 2)

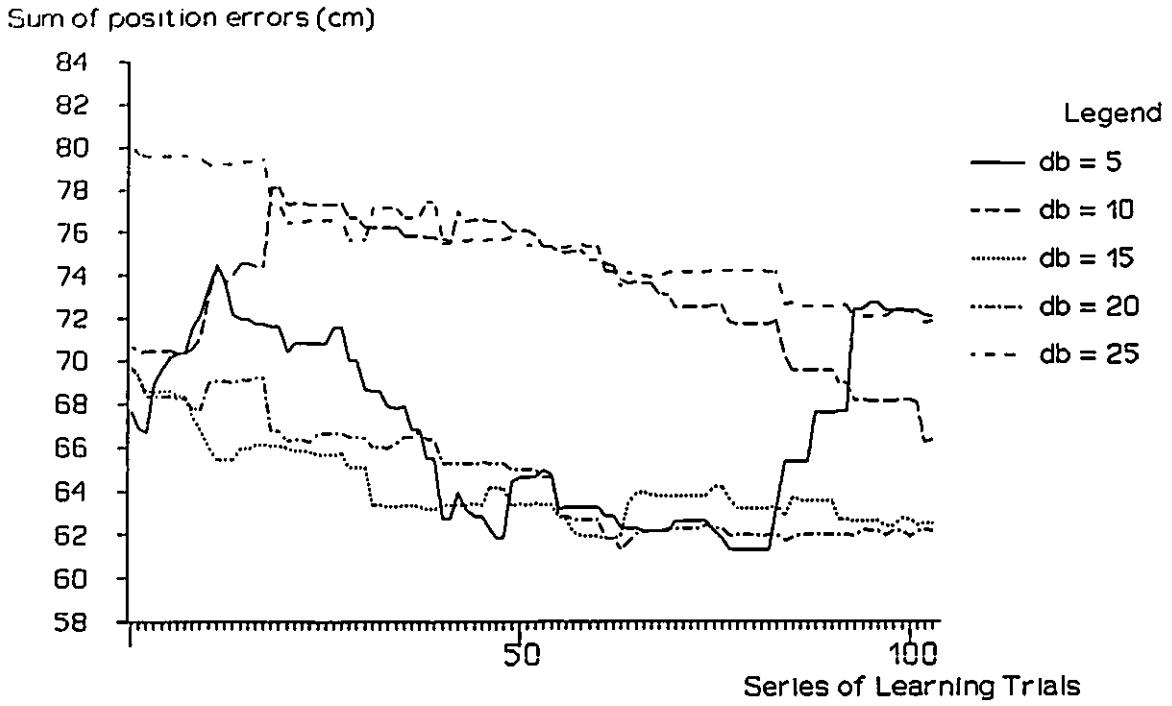


Figure 6.52 4 Zone Velocity Learning at 90 cm/s (Test Trajectory 3)

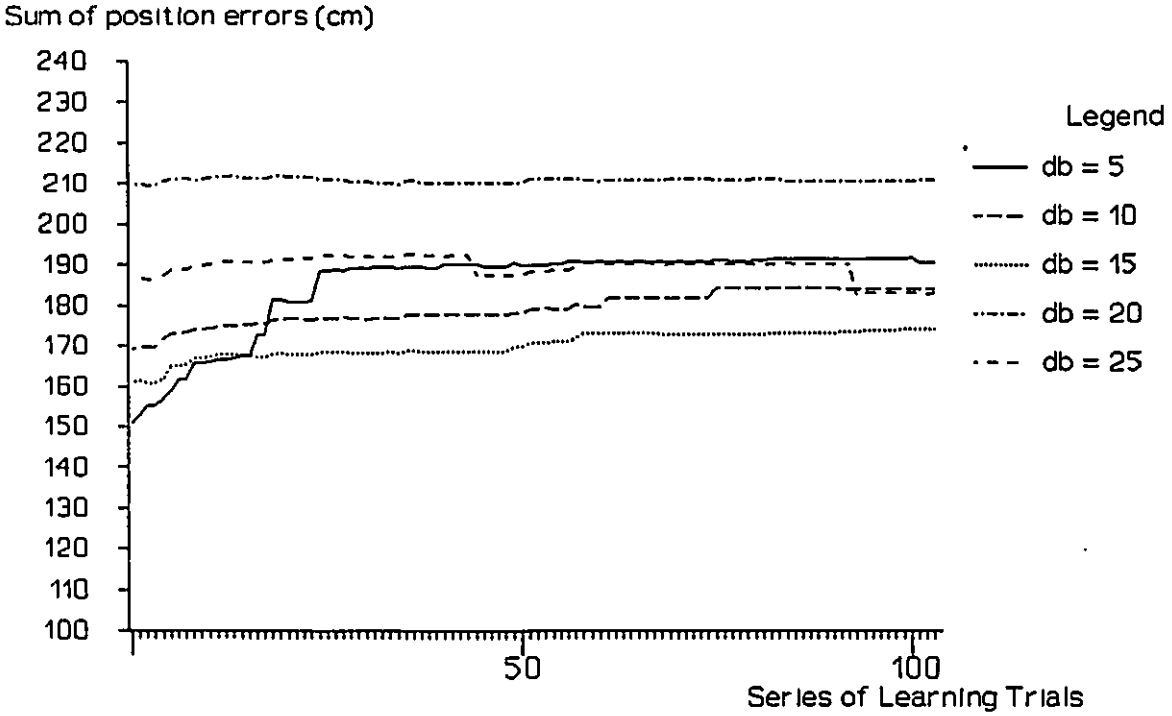


Figure 6.53 4 Zone Velocity Learning at 90 cm/s (Test Trajectory 4)

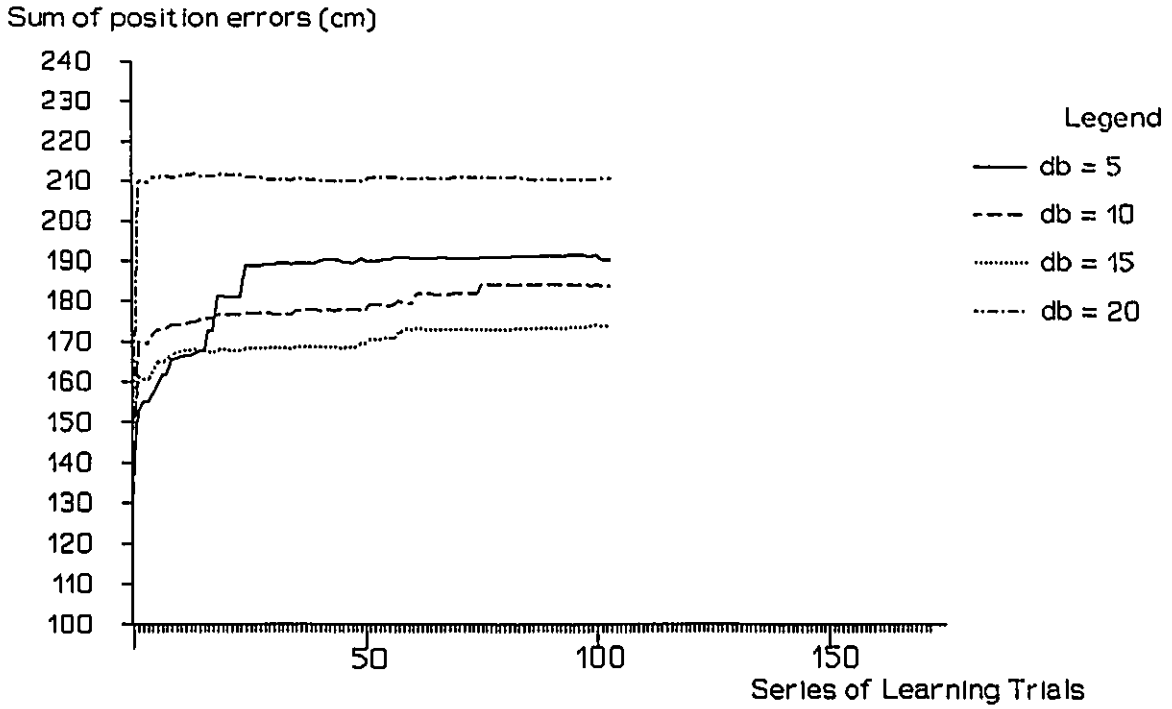


Figure 6.54 4 Zone Velocity Learning at 90 cm/s (Test Trajectory 5)

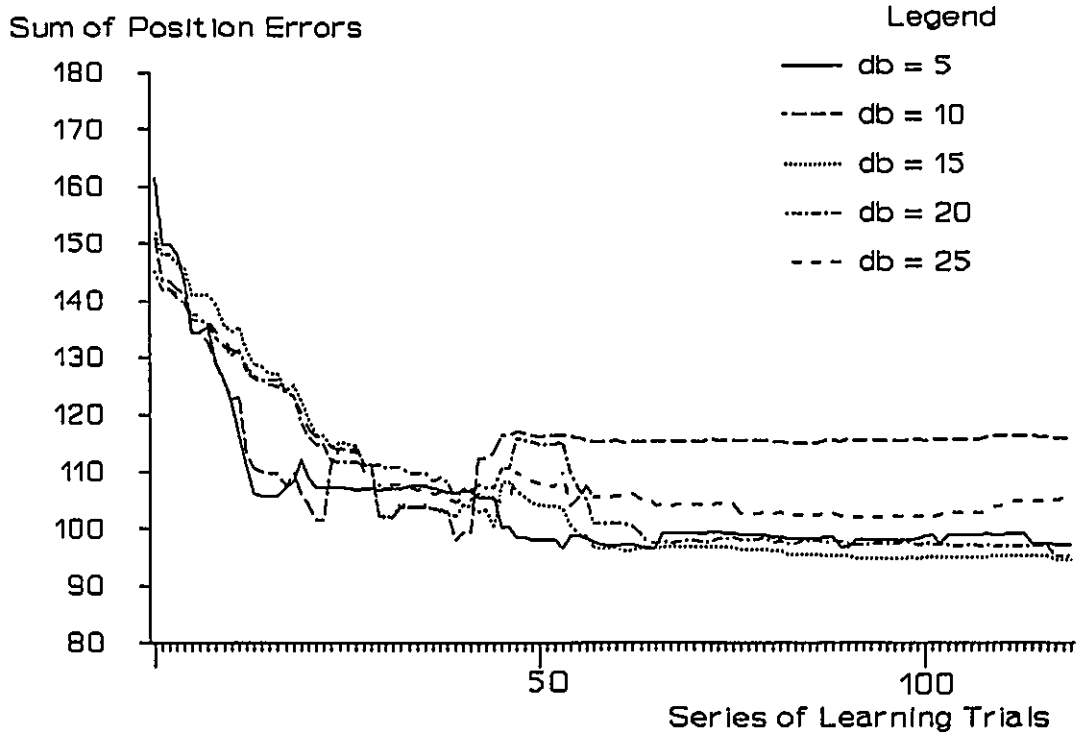


Figure 6.55 Velocity Learning at 60 cm/s (Quasi-Static Base) (Trajectory 1)

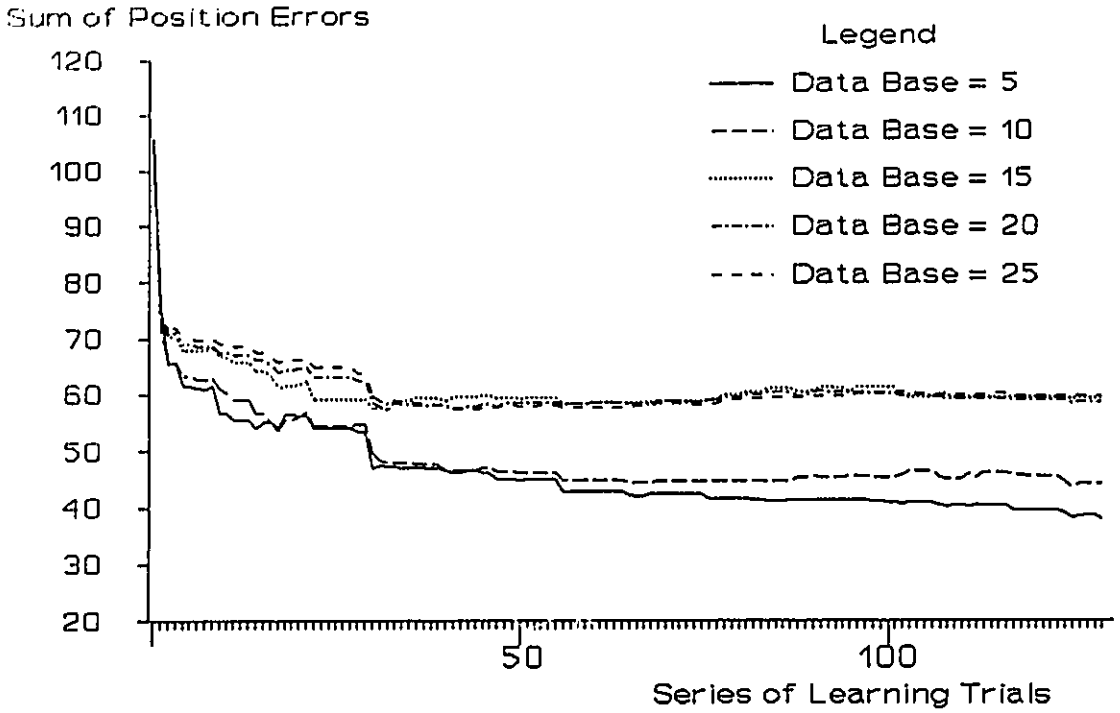


Figure 6.56 Velocity Learning 60 cm/s (Quasi-Static Base)(Traj 2)

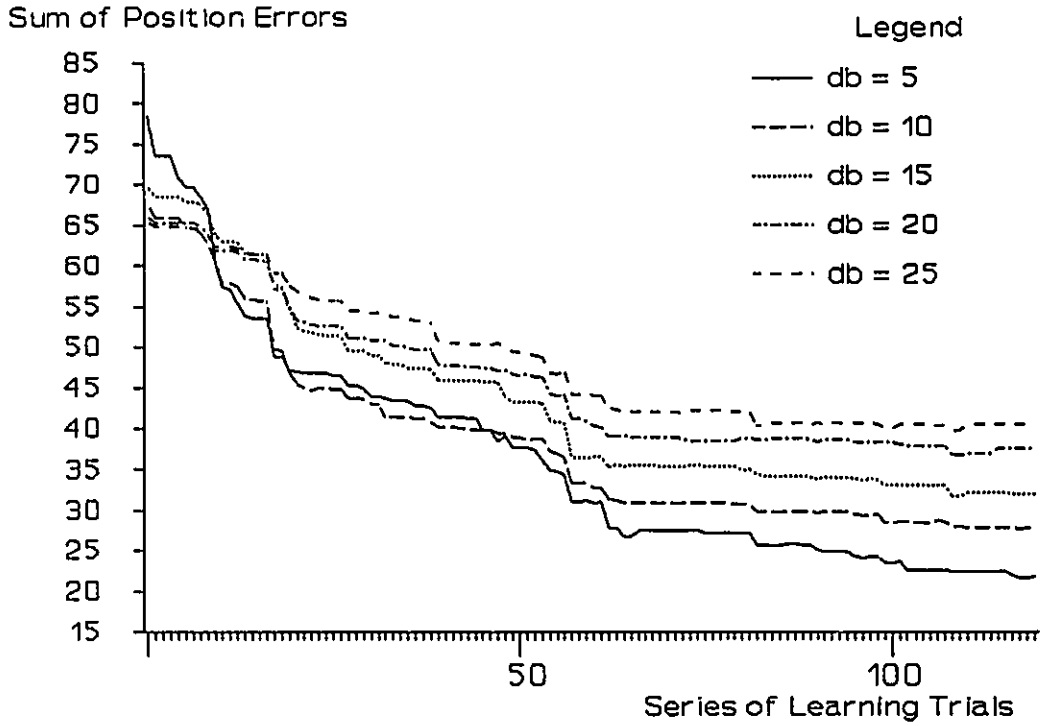


Figure 6.57 Velocity Learning 60 cm/s (Quasi-Static Base)(Traj 3)

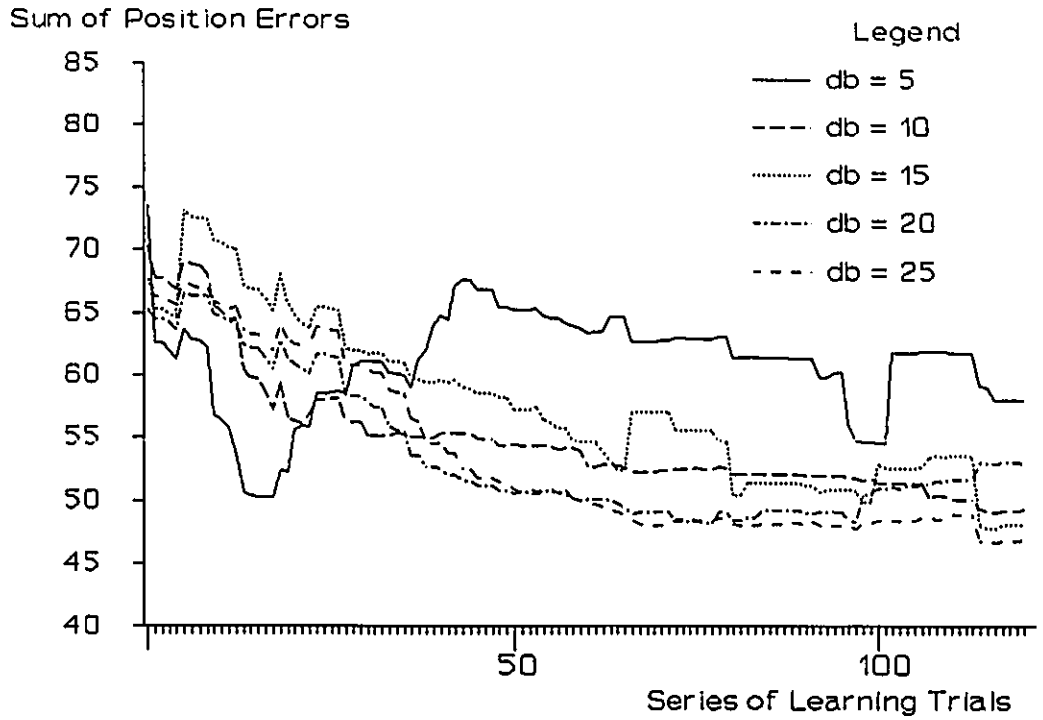


Figure 6.58 Velocity Learning 60 cm/s (Quasi-Static Base)(Traj 4)

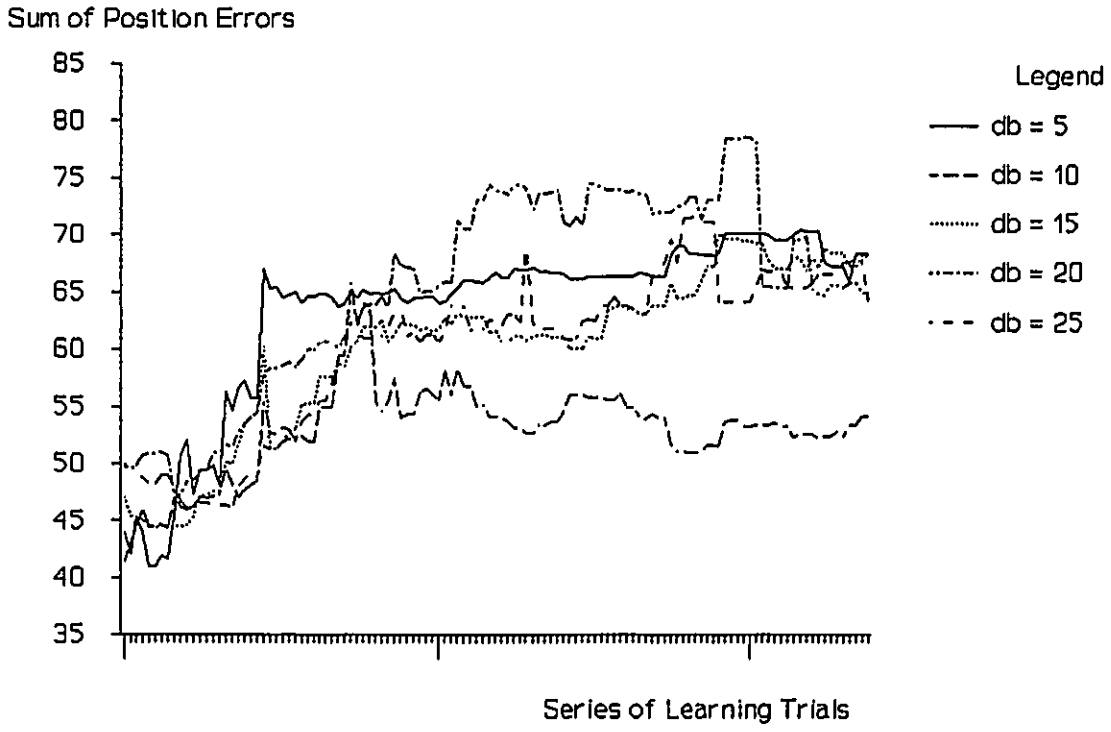


Figure 6.59 Velocity Learning 60 cm/s (Quasi-Static Base)(Traj 5)

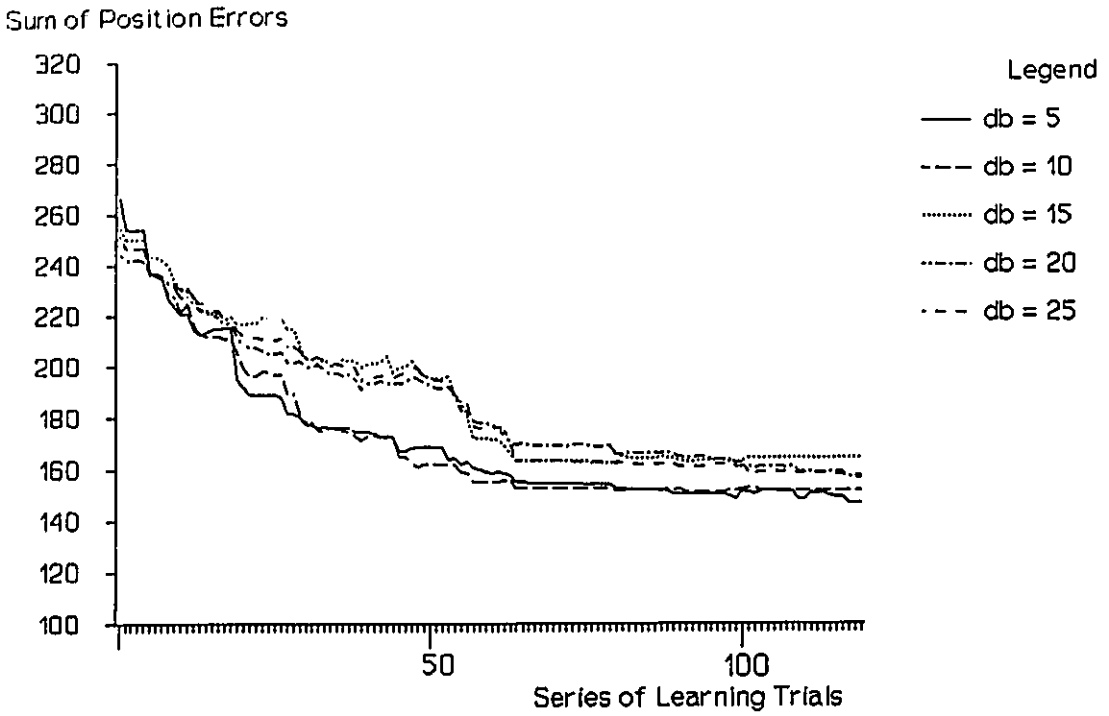


Figure 6.60 Velocity Learning at 90 cm/s (Quasi-Static Base)(Traj 1)

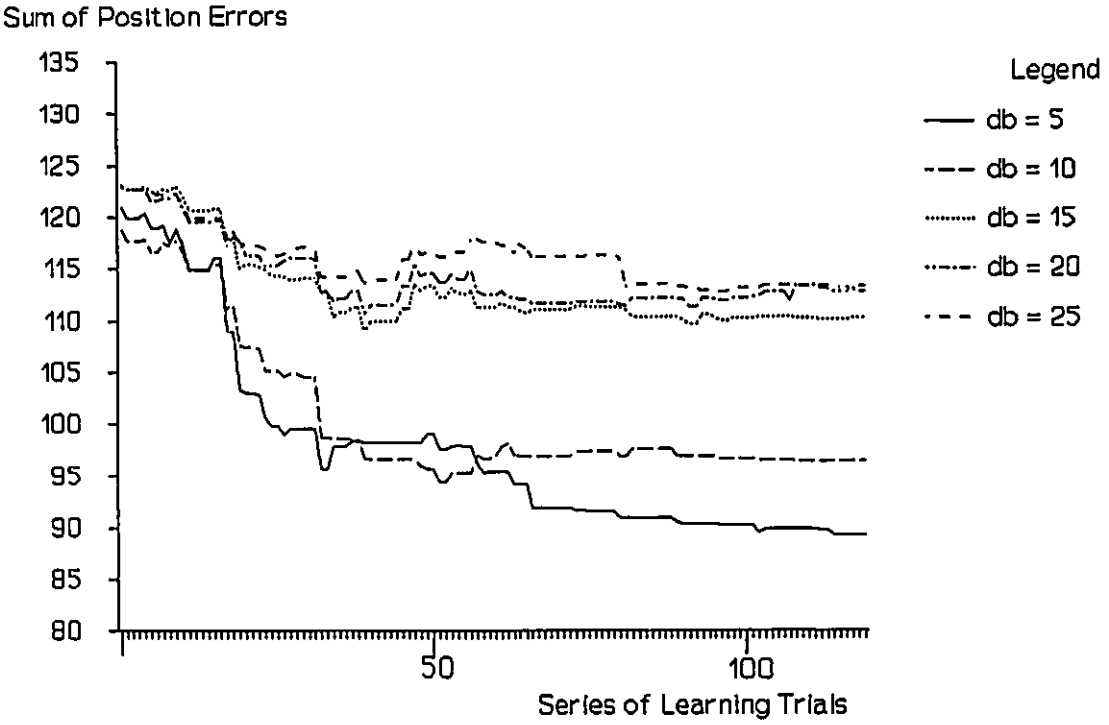


Figure 6.61 Velocity Learning at 90 cm/s (Quasi-Static Base)(Traj 2)

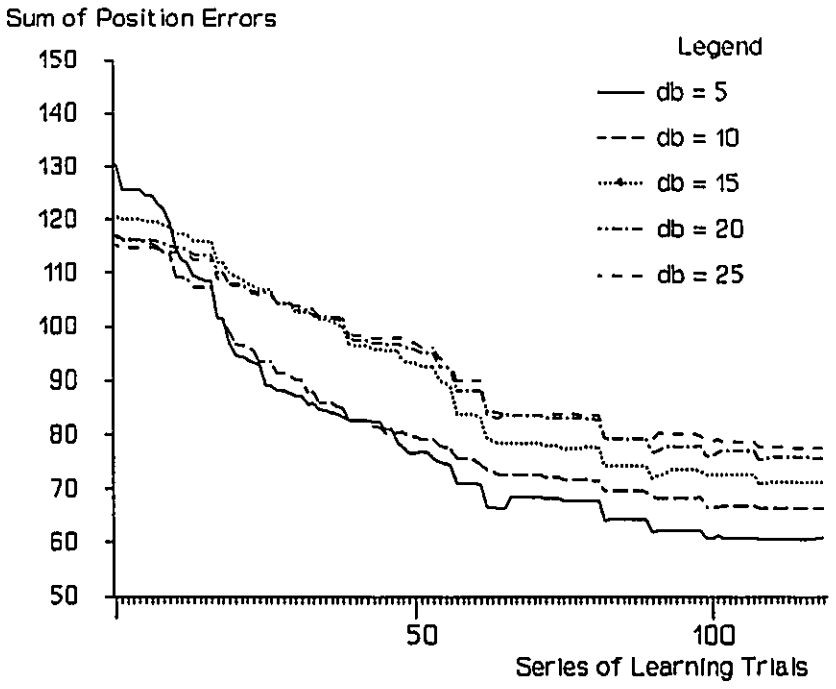


Figure 6.62 Velocity Learning at 90 cm/s (Quasi-Static Base)(Traj 3)

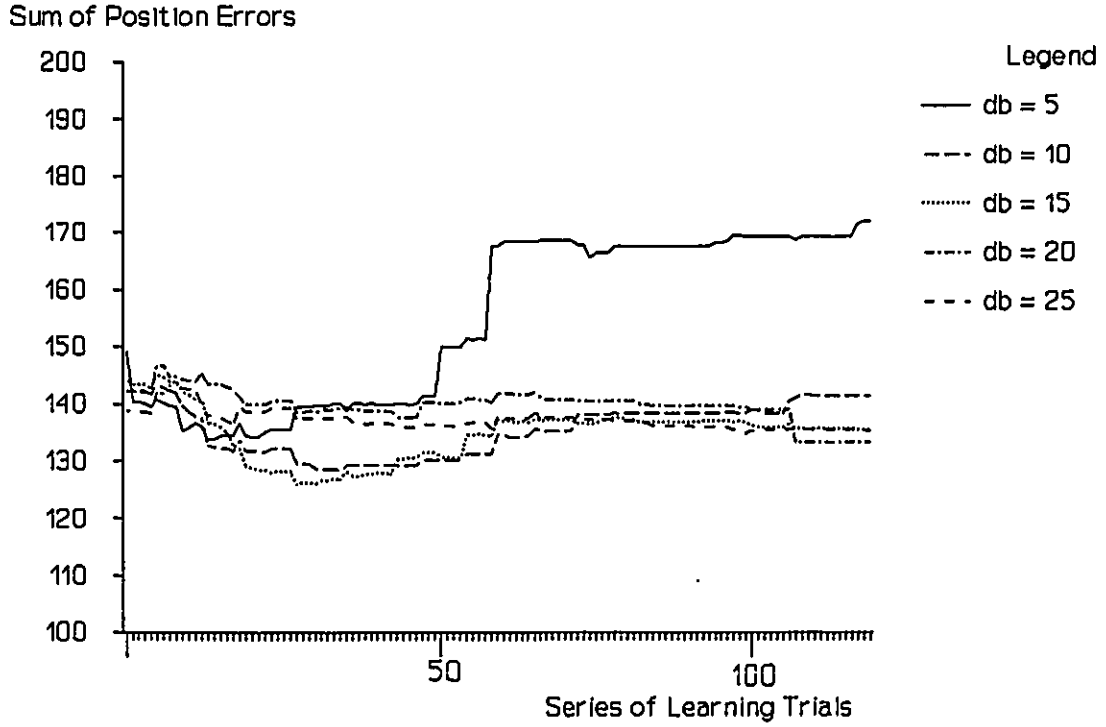


Figure 6.63 Velocity Learning at 90 cm/s (Quasi-Static Base)(Traj 4)

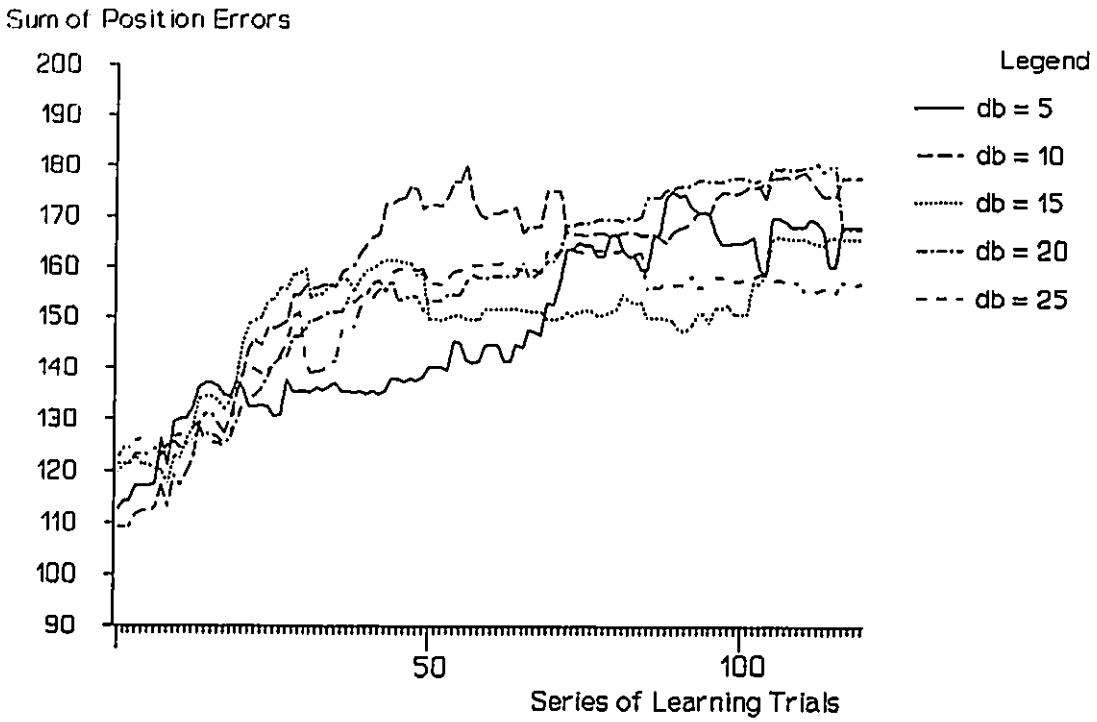


Figure 6.64 Velocity Learning at 90 cm/s (Quasi-Static Base)(Traj 5)

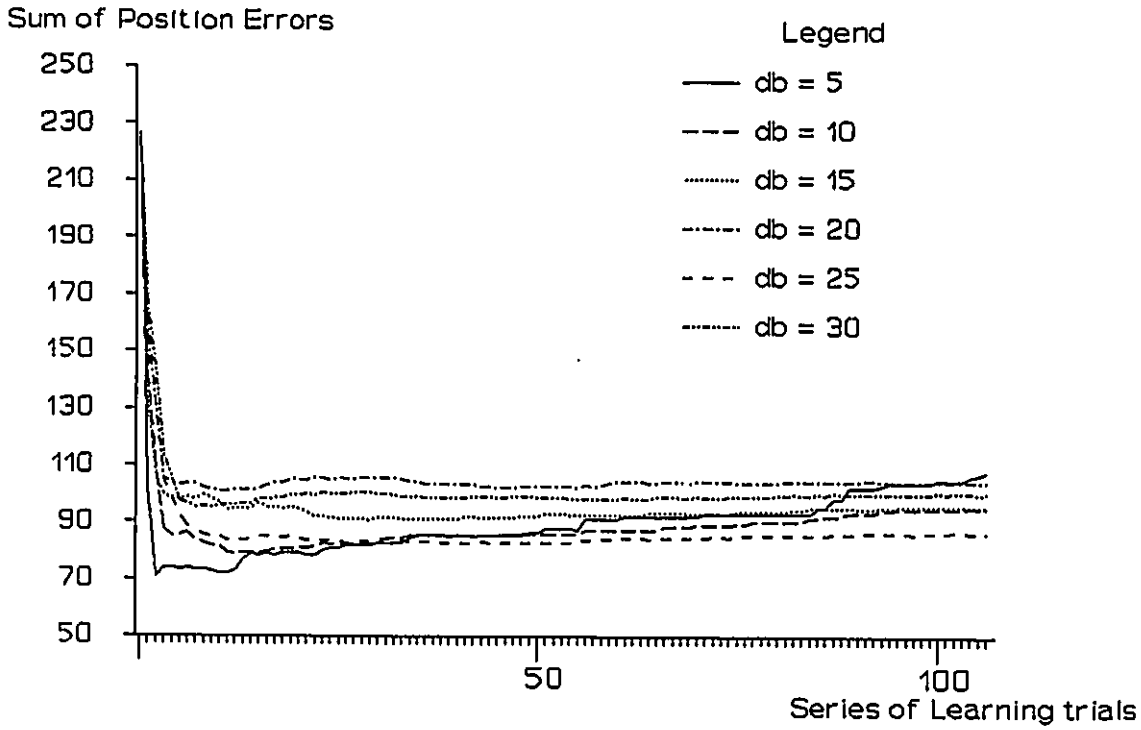


Figure 6.65 Configuration Learning at 60 cm/s with Stop Pt Isolation (Traj 1)

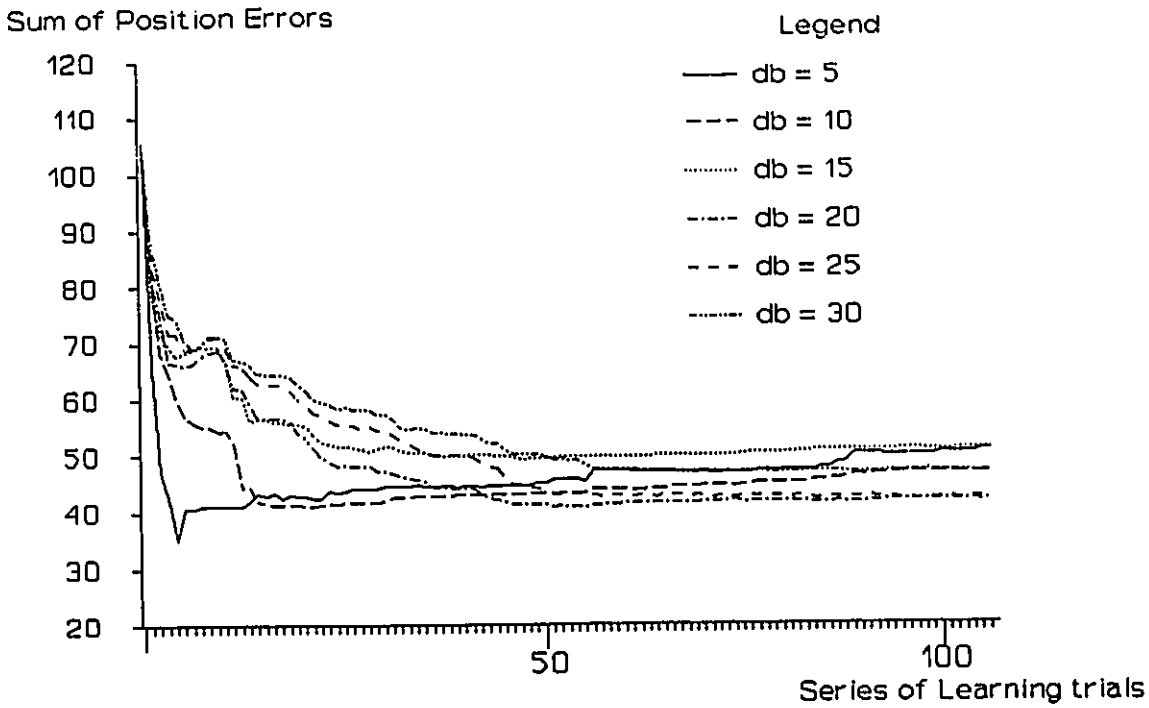


Figure 6.66 Configuration Learning at 60 cm/s with Stop Pt Isolation (Traj 2)

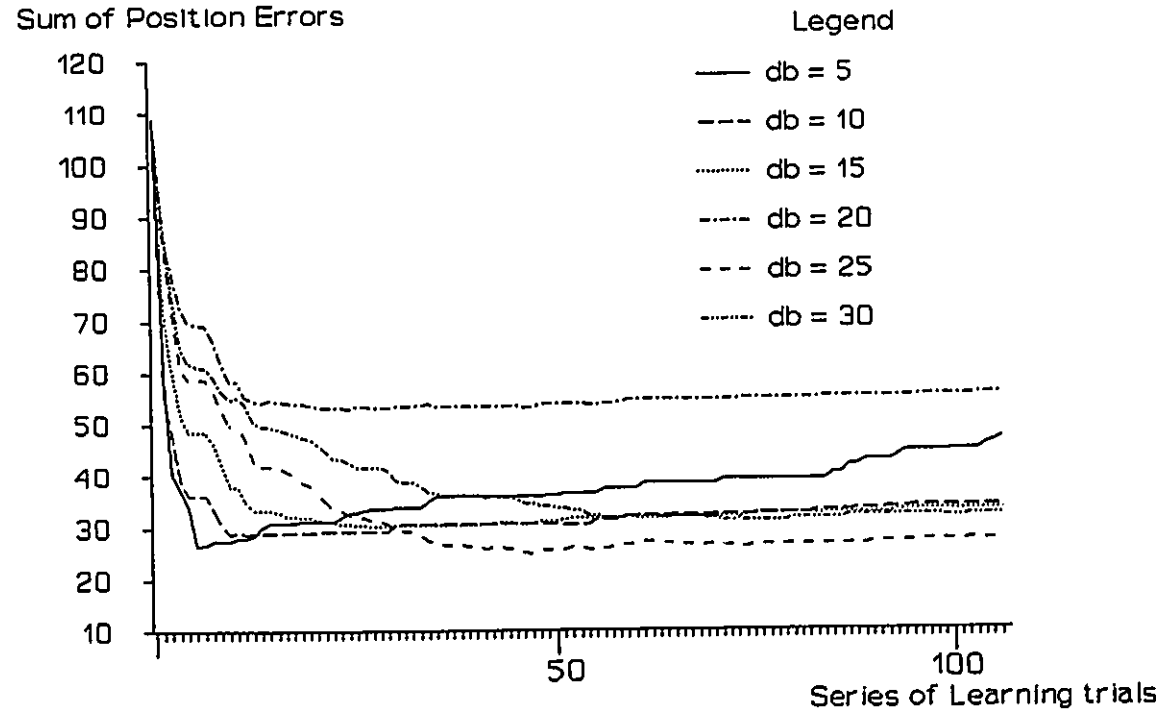


Figure 6.67 Configuration Learning at 60 cm/s with Stop Pt Isolation (Traj 3)

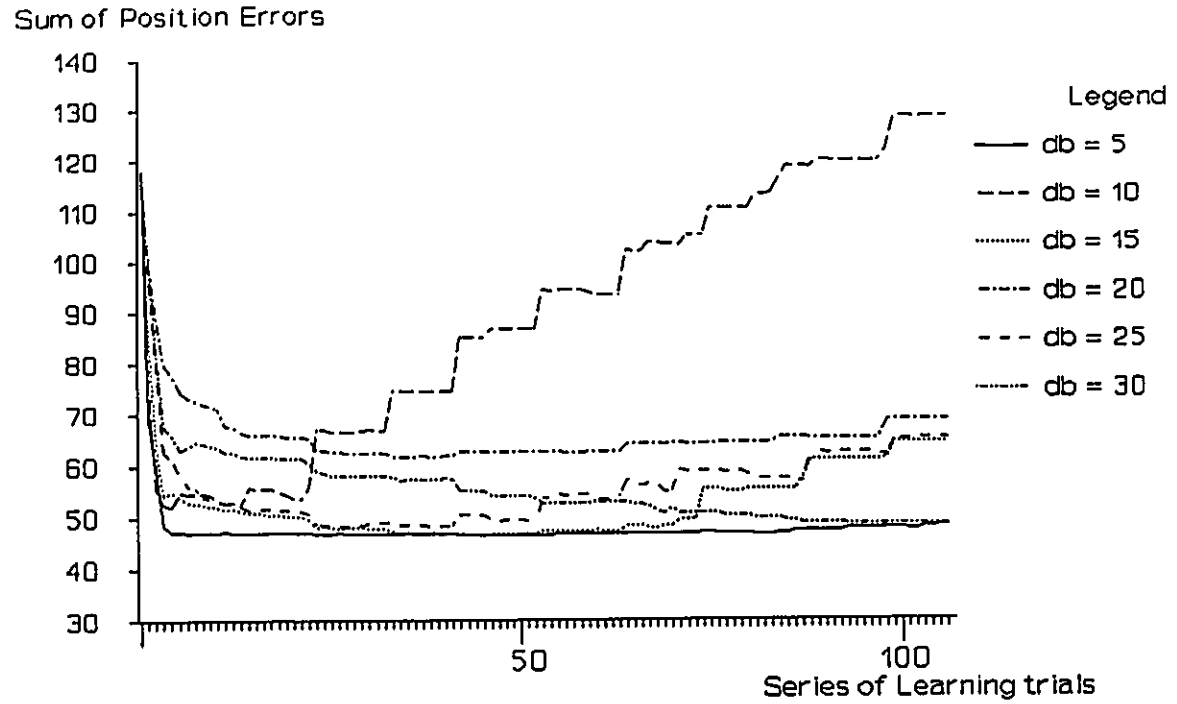


Figure 6.68 Configuration Learning at 60 cm/s with Stop Pt Isolation (Traj 4)

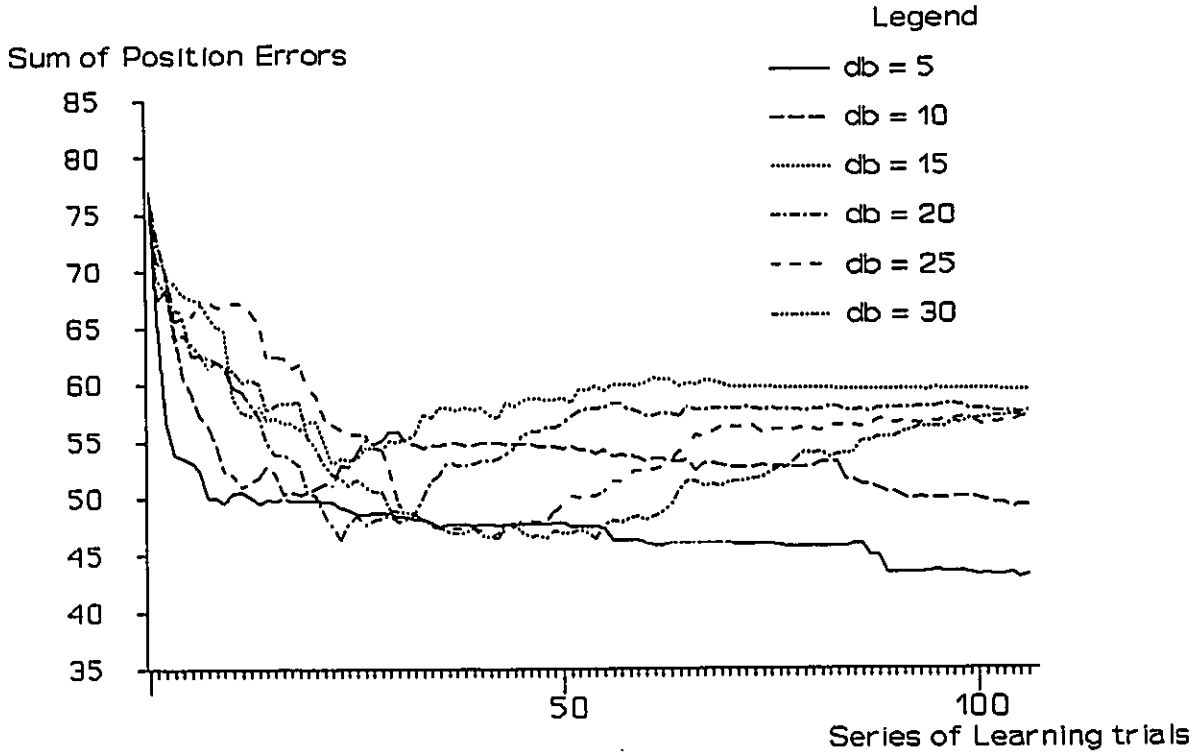


Figure 6.69 Configuration Learning at 60 cm/s with Stop Pt Isolation (Traj 5)

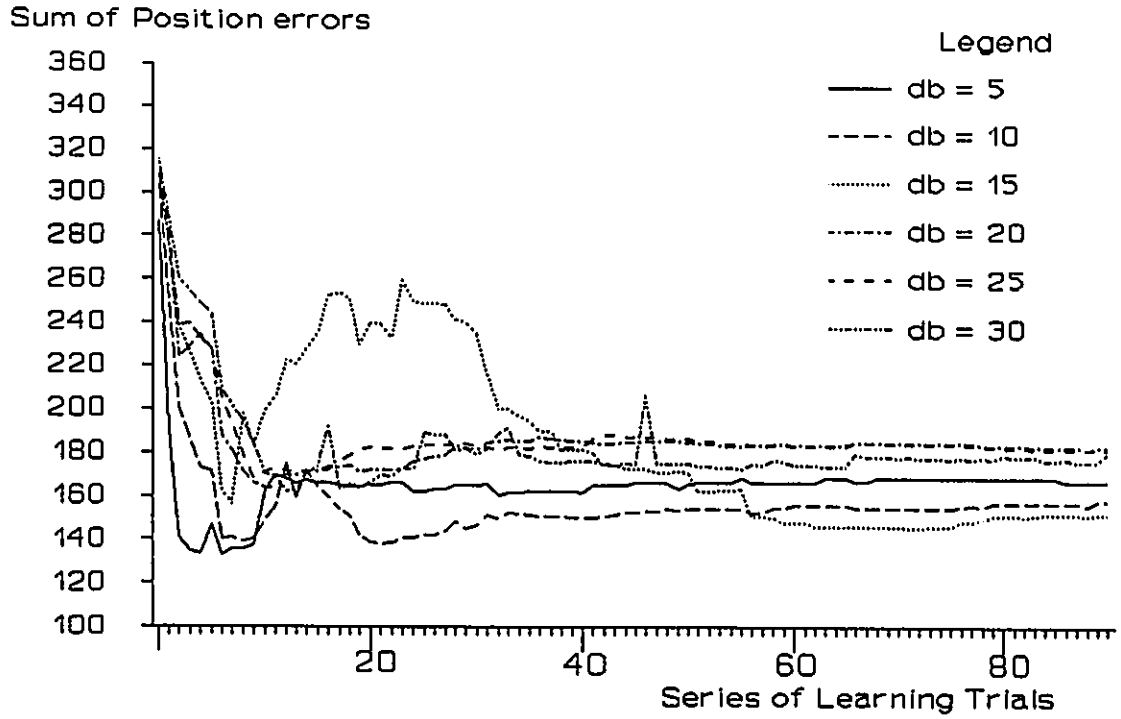


Figure 6.70 Configuration Learning at 90 cm/s with Stop Pt Isolation (Traj 1)

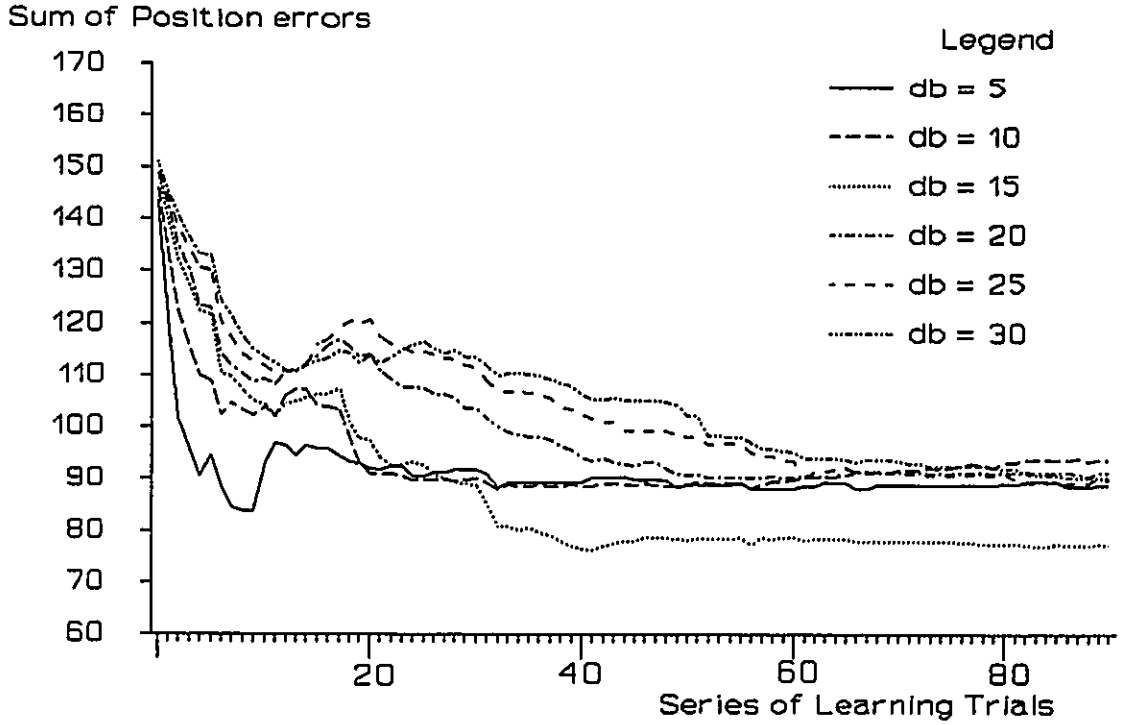


Figure 6.71 Configuration Learning at 90 cm/s with Stop Pt Isolation (Traj 2)

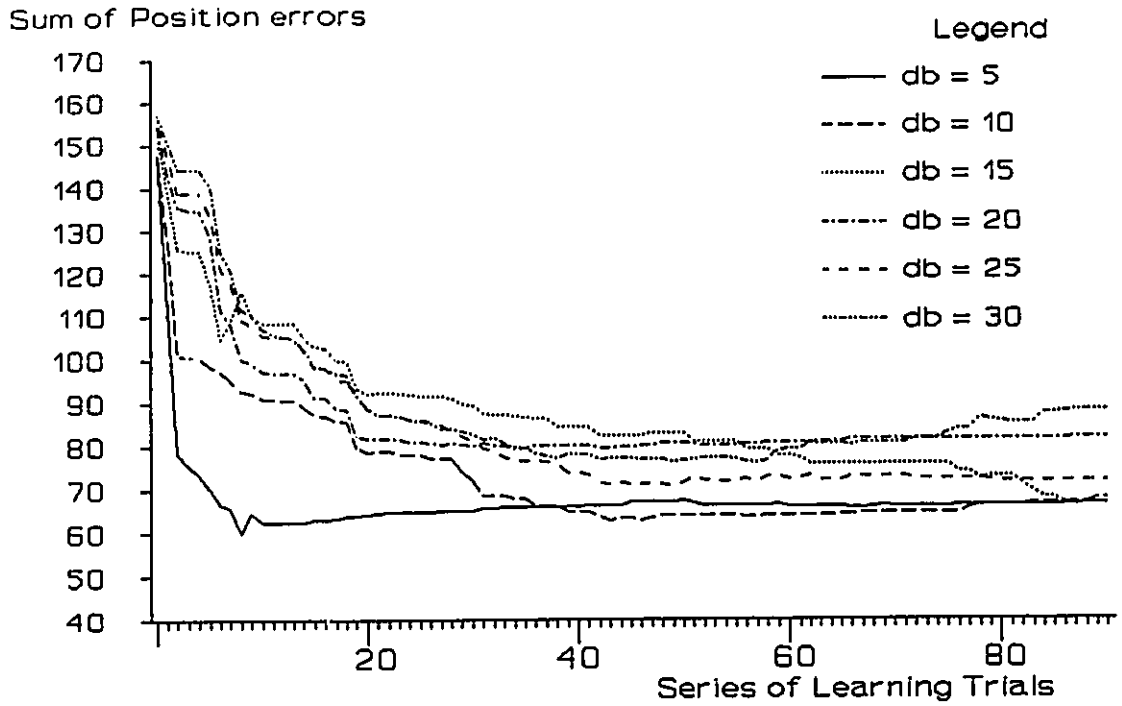


Figure 6.72 Configuration Learning at 90 cm/s with Stop Pt Isolation (Traj 3)

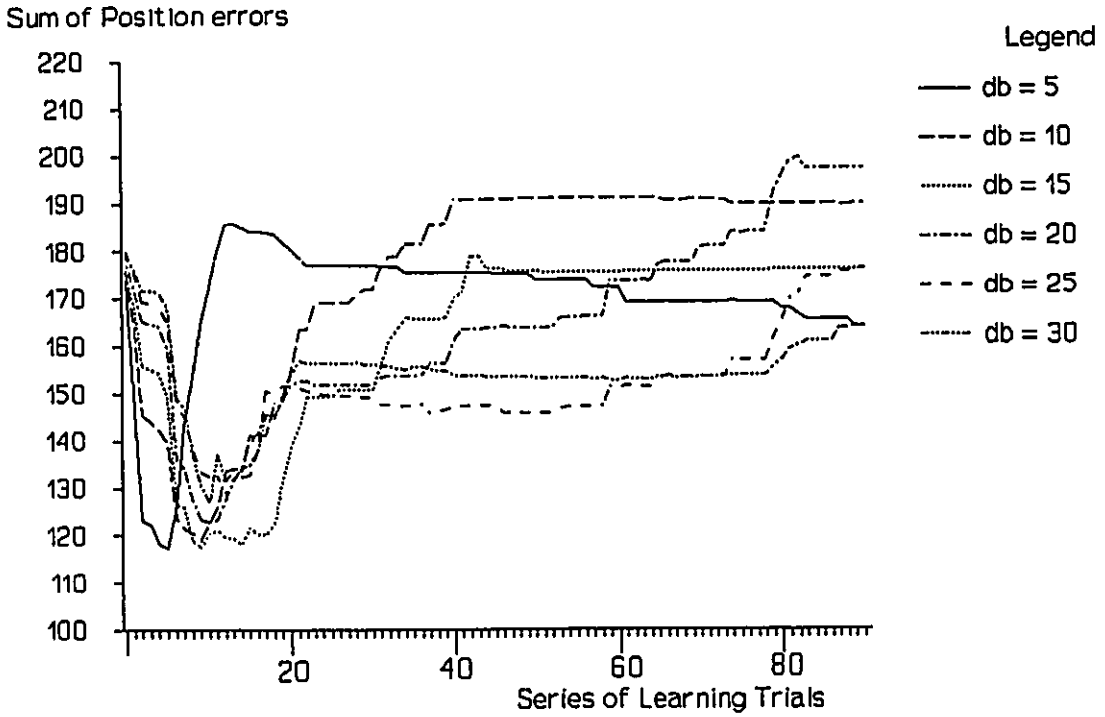


Figure 6.73 Configuration Learning at 90 cm/s with Stop Pt Isolation (Traj 4)

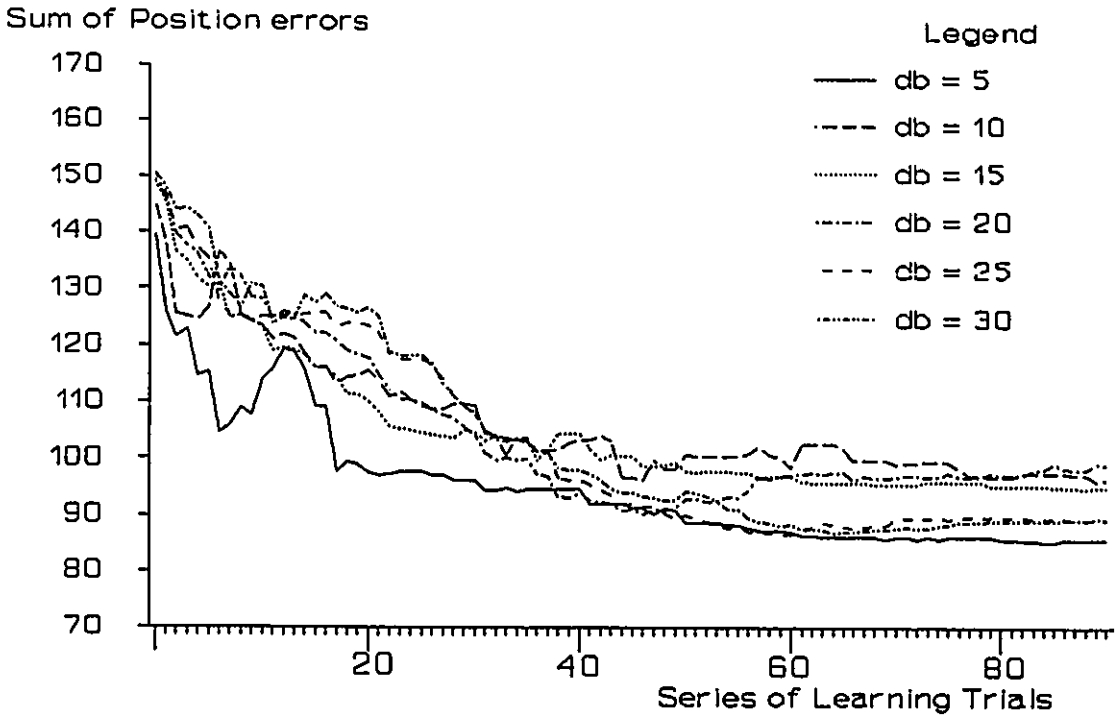


Figure 6.74 Configuration Learning at 90 cm/s with Stop Pt Isolation (Traj 5)

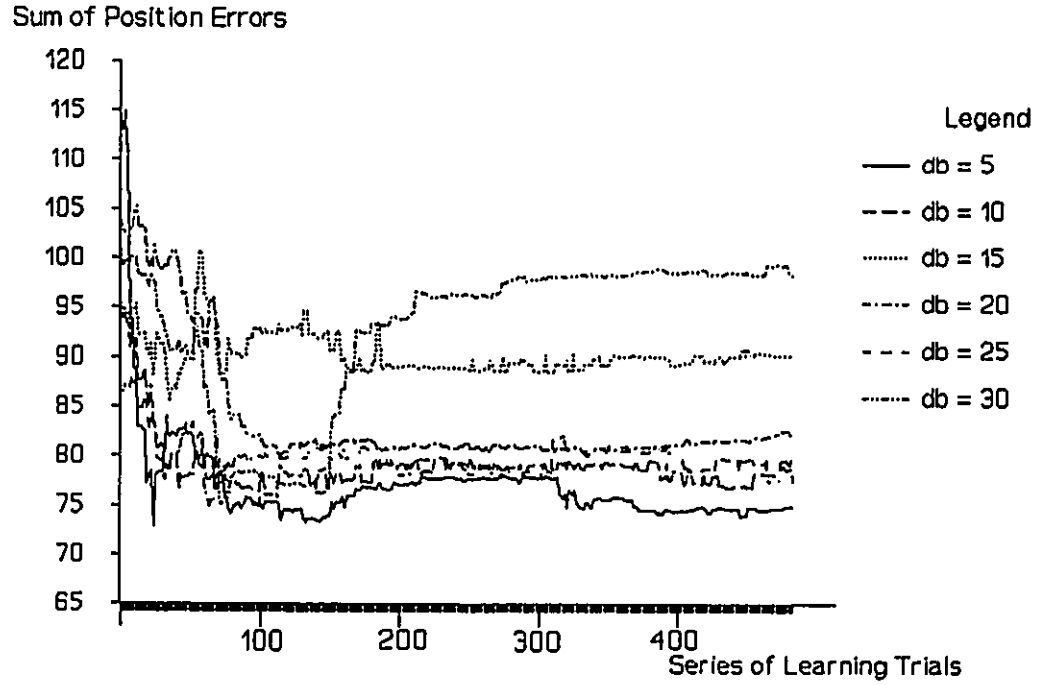


Figure 6.75 4 Zone Velocity Learning at 60 cm/s with Stop Pt Isolation (Traj 1)

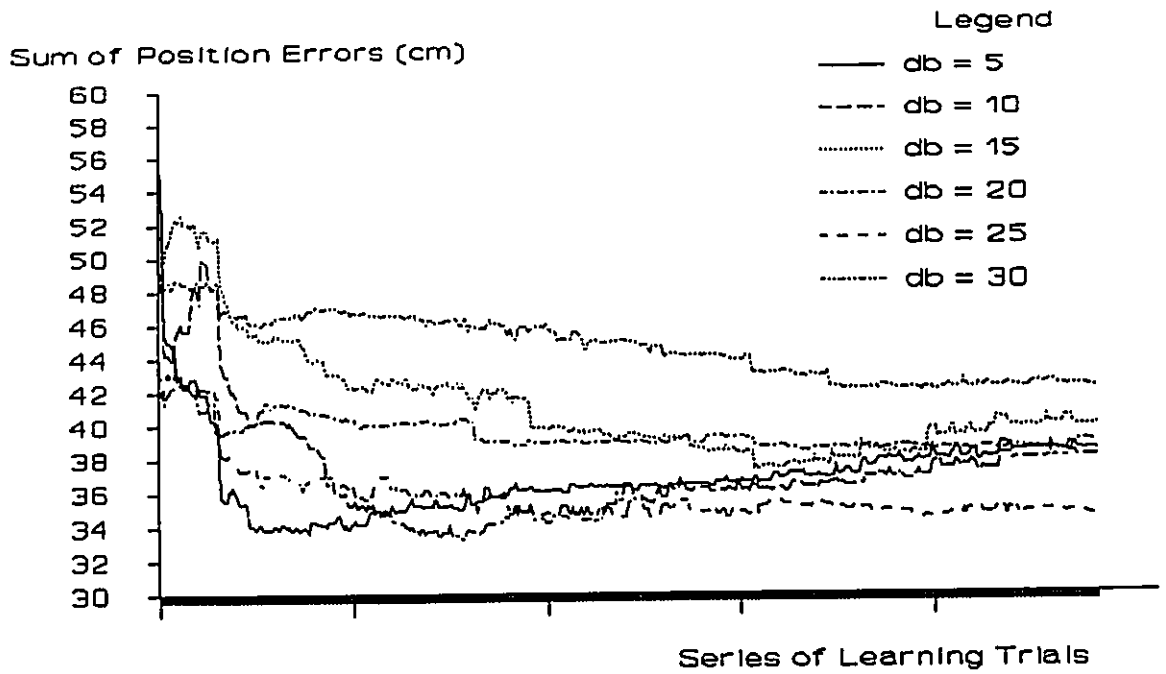


Figure 6.76 4 Zone Velocity Learning at 60 cm/s with Stop Pt Isolation (Traj 2)

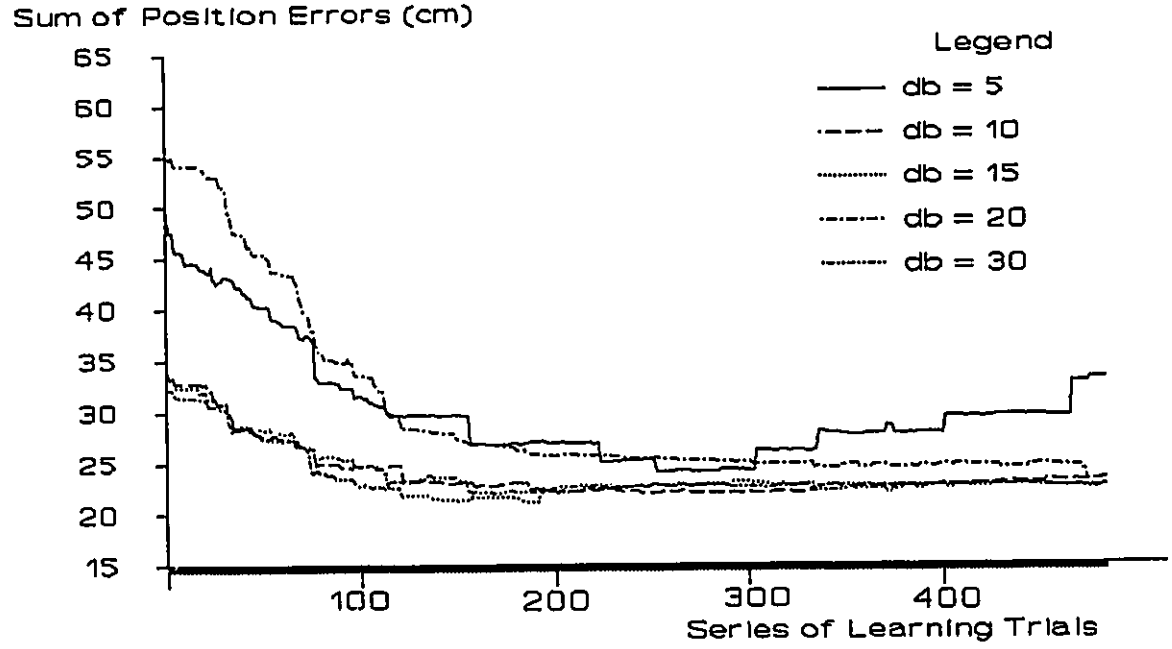


Figure 6.77 4 Zone Velocity Learning at 60 cm/s with Stop Pt Isolation (Traj 3)

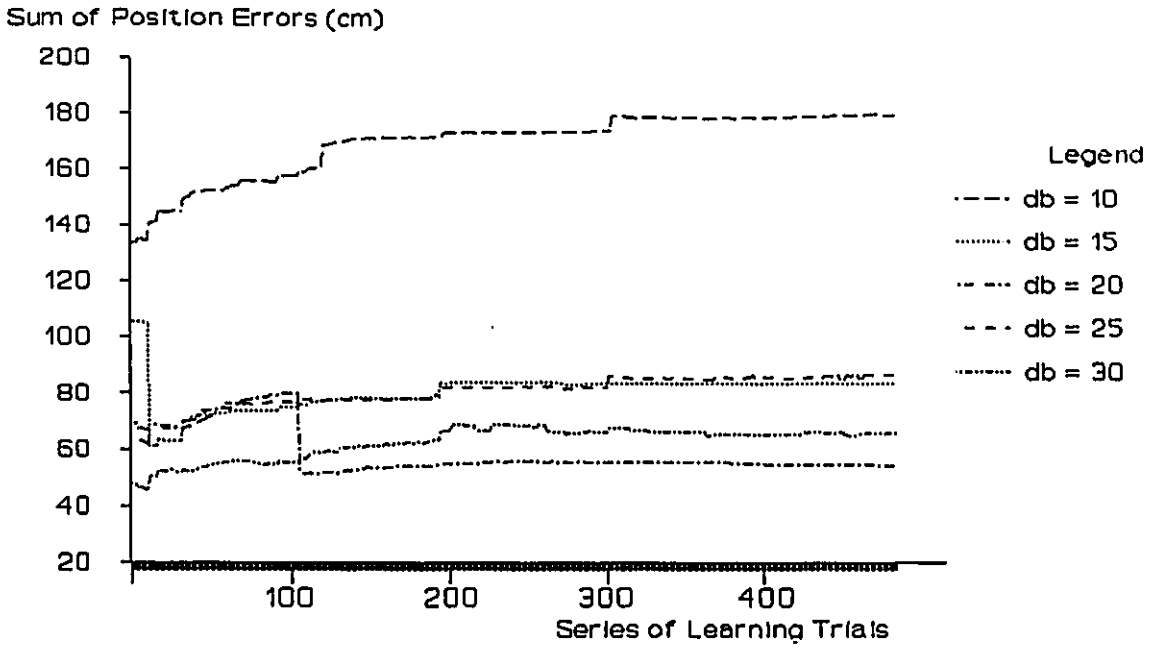
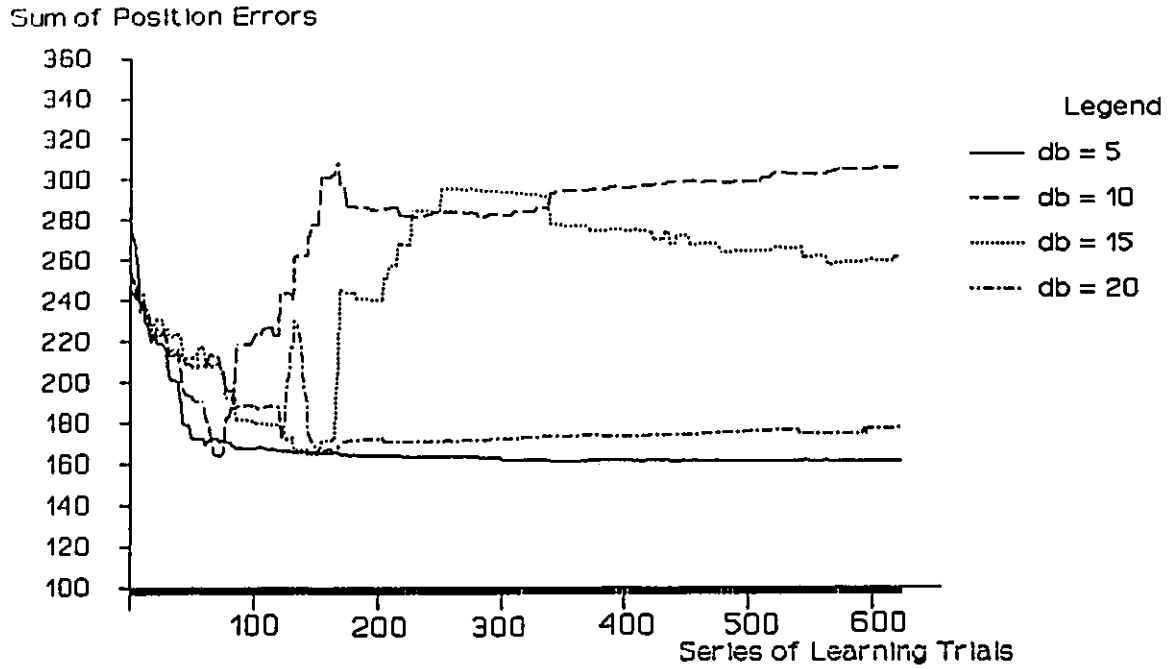


Figure 6.78 4 Zone Velocity Learning at 60 cm/s with Stop Pt Isolation (Traj 4)

Figure 6.79 4 Zone Velocity Learning at 60 cm/s with Stop Pt Isolation (Traj 5)



Figire 6.80 4 Zone Velocity Learning at 90 cm/s with Stop Pt Isolation (Traj 1)

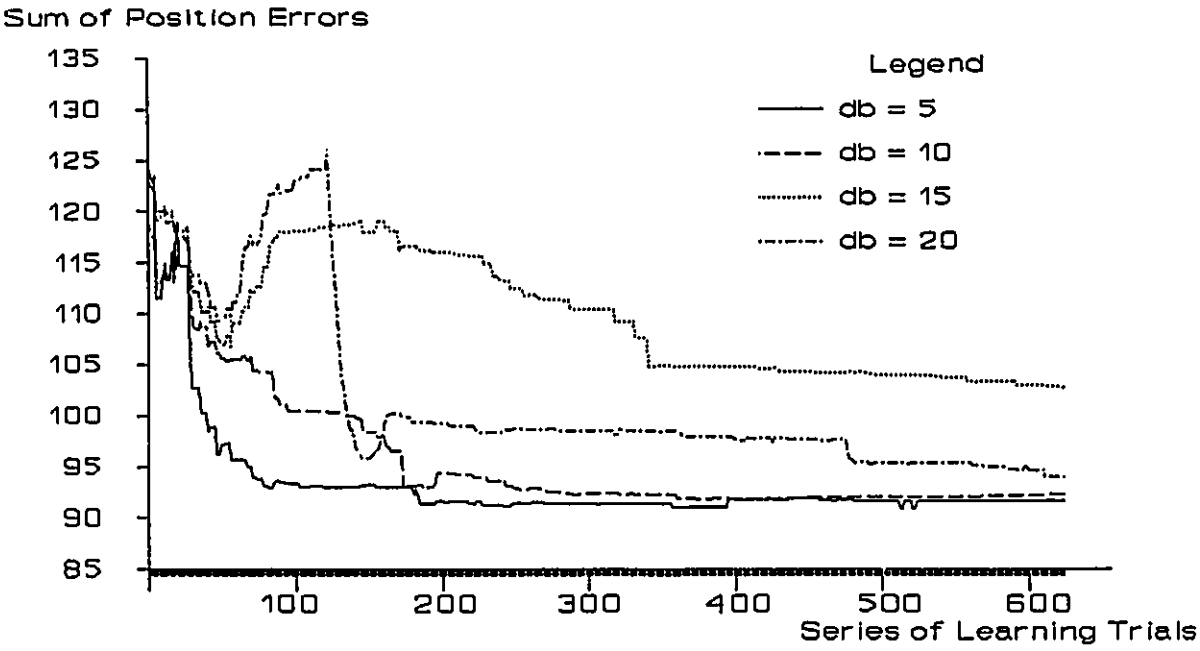


Figure 6.81 4 Zone Velocity Learning at 90 cm/s with Stop Pt Isolation (Traj 2)

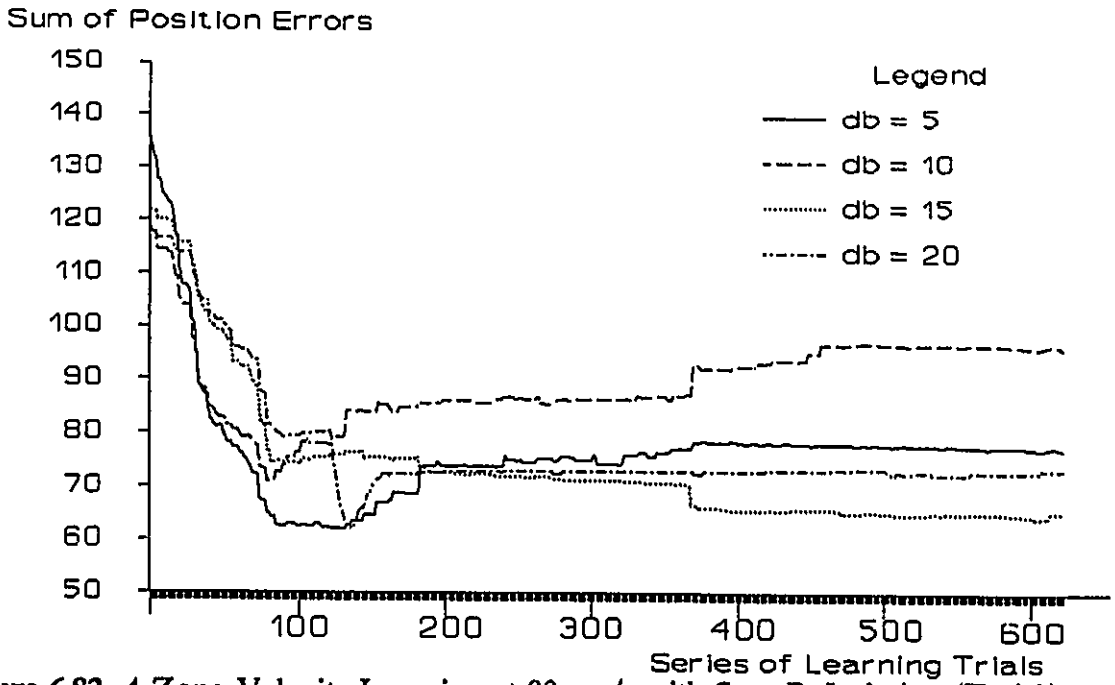


Figure 6.82 4 Zone Velocity Learning at 90 cm/s with Stop Pt Isolation (Traj 3)

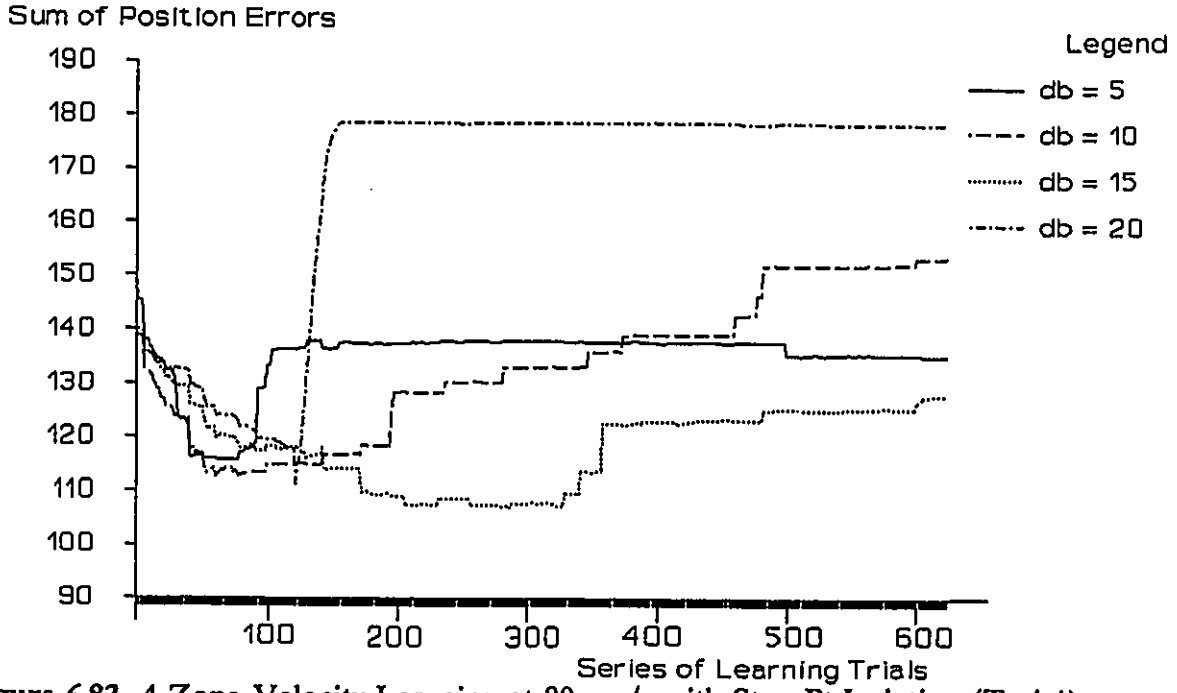


Figure 6.83 4 Zone Velocity Learning at 90 cm/s with Stop Pt Isolation (Traj 4)

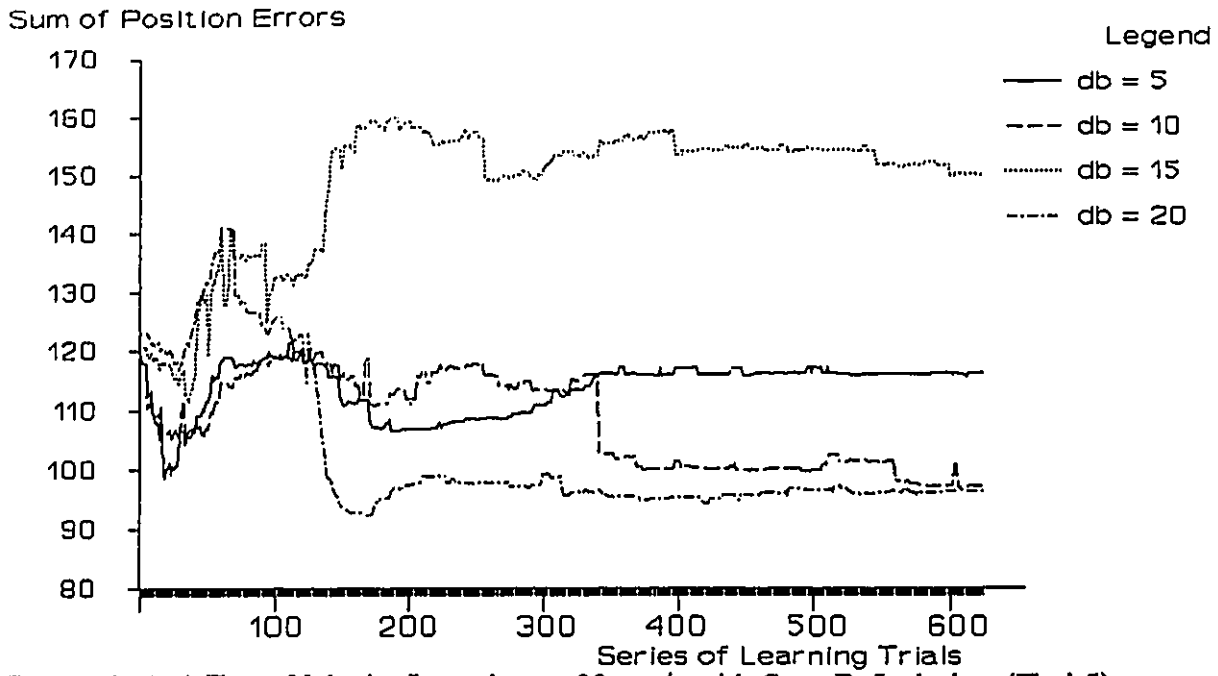


Figure 6.84 4 Zone Velocity Learning at 90 cm/s with Stop Pt Isolation (Traj 5)

Trajectory 1: Cartesian Straight Line							
Type of Learning	**Data Base Size**						
	1	**5**	**10**	**15**	**20**	**25**	**30**
Velocity = 10 cm/s (original error = 19.4)							
Random	7.7 (60)	6.9 (64)	6.0 (69)	5.9 (70)	5.8 (70)	5.7 (71)	5.7 (71)
Velocity = 60 cm/s (original error = 227)							
Repetitive	84 (63)	79 (65)	80 (65)	77 (66)	82 (64)	85 (63)	80 (65)
Random (Config)	100 (56)	91 (60)	79 (65)	86 (62)	89 (61)	88 (61)	84 (63)
Velocity Direction	94 (59)	74 (67)	76 (66)	82 (64)	91 (60)	74 (67)	76 (66)
4 Zone Velocity		80 (65)	94 (59)	90 (60)	74 (67)	76 (66)	78 (66)
Vel (10 cm/s base)		97 (57)	11 (49)	94 (59)	95 (58)	106(53)	
Random Config(Stop)	110 (51)	117 (48)	97 (57)	96 (58)	103(55)	87 (62)	101 (55)
4 Zones Vel (Stops)	94 (59)	74 (67)	77 (66)	90 (60)	82 (64)	79 (65)	99 (56)
Velocity = 90 cm/s (original error = 325)							
Repetitive	148 (54)	160 (50)	154 (52)	168 (48)	154 (52)	150 (53)	166 (48)
Random (Config)	197 (49)	178 (55)	166 (59)	171 (57)	175 (56)	175 (56)	174 (56)
Vel Direction		162 (49)	188 (42)	158 (51)	171 (47)	167 (48)	177 (45)
4 Zone Velocity		160 (50)	187 (42)	356{10}	332 {3}	151 (53)	
Vel (10 cm/s base)		146 (55)	151 (53)	164 (49)	157 (51)	156 (52)	
Random Config(Stop)	187 (42)	167 (48)	158 (51)	152 (53)	182 (43)	181 (44)	180 (44)
4 Zone Vel (Stops)		161 (50)	309 (4)	264 (18)	176 (45)		

Table 6.2 Results of Zone Learning Method Strategies (Test Path 1)

Trajectory 2: Cartesian Straight Line							
Type of Learning	**Data Base Size**						
	1	**5**	**10**	**15**	**20**	**25**	**30**
Velocity = 10 cm/s (original error = 8.7)							
Random	3.5 (60)	3.2 (63)	2.5 (71)	2.6 (70)	2.5 (71)	2.5 (71)	2.5 (71)
Velocity = 60 cm/s (original error = 106)							
Repetitive	36 (66)	43 (59)	34 (68)	34 (68)	41 (61)	32 (70)	32 (70)
Random Config	50 (53)	44 (58)	47 (56)	50 (53)	38 (64)	42 (60)	46 (57)
Vel Direction	46 (57)	35 (67)	40 (62)	44 (58)	37 (65)	36 (65)	43 (59)
4 Zone Velocity		38 (64)	41 (61)	49 (54)	36 (66)	40 (62)	46 (57)
Vel(10 cm/s base		38 (64)	44 (58)	59 (44)	58 (45)	59 (44)	
Ran Con (Stop)	54 (49)	54 (49)	47 (56)	50 (53)	42 (60)	42 (60)	47 (56)
4 Zone Vel (Stops)	46 (57)	38 (64)	37 (65)	38 (64)	39 (63)	35 (67)	42 (60)
Velocity = 90 cm/s (original error = 153)							
Repetitive	76 (50)	81 (47)	83 (46)	105 (31)	82 (46)	83 (46)	95 (38)
Random Config	107(30)	100 (35)	89 (42)	89 (42)	90 (41)	101 (34)	105(31)
Vel Direction		91 (40)	89 (42)	84 (45)	84 (45)	89 (42)	83 (46)
4 Zone Velocity		86 (44)	81 (47)	83 (36)	90 (31)	102 (33)	
Vel(10 cm/s base		89 (42)	96 (37)	110 (28)	112 (27)	113 (26)	
Ran Config(Stop)	106(31)	88 (42)	93 (39)	77 (50)	91 (40)	90 (41)	89 (42)
4 Zone Vel(Stop)		91 (40)	92 (40)	102 (33)	94 (39)		

Table 6.3 Results of Zone Learning Method Strategies (Test Path 2)

Trajectory 3: Cartesian Straight Line							
Type of Learning	**Data Base Size**						
	1	**5**	**10**	**15**	**20**	**25**	**30**
Velocity = 10 cm/s (original error = 8.9)							
Random	3.7 (58)	3.0 (66)	2.3 (74)	2.2 (75)	2.2 (75)	2.1 (76)	2.0 (77)
Velocity = 60 cm/s (original error = 110)							
Repetitive	21 (81)	21 (81)	23 (79)	25 (77)	19 (83)	19 (83)	18 (84)
Random (Config)	45 (59)	34 (69)	25 (77)	29 (74)	29 (74)	28 (74)	30 (73)
Velocity Direction	35 (68)	22 (80)	21 (81)	23 (79)	22 (80)	22 (80)	27 (75)
4 Zone Velocity		24 (78)	20 (82)	22 (80)	20 (82)	21 (81)	22 (80)
Velocity(10 cm/s base)		22 (80)	28 (74)	32 (71)	37 (66)	40 (64)	
Random Config (Stops)	53 (52)	50 (54)	34 (69)	33 (70)	55 (50)	28 (74)	40 (64)
4 Zone Vel (Stops)	51 (53)	29 (74)	23 (79)	23 (79)	25 (77)	21 (81)	22 (80)
Velocity = 90 cm/s (original error = 162)							
Repetitive	61 (62)	67 (59)	67 (59)	59 (64)	79 (51)	71 (56)	61 (62)
Random (Config)	98 (39)	75 (54)	77 (52)	67 (59)	71 (56)	77 (52)	79 (51)
Vel Direction		70 (57)	79 (51)	59 (64)	71 (56)	80 (52)	84 (51)
4 Zone Velocity		72 (56)	66 (59)	62 (62)	62 (62)	72 (56)	
Vel (10 cm/s base)		61 (62)	66 (59)	71 (56)	76 (53)	77 (52)	
Random Config (Stops)	86 (47)	66 (59)	67 (59)	67 (59)	82 (49)	72 (56)	88 (46)
4 Zone Vel (Stops)		80 (51)	95 (41)	66 (59)	73 (55)		

Table 6.4 Results of Zone Learning Method Strategies (Test Path 3)

Trajectory 4: Cartesian Straight Line							
Type of Learning	**Data Base Size**						
	1	**5**	**10**	**15**	**20**	**25**	**30**
Velocity = 10 cm/s (original error = 11.1)							
Random	3.6 (68)	2.5 (77)	2.1 (81)	1.9 (83)	1.8 (84)	1.7 (85)	1.7 (85)
Velocity = 60 cm/s (original error = 119)							
Repetitive	55 (54)	51 (57)	43 (64)	45 (62)	42 (65)	39 (67)	39 (67)
Random Config	57 (53)	112 (16)	116 (2)	50 (48)	86 (28)	60 (50)	57 (52)
Vel Direction	56 (53)	111 (7)	120 (0)	110 (8)	84 (29)	89 (25)	95 (21)
4 Zone Velocity		134 {13}	153 {29}	85 (29)	69 (42)	62 (48)	68 (43)
Vel(10 cm/s base)		58 (51)	49 (59)	48 (60)	53 (55)	46 (61)	
Ran Conf (Stop)	53 (55)	49 (59)	131 {10}	64 (46)	68 (43)	65 (45)	48 (60)
4 Zone Vel(Stops	89 (25)	87 (27)	178 {50}	83 (30)	56 (53)	85 (29)	65 (45)
Velocity = 90 cm/s (original error = 187)							
Repetitive	113 (40)	133 (29)	131 (30)	128 (32)	128 (32)	139 (26)	127 (32)
Random Config	132 (29)	149 (20)	168 (10)	171 (9)	213 {14}	195 {4}	194 {4}
Vel Direction		225 {20}	205 {10}	226 {21}	195 {4}	232 {24}	224 {20}
4 Zone Velocity		190 {2}	184 (2)	174 (7)	211 {13}	183 (2)	
Vel(10 cm/s base)		192 {3}	141 (25)	135 (28)	133 (29)	135 (28)	
Ran Conf (Stop)	125 (31)	167 (11)	189 {1}	175 (6)	197 {5}	176 (6)	164 (12)
4 Zone Vel(Stops		136 (27)	151 (19)	143 (23)	178 (5)		

Table 6.5 Results of Zone Learning Method Strategies (Test Path 4)

Trajectory 5: Oblique Rectangle							
Type of Learning	**Data Base Size**						
	1	**5**	**10**	**15**	**20**	**25**	**30**
Velocity = 10 cm/s (original error = 32.8)							
Random	11.7 (64)	10.5 (68)	9.5 (71)	9.7 (70)	10. (69)	9.9 (70)	9.9 (70)
Velocity = 60 cm/s (original error = 78)							
Repetitive	35 (55)	25 (68)	73 (6)	36 (54)	86 {10}	63 (19)	38 (51)
Random Config	40 (49)	65 (17)	80 {3}	73 (6)	85 {9}	59 (24)	79 {1}
Vel Direction	59 (24)	87 {12}	117 {50)}	94 {21}	100 {28}	74 (5)	100 {28}
4 Zone Velocity		92 {18}	100 {28}	88 {13}	121 {55}	99 {27}	111 {42}
Vel(10 c/s base		69 (11)	53 (39)	68 (13)	65 (17)	64 (12)	
Ran Conf(Stops)	42 (46)	42 (46)	47 (40)	57 (27)	58 (26)	55 (29)	56 (28)
4 Zone Vel(Stop	61 (12)	70 (10)	73 (6)	67 (14)	78 (0)	68 (13)	64 (18)
Velocity = 90 cm/s (original error = 153)							
Repetitive	98 (36)	119 (22)	106 (31)	105 (31)	120 (22)	100 (35)	122 (21)
Random Config	121 (21)	137 (10)	151 (1)	155 {1}	178 {16}	156 {2}	161 {5}
Vel Direction		151 (1)	175 {14}	175 {14}	175 {14}	166 {8}	171 {12}
4 Zone Velocity		190 {24}	183 {20}	174 {14}	210 {37}	183 {20}	
Vel(10 c/s base		168 {10}	177 {16}	165 {8}	167 {9}	156 {2}	
Ran Conf(Stops)	99 (35)	86 (44)	96 (37)	95 (38)	99 (35)	89 (42)	89 (42)
4 Zone Vel(Stop		114 (25)	97 (37)	153 (0)	96 (37)		

Table 6.6 Results of Zone Learning Method Strategies (Test Path 5)

6.4 Interpretation of the Test Results for the Different Approaches

6.4.1 Repetitive Learning

The use of the repetitive learning strategy resulted in significant reductions in the value of the position error objective function β. For the operating velocity of 60 cm/s the reductions range from 60 to 80 percent with a few exceptions. (Tables 6.2 to 6.6) The improvement for the operating velocity of 90 cm/s is slightly less (45 to 65 percent for test paths 1,2 and 3 and 20 to 35 percent for test paths 4 and 5).

While repetitive learning had been demonstrated before (Chapter 2) this particular application of repetitive learning (zone method) is unique. The successful results demonstrate the feasibility of the method and that the velocity feedback gain is a good parameter to adjust in order to improve manipulator performance. The learning rate for this strategy is fast as can be observed in Figures 6.10 to 6.14 for the 60 cm/s operating velocity and 6.15 to 6.19 for an operating velocity of 90 cm/s. This is to be expected since the same zones are continually being traversed under similar conditions. Poorer results were obtained with test path 5, especially at 90 cm/s. This is due to the stop points within the path and will be discussed later.

6.4.2 Random Configuration Learning

The random configuration zone learning approach was able to greatly reduce the trajectory position errors for paths 1, 2 and 3. The reductions are in the range to 60 to 75 percent for manipulator operating velocities of 10 cm/s and 60 cm/s (Tables 6.2 to 6.6). These reductions are almost as good as those obtained with the repetitive learning strategy. The improvements are slightly less (35 to 55 percent) for a 90 cm/s manipulator operating velocity.

Although the amount of improvement was not as large as those for the repetitive learning, the success of this strategy supports a major objective of the work; that of transferring knowledge between trajectories in the work volume. Two trajectories are considered related when they both traverse at least one of the configuration zones. In this case the knowledge gained in one may be applied to the other in the common zones. This learning approach accumulates knowledge concerning the robot itself independent of trajectory. However, the learning rate for this strategy, as expected, is slower than that of repetitive learning. The learning curves are shown in Figures 6.20 to 6.22 for a 60 cm/s operating velocity and 6.25 to 6.27 for a 90 cm/s operating velocity for test paths 1 to 3.

The learning was much less successful and more erratic when the method was applied to test paths 4 and 5 (Figures 6.23 to 6.24 and 6.28 to 6.29). This difficulty is due to the intermediate stop points within those paths. Other strategies which will be discussed later have obtained better results for paths 4 and 5 using the operating velocities of 60 and 90 cm/s.

Unlike the higher velocity tests, the error reductions obtained at the slow operating velocity of 10 cm/s did not vary with the trajectory (Figures 6.30 to 6.34). The error reductions recorded for test path 4 (68% to 85%) and test path 5 (64% to 70%) were just as large as those recorded for the other three test paths (Tables 6.2 to 6.6). This is because of the slow operating velocity of the manipulator. In this case (10 cm/s) the intermediate stop points have almost no effect on the learning and test process since each of the links almost stops at every point anyway. This data base will be used later in combination with a velocity learning strategy (quasi-static method) to will alleviate some of the problems posed by intermediate trajectory stop points.

6.4.1 Velocity Zone Learning

Two applications of the velocity zone method of learning were tested. They are referred to as "velocity direction" learning and "4 zone velocity" learning. These are essentially the same strategy with the exception that for the former only two zones are defined for the velocities (positive and negative) while 4 zones are defined in the "4 zone" strategy. This learning process begins after the completion of the configuration zone learning. Therefore, the starting point is already much improved over the initial fixed gain system.

Both strategies resulted in further improvements to manipulator performance for test paths 1, 2 and 3 (Tables 6.2 to 6.6). Additional improvements of approximately 10 percent were obtained above the configuration zone results for a manipulator velocity of 60 cm/s. This yields a total improvement from the original system of in the range of 60 to 80 percent. For both types

of velocity learning the rates of improvement were more consistent for trajectories 2 and 3 (Figures 6.37 to 6.37 and 6.46 to 6.47) than for trajectory 1 (Figures 6.35 and 6.45).

However, at 90 cm/s, the velocity direction was able to improve the result of only test trajectory 2 (Figure 6.41) by approximately 10 percent over the result obtained through configuration learning. The method was not able to consistently reduce the errors for the other 4 trajectories (Figures 6.40 and 6.42 to 6.44). The 4 zone velocity learning method fared slightly better as the errors from trajectories 2 and 3 were generally reduced by 5 to 10 percent (Figures 6.51 and 6.52) from the configuration learning result although variation between sizes of data bases were noted. Again trajectories 4 and 5 showed no improvement (Figures 6.53 and 6.54). The velocity learning is less predictable and less consistent than that of the configuration method.

6.4.3 Velocity Learning using a Quasi-Static Base

This strategy was implemented to achieve two goals: (1) to provide a single data base which will be applicable to all manipulator operating velocities rather than having a different data base for each velocity, and (2) to rectify the learning problems which were encountered with test paths 4 and 5 due to the existence of intermediate stop points.

The learning process is very successful for paths 1, 2 and 3 for both manipulator operating velocities of 60 cm/s (Figures 6.55 to 6.57) and 90 cm/s (Figures 6.60 to 6.62). The trajectory error reductions were from 50 to 70 percent for the 60 cm/s velocity but were slight less at

between 30 and 60 percent for the higher velocity of 90 cm/s (Tables 6.2 to 6.6).

This strategy was partially successful in addressing its second objective. The results for one of the two problem paths was very good. The total improvements in test path 4 were 50 to 60 percent for the 60 cm/s velocity (Figure 6.58) and 25 to 30 percent for the 90 cm/s velocity (Figure 6.63). Small improvements from the original fixed gain system were registered for test path 5 at 60 cm/s (10 to 20 percent) while slight error increases were noted for 90 cm/s. However, from the plots (Figures 6.59 and 6.64) it is seen that, in this case, the result would have been better using the 10 cm/s base without the velocity adaptation.

6.4.4 Random Configuration Learning with Stop Point Isolation

The method of isolating intermediate stop points proved successful in reducing the errors compiled during the execution of path 5 at both operating velocities of 60 and 90 cm/s (Figures 6.69 and 6.74). Reductions ranging from 30 to 45 percent were achieved using this method. (Table 6.6) This strategy was the most successful at reducing the position errors for test trajectory 5. This method was also able to improve the results for test path 4 at 60 cm/s (40 to 60 percent) (Figure 6.68) but could only provide marginal improvements up to 12 percent with one value of 31 percent for the 90 cm/s velocity (Figure 6.73). Although that strategy is better than most others tested, this was not as good as the velocity built upon a quasi-static base.

For the first three test paths the results were very good although they were generally

slightly less than the normal configuration learning approach (Figures 6.65 to 6.67 and 6.70 to 6.72). The results generally ranged from 40 to 60 percent rather than the 50 to 70 percent improvement obtained by the configuration learning without the stop point isolation. This is because a few of the intermediate points of these paths are stop points. In these cases, the link damping is increased resulting in slower response.

6.4.5 4 Zone Velocity Learning with Stop Point Isolation

This strategy, applied after the completion of configuration learning, was able to even further improve the result in the cases of test paths 1, 2 and 3 at 60 cm/s (Figures 6.75 to 6.77). These reductions were in the range of 60 to 80 percent (Tables 6.2 to 6.9). However, the results were generally poorer for the higher operating velocity of 90 cm/s (Figures 6.80 to 6.82). In this case the error reductions ranged from 4 to 50 percent for path 1, about 40 percent for path 2 and 40 to 60 percent for path 3. This method was not successful in further improving the results already obtained by the configuration learning with stop points for test paths 4 and 5 for either of the two velocities tested (Figures 6.78 to 6.79 and 6.83 to 6.84).

6.5 Effect of Data Base Size

Because the larger number of zones allow for less damping within certain zones, the general effect of increasing the number of zones is to improve the performance of the

manipulator. This effect is seen in Figure 6.85. This plot displays the value of the objective function β (sum of position errors) for all the five test trajectories after a complete set of random configuration learning trials executed at a manipulator operating velocity of 10 cm/s. Seven points are shown for each of the five trajectories. These points represent the value of β for different sizes of data sets corresponding to partitions of 1, 5, 10, 15, 20, 25 and 30. The graphs show that significant reductions in position errors can be achieved by increasing the size of the data sets used from 1 to 5 to 10. After 10 only small increases are registered. Increasing the number of data sets is accomplished by increasing the number of configuration zones within the manipulator's work volume.

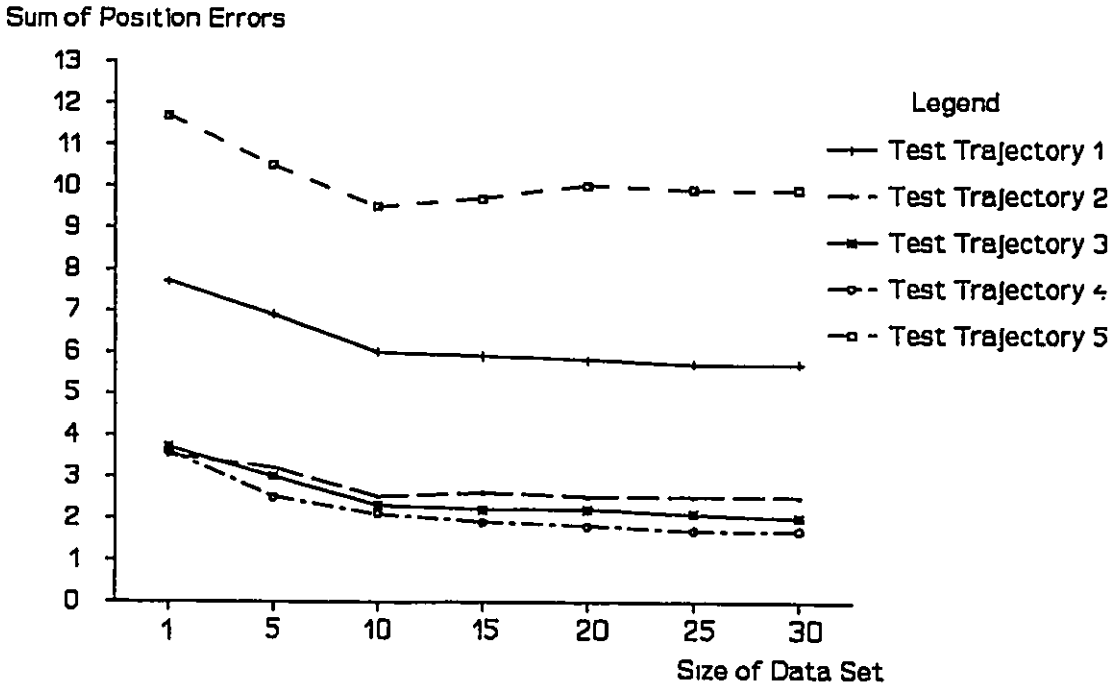


Figure 6.9: Effect of Data Base Size on Sum of Position Errors (10 cm/s)

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

7.1 Major Conclusions

One of the major objectives of this work was to develop a strategy to improve the performance of proportional controlled robot manipulators with velocity feedback without the need for major overhaul of the controller hardware or software. This objective has been attained to a great extent with the proposed zone method of learning presented in this thesis. The zone learning method allows a simple control system to be maintained yet achieve high performance by the modification of the feedback velocity gain according to specific configuration and velocity conditions. The zone learning method can be easily added to existing robot manipulators.

The performance of proportional controlled robots with velocity feedback is limited due to the fact that a linear controller is being applied to a highly non-linear system. The zone learning system in effect transforms the non-linear problem into a piece-wise linear problem where the zones result in a linearized approximation of the system.

The results of the evaluation trials demonstrate that the zone method of learning is successful in improving the performance of a robot manipulator using proportional joint controllers with velocity feedback. Learning based on manipulator configuration zones provided

very significant improvement for most trajectories. However, the errors over some of the trajectories were not significantly reduced by the learning method. This was due to intermediate stop points within the path. This difficulty was alleviated by a modified learning strategy which isolated the stop points and compensated for them.

Additional improvements were demonstrated by the implementation of a velocity zone learning process following the completion of the configuration zone learning. However, these results were neither as dramatic nor as consistent as those of configuration learning.

One important concept which was being investigated was the ability to transfer knowledge from one type of trajectory to another, provided that some commonality exists (i.e. passes through the same zones). Repetitive learning proved very successful but the knowledge gained is trajectory specific and a new trajectory requires a new set of learning trials. The zone learning method was able to successfully transfer knowledge from one trajectory to another. This was demonstrated by the random learning strategy. The zone learning method therefore results in manipulator specific knowledge rather than trajectory specific knowledge.

7.2 Future Work

The learning algorithm which was used in the zone learning method was not a sophisticated one. It was simply a "step and test" method. The results of a previous iteration were used to determine the direction for the step (increasing or decreasing the damping). A

constant step size was used which is not a very efficient search method. When overshoot occurred a limit value was set to prevent the damping from being further reduced. This learning algorithm was generally slow to converge, especially when large data bases were used. It should be possible to devise better learning algorithms by the application of optimization techniques. This would result in faster learning.

The non-repetitive (or random) learning process which was used consisted of trajectories which were randomly generated by a computer program. The purpose of the random learning was to demonstrate that, using this type of data base, knowledge can be transferred from one trajectory to another. However, since the learning was not directed, some zones were explored many more times than others. It was not a very efficient method. It should be possible to devise an algorithm which would seek out unexplored or rarely explored zones and generate trajectories which pass through them.

The existence of intermediate link stop (or change of direction) points within the robot trajectories caused problems in the learning process and for certain trajectories. This is because the learning strategy was based on the link moving through the intermediate points. A strategy was developed to alleviate this problem by isolating the stop points and overdamping the system to ensure that no overshoot occurs. However, since some inconsistencies persist the effects of the intermediate stop points on the learning process should be further investigated. This could lead to a better strategy for dealing with these points.

As this work was a proof of concept study, it was not possible to produce an optimum implementation for specific cases. Some of the factors which are left to examine are optimum data base sizes, best strategies for specific applications, optimum fixed learning step size or variable learning step, and the choice of overlapping or non-overlapping zone partition strategies.

One particular strategy of implementation which should be explored consists of sequentially increasing the number of zones as the learning process progresses. In this implementation, the learning process is initiated using a small number of zones only. Once the learning is complete, each zone can be partitioned into two, effectively doubling the number of zones. The initial multiplier values for both new subzones would be the limit value in the parent zone. The learning would then continue. This method of progressively increasing the number of zones may considerably increase the rate of learning which is slow for large numbers of zones.

The concept of zone learning was developed for a proportional controller with velocity feedback and demonstrated on a three DOF revolute robot. The implementation of the method on differently configured robots should be explored. Also the extension of this learning method to other types of controller should be investigated.

The learning algorithm and data base structure should be tested on an actual manipulator to verify the results of the simulations. More uncertainties are encountered on a real system than can be programmed into a simulator. Also a real system has some variability from day to day which is not present in a simulator. The true test for any new method is its practical application.

REFERENCES

1. Lunde E., Balchen J.G., "Practical Trajectory Learning Algorithms for Robotic Manipulators", Proceedings of the 1990 IEEE Conference on Robotics and Automation, Cincinnati, Ohio, May 13-18, 1990, pp 1970-1975.

2. Kawamura S., Miyazaki F., Arimoto S., "Iterative Learning Control for Robotic Systems", Proc. IECON '84, Tokyo, pp 393-398.

3. Kawamura S., Miyazaki F., Arimoto S., "Applications of Learning Method for Dynamic Control of Robot Manipulators", Proceedings of the 24th IEEE Conference on Decision and Control, Ft. Lauderdale, FL, December 1985, pp 1381-1386.

4. Arimoto S., Kawamura S., Miyazaki, Tumaki S., "Learning Control Theory for Dynamical Systems", Proceedings of the 24th IEEE Conference on Decision and Control, Ft. Lauderdale, FL, December 1985, pp 1375-1380.

5. Kawamura S., Miyazaki F., Arimoto S., "Realization of Robot Motion Based on a Learning Method", IEEE Transactions on Systems, Man, and Cybernetics, Vol. 18, No.1, Jan/Feb 1988, pp 126-134.

6. Kawamura S., Miyazaki F., Arimoto S., "Intelligent Control of Robot Motion Based on Learning Method", Proc. of "87 IEEE International Symposium on Intelligent Control, Philadelphia, Pennsylvania, pp 365-370.

7. Arimoto S., Miyazaki F., Kawamura S., "Motion Control of Robotic Manipulator Based on Motor Program Learning", Proc. IFAC Symp on Robot Control, Karlsruhe, FRG, 1988, pp. 169-176.

8. Tsai M.-c., Anwar G., Tomizuka M., "Discrete Time Repetitive Control for Robotic Manipulators", Proceedings of the 1988 IEEE Conference on Robotics and Automation, Philadelphia, Pennsylvania, 1988, pp. 1341-1346.

9. Cosner C., Anwar G., Tomizuka M., "Plug In Repetitive Control for Industrial Robotic Manipulators", Proceedings of the 1990 IEEE Conference on Robotics and Automation, Cincinnati, Ohio, May 13-18, 1990, pp 1516-1521.

10. Atkeson C.G., McIntyre J., "Robot Trajectory Learning Through Practice", Proceedings of the 1986 IEEE Conference on Robotics and Automation, San Francisco, California, 1986, pp. 1737-1742.

11. Guglielmo K., Sadegh N., "Experimental Evaluation of a New Robot Learning Controller", Proceedings of the 1991 IEEE Conference on Robotics and Automation, Sacramento, California, April 1991, pp. 734-739.

12. Sadegh N., Guglielmo K., "A New Learning Controller for Mechanical Manipulators", Proceedings of the American Control Conference, 1989, pp. 2827-2833.

13. Horowitz R., Kao W.-W., Boals M., Sadegh N., "Digital Implementation of Repetitive Controllers for Robotic Manipulators", Proceedings of the 1989 IEEE Conference on Robotics and Automation, Scottsdale, Arizona, May 1989, pp. 1497-1503.

14. Natarajan B.K., "Machine Learning: A Theoretical Approach", Morgan Kaufmann Publishers, Inc., San Mateo, California, 1991.

15. Widrow B., Nguygen D.H., "Neural Network for Self Learning Control Systems", IEEE Control Systems Magazine, April 1990, pp. 18-23.

16. Miller W.T., "A Nonlinear Learning Controller for Robotic Manipulators", Proceedings of the SPIE: Intelligent Robots and Computer Vision, Vol. 726, pp. 416-423, Oct 1986.

17. Miller W.T., "Real Time Experiments in Neural Network Based Learning Control During High Speed Non-Repetitive Robotic Operations", in Proceedings of the Third International Symposium on Intelligent Control, Arlington, VA, 24-26 Aug 1988, pp 513-518.

18. Gawthrop P.J., "Self-Tuning PID Controllers: Algorithms and Implementation", IEEE Transactions on Automatic Control, Vol. AC-31, No. 3, March 1986, pp. 201-209.

19. Gawthrop P.J., Nomikos P.E., "Automatic Tuning of Commercial PID Controllers for Single Loop and Multiloop Applications", IEEE Control Systems Magazine, January 1990, pp. 34-42.

20. Nishikawa Y., Sannomiya N., Ohta T., Tanaka H., "A Method for Auto-tuning of PID Parameters", Automatica, Vol. 20, No. 3, pp. 321-332, 1984.

21. Farsi M., Finch J.W., Warwick K., Tzafestas S.G., Stasinopoulos G., "Simplified PID Self Tuning Controller for Robotic Manipulators", Proceedings of the 25th Conference on Decision and Control, Athens, Greece, December 1986, pp. 1886-1887.

22. Tzafestas S.G., Stasinopoulos G., Farsi M., Finch J.W., Warwick K., "Decentralized PID Self Tuning Control of Industrial Robots", Proceedings of the 25th Conference on Decision and Control, Athens, Greece, December 1986, pp. 1888-1890.

23. Chen Y., "Parameter Fine-Tuning for Robots", IEEE Control Systems Magazine, February, 1989, pp. 35-39.

24. Craig, J.J., "Introduction to Robotics, Mechanics and Control, Second Edition", Addison-Wesley Publishing Company, New York, 1989.

25. Koivo A.J., "Fundamentals for Control of Robotic Manipulators", John Wiley & Sons, Inc., New York, 1989.

26. Emanuel P., Leff E., "Introduction to Feedback Control Systems", McGraw-Hill Book Company, New York, 1979.

27. SRI International, Gerry B. Arden, Editor in chief, "Robot Design Handbook", McGraw-Hill Book Company, New York, 1988.

28. Vukobratovic M., "Introduction to Robotics", Springer-Verlag, New York, 1989.

29. Ziegler J.G., Nichols N.B., "Optimum Settings for Automatic Controllers", Trans. ASME, Vol. 64, pp. 759-768 (Nov. 1942).

30. Korn, G.A., Korn, T.M., "Mathematical Handbook for Scientists and Engineers", Second Edition, McGraw-Hill Book Company, Toronto, 1968.

31. Paul R.P., "Robot Manipulators: Mathematics, Programming and Control", Cambridge, MA, MIT Press 1981.

32. Luh J.Y.S., "Conventional Controller Design for Industrial Robots --- a Tutorial", IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-13, No. 3, May/June 1983.

33. Specht R., Isermann R., "On-line Identification of Inertia, Friction and Gravitational Forces Applied to an Industrial Robot", selected papers from the 2nd IFAC Symposium on Robot Control, Karlsruhe, FRG, 5-7 Oct 1988, ed U. Rembold, Pergamon Press, Oxford, U.K., 1989, pp 219-224.

34. Meriam J.L., "Dynamics", John Wiley & Sons, Inc., New York, 1975.

APPENDIX A

DYNAMIC SIMULATION OF A THREE LINK REVOLUTE ROBOT

The testing of the learning method presented in this thesis was conducted by computer simulation of a three link revolute robot manipulator. The derivation of the equations of motion for the manipulator is presented in this section.

A schematic of the simulated robot is presented in Figure A-1.

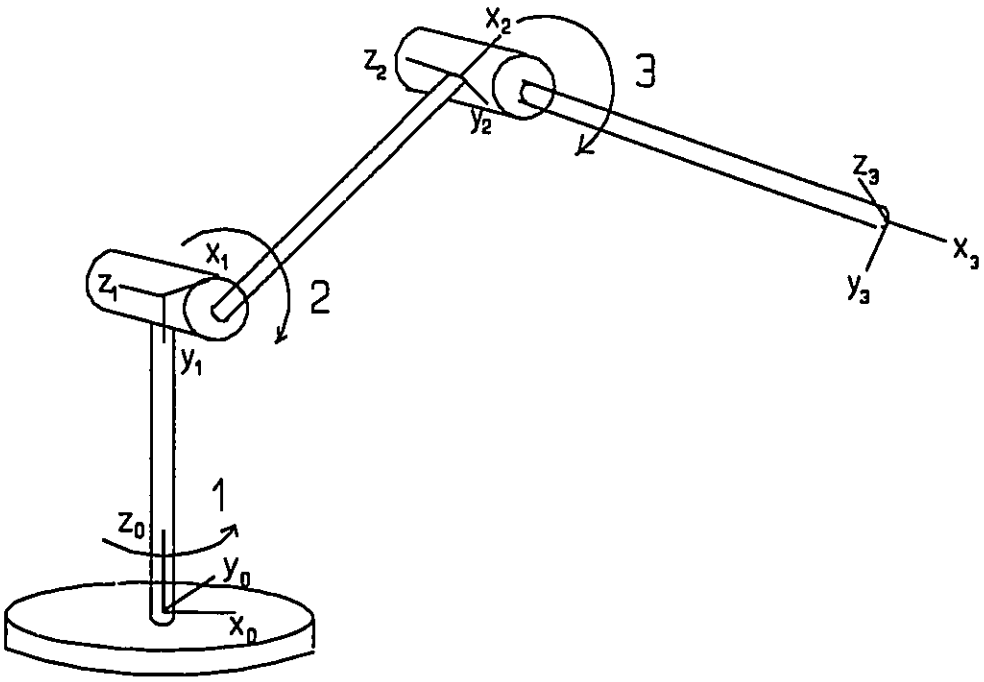


Figure A.1 Three DOF robot with coordinate systems

The three links are modelled by long slender cylindrical sections. For the mass moments of inertia calculations, the motors are also modelled by cylinders, but of different dimensions and mass.

The Denavit-Hartenberg notation is used in this work to develop the kinematics of the manipulator. [31] The D-H matrix for the three link revolute robot simulator which was constructed for use in this study is shown in Table A-1.

Link	θ_i	α_i	a_i	d_i
1	θ_1	-90	0	d_1
2	θ_2	0	a_2	d_2
3	θ_3	0	a_3	d_3

Table A.1 Denavit Hartenberg matrix for the revolute robot used in this work

The manipulator position matrix can be calculated using the D-H convention method where the final position matrix is determined by the matrix equation $A_0^3 = A_1A_2A_3$ with A_0^3 representing the final position matrix of the manipulator end effector, and A_n representing the transformation matrix for link n which is shown below in a general form.

$$A_n = \begin{bmatrix} Cos\theta_n & -Sin\theta_n Cos\alpha_n & Sin\theta_n Sin\alpha_n & a_n Cos\theta_n \\ Sin\theta_n & Co\,\theta_n Sin\alpha_n & -Cos\theta_n Sin\alpha_n & a_n Sin\theta_n \\ 0 & Sin\alpha_n & Cos\alpha_n & d_n \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

From A_n, the following three transformation matrices are derived.

$$A_1 = \begin{bmatrix} C_1 & 0 & -S_1 & 0 \\ S_1 & 0 & C_1 & 0 \\ 0 & -1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_2 = \begin{bmatrix} C_2 & -S_2 & 0 & a_2 C_2 \\ S_2 & C_2 & 0 & a_2 S_2 \\ 0 & 0 & 1 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_3 = \begin{bmatrix} C_3 & -S_3 & 0 & a_3 C_3 \\ S_3 & C_3 & 0 & a_3 S_3 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

with $C_1=Cos\theta_1$ $C_2=Cos\theta_2$ $C_3=Cos\theta_3$ $S_1=Sin\theta_1$ $S_2=Sin\theta_2$ $S_3=Sin\theta_3$

The position and orientation of the local coordinate system $[x_1\ y_1\ z_1]$ located at the end of link 1, is given by matrix A_1. The position and orientation of the local coordinate system $[x_2\ y_2\ z_2]$, located at the end of link 2, is given by the transformation matrix $A_0^2 = A_1\ A_2$.

$$A_0^2 = A_1\ A_2 = \begin{bmatrix} C_1C_1 & -C_1S-2 & -S_1 & a_2C_1C_2-S_1d_2 \\ S_1C_2 & -S_1S_2 & C_1 & a_2S_1C_2+C_1d_2 \\ -S_2 & -C_2 & 0 & -a_2S_2+d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Similarly, the position and orientation of local coordinate system $[x_3\ y_3\ z_3]$, located at the end of link 3, is given by the transformation matrix $A_0^3 = A_1\ A_2\ A_3$.

$$A_0^3 = \begin{bmatrix} -- & -- & -- & a_3C_1C_{23} - d_3S_1 + a_2C_1C_2 - d_2S_1 \\ -- & -- & -- & a_3S_1C_{23} + C_1(d_2 + d_3) + a_2S_1C_2 \\ -- & -- & -- & -a_3S_{23} - a_2S_2 + d_1 \\ -- & -- & -- & 1 \end{bmatrix}$$

For the manipulator model used in this work the values of a_i and d_i are as follows:

$$d_1 = l_{12}, \quad d_2 = \frac{1}{2}l_{13}, \quad d_3 = \frac{1}{2}l_{22}, \quad a_2 = l_{21}, \quad a_3 = l_3$$

The first three rows of column four define the location, in the base coordinate system $[x_0\ y_0\ z_0]$, of the local coordinate system $[x_3\ y_3\ z_3]$ which represents the end effector position. In this simulation, only the final column of the transformation matrix is of interest since only the positions of the joints and end effector are being studied. The first three columns of A_0^3 define the orientation of the end effector.

Dynamics of Robot Simulator Using Lagrangian Formulation

The energy-based Lagrangian method was used to determine the dynamic equations of the robot manipulator. The Lagrangian L is defined as the total kinetic energy of a dynamic system minus the total potential energy.

$$L = \sum_{i=1}^{n} (K_i - U_i) \tag{A.1}$$

The equations of motion of the system are derived using the following general differential equations.

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q} = Q_{q_i} \quad for \quad i=1,..n \qquad \text{(A.2)}$$

In the case of a three DOF revolute robot

$q_i = \theta_i$, and $Q_i = T_i$ (torque), for $i = 1,2,3$

The Lagrangian equation therefore becomes

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}_i} - \frac{\partial L}{\partial \theta_i} = T_i \quad i=1,2,3 \qquad \text{(A.8)}$$

Determination of the Lagrangian of the Robotic Manipulator

Link One:

The first link, shown in Figure A.2, possesses only rotational kinetic energy and no potential energy.

$$K_1 = \frac{1}{2} \bullet I_{1zz} \bullet \dot{\theta}_1^2 \qquad \text{(A.9)}$$

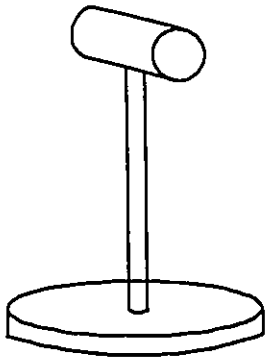


Figure A.2 Robot Link 1

Link Two:

The second link (Figure A.3) possesses both kinetic and potential energy. In addition, since the link possesses both translational and rotational velocity, the kinetic energy is calculated in two parts: (1) the energy due to the translational velocity of the centres of mass or the link components, and (2) the energy due to the rotation of the link components about their centres of

mass. The angular velocity is the same for all the components since a rigid body is assumed. The link kinetic energy is calculated as follows:

$$K_2 = \frac{1}{2}\sum_i m_{2_i} v_{2_i}^2 + \frac{1}{2}\sum_i I_{2_{jj}} \omega_{2_{jj}}^2 \qquad \text{(A.5)}$$

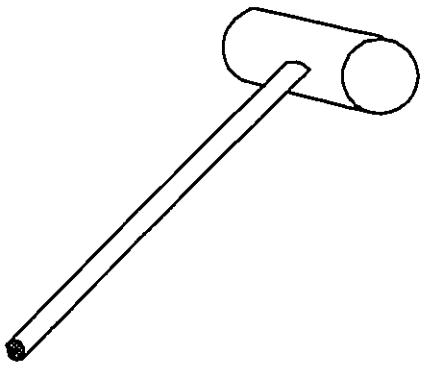


Figure A.3 Robot Link 2

where m_{2i} and v_{2i} = the mass and velocity of link 2 component i.

I_{2jj} = total moment of inertia of link 2 components, measured at the component centres of mass, about axis of rotation jj

ω_j = angular velocity of link 2 about axis jj

The position of the centre of mass of component 2 (the motor) can be extracted from the last column of the D-H transformation matrix A_0^2. The position of component 1 (slender rod) if determined by replacing a_2 with $\frac{1}{2}a_2$ in the expression for component 2. Replacing d_1, d_2, and a_2 with the proper values for each of the components of link 2, the following Cartesian position equations are derived, for the centre of mass of each component.

$$x_{21} = \frac{1}{2}l_{21}C_1C_2 - \frac{1}{2}l_{13}S_1$$

$$z_{22} = l_{21}S_2 + l_{12}$$

$$
\begin{aligned}
y_{21} &= \frac{1}{2}l_{21}S_1C_2 + \frac{1}{2}l_{13}C_1 \\
z_{21} &= -\frac{1}{2}l_{21}S_2 + l_{12} \\
x_{22} &= l_{21}C_1C_2 - \frac{1}{2}l_{13}S_1 \\
y_{22} &= l_{21}S_1C_2 + \frac{1}{2}l_{13}C_1
\end{aligned}
\tag{A.6}
$$

The time derivative of the above position equations will yield the velocities of the centres of mass of the components of link 2.

$$
\begin{aligned}
\dot{x}_{21} &= -\dot{\theta}_1(\frac{1}{2}l_{21}S_1C_2 + \frac{1}{2}l_{13}C_1) - \dot{\theta}_2(\frac{1}{2}l_{21}C_1S_2) \\
\dot{y}_{21} &= \dot{\theta}_1(\frac{1}{2}l_{21}C_1C_2 - \frac{1}{2}l_{13}S_1) - \dot{\theta}_2(\frac{1}{2}l_{21}S_1S_2) \\
\dot{z}_{21} &= -\frac{1}{2}l_{21}C_2\dot{\theta}_2 \\
\dot{x}_{22} &= \dot{\theta}_1(l_{21}S_1C_2 + \frac{1}{2}l_{13}C_1) - \dot{\theta}_2(l_{21}C_1S_2) \\
\dot{y}_{22} &= \dot{\theta}_1(l_{21}C_1C_2 - \frac{1}{2}l_{13}S_1) - \dot{\theta}_2(l_{21}S_1S_2) \\
\dot{z}_{22} &= -l_{21}C_2\dot{\theta}_2
\end{aligned}
\tag{A.7}
$$

The angular velocity about each component's centre of mass is the same and is given by the vector ω_2 in local coordinates $[x_1\ y_1\ z_1]$.

$$
\begin{aligned}
\omega_{2x} &= 0 \\
\omega_{2y} &= \dot{\theta}_1 \\
\omega_{2z} &= \dot{\theta}_2
\end{aligned}
\tag{A.8}
$$

The mass moments of inertia about the centre of mass of each of the components (masses m_{21} and m_{22}) in the local coordinate system, are expressed as a diagonal matrix.

$$I_{21} = \begin{bmatrix} I_{21xx} & 0 & 0 \\ 0 & I_{21yy} & 0 \\ 0 & 0 & I_{21zz} \end{bmatrix} \qquad I_{22} = \begin{bmatrix} I_{22xx} & 0 & 0 \\ 0 & I_{22yy} & 0 \\ 0 & 0 & I_{22zz} \end{bmatrix}$$

with

$$I_{21xx}=m_{21}r_{21}^2 \qquad I_{21yy}=\tfrac{1}{2}m_{21}r_{21}^2+\tfrac{1}{2}m_{21}l_{21}^2 \qquad I_{21zz}=\tfrac{1}{2}m_{21}r_{21}^2+\tfrac{1}{2}m_{21}l_{21}^2$$

$$I_{22xx}=m_{22}r_{22}^2 \qquad I_{22yy}=\tfrac{1}{4}m_{22}r_{22}^2+\tfrac{1}{12}m_{22}l_{22}^2 \qquad I_{22zz}=\tfrac{1}{2}m_{22}r_{22}^2+\tfrac{1}{12}m_{22}l_{22}^2$$

The kinetic energy of link 2 can now be expressed

$$\begin{aligned} K_2 = {} & \tfrac{1}{2}\begin{bmatrix} x_{21} & y_{21} & z_{21} \end{bmatrix} \begin{bmatrix} m_{21} & 0 & 0 \\ 0 & m_{21} & 0 \\ 0 & 0 & m_{21} \end{bmatrix} \begin{bmatrix} x_{21} \\ y_{21} \\ z_{21} \end{bmatrix} \\ & + \tfrac{1}{2}\begin{bmatrix} x_{22} & y_{22} & z_{22} \end{bmatrix} \begin{bmatrix} m_{22} & 0 & 0 \\ 0 & m_{22} & 0 \\ 0 & 0 & m_{22} \end{bmatrix} \begin{bmatrix} x_{22} \\ y_{22} \\ z_{22} \end{bmatrix} \\ & + \tfrac{1}{2}\begin{bmatrix} \omega_{2x} & \omega_{2y} & \omega_{2z} \end{bmatrix} \begin{bmatrix} I_{21xx}+I_{22xx} & 0 & 0 \\ 0 & I_{21yy}+I_{22yy} & 0 \\ 0 & 0 & I_{21zz}+I_{22zz} \end{bmatrix} \begin{bmatrix} \omega_{2x} \\ \omega_{2y} \\ \omega_{2z} \end{bmatrix} \end{aligned} \qquad \textbf{(A.9)}$$

The potential energy of link 2 (U_2) is

$$U_2 = -(\frac{m_{21}}{2} + m_{22})\, g\, S_2 l_{21} \qquad \text{(A.10)}$$

Link Three:

Link 3 (Figure A.4) possesses potential energy along with both translational and rotational kinetic energy. The calculations of energy are therefore similar to those of link 2.

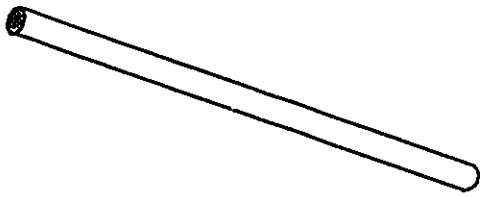


Figure A.4 Link 3

The position of the centre of mass of link 3 with respect to the base coordinate system is determined from the last column of transformation matrix A_0^3 by replacing a_3 by $\frac{1}{2}a_3$.

$$x_3 = \frac{1}{2}l_3 C_1 C_{23} - S_1(\frac{1}{2}l_{13}+\frac{1}{2}l_{22}) + l_{21} C_1 C_2$$

$$y_3 = \frac{1}{2}l_3 S_1 C_{23} + C_1(\frac{1}{2}l_{13} + \frac{1}{2}l_{22}) + l_{21} S_1 C_2 \qquad \text{(A.11)}$$

$$z_3 = l_{12} - l_{21} S_2 - \frac{1}{2}l_3 S_{23}$$

The translational velocity of the centre of mass of link 3 is determined by differentiating the expressions for the position of the centre of mass of the link. This operation results in

$$\dot{x}_3 = -\dot{\theta}_1(\frac{1}{2}l_3 S_1 C_{23} + C_1(\frac{1}{2}l_{13}+\frac{1}{2}l_{22}) + l_{21} S_1 C_2) - \dot{\theta}_2(\frac{1}{2}l_3 C_1 S_{23} + l_{21} C_1 S_2) - \dot{\theta}_3 \frac{1}{2}l_3 C_1 S_{23}$$

$$\dot{y}_3=\dot{\theta}_1(\frac{1}{2}l_3C_1C_{23}-S_1(\frac{1}{2}l_{13}+\frac{1}{2}l_{22})+l_{21}C_1C_2)-\dot{\theta}_2(\frac{1}{2}l_3S_1S_{23}+S_1S_2)-\dot{\theta}_3(\frac{1}{2}l_3S_1S_{23}) \qquad \text{(A.12)}$$

$$\dot{z}_3=-\dot{\theta}_2(l_{21}C_2+\frac{1}{2}l_3C_{23})-\dot{\theta}_3(\frac{1}{2}l_3C_{23})$$

The angular velocity of link three about its centre of mass is defined by the vector ω_3 in local coordinates $[x_2\ y_2\ z_2]$.

$$\begin{aligned} \dot{\omega}_{3x} &= 0 \\ \dot{\omega}_{3y} &= \dot{\theta}_1 \\ \dot{\omega}_{3z} &= \dot{\theta}_2 + \dot{\theta}_3 \end{aligned} \qquad \text{(A.13)}$$

The mass moment of inertia of link three is defined by the diagonal inertia matrix below.

$$I_3 = \begin{bmatrix} I_{3xx} & 0 & 0 \\ 0 & I_{3yy} & 0 \\ 0 & 0 & I_{3zz} \end{bmatrix} \qquad \begin{aligned} I_{3xx} &= m_3r_3^2 \\ I_{3yy} &= \tfrac{1}{2}m_3r_3^2+\tfrac{1}{12}m_3l_3^2 \\ I_{3zz} &= \tfrac{1}{2}m_3r_3^2+\tfrac{1}{12}m_3l_3^2 \end{aligned}$$

The matrix is diagonal since the angular velocities are defined in local coordinates.

The total kinetic energy of link three can thus be expressed by the matrix equation

$$K_3 = \tfrac{1}{2}\begin{bmatrix} x_3 & y_3 & z_3 \end{bmatrix}\begin{bmatrix} m_3 & 0 & 0 \\ 0 & m_3 & 0 \\ 0 & 0 & m_3 \end{bmatrix}\begin{bmatrix} x_3 \\ y_3 \\ z_3 \end{bmatrix}$$

$$+ \frac{1}{2}\begin{bmatrix} \omega_{3x} & \omega_{3y} & \omega_{3z} \end{bmatrix} \begin{bmatrix} I_{3xx} & 0 & 0 \\ 0 & I_{3yy} & 0 \\ 0 & 0 & I_{3zz} \end{bmatrix} \begin{bmatrix} \omega_{3x} \\ \omega_{3y} \\ \omega_{3z} \end{bmatrix} \tag{A.14}$$

The potential energy of link three (U_3) is

$$U_3 = -m_3 g(S_2 l_{21} + S_{23}\frac{l_3}{2}) \tag{A.15}$$

The Lagrangian of the robotic manipulator can thus be expressed as

$$L = K_1 + K_2 + K_3 - U_2 - U_3 \tag{A.16}$$

Determination of the Equations of Motion by the Lagrangian Method

The equations of motion of the robot manipulator are derived by the direct application of the Lagrange equation.

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_i}\right) - \frac{\partial L}{\partial \theta_i} = T_i \qquad \text{(A.17)}$$

The expansion of this equation, for the 3 DOF revolute robot, yields the following terms.

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_i}\right) = I_{1zz}\left[(d\omega_{1z})(\dot{\theta}_i\omega_{1z}) + \omega_{1z}(d\dot{\partial}_i\omega_{1z}) \right]$$

$$+ m_{21}[(d\dot{x}_{21})(\dot{\partial}_i\dot{x}_{21} + (\dot{x}_{21})(d\dot{\partial}_i x_{21}) + (d\dot{y}_{21})(\dot{\partial}_i\dot{y}_{21})$$

$$+ (\dot{y}_{21})(d\dot{\partial}_i\dot{y}_{21}) + (d\dot{z}_{21})(\dot{\partial}_i\dot{z}_{21}) + (\dot{z}_{21})(d\dot{\partial}_i\dot{z}_{21})]$$

$$+ m_{22}[(d\dot{x}_{22})(\dot{\partial}_i\dot{x}_{22}) + (\dot{x}_{22})(d\dot{\partial}_1\dot{x}_{22}) + (d\dot{y}_{22})(\dot{\partial}_i\dot{y}_{22})$$

$$+ (\dot{y}_{22})(d\dot{\partial}_i\dot{y}_{22}) + (d\dot{z}_{22})(\dot{\partial}_i\dot{z}_{22}) + (\dot{z}_{22})(d[\dot{\partial}_1\dot{z}_{22})]$$

$$+ I_{2xx}[(d\omega_{2x})(\dot{\partial}_i\omega_{2x}) + (\omega_{2x})(d\dot{\partial}_i\omega_{2x})] + I_{2yy}[(d\omega_{2y})(\dot{\partial}_i\omega_{2y})$$

$$+ (\omega_{2y})(d\dot{\partial}_i\omega_{2y})] + I_2zz[(d\omega_{2z})(\dot{\partial}_i\omega_{2z}) + (\omega_{2z})(d\dot{\partial}_i\omega_{2z})]$$

$$+ m_3[(d\dot{x}_3)(\dot{\partial}_i\dot{x}_3) + (\dot{x}_3)(d\dot{\partial}_i\dot{x}_3) + (d\dot{y}_3)(\dot{\partial}_i\dot{y}_3)$$

$$+ (\dot{y}_3)(d\sigma_i\dot{y}_3) + (d\dot{z}_3)(\dot{\partial}_i\dot{z}_3) + (\dot{z}_3)(d\dot{\partial}_i\dot{z}_3)]$$

$$+ I_{3xx}[(d\omega_{3x})(\dot{\partial}_i\omega_{3x}) + (\omega_{3x})(d\dot{\partial}_i\omega_{3x})] + I_{3yy}[(d\omega_{3y})(\dot{\partial}_i\omega_{3y})$$

$$+ (\omega_{3y})(d\dot{\partial}_i\omega_{3y})] + I_{3zz}[(d\omega_{3z})(\dot{\partial}_i\omega_{3z}) + (\omega_{3z})(d\dot{\partial}_i\omega_{3z})]$$

$$\frac{\partial L}{\partial \theta_i} = \omega_{1z} I_{1zz} \partial_i \omega_{1z} + m_{21}[\dot{x}_{21}(\partial_i \dot{x}_{21}) + \dot{y}_{21}(\partial_i \dot{y}_{21}) + \dot{z}_{21}(\partial_i \dot{z}_{21})]$$

$$+ m_{22}[\dot{x}_{22}(\partial_i \dot{x}_{22}) + \dot{y}_{22}(\partial_i \dot{y}_{22} + \dot{z}_{22}(\partial_i \dot{z}_{22})]$$

$$+ m_3[\dot{x}_3(\partial_i \dot{x}_3) + \dot{y}_3(\partial_i \dot{y}_3) + \dot{z}_3(\partial_i \dot{z}_3)] + \omega_{3x}(\partial_i \omega_{3x})(I_{3xx})$$

$$+ \omega_{3y}(\partial_i \omega_{3y})(I_{3yy}) + \omega_{3z}(\partial_i \omega_{3z})(I_{3zz})$$

$$+ \frac{\partial U}{\partial \theta_i}$$

where ∂_i indicates the partial derivative with respect to θ_i

$\dot{\partial}_i$ indicates the partial derivative with respect to $\dot{\theta}_i$

$d\dot{\partial}_i$ indicates time derivative of the partial with respect to $\dot{\theta}_i$

The translational and angular acceleration terms which are present in all three equations of motion are listed below. Note that the angular accelerations are defined in coordinates systems local to each specific joint or degree of freedom. (i.e. ω_{1z} is defined in $[x_0\ y_0\ z_0]$ coordinates; ω_{2x}, ω_{2y} and ω_{2z} are defined in $[x_1\ y_1\ z_1]$ coordinates; ω_{3x}, ω_{3y} and ω_{3z} are defined in $[x_2\ y_2\ z_2]$ coordinates. The translational velocities, however, are all expressed in terms of the base coordinate system $[x_0\ y_0\ z_0]$. This is due to the method which was used to derive the components of the link kinetic energy: translational kinetic energy of the masses with respect to the base coordinates plus rotational kinetic energy with respect to the component centres of mass.

Translational Accelerations:

$$\ddot{x}_{21} = -\ddot{\theta}_1(\frac{1}{2}l_{21}S_1C_2 + \frac{1}{2}l_{13}C_1) - \dot{\theta}_1[\frac{1}{2}l_{21}(C_1C_2\dot{\theta}_1 - S_1S_2\dot{\theta}_2) - \frac{1}{2}l_{13}S_1\dot{\theta}_1]$$

$$-\ddot{\theta}_2(\frac{1}{2}l_{21}C_1S_2) + \dot{\theta}_2\frac{1}{2}l_{21}(S_1S_2\dot{\theta}_1 - C_1C_2\dot{\theta}_2)$$

$$\ddot{y}_{21} = \ddot{\theta}_1(\frac{1}{2}l_{21}C_1C_2 - \frac{1}{2}l_{13}S_1) - \dot{\theta}_1[\frac{1}{2}l_{21}(S_1C_2\dot{\theta}_1 + C_1S_2\dot{\theta}_2) + \frac{1}{2}l_{13}C_1\dot{\theta}_1]$$

$$-\ddot{\theta}_2(\frac{1}{2}l_{21}S_1S_2) - \dot{\theta}_2\frac{1}{2}l_{21}(C_1S_2\dot{\theta}_1 + S_1C_2\dot{\theta}_2)$$

$$\ddot{z}_{21} = -\frac{1}{2}l_{21}(\ddot{\theta}_2C_2 - S_2\dot{\theta}_2^2)$$

The expressions for the accelerations of x_{22}, y_{22} and z_{22} are the same as those of x_{21}, y_{22} and z_{22} except that the term $\frac{1}{2}l_{21}$ is replaced by l_{21}.

$$\ddot{x}_3 = -\ddot{\theta}_1[\frac{1}{2}l_3S_1C_{23} + C_1(\frac{1}{2}l_{13} + \frac{1}{2}l_{22}) + l_{21}S_1C_2] - \ddot{\theta}_2(\frac{1}{2}l_3C_1S_{23} + l_{21}C_1S_2) - \ddot{\theta}_3(\frac{1}{2}l_3C_1S_{23})$$

$$-\dot{\theta}_1[\frac{1}{2}l_3(C_1C_{23}\dot{\theta}_1 - S_1S_{23}(\dot{\theta}_2 + \dot{\theta}_3)) - s_1\dot{\theta}_1(\frac{1}{2}l_{13} + \frac{1}{2}l22) + l_{21}(C_1C_2\dot{\theta}_1 - S_1S_2\dot{\theta}_2)]$$

$$-\dot{\theta}_2[\frac{1}{2}l_3(-S_1S_{23}\dot{\theta}_1 + C_1C_{23}(\dot{\theta}_2 + \dot{\theta}_3)) + l_{21}(-S_1S_2\dot{\theta}_1 + C_1C_2\dot{\theta}_2)]$$

$$-\dot{\theta}_3[\frac{1}{2}l_3(-S_1S_{23}\dot{\theta}_1 + C_1C_{23}(\dot{\theta}_2 + \dot{\theta}_3))]$$

$$\ddot{y}_3 = \ddot{\theta}_1[\frac{1}{2}l_3C_1C_{23}-S_1(\frac{1}{2}l_{13}+\frac{1}{2}l_{22})+l_{21}C_1C_2]-\ddot{\theta}_2[\frac{1}{2}l_3S_1S_{23}+l_{21}S_1S_2]-\ddot{\theta}_3[\frac{1}{2}l_3S_1S_{23}]$$
$$+\dot{\theta}_1[\frac{1}{2}l_3(-S_1C_{23}\dot{\theta}_1-C_1S_{23}(\dot{\theta}_2+\dot{\theta}_3))-C_1\dot{\theta}_1(\frac{1}{2}l_{13}+\frac{1}{2}l_{22})-l_{21}(S_1C_2\dot{\theta}_1+C_1S_2\dot{\theta}_2)]$$
$$-\dot{\theta}_2[\frac{1}{2}l_3(C_1S_{23}\dot{\theta}_1+S_1C_{23}(\dot{\theta}_2+\dot{\theta}_3))+l_{21}(C_1S_2\dot{\theta}_1+S_1C_2\dot{\theta}_2)]$$
$$-\dot{\theta}_3[\frac{1}{2}l_3(C_1S_{23}\dot{\theta}_1+S_1C_{23}(\dot{\theta}_2+\dot{\theta}_3))]$$

$$\ddot{z}_3 = -\ddot{\theta}_2(l_{21}C_2+\frac{1}{2}l_3C_{23})+\dot{\theta}_2[l_{21}S_2\dot{\theta}_2+\frac{1}{2}l_3S_{23}(\dot{\theta}_2+\dot{\theta}_3)]$$
$$-\ddot{\theta}_3(\frac{1}{2}l_3C_{23}) + \dot{\theta}_3\frac{1}{2}l_3S_{23}(\dot{\theta}_2+\dot{\theta}_3)$$

Angular Accelerations:

$$d\omega_{1x}=\ddot{\theta}_1,\ d\omega_{2x}=0,\ d\omega_{2y}=\ddot{\theta}_1,\ d\omega_{2z}=\ddot{\theta}_2$$
$$d\omega_{3x}=0,\ d\omega_{3y}=\ddot{\theta}_1,\ d\omega_{3z}=\ddot{\theta}_2+\ddot{\theta}_3$$

Terms for the First Lagrange Equation:

$$\partial_1\omega_{1z}=\partial_1\omega_{21x}=\partial_1\omega_{21y}=\partial_1\omega_{21z}=\partial_1\omega_{22x}=\partial_1\omega_{22y}=\partial_1\omega_{22z}=\partial_1\omega_{3x}=\partial_1\omega_{3y}=\partial_1\omega_{3z}=0$$

$$\partial_1\dot{x}_{21}=-\dot{\theta}_1(\frac{1}{2}l_{21}C_1C_2-\frac{1}{2}l_{13}S_1)+\dot{\theta}_2(\frac{1}{2}l_{21}S_1S_2)$$

$$\partial_1\dot{y}_{22}=-\dot{\theta}_1(\frac{1}{2}l_{21}S_1C_2+\frac{1}{2}l_{13}C_1)-\dot{\theta}_2(\frac{1}{2}l_{21}C_1S_2),\ \partial_1\dot{z}_{21}=0$$

$$\partial_1\dot{x}_{22}=-\dot{\theta}_1(l_{21}C_1C_2-\frac{1}{2}l_{13}S_1)+\dot{\theta}_2(\frac{1}{2}l_{21}S_1S_2),\ \partial_1\dot{z}_{22}=0$$

$$\dot{\partial}_1\omega_{1z}=1,\ \dot{\partial}_1\omega_{2x}=0,\ \dot{\partial}_1\omega_{2y}=1,\ \dot{\partial}_1\omega_{2z}=0,\ \dot{\partial}_1\omega_{3x}=0,\ \dot{\partial}_1\omega_{3y}=1,\ \dot{\partial}_1\omega_{3z}=0$$

$$\partial_1\dot{y}_{22}=-\dot{\theta}_1(l_{21}S_1C_2+\frac{1}{2}l_{13}C_1)-\dot{\theta}_2(\frac{1}{2}l_{21}C_1S_2),\quad \partial_1\dot{z}_3=0$$

$$\partial_1\dot{x}_3=-\dot{\theta}_1(\frac{1}{2}l_3C_1C_{23}-s_1(\frac{1}{2}l_{13}+\frac{1}{2}l_{22})+l_{21}C_1C_2)+\dot{\theta}_2(\frac{1}{2}S_1S_{23}+l_{21}S_1S_2)+\dot{\theta}_3(\frac{1}{2}l_3S_1S_{23})$$

$$\partial_1\dot{y}_3=-\dot{\theta}_1(\frac{1}{2}l_3S_1C_{23}+C_1(\frac{1}{2}l_{13}+\frac{1}{2}l_{22})+l_{21}S_1C_2)-\dot{\theta}_2(\frac{1}{2}l_3C_1S_{23}+l_{21}C_1S_2)-\dot{\theta}_3(\frac{1}{2}l_3C_1S_{23})$$

$$\dot{\partial}_1\dot{x}_{21}=-\frac{1}{2}l_{21}S_1C_2-\frac{1}{2}l_{13}C_1,\quad \dot{\partial}_1\dot{y}_{21}=\frac{1}{2}l_{21}C_1C_2-\frac{1}{2}l_{13}S_1,\quad \dot{\partial}_1\dot{z}_{21}=0$$

$$\dot{\partial}_1\dot{x}_{22}=-l_{21}S_1C_2-\frac{1}{2}l_{13}C_1,\quad \dot{\partial}_1\dot{y}_{22}=l_{21}C_1C_2-\frac{1}{2}l_{13}S_1,\quad \dot{\partial}_1\dot{z}_{22}=0$$

$$\dot{\partial}_1\dot{x}_3=-\frac{1}{2}L_3S_1C_{23}-C_1(\frac{1}{2}l_{13}+\frac{1}{l_{22}})-l_{21}S_1C_2$$

$$\dot{\partial}_1\dot{Y}_3=\frac{1}{2}l_3C_1C_{23}-S_1(\frac{1}{2}l_{13}+\frac{1}{2}l_{22})+l_{21}C_1C_2,\quad \dot{\partial}_1\dot{z}_3=0$$

$$d\dot{\partial}_1w_{ij}=0,\quad for\ i=1,2,3;\quad j=x,y,z;$$

$$d\dot{\partial}_1\dot{x}_{21}=-\frac{1}{2}l_{21}(C_1C_2\dot{\theta}_1-S_1S_2\dot{\theta}_2)+\frac{1}{2}l_{13}S_1\dot{\theta}_1$$

$$d\dot{\partial}_1\dot{y}_{21}=-\frac{1}{2}l_{21}(S_1C_2\dot{\theta}_1+C_1S_2\dot{\theta}_2)-\frac{1}{2}l_{13}c_1\dot{\theta}_1,\quad d\dot{\partial}_1\dot{z}_{21}=0$$

$$d\dot{\partial}_1\dot{x}_{22}=-l_{21}(C_1C_2\dot{\theta}_1-S_1S_2\dot{\theta}_2)+\frac{1}{2}l_{13}S_1\dot{\theta}_1$$

$$d\dot{\partial}_1\dot{Y}_{22}=-l_{21}(S_1C_2\dot{\theta}_1+C_1S_2\dot{\theta}_2)-\frac{1}{2}l_{13}C_1\dot{\theta}_1,\quad d\dot{\partial}_1\dot{z}_{22}=0$$

$$d\dot{\partial}_1\dot{x}_3=-\frac{1}{2}l_3(C_1C_{23}\dot{\theta}_1-S_1S_{23}(\dot{\theta}_2+\dot{\theta}_3))+S_1\dot{\theta}_1(\frac{1}{2}l_{13}+\frac{1}{2}l_{22})-l_{21}(C_1C_2\dot{\theta}_1-S_1S_2\dot{\theta}_2)$$

$$d\dot{\partial}_1\dot{y}_3=-\frac{1}{2}l_3(S_1C_{23}\dot{\theta}_1+C_1S_{23}(\dot{\theta}_2+\dot{\theta}_3))-C_1\dot{\theta}_1(\frac{1}{2}l_{13}+l_{22})-l_{21}(S_1C_2\dot{\theta}_1+C_1S_2\dot{\theta}_2)$$

$$d\dot{\partial}_1\dot{z}_3=0$$

Terms for the Second Lagrange Equation:

$$\partial_2\omega_{1z}=0,\ \partial_2\omega_{2x}=0,\ \partial_2\omega_{2y}=0,\ \partial_2\omega_{2z}=0,\ \partial_2\omega_{3x}=0,\ \partial_2\omega_{3y}=0,\ \partial_2\omega_{3z}=0$$

$$\partial_2\dot{x}_{21}=\frac{1}{2}l_{21}(\dot{\theta}_1S_1S_2-\dot{\theta}_2C-1C_2)$$

$$\partial_2\dot{y}_{21}=-\frac{1}{2}l_{21}(\dot{\theta}_1C_1S_2+\dot{\theta}_2S_1C_2)$$

$$\partial_2\dot{z}_{21}=\frac{1}{2}l_{21}\dot{\theta}_2S_2$$

$$\partial_2\dot{x}_{22}=l_{21}(\dot{\theta}_1S_1S_2-\dot{\theta}_2C_1C_2)$$

$$\partial_2\dot{y}_{22}=-l_{21}(\dot{\theta}_1C_1S_2+\dot{\theta}_2S_1C_2)$$

$$\partial_2\dot{z}_{22}=l_{21}\dot{\theta}_2S_2$$

$$\partial_2\dot{x}_3=\dot{\theta}_1(\frac{1}{2}l_3S_1S_{23}+l_{21}S_1S_2)-\dot{\theta}_2(\frac{1}{2}l_3C_1C_{23}+l_{21}C_1C_2)-\dot{\theta}_3(\frac{1}{2}l_3c_1C_{23})$$

$$\partial_2\dot{y}_3=-\dot{\theta}_1(\frac{1}{2}l_3C_1S_{23}+l_{21}C_1S_2)-\dot{\theta}_2(\frac{1}{2}l_3S_1C_{23}+l_{21}S_1C_2)-\dot{\theta}_3(\frac{1}{2}l_3S_1C_{23})$$

$$\partial_2\dot{z}_3=\dot{\theta}_2(l_{21}S_2+\frac{1}{2}l_3s_{23})+\dot{\theta}_3(\frac{1}{2}l_3S_{23})$$

$$\dot{\partial}_2\omega_{1z}=0,\ \dot{\partial}_2\omega_{2x}=0,\ \dot{\partial}_2\omega_{2y}=0,\ \dot{\partial}_2\omega_{2z}=1,\ \dot{\partial}_2\omega_{3x}=0,\ \dot{\partial}_2\omega_{3y}=0,\ \dot{\partial}_2\omega_{3z}=1$$

$$\dot{\partial}_2\dot{x}_{21}=-\frac{1}{2}l_{21}C_1S_2,\ \dot{\partial}_2\dot{y}_{21}=-\frac{1}{2}l_{21}S_1S_2,\ \dot{\partial}_2\dot{z}_{21}=-\frac{1}{2}l_{21}C_2$$

$$\dot{\partial}_2\dot{x}_{22}=-l_{21}C_1S_2,\ \dot{\partial}_2\dot{y}_{22}=-l_{21}S_1S_2,\ \dot{\partial}_2\dot{z}_{22}=-l_{21}C_2$$

$$\dot{\partial}_2\dot{x}_3=-C_1S_2l_{21}-\frac{1}{2}l_3C_1S_{23},\ \dot{\partial}_2\dot{y}_3=-S_1S_2L_{21}-\frac{1}{2}l_3S_1S_{23},\ \dot{\partial}_2\dot{z}_3=-\frac{1}{2}l_3C_{23}-l_{21}C_2$$

$$d\dot{\partial}_2\omega_{1z}=0,\quad d\dot{\partial}_2\omega_{2x}=0,\quad d\dot{\partial}_2\omega_{2y}=0,\quad d\dot{\partial}_2\omega_{2z}=0,\quad d\dot{\partial}_2\omega_{3x}=0,\quad d\dot{\partial}_2\omega_{3y}=0,\quad d\dot{\partial}_2\omega_{3z}=0$$

$$d\dot{\partial}_2\dot{x}_{21}=\frac{1}{2}l_{21}(\dot{\theta}_1S_1S_2-\dot{\theta}_2C_1C_2)$$

$$d\dot{\partial}_2\dot{y}_{21}=-\frac{1}{2}l_{21}(\dot{\theta}_1C_1S_2+\dot{\theta}_2S_1C_2)$$

$$d\dot{\partial}_2\dot{z}_{21}=\frac{1}{2}l_{21}\dot{\theta}_2S_2$$

$$d\dot{\partial}_2\dot{x}_{22}=l_{21}(\dot{\theta}_1S_1S_2-\dot{\theta}_2C_1C_2)$$

$$d\dot{\partial}_2\dot{y}_{22}=-l_{21}(\dot{\theta}_1C_1S_2+\dot{\theta}_2S_1C_2)$$

$$d\dot{\partial}_2\dot{z}_{22}=l_{21}\dot{\theta}_2S_2$$

$$d\dot{\partial}_2=l_{21}(\dot{\theta}_1S_1S_2-\dot{\theta}_2C_1C_2)+\frac{1}{2}l_3(\dot{\theta}_1S_1S_{23}-(\dot{\theta}_2+\dot{\theta}_3)C_1C_{23})$$

$$d\dot{\partial}_2\dot{y}_3=-l_{21}(\dot{\theta}_1C_1S_2+\dot{\theta}_2S_1C_2)-\frac{1}{2}l_3(\dot{\theta}_1C_1S_{23}+(\dot{\theta}_2+\dot{\theta}_3)C_{23}S_1)$$

$$d\dot{\partial}_2\dot{z}_3=\frac{1}{2}l_3(\dot{\theta}_2+\dot{\theta}_3)S_{23}+l_{21}\dot{\theta}_2S_2$$

Terms for the Third Lagrange Equation:

$$\partial_3\omega_{1z}=\partial_3\omega_{2x}=\partial_3\omega_{2y}=\partial_3\omega_{2z}=\partial_3\omega_{3x}=\partial_3\omega_{3y}=\partial_3\omega_{3z}=0$$

$$\partial_3\dot{x}_{21}=\partial_3\dot{x}_{22}=\partial_3\dot{y}_{21}=\partial_3\dot{y}_{22}=\partial_3\dot{z}_{21}=\partial_3\dot{z}_{22}=0$$

$$\partial_3\dot{x}_3=\frac{1}{2}l_3[\dot{\theta}_1S_1S_{23}-\dot{\theta}_2C_1C_{23}-\dot{\theta}_3C_1C_{23}]$$

$$\partial_3\dot{y}_3=-\frac{1}{2}l_3[\dot{\theta}_1C_1S_{23}+\dot{\theta}_2S_1C_{23}+\dot{\theta}_3S_1C_{23}]$$

$$\partial_3\dot{z}_3=\frac{1}{2}l_3(\dot{\theta}_2S_{23}+\dot{\theta}_3S+23)$$

$$\dot{\partial}_3\omega_{1z}=\dot{\partial}_3\omega_{2x}=\dot{\partial}_3\omega_{2y}=\dot{\partial}_3\omega_{2z}=\dot{\partial}_3\omega_{3x}=\dot{\partial}_3\omega_{3y}=0,\quad \dot{\partial}_3\omega_{3z}=1$$

$$\dot{\partial}_3\dot{x}_{21}=\dot{\partial}_3\dot{x}_{22}=\dot{\partial}_3\dot{y}_{21}=\dot{\partial}_3\dot{y}_{22}=\dot{\partial}_3\dot{z}_{21}=\dot{\partial}_3\dot{z}_{22}=0$$

$$\dot{\partial}_3\dot{y}_3=-\frac{1}{2}l_3C_1S_{23},\quad \dot{\partial}_3\dot{y}_3=-\frac{1}{2}l_3S_1S_{23},\quad \dot{\partial}_3\dot{z}_3=-\frac{1}{2}l_3C_{23}$$

$$d\dot{\partial}_3\omega_{1z}=d\dot{\partial}_3\omega_{2x}=d\dot{\partial}_3\omega_{2y}=d\dot{\partial}_3\omega_{2z}=d\dot{\partial}_3\omega_{3x}=d\dot{\upsilon}_3\omega_{3y}=d\dot{\partial}_3\omega_{3z}=0$$

$$d\dot{\partial}_3\dot{x}_{21}=d\dot{\partial}_3\dot{x}_{22}=d\dot{\partial}_3\dot{y}_{21}=d\dot{\partial}_3\dot{y}_{22}=d\dot{\partial}_3\dot{z}_{21}=d\dot{\partial}_3\dot{z}_{22}=0$$

$$d\dot{\partial}_3\dot{x}_3=\frac{1}{2}l_3[\dot{\theta}_1S_1S_{23}-(\dot{\theta}_2+\dot{\theta}_3)C_1C_{23}]$$

$$d\dot{\partial}_3\dot{y}_3=-\frac{1}{2}l_3[\dot{\theta}_1C_1S_{23}+(\dot{\theta}_2+\dot{\theta}_3)S_1C_{23}]$$

$$d\dot{\partial}_3\dot{z}_3=\frac{1}{2}l_3(\dot{\theta}_2+\dot{\theta}_3)S_{23}$$

The Lagrangian method results in a set of three coupled equations of the form

$$\sum_{i=1}^{3} \left(\sum_{j=1}^{3} A_{ij} \ddot{\theta}_j \right) = C_i$$

where A_{ij} = the sum of all the coefficients of the accelerations $\ddot{\theta}_i$ in Lagrange equation i.

C_i = all the non acceleration terms in Lagrange equation i.

The equations can be decoupled in acceleration resulting in three equations of the form

$$\ddot{\theta}_i = D_i \qquad i=1,2,3$$

From these, a set of six first order ordinary differential equations are formulated as follows

$$\frac{d\dot{\theta}_i}{dt} = D_i \qquad i=1,2,3$$

$$\frac{d\theta_i}{dt} = \dot{\theta}_i \qquad i=1,2,3$$

This set of six equations must be solved at each iterative time interval of the computer simulation. This is accomplished numerically using the Adams-Bashford predictor-modifier-corrector method.

All the software for the robot simulator is written in "C" and implemented and executed on 386 or 486 microcomputers.

APPENDIX B

STATIC BALANCING OF LINKS 2 & 3

B.1 Benefits of Static Balancing of Manipulator Links

Certain links of a manipulator are subject to gravity torques which act as disturbances to the link/controller system. The effects of gravity can be greatly reduced by the addition of a feedforward gravity compensation signal in the control loop. The magnitude of the compensation is calculated from the robot model. The link actuator must therefore provide enough force to accomplish two major tasks: (1) counteract the gravity force which is a function of the mass and configuration of the links, and (2) accelerate or decelerate the link which is a function of the mass moments of inertia of the links.

While the acceleration torque is always in the direction of link motion, the gravity compensation can be either in the direction of motion or opposite to the direction of motion. This asymmetry in the input torque requirements for link motion results in different control responses based on direction of motion. It also forces larger electric motors to be used as actuators.

Static balancing consists of adding a counterweight to each link in order to null the

resultant moment about the link's pivot point. Since the robot load is variable, and generally unknown, passive balancing cannot be exactly achieved for most conditions. Therefore, the compromise generally used is to balance for a load of one-half of the maximum load. This simple method greatly reduces the gravity disturbance torque. However the addition of the counterweight increases the link's mass moment of inertia and therefore additional torque is required for acceleration or deceleration. This, however, is generally less than that which would have been required for full gravity compensation. Static balancing results in more uniform system response for all directions of link motion and therefore reduces the maximum driving torques required.

Links two and three of the three link revolute robot simulator used for this work were statically balanced for a load of one-half the maximum load. Link one is not subject to a gravity torque about its axis of rotation therefore no balancing was required. The terms of the balancing components were added to the equations of motion after they were derived. This is a straightforward procedure which will be presented for links two and three.

Static Balancing of Link 3

The counterweight for link three was defined as a cylinder attached to a rod extending from the link's pivot point opposite to the link itself. The free body diagram for link three including the counterweight and a three-dimensional representation of the counterweight assembly is shown in Figure B.1.

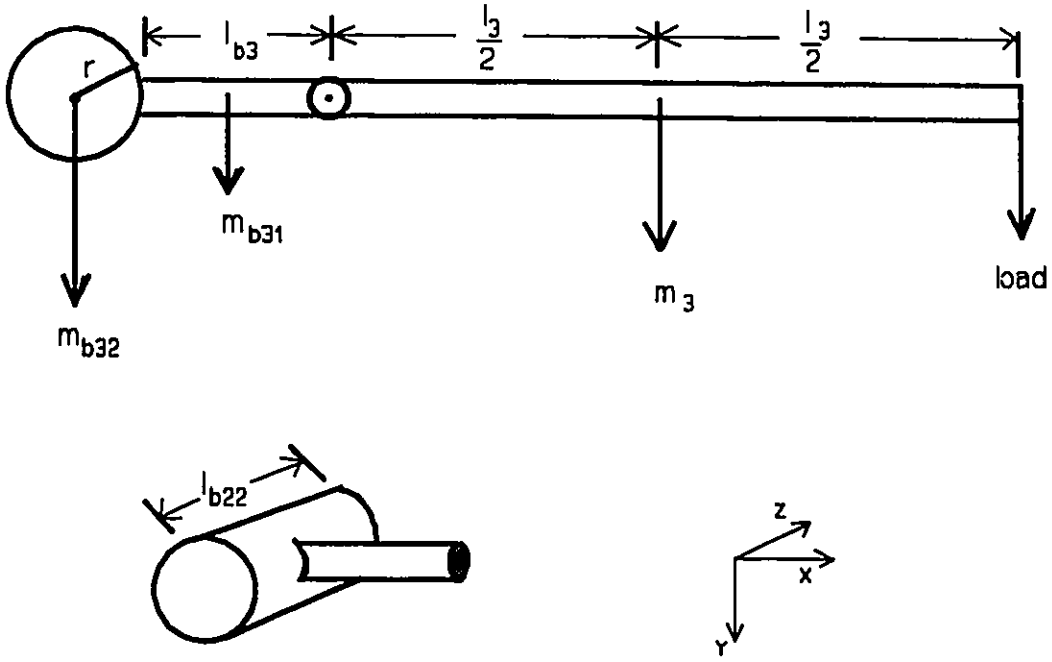


Figure B.1 Link 3 with Counterweight

The sum of moments about the link's pivot point yields

$$m_3\, g\frac{l_3}{2} + \frac{load_{max}}{2} g l_3 = m_{b31}\, g\frac{l_{b3}}{2} + m_{b32}\, g(l_{b3}+r) \qquad \text{(B.1)}$$

Choosing $l_{b3}=\frac{1}{4}l_3$, and $m_{b31}=\frac{1}{4}m_3$ the mass of the counter weight can be determined

$$m_{b32} = \frac{m_3\frac{l_3}{2} + \frac{load_{max}}{2} l_3 - m_{b31}\frac{l_{b3}}{2}}{(l_{b3} + r)} \qquad \text{(B.4)}$$

If r is chosen to be the same as motor radii r_{22}, the mass moments of inertia for the counterweight assembly are as follows:

$$I_{xx} = m_{b31}r_3^2 + \frac{1}{4}m_{b32}r_{22}^2 + \frac{1}{12}m_{b32}l_{22}^2 \quad \text{(B.3)}$$

$$I_{yy} = \frac{1}{2}m_{b31}r_3^2 + \frac{1}{2}m_{b31}l_{b3}^2 + \frac{1}{4}m_{b32}r_{22}^2 + \frac{1}{12}m_{b32}l_{22}^2 \quad \text{(B.4)}$$

$$I_{zz} = \frac{1}{2}m_{b31}r_r^2 + \frac{1}{2}m_{b31}l_{b3}^2 + \frac{1}{2}m_{b32}r_{22}^2 \quad \text{(B.5)}$$

The counterweight terms are added into the equations of motion in the following manner:

(1) The gravity torque is modified to include the counterweight components.

(2) Since the angular velocity of the counterweight is identical to that of link three, the equations are updated by adding the mass moments of inertia of the counterweight assembly to that of the link itself.

(3) The translational velocities of the centres of mass of the counterweight components will be inversely proportional to that of the centre of mass of the link itself. The proportionality factor is the ratio of the distance from the centre of mass of each counterweight component to the link pivot to the distance from the link centre of mass to the pivot point. For component m_{b31}, the proportionality factor is $-\frac{l_{b3}}{l_3}=-\frac{1}{4}$. For component m_{b32}, the factor is $-2(\frac{l_{b3}+r}{l_3})$. The square of these proportional terms, multiplied by their respective masses, are added to the link mass m_3 in all the equations of motion. This results in a new mass for link 3 (M_3).

$$M_3 = m_3 + (-\frac{1}{4})^2 m_{b31} + (-2\frac{l_{b3}+r}{l_3})^2 m_{b32} \quad \text{(B.6)}$$

Static Balancing of Link 2

The balancing procedure for link two is identical to that of link three, except that there are more terms. Again the counterweight for link two was defined as a cylinder with an attached rod extending from the link's pivot point opposite to the link itself (Figure B.2).

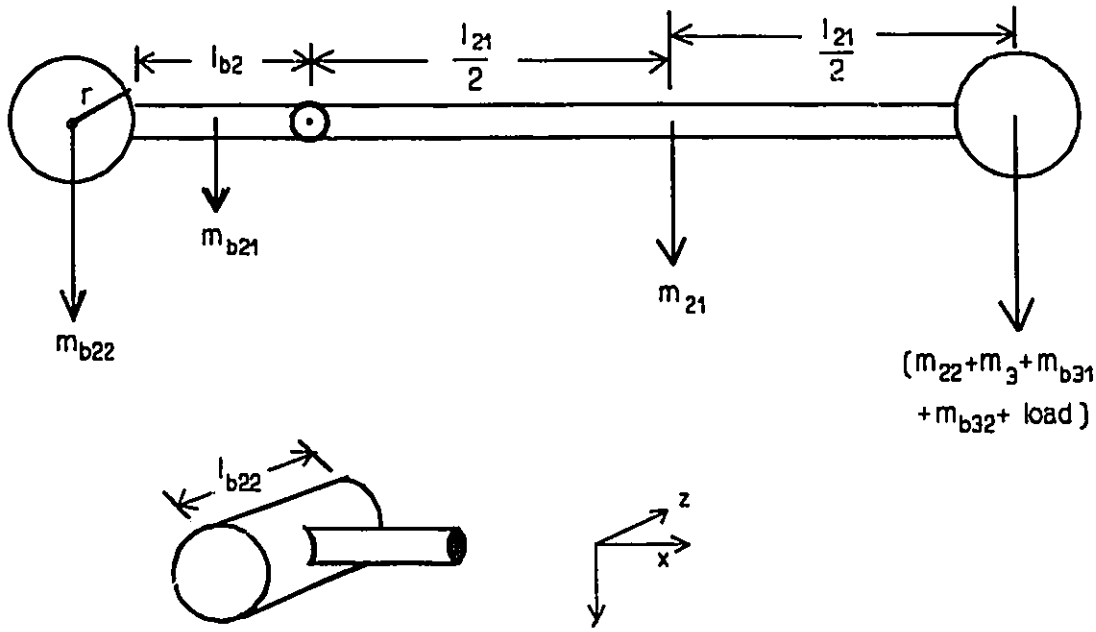


Figure B.2 Link 2 with Counterweight

The sum of moments about link two's pivot point yields

$$(\frac{m_{21}}{2}+m_{22}+m_3+\frac{load_{max}}{2}+m_{b31}+m_{b32})gl_{21} = m_{b21}g\frac{l_{b2}}{2} + m_{b22}g(l_{b2}+r) \qquad \text{(B.7)}$$

Again choosing $l_{b2}=\frac{1}{4}l_{21}$, $m_{b21}=\frac{1}{4}m_{21}$, and $r=r_{22}$ the mass and the mass moments of inertia of

the counterweight can be determined

$$m_{b22} = \frac{(\frac{m_{21}}{2}+m_{22}+m_3+\frac{load_{max}}{2}+m_{b31}+m_{b32})l_{21} - m_{b21}\frac{l_{b2}}{2}}{(l_{b2}+r)} \quad \text{(B.8)}$$

$$I_{xx} = m_{b21}r_3^2 + \frac{1}{4}m_{b22}r_{22}^2 + \frac{1}{12}m_{b22}l_{22}^2 \quad \text{(B.9)}$$

$$I_{yy} = \frac{1}{2}m_{b21}r_3^2 + \frac{1}{2}m_{b21}l_{b2}^2 + \frac{1}{4}m_{b22}r_{22}^2 + \frac{1}{12}m_{b22}l_{22}^2 \quad \text{(B.10)}$$

$$I_{zz} = \frac{1}{2}m_{b21}r_3^2 + \frac{1}{2}m_{b21}l_{b2}^2 + \frac{1}{2}m_{b22}r_{22}^2 \quad \text{(B.11)}$$

The counterweight terms are added into the equations of motion in an identical manner to those of link three. The translational velocity proportionalities are: for component m_{b21}, $-\frac{l_{b2}}{l_{21}}=-\frac{1}{4}$, and for component m_{b22}, $-2(\frac{l_{b2}+r}{l_{21}})$. Again, the square of these proportional terms, multiplied by their respective masses, are added to the link mass m_{21} in all the equations. This results in a new mass for component 21 of link 2 (M_{21}).

$$M_{21} = m_{21} + (-\frac{1}{4})^2 m_{b21} + (-2\frac{l_{b2}+r}{l_{21}})^2 m_{b22} \quad \text{(B.12)}$$

This new value for M_{21} is used in the equations of motion to incorporate the effects of the balancing components.

APPENDIX C

AUTOMATIC GRAVITY COMPENSATION FOR PROPORTIONAL CONTROL UNDER UNCERTAINTY

C.1 Gravity Disturbance Torques and Compensation

The gravity torque T_g is considered to be a disturbance which is always present for certain links of a robotic manipulator. For a step input, a proportional controlled link with velocity feedback subject to a gravity torque will always have a steady state error. The output of the system $\theta(s)$ when it is subjected to a step input $\theta_d(s)$ and in the presence of the gravity disturbance T_g, also considered as a step input, is given in the Laplace domain as

$$\theta(s) = \frac{\theta_d(s)K_pK_mN - T_g(s)N}{Js^2 + s(K_vK_mN+F_m+B) + NK_pK_m} \qquad \text{(C.1)}$$

Applying of the final value theorem yields the following steady state response.

$$\theta_{steadystate} = \theta_d - \frac{T_g}{K_pK_m} \qquad \text{(C.2)}$$

Explicit gravity compensation can be used to theoretically eliminate this error. The common method consists of generating a feedforward compensating torque signal which will

exactly counteract the gravity torque [31] [32]. However, this compensation signal is usually calculated from a theoretical robot model and therefore may not compensate exactly the actual gravity torque. Also, if the load is not known, the calculated compensation is very inaccurate.

System identification techniques can be used for on-line estimation of the gravity torques [33] These techniques can be accurate but may require significant computation and add complexity to the control program.

An integral term may be added to the controller to eliminate the steady state error and provide automatic gravity compensation even for unknown loads.

$$u(t) = K_p[\theta_d(t)-\theta(t)] + K_I\int_0^t[\theta_d(t)-\theta(t)]dt - K_v\dot{\theta}(t) \qquad \text{(C.3)}$$

When the link is moving in a direction opposed to the gravity torque the integration of the position error will eventually accumulate a correction signal which will compensate for gravity. For a properly tuned manipulator the gravity torque will be exactly compensated for by the integral component of the control torque when the position error and velocity are zero. This will occur without over-shoot.

However, the PI controlled link with velocity feedback is inherently flawed when faced with a gravity disturbance force T_g. The link will overshoot the target point in cases where the

link is moving in the direction of the gravity torque. The over-shoot occurs because the integral component, which will eventually compensate for gravity, integrates directly the position error. In the case where the position error results in a torque of the same sign as the gravity torque, the integration, only agravates the situation. The compensation will only occur after the target point has been passed and the sign of the position error has changed. For security reasons, over-shoot cannot be tolerated in industrial robots.

The over-shoot problem may be solved by determining the direction of the gravity torque on each individual link and directing the integral component force accordingly. However, for a statically balanced robot, it may not always be possible to determine the direction of the gravity torque in the presence of an unknown load.

C.2 Gravity Compensation by Trajectory Modification

A new method of gravity compensation is proposed in which the desired trajectory points are modified by the control algorithm. The proportional controller with velocity feedback uses the modified target point as the normal input. Although the steady-state gravity error of the proportional controlled link will still exist with respect to the modified target point the real error with respect to the original target point will be zero.

Successful application of the proposed method hinges on the desired link path being defined as a series of step inputs. The step inputs are timed so as to create a rudimentary

velocity control over the trajetory. Such a trajectory definition is not uncommon for industrial robot manipulators.

Modifying the desired trajectory can reduce the steady-state error to zero. However, under uncertain conditions (imperfect knowledge of the manipulator or unknown load) the gravity error cannot be predicted. However, the gravity compensation torque is implicity, and iteratively, calculated by the controller. The compensation becomes more accurate as the position error is reduced since as the link velocity approaches zero an increasing amount of the torque provided by the actuator is used to compensate for gravity. In the limit when the link velocity is zero

$$u(t) = T_g \tag{C.4}$$

This can be used to modify the desired link position to eliminate the steady-state error.

$$\theta_{mod}(t) = \theta_d(t) + \frac{u(t)}{K_p K_m} \tag{C.5}$$

In reality the controller output signal $u(t)$ is not a torque but an electrical input to the link motor. The motor then applies a torque on the link according to equation (C.6).

$$Torque = \frac{I_m K_m - F_m \omega}{N} \tag{C.6}$$

where I_m = current to the motor

K_m = motor constant

F_b = motor back emf constant

The trajectory modification expression thus becomes slightly more complex.

$$\theta_{mod} = \vartheta_d + \frac{I_m K_m - F_m \dot{\theta} N}{K_p K_m} \qquad \text{(C.7)}$$

This correction accurately compensates for the gravity torque in the neighborhood of the target point when the velocity is small. However, away from the target point, with a non-negligeble velocity, the compensation will be inaccurate. This is not a serious concern since the trajectory is not controlled between the target points. For this reason this particular compensation method is applicable only to a trajectory defined as a time-referenced series of step inputs, for which no control is expected between the target points.

Another drawback of the method is that, in the above form, instabilities may be encoutered near the target point. This is because a variation in the control force moves the target point θ_{mod}. This then forces a change in the control force $u(t)$ which in turn results in moving the target point, again, and so on. To prevent this from causing system instability, the value of the control force used in the modification expression (for a discrete application of the method) is averaged over the last n iterations before being applied to the correction factor. The motor current for the i^{th} iteration is calculated as follows:

$$I_{m_i} = \frac{\sum_{j=i-1}^{i-n} I_{m_j}}{n} \qquad \text{(C.8)}$$

This smoothing greatly increases the stability of the method.

APPENDIX D

CHANGES IN MASS MOMENTS OF INERTIA

D.1 Transferring Mass Moments of Inertia about Different Axes

In general, the mass moment of inertia of a rigid body about any axis OM, where O is the origin of the reference axis xyz, can be expressed in terms of the reference axis xyz as follows. [34]

$$I_{OM} = I_{xx}l^2+I_{yy}m^2+I_{zz}n^2-2I_{xy}lm-2I_{xz}ln\ -2_{yz}mn \qquad \textbf{(D.1)}$$

where l, m, and n are the direction cosines of axis OM with respect to the coordinate system.

D.2 Parallel axis theorem:

The mass moment of inertia I_{11} of a rigid body about a given axis **11** which is parallel to another axis **22** that passes through the centre of mass of the rigid body can be expressed as

$$I_{11} = I_{22} + md^2 \qquad \textbf{(D.2)}$$

where m = mass of the rigid body

d = perpendicular distance between the parallel axes

D.3 Three Link Revolute Robot

The total mass moment of inertia about θ_1 is the sum of the inertias of all three links about the axis of rotation zz.

$$I_{zz} = I_{zz_1} + I_{zz_2} + I_{zz_3} \qquad \textbf{(D.3)}$$

The inertia of link 1 I_{zz_1} does not vary, but the inertia of links 2 and 3 (I_{zz_2} and I_{zz_3}) about link 1's axis of rotation are functions of their positions. Applying equations D.1 and D.2 the contributions of links 2 and 3 to the inertia about θ_1 can be expressed as follows:

$$I_{zz_2} = I_{y'y'2}Sin^2\theta_2 + I_{z'z'2}Cos^2\theta_2 - 2I_{y'z'2}Cos\theta_2 Sin\theta_2 + m_2(l_2 Cos\theta_2)^2 \qquad \textbf{(D.8)}$$

$$I_{zz_3} = I_{y'y'3}Sin^2(\theta_2+\theta_3) + I_{z'z'3}Cos^2(\theta_2+\theta_3) \qquad \textbf{(D.9)}$$

$$-2I_{y'z'3}Sin(\theta_2+\theta_3)Cos(\theta_2+\theta_3) + m_3[l_2 Cos\theta_2 + \frac{l_3}{2}Cos(\theta_2+\theta_3)]^2 \qquad \textbf{(D.10)}$$

where $I_{x'x'2}$, $I_{y'y'2}$ and $I_{z'z'2}$ are the mass moments of inertia of link 2 about its centre of mass and $I_{x'x'3}$, $I_{y'y'3}$ and $I_{z'z'3}$ are the mass moments of inertia of link 3 about its centre of mass.

The inertia with respect to θ_2 also varies with the manipulator configuration, although it only depends on θ_3 as follows.