MASSACHUSETTS INSTITUTE OF TECHNOLOGY

ARTIFICIAL INTELLIGENCE LABORATORY

Working Paper No. 129 August 1976

# DIGITAL CONTROL OF A SIX-AXIS MANIPULATOR

by

David C. Blanchard

ABSTRACT. This paper describes a scheme for providing low-level control of a multi-link serial manipulator. The goal was to achieve adaptive behavior without making assumptions about the environment.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-75-C-0643.

## ACKNOWLEDGEMENTS

I am indebted to the following individuals for their contributions to this work.

B. K. P. Horn (Thesis Advisor) -- for patiently guiding me in the selection of a topic and locating the hardware to test my ideas. Also for agreeing to proof-read the draft.

Meyer Billmers -- for familiarizing me with the Mini-Robot Group and for his willingness to help me out of the holes I frequently dug myself into.

Fred Drenckhahn -- (keeper of the VIC-ARM) for helping me to modify the The Arm without damaging it.

Dave Tanzer -- for introducing me to an easier way to accomplish basic I/O on the Mini-Robot System. His monitor routines saved many hours of auxilliary fiddling.

Melissa Weiksnar -- For her professional editing and formatting suggestions.

# TABLE OF CONTENTS

## ABSTRACT

The current approaches to low-level control of manipulators seem to be roughly divisible into three catagories. The first contains the simple *rote memorization* schemes which perform only under the most limited conditions. They are often "hand guided" along a path which is then mindlessly retraced *ad infinitum*. Almost all current industrial robots reside in this catagory.

Secondly *mathematical open-loop* methods attempt to convert a desired joint motion into an accurate sequence of motor torques based on an internalized model of the manipulator. All too often the computation required prohibits the use of this method in real time. Open loop mathematical control techniques under favorable circumstances (no unforseen disruptions) approximate optimal behavior on the first try.

Thirdly *heuristic learning* schemes attempt to revise and improve their technique based on earlier mistakes. These schemes employ the premise that human arm movement is essentially an open loop phenomena with some closed-loop information used in refining the final position. To quote Carl Flatau [1],

> Critical examination of the human anatomy fails to detect anything which could possibly be analogous to a position transducer. Position accuracy seems to be achieved soley by successive approximations in the arm-eye-force sensing systems.

For example, the amazing improvement people experience with practice at racket sports implies that the arm-eye-force feedback mechanism without memory is a description of a rank beginner. The advanced player draws on his accumulated practice to the extent that he requires only slight correctional feedback to *refine* the motion.

One generates a long term learning program in the hopes that it will absorb and eventually compensate any unforeseen aspect. The problem with these controllers is that they tend to require (much like the humans they emulate) prohibitive amounts of practice with many different loads and maneuvers before they can perform.

I strongly favor a control scheme which adjusts and adapts during the course of a single movement. For one thing it can't be "surprised" by a change in load or any other parameter. As I intend to show it needn't involve lengthy calculations that would make it unsuitable for real time operation. Lastly it is able to perform instantly without tiresome rehearsals.

In this paper I define and implement a position and velocity controller for a real-world mechanical manipulator of several links using a hybrid method and taking particular advantage of the ability to make decisions *during* a single movement.

## Historical Remarks

The earliest form of feedback control devised by man was simple proportional control. This method defines the input to a system to be the product of a scalar gain and the difference between the observed output and the desired output. If the desired output is constant over time the result is termed a *regulator* otherwise the term is *controller*. Proportional control has been used for thousands of years. It was not well understood until a century ago when its application--and consequent failure--to control Watt's popular Steam Engine, excited the curiosity of James Clerk Maxwell. Soon, he established enough of a theoretical base to suggest the use of derivative and integral signal as well. This combination (known as PID control) remains popular for its generality. Yet it may be tailored by only limited amounts to a particular task.

PID control seems a reasonable choice for a range of application. Until the advent of digital computers it was at least one of the few simple approaches.

Digital computation brought a new dimension of possibilities to the field of control. Because the possibilities are so new and so vast, most of them lack the analytical foundations of the older analog techniques. Therefore many designers choose not to exploit these new abilities directly. Instead, they merely program the new equipment to simulate analog operation. Although this simulation is entirely appropriate in some cases, one shouldn't feel bound to it.

In pursuit of practical methods for controlling complex serial manipulators (or "kinetic chains") modern control theorists busy themselves by generating reams of analytical literature whose applicability to real-world tasks remains to be demonstrated. Some modern theorists believe that mathematical formulations which embody all masses, energies, joint properties, etc. and purport to provide all necessary information for open loop operation, are essential for good control. Kahn [9] provides such formulation in detail and establishes the notion of a "J matrix"—a state-space description of velocities and orientations of the links in a kinetic chain. Using this, one can solve for a string of current (force) vectors which controls the motor torques. Richard Waters [10] suggests a method of computing forces based on recurrence relations which use this matrix. The fifty-one four by four matrix products (and the few sums) required can take 50 milliseconds (ms)—just on the fringes of feasibility. But what if the mass is unknown? What if the body has a substantial moment as well as mass? I do not mean to suggest that dynamically deducing the J (the matrix description of the link inertias) is a futile excercise. But, it simply doesn't adapt itself generally to unforseen conditions.

Richard Paul [5] confronts the difficulty of servoing in the presence of load variation as follows:

> It can be seen that the system response is dependent on J as would be expected. Because the effective link inertia can vary by 10 to 1 as the arm configuration changes, we are unable to maintain a given response independent of arm configuration. If however we add a gain of minus J ...then we obtain ...servo response independent of arm configuration.

This approach is highly reasonable in the context of a loading dominated by the links themselves. It doesn't fare well with larger, unannounced loadings of external origin.

## Description of the Hardware

Since it is difficult to tailor a controller to a manipulator without knowledge of its basic properties I will provide a short description of the VIC-ARM (see Figure 1) used in this study.

The Arm was designed by Victor Scheinman [8] to be a light, general-purpose, six-axis manipulator for use in artificial intelligence work. Its main appeal is its size (just about "life size"), its flexibility (six axis motion plus grip) and its commendable dynamics—at least compared to its industrial counterparts.

The Arm provides both positional and velocity output and accepts current (force) inputs to its motors. It is gear-driven and has noticable—though not serious—backlash. The noise in its velocity signal is significant (up to 20 per cent) and seems to increase linearly with the velocity.

One very important observation in terms of controlling The Arm is its tendency to velocity-saturate at a relatively low velocity. This is due to current saturation, windage and friction in the motors. These effects tend to de-emphasize the The Arm's mass for all but very short and sudden movements. Therefore, describing gross motions requires a *viscous* as well as an *inertial* model. I will later appeal to a hypothetical arm which may be loaded over a wider range than the VIC-ARM.

## Moving from Here to There in One Dimension

Perhaps the easiest method of getting from here to there is by using PID control. With its three scalar gains properly set, PID control will drive a *linear load* rather nicely to a commanded position. However, once the three scalar gains have been set, the intermediate velocities and displacements which result are beyond one's influence. PID control can make a body with "bad" movement properties behave as if it were lively but stable. Often, the desired goal is to achieve critically damped movement. But for certain applications some criteria are relaxed in order to permit tighter restrictions to be placed on others. For example, one often permits overshoot in the response of an amplifier to achieve a more rapid rise-time. Note, however, that using this trick indiscriminately (e.g., applied to the vertical component of the landing control for an aircraft) would have embarrasing consequences.

So far I have assumed that the load has both inertial and viscous properties (as does The Arm itself). But The Arm has backlash and stiction as well. The backlash essentially disembodies the arm into two loosely connected masses—only one of which may be directly driven. The stiction assures that the velocity will not move from zero until the applied torque exceeds a certain limit.

These effects seriously crimp the expectations for a PID controller applied to The Arm. For example, one must beware of halting *close to* but not *at* the goal position. Normally, an Integral component would correct this discrepancy by gradually nudging The Arm into place. With stiction, however, the nudge won't be felt until it is large enough to cause overshoot—and the process continues to limit cycle. For this reason, the value of integral gain to a manipulator lies mostly in its ability to combat the steady-state error which results from gravity loading.

Using PID control to command the position of a single joint of The Arm presents a deeper problem: PID doesn't account for changes in *effective* mass. These changes depend on the object the arm is holding, its motion relative to gravity and the medium in which it moves. These considerations compel one to be overly conservative with the damping of the joint, lest overshoot occur during movements dominated by inertial rather than friction forces. As a result, the movement of less massive objects is too sluggish. This effect may be demonstrated with the VIC-ARM even without an external load. First a PID controller is tuned such that the extended arm moving in the horizontal plane moves rapidly with no overshoot. Next, the arm is raised to lie along the vertical axis (minimizing its moment of inertia) and the test is repeated with the same PID gains. The motion doesn't overshoot... but it is now possible to increase performance by raising the Proportional gain. Thus using a single set of gains for all arm configurations is a compromise at best.

As previously indicated, PID control applied to a large movement precludes control over intermediate velocities. This limitation may not be tolerable, say, for a motion which involves transporting a glass of water. It is easy enough to imagine tasks requiring better control over the velocity (and displacement) than PID affords. Nonetheless, one may dictate a distance profile to be followed by a PID controller, thereby "leading it by the nose" rather than allowing it to choose its own intermediate trajectory. My earlier experimenting with a single joint showed this method actually works fairly well although a trajectory input is required instead of a scalar destination. For an average movement, a triangular velocity profile (or equally, a parabolic distance profile) gives generally good results.

I have assumed all along that a *D* component is available (for a positional servo this is a velocity input)---not a light-hearted consideration. Velocity input is necessary to dampen the effect of a large proportional gain, and is even more necessary when Integral gain is used. In theory, one could obtain the velocity signal from a differencing of the positional input; in practice such is not recommended. For most real systems the granularity of the position signal and/or its inherent uncertainty are formidible enough to doom the idea. Smoothing the positional readings to obtain a sense of the slope introduces an effective delay owing to the deletion of the *valid* hi-frequency components along with the noise. This delay will make it impossible to attach a reasonable significance (gain) to this input without producing instability in the total result. More will be said on the effects of smoothing later.

The sampling rate is a vital parameter to any digital control scheme. Examining the input too frequently wastes computation time, yet looking too seldom limits the bandwidth of the observed input signal which eventually results in instability from too much delay. Since evaluating a control strategy in the face of unstable operation is difficult, I have been deliberately conservative by using a sampling rate of one millisecond for my tests.

## Moving in N-Dimensional Joint Space

The issues are further constrained in considering simultaneous motion of two or more joints. Unlike an XYZ table moving in cartesian space, the motions of joints interact in angular joint-space. The effect of the Coreolis force is very evident in some movements, the perceived mass changes as a function of its angle and of course there is always gravity to battle.

I do not intend to belabor conversion from joint space to cartesion space—for many tasks it is not even necessary to invoke cartesian space at all. When it *is* necessary, a conversion routine would be placed between the user's calls and my servoing routines to insulate the two geometric systems from each other. B. K. P. Horn [2] has pointed out that an efficient transformation from Joint to Cartesian Space for six joints may be achieved using only 63 muliplications by substituting rotation operators for some matrix multiplication. The reverse transformation is considerably more intricate. In addition to the eight-fold ambiguity due to three arbitrary sign choices, the closed form solution is a polynomial of unholy degree. Fortunately the system may be decomposed into three arm joints and three hand joints, and solved in two parts.

When designing a low-level controller for a multi-axis manipulator, it seems reasonable to grant the user maximum flexibility while doing enough for him to preclude his feeling that he must control every detail of the motion himself. My compromise is to permit him to supply final velocities and positions for all joints. This format facilitates the chaining of calls to produce a large smooth movement. I further intend that each call be capable of defining a fairly large movement (on the order of several inches) thereby preventing the higher level processing (e.g. the Cartesian <=> Joint Space) from bogging down.

It is soon evident that identification and rejection of *illegal* user calls is needed. Certainly a request for The Arm to reach past its linkage span should not be attempted. Beyond the obvious limits of the hardware to generate maximum displacements lurk *illogical* calls. For example, a user might demand a positive position change and yet insist upon arriving at the new location with a negative velocity. This achievement might be possible (by overshooting the mark by an arbitrary distance and back-pedaling to the correct position) but the intent is nontheless fuzzy. It seems better to constrain the user to dividing such maneuvers into smaller, logical, motions to accomplish the same task. Finally there are *daring* calls... e.g. a request for a fast motion whose feasibility isn't known until it is tried. These should be attempted and completed positionally with possible velocity reduction. More on this later.

Most importantly, when the motion of more than one joint is involved, coordination is required to ensure that all movements are simultaneously completed. This outcome is a direct consequence of allowing the user to dictate the final joint velocities. It should be clear that this freedom is not strictly vital: but it does allow the user definite advantages in composing larger movements, and therefore should be retained if it doesn't cause extreme difficulties for the servoing routine. It doesn't. It puts tighter bounds on both the number of feasible calls and the quantity of possible trajectories... but these effects are not severe. The following ground rules emerge for choosing trajectories:

1) A single *time-to-go* for each movement must be established for all joints.

2) The above time will be determined by the joint whose minimum-time trajectory is the *slowest* of the group.

3) Each velocity trajectory is constrained at *both* ends.

4) The *area* under each velocity curve is specified (due to the position constraint).

5) The velocity profiles shall not contain *step changes* which are impossible to realize. However, slope discontinuities will be permitted.

6) Within the limits of the above rules the velocity profiles shall be reasonably devoid of odd bumps and uneven speed changes.

The motivation behind criterion number six is simply that of the family of strictly possible trajectories (based on one thru five) the final choice should favor a smooth ascent over a sawtooth or an unbalanced (time-wise) curve. Obviously it is preferable to spread velocity changes over the full interval, thereby improving the chances that the *total* motion will be as nearly linear in cartesian space as its creator probably intended. If he really wants an L-shaped movement he is entirely free to order it (using two or more calls)... yet it is poor form to arbitrarily hand him one if alternatives exist.

The following scheme preserves the above properties and serves also as the basis for the adaptive method of the next chapter. For every joint we are given a desired position (from which we derive the distance); analogously with the velocity. Additionally we have a parameter for limiting acceleration (slope of the velocity curve). Assuming for the moment that no velocity overshoot or undershoot is permissible, the distance is immediately bounded. Figure 5.1 shows the minimum and maximum distances attainable.

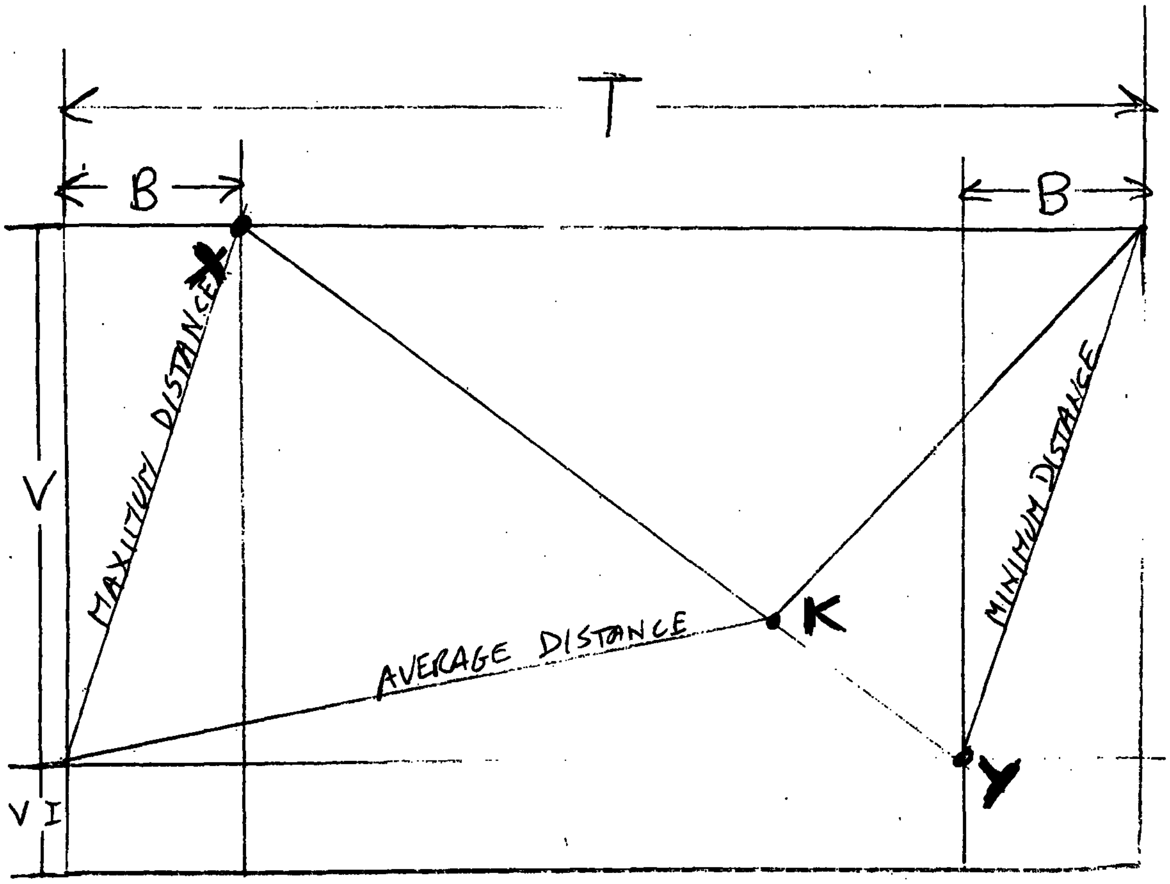

**Figure 5.1 Locus of Velocity Trajectories.**

What remains is only to select the proper area (distance) without changing the endpoint velocities. I have chosen to construct a dividing line which runs from point X to point Y along which I place an intermediate target point called K. I then construct straight lines from the initial velocity to this point, and from this point to the final velocity. By varying the location of this target point along the dividing line any attainable distance may be planned within both the velocity and time constraints. There is, of course a step change in acceleration along the way but this is allowable. The time constants of the electronics are short enough compared to those of The Arm to pretend that a current step is realizable.

The formula for finding the distance A which locates point K given a distance, initial and final velocities, time-to-go and maximum slope is given below:

$$K = \frac{(T - 2B)\left(2T - B - \dfrac{2(D - T(VI))}{V}\right)}{2(T - B)}$$

| | |
|---|---|
| B | - Braking Distance (Defined as V/(Maximum Slope)) |
| T | - Total Time |
| D | - Desired Distance |
| V | - Delta Velocity |
| VI | - Initial Velocity |

Note that *two* design iterations are necessary for each finished group of velocity trajecories. On the first iteration (since no time constraint is dictated) the planner arranges the minimum time for each joint. Since this time through generally produces a spread of transit times, a second round of planning is executed using (for all joints) the slowest time found by the first iteration. The PID controller will actually servo around this trajectory group.

## Moving in R-Space with Tracking Sensitivity.

The sucess of the method outlined in the preceeding chapter hinges on the ability to plan a trajectory which the lower-level PID servoing is *capable* of following. The limitation is presumed to be the finite motor currents one may use. Naturally one is not allowed to load The Arm such that it can't move at all, but it isn't hard to imagine a loading which significantly alters its dynamic properties. Many manipulataors can operate with up to a ten-to-one [1] load/no-load ratio. Here I depart somewhat from the VIC-ARM as I had warned---it will bear roughy one pound without damage to its two pound main-link housing. To argue the value of my new plan I must assume a manipulator with a greater range of carrying capacity. Granted this liberty, one would prefer to retain the condition of minimum time travel but without having to prespecify the load.

One way to deal with the problem of an unknown load is to monitor its progress as one applies known signals and try to deduce the mass. Richard Waters [10] suggests that in lieu of a priori knowledge that the arm "discover the inertial characteristics of the object it is holding. [This] can be done by applying small canonical forces near the hand and observing the accelerations produced." As I already mentioned it isn't clear this monitoring is feasible with the VIC-ARM, particularly due to a lack of direct acceleration feedback and a not-so-pure velocity signal. Furthermore, not only the simple inertial characteristics of the load must be known to allow open loop operation. Suppose the task involves moving in mixed media---e.g. moving objects in and around a tank of water. Unless one specifically programs the controller to expect a watery medium, the confusion of a mass-detecting controller would be boundless. As an early experiment I found the general approach of "mass-deduction" quite frustrating. It works marginally in the one-dimensional case, but only with an *a priori* notion of the friction which will be encountered. The problem is separating the effects of friction from the effects of the mass---which is terribly hard to do from a noisy velocity signal. By "knowing" the friction in advance one may read several velocities (over time) to pin down the mass. Notice that this method is confined to joints with known friction and no gravity loading... not that gravity couldn't also be deduced from the positon of all the supporting links; but such approaches tend to get out of hand.

Easing off a little, one might wish that the controller follow the planned trajectory until it becomes obvious that it *cannot* (for whatever reason) at which time a new *less demanding* trajectory is issued to it. I have simply sidestepped the question of why the old trajectory can't be followed. If it can't be followed, it must be replanned.

Having resolved to make trajectory substitutions en route one must not forget the larger context outlined in the last chapter. All joints must have the same time-to-go. When a less demanding trajectory is bestowed upon a joint in mid-flight it will require more time. Since from this point onward its predicted arrival time will exceed that of its neighbors *the entire group of trajectories must be replanned.* I have outlined the basic steps to be taken below:

## ** A Recipe for Coordinating Tracking Feedback Amoung N Joints **

Establish maximum slope values for all joints.

Plan the movement of each joint to be minimum time.

Identify the slowest movement of the above group.

Replan the others to use this longer time.

Activate movement.

Sound alarm when any joint generates a integrated
error above a maximum limit.

Decide what slope can be expected of it based on how long it had taken to change its velocity by so much.

Ease off this expected slope by 20%.

Replan all trajectories from this point on.

If a trajectory is overconstrained, say, if the time is too large for the velocities and distances given sacrifice the final velocity to save the distance.

Do not permit overspeeding (ie require $v_i<v<v_f$ or $v_i>v>v_f$).

Permit underspeeding to waste time waiting for another joint.

## Results

One of the early problems with this plan involved computation time. I had found a sampling interval of one millisecond to be adequate in previous work and used it here. But when the replanning was triggered (cutting power temporarily), a significant change occured in the velocities of the links. This time is on the order of 20 milliseconds---too long to allow the primary joints to be without power. The remedy was to allow the servoing routine to interrupt the planner at one millisecond intervals and servo the links to the last actual velocity attained before the onset of the new planning. In effect the arm hovers while new orders are formulated.

This system degrades gracefully. When there is no hope of reaching the destination with all the proper velocities the system slips the velocity requirement such that the correct positions are always attained simultaneously.

An example of this method is given below. I used the three major joints (zero, one and two) and the movement represented was quite small scale (on the order of 10 degrees of motion about each axis). I have plotted the results only for joints zero and one. The graphs will be found in Appendix A.

The PID gains were set at 5, 3/16 and 8 respectivly. The P term here is integrated velocity error (shown in Figures three and four). The above gains were chosen through simple experimentation. Indeed there are cookbook methods (like Newton-Ziegler Tuning) which determine the PID gains of a black-box system. Seemingly one can do as well or better if permitted to see the results of one's guesses.

Note that a double integration of the velocity error is used (the I component). It is necessary to avoid steady state error in any joint which would occur from acceleration or gravity.

Figures one and two show the actual velocities of both joints. Figures three and four are the cumulative error. Figures five and six are the initial minimum-time trajectories for both joints. Figures seven and eight are the initial constant-time trajectories and figures nine thru twelve show the final trajectory.

The velocity-chatter in figure two results from too much proportional gain. It looks worse than it is but can be made much smaller by reducing the P-gain.

Accounting for the fact that each dot on the graphs combines two sample intervals, Figure three shows that sixty-eight intervals have elapsed before joint one exceeds the critical cumulative error. At this time, Figures nine through twelve are generated (eleven and twelve being the final trajectories).

I did not provide a graph of motor currents because it would not have been informative. The gains I used (particularly for the velocity error) were high enough to transmit considerable noise to the motors. I tolerated this because using lower gains resulted in mushy response. This problem was particularly evident on gravity loaded joints. Raising the Integral gain lowers the steady state error but one is then controlling the velocity through two integrations; first the velocity error is integrated into cumulative error, second, the force commands (accelerations) are integrated into velocities. Like any second order system, increasing the gain will never cause instability but this doesn't preclude stiff oscillation. The usual methodology is to choose a high enough Proportional gain to satisfy the error requirements and add whatever Derivative gain is nescessary to damp the oscillation. I discovered a very low ceiling existed on the maximum simple error gain. Above this, an offensive jitter occured in the Arm's velocity. In light of this I used the maximum simple error gain and cranked up the cumulative gain until oscillation became noticable.

There remained a problem with gravity-loaded joints. At the start of a movement the brakes release and the limb falls until enough proportional error plus integral error exists to reverse its direction. This situation is not entirely correctable unless one conceeds the use of *a priori* knowledge of the loading—which I had vowed to avoid. Worse still, the cummulative error had a tendency to remain at a constant level in order to provide the torque to counter gravity *independent* of the maneuver commanded of the limb. Since the cummulative error is an equivalent expression of the distance constraint it ought to be nearly zero by the end of the movement even if not before.

The "natural" next step is to introduce yet another integration (doubly integrated error). A threat looms immediately; the entire servo loop is now a third order system with definity potential for instability. I found it impossible to use enough double-integral gain to be useful without further de-stablizing the already shaky control structure. By examining the resulting output I decided that the effect of the double integration should be to *average* rather than to strictly *integrate* the acumulated error. This would allow the double integrator to take prompt effect yet prevent it from driving the cumulative error (and hence itself) violently later in the movement. This did in fact yield a better performance. The final cumulative errors were lower and no less stable. Admittedly there is still room for improvement.

I wondered at first if the direct use of a noisy velocity signal was a mistake. It seemed reasonable that the noise might escalate when fed to the motors and read in again as velocity. I constructed a "running averager" which passes as a crude low-pass filter. The input-output relation is given below:

$$Y_n = \frac{U + Y_{n-1} + Y_{n-2} + \cdots Y_{n-a}}{a+1}$$

|  |  |
|---|---|
| $Y_n$ | -- The current output. |
| $U$ | -- The current input. |
| $a$ | -- Memory length |

I decided not to smooth the two integrated signals; the integrations themselves constitute low-pass filtering. I observed that the dither in the velocity error was noticeably lower after smoothing with memory length of at least five. However the overall performance of the system went down the drain. Evidently the delay induced by the smoothing totally compromised the usefulness of the signal. Perhaps then, since I really only wanted to break the loop (for the noise, not the valid signal) it would have been better to smooth the motor currents instead thereby not altering the velocity data seen by the controller. Once again, the performance was poor... but it improved markedly as the memory length of the filter was reduced towards one. The moral seems to be that simple smoothing is more damaging than helpful. Signals are more attractively displayed but the performance suffers for it. The dismal go-around described above compelled me to remove all invocations of smoothing from the program.

## Conclusions

I believe that a weak link in the results of the actual servoing is the quality of the input signals; particularly of the velocity. I venture that a hardware improvement here would be a great gift. If this isn't possible, there are untested (by me) alternatives. One could employ fancier signal processing methods (such as Kalman filtering) to separate the noise from the signal... though using a Kalman filter entails constructing highly accurate model of the physical system which wouldn't be a simple task.

Accelerometer input would have been very helpful for rapidly determining whether or not the proper trajectory was being followed. It would no longer have been necessary to wait until enough drift had occured to realize that the velocity-noise wasn't causing it. For rapid movement of The Arm I would deem this a crucial input.

One might ease the computational load on the computer by delegating the lowest level velocity servoing to hardware. Since the gains are fixed this change is easily made.

## Suggestions for Further Work

The not-so-distant future will see increasing use of manipulators performing industrial assembly. To this end greater precision will require more rigid construction of the manipulator links if the primary source of positonal feedback are angle sensors in actual joints. However, there is strong evidence to suggest that a compliant medium between the hand and the load offers advantages in close tolerance work. Kawasaki engineers [9] have suceeded in obtaining faster performance of high precision assembly tasks using this method.

Most researchers seem to agree that more capability is needed in the hand to provide an adequate sense of touch. I would venture that some *proximity* sensing would be equally useful. It would be preferable for the arm to be able to roughly sense the presence of an object before actually colliding with it. This sensing would allow The Arm to plan either slowing down to approach a goal object or taking early steps to avoid an obstacle. This might take the form of crude sonar. On a grander scale, the possibility of mounting vision hardware on the Arm itself seems interesting... this would eliminate the present need to parse a visual scene *both* for the location of the arm and the other objects and then difference the information.

## APPENDIX A

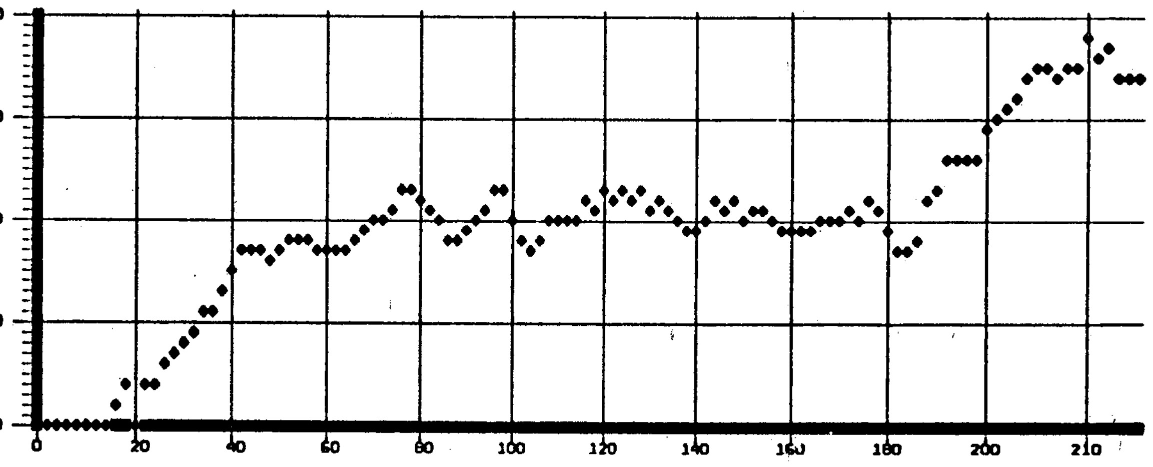

Figure 1 Velocity of Joint Zero.

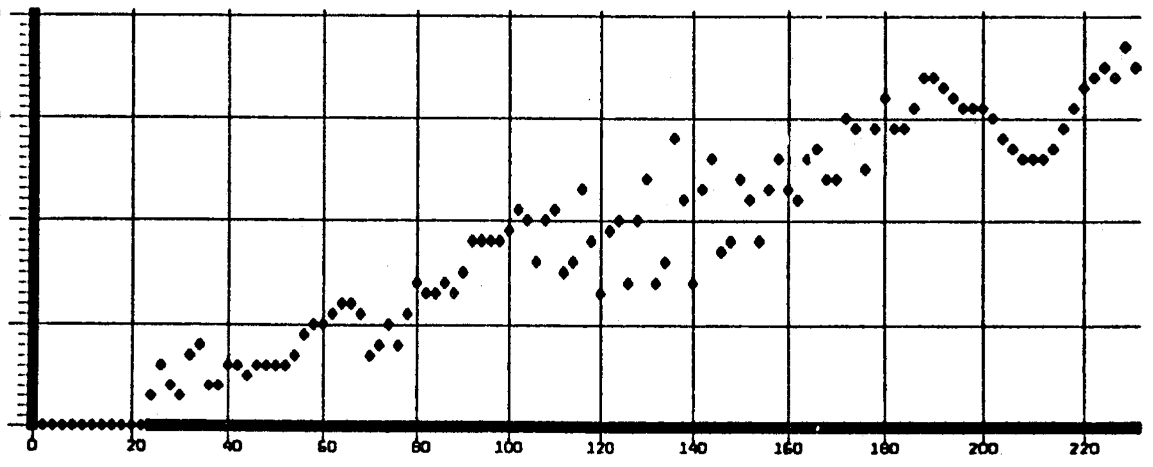

Figure 2 Velocity of Joint One.

**NOTE** All graphs are plotted against time.

All times are quoted in milliseconds.

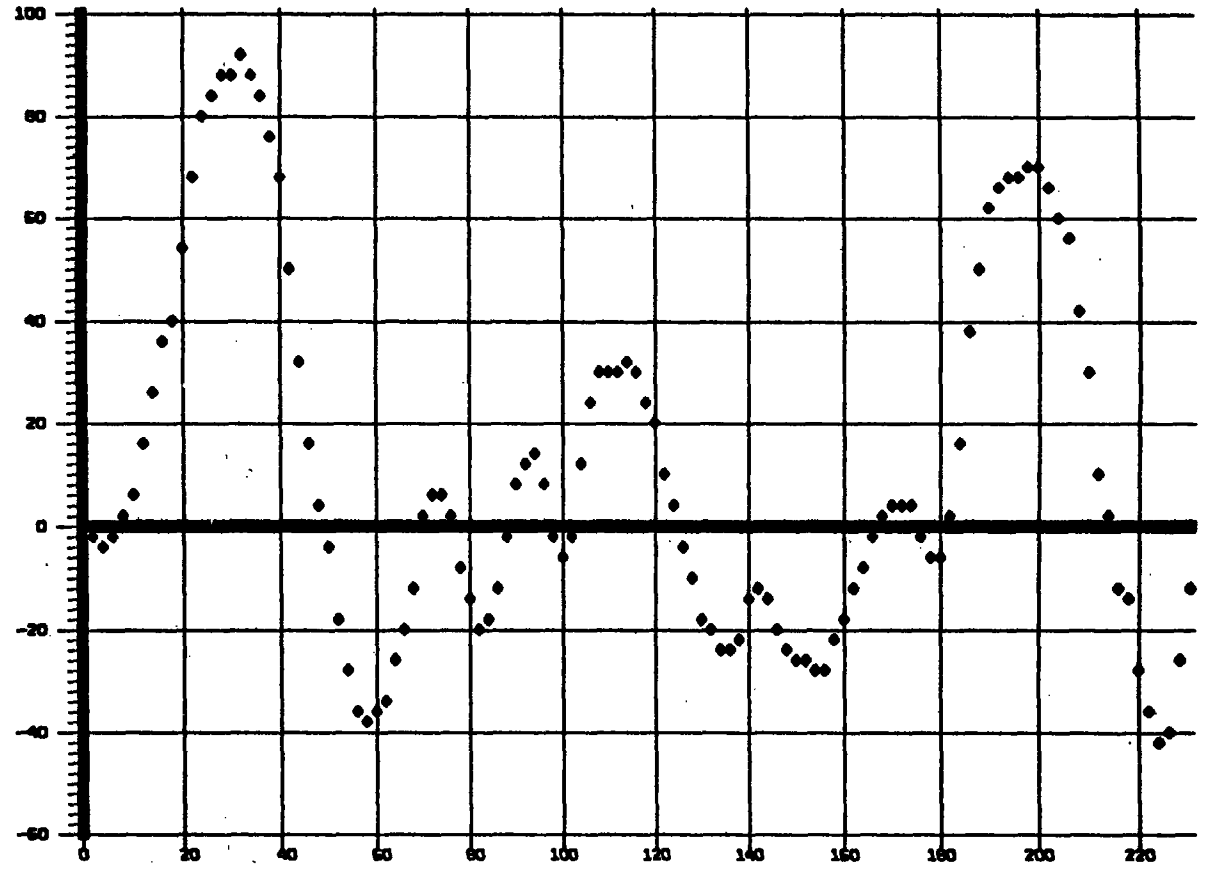

**Figure 3 Cummulative error of Joint Zero.**

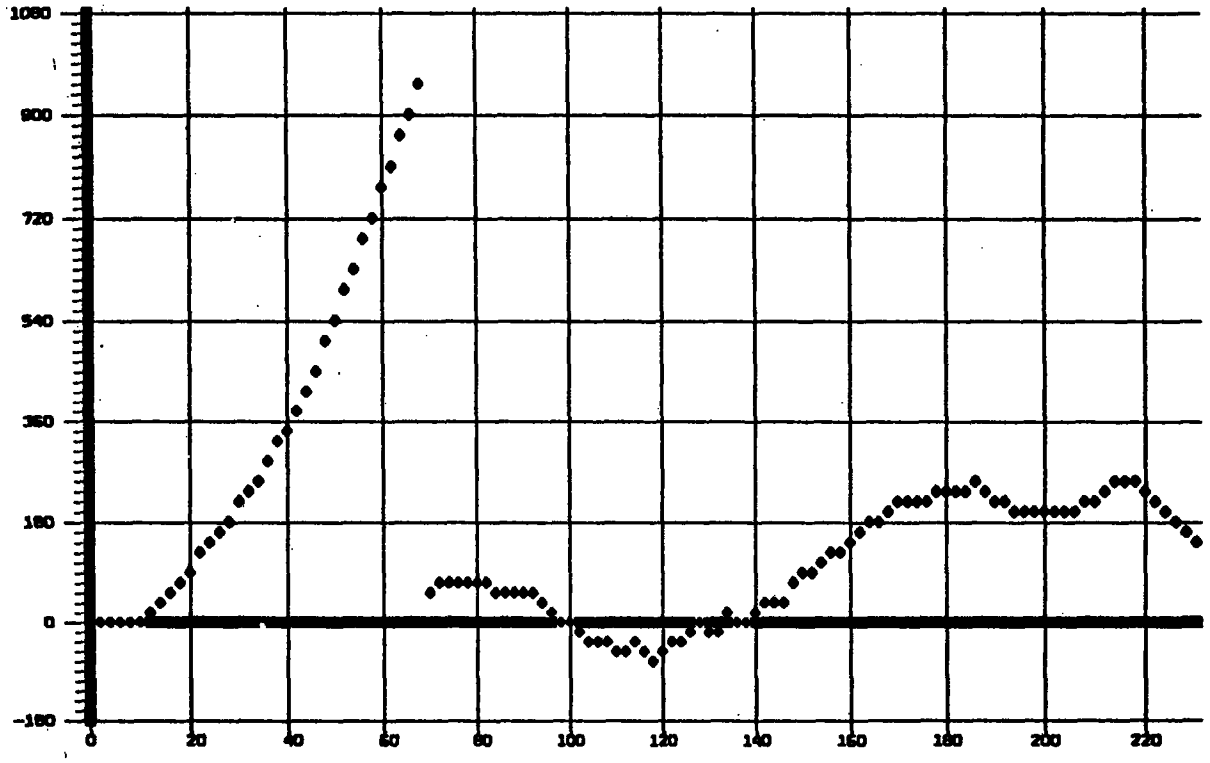

**Figure 4 Cummulative Error of Joint One.**

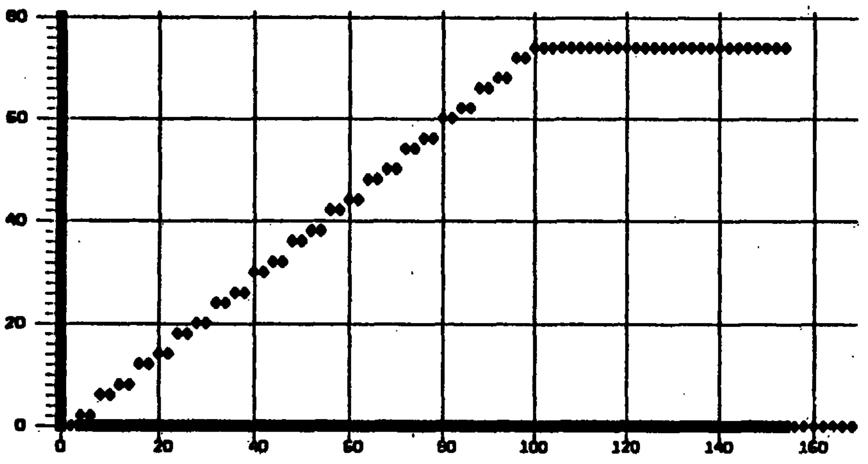

**Figure 5 Minimum Time Tajectory for Joint Zero.**

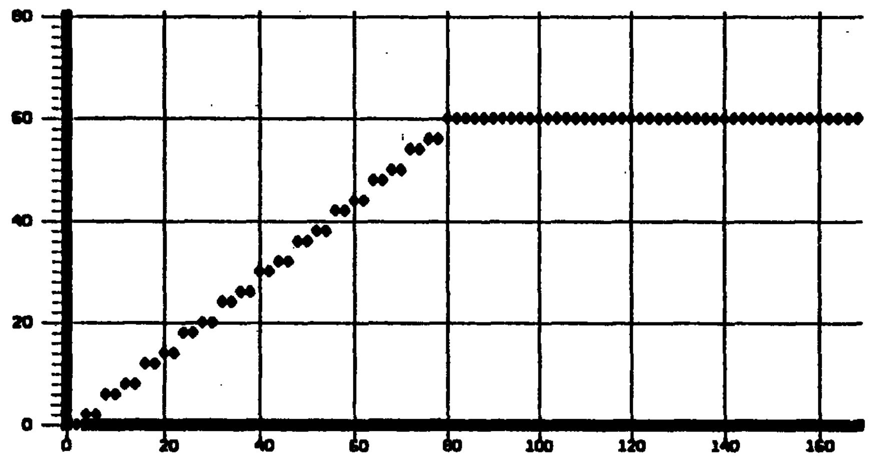

**Figure 6 Minimum Time Trajectory for Joint One.**

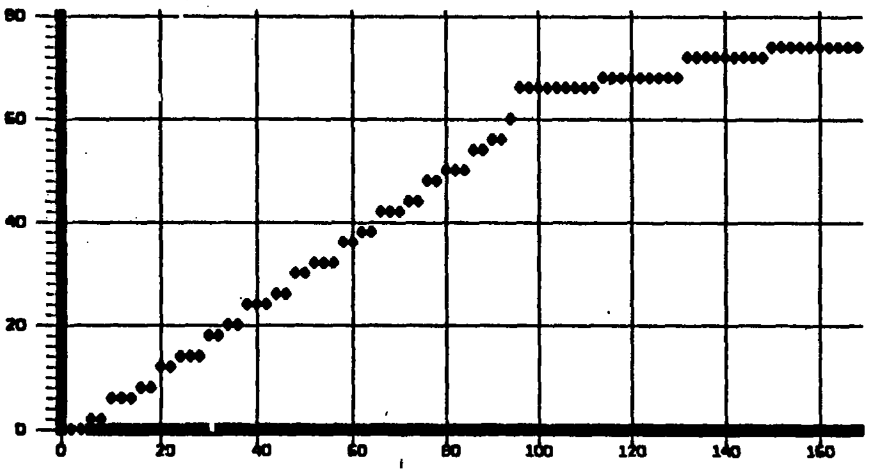

**Figure 7 Actual Trajectory for Joint Zero.**

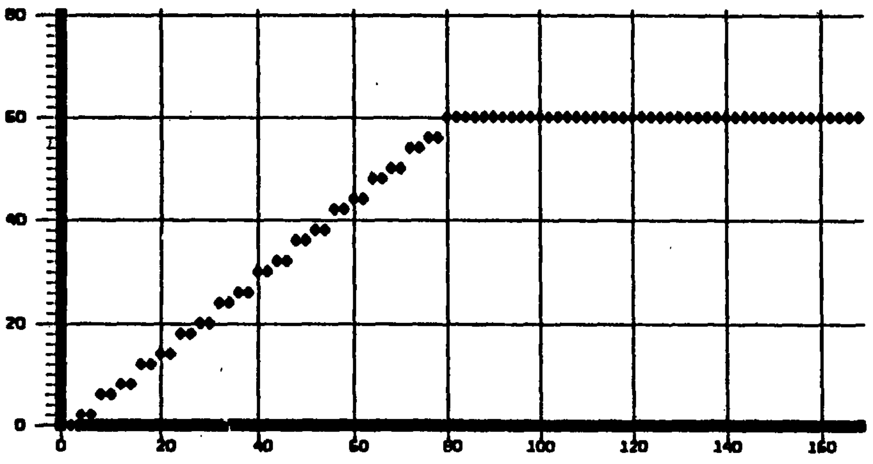

**Figure 8 Actual Trajectory for Joint One.**

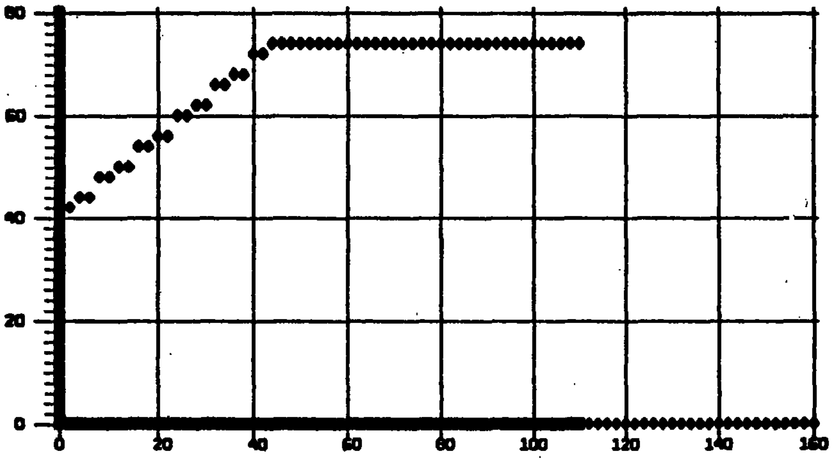

**Figure 9  Minimum Time Trajectory for Joint Zero.**

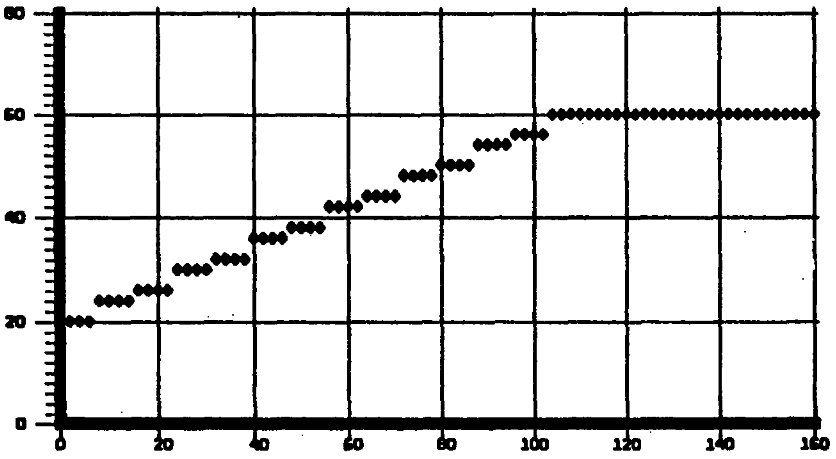

**Figure 10  Minimum Time Trajectory for Joint One.**

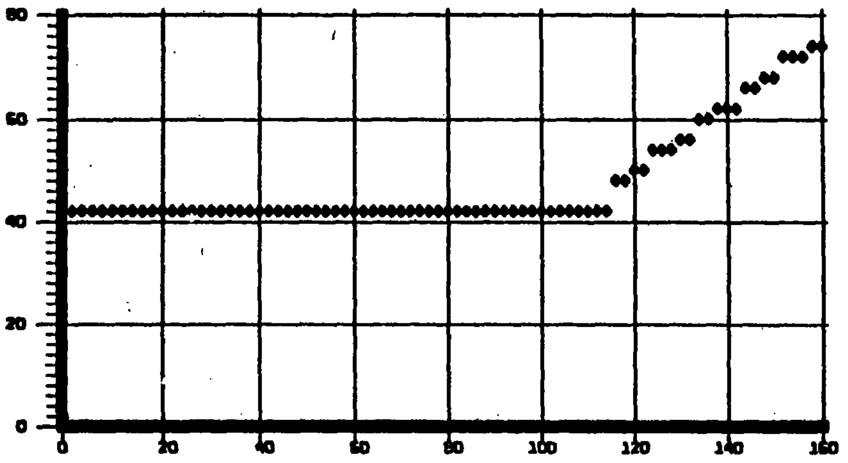

**Figure 11  Current Trajectory for Joint Zero.**

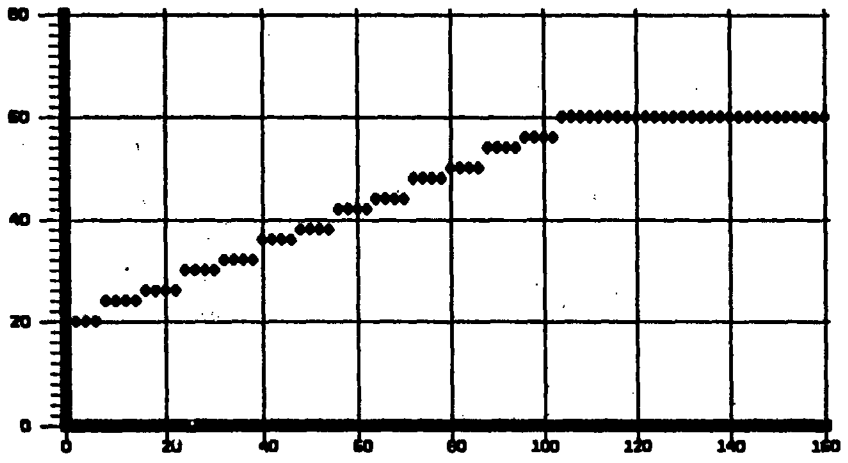

**Figure 12  Current Trajectory for Joint One.**

## BIBLIOGRAPHY

[1] Flatau, Carl R., *Desing outline for Mini-Robot Arms Based on Manipulator Technology.* May 1973.

[2] Horn, B. K. P., *Working Paper 69,* May 1974.

[3] Kahn, Micheal E., *The Near-Minimum-Time Control of Open- Loop Articulated Kinematic Chains,* Stanford AI-Memo 106, Dec 1969.

[4] Kuo, Benjamin C., *Automatic Control Systems* Prentice Hall, Englewood New Jersey c 1975.

[5] Paul, Richard, *Modelling, Trajectory Calculation and Servoing of a Computer Controlled Arm, Stanford AI-Memo 177, November 1972.*

[6] Pieper, Donald L., *The Kinematics of Manipulators under Computer Control.*

[7] Raibert, Marc, *A State Space for Sensorimotor Control and Learning,* MIT AI-memo 351, January 1976.

[8] Scheinman, Victor, *Design of a Computer Controlled Manipulator,* Stanford AI-Memo 92, June 1969.

[9] Symposium Proceedings, *The Fifth Annual Symposium on Industrial Robotics,* 1975.

[10] Waters, Richard, *Vision Flash 42,* March 1973.