TIME-OPTIMAL CONTROL OF AN OPTICAL-MEMORY INFORMATION RETRIEVAL SYSTEM

by

Abdul Mohammed Javery

This work was supported by the Joint Services Electronics Program (U.S. Army, U.S. Navy and U.S. Air Force) under Contract DAAB-07-72-C-0259.

TIME-OPTIMAL CONTROL OF AN OPTICAL-MEMORY INFORMATION RETRIEVAL SYSTEM

BY

ABDUL MOHAMMED JAVERY

B.S., University of Illinois (Chicago Circle), 1976

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1977

Thesis Adviser: Professor William R. Perkins

Urbana, Illinois

ACKNOWLEDGEMENT

The author would especially like to thank his advisor Professor W. R. Perkins for his encouragement, interest and understanding. Special thanks to Mr. John Squire for his help in defining the control problem and to Mr. S. H. Javid for helping the author understand the time-optimal control iterative procedure.

TABLE OF CONTENTS

LIST OF FIGURES

Figure Page

CHAPTER 1

INTRODUCTION

1.1 The Overall System.

In recent years a significant amount of work has been done in the field of optics. These advances has made it possible to use optics as a communication medium. One apparent application is seen in the video-disc systems presently being developed by the television industry. Recently some interest has evolved in applying the video-disc technology towards an optical memory device having a large information storage capability.

A memory device of this kind is presently being developed at the Coordinated Science Laboratory at the University of Illinois, Urbana. The storage medium for this system is a mylar disc coated with a thin film of bismuth. Information is stored in 4096 concentric tracks on the disc where the metallic film has been selectively removed. This memory device has an expected storage capacity of 10^8 to 10^{10} bits of information per disc. The laser optics serves as a communication medium between the disc and the optical sensor, which senses the presence or absence of metal in accordance with the information stored. Fig. 1.1 shows a schematic diagram of the system. The laser source is fixed and the laser beam, (after passing through two lenses for proper convergence), is reflected off a mirror onto the

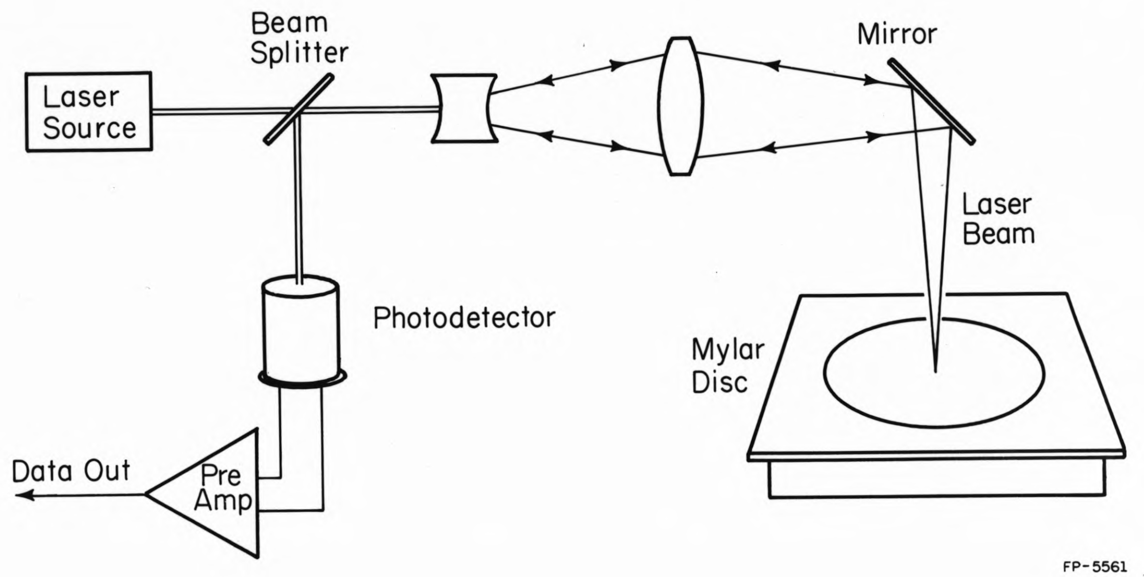

Fig. 1.1 The Overall System.

disc. If the metallic film is present at the region, (a circular region of a radius of about 5μm where the beam is focussed), the beam will be reflected back and sensed by the optical sensor. The disc rotates at 1800 rpm so that any portion of the disc can be scanned and the information retrieved by merely deflecting the mirror by an appropriate amount. The mirror itself is mounted on the arm of a moving iron galvanometer.

The purpose of this work is to design a time-optimal control algorithm to position the mirror so as to focus the laser beam on any desired track for information retrieval. The application of this optical memory device can be varied, anywhere from replacing the conventional disc drives to storing audio messages for air traffic control.

1.2 The System Model.

The mechanism used to deflect the mirror is a moving Iron Galvanometer, with the mirror being mounted on its arm. The galvanometer used is a commercially available General Scanning G-300PD Model. The plant consists of an iron vane rotor supported by a torsion spring immersed in a permanent magnetic field. Windings are used to produce a bias field such that the torque generated is proportional to the current in the drive coils. A schematic diagram of the plant is given in Fig. 1.2.

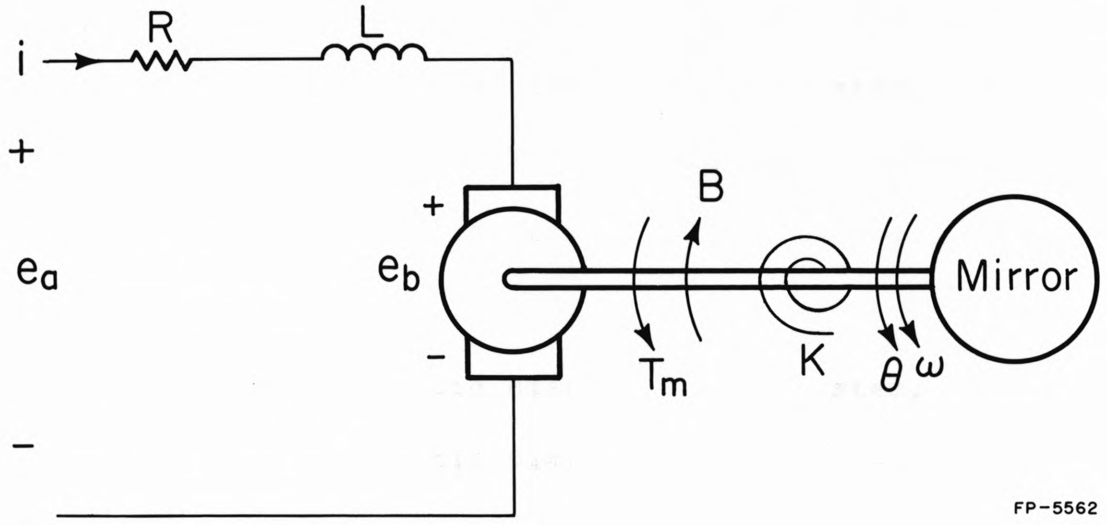

Fig. 1.2 Schematic Diagram of the System.

Let

R	=	armature resistance
L	=	armature inductance
i	=	armature current
e_a	=	applied voltage
e_b	=	back emf
θ	=	angular position
ω	=	angular velocity
B	=	back emf constant
J	=	total inertia
B_m	=	bearing (viscous) friction coefficient
K_m	=	motor torque constant
K	=	torsion spring constant.

Refering to the schematic diagram of Fig. 1.2, the variables $i(t)$, $\theta(t)$ and $\omega(t)$ are assigned as the states of the system. Writing Kirchhoff's voltage law around the loop

$$L\frac{di}{dt} = -Ri - e_b + e_a \ .$$

The back emf voltage is proportional to the motor speed,

$$e_b = B\omega \ .$$

Therefore

$$\frac{di}{dt} = \frac{-Ri}{L} - \frac{B\omega}{L} + \frac{e_a}{L} \ . \qquad (1.1)$$

Assuming the spring to be linear, the torque-spring relationship is given by

$$T_s = K\theta .$$

The torque constant K_m relates the motor torque T_m and the armature current $i(t)$ by

$$T_m = K_m i(t) .$$

The torque T_f produced due to the viscous friction is proportional to the motor speed,

$$T_f = B_m \omega .$$

The algebraic sum of the torques must be equal to the product of the total inertia and motor acceleration,

$$J\frac{d\omega(t)}{dt} = K_m i(t) - K\theta(t) - B_m\omega(t)$$

or

$$\frac{d\omega(t)}{dt} = \frac{K_m i(t)}{J} - \frac{K\theta(t)}{J} - \frac{B_m\omega(t)}{J} . \qquad (1.2)$$

The third state equation is simply

$$\frac{d\theta(t)}{dt} = \omega(t) . \qquad (1.3)$$

Defining

$$x_1 = i(t)$$
$$x_2 = \theta$$
$$x_3 = \omega \qquad \text{and} \qquad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} .$$
$$e_a = u(t)$$

Equations (1.1) - (1.3) can be written in the form of state equations:

$$\dot{x} = \begin{bmatrix} -R/L & 0 & -B/L \\ 0 & 0 & 1 \\ K_m/J & -K/J & -B_m/J \end{bmatrix} x + \begin{bmatrix} 1/L \\ 0 \\ 0 \end{bmatrix} u \quad . \qquad (1.4)$$

The values of the system parameters are obtained from the specifications provided by the galvanometer manufacturer (General Scanning Co.). The sytem parameters are:

$R = 8.0$ Ohms

$B = 0.166$ V-Sec/rad

$K_m = 0.14$ Nt-m/A

$K = 0.28$ Nt-m/rad

$B_m = 3.39\text{x}10^{-5}$ (Nt-m)/(rad/sec)

$J = 9.8\text{x}10^{-7}$ Kg-m^2

$L = 0.03$ H

Substituting the values for the system parameters, the state equation (1.4) becomes:

$$\dot{x} = \begin{bmatrix} -266.667 & 0 & -5.5333 \\ 0 & 0 & 1 \\ 142{,}860 & -285{,}710 & -34.592 \end{bmatrix} x + \begin{bmatrix} 33.333 \\ 0 \\ 0 \end{bmatrix} u \quad . \qquad (1.5)$$

The state diagram of the system is as shown in Fig. 1.3. The transfer function between the motor displacement and the input voltage is obtained from the state diagram as:

$$\frac{\theta(s)}{U(s)} = \frac{4.761995\times10^{6}}{s^{3} + 3.012\times10^{3}s^{2} + 1.085\times10^{7}s + 7.619\times10^{8}}$$

The transfer function has no zeroes and has the three poles located at $\lambda_1=-71.2696$, $\lambda_{2,3}=-114.99\pm j1027.5$. The system is basically open loop except for the feedback loop caused by the back emf (Fig. 1.3). The back emf effect is equivalent to an 'electrical friction', which tends to improve the stability of the system.

Fig. 1.4 shows the natural time response of the system with an initial condition of x(0)=[0.2 0.1 0]'. The plot for the state x_1 (current) shows that the decay in the current is not pure exponential but it is affected by the oscillatory modes in the system. The oscillations in x_1 are due mainly to the back emf voltage. A comparison of the plots for x_1 and x_2 (current & position) show an almost linear relationship. The constant relating the two is about 0.5 rad/A,

$$x_2(t) = 0.498x_1(t) \tag{1.6}$$

The phase-plane trajectory of the system in the x_2-x_3 plane is shown in Fig. 1.5. Again the effect of back emf voltage is evident. As the velocity increases so does the

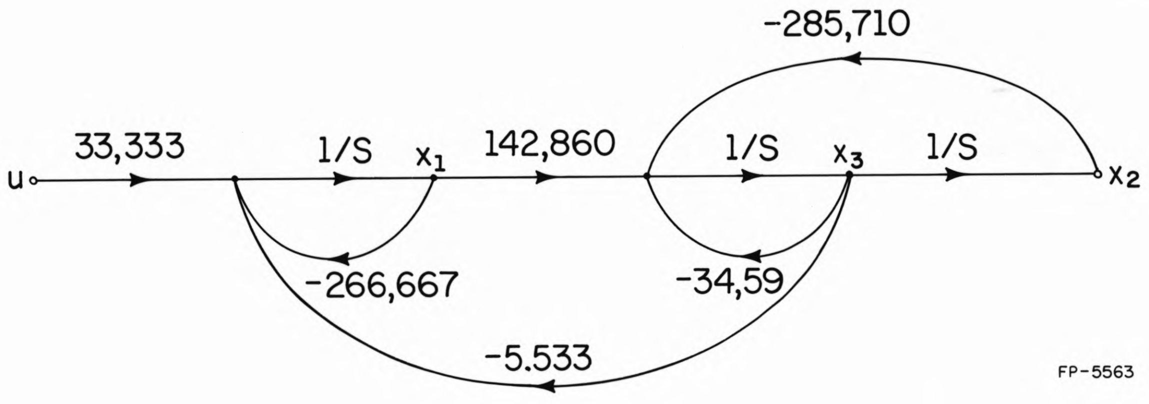

Fig. 1.3 State Diagram of the System.

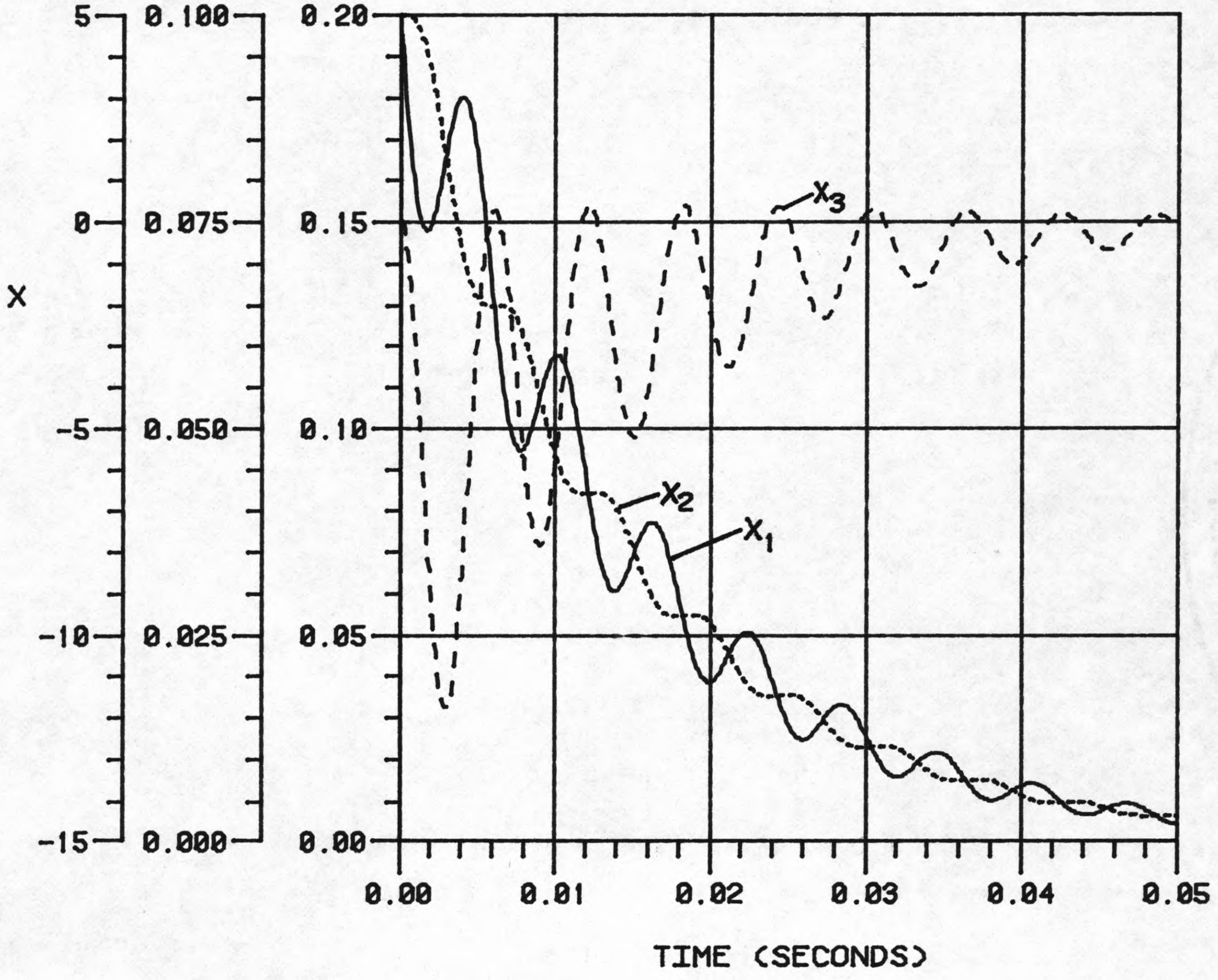

Fig. 1.4 Natural Time Responses of the System States.

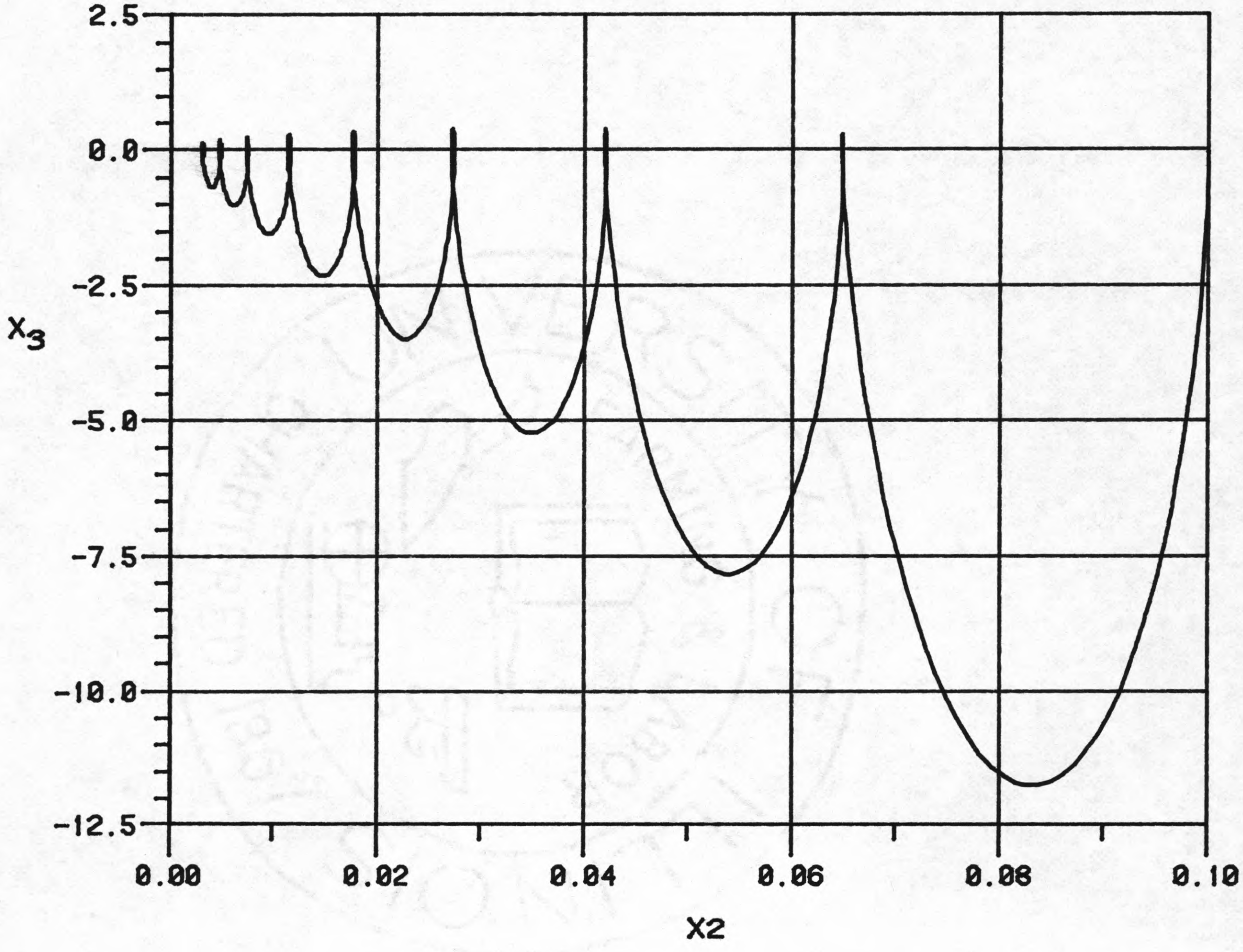

Fig. 1.5 Phase-Plane Trajectory in (x_2,x_3) Plane.

back emf voltage which tends to decrease the velocity. The back emf voltage gradually increases and at a point exceeds the control voltage in magnitude (but having opposite sign), rendering the control ineffective and decreasing the velocity to zero. With the decrease in velocity, the back emf voltage also decreases and eventually becomes so small that the control voltage becomes effective again. This cycle is repeated as the state x_2 (position) moves towards the origin.

1.3 Discussion of the Problem.

In the past it was attempted [1] to solve this problem of positioning the laser beam by neglecting the inductance in the plant and thus considering it to be a second order system instead of third order. It was assumed in [1] that the control for travelling absolute distances is the same, thus having the same switching curve, but merely shifting the coordinates for different final points. This assumption was found to be incorrect. The switching curve is seen to be final-point dependent. So the problem of designing a controller for the optical memory is essentially unsolved in [1], but an appreciable amount of insight to the system was gained through the work done in [1].

In this thesis, the basic approach taken in solving the third order problem is to apply the ideas of singular perturbation. But from the system eigenvalues it is evident that the system does not really possess a "two time scale property" (the system eigenvalues are not very widely separated), so that the theory of singular perturbation e.g. in [2] or [3], is not directly applicable. However, an iterative procedure, proposed in [4], based on the theory of singular perturbation can be used with some modifications.

An unusual property of the system is that the electrical mode (current) is slower than the mechanical modes (position and velocity). This adds to the complexity of the problem because it is required to control the fast modes of the system through its slow mode.

CHAPTER 2
TIME-OPTIMAL CONTROL

2.1 The Iterative Procedure.

The basic idea involved in the iterative procedure, as proposed in [4], is to decouple the higher order system into 'slow' and 'fast' subsystems (subsystems having slow and fast eigenvalues) and then try to control them separately. The 'slow states', x_s(states of the slow subsystem), are first steered to their final values and then the 'fast states', x_f(states of the fast subsystem), are rapidly steered to their desired final values. If the eigenvalues of the subsystems are widely separated then the slow states would not move away from its final position, while the fast states are being steered. But if the separation is not wide enough then the slow states would drift away (from their final values) while trying to steer the fast states.

The algorithm proposed in [4] gives a method to iteratively find a point x_{si} by integrating the slow subsystem backwards from its desired final value using the 'fast control'(control required to steer the fast subsystem to its final value). Now the slow states are taken to this calculated point x_{si} (instead of their exact final values). If the fast states are then steered, the slow states would automatically drift to their final values under the influence of the fast control.

In this procedure the fast states, x_f, are at their desired positions but the slow states, x_s, are somewhat drifted from their final values. Applying the procedure iteratively, the slow states move towards their desired final values in steps and eventually converges to their final values. The algorithm may fail to converge if the steps taken are too large. To assure convergence, the algorithm can be modified to reduce the steps taken by the slow states by multiplying them by a number s, where $0<s<1$. The value of s is picked in such a way that there is no divergence but at the same time convergence is achieved in an acceptable number of iterations. Note that making s arbitrarily small would considerably decrease the rate of convergence. There is no closed form formula available to calculate the value of s. But, experimenting with the system it was found that the value of s does not vary rapidly so that s can be taken as constant over some ranges of initial and final conditions. The value of s for the different ranges of the system was found by trial and error.

2.2 Linear Transformation

To use the iterative procedure the system has to be decoupled into slow and fast subsystems. A Jordan Transformation is applied here. Rewriting the system equations (1.5)

$$\dot{x} = \begin{bmatrix} -266.667 & 0 & -5.5333 \\ 0 & 0 & 1 \\ 142,860 & -285,710 & -34.592 \end{bmatrix} x + \begin{bmatrix} 33.333 \\ 0 \\ 0 \end{bmatrix} u \ . \quad (2.1)$$

Let x = Tz where T is the transformation matrix and z is the transformed state vector, so that

$$\dot{z} = T^{-1}ATz + T^{-1}Bu.$$

The transformation matrix T is found to be

$$T = \begin{bmatrix} 0.18697\text{x}10^{7} & 0.82075\text{x}10^{6} & -0.20646\text{x}10^{6} \\ 0.92636\text{x}10^{6} & -0.13451\text{x}10^{6} & 0.74260\text{x}10^{5} \\ -0.66023\text{x}10^{8} & -0.60836\text{x}10^{8} & -0.14675\text{x}10^{9} \end{bmatrix} . \quad (2.2)$$

The Jordan form of system (2.1) is

$$\dot{z}_1 = -71.2695z_1 + 0.486\text{x}10^{-5}u \quad (2.3a)$$

$$\begin{aligned} \dot{z}_2 &= -114.99z_2 + 1027.5z_3 + .26254\text{x}10^{-4}u \\ \dot{z}_3 &= -1027.5z_2 - 114.99z_3 - 0.1307\text{x}10^{-4}u \ . \end{aligned} \quad (2.3b)$$

Thus the system is decoupled into slow subsystem (2.3a) and fast subsystem (2.3b). The general approach is to control the slow subsystem first and steer it to some desired point (to z_{1f}, if the system has widely separated eigenvalues) and then to control the fast subsystem. The problem here is to iteratively find a point $z_{1i}(t_s)$, (intermediate z_1 at time t_s), depending on z_{1f} (final z_1) so that z_1 is steered to z_{1i} instead of z_{1f}. Now if the 'fast states' (z_2 and z_3) are steered to their final values (z_{2f} and z_{3f}), the slow state (z_1) would automatically drift from z_{1i} to z_{1f} under the influence of the fast control u_f.

2.3 <u>The Slow Subsystem.</u>

The equation for the slow subsystem is

$$\dot{z}_1 = -71.2696 z_1 + 0.486\text{x}10^{-5} u \ . \tag{2.4}$$

The performance index to be minimized is

$$J = \int_0^{t_f} dt \qquad \text{with} \quad |u| \leq k \ .$$

The Minimum Principle is applied to find the time-optimal control. The Hamiltonian is

$$H = 1 - 71.2696 z_1(t) p_1(t) + 0.486\text{x}10^{-5} u(t) p_1(t) \ .$$

The costate $p_1(t)$ satisfies the equation

$$\dot{p}_1(t) = 71.2696 p_1(t) \ . \tag{2.5}$$

The control u(t) that minimizes the Hamiltonian is given by

$$u(t) = u_s(t) = -\text{Sgn}\{p_1(t)\} * k \ .$$

Solving (2.5)

$$p_1(t) = p_1(0)e^{71.2696t} . \tag{2.6}$$

From (2.6) it is seen that, since the exponential is always positive, $p_1(t)$ does not change sign and has the same sign as that of $p_1(0)$. This implies that for the slow subsystem there is no switching. It can be shown [5] that for this first order system the optimal u is given by

$$u_s = -\mathrm{Sgn}\{z_1(0)-z_1(t_f)\}*k$$

Thus the control law is

$$\begin{aligned}&\text{If } z_1(0) < z_1(t_f), \text{ then } u_s = k.\\ &\text{If } z_1(0) > z_1(t_f), \text{ then } u_s = -k.\\ &\text{If } z_1(0) = z_1(t_f), \text{ then } u_s = 0.\end{aligned} \tag{2.7}$$

Solving (2.4) for $z_1(t)$ gives

$$z_1(t) = 6.8192\text{x}10^{-8}u+(z_1(0)-6.8192\text{x}10^{-8}u)e^{-71.2696t} . \tag{2.8}$$

Equation (2.8) can be used to find the intermediate point $z_1(t_s)$, for the iterative procedure. Thus

$$z_{1i}(t_s)=6.8192\text{x}10^{-8}u_f+(z_1(t_f)-6.8192\text{x}10^{-8}u_f)e^{-71.2696t} . \tag{2.9}$$

In the above equation u_f is fast control (control required for the fast subsystem), $z_1(t_f)$ is the desired final point and t should be substituted by the negative of the time for which the fast control was applied in the previous

iteration. Note that if the value of z_{1i} being calculated is for the (n+1)th iteration then the control and time (u_f and t) used is from the nth iteration.
Also from (2.8)

$$t_s = -\ln[a_1/a_2]/71.2696 \tag{2.10}$$

where

$$a_1 = z_{1i}(t_s) - 6.81918u_s$$
$$a_2 = z_1(0) - 6.81918u_s$$

Thus the time required to reach any given intermediate point can be calculated using Eq. (2.10). This then is the time for which the slow control is to be applied for the next iteration. Thus for each iteration the new intermediate point $z_{1i}(t_s)$ is given by Eq. (2.9) and the time t_s is given by Eq. (2.10).

2.4 The Fast Subsystem

2.4.1 Application of the Minimum Principle.

The state equations for the fast subsystem are

$$\dot{z}_2 = -114.99z_2 + 1027.5z_3 + 0.2625\times10^{-4}u$$
$$(2.11)$$
$$\dot{z}_3 = -1027.5z_2 - 114.99z_3 - 0.1307\times10^{-4}u \ .$$

This second order system is a stable but very lightly damped harmonic oscillator. It has a natural frequency of 1033.9 rad/sec and a damping coefficient of 0.1112. A detailed analysis and the time-optimal control of a system of this type is given in [5]. Using polar coordinates and eliminating time, the equation for the phase-plane trajectories is found to be

$$R = R_0 e^{[(\theta-\theta_0)/8.9356]} \tag{2.12}$$

where

$$R = \sqrt{(z_2-z_{2e})^2 + (z_3-z_{3e})^2}$$

$$R_0 = \sqrt{(z_{20}-z_{2e})^2 + (z_{30}-z_{3e})^2}$$

$$\theta = \mathrm{Cos}^{-1}((z_2-z_{2e})/R)$$

$$\theta_0 = \mathrm{Cos}^{-1}((z_{20}-z_{2e})/R_0)$$

and where (z_{20},z_{30}) and (z_{2e},z_{3e}) are the initial and equilibrium points, respectively.

Eq.(2.12) describes a family of logarithmic spirals tending to the equilibrium point. The calculations involved in deriving the equation of the phase-plane trajectory are given in Appendix A. Fig. 2.1 shows the phase-plane trajectories of the system in the transformed z_2-z_3 plane with no control on the system. The Minimum Principle is utilized again to find the time-optimal control for the second order subsystem.

Rewriting the transformed state equations (2.11)

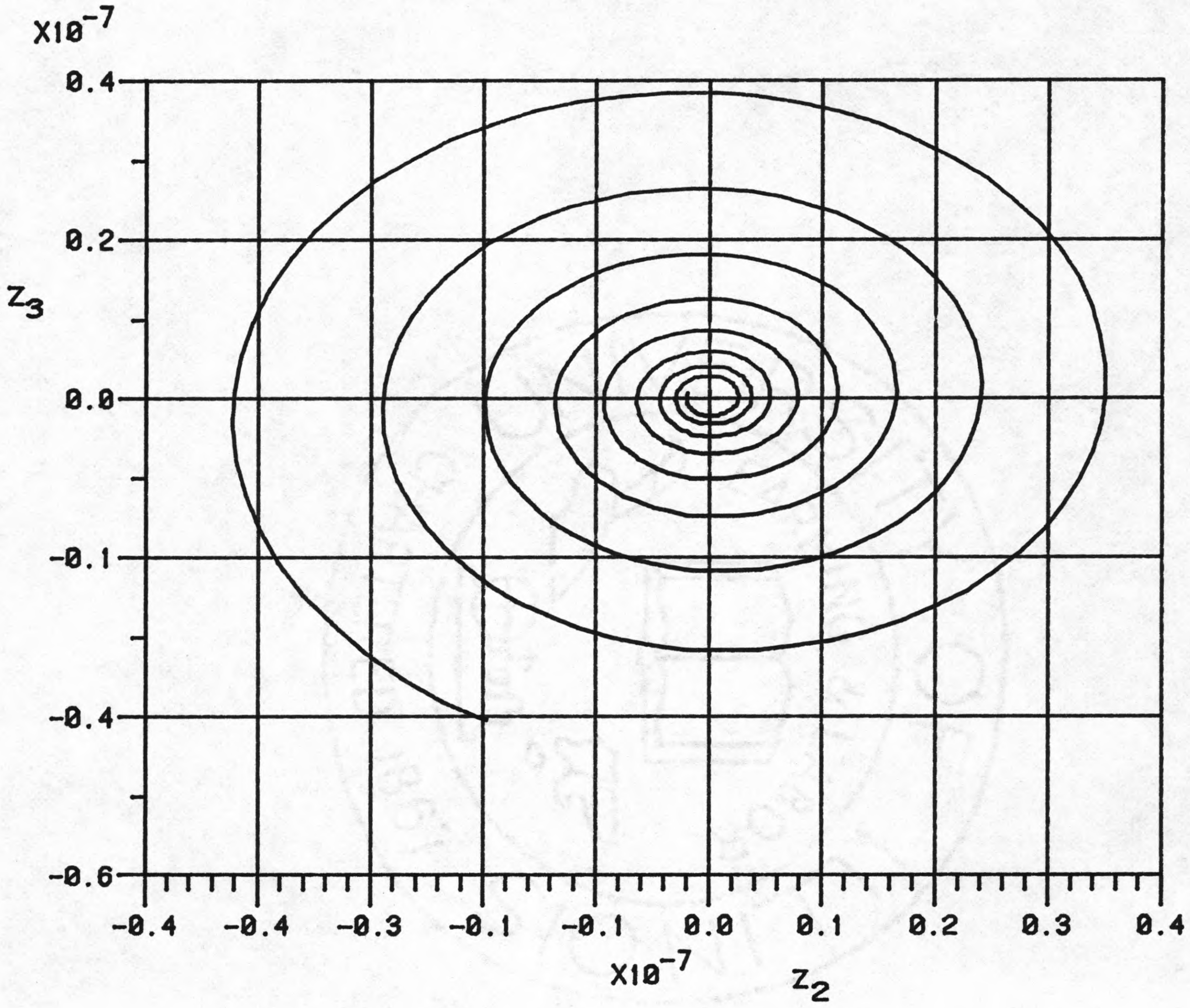

Fig. 2.1 Phase-Plane Trajectory in the Transformed (z_2, z_3) Plane.

$$\dot{z}_2 = -114.99z_2 + 1027.5z_3 + 0.2625\text{x}10^{-4}u$$

$$\dot{z}_3 = -1027.5z_2 - 114.99z_3 - 0.1307\text{x}10^{-4}u \ . \qquad (2.13)$$

The Hamiltonian is

$$H = 1 - 114.99z_2p_2 + 1027.5z_3p_2 + 0.2625\text{x}10^{-4}up_2 - 1027.5z_2p_3 - 114.99z_3p_3 - 0.1307\text{x}10^{-4}up_3 \ .$$

The control that minimizes the Hamiltonian is

$$u(t) = -\text{Sgn}\{0.2625\text{x}10^{-4}p_2(t) - 0.1307\text{x}10^{-4}p_3(t)\}*k \ .$$

The costate variables $p_2(t)$ and $p_3(t)$ satisfy the following differential equations

$$\dot{p}_2 = 114.99p_2 + 1027.5p_3$$

$$\dot{p}_3 = -1027.5p_2 + 114.99p_3 \ . \qquad (2.14)$$

The transition matrix for system (2.14) is

$$\Psi(t) = e^{114.99t}\begin{bmatrix} \text{Cos}\alpha t & \text{Sin}\alpha t \\ -\text{Sin}\alpha t & \text{Cos}\alpha t \end{bmatrix}$$

where $\alpha = 1027.5$.

The solution for (2.14) is given by

$$p_2(t) = e^{114.99t}(p_{20}\text{Cos}\alpha t + p_{30}\text{Sin}\alpha t)$$

$$p_3(t) = e^{114.99t}(p_{30}\text{Cos}\alpha t - p_{20}\text{Sin}\alpha t)$$

where

$$p_0 = p(0) .$$

A plot of the function $(0.2625 \times 10^{-4} p_2(t) - 0.1307 \times 10^{-4} p_3(t))$ versus time and the corresponding control u(t) is shown in Fig. 2.2. Note that the sign of the control u(t) depends on the sign of this function. From the figure it is seen that the control cannot remain constant for more than $114.99\pi/1027.5$ secs. and also that there could be an infinite number of switchings.

2.4.2 Switching Curves and Controls.

The final point is any allowable point including the origin. The shape of the switching curve varies with the final point. Due to the linear relationship (Eq. 1.6) between the system states x_1 and x_2, the transformed initial and final points will always lie on a straight line in the z_2-z_3 plane given by

$$z_3 = 2.86586 z_2 . \tag{2.15}$$

Eq. (2.15) is the equation of a line that passes through the origin and the two equilibrium points (due to $u = \pm 5$). Furthermore the initial and final states are constrained to the portion of the straight line between the two equilibrium points. This implies that there could be at most one switching. A more detailed description is given in Appendix B. The equation of the switching curve is

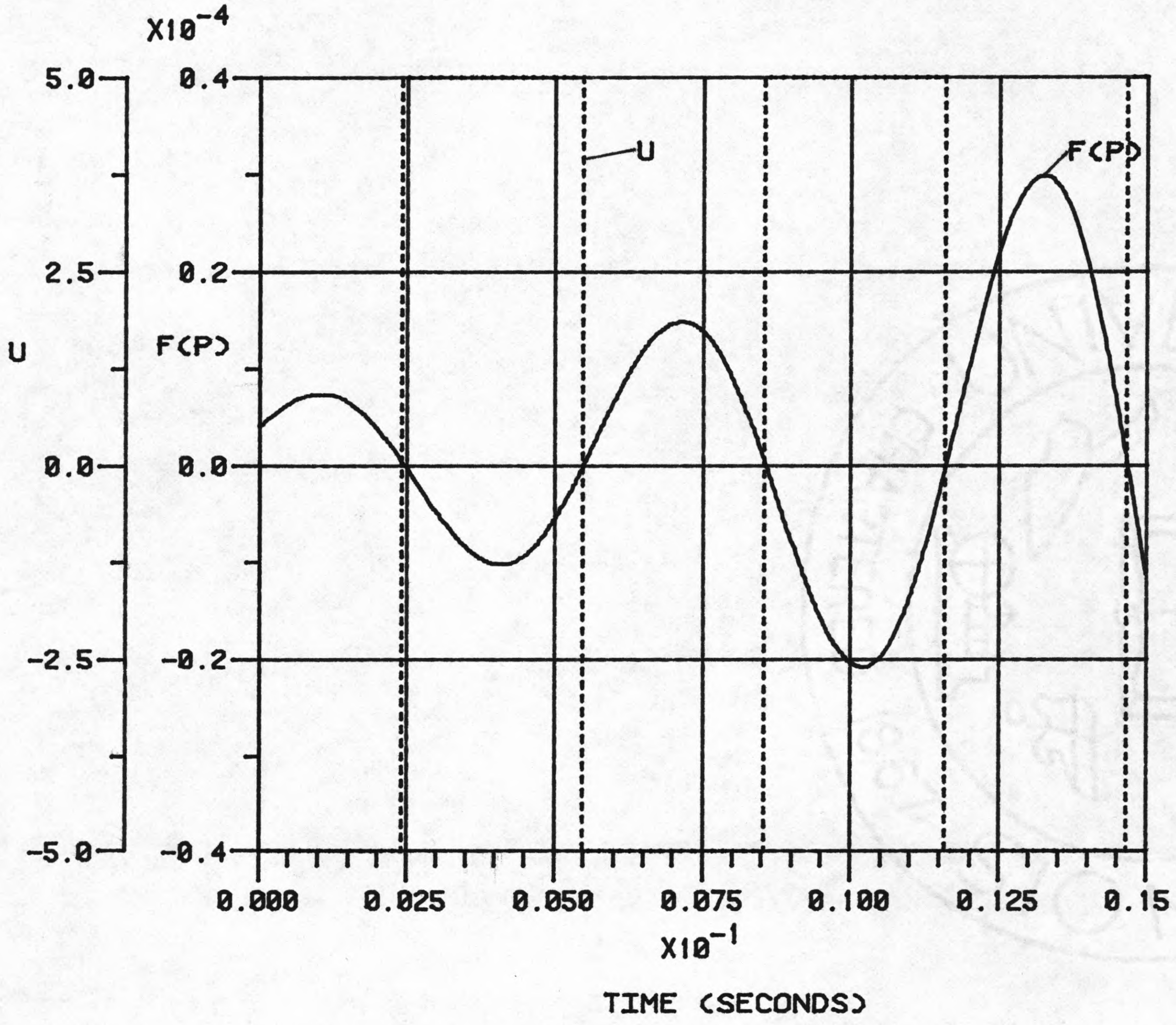

Fig. 2.2 Function of the Costate Variable f(p) and the Corresponding Control u(t) Vs. Time.

$$R_s = R_{s0} * \exp[(\theta_s - \theta_{s0})/8.9356] \tag{2.16}$$

where

$$R_s = \sqrt{(z_2 - z_{2e})^2 + (z_3 - z_{3e})^2}$$

$$R_{s0} = [\sqrt{(z_{2f} - z_{2e})^2 + (z_{3f} - z_{3e})^2}\,] * \exp(\pi/8.9356)$$

$$\theta_s = \text{Cos}^{-1}((z_2 - z_{2e})/R_s)$$

$$\theta_{s0} = 1.22 \;.$$

The switching curves are obviously final point dependent as can be seen from Eq. (2.16). Figs. 2.3(a) & 2.3(b) show typical switching curves for track 1100 and 3600 as the final points respectively. Thus the switching curves consist of exactly two "semiloops", one on each side of the final point, which are portions of the phase-plane trajectory. Any point on the switching curve can, therefore, be taken to the final point by the control u=+5V or u=-5V.

The time-optimal control, as a function of the transformed states (z_2 & z_3), is given by

If (z_2, z_3) is above the switching curve, then $u^* = u_f = +k$.
If (z_2, z_3) is below the switching curve, then $u^* = u_f = -k$.
If (z_2, z_3) is on the left half of the switching curve, then $u^* = u_f = +k$.
If (z_2, z_3) is on the right half of the switching curve, then $u^* = u_f = -k$.

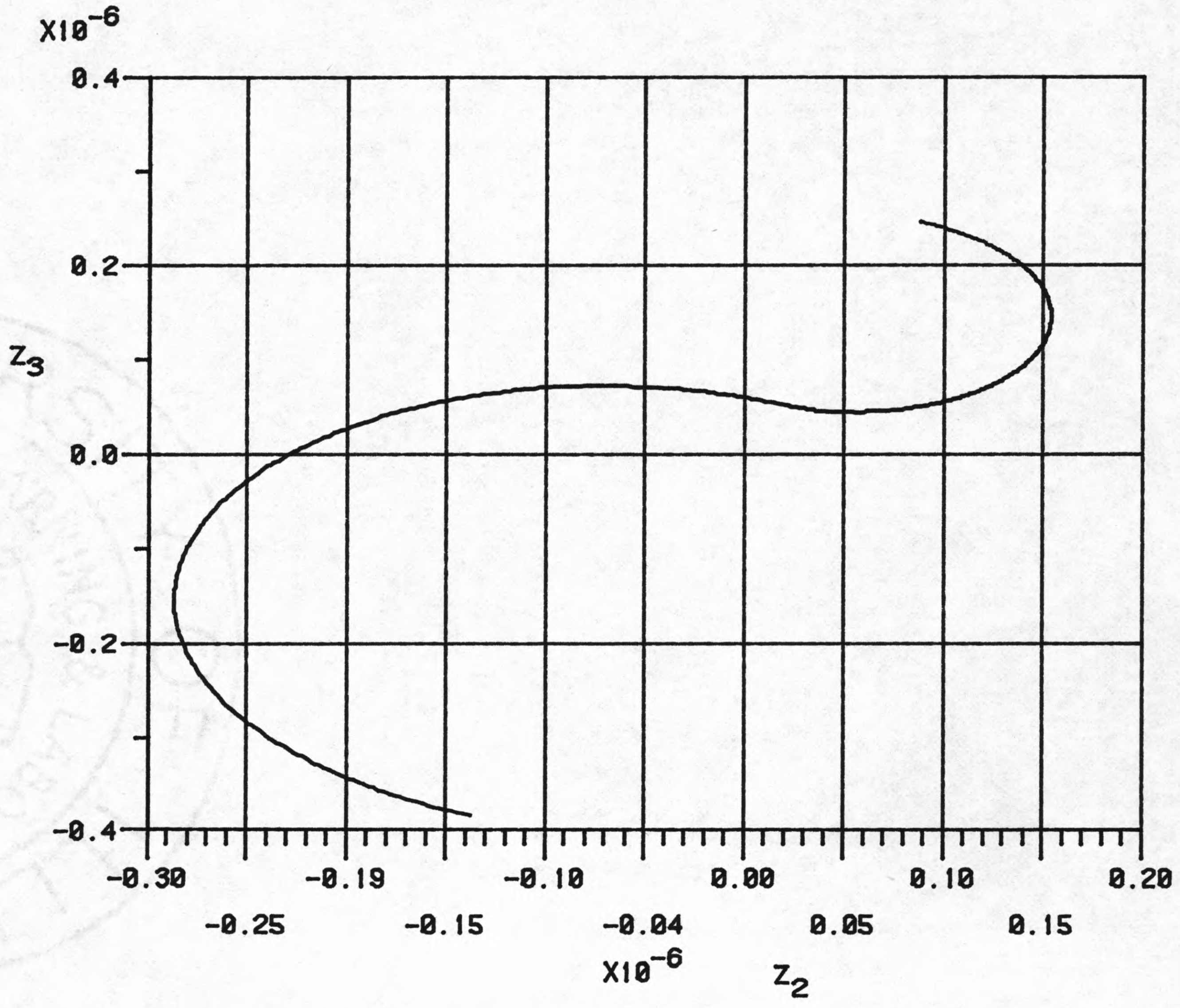

Fig. 2.3(a) Switching Curve for Final Track = 1100.

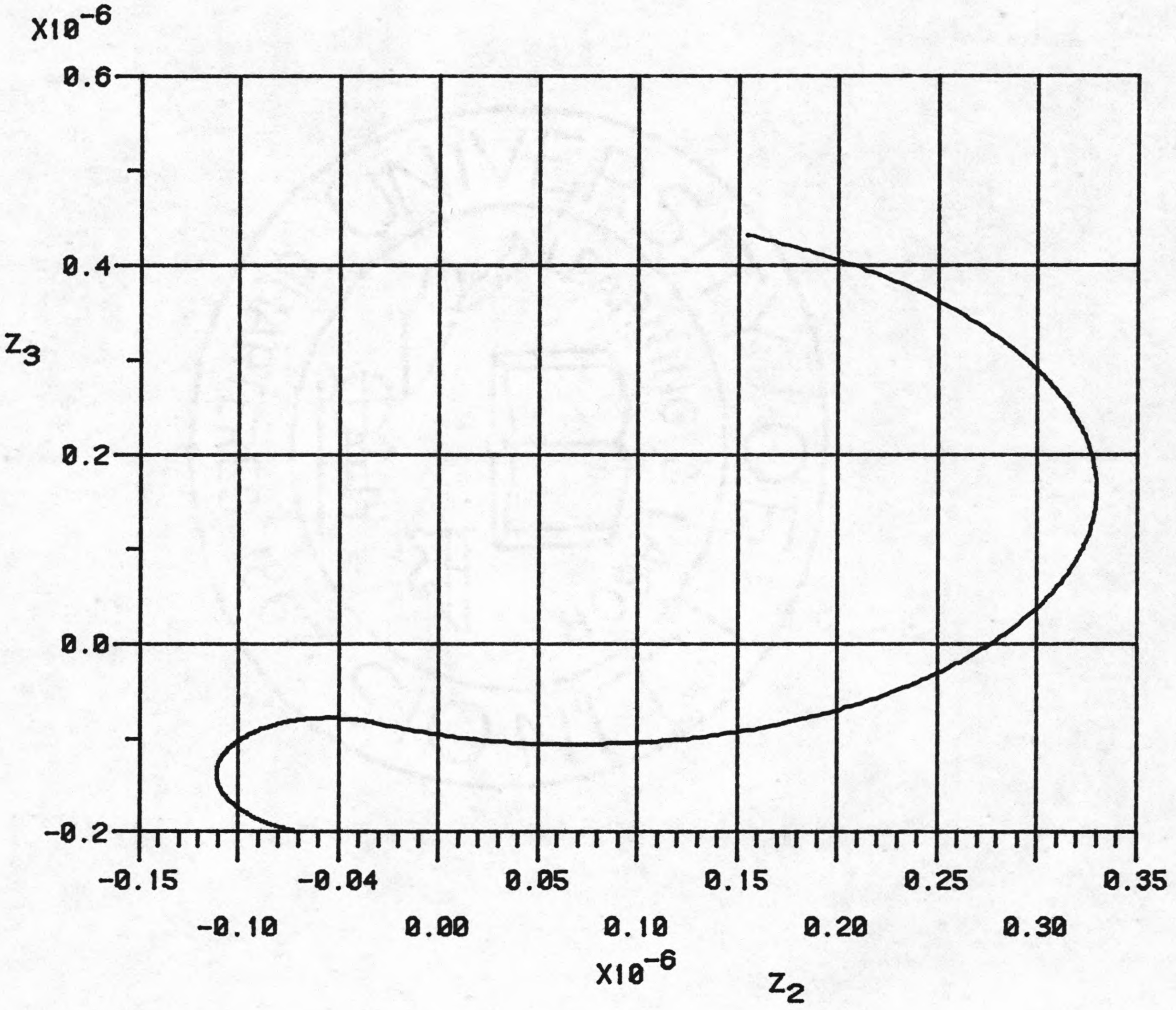

Fig. 2.3(b) Switching Curve for Final Track = 3600.

Rewriting the equation of the switching curve:

$$R_s - R_{s0}*\exp[(\theta_s - \theta_{s0})/8.9356] = 0 \qquad (2.17)$$

with R_s, R_{s0}, θ_s and θ_{s0} as in Eq. (2.16).

The location of any given point with respect to the switching curve can be found by substituting its value in the left hand side of Eq. (2.17). Let EQ be the value obtained by this substitution. The value of EQ will be identically zero only if the point lies on the switching curve. Refering to Fig. 2.4 (switching curve with origin as the final point), the point under consideration lies in the shaded region if the value of EQ is negative and in the unshaded region if EQ is positive. Thus knowing which half of the plane the point lies in (by the value of z_2), it can be inferred by the sign of EQ if the point lies above, below or on the switching curve.

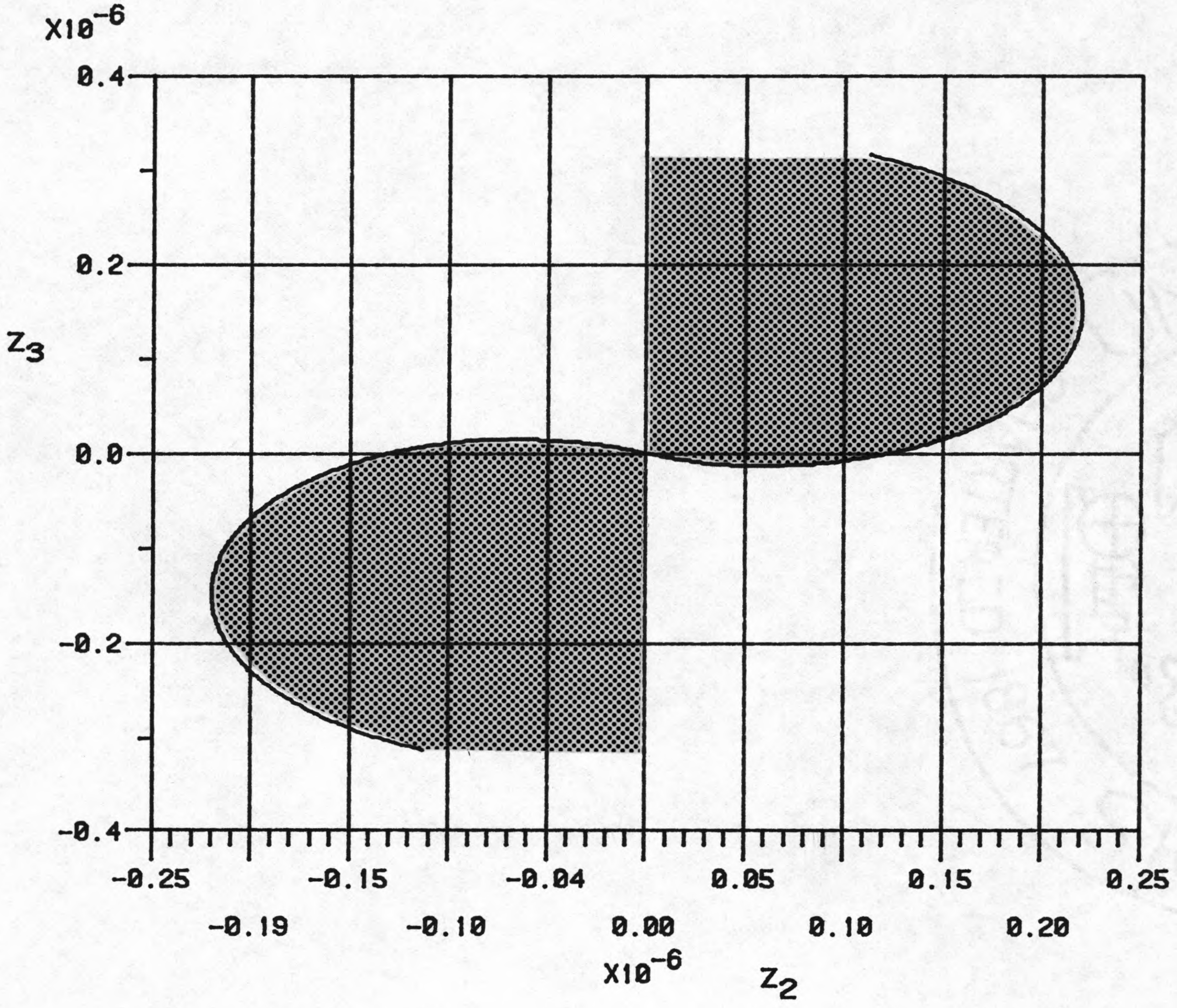

Fig. 2.4 Switching Curve for Going to the Origin (Track 2048).

CHAPTER 3

SIMULATION AND RESULTS

3.1 Simulation on Hybrid Computer and DEC-10.

The system was simulated on the AD-5 (Applied Dynamics) Analog Computer with PDP-11/40 as the digital controller. The ultimate goal is to be able to use a microcomputer system (e.g. INTEL 8080 based system) as the digital controller, which would physically reside on the system. A documentation of the existing 8080A based microcomputer system with a description of the available software is given in Appendix E. For simulation purposes it was assumed that all three states are available. The equation for the switching curve being fairly complicated (involving exponentials and arc cosines), it was inferred that the PDP-11 could not perform the required calculations to control the system in real time. Therefore, the simulated system was time scaled by a factor of 1000. The state x_3 was also magnitude scaled by 1000 to keep the simulation within the dynamic range of the hybrid computer. Thus the scaled system equations are:

$$\dot{x} = \begin{bmatrix} -0.2667 & 0 & -5.5333 \\ 0 & 0 & 1 \\ 0.1428 & -0.2857 & -0.0346 \end{bmatrix} x + \begin{bmatrix} 0.3333 \\ 0 \\ 0 \end{bmatrix} u/10. \quad (3.1)$$

Fig. 3.1 shows the patching diagram of the system on the Hybrid Computer. It was attempted to control this simulated system, but the accuracy achieved was not satisfactory. This is mainly due to the fact that the calculations involved are too time consuming making the control algorithm comparatively slow even for the time scaled system. However, this simulation was found very useful in obtaining system responses and studying system behaviour under open loop control.

To achieve better accuracy, the system was simulated on the DEC-10 Digital Computer. A simple Euler approximation for integration was used with very small value for Δt (5×10^{-7} sec.). For this simulation also it was assumed that all the states were available. The control algorithm was applied to this simulated system and the desired accuracy was achieved so that the criteria of reaching within ± 10 tracks was satisfied.

3.2 The Algorithm.

The iterative procedure described in Sec. 2.1 with the step size modification is used to find the control for the system. The initial and final tracks are given and it is assumed that the states are available at all times. Applying the linear transformation discussed in Sec. 2.2, the transformed states are formed. To find the control required to move the system from any arbitrary initial point

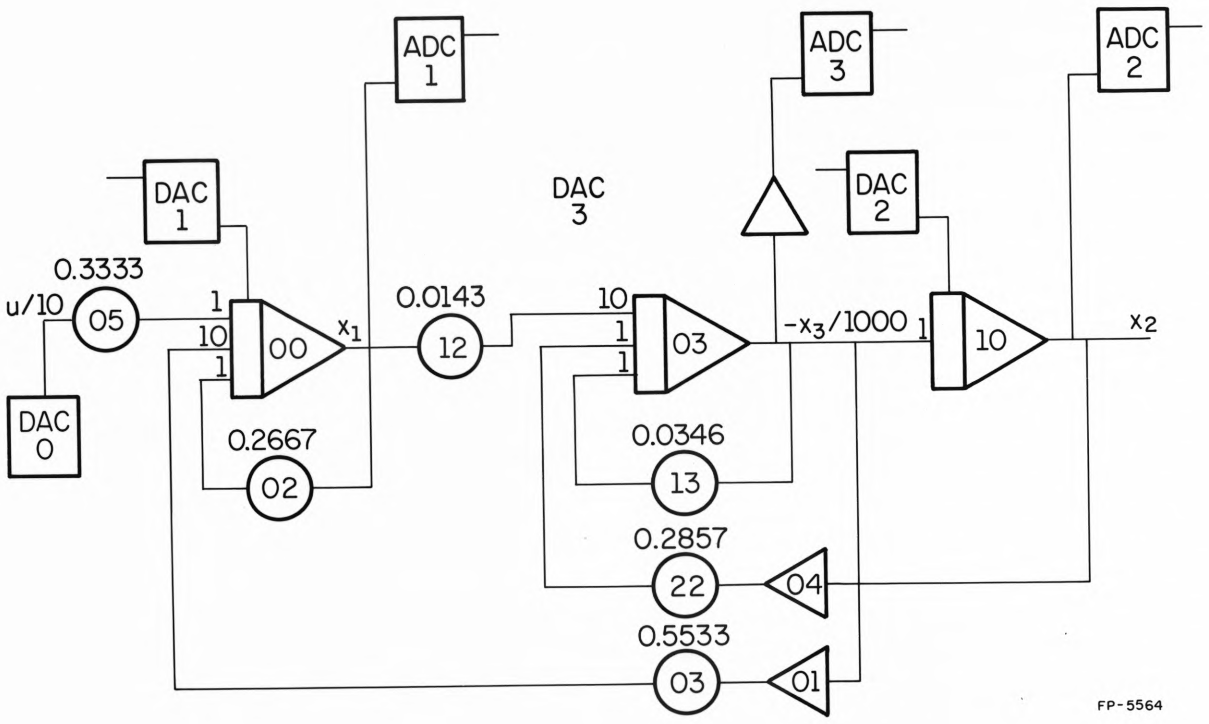

Fig. 3.1 Patching Diagram of the System on the Analog Computer.

to any final point, the slow state z_1 is first steered to its final value z_{1f}. The required control is given by (2.7). At $z_1=z_{1f}$, the location of the transformed system states in the z_2-z_3 plane is determined with respect to the switching curve. This is done by substituting the present state values in (2.17) and calculating the value of EQ as described in Sec. 2.4.1. Knowing the position of the states with respect to the switching curve, the control sequence for the fast subsystem is formed. The proper control is then applied and the position of the system states are evaluated at every sampling instant. The values of the system states are substituted in the equation for the switching curve (2.17) after very sampling period to check if the switching curve is reached. When the states reach the switching curve or are in a very small neighborhood ϵ ($=0.2\times10^{-9}$), of the switching curve, the control is switched. This (switched) control is now applied until the states reach a small neighborhood $\sigma(0.5\times10^{-8})$ of the final point (z_{2f},z_{3f}). While z_2 and z_3 are steered to their final values, the slow state z_1 drifts away from its final value (to which it was steered). The iterative procedure is now applied and an intermediate value of z_1, (z_{1i}), is calculated using (2.9). The calculated z_{1i} is multiplied by the step size s and on second iteration, z_1 is steered to z_{1i}*s instead of z_{1f}. The whole procedure is repeated until the final track value reaches within ±10 tracks of the desired value. Fig. 3.2 shows a flow chart for the

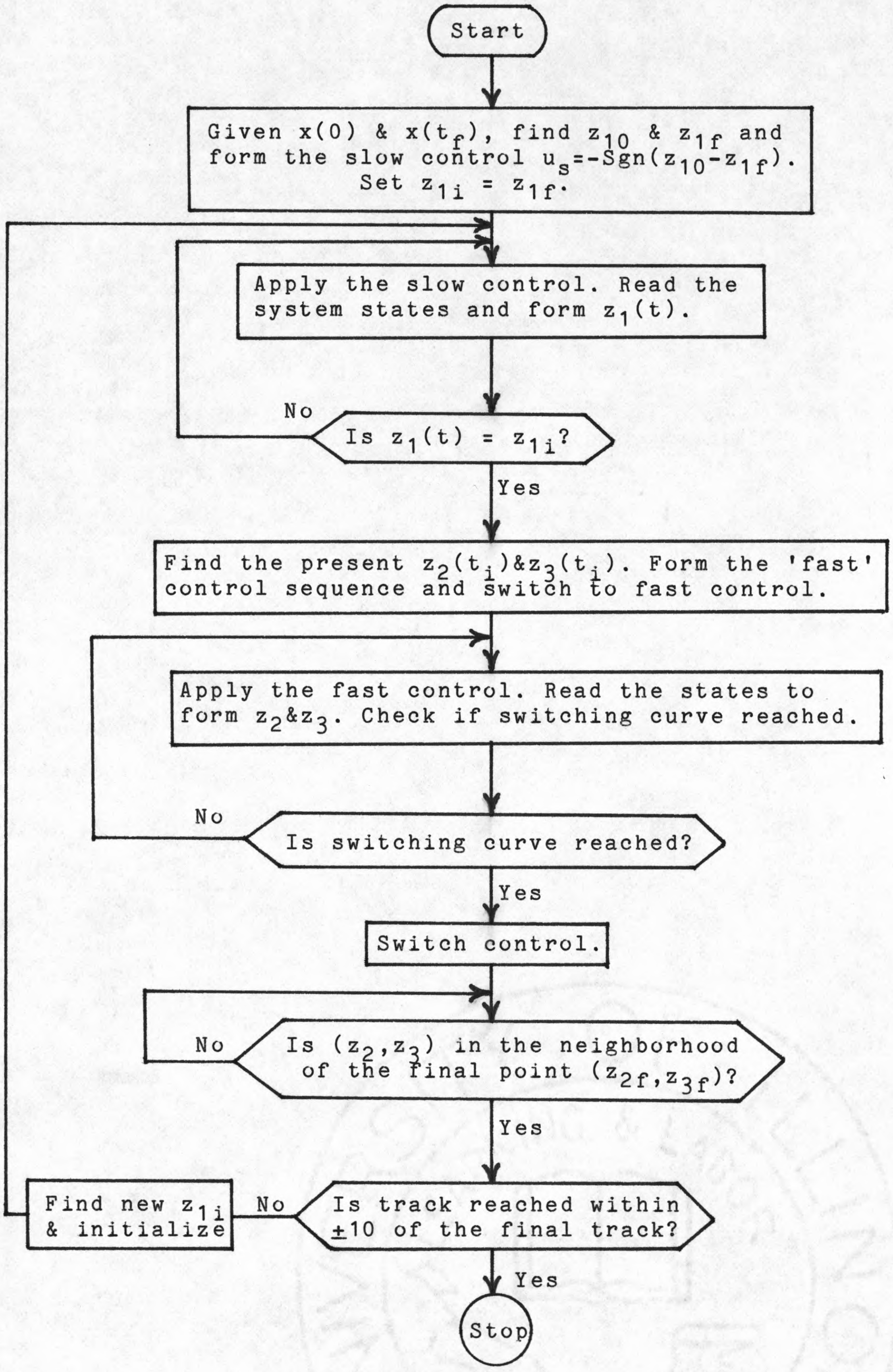
Start
Given $x(0)$ & $x(t_f)$, find z_{10} & z_{1f} and form the slow control $u_s = -\mathrm{Sgn}(z_{10} - z_{1f})$. Set $z_{1i} = z_{1f}$.
Apply the slow control. Read the system states and form $z_1(t)$.
No
Is $z_1(t) = z_{1i}$?
Yes
Find the present $z_2(t_i)$&$z_3(t_i)$. Form the 'fast' control sequence and switch to fast control.
Apply the fast control. Read the states to form z_2&z_3. Check if switching curve reached.
No
Is switching curve reached?
Yes
Switch control.
No
Is (z_2, z_3) in the neighborhood of the final point (z_{2f}, z_{3f})?
Yes
Find new z_{1i} & initialize
No
Is track reached within ± 10 of the final track?
Yes
Stop

Fig. 3.2 Flow Chart for the Time-Optimal Algorithm.

algorithm. The computer program for implementing this algorithm is given in Appendix C.

3.3 Results.

Utilizing the varying step size, the iterative procedure converges, on an average, in 8 iterations. Although on the actual system a control voltage of upto $\pm 15V$ is allowed, here the control voltage considered is $\pm 5V$. This does not change the problem of designing the control in any way. Changing the control from $\pm 5V$ to $\pm 15V$ merely speeds up the system, with the proposed time-optimal algorithm remaining the same. Fig. 3.3(a) and 3.3(b) show the time response and the phase-plane trajectories, respectively, of the controlled system in going from track 100 to track 4000 (picked arbitrarily). From the figures it is seen that state x_1 oscillates as it moves towards its desired final value but the response for x_2 is not very oscillatory. The response for x_2 is that of a monotonic non-decreasing (or non-increasing) function so that the mirror deflection is always in the right direction, without oscillating back and forth. The actual tracks traversed and the corresponding control is shown in Figs. 3.3(c) and 3.3(d) respectively. The track reached is very close to the actual desired track (track 4004 was reached with the desired being 4000). The control required shows that the slow control is applied for the longest time with two fast

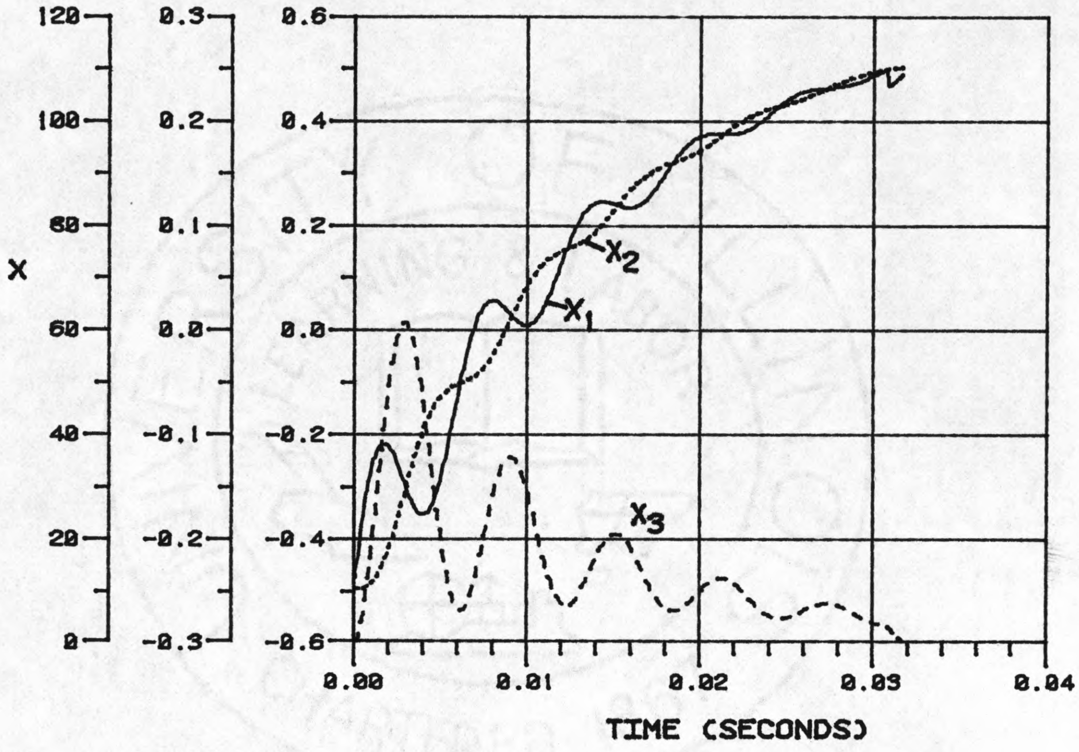

Fig. 3.3(a) Time Responses of the Controlled System States.

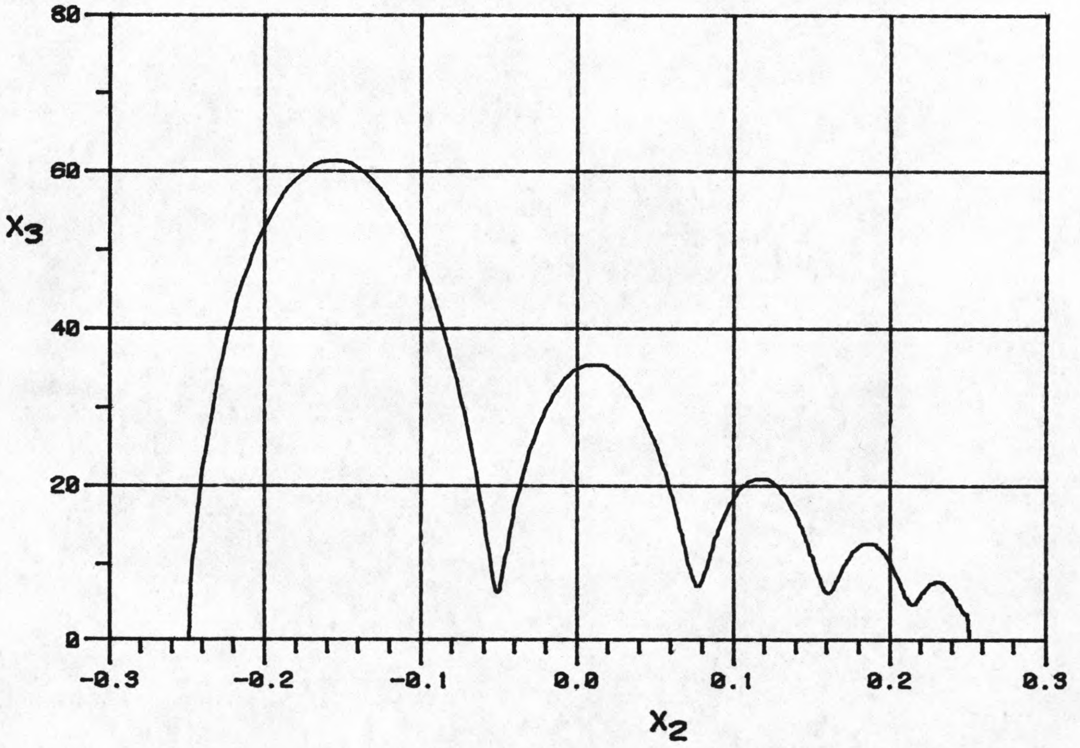

Fig. 3.3(b) Phase-Plane Trajectory of the Controlled System in the (x_2, x_3) Plane.

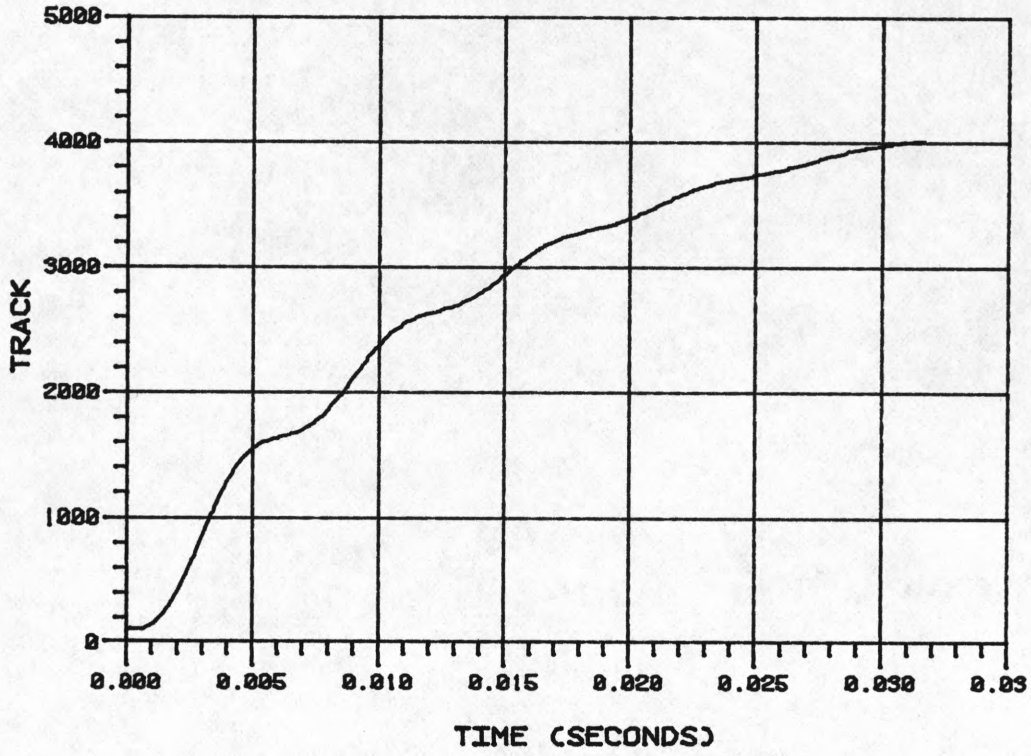

Fig. 3.3(c) Actual Track Movement Versus Time.

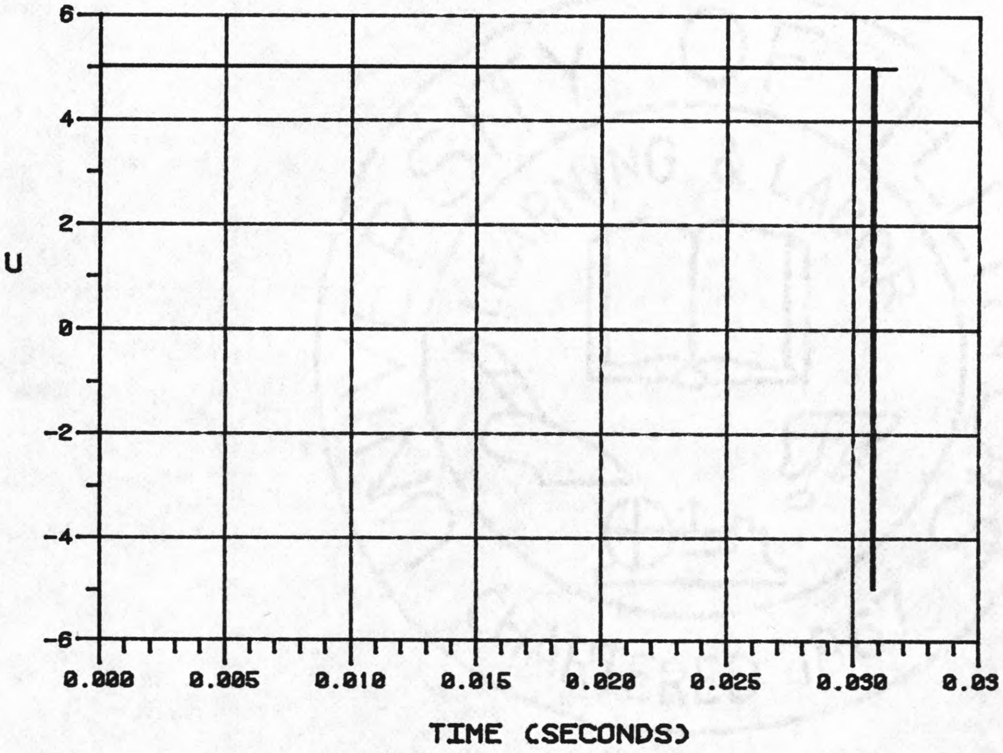

Fig. 3.3(d) Control u(t) Versus Time.

switchings at the end. Refering to Fig. 3.3(c), the slow control takes the system fairly close to the final value (track 3980) and then the fast control takes the system to the desired position. Figs. 3.3(e) and 3.3(f) show the time response and phase-plane trajectory of the transformed system (controlled) states respectively for the same track movement (track 100 to track 4000). The effect of the slow and fast controls is clearly seen in Fig. 3.3(f). The system moves towards the final point in an oscillatory motion and as soon as the desired slow state is reached, two fast switchings takes the system to the final point. Figs. 3.4(a) through 3.4(f) are identical to Figs. 3.3(a) through 3.3(f) except that the tracks traversed is from track 3500 to 300.

The time required to make these large excursions are around 30ms. The time decreases as the number of tracks to be traversed is decreased. Also applying a control of $\pm$15V instead Of $\pm$5V would decrease the required time by a factor of 2.

3.4 Real Time Implementation.

3.4.1 Feasibility Study.

The control required for travelling absolute distances is not the same, rather it depends on the initial and final points. If the control for travelling absolute distances

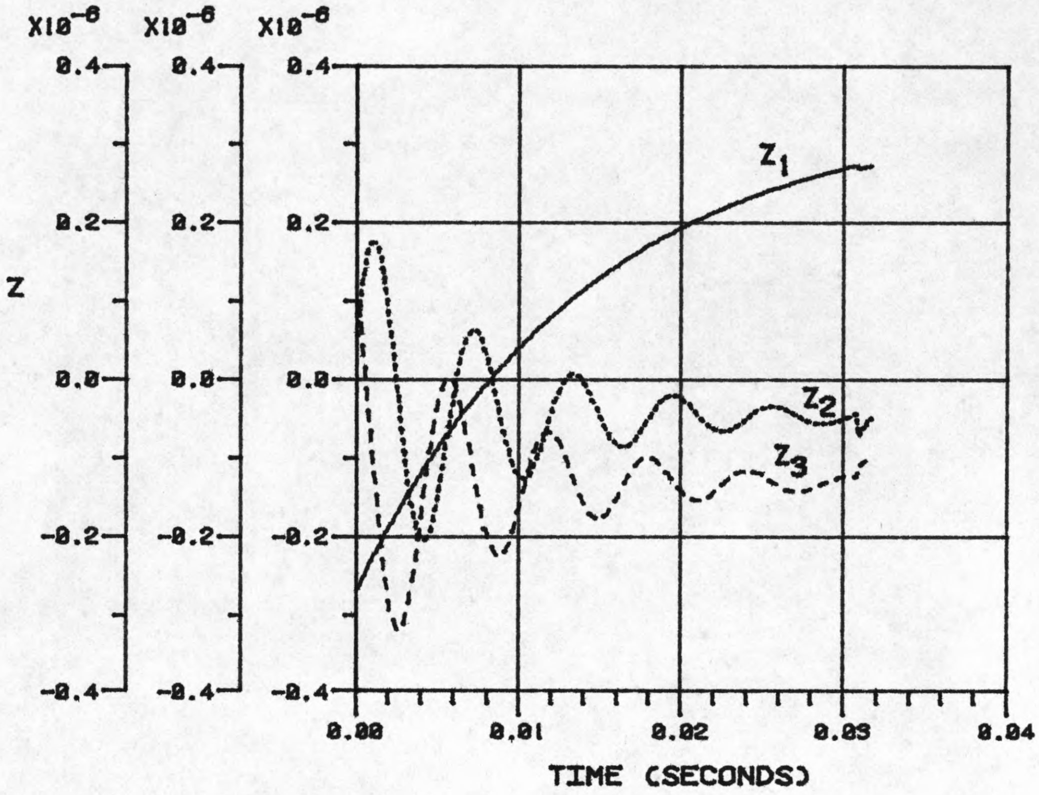

Fig. 3.3(e) Time Responses of the Transformed States of the Controlled System.

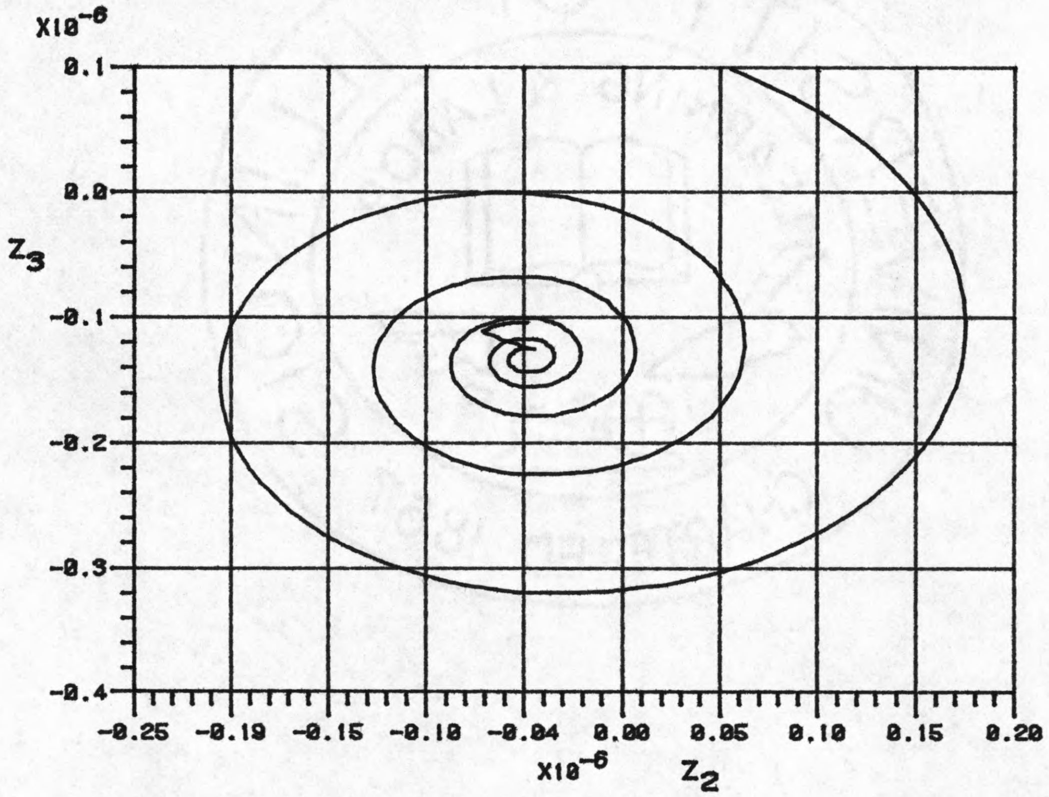

Fig. 3.3(f) Phase-Plane Trajectory of the Transformed Controlled System.

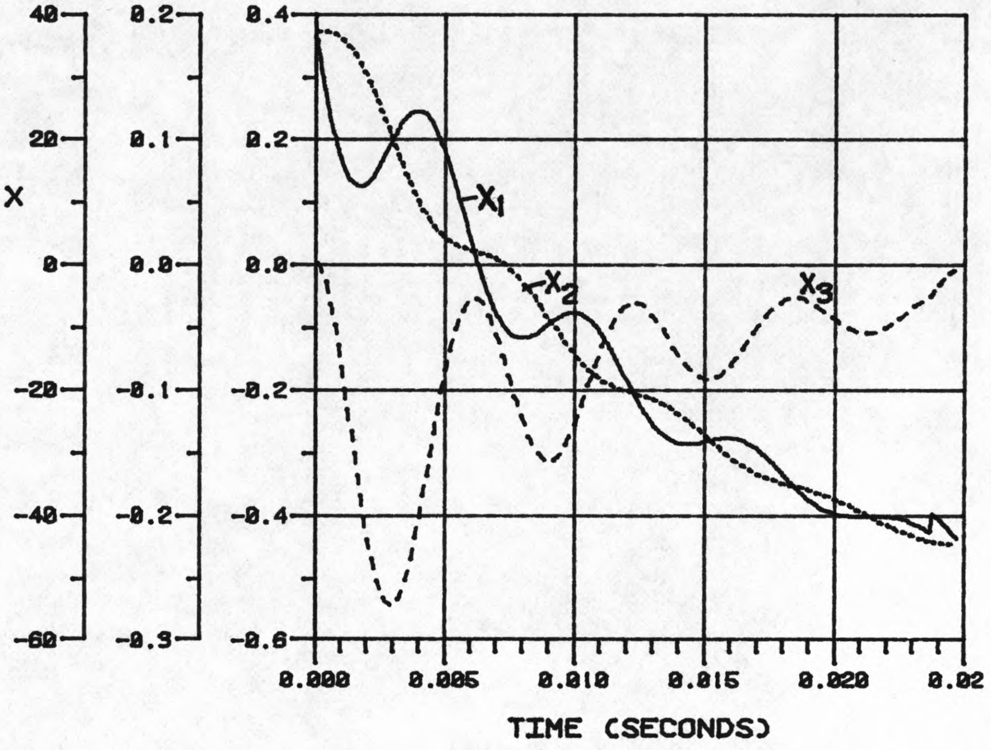

Fig. 3.4(a) Time Responses of the Controlled System States.

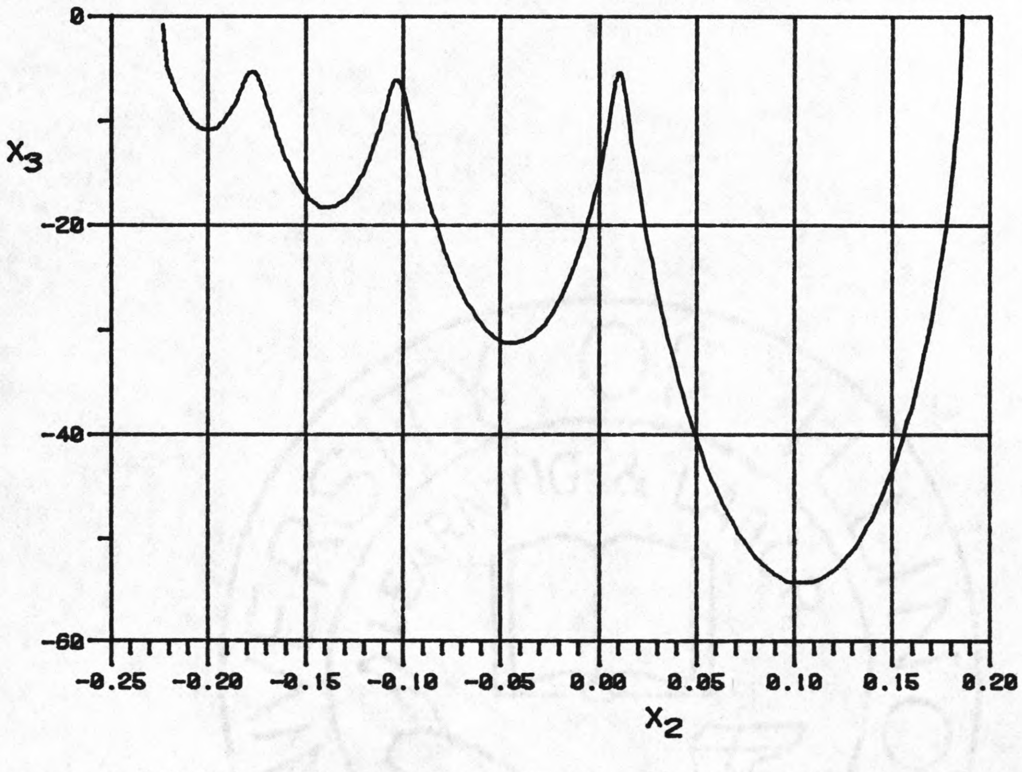

Fig. 3.4(b) Phase-Plane Trajectory of the Controlled System in the (x_2, x_3) Plane.

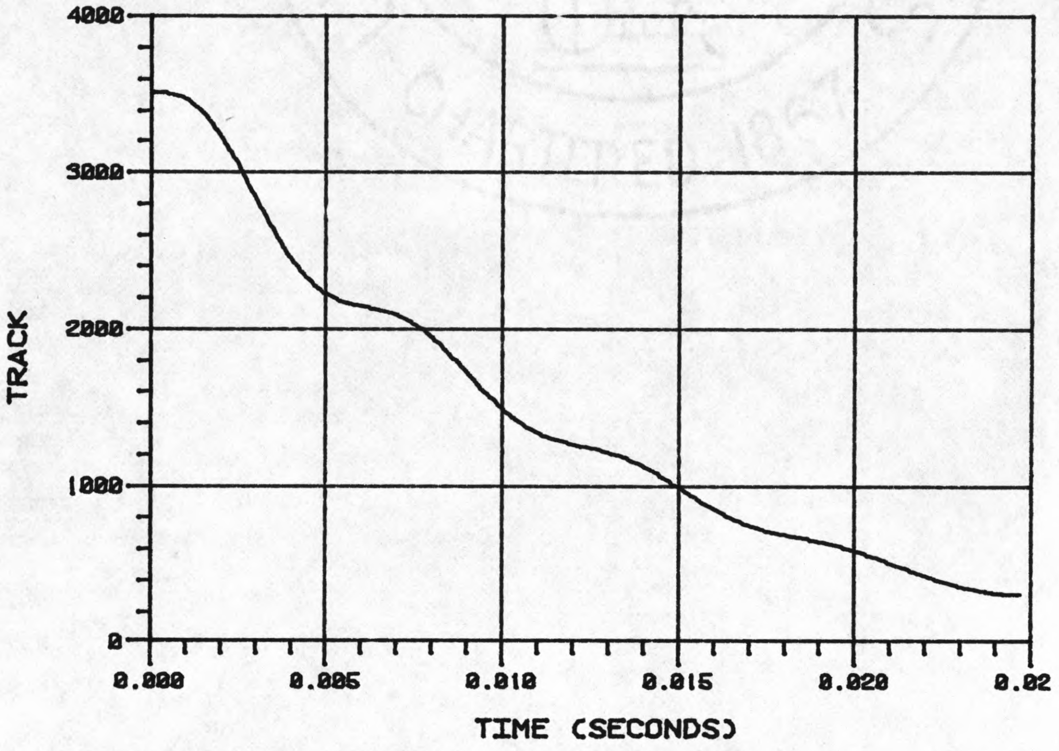

Fig. 3.4(c) Actual Track Movement Versus Time.

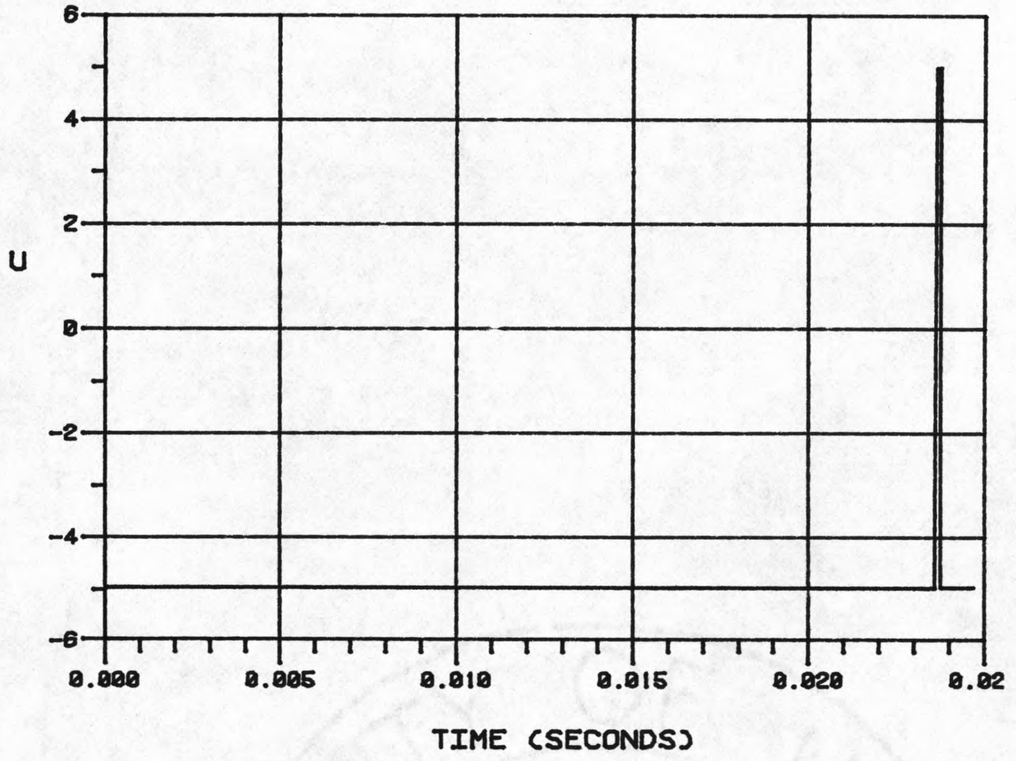

Fig. 3.4(d) Control u(t) Versus Time.

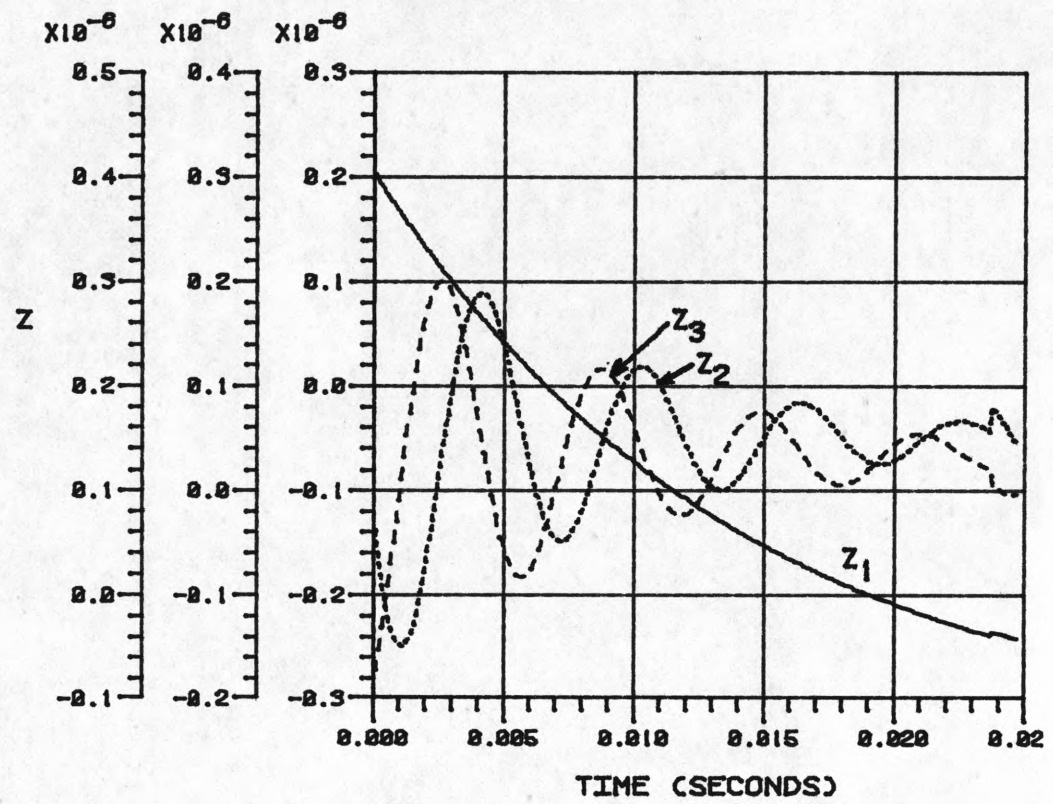

Fig. 3.4(e) Time Responses of the Transformed States of the Controlled System.

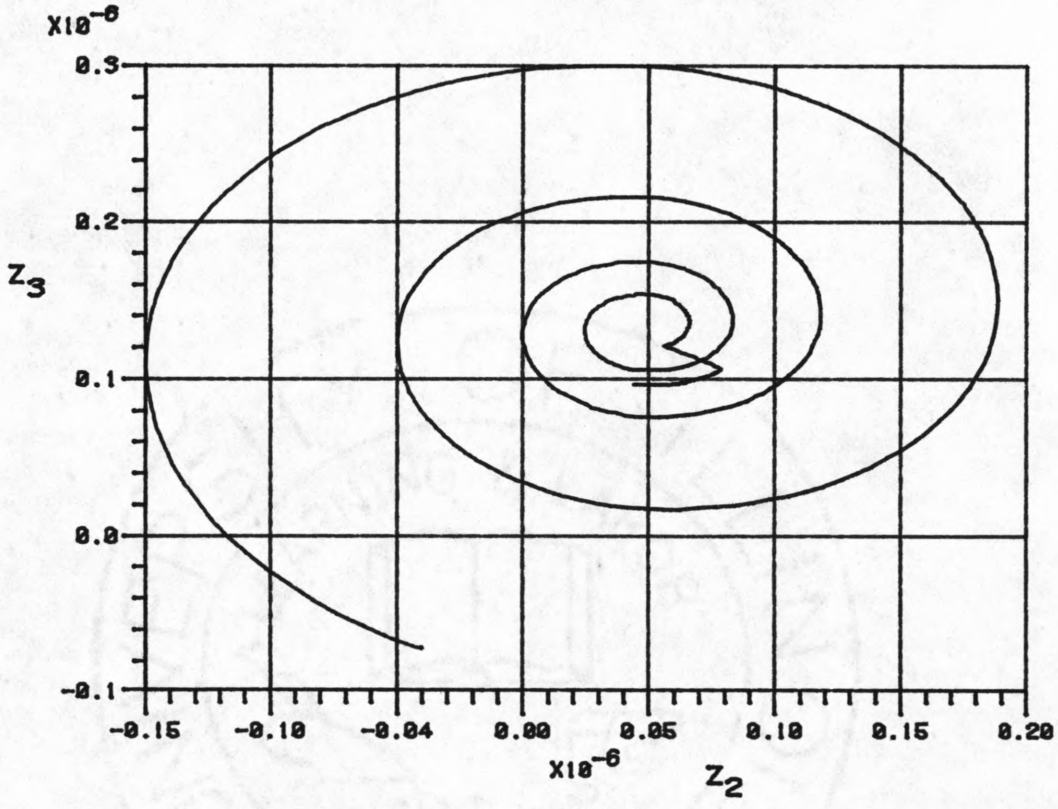

Fig. 3.4(f) Phase-Plane Trajectory of the Transformed Controlled System.

were the same there would be 4096 different solutions, which could be conveniently stored up to form a look-up table. But here the solution for going from every track to every other track is different. Thus there are n(n+1)/2, i.e. about 16.78 million different solutions. Considering the symmetry about the origin the number of different solutions can be reduced to half, leaving 8.39 million solutions. A look-up table of this size is impractical, specially keeping in mind that the control being designed is for a "memory device". The only reasonable thing left to do is to calculate the control in real time. But, as mentioned earlier, this is not feasible because calculating the control involves finding exponentials and arc cosines at every sampling period. Furthermore there would be a need for position feedback, which is difficult to obtain. Calculating exponentials and arc cosines on a microcomputer in real time does not seem feasible at all. The other possibility is to store the tables of exponentials and arc cosines, but this again involves a huge amount of memory.

Experimenting with the system, a very peculiar property of the system was discovered for which no theoretical justification is known at the moment. This property (explained in the next section) has made it possible to form a look-up table, containing only 4096 entries, from which all the solutions can be found by a very simple iterative algorithm. This algorithm involves one multiplication and one addition per iteration. Five iterations are needed so that the algorithm can be implemented in real time.

3.4.2 Algorithm for Real Time Implementation.

The property of the system that has made it possible to have a look-up table (containing 4096 solutions) will be discussed here first. Suppose that all the solutions for going from track 1 to any other track is stored in a table. Let u_n be the solution for going from track 1 to track n. The solution u_{500} (arbitrary) is picked. Now this solution is applied to the system but with track 100 as the initial track (instead of track 1). One might expect the final track to be 600 (a distance of 500 tracks away). But it is seen that instead of going to track 600, the system goes to track 580. This means that there is a "loss" of 20 tracks for a "shift" of 100 tracks in the initial value. Now if instead of track 100 the same solution (u_{500}) is applied starting at track 200, the system ends up on track 660; a "loss" of 40 tracks. Thus there exists a linear relationship between the loss in the number of tracks traversed and the "shift" in the initial point. For the above example, the loss per track shift is 0.2. That is,

Tracks "lost" = 0.2*Initial track "shift".

The factor relating the number of tracks lost and initial track shift will be called "loss factor" from now on. In the above equation the loss factor is 0.2. Thus, if it is required to move from track 100 to track 580 or from track 200 to track 660, the solution u_{500} can be applied. Considering every possibility for arbitrary track movements, there is always a solution available in the look-up table that can be used. The only thing required is an algorithm to pick the right solution from the table (of 4096 solutions).

Unfortunately, the loss factor varies from solution to solution. That is, the loss factor for u_{500} is different from the one for u_{1000} or u_{1200}. But it is important to note here that the loss factor for a particular u_n is constant for all shifts in the initial point. Furthermore, the values for the loss factors do not have jumps in them. It changes slowly between 0.07 and 0.92. Thus the value of the loss factor can be taken as a constant over certain ranges of track numbers. Experimentally it was found that the value of the loss factor do not change appreciably over a range of 20 tracks, for which the value can be taken as constant. Thus, only 205 different values of loss factor are required.

Due to the symmetry in the system in going from left to right or right to left on the disc, everything that is true for travelling in one direction is also true for the other direction. The innermost track on the disc is numbered 1 and the outermost is numbered 4096. The laser beam is focused to the center of the disc (on track 2048) for zero current and zero deflection. Thus to focus on track 1 or track 4096 would require maximum current and maximum deflection on either sides. Thus the solution for going from track 1 to any track is exactly the same as for going from track 4096 to a track same distance away, except that the sign of the controls are reversed.

An algorithm to pick a solution from the table, that is also a solution for a given initial and final point, is given below. The basic idea is to take a solution u_n and apply it starting at the given initial track, at the same time taking into account the "loss" encountered due to the "shift". The further away the initial track is (from track 1), the greater is the loss. Let u_n be the control for moving from track 1 to track n (as defined earlier) and let L_n be the corresponding loss factor. Suppose it is required to move from track m to track k.

1. Find the number of tracks to be traversed,

$$d = k-m$$

2. Find the associated loss factor L_d. The shift in the initial track is m-1 (from 1 to m). Therefore, the loss is

$$L = L_d*(m-1)$$

3. Let $n = d+L$.

4. Find L_n. Note that L_n is different from L_d. Find the new value of loss,

$$L = L_n*(m-1)$$

5. Repeat steps (3) & (4) five times. Note that the value of d (distance to be traversed) remains constant for a

particular set of iterations.

6. Look-up the value of u_n. This is the required control for going from track m to track k. The value of n is the one obtained after five iterations.

It was found that five iterations were sufficient to achieve the desired accuracy of reaching within ± 10 tracks. Each iteration in the above algorithm involves one addition, one multiplication and 'looking up' the value of the loss factor from the table. Fairly cheap Random Access Memory is available with an access time of a few hundred nanoseconds. The time required in performing an addition is in the order of few microseconds. The only time consuming operation is that of multiplication, which might require 200 to 300 μsec. Therefore the time required for the algorithm to find the control would be around 1.5 to 2 msec. This time could be drastically reduced by employing hardware multiplier instead of software multiplication routine. Typical time for finding the control in this case would be less than 300 μsec.

Fig. 3.5 shows the flow chart for the algorithm. The computer program to implement this algorithm is given in Appendix D.

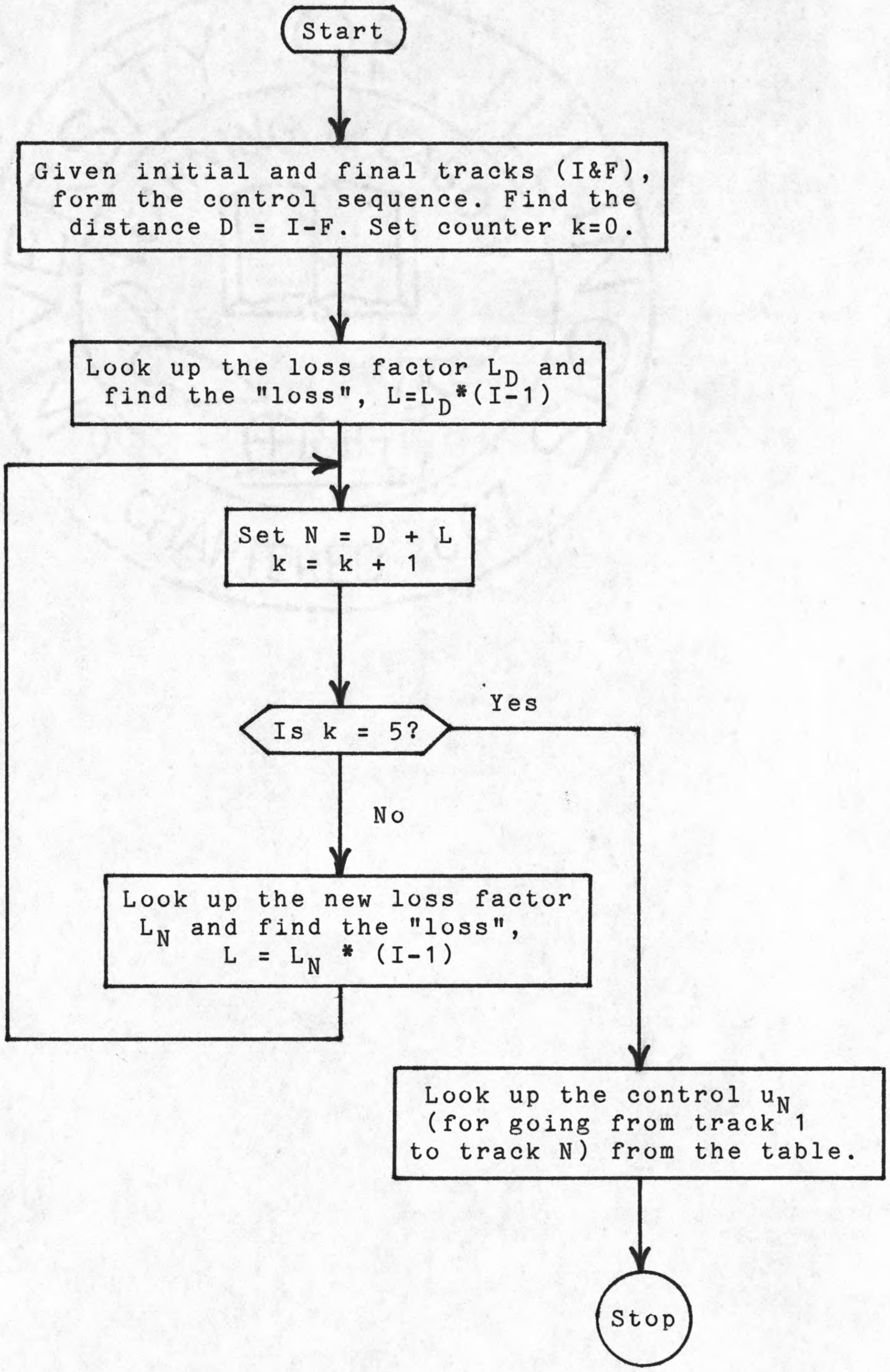
Start
Given initial and final tracks (I&F), form the control sequence. Find the distance D = I-F. Set counter k=0.
Look up the loss factor L_D and find the "loss", $L=L_D*(I-1)$
Set N = D + L
k = k + 1
Is k = 5?
Yes
No
Look up the new loss factor L_N and find the "loss", $L = L_N * (I-1)$
Look up the control u_N (for going from track 1 to track N) from the table.
Stop

Fig. 3.5 Flow Chart for Real-Time Algorithm.

3.4.3 Results and Discussion.

The proposed algorithm for real time implementation was applied to the system model. The results obtained were quite satisfactory. Figs. 3.6(a) & 3.6(b) show the result of applying this new algorithm (referred to as 'real time algorithm' from now on) and the time-optimal algorithm (discussed in Sec. 3.2) respectively, for going from track 50 to track 2000. For the real time algorithm, track 1992 is reached with the required total time of 9.76ms. With the time-optimal algorithm track 1994 is reached in 9.74ms. Figs. 3.7(a) & 3.7(b) show a similar comparison in going from track 3600 to track 1200. The track reached by the real time algorithm in this case is 1193 with an access time of 14.54ms. The time-optimal algorithm takes the system to track 1192 in 14.39ms. In both the above cases it is seen that the accuracy of the real time algorithm is acceptable and also the access time is within 1.1% of the one required by the time-optimal algorithm (which cannot be implemented in real time). Several different points were tried and the real time algorithm performed well. In every case the criteria of reaching within ± 10 tracks was satisfied. For the worst case, the time required by the real time algorithm was 1.5% more than that required by the time-optimal algorithm.

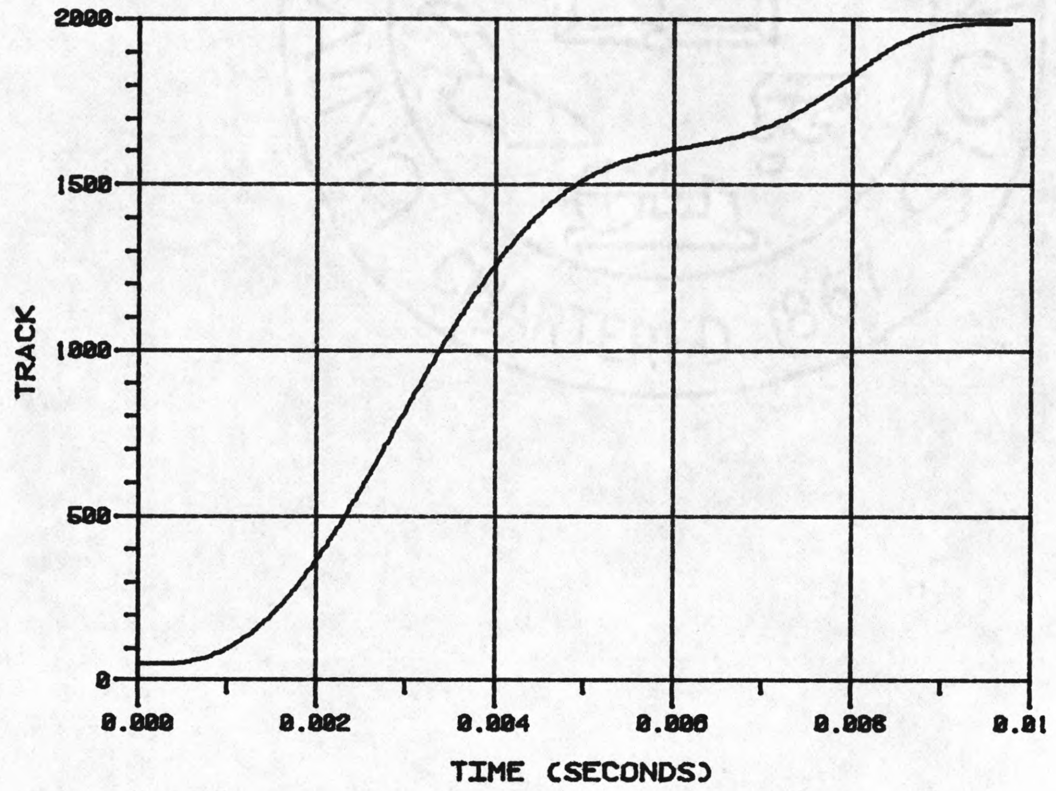

Fig. 3.6(a) Track Movement Vs. Time Applying the Real-Time Algorithm.

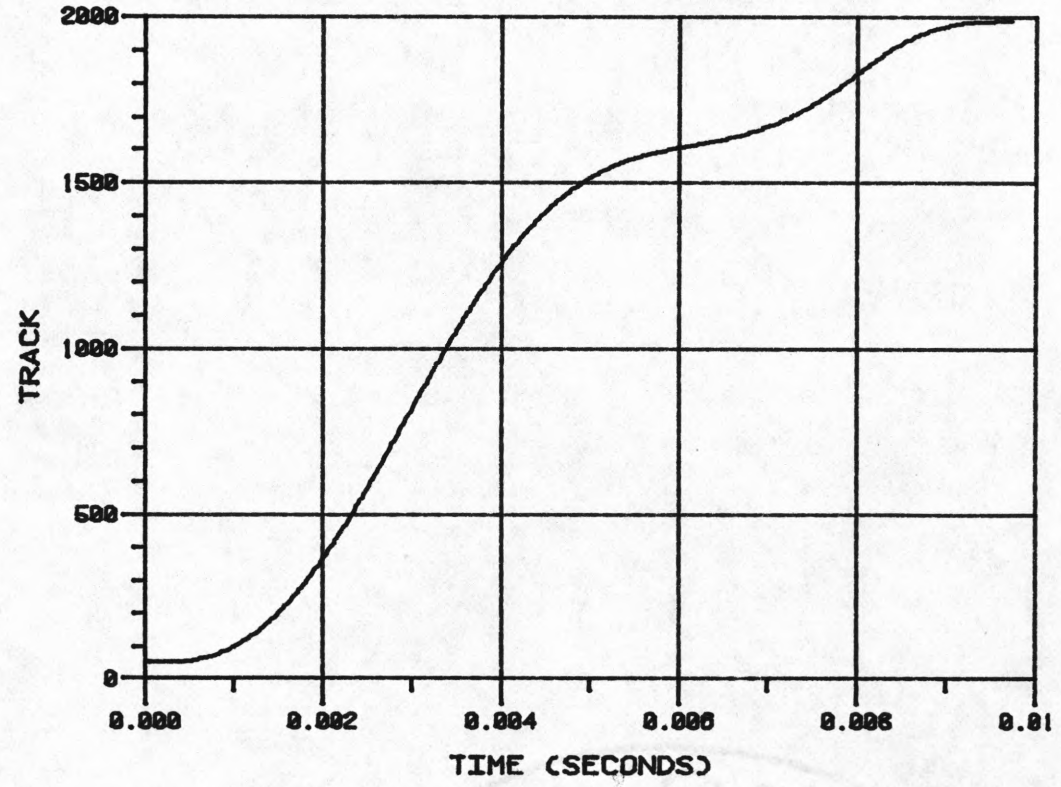

Fig. 3.6(b) Track Movement Vs. Time Applying the Time-Optimal Algorithm.

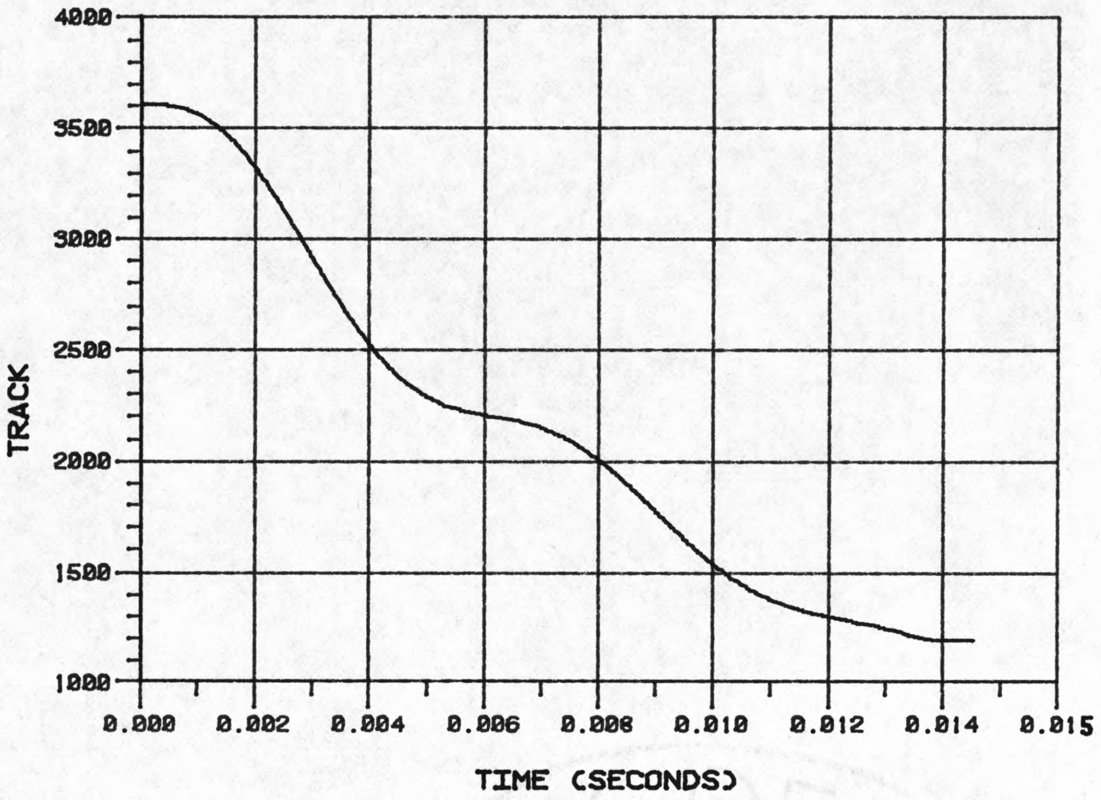

Fig. 3.7(a) Track Movement Vs. Time Applying the Real-Time Algorithm.

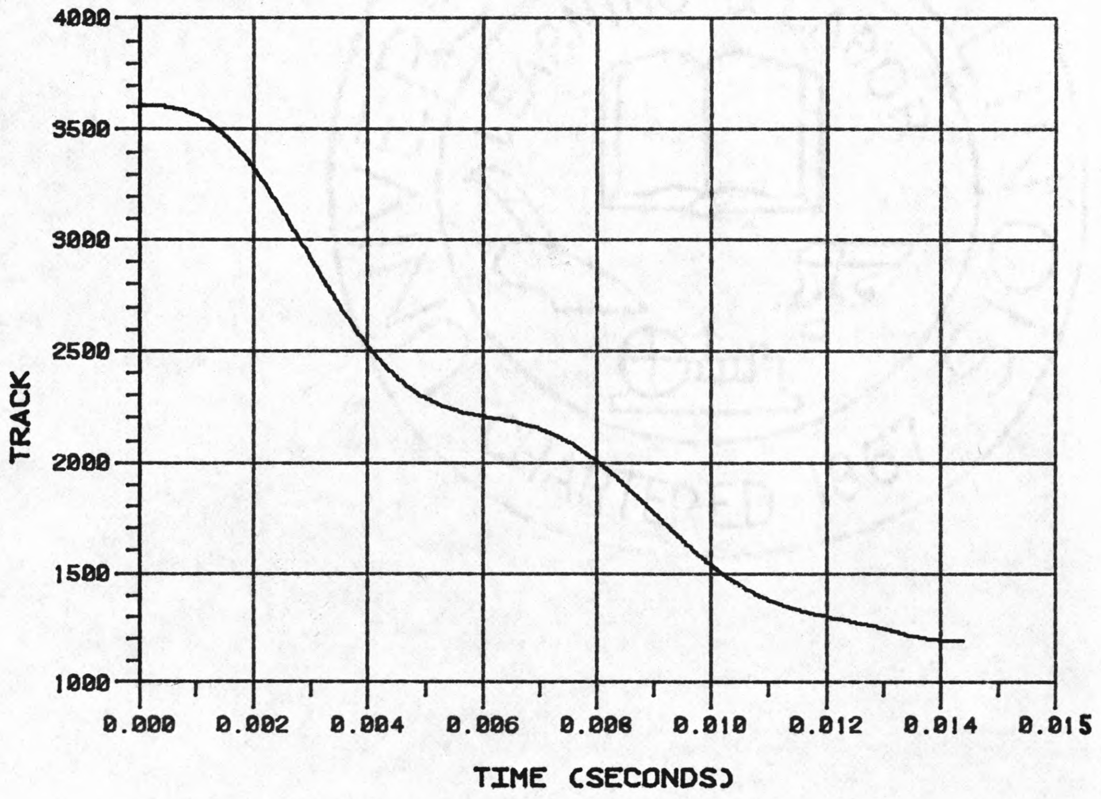

Fig. 3.7(b) Track Movement Vs. Time Applying the Time-Optimal Algorithm.

Nevertheless, the time-optimal algorithm is the basic algorithm that is used to generate all the solutions in going from track 1 to any other track. These 4096 different solutions are stored in the form of a table which is utilized by the real time algorithm.

CHAPTER 4
CONCLUSIONS

The purpose of this thesis was to design a time-optimal control algorithm for positioning the mirror so as to focus the laser beam. The focussing of the laser beam on a desired track acts as an information retrieval system for the optical memory device. The mirror to be positioned is mounted on the arm of a moving iron galvanometer. It is required to control the angular movement of the armature for precise deflection of the laser beam (reflected off the mirror).

The overall system description of the optical memory device alongwith the system model was given in Chapter 1. A third order model of the system was considered taking into account the inductance in the motor and also the back emf effect.

The basic approach taken in solving the third order system was to apply the theory of singular perturbation, i.e. decompose the system into 'slow' and 'fast' subsystems and then try to control them separately. An iterative procedure based on the theory of singular perturbation was applied. The first two sections in Chapter 2 describes the iterative procedure and the linear transformation for decomposing the system into slow and fast subsystems. The time-optimal control for the subsystems, utilizing Minimum Principle, was developed in the remaining part of Chapter 2.

The control algorithm alongwith the simulation and results was given in Chapter 3. The simulation results show that the proposed control algorithm satisfies the criteria of reaching within ±10 tracks of the final value. But, it is seen that the algorithm is too complicated to be implemented in real time. The feasibility of having a open-loop control (with look-up table) was discussed next. It was found that the number of different solutions to be stored in the table was over 8 millions, making the idea of open-loop control seemingly impractical. A very peculiar property of the system was then discussed that made it possible to have a table containing 4096 different solutions only. Finally, an algorithm was proposed that could be implemented in real time to generate all the solutions to the system from the table (of 4096 different solutions).

Thus, it is feasible to have an open-loop control for the system by forming a look-up table of 4096 solutions and using the 'real time algorithm' to find the control for any desired track movement.

LIST OF REFERENCES

[1] G. S. Gurley, "Time-Optimal Control of a Video Disc Scanner", M. S. Thesis, University of Illinois, Urbana, Illinois, 1977

[2] P. V. Kokotovic and A. H. Haddad, "Singular Perturbation of a Class of Time-Optimal Controls", IEEE Trans. Automatic Control, Vol. AC-20, No. 1, pp 163-164, Feb. 1975

[3] P. V. Kokotovic and A. H. Haddad, "Controllability and Time-Optimal Control of System With Slow and Fast Modes", IEEE Trans. Automatic Control Vol. AC-20, No. 1, pp 111-113, Feb. 1975

[4] S. H. Javid, "The Time-Optimal Control of a Class of Singularly Perturbed Systems", M. S. Thesis, University of Illinois, Urbana, Illinois, 1975

[5] Michael Athans and Peter L. Falb, "OPTIMAL CONTROL: An Introduction to the Theory and Its Application", New York: McGraw Hill, 1966

APPENDIX A

THE PHASE-PLANE TRAJECTORIES.

The state equations of the fast subsystem are

$$\dot{z}_2 = -114.99z_2 + 1027.5z_3 + 0.2625\times10^{-4}u$$
$$\dot{z}_3 = -1027.5z_2 - 114.99z_3 - 0.1307\times10^{-4}u \; . \qquad (A.1)$$

Let (z_{2e},z_{3e}) be the equilibrium point with $u = k$. Define a new coordinate system (z_2',z_3') with its origin at (z_{2e},z_{3e}) i.e.

$$z_2' = z_2 - z_{2e}$$
$$z_3' = z_3 - z_{3e} \; .$$

Transforming to polar coordinates,

$$z_2' = R*\mathrm{Cos}\theta = z_2 - z_{2e}$$
$$z_3' = R*\mathrm{Sin}\theta = z_3 - z_{3e}$$

or

$$z_2 = R*\mathrm{Cos}\theta + z_{2e}$$
$$z_3 = R*\mathrm{Sin}\theta + z_{3e} \; . \qquad (A.2)$$

Substituting (A.2) in (A.1), the state equations take the form

$$\dot{R}^*\mathrm{Cos}\theta-R^*\mathrm{Sin}\theta\dot{\theta}= -\alpha(R^*\mathrm{Cos}\theta+z_{2e})+\beta(R^*\mathrm{Sin}\theta+z_{3e})+\zeta u \qquad (A.3(a))$$

$$\dot{R}^*\mathrm{Sin}\theta+R^*\mathrm{Cos}\theta\dot{\theta}= -\beta(R^*\mathrm{Cos}\theta+z_{2e})-\alpha(R^*\mathrm{Sin}\theta+z_{3e})-\eta u \qquad (A.3(b))$$

where the symbols used for writing convenience are:

$\alpha = 114.99$, $\beta = 1027.5$, $\zeta = .2625\times10^{-4}$ and $\eta = 0.1307\times10^{-4}$.

From (A.1), for equilibrium point

$$\begin{aligned}&-114.99z_{2e}+1027.5z_{3e}+0.2625\times10^{-4}u = 0\\&-1027.5z_{2e}-114.99z_{3e}-0.1307\times10^{-4}u = 0 .\end{aligned} \qquad (A.4)$$

Substituting (A.4) in (A.3),

$$\dot{R}^*\mathrm{Cos}\theta-R^*\mathrm{Sin}\theta\,\dot{\theta} = -\alpha R^*\mathrm{Cos}\theta + \beta R^*\mathrm{Sin}\theta \qquad (A.5(a))$$

$$\dot{R}^*\mathrm{Sin}\theta+R^*\mathrm{Cos}\theta\,\dot{\theta} = -\beta R^*\mathrm{Cos}\theta - \alpha R^*\mathrm{Sin}\theta . \qquad (A.5(b))$$

Dividing (A.5(a)) by (A.5(b)),

$$\frac{\mathrm{Cos}\theta dR - R^*\mathrm{Sin}\theta d\theta}{\mathrm{Sin}\theta dR + R^*\mathrm{Cos}\theta d\theta} = \frac{-\alpha^*\mathrm{Cos}\theta + \beta^*\mathrm{Sin}\theta}{-\beta^*\mathrm{Cos}\theta - \alpha^*\mathrm{Sin}\theta} . \qquad (A.6)$$

After some algebraic manipulations (A.6) becomes

$$\alpha R dR - \beta R^2 d\theta = 0 \qquad (A.7)$$

or

$$\frac{dR}{R} = \frac{\beta d\theta}{\alpha} . \qquad (A.8)$$

Integrating both sides of (A.8) and taking the exponentials,

$$R = R_0^*\exp[(\theta-\theta_0)\beta/\alpha]$$

$$= R_0 * \exp[(\theta - \theta_0)/8.93556] \ . \qquad (A.9)$$

From (A.2)

$$R = \sqrt{(z_2 - z_{2e})^2 + (z_3 - z_{3e})^2}$$

and $\theta = \cos^{-1}[(z_2 - z_{2e})/R]$.

The constant of integration R_0 and $_0$ are found to be

$$R_0 = \sqrt{(z_{20} - z_{2e})^2 + (z_{30} - z_{3e})^2}$$

$$\theta_0 = \cos^{-1}[(z_{20} - z_{2e})/R_0]$$

where (z_{20}, z_{30}) is the initial point.

APPENDIX B

THE SWITCHING CURVE.

The switching curve for the fast system (2.11) in general would consist of an infinite number of "semi-loops" as discussed in [4]. The first of these "semi-loops" on either side of the final point are portions of the phase-plane trajectories, so that any point on this part of the switching curve can be driven to the final point in no more than $\pi/8.9356$ sec. by applying a control of $u = \pm 5$. Let $\gamma = \gamma^+ \cup \gamma^-$ denote this portion of the switching curve where γ^+ is the portion that requires $u = +5$ and γ^- is the portion that requires $u = -5$ to drive a point on it to the final point.

In the steady state, the system state $x_3=0$ and owing to the system property, $x_2=0.5x_1$. Calculating the inverse of the transformation matrix T given by (2.2), the transformation equations are found to be

$$z_2=7.8763\times10^{-7}x_1-1.7311\times10^{-6}x_2-1.9841\times10^{-9}x_3 \qquad \text{(B.1(a))}$$

$$z_3=-3.9211\times10-7x_1+3.5797\times10^{-7}x_2-6.0815\times10^{-9}x_3 \ . \qquad \text{(B.1(b))}$$

Expressing (B.1(a)) and (B.1(b)) in terms of x_2 only in the steady state,

$$z_2 = -1.5585\times10^{-7}x_2$$

$$z_3 = -4.2625\times10^{-7}x_2 \ .$$

Therefore

$$z_3 = 2.735z_2 \ . \qquad \text{(B.2)}$$

Equation (B.2) implies that the initial and final points always lie on a straight line. Evidently the equilibrium points also lie on this line. Due to the following state constraints,

$x_2 \leqslant 0.2618$ rad (15 deg. deflection on each side)

$x_1 \leqslant 0.5236$ A

the transformed states are constrained to

$$z_2 \leqslant 4.0801 \times 10^{-8}$$
$$z_3 \leqslant 1.1159 \times 10^{-7} \ .$$

Now the equilibrium point due to the control of $u = 5V$ is $(z_2, z_3) = (4.8694 \times 10^{-8}, 1.3321 \times 10^{-7})$. This implies that due to the constraint on z_2 & z_3, all the initial and final points of the system in the transformed plane will lie on the straight line (B.2) between the two equilibrium points. Note that the portion of the switching curve denoted by $\gamma = \gamma^+ \cup \gamma^-$, encompasses the two equilibrium points, so that for all initial and final points only the portion γ of the switching curve is required. This also implies that for all feasible initial points there could be at most one switching.

The equation of the switching curve is the same as that for a particular portion γ of the phase-plane trajectory. The portion considered is that which passes through the desired final point and also every point on it can be forced to the final point in no more than $\pi/8.9356$ sec. Thus the equation of the switching curve is

$$R_s = R_{s0}*\exp[(\theta_s - \theta_{s0})/8.9356]$$

where

$$R_s = \sqrt{(z_2-z_{2e})^2 + (z_3-z_{3e})^2}$$

$$R_{s0} = [\sqrt{(z_{2f}-z_{2e})^2+(z_{3f}-z_{3e})^2}\,]*\exp(\pi/8.9356)$$

$$\theta_s = \mathrm{Cos}^{-1}[(z_2-z_{2e})/R_s]$$

$$\theta_{s0} = \mathrm{Tan}^{-1}(z_{30}/z_{20})=1.22 \text{ rad.}$$

APPENDIX C

PROGRAM FOR TIME-OPTIMAL ALGORITHM

```
C       GIVEN THE INITIAL AND FINAL TRACKS, THIS PROGRAM
C       CALCULATES THE TIME OPTIMAL CONTROL ITERATIVELY.
C       THE TIMES GIVEN BY THE PROGRAM ARE THE TIMES FOR
C       WHICH THE CONTROL SEQUENCE IS TO BE APPLIED. THE
C       CONTROL SEQUENCE IS ALSO GIVEN BY THE PROGRAM.
C       APPLYING THIS CONTROL TO THE SYSTEM WOULD STEER
C       THE SYSTEM TO WITHIN ±10 TRACKS OF THE DESIRED
C       FINAL TRACK.
C
C
C       ENTER INITIAL AND FINAL TRACKS.
C
10      TYPE 20
20      FORMAT(' INITIAL TRACK ?'/)
        ACCEPT 30,ITR
30      FORMAT(I4)
        TYPE 40
40      FORMAT(' FINAL TRACK?'/)
        ACCEPT 30,IFT
C
C       ITR = INITIAL TRACK,  IFT = FINAL TRACK
        DIST=ABS(ITR-IFT)
C
C       CHECK IF THE DESIRED DISTANCE TO BE TRAVERSED
C       IS MORE THAN 10 TRACKS.
        IF(DIST.LE.10)        GO TO 50
        GO TO 70
50      TYPE 60
60      FORMAT(' YOU ARE WITHIN 10 TRACKS.'/)
        GO TO 10
70      CALL STEP(DIST,ST)
C
C       FORM THE INITIAL AND FINAL STATES.
        X10=(ITR-2048)*0.2563476E-3
        X20=0.4986654839*X10
        X1F=(IFT-2048)*0.2563476E-3
        X2F=0.4986654839*X1F
        X3F=0.0
        CALL JORDAN(X1F,X2F,X3F,Z1F,Z2F,Z3F)
C
C       U IS THE CONTROL, EPS (EPSILON) IS THE
C       NEIGHBORHOOD OF THE SWITCHING CURVE,
C       SIGMA IS THE NEIGHBORHOOD OF THE FINAL
C       POINT AND DTF IS THE TIME FOR EULER
C       INTEGRATION.
        U=5.0
        EPS=0.2E-9
        SIGMA=.7E-8
```

```
      DTF=5.0E-7
C
C     FIND THE CONTROL FOR THE SLOW SURSYSTEM.
C
      Z10=1.457987E-7*X10+7.994377E-7*X20
      IF(Z10-Z1F)         80,90,90
80    U1=U
      GO TO 100
90    U1=-U
C     (Z2E,Z3E) IS THE EQUILIBRIUM POINT.
100   Z2E=4.869355725E-8
      Z3E=1.33206105E-7
C
C     CALCULATE THE INITIAL RADIUS OF THE SWITCHING
C     CURVE. THIS DEPENDS ON THE FINAL POINT.
C     R01 IS THE INITIAL "RADIUS" OF THE RIGHT HALF
C     OF THE SWITCHING CURVE AND R02 IS FOR THE LEFT
C     HALF.
C
      R11=SQRT((Z2E-Z2F)*(Z2E-Z2F)+(Z3E-Z3F)*(Z3E-Z3F))
      R01=R11*1.35764436
      R12=SQRT((Z2E+Z2F)*(Z2E+Z2F)+(Z3E+Z3F)*(Z3E+Z3F))
      R02=R12*1.35764436
      Z1I=Z1F
C     SET ITERATION COUNT TO ZERO.
      NITR=0
110   CONTINUE
C
C     CALCULATE THE TIME TS REQUIRED TO REACH THE
C     DESIRED VALUE OF Z1 (Z1I).
C
      ARG=(Z1I-6.81917676E-8*U1)/(Z10-6.81917676E-8*U1)
      TS=-(ALOG(ARG))/71.2696
      DTS=TS/1000.
      X1K=X10
      X2K=X20
      X3K=0.0
      I=1
C     APPLY THE SLOW CONTROL FOR TIME TS.
      DO 120 J=1,1000
      X1=X1K+DTS*(-266.667*X1K-5.5333*X3K+33.3333*U1)
      X2=X2K+DTS*X3K
      X3=X3K+DTS*(142860*X1K-285710*X2K-34.592*X3K)
      X1K=X1
      X2K=X2
      X3K=X3
120   CONTINUE
C
C     FIND THE LOCATION OF THE STATES IN THE
C     TRANSFORMED STATES AFTER THE SLOW CONTROL
C     HAS BEEN APPLIED AND FORM THE CONTROL
C     SEQUENCE.
C
      CALL JORDAN(X1,X2,X3,Z1,Z2,Z3)
      FLAG=0
      CALL FCNTRL(EQ,FLAG,R01,R02,Z2,Z3)
      IF(FLAG.EQ.1)          GO TO 140
      IF(EQ.LT.0)        GO TO 130
      U2=-U
      GO TO 160
130   U2=U
      GO TO 160
140   IF(EQ.LT.0)        GO TO 150
      U2=U
      GO TO 160
150   U2=-U
160   U3=-U2
      N=1
C     APPLY THE FAST CONTROL.
170   CONTINUE
      X1=X1K+DTF*(-266.667*X1K-5.5333*X3K+33.3333*U2)
      X2=X2K+DTF*X3K
      X3=X3K+DTF*(142860*X1K-285710*X2K-34.592*X3K)
      X1K=X1
      X2K=X2
      X3K=X3
      CALL JORDAN(X1,X2,X3,Z1,Z2,Z3)
C
C     CHECK IF SWITCHING CURVE IS REACHED.
C
      CALL FCNTRL(EQ,FLAG,R01,R02,Z2,Z3)
      EQ=ABS(EQ)
      IF(EQ.LE.EPS)          GO TO 180
      N=N+1
      GO TO 170
180   TF1=DTF*N
C     TF1 IS THE TIME FOR WHICH THE FAST CONTROL
C     IS APPLIED BEFORE SWITCHING.
      M=1
C     APPLY THE "SWITCHED CONTROL".
190   CONTINUE
      X1=X1K+DTF*(-266.667*X1K-5.5333*X3K+33.3333*U3)
      X2=X2K+DTF*X3K
      X3=X3K+DTF*(142860*X1K-285710*X2K-34.592*X3K)
      X1K=X1
      X2K=X2
      X3K=X3
C     CHECK IF THE NEIGHBORHOOD OF THE FINAL POINT
C     HAS BEEN REACHED.
      CALL JORDAN(X1,X2,X3,Z1,Z2,Z3)
      R=SQRT((Z2-Z2F)*(Z2-Z2F)+(Z3-Z3F)*(Z3-Z3F))
      IF(R.LE.SIGMA)          GO TO 200
      M=M+1
      GO TO 190
200   TF2=DTF*M
      IFTR=7822.785741*X2+2048
C
C     TF2 IS THE TIME FOR WHICH THE SWITCHED FAST
C     CONTROL IS APPLIED AND IFTR IS THE FINAL
C     TRACK REACHED.
```

```
      NITR=NITR+1
C     STOP IF MORE THAN 20 ITERATION IS REQUIRED.
      IF(NITR.GE.20)    GO TO 210
C
C     STOP IF WITHIN 10 TRACKS OF THE DESIRED VALUE
C     OTHERWISE ITERATE.
      MISS=ABS(IFT-IFTR)
      IF(MISS.LE.10)    GO TO 230
C     FIND THE NEW Z1I.
      Z1S=6.8192E-8*U3+(Z1F-6.8192E-8*U3)*EXP(71.26966*TF2)
      Z1I=6.8192E-8*U2+(Z1S-6.8192E-8*U2)*EXP(71.26966*TF1)
      Z1I=Z1I/ST
      GO TO 110
210   TYPE 220
220   FORMAT(' CONVERGENCE IS TOO SLOW.'/)
230   TYPE 240,TS,TF1,TF2
240   FORMAT(' TS=',F12.8,2X,'TF1=',F12.8,2X,'TF2=',F12.8)
      TYPE 250,IFTR,NITR
250   FORMAT(' TRACK= ',I7,'   OF ITR= ',I3)
      TYPE 260,U1,U2,U3
260   FORMAT(' THE CONTROL SEQUENCE IS ',3G)
      TYPE 270
270   FORMAT(' TYPE 1 TO START AGAIN,0 TO END'/)
      ACCEPT 280,I
280   FORMAT(I2)
      IF(I.EQ.1)        GO TO 10
      STOP
      END
C
C
C            SUBROUTINE FCNTRL (FAST CONTROL)
C     THIS ROUTINE DETERMINES THE POSITION OF A POINT
C     WITH RESPECT TO THE SWITCHING CURVE. IT IS USED
C     TO FORM THE FAST CONTROL SEQUENCE AND ALSO TO
C     DECIDE IF THE SWITCHING CURVE HAS BEEN REACHED.
C
      SUBROUTINE FCNTRL(EQ,FLAG,R01,R02,Z2,Z3)
      Z2E=4.869355725E-8
      Z3E=1.33206105E-7
      Z2BAR=Z2-Z2F
      IF(Z2BAR.GE.0)        GO TO 10
      Z2=-Z2
      Z3=-Z3
      R0=R02
      FLAG=1
      GO TO 20
10    R0=R01
20    A1=Z2-Z2E
      A2=Z3-Z3E
      A=SQRT(A1*A1+A2*A2)
      ARG=A1/A
      C=ACOS(ARG)
      IF(Z3.LE.Z3E)          C=-C
      D=R0*EXP((C-1.23057)/10.275)
      EQ=A-D
```

```
        RETURN
        END
C
C
C           SUBROUTINE STEP
C       THIS ROUTINE IS ESSENTIALLY A LOOK-UP TABLE.
C       GIVEN THE NUMBER OF TRACKS TO BE MOVED, THE
C       ROUTINE GIVES THE VALUE OF THE STEP SIZE.
C
        SUBROUTINE STEP(DIST,ST)
        DIMENSION A(16),B(16)
        DATA(A(I),I=1,5)/160,250,350,450,500/
        DATA(A(I),I=6,10)/600,800,850,900,1000/
        DATA(A(I),I=11,16)/1050,1950,2050,2200,4000,4095/
        DATA(B(I),I=1,5)/1.0,1.01,1.02,1.03,1.035/
        DATA(B(I),I=6,10)/1.04,1.045,1.035,1.03,1.02/
        DATA(B(I),I=11,16)/1.01,1.0,1.01,1.02,1.0,1.01/
        DO 10 K=1,16
        ST=B(K)
        IF(DIST.LE.A(K))    GO TO 20
10      CONTINUE
20      RETURN
        END
C
C
C       SUBROUTINE JORDAN
C       THIS ROUTINE FINDS THE TRANSFORMED STATES GIVEN
C       THE SYSTEM STATES.
C
        SUBROUTINE JORDAN(X1,X2,X3,Z1,Z2,Z3)
        Z1=1.457987E-7*X1+7.994377E-7*X2+1.994189E-10*X3
        Z2=7.876269E-7*X1-1.731102E-6*X2-1.98409E-9*X3
        Z3=-3.9211127E-7*X1+3.579705E-7*X2-6.081508E-9*X3
        RETURN
        END
```

APPENDIX D

PROGRAM FOR 'REAL-TIME' ALGORITHM

```
C       THIS PROGRAM CALCULATES THE TIME OPTIMAL
C       CONTROL, FOR THE OPTICAL MEMORY DEVICE,
C       IN REAL TIME. IT UTILIZES A LOOK-UP TABLE
C       CONTAINING 4096 SOLUTIONS FOR GOING FROM
C       TRACK 1 TO ANY OTHER TRACK ON THE DISC.
C       IT ALSO REQUIRES A TABLE OF "LOSS FACTORS.
C
C       ENTER INITIAL AND FINAL TRACK.
10      TYPE 20
20      FORMAT(' INITIAL TRACK?'/)
        ACCEPT 30,ITR
30      FORMAT(I6)
        TYPE 40
40      FORMAT(' FINAL TRACK?'/)
        ACCEPT 30,IFT
C       ITR = INITIAL TRACK, IFT = FINAL TRACK
C
C       FORM THE CONTROL SEQUENCE.
        U=5.0
        IF(ITR-IFT)  50,60,60
50      U1=U
        GO TO 70
60      U1=-U
70      U2=-U1
        U3=U1
        ITR1=ITR
        IFT1=IFT
C       FIND THE NUMBER OF TRACKS TO BE MOVED.
        IDIST=ABS(IFT-ITR)
        IMOV=IDIST
C
C       IF MOVEMENT IS FROM HIGHER TRACK NUMBER
C       TO LOWER TRACK, MAKE USE OF SYMMETRY.
        IF(IFT.LT.ITR)  ITR=4096-ITR
C
C       FIND ITERATIVELY THE SOLUTION TO BE
C       PICKED FROM THE LOOK-UP TABLE.
        DO 80 KK=1,5
        TAB=IDIST/20.
        ITAB=INT(TAB)
        FRAC=TAB-ITAB
        IF(FRAC.GE.0.5)  ITAB=ITAB+1
        CALL LOSFAC(ITAB,ALOSS)
        IADD=ALOSS*ITR
80      IDIST=IMOV+IADD
C
C       IDIST IS THE REQUIRED TRACK NUMBER.
C       THAT IS, THE CONTROL REQUIRED FOR
```

```
C        GOING FROM TRACK ITR TO TRACK IFT IS
C        THE AS FOR GOING FROM TRACK 1 TO TRACK
C        IDIST.
C
C        ASSUME THE 4096 SOLUTIONS ARE STORED IN
C        THE MEMORY WHICH CAN BE ACCESSED THROUGH
C        SOME SUBROUTINE LOOKUP. NOTE THAT THE
C        PREVIOUS PROGRAM TOPT.FOR CAN BE USED TO
C        GENERATE SUCH A TABLE.
C
         CALL LOOKUP(IDIST,TS,TF1,TF2)
         TYPE 90,TS,TF1,TF2
90       FORMAT(' THE TIMES ARE = ',3G)
         TYPE 100
100      FORMAT(' TYPE 1 TO RESTART,0 TO END'/)
         ACCEPT 110,K
110      FORMAT(I2)
         IF(K.EQ.1)  GO TO 10
         STOP
         END
C
C
C             SUBROUTINE LOSFAC (LOSS FACTOR)
C        THIS ROUTINE IS A 'LOOK-UP TABLE' FOR THE
C        LOSS FACTOR. GIVEN THE DESIRED DISTANCE TO
C        BE MOVED, THIS ROUTINE FINDS THE ASSOCIATED
C        LOSS FACTOR.
C
C
         SUBROUTINE LOSFAC(ITAB,ALOSS)
         DIMENSION AR(205)
         DATA(AR(J),J=1,7)/.07,.09,.11,.125,.14,,.16/
         DATA(AR(J),J=8,14)/.165,.18,.18,.185,.19,.195,.205/
         DATA(AR(J),J=15,21)/.21,.215,.22,.23,.23,.235,.24/
         DATA(AR(J),J=22,28)/.24,.24,.25,.26,.26,.27,.265/
         DATA(AR(J),J=29,35)/.27,.27,.275,.28,.28,.285,.29/
         DATA(AR(J),J=36,42)/.295,.29,.3,.3,.3,.305,.3/
         DATA(AR(J),J=43,49)/.31,.31,.315,.32,.32,.325,.33/
         DATA(AR(J),J=50,56)/.33,.33,.335,.34,.34,.335,.34/
         DATA(AR(J),J=57,63)/.34,.34,.345,.345,.34,.34,.345/
         DATA(AR(J),J=64,70)/.35,.35,.35,.35,.35,.355,.35/
         DATA(AR(J),J=71,77)/.35,.355,.36,.355,.35,.355,.35/
         DATA(AR(J),J=78,84)/.36,.36,.37,.38,.395,.4,.415/
         DATA(AR(J),J=85,91)/.42,.43,.44,.45,.455,.46,.47/
         DATA(AR(J),J=92,98)/.475,.48,.485,.5,.5,.505,.51/
         DATA(AR(J),J=99,105)/.51,.515,.52,.525,.53,.535,.535/
         DATA(AR(J),J=106,112)/.54,.54,.545,.55,.555,.555,.55/
         DATA(AR(J),J=113,119)/.56,.56,.565,.57,.57,.57,.57/
         DATA(AR(J),J=120,126)/.57,.58,.575,.57,.58,.58,.58/
         DATA(AR(J),J=127,133)/.585,.58,.59,.595,.6,.605,.61/
         DATA(AR(J),J=134,140)/.62,.625,.63,.64,.65,.65,.66/
         DATA(AR(J),J=141,147)/.67,.675,.68,.68,.685,.59,.695/
         DATA(AR(J),J=148,154)/.7,.705,.71,.705,.71,.72,.71/
         DATA(AR(J),J=155,161)/.72,.71,.725,.72,.72,.73,.725/
         DATA(AR(J),J=162,168)/.735,.74,.74,.74,.75,.75,.76/
DATA(AR(J),J=169,175)/.77,.78,.78,.785,.79,.795,.805/
DATA(AR(J),J=176,182)/.81,.805,.81,.81,.81,.82,.82/
DATA(AR(J),J=183,189)/.825,.83,.83,.835,.84,.85,.85/
DATA(AR(J),J=190,196)/.86,.865,.87,.87,.87,.87,.88/
DATA(AR(J),J=197,203)/.88,.885,.89,.9,.91,.9,.91/
DATA(AR(J),J=204,205)/.92,.92/
ALOSS=AR(ITAB)
RETURN
END
```

APPENDIX E

THE 8080A BASED MICROCOMPUTER SYSTEM.

E.1 Introduction.

The purpose of this Appendix is to provide adequate information on the existing 8080 microcomputer system (its hardware & software) built around intel 8080A CPU. Here an effort has been made to collect the important information pertaining to the system's hardware and software and present it with some comments on its functional aspect. This Appendix may be considered as a documentation on the existing hardware and software. The Appendix has been, basically, divided into two sections: Hardware and Software.

The hardware section gives an overview of the system organization. It deals with the layouts and functions of each of the boards in the system, i.e. the different chips being used, their interconnections, control & addressing scheme. A detailed description of the various chips can be found in [1] & [2]. The figures included here are in block diagram form only showing the important connections. The software section deals with the functional description of the utility routines available on the ROM, their locations, etc. A complete listing of these utility routines is given at the end.

E.2 Hardware.

The system consists of the following four boards:

i) 8080A CPU, Clock and Controller.

ii) 8K PROM

iii) 2K RAM

iv) ADC-DAC Interface Board

Each of these boards will be described individually and then the overall system organization will be described in block diagram form. The CPU-Controller and memory layouts are standard in the sense that these three boards are laid out, more or less, as suggested in the Intel 8080 User's Manual. A brief description of these boards will be given here, however, the ADC-DAC board will be described in some detail.

i) 8080A CPU, Clock and Controller:

This board (Fig. E.1) is the heart of the system as it has the CPU chip on it. The 8080A CPU is a 8-bit parallel processor. It has a 16-bit address bus so that upto 64K 8-bit words can be addressed. The CPU provides the following signals:

SYNC - signal to indicate the beginning of each machine cycle.

DBIN(Data Bus In) - signal to indicate that the data bus is in input mode.

WR (Write) - this signal is used for memory write or I/O

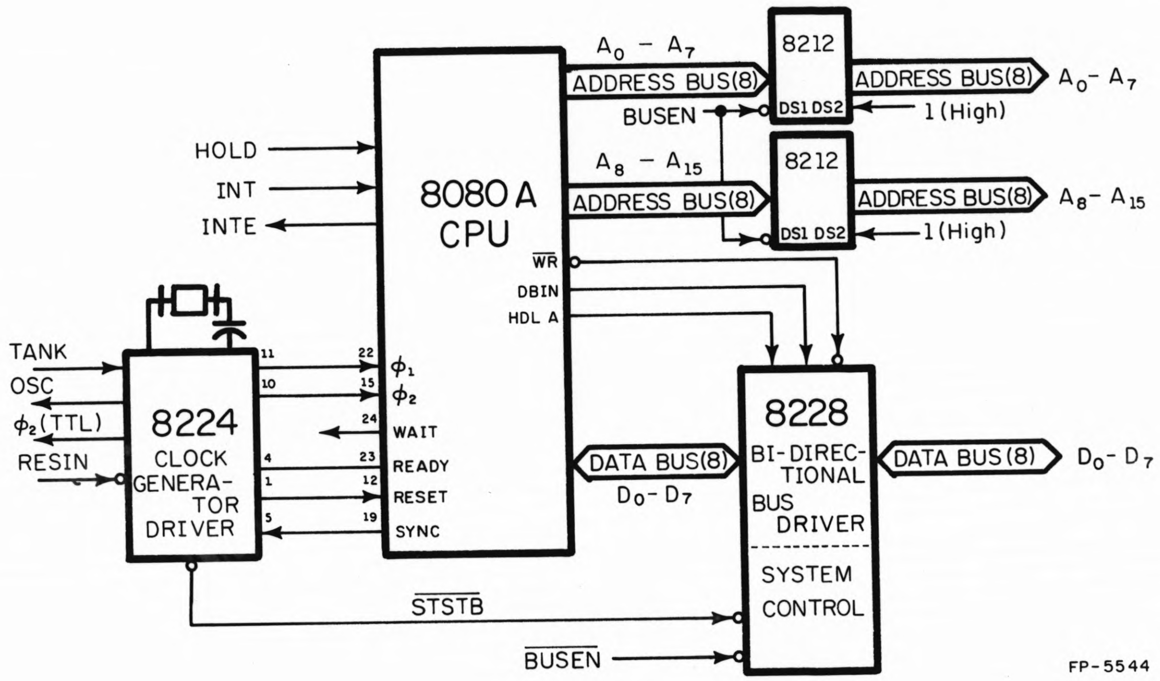

Fig. E.1 CPU, Clock and Controller.

output.

WAIT -signal to indicate CPU is in wait state.

INTE (Interrupt Enable) - indicates content of interrupt flip/flop.

HLDA(Hold Acknowledge) - indicates CPU is in hold state.

The CPU accepts the following signals for its proper operation :

READY - indicates to the CPU that valid input data is available on the 8080A data bus.

HOLD - requests CPU to enter hold state.

INT (Interrupt) - signal to the CPU for an interrupt request.

RESET - this signal when activated (3 clock cycle min) clears the program counter and returns the program to the begining of the monitor.

The 8224 chip is used as the clock generator/driver for the CPU. The oscillator circuit in the 8224 derives its basic operating frequency from an external crystal. The chip has a divide by 9 counter so that the oscillator operates at 9 times the desired processor speed. A 18.432mHz crystal is used so that the 8080A runs at 2.048mHz. Since the frequency is high (>10mHz) a 10pF capacitance is connected in series with the crystal which has the effect of "trimming" the frequency to obtain the desired frequency. The 8224 provides the CPU with the two phase clock (ϕ_1&ϕ_2) which are driven internally and are directly interfaced to the CPU. A power-on reset signal is also provided by the 8224 and is connected to the reset of the CPU for initialization of the system. The 8224 recieves a SYNC (synchronization) signal from the CPU. Gating this signal with the phase 1 (ϕ_1) of the clock it produces a STSTB (status strobe) signal for the system controller (8228).

The function of system controller and data bus driver for the 8080A is performed by the 8228 chip. The 8228 generates all control signals to interface directly with RAM, ROM and I/O components. The 8080A data bus is connected to memory & I/O devices through the 8-bit bi-directional bus driver in the 8228 chip. The bi-directional data bus is controlled so that proper bus flow is maintained. The STSTB of the controller(8228) is connected to the STSTB signal of the Clock/Driver chip(8224). The controller uses the STSTB signal to latch the "status" information issued by the CPU. This status information alongwith HLDA, DBIN & WR signals from the CPU is used to generate control signals (MEMR, MEMW, I/OR, I/OW & INTA).

The 16 bit address issued by the CPU is latched and buffered by two 8212 chips (8-bits each). The device select bit DS_2 of 8212 is set high and bit DS_1 (of 8212) is connected to BUSEN (Bus Enable) signal, so that the device is selected only if the bus is enabled (active low).

ii) <u>Read Only Memory:</u>

8K of Read Only Memory is available on this board (Fig. E.2). There are eight 8708 EPROMS(Erasable Programable Read Only Memory) with 1K X 8 bits each. The 16 bit address from the CPU can be decoded to address any byte in the memory bank. The addressing scheme is as follows. the lower 10 bits (A_0-A_9) is used to address the ROM chip. Bits A_0-A_9 of the CPU address bus is connected to A_0-A_9 bits of each ROM through a buffer (8212). The next 3 bits (A_{10}-A_{12}) are inputs to a "1 out of 8 binary decoder" (8205). The output of the decoder is connected to the CS (Chip Select) input of the ROMs (8 output of decoder, 1 to each ROM) so that any one of the eight ROM chips on the board can be selected by specifying these bits (A_{10}-A_{12}). The higher 3 bits (A_{13}-A_{15}) of the address bus is used for 'board select'. The board select is done by setting a switch dip on the board and comparing it with bits A_{13}-A_{15} using a 4-bit comparator (SN7485). Bit 6 (high if the two inputs are equal) of the comparator is connected to the enable of the decoder. Thus when the bits specified by A_{13}-A_{15} matches the ones set on the board (by switch dip) the decoder is enabled which in turn selects one of the 8 ROM chips on the board (specified by A_{10}-A_{12}) and the particular byte on the ROM is selected by A_0-A_9. Note that MEMR & MEMEN (Memory Enable) are connected to the other two enable input (E_1 & E_2) of the decoder, so that in order to enable the

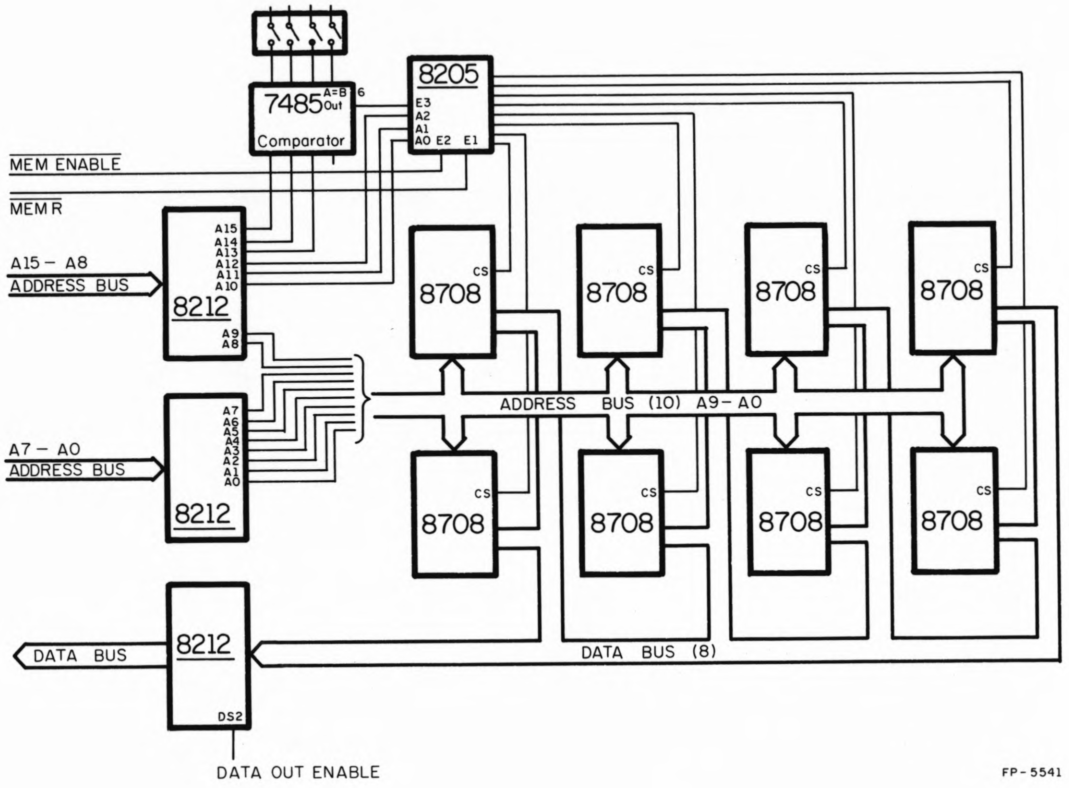

Fig. E.2 8K PROM Board.

decoder the CPU must be in the right mode (that is, MEMR & MEMEN signals must both be "low").
The switch dip for this board is set to 0000. The existing software in the ROM is:

Location 00H to 3FFH (1K) ------ 8080 MOnitor

Location 400H to 7FFH (1K) ------ Utility Routines (to be described later)

iii) Random Access Memory:

There are sixteen 8102A-4 RAM chips on this board (Fig. E.3), each with 1K X 1 bits of memory, so that 2K X 8 bits of Random Access Memory is available on this board (some boards might have only 1K of RAM). The 16-bit address from the CPU is latched and buffered by two 8212 chips(8-bits each). The addressing scheme is somewhat similar to the one used for the ROMs. The lower 10-bits of the address bus (A_0-A_9) is connected to the address inputs (A_0-A_9) of each RAM chip and is used to address a particular bit in the RAM. The higher 5-bits (A_{11}-A_{15}) is used for board select. The switch dip is set to a particular value which is a characteristic of the board (for this board the 5 bits on the switch dip is set to 00100). These bits are compared to the bits A_{11}-A_{15} of the address bus using two 4-bit comparators (SN7485). If there is a match, a signal is issued by the comparator (bit 6) which is gated with bit A_{10} of the address bus to produce an enable signal for the

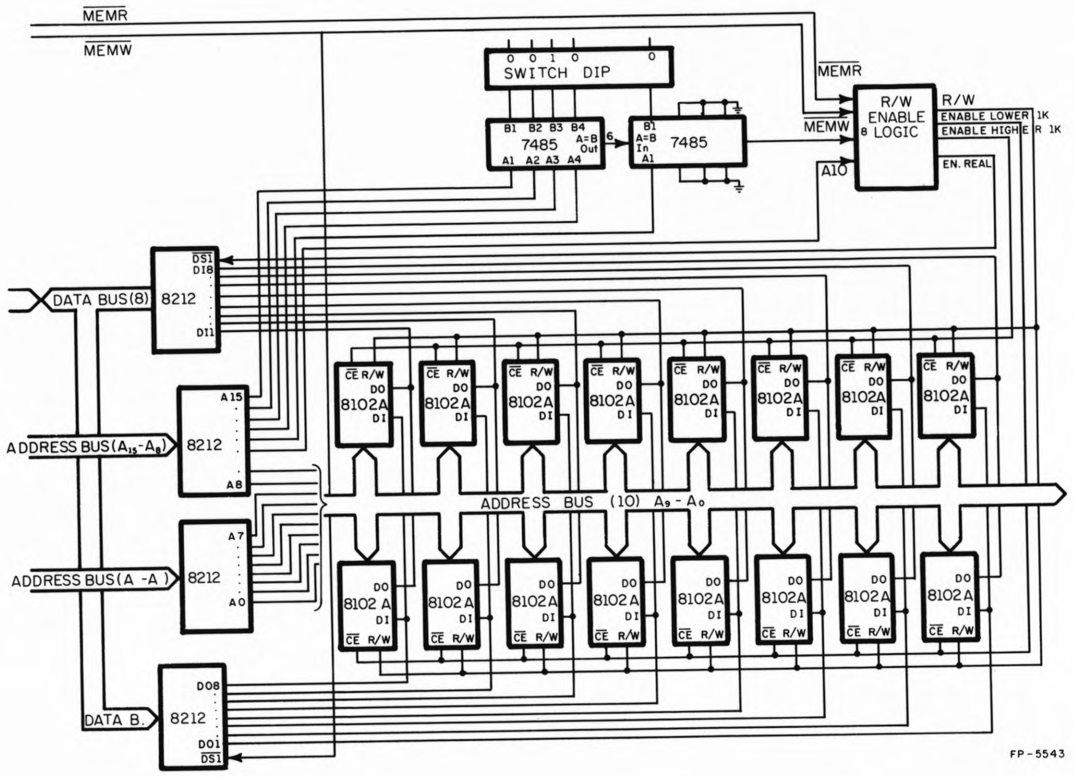

Fig. E.3 2K RAM Board.

board. Bit A_{10} is used to enable 1K of memory (either the higher or the lower 1K of the memory is enabled at a time). With $A_{10}=0$ the lower 1K of memory is selected and with $A_{10}=1$ the higher 1K of memory is selected (ofcourse to select any memory the result of the "board select" comparison must be high (bit 6 = 1)). With the above set up the address of the RAM goes from 2000H to 27FFH (2K of memory) or from 2000H to 23FFH (if only 1K of memory is available).

This being random access memory the flow of data is bi-directional (from and to the 8080A). The 8-bit data bus from the CPU is connected to two unidirectional (8212) latch/buffers. One of these 8212 is for "read" instruction (data from memory to CPU) and the other for "write" instructions (data from the CPU to memory). One of these chips is selected depending on whether a "read" or "write" instruction is being carried on. The 'MEMR' and 'MEMW' signals provided by the controller is used to select the particular chip.

iv) <u>ADC-DAC</u> <u>Interface</u> <u>Board:</u>

Interfacing the 8080 system is made very easy through the use of the 8255 parallel interface chip. This chip is used to interface the DAC to the 8080A CPU (Fig. E.4). The 8228 system controller is used as an interface to the 8080 data bus. For I/O devices, this implies that when data is being requested by the 8080 the I/O R signal is active

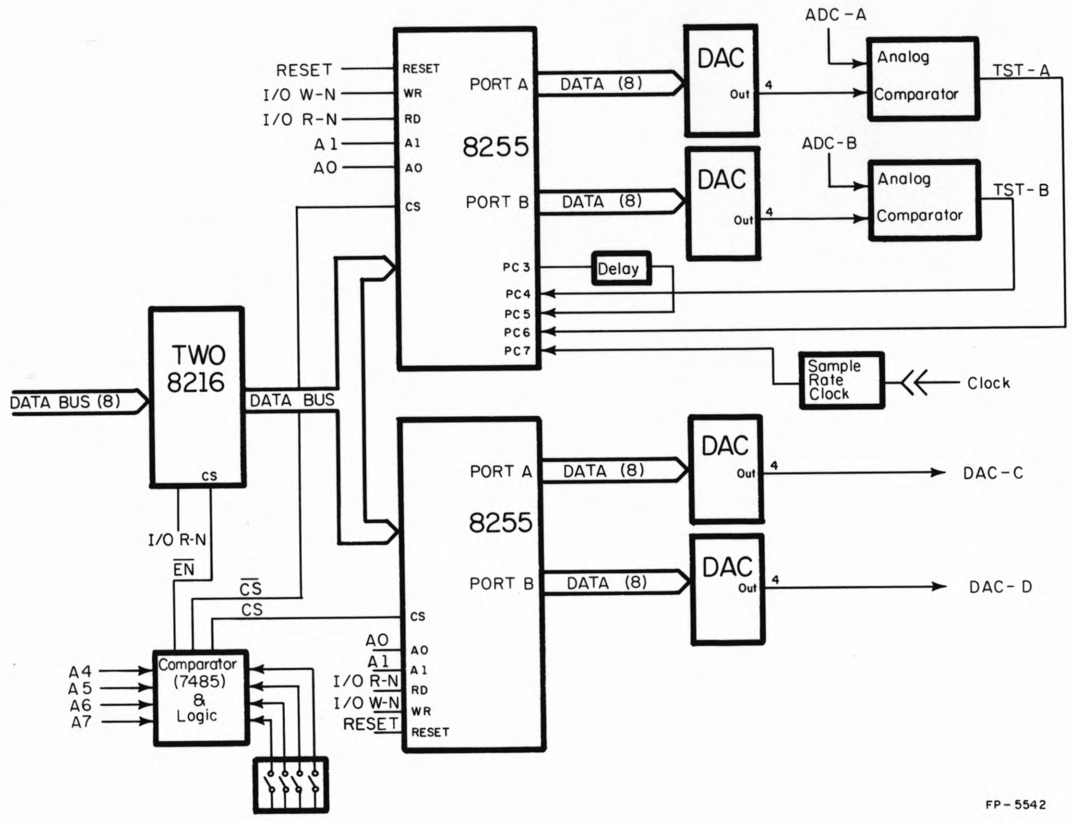

Fig. E.4 ADC-DAC Interface Board.

(=low). If data is being output to the I/O device, the I/O W signal is active (=low). The lower half of the address bus (A_0-A_7)will be decoded to specify which device the 8080 is trying to access. These actions of the hardware allow the operation of the 'IN' and 'OUT' instructions.

The heart of the interface is the 8255. It provides three ports: ports A and B can be unidirectional or bi-directional with latched or unlatched input and output. The third, port C, can be split in half to provide four bits of input and output or it can be used an unlatched input and/or output port. The configuration for a particular application is specified by user software. In this application, ports A and B have latched outputs with unlatched input. Port C is split in half to allow flow of control signals from and to the 8080. For our operation the following port assignment is required:

Port A	--------	Output
Port C (Upper, PC_7-PC_4)	-----	Input
Port C (Lower, PC_3-PC_0)	-----	Output
Port B	--------	Output

The control word for this assignment is 88H. In order to set up the ports in the above manner the control word (88H) is "output" by the CPU to the control word register under software instructions. Therefore to set up the ports the following two instructions must be executed:

```
        MVI    A,88H
        OUT    CNTRL
```

(where 'CNTRL' should be equated to the address of the Control Word Register of the 8255 in question)

Since there are 4 DACs it is necessary to use two 8255s, each placed in the 88H mode. Having two 8255 chips there are 6 ports (3 in each) and two control word registers (CNTRL), one for each 8255. The first (upper) 8255 is connected to the pair of DACs that under software control performs the function of ADCs. The second (lower) 8255 is connected to the pair of DACs that performs its normal function of D/A conversion. The 3 ports and the control word register of the upper 8255 connected to the "ADC" has the following addresses:

Port A	-------	04H
Port B	-------	05H
Port C	-------	06H
CNTRL	-------	07H

The 3 ports and Control Word Register of the lower 8255 connected to the "DAC" has the following addresses:

Port A	-------	0CH
Port B	-------	0DH
Port C	-------	0EH
CNTRL	-------	0FH

However, control signals need only be passed through one of the C ports with the other being unused. Specifying a port of an 8255 is done via the lower two bits of the address bus, AB_0 and AB_1. A detailed bit assignment is given in [1]. Bits AB_0 and AB_1 are connected to the A_0 and A_1 inputs of each 8255. An 8255 chip is be seclected using bit three of the address bus (AB_3), which being the input to one of the chip select inputs of the 8255's and its complement the input to the other, so that only one of the 8255 is selected at a time. To allow more than one DAC boards, the remaining bits of the address bus is decoded as a board select with the exception of bit two. This bit is used to distinguish between serial and parrallel interface boards. Since the 8255 sinks only 1.6mA it is necessary to buffer its connections to the 8080 data bus. This is done with the 8216 bi-directional bus driver chip which can sink 50mA per input. A chip select and data in enable signal are required for its use. The chip select is made active when either 8255 is selected. The data in enable input is tied to the I/O W signal.

There now remains the details of interface to the DAC chips themselves. In order to have bipolar operation using two's complement number representation inverters are placed on the lower seven digital inputs of the DAC chips. Sample rate is defined by bit 7 of Port C of the upper 8255. It is assumed to be high active and can be tested using an 'IN' instruction and masking the result with 80H.(See Subroutine SPWT). The remainder of Port C is used for signals to implement a software analog to digital conversion routine (See Subroutine CVRT). The 8080 sends out a digital word which (after conversion to analog by the upper DAC) is compared to the analog signal to be converted (to digital). This analog comparison is done using 741 Op-Amps. The result of the comparison is sent to bit 6 of Port C (of the upper 8255). The CVRT routine checks this bit and depending on the result of the comparison, the routine either increases or decreases the value of the next word it puts out (for further details see Subroutine CVRT). The bit assignment of Port C of the first (upper) 8255 is as follows:

<u>Bit 7:</u> Sample time flag. This bit is connected to the clock so that it goes high at every sampling time. It is used by Subroutine SPWT (wait for sampling time).

<u>Bit 6:</u> The result of comparison of two analog signals (the one to be converted and the other "guessed" by the CPU).

If 1 : Input to ADC-A>DAC-A Output.

If 0 : Input to ADC-A<DAC-A Output.

Note that an ambiguity occurs when the two signals are equal, in which case the result of the comparator is arbitrary.

<u>Bit 5:</u> Pause flag. This bit is connected to bit 3 through some "gates" acting as hardware delays. It is tested (by Subroutine WAIT) after toggling bit 3. The

purpose of this delay is to provide the op-amp comparator enough time to perform the comparison.

<u>Bit 4:</u> The function of this bit is similar to bit 6. Here the result is from the comparison of ADC-B & DAC-B instead of 'comparison' of ADC-A & DAC-A.

If 1 : Input to ADC-B > DAC-B Output

If 0 : Input to ADC-B < DAC-B Output

<u>Bit 3:</u> Toggle for 'hardware delay'.

The only remaining important thing to be discussed on this board is the I/O communication interface, that is, connecting up the 8080 to a TTY or a CRT, etc. The 8251, a Universal Synchronous/Asynchronous Receiver/Transmitter (USART), is used for I/O communication interface this chip accomodates all the data communication required by the 8080 and is programmed by the CPU. the USART accepts data characters from the CPU in parallel format and then converts them into a continuous serial data stream for transmission (to the TTY or CRT). Simultaneously it can receive serial data streams (from TTY or CRT) and convert them into parallel data for the CPU. The rate of transmitting and receiving data is governed by the Buad Rate Generator (which is simply a divider which receives the clock ϕ_2(TTL) from the clock generator (8224)). Further information on baud grate generator and baud rate selection can be found in [2]. The overall system can be best understood by refering to the complete system diagram (Fig. E.5). The four boards

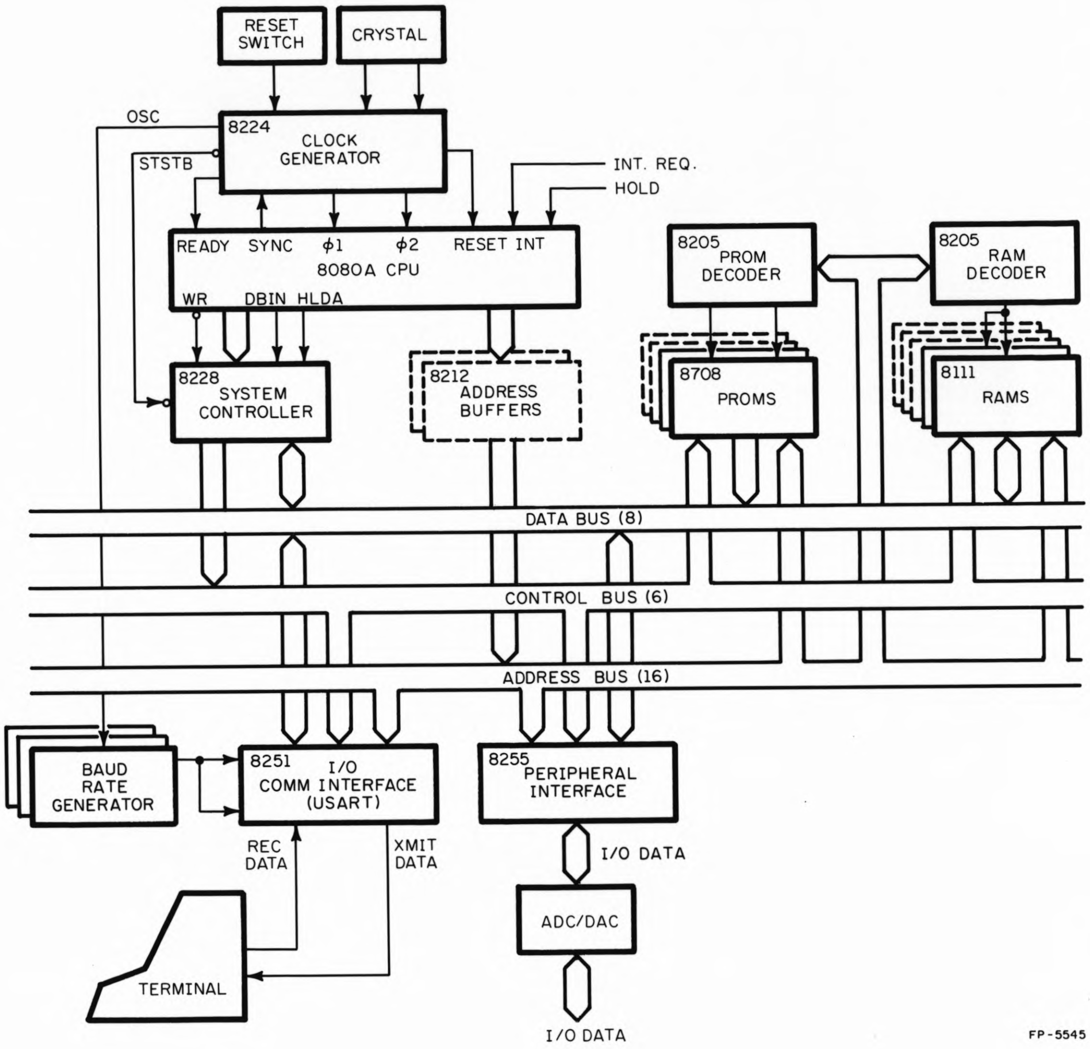

Fig. E.5 System Functional Block Diagram.

described above are connected together. The CPU is connected to the memory (ROM & RAM) by data and address buses. The CPU also communicates with the outside world through the interface board (8255, 8251 & DACs in our case). This essentially completes the brief discussion about the hardware layouts. A more detailed description on the various signals available (from 8080 & other chips) and the timing diagram can be found in [1].

E.3 Software.

Besides the very powerful instruction set for the 8080, there is quite a bit of software available in the form of subroutines in the ROM. User can access these subroutine by simply using the 'CALL' instruction. It is necessary to know the locations of these subroutines in the ROM in order to initiate these calls. The 8080 monitor resides in the first 1K of the memory (locations 00H to 3FFH) and the second 1K is taken up by the utility routines (locations 400H to 7FFH). The listing and description of the monitor routines is given in [2]. The monitor and the utility routines assume certain variables to be present at certain fixed locations in the memory. The meaning of these variables will become more apparent as they are discussed in context with various subroutines below. Here a list of those variables with their locations is given for quick reference:

Name	Description	Subroutine	Location
DIM	Length of Vector	VCMLT	2314H
AD1	Pointer to 1st Vector	VCMLT	2315H - 2316H
AD2	Pointer to 2nd Vector	VCMLT	2317H - 2318H
XSIZE	Incr. for X-axis	GRID	2319H
YSIZE	Incr. for Y-axis	GRID	231AH
DCNT	# pts. to be plotted	DSPLY	231BH
DXPND	# times a pt.to be plotted	"	231CH

The monitor save area is in locations 23EDH thru 23FCH. User RST 7 branches to location 237DH. The stack pointer is random with power-on (or RST 1) and must be initialized to a higher location in RAM (23ECH) so that it does not interfere with the user's programs in the RAM. The locations of the utility routines are listed in Table 1 and can be used for quick reference.

Table 1

Subroutine	Location (Address)
YAD	0412H
XAD	0424H
GRID	0436H
DSPLY	04CBH
TRANS	04ECH
FADD	0561H
FMULT	05E4H
MLT	0649H
VCMLT	0670H
TST	06ADH
INVRT	06C2H
TRUNC	06EDH
DSFT	070BH
CNVRT	0738H
CNRT	0759H
CVRT	077EH
SPWT	07E6H

The functional description of the available subroutines are given below. A complete program listing can be found at the end of this Appendix.

Warning: In using these routines, it should be noted that the content of the registers are destroyed (altered) by the subroutines, so that the content of a certain register does not necessarily remain the same after a subroutine has been called.

Subroutine CTS

Operand in accumulator.

Result is placed in D-E pair.

This routine converts the eight bit two's complement number in the accumulator to a ten bit offset binary number in the D-E register pair. This routine is used by routines YAD & XAD.

Subroutine YAD

Operand in accumulator.

Result: output to the graphics display.

This routine calls subroutine CTS to convert the eight bit two's complement operand in the accumulator to a ten bit off set binary operand in the D-E pair. The routine then takes this operand and adds the codes for HIY (20H) to register D and LOY (60H) to register E. The routine then calls the monitor routine 'CO' to output the result to the graphics display. This routine is used by routines GRID & DSPLY to draw vertical vertors.

Subroutine XAD (0424H)

Operand in accumulator.

Result: output to graphic display.

This routine is very similar to the YAD routine except that it is used by the GRID and DSPLY routines to draw horizontal vectors instead of vertical.

Subroutine GRID (0436H)

Initialization:

XSIZE (2319H), The step size along X-axis.

YSIZE (231AH), The step size along Y-axis.

Equate GRCMD to 1DH. This character (GRCMD) puts graphics display in graph mode.

Equate TTCMD to 1FH. This character (TTCMD) is the control word to put the graphics display in TTY mode.

This routine generates a grid of specified size as an aid in reading the graph. This subroutine is not normally called by another routine (that is, it is generally not a part of the program). After its execution it returns to the monitor.

Subroutine DSPLY (04CBH)

Operand (point to be plotted) in register B.

Initializations:

Put terminal in graphics mode:

```
        MVI  C,1DH
        CALL CO
```

Set iteration count to 80H:

```
        MVI  A,80H
        STA  DCNT
```

Set # times a sample to be repeated (normally 1):

```
        MVI  A,XPND
        STA  DXPND
```

Specify the location for DCNT & DXPND:

```
    DCNT   EQU   231BH
    DXPND  EQU   231CH
```

The above mentioned initialization must be done before calling this routine. This routine plots the content of register B (which is updated after a point has been plotted) at 256 points, that is, the routine quits after plotting 256 points and returns to the point from which it was called.

Subroutine TRANS (04ECH)

Transfer of data from DEC-10 to 8080.

The pointer to the beginning of the ram area where data is to be loaded is placed in the H-L pair (prior to calling the routine). This routine transfer data from the DEC-10 to the 8080 (down line loading). The routine stays in the wait loop until it receives the character 'I' (ignoring all other characters before that). The routine then assumes all the succeeding data to be ASCII encoded hex digits, at all times ignoring null characters. Each successive pairs of ASCII characters received is decoded into a pair of hex digits and placed in the location pointed to by the H-L register pair. The H-L pair is incremented after every two hex characters have been received. The process is terminated by a '$' (shift-4).

Down Line Loading:

In order to transfer a set of data from DEC-10 (at CSL) to the 8080 the following is done:

1) Set the stack pointer to a desirable location (23ECH) in the RAM (can be done by using monitor command XS).

2) Place the address of the beginning of the RAM where data is is to be loaded in the H-L pair (can be done by using monitor command XM).

3) Execute subroutine TRANS and while it is waiting for the data (from the DEC-10), turn the switch to connect the terminal to the DEC-10.

4) Put the data to be transfered in file named 'PROG.DAT' and then type the following:

```
.TT NO CRLF (no carraige return or line feed)
```

5) The program can now be transferred by typing:

```
.SUB 8080A.C**:OUT/NOLOG
```

The two asteriks after 'C' is the terminal number.

6) Logoff (K/N) from the DEC-10 and immediately flip the

switch to the 'LOAD' position. After the data has been loaded in the 8080 (it will be displayed on the screen) type in a '$' (Shift-4) to terminate the TRANS routine.

Subroutine FADD (0561H)

Floating point addition.

Addend: B-C pair.

Augend: D-E pair.

Result is in B-C pair.

This routine takes the contents of register pairs B-C & D-E and forms their sum using floating point arithmetic. The result is placed in the B-C pair.

Subroutine FMULT (05E4H)

Floating point multiplication.

Multiplicand:	B-C pair.
Multiplier:	D-E pair.
Result:	B-C pair.

This routine takes two floating numbers (content of B-C & D-E pairs) and forms their product. The result is placed in B-C pair.

Subroutine VCMLT (0670H)

Vector multiplication.

This routine assumes the following initialization:

Length of the vectors (DIM): at 2314H

Pointer to the first vector (AD1): at 2315H-2316H

Pointer to the second vector(AD2): at 2317H-2318H

This routine accepts two vectors and forms the sum of their products. The corresponding elements in the two vectors are multiplied and their sum is formed. Note that the two vectors are of the same length. The result is placed in the B-C register pair.

Subroutine TST (06ADH)

This routine compares the operand in the B-C pair to 0.75.

Result:

If operand > 0.75,	no action taken.
If operand < 0.75,	return to monitor.

Note that the stack pointer must be reinitialized after use of this routine since there may remain a return address on the stack.

Subroutine INVRT (06C2H)

Invert the positive operand.

Operand to be inverted: B-C pair.

Result: B-C pair.

This routine assumes that the operand to be inverted is positive. It places the result in the B-C pair.

Subroutine TRUNC (06EDH)

Truncate a floating point number.

Operand to be truncated in B-C pair.

Result is placed in B-C pair.

This routine takes a floaing point number in the B-C pair and truncates it to a eight bit fixed point number.

Subroutine CNVRT (0738H)

Operand: Register B

Result: B-C pair.

This routine converts a fixed point number in register B to a floating point number. The result is placed in B-C pair.

Subroutine CNRT (0759H)

Operand: B-C pair.

Result: Register B

This routine converts a floating point number in the B-C pair to a fixed point number and places the result in register B.

Subroutine CVRT (077EH)

Software analog to digital conversion.

Register assignments:

Current ADC-A result: Register B

Current ADC-B result: Register C

nth increment: Register D

Status of ADC (Port C): Register E

In addition to the register an iteration count is located at 22BDH. This routine uses the upper (first) parallel port chip (8255) to perform a binary successive approximation of two analog signal at the same time. The routine first checks for the 'sign' of the analog signal (tests bit 6 for ADC-A) and accordingly sets the sign bit (bit 7 in register B for ADC-A) to 1 or 0. It then sets the next higher order bit (bit 6) to 1 and outputs this number to the DAC. The digital number after conversion to analog by the DAC, is compared to the actual analog signal to be converted. Depending upon the result of the comparison bit 6 is either left as 1 or reset to 0. Next, bit 5 (of register B for ADC-A) is set to 1 and a similar operation is repeated as for bit 6 and so on. Thus the whole procedure of getting the digital number approximately equal to the analog signal, 8 iterations are required. The conversion of the second analog signal (ADC-B) is performed in a similar way except that the result is placed in register C. The time required for the complete conversion is about 500-600 μs which is relatively fast compared to the time required by some of the ADC available.

Result:

A/D conversion of signal in ADC-A in register B.

A/D conversion of signal in ADC-B in register C.

Subroutine SPWT (07E6H)

Wait for next sample time.

This routine tests bit 7 of port C of the upper (first) 8255 and when the signal on this bit goes high (=1), the routine returns to where it is called from. Recall that bit 7 of port C is connected to sampling rate clock of the system. This routine also stores the 'status' of port C in register E, which is used by subroutine CVRT.

LIST OF REFERENCES

[1] Intel 8080 Microcomputer System User's Manual, Sept. 1975

[2] Intel MCS-80 System Design Kit User's Guide, 1976

PROGRAM LISTING

SUBROUTINE CTS

```
;THIS ROUTINE CONVERTS THE EIGHT
;BIT TWO'S COMPLEMENT NUMBER IN THE
;ACCUMULATOR TO A TEN BIT OFFSET
;BINARY NUMBER IN THE D-E REGISTER
;PAIR.  AFTER CONVERSION TO OFFSET
;BINARY, THE MOST SIGNIFICANT FOUR
;BITS ARE RIGHT JUSTIFIED AND PLACED IN
;THE LOWER FIVE BITS OF REGISTER D
;AFTER BEING LEFT JUSTIFIED, THE LEAST
;SIGNIFICANT FOUR BITS ARE PLACED IN THE
;LOWER FIVE BITS OF REGISTER E
;
;
CTS:    ADI     80H
        RRC
        RRC
        RRC
        RRC
;CONVERT THE TWO'S COMPLEMENT NUMBER
;TO OFFSET BINARY AND ROTATE RIGHT
;FOUR PLACES
;
        MOV     E,A
;PLACE RESULT IN E REGISTER
```

```
;
        ANI     OFH
;GET RID OF THE UPPER FOUR BITS
;
        MOV     D,A
;PLACE RESULT IN REGISTER D
;
        MOV     A,E
;GET TWO'S COMPLEMENT OPERAND
;FROM REGISTER E
;
        ANI     OFOH
;MASK OUT LOWER FOUR BITS
;
        RRC
        RRC
        RRC
;ROTATE RIGHT THREE PLACES
;
        MOV     E,A
;PLACE RESULT IN REGISTER E
;
        RET
    ******************************
```

SUBROUTINE YAD

```
;THIS ROUTINE CALLS CTS TO
;CONVERT THE EIGHT BIT TWO'S
;COMPLEMENT OPERAND IN THE
;ACCUMULATOR TO A TEN BIT
;OFFSET BINARY OPERAND IN THE D-E
;PAIR,THEN TAKES THIS OPERAND AND
;ADDS THE CODES FOR HIY,20H,TO REGISTER D
;AND LOY,60H,TO REGISTER E
;THE RESULTS ARE OUTPUT TO THE GRAPHICS
;DISPLAY
;
CO      EQU     01E3H
;THIS IS STARTING ADDRESS FOR THE
;ROUTINE TO OUTPUT A CHARACTER TO THE
;GRAPHICS DISPLAY OR TTY
HIY     EQU     20H
LOY     EQU     60H
;
YAD:    CALL    CTS
        MOV     A,D
        ADI     HIY
        MOV     C,A
        CALL    CO
;OUTPUT FIRST BYTE OF Y ADDRESS
;TO TERMINAL
        MOV     A,E
        ADI     LOY
        MOV     C,A
        CALL    CO
;OUTPUT SECOND BYTE OF ADDRESS
;
        RET
    ******************************
```

<u>SUBROUTINE</u> <u>XAD</u>

```
;THIS ROUTINE CALLS CTS TO CONVERT
;THE EIGHT BIT TWO'S COMPLEMENT
;OPERAND IN THE ACCUMULATOR TO
;A OFFSET BINARY TEN BIT OPERAND
;IN THE D-E PAIR. THEN TAKES THIS RESULT
;AND ADDS THE CODE FOR HIX,20H,TO
;REGISTER D;AND ADDS THE CODE FOR LOX
;,40H,TO REGISTER E
;THE RESULTS ARE THEN OUTPUT TO THE
;TERMINAL
;
HIX     EQU     20H
LOX     EQU     40H
;
XAD:    CALL    CTS
        MOV     A,D
        ADI     HIX
        MOV     C,A
        CALL    CO
;OUTPUT HIX
;
        MOV     A,E
        ADI     LOX
        MOV     C,A
        CALL    CO
;OUTPUT LOX
;
        RET
    ********************************
```

<u>SUBROUTINE</u> <u>GRID</u>

```
;THIS ROUTINE GENERATES A GRID FOR
;READING GRAPH
;
XSIZE   EQU     2319H
;
GRCMD   EQU     1DH
;THIS CHARACTER PUTS GRAPHICS DISPLAY
;IN GRAPHING MODE
TTCMD   EQU     1FH
;THIS CHARACTER PUTS GRAPHICS DISPLAY
;IN TTY MODE
;
GRID:   MVI     C,TTCMD
        CALL    CO
```

```
;PUT GRAPHICS DISPLAY IN TTY MODE
;
        MVI     C,GRCMD
        CALL    CO
;PUT GRAPHICS DISPLAY IN GRAPH MODE
;THIS NOW ALLOWS THE FIRST VECTOR
;NOT TO BE DRAWN, THAT IS, THE FIRST
;VECTOR WILL BE TO THE ORIGIN FOR PURPOSES OF
;INITIALZATION
;
;GENERATE HORIZONTAL PART FIRST
        LXI     H,XSIZE
;LOAD THE H-L PAIR WITH THE ADDRESS
;OF THE STEP SIZE FOR THE HORIZONTAL
;GRID
;
        MVI     B,80H
XLP:    MVI     A,80H
        CALL    YAD
        MOV     A,B
        CALL    XAD
        CALL    XTST
;TEST FOR OVERFLOW FROM
;POSITIVE TO NEGATIVE
;
        MOV     A,M
        ADD     B
        MOV     B,A
        MVI     A,80H
        CALL    YAD
        MOV     A,B
        CALL    XAD
;MOVE TO NEXT LOCATION ON HORIZONTAL
;AXIS FOR VECTOR
;
        MVI     A,7FH
        CALL    YAD
        MOV     A,B
        CALL    XAD
;DRAW THE VECTOR
;
        MVI     A,7FH
        CALL    YAD
        CALL    XTST
;TEST FOR OVERFLOW FROM POSITIVE
;TO NEGATIVE
;
        MOV     A,M
        ADD     B
        MOV     B,A
        CALL    XAD
;DRAW VECTOR TO NEXT HORIZONTAL LOCATION
;
        JMP     XLP
```

```
XTST:   MOV     A,B

        ANA     A

        RM

        ADD     M

        XRA     B

        RP

;THIS LOOP COMPLETES HORIZONTAL GRID

;

;BEGIN VERTICAL GRID HERE

;

YSIZE   EQU     231AH

;VERTICAL GRID SIZE

;

YGRID:  MVI     C,TTCMD

        CALL    CO

        MVI     C,GRCMD

        CALL    CO

;

;PREPARE FOR INITIALIZATION

;AT ORIGIN

;

        LXI     H,YSIZE

;LOAD H-L PAIR WITH ADDRESS FOR

;VERTICAL GRID SIZE

;

        MVI     B,80H

YLP:    MOV     A,B
```

```
        CALL    YAD
        MVI     A,80H
        CALL    XAD
;GET ORIGIN FOR NEXT VECTOR
;
        MOV     A,B
        CALL    YAD
        MVI     A,7FH
        CALL    XAD
        MOV     A,B
        CALL    YAD
        MVI     A,80H
        CALL    XAD
;
;DRAW THE VECTOR
;
        CALL    YTST
        MOV     A,M
        ADD     B
        MOV     B,A
        JMP     YLP
YTST:   MOV     A,B
        ANA     A
        RM
        ADD     M
        XRA     B
        RP
```

```
;THIS FINISHES VERTICAL GRID
;
        MVI     A,7FH
        CALL    YAD
        MVI     A,80H
        CALL    XAD
        MVI     A,7FH
        CALL    YAD
        MVI     A,7FH
        CALL    XAD
        RST     1
    ******************************
```

SUBROUTINE DSPLY

```
;THIS ROUTINE GRAPHS THE CONTENTS
;OF REGISTER B AT 256 POINTS. AFTER
;THIS IT SIMPLY RETURNS TO THE POINT
;FROM WHICH IT WAS CALLED
;
;INITIALIZATION IS ASSUMED AS FOLLOWS:
;       MVI     C,1DH
;       CALL    CO
;THIS PUTS GRAPHICS TERMINAL IN
;GRAPHICS MODE
;       MVI     A,80H
;       STA     DCNT
;THIS SETS THE ITERATION
```

```
;COUNT TO 80H
;       MVI     A,XPND
;       STA     DXPND
;THIS REPEATS SAMPLE XPND TIMES
;THIS IS NORMALLY 1(ONE)
;
DCNT    EQU     231BH
DXPND   EQU     231CH
        ANA     A
        JM      DADD
        LXI     H,DXPND
        ADD     M
        JP      DCONT
        RET
DADD:   LXI     H,DXPND
        ADD     M
DCONT:  MOV     C,A
        STA     DCNT
        MOV     A,B
        MOV     B,C
        CALL    YAD
        MOV     A,B
        CALL    XAD
        RET
        END
```

****

<u>SUBROUTINE</u> <u>FADD</u>

```
FADD:   MOV A,B
        ANA A
        JZ RSLTD
        MOV A,D
        ANA A
        RZ
        MOV A,C
        SUB E
        MOV H,A
        JZ AD;TEST FOR SHIFT
        JP SFTS;SHIFT REG D IF
;R-S IS GREATER THAN ZERO
SFTB:   MVI A,00H
        SUB H
        MOV H,A
        MOV C,E
        CPI 08H
        JP RSLTD
        SFTR B,H
        JMP AD
SFTS:   CPI 08H
        RP
        SFTR D,H
AD:     MOV A,B
        XRA D
        JM ADZ;TEST FOR SIGNS OF
```

```
;B AND D THE SAME
;IF DIFFERENT BRANCH TO ADZ
;THIS BRANCH IS FOR SAME SIGN
        MOV A,B
        ANA A
        JM LZRO;TEST FOR RESULT LESS THAN ZERO
        ADD D
        JP POSS
NRM:    RAR
        JNC NNCR
        INR A
NNCR:   INR C
DON:    MOV B,A
        RET
LZRO:   ADD D
        JM NEGG;TEST FOR RESULT LESS THAN ZERO
        JMP NRM
ADZ:    MOV A,B
        ADD D
        JZ ZER
        JM NEGG
LL:     DCR C
        ADD A
        JP LL
        RAR
        INR C
        MOV B,A
        RET
ZER:    MVI B,00H
        MVI C,00H
        RET
RSLTD:  MOV B,D
        MOV C,E
        RET
POSS:   DCR C
        ADD A
        JP POSS
        RAR
        INR C
        MOV B,A
        RET
NEGG:   DCR C
        ADD A
        JM NEGG
        RAR
        INR C
        MOV B,A
        RET
```

****

SUBROUTINE FMULT

```
FMULT:  MOV A,C;GENERATE EXP
        ADD E;FROM EXP'S IN CE REG'S
        MOV L,A;PUT RESULT IN L REG
        MOV A,B;TEST FOR POSITIVE
        XRA D;OR NEGATIVE RESULT WITH XRA
        JM NG
        MOV A,B;POSITIVE RESULT BRANCH
        ANA A
        JP BPOS;TEST FOR BOTH POSITIVE
        CMA;NEGATE BOTH IF FOUND B
        INR A;TO BE NEGATIVE
        MOV B,A
        MOV A,D
        CMA
        INR A
        MOV D,A
BPOS:   MOV C,B
        CALL MLT;FORM UNSGNED 16 BIT PRODUCT
LO:     MOV A,B;GET RID OF UNNECESSARY ZEROS
        ANA A
        JM L101
        MOV A,C
        ADD A
        MOV C,A
        MOV A,B
        RAL
        MOV B,A
        DCR L
        JMP L0
L101:   RAR;HAVING FOUND FIRST ONE
        MOV B,A;RIGHT SHIFT
        JNC NAD
        INR B;ROUND TO 8 BITS
NAD:    INR L
        MOV C,L;RESULT IN B,C REGS
        RET
NG:     MOV A,B;NEG BRANCH
        ANA A;SET STATUS
        JP DNEG;FIND WHICH MANTISSA
        CMA;IS NEGATIVE
        INR A
        MOV B,A
        JMP L202
DNEG:   MOV A,D
        CMA
        INR A
        MOV D,A
L202:   MOV C,B
        CALL MLT
L3:     MOV A,B
        ANA A
        JM L4
        MOV A,C
        ADD A
        MOV C,A
        MOV A,B
        RAL
        MOV B,A
        DCR L
        JMP L3
L4:     RAR
        MOV B,A
        JNC NNAD
        INR B
NNAD:   INR L
        MOV C,L
        MOV A,B
        CMA
        INR A
        MOV B,A
        RET
ZRO:    MVI B,00H
        MVI C,00H
        POP    H
        RET
      ********************************
```

SUBROUTINE MLT (MULTIPLICATION)

```
MLT:    MOV A,C
        ANA A;SET STATUS
        JZ ZRO
        MOV C,A
        MOV A,D
        ANA A;SET STATUS
        JZ ZRO
        MVI B,00H
        MVI E,09H
MULT0:  MOV A,C
        RAR
        MOV C,A
        DCR E
        JZ DONE
        MOV A,B
        JNC MULT1
        ADD D
MULT1:  RAR
        MOV B,A
        JMP MULT0
DONE:   MOV A,C
        ADD A
        MOV C,A
        MOV A,B
        RAL
        MOV B,A
        RET
    ********************************
```

SUBROUTINE VCMLT

```
VCMLT:  LDA     DIM
        STA     CNT
        LDM     TEMP,0000H
L10:    LRPI    B,C,AD1
        INX     H
        SHLD    AD1
        LRPI    D,E,AD2
        INX     H
        SHLD    AD2
        CALL    FMULT
        LDRP    D,E,TEMP
        CALL    FADD
        STRP    TEMP,B,C
        DCRM    CNT
        JNZ     L10
        RZ
    ******************************
```

SUBROUTINE TST

```
TST:    MOV     A,B
       ANA     A;SET FLAGS
        JZ      STOP
        JM      STOP
;TEST LESS THAN OR EQUAL ZERO
        MOV     A,C
        ANA     A
;TEST EXP GREATER THAN ZERO
        JM      STOP
        RNZ
        MOV     A,B
        CPI     50H
        RP
        JMP     STOP
    ******************************
```

SUBROUTINE INVRT

```
;TEST MANTISSA GREATER THAN 0.5
;REGISTER ASSIGNMENTS
;B=NUMBER TO BE INVERTED
;C=RESULT
;D=REMAINDER
;E=CNTR
INVRT:  MOV     A,C
        STA     TEMP
;SAVE EXP IN TEMP
        MVI     C,00H
        MVI     D,080H
        MVI     E,06H
L50:    MOV     A,D
        CMP     B
        JM      SFT
        SUB     B
        INR     C
SFT:    ADD     A
        MOV     D,A
        MOV     A,C
        ADD     C
        MOV     C,A
        DCR     E
        JNZ     L50
        MOV     A,D
        ADD     A
        JP      NOINR
        INR     C
NOINR:  MOV     B,C
;PUT RESULT IN B
        LDA     TEMP
;GET EXP
        MOV     C,A
        MVI     A,00H
        SUB     C
        INR     A
        MOV     C,A
        RET
```

****

<u>SUBROUTINE</u> <u>TRUNC</u>

```
TRUNC:  MOV     A,C
        ANA     A
        RP
        MVI     A,00H
        SUB     C
        CPI     07H
        JM      NZR
        MVI     B,00H
        MVI     C,00H
        RET
NZR:    MVI     D,0FFH
        MOV     E,A
        MOV     A,D
TRLP:   ADD     A
        MOV     D,A
        DCR     E
        JNZ     TRLP
        MOV     A,B
        ANA     D
        MOV     B,A
        RET
    ********************************
```

<u>SUBROUTINE</u> <u>DSFT</u>

```
DSFT:   LHLD    AD1
        MVI     D,00
        LDA     MDIM
        ADD     A
        MOV     C,A
        LDA     LDIM
        ADD     A
        ADI     0FEH
        MOV     E,A
        DAD     D
        MOV     E,L
        INX     H
        MOV     D,L
        MOV     A,C
        ADI     2
        PUSH    D
DLP:    MOV     L,E
        DCR     E
        MOV     B,M
        MOV     L,D
        DCR     D
        MOV     M,B
        DCR     A
        JNZ     DLP
        MOV     A,C
        CPI     00
        RZ
        POP     D
        MVI     C,00
        JMP     DLP
```

****

SUBROUTINE CNVRT

```
;THIS ROUTINE CONVERTS A FIXED POINT NUMBER
;IN REGISTER B TO A FLOATING POINT NUMBER IN
;THE B-C REGISTER PAIR
;
CNVRT:  MVI     C,00H
        MOV     A,B
        ANA     A
        JZ      ZZ
        JM      LTZ
LP:     DCR     C
        ADD     A
        JP      LP
FIN:    RAR
        INR     C
        MOV     B,A
        RET
LTZ:    DCR     C
        ADD     A
        JM      LTZ
        JMP     FIN
        RET
ZZ:     MVI     B,00H
        MVI     C,00H
        RET
```

****

SUBROUTINE CNRT

```
;THIS ROUTINE CONVERTS A FLOATING POINT NUMBER IN
;THE B-C REGISTER PAIR TO A FIXED POINT NUMBER IN
;REGISTER B
;
CNRT:   MOV     A,C
        CPI     OF8H
        JP      NTZ
        MVI     B,00
        RET
NTZ:    ANA     A
        RZ
        CPI     1
        JP      SAT
        MOV     A,B
LPP:    ANI     OFEH
        JP      SFTRP
        CMC
SFTRP:  RAR
        INR     C
        JNZ     LPP
        MOV     B,A
        RET
SAT:    MOV     A,B
        ANA     A
        MVI     B,7FH
        RP
        INR     B
        RET
    ******************************

    SUBROUTINE CVRT

;THIS ROUTINE PERFORMS A BINARY CHOP
;SUCCESSIVE APPROXIMATION OF TWO ANALOG
;INPUTS USING A 8255 PARALLEL PORT CHIP
;TO PASS DATA. ITS MODE IS DEFINED AS
;PORTA OUTPUT
;PORTB OUTPUT
;PORTC-UPPER HALF INPUT
;      LOWER HALF OUTPUT
;
PORTA   EQU     14H
PORTB   EQU     15H
PORTC   EQU     16H
CNTRL   EQU     17H
CMD     EQU     88H
CVRT:   MVI     A,CMD
        OUT     CNTRL
        MVI     D,40H
        MVI     A,7
        STA     CNT
        MVI     A,0
        OUT     PORTA
        OUT     PORTB
        OUT     PORTC
        MOV     B,A
        MOV     C,A
        CALL    WAIT
        CALL    WAIT
;WAIT FOR COMPARATOR TO RESPOND
;
;
        MOV     A,E
        ANI     40H
;TEST BIT 6 TO DETERMINE IF INPUT TO ADC-A
; IS GREATER OR LESS THAN ZERO
;
        JNZ     TSTB
        MVI     B,80H
TSTB:   MOV     A,E
        ANI     10H
;TEST BIT 4 TO DETERMINE IF INPUT TO ADC-B
; IS GREATER OR LESS THAN ZERO
        JNZ     NXT
        MVI     C,80H
NXT:    MOV     A,B
        ADD     D
        OUT     PORTA
        MOV     A,C
        ADD     D
        OUT     PORTB
```

```
;GET NTH INCREMENT FOR BINARY CHOP AND
;ADD IT TO N-1TH APPROXIMATION FOR BOTH
;ADC CHANNELS
        CALL    WAIT
        CALL    WAIT
;WAIT FOR RESPONSE
        MOV     A,E
        ANI     40H
;TEST BIT 6 OF STATUS
;WORD FOR ADC'S TO DETERMINE
;IF PRESENT INCREMENT IS TO BE
;ADDED TO N-1TH APPROXIMATION
;OF ADC-A
        JZ      NADA
        MOV     A,B
        ADD     D
        MOV     B,A
NADA:   MOV     A,E
        ANI     10H
;TEST BIT 4 OF STATUS WORD
;TO DETERMINE IF PRESENT
;INCREMENT IS TO BE ADDED TO
;N-1TH APPROXIMATION OF ADC-B
        JZ      NADB
        MOV     A,C
        ADD     D
        MOV     C,A
```

```
NADB:   MOV     A,D
        ANA     A
        RAR
        MOV     D,A
;FORM N+1TH INCREMENT BY SHIFTING
;CONTENTS OF REGISTER D
        LXI     H,CNT
        DCR     M
;GET ITERATION COUNT AND DECREMENT
;IF NOT ZERO PERFORM NEXT ITERATION
        JNZ     NXT
        RET
;THIS ROUTINE PAUSES A TIME
;DETERMINED BY HARDWARE BY
;TOGGLING BIT 3 OF PORTC
;AND WAITING FOR BIT 5 TO
;GO HIGH
WAIT:   MVI     A,8
        OUT     PORTC
        MVI     A,0
        OUT     PORTC
PAUSE:  IN      PORTC
        MOV     E,A
        ANI     20H
        JZ      PAUSE
        RET
    ******************************
```

SUBROUTINE SPWT

```
;THIS ROUTINE WAITS FOR NEXT
;SAMPLE TIME AS DETERMINED BY
;EXTERNAL CLOCK FLAGGED BY BIT
;7 OF PORTC OF FIRST 8255 AND
;SAVES STATUS IN REGISTER E
;
SPWT:   IN      PORTC
        MOV     E,A
        ANI     80H
        JZ      SPWT
        RET
        END
```

****