A NEW CONTOURING ALGORITHM FOR THE UBC CONTROLLER

by

Rudolf Seethaler

B.A.Sc., University of Toronto, 1991

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

THE FACULTY OF GRADUATE STUDIES

MECHANICAL ENGINEERING

We accept this thesis as conforming

to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

April 1993

Abstract

Conventional servo controllers are not able to provide the high degree of accuracy that is required in many robotic and contour machining applications. New control algorithms are usually difficult to implement in practical machining or robotics environments, since they require extremely fast computer hardware, or a very exact model of the plant. Such algorithms also usually assume that the drive system may be modeled as a linear system; unfortunately this assumption is often violated during contouring operations. The objective of the work reported in this thesis is to utilize the special architecture of the UBC controller to allow the development of control algorithms that allow high speed contouring operations to be undertaken with automatic, real time, error control. The system has been simulated and tested in situations which result in significant non linearity, (sharp corner tracking being a particularly important example.) The results of these tests indicate that the system is able to achieve contouring performance that is better than other systems described in the literature to date.

Table of Contents

List of Tables

List of Figures

Nomenclature

A :	Filter lead constant in z-domain
A(z) :	Denominator (poles) of position loop transfer function
B :	Filter lag constant in z-domain
B(z) :	Numerator (zeroes) of position loop transfer function
$B^+(z)$:	Cancelable zeroes of position loop transfer function
$B^-(z)$:	Non cancelable zeroes of position loop transfer function
D(z) :	Drive transfer function
E :	Error signal
Ex \ Ey :	Position errors
F(s) :	Filter transfer function in s-domain
F(z) :	Filter transfer function in z-domain
G(s) :	Position open loop transfer function
$G_0(z)$:	Inverse of position loop transfer function
$G_f(z)$:	ZPETC transfer function
H(z) :	Controller transfer function
I :	Current
I_{max} :	Current saturation limit
IKF(z) :	Inverse compensation filter transfer function
Je :	Equivalent inertia
K :	Filter gain in s-domain
K1 :	Loop gain
KD	Derivative gain
KI	Integral gain
KP	Proportional gain
Ka :	Amplifier gain
Kd :	D/A gain
Ke :	Encoder gain
Kp :	Filter gain in z-domain
Kss :	Velocity loop gain
Kt :	Torque constant
Ktsa :	Velocity feedback gain
LP(z) :	Linear phase low pass filter transfer function

Mx \ My :	First Stage Interpolated Positions (Master Position)
N :	Order of linear phase low pass filter
N_a :	Slave samples required for acceleration
N_d :	Slave samples required for deceleration
O :	Filtered signal
R :	Angular reference position
Rx \ Ry :	Reference positionsfor lead-lag controllers
SLx \ SLy :	State line flags
SL :	State line
Sx \ Sy :	Second stage interpolated positions
T :	Available torque
Td :	Disturbance torque
U :	Control signal
Vx \ Vy :	Velocity spline signals
V_i :	Master Controller velocities
W(z) :	Cross coupling controller
X_i :	Master positions
a :	Filter lead constant in s-domain
a_0 :	initial acceleration in constant jerk position spline
a_1, a_2, a_3 :	Denominator coefficients of position loop transfer function in the z-domain
b :	Filter lag constant in s-domain
b_1, b_2, b_3 :	Numerator coefficients of position loop transfer function in the z-domain
e_{ss} :	Ramp following error
$e_{allowable}$:	Allowable error
f :	Circular interpolation frequency
f_b :	Position loop bandwidth
f_{max} :	Maximum circular interpolation frequency
j_0 :	Jerk in constant jerk position spline
r :	Circular interpolation radius
r_{max} :	Maximum circular interpolation radius
s :	Distance traveled
t :	Time
v :	Velocity spline velocity
v_0 :	Initial velocity

v_f :	Final velocity
x :	Interpolated position
x_0 :	Initial position
α :	Angular acceleration
α_1, α_2 :	Parameters for transferring the velocity loop to the z-domain
α_{max} :	Maximum angular acceleration
β_1, β_2 :	Parameters for transferring the velocity loop to the z-domain
δ_{static} :	Allowable static error
ΔT :	Slave sampling time
Δt :	Master sampling time
ζ :	Damping ratio of position loop
η :	Performance factor in corner contouring
θ :	Angular actual position
τ :	Amplifier time constant
τ_1 :	Velocity loop time constant
ϕ_{ss} :	Steady state ramp phase lag
ψ :	Angle between master samples at circular interpolation
ω :	Angular velocity
ω_c :	Cutoff frequency of linear phase low pass filter
ω_{max} :	Maximum angular velocity
ω_n :	Natural frequency of position loop

Acknowledgments

I wish to thank my supervisor Dr. Ian Yellowley for providing me with valuable technical and moral support during all stages of this research. He also gave me the benefit of extended commentaries rich in insight and helpful criticism.

I would also like to thank my colleagues for their friendship and support during the course of this work.

Finally funding from the Natural Science and Engineering Research Council of Canada and the Advanced Systems Institute is gratefully acknowledged.

Chapter 1

An Introduction to Robots, Applications, and Previous Research

1.1 Introduction

The necessity to achieve higher quality control and increased productivity has lead progressive companies to examine computer-based automation. At the present time, many tasks are carried out by special purpose machines that are able to perform specific tasks at high production rates. Such systems are generally referred to, in a generic sense, as hard automation systems. It is evident however that in the future, it may well be better to have general purpose machines, that are able to overcome the inflexibility and the high cost of product changes. This usually implies the application of robotic devices.

Early applications of robots usually involved only simple pick and place operations. The robots were required to accurately position objects at certain positions in space, but they did not have to follow a specified path between the final positions with a high degree of accuracy. During the last decade, robots have been put into use in continuous operations such as welding, abrasive finishing, and adhesive applications. These tasks require the actuator to follow the prescribed paths very accurately. At the same time, one seeks to run at the highest speed possible, (consistent with path and process constraints.) Conventional servo controllers are not usually able to achieve the necessary path accuracy at high speeds without extensive path preprocessing.

The objective of the research described in this thesis is to develop a contouring control algorithm that is able to address the requirements of high speed profiling and process control. In the course of the work a candidate robot, to be used in further research, has been selected and refurbished. The robot (a GMF S-108) was retrofitted with high performance actuators and PWM amplifiers. The overall aim of the controller design was to perform all path control activities in real time, and to reduce the cumbersome preprocessing activity used in other approaches. The controller used is an existing UBC designed, multi-processor based system. The architecture allows not only master-slave communication, but also high speed interaction between slave processors. The latter feature has proven to be critical in this application.

1.2 Historical Development of Robots

The overview in this section is based on Craig [1]. References in this section are directly quoted from Craig.

The first work on industrial robots started in the late 1940's. The Oak Ridge and Argonne National Laboratories began work on a remotely controlled manipulator to handle radioactive materials. These systems were designed to reproduce the hand and arm motions of the operator, and they were later equipped with force feedback, allowing the operator to "feel" the forces acting between the slave manipulator and its environment.

In the 1950's the first systems capable of autonomous, repetitive operations were developed. The first industrial robot was introduced by Unimation INC. in 1959. It used a computer in conjunction with a manipulator that could be taught to carry out a variety of tasks.

It quickly became evident that sensory feedback would improve the usefulness of robots. Ernst[2] introduced a mechanical hand with tactile feedback that could feel blocks and use that information to stack them without operator assistance. In 1968 a manipulator equipped with a TV camera and a microphone was built at Stanford. It was able to recognize spoken messages, see blocks scattered on a table, and manipulate the blocks (in a variety of ways.)

In the late 1960's Pieper [3] studied the general kinematics problem of manipulators, while Kahn and Roth [4] analyzed robot dynamics and proposed the first near minimum time control strategy.

During the 1970's, most work was focused on the use of sensors to facilitate manipulative or assembly tasks.

Today robotics research is a very broad field that includes kinematics, dynamics, planning systems, control, sensors, programming languages, and machine intelligence.

1.3 Kinematics and Dynamics

Kinematic analysis is used to study the geometry of motion of a robot with respect to a base coordinate system. It describes the displacement of the manipulator without regard for the forces and moments that cause the motions. The analysis examines the relationship between the joint-variable space and the position and orientation of the end-effector in Cartesian space. Figures 1.1 to 1.4 show a number of typical manupulator types. Paul [5] presents a comprehensive introduction to Kinematics and Dynamics.

There are two basic problems that are of interest. The so-called forward kinematic solution describes the position of the end-effector in terms of the joint variables, while the backward kinematic solution determines the joint-angles in terms of the position and orientation in Cartesian space. Usually the task is stated in terms of the reference coordinate frame. This requires that backward kinematic solutions must be available in order to determine the joint-angles.

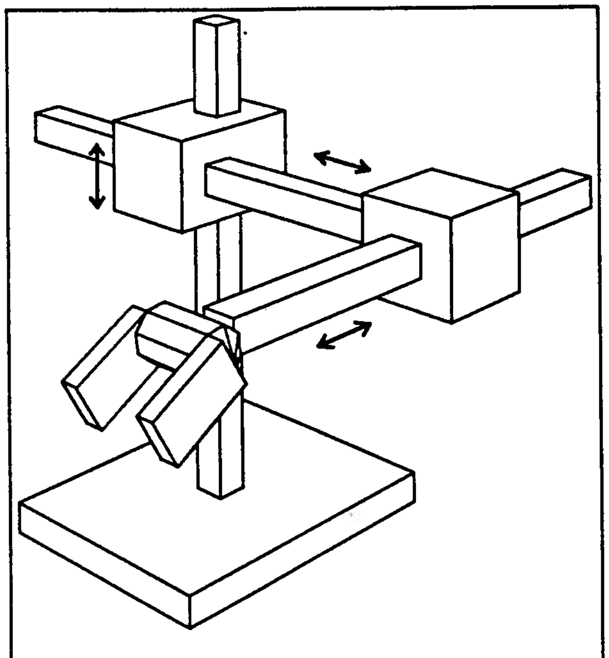


***Figure 1.1.* Cartesian Robot**

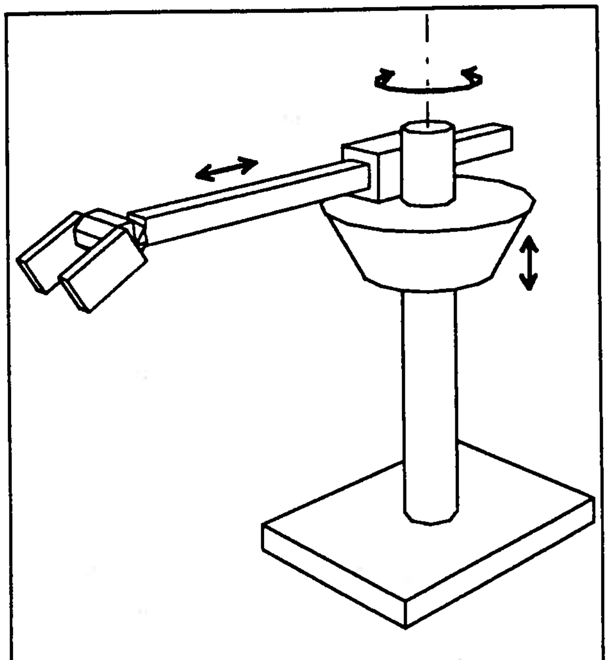


***Figure 1.2.* Cylindrical Robot**

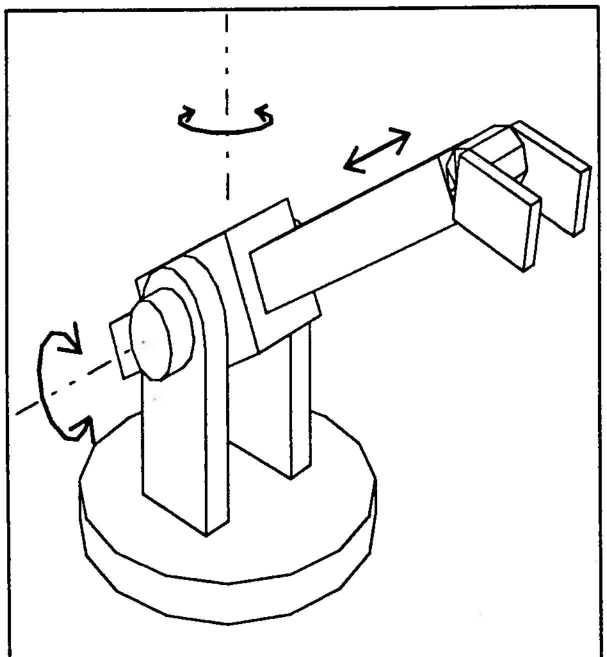


***Figure 1.3.* Spherical Robot**

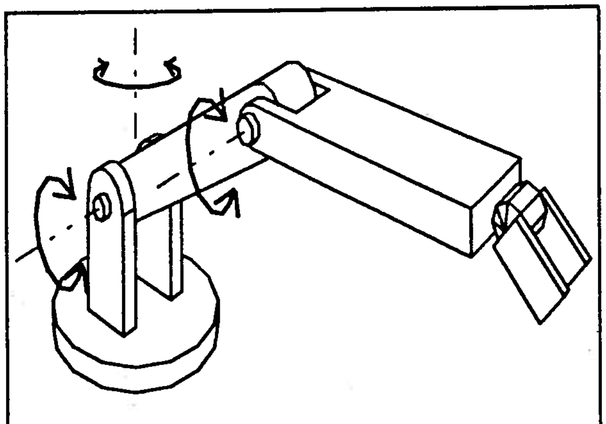


***Figure 1.4.* Revolute Robot**

A matrix representation of link geometries was first presented by Denavit and Hartenberg [6]. This approach allows one to represent the spatial geometry of the links with respect to a base reference frame. It utilizes 4x4 homogeneous transformation matrices that relate adjacent links, and reduces the problem to finding an equivalent 4x4 matrix relating the hand coordinate frame to the reference coordinate frame.

Robot dynamics deals with the mathematical formulation of the equations of robot arm motion. They are useful in computer simulations and the design of robot motion controllers. The dynamic equations are derived from the laws of Newtonian and Lagrangian Mechanics. Using the Denavit-Hartenberg convention it is possible to develop standard procedures to derive the equations of motion in terms of specified geometric and inertial parameters. Two standard procedures for this task are the Lagrange-Euler formulation, first developed by Uicker [7], and the Newton-Euler procedure, introduced by Stepanenko and Vukobratovic [8].

1.4 Drive Systems

There are a number of different drive systems that can be used to power a robot. The type of drive is usually determined by the application (task) for which the robot is designed. Koren [9] describes some of the basic drive systems.

Electrical drive systems are usually used for small to medium size manipulators. Direct current actuators provide good speed regulation, high torques and efficiencies, and have therefore traditionally been used for high precision operations. However the low power to size ratio usually requires gearing systems. Being angular actuators, they require ball screws or similar devices to provide linear motion.

With the development of new high power electronics, alternating current actuators have become more available. They are beginning to replace DC actuators, because they have slightly better characteristics.

Stepping motors are incremental digital drive systems that rotate one angular increment per command pulse. They usually do not have position feedback. This means that they operate in an open loop control system that does not let the controller know whether or not the actuator has actually performed the prescribed motion. Large position and velocity errors cannot be detected. Usually such actuators have a very limited power and torque capacity.

Hydraulic actuators are used for large robots. They provide high power with small units, and the response characteristics are very good. Linear and rotary actuators are readily available and no gearing is required. Hydraulic actuators however have several disadvantages. They require highly pressurized liquid as the operating fluid, which often causes maintenance problems. The response characteristics change with the temperature of the working fluid and they are also strongly influenced by entrained materials. The relative cost of a hydraulic system does not decrease with size and hence they are too expensive for small systems. Finally the various devices used in hydraulic control elements are often distinctly nonlinear, leading to significant design problems.

Pneumatic actuators are usually used in small manipulators, which are designed for point to point operations. They have fast response times, and compressed air is usually readily available. On the other hand they require relatively large actuators, because of the low pressures, and they produce a considerable amount of noise.

1.5 Trajectory Planning

With the knowledge of kinematics and dynamics of the actuators one would like to control the actuators to make the manipulator follow a desired path. The space curve that the manipulator describes while moving from an initial location (position and orientation) to a final location, is called a path. In trajectory planning one is interested in constructing such a path. Craig [10] discusses trajectory planning in a broader scope than this introduction allows.

There are three constraints that enter trajectory planning. The path constraint specifies what path the manipulator has to follow. The obstacle constraint restricts the path to manipulator motions that will not interfere with other objects (obstacles) along the trajectory of the manipulator. Finally it is of course impossible to put infinite power into an actuator, and hence discontinuities in the path and its first second and third derivative should be avoided. This last constraint is referred to as the manipulator's dynamic constraint.

There are basically two approaches to trajectory planning. In the first approach knot points are selected along the path in the manipulator's generalized coordinate frame (joint coordinates). Using constraints such as continuity and smoothness of position, velocity, and acceleration, polynomials are fitted between the knot points. They guarantee that the constraints are met at the knot points. With this method it is difficult to check for obstacle avoidance, because the trajectory is planned in joint coordinates, making it difficult to trace the path of the manipulator in Cartesian space.

The second approach specifies the path in an analytical function such as a straight line or a circle in Cartesian space. Thus, obstacle avoidance can be checked for in the initial stage of calculation. The Cartesian path constraint must then be transferred to joint path constraints. Again a splining operation is conducted during the fit between the joint knot points, which will ensure continuity and smoothness of position, velocity, and acceleration.

Usually the actuator dynamics are not considered in the trajectory planning phase, because of their computational complexity. Thus, large tracking errors will result within the servo control of the manipulator, if the actuators cannot supply the torque required for the desired path.

1.6 Motion Control

Motion control is the process of controlling the actual actuators of a manipulator in order to achieve a desired trajectory. Motion control is usually divided into two parts. First the dynamic model of the manipulator is obtained. This model is then used to determine a suitable control law. There are usually two distinct phases in a motion segment. First the manipulator is moved from its initial position to the vicinity of the final location along a planned trajectory. In the second segment fine-motion control often in combination with sensory feedback is employed to make the end-effector dynamically interact with the object.

Currently most industrial robot controllers do not utilize information from the dynamic equations of motion, but rather use simple servo controllers. This leads to the requirement for slow speeds, (often accompanied by unnecessary vibrations). In order to improve the performance, more sophisticated control approaches and dedicated computer architectures are often utilized. There are two basic philosophies in modern contouring control. The first attempts to minimize path error by minimizing the position error of each axis. The usual way that this is achieved, is to insert a filter before the control loop that is the exact inverse of the actual control loop. In theory this should lead to a unity transfer function. The initial problem, that such a controller is often unstable, was solved by Tomizuka [11] by only canceling the stable zeroes of the system. Still this approach had two flaws. First it requires a very accurate model of the system, since otherwise the position loop cannot be canceled exactly. One would therefore have to use an exact dynamic model of the actuators, and the equations of motions of the robot arm, including loads and disturbances. This is extremely computationally complex, and it is often impossible to extract exact information on the load and disturbances. The second problem arises from the saturation limits of the amplifiers and motors. In order to cancel the position loop dynamics, large control signals are required, which can saturate the amplifiers. This problem becomes more pronounced when the frequency components of the reference path increase. (Thus, the system performance deteriorates around corners.) Both problems have been addressed by Weck [12]. He proposed the use of a regressive system characterization to determine the position loop model, and used this information, instead of an analytical model, to design the inverse filter. In his proposal the system characterization is carried out in the start up phase. Therefore the system model does not contain loads and position dependent arm dynamics. In order to get a truly accurate model it would be necessary to perform system characterizations throughout the complete operating cycle. This in turn poses problems with large computational efforts. Weck also proposed the use of a linear phase low pass filter in order to reduce the high frequency components in the reference signal. This idea seems to work very well for improving corner tracking.

The second approach was first proposed by Koren [13]. This approach attempts to minimize the path error directly instead of the single axis error. The approach requires the controller to calculate the path error from the single axes errors and then use this path error to compensate the different axes, a strategy which clearly requires cross coupling of errors. The controller is complicated and the computational effort rises with the square of the number of axes. To this date the author has not been able to find a paper that discusses the implementation of this system on high performance servos, which require fast sampling times in order to fully utilize their capabilities. It seems that the computational effort is too large to actually implement this system with today's computing capacity.

The algorithm that is proposed in this thesis does not rely on an exact dynamic model. It therefore does not require regressive system characterizations. The algorithm relies on the principle that an error proportional to the reference velocity in each axis leads to a phase lag between the actual and the reference signal, but the path accuracy itself is not compromised by such an error. The algorithm then predicts an allowable error proportional to the velocity, compares it with the actual error, and inserts extra time steps in those cases where the allowable error is exceeded. This algorithm is very fast and has been implemented in real time. It can be implemented on other industrial equipment without major changes, because it is not dependent on an exact dynamic model.

1.7 Sensing

The use of external sensors allows a robotic manipulator to interactively adapt to changes in its environment. Even though, at present, most tasks only require the robot to perform preprogrammed repetitive tasks, a great deal of effort has gone into the development of new sensory techniques. Merrit [14] has presented a detailed survey article on robot sensing.

There are two categories of sensors. Internal sensors detect variables such as arm joint position that are used by the controller. External sensors detect variables such as range, proximity, touch, force and torque, and vision. They are used for guidance as well as object identification. Vision is potentially the most powerful robot sensory capability. It can be defined as the process of extracting, characterizing and interpreting information from images of a three dimensional world. It is often divided into sensing, preprocessing, segmentation, description, and interpretation.

The unique architecture of the UBC controller that is described in Chapter 3 allows the integration of contouring control with external sensory inputs. Force sensors in metal cutting operations have been used to ensure a constant cutting force and a constant equivalent chip thickness.

1.8 Programming Languages

Communication between the operator and the manipulator is the major obstacle in utilizing a general purpose assembly machine. Fu et al. [15] have presented an introduction to this topic.

There are three major approaches to this problem. Discrete word recognition is very limited, because state of the art speech recognition can only recognize discrete words from a limited vocabulary. It is also speaker dependent, it needs very powerful computers for efficient real time algorithms, and it requires a large memory space to store the speech data. As a result discrete word recognition has not yet found its way into real industrial applications.

When using teach and playback, an operator slowly leads the robot (off line) through the entire assembly task with a manual control. A computer records points along that path in order to be able to replay the motion at a later time. The taught motion is played back and edited until it contains no errors. Finally the robot is run on line at its appropriate speed to perform the taught motion repetitively.

The most advanced approach to solve the communication bottleneck between the operator and the manipulator, is to use high level programming. When using a robot for such tasks as welding or spray painting, no interaction is required between the robot and its environment. If the manipulator is used for assembly purposes, sensory feedback is usually required. This can only be handled by conditionally programmed methods that are best implemented using high level programming techniques. There are two programming categories for robotics applications.

In robot-oriented programming assembly tasks are divided into a number of robot motions. The programmer has to guide the robot through the task by sequentially programming every single motion segment. This procedure is very cumbersome making it difficult to use.

In task-oriented programming, assembly tasks are described in terms of objects being manipulated rather than by robot motions. This means that the programmer can specify a task such as picking up an object in a high-level language, and the program then uses a database to convert the task to a robot-level program. This procedure is much more user friendly than robot-oriented task planning, but many problems, notably task planning, obstacle avoidance, and sensory information utilization still remain to be solved.

The UBC controller, which is described in detail in Chapter 3, uses high level programming that allows easy trajectory programming which can be integrated with technological sensory feedback to ensure optimal operating conditions.

Chapter 2

Previous Approaches to the Development of Fast Contouring Controllers

2.1 Introduction

In the last decade many researchers have begun to examine the development of high speed contouring algorithms. This has been accompanied by the development of faster computer hardware that allows one to run control strategies that were not previously feasible.

Two main approaches have arisen from this previous work. The first control strategy, proposed by Tomizuka [11] tries to eliminate path error by achieving zero axis following error. It attempts to cancel the traditional control loop with an inverse filter of the position loop. For this system to perform adequately, it is necessary to have a very exact model of the plant.

This strategy unfortunately often leads to the saturation of amplifier current and the violation of other torque-related constraints, during phases of large acceleration. Weck [12] proposed a solution to this problem which involves placing a low pass filter in front of the controller. This alleviated the saturation problem, and better corner tracking was achieved.

The second approach, introduced by Koren [13], tries to minimize path error alone, while still allowing actual axial position errors. It calculates a path error from the individual errors of the axes and then uses this path error to compensate the individual axes. This strategy requires very fast hardware, and to date the author has not been able find a paper on an experimental setup using high performance servos in combination with this strategy.

2.2 Comparison of Existing Approaches

A paper comparing the various advanced controller strategies has been presented by Koren and Lo [16]. The course of their argument will be followed in this introduction, following which an introduction to existing methods and the new concepts arising from the author's work, will be compared with the simulated and experimental results obtained by Koren.

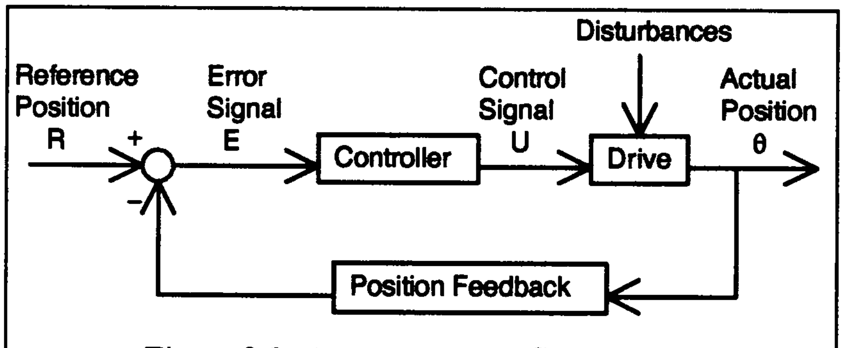


Figure 2.1. Typical Position Control Loop

Figure 2.1 shows a typical servo system. The actual position is fed back and subtracted from the desired position. This error signal is fed into a controller. The controller then feeds a velocity command to the plant. Usually the plant may be modeled as a first order lag. In the simplest case, the controller has a simple proportional gain. This system has been used in many NC machines. Advanced CNC applications require very high feedrates, which lead to large contouring errors with P-controllers. Thus, P-controllers are often not adequate for advanced CNC applications.

Three basic methods have been developed to overcome the limitations of a simple proportional control action:

The first approach applies more sophisticated axial controllers such as PID or state feedback controllers. These controllers attempt to reduce the position errors of the individual axes.

The second approach adds a feed forward controller, such as the ZPETC or IKF approaches discussed in the introduction, to compensate for axial position errors.

The third method uses cross coupling in order to reduce the contour error rather than the axial tracking error. The controller proposed in this thesis has some of the characteristics of the third method, since the axes can communicate with each other via a state line. Unlike the original system proposed by Koren, this proposed system is very simple. It can handle more degrees of freedom and it is possible to optimize contouring performance and technological factors such as maximum cutting forces, at the same time and with very little more effort.

There are variations and accessories to these three basic control approaches. The most prominent are adaptive control and predictive control as mentioned in the introductory chapter.

2.3 Sources of Error

There are three different types of error:

The first are due to mechanical errors. This category contains backlash or nonstraightness of the table motion. It cannot be improved upon by the simple addition of more sophisticated controllers. Those errors can only be reduced by using better hardware or compensation techniques.

The source for the second type of error is the effects of the production process. Machine tool operations for instance lead to errors due to tool wear or tool deflection. These errors must also be handled through compensation.

The third type of error arises from the controller and drive dynamics. These errors are often dominant in high speed operations, and they can be alleviated by improving the control algorithm. The three above controller types and the one proposed in this thesis all concentrate on this third class of error, which will further be subdivided into three categories:

The first category is parameter mismatch. In a type-I control system, a mismatch in the open loop gains will lead to a steady-state error when following a straight line. Different time constants on the other hand, will lead to transient contour errors that may be more prominent than those due to mismatched gains.

The second type of error is due to disturbances. The disturbances result from cutting forces or other parameters imposed by the process.

The third kind of error stems from the contour path and the machine dynamics. When the desired path contains a discontinuity or a sharp corner, the system is often unable to follow the path within the prescribed tolerances. The major problem in such cases is the fact that the maximum acceleration is limited by the saturation current of the amplifier.

Most of these errors increase with feeding speed. This means that more effective servo controllers are needed for high speed operations.

2.4 Feedback Controllers

In this category controllers with basic feedback only will be listed.

2.4.1 P-Controller

The P-controller is a proportional gain that is usually tuned such that the closed loop damping ratio is approximately equal to 0.707. This controller works well for low feed rates.

2.4.2 PID-Controller

The PID-controller adds the integrated error and a derivative of the error to the control signal. The integral term ensures that the system has zero tracking error for ramp inputs (assuming the uncompensated system is type I), and the derivative term shapes the dynamic response of the system. The PID-controller provides poor tracking and significant overshoots for nonlinear contours like sharp corners which cannot be described with first order polynomials. These problems are addressed by preprogramming a path with sufficiently small accelerations.

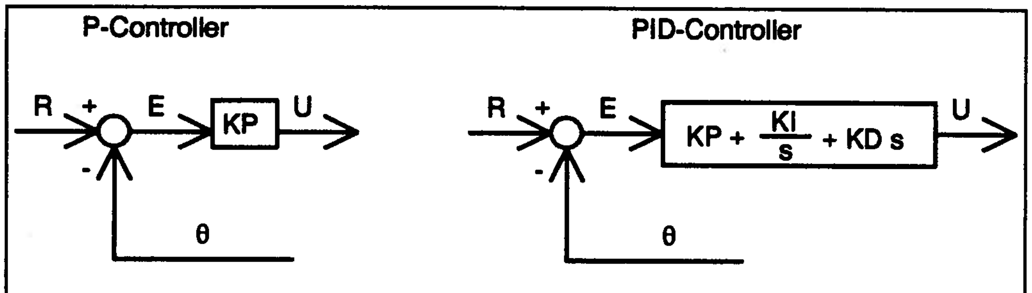


Figure 2.2. P-Controller and PID-Controller Structure

2.5 Feedforward Controllers

The idea behind this type of controller is to implement a filter in the control loop that leads to a unity transfer function of the system.

2.5.1 Zero Phase Error Tracking Controller (ZPETC)

The ZPETC controller was first proposed by Tomizuka [11]. It introduces a filter before the position feedback loop in an attempt to cancel the zeroes and poles of the system. A block diagram of this controller is shown in Figure 2.3.

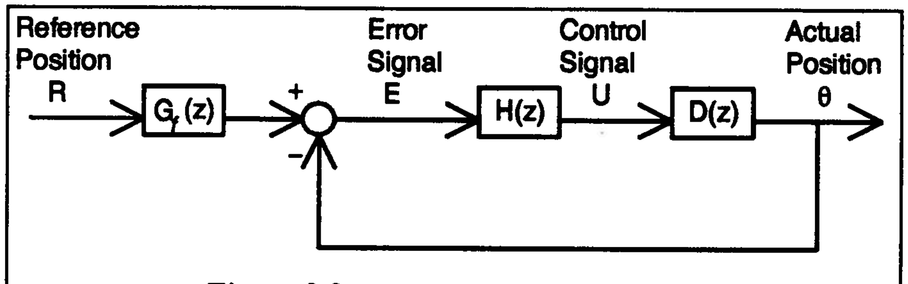


Figure 2.3. Block Diagram of ZPETC

$G_0(z)$ is the feedforward filter that is intended to cancel the position loop. Ideally it should have the following form:

$$G_0(z) = \left(\frac{B(z)}{A(z)}\right)^{-1} = \left(\frac{H(z)\,D(z)}{1+H(z)\,D(z)}\right)^{-1} \tag{2.1}$$

It is not possible to cancel the zeroes outside the unit circle because to do so would lead to an unstable controller. The zeroes are divided into cancelable and non cancelable zeroes:

$$G_0(z) = \left(\frac{B^+(z)\,B^-(z)}{A(z)}\right)^{-1} \tag{2.2}$$

Tomizuka only cancels the cancelable zeros $B^+$and arrives at the following controller:

$$G_f(z) = \frac{A(z^{-1})\,B^-(z)}{B^+(z^{-1})\left[B^-(1)\right]^2} \tag{2.3}$$

The phase angle of the overall transfer function of this system is zero, which implies that this system achieves zero phase error tracking. Haack and Tomizuka [17] proposed the reduction of the possible gain error by adding zeroes to the feedforward controller.

This system requires that the model of the system be very accurate. Disturbance torques will severely reduce the performance of the system.

This controller also produces very large control signals that can saturate the amplifiers. Buttler et al. [18], recognizing this problem, have proposed the use of path planning to avoid large accelerations.

2.5.2 Inverse Compensation Filter (IKF)

The ZPETC controller inherently requires large control signals which saturate the amplifiers and motors during moves with high frequency components. This problem was alleviated by Weck [12] by putting a low pass filter with linear phase in front of the ZPETC. Figure 2.4 shows the a block diagram of the IKF Controller.

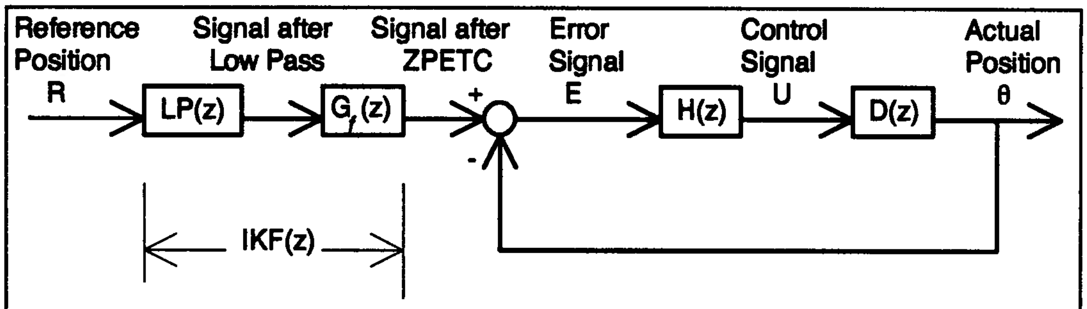


Figure 2.4. Block Diagram of IKF

The ideal IKF filter has the transfer function described in equation 2.4:

$$IKF(z) = \left\{ \frac{\omega_c \, \Delta T}{\pi} + \sum_{i=1}^{N} \frac{[1+\cos(i\,\pi/N)]\sin(i\,\omega_c\,\Delta T)}{2\,i\,\pi}\left(z^i + z^{-i}\right) \right\} \left\{ \frac{A(z^{-1})\,B^-(z)\,z^d}{B^+(z^{-1})\left|B^-(1)\right|^2} \right\} \quad (2.4)$$

Here *N* is the filter order, and Weck proposed that *N* be equal to the ratio of the drive time constant and the controller sampling time. ΔT is the controller sampling time, ω_c is the cutoff frequency of the speed controlled drive system, and *d* is the estimated dead time.

Weck also proposed the use of a recursive least square model identification according to the Bierman factorization method, instead of an arithmetical model. This should eliminate large computing efforts and the insecurity of model design.

The results obtained with this method seem to solve most of the ZPETC controller's problem. Still the system tends to become unstable for sampling times of less than 5 ms. The threat of saturating the amplifiers has been considerably decreased, but not eliminated.

2.6 Cross Coupling Controller (CCC)

This controller was first proposed by Koren [13]. It is based on the idea that, in order to improve the contouring performance, it is necessary to eliminate the path error rather than individual axis errors.

This controller uses a contour error model that is then implemented in a simple control law. Figure 2.5 shows a block diagram of a cross coupled controller.

The error model is given by equations for *Cx* and *Cy*, which are functions of the instantaneous slope and the radius of the contour, and the errors of the system, respectively.

The cross coupling controller uses a simple PID control law of the form described in equation 2.5 :

$$W(z) = KP + KI\frac{\Delta T\, z}{z-1} + KD\frac{z-1}{\Delta T\, z} \tag{2.5}$$

This controller structure has been tested by Koren and Lo [19]. It was able to give an improvement in performance of 1:5 to 1:10 compared to a simple P-controller.

The main disadvantage of this controller is that it requires very fast hardware in order to perform in real time. At this time a three axis system is operating satisfactorily. This three axis system has a slow time constant, which in turn requires only a low sampling rate. It seems that it will be difficult to use this system with high performance servos, as such servos need very fast sampling rates to give optimum results.

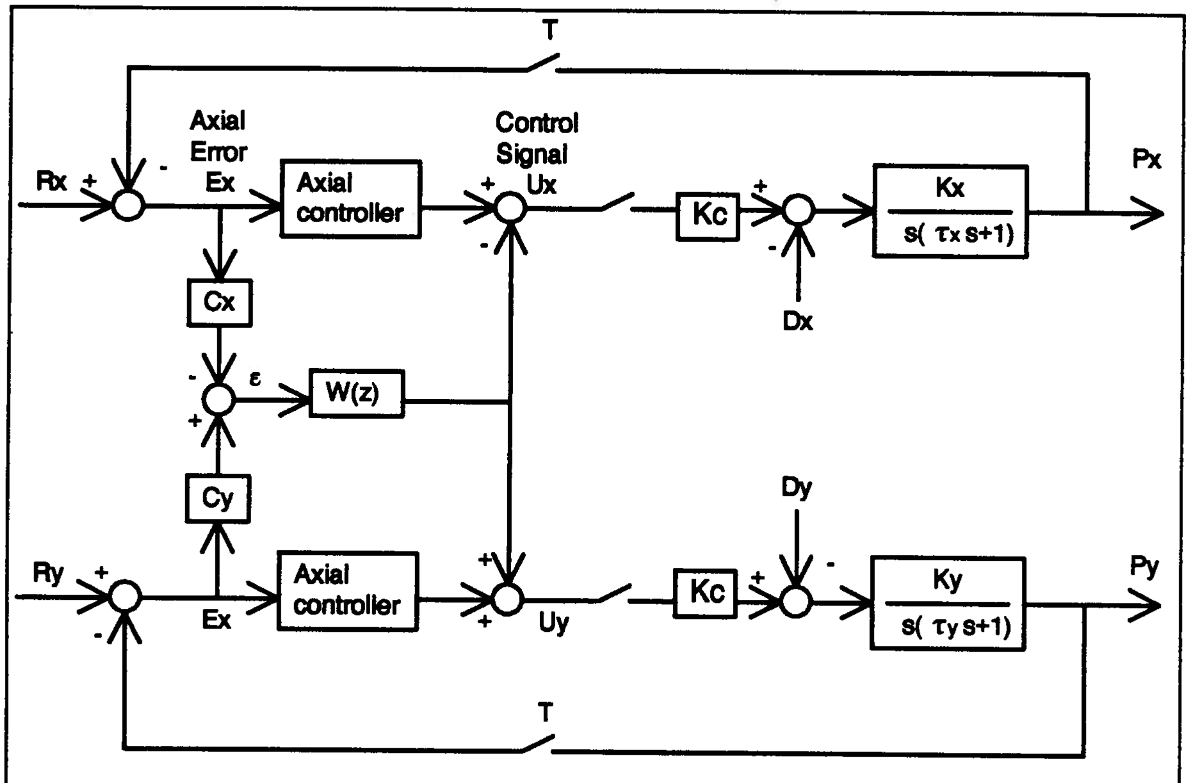


Figure 2.5. Block Diagram of a Two Axis Cross Coupled Controller

Chapter 3

UBC Controller

The UBC controller is an open architecture multi-axis controller that was specifically designed to coordinate a large number of axes, and to allow integration of process control with position and velocity control.

3.1 Controller Architecture

The controller utilizes the STD 32 Bus. The master is an IBM PC compatible computer, which can control up to 15 slave controllers. In the work described in this thesis, the master updates the position of each slave every 16 ms; this corresponds to approximately twice the time constant of the slave servos. The slaves are digital lead lag controllers that interpolate 32 times between every master position. They have a sampling time of approximately 0.5 ms.

The architecture ensures that most of the workload is taken from the master and transferred to the slave controllers.

Every slave controller is connected to a so-called state line. If the state line is high, the slaves perform as ordinary lead-lag controllers. If the state line is low however, the slaves will not increment position, but will continue to close the position loop. Each slave is able to pull the state line low, in response to conditions where some error criterion is met. The arrangement described means that each slave is able to slow down or stop the complete system when it cannot follow the required path at the programmed velocity.

Figure 3.1 shows a schematic diagram of the controller architecture :

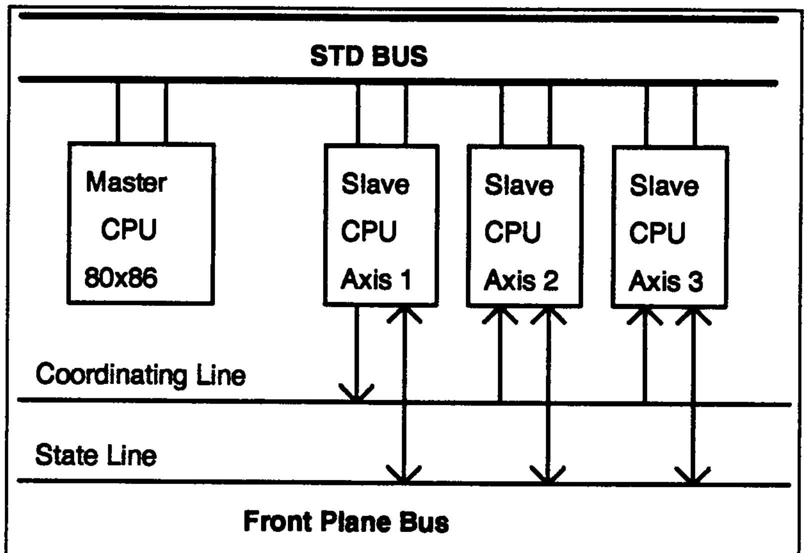
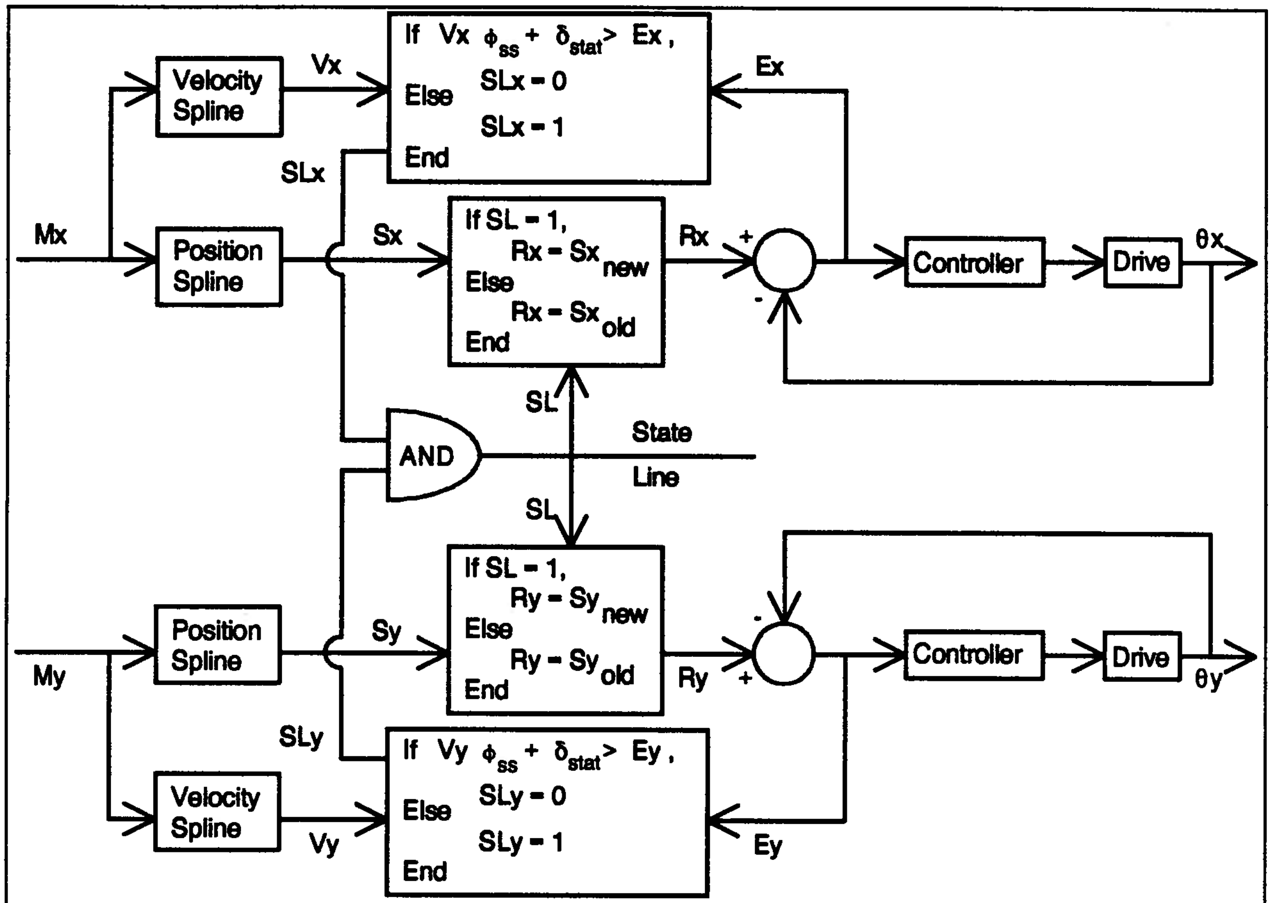


Figure 3.1. UBC Controller Architecture

The slaves receive position incremnents every master sampling time. These position increments are modified twice. First the slaves perform a second stage interpolation. This procedure reduces the sampling time to half a millisecond and it also provides velocity shaping in order to reduce acceleration levels between position increments. The second operation that is performed on the reference signal is velocity modulation with the state line. This concept allows the reduction of acceleration levels below the saturation limits of the amplifiers. The signal modified in this way is then fed into a simple lead-lag servo controller. It should be noted, that it is quite difficult to analyze the complete slave controller with conventional control techniques, because the system is not linear and the author was not able to obtain a closed transfer function of the system. A comprehensive stability analysis of the system has not been performed, but simulations and experiments of the control system including the state line strategy tend to be more stable than the simple servo controller. A schematic block diagram of a two axis control system is shown in Figure 3.2.



If Vx ϕ_{ss} + δ_{stat} > Ex,
SLx = 0
Else
SLx = 1
End
Velocity Spline
Vx
Ex
SLx
Mx
Position Spline
Sx
If SL = 1, Rx = Sx$_{new}$ Else Rx = Sx$_{old}$ End
Rx
Controller
Drive
θx
SL
AND
State Line
If SL = 1, Ry = Sy$_{new}$ Else Ry = Sy$_{old}$ End
Sy
My
Ry
θy
SLy
If Vy ϕ_{ss} + δ_{stat} > Ey, SLy = 0 Else SLy = 1 End
Vy
Ey

Figure 3.2. Schematic Block Diagram of a Two Axis Control System

Mx \ My	First Stage Interpolated Position
Sx \ Sy	Second Stage Interpolated Position
Rx \ Ry	Reference Position of Lead-Lag Controller
Vx \ Vy	Velocity Spline
Ex \ Ey	Position Error
SLx \ Sly	State Line Flags
SL	State Line

In the following sections the lead-lag controller, the position spline, and velocity modulation using the state line will be discussed.

3.2 Slave Transfer Function

The block diagram of the position loop of the conventional lead-lag controller of the slaves is shown in Figure 3.3:

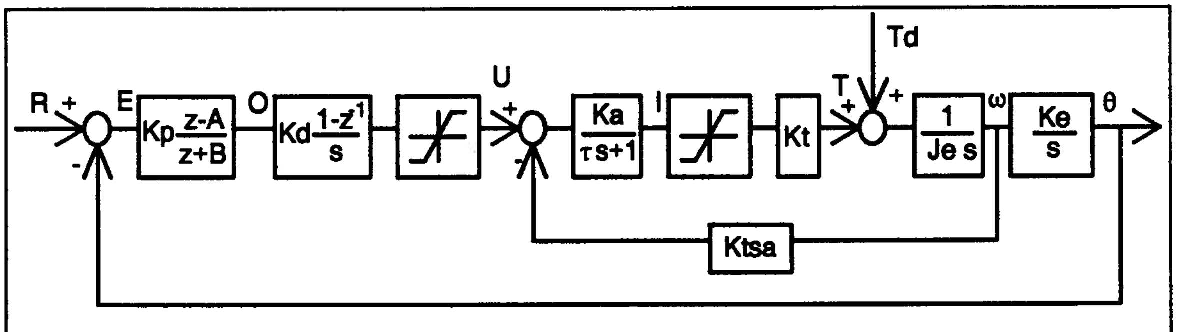


Figure 3.3. Block Diagram of Position Loop

The block diagram shows that the system is half continuous and half discrete. In order to analyze the system it is first necessary to transform it into either form.

The easiest approach in this case, where the sampling frequencies are relatively high, is to convert the digital filter and the zero order lag to a continuous filter. The continuous filter will have the following form :

$$F(s)= \frac{O}{E} = K\frac{s+a}{s+b} \tag{3.1}$$

Using forward difference approximations this can be rewritten as :

$$K\,E_{k+1} + K(a\,\Delta T - 1)E_k = O_{k+1} + (b\,\Delta T - 1)O_k \tag{3.2}$$

This corresponds to the lead lag filter in the z-Domain given by :

$$F(z) = K\frac{z+(a\,\Delta T-1)}{z+(b\,\Delta T-1)} = Kp\frac{z-A}{z+B} \tag{3.3}$$

The transformations from the continuous to the discrete domain are :

$$\begin{aligned} A &= 1 - a\,\Delta T \\ B &= b\,\Delta T - 1 \\ Kp &= K \end{aligned} \tag{3.4}$$

A simplified continuous block diagram is shown in Figure 3.4. (Both saturation limits and the amplifier bandwidth are neglected in this case.)

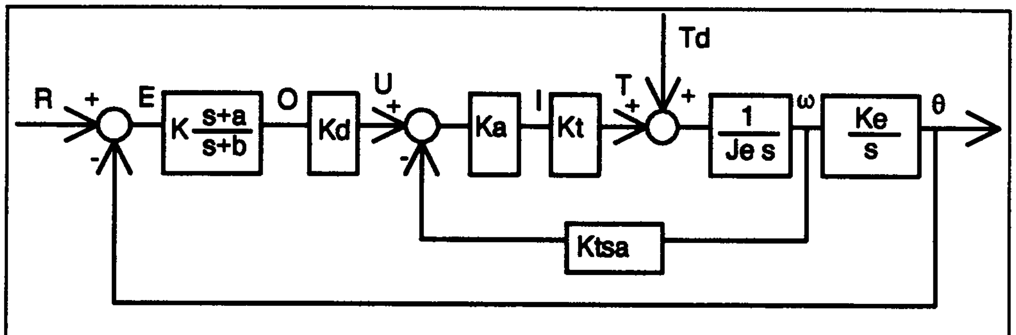


Figure 3.4. Block Diagram of Continuous Position Loop

The velocity loop is a first order lag of the form:

$$\frac{\omega}{U} = \frac{Kss}{\tau_1 s + 1} \tag{3.5}$$

where :
$$Kss = \frac{1}{Ktsa}$$
$$\tau_1 = \frac{Je}{Ka\,Kt\,Ktsa}$$

The resulting block diagram is shown in Figure 3.5

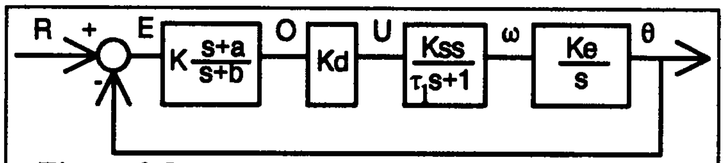


Figure 3.5. Position loop of simplified continuous model

There are two very important characteristics of this system. First, it has a limited acceleration due to current saturation. By inspecting Figure 3.3 it becomes apparent that the maximum achievable acceleration is :

$$\alpha_{max} = I_{max} Kt \frac{Ke}{Je} \tag{3.6}$$

The second important characteristic is the expected following error to a ramp input. The open loop transfer function of the model in Figure 3.5 is :

$$G(s) = K\, Kd\, Kss\, Ke \frac{s+a}{(s+b)(\tau_1 s+1)\, s} \tag{3.7}$$

This is a type I system and the steady state error to a ramp input can be shown to be:

$$e_{ss} = v \frac{b\, Ktsa}{a\, K\, Kd\, Ke} = v\, \phi_{ss} \tag{3.8}$$

This means that the following error is proportional to velocity. In other words during a ramp, the steady state response will result in the actual signal always lagging a constant time behind the reference signal. If the different axes have the same gains, the path error will be zero, even though the actual path lags behind the reference.

Equation 3.8 also shows that it is possible to control the velocity of the system by controlling the error.

An equivalent expression for the following error in the z-domain is derived in Appendix 1. It yields the same numerical values as equation 3.8, but it is much more complex. The simple expression derived in the s-domain is used for further analysis and calculation.

3.3 Position Spline

Before the reference positions are transferred from the master to the lead-lag slave controllers, the slaves performs an interpolation operation. This operation is called position splining and it aims to reduce large accelerations of the reference signal by shaping the velocity during master samples. The position spline also transfers some of the work of the master to the slaves, because the master needs to send less position increments to the slaves. This enables the master to concentrate on other operations such as task planning.

The Interpolation scheme was developed by Yellowley and Pottier [20]. It assumes that points are evenly spaced in time, and it assumes a linear variation in velocity over each master sampling period. This leads to the following formula for displacement:

$$x = X_i + \left[X_{i+1} - X_i\right]\frac{t}{\Delta t} + \frac{\left[X_{i+2} - X_{i+1} - X_i + X_{i-1}\right]}{2\Delta t^2}\cdot\left[\frac{t^2}{2} - \frac{\Delta t}{2}t\right] \tag{3.9}$$

where $0 \leq t \leq \Delta t$

Figure 3.6 shows the velocity profile which results from this scheme. The spline assumes a constant acceleration and zero jerk over a master sampling interval. At every master sample a velocity discontinuity is encountered.

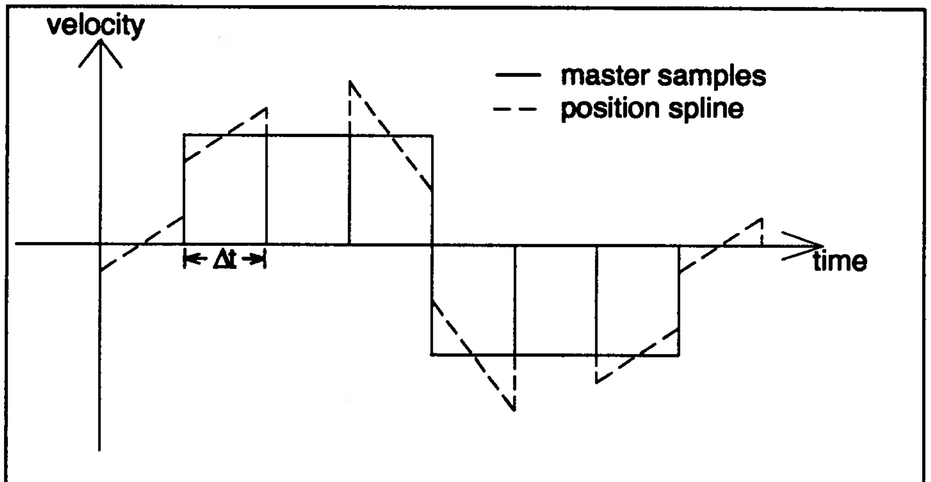


Figure 3.6. Position Spline

A position spline with linear acceleration and constant jerk that would have a continuous velocity profile was investigated, as a first alternative, and its derivation is given in Appendix 2. The added computational complexity did not yield a significant improvement in performance and so the constant acceleration spline was maintained, together with the addition of a velocity spline which is described in a later section.

3.4 The Application of the State Line to the Minimization of Error

It was shown in the previous section that the position spline contains infinite accelerations. As shown in equation 3.5 however, there is a maximum achievable acceleration. The state line will be used to smooth the velocity discontinuities. The basic idea, which will be expanded upon in this section, is to allow the smoothing of velocity and acceleration by triggering the state line based upon error. The reader will realize that this is the inverse of the normal procedure. This is achieved by controlling the amount of error, and hence the velocity before and after a large acceleration. In the proposed system one is indirectly controlling the velocity through error.

3.4.1 Velocity Spline

Equation 3.8 allows one to stipulate an error criterion for the state line :

$$e_{allowable} = v\left[\frac{b}{a\,K\,Kd\,Kss\,Ke}\right] + \delta_{stat} = v\phi_{ss} + \delta_{stat} \tag{3.10}$$

The first term allows for the expected error proportional to velocity and δ_{stat} allows for friction, and other imperfections in the system.

Ideally an actual reference velocity should be used to calculate the allowable error. The reference velocity based upon the position spline contains accelerations that are beyond the possible system limits described by equation 3.5. It is thus necessary to form a reasonable independent velocity spline to permit the allowable error to be calculated. The most logical approach is to assume a velocity spline that linearly connects the average velocities of two adjacent master velocities and thus contains no velocity discontinuities. It takes two master sampling intervals to accelerate from zero to a constant velocity; this corresponds to about 4 times the time constant of the non-saturated position loop. The spline is illustrated in Figure 3.7 :

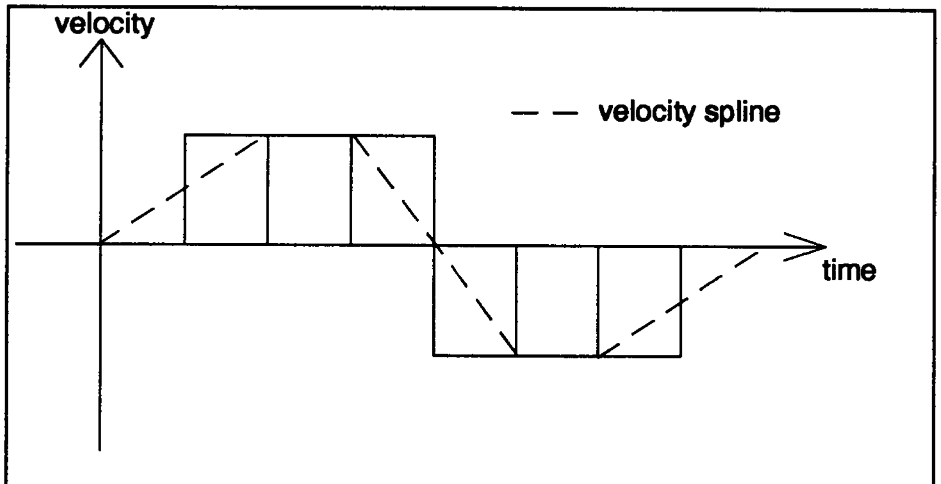


Figure 3.7. Velocity Spline

The spline itself can be described mathematically in the form :

$$v(t) = \frac{1}{\Delta t}\left[(X_{i+1} - X_{i-1}) + (X_{i+2} - X_{i+1} - X_i + X_{i-1})\frac{t}{\Delta t}\right] \tag{3.11}$$

3.4.2 State Line Triggering Logic

The intention of the approach should by now be clear; the author intends to trigger the state line when the actual error is greater than would be expected, should the velocity be that of the velocity spline. There are however several ways in which the line may be triggered.

In the first series of experiments the state line was triggered using an absolute error criterion of the form:

$$|e_{allowable}| \leq |\phi_{ss}\, v| + \delta_{static} \tag{3.12}$$

This criterion then allows an error band around the actual position. It means that the state line will be triggered when the actual position is leading the reference position. Such an approach forces the system to slow down more than necessary. A sign sensitive error criterion was therefore formulated.

When the reference velocity is positive, the criterion is:

$$\begin{aligned} &v > 0: \\ &e_{allowable} < (\phi_{ss}\, v) + \delta_{static} \end{aligned} \tag{3.13}$$

In the case of negative reference velocity, the criterion changes to :

$$\begin{aligned} &v < 0: \\ &e_{allowable} > (\phi_{ss}\, v) - \delta_{static} \end{aligned} \tag{3.14}$$

In order to combine both constraints into one single equation it is necessary to negate the second constraint. This is equivalent to multiplying either equation by the sign of the velocity. Hence the final constraint used is given by :

$$sign(v)\, e_{allowable} < sign(v)\, \phi_{ss}\, v + \delta_{static} \tag{3.15}$$

3.5 Sampling Period

For the velocity spline to give satisfactory results, it is necessary to choose a master sampling time that allows the system sufficient time to slow down from maximum speed to zero speed within one master sampling period.

One way to determine the required master sampling period is to calculate the distance required for the above mentioned deceleration, and transform it to the required number of slave samples per master sample:

The time required to accelerate from v_0 to v_1 can be calculated using :

$$t = \frac{v1 - v0}{a_{max}} \tag{3.16}$$

This requires a distance of

$$s = v_0\, t + \frac{1}{2}\, a_{max}\, t^2 = \frac{v_1^{\,2} - v_0^{\,2}}{2\, a_{max}} \tag{3.17}$$

During a deceleration phase, this requires that the state line starts triggering N_d slave samples before the velocity discontinuity, where :

$$N_d = \frac{s}{v_0\, \Delta T} = \frac{v_1^{\,2} - v_0^{\,2}}{2\, v_0\, a_{max}\, \Delta T} \tag{3.18}$$

During an acceleration phase, this requires that the state line stops triggering Na slave samples after the velocity discontinuity, where :

$$N_a = \frac{s}{v_1\, \Delta T} = \frac{v_1^{\,2} - v_0^{\,2}}{2\, v_1\, a_{max}\, \Delta T} \tag{3.19}$$

3.6 Filter Parameters

The position loop without the state line is a third order system :

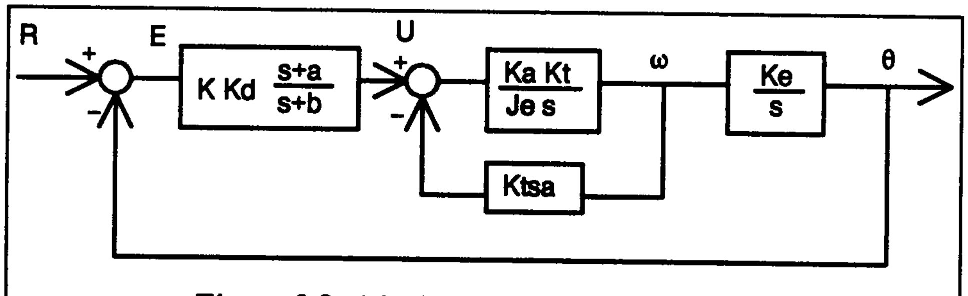


Figure 3.8. Block Diagram of Position Loop

This third order system can be reduced to a second order system by canceling the dynamics of the velocity loop with the filter. This leads to :

$$a = \frac{Ktsa\ Ka\ Kt}{Je} \tag{3.20}$$

The position loop block diagram reduces to the form shown in Figure 3.9:

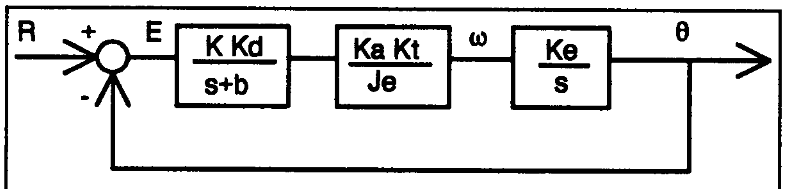


Figure 3.9. Block Diagram of Reduced Position Loop

The transfer function of this system is given by :

$$\frac{\theta}{R} = \frac{K\ Kd\ Ka\ Kt\ Ke}{Je\ s^2 + Je\ b\ s + K\ Kd\ Ka\ Kt\ Ke} \tag{3.21}$$

The natural frequency of this second order system is given by :

$$\omega_n = \sqrt{\frac{K\ Kd\ Ka\ Kt\ Ke}{Je}} \tag{3.22}$$

The damping ratio of that system is given by :

$$\zeta = \frac{b}{2\,\omega_n} \tag{3.23}$$

The final parameter of interest is the steady state phase lag to a unit ramp input :

$$\phi_{ss} = \frac{b\ Je}{K\ Kd\ Ka\ Kt\ Ke} \tag{3.24}$$

One is thus left with three equations, but only two system parameters. In a normal servo system, one would pick a natural frequency and a damping ratio in order to calculate the filter gain K and the lag time constant 1/b.

In the case of the velocity spline, one actually needs to specify the phase lag and the damping ratio of the system. This leads to a filter gain of :

$$K = \frac{4\ Je}{Kd\ Ka\ Kt\ Ke}\left(\frac{\zeta}{\phi_{ss}}\right)^2 \tag{3.25}$$

The resulting inverse of the filter lag time constant is given by :

$$b = 2\zeta \sqrt{\frac{K\,Kd\,Ka\,Kt\,Ke}{Je}} \tag{3.26}$$

and the resulting natural frequency of the system is :

$$\omega_n = \frac{b}{2\zeta} \tag{3.27}$$

This parameter choice usually yields a stable and fast result. In the case of the system with a sampling time of 0.5 ms, a desired phase lag of 3 ms and a desired damping ratio of one, the resulting natural frequency of the system was 666 rad/sec.

3.7 Mismatched Gains

In some circumstances it is possible (but in the author's opinion not very probable), that the gains of different axes will be mismatched. In such an event the following error of the different axes will have different proportionality constants to the velocity. A steady state velocity path error will then result. Simply choosing the smallest proportionality constant as the common velocity error constant will trigger the state line and slow down the system. Such an approach however will still result in a path error which is proportional to the new, slower, velocity.

In order to run at the maximum speed with no path error it is necessary to feedforward the difference in errors between the fastest and the slower axes to the slower axes. In this manner all the axes have their own gain dependent velocity error constants, and the system can still run at the reference speed.

It is relatively easy to implement this concept for linear motions within the UBC controller. The axis with the largest gain still operates in the usual fashion. Any slower axis will have a larger velocity error constant. The actual steady state error will be:

$$e_{actual} = \phi_{ss\,actual}\, v \tag{3.28}$$

The faster axis requires a steady state error of:

$$e_{required} = \phi_{ss\ fastest\ axis}\ v \tag{3.29}$$

The error may be reduced by the difference of the above two equations:

$$compensation = v\left(\phi_{ss\ fastest\ axis} - \phi_{ss\ actual}\right) \tag{3.30}$$

In a linear move this is achieved by adding the compensation to the first master sample and subtracting it from the last. The basic scheme is illustrated in Figure 3.10:

The first and the last master sample position increments are updated in the following fashion:

$$\begin{aligned} X_1' &= X_1 + v\left(\phi_{ss\ fastest\ axis} - \phi_{ss\ actual}\right) \\ X_n' &= X_n - v\left(\phi_{ss\ fastest\ axis} - \phi_{ss\ actual}\right) \end{aligned} \tag{3.31}$$

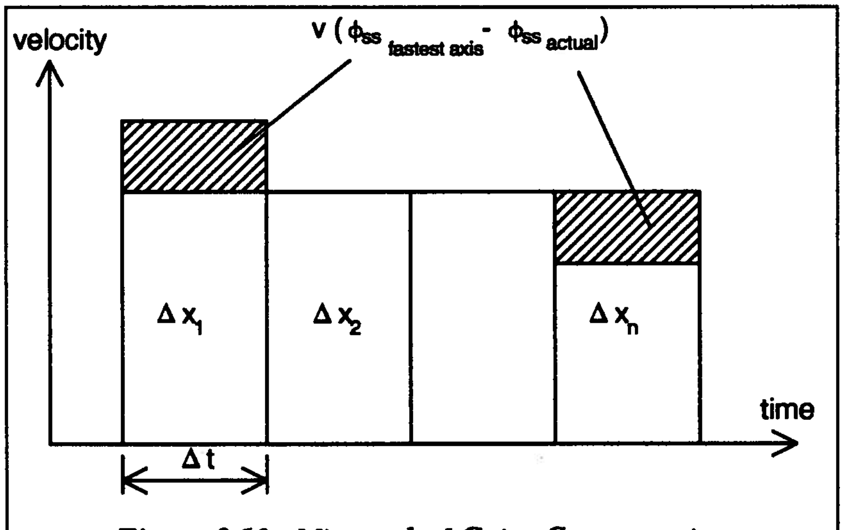


Figure 3.10. Mismatched Gains Compensation

3.8 Conclusions

In this chapter the UBC controller was described in detail. A new contouring algorithm was introduced. This algorithm smoothes out velocity discontinuities by controlling the amount of error of each axis. The procedure ensures path accuracy without extensive path preprocessing or fast computer hardware.

Simulations and experimental results from a real system are presented in the next chapter. They will show that the strategy utilizing a velocity spline in order to trigger the state line provides good results which are comparable to the ones obtained with other algorithms described in the literature.

Chapter 4

Simulation and Experimental Results

4.1 Introduction to Computer Simulations

The first phase of the work concerned with the evaluation of new control strategies, involved the creation of a realistic simulation model of the system. The program was developed using Matlab; it allows the examination of the path performance of an arbitrary number of axes and the introduction of nonlinear influences. A typical high speed servo system was also constructed and tested to validate the results from the simulated model. Results from the experimental system are given later in this chapter.

The actual MATLAB program is reproduced in its entirety in Appendix 3. As mentioned previously, the program allows the calculation of actual path errors as well as individual axis errors in an (*N*) axis system. The major non linearity in the basic system comprises the current limit of the amplifier, which is included in the simulation. The ordinary differential equations corresponding to the analog part of the loop were solved using a second order Runge-Kutta algorithm, while the time domain equations of the digital part (including the state line), were solved directly. The values of the various parameters in the model were chosen to correspond to the high performance experimental servo system, which is described later.

The majority of the simulations have been concerned with corner tracking capability. In order to simulate this, the path shown in Figure 4.1 has been used. The reader should realize that the path chosen requires one of the servos to change direction, while the second may, in essence continue at constant velocity. Most of the simulations of this rather difficult contour have used a nominal speed of 500 RPM (one motor will thus reverse to -500 RPM during the corner.)

4.1.1 Simple Servo System

The first set of simulations were conducted on a conventional system with no state line in order to provide a reference from which the degree of improvement could be gauged. Figure 4.1 shows the corner tracking ability of the system. It has large overshoots that obviously cannot be tolerated in practical contouring operations. It is important for the comparison with attempted improved strategies, which are described later, (with a nominal speed of 500 RPM) to point out that the whole traverse time required, according to the simulation is 0.225 seconds.

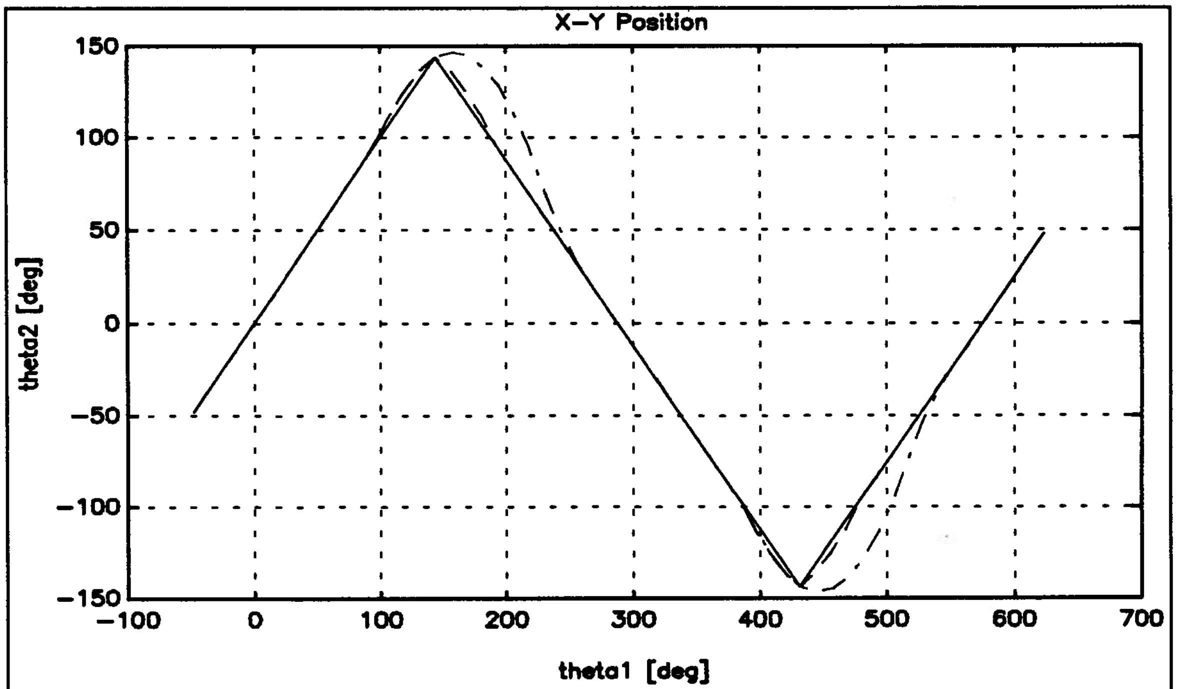


Figure 4.1. Corner Tracking With a Simple Servo Controller

Solid : Master Position Spline

Dashed : Reference Position

Dash-dot : Actual Position

4.1.2 State Line Strategies

There are many different approaches which may be used to trigger the state line. The most promising have been simulated.

4.1.2.1 Allowable Static Error

This is the simplest way of using the state line to control error. The state line in this strategy is triggered whenever the position error of a single axis exceeds a certain set constant limit. The error criterion is given by equation 1.1.

$$sign(v)\, e_{allowable} \;<\; \delta_{static} \tag{4.1}$$

Figure 4.2 shows the simulation of this method. The path accuracy is very good, but the system has been slowed down to a third of the original speed; this results from the fact that only a very small following error is allowed. (The complete simulation lasted 0.68 seconds.)

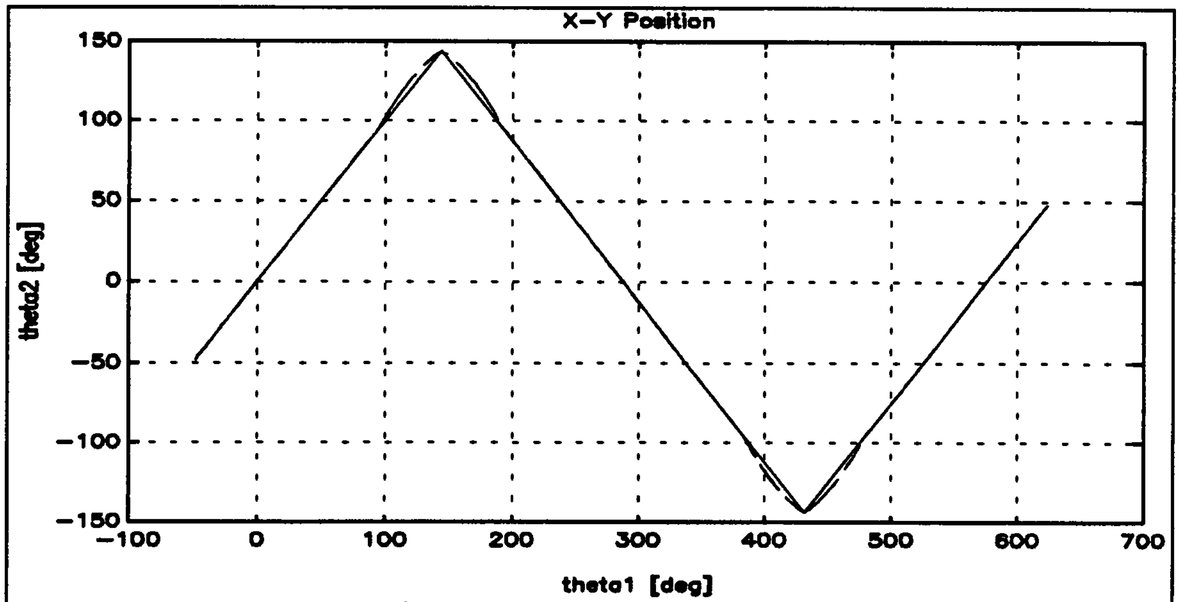


Figure 4.2. Corner Tracking with Static Error Criterion
Solid : Master Position Spline
Dashed : Reference Position
Dash-dot : Actual Position

4.1.2.2 The Use of a Velocity Spline

It was shown in section 2.2 that the expected following error is proportional to velocity. The fact that this kind of following error does not lead to a path error but rather to a phase lag, was used in section 2.4.1 to derive an error criterion that uses a velocity spline. The simulated response resulting from this approach is presented in Figure 4.3. It shows that this method yields excellent path accuracy while only slowing down the mean speed over the complete simulation by 20%, (the velocity spline approach took 0.285 seconds compared to 0.225 seconds for the simple servo.) The actuators were operating at the reference speed during most of the simulation; at the corners however the state line actively slowed down the system appreciably, allowing considerably reduced errors.

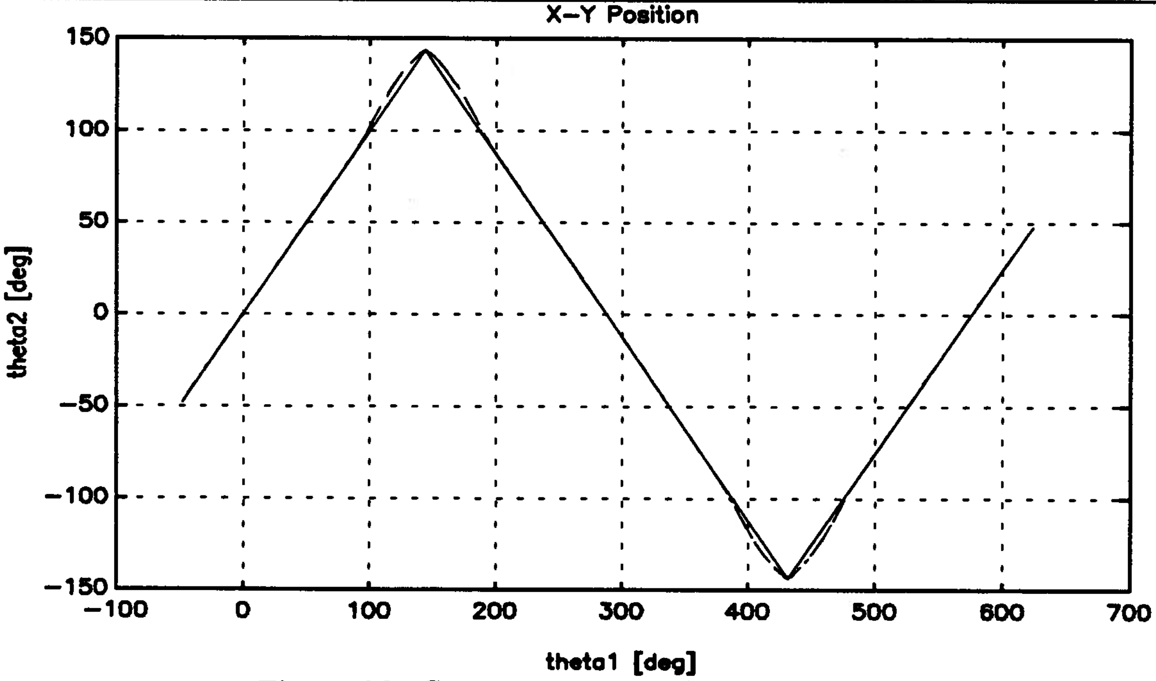


Figure 4.3. Corner Tracking with Velocity Spline
Solid : Master Position Spline
Dashed : Reference Position
Dash-dot : Actual Position

4.1.3 Velocity Feedforward

It may be advantageous to eliminate the phase lag that is inherent in the velocity spline, (this is the normal approach taken in the literature). One way of achieving this goal is to feedforward the expected velocity error. A simulation of this approach is shown in Figure 4.4. The system is now able to eliminate the phase lag, but corner tracking performance has deteriorated. When the axis with no discontinuity is slowed down at the corner, the velocity feedforward is no longer proportional to the actual velocity (0 at the corner itself.) Thus when the state line is actuated, the velocity feedforward for the second axis is larger than it should be, and a path error is encountered.

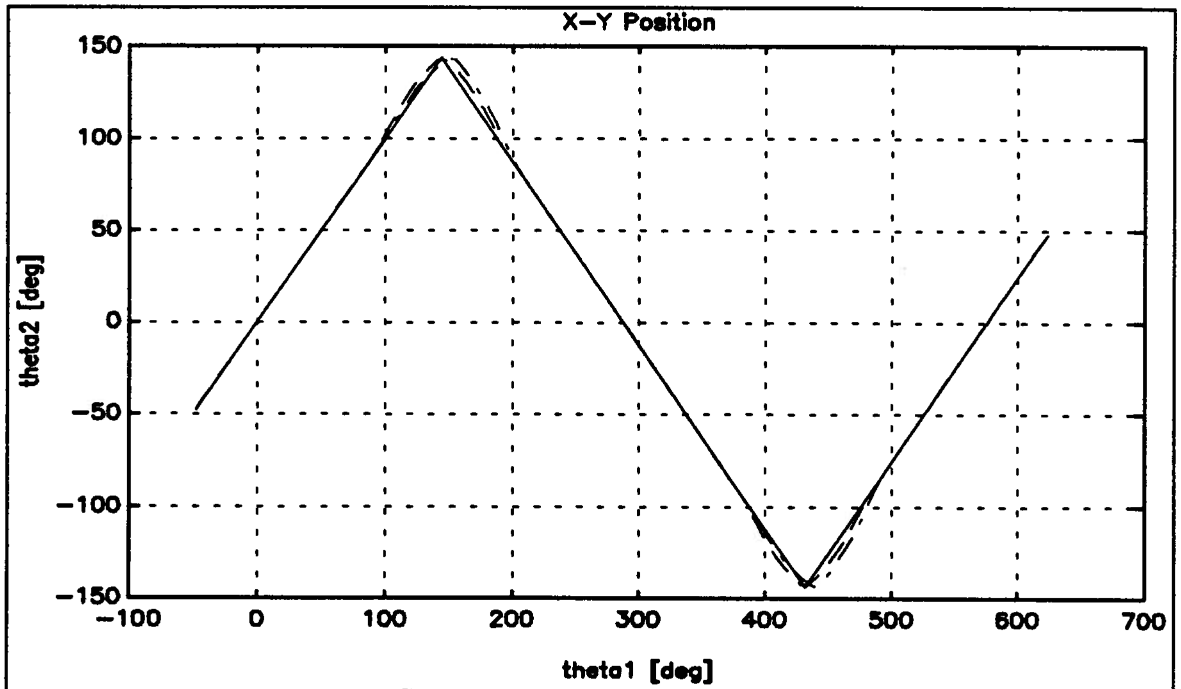


Figure 4.4. Corner Tracking with Velocity Feedforward
Solid : *Master Position Spline*
Dashed : *Reference Position*
Dash-dot : *Actual Position*

4.1.4 Constant Jerk Position Spline

It was shown in section 2.3 that the original position spline has velocity discontinuities. A new spline with constant jerk during a master sampling period was derived and tested. Figure 4.5 shows the tracking ability of this new spline. It shows no great improvement over the existing constant acceleration spline.

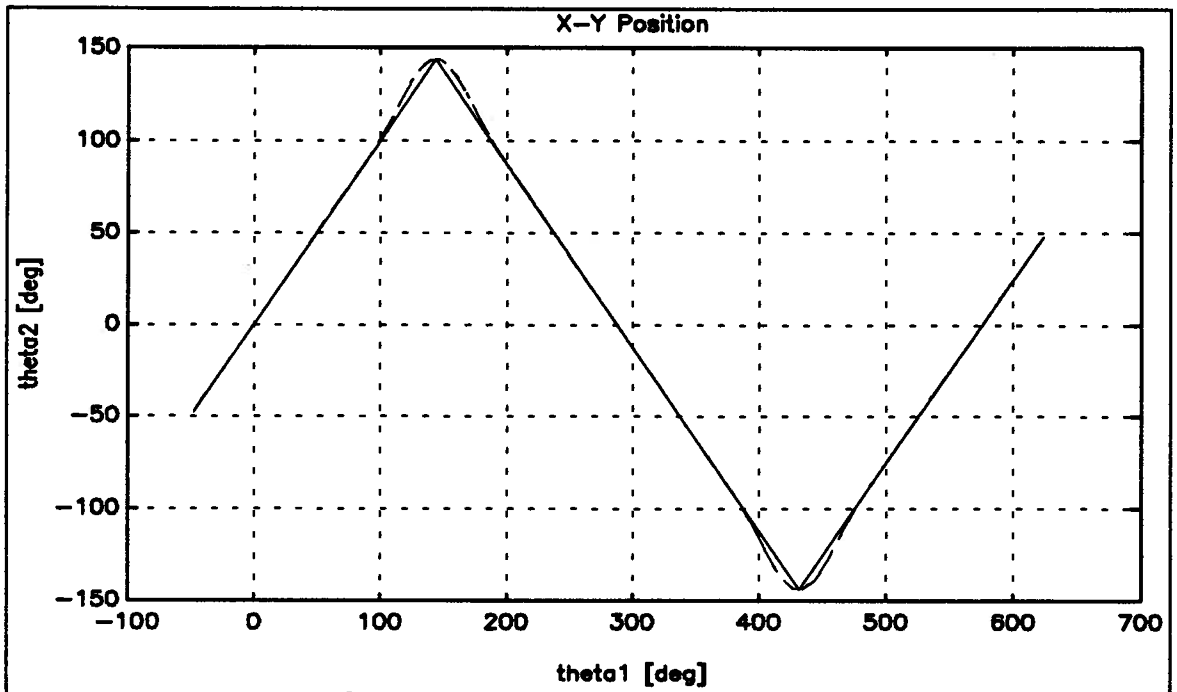


***Figure 4.5.** Corner Tracking with Constant Jerk Position Spline*

Solid :	*Master Position Spline*
Dashed :	*Reference Position*
Dash-dot :	*Actual Position*

4.1.5 Conclusions from Simulations

Figures 4.6 and 4.7 show enlargements of the simulated system performance with various strategies (500 RPM and 1000 RPM). It is seen that the best approach involves using the velocity spline in combination with the original position spline. A static error criterion is not feasible, because it does not allow the system to speed up to the reference velocity. The modified position spline does not improve the performance of the system over the previous simpler approach. Velocity feedforward is not feasible, because it introduces a path error when the state line is activated.

Table 4.1 shows the path error encountered and the time the different strategies take to contour the required path at 500 RPM:

Table 4.1. Execution Time and Path Error of Simulated Strategies

Strategy	Time [sec]	Path Error [BLU]
Simple Servo	0.2250	260
Static Error Criterion	0.6800	007
Velocity Spline	0.2850	005
Velocity Feedforward	0.2875	045
Constant Jerk Position Spline	0.2795	005

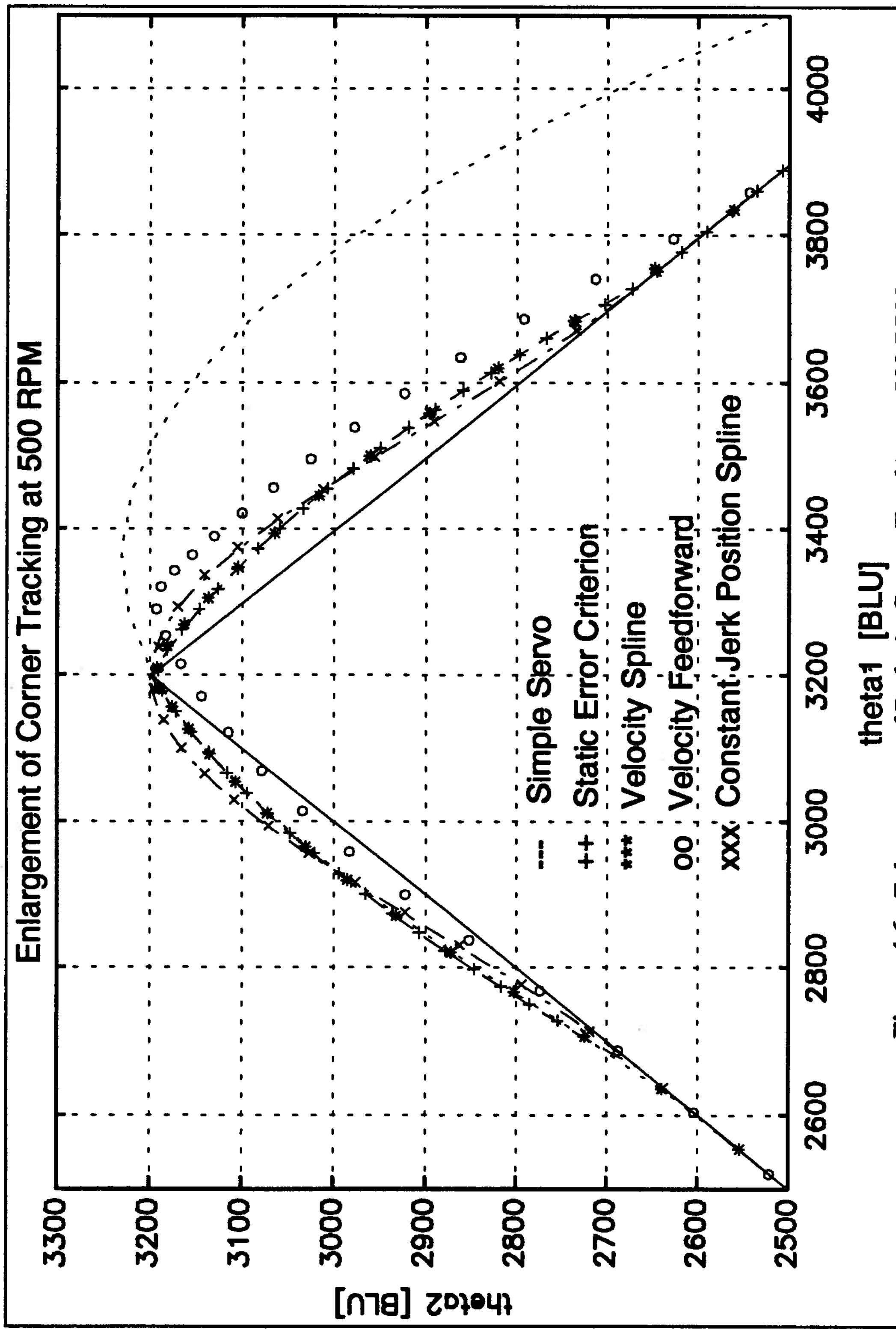


Figure 4.6. *Enlargements of Paths in Corner Tracking at 500 RPM*
Solid: *Master Spline*
Dashed : *Constant Acceleration position Spline*
Dash-Dot : *Constant Jerk Position Spline*

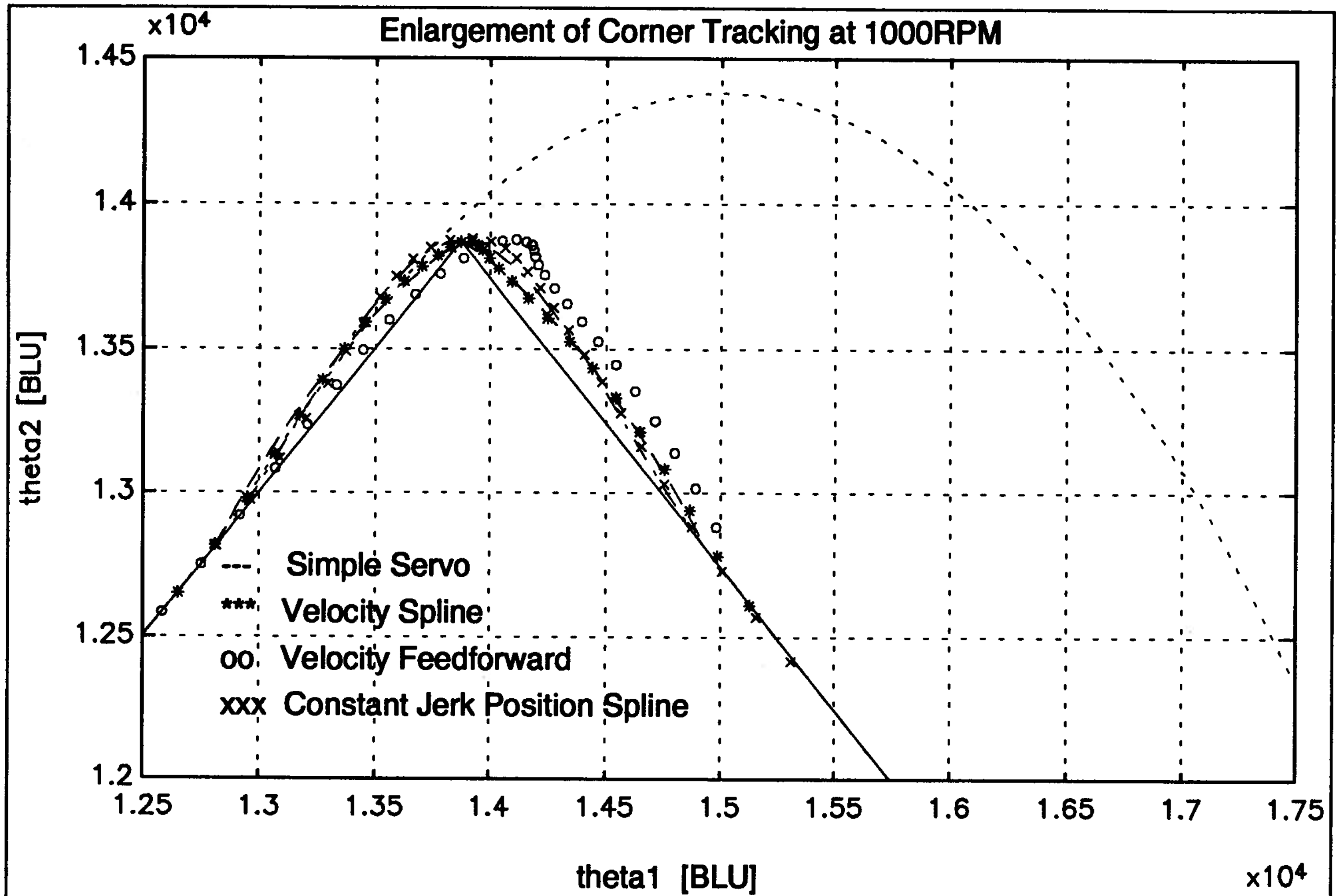


Figure 4.7. Enlargements of Paths in Corner Tracking at 1000 RPM

Solid:	*Master Spline*
Dashed :	*Constant Acceleration position Spline*
Dash-Dot :	*Constant Jerk Position Spline*

4.2 Experimental Results

After the velocity spline had been chosen as the most efficient method to trigger the state line, actuators, amplifiers, and controllers from a retrofitted GMF s-108 robot were used to validate the simulated results.

4.2.1 Experimental Apparatus

The experimental apparatus consists of a controller, amplifiers, and actuators. The actuators are equipped with tachometers and encoders. The tachometer signals are fed back to the amplifiers, and the encoder signals are utilized by the control computer.

Figure 4.8 shows a schematic diagram of the drive system.

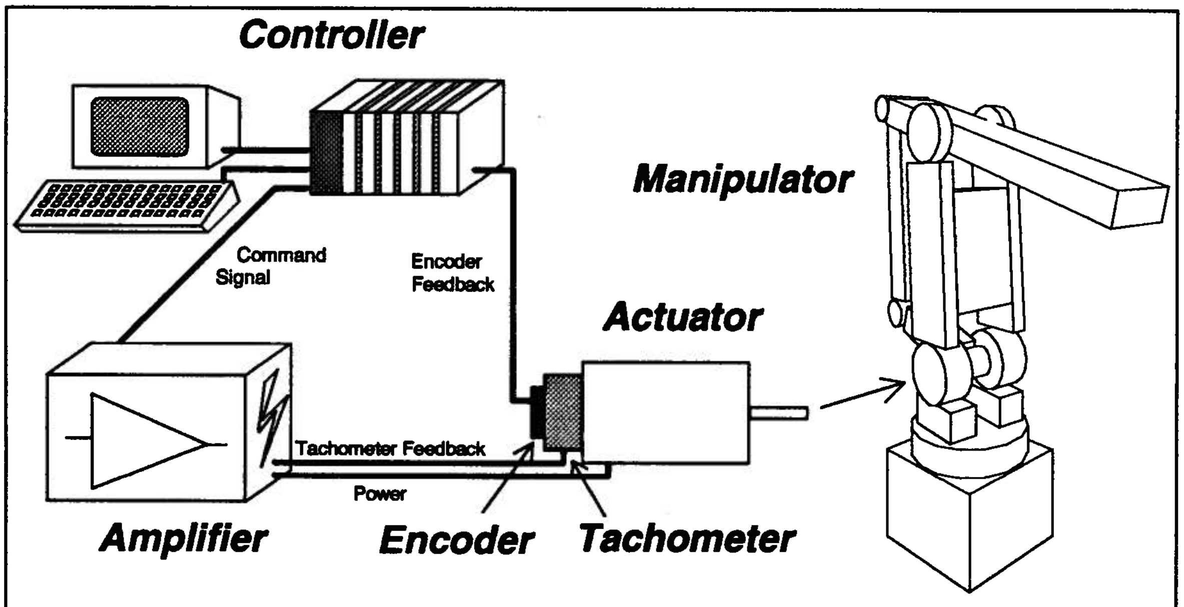


Figure 4.8. Schematic Diagram of Drive System

Figure 4.9 shows a schematic block diagram of a single axis of the system.

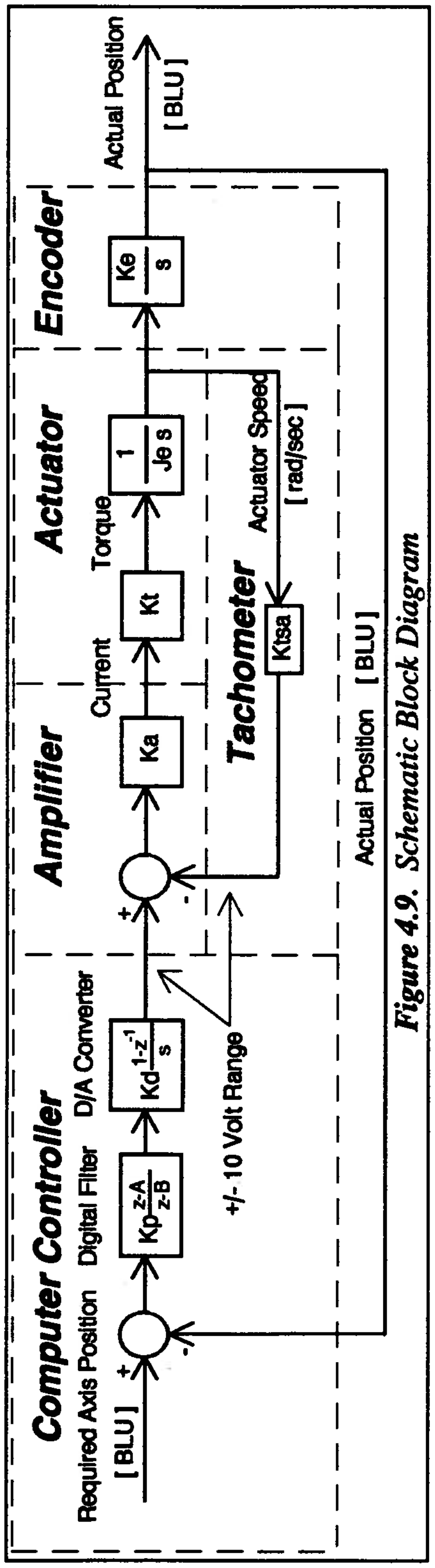


Figure 4.9. Schematic Block Diagram

4.2.1.1 Actuators

The actuators are DC motors which are supplied by Inertial Motors. They are equipped with tachometers for measuring actuator velocity and encoders for measuring actual position. Table 4.2 lists the motor parameters:

Table 4.2. Motor Parameters

Name	Symbol	Value	Units
Motor Torque Constant	Kt	0.2967	Nm/A
Motor Inertia	Je	0.9636e-3	Nm s^2
Tachometer Feedback	Ktsa	0.047	V/(rad/sec)
Encoder Gain	Ke	636.6198	BLU/rad
Maximum Current	I_{max}	10	A
Maximum Speed	ω_{max}	2000	RPM

4.2.1.2 Amplifiers

The amplifiers were built by Glentek specifically for the Inertial motors. They are PWM amplifiers with a bandwidth of more than 750 Hz. Table 4.3 shows all pertinent parameters of the amplifiers:

Table 4.3. Amplifier parameters

Name	Symbol	Value	Units
Amplifier Gain	Ka	13.6136	A/V
Maximum Current	I_{max}	10	A
Bandwidth	$fc_{amplifier}$	>750	Hz

4.2.1.3 Controller

The control algorithm was implemented on a Ziatech STD32-bus industrial computer. The master is a V53 and the slaves are 80c196 based STD32 boards from Universal Systems.

The controller parameters are listed in Table 4.4 :

Table 4.4. Table of Model Constants

Name	Symbol	Value	Units
D/A Converter Gain	Kd	0.0049	
Filter Gain	Kp	34.1333	
Filter Lead Parameter	A	0.9	
Filter Lag Parameter	B	-0.333	
Master Sampling Period	Δt	16	m sec
Slave Sampling Period	ΔT	0.5	ms
Position Loop Damping Ratio	ζ	1	
Position Loop Steady State Ramp Phase Lag	ϕ_{ss}	3	ms
Position Loop Natural Frequency	ω_n	666.67	rad / sec
Position Loop Bandwidth	f_b	67	Hz

The filter parameters were chosen to yield a damping ratio of unity and a steady state ramp phase lag of 3 msec. Obviously the theoretical bandwidth of 67 Hz cannot be achieved, unless sinusoidal reference commands with very small amplitudes are utilized.

The Bode plots of the position loop are plotted in Figure 4.10 and the step response of the experimental system is shown Figure 4.11.

The slave sampling period was chosen to be 0.5 ms to ensure that the slave board itself and the communication protocol between the master and the slaves was always reliable. At the same time this sampling period is small enough to ensure that the continuous analysis presented in this work remains valid.

The master sampling period was chosen to provide a compromise between the possible acceleration of the system and the bandwidth of the position spline. According to equation 2.19, a master sampling period of 16 ms ensures that the system is able to reverse smoothly at 1000 RPM. Larger sampling periods are not advisable, because they would decrease the reference bandwidth below an acceptable level. Increasing the current limit to 20 A or averaging the velocity spline over 4 instead of 2 master samples would make a reversal at 2000 RPM possible.

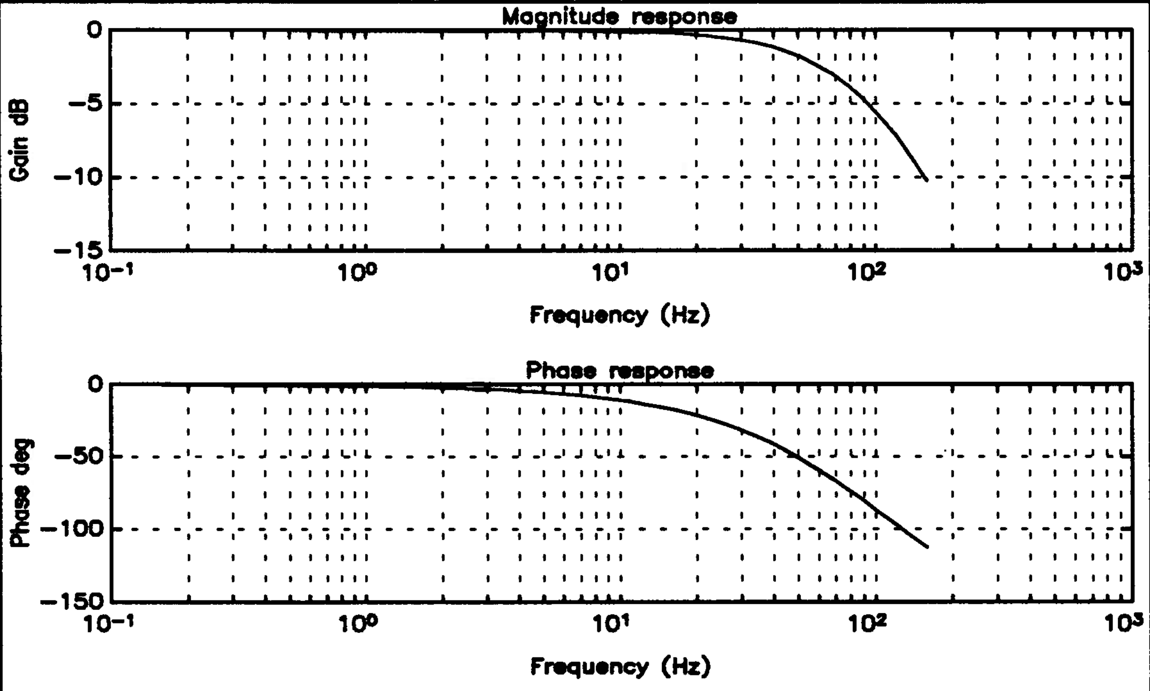


Figure 4.10. Bode Plots of Experimental System

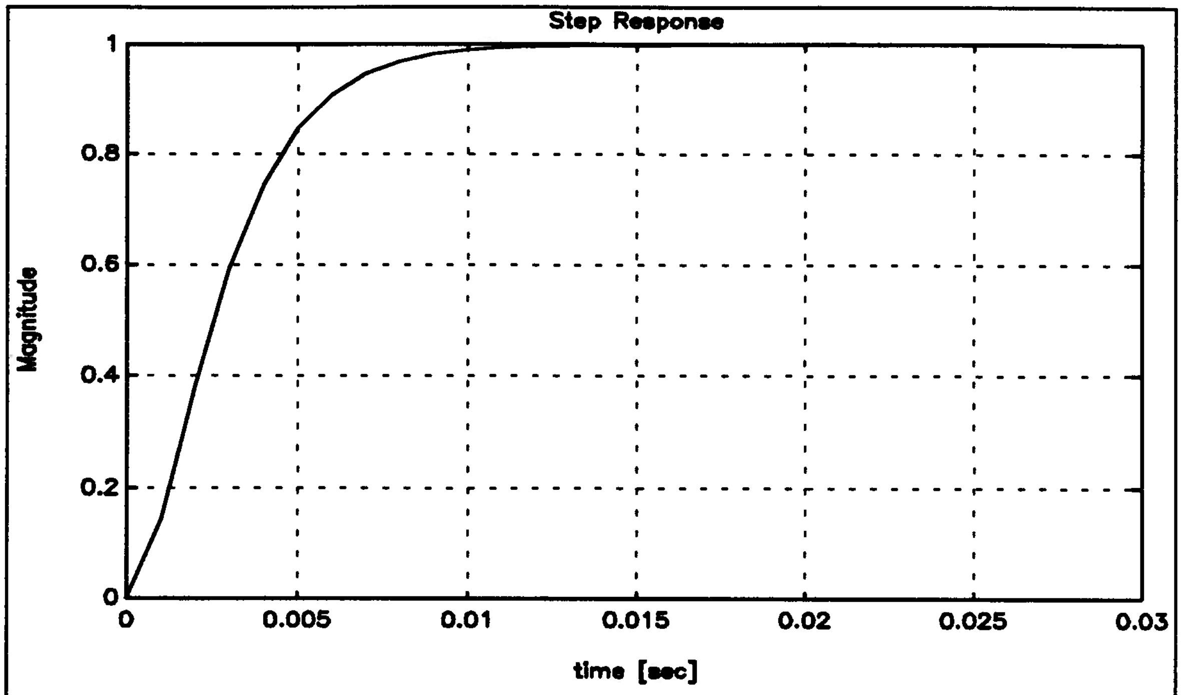


Figure 4.11. Step Response of Experimental System

4.2.1.4 Data Recording Device

It is necessary to have an accurate data recording scheme in order to achieve reliable test results. In the experiments that follow, a forth program was run on the master. This program continuously reads the values of position and error from the slaves and writes them to an array. The program is initiated after all the position increments for an experiment have been transferred to the slaves. At this point the slaves are slowly emptying their buffers, and the recording program simply writes current positions and errors to the array. As soon as the array is filled, the program stops. At a later stage the array is transferred to a file and imported into Matlab for post processing. This procedure is extremely simple, but unfortunately it cannot capture the beginning of an experiment. It also does not provide an accurate measure of time, because data is simply read as quickly as possible from the slaves. The data recording program and programs to create the input required for the experiments themselves are listed in Appendix 4.

4.2.2 Circular Interpolation

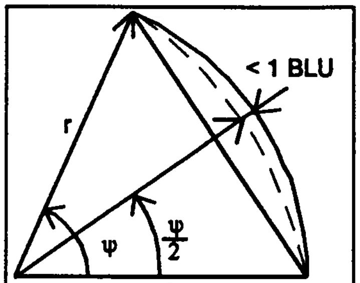


Figure 4.12. Circular Interpolation

During machining operations, the contouring of circles is the second most common geometry, (after contouring of straight lines.) It is therefore important to determine the limits of the system in circular interpolation. In addition, circular interpolation can give some insight into the frequency response of the system. In this section relationships will be derived which specifically define the limits that the simple servo system and the modified system with a velocity spline can follow. It will be shown that a frequency response in the usual sense is difficult to derive in those cases where the simple servo system would show a deteriorated performance due to the imposed saturation limit. The state line system, when subjected to the same conditions, will decrease the reference frequency to acceptable levels, hence avoiding large errors.

In order to achieve good path accuracy, it is necessary to meet the constraint set by the position spline. A single axis is required to have a maximum path error of less than one BLU. (One BLU is the resolution of the encoder. One revolution of the actuator corresponds to 4000 BLUs.) According to Pottier [21] this requires :

$$\psi = \sqrt[4]{\frac{42.6667}{r}} \tag{4.2}$$

This can be rewritten in the form :

$$2\pi\,\Delta t\, f_{\max} = \sqrt[4]{\frac{42.6667}{r_{\max}}} \tag{4.3}$$

Thus relating the maximum radius of the circle to the maximum allowable frequency.

The maximum achievable velocity puts a second constraint on the system. During circular interpolation the actuator position is described by :

$$\theta = r \sin(2\pi f t) \tag{4.4}$$

The corresponding velocity is :

$$\omega = 2\pi f r \cos(2\pi f t) \tag{4.5}$$

The maximum velocity takes place when the cosine term equals one (or -1). This leads to the final form of the second constraint:

$$\omega_{max} = 2\pi f_{max} r_{max} \tag{4.6}$$

The third constraint is current saturation of the amplifier, which limits the possible acceleration. The acceleration corresponding to the reference signal is:

$$\alpha = -(2\pi f)^2 r \sin(2\pi f t) \tag{4.7}$$

The maximum acceleration occurs when the sine term equals one (or -1). This leaves a third constraint that correlates the maximum radius to the maximum frequency:

$$\alpha_{max} = I_{max} Kt \frac{Ke}{Je} = (2\pi f_{max})^2 r_{max} \tag{4.8}$$

The maximum velocity constraint is only active at low frequencies and large radii. The problem then is to determine which of the two other constraints is dominant. For the system under consideration with a slave sampling time of 0.5 ms and a master sampling time of 16 ms, Figure 4.13 shows a plot of frequency versus circle radius.

The two constraints meet at r_{crit} = 5900 BLU and f_{crit} = 2.9 Hz. At radii less than r_{crit} the position spline imposes the dominant constraint. At radii higher than the critical radius, current saturation is dominant.

If one attempts to use reference commands along the position spline constraint for radii larger than r_{crit}, the state line should pull the actual response down to the limiting acceleration constraint line. This means that it would decrease the actual output frequency. The shape of the sine curve would of course change, because the state line would only try to alter the portions that exceed the limiting acceleration. In the extreme case the final output would have a constant curvature that changes sign every half cycle.

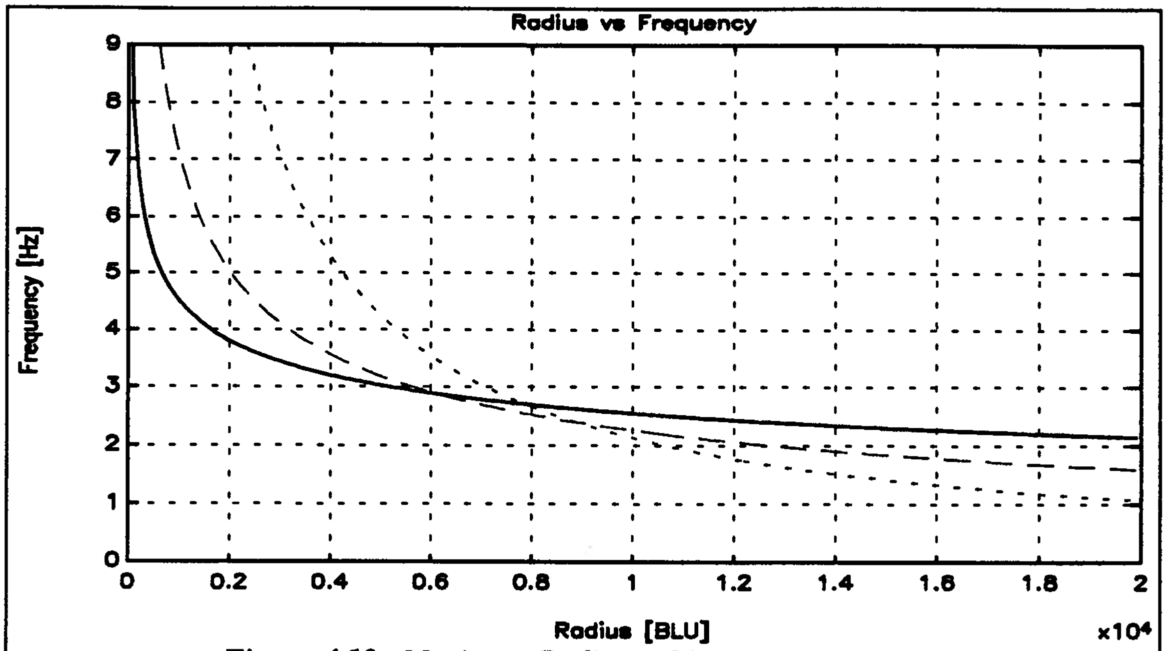


Figure 4.13. Maximum Radius vs Maximum Frequency
Dotted Line : Limiting Velocity Constraint
Dashed Line : Limiting Acceleration Constraint
Solid Line : Position Spline Constraint

This resulting non sinusoidal response complicates a true frequency analysis, because the error is superimposed on the non sinusoidal carrier signal. The two responses are difficult to separate, because one cannot easily determine the frequency components of the carrier signal.

4.2.2.1 Low Feedrate Experiment

In this experiment ten circles were contoured and the third is shown in the Figures below. A feedrate was chosen that would not saturate the system. The state line is not activated and the system behaves as a normal servo system. Table 4.5 gives pertinent parameters :

Table 4.5. Parameters for Low Feedrate Circular Interpolation

Name	**Value**	**Units**
Frequency	3	Hz
Radius	5000	BLU
Feedrate	94248	BLU/sec

Figure 4.14 shows the actual path and Figure 4.15 the path error during the experiment.

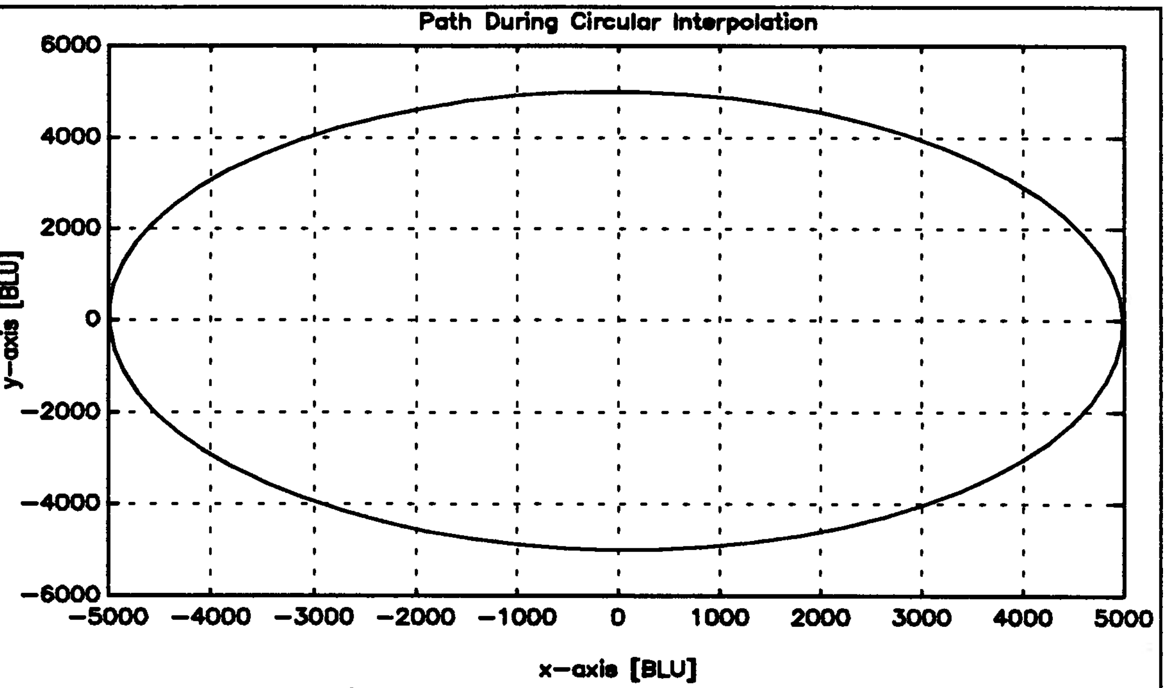


Figure 4.14. Path of Low Feedrate Circular Interpolation Experiment

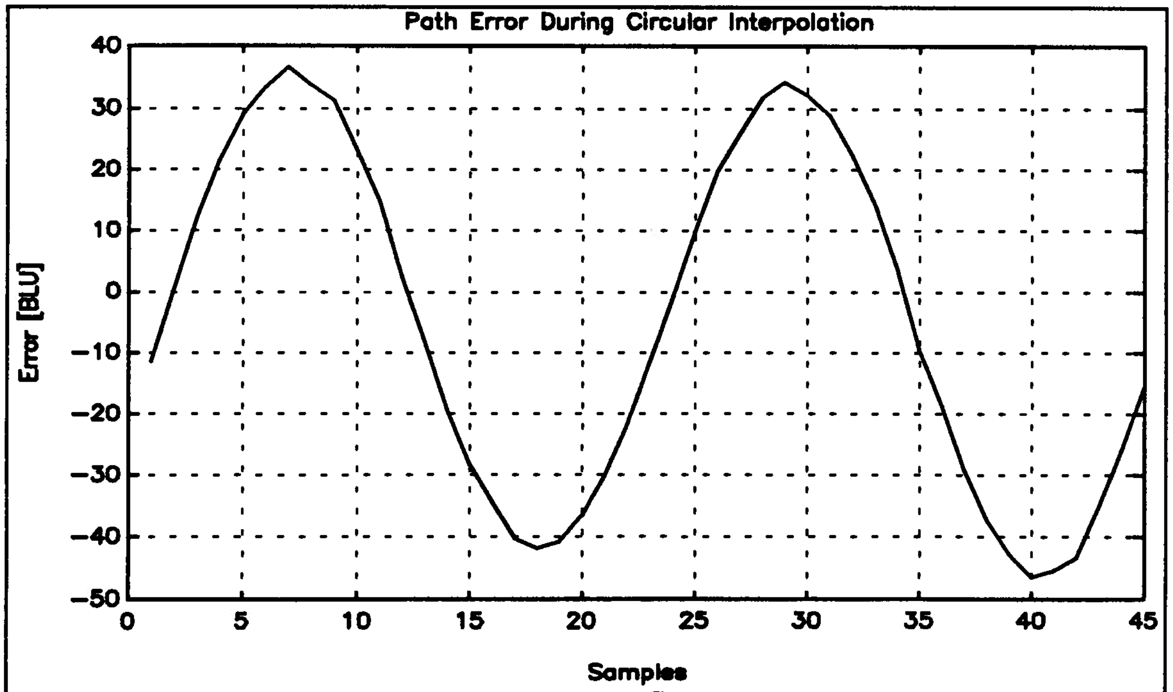


Figure 4.15. Path Error of Low Feedrate Circular Interpolation Experiment

Clearly the system is able to follow the contour well. No state line is needed, because the saturation limit is not reached. The maximum error is approximately 45 BLUs.

4.2.2.2 High Feedrate Experiment

In this experiment the feedrate was chosen such that current saturation and the maximum required speed would make it impossible for the simple servo to follow the path. Table 4.6 shows the parameters for this experiment :

Table 4.6. Parameters for High Feedrate Circular Interpolation

Name	Value	Units
Frequency	3	Hz
Radius	8000	BLU
Feedrate	150796	BLU/sec

Figure 4.16 shows the actual path and Figure 4.17 the corresponding path error during the high feedrate experiment using a simple servo with no velocity spline :

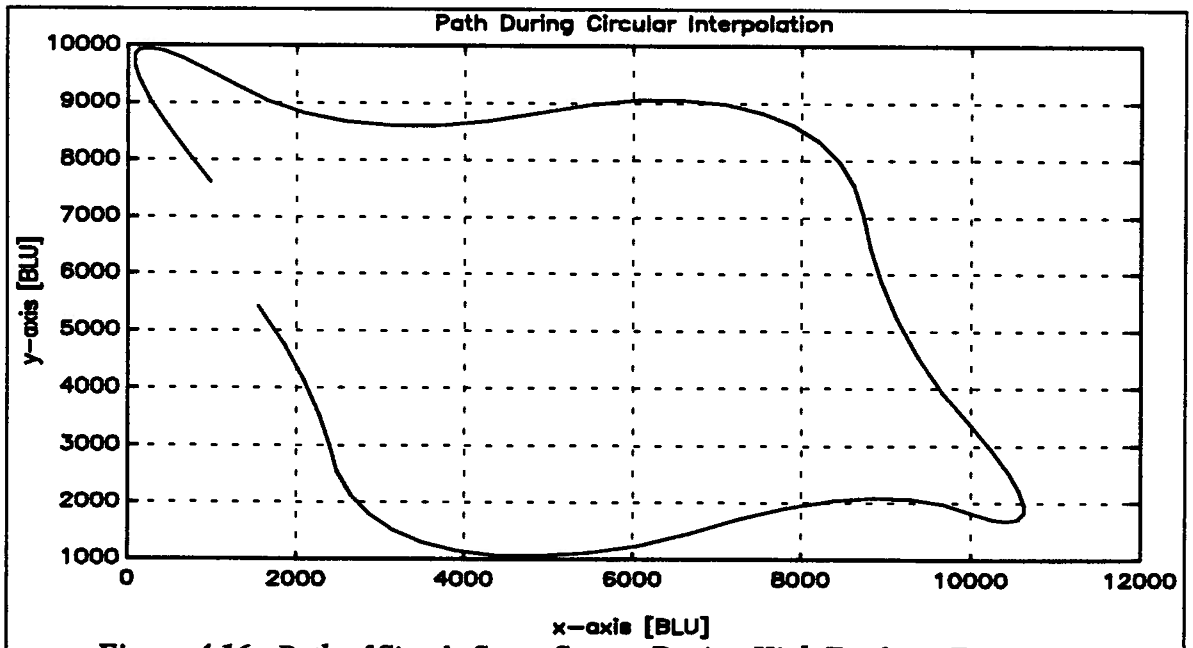


Figure 4.16. Path of Simple Servo System During High Feedrate Experiment

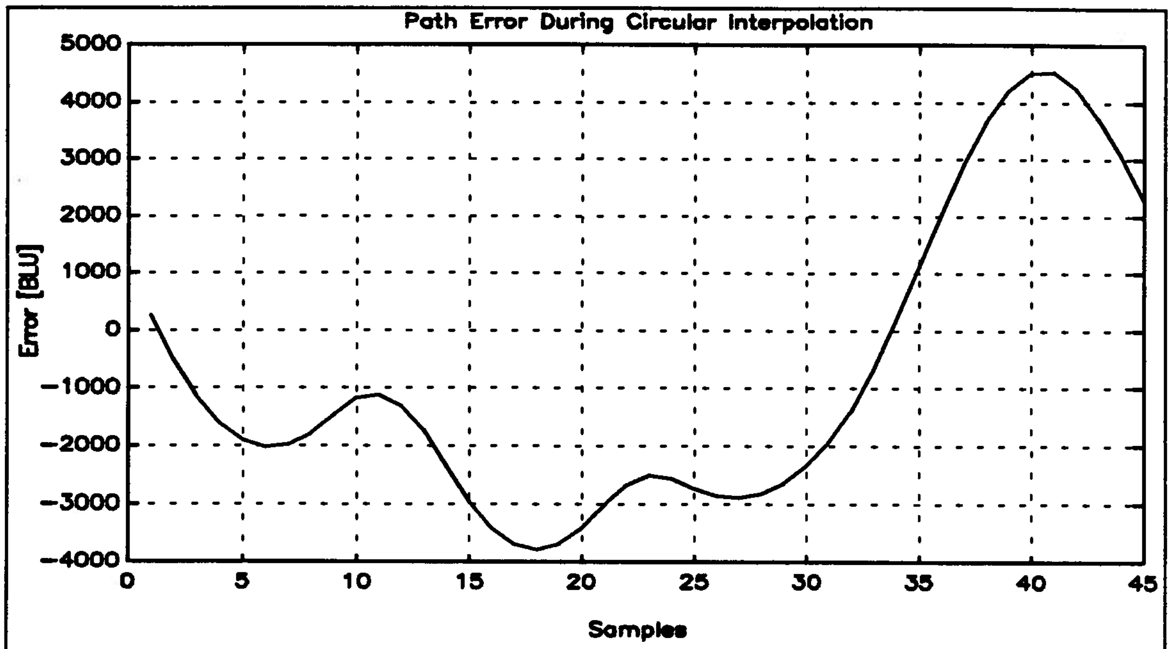


Figure 4.17. Path Error of Simple Servo System During High Feedrate Experiment

The system is not able to follow the reference path, because acceleration levels have driven the amplifiers into saturation, and the required velocity exceeds the maximum velocity of the actuators..

Figure 4.18 shows the path and Figure 4.19 the path error of a high feedrate experiment with the state line being activated using the velocity spline error criterion.

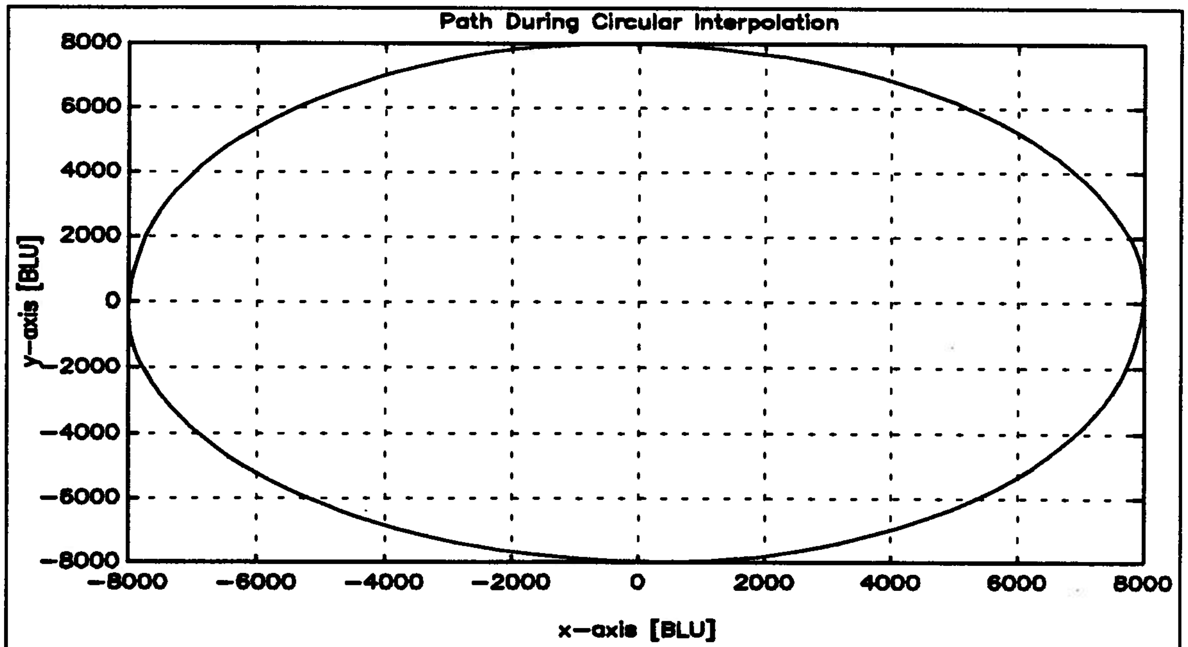


Figure 4.18. Path of Servo System with Active State Line During High Feedrate Experiment

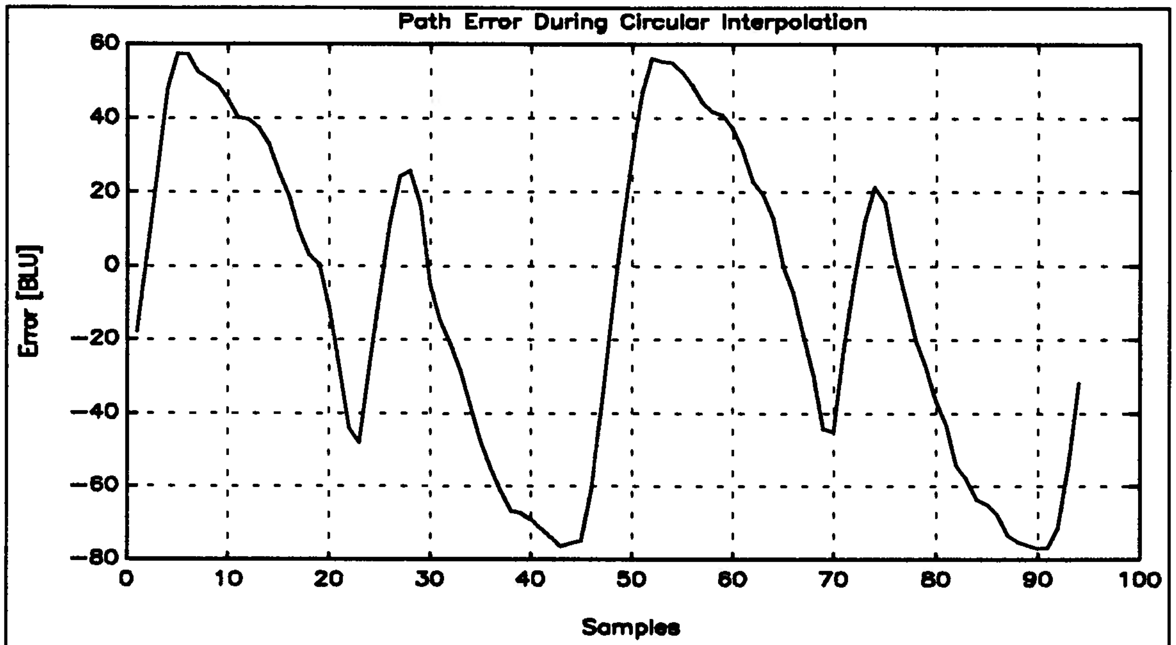


Figure 4.19. Path Error of Servo System with Active State Line During High Feedrate Experiment

Clearly the velocity spline has slowed down the system at the places of high acceleration to an acceptable level. The path error is approximately 78 BLUs. This is good considering that the maximum velocity constraint and the saturation constraint from Figure 4.13 were exceeded.

4.2.3 Mismatched Gains

An experiment was arranged in which the second axis had half the gain of the first axis. Two straight line contouring experiments were performed with this system. In the first experiment no compensation was utilized, and in the second the algorithm described in section 2.7 was implemented to compensate for the mismatched gains. Figure 4.20 shows the reference path, and the actual path of both experiments on the same plot. Figure 4.21 shows the error for both experiments. In the first experiment with no compensation, a steady state path error of approximately 75 BLU was encountered. In the second experiment the compensation scheme yielded a steady state error of approximately 5 BLU. This shows that the scheme described in section 2.7 works well for compensating mismatched gains.

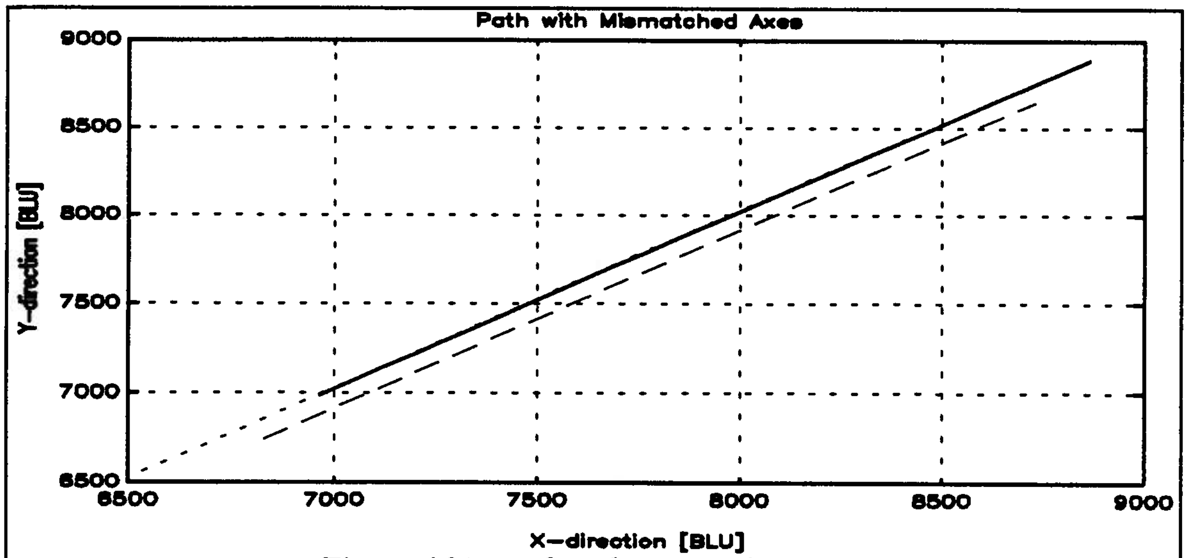


Figure 4.20. Path with Mismatched Gains
Solid Line : Reference Path
Dashed Line : Simple Servo Path
Dotted Line : Compensated Path

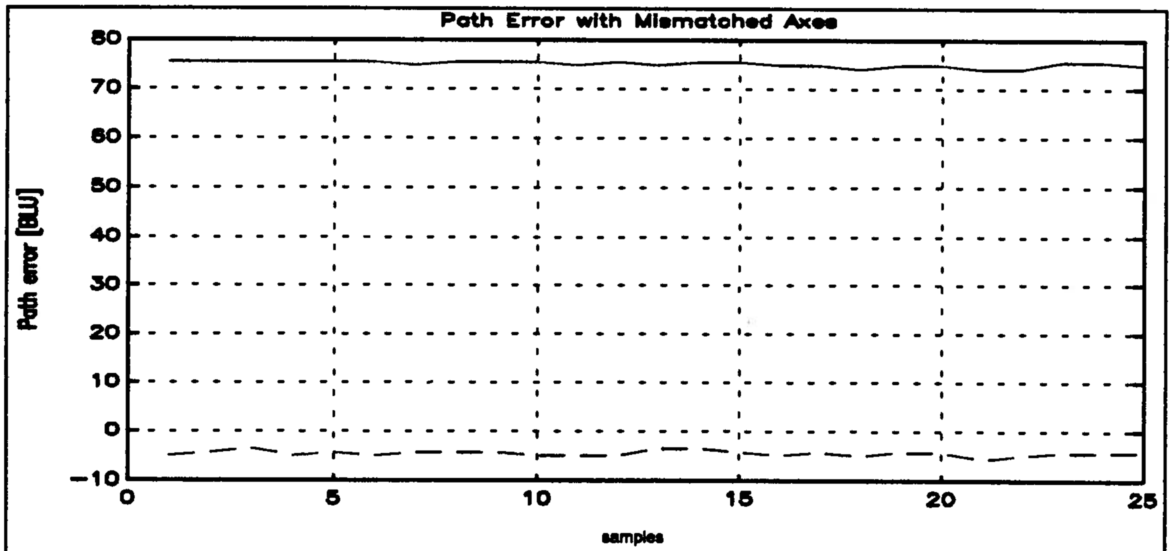


Figure 4.21. Path Error with Mismatched Gains
Solid Line : Simple Servo Error
Dashed Line : Compensated Error

4.2.4 Disturbance Torques

The controller strategies described in the previous sections should be able to compensate for disturbances that might result from loads on the actuator or cutting torques. It will simply slow down the system until the error criterion imposed on the system in order to achieve the necessary path accuracy is met.

In order to check the performance of the system with severe disturbances, it was decided to abruptly stop the second axis during a linear move. This should then make the first axis stop at the same instant. The experiment was performed at 65 RPM for each axis respectively. The requested distance was 4000 BLU and the y-axis was stopped at approximately 2200 BLU.

For the system with no state line acting, the required and the actual path of the system in response to the described disturbance are shown in Figure 4.22. The path error of this system is shown in Figure 4.23.

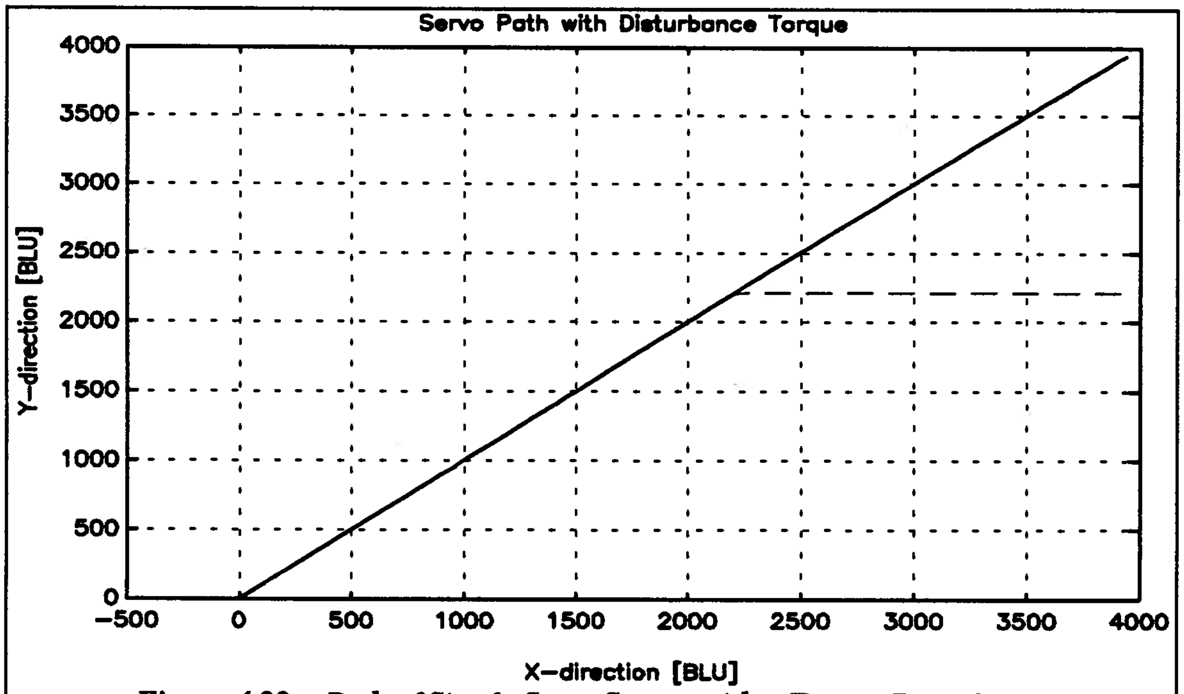


Figure 4.22 : Path of Simple Servo System with a Torque Disturbance
Solid : Reference position
Dashed : Actual Position

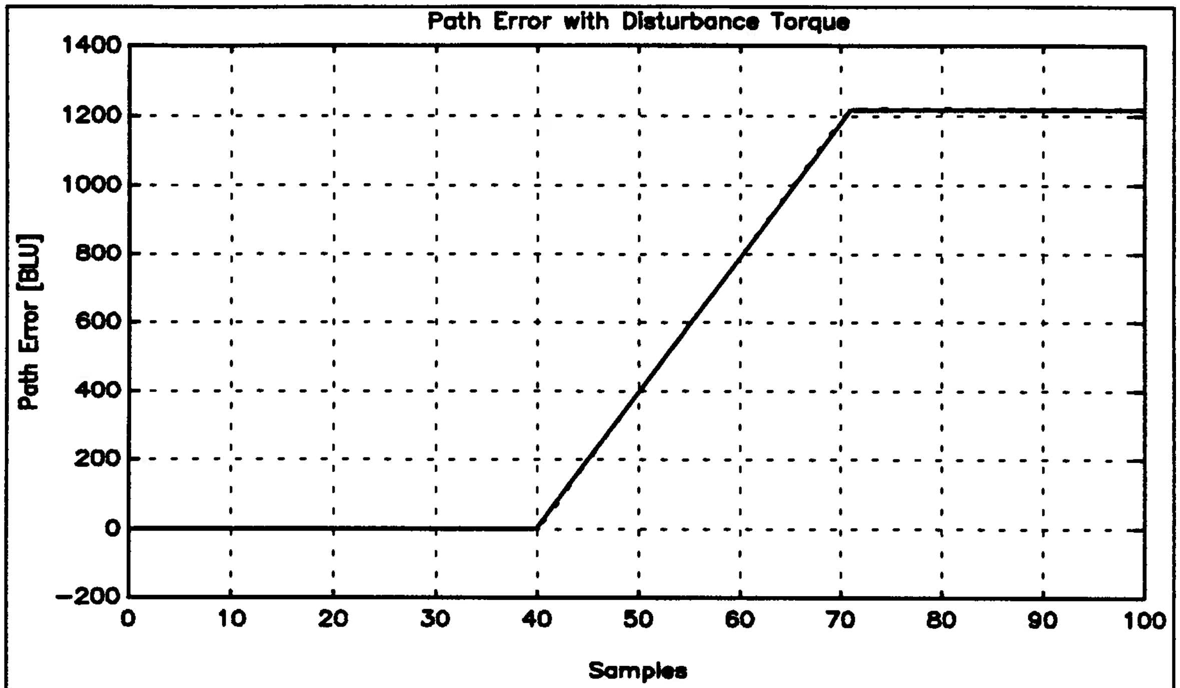


Figure 4.23. Path Error of Simple Servo System with a Torque Disturbance

Figures 4.22 and 4.23 show that when the second axis is stopped, the first axis (of course) continues to operate normally. This introduces a large path error.

Figure 4.24 shows the path in a similar experiment with the state line active. The state line stops the system as soon as it detects the large torque applied to the second axis. (this is done through monitoring error rather than torque itself.) Figure 4.25 shows the path error in this experiment. It is reduced to 15 BLUs.

This simple experiment would lead one to believe that the system is able to compensate for disturbance torques, even in extreme cases.

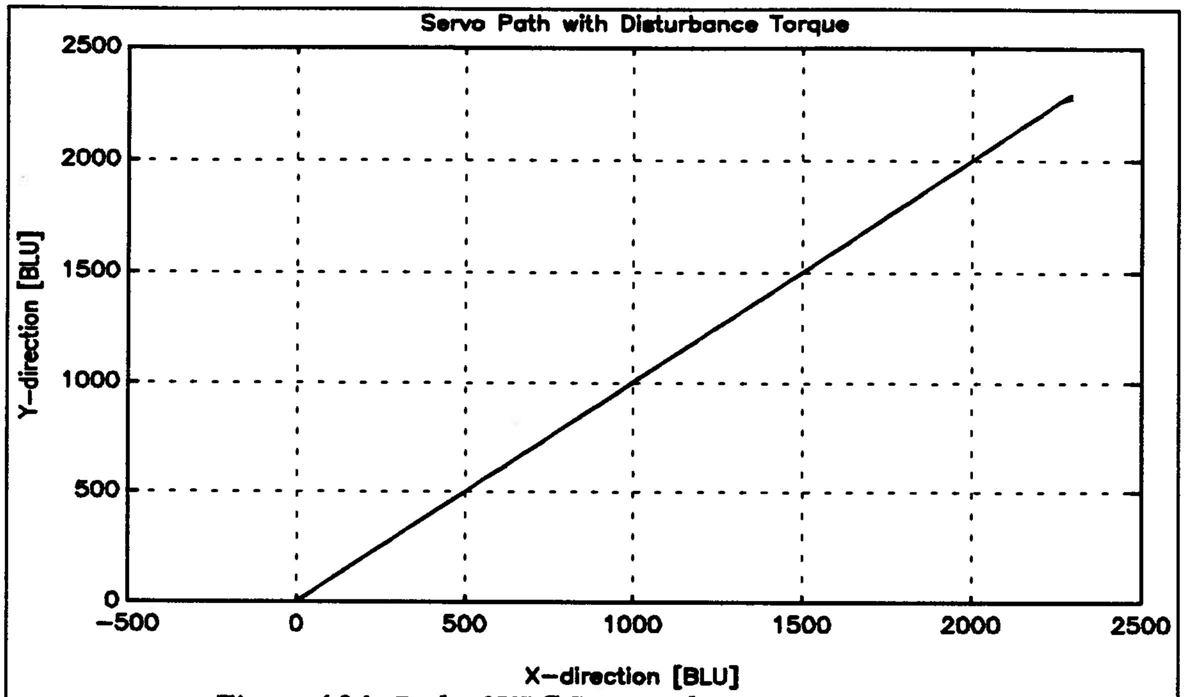


Figure 4.24. Path of UBC Servo with Torque Disturbance

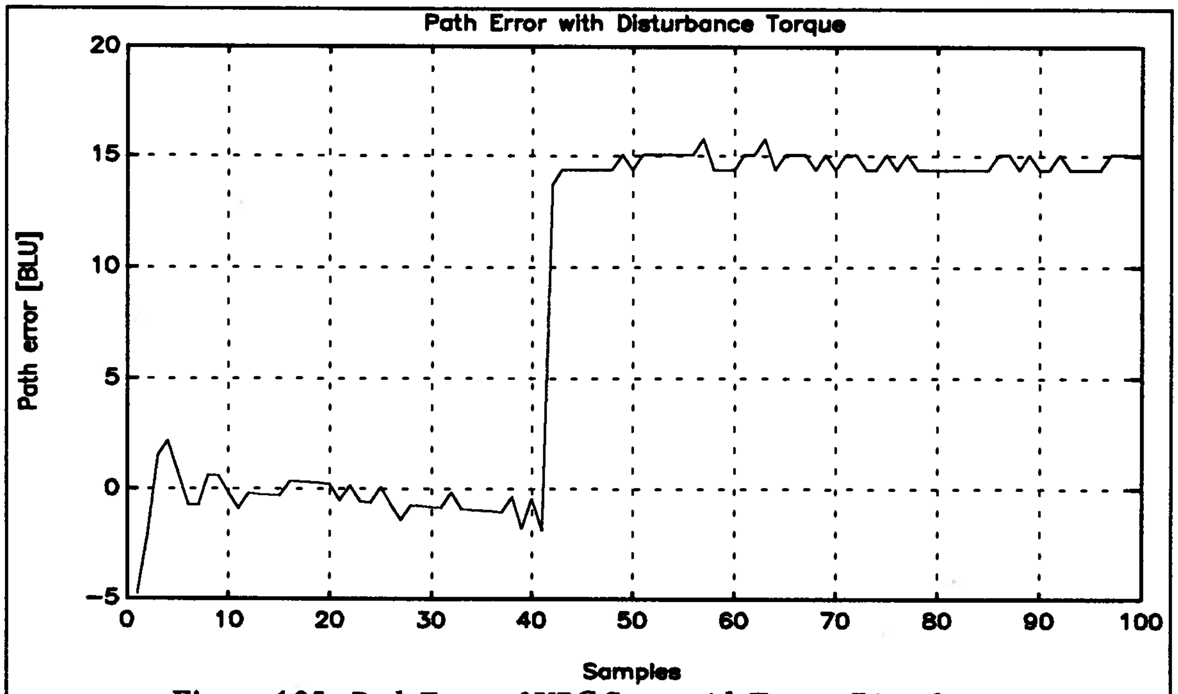


Figure 4.25. Path Error of UBC Servo with Torque Disturbance

4.2.5 Corner Tracking

In terms of robotic and high speed machining operations, the most important test criterion is corner tracking ability. This task requires the controller to compensate for large motor inertia torques that can lead to saturation of the amplifiers. Usually one attempts to avoid saturation by choosing a sufficiently small speed around a corner. The UBC controller on the other hand will recognize a large acceleration and slow down the system automatically, in order to allow the cornering to be achieved at maximum possible deceleration and acceleration levels.

The experiment was arranged as described in the previous section so that the actuator of the first axis would travel at constant speed while the second actuator would reverse at the same speed a number of times during the experiment. The data shown depict one such cycle in the middle of the experiment.

4.2.5.1 Reversal at 500 RPM

Figure 4.26 shows a plot of the first axis versus the second axis of a simple servo system. Figure 4.27 shows the result of the same experiment with an active state line. Clearly there is a vast improvement in path accuracy.

This experiment also allows the validation of the results obtained from the simulations. Clearly Figures 4.1 and 4.26, and Figures 4.3 and 4.27 show similar responses.

Figure 4.29 shows a magnified view of the performance of the simple servo in the first corner. Figure 4.28 shows an enlargement of the first corner of the performance of the system with an active state line. The simple servo has a path error of approximately 26° while the system with a state line shows less than 2° of path error. The "non smoothness" of the reference path is a result of the data acquisition program. It calculates the reference path by adding the actual position to the error. Data acquisition is relatively slow and sometimes the error and the actual position may be a sample out of phase.

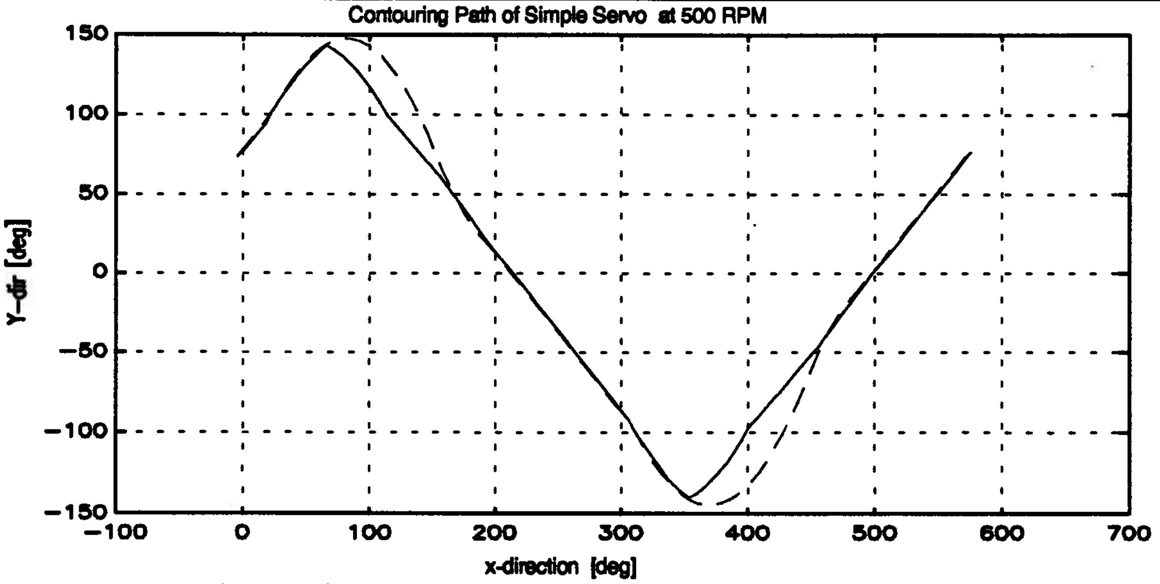


Figure 4.26. Corner Tracking of Simple Servo at 500 RPM
Solid : Reference Position
Dashed : Actual Position

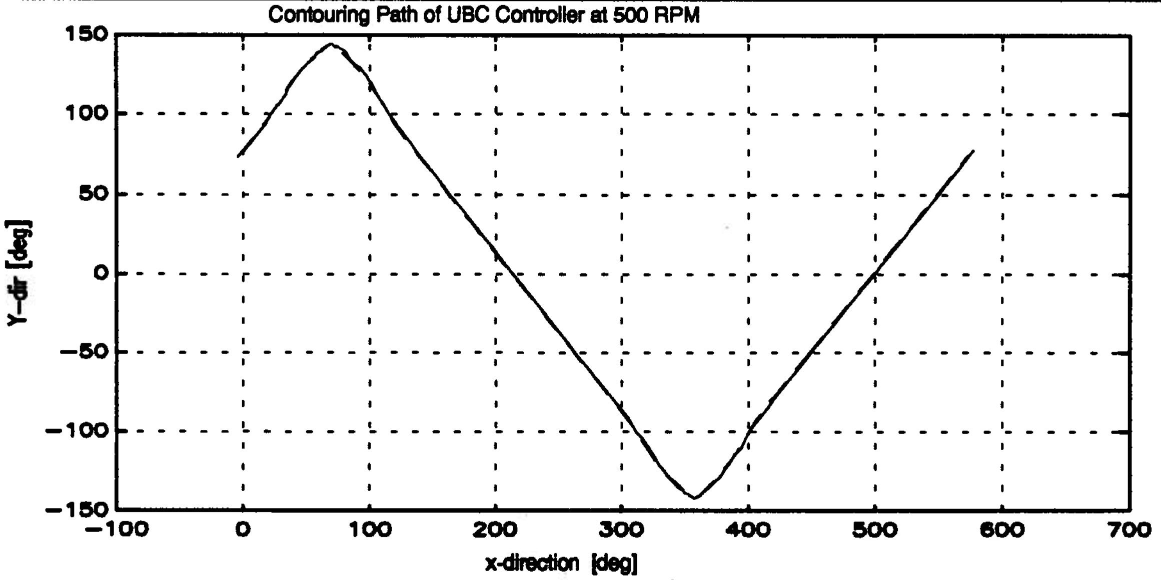


Figure 4.27. Corner Tracking of UBC Controller at 500 RPM
Solid : Reference Position
Dashed : Actual Position

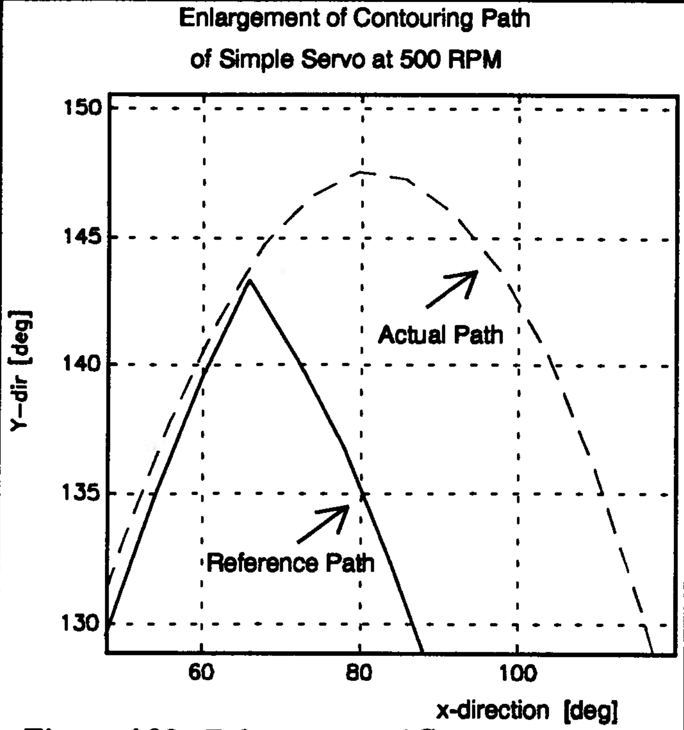


Figure 4.29. Enlargement of Corner Tracking of Simple Servo at 500 RPM
Solid : Reference Pos.
Dashed : Actual Pos.

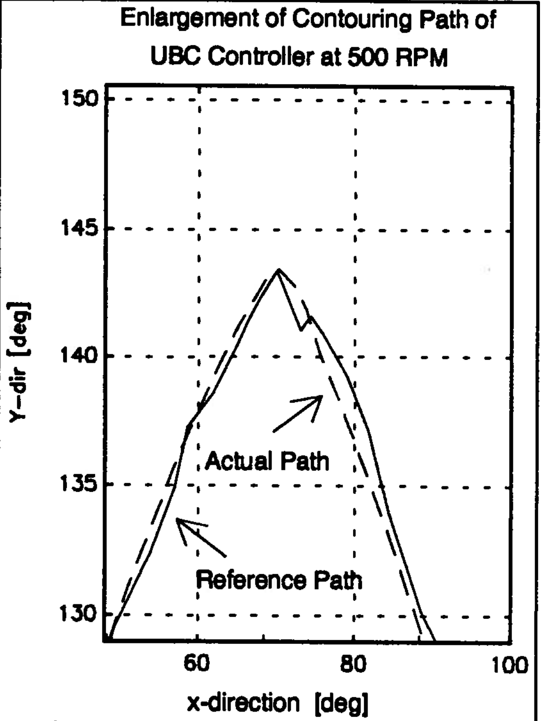


Figure 4.28. Enlargement of Corner Tracking of UBC Controller at 500 RPM
Solid : Reference Pos.
Dashed : Actual Pos.

4.2.5.2 Reversal at 1000 RPM

Figure 4.30 shows a plot of the first axis versus the second axis of a simple servo system. Figure 4.31 shows the response of this same system with an active state line. Figure 4.32 is a magnified view of the first corner of the simple servo system. Figure 4.33 is an enlargement of the performance during the first corner of the system with an active state line. The simple servo system now has a path error of approximately 90° and the system with the active state line has a path error of approximately 7°.

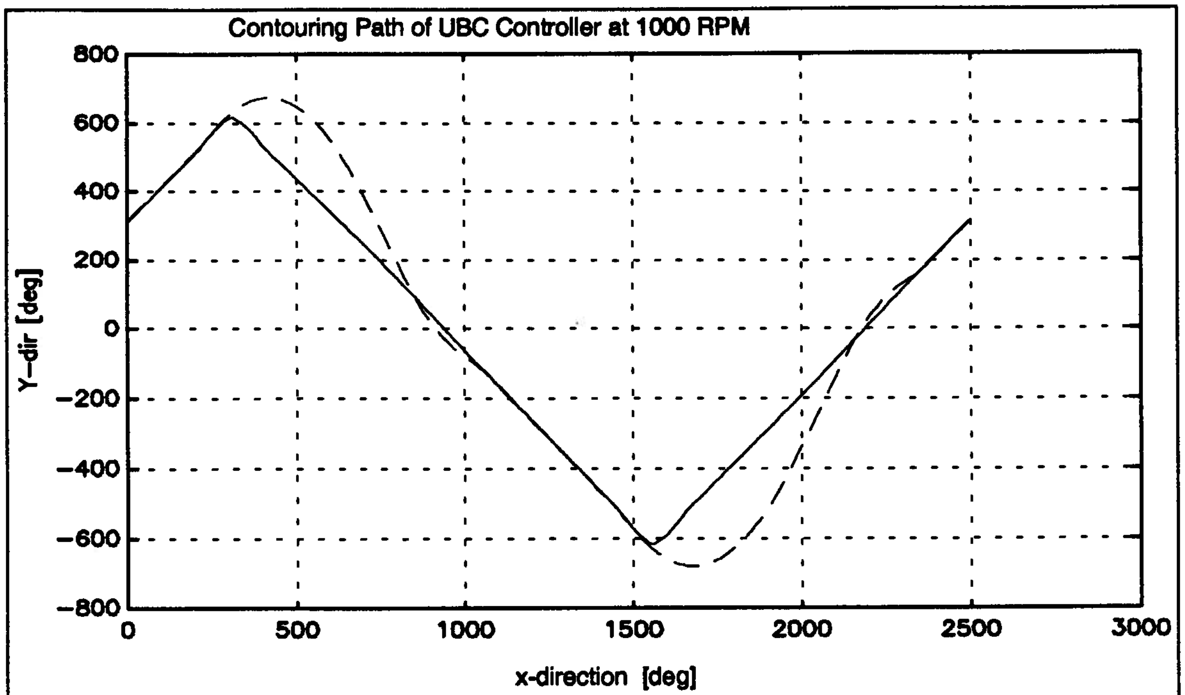


Figure 4.30. Corner Tracking of Simple Servo at 1000 RPM
Solid : Reference Position
Dashed : Actual Position

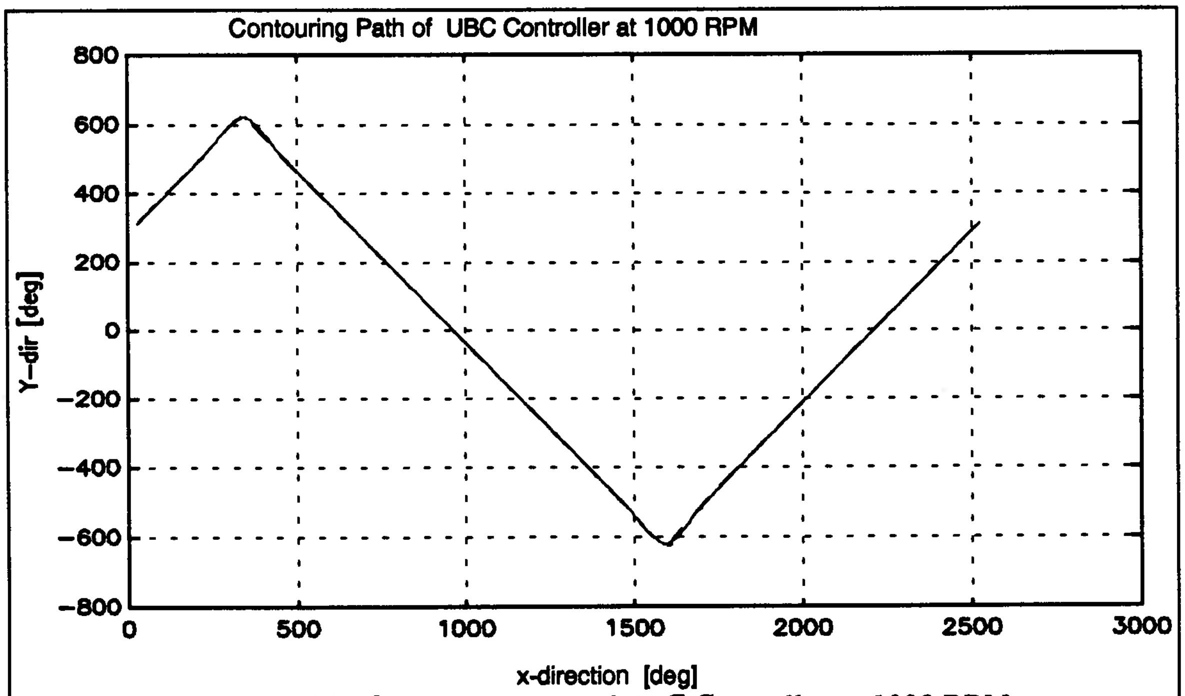


Figure 4.31. Corner Tracking of UBC Controller at 1000 RPM
Solid : Reference Position
Dashed : Actual Position

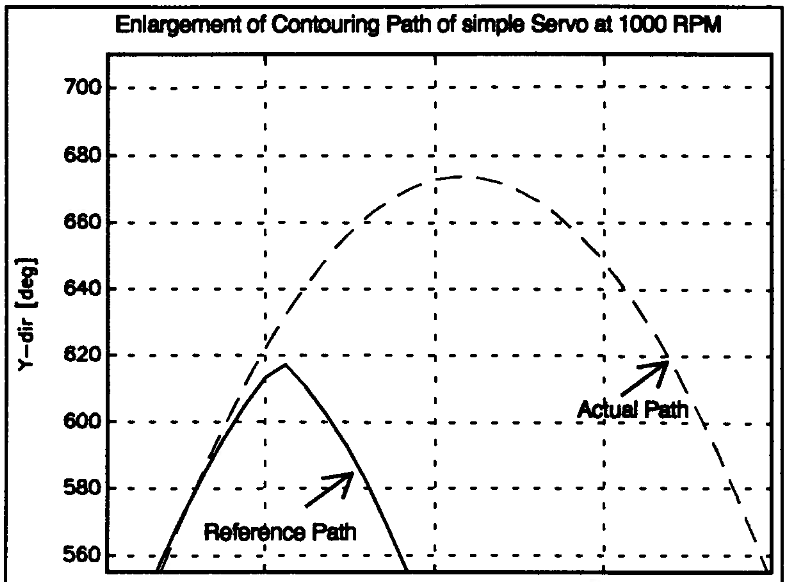


Figure 4.32. Close-up of Corner Tracking at 1000 RPM
Solid : Reference Position
Dashed : Actual Position

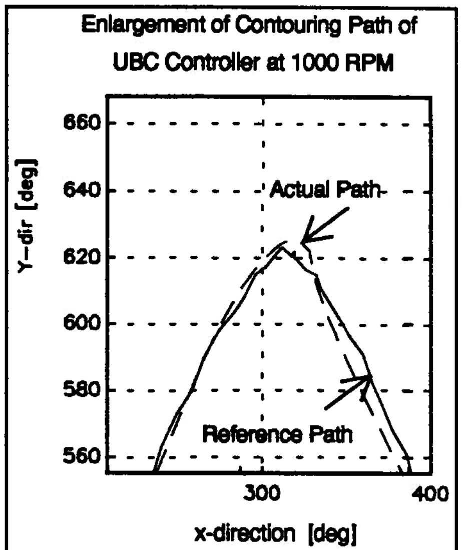


Figure 4.33. Close-up of Corner Tracking of UBC Controller at 1000 RPM
Solid : Ref. Pos.
Dashed : Act. Pos.

4.2.5.3 Comparison with Other Proposed Controllers

The results shown in the previous section are fairly good. In order to compare them with other algorithms described in the literature, it is necessary to select a performance parameter (since no two researchers use the same experiment.) It was decided to choose path velocity over path error for this comparison :

$$\eta = \frac{\textit{path velocity}}{\textit{path error}} \tag{4.9}$$

This parameter should stay almost constant for different velocities, but it may show poorer results for higher speeds. The arrangement should not to the advantage of the experiments reported here, since these experiments were carried out at very high speeds.

Koren has presented computer simulations for the ZPETC arrangement proposed by Tomizuka as well as his proposed CCC arrangement [16] (both of which have been described in Chapter 3.) Motor velocity in this case was approximately 25 RPM. It should be noted that the data presented is derived from simulations only. Simulations will often give better results than real experiments, because they utilize absolutely accurate models with matched gains, and no unexpected disturbances.

Table 4.7 shows however, that the system proposed in this thesis performs favorably when compared to the simulation results of other systems.

Table 4.7. Performance in Corner Tracking

Control Algorithm	**Single Axis Speed**	**Performance Factor η**
Zero Phase Error Tracking Control	25 RPM	375
Cross Coupled Control	25 RPM	750
UBC Controller	500 RPM	1500
UBC Controller	1000 RPM	850

4.3 Conclusions

In this chapter simulations of different controller strategies have been presented. It is concluded from these simulations that a velocity spline approach to trigger the state line is the best compromise between computational effort and contouring performance.

The experimental results are in good agreement with the simulations., validating the conclusions drawn from the simulations.

Constraints have been derived for circular interpolation. They limit the maximum amplitude or frequency of the system according to saturation limits (acceleration), position spline accuracy, and maximum velocity. The experiments carried out show that the velocity spline approach can in fact overcome the amplifier saturation limit.

It has also been shown that the compensation scheme for mismatched axes described in section 2.7 successfully eliminates path errors in linear moves.

The controller's ability to compensate for disturbance torques has been proven by showing that it is able to slow down the complete system almost instantaneously, when one of the axes is stopped by an external torque. This result suggests that the controller will be able to compensate for disturbance torques that are not as drastic as the one chosen in this experiment.

Finally the corner tracking ability of the control algorithm has been extensively tested. The experimental results compare very well with the simulations obtained in previous sections. The system can indeed track a sharp corner at high speeds. The results from the author's experiments have been compared with simulated results, shown in the literature. The UBC controllers performed equally well or better than any other control strategy found.

Chapter 5

Conclusions and Future Developments

Many robotic and contour machining applications require a high degree of accuracy from the feed drives. This goal cannot be accomplished with conventional servo controllers. A great deal of research has gone into the development of advanced servo controllers, however most of these controllers are difficult to implement in practical machining or robotics environments. The latter point arises because the time constants of the plants are so short that digital controllers have difficulties performing the required arithmetic in the allowable time. Finally some of the algorithms rely on an exact model of the plant which may not be available in a real system.

Computer simulations show that the limiting factor in many cases is not the control algorithm, but the physical limits of the system. When a system has to follow a path with a discontinuity or a sharp corner, it is often impossible to outperform a simple servo controller, because the possible acceleration of the system is limited by the limiting current of the actuator. This in turn means that the system will become nonlinear at high accelerations. Such behavior is often neglected in algorithms and simulations of controller performance.

In this work a new approach to contouring is proposed, based on a special controller architecture developed at UBC. Most industrial machines today are servo controlled. If one axis cannot follow its prescribed path due to a current limit or a disturbance torque, the whole system may have large errors. By using the UBC controller architecture, it is possible to slow down the complete system in those cases where one axis cannot follow the prescribed path. This is achieved by using a state line on the front plane bus, which connects all the slaves. When the state line is high, all slaves operate normally as simple servos. If one of the slaves encounters a constraint, it pulls the state line low. This tells the other slaves to stop processing new position increments until the slave in question has resolved the problem and pulled the state line high again.

The system has been simulated and the results from the simulations have been used to choose an algorithm for the triggering of the state line, which represents a compromise between computational efficiency and resulting path accuracy.

The system has been analyzed in Chapter 3 of this thesis. A proposal for the selection of all pertinent parameters is given. This selection process is quite simple, and usually provides reliable and stable controls.

In a series of experimental tests, the simulated results have been validated. The results of these tests indicate that this system is able to achieve contouring performances that are better than other systems found in the literature to this point.

A simple algorithm for compensation of mismatched gains was also developed and tested in a straight line move on the experimental system. The experiments indicate that the algorithm is able to compensate for mismatched gains.

Since disturbance torques compromise the contouring performance of real systems, a test was carried out to demonstrate the robustness to external disturbances of the controller. During a linear move, one axis was stopped abruptly. In a normal contouring system other axes would continue to operate normally, and thus a large path error would be introduced. The UBC controller on the other hand was able to detect the error and stop the second axis in its path. A minimal path error was detected.

In the future it would be of interest to combine the control algorithm with a path planning algorithm. A team under the leadership of Dr. Cherchas at UBC has been working on a path planning program for robotic applications, which utilizes Auto Cad for solid modeling. It should be possible to utilize their findings in a real time as opposed to preprocessing stage. A separate master computer could be used to perform the path planning, which would then pass the required path directly over the STD-32 bus to the master controller.

This controller can be used in machining operations. These applications often require technological input to the controller. Mr. Ramin Ardekani [22], has developed a control algorithm for the UBC controller that is able to use the state line in order to limit the cutting force or equivalent chip thickness on a lathe. It would be of interest to fuse this system with the new contouring control. This should be possible, because both applications rely on the state line to slow down the system when a constraint is encountered. This means that the system can either be slowed down by contouring constraints or machining constraints, depending on which constraint is dominant.

It should be mentioned that the control algorithm, as it stands, is designed for Cartesian machines. Unfortunately there are a lot of non Cartesian machines used in industry (especially robots). It is possible to transform the Cartesian contour errors to joint coordinates using a Jacobean transformation. In this way the controller can be modified for non Cartesian machines with little more computational effort.

Bibliography

[1] Craig, J. J., 1988, *Adaptive Control of Mechanical Manipulators*, Addison Wesley Publishing Company, Reading, Mass., pp. 2-4.

[2] Ernst, H. A., 1962, "MH-1, A Computer-Oriented Mechanical Hand", *Proc. 1962 Spring Joint Computer Conf.*, San Francisco, Calif., pp. 39-51.

[3] Pieper, D. L., 1968, "The Kinematics of Manipulators under Computer Control", Artificial Intelligence Project Memo no. 72, Computer Science Department, Stanford University, Palo Alto, Calif.

[4] Kahn, M. E., Roth, B., 1971, "The Near-Minimum Time Control of Open Loop Articulated Kinematic Chains", *Trans. ASME, J. Dynamic Systems, Measurement and Control*, vol. 93, March, pp. 164-172.

[5] Paul, R. P., 1981, *Robot Manipulator: Mathematics, Programming, and Control*, MIT Press, Cambridge, Mass.

[6] Denavit, J., Hartenberg, R. S., 1955, " A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices", *J. App. Mech.* , vol 77, pp. 215-221.

[7] Uicker, J. J., 1965, "On the Dynamic Analysis of Spatial Linkages using 4x4 Matrices", Ph.D. dissertation, Northwestern University, Evanston, I11.

[8] Stepanenko, Y., Vukobratovic, M., 1976, "Dynamics of Articulated Open-Chain Active Mechanisms", *Math-Biosciences*, vol. 28, pp. 137-170.

[9] Koren, Y., 1985, *Robotics for Engineers*, McGraw-Hill Book Company, New York, pp 51-59

[10] Craig, J. J., 1986, *Introduction to Robotics, Mechanics and Control*, Addison Wesley Publishing Company, Reading, Mass., pp. 191-222.

[11] Tomizuka, M., 1987, " Zero Phase Error Tracking Algorithm for Digital Control", *ASME Transaction* , *Journal of Dynamic Systems, Measurement and Control*, vol 109, March, pp 65-68

[12] Weck, M. and Ye, G., 1990, "Sharp Corner Tracking Using The IKF Control Strategy", *Annals of CIRP*, vol. 39, January, pp. 437-441.

[13] Koren, Y., 1980, "Cross-Coupled Biaxial Computer Control for Manufacturing Systems", *ASME Transaction* , *Journal of Dynamic Systems, Measurement and Control*, vol 102, no. 4, December , pp 265-272

[14] Merrit, R., 1982, "Industrial Robots: Getting Smarter All The Time", Instruments and Control Systems, vol. 55, no. 7, pp. 32-38.

[15] Fu, K. S., Gonzalez, R. C., Lee, C. S. G., 1987, *Robotics: Control, Sensing, Vision, and Intelligence*, McGraw-Hill Book Company, New York., pp. 450-473.

[16] Koren, Y. and Lo, C. C., 1992, "Advanced Controllers for Feed Drives", *Annals of CIRP*, vol. 41, Feb., pp. 689-698.

[17] Haack, B. and Tomizuka, M., 1991, "The effect of Adding Zeroes to Feed forward Controllers", *ASME Journal of Dynamic Systems, Measurement and Control*, vol. 113, March, pp. 6-10.

[18] Buttler, J., Haack, B. and Tomizuka, M., 1991, "Reference Input Generation for High Speed Coordinated Motion of a Two Axis System", *ASME Journal of Dynamic Systems, Measurement and Control*, vol. 113, March, pp. 67-74.

[19] Koren, Y. and Lo, C. C., 1991, "Variable Gain Cross-Coupling Controller for Contouring", *Annals of CIRP*, vol. 104, Aug., pp. 371-374.

[20] Yellowley, I. and Pottier, P., 1989, " A Note On a Simple Method for the Improvement Of Interpolation accuracy in a General Purpose, Multiprocessor Based Motion Controller", *Int. J. Mach. Tools Manufacturing.*, vol. 29, no. 2, pp. 287-292

[21] Pottier, P., 1991, "Advanced Controller for Machine Tools", M.A.Sc. Thesis, McMaster Univ.

[22] Ardekani, R., 1992, "Integration of Manufacturing Process Control and Optimization With a General Purpose Multi-Processor Controller", M.A.Sc. Thesis, Univ. of British Columbia.

Appendix 1

Derivation of Following Error in the z-domain

The derivations in Chapter 3 were based on a very simple forward difference approximation of the discrete filter. It was deemed desirable to prove that this representation is adequate for the design of the contouring algorithm of the UBC controller. Thus the following error was derived in the z-domain and compared with the results obtained in chapter 3.

The simplified continuous part of Figure 3.2 can be transformed into a third order discrete system, assuming that the D/A converter behaves like a zero order hold. The closed loop position transfer function can be described by :

$$\frac{\theta(z)}{R(z)} = \frac{K1\left(\beta_1 z^{-1} + (\beta_2 - A\beta_1)z^{-2} - A\beta_2 z^{-3}\right)}{1 + (B + \alpha_1 + K\beta_1)z^{-1} + (B\alpha_1 + \alpha_2 + K\beta_2 - KA\beta_1)z^{-2} + B\alpha_2 - KA\beta_2} \qquad \text{(A1.1)}$$

where:

$$\beta_1 = \tau_1\left(\frac{\Delta T}{\tau_1} - 1 + e^{-\frac{\Delta T}{\tau_1}}\right) \qquad \beta_2 = \tau_1\left(1 - e^{-\frac{\Delta T}{\tau_1}}\left(1 + \frac{\Delta T}{\tau_1}\right)\right)$$

$$\alpha_1 = -\left(1 + e^{-\frac{\Delta T}{\tau_1}}\right) \qquad \alpha_2 = e^{-\frac{\Delta T}{\tau_1}}$$

$$K1 = Kp\, Kd\, Kss\, Ke$$

Equation A1.1 can be rewritten into :

$$\frac{\theta(z)}{R(z)} = K1\frac{b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3}}{1 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3}} \qquad \text{(A1.2)}$$

The transfer function for the error is:

$$\frac{E(z)}{R(z)} = K1\frac{1+(a_1-b_1)z^{-1}+(a_2-b_2)z^{-2}+(a_3-b_3)z^{-3}}{1+a_1z^{-1}+a_2z^{-2}+a_3z^{-3}} \tag{A1.3}$$

Applying the final value theorem to a ramp input yields:

$$e_{ss} = \lim_{z\to 1}\left(1-z^{-1}\right)\frac{v\Delta T z^{-1}}{\left(1-z^{-1}\right)^2}K1\frac{1+(a_1-b_1)z^{-1}+(a_2-b_2)z^{-2}+(a_3-b_3)z^{-3}}{1+a_1z^{-1}+a_2z^{-2}+a_3z^{-3}} \tag{A1.4}$$

Using long division this can be simplified to:

$$e_{ss} = v\Delta T\,K1\frac{3+2(a_1-b_1)+(a_2-b_2)}{1+a_1+a_2+a_3} \tag{A1.5}$$

This representation yields the same result as the continuous solution, which was presented in section 3.2. It is much longer and there is no advantage over the continuous solution.

Appendix 2

Constant Jerk Position Spline

The original position spline that was proposed by Yellowley and Pottier [20] has velocity discontinuities. During the course of this research a new spline was proposed and tested. It has constant jerk and a linear change of acceleration over one master sampling period. This results in a continuous velocity, with acceleration discontinuities.

The displacement over one master sampling interval for a path with constant jerk is described by :

$$x = x_0 + v_0 \frac{t}{\Delta t} + \frac{1}{2} a_0 \left(\frac{t}{\Delta t}\right)^2 + \frac{1}{6} j_0 \left(\frac{t}{\Delta t}\right)^3 \qquad \text{(A2.1)}$$

where : $0 \le t \le \Delta t$

The spline assumes that the starting and finishing velocities are the averages of the two adjacent velocities at those points. This assumption provides two boundary conditions.

The boundary conditions at t=0 is :

$$v_0 = \frac{V_i + V_{i-1}}{2} \qquad \text{(A2.2)}$$

where : $V_i = \dfrac{X_{i+1} - X_i}{\Delta t}$

The boundary condition at t=Δt is :

$$v_f = \frac{V_{i+1} + V_i}{2} \qquad \text{(A2.3)}$$

The instantaneous velocity is defined as :

$$v = v_0 + a_0 \frac{t}{\Delta t} + \frac{1}{2} j_0 \left(\frac{t}{\Delta t} \right)^2 \tag{A2.4}$$

The boundary condition at t=0 is met automatically, but the boundary condition at t=Δt requires :

$$v_f = \frac{V_{i+1} + V_i}{2} = v_0 + a_0 + \frac{1}{2} j_0 \tag{A2.5}$$

Substituting equation A2.2 into equation A2.5 leads to :

$$\frac{V_{i+1} - V_{i-1}}{2} = a_0 + \frac{1}{2} j_0 \tag{A2.6}$$

It is necessary to have the correct mean velocity over a master sampling period. This requirement provides the second equation in order to solve for the initial acceleration and jerk :

$$V_i = v_0 + \frac{1}{2} a_0 + \frac{1}{6} j_0 \tag{A2.7}$$

Substituting equation A2.2 for v0 leads to :

$$\frac{V_i - V_{i-1}}{2} = \frac{1}{2} a_0 + \frac{1}{6} j_0 \tag{A2.8}$$

Solving equations A2.6 and A2.8 yields:

$$a_0 = -V_{i+1} + 3V_i - 2V_{i-1} \tag{A2.9}$$

$$j_0 = 3V_{i+1} - 6V_i + 3V_{i-1} \tag{A2.10}$$

This spline ensures continuous displacement and velocity profiles. It is an improvement over the old spline that assumes a constant acceleration over a master sampling time interval. Figure A2.1 shows an example of a velocity profile of the constant jerk position spline.

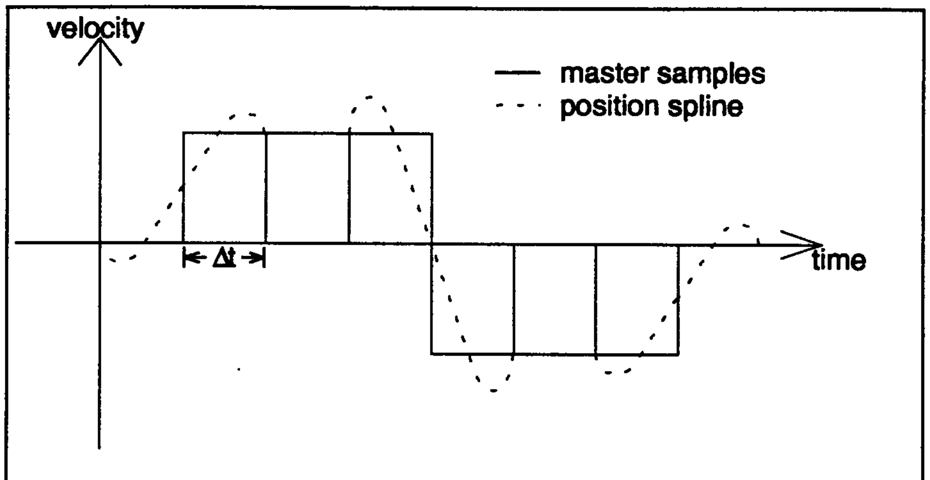


Figure A2.1 Constant Jerk Position Spline

The velocity spline of this position spline is described by equation A2.4. In those cases where the acceleration exceeds the maximum allowable limit imposed on the system by current saturation, an new average velocity is chosen such that the acceleration stays within its bounds. Maximum acceleration can either occur at the beginning or at the end of the sampling period.

case 1: maximum acceleration at t=0 :

$$a_0 = a_{\max} \tag{A2.11}$$

Substituting into equation A2.9 and rearranging yields :

$$V_i = \frac{a_{\max} + V_{i+1} + 2V_{i-1}}{3} \tag{A2.12}$$

This result can be used to calculate the remaining coefficients :

$$v_0 = \frac{V + V_{i+1i}}{2} = \frac{2V_{i+1} + 7V_{i-1} + 2a_{\max}}{6} \tag{A2.13}$$

$$j_0 = 3V_{i+1} - 6V_i + 3V_{i-1} = V_{i+1} - V_{i-1} - 2a_{\max} \tag{A2.14}$$

case 2: maximum acceleration at t=Δt :

$$a_0 + j_0 = a_{\max} \tag{A2.15}$$

Substituting into equations A2.9 & A2.10 leads to:

$$V_i = \frac{2V_{i+1} + V_{i-1} - a_{\max}}{3} \tag{A2.16}$$

This can be used to derive the remaining coefficients :

$$v_0 = \frac{V_i + V_{i-1}}{2} = \frac{2V_{i+1} + 4V_{i-1} - a_{\max}}{6} \tag{A2.17}$$

$$a_0 = -V_{i+1} + 3V_i - 2i = V_{i+1} - V_{i-1} - a_{\max} \tag{A2.18}$$

$$j_0 = 3V_{i+1} - 6V_i + 3V_{i-1} = -V_{i+1} + V_{i-1} + 2a_{\max} \tag{A2.19}$$

Simulations showed that the added complexity of this spline does not significantly improve the performance of the system. It was decided to keep the original spline with constant acceleration over one sampling period.

Appendix 3

Listings of Matlab Simulation Source Code

During the course of this thesis, a computer program was generated using Matlab, that is able to simulate an n-axis servo system with current amplifiers. The program uses a Runge-Kutta second order algorithm to solve the differential equations of the velocity loop. It should be mentioned that the program allows for current saturation and limited bandwidth of the amplifiers.

A3.1 Main Program

Filename : *main.m*

```
% *******************************
% Main program to simulate servomotors
% *******************************

clear;                                  % Clear variables

% Number of links

n=2;                                    % 2 motors

% Step response data

Kss=2000*2*pi/60/10*ones(n,1);          % Steady state velocity 2000 rpm /
10 V
tau=5.e-3*ones(n,1);                    % Time constants 5 ms

% Global parameters

global Je Kt Ka Itau Ktsa Ke VCMD Imax Vmax Kd Rt;

Imax=10*ones(n,1);                      % 12 amp current limit during
simulation
Vmax=10*ones(n,1);                      % 10 Volts D/A maximum output
DAres=2^11*ones(n,1);                   % 12-bit D/A resolution
```

```
Je=(0.135+1.5e-3)*7.0616e-3*ones(n,1);        % Equivalent inertia [N*m*s^2/A]
Kt=42*7.0616e-3*ones(n,1);                     % Torque constant [N*m/A]
Ka=(Kss./tau).*(Je./Kt);                       % Amplifier constant [A/V]
Itau=.63/750/2/pi*ones(n,1);                   % Amplifier time constant
Ktsa=Kss.^(-1);                                % Tach generator constant
Ke=4*1000/2/pi*ones(n,1);                      % Encoder gain
Kd=Vmax./DAres;                                % D/A gain
C2D=Ke.^(-1)*360/2/pi;                         % Conversion counts to degrees

psi=3e-3*ones(n,1);                            % following error of 3 ms
zeta=.707*ones(n,1);                           % damping ratio of one

% Sampling times

Ts=0.5e-3;                             % Slave sampling interval
Tms=32*Ts;                             % Master sampling

% Filter parameters in s-domain

a=Ktsa.*Ka.*Kt./Je;                    % filter lead cancels plant lag
K=4*Je./Kd./Ka./Kt./Ke.*(zeta./psi).^2; % filter gain
b=K.*Kd.*Ka.*Kt.*Ke./Je.*psi;          % filter lag

% Filter parameters in z-domain

Kp=K;                                  % filter gain
A=ones(n,1)-Ts*a;                      % filter lead
B=Ts*b-ones(n,1);                      % filter lag

% Error limits

global Kve del amax;

Kve=psi;                               % allowable velocity error
del=2*ones(n,1);                       % 5 BLU allowablwe static error
amax=Ivmax.*Kt.*Ke./Je;                % Acceleration limit


% Master and slave sample generation

sample                                 % Calculate master samples

% subsample using constant acceleration
subsample;                             % Calculate subsamples
```

```
% subsample using constant jerk
% lsubsample;                         % Calculate subsamples

% Simulation

position;                             % Simulate motors

% Plot results

plo;                                  % Plot results
```

A3.2 Subprogram to Create Master Samples

Filename : sample.m

```
% ******************
% Create Master Samples
% ******************

% clear master samples

clear MR;

% Master samples for 500 RPM reversal

% MR=533*[[2:1:6;2:1:6],[7:1:12;5:-1:0],[13:1:16;1:1:4]];

% Master samples for 1000 RPM reversal

MR=1067*[[8:1:13;8:1:13],[14:1:26;12:-1:0],[27:1:33;1:1:7]];

% Master samples for sin wave

% MR=1000*[sin(0:pi/8:2*pi);cos(0:pi/8:2*pi)];
```

A3.3 Subprogram to Create Slave Samples

There are two versions of this file. the first one creates a position spline with linear acceleration, and the second one creates a position spline with linear jerk.

A3.3.1 Subsample with Linear Acceleration Position Spline

Filename : subsample.m

```
% ****************************
% Subsample with linear acceleration
% ****************************

clear R;
clear Ve;

[k,q]=size(MR);
m=Tms/Ts;
k=2;

MR=[MR(1:n,1)*ones(1,k/2),MR(1:n,:),MR(1:n,q)*ones(1,k/2+1)];
V=(MR(1:n,2:q+k+1)-MR(1:n,1:q+k))/Tms;

R(1:n,1)=MR(1:n,1);
Ve=sum(V(1:n,1:k)')'/k;

V1=Ve;
for i=k/2+1:q+k/2-1,
        V0=V1;
        V1=sum(V(1:n,i-k/2+1:i+k/2)')'/k;
        a=(V1-V0)/Tms;

% Interpolate

        for t=1:m,
                Ve(1:n,(i-k/2-1)*m+t+1)=V0+t/m*a*Tms;
                R(1:n,(i-k/2-1)*m+t+1)=MR(1:n,i)+(MR(1:n,i+1)-MR(1:n,i))*t/m+(t^2-
m*t)*(MR(1:n,i+2)-MR(1:n,i+1)-MR(1:n,i)+MR(1:n,i-1))/(4*m^2);
        end;
end;

MR=MR(:,k/2+1:q+k/2);
```

A3.3.2 Subsample with Linear Jerk Position Spline

Filename :   lsubsample.m

```
% ************************************
% subsample in with linear jerk position spline
% ************************************

clear R;
clear Ve;

Amax=amax*Tms^2;

[k,q]=size(MR);
m=Tms/Ts;

MR=[MR(1:n,1),MR(1:n,:),MR(1:n,q),MR(1:n,q)];

V=MR(1:n,2:q+3)-MR(1:n,1:q+2)

R(1:n,1)=MR(1:n,1);
for i=2:q+1,

        v0=(V(1:n,i)+V(1:n,i-1))/2;
        a0=-V(1:n,i+1)+3*V(1:n,i)-2*V(1:n,i-1);
        j0=3*V(1:n,i+1)-6*V(1:n,i)+3*V(1:n,i-1);

        for j=1:n,
                if abs(a0(j,1))<Amax(j,1),
                        ve(j,1)=v0(j,1);
                        ae(j,1)=a0(j,1);
                        je(j,1)=j0(j,1);
                else,
                        ve(j,1)=1/6*(2*V(j,i+1)+7*V(j,i-1)+2*Amax(j,1)*sign(a0(j,1)));
                        ae(j,1)=Amax(j,1)*sign(a0(j,1));
                        je(j,1)=3*V(j,i+1)-6*V(j,i)+3*V(j,i-1)-2*ae(j,1);
                end;
                if abs(a0(j,1)+j0(j,1))>Amax(j,1),
                        ve(j,1)=1/6*(2*V(j,i+1)+4*V(j,i-1)-
Amax(j,1)*sign(a0(j,1)+j0(j,1)));
                        ae(j,1)=V(j,i+1)-V(j,i-1)-Amax(j,1)*sign(a0(j,1)+j0(j,1));
                        je(j,1)=-V(j,i+1)+V(j,i-1)+2*Amax(j,1)*sign(a0(j,1));
                end;
        end;

        for t=1:m,
            R(1:n,(i-2)*m+t+1)=MR(1:n,i)+v0*t/m+1/2*a0*(t/m)^2+1/6*j0*(t/m)^3;
            Ve(1:n,(i-2)*m+t+1)=(ve+ae*(t/m)+1/2*je*(t/m)^2)/Tms;
            Ae(1:n,(i-2)*m+t+1)=(ae+je*(t/m))/Tms;
      end;
end;

MR=MR(:,2:q+1);
```

A3.3 Subprogram to Simulates the Position Loop

Filename : position.m

```
% ***********************************
% Position loop subprogram called from main
% ***********************************

% Clear variables

clear theta;
clear time;
clear omega;
clear error;
clear filt;
clear Vcmd;
clear Rp;
clear Vp;
clear R1;

% Order of predictor
po=1;
tim=po+1;

% Initial conditions

[k,l]=size(R);
time=Ts*[-po:0];
theta=[zeros(n,n-1),R(:,1)]*ones(n,tim);
omega=zeros(n,tim);
I=zeros(n,tim);
error=zeros(n,tim);
filt=zeros(n,tim);
Vcmd=zeros(n,tim);
Rp=[theta(:,1:po-1),R,R(:,l),R(:,l),R(:,l),R(:,l),R(:,l),R(:,l),R(:,l),R(:,l),R(:,l)];
Vp=[zeros(k,po-1),Ve,Ve(:,l),Ve(:,l),Ve(:,l),Ve(:,l),Ve(:,l),Ve(:,l),Ve(:,l),Ve(:,l),Ve(:,l)];
```

```
[y,z]=size(Vp);
Va=sign(Vp);
for i=1:n,
        for j=2:z,
                if Va(i,j) == 0,
                        Va(i,j)=Va(i,j-1);
                end;
        end;
end;

R1(:,1)=Rp(:,1);
pf=16;

% Start simulation

for step=po+1:l+po-1,

        sflag=1;

        while ((errflag(Rp,R1,Vp,Va,step,theta,tim,n) > 0.5) | (sflag > 0.5)),

                sflag=0;

                R1(:,tim)=Rp(:,step);

                if ((tim-po-1)/16-floor((tim-po-1)/16)) < 1e-5,
                        plot(time,C2D(1)*R1(1,:),time,C2D(1)*theta(1,:),time,C2D(2)*R1
                        (2,:),time,C2D(2)*theta(2,:)),
                        title('Response, Motor 1 & 2'),
                        xlabel('time [sec]'),ylabel('angle [deg]'),grid;
                end;

                error(:,tim)=R1(:,tim)-theta(:,tim);

%       Velocity feedforward
%               error=error+Kve.*Vp(:,step);

                filt(:,tim)=Kp.*(error(:,tim)-A.*error(:,tim-1))-B.*filt(:,tim-1);
                VCMD=filt(:,tim);

                for i=1:n,
                        if abs(VCMD(i))>DAres(i),
                                VCMD(i)=sign(VCMD(i))*DAres(i);
                        end;
                end;
                VCMD=VCMD.*Kd;

                Vcmd(:,tim)=VCMD;

% solve differential equation
                clear t;
                clear y;
                [t,y]=runge2((tim-po-1)*Ts,Ts/3,3,[omega(:,tim);theta(:,tim);I(:,tim)],n);
                tim=tim+1;
                [k,q]=size(t);
                k=k*q;
                time(tim)=t(k);
                omega(:,tim)=y(k,1:n)';
                theta(:,tim)=y(k,n+1:n*2)';
                I(:,tim)=y(k,2*n+1:3*n)';

        end;
        R1(:,tim)=Rp(:,step+1);
end;

% calculate current
for i=1:n,
        for j=1:tim,
                if abs(I(i,j))>Imax(i),
                        Is(i,j)=sign(I(i,j))*Imax(i);
                else
                        Is(i,j)=I(i,j);
                end;
        end;
end;
```

A3.3.1 Function to perform Runge Kutta Second Order Integration

Filename : *runge2.m*

```
% ******************************
% Runge Kutta Second Order Algorithm
% ******************************

function [time,y1]=runge2(t0,h,tn,y0,n)

tim=1;
y1(tim,:)=y0';
```

```
time(tim)=t0;
for t=1:tn,
        k1=h*rdyn(y1(tim,:)',n);
        k2=h*rdyn((y1(tim,:)+k1)',n);
        tim=tim+1;
        time(tim,1)=t0+h*t;
        y1(tim,:)=y1(tim-1,:)+(k1+k2)/2;
end;
```

A3.3.1.1 Function Containing Differential Equations of Velocity Loop

Filename : rdyn.m

```
% ******************************
% Velocity Loop Differential Equations
% ******************************

function yprime=rdyn(y,n)

% current limit

% I=y(2*n+1:3*n);
I=(VCMD-Ktsa.*y(1:n)).*Ka;

for i=1:n,
        if abs(I(i))>Imax(i),
                I(i)=sign(I(i))*Imax(i);
        end;
end;

% Disturbance torque
Td=0*ones(n,1);

% dw/dt
yprime(1:n)=(Kt.*I-Td)./Je;

% dtheta/dt
yprime(n+1:2*n)=y(1:n).*Ke;


% Bandwidth of Amplifier

% dI/dt
yprime(2*n+1:3*n)=((VCMD-Ktsa.*y(1:n)).*Ka-y(2*n+1:3*n))./Itau;
```

A3.3.2 Function Containing Error Checking for State Line

Filename : errflag.m

```
% ********************
% State line Error Checking
% ********************

function f=errflag(Rp,R1,Vp,Va,step,theta,tim,n);

if tim > 3,
        error=(Rp(:,step)-theta(:,tim))+((Rp(:,step)-theta(:,tim))-(R1(:,tim-1)-theta(:,tim-
1)))*0;
else,
        error=(Rp(:,step)-theta(:,tim));
end;

% velocity feedforward

% error=error+Kve.*Vp(:,step);

f=0;
if tim > 2,
        for i=1:n,
                if (sign(Va(i,step))*error(i)) >
(sign(Va(i,step))*Kve(i)*Vp(i,step)+del(i)),
                        f=1;
                end;
        end;
end
```

A3.4 Subprogram to Plot Results

Filename : plot.m

```
% *********
% plot results
% *********

!rm metatmp.met

[k,l]=size(time);
k=k*l;
times(1)=time(1);
```

```
R1s(:,1)=R1(:,1);

for i=2:k,
        times((i-1)*2:(i-1)*2+1)=[time(i),time(i)];
        R1s(:,(i-1)*2:(i-1)*2+1)=[R1(:,i-1),R1(:,i)];
end;

% plot Theta1 and Theta2 vs Time

plot(times,C2D(1)*R1s(1,:),time,C2D(1)*theta(1,:),times,C2D(2)*R1s(2,:),time,C2D(2)
*theta(2,:)),
title('Response, Motor 1 & 2'),xlabel('time [sec]'),ylabel('angle [deg]'),grid;
gtext('Ref 1 -->'),gtext('Ref 2 -->'),gtext('<-- Theta 1'),gtext('<-- Theta 2');
meta metatmp;

% plot Theta1 vs Theta2

plot(C2D(1)*MR(1,:),C2D(2)*MR(2,:),C2D(1)*R1(1,:),C2D(2)*R1(2,:),C2D(1)*theta(1
,:),C2D(2)*theta(2,:)),
title('X-Y Position'),xlabel('theta1 [deg]'),ylabel('theta2 [deg]'),grid;
gtext('Master Ref ->'),gtext('Slave Ref ->'),gtext('<- Position');
meta metatmp;

% plot closeup of theta1 vs theta 2

% 500 RPM
% axis([260,340,265,300]);
axis([260,315,265,300]);

% 1000 RPM
% axis([1200,1500,1200,1300]);
% axis([1200,1300,1200,1300]);

plot(C2D(1)*MR(1,:),C2D(2)*MR(2,:),C2D(1)*R1(1,:),C2D(2)*R1(2,:),C2D(1)*theta(1
,:),C2D(2)*theta(2,:)),
title('X-Y Position'),xlabel('theta1 [deg]'),ylabel('theta2 [deg]'),grid;
meta metatmp;

axis;

% plot filter output vs time

plot(time(1:k-1),filt(1,:),time(1:k-1),filt(2,:)),title('Filter output'),
xlabel('time'),ylabel('Magnitude');
pause;
```

```
% Plot command voltage vs time

plot(time(1:k-1),Vcmd(1,:),time(1:k-1),Vcmd(2,:)),title('Command Voltage');
xlabel('time'),ylabel('Volts');
pause;

% plot current vs time

plot(time(1:k),I(1,:),time(1:k),I(2,:)),title('Current');
xlabel('time'),ylabel('Amperes');
pause;
```

A3.5 Program to Calculate Filter Parameters

Filename : param.m

```
% ***********************
% Determine filter parameters
% ***********************

clear;

% number of links

n=2;                                                    % 2 motors

% Step response data

Kss=2000*2*pi/60/10*ones(n,1);          % Steady state velocity 2000 rpm / 10 V
tau=5e-3*ones(n,1);                     % Time constants 5 ms

% Position loop parameters

Je=(0.135+1.5e-3)*7.0616e-3*ones(n,1);  % Equivalent inertia [N*m*s^2/A]
Kt=42*7.0616e-3*ones(n,1);              % Torque constant [N*m/A]
Ka=(Kss./tau).*(Je./Kt);                % Amplifier constant [A/V] (from step
response)
Ktsa=Kss.^(-1);                         % Tach generator constant
Ke=4*1000/2/pi*ones(n,1);               % Encoder gain
Kd=Vmax./DAres;                         % D/A gain
C2D=Ke.^(-1)*360/2/pi;                  % conversion counts to degrees

% Sampling times
```

```
Ts=.5e-3;                             % Slave sampling interval
N=5;
Tms=2^N*Ts;                           % Master sampling

%Filter design parameters

zeta=.707*ones(n,1);                  % damping ratio of position loop
fw=6;                                 % 6*Ts (3 ms) following error

% Filter parameters

a=Ka.*Kt.*Ktsa./Je;                   % filter cancels dynamics

% b for specified following error and zeta

K=(2*zeta./a./Kss/(Ts*fw)).^(2).*Ka.*Kt./Je./Kd./Ke;
b=2*zeta.*(K.*Ke.*Kd.*Ka.*Kt./Je).^(0.5);

% forward differences

A=ones(n,1)-a*Ts;
B=b*Ts-ones(n,1);
Kp=K;

% Tustins approximation

% A=(2*ones(n,1)-a*Ts)./(2*ones(n,1)+a*Ts);
% B=(b*Ts-2*ones(n,1))./(b*Ts+2*ones(n,1));
% Kp=K.*(2*ones(n,1)+a*Ts)./(2*ones(n,1)+b*Ts);

num=Kp(1)*Kd(1)*Kss(1)*Ke(1)*[0,0,1,a(1)];
den=num+[tau(1),b(1)*tau(1)+1,b(1),0];


% bode plot of the position loop
w = logspace(0,3);
[mag,phase] = bode(num,den,w);
clc
subplot(211)
semilogx(w/2/pi,20*log(mag)/log(10)), grid,title('Magnitude response'),..
xlabel('Frequency (Hz)'), ylabel('Gain dB'), subplot(212),..
semilogx(w/2/pi,phase), grid,title('Phase response'), xlabel('Frequency (Hz)'),..
ylabel('Phase deg'),pause;
!del bode.met;
meta bode;
subplot(111);
```

```
clc;

% Plot step response

t=[0:.001:.03];
y=step(num,den,t);
plot(t,y),title('Step Response'),xlabel('time [sec]'),ylabel('Magnitude'),
grid;
!del step.met
meta step;
pause;

% print filter parameters for forth file

G4=ceil(log(Kp)/log(2));
G1=Kp./2.0.^G4
G2=G1.*A
G3=B
G4

% print velocity error constant

Kve=fw*Ts./Tms
Kve_recommendet=1.25*Kve

% print maximum acceleration

amax=Imax.*Kt.*Ke./Je

% print second order system characteristics

omega=(Kp.*Kd.*Ka.*Kt.*Ke./Je).^(0.5)
zeta=0.5*b./omega
fe=b./a./Kd./Kss./Ke./Kp
```

Appendix 4

Listings of Forth Code for Controller Experiments

The forth code listed in this appendix was used to calibrate the velocity loop and perform contouring experiments.

A4.1 Program to Calibrate Amplifiers

```
\ AMPLIFIER CALIBRATION
 HEX

\ PUT VOLTAGE ONTO DAC ( FFF=10V, 800=0V, 0=-10V )
: DACX DUP 0FF AND SWAP -8 SHIFT 1FD1 BWX 1FD0 BWX ;
: DACY DUP 0FF AND SWAP -8 SHIFT 1FD1 BWY 1FD0 BWY ;

\ STEP INPUT 1.875V=375RPM 980 680
: PXY 888 DACX 880 DACY ;
: ZXY 800 DACX 800 DACY ;
: NXY 780 DACX 780 DACY ;

 DECIMAL
-->


\ READ AND PRINT  SLAVE VARIABLES

\ X-DIRECTION
: PEX 102 WRX  . ;             \ POSITION ERROR
: PAX 144 LRX D. ;             \ ACTUAL POSITION
: RLX 156 WRX  . ;             \ VELOCITY SPLINE LIMIT

\ Y-DIORECTION
: PEY 102 WRY  . ;             \ POSITION ERROR
: PAY 144 LRY D. ;             \ ACTUAL POSITION
: RLY 156 WRY  . ;             \ VELOCITY SPLINE LIMIT
-->
```

A4.2 Data Array Routines

```
\ STORE CURRENT POSITION INTO AN ARRAY

VARIABLE POSSIZE                    \ SIZE OF ARRAY
12000 POSSIZE !
CREATE POS POSSIZE @ ALLOT          \ ALLOCATE ARRAY
: STO
  144 LRX 2 PICK 2! 4 +             \ STORE X & Y POSITION
  144 LRY 2 PICK 2! 4 +
  102 WRX 1 PICK  ! 2 +             \ STORE POSITION ERROR
  102 WRY 1 PICK  ! 2 +
\ 156 WRX 1 PICK  ! 2 +             \ STORE R_LIMIT
\ 156 WRY 1 PICK  ! 2 +;
-->



\ FILL UP THE DATA ARRAY

: FILLPOS                           \ FILL THE ARRAY
  POS
  POSSIZE @ 0 DO
    STO
  12 +LOOP ;
-->



\ WRITE POSITION ARRAY TO THE SCREEN

: PP
  POS POSSIZE @ 0 DO
    DUP 2@ D. 4 +                   \ X-POSITION
    DUP 2@ D. 4 +                   \ Y-POSITION
    DUP  @  . 2 +                   \ X-ERROR
    DUP  @  . 2 +                   \ Y-ERROR
\   DUP  @  . 2 +                   \ X-R_LIMIT
\   DUP  @  . 2 +                   \ Y-R_LIMIT
    CR
  12 +LOOP ;
-->
```

```
\ WRITE POSITION ARRAY TO A FILE
: WP
   SHELL" ATTRIB -R TPOS"                    \ DELETE OLD FILE
   SHELL" DEL TPOS"
   >FILE TPOS                                \ OPEN FILE TPOS
   PP                                        \ TRANSFER DATA
   CONSOLE ;                                 \ CLOSE FILE
-->


\ COPY POSITION FILE TO A:SERVO.MAT
: P2S
    WP
    SHELL" COPY TPOS A:SERVO.MAT" ;

\ COPY POSITION FILE TO A:VELO.MAT
: P2V
    WP
    SHELL" COPY TPOS A:VELO.MAT" ;
-->
```

A4.3 Corner Tracking Using XYL

```
\ TESTRUN AXIS REVERSAL USING XYL
DECIMAL
: RUPERT 6075. X  6075. Y XYL ;

: WORRY 6075. X -6075. Y XYL ;

: DONE  1061. VEL
     10 0 DO RUPERT WORRY LOOP
     FILLPOS ;
-->
```

A4.4 Corner Tracking Using Lower Level Commands

```
\ TESTRUN REVERSAL USING STUFFX/STUUFY

VARIABLE RVEL
VARIABLE MS

: RM1 DECIMAL
   5 N !                        \ 2^3 SLAVE SAMPLES
   CALC_T2^N                    \ CALCULATE MASTER SAMPLING TIME
   500.E0                       \ SET A VELOCITY OF 500 RPM
   4.0E3 F* 6.0E1 F/ FDUP       \ CORESPONDING BLU/SEC
   T2^N F@ F* F>S RVEL !        \ VEL=BLU/SEC*T2^N=BLU/MAST.SAMP.
   2.0E6 F/ T2^N F@ F/
   6.0E0 F* F>S DUP MS !        \ MS=MASTER SAMPLES IN 1 DIREC.
   S>F 20.0E0 F* K2 F!  ;       \ K2=# OF MASTER SAMPLES
-->


: RMOVE
   RM1 CALC_CVS STUFF_INITX STUFF_INITY SEND_HEADER
   10 0 DO
      MS @ 0 DO
         RVEL @  STUFFX  RVEL @ STUFFY
      LOOP
      MS @ 0 DO
         RVEL @ STUFFX RVEL @ NEGATE STUFFY
      LOOP
   LOOP
   0 STUFFX 0 STUFFY
   DUMP_SMALLX DUMP_SMALLY
   FILLPOS;
-->
```

A4.5 Frequency Response Experiment

```
\ FREQUNCY REPOMSE

\ SET UP FREQUENCY RESPOSE EXPEERIMENT

FVARIABLE MAG           \ DESIRED MAGNITUDE
FVARIABLE FREQ          \ DESIRED FREQUENCY
VARIABLE TTIME          \ LENGTH OF EXPERIMENT IN MASTER SAMPLES
-->


\ WRITE MASTER SAMPLES TO TABLE

: FR1
  5 N !  CALC_T2^N                       \ DEFINE MASTER SAMPL
  4.0E0 FREQ F!  3.0E3 MAG F!            \ DEFINE MAG & FRE
  6.283185E0 FREQ F@ F* T2^N F@ F*       \ OMEGA * DELTA TIME
  0E0                                    \ OMEGA0
  5E0 T2^N F@ F/ F>S TTIME !             \ EXP. LASTS 5 SEC.
  POS TTIME @ 4 * + POS DO
    FDUP FSIN MAG F@ F* F>S I !          \ MAG*SIN(OMEGA0*TIME)
    FDUP FCOS MAG F@ F* F>S I 2 + !
    1 FPICK F+                           \ OMEGA0+OMEGA*DEL(TIM)
  4 +LOOP  FDROP FDROP ;
-->


\ CONVERT SAMPLES TO INCREMENTS

VARIABLE OX
VARIABLE OY

: FR2
  POS @ OX ! POS 2 + @ OY !
  POS TTIME @ 4 * + POS DO
    I @ DUP OX @ - I ! OX !
    I 2 + @ DUP OY @ - I 2 + ! OY !
  4 +LOOP;
-->
```

```
\ SHUFFLE TABLE TO CONTROLLER BOARDS

: FR3
  TTIME S>F K2 F!                \ # OF MASTER SAMPLES
  CALC_CVS                       \ PREPARE TO SEND
  STUFF_INITX STUFF_INITY        \ COMMANDS TO BOARDS
  SEND_HEADER
  POS                            \ BEGINNING OF TABLE
  TTIME @ 4 * 0 DO
    DUP @ STUFFX 2 +             \ STUFF POSITION
    DUP @ STUFFY 2 +
  4 +LOOP DROP
  0 STUFFX 0 STUFFY              \ FLUSH PIPE
  DUMP_SMALLX DUMP_SMALLY;
-->


\ FILL UP TABLE WITH RESULTS

: FR4
  POS                            \ BEGINNING OF TABLE
  POSSIZE @ 0 DO
    144 LRX 2 PICK 2! 4 +        \ STORE POSITION
    144 LRY 2 PICK 2! 4 +
    102 WRX 1 PICK  ! 2 +        \ STORE ERROR
    102 WRY 1 PICK  ! 2 +
     56 WRX 1 PICK  ! 2 +        \ STORE STEP
  14 +LOOP DROP;
-->


\ TRANSFER RESULTS FROM TABLES TO SCREEN

: PRES
  POS                            \ BEGINNING OF TABLE
  POSSIZE @ 0 DO
    DUP 2@ D. 4 + DUP 2@ D. 4 +  \ TRANSFER POSITION
    DUP  @  . 2 + DUP  @  . 2 +  \ TRANSFER ERROR
    DUP  @  . 2 +                \ TRANSFER STEP
    CR
  14 +LOOP ;
-->
```

```
\ TRANSFER RESULTS FROM TABLES TO FILE

: FR FR1 FR2 FR3 FR4
   SHELL" DEL FREQRESP"
   >FILE FREQRESP                  \ OPEN NEW FILE
   PRES                            \ WRITE DATA
   CONSOLE                         \ CLOSE FILE
   SHELL" COPY FREQRESP B:FREQRESP.MAT" ;
```