Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

Real Time Control of A Dual Resonator System

by
Penghuai Shen

An experimental multi-resonator system has been developed to study modelling and robust control of such systems in a real-time environment. The plant to be controlled consists of a pair of interconnected resonators whose resonant frequency and plant structure are known. The control objectives are to provide closed loop stability and asymptotic regulation of the frequency, gain and phase of each of the resonators. Furthermore, it is desired to use a minimal controller configuration so as to reduce implementational complexity.

The theoretical results follow directly the paper " Decentralized Robust Control of Interconnected Resonators" written by my advisor Dr. Timothy N. Chang. The main theoretical results have been applied in this experiment and verified by the experimental results.

The controller identification approach has been employed, and feedforward and robust feedback control are implemented in this work. Various experimental conditions such as: single loop amplitude control, 2-loop amplitude control, phase control, and 2-loop amplitude & phase control have been tested. The data obtained are in good agreement with the theoretical predication and simulation results. The stated objectives have all been met.

Data analysis was carried out by porting the experimental data into PC MATLAB environment whereas simulations were done by means of the ALSIM software.

REAL TIME CONTROL
OF
A DUAL RESONATOR SYSTEM

by
Penghuai Shen

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Electrical Engineering

Department of Electrical and Computer Engineering

October 1993

APPROVAL PAGE

Real Time Control
Of
A Dual Resonator System

Penghuai Shen

Dr. Timothy N. Chang, Thesis Advisor (date)
Assistant Professor of Electrical and Computer Engineering, NJIT

Dr. Andrew Meyer, Committee Member (date)
Professor of Electrical and Computer Engineering, NJIT

Dr. Edwin Hou, Committee Member (date)
Assistant Professor of Electrical and Computer Engineering, NJIT

BIOGRAPHICAL SKETCH

Author: Penghuai Shen

Degree: Master of Science in Electrical Engineering

Date: October 1993

Undergraduate and Graduate Education:

- Master of Science in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 1993
- Bachelor of Science in Electrical Engineering,
  Shanghai Institute of Engineering & Technology, Shanghai, China, 1983

Major: Electrical Engineering

This thesis is dedicated to
my wife and my daughter

ACKNOWLEDGMENT

I wish to express my deep gratitude to Dr. Timothy N. Chang, my advisor, for his invaluable help and support during the courses of this work. I have profited greatly from his guidance and insight. I am also very thankful to him for spending his precious time to refine this report. Furthermore, I am indebted to him for providing me with the financial support needed to carry out my study and research.

Many thanks are also due to Dr. Andrew U. Meyer and Dr. Edwin Hou for serving as my thesis committee and also for their many constructive comments and suggestions concerning this work.

I am indebted to many persons for their help and concern during the courses of this work.

Special thanks to the Center for Manufacturing System for supplying the DSP board and testing equipments. Furthermore, assistant from Mr. Alan Bondhus, CIM Operations Manager, is graciously acknowledged.

Finally, I am deeply indebted to my wife for her fully support during my studies and research at NJIT.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

Studies of the control of flexible materials are receiving much attention recently. Research and development of the control of such completely integrated structures with embedded actuators, sensors, signal processing, and control systems are of prime interest.

The behavior of flexible structures, such as a circular disk resonator, is essentially governed by their mode shapes, i.e. their amplitude and phase at one of the flexible modes. The control objective for flexible structure is aimed at stabilizing and/or regulating the device's mechanical behavior.

The plant to be controlled in this work is a dual resonators made with the "smart" material, PZT, a piezoceramic with artificial piezoelectricity

Piezoelectricity is a property of certain classes of crystalline materials. When mechanical pressure is applied to one of these materials, the crystalline structure produces a voltage proportional to the pressure. Conversely, when an electric field is applied to one of these materials, the crystalline structure changes shape, producing dimensional changes in the material.

This property enables the piezoceramic to be used as both actuator and sensor with a homogeneous structure, and therefore the term "smart material". In general, when we control a resonator, we need to use sensor to measure its variables such as amplitude and phase angle over a range of frequency. Similarly, we need to use actuator to execute the control command. But by making use of PZT material, the problem becomes quite simple, and the system setup becomes more compact: all the sensors and actuators are embedded in a device.

In our experiment, the resonator device is a small piece of PZT disk with 1 inch diameter and 0.05 inch thickness. All the actuators and sensors are located on

the same disk. When we apply a voltage signal to the device, a radial deformation of the disk will be produced, meanwhile, due to such mechanical deformation, an electrical charge is also generated. A pickoff signal, indicative of the mechanical motion is interfaced to an A/D converter while the drive signal is generated by a D/A converter. In a wide range of industrial problems, the objective is to control the device's behavior at resonance - when the transfer gain achieves a maximum.

For this particular experiment, each resonator is plated with a central circular electrode surrounded by six equal drive sectors on one side while the other side of the PZT disk is to be used as a ground. The central circular part serves as the sensor, while the six equal sectors, connected by wire, function as the actuator. With two such resonators, the open loop system consists of two inputs and two outputs. The mechanical interface medium is a square flat plate made with aluminum, on which the two resonators are epoxyed. Figure 1.1 shows the configuration of the resonator system.

Since a PZT device is essentially a charge generator, the electrical charge generated by the resonators is first converted by a charge amplifier into a voltage signal. Analog filtering then is used to remove environmental noise sources. Finally a voltage amplifier is used to scale the pickoff signal to match the input range of the A/D converter.

The A/D and D/A converter used here is the Sonitech SAIB (Stereo Audio Interface Box) which has a dual channel, 16-bit A/D, D/A capability with a maximum conversion rate of 48 KHz per channel.

The SAIB resides externally to the host machine and connects serially to the SPIRIT-30 Digital Signal Processor (DSP) board. The incoming signal can be looped back to the output while the host or the DSP is processing or streaming the data (See APPENDIX B for details).

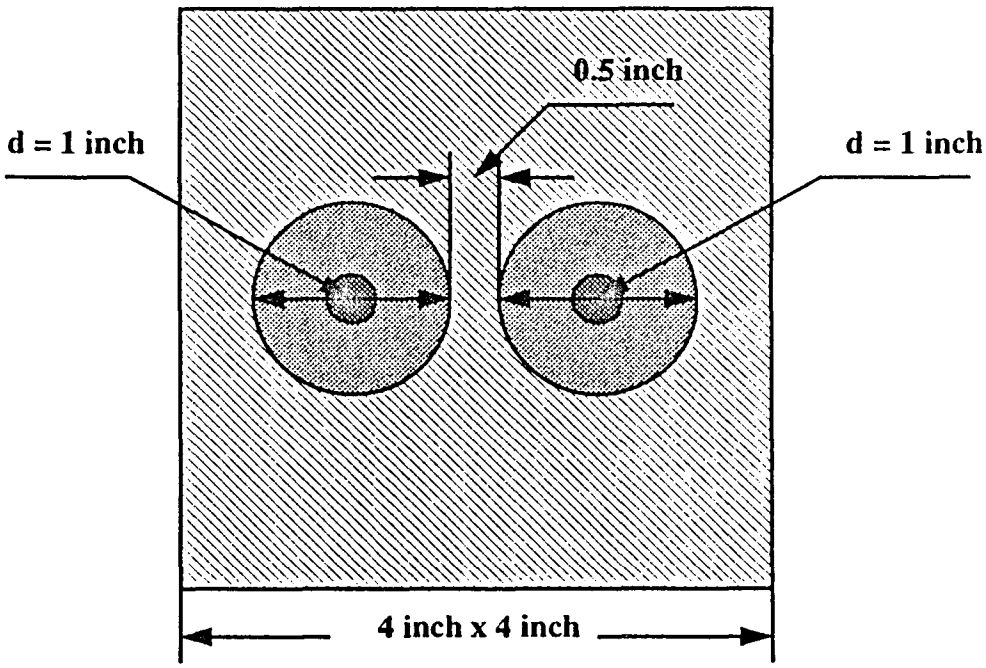


Figure 1.1 Configuration of A Two Resonators Device

The controller is implemented in the Sonitech SPIRIT-30 System. The heart of the SPIRIT-30 is the Texas Instrument's TMS320C30 Digital Signal Processor (DSP) with a 60 ns instruction cycle time, 2Kx32 words of internal RAM, single cycle floating- point multiply/accumulate and an on chip DMA controller. The host system is an IBM compatible 386-PC.

The functional block diagram of the system is shown in Figure 1.2

The organization of this thesis is as follows: In Chapter 2, the theoretical results concerning the control of multiple, interconnected resonators under plant parameter uncertainty are presented along with an synthesis algorithm. Modelling of the two resonators is also provided in this chapter. Chapter 3, the experimental results for the four conditions are presented, they are: 1) single loop amplitude control, 2) 2-loop amplitude control, 3) phase control, and 4) 2-loop amplitude and phase control. The on-line tuning method is employed in this experiment to derive the optimal gain(s) for

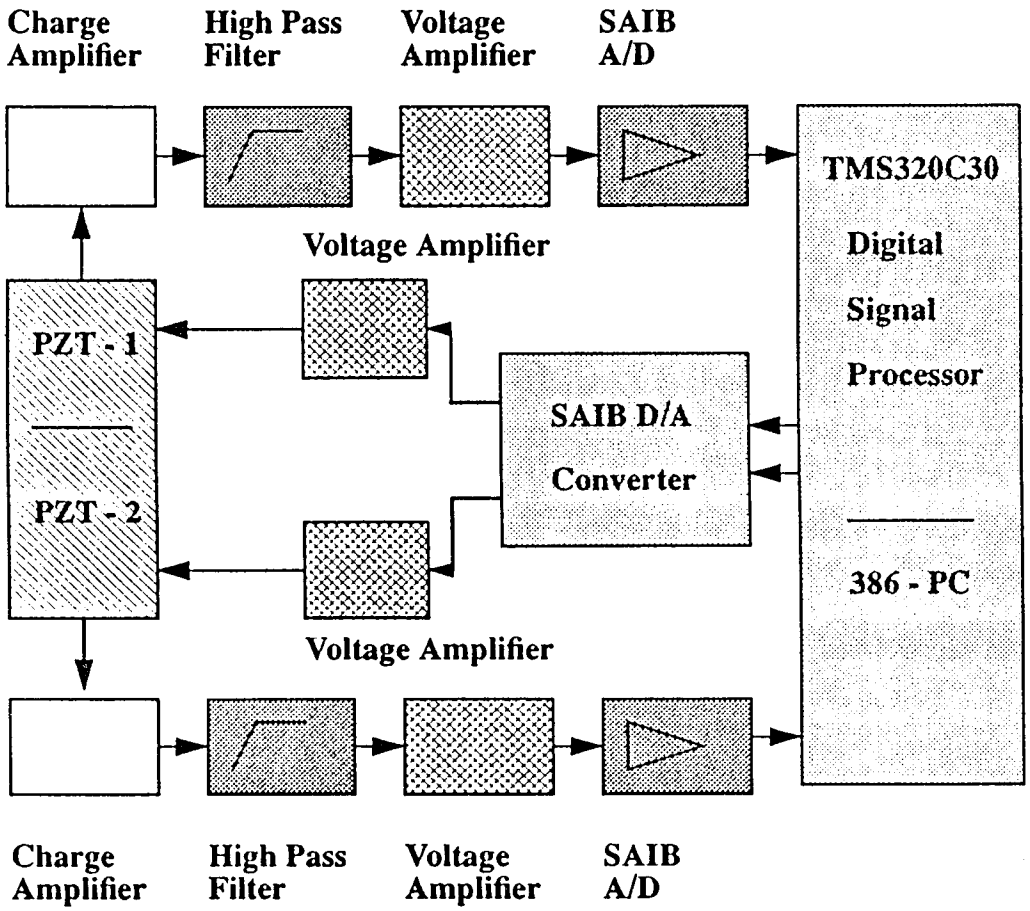


Figure 1.2 Functional Block Diagram of the System

each case, it is not necessary to know the mathematical model of the plant. Chapter 4 is the simulation results of the closed loop system for several control configurations using the same gain(s) as those used in the experiment. Finally, in Chapter 5, the conclusions and future work are discussed.

CHAPTER 2

DEVELOPMENT

2.1 Theoretical Results

Many physical generators and sensors operate in their resonant mode to maintain high efficiency and/or accuracy. For examples, devices such as electric generators, vibrating beam accelerometers, laser devices, and acoustic sensors all base their operation on sustained oscillations where the frequency, amplitude and phase of the resonators are critical. The traditional way of regulating the resonators typically assumes the knowledge of a plant model and a centralized control structure. Such assumptions are either unfeasible or not cost effective on a commercial mass production basis where individual devices may exhibit significant deviation in their parametric values (such as damping and coupling). Furthermore, the resultant controller may also be too complex/expensive to implement.

Therefore, besides providing robust closed loop stability and asymptotic regulation, the resonator control system should also be easy to apply and inexpensive to implement. The decentralized robust controller, when combined with the controller identification approach, would be one such candidate. The imposition a decentralized control structure minimizes the controller hardware whereas the controller identification approach removes the requirement of a plant model. The controller gains are derived from performing a limited number of steady-state experiments on the plant. In the event that a good plant model is available, the parameter optimization method can be used to further shape the transient of the closed loop response.

A definition of the Hilbert transform is given in the next section, to be followed by a description of the plant structure.

Hilbert Transform

The Hilbert transform of a continuous time signal $x(t)$ is given by:

$$\hat{x}(t) = \frac{1}{\pi}\int_{-\infty}^{\infty}\frac{x(\tau)}{t-\tau}d\tau \tag{2.1}$$

and the inverse transform is given by:

$$x(t) = -\frac{1}{\pi}\int_{-\infty}^{\infty}\frac{\hat{x}(\tau)}{t-\tau}d\tau \tag{2.2}$$

where the integrals are understood to be Cauchy's principal values.

Denote $\hat{x}(t) = \mathcal{H}[x(t)]$ and $x(t) = \mathcal{H}^{-1}[\hat{x}(t)]$.

The frequency domain relationship between $x(t)$ and $\hat{x}(t)$ is given by:

$$\hat{X}(\omega) = -jsgn(\omega)X(\omega) \tag{2.3}$$

where $\hat{X}(\omega)$ and $X(\omega)$ are, respectively, the Fourier transform of $\hat{x}(t)$ and $x(t)$.

2.1.1 Plant Structure

The plant consists of r interconnected resonators and is assumed to be represented by an open-loop stable, proper transfer matrix $P(s)$. Furthermore, $P(s)$ is assumed to be narrow-band with center frequency ω_c and bandwidth $2\omega_B$.

The plant output $y(t) \in \Re^r$ is therefore also narrow-band given by:

$$y(t) = [y_1(t), \ldots, y_r(t)]' \tag{2.4}$$

$$y_i(t) = A_i(t)\sin(\omega_c t + \theta_i(t)) \tag{2.5}$$

$$= y_i^c(t)\cos\omega_c t - y_i^s(t)\sin\omega_c t \quad i = 1, 2, \ldots, r \tag{2.6}$$

The reference signal $y^{ref}(t) \in \Re^r$ is defined as :

$$y^{ref}(t) = [y_1^{ref}(t), \ldots, y_r^{ref}(t)]' \tag{2.7}$$

$$y_i^{ref}(t) = A_i^{ref} \sin(\omega_c t + \theta_i^{ref}) \quad i = 1, 2, \ldots, r \tag{2.8}$$

where A_i^{ref} and θ_i^{ref}, $i = 1, 2, \ldots, r$ are assumed constant.

A block diagram showing the plant structure is given in Figure 2.1.

Let the output of the amplitude/phase detectors N_D be defined as:

$$z(t) = [A_1(t), \theta_1(t), \ldots, A_r(t), \theta_r(t)]' \tag{2.9}$$

Denote $z^{ref} = [A_1^{ref}, \theta_1^{ref}, \ldots, A_r^{ref}, \theta_r^{ref}]'$ as the aggregated reference vector and $e(t)$ as the error vector. Then,

$$e(t) = z(t) - z^{ref} \tag{2.10}$$

The control objective is $y(t) \rightarrow y^{ref}(t)$, or equivalently,

$$\lim_{t \rightarrow \infty} e(t) = 0 \tag{2.11}$$

The feedback control system $C : e(t) \rightarrow \delta v(t)$ should have the following properties:

1. Maintains closed loop stability
2. Provides asymptotic tracking, i.e. (2.11).
3. Possesses robustness against plant parameter variations, i.e. Property 2 above holds under small plant perturbations provided that the perturbed closed loop system remains stable.

The overall control signal $v(t) \in \Re^{\nu}$, $\nu = 2r$ is lowpass and consists of a feedforward component v^0 and the feedback component $\delta v(t)$:

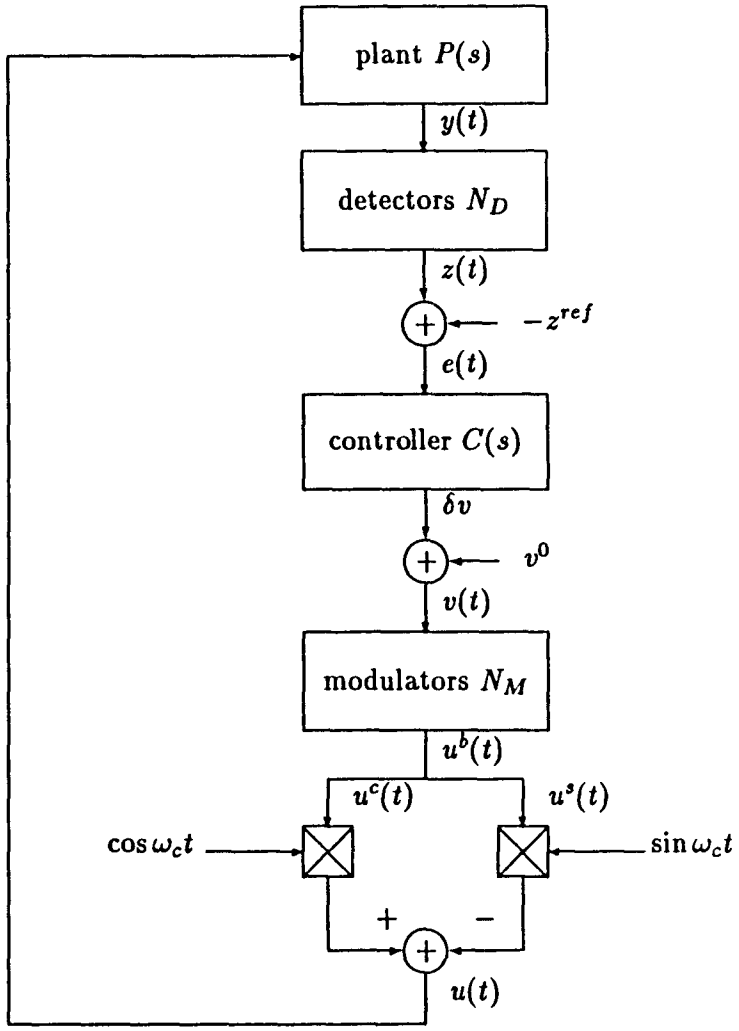


Figure 2.1 Block Diagram of the r-resonator System

$$v(t) = v^0 + \delta v(t) \tag{2.12}$$

Application of v^0 serves two purposes: 1) it speeds up the plant response and 2) it brings the closed loop response to a neighborhood of the desired set points.

The amplitude/phase modulator $N_M : v(t) \rightarrow u^b(t)$ generates the lowpass plant inputs $u^b(t) = [u_1^b(t), \ldots, u_\nu^b(t)]'$, $\nu = 2r$ according to:

$$\begin{bmatrix} u_{2i-1}^b(t) \\ u_{2i}^b(t) \end{bmatrix} = \begin{bmatrix} v_{2i-1}(t) \sin v_{2i}(t) \\ v_{2i}(t) \cos v_{2i}(t) \end{bmatrix} \quad i = 1, 2, \ldots, r \tag{2.13}$$

The plant input $u(t)$ is synthesized as:

$$u_i(t) = u_{2i-1}^b(t) \cos \omega_c t - u_{2i}^b \sin \omega_c t \tag{2.14}$$

and $u(t) = [u_1(t), \ldots, u_r(t)]'$.

2.1.2 Low Frequency Equivalent Model

In this section, the low frequency equivalence of $u(t)$, $y(t)$, and the plant impulse response are derived via the Hilbert transform.

Let $\hat{u}(t) = \mathcal{H}[u(t)]$ and $\bar{u}(t) = u(t) + j\hat{u}(t) \in \mathcal{C}^r$ be the pre-envelop of $u(t)$. Then it is observed that,

$$\begin{aligned} u(t) &= \mathrm{Re}[\bar{u}(t)] \\ &= \mathrm{Re}[(\bar{u}(t)e^{-j\omega_c t})e^{j\omega_c t}] \\ &= \mathrm{Re}[\tilde{u}(t)e^{j\omega_c t}] \end{aligned} \tag{2.15}$$

The quantity $\tilde{u}(t) = \bar{u}(t)e^{-j\omega_c t} \in \mathcal{C}^r$ is known as the complex envelop of $u(t)$. $\tilde{u}(t)$ is a lowpass signal with bandwidth ω_B and it can be expressed as :

$$\tilde{u}(t) = u^c(t) + ju^s(t) \tag{2.16}$$

where $u^c(t)$, $u^s(t) \in \Re^r$ are the low frequency quadrature components of $u(t)$. From (2.15), $u(t)$ may be expressed as:

$$u(t) = u^c(t)\cos\omega_c t - u^s(t)\sin\omega_c t \tag{2.17}$$

The plant impulse response matrix $H(t) = \mathcal{L}^{-1}P(s) = \{h_{ij}(t)\}, i,j = 1,2,\ldots,r$ can also be decomposed into its low frequency quadrature pair $H^c(t) = \{h^c_{ij}(t)\}$ and $H^s(t) = \{h^s_{ij}(t)\}$ in a similar manner:

$$H(t) = 2H^c(t)\cos\omega_c t - 2H^s(t)\sin\omega_c t \tag{2.18}$$

The factor of 2 is introduced for notational convenience. It is obvious that $H^c(t)$ and $H^s(t)$ are stable iff $H(t)$ is stable.

The plant output $y(t)$ can be expressed as:

$$\begin{aligned} y(t) &= H(t) * u(t) \\ &= \mathrm{Re}[\bar{H}(t) * \bar{u}(t)] \end{aligned}$$

where $\bar{H}(t) = H(t) + j\hat{H}(t)$, $\hat{H}(t) = \mathcal{H}[H(t)]$ and " * " denotes convolution. Following the same development for $u(t)$,

$$\begin{aligned} y(t) &= \mathrm{Re}[(\tilde{H}(t)e^{j\omega_c t}) * (\tilde{u}(t)e^{j\omega_c t})] \\ &= \mathrm{Re}[e^{j\omega_c t}(\tilde{H}(t) * \tilde{u}(t))] \end{aligned}$$

On letting

$$\begin{aligned} \dot{y}(t) &= \dot{H}(t) * \dot{u}(t) \\ &= (H^c(t) + jH^s(t)) * (u^c(t) + ju^s(t)) \end{aligned}$$

and observing that $y(t) = \mathrm{Re}[\tilde{y}(t)e^{j\omega_c t}]$, it follows that

$$y(t) = y^c(t)\cos\omega_c t - y^s(t)\sin\omega_c t \tag{2.19}$$

where

$$\begin{aligned} y^c(t) &= H^c(t) * u^c(t) - H^s(t) * u^s(t) \\ y^s(t) &= H^s(t) * u^c(t) + H^c(t) * u^s(t) \end{aligned}$$

and $y^c(t) = [y_1^c(t), \ldots, y_r^c(t)]'$, $y^s(t) = [y_1^s(t), \ldots, y_r^s(t)]' \in \Re^{r\times r}$

The baseband equivalent model of the resonators is now obtained as:

$$y^b(s) = G(s)u^b(s) \tag{2.20}$$

where

$$\begin{bmatrix} u_{2i-1}^b(t) \\ u_{2i}^b(t) \end{bmatrix} = \begin{bmatrix} u_i^c(t) \\ u_i^s(t) \end{bmatrix} \quad i = 1, 2, \ldots, r \tag{2.21}$$

$$\begin{bmatrix} y_{2i-1}^b(t) \\ y_{2i}^b(t) \end{bmatrix} = \begin{bmatrix} y_i^c(t) \\ y_i^s(t) \end{bmatrix} \quad i = 1, 2, \ldots, r \tag{2.22}$$

$$G(s) = \begin{bmatrix} G_{11}(s) & \cdots & G_{1r}(s) \\ \vdots & & \vdots \\ G_{r1}(s) & \cdots & G_{rr}(s) \end{bmatrix} \tag{2.23}$$

$$G_{ij}(s) = \begin{bmatrix} h_{ij}^c(s) & -h_{ij}^s(s) \\ h_{ij}^s(s) & h_{ij}^c(s) \end{bmatrix} \quad i, j = 1, 2, \ldots, r \tag{2.24}$$

Again, $G(s)$ is stable iff $P(s)$ is stable. It should be noted that $G(s)$ may also include the dynamics of the amplitude/phase detectors (typically lowpass filters). Figure 2.2 shows the equivalent model of the r-resonator problem.

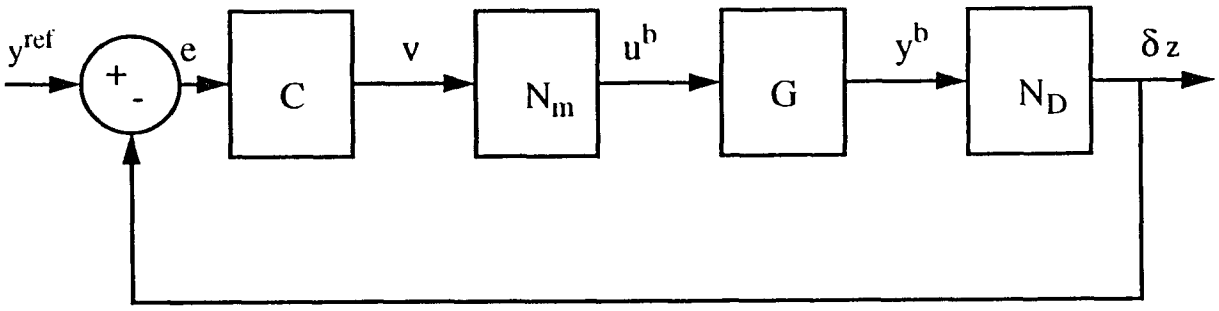


Figure 2.2 Low frequency Equivalent Model of the Plant

2.1.3 Modulators and Detectors

The modulators perform gain/phase modulation and frequency translation. To implement the latter, a pair of multipliers and two quadrature signals $\cos\omega_c t$ and $\sin\omega_c t$ are required for each resonator. These two signals can be derived from a number of sources such as:

1. The master resonator
2. An internal reference oscillator
3. The filtered output signals

The exact source depends upon the nature of the application. For example, in situation where ω_c is expected to vary with environmental factors (temperature, pressure, acceleration, etc.), 1. or 3. should be used. If, on the other hand, ω_c is

to be held constant , then 2. should be used. All resonator control systems should receive the same quadrature signals to avoid phase uncertainties.

The amplitude/phase demodulators also perform two tasks:

1. Convert $y(t)$ into its baseband equivalent $y^b(t)$

2. Transform $y^b(t)$ into the amplitude/phase vector $z(t)$ according to (2.9) and:

$$\begin{aligned} A_i(t) &= \sqrt{(y^b_{2i-1})^2 + (y^b_{2i})^2} \\ \theta_i(t) &= \tan^{-1}\left(\frac{y^b_{2i-1}}{-y^b_{2i}}\right), \quad i = 1, 2, \ldots, r \end{aligned} \tag{2.25}$$

Tasks 1 and 2 may be carried out simultaneously by the demodulator but they are separated for the purpose of analysis. In actual implementation, the amplitude and phase signals $A_i(t)$ and $\theta_i(t)$ can be obtained by a number of standard methods, e.g. an envelop detector for $A_i(t)$ and a phase discriminator for $\theta_i(t)$. In each case, the detectors can be modelled by a cascade combination of a nonlinear element and a linear filter. Since the nonlinear elements are algebraic in nature, it is possible to lump the remaining dynamics into the low frequency equivalent plant model and treat the demodulators as ideal, given by (2.25).

2.1.4 Control Systems Synthesis

The traditional way of synthesizing a control system typically assumes the following conditions:

1. Centralized control structure

2. Known plant model

In actual application, especially in commercial mass production of resonators systems, such requirements are either unfeasible or not cost effective due to the variability of each physical units.

In this case, a preferred alternative is the so-called "controller identification" approach. It entails three steps:

1. Determine the necessary controller type and structure for the particular application.

2. Perform a limited number of steady-state experiments on the plant.

3. Derive the necessary controller gains based on the experimental data.

It is not necessary to know the plant model or even the model order. All that is required here is a limited number of steady-state data to determine the control system. The following two sections describe control structure and gain determination base on this approach.

2.1.5 Controller Type and Structure

The control signal $v(t)$ consists of a constant feedforward term v^0 and a dynamic feedback term $\delta v(t)$. The feedforward vector is computed from the experimental data as follows:

Let $\xi^0 = [\xi_1^0, \ldots, \xi_r^0]' \in \Re^r$ be given by:

$$\xi_i^0 = A_i^{ref} e^{j\theta_i^{ref}}, \quad i = 1, 2, \ldots, r \tag{2.26}$$

and

$$\zeta^0 = P^{-1}(j\omega_C)\xi^0 \tag{2.27}$$

Then,

$$v^0 = \begin{bmatrix} |\zeta_1^0| \\ -\angle\zeta_1^0 \\ \vdots \\ |\zeta_r^0| \\ -\angle\zeta_r^0 \end{bmatrix} \tag{2.28}$$

where $|\cdot|$ and $\angle$ represent the magnitude and phase angle respectively. From (2.27), it is clear that the plant $P(s)$ must not have a transmission zero at $s = j\omega_c$.

Synthesis of $\delta v(t)$ can be carried out by first linearizing the modulators N_M and the detectors N_D:

From (2.13), N_M can be linearized around v^0 as:

$$\delta u^b(t) = \mathcal{B}\delta v(t) \tag{2.29}$$

where

$$\mathcal{B} = \text{block diagonal}[\Lambda_1, \Lambda_2, \ldots, \Lambda_r] \tag{2.30}$$

$$\Lambda_i = \begin{bmatrix} \sin v_{2i}^0 & v_{2i-1}^0 \cos v_{2i}^0 \\ -\cos v_{2i}^0 & v_{2i-1}^0 \sin v_{2i}^0 \end{bmatrix} \quad i = 1, 2, \ldots, r \tag{2.31}$$

Similarly, from (2.25) the detectors N_D is linearized around v^0 as:

$$\delta z(t) = \mathcal{C}\delta y^b(t) \tag{2.32}$$

where

$$\mathcal{C} = \text{block diagonal}[\Gamma_1, \Gamma_2, \ldots, \Gamma_r] \tag{2.33}$$

$$\Gamma_i = \begin{bmatrix} \frac{y_{2i-1}^{b0}}{\sqrt{(y_{2i-1}^{b0})^2+(y_{2i}^{b0})^2}} & \frac{y_{2i}^{b0}}{\sqrt{(y_{2i-1}^{b0})^2+(y_{2i}^{b0})^2}} \\ \frac{-y_{2i}^{b0}}{(y_{2i-1}^{b0})^2+(y_{2i}^{b0})^2} & \frac{y_{2i-1}^{b0}}{(y_{2i-1}^{b0})^2+(y_{2i}^{b0})^2} \end{bmatrix} \quad i = 1, 2, \ldots, r \tag{2.34}$$

The baseband incremental model of the plant is now obtained as:

$$G^c(s) = \mathcal{C}G(s)\mathcal{B} \tag{2.35}$$

A block diagram for the feedback control system is shown in Figure 2.3.

Lemma 1 below follows directly from the frequency translation of the passband spectrum of the plant:

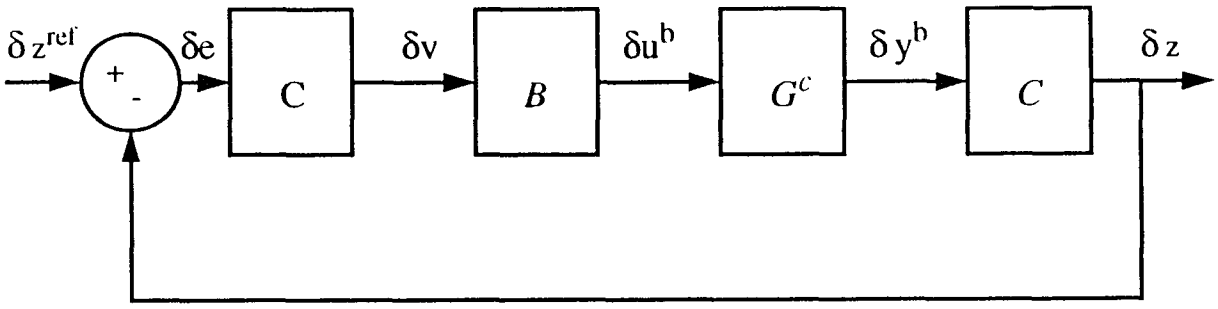


Figure 2.3 Low Frequency Incremental Model of the Plant

Lemma 1 *If the nominal $P(s)$ has no transmission zeros at $\pm j\omega_c$, then $G^c(s)$ has no transmission zero at the origin*

The above Lemma assures the steady-state invertibility of the dc gain of the incremental plant. Denote

$$\mathcal{T} = -G^c(0) \tag{2.36}$$

From Lemma 1 the above inverse exists iff $P(j\omega_c) \neq 0$.

In order to achieve asymptotic stability and tracking, the controller C must supply the closed loop transmission zeros at the origin for the error transfer matrix. The minimal controller type is given by:

$$C(s) = \frac{K}{s} \tag{2.37}$$

where $K \in \Re^{\nu \times \nu}$ is the feedback gain matrix that may take on one of the following control structures:

1. Centralized – K has ν^2 non-zero parameters.

2. Partially decentralized – K has 2ν non-zero parameters, i.e.

$$K = \text{block diagonal}[K_1, K_2, \ldots, K_r], \quad K_i \in \Re^{2\times 2}, \; i = 1, 2, \ldots, r \tag{2.38}$$

3. Decentralized – K has only ν non-zero parameters.

$$K = \text{diagonal}[K_1, K_2, \ldots, K_\nu], \quad K_i \in \Re^{1}, \; i = 1, 2, \ldots, \nu \tag{2.39}$$

The use of decentralized structure can significantly reduce the number of interconnections and thence the computation/hardware cost. For example, the centralized controller of a six-resonator plant ($\nu = 12$) requires 144 interconnections whereas a decentralized controller only needs 12 interconnections.

2.1.6 Controller Gain Determination

The controller gain K can be determined as follows:

1. Centralized Gain:

$$K = \varepsilon \mathcal{T}^{-1} \tag{2.40}$$

where $\varepsilon > 0$ is a tuning gain.

2. Partially Decentralized Gain:

Partition $\mathcal{T}$ into 2×2 submatrices as:

$$\mathcal{T} = \begin{bmatrix} \mathcal{T}_{11} & \cdots & \mathcal{T}_{1r} \\ \vdots & & \vdots \\ \mathcal{T}_{r1} & \cdots & \mathcal{T}_{rr} \end{bmatrix} \tag{2.41}$$

where $\mathcal{T}_{ij} \in \Re^{2\times 2}$, $i,j,=1,2,\ldots,r$. The controller gain is given as:

$$\begin{aligned} K_1 &= \varepsilon \mathcal{T}_{11}^{-1} \\ K_2 &= \varepsilon[\mathcal{T}_{22} - \mathcal{T}_{21}\mathcal{T}_{11}^{-1}\mathcal{T}_{12}]^{-1} \\ &\vdots \quad\quad \vdots \\ K_r &= \varepsilon \left[\mathcal{T}_{rr} - [\mathcal{T}_{r1},\ldots,\mathcal{T}_{r,r-1}] \begin{bmatrix} \mathcal{T}_{11} & \cdots & \mathcal{T}_{1,r-1} \\ \vdots & & \vdots \\ \mathcal{T}_{r-1,1} & \cdots & \mathcal{T}_{r-1,r-1} \end{bmatrix}^{-1} \begin{bmatrix} \mathcal{T}_{1r} \\ \vdots \\ \mathcal{T}_{r-1,r} \end{bmatrix} \right]^{-1} \end{aligned} \tag{2.42}$$

provided the inverses exist. It is also possible to assign an individual tuning gain to each local control agent. In this case, the number of tuning gains equals to r.

3. Fully Decentralized Gain:

Let $|\mathcal{T}(i)|$ denote the i-th leading principal minor of $\mathcal{T}$. Assume that $|\mathcal{T}(i)| \neq 0$, $i = 1,2,\ldots,r$, define $I(i)$ as:

$$\begin{aligned} I(1) &= 1 \\ I(i) &= |\mathcal{T}(i)||\mathcal{T}(i-1)|^{-1}, \quad i = 2,3,\ldots,r \end{aligned} \tag{2.43}$$

then the controller gains $K_i \in \Re^1$ are given as:

$$K_i = \varepsilon[I(i)t_{ii}]^{-1} \quad i = 1,2,\ldots,\nu \tag{2.44}$$

where t_{ii} is the i-th diagonal element of $\mathcal{T}$. Again, each local control agent can be assigned a separate tuning gain, resulting in ν adjustments.

Lemma 2 is now obtained from Theorem 1 of [4] and Lemma 1 above.

Lemma 2 *Assume that $P(s)$ is open loop stable and has no transmission zero at $\pm j\omega_c$, then*

1. *$\mathcal{T}^{-1}$ exists*
2. *There exists a permutation matrix R_1 such that $\mathcal{T} = -[R_1 G^c(0) R_1^{-1}]^{-1}$ possesses non-zero leading principal minors.*

The above results indicate that, by choosing a proper I/O pairing, the inverses in (2.42) and (2.44) all exist so that the controller gains are well-defined.

Lemma 3 (5) *Assume that $P(s)$ is open loop stable and the controller gains are well-defined, $\exists \varepsilon^* > 0$ such that $\forall \varepsilon \in (0, \varepsilon^*]$, the closed system is locally, asymptotically stable.*

It should be noted that Lemma 3 implicitly assumes that $P(s)$ has no transmission zero at $\pm j\omega_c$. From Lemma 1 and the structure of controller (2.37), robust asymptotic tracking occurs for the closed loop system.

2.1.7 Synthesis Algorithm

The following algorithm provides a systematic way of synthesizing the resonator control system:

1. Determine the gain matrix $P(j\omega_C)$ by standard frequency response methods.
2. Verify that $\det P(j\omega_c) \neq 0$.
3. Calculate the feedforward control vector v^0 using (2.28).
4. Determine $G^c(0)$ by either using equations (2.23), (2.31), and (2.34) or by experimentally calculating the DC gain matrix from $\delta v(t)$ to $e(t)$ directly.

5. Determine a suitable I/O pairing (i.e. the permutation matrix R_1) and a control configuration (i.e. centralized, partially decentralized , or fully decentralized).

6. Calculate the controller gains using (2.40), (2.42), or (2.44).

7. Tune the system by adjusting ε on-line so that desirable transient characteristics are obtained. Use a separate tuning gain for each local control agent if necessary. Such tuning gain(s) always exists if the conditions of Lemma 3 are satisfied.

2.2 Modelling & Controller Structure

Modelling of the plant is studied in this section along with the determination of the controller type and structure.

2.2.1 Open Loop Experiment

The open loop experiment consists of three steps: 1) wide band frequency sweep test, 2) linearity check, and 3) local modelling.

To conduct the wide band frequency sweep, a sinusoidal signal of 5-v amplitude is applied to the 2 inputs of the plant. Magnitude response, from 1 kHz to 12.5 kHz, is derived for the following combination: 1) transfer characteristics of the first resonator (labelled as R) (p_{11}), 2) cross coupling characteristics from the second resonator (labelled as B) to resonator R, (p_{12}). 3) cross coupling characteristics from resonator R to resonator B(p_{21}). and 4) transfer characteristics of resonator B(p_{22}).

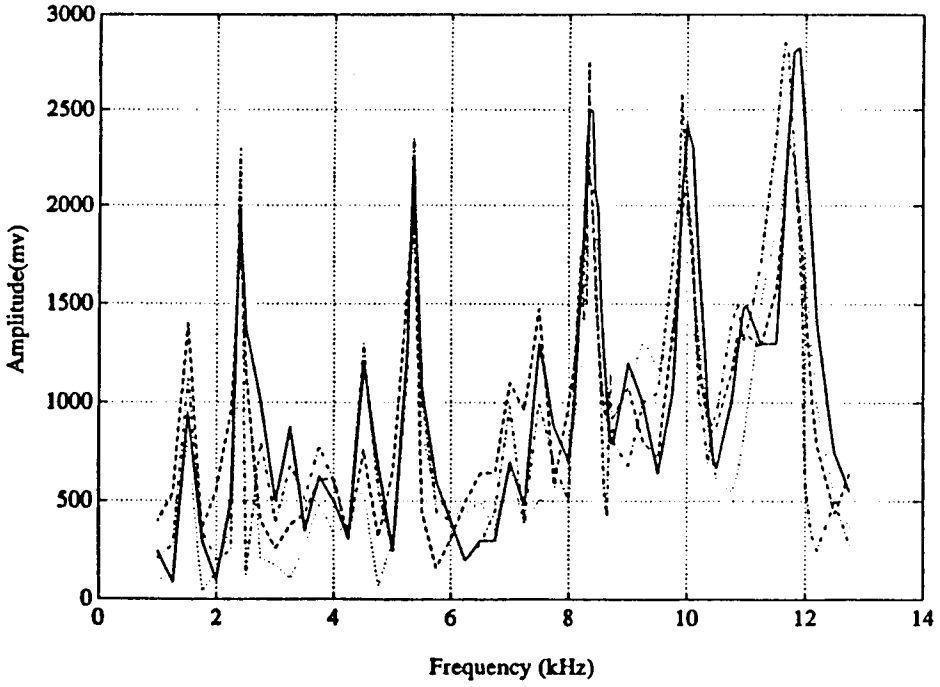


Figure 2.4 Spectrum of Frequency Response of the Resonator Device

It is observed that within the 1 kHz to 12.5 kHz range, there exist 5 dominant resonant peaks at 2.465 kHz, 5.375 kHz, 8.36 kHz, 10.25 kHz and 11.985 kHz.

The transfer characteristics of the devices are quite close, reflecting the fact that the resonator structure is highly symmetrical. The resonance at 8.36 kHz is found to be most consistent (less sensitive to mounting, temperature, etc.) and is therefore selected for the next test procedure: linearity check.

For linearity check, three excitation voltage level at 1-v, 5-v, and 10-v are in turn applied to the plant. The frequency range starts from 7 kHz and ends at 9 kHz with 18 quantization intervals.

The corresponding test results for p_{11}, p_{12}, p_{21}, and p_{22} are plotted in Figures A.1, A.2, A.3 and A.4 (refer to APPENDIX A).

From the experimental data, it is concluded that the nonlinearity in each device is within 10 %, a figure that can be tolerated by robust controllers.

Since the consistency and linearity of the plant around 8.36 kHz are acceptable, local moddelling of the resonator around this frequency can be carried out.

2.2.2 Modelling

The dynamics of the resonators around the resonance at f_c=8.36 kHz can be represented by the following 2x2 transfer matrix:

$$P(s) = [p_{ij}(s)],\ i,j,=1,2 \tag{2.45}$$

$$p_{11}(s) = \frac{A_{11}+B_{11}s}{s^2+\frac{\omega_c}{Q_{11}}s+\omega_c^2} \tag{2.46}$$

$$p_{12}(s) = \frac{A_{12}+B_{12}s}{s^2+\frac{\omega_c}{Q_{12}}s+\omega_c^2} \tag{2.47}$$

$$p_{21}(s) = \frac{A_{21}+B_{21}s}{s^2+\frac{\omega_c}{Q_{21}}s+\omega_c^2} \tag{2.48}$$

$$p_{12}(s) = \frac{A_{22}+B_{22}s}{s^2+\frac{\omega_c}{Q_{22}}s+\omega_c^2} \tag{2.49}$$

The parameters A_{ij}, B_{ij}, and Q_{ij} , i,j = 1,2 , are obtained as follows:

First Q_{ij} : bandwidth of p_{ij} around 8.36 kHz derived by this center frequency. Second at $\omega = \omega_c$:

$$p_{ij}(j\omega_c) = \frac{A_{ij} + B_{ij}(j\omega_c)}{-\omega_c^2 + \frac{\omega_c}{Q_{ij}}(j\omega_c) + \omega_c^2} \tag{2.50}$$

$$K_{ij} = |p_{ij}(j\omega_c)| \tag{2.51}$$

equivalent to:

$$B_{ij} = \sqrt{\frac{(\frac{K_{ij}\omega_c}{Q_{ij}})^2}{1 + \tan^2(\theta_{ij})}} \tag{2.52}$$

$$A_{ij} = \tan\theta_{ij}\omega_c B_{ij} \tag{2.53}$$

where: θ_{ij} is the corresponding phase angle at 8.36 kHz.

For a drive level of 5-v, the corresponding Q_{ij}, θ_{ij}, and K_{ij}'s are shown in Table 2.1, 2.2, and 2.3.

Thus, A_{ij} and B_{ij} are calculated using (2.52) (2.53) as:

$$A_{11} = -8.3568 \times 10^7 \qquad B_{11} = 280.5265 \tag{2.54}$$

$$A_{12} = 8.3549 \times 10^7 \qquad B_{12} = 280.4634 \tag{2.55}$$

$$A_{21} = 6.0856 \times 10^7 \qquad B_{21} = 204.2843 \tag{2.56}$$

$$A_{22} = -7.7877 \times 10^7 \qquad B_{22} = 261.42 \tag{2.57}$$

Table 2.1 Q value of the Resonator

-	Q value
Q_{11}	32.5149
Q_{12}	29.27
Q_{21}	35.72
Q_{22}	39.0783

Table 2.2: Open Loop Gain K_{ij} of the Resonator Device, around Resonant Frequency = 8.36kHz

-	K value
K_{11}	1.0
K_{12}	0.9
K_{21}	0.8
K_{22}	1.12

Table 2.3 Phase Angle , θ_{ij}, at Resonance

-	θ
θ_{11}	-80^o
θ_{12}	80^o
θ_{21}	80^o
θ_{22}	-80^o

Thus, the transfer function for the resonators are:

$$p_{11}(s) = \frac{-8.3568 \times 10^7 + 280.5265s}{s^2 + 1615.5s + 2.759 \times 10^9} \tag{2.58}$$

$$p_{12}(s) = \frac{8.3549 \times 10^7 + 280.4634s}{s^2 + 1794.6s + 2.759 \times 10^9} \tag{2.59}$$

$$p_{21}(s) = \frac{6.0856 \times 10^7 + 204.2843s}{s^2 + 1470.56s + 2.759 \times 10^9} \tag{2.60}$$

$$p_{22}(s) = \frac{-7.7877 \times 10^7 + 261.42s}{s^2 + 1344.2s + 2.759 \times 10^9} \tag{2.61}$$

The transfer matrix at the resonant frequency is then:

$$p(j\omega_c) = \begin{bmatrix} p_{11}(j\omega_c) & p_{12}(j\omega_c) \\ p_{21}(j\omega_c) & p_{22}(j\omega_c) \end{bmatrix} \tag{2.62}$$

or:

$$p(j52527.3848) = \begin{bmatrix} 0.1738 + 0.9848i & 0.1563 - 0.8863i \\ 0.1389 - 0.7878i & 0.1945 + 1.1030i \end{bmatrix} \tag{2.63}$$

The state space model of the plant can be written as:

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{2.64}$$

$$y(t) = Cx(t) \tag{2.65}$$

where:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -\omega_c^2 & -\frac{\omega_c}{Q_{11}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\omega_c^2 & -\frac{\omega_c}{Q_{12}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\omega_c^2 & -\frac{\omega_c}{Q_{21}} & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\omega_c^2 & -\frac{\omega_c}{Q_{22}} \end{bmatrix} \tag{2.66}$$

$$B = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \tag{2.67}$$

$$C = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & c_{21} & c_{22} & c_{23} & c_{24} \end{bmatrix} \tag{2.68}$$

Equivalently,

$$\dot{x}_1(t) = x_2(t) \tag{2.69}$$

$$\dot{x}_2(t) = -\frac{\omega_c}{Q_{11}} x_2(t) - \omega_c^2 x_1(t) + u_1(t) \tag{2.70}$$

$$\dot{x}_3(t) = x_4(t) \tag{2.71}$$

$$\dot{x}_4(t) = -\frac{\omega_c}{Q_{12}}x_4(t) - \omega_c^2 x_3(t) + u_2(t) \tag{2.72}$$

$$\dot{x}_5(t) = x_6(t) \tag{2.73}$$

$$\dot{x}_6(t) = -\frac{\omega_c}{Q_{21}}x_6(t) - \omega_c^2 x_5(t) + u_1(t) \tag{2.74}$$

$$\dot{x}_7(t) = x_8(t) \tag{2.75}$$

$$\dot{x}_8(t) = -\frac{\omega_c}{Q_{22}}x_8(t) - \omega_c^2 x_7(t) + u_2(t) \tag{2.76}$$

$$y_1 = c_{11}x_1(t) + c_{12}x_2(t) + c_{13}x_3(t) + c_{14}x_4(t) \tag{2.77}$$

$$y_2 = c_{21}x_5(t) + c_{22}x_6(t) + c_{23}x_7(t) + c_{24}x_8(t) \tag{2.78}$$

Or

$$\dot{x}_1(t) = x_2(t) \tag{2.79}$$

$$\dot{x}_2(t) = -1615.5x_2(t) - 2.75 \times 10^9 x_1(t) + u_1(t) \tag{2.80}$$

$$\dot{x}_3(t) = x_4(t) \tag{2.81}$$

$$\dot{x}_4(t) = -1794.6x_4(t) - 2.759 \times 10^9 x_3(t) + u_2(t) \tag{2.82}$$

$$\dot{x}_5(t) = x_6(t) \tag{2.83}$$

$$\dot{x}_6(t) = -1470.5x_6(t) - 2.759 \times 10^9 x_5(t) + u_1(t) \tag{2.84}$$

$$\dot{x}_7(t) = x_8(t) \tag{2.85}$$

$$\dot{x}_8(t) = -1344.2x_8(t) - 2.75 \times 10^9 x_7(t) + u_2(t) \tag{2.86}$$

$$y_1 = -8.36 \times 10^7 x_1(t) + 280.5x_2(t) + 8.3 \times 10^7 x_3(t) + 280.4x_4(t) \tag{2.87}$$

$$y_2 = 6.08 \times 10^7 x_5(t) + 204.3x_6(t) - 7.7 \times 10^7 x_7(t) + 261.4x_8(t) \tag{2.88}$$

Frequency responses of the local theoretical model are plotted in Figures A.5, A.6, A.7 and A.8 for p_{11}, p_{12}, p_{21}, and p_{22}, respectively (see APPENDIX A for details).

It is observed that the theoretical curves and the experimental curves are in agreement to within 1 dB.

2.2.3 Controller Structure

A digital controller is used in the control of the resonators. Most of the signal processing such as amplitude & phase modulation and demodulation, reference sine wave generation, lowpass filtering, and control are implemented in software.

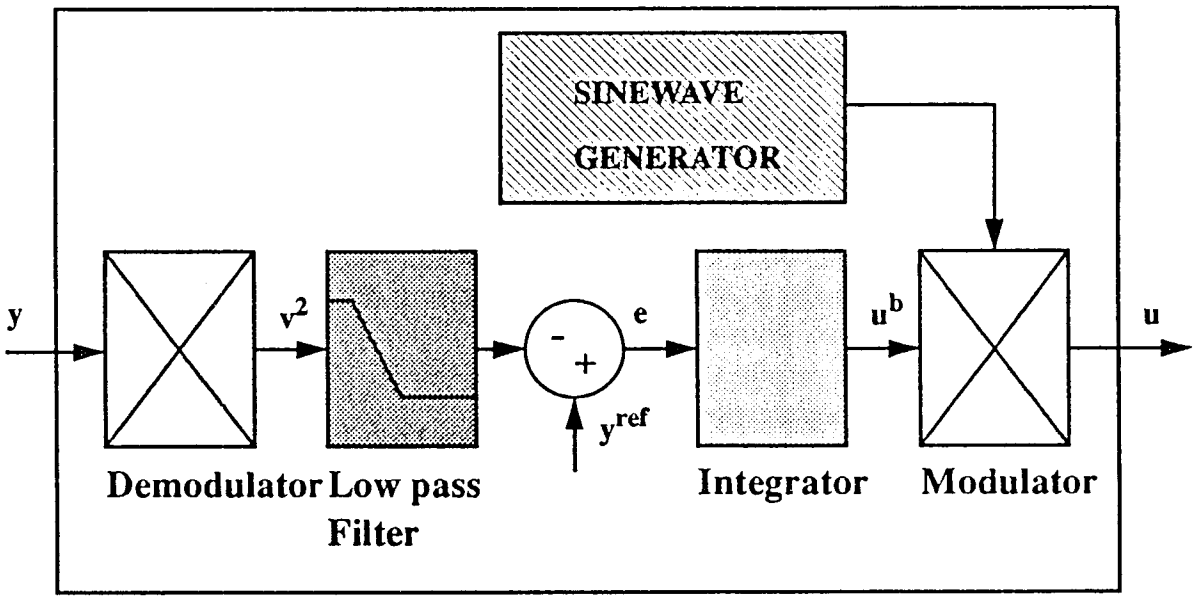


Figure 2.5 Block Diagram of A Digital Controller

The plant output is received by the controller through the A/D convertors. The amplitude & phase angle of the plant output signals are then detected by the amplitude & phase detectors by means of square law demodulation . A lowpass filter is used to filter the high frequency components generated during demodulation to obtain the baseband amplitude information which is later subtracted from the amplitude reference to form the error. This error signal is then fed to a integrator that in turn produces the necessary corrective action to achieve asymptotic tracking & regulation. This control signal is sent via the D/A converter to the plant input. The block diagram of a digital controller is shown in Figure 2.5.

The following sections provide a description of the various software signal processing components:

2.2.3.1 Reference Sine wave Generation The reference sine wave is synthesized with the frequency f_c = 8.36 kHz and equivalent amplitude = 1 volt.

The sine wave generated by software is obtained by discretizing the continuous time transfer function:

$$H(s) = \frac{\omega_c^2}{s^2 + \omega_c^2}$$

Bilinear transformation is applied to H(s) whereby the continuous variables s is replaced by the expression that involves the discrete variable z:

$$S = \frac{2}{T_s}\frac{z-1}{z+1}$$

to translate the H(s) to H(z).

To balance the software overhead and controller delay, a sampling frequency of 27.1 kHz is chosen. This results in the sampling period T_s being :

$$\begin{aligned} T_s &= \frac{1}{f_s} \\ &= 36.4(\mu s) \end{aligned}$$

A second-order direct-form II Infinite Impulse Response lattice structure is used for the implementation. The difference equations for this structure are :

$$d(n) = x(n) + \alpha_{11}d(n-1) + \alpha_{12}d(n-2) \quad (2.89)$$

$$s(n) = \beta_{10}d(n) + \beta_{11}d(n-1) + \beta_{12}d(n-2) \quad (2.90)$$

where, the coefficients obtained using MATLAB routine are tabulated below:

$$\alpha_{11} = -0.0867$$

$$\alpha_{12} = 1.$$

$$\beta_{10} = 0.4783$$

$$\beta_{11} = 0.9566$$

$$\beta_{12} = 0.4783$$

and $s(n)$ is the sine wave output.

2.2.3.2 Amplitude & Phase Detector The input signal is assumed to have the form :

$$v = A\sin(\omega t + \theta)$$

The square law demodulation is first applied to the signal v, i.e.:

$$v^2 = A^2\sin^2(\omega t + \theta)$$

or

$$v^2 = \frac{A^2}{2}(1 - \cos 2(\omega t + \theta))$$

which includes the baseband amplitude component

$$\frac{A^2}{2}$$

and the high frequency component

$$\frac{A^2}{2}\cos 2(\omega t + \theta)$$

After filtering the signal using a lowpass filter, the baseband component is obtained. The same procedure is applied to phase detection:

Suppose the two input signals are $A\sin\omega t$ and $B\sin(\omega t + \theta)$, multiplication of these two signals yields:

$$v = A\sin\omega t B\sin(\omega t + \theta) \quad (2.91)$$

$$= \frac{AB}{2}(\cos\theta - \cos(2\omega t + \theta)) \quad (2.92)$$

which involves the baseband phase component

$$\frac{AB}{2}\cos\theta$$

and the high frequency components

$$\frac{AB}{2}(\cos(2\omega t+\theta))$$

Again, the baseband phase information can be extracted by means of lowpass filtering

2.2.3.3 Lowpass Filter Three second-order digital Butterworth filters of identical structure are used for the two amplitude and the phase demodulation outputs. The cutoff frequency is set to 1256.6 rad/s under the sampling frequency $\omega_s = 172332$ rad/s . The transfer function of the second order digital Butterworth lowpass filter is given by

$$H(z)=\frac{\beta_{20}+\beta_{21}z^{-1}+\beta_{22}z^{-2}}{1-\alpha_{21}z^{-1}+\alpha_{22}z^{-2}} \tag{2.93}$$

The difference equations for this transfer function:

$$d_{pi}(n) = y_{pi}(n)+\alpha_{21}d_{pi}(n-1)+\alpha_{22}d_{pi}(n-2) i=1,2,3 \tag{2.94}$$

$$y_{fi}(n) = \beta_{20}d_{pi}(n)+\beta_{21}d_{pi}(n-1)+\beta_{22}d_{pI}(n-2) \tag{2.95}$$

where, the coefficients are generated by MATLAB as:

$$\alpha_{21} = -1.9325 \tag{2.96}$$

$$\alpha_{22} = 0.9373 \tag{2.97}$$

$$\beta_{20} = 0.0005 \tag{2.98}$$

$$\beta_{21} = 0.001 \tag{2.99}$$

$$\beta_{22} = 0.0005 \tag{2.100}$$

and:

y_{p1} & y_{f1} are the filter input and output of the first resonator amplitude detector.

y_{p2} & y_{f2} are the filter input and output of the second resonator. amplitude detector.

y_{p3} & y_{f3} are the filter input and output for the phase detector.

d_{pi} is the filter state.

2.2.3.4 Integrator Controller & Feedforward Controller The control signal consists of a constant feedforward term and a dynamic feedback term. The integrator severs as a dynamic feedback controller which has the form:

$$C = \frac{K_I}{s}$$

It performs the task to stabilize and regulate the output to the desired value, where:

K_I is the integral gain.

To realize the integral function in software, the rectangular form

$$\eta(n+1) = \eta(n) + e(n) \tag{2.101}$$

$$u_b(n) = K_I \eta(n) \tag{2.102}$$

is applied.

Where $e(n)$ is the error formed by subtracting the reference from the output of the demodulator, ie:

$$e(n) = y(n) - A_i^{ref} \tag{2.103}$$

and the $\eta(n)$ is the integrator state, $u_b(n)$ is the output of the integrator.

The feedforward controller in this structure is just a constant value computed by the procedures given by previous section (2.26),(2.27) and (2.28). Since the transfer matrix at the resonant frequency s= $j\omega_c$ is given by:

$$P(j\omega_c) = \begin{bmatrix} p_{11}(j\omega_c) & p_{12}(j\omega_c) \\ p_{21}(j\omega_c) & p_{22}(j\omega_c) \end{bmatrix} \tag{2.104}$$

or:

$$P(j52527.3848) = \begin{bmatrix} 0.1738 + 0.9848i & 0.1563 - 0.8863i \\ 0.1389 - 0.7878i & 0.1945 + 1.1030i \end{bmatrix} \tag{2.105}$$

Then the inverse of the $P(j52527.3848)$, obtained by using MATLAB routine, is:

$$P(j52527.3848)^{-1} = \begin{bmatrix} 1.1558 - 0.9994i & 1.1474 - 0.0.437i \\ 1.0199 - 0.3884i & 1.0319 - 0.8923i \end{bmatrix} \tag{2.106}$$

Assuming that the reference vector is:

$$y^{ref}(t) = \begin{bmatrix} 1 \times \sin(\omega_c t + 0.5) \\ 1 \times \sin(\omega_c t + 1) \end{bmatrix} \tag{2.107}$$

Then, the feedforward control vector v^0 is calculated as:

$$\begin{aligned} \xi^0 &= \begin{bmatrix} 1e^{j0.5} \\ 1e^{j1} \end{bmatrix} \\ \zeta^0 &= P^{-1}(j\omega_c)\xi^0 \\ &= \begin{bmatrix} 2.5e^{j0.16} \\ 2.45e^{j0.22} \end{bmatrix} \end{aligned}$$

Thus

$$v^0 = [2.5\ 0.16\ 2.45\ 0.22]'$$

CHAPTER 3

EXPERIMENTAL RESULTS

3.1 Experimental Results

The experimental setup shown in Figure 3.1 consists of the following components: two circular disk resonators, an aluminum base plate between the resonators, an interface circuit (including charge amplifier, filter and voltage amplifier), two A/D, D/A convertors (sonitech SAIB) and a DSP system. The host computer is an IBM compatible 386 PC.

The dual resonator system has two inputs and two outputs. The plant and controller outputs can be uploaded from the DSP to PC and then ported to the Matlab enviroment for display and analysis. The control command can be modified interactively and then downloaded into the DSP for execution. All the sequences are software controllable.

3.2 Main Results

In this work, the controller identification approach [1] has been used in the control of resonator's amplitude and relative phase. Feedforward and robust feedback control have been implemented[1]. Various experimental conditions such as: single loop amplitude control, 2-loop amplitude control, phase angle control, and 2-loop amplitude & phase control have been tested. Finally, a comparison between results obtained with and without feedforward control is made. A decentralized control structure is chosen for this control configuration to minimize the controller complexity, so that there are no coupling gains between the control agents C1 & C2. Refer to Figure 3.2 for a block diagram of the decentralized control structure.

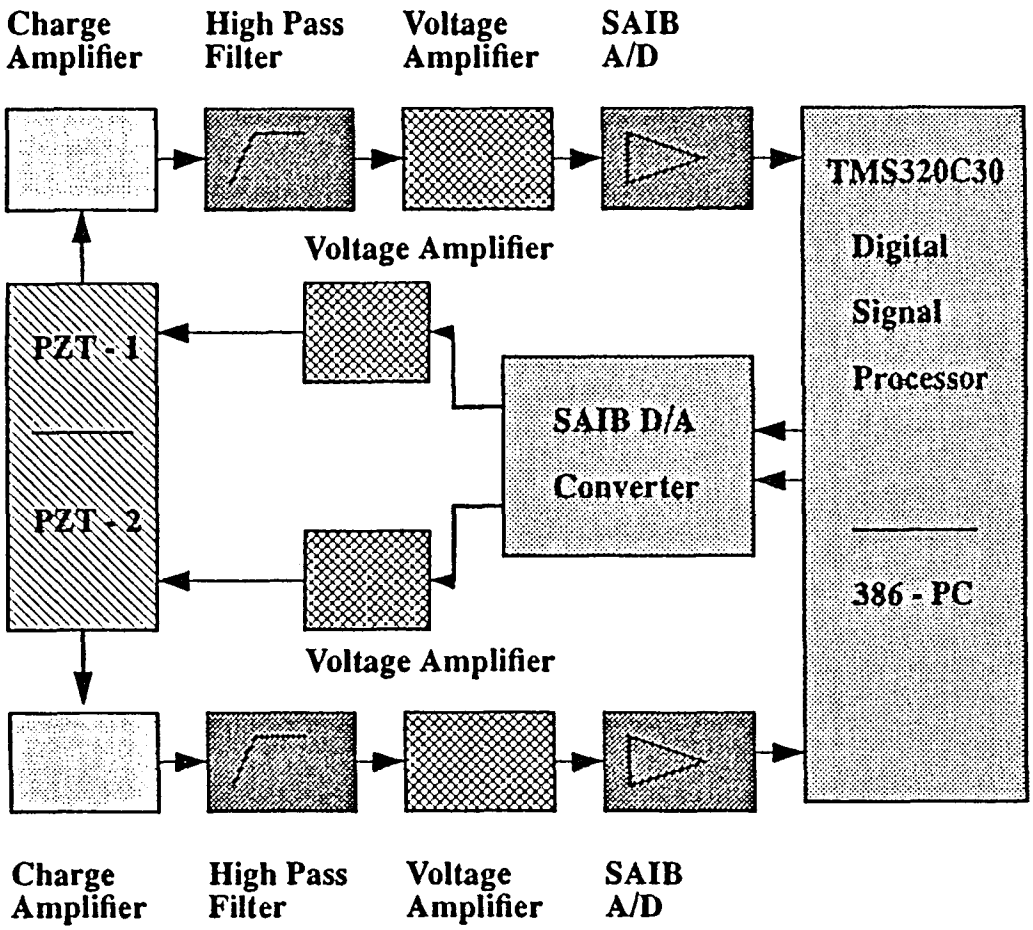


Figure 3.1 Block Diagram of Experimental Setup

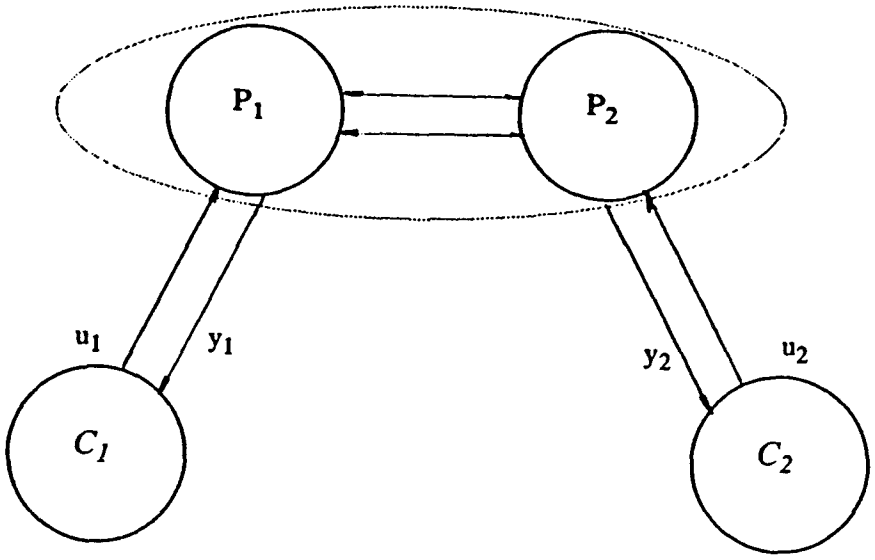


Figure 3.2 Block Diagram of the Decentralized Control Structure

Several experimental conditions have been chosen: 1) single loop amplitude control, 2) 2-loop amplitude control, 3) phase control, and 4) 2-loop amplitude and phase control. The respective experimental results are described in the following sections.

3.2.1 Single Loop Amplitude Control

In this case, only one of the two resonators's amplitude is controlled. The objectives are 1) regulation of the resonator's output amplitude to a desired level and 2) fast speed of response.

The configuration of single loop amplitude control system is shown in Figure 3.3.

The demodulator is applied to get the amplitude of the plant output signal y_1 (or y_2). The amplitude of the output signal is then compared with the system reference variable to form the error. The integrator is employed to eliminate the error. Finally, the modulator converts the baseband controller signal to the passband .

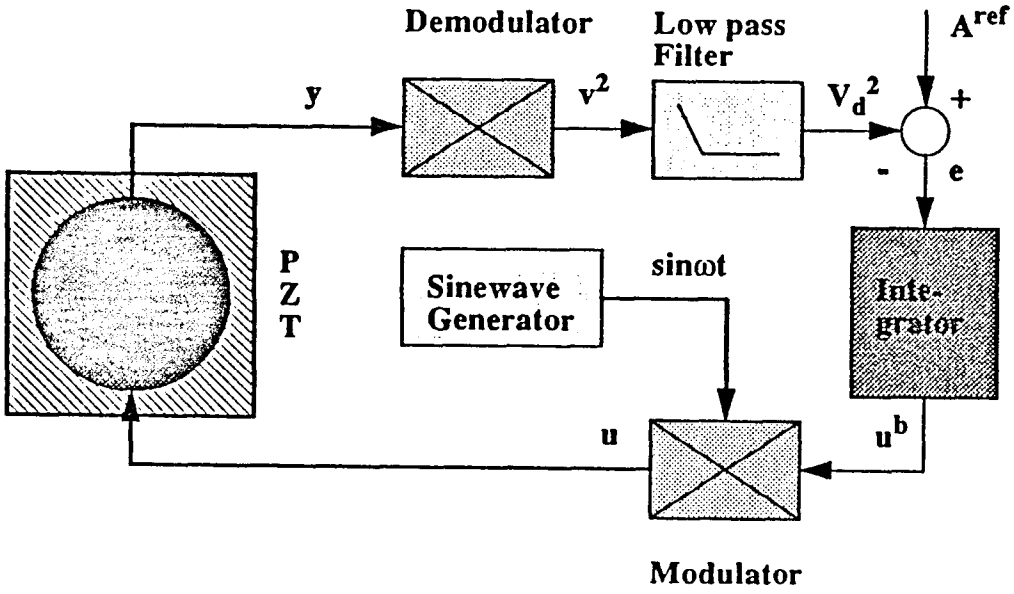


Figure 3.3 Block Diagram of Single Loop Amplitude Control

The amplitude command trajectories are shown in Figures 3.4- 3.5, the first trajectory starts from 0 to $\sqrt{0.5}$, and then from $\sqrt{0.5}$ to 1, from 1 to $\sqrt{1.5}$, and finally, from $\sqrt{1.5}$ to $\sqrt{2}$. The second trajectory is from 0 to $\sqrt{2}$, and to $\sqrt{0.5}$ in a reversed order as in the first trajectory. The trajectories are automatically synthesized by the software.

The on-line tuning method is applied here: the integral gain K_I is progressively increased until an optimal value is obtained. The initial value of K_I was chosen to be 0.0025.

The closed loop responses are shown in Figures C.1, C.2 (refer to APPENDIX C). From figures, it is observed that the gain 0.0025 results in the settling time about 0.7 second with no overshoot.

The gain is further increased to 0.025. Figures C.3 & C.4 (see APPENDIX C) show that the settling time is now about 0.08 second and the overshoot is about 2.5%. Here, we define that the the response time is the time that the output signal is within $\pm$ 2 % of the reference level.

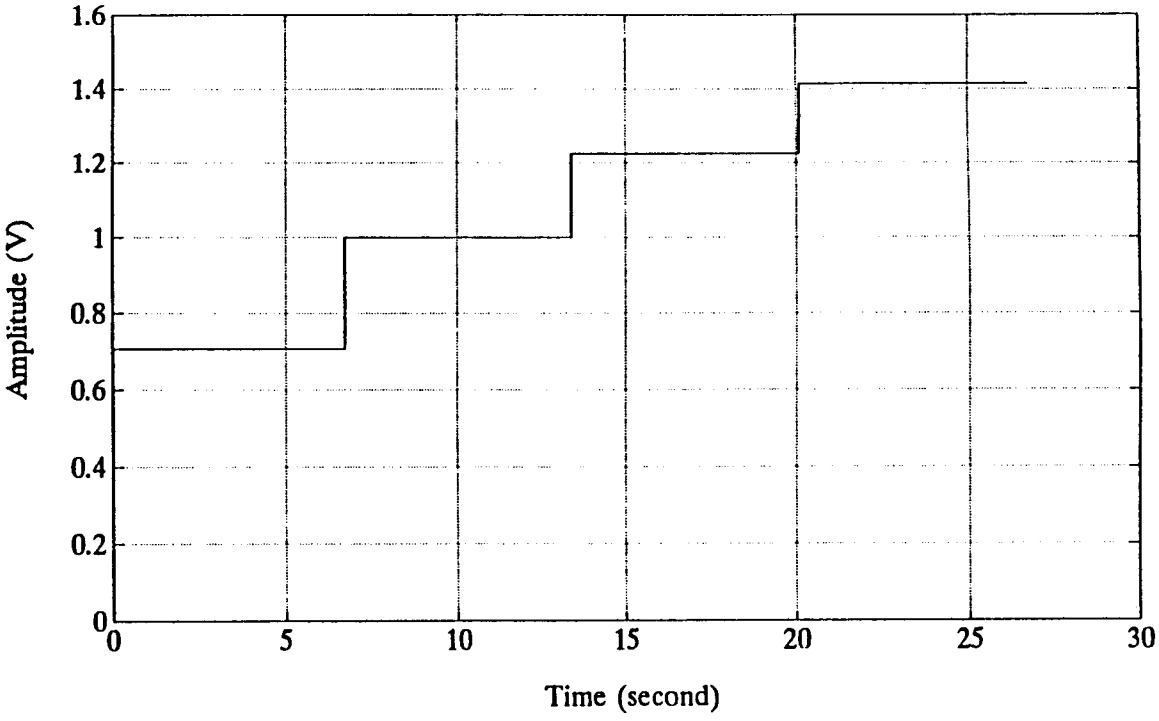


Figure 3.4 First Command Trajectory For Single Loop Amplitude Control

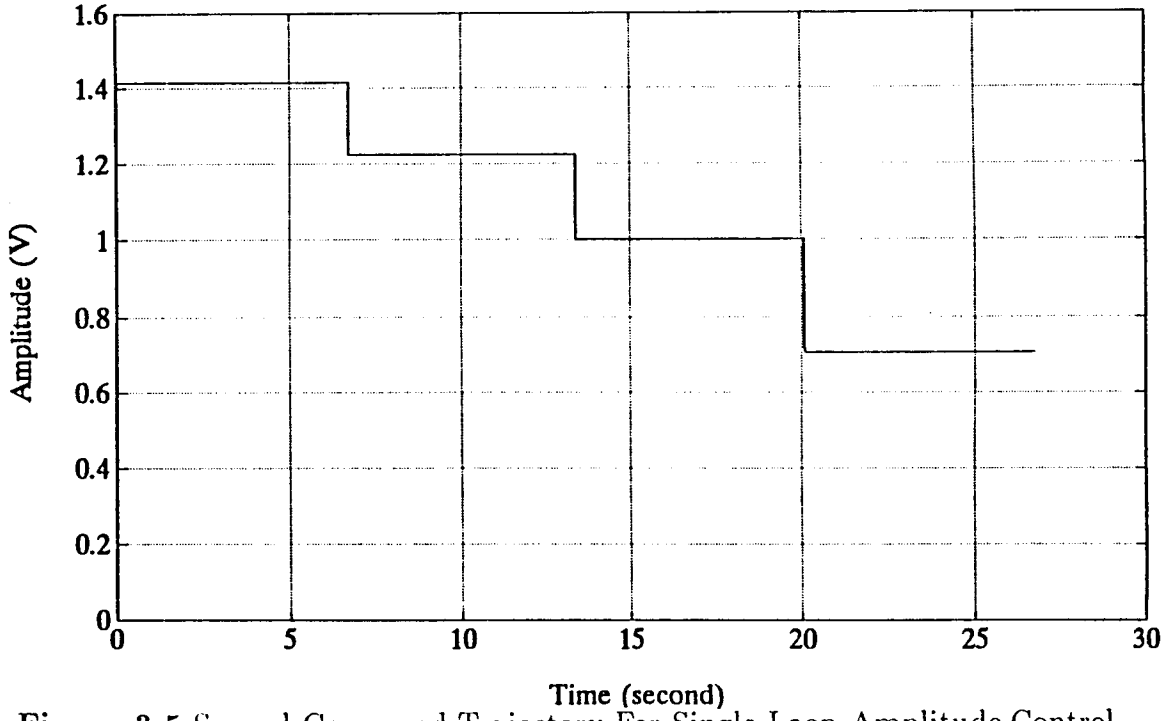


Figure 3.5 Second Command Trajectory For Single Loop Amplitude Control

Table 3.1 Experimemtal Results for Single Loop Amplitude Control

Case	gain K_I	command trajectory	Response time	Overshoot	Fig. No.
1	0.0025	output: $0 \rightarrow \sqrt{0.5}$	0.68 s	no overshoot	Fig. C.1
		output: $\sqrt{0.5} \rightarrow 1$	0.68 s	no overshoot	
		output: $1 \rightarrow \sqrt{1.5}$	0.69 s	no overshoot	
		output: $\sqrt{1.5} \rightarrow \sqrt{2}$	0.68 s	no overshoot	
		output: $0 \rightarrow \sqrt{2}$	0.45 s	no overshoot	Fig. C.2
		output: $\sqrt{2} \rightarrow \sqrt{1.5}$	0.45 s	no overshoot	
		output: $\sqrt{1.5} \rightarrow 1$	0.45 s	no overshoot	
		output: $1 \rightarrow \sqrt{0.5}$	0.45 s	no overshoot	
2	0.025	output: $0 \rightarrow \sqrt{0.5}$	0.08 s	2.5 %	Fig. C.3
		output: $\sqrt{0.5} \rightarrow 1$	0.08 s	2.5 %	
		output: $1 \rightarrow \sqrt{1.5}$	0.082 s	2.5 %	
		output: $\sqrt{1.5} \rightarrow \sqrt{2}$	0.08 s	2.5 %	
		output: $0 \rightarrow \sqrt{2}$	0.0785 s	2.5 %	Fig. C.4
		output: $\sqrt{2} \rightarrow \sqrt{1.5}$	0.0785 s	2.5 %	
		output: $\sqrt{1.5} \rightarrow 1$	0.0785 s	2.5 %	
		output: $1 \rightarrow \sqrt{0.5}$	0.0785 s	2.5 %	

Increasing the gain to 0.05 results in instability. Therefore, it is concluded that K_I=0.025 is the "optimal" value for this particular control configuration.

The experimental results are summarized in Table 3.1

3.2.2 Two Loop Amplitude Control

In the 2-loop amplitude control problem, both resonators are under closed loop control, with the same performance objectives being imposed. It is to be noted that a decentralized control structure has been imposed: there are no coupling gain between the two control agents. Furthermore, since the resonator structure is nearly identical, the control gains for the two control agents can be chosen to be the same to facilitate on-line tuning of the system.

A block diagram of the two loop amplitude control is shown in Figure 3.6. The same controller structure as the single loop amplitude controller has been used here again.

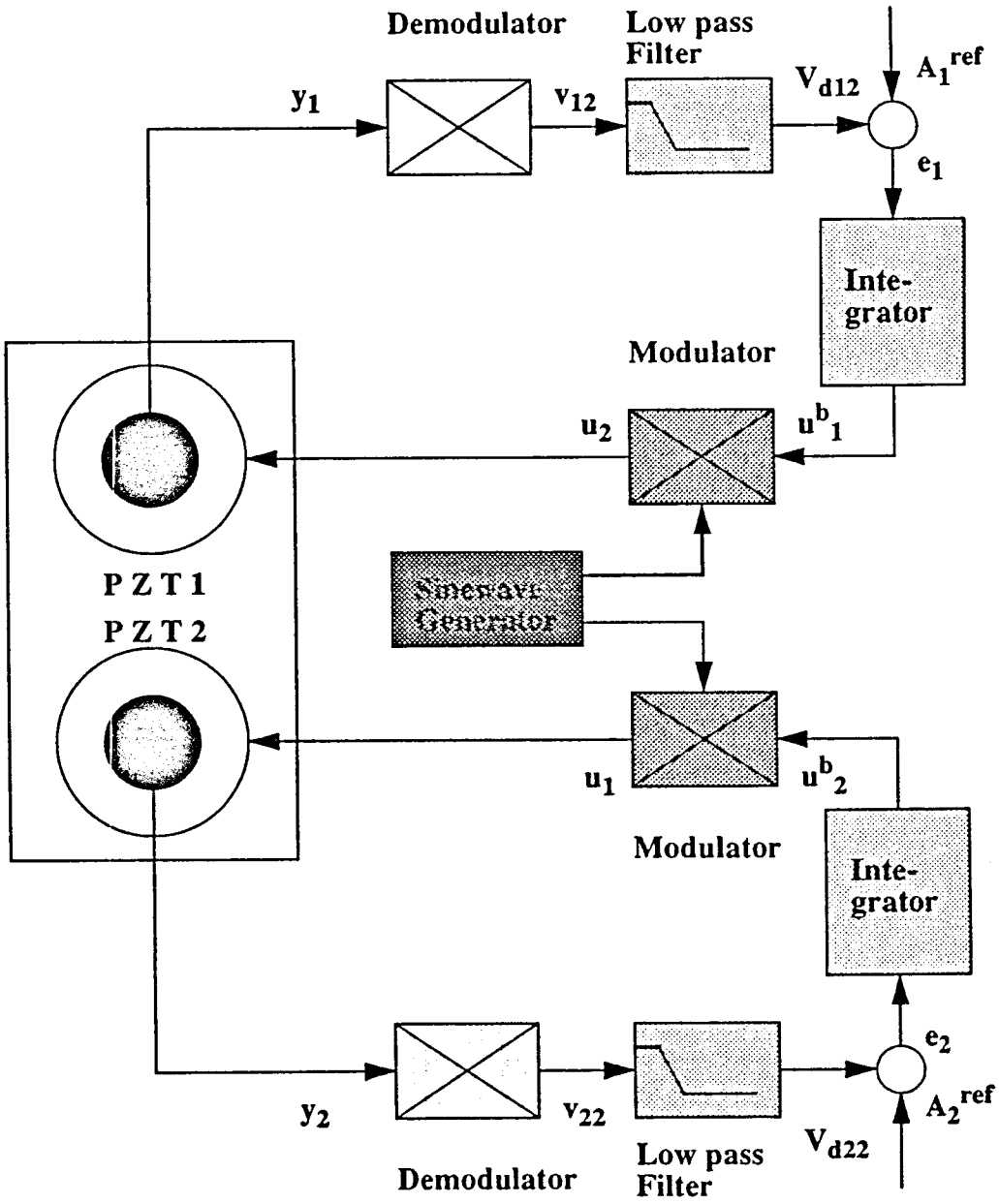


Figure 3.6 Block Diagram of Two Loop Amplitude Control

Table 3.2 Experimental Results for Two Loop Amplitude Control

Case	gain K_I	command trajectory	Response time	Overshoot	Fig. No.
1	0.0025	output1: $0 \rightarrow 1$	< 1 s	no overshoot	Fig. C.5
		output2: $0 \rightarrow 1$	< 1 s	< 5 %	to
		output1: $1 \rightarrow \sqrt{2}$	< 1 s	no overshoot	Fig. C.10
		output2: $1 \rightarrow 1$	< 1 s	15 %	
		output1: $1 \rightarrow 1$	< 1 s	$< 3\%$	
		output2: $1 \rightarrow \sqrt{2}$	< 1 s	no overshoot	
		output1: $1 \rightarrow \sqrt{0.5}$	< 1 s	no undershoot	
		output2: $1 \rightarrow 1$	< 1 s	8% undershoot	
		output1: $1 \rightarrow 1$	< 1 s	1 %	
		output2: $1 \rightarrow \sqrt{0.5}$	< 1 s	no overshoot	
2	0.025	output1: $0 \rightarrow 1$	0.07 s	no overshoot	Fig. C.11
		output2: $0 \rightarrow 1$	0.08 s	< 3 %	to
		output1: $1 \rightarrow \sqrt{2}$	0.05 s	3 %	Fig. C.16
		output2: $1 \rightarrow 1$	0.08 s	18 %	
		output1: $1 \rightarrow 1$	0.07 s	$< 8\%$	
		output2: $1 \rightarrow \sqrt{2}$	0.08 s	no overshoot	
		output1: $1 \rightarrow \sqrt{0.5}$	0.10 s	no undershoot	
		output2: $1 \rightarrow 1$	0.08 s	10% undershoot	
		output1: $1 \rightarrow 1$	0.06 s	2.5 %	
		output2: $1 \rightarrow \sqrt{0.5}$	0.08 s	no overshoot	

The command trajectory for the channel 1 varies from 0 to 1, 1 to $\sqrt{2}$, then 1 to $\sqrt{0.5}$ while that of the channel 2 varies from 0 to 1, from 1 to $\sqrt{0.5}$, and finally from 1 to $\sqrt{2}$;

The experimental results for 2-loop amplitude control are summerized in Table 3.2 and plotted in in Figures C.5 - C.18 (refer to APPENDIX C). Again, the tuning gains K_I=0.0025, 0.025, and 0.05 have been applied.

Figures C.5 & C.10 show the results with gain K_I set to 0.0025.

Although results with K_I=0.0025 lead to asymptotic tracking and fast response, the results with K_I = 0.025 behave nearly ideally for tracking the command trajectories: the settling time is only 0.07s, the overshoot is very small (about 2 %) (see Figures C.11 - C.16 in APPENDIX C).

Further increasing the gain to 0.05 results in a marginally stable system (see Figures C.17 - C.18). Thus, for the purpose of this experiment, the gain K_I=0.025 is chosen as the "optimal" value.

3.2.3 Phase Control

There are several configurations in phase control. First, the phase angle between two measured outputs can be regulated. Second, the phase angle between an output and a reference signal can be regulated. Third, the phase angle between the input and the output can be regulated. All these are based on the same control methodology. For the present experimental setup, the phase angle between two plant outputs is regulated. To simplify software coding requirment, the cosine of the phase angle ($\cos\theta$) is regulated instead of the phase angle (θ) itself. Conversion between the two can be readily accomodated. The test range of the actual phase angle varies from 0 degree to 30 degrees.

Phase compensator is based on a phase shifter with the following equation of operation:

In accordance with the formula

$$Asin(\omega t + \theta) = A(sin\omega t cos\theta + cos\omega t sin\theta) \quad (3.1)$$

$$= Asin\omega t cos\theta + Acos\omega t sin\theta \quad (3.2)$$

When θ is small (within $\pm$ 15°), the approximation

$$sin\theta \simeq \theta \quad (3.3)$$

$$cos\theta \simeq 1 \quad (3.4)$$

can be made, resulting in

$$Asin\omega t + A\theta cos\omega t = asin\omega t + bcos\omega t \quad (3.5)$$

Table 3.3 Experiment Results for Phase Control

Case	gain K_p	command trajectory	Response time	Overshoot	Fig. No.
1	0.0025	output: $0^o \rightarrow 10^o$	0.68 s	no overshoot	Fig. C.19
		output: $10^o \rightarrow 20^o$	0.68 s	no overshoot	
		output: $20^o \rightarrow 30^o$	0.69 s	no overshoot	
		output: $30^o \rightarrow 20^o$	0.68 s	no overshoot	Fig. C.20
		output: $20^o \rightarrow 10^o$	0.45 s	no overshoot	
2	0.025	output: $0^o \rightarrow 10^o$	0.08 s	< 2.5 %	Fig. C.21
		output: $10^o \rightarrow 20^o$	0.08 s	< 2.5 %	
		output: $20^o \rightarrow 30^o$	0.082 s	< 2.5 %	
		output: $30^o \rightarrow 20^o$	0.08 s	< 2.5 %	Fig. C.22
		output: $20^o \rightarrow 10^o$	0.0785 s	< 2.5 %	

where:

$$a = A \tag{3.6}$$

$$b = A\theta \tag{3.7}$$

Therefore, by fixing a and varying b, the desired phase change can be effected. Again, this is based on the assumption the θ is small.

The command trajectories are chosen as follows: first, the phase command is switched from 0^o to 10^o, and held at the 10^o, and then switched from 10^o to 20^o, held at 20^o 5 second , finally, switched from 20^o to 30^o. The other trajectory is from 30^o to 10^o in a reversed order. Two sets of tuning gains K_P=0.0025, and K_P=0.025 are tested. The experimental results are shown in Figures C.19 - C.22, respectively. For the case the control gain K_P =0.0025, the system settling time is about 0.35 second. Further increasing gain from 0.0025 to 0.025 reduces the settling from 0.35 second to 0.05 second with less than 2.5 % overshoot. Since the gain K_P = 0.05 leads to instability, the "optimal" gain for this phase control configuration will be 0.025.

The experimental results for phase control are summarized in Table 3.3.

3.2.4 Two-Loop Amplitude & Phase Control

In a 2-loop amplitude & phase control problem, the control objective is to control the resonators' output amplitude and their phase angle. Again, the requirments of asymptotic regulation and fast speed of response are imposed. All the goals are achieved in the experiment. Controller structure remains the same as the one used in previous section, so the experiment can be carried out in almost the same way as before. The block diagram of 2-loop amplitude & phase control is shown in Figure 3.7.

In this experiment, the amplitude is controlled at 1 Volt for both outputs, and the phase angle regulated is limited between 0 degree and 15 degree; the $\cos(\theta)$ is still used as phase signal instead of using θ.

The experimental results are summarized in Table 3.4.

Figures C.23 & C.29 show the amplitude output after the square law demodulator, but before filtering, and Figures C.24 & C.30 show their baseband components after filtering. Figures C.25 - C.28 present the phase angle to be controlled between the two outputs with gains K_I=0.0025, and K_P= -0.0025 and K_I =0.025, K_P = -0.025; where K_I and K_P represent the controller gain for amplitude and phase , respectively.

As evident from the figures, when the gain pair are set to be K_I =0.0025 & K_P = -0.0025, it takes about 3 to 5 second for the system to reach steady state, and there is no overshoot or undershoot. So the control gains are raised to 0.025 and -0.025 to get fast system response. In this case, it takes about 1 second for the system to reach steady state. If the gain pair are further increased to 0.05 and -0.05, the system becomes unstable. So the gain pair 0.025, -0.025 is chosen as the "optimal" gain for this test.

Several other cases are tested just for further verification (Figures C.29 - C.32). Figures C.31 and C.32 are the results with gain 0.025 and -0.025 but more complex

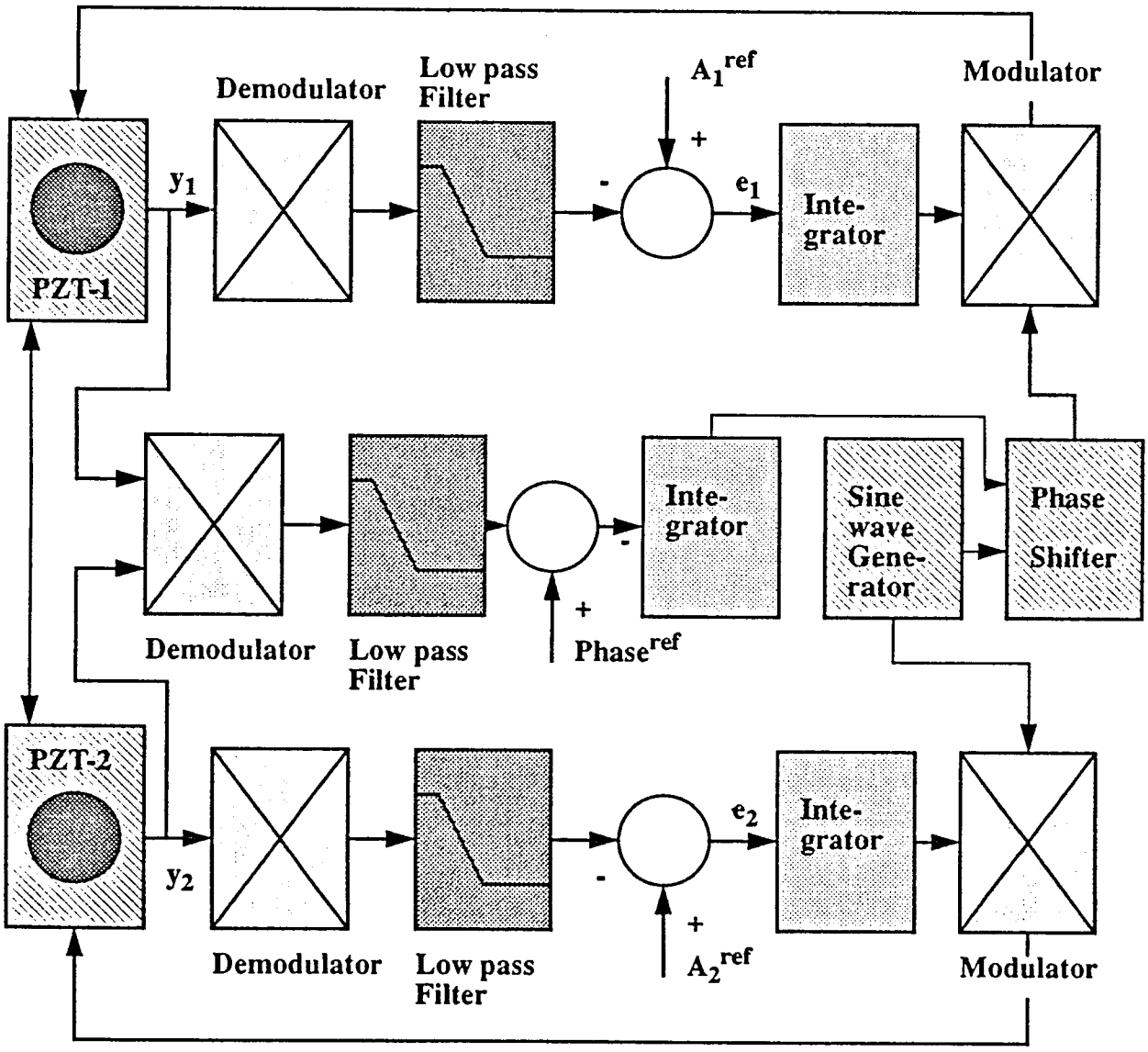


Figure 3.7 Block Diagram of 2-Loop Amplitude & Phase Control

Table 3.4 Experimental Results for 2-Loop Amplitude & Phase Control

Case	gains	command trajectory	Respose time	Overshoot	Fig. No.
1	0.0025	output: 0 → 10	3.55 s	no overshoot	Fig. C.25
	-0.0025	output: 10 → 15	3.45 s	no overshoot	
		output: 0 → 10	3.55 s	no overshoot	Fig. C.26
		output: 10 → 5	3.45 s	no overshoot	
2	0.025	output: 0 → 10	0.355 second	no overshoot	Fig. C.27
	-0.025	output: 10 → 15	0.345 s	no overshoot	
		output: 0 → 10	0.355 s	no overshoot	Fig. C.28
		output: 10 → 5	0.345 s	no overshoot	

command trajectories. It can be seen from these figures that the same performances as above are obtained .

The experiment results are summarized in the Table 4.4.

3.2.5 Comparison Between With and Without Feedforward Control

The implementation of feedforward control mainly provides two advantages: 1) speed up system response and 2) improve system performance such as reducing the overshoot, etc. The constant feedforward term is computed based on (2.26), (2.27) and (2.28). Figure 3.8 shows the system with feedforward control while the Figure 3.9 is the system response without feedforward control. Figure 3.8 presents a much fast response (about 2 times faster) than the one without feedforward.

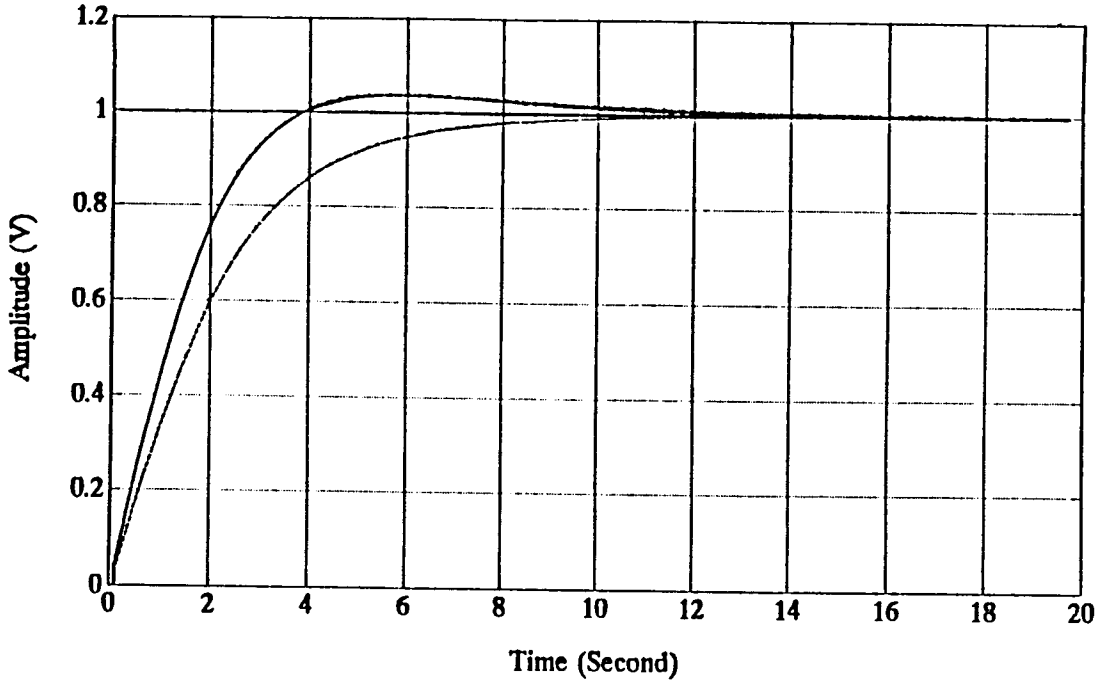


Figure 3.8 System Response with Feedforward Control

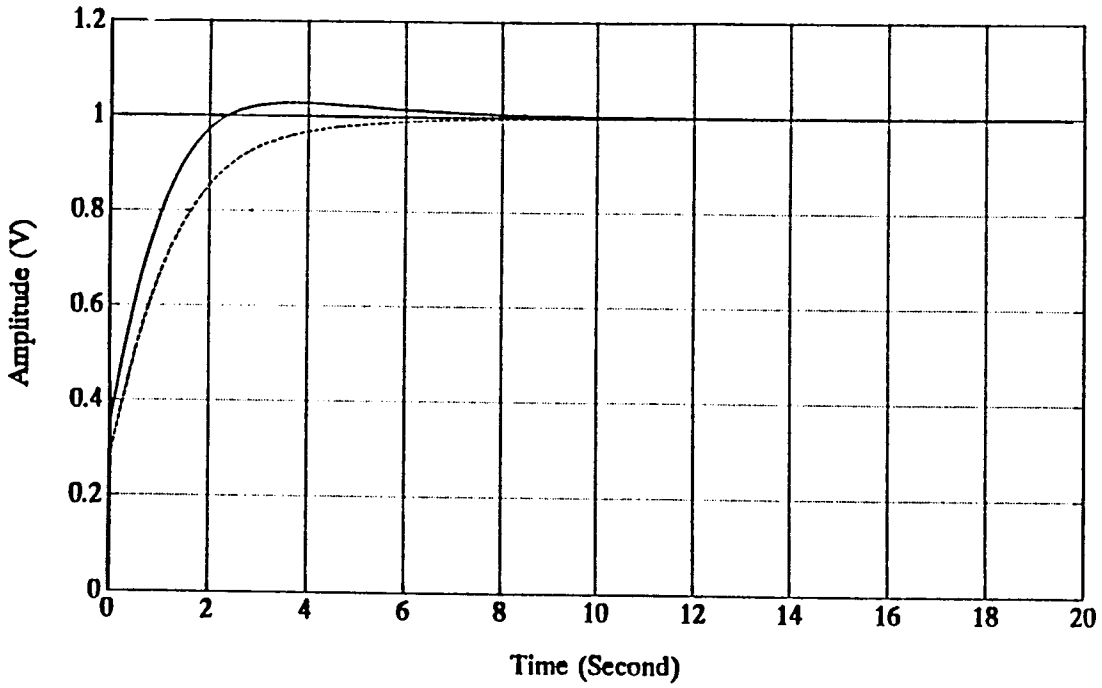


Figure 3.9 System Response without Feedforward Control

CHAPTER 4

SIMULATION RESULTS

In order to verify the experimental results of the controlled resonator system, numerical simulations have also been carried out using the ALSIM software. The plant model described in equation (2.58), (2.59), (2.60) and (2.61), together with the controller equations are simulated for various command trajectories and control gains. Since the number of experimental cases is large, only yhe major ones are presented here to be compared with the corresponding experimental results. The following cases are simulated: 1) single loop amplitude control with gain K_I = 0.0025, and K_I = 0.025, 2) 2-loop amplitude control with gain K_I = 0.0025, and K_I = 0.025.

4.1 Simulation for Single Loop Amplitude Control with gain K_I = 0.0025 and K_I = 0.025

The model for a single loop resonator can be written as:

$$p(s) = \frac{a + bs}{s^2 + \frac{\omega_c}{Q}s + \omega_c^2} \tag{4.1}$$

Since the two resonators have nearly identical characteristics, any one of the two can be selected for simulation.

The first resonator p_{11} is given by equation(2.58) as:

$$p_{11}(s) = \frac{-8.3568 \times 10^7 + 280.5253s}{s^2 + 1615.5s + 2.759 \times 10^9} \tag{4.2}$$

The corresponding state-space model of the plant and the control is given as:

$$\dot{x}_1(t) = x_2(t) \tag{4.3}$$

$$\dot{x}_2(t) = -1615.5x_2(t) - 2.759 \times 10^9 x_1(t) + u(t) \quad (4.4)$$

$$y(t) = -8.3568 \times 10^7 x_1(t) + 280.5256x_2(t) \quad (4.5)$$

$$\dot{x}_3(t) = x_4(t) \quad (4.6)$$

$$\dot{x}_4(t) = -\sqrt{2}\omega_f x_4(t) - \omega_f^2(x_3(t) - v^2(t)) \quad (4.7)$$

$$e(t) = A^{ref} - x_3(t) \quad (4.8)$$

$$\dot{x}_5(t) = K_I e(t) \quad (4.9)$$

$$u(t) = x_5(t)\sin(\omega t) \quad (4.10)$$

where: $x_1(t)$ and $x_2(t)$ are the plant states, $x_3(t)$ is the lowpass filter output which represents the amplitude information of the plant, $e(t)$ is the error between the reference and the signal amplitude, $x_5(t)$ is the integrator output and $u(t)$ is the controller output. The controller used here is the continuous time but nevertheless has the same essential structure as the discretized version used in the experiment. The plant output $y(t)$ is first squared and, then the lowpass filter with cutoff frequency $w_f = 1256.63$ rad/s is applied the get the amplitude of the signal. Finally, the integrator is used to regulate the amplitude to the desired reference. The simulation results and the corresponding experimental results are presented in Figure D.1 through Figure D.8 (refer to APPENDIX D). Figures D.1, D.2 are the results with gain 0.0025 for the first command trajectories (refer to Figure 3.4) and Figures D.3, D.4 are the simulation results and experimental results with gain 0.0025 for the second command trajectories (refer to Figure 3.5). Figures D.5 - D.8 are the results using gain 0.025.

From these figures, it can be seen that both the simulation results and experimental results show about 0.35 second settling time with no overshoot for the case of gain $K_I = 0.0025$. In the case of gain $K_I = 0.025$, the settling time for both results are about 0.07 second with no overshoot. Therefore it is concluded that simulation results match the experimental results well in single loop amplitude control.

4.2 Simulation for Two Loop Amplitude Control with gain $K_I = 0.0025$ and $K_I = 0.025$

The math model for the two loop resonator system is given in equations(2.58), (2.59), (2.60), and (2.61). The corresponding state-space model of the closed loop system is given by:

$$\dot{x}_1(t) = x_2(t) \tag{4.11}$$

$$\dot{x}_2(t) = -\frac{\omega_c}{Q_{11}}x_2(t) - \omega_c^2 x_1(t) + u_1(t) \tag{4.12}$$

$$\dot{x}_3(t) = x_4(t) \tag{4.13}$$

$$\dot{x}_4(t) = -\frac{\omega_c}{Q_{12}}x_4(t) - \omega_c^2 x_3(t) + u_2(t) \tag{4.14}$$

$$y_1(t) = -8.36 \times 10^7 x_1(t) + 280.5x_2(t) + 8.4 \times 10^7 x_3(t) + 280.5x_4(t) \tag{4.15}$$

$$\dot{x}_5(t) = x_6(t) \tag{4.16}$$

$$\dot{x}_6(t) = -\sqrt{2}\omega_f x_6(t) - \omega_f^2(x_5(t) - y_1^2(t)) \tag{4.17}$$

$$e_1(t) = A_1^{ref} - x_5(t) \tag{4.18}$$

$$\dot{x}_7(t) = K_I^{11} e_1(t) \tag{4.19}$$

$$u_1(t) = x_7(t)\sin(\omega_c t) \tag{4.20}$$

$$\dot{x}_8(t) = x_9(t) \tag{4.21}$$

$$\dot{x}_9(t) = -\frac{\omega_c}{Q_{21}}x_9(t) - \omega_c^2 x_8(t) + u_1(t) \tag{4.22}$$

$$\dot{x}_{10}(t) = x_{11}(t) \tag{4.23}$$

$$\dot{x}_{11}(t) = -\frac{\omega_c}{Q_{22}}x_{11}(t) - \omega_c^2 x_{10}(t) + u_2(t) \tag{4.24}$$

$$y_2(t) = 6.1 \times 10^7 x_8(t) + 204.3x_9(t) - 7.8 \times 10^7 x_{10}(t) + 261.4x_{11}(t) \tag{4.25}$$

$$\dot{x}_{12}(t) = x_{13}(t) \tag{4.26}$$

$$\dot{x}_{13}(t) = -\sqrt{2}\omega_f x_{13}(t) - \omega_f^2(x_{12}(t) - y_2^2(t)) \tag{4.27}$$

$$e_2(t) = A_2^{ref} - x_{12}(t) \tag{4.28}$$

$$\dot{x}_{14}(t) = K_I^{22} e_2(t) \tag{4.29}$$

$$u_2(t) = x_{14}(t)\sin(\omega_c t) \tag{4.30}$$

where: $x_1(t)$, $x_2(t)$, $x_8(t)$ and $x_9(t)$ are the corresponding state variables of the p_{11}, p_{12}, p_{21} and p_{22}. $x_5(t)$, $x_7(t)$ and $u_1(t)$ are, respectively, the demodulator output, the integrator output, and the controller output for the first control agent. Similarly, $x_12(t)$, $x_14(t)$ and $u_2(t)$ are, respectively, the demodulator output, the integrator output, and the controller output for the second control agent.

Again, the respective experimental results in Chapter 3 are reproduced here along with the simulation results for comparison. It is observed from Figures D.9-D.16 all the cases with gain $K_I = 0.0025$ present a 0.35 second settling time and a small overshoot/undershoot. Comparing with the experimental results, the same settling time is observed except that there exist a slight discrepancy between the direction of overshoot. Similarly, Figures D.17 -D.20 show the close agreement between simulation results and experimental results in all case, the settling time is about 0.075 second.

4.3 Discussion

Through comparing the simulation results with experimental results, it is observed that the simulated closed loop performance is in good agreement with that observed experimentally. The slight discrepancies in overshoot/undershoot are mainly due to unmodelled dynamics and the intrinsic nonlinearity in the actual PZT devices. It is therefore concluded that the mathematical models obtained by experimental method in this work provide good description of the resonator dynamics.

CHAPTER 5
CONCLUSIONS

In thesis, the problem of real-time control of a dual, interconnected resonator system is considered. The experimental apparatus consists of a pair of PZT disks, an aluminum base plate , and the necessary digital control hardware (A/D, D/A TMS320C30 Digital Signal Processor, and electronic circuits). The objective is to regulate the gain/phase characteristics of the two PZT disk at a resonant frequency of 8.36 kHz.

A number of preliminary testings have been carried out to verify the linearity and wideband frequency response characteristics. Local modelling around the 8.36 kHz resonance is then carried. The mathematical models are used for simulation purpose.

Experimentally, a number of test configurations have been selected: single loop amplitude control, 2-loop amplitude control, phase control, and 2-loop amplitude & phase control. The controller structure and synthesis method are based on the theoretical results described in [1]. From the experimental results, it is found that a decentralized controller combined with the on-line tuning method yields satisfactory closed loop response with minimum controller complexity.

Simulation results further confirm the validity of the approach as well as the structure of the experimental, real-time controller.

The furture work can be suggested : 1) control of more complex and more compact resonator structure such as the resonator configured as sandwich form; 2) control the system with more input and output, say in the pattern, there can be seven inputs and seven outputs and 3) further improve interface between the plant and the controller which includes the hardware and software. Finally, 4) expend amplitude & phase control to more flexible values to fit the requirements of realistic applications.

APPENDIX A

FREQUENCY RESPONSE OF RESONATORS WITH CENTRAL FREQUENCY 8.36 kHZ, BY EXPERIMENTAL AND THEORETICAL METHOD

A.1 Frequency Response of Resonators with Central Frequency 8.36 kHz by Experimental Method

In this section, the results of the frequency response of the resonators with central frequency equals 8.36 kHz obtained by experimental method are presented.

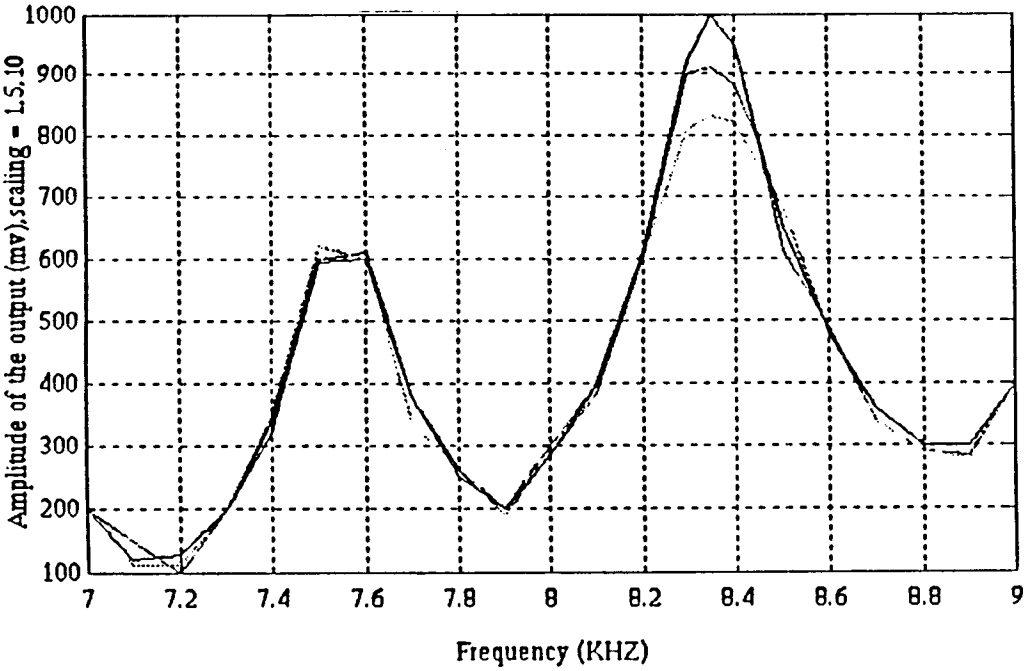


Figure A.1: Frequency Response of Resonator Device with Central Frequency f_c =8.36 kHz p_{11}

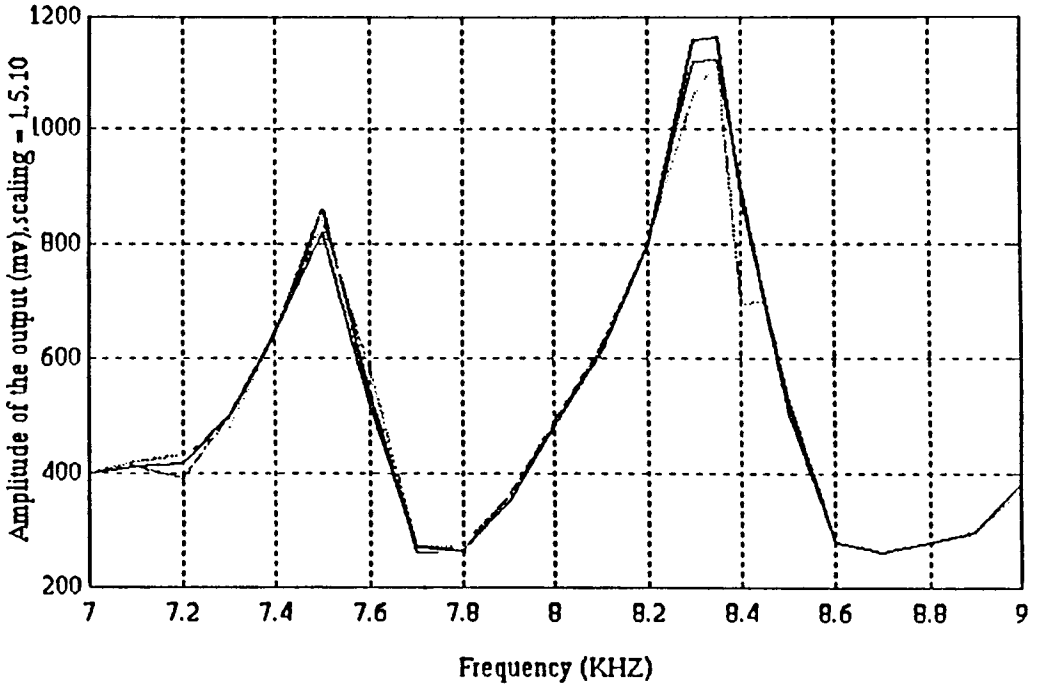


Figure A.2: Frequency Response of Resonator Device with Central Frequency f_c =8.36 kHz p_{12}

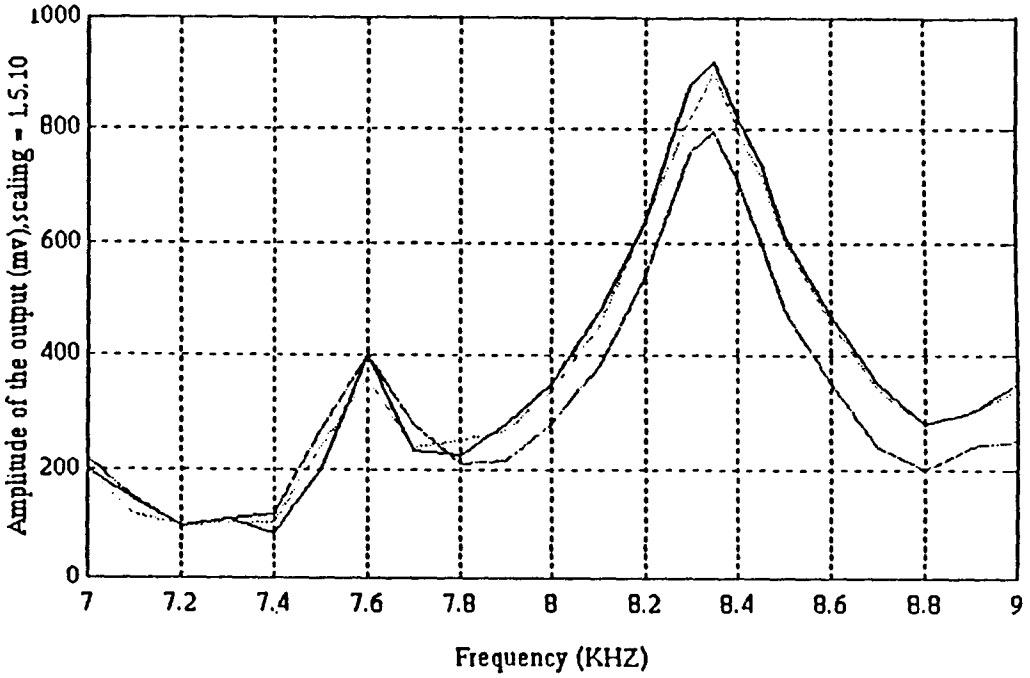


Figure A.3: Frequency Response of Resonator Device with Central Frequency f_c =8.36kHz p_{21}

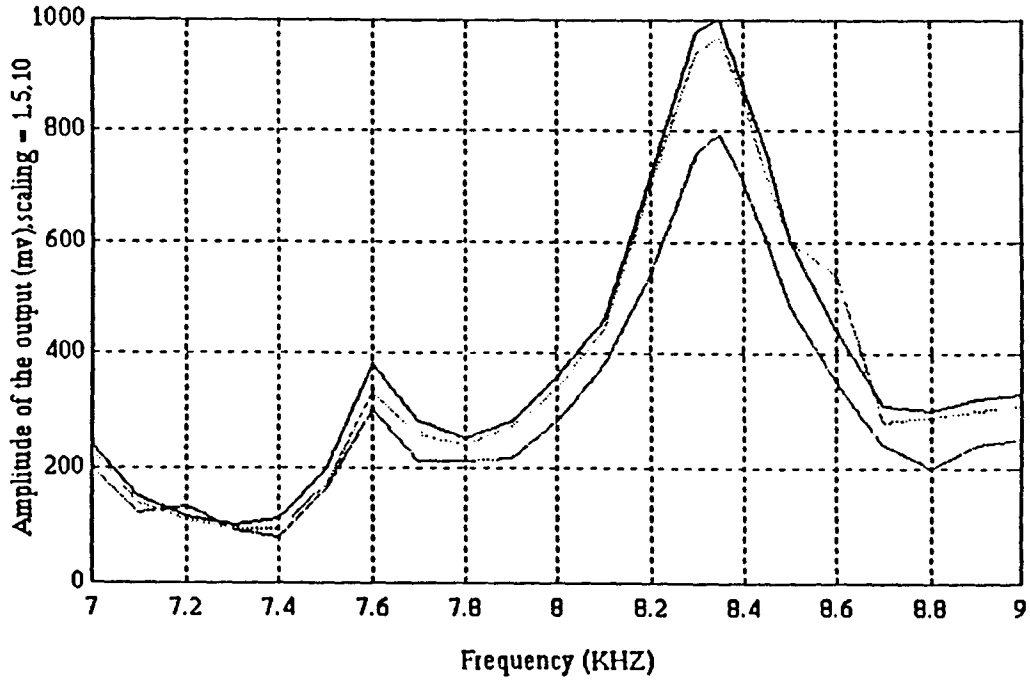


Figure A.4: Frequency Response of Resonator Device with Central Frequency f_c =8.36 kHz p_{22}

A.2 Frequency Response of Resonators with Central Frequency 8.36 kHz by Theoretical Method

In this section, the results of the frequency response of the resonators for central frequency equals 8.36 kHz obtained by theoretical method, model matrix, are presented.

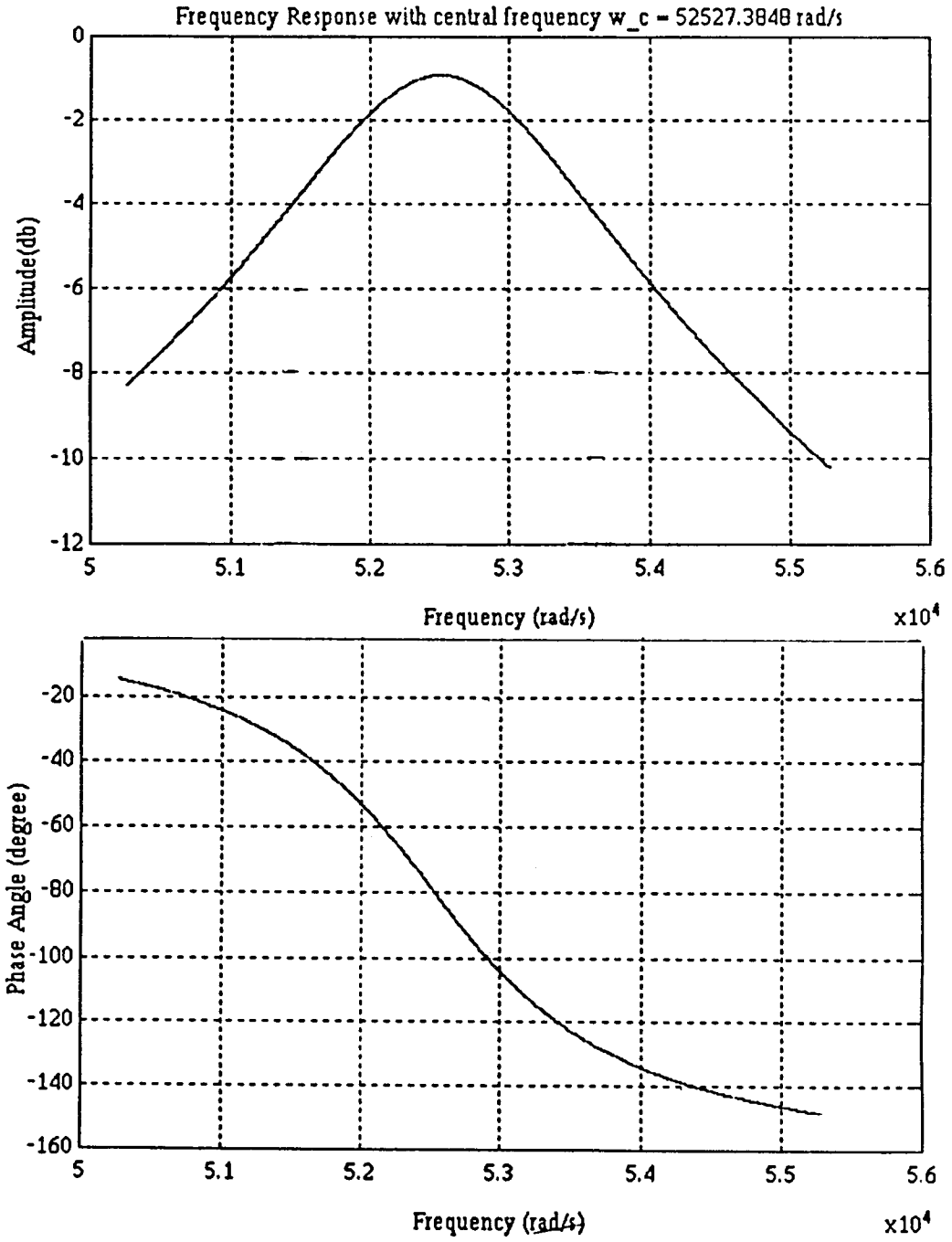


Figure A.5: Frequency Response of Resonator Device with Central Frequency f_c =8.36 kHz P_{11}, (theoretically)

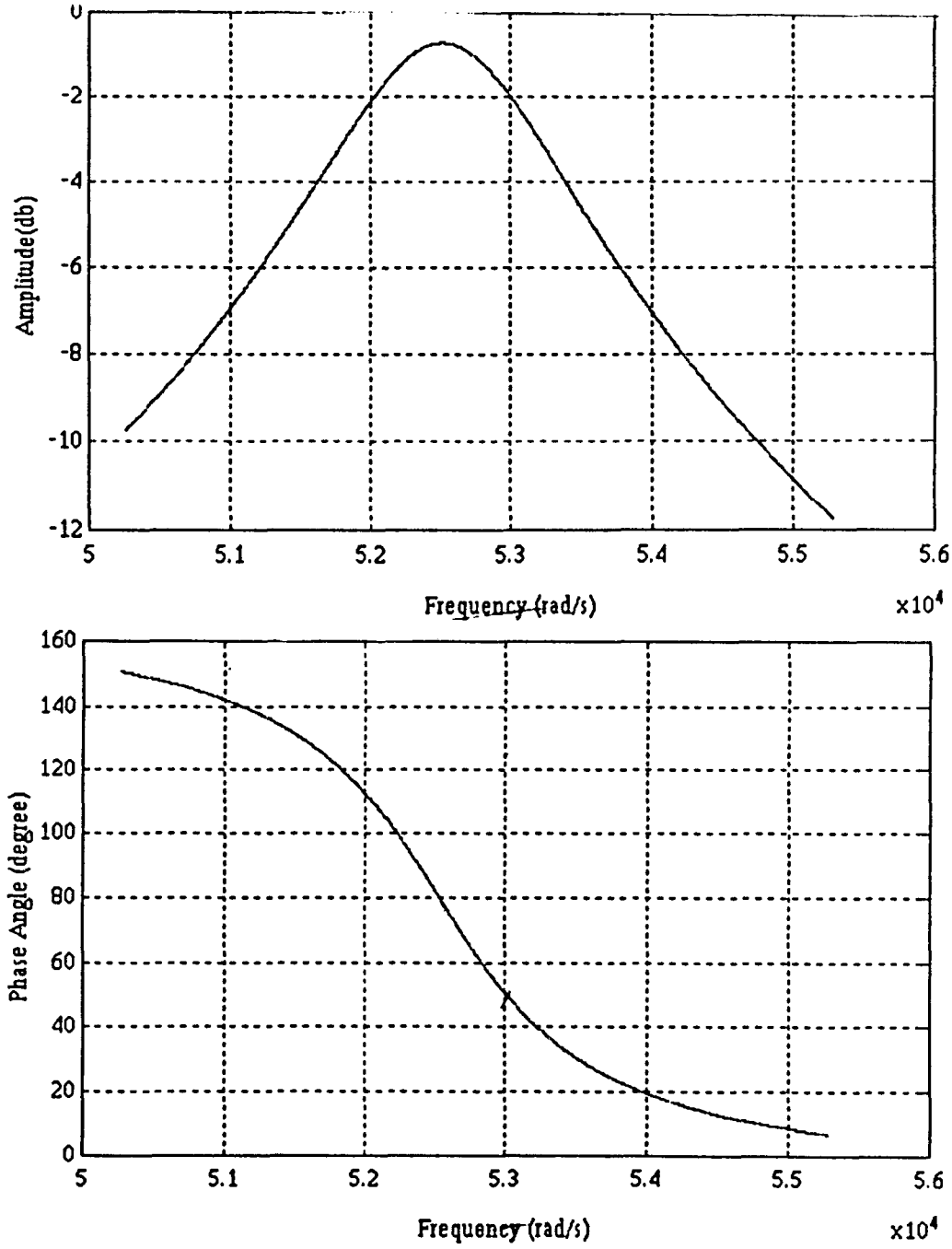


Figure A.6: Frequency Response of Resonator Device with Central Frequency f_c =8.36kHz P_{12}, (theoretically)

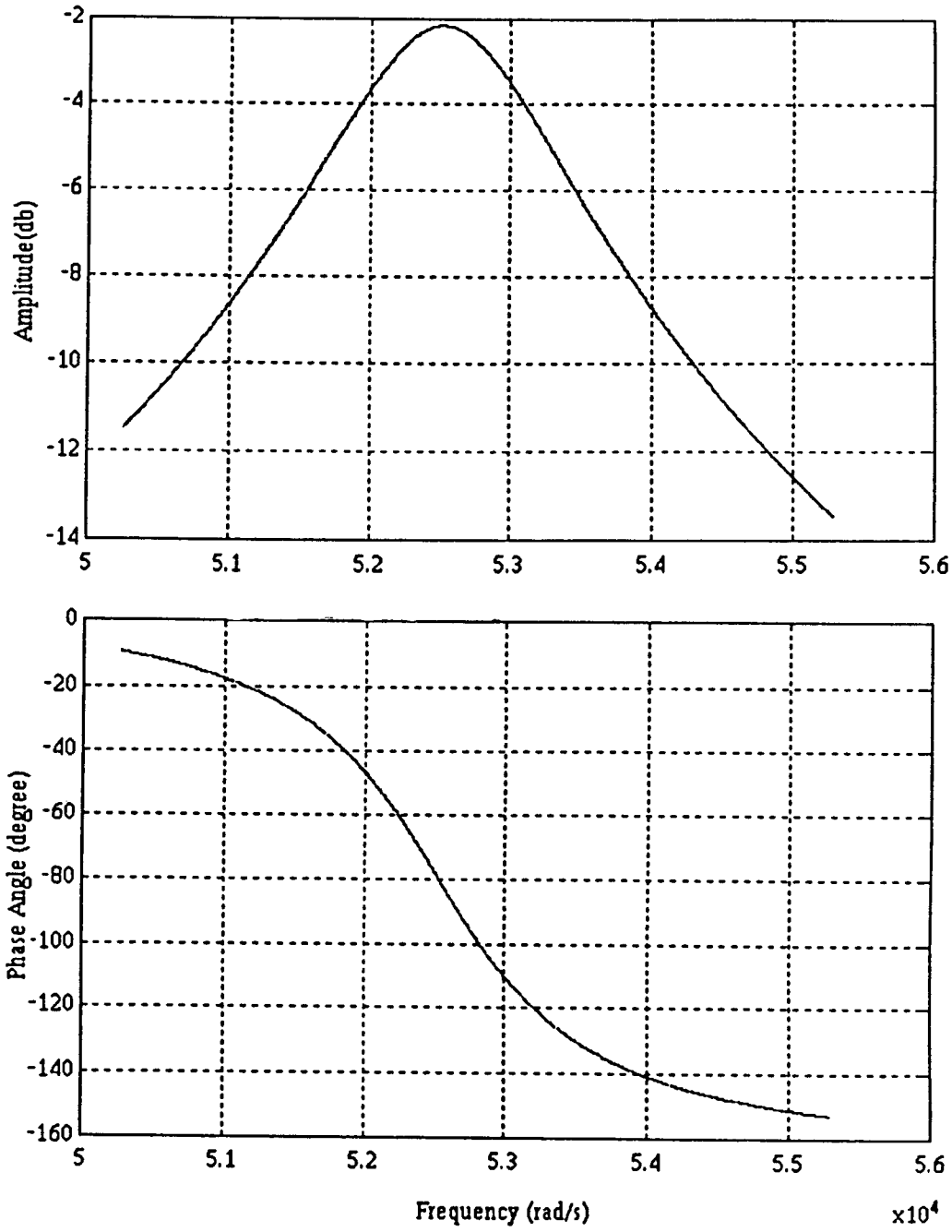


Figure A.7: Frequency Response of Resonator Device with Central Frequency f_c =8.36 kHz P_{21}, (theoretically)

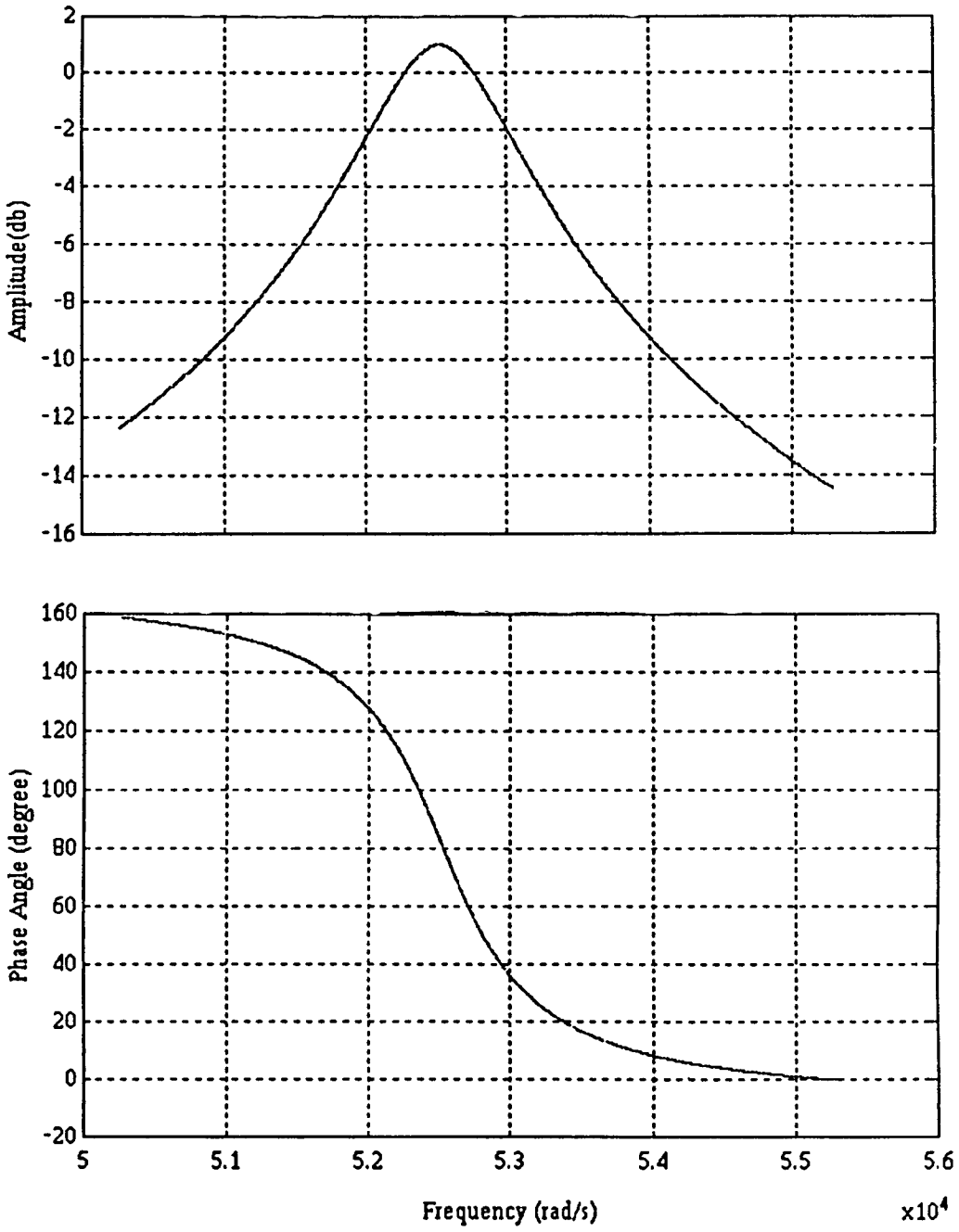


Figure A.8: Frequency Response of Resonator Device with Central Frequency f_c =8.36 kHz P_{22}, (theoretically)

APPENDIX B

DESCRIPTION OF HARDWARE AND SOFTWARE

B.1 Resonators

Dimension of Disk	: diameter = 1 in, thickness = 0.05in
Material	: LZT-2
Dimension of Plate	: 4 in × 4 in × 0.1 in
Material	: Aluminum

B.2 HARDWARE DESCRIPTION

B.2.1 SPIRIT-30 SYSTEM

B.2.1.1 System Overview The SPIRIT-30 system offers 33.3 MFLOPS (million floating-point operations per second) and 16.7 MIPS (million instructions per second) of performance for PC AT/386 and 100 % compatible computers and is embraced by a complete application development environment. The SPIRIT-30 board, along with comprehensive development software (SPIRIT-EDSP) and DSP program library (DSPL) is well suited for embedded applications for image analysis, graphics, numerical computation, control systems, telecommunications and other high performance applications.

The heart of the SPIRIT-30 system is the Texas Instrument's TMS320C30 Digital Signal Processor (DSP) with 60 ns instruction cycle time, 2Kx32 words of internal RAM, single cycle floating-point multiply/accumulate and an on chip DMA controller.

A spirit-30 functional block diagram is given in Figure B.1.

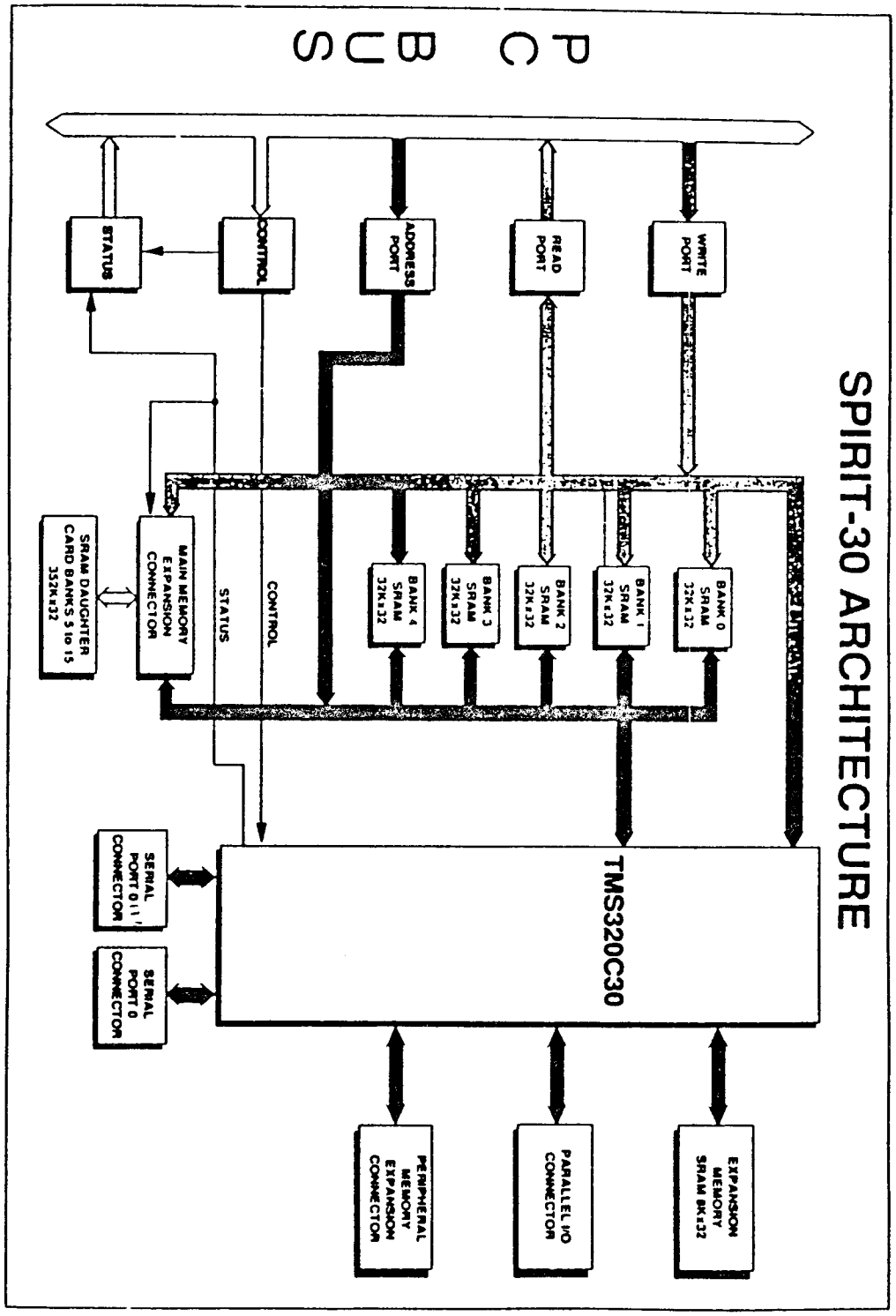


Figure B.1 Functional Block Diagram of SPIRIT - 30 System

B.2.1.2 System Features The SPIRIT-30 system has the following features:

- Texas Instrument's high performance floating-point Digital Signal Processing chip (TMS320C30)
- Single slot IBM AT/386 direct plug-in solution
- All communication with the SPIRIT-30 are 16 bits
- Up to 640 Kbytes of dual access RAM for 0 wait state
- Capability to expend entire memory range, up to 64 Mbytes
- Very versatile communication between the host PC and the SPIRIT-30
- Two schemes of data transfer between the host and SPIRIT-30 , namely DMA and program transfer

B.2.1.3 Processor The processor used for application acceleration is the TMS320C30 Digital Signal Processor. This processor offers the following features:

- 60-ns single-cycle instruction execution time
- One 4K x 32-bit -cycle dual access on chip ROM block
- Two 1K x 32-bit single cycle dual-access on chip RAM block
- 64 x 32-bit instruction cache
- Up to 16M x 32-bit of addressable memory space
- 40/32-bit floating point/integer multiply and ALU
- 32 bits barrel shifer
- on chip Direct Memory Access (DMA) controller for concurrent I/O and CPU operation

- Parallel multiply and ALU instructions in single cycle
- Zero overhead loops with single cycle branches
- Two serial ports to support 8/16/32 bits transfers
- Two 32 bits timers

B.2.1.4 Communication between Host and the SPIRIT-30 There are several means of communication between the host and SPIRIT-30

- Interrupt from the SPIRIT-30 to the host

  The SPIRIT-30 can interrupt the host via the TCLK0 pin of the DSP.

- Interrupt from the host to the SPIRIT-30

  The host can interrupt the SPIRIT-30 via the control register. One of the control register i connected to the INT0* line of the DSP. A library routine called pc _dsp _ int() can be used to perform this function.

- Status Register read by the host of the SPIRIT-30 flags

- Status read by the DSP of the INT0* line

  The DSP has a register in which all interrupts are latched irrespective of whether, they are enabled or not. Thus, a programmer can disable INT0*, but can still determine whether an interrupt happened from the host by polling the status of the INT0* bit in the interrupt flag (IF) register of the DSP.

B.3 Stereo Audio Interface Box (SAIB)

B.3.1 General Description

The SAIB provides the user with high quality, low cost stereo A/D and D/A capability. The SAIB features are as following:

- Low cost dual-channel stereo audio interface
- 16-bit resolution and 80 dB dynamic range
- Programmable sampling rates and gain
- Built-in input anti-aliasing filters and output low-pass smoothing filters
- Versatile line, microphone, handset, headphone, and speaker connections
- Standard ASM-S serial interface

The SAIB resides externally to the host machine and connects serially to the SPIRIT-30 Digital Signal Processor (DSP) board. The incoming signal can be looped back to the output while the host or the DSP is processing or streaming the data.

B.3.2 SAIB Modes of Operation

The SAIB has two modes of operation, Data-Mode, and Control-Mode. Control-Mode is used for software configuration of bit fields that remain static for the duration of an acquisition, such as the data formatted the conversion clock frequency. The SAIB operates in Data-Mode when performing conversions. In either mode of operation. two 32-bit words must be send to (and received from) the SAIB via the C30 serial port for each SAIB conversion cycle (The SAIB requires, and provides, 64 bits per stereo sample). The function of the nits in serial data stream is different in each of the two modes.

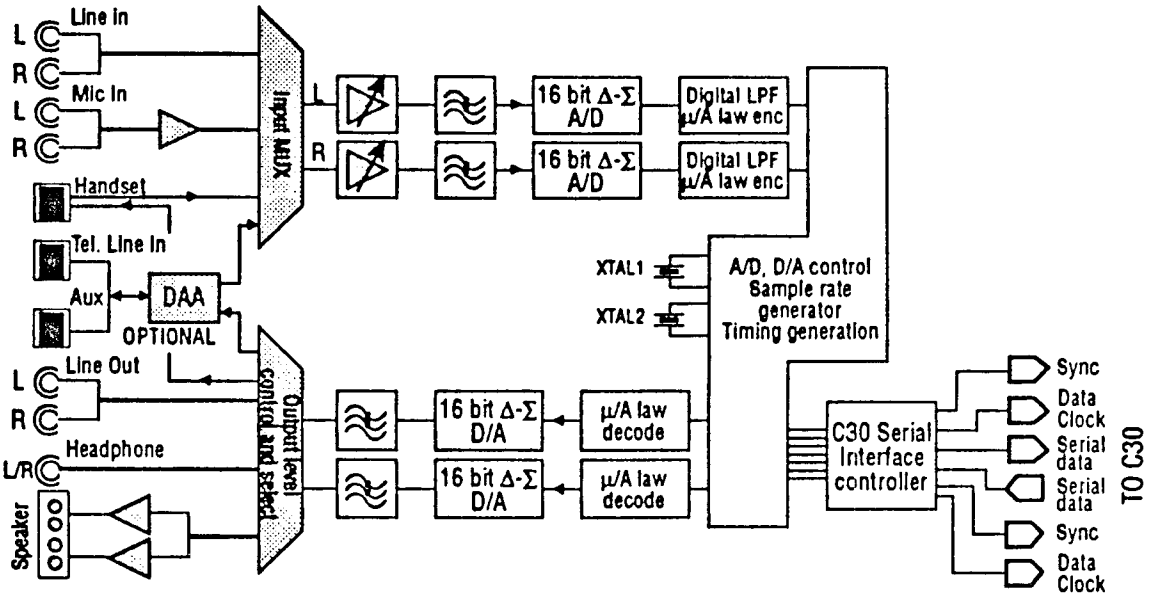
Figure B.2 is the SAIB Block Diagram.

B.3.3 SAIB Hardware Specifications

B.3.3.1 Analog Input and Output Channels : 2 channels A/D and 2 channels D/A; AC Coupled; fc 4Hz

Signal Coding : 16-bit linear, 8-bit μ-law coding

Sampling rates : Group 1: 8,9.6,16,27.42857,32,48 kHz (software selectable);



SAIB Block Diagram

Figure B.2 SAIB Block Diagram

Group 2; 5.5125,6.615,11.025,18.9,22.05,33.075,37.8,44.1 kHz (software selectable)

1. Analog Input

   Dynamic range: Line 80 dB; All others 72 dB

   S/(N+D): Linear 74 dB; All others 66 dB

   Inter-channel isolation: Line 80 dB; Mic 60 dB

   Full scale input: Line 2.8 Vpp; Mic .29 Vpp

   Gain: Software programmable for line inputs .2 to 22.7 dB; Mic inputs 19.8 to 42.7 dB

   Filtering: Passband 0 – .45 fs + .1 dB; Stopband > .55 fs >74 dB

   Inputs: Line level, Microphone ,etc.

2. Analog Output

   Dynamic range: 80 dB

   S/(N+D): Linear 74 dB; All others 60 dB

   Inter-channel isolation: Line 78 dB; Mic 40 dB

   Differential non-linearity: +/−.9 LSB

   Output Voltage: Line & headphone 2.8 Vpp;

   Attenuation: Software programmable from -.2 to -94.7 dB

   Rated output impedance: Line 10 kOhm

   Filtering: Passband 0 - .45 fs + .1 dB; Stopband > .55 fs >74 dB

   Outputs: Line level, Microphone ,etc.

   LED indicators: Red off-hook; Yellow handset sect; Green ring detect

B.4 SOFTWARE DESCRIPTION

B.4.1 SAIB Programming Steps

The SAIB is a data-streaming device which interface to the Sonitech SPIRIT-30 digital signal processor via one of its two TMS320C30 serial ports. All features of the SAIB are software controlled thru commands embedded in the data-stream. The SAIB has two 16 bits analog I/O channels, and therefore, produces (or expects) 32 bit audio data words. In order to include the additional flexibility allowed by the software programmable controls, an additional 32 bits control word must be sent along with each audio data word. This combination of data and control forms a Word-Pair, one of which is required to be sent for each conversion cycle. What follows is a sequence of steps required to take the SAIB from a reset state, to an operating one.

The SAIB is hardware-reset during power-up and begins operating in Data-Mode. Until the SPIRIT-30's serial-port is configured, and code has been downloaded, the SAIB cycles in-and-out of its reset state. A disabled serial-port, or a dis-connected

serial-port cable will appear to the SAIB as Data(DMD-Word) and Control(CMC-Word) words with all bits set (1), and is interpreted by the SAIB as a Soft-RESET command. The following is the programming steps:

1. Configure the TMS320C30 serial-port:

   The SAIB requires that the serial-port be configured for:

   - Fixed Continuous Mode With Frame Sync
   - 32 bit receive and transmit word length
   - FSR/FSX active-high polarity
   - DR/DX active-high polarity
   - CLKR/CLKX active-high polarity
   - CLKR/CLKX externally generated
   - FSX externally generated

2. Transition SAIB to Control-Mode operation

   - Write a DMD-Word of 0x00000000 (Null data) to the serial-port data transmit register.
   - Write a DMC-Word of 0x3f3fc0f0 to the serial-port data transmit register.
   - At the end of the present DMC-Word transmission, the SAIB will enter Control-Mode. The SAIB expects the next word-pair to be a CMC0-Word/CMC1-Word pair.
   - Set-up the Control-Mode control data
   - Repeatedly send the CMC0/CMC1-Word pair to the serial data-transmit register with the DCB bit clear(DCB = 0)
   - Read back and verify the control information from the SAIB until the DCB bit clears (DCB = 0)

3. Transition SAIB to Data-Mode operation

   - Send the CMC0/CMC1-Word-pair to the serial-port data-transmit register with the DCB bit set (DCB=1). (This enables the CMC0/CMC1 control information into the SAIB's internal control registers).
   - Send the CMC0/CMC1-Word-pair to the serial-port data-transmit register with the DCB bit set (DCB = 1) AND the D/C* bit set (D/C* = 1).(This latches the CMC0/CMC1 control information into the SAIB's internal control registers and transitions the SAIB into Data-Mode operation).
   - At the end of present CMC1-Word transmission, the SAIB will enter Data-Mode. The SAIB expects the next word-pair to be a DMD-Word/DMC-Word Pair.

4. Calibrate the SAIB

   - The SAIB performs an internal offset-calibration cycle each time it exits from Control-Mode. 256 conversion cycles of Null Data required to perform the calibration.
   - Write a DMD-Word of 0x00000000 (Null Data) to the serial-port data transmit register.
   - Write a DMC-Word of 0x7F7FD0F0 to the serial-port data-transmit register.
   - Send the DMD/DMC-Word pair to the serial-port data-transmit register 255 times.

5. Begin SAIB Data-Mode operation

   After calibration, the SAIB begins normal Data-Mode operation.

   Because SAIB control information is contained in the serial data-stream, the

SAIB requires a properly sequenced DMD/DMC word-pair sent to it for each conversion cycle, regardless of whether A/D, D/A, or A/D and D/A conversions are desired.

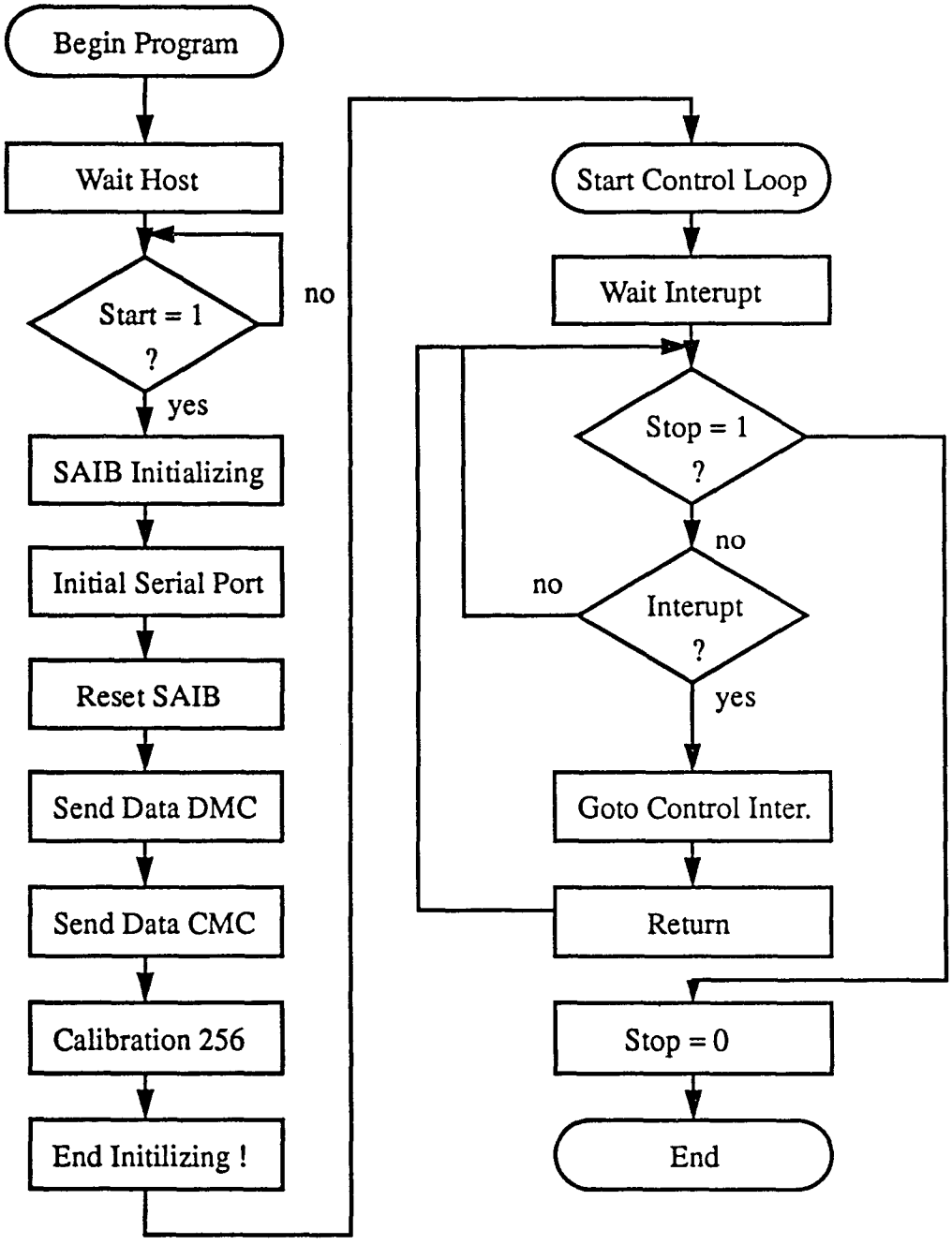
6. During the Data-Mode operation, execute the REAL-TIME CONTROL MODE

   After end the calibration, the program enter the real-time control mode.

   - Receive Data from serial-port, and transfer to the floating point number.
   - Modulator to get the amplitude and phase angle of the input signal.
   - Compare to the reference to form an error.
   - Integrator and feedforward controller.
   - Demodulator to form output.
   - Transfer data back to unsigned long integer and output.
   - Send data to the file for Matlab processing.

B.4.2 Program Flowchart and Program List

In this section, programming flowchart and programs are listed. Program 1, Host program, is the host computer, PC, program which control the DSP program and data transfer. Program 2 is the DSP program which performances the tasks included initialize SAIB and serial-port, loop back data information and execute real-time control. Program 3 is used for transfering data from DSP to Matlab for processing and analysing. Program 4 is the ALSIM program.



Flowchart 1

Figure B.3 Programming Flowchart-1

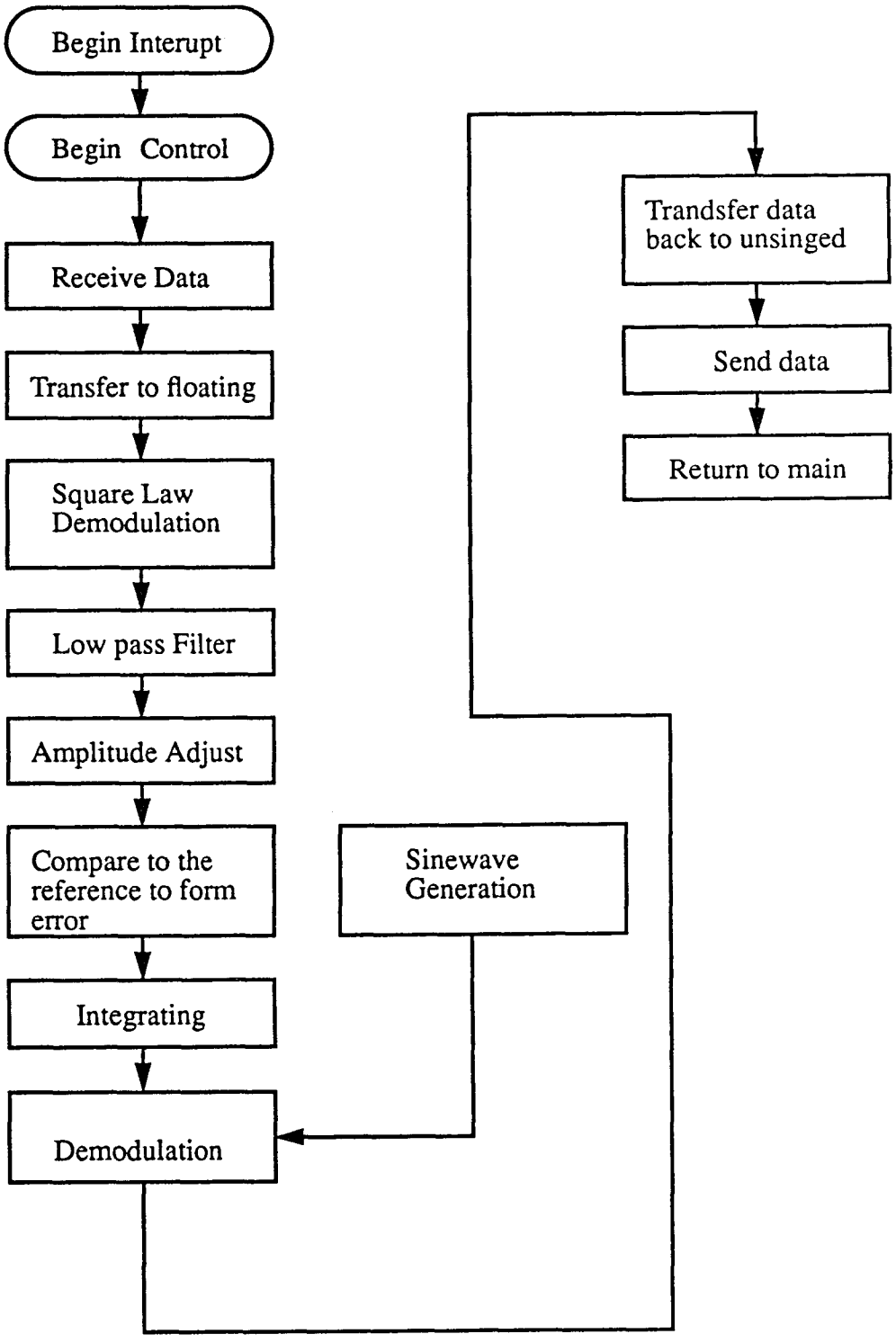


Figure B.4 Programming Flowchart-2

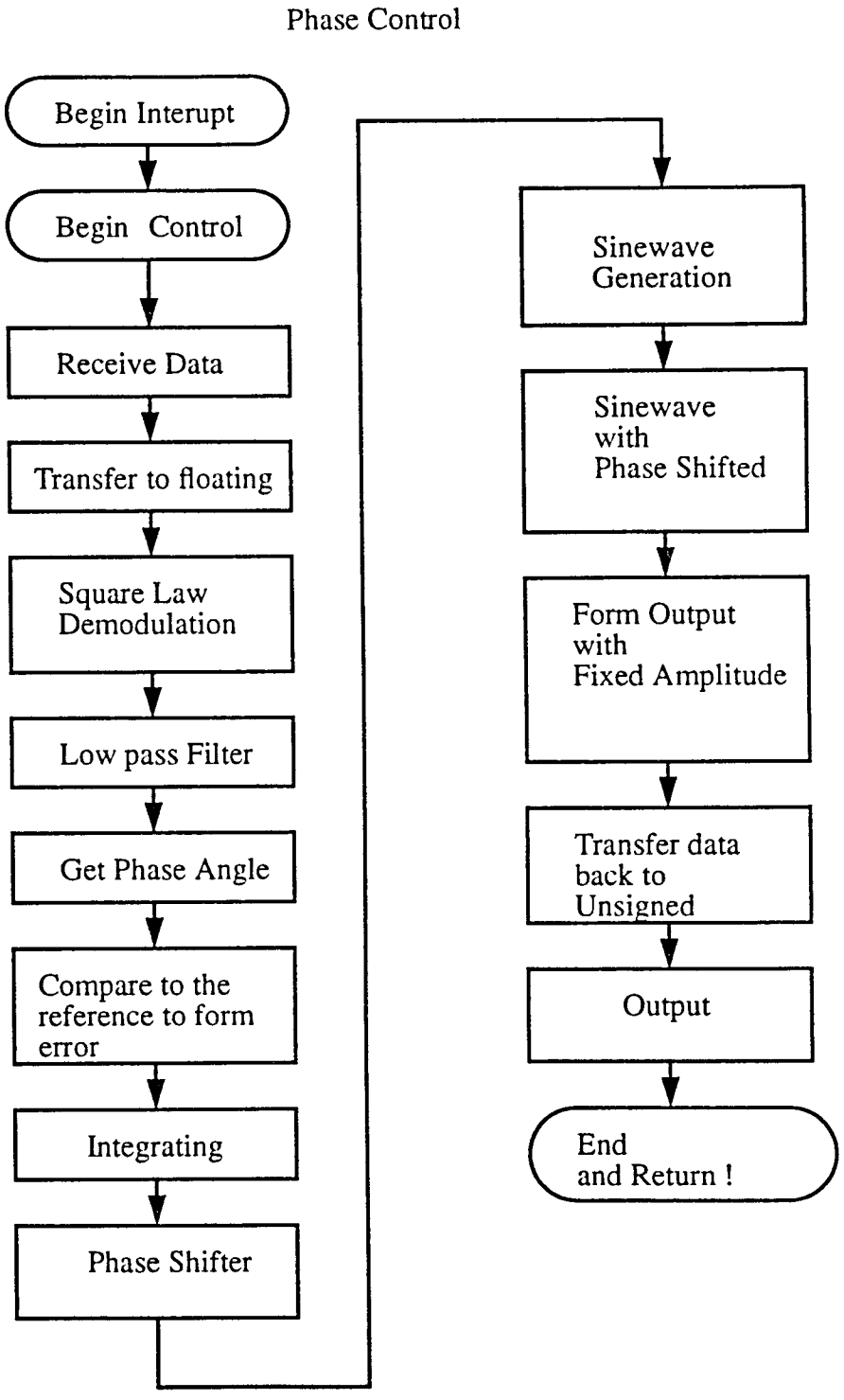


Figure B.5 Programming Flowchart-3

Program 1 Controlh.c in Host PC

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dos.h>
#include <conio.h>

#include <menus.h>
#include <bit_defn.h>
#include <display.h>
#include <s30tools.h>

#define BUFFER_SIZE 4096                /* *** */

unsigned long buffer0[BUFFER_SIZE];     /* Buffer 0 */
unsigned long buffer1[BUFFER_SIZE];     /* Buffer 1 */

unsigned int port_num = 0;              /* Serial port number (0/1) */
unsigned int rx_cmc1 = 0;               /* *** */
int dsp_exe_dl_status;                  /* */
unsigned int base_address;              /* */
int   j,k,l,i;                          /* Miscellaneous integers */
int m = 0;                              /* First-iteration flag */
int n = 0;                              /* character counter */

/* File Operations Variables */

FILE *record1_file;                     /* File to accept record-data */
FILE *record2_file;                     /* File to source playback-data */
int write1_ret = 0;                     /* */
int write2_ret = 0;                     /* */
char ch = '0';                          /* */
char s[3];                              /* */

int y11_ref = 0;                        /* Define Some Control Variables */
int y12_ref = 0;
int ki1   = 0;
int ki2   = 0;

float k1   = 0.;
float k2   = 0.;
float y1_ref = 0.;
float y2_ref = 0.;
```

```
/* Semaphores and Flags */

unsigned int start =  1;                    /* semaphore: Host starts dsp */
unsigned int stop =   1;                    /* semaphore: Host stops dsp */

unsigned int dmcflag = 1;                   /* flag: *** */
unsigned int buf0_full = 0;                 /* flag: buffer 0 full */
unsigned int buf1_full = 0;                 /* flag: buffer 1 full */
unsigned int init_complete = 0;             /* flag: 1 => init. complete */

/* Register Copies */

unsigned long newdmc =  0;                  /* *** */
unsigned long dmc = 0;                      /* *** */
unsigned long cmc0 = 0;                     /* *** */
unsigned long cmc1 = 0;                     /* *** */


long unsigned buffer0_addr = 0;             /* Buffer 0 */
long unsigned buffer1_addr = 0;             /* Buffer 1 */

long unsigned port_num_addr = 0;            /* Serial port number (0/1) */
long unsigned rx_cmc1_addr = 0;             /* SAIB chip number */

long unsigned start_addr = 0;               /* semaphore: Host says start */
long unsigned stop_addr =  0;               /* semaphore: Hots says stop */

long unsigned init_complete_addr = 0;       /* flag: initialization complete */
long unsigned buf0_full_addr =     0;       /* flag: 1 => Buffer 0 full */
long unsigned buf1_full_addr =     0;       /* flag: 1 => Buffer 1 full */
long unsigned dmcflag_addr =       0;       /* flag: 1 => *** */

long unsigned dmc_addr =           0;       /* DMC */
long unsigned newdmc_addr =        0;       /* *** */
long unsigned cmc0_addr =          0;       /* */
long unsigned cmc1_addr =          0;       /* */

long unsigned ki1_addr =           0;
long unsigned ki2_addr =           0;
long unsigned y11_ref_addr =       0;
long unsigned y12_ref_addr =       0;
```

```
main(){
                base_address = 0x300;                   /* Get address */

                BASE_ADD = base_address;                /* Convert s[] to binary */
                outpw(BASE_ADD+6,0);                    /* Write null to address */

                /*** Download SPIRIT-30 program ***/

    if(( dsp_exe_dl_status = dsp_dl_exec("control.out"))==-1){
                 printf("\n Could not find control.out --Download Error\n");
                 exit(0);     /* Message & exit if error */
                }

    dsp_reset();      /*** Reset SPIRIT-30 ("DSP") card ***/

    if ((start_addr = get_laddr("_start"))==-1){
                 printf("start flag is not present in dsp code\n");
                 printf(" Or error while getting the address of start flag\n");
                 exit(1);
                }
    if ((stop_addr = get_laddr("_stop"))==-1){
                 printf("stop flag is not present in dsp code\n");
                 printf(" Or error while getting the address of stop flag\n");
                 exit(1);
                }
    if ((dmcflag_addr = get_laddr("_dmcflag"))==-1){
                 printf("DMCFLAG flag is not present in dsp code\n");
                 printf(" Or error while getting the address of DMCFLAG flag\n");
                 exit(1);
                }
    if ((cmc0_addr = get_laddr("_cmc0"))==-1){
                 printf("CMC0 is not present in dsp code\n");
                 printf(" Or error while getting the address of CMC0\n");
                 exit(1);
                }
    if ((cmc1_addr = get_laddr("_cmc1"))==-1){
                 printf("CMC1 is not present in dsp code\n");
                 printf(" Or error while getting the address of CMC1\n");
                 exit(1);
                }
    if ((newdmc_addr = get_laddr("_newdmc"))==-1){
                 printf("NESDMC pointer is not present in dsp code\n");
                 printf(" Or error while getting the address of NEWDMC pointer\n");
                 exit(1);
                }
    if ((dmc_addr = get_laddr("_dmc"))==-1){
```

```
                printf("DMC pointer is not present in dsp code\n");
                printf(" Or error while getting the address of DMC pointer\n");
                exit(1);
               }
if ((port_num_addr = get_laddr("_port_num"))==-1){
                printf("PORT_NUM flag is not present in dsp code\n");
                printf(" Or error while getting the address of PORT_NUM flag\n");
                exit(1);
               }
if ((buf0_full_addr = get_laddr("_buf0_full"))==-1){
                printf("BUF0_FULL flag is not present in dsp code\n");
                printf("Or error while getting the address of BUF0_FULL flag\n");
                exit(1);
               }
if ((buf1_full_addr = get_laddr("_buf1_full"))==-1){
                printf("BUF1_FULL flag is not present in dsp code\n");
                printf(" Or error while getting the address of BUF1_FULL flag\n");
                exit(1);
               }
if ((buffer0_addr = get_laddr("_buffer0"))==-1){
                printf("BUFFER0 array is not present in dsp code\n");
                printf(" Or error while getting the address of BUFFER0 array\n");
                exit(1);
               }
if ((buffer1_addr = get_laddr("_buffer1"))==-1){
                printf("BUFFER1 array is not present in dsp code\n");
                printf(" Or error while getting the address of BUFFER1 array\n");
                exit(1);
               }
if ((rx_cmc1_addr = get_laddr("_rx_cmc1"))==-1){
                printf("RX_CMC1 is not present in dsp code\n");
                printf(" Or error while getting the address of RX_CMC1\n");
                exit(1);
               }

               ki1_addr = get_laddr("_ki1");              /* Get control variables */

               ki2_addr = get_laddr("_ki2");              /* Address

               y11_ref_addr = get_laddr("_y11_ref");

               y12_ref_addr = get_laddr("_y12_ref");

               port_num = 0;
               dsp_dl_int_array(port_num_addr,1,&port_num);
               cmc0 = 0xe0140000;
```

```
cmc1 = 0x0;
dmc = 0xc040c0f0;
newdmc = 0xc040c0f0;
dsp_dl_long_array(cmc0_addr,1,&cmc0);    /* Download cmc0 */
dsp_dl_long_array(cmc1_addr,1,&cmc1);    /* Download cmc1 */
dsp_dl_long_array(dmc_addr,1,&dmc);      /* Download dmc */
dsp_dl_long_array(newdmc_addr,1,&newdmc);/* Download newdmc */

/*** Set "start" semaphore ***/

dsp_dl_int_array(start_addr,1,&start);

/*** Initialize SAIB ***/

system("cls");     /* Clear screen */
printf("\n\n\n\n\n\n\n\n\n\n\n\n");              /* Display XXX Menu */
printf("\t\t\tInitializing SAIB.....\n\n");

  /* WAIT for SAIB program init */

dsp_up_int_array(init_complete_addr,1,&init_complete);
while((!init_complete) && (!kbhit()))
dsp_up_int_array(init_complete_addr,1,&init_complete);

do{
system("cls");
printf("\n\n\n\n\n\n\n\n\n\n");
printf("\t\t\tSystem is Running....\n\n");
printf("\t\t\tData is being Send to Files\n\n");
printf("\t\t\tPlease Wait.....\n\n");
dsp_up_int_array(rx_cmc1_addr,1,&rx_cmc1);

record1_file = fopen("out.dat","wb");
record2_file = fopen("control.dat","wb");
dsp_up_int_array(buf0_full_addr,1,&buf0_full);
while (!buf0_full)    /* WAIT for Bfr0 not full */
dsp_up_int_array(buf0_full_addr,1,&buf0_full);
dsp_up_long_array(buffer0_addr,BUFFER_SIZE,buffer0);

write1_ret = fwrite((char *)buffer0,4,BUFFER_SIZE,record1_file);
dsp_up_int_array(buf1_full_addr,1,&buf1_full);
while (!buf1_full)       /* WAIT for Bfr1 full */
dsp_up_int_array(buf1_full_addr,1,&buf1_full);
dsp_up_long_array(buffer1_addr,BUFFER_SIZE,buffer1);
write2_ret = fwrite((char *)buffer1,4,BUFFER_SIZE,record2_file);
system("cls");
```

```
printf("\n\n\n\n\n\n\n\n\n");
printf("\t\t\tOutput Data is send to out.dat\n\n");
printf("\t\t\tControl Data is send to control.dat\n\n\n\n\n");
printf("\t\t\tPress Anykey to Continue.....\n\n\n");
getch();

system("cls");
printf("\n\n\n\n\n\n\n\n\n\n");
printf("\t\t\tPress 'M' to modify references and gains\n\n\n ");
printf("\t\t\tPress SPACEBAR  to continue.....\n\n\n");
ch = getch();
if (( ch=='M') || (ch =='m'))
{

system(" cls");
printf("\n\n\n\n\n\n");
printf("\t\t\tPlease Input New References and Gains\n\n\n");
printf("\t\t\tPlease enter gain1 \n");
scanf("%f",&k1);

ki1 = 100000*k1;
printf("\t\t\tk1 = %f\n\n",k1);
dsp_dl_int_array(ki1_addr,1,&ki1);
printf("\t\t\tPlease enter gain2 \n");
scanf("%f",&k2);

ki2 = 100000*k2;
printf("\t\t\t k2 = %f\n\n",k2);
dsp_dl_int_array(ki2_addr,1,&ki2);
printf("\t\t\tPlease enter y1_ref \n");
scanf("%f",&y1_ref);

y11_ref = 100*y1_ref;
printf("\t\t\ty1_ref = %f\n\n",y1_ref);
dsp_dl_int_array(y11_ref_addr,1,&y11_ref);
printf("\t\t\tPlease enter y2_ref\n ");
scanf("%f",&y2_ref);

y12_ref = 100*y2_ref;
printf("\t\t\ty2_ref = %f\n\n",y2_ref);
dsp_dl_int_array(y12_ref_addr,1,&y12_ref);
dmcflag = 1;       /* Set dmcflag = 1 to notify dsp to change
                                        conreol variables */
dsp_dl_int_array(dmcflag_addr,1,&dmcflag);
dmcflag = 0;
printf("\t\t\tPress any key to continue....\n\n");
```

```
                getch();
                 }
                else if (ch ==32)
                {
                break;
    }
}
                while(ch!=32);
                stop = 0;
                system("cls");
                printf("\n\n\n\n\n\n\n");
                printf("\t\tPress Any Key To Transfer data to matlab and Stop.");
                while(!kbhit());
                stop = 1;
                dsp_dl_int_array(stop_addr,1,&stop);/* "Stop" DSP program */

}
```

Program 2 Control.c in DSP

```
#include <c30.h>
#include <sp.h>
#include <cs4215.h>
#include <dram.h>

#define CalibCount     256
#define buffersize     4096
#define buffersizeM1   buffersize-1

#define GCRval   0x00BC0000;                    /* Value of GCR (applic. specific) */
#define ts2_val   0x3f3fc0f0;                   /* Value of tx_tempreg_two   */
SP      sp;                                     /* Serial Port */
SP     *pSP = &sp
unsigned int gcr_vals;                          /* Global Control Register */
unsigned int dcb = 0X04000000;                  /* *** */
unsigned long cmc0 = 0;                         /* Control-Mode Control-Word 0 */
unsigned long cmc1 = 0;                         /* Control-Mode Control-Word 1 */
unsigned long dmc =  0;                         /* Data-Mode Control-Word */
unsigned long newdmc = 0;                       /* Temporary "New" dmc */
unsigned long TxTempReg1=0X0;                   /* Temp Regs to TX Control or Data */
unsigned long TxTempReg2=0X0;                   /* Words for Control or Data Mode */
unsigned long RxTempReg1 =  0X0;                /* Temp Regs to RX Control or Data */
unsigned long RxTempReg2 =  0X0;                /* Words for Control or Data Mode */
unsigned long bufout = 0x0;
unsigned long bufcon = 0x0;
unsigned int start = 0;                         /* semaphore: Host says start */
unsigned int stop =  0;                         /* semaphore: Host says stop */
unsigned int buf0_full = 0;                     /* flag: 1 => buffer 0 full */
unsigned int buf1_full = 0;                     /* flag: 1 => buffer 1 full */
unsigned int init_complete =0;                  /* flag: 1 => initialization done */
unsigned int dmcflag = 0;                       /* flag: 1 => *** */
unsigned int i = 0;                             /* interrupt counter */
unsigned int rx_cmc1;                           /* SAIB chip number */
unsigned int port_num = 0;                      /* Serial port number to use (0/1) */
unsigned int delay_counter;
int    mode = 0;                                /* 0=> Data mode; 1=> Control mode */
int    m = 0,n=0;
unsigned int cont = 0;
unsigned int usd1 = 0, usd2 = 0;
unsigned int uxd1 = 0, uxd2 = 0;
int sd1 = 0, sd2 = 0,xd1 = 0,xd2 = 0;
int ix = 0;
unsigned long buffer0[buffersize];              /* buffer 0 */
```

```
unsigned long buffer1[buffersize];                    /* buffer 1 */
unsigned long ustemp1, ustemp2;

/* define some variables for the controller */

unsigned int usx1=0, usx2=0;

int S1=0, S2=0, sk1=0, sk2=0;
float y_10 =0., y_11 =0., y_12 =0., x_10 =0., x_11 =0., x_12 =0.;
float y_20 =0., y_21 =0., y_22 =0., x_20 =0., x_21 =0., x_22 =0.;

float V1_in = 0.,V2_in = 0.;
float e1   = 0., e2   = 0.;
float n1_1  = 0., n1_2  = 0.;
float n2_1  = 0., n2_2  = 0.;

float u1=0.,u2=0., ub1 = 0., ub2 = 0.;
float uf1 = 0., uf2 = 0.;                             /* Amplitude Reference */
float y1_ref=1., y2_ref=1.;
float fc1 = 2.5, fc2 = 2.5;                           /* Feedforward Constant */

float Y10=0,Y11=0,Y12=0,X10=0,X11=0;
float Y20=0,Y21=0,Y22=0,X20=0,X21=0;
float X12=1., X22=1.;

float p1_0 = 0,p1_1 = 0,p1_2 = 0,phase_10 = 0,phase_11 = 0,phase_12 = 0;
float ep1 = 0,np1_1 = 0,np1_2 = 0,upb1 = 0,upf1 = 0.,up1 = 0.;
float sh_10 = 0, sh_11 = 0, sh_12 = 0;
float phase_ref = 1.0;                                /* Phase Reference */

float a=0.,b=0.,kc1 = 0.;
float d0 = 0, d1 = 0, d2 = 0;
float w=0.;

int ki1=0,ki2=0;
float k1=0.025,k2=0.025;                              /* Control gains */

int y11_ref=0,y12_ref=0;
float kp = -0.025;                                    /* Phase Control Gain */
float kpf = 0.;
int kpi = 0;
int kpfi = 0;
int phasei_ref = 0;
```

```
/* 2. sampling rate 27.4khz,cutoff frequence 200hz */

loat a_11 = -1.9352,a_12 = 0.9373;
float b_10 = 0.0005, b_11 = 0.001, b_12 = 0.0005;

/* SINE WAVE FOR FREQUENCY = 8.36K, SAMPLING RATE 27.4K */

float A11 = -0.0867, A12 = 1., B10 = 0.4783, B11 = 0.9566, B12 = 0.4783;

main()
{
                waitst(0);                      /* Initialize S30 WAIT states */

                asm("   AND 0dfffh,ST");        /* Disable "global" interrupt */

  cmc0 = 0xe0140000;                            /* 27.4k */
  cmc1 = 0x0;                                   /* Set Control mode low word */
  dmc  = 0xc040c0f0;                            /* Set Data mode control word */
  newdmc = 0Xc040c0f0;                          /* Setup temp dmc reg for Host updates */

    while(!start);                              /* WAIT until Host sets start semaphore */
  saib_init(port_num);                          /* Init serial port 0/1 and SAIB */

  asm(" OR 10H,IE");                            /* Enable serial port 0 TX interrupt */
  asm(" OR 2000H,ST");                          /* Enable global interrupt */

                do_clt_data();                   /* Perform Record/Play functions for Host */

                                                /* Reset SAIB before completion */
                SET_SP_X_PCR (pSP, 0x60);/* Set serial port DX pin as output = 1 */
                                                /* (this resets the SAIB) */
                asm("   AND 0dfffh,ST");        /* Disable global interrupt */

                while(1);                       /* WAIT here for Host reset */
}

void
saib_init (unsigned int sp_no)
{
int i;
                                                /* Initialize C30 serial port */
                pSP = &sp;                      /* Setup pointer to serial port */
                INIT_SP (sp_no, pSP);           /* Function initializes serial port */

#ifdef SBUS                                     /* For SBUS only: Setup serial port's */
  int *spdirreg = (int *) 0x80a001;             /* direction register to configure */
```

```
    *spdirreg = 0;                          /* CLKX, CLKR, FSX, FSR as inputs */
#endif
                                            /* Reset SAIB */
                SET_SP_X_PCR (pSP, 0x60); /* Config ser-port DX pin as output. */
                                            /* with DX =1 (this resets the SAIB) */
                DELAY_140MS;                /* WAIT 140 ms for SAIB reset */
                                            /* Reset Serial Port */
                SET_SP_X_PCR (pSP, 0x0);/* Reset ser-port "condition" */
                *(pSP)->gcr &= 0xf3ffffff;  /* Reset ser-port with GCR */
                asm (" nop");               /* WAIT three cycles for reset */
                asm (" nop");
                asm (" nop");

                SET_SP_X_PCR (pSP, 0x111);          /* Set serial port mode for port pins */
                SET_SP_R_PCR (pSP, 0x111);

                gcr_vals = GCRval;          /* Setup GCR */
                SET_SP_GCR (pSP, gcr_vals);
                                            /* Initialize SAIB */
                *(pSP)->gcr |= 0x0C000000; /* "Unreset" serial port */
                TxTempReg1 = 0x0000000;  /* Setup TX data and control words */
                TxTempReg2 = ts2_val;
                                            /* (SAIB powers up in Data mode) */
                SEND_DATA (pSP, TxTempReg1);  /* Send data word to TX */
                                            /* (SAIB now in control mode) */
                                            /* Send Control word */
                while(!XRDY(pSP));          /* WAIT for TX ready */
                SEND_DATA (pSP, TxTempReg2);  /* Control word to TX */
                                            /* Write control words until read dcb=0 */
                TxTempReg1 = cmc0;          /* Setup TX data and control words */
                TxTempReg2 = cmc1;
                dcb = 0x04000000;           /* Setup dcb to check RX'd DCB */

                do{                         /* UNTIL (read DCB=0) */
                        while (!XRDY(pSP));                 /* WAIT for TX ready */
                        SEND_DATA (pSP, TxTempReg1);        /* Send control word */
                        RxTempReg1 = RECEIVE_DATA(pSP); /* Receive word */
                        dcb &= RxTempReg1;          /* *** */
                        while(!XRDY(pSP));          /* WAIT for TX ready */
                        SEND_DATA (pSP, TxTempReg2);        /* Send control word */
                        RxTempReg2 = RECEIVE_DATA(pSP); /* Receive word */
                }
                while(dcb != 0);            /* (test for do-while loop) */
                                            /* Send DCB=1 & send control words again */
                cmc0 |= 0x04000000;         /* Setup DCB=1 */
                TxTempReg1 = cmc0;
```

```
                while (!XRDY(pSP));          /* WAIT for TX Ready */
                SEND_DATA (pSP, TxTempReg1);  /* Send *** to SAIB */
                while (!XRDY(pSP));          /* WAIT for TX ready */
                SEND_DATA (pSP, TxTempReg2);  /* Send *** to SAIB */
                                              /* Send D/C*=1 (TS6) for data mode */
                cmc1 |= 0x00400000;          /* Setup D/C*=1 (TS6) */
                TxTempReg2 = cmc1;
                while (!XRDY(pSP));          /* WAIT for TX Ready */
                SEND_DATA (pSP, TxTempReg1);  /* Send *** to SAIB */
                while (!XRDY(pSP));          /* WAIT for TX ready */
                SEND_DATA (pSP, TxTempReg2);  /* Send *** to SAIB */

                TxTempReg1 = 0x0;                       /* Setup desired configuration */
                TxTempReg2 = dmc;

                for (i=0; i<CalibCount; i++) {          /* FOR (256 times) */
                        while (!XRDY(pSP)) ;            /* WAIT for TX ready */
                        if (!mode) {                    /* IF (mode 0) */
                SEND_DATA(pSP, TxTempReg1);             /* Send control word */
                TxTempReg1 = RECEIVE_DATA(pSP);         /* Receive control */
                                        mode++;         /* Incr to mode 1 */
                        }
                        else {                          /* ELSE ( mode 1 ) */
                 SEND_DATA(pSP, TxTempReg2);            /* Send control word */
                RxTempReg1 = RECEIVE_DATA(pSP);         /* Receive control */
                                        mode--;         /* Back to mode 0 */
                        }
                }

                init_complete = 1;              /* Note initialization done */

                while (!RRDY(pSP));             /* Clear RX port and WAIT for Host int */
}

void do_clt_data(void){

                do{                             /* UNTIL (Host semaphore says stop) */
                cont = 0;                                       /* Set counter = 0 */
                p1_2 = V1_in*V2_in;                             /* Multiplication of channel 1 */
                                                                /*and channel 2 */
                                                                /* Low pass filter */
                phase_12 = b_10*p1_2 + b_11*p1_1 + b_12*p1_0
                                        - a_11*phase_11 - a_12*phase_10;
                phase_10 = phase_11;
                phase_11 = phase_12;
                p1_0 = p1_1;
```

```
p1_1 = p1_2;
                                        /* Form error of the phase angle */
ep1 = phase_ref - phase_12*0.8352825;
np1_1 = np1_1 + ep1;                    /* Integrating */
up1 = kp*np1_1 + upf1;

if(up1>1){                              /* Phase saturation Protecting */
up1=1;}
else if(up1<-1){
up1=-1.;}

sh_12 = Y12 + up1*d2;                   /* Phase shifting, where Y12 is*/
                                        /* the sin, d2 is the cos */

xd1 = phase_12*835.2825;                /* Numeric adjusting */
xd2 = 1000*phase_ref;
                                        /* Form control signal buffer used for
                                        /* data  transferong */
uxd1 = xd1;
uxd2 = xd2;
bufcon = ((uxd1<<16)>>16) | (uxd2<<16);/* Form 32 bit word */

sk1 = 14000*(sh_12-1);                  /* Amplitude saturation protecting */
if(sk1>32767){                          /* for Channel 1 */
sk1 = 32767;
}
else if (sk1<-32767){
sk1 = -32767;
}
usx1 = sk1;
ustemp1 = usx1;

sk2 =14000*(Y12-1);                     /* Amplitude saturation protecting */
if(sk2>32767){                          /* for Channel 2 */
sk2 = 32767;
}
else if(sk2<-32767){
sk2 = -32767;
}
usx2 = sk2;
ustemp2 = usx2;                         /* Form 64 bit word for output */

RxTempReg1 = ((ustemp1<<16)>>16) | (ustemp2<<16);
```

```
            if(ix<4096)                          /* Send data to buffer0 and buffer 1 */
            {                                    /* 80*4096 */
             if(n>80){
             buffer0[ix] = bufout;
             buffer1[ix] = bufcon;
             ix++;
             n=0;}
             else {n++;}
             do{                                 /* Loop and wait for 80, any number */
                                                 /* is OK, all depends how long the data */
             TxTempReg1 = RxTempReg1;            /* intervals you want */

                        if(dmcflag){             /* If dmcflag = 1, change control */
                          dmcflag = 0;           /* gain and referenece values */
                          ix = 0;
                        k1=0.00001*ki1;
                        k2=0.00001*ki2;
                        y1_ref=0.01*y11_ref;
                        y2_ref=0.01*y12_ref;
                        }
            }
            while(!cont);                        /* Count = 1 means data in is ready */
            }
            else
            {
            buf0_full = 1;                       /* Buffer full, stop sending data but */
            buf1_full = 1;                       /* keep control function */

            do{
            TxTempReg1 = RxTempReg1;

              if(dmcflag){                       /* If dmcflag = 1, change control gain */
                dmcflag = 0;
                ix = 0;
              k1=0.00001*ki1;
              k2=0.00001*ki2;
              y1_ref=0.01*y11_ref;
              y2_ref=0.01*y12_ref;
              }
}
while(!cont);
}
            }                                    /* If stop = 1, end control */
            while(!stop);
            asm(" AND   0FFEFh,IE");             /* Disable port 0 TX interrupt */
            asm(" AND   0DFFFh,ST");             /* Disable global interrupt */
```

```
                SET_SP_X_PCR (pSP, 0x0);        /* Reset serial port "condition" */
                SET_SP_R_PCR (pSP, 0x0);        /* *** */
                DELAY_140MS;                    /* Delay for 140 ms. for SAIB reset */
                *(pSP)->gcr &= 0xf3ffffff;      /* Reset the serial port */
                asm (" nop");                   /* WAIT three cycles for port reset */
                asm (" nop");
                asm (" nop");

}


void c_int05(){

                if(!mode){                      /* IF (mode is zero) */
                SEND_DATA (pSP, TxTempReg1);        /* send data received last */
                RxTempReg1 = RECEIVE_DATA(pSP);     /* receive data */
                cont = 1;                           /* Set count = 1 */
                usx1 = ((RxTempReg1<<16)>>16);      /* Get 2 cahnnels data */
                usx2 = (RxTempReg1>>16);

/*      m++;                                        /* This is for change reference */
                if(m<50*4096)                       /* automatically */
                {
                if(m<50*2048){
                y1_ref = 1.;
                y2_ref = 0.5;
                fc1 = 0.;
                fc2 = 0.;
                }
                else
                {
                y1_ref = 0.5;
                y2_ref = 1.;
                fc1 = 0;
                fc2 = 0.;
                }
                }
                else m=0; */                        /* Change data to floating point */
                if(usx1>32678)
                {
                sk1 = (usx1 - 65535);
                }
                else
                {
                sk1 = usx1;
```

```
}
V1_in = sk1*0.00021;                    /* Data adjusting */

V_12 = V1_in*V1_in;                     /* Square law demodulation */

if(usx2>32768)
{
sk2 = (usx2 - 65535);
}
else
{
sk2 = usx2;
}
V2_in = sk2*0.00021;
x_22 = V2_in*V2_in;                     /* Square law demodulation */

/***************   Low Pass Filter and Sinewave Generation   ***************/

y_12 = b_10*x_12 + b_11*x_11 + b_12*x_10 - a_11*y_11 - a_12*y_10;
y_10 = y_11;
y_11 = y_12;
x_10 = x_11;
x_11 = x_12;

sd1 = V1_in;
usd1 = sd1;

e1 = y1_ref - y_12*2;           /* Form error */
n1_1 = n1_1 + e1;               /* Integrating and feedforwarrd control */
u1 = k1*n1_1 + fc1;

y_22 = b_10*x_22 + b_11*x_21 + b_12*x_20 - a_11*y_21 - a_12*y_20;
y_20 = y_21;
y_21 = y_22;
x_20 = x_21;
x_21 = x_22;

sd2 = V2_in;
usd2 = sd2;

e2 = y2_ref - y_22*2;
n2_1 = n2_1 + e2;
u2 = k2*n2_1 + fc2;

xd1 = u1;
xd2 = u2;
```

```
                uxd1 = xd1;
                uxd2 = xd2;

                d2 = 1 - A11*d1 - A12*d0;                /* Sinewave Generation */
                Y12 = B10*d2 + B11*d1 + B12*d0;
                d0 = d1;
                d1 = d2;

                bufout = ((usd1<<16)>>16) | (usd2<<16)/* Output Buffer */

                mode++;                                  /* flip the mode */

 } else{                                                 /* ELSE (mode is one) */
                SEND_DATA (pSP, TxTempReg2);             /* send data mode control word */
                RxTempReg2 = RECEIVE_DATA(pSP);          /* receive control word */
                mode--;                                  /* flip the mode */
                i++;
  }
}

void c_int06(){}
void c_int07(){}
```

Program 3 Read.c in PC - This program is used to transfer data from DSP to Matlab.

```
#include <stdio.h>
#define BUFFER_SIZE 4096
main()
{
   unsigned long buffer[BUFFER_SIZE];
   int i;
   char filename[8];
   FILE *outfile,*infile1,*infile2;
       printf("\n\t\tPlease enter the filename:");
       scanf("%s",filename);
       outfile = fopen(filename,"rb");
       fread((char *)buffer,4,BUFFER_SIZE,outfile);
       infile1 = fopen("data1.mat","w");
       infile2 = fopen("data2.mat","w");
       for(i=0;i<4096;i++)
       {
       fprintf(infile1,"%d\n",buffer[i]>>16);
       fprintf(infile2,"%d\n",((buffer[i]<<16)>>16));
       }
}
```

4. Simulation Program and Data File

```
#include "\ALSIM\ALSIM.H"

#define w_over_Q11     fpar[1]
#define w_over_Q12     fpar[2]
#define w_over_Q21     fpar[3]
#define w_over_Q22     fpar[4]
#define w              fpar[5]
#define w2             fpar[6]
#define root2_wf       fpar[7]
#define wf2            fpar[8]
#define amp_ref1        fpar[9]
#define K_I_amp1        fpar[10]
#define amp_ref2        fpar[11]
#define K_I_amp2        fpar[12]

/*
2-Loop Amplitude Control
*/

derv(t, x, dxdt)
double t, *x, *dxdt;
{

dxdt[1]=x[2];
dxdt[2]=-w_over_Q11*x[2]-w2*x[1]+ plotout[1];
dxdt[3]=x[4];
dxdt[4]=-w_over_Q12*x[4]-w2*x[3]+ plotout[2];

/* plotout[3] is the output of channel 1 */
plotout[3]= -8.3568e7*x[1]+280.5256*x[2] + 8.3549e7*x[3] +
280.4634*x[4];

/* demodulation and lowpass filter */
/* x[5] is amplitude info */
dxdt[5]=x[6];
dxdt[6]=-root2_wf*x[6]-wf2*(x[5]-plotout[3]*plotout[3]*0.02);

/* plotout[4] is the error */
plotout[4]=amp_ref1-x[5];

/* Integral */
dxdt[7]= 27427.58*K_I_amp1*plotout[4];

/* modulation */
plotout[1]=x[7]*sin(w*t);
```

Simulation Program Cont'

```
dxdt[8]=x[9];
dxdt[9]=-w_over_Q21*x[9]-w2*x[8]+ plotout[1];
dxdt[10]=x[11];
dxdt[11]=-w_over_Q22*x[11]-w2*x[10]+ plotout[2];

/* plotout[5] is the output of the channel 2 */
plotout[5]=6.0856e7*x[8]+204.2843*x[9] -7.7877e7*x[10] +
261.422*x[11];

/* demodulation and lowpass filter */
dxdt[12]=x[13];
dxdt[13]=-root2_wf*x[13]-wf2*(x[12]-plotout[5]*plotout[5]*0.02);

/* plotout[6] is the error */
plotout[6]=amp_ref2 - x[12];

/* integral */
dxdt[14]= 27427.58*K_I_amp2*plotout[6];

/* modulation */
plotout[2]=x[14]*sin(w*t);

}
```

Simulation Program Cont'

```
0.       ;initial time
1.       ;final time
1.e-5    ;maximum stepsize
1.0e-9   ;minimum stepsize
0.01     ;fractional error criterion

10000    ;multiple of maximum stepsize for print output
100      ;multiple of maximum stepsize for plot output

14       ;number of plant states
0        ;number of plant inputs
0        ;number of plant outputs
0        ;number of controller states

6        ;size of user defined plot vector
0        ;size of user common area
0        ;size of gaussian random number vector
        ;vector multiplied by sqrt(hmax) to provide approx. uniform
         ;variance for variable stepsize
318      ;random number seed
272      ;random number seed
190      ;random number seed
0        ;number of user defined integer input parameters
0,0      ;end integer input parameters
12          ; number of user defined floating point input parameters
1,1615.5       ; w_over_Q11
2,1794.6       ; w_over_Q12
3,1470.5       ; w_over_Q21
4,1344.2       ; w_over_Q22
5,52527.3837   ; w
6,2.759e9      ; w2
7,1777.14      ; root2_wf
8,1.58e6       ; wf2
9,1.           ; amp_ref1
10,0.025       ; K_I_amp1
11,1.          ; amp_ref2
12,0.025       ; K_I_amp2
0,0.           ; end floating point input parameters
1,0.
2,0.
3,0.
4,0.
0,0            ; end plant initial conditions

0,0            ; end controller initial conditions
```

Simulation Program Cont'

```
0.       ;initial time
2.       ;final time
1.e-5    ;maximum stepsize
1.0e-9   ;minimum stepsize
0.01     ;fractional error criterion

10000    ;multiple of maximum stepsize for print output
100      ;multiple of maximum stepsize for plot output

14       ;number of plant states
0        ;number of plant inputs
0        ;number of plant outputs
0        ;number of controller states

6        ;size of user defined plot vector
0        ;size of user common area
0        ;size of gaussian random number vector
        ;vector multiplied by sqrt(hmax) to provide approx. uniform
         ;variance for variable stepsize
318      ;random number seed
272      ;random number seed
190      ;random number seed
0        ;number of user defined integer input parameters
0,0      ;end integer input parameters
12          ; number of user defined floating point input parameters
1,1615.5       ; w_over_Q11
2,1794.6       ; w_over_Q12
3,1470.5       ; w_over_Q21
4,1344.2       ; w_over_Q22
5,52527.3837   ; w
6,2.759e9      ; w2
7,1777.14      ; root2_wf
8,1.58e6       ; wf2
9,1.           ; amp_ref1
10,0.0025      ; K_I_amp1
11,1.          ; amp_ref2
12,0.0025      ; K_I_amp2
0,0.           ; end floating point input parameters
1,0.           ; initial conditions of four states
2,0.
3,0.
4,0.
0,0            ; end plant initial conditions

0,0            ; end controller initial conditions
```

APPENDIX C

EXPERIMENTAL RESULTS - FIGURES

The experimental results for four different cases with gains equal 0.0025 and 0.025 are presented here for references. The four cases are 1) single loop amplitude control, 2) 2-loop amplitude control, 3) phase control and 4) 2-loop amplitude & phase control, respectively.

C.1 RESULTS FOR SINGLE LOOP AMPLITUDE CONTROL

Four figures are presented here for two different command trajectories with control gains $K_I = 0.0025$, and 0.025.

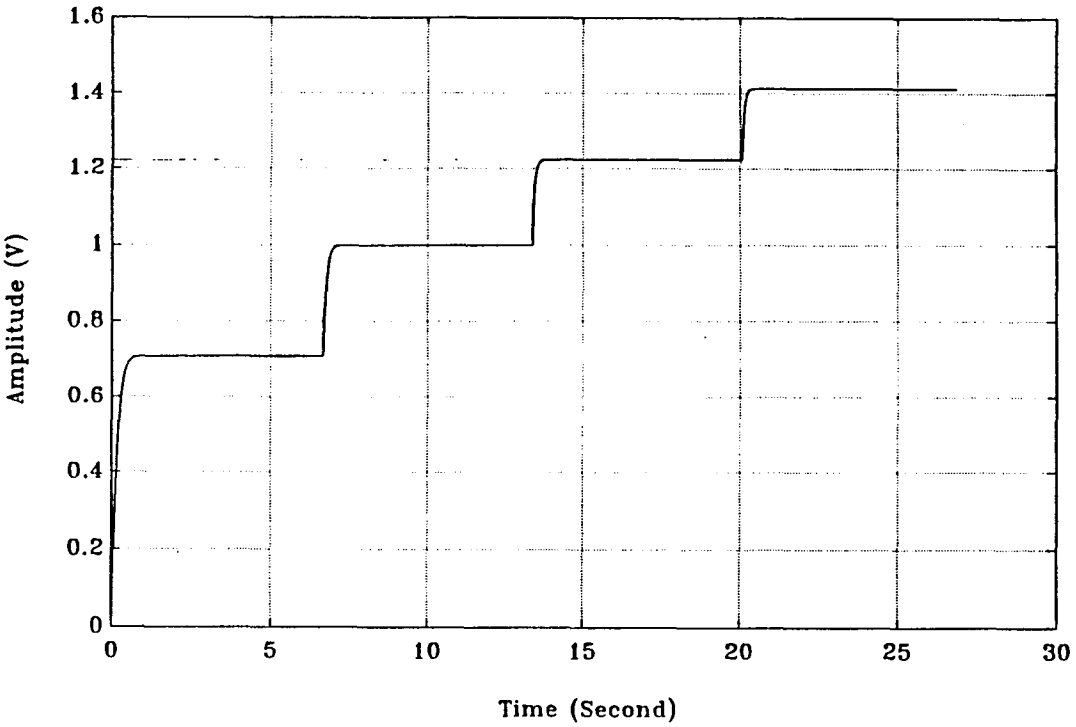


Figure C.1 Single Loop Amplitude Control for First Trajectory,K_I =0.0025

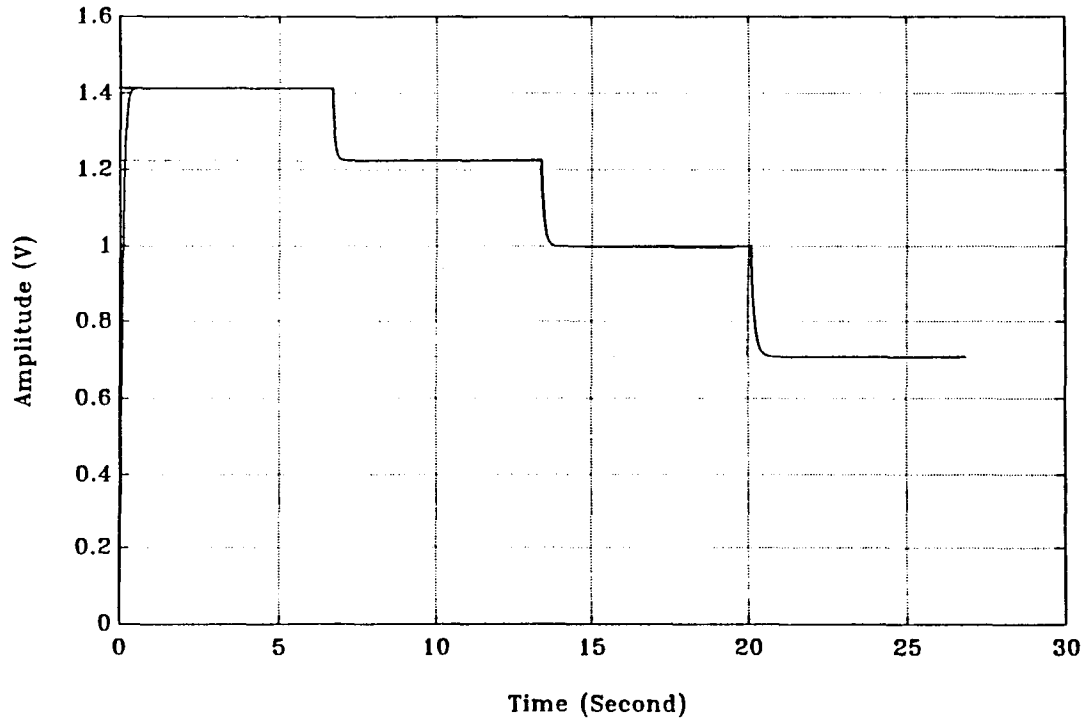


Figure C.2 Single Loop Amplitude Control for Second Trajectory, K_I=0.0025

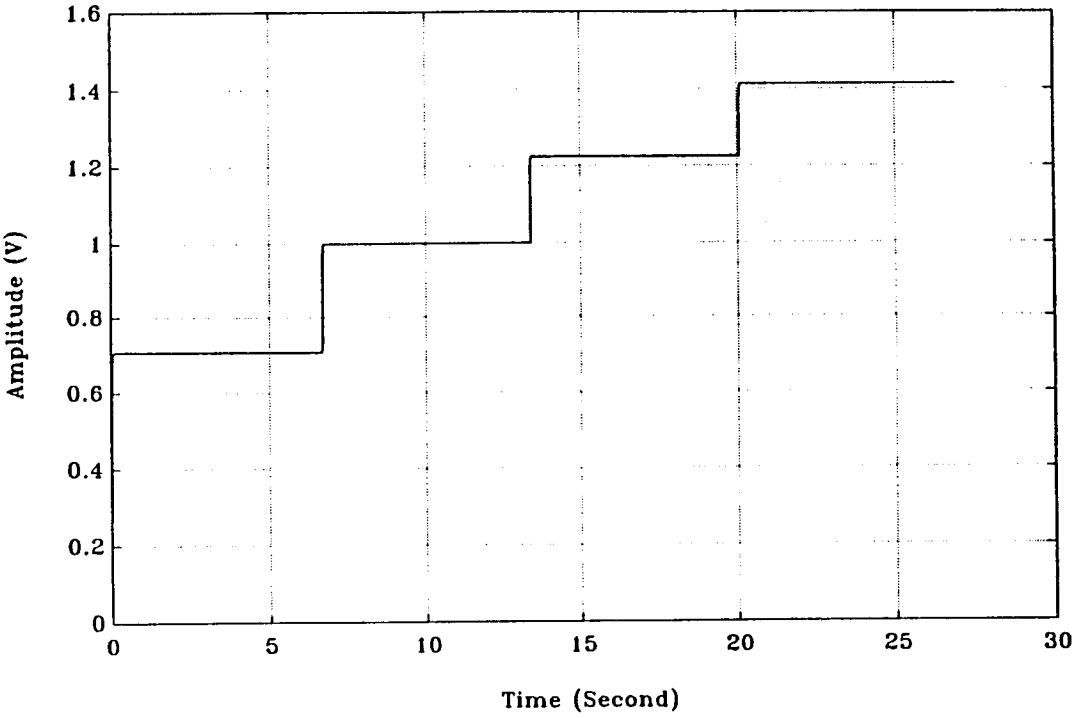


Figure C.3 Single Loop Amplitude Control for First Trajectory, K_I=0.025

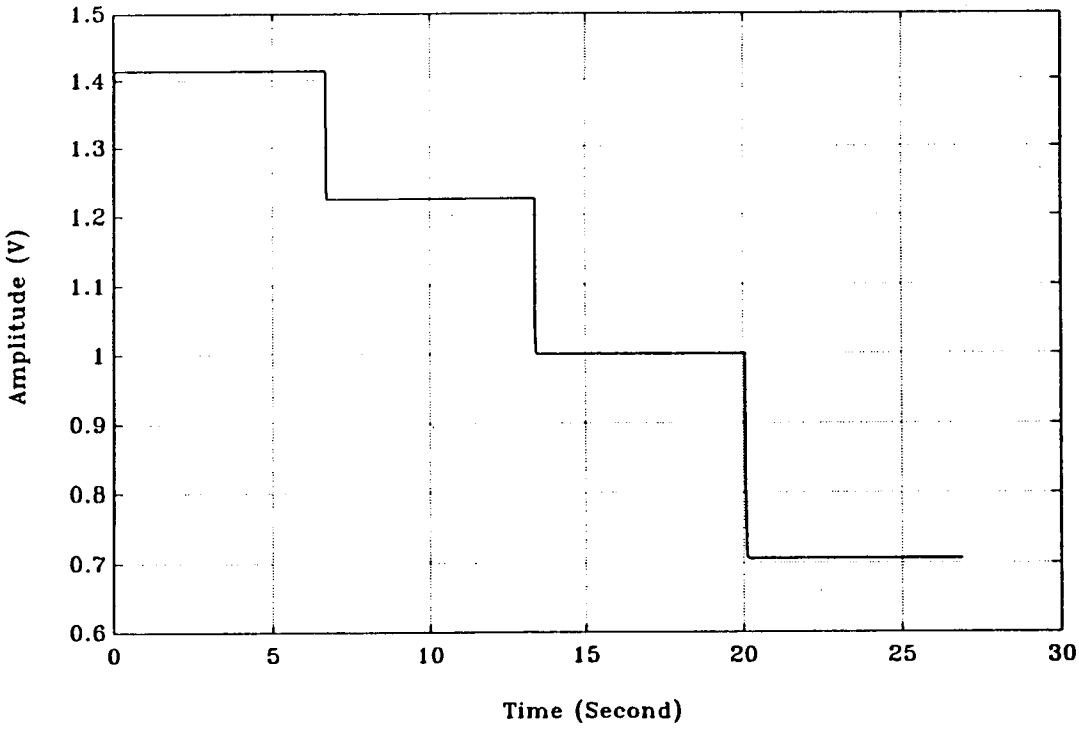


Figure C.4 Single Loop Amplitude Control for Second Trajectory, K_I=0.025

C.2 RESULTS FOR TWO LOOP AMPLITUDE CONTROL

Following figures are the results of 2-loop amplitude control with control gain(s) K_I equals 0.0025, 0.025, and 0.05 respectively, where figures with $K_I = 0.025$ are the "optimal"cases in this configuration, figures with gain(s) equals 0.05 are the unstable results. Two different command trajectories are used in this configuration.

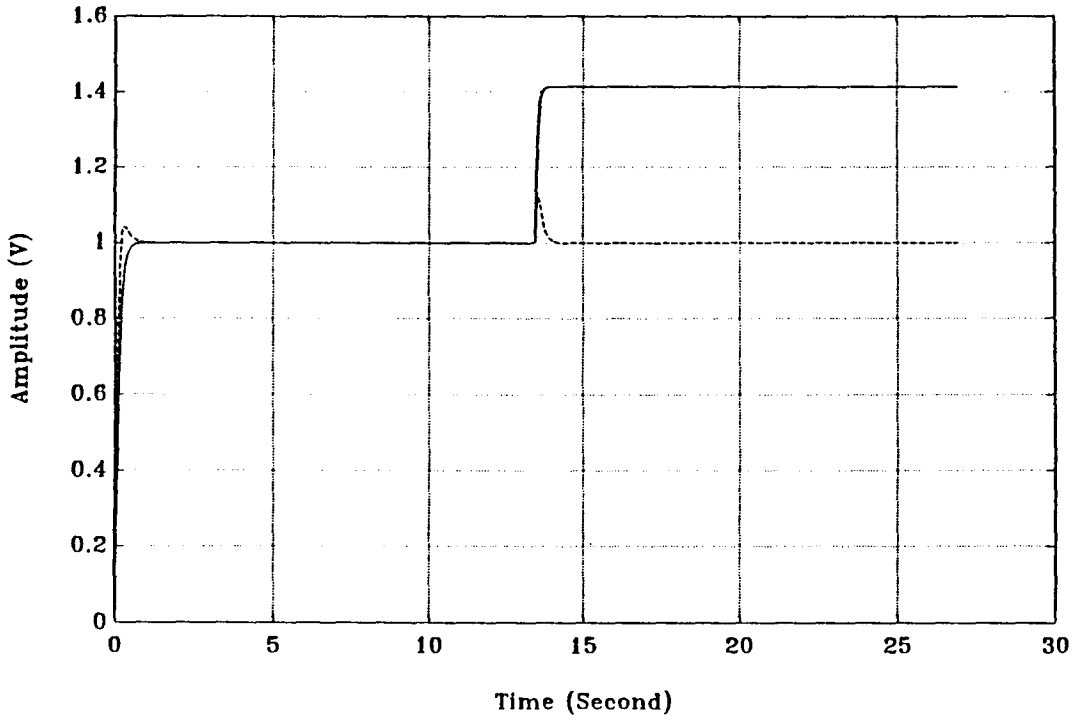


Figure C.5 2-Loop Amplitude Control for First Trajectory (1) with $K_I = 0.0025$

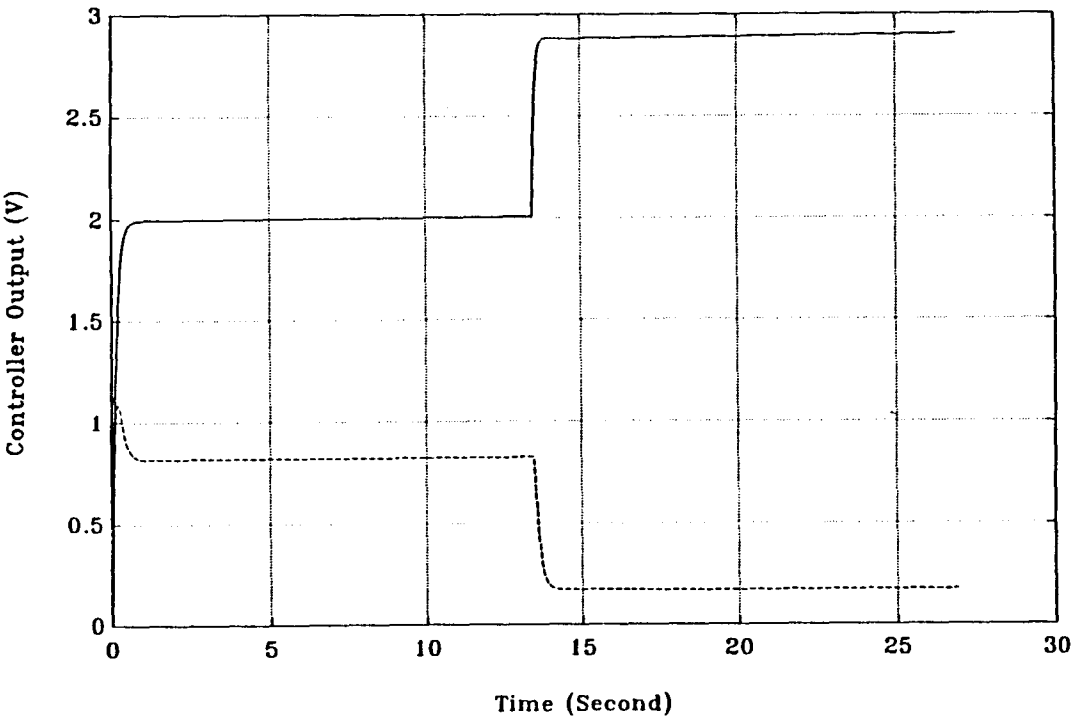


Figure C.6 2-Loop Amplitude Control for First Trajectory, Controller output

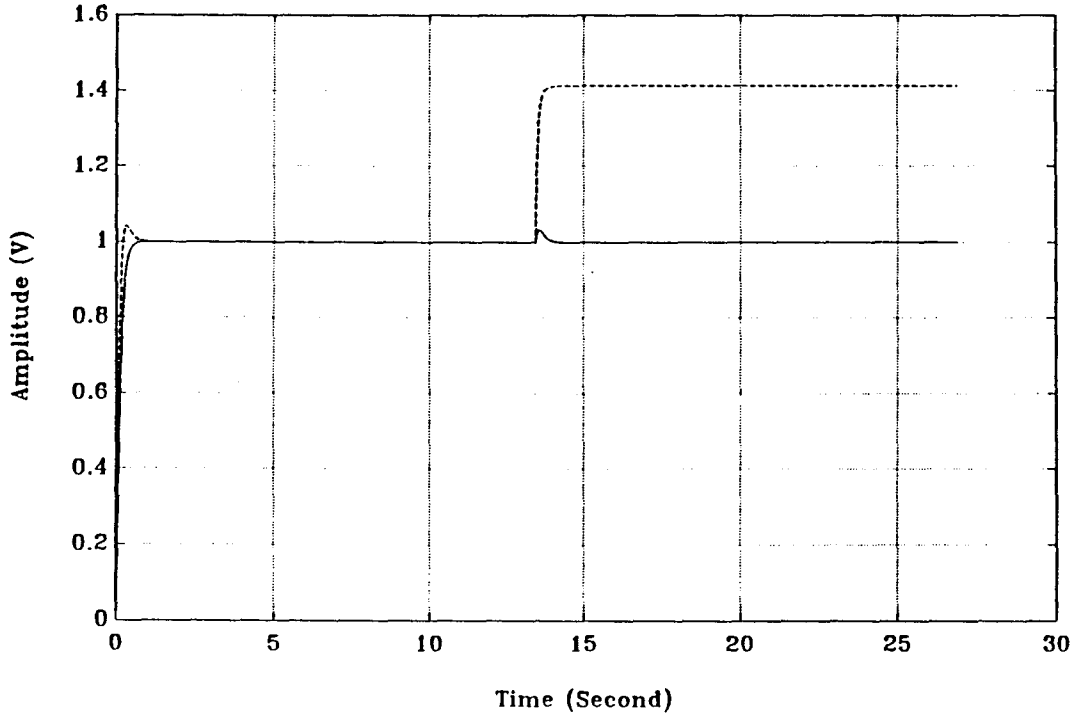


Figure C.7 2-Loop Amplitude Control for First Trajectory (2) with $K_I = 0.0025$

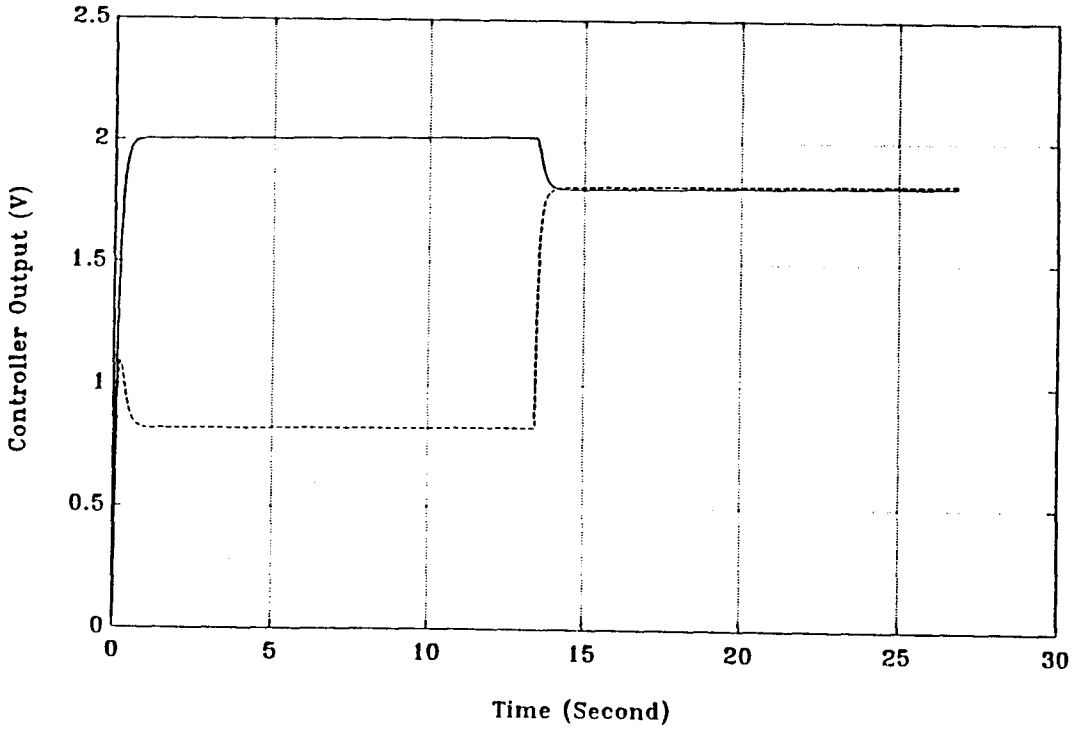


Figure C.8 2-Loop Amplitude Control for First Trajectory, Controller output

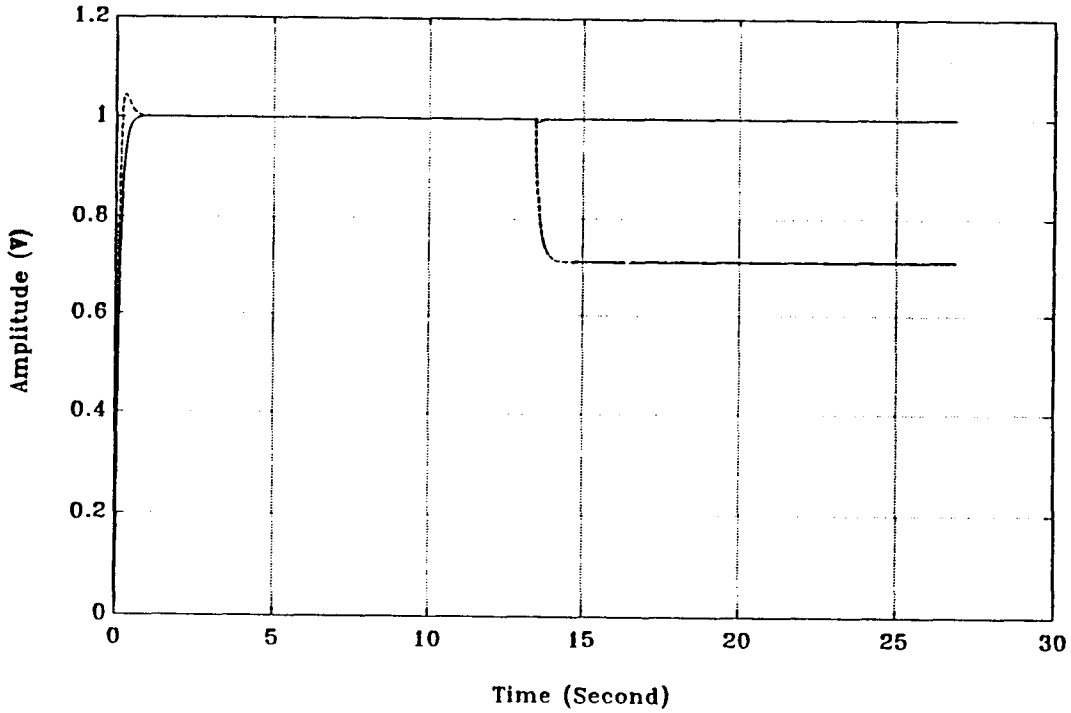


Figure C.9: 2-Loop Amplitude Control for Second Trajectory (1) with $K_I = 0.0025$

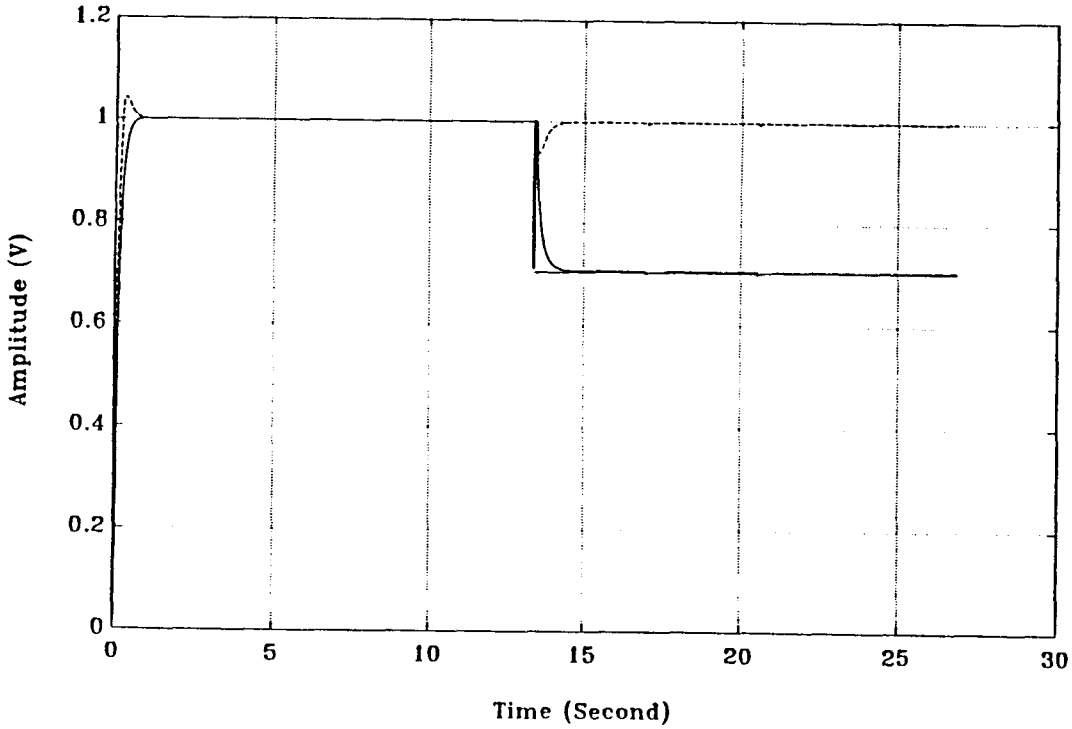


Figure C.10: 2-Loop Amplitude Control for Second Trajectory (2) with $K_I = 0.0025$

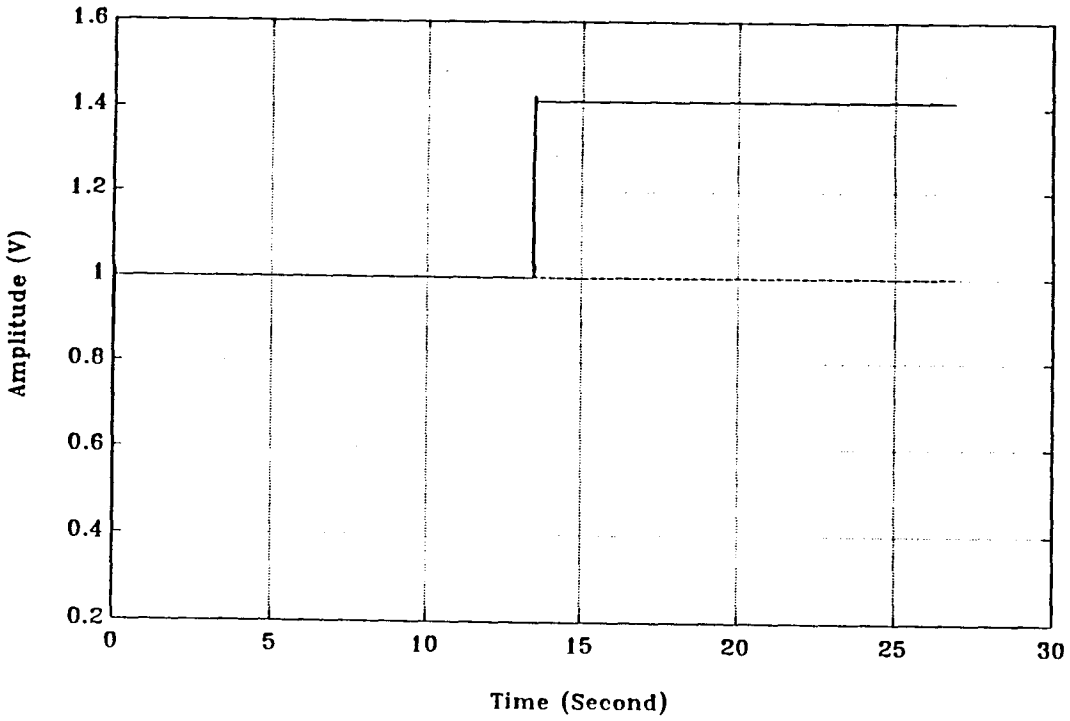


Figure C.11 2-Loop Amplitude Control for First Trajectory (1) with $K_I = 0.025$

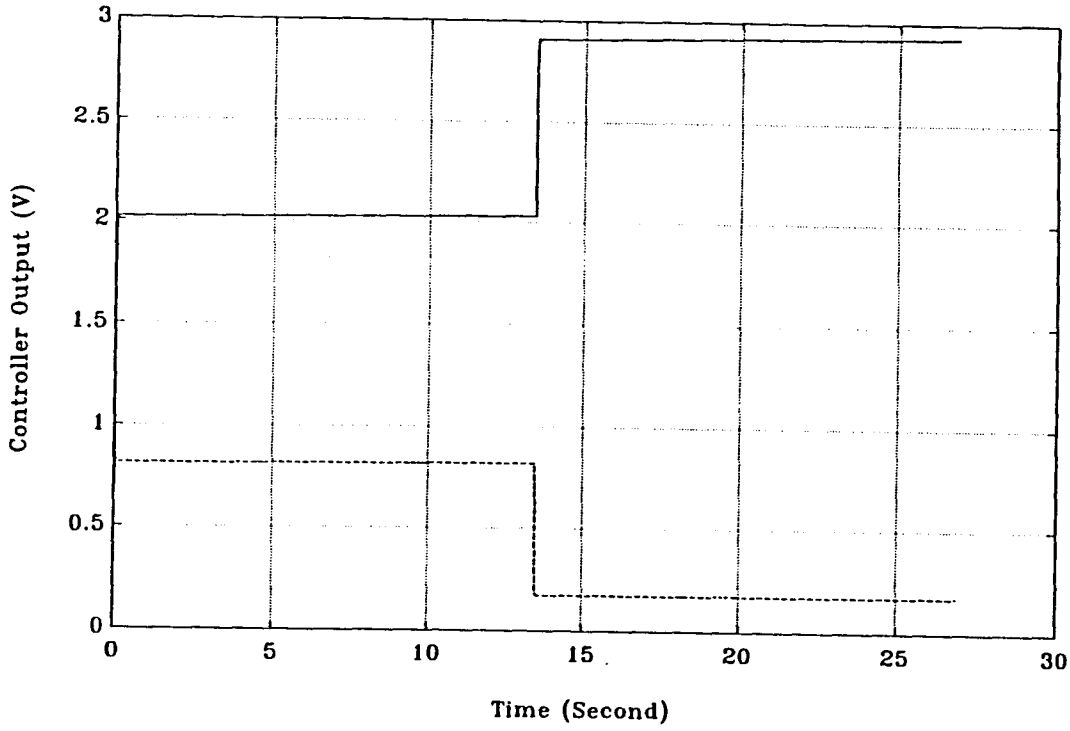


Figure C.12 2-Loop Amplitude Control for First Trajectory, Controller output

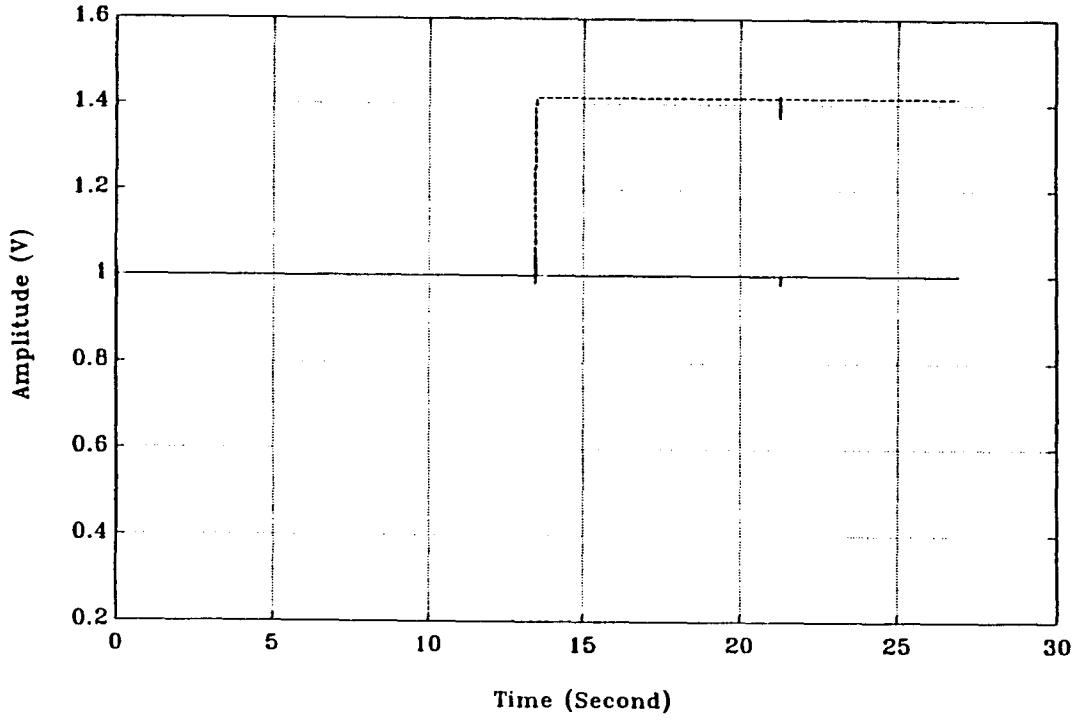


Figure C.13 2-Loop Amplitude Control for First Trajectory (2) with $K_I = 0.025$

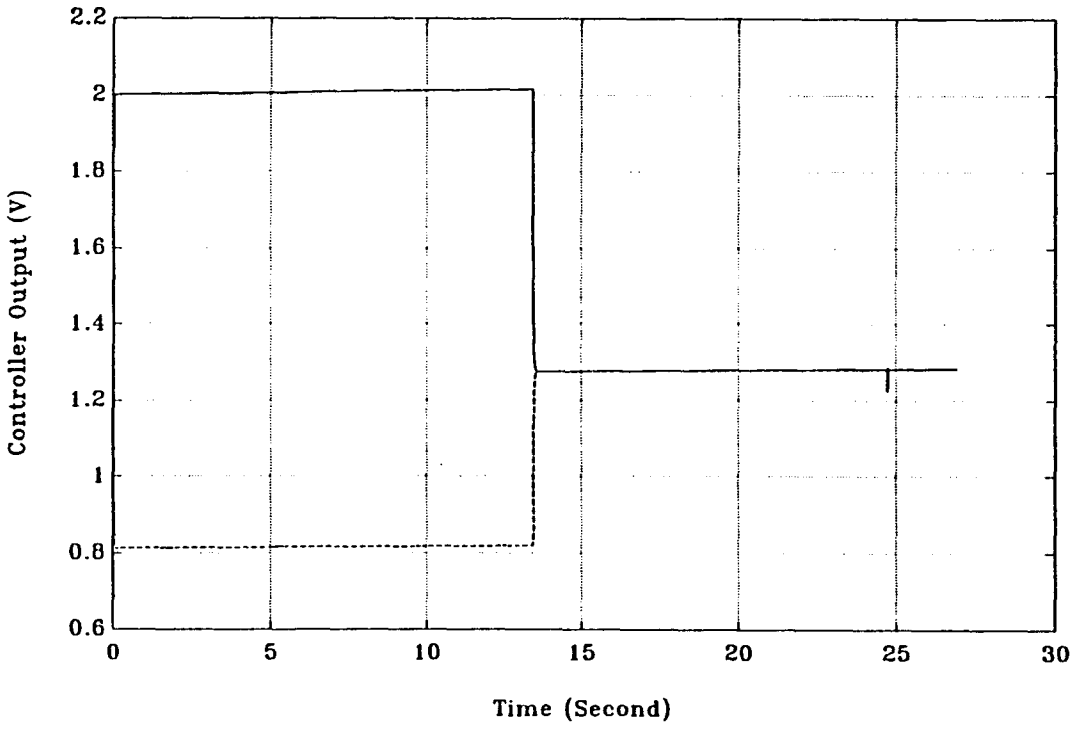


Figure C.14 2-Loop Amplitude Control for First Trajectory, Controller output

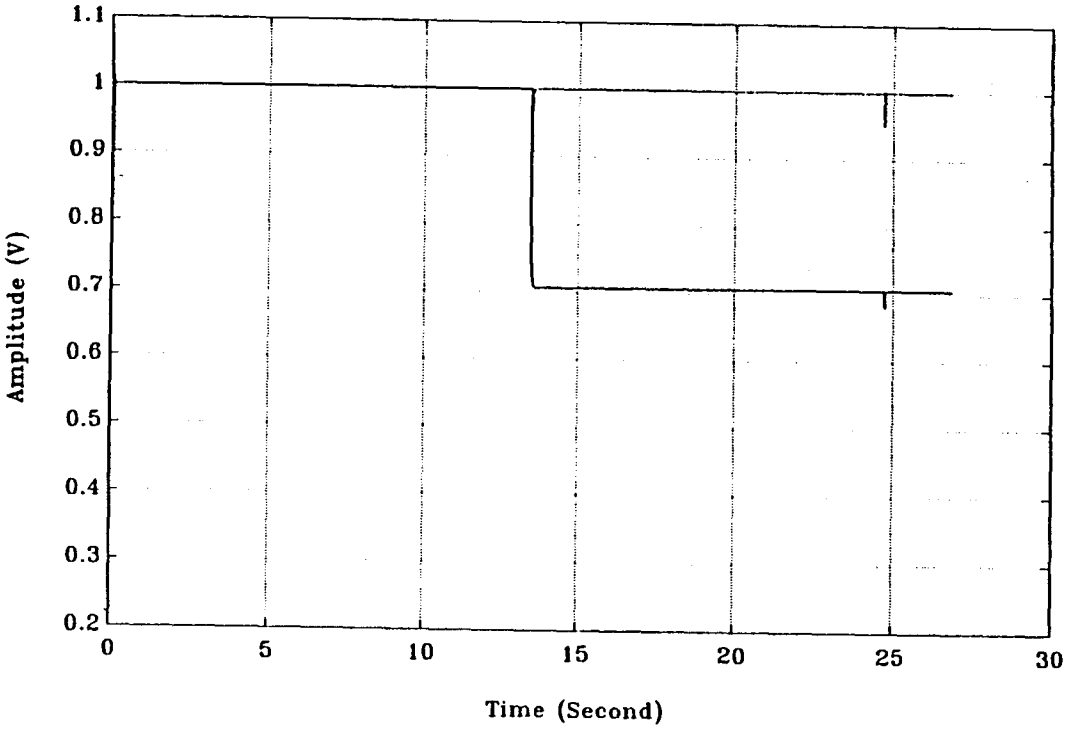


Figure C.15: 2-Loop Amplitude Control for Second Trajectory (1) with $K_I = 0.025$

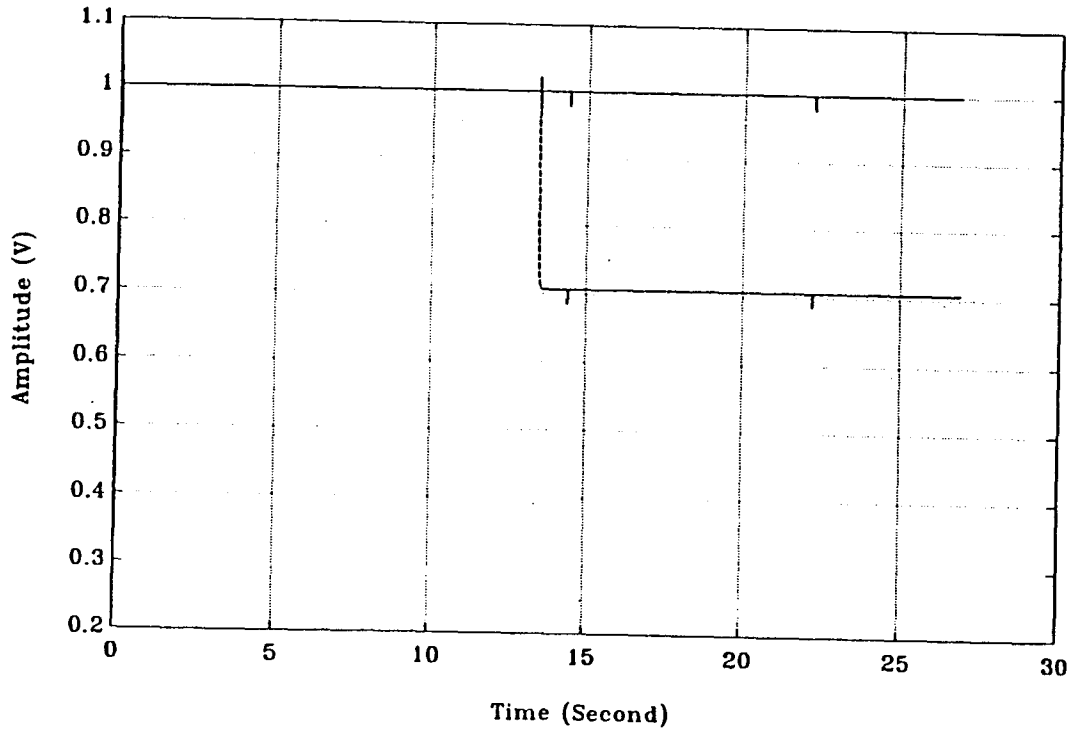


Figure C.16: 2-Loop Amplitude Control for Second Trajectory (2) with $K_I = 0.025$

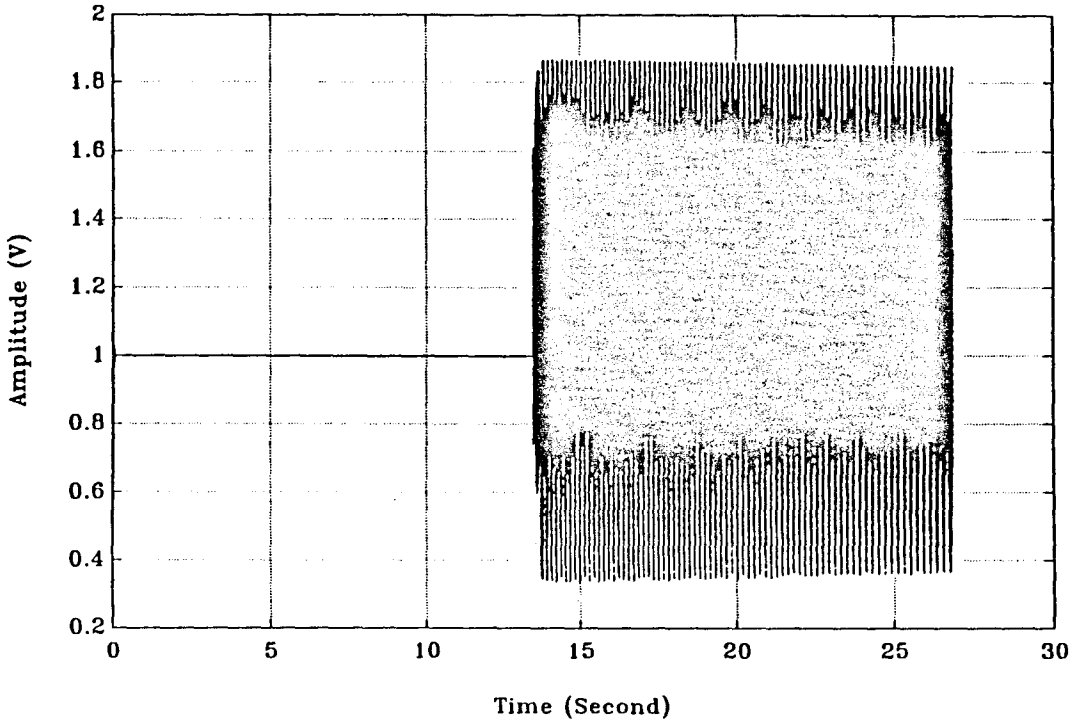


Figure C.17: 2-Loop Amplitude Control, Plant Output for the First Trajectory, $K_I = 0.05$

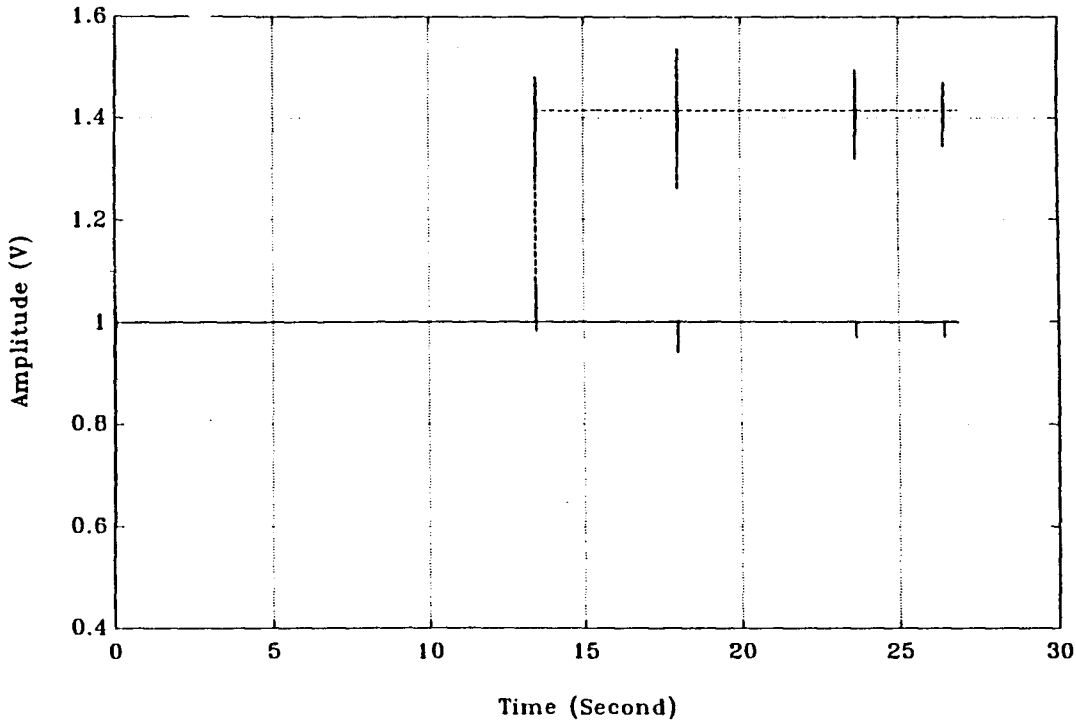


Figure C.18: 2-Loop Amplitude Control, Plant Output for the Second Trajectory $K_I = 0.05$

C.3 RESULTS FOR PHASE CONTROL

Four figures are shown in this section with control gain equals 0.0025 and 0.025. Two command trajectories are used for this experiment. The figures with gain equals 0.025 are the "optimal" cases for the phase control.

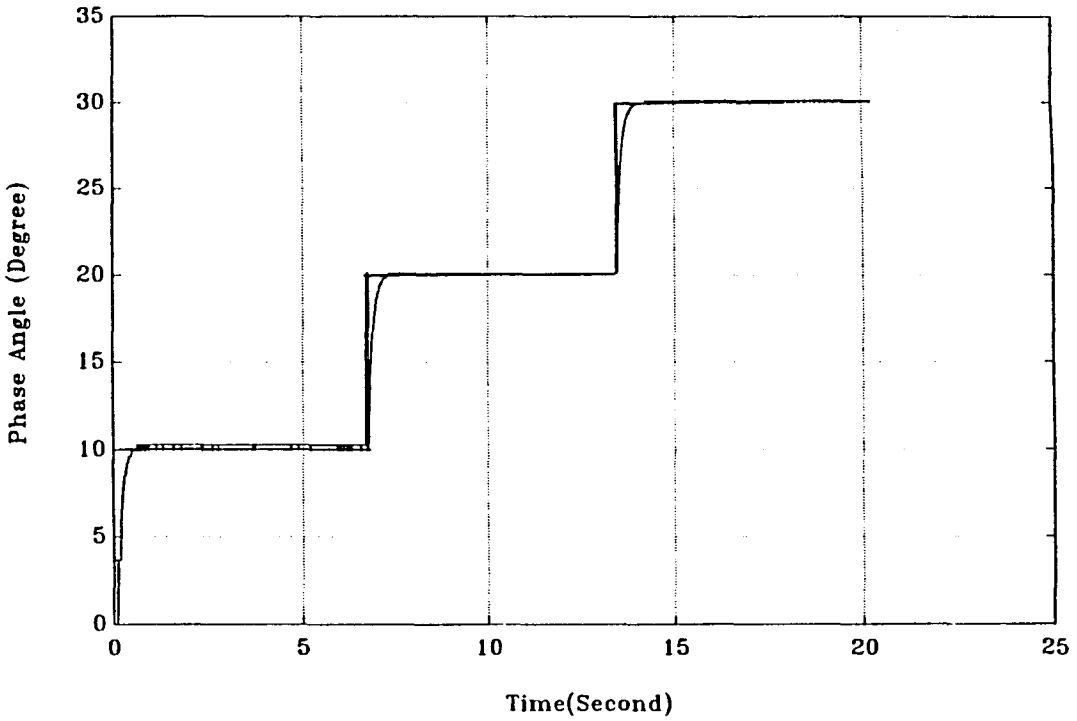


Figure C.19 Phase Control for First Trajectory with Gain $K_p = 0.0025$

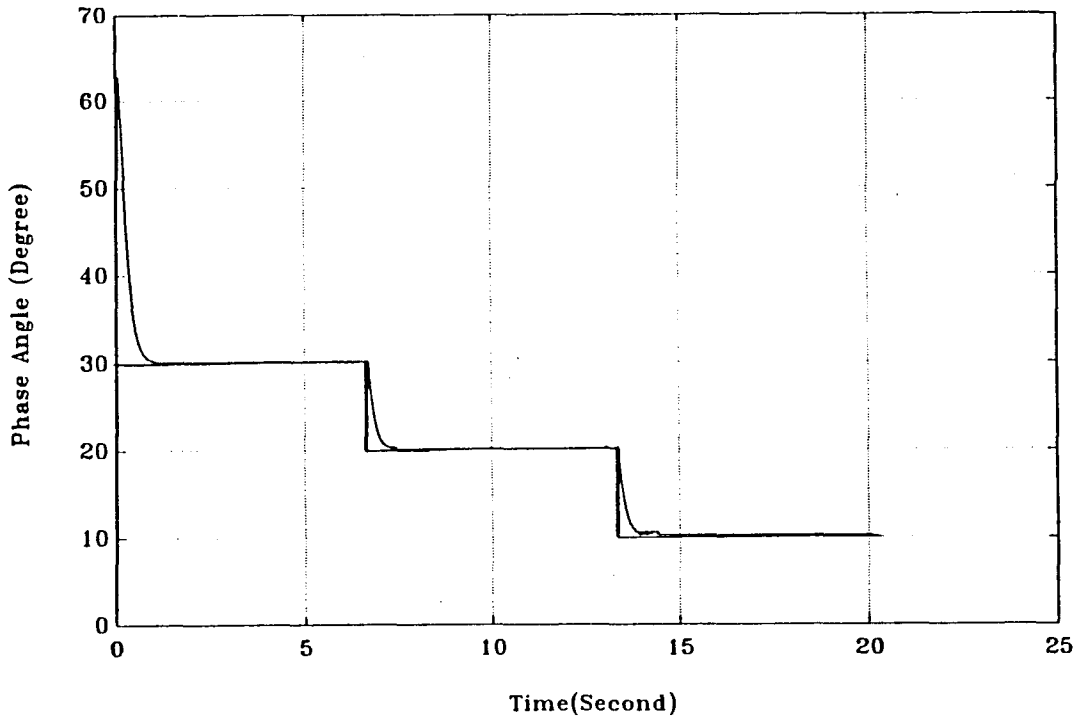


Figure C.20 Phase Control for Second Trajectory with Gain $K_p = 0.0025$

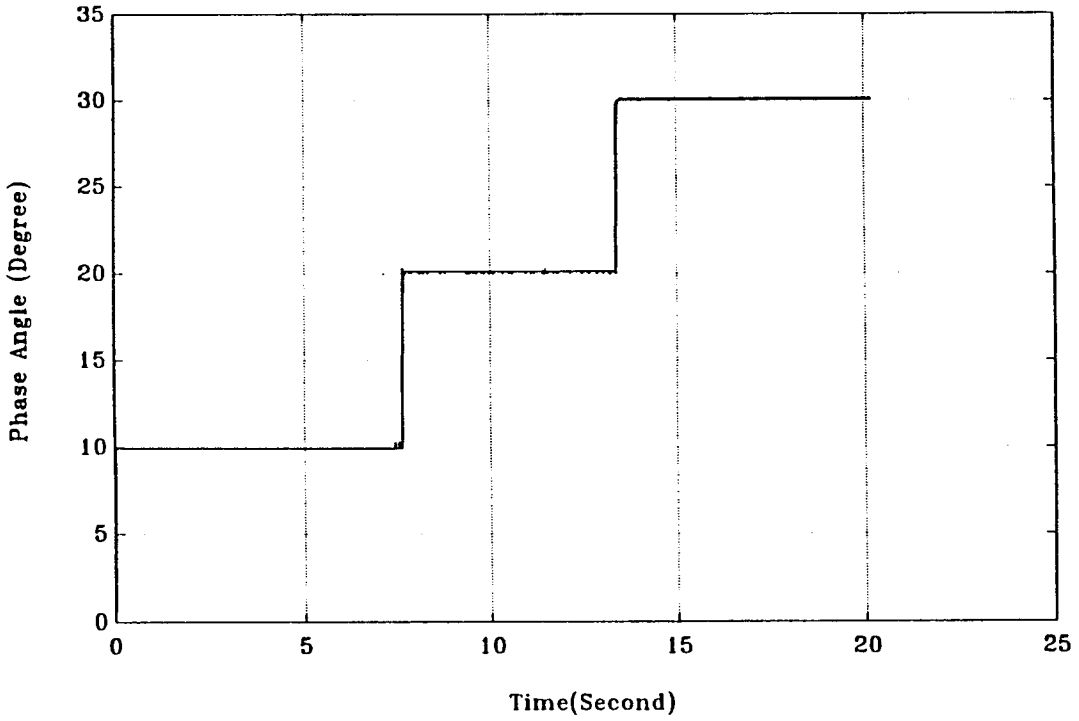


Figure C.21 Phase Control for First Trajectory with Gain $K_p = 0.025$

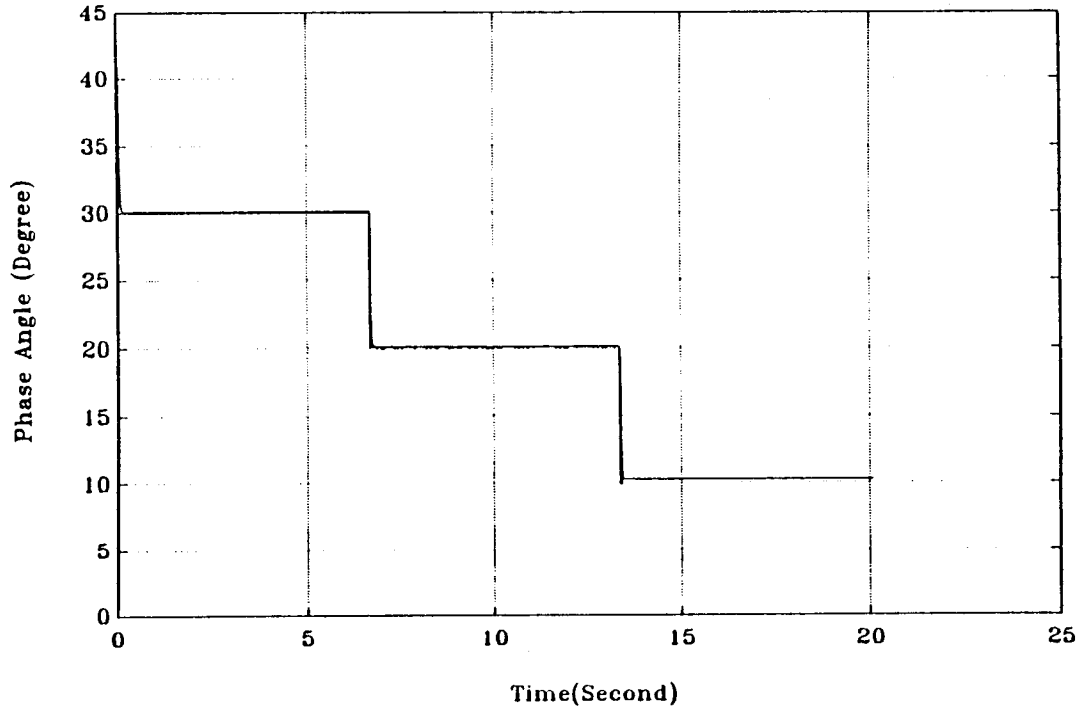


Figure C.22 Phase Control for Second Trajectory with Gain $K_p = 0.025$

C.4 RESULTS FOR TWO LOOP AMPLITUDE & PHASE CONTROL

Results for 2-loop amplitude & phase control are presented in this section. The figures with gain pair equal 0.025 and -0.025 are the "optimal" cases for this configuration.

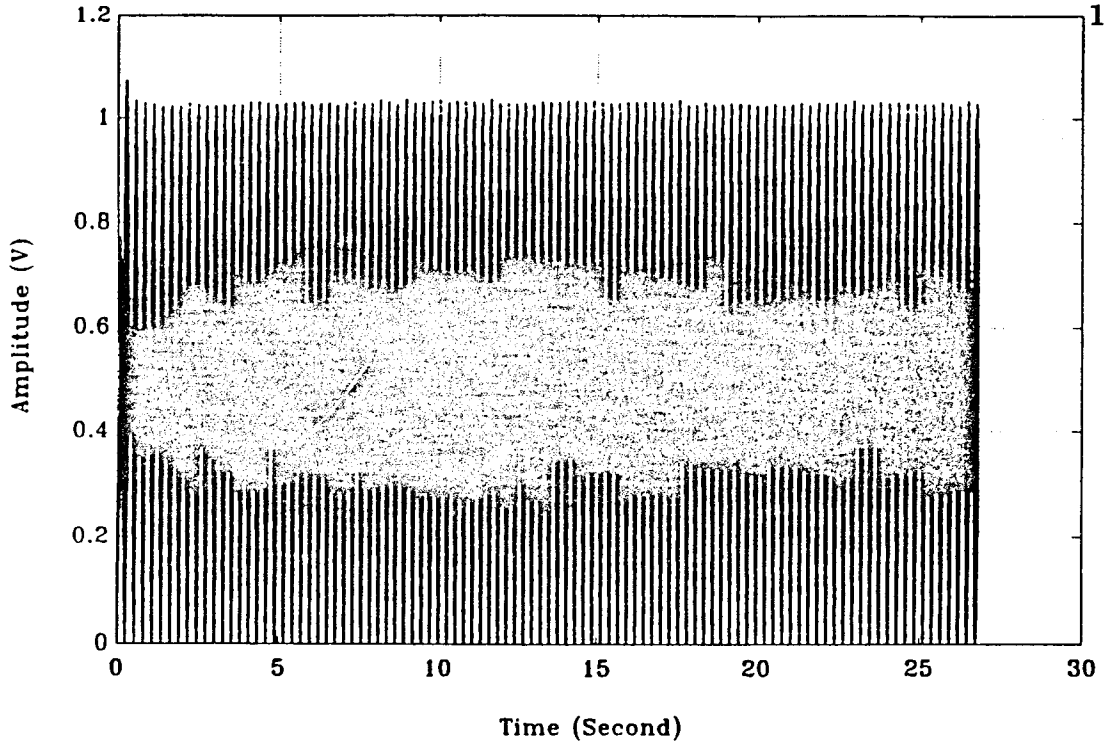


Figure C.23: 2-Loop Amplitude & Phase Control, Amplitude Output before Filtering

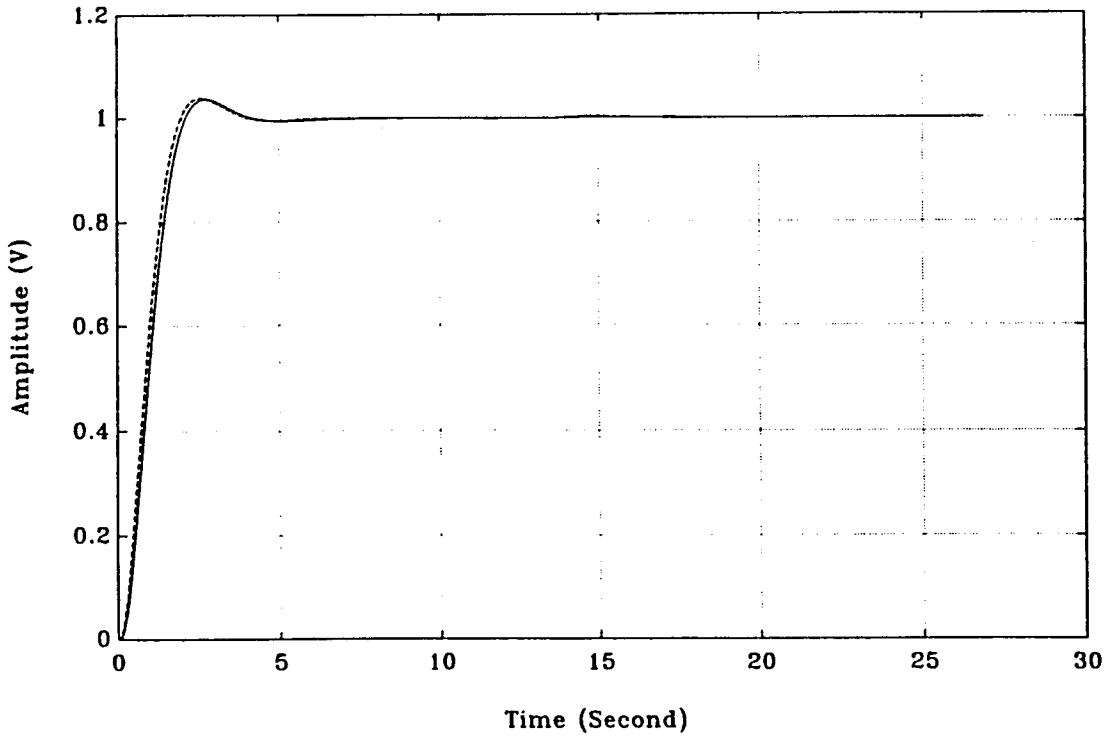


Figure C.24: 2-Loop Amplitude & Phase Control, Amplitude Output after Filtering

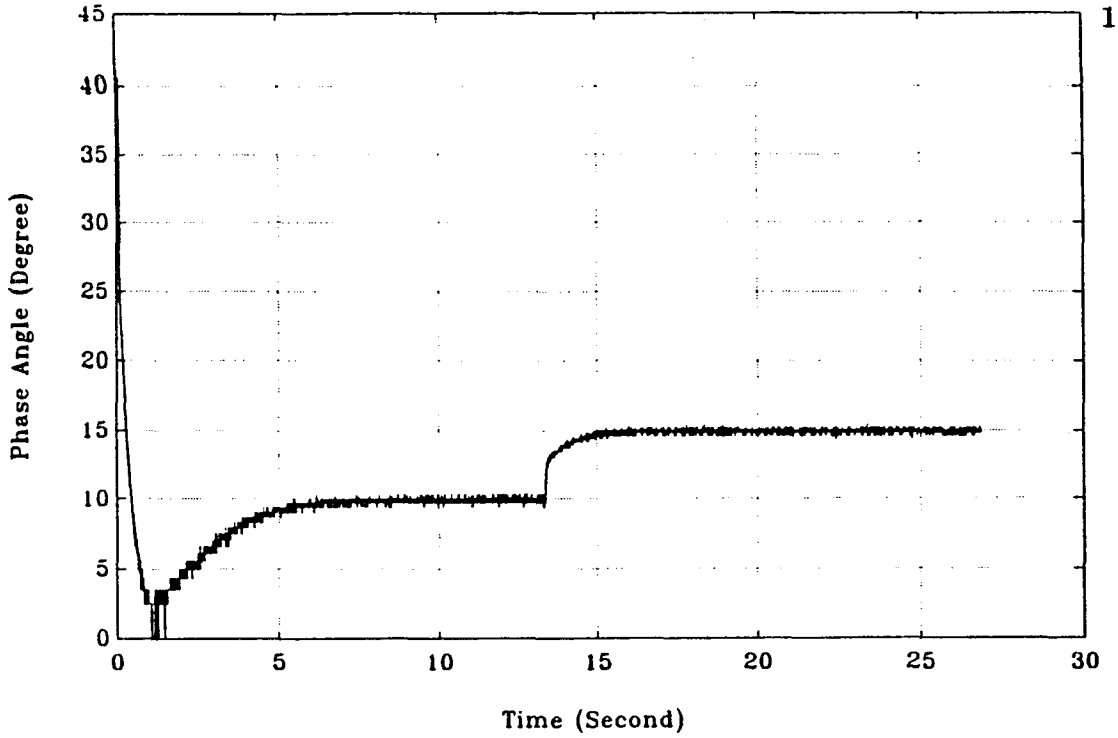


Figure C.25: 2-loop Amplitude & Phase Control, Phase Angle Between Two Outputs,with K_I=0.0025, K_P=-0.0025

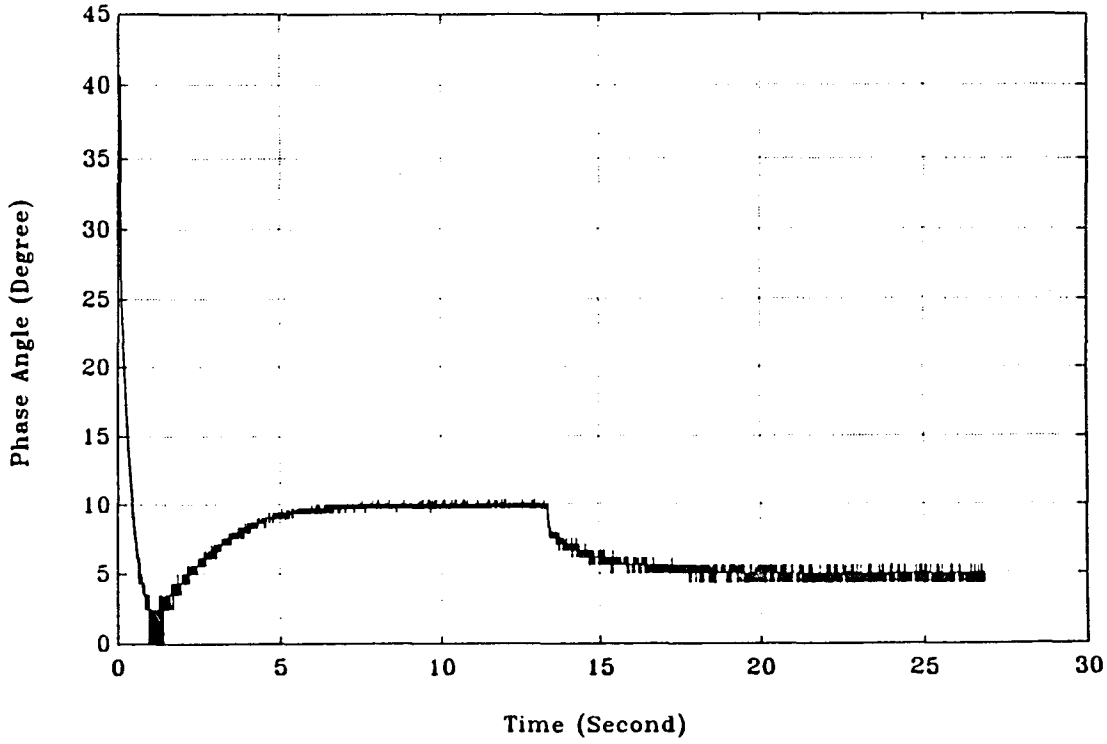


Figure C.26: 2-Loop Amplitude & Phase Control, Phase Angle Between Two Outputs,with K_I=0.0025, K_P=-0.0025

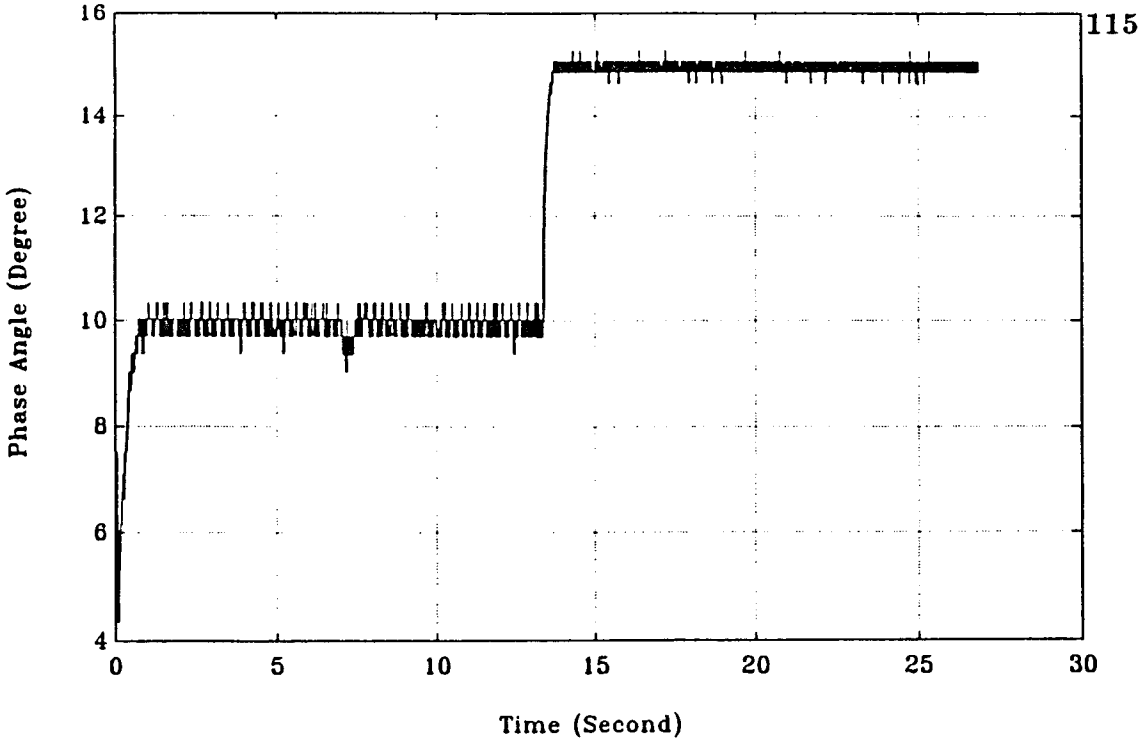


Figure C.27: 2-Loop Amplitude & Phase Control, Phase Angle Between Two Outputs,with K_I=0.025, K_P=-0.025

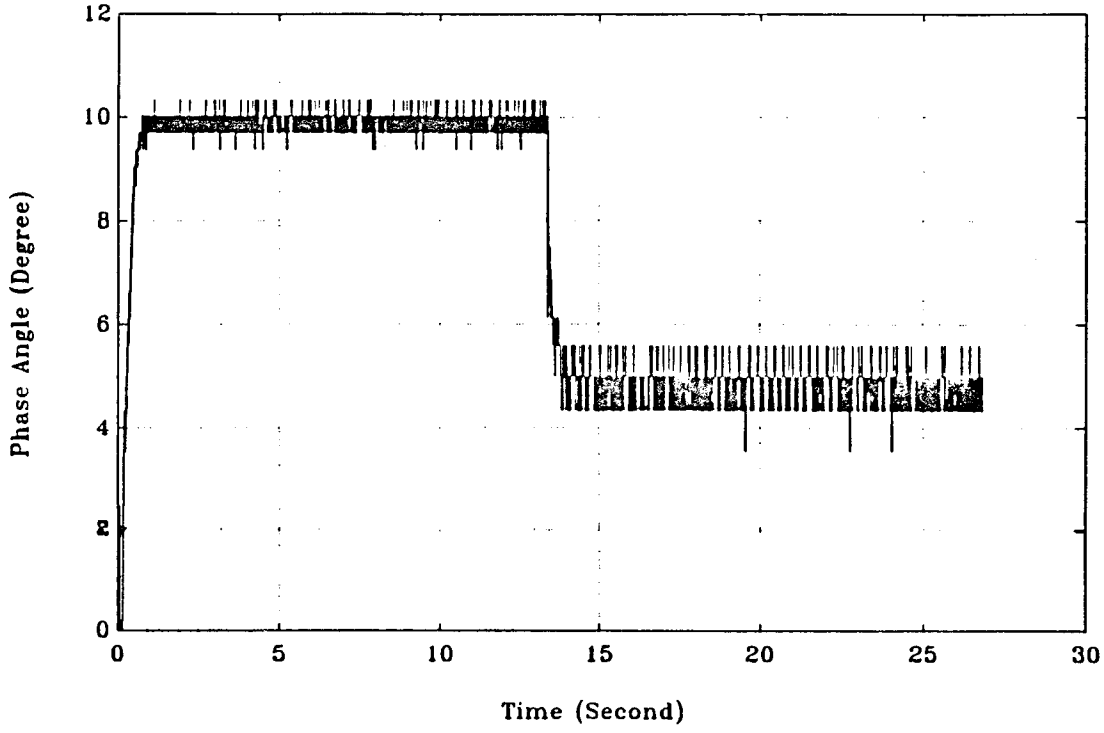


Figure C.28: 2-Loop Amplitude & Phase Control, Phase Angle Between Two Outputs,with K_I=0.025, K_P=-0.025

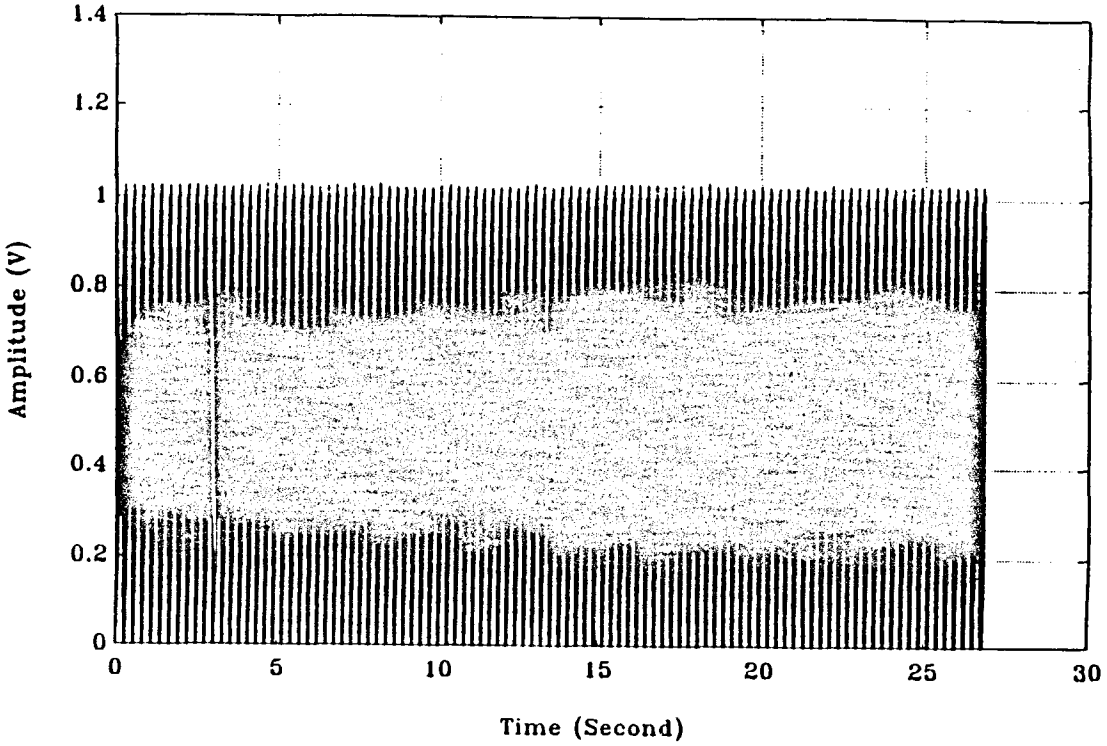


Figure C.29: 2-Loop Amplitude & Phase Control, Amplitude Output before Filtering

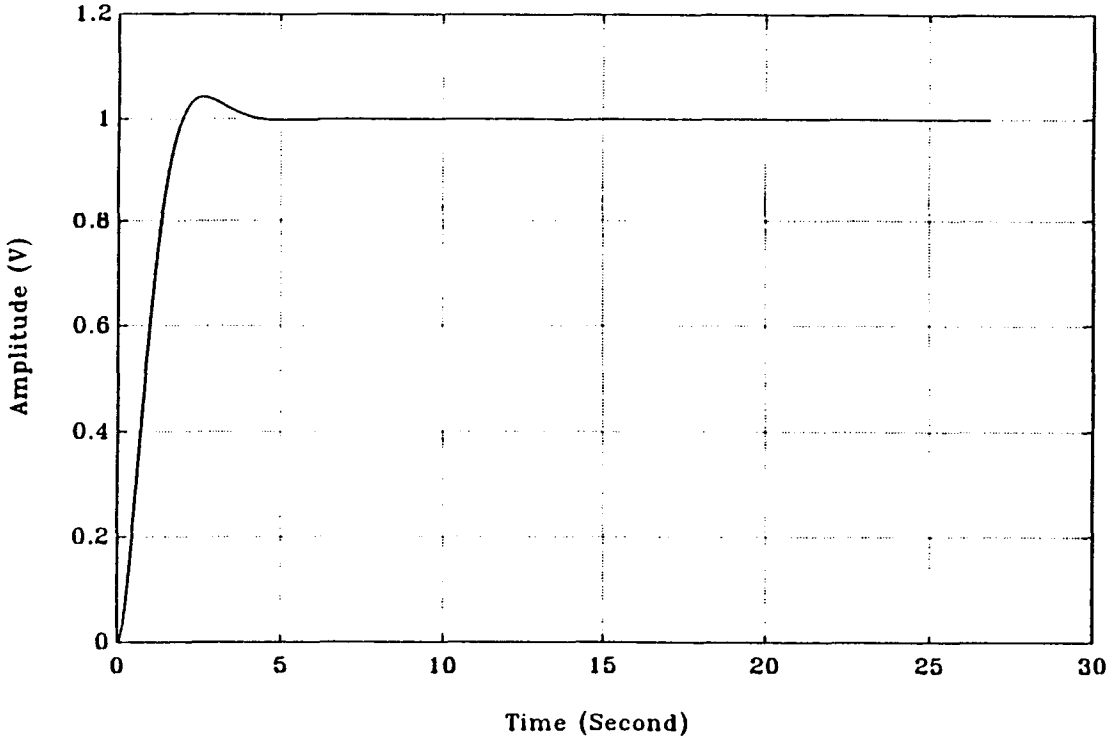


Figure C.30: 2-Loop Amplitude & Phase control, Amplitude Output after Filtering

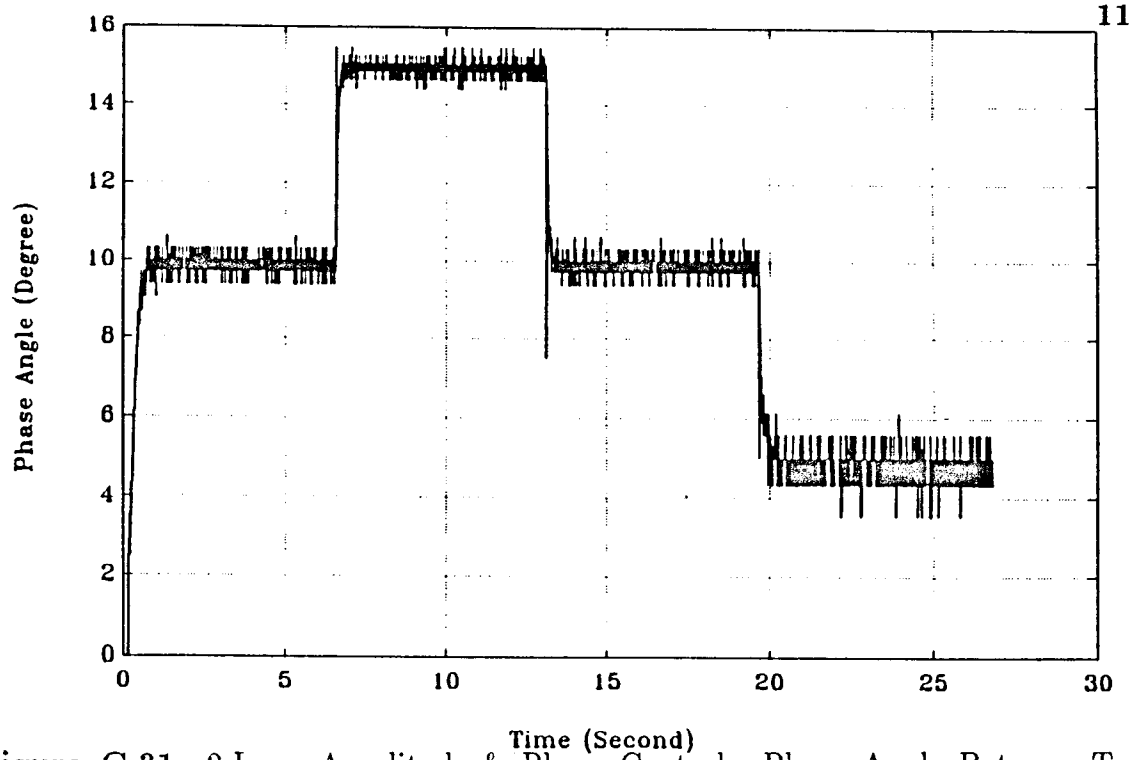


Figure C.31: 2-Loop Amplitude & Phase Control. Phase Angle Between Two Outputs with K_I=0.025 & K_P=-0.025

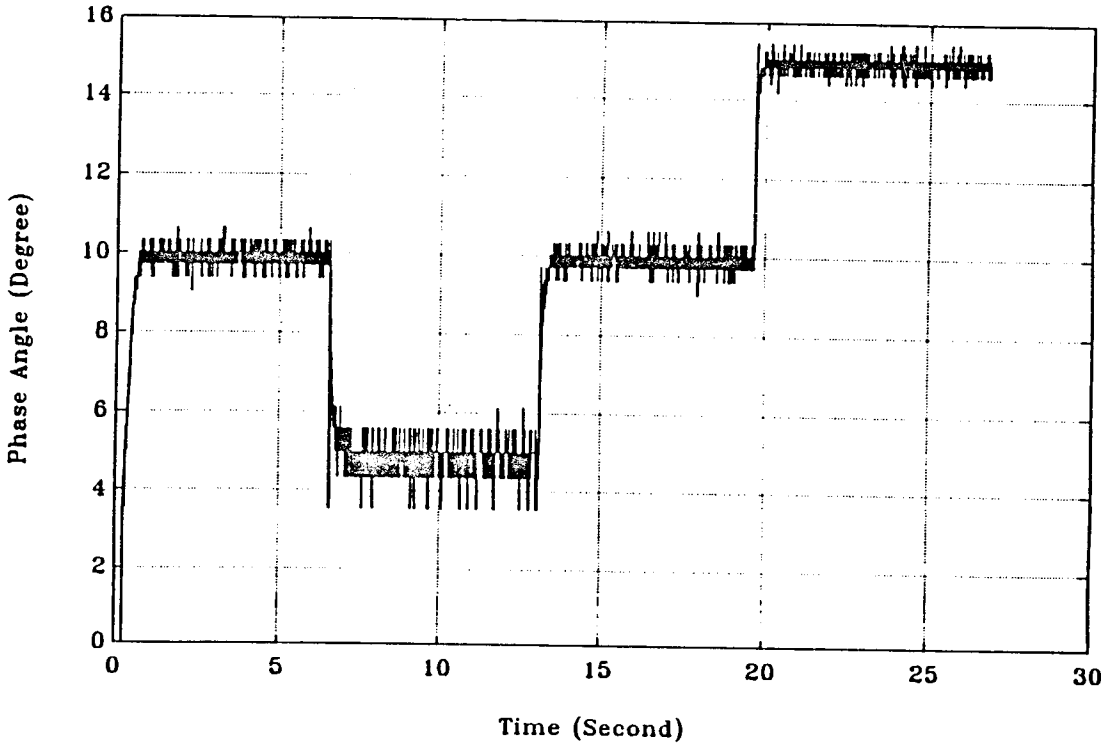


Figure C.32: 2-Loop Amplitude & Phase Control. Phase Angle Between Two Outputs with K_I=0.025 & K_P=-0.025

APPENDIX D

SIMULATION RESULTS AND CORRESPONDING EXPERIMENTAL RESULTS - FIGURES

The simulation results for 1) single loop amplitude control, and 2) 2-loop amplitude control, with two different gain(s) and corresponding results obtained by experimental mathod are presented in following sections.

D.1 RESULRTS FOR SINGLE LOOP AMPLITUDE CONTROL

In this section, simulation results for single loop amplitude control with two different gains are shown along with the corresponding experimental results using same gain(s).

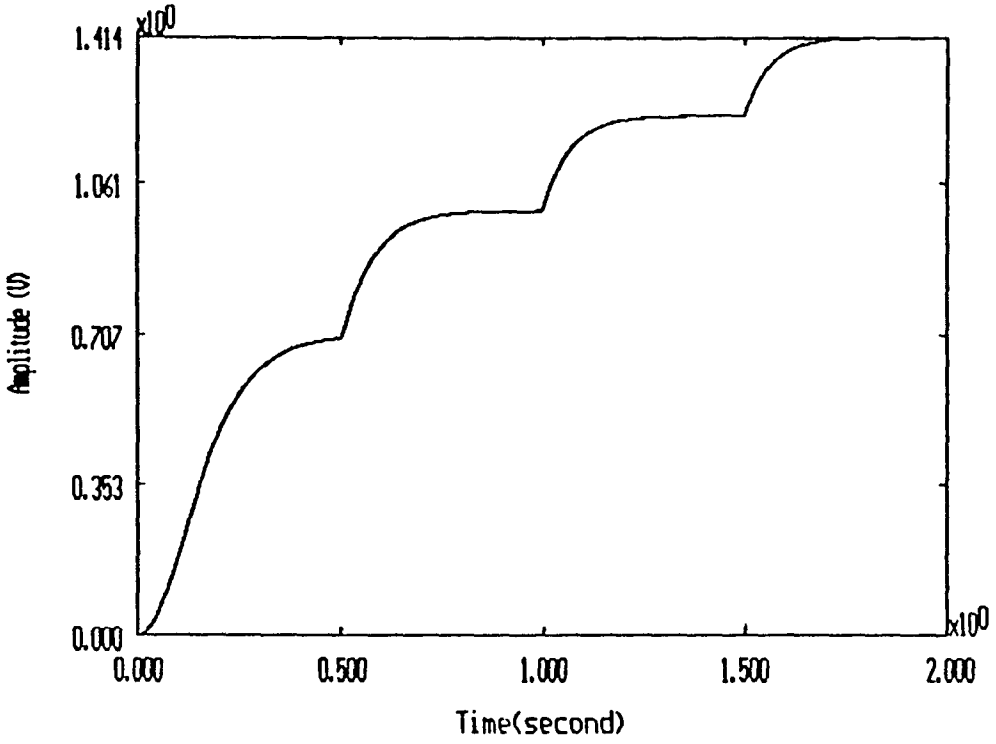


Figure D.1: Simulation for the Single Loop Amplitude Control with Gain K_I = 0.0025 (case 1)

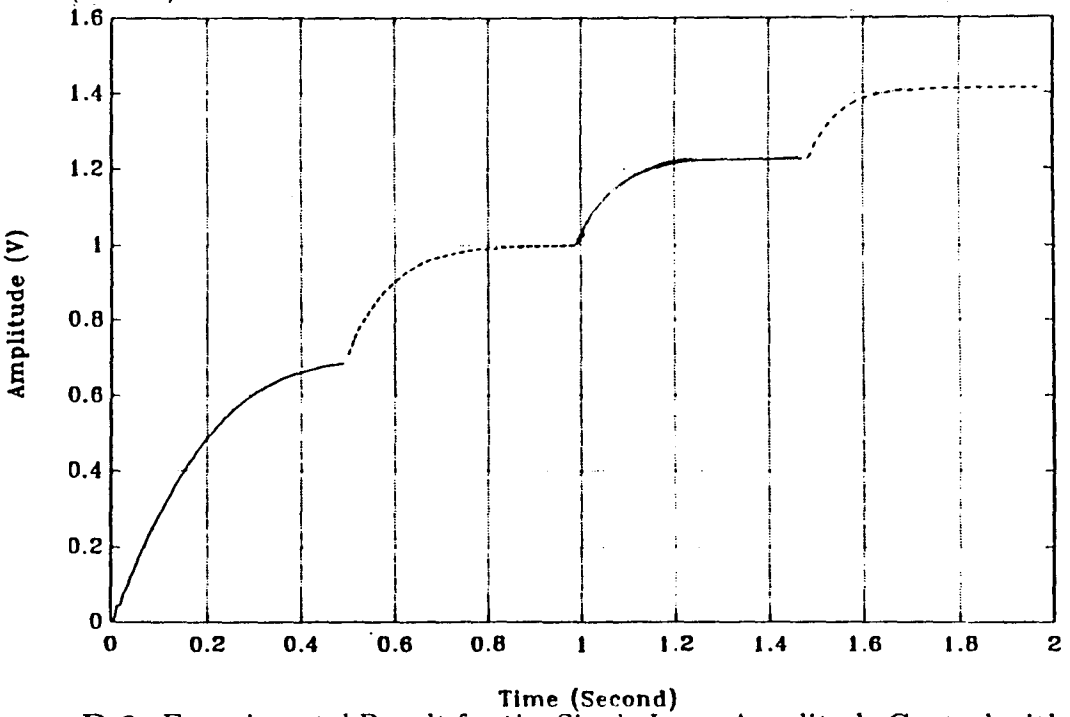


Figure D.2: Experimental Result for the Single Loop Amplitude Control with Gain K_I = 0.0025 (case 1)

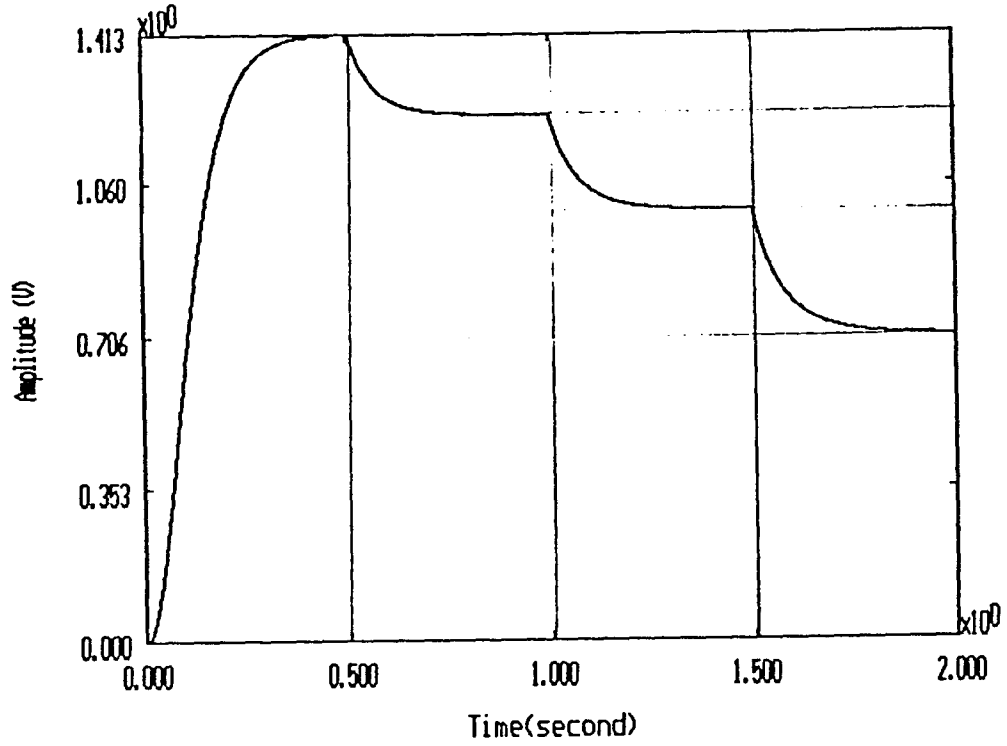


Figure D.3: Simulation for the Single Loop Amplitude Control with Gain K_I = 0.0025 (case 2)

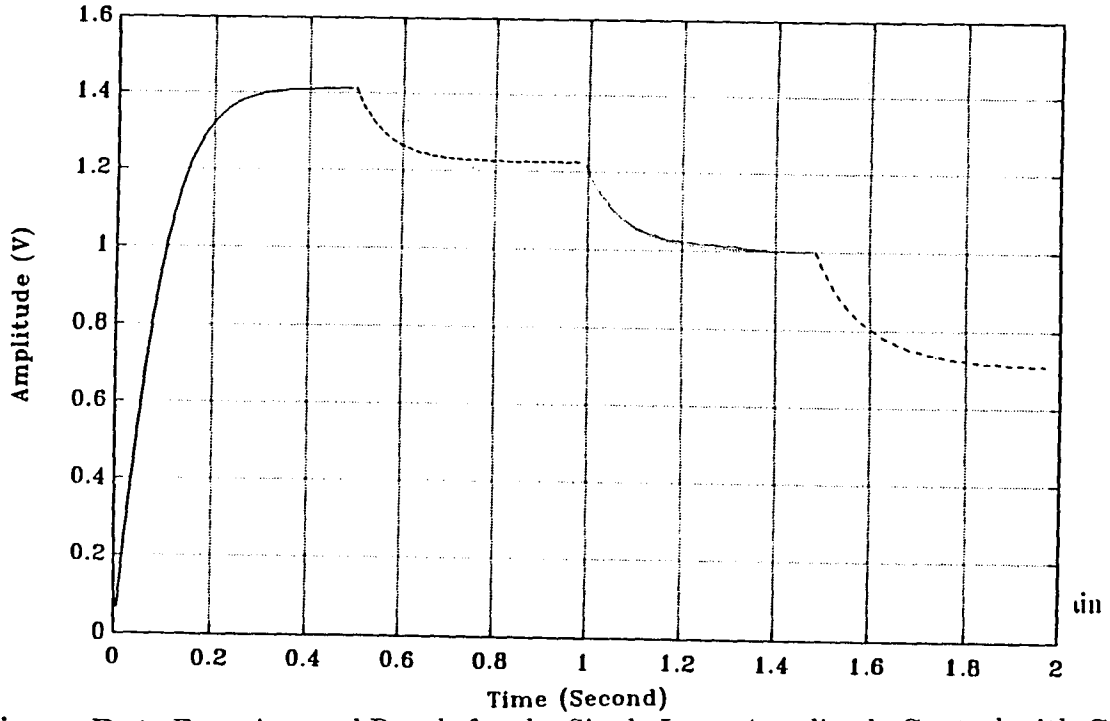


Figure D.4: Experimental Result for the Single Loop Amplitude Control with Gain K_I = 0.0025 (case 2)

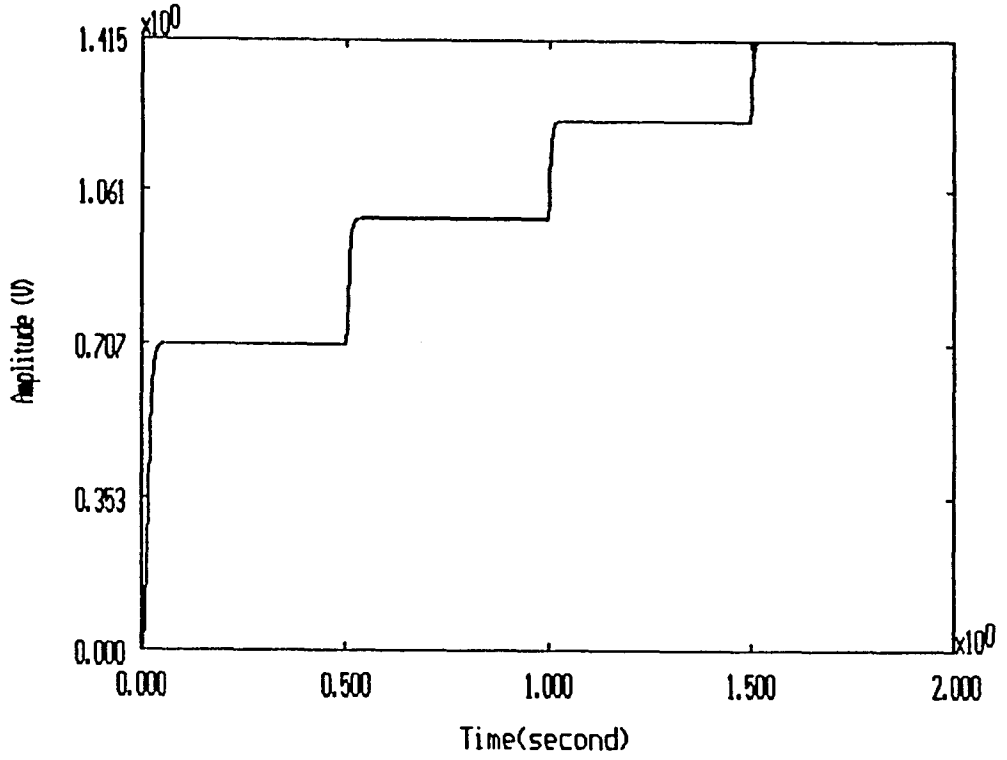


Figure D.5: Simulation for the Single Loop Amplitude Control with Gain K_I = 0.025 (case 1)

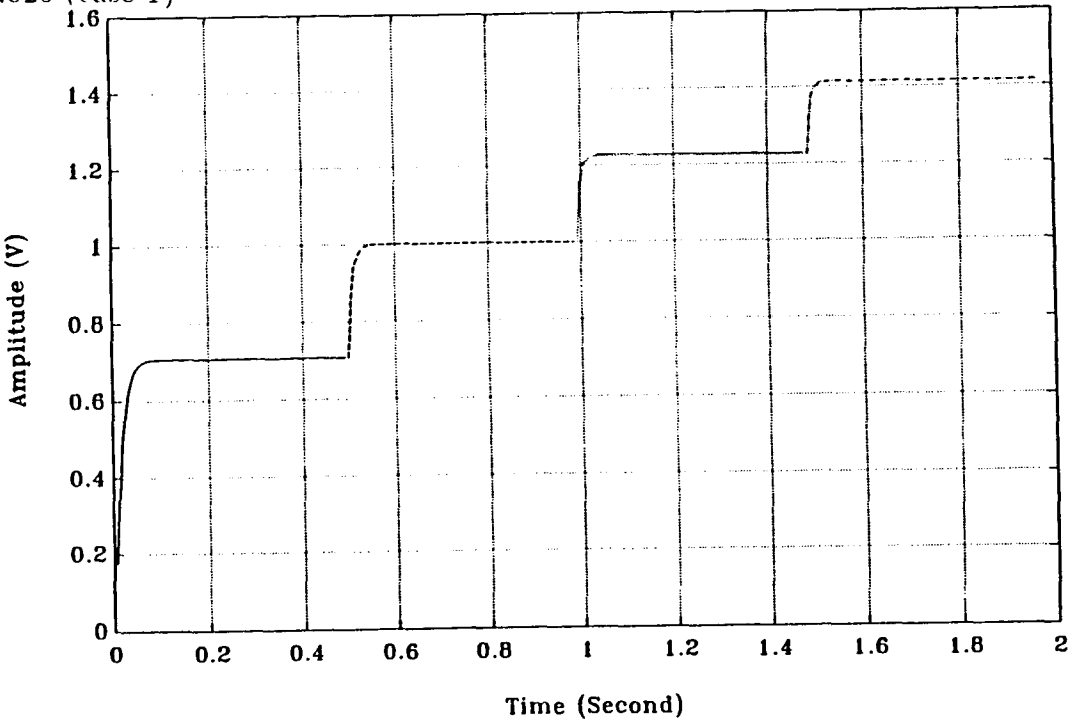


Figure D.6: Experimental Result for the Single Loop Amplitude Control with Gain K_I = 0.025 (case 1)

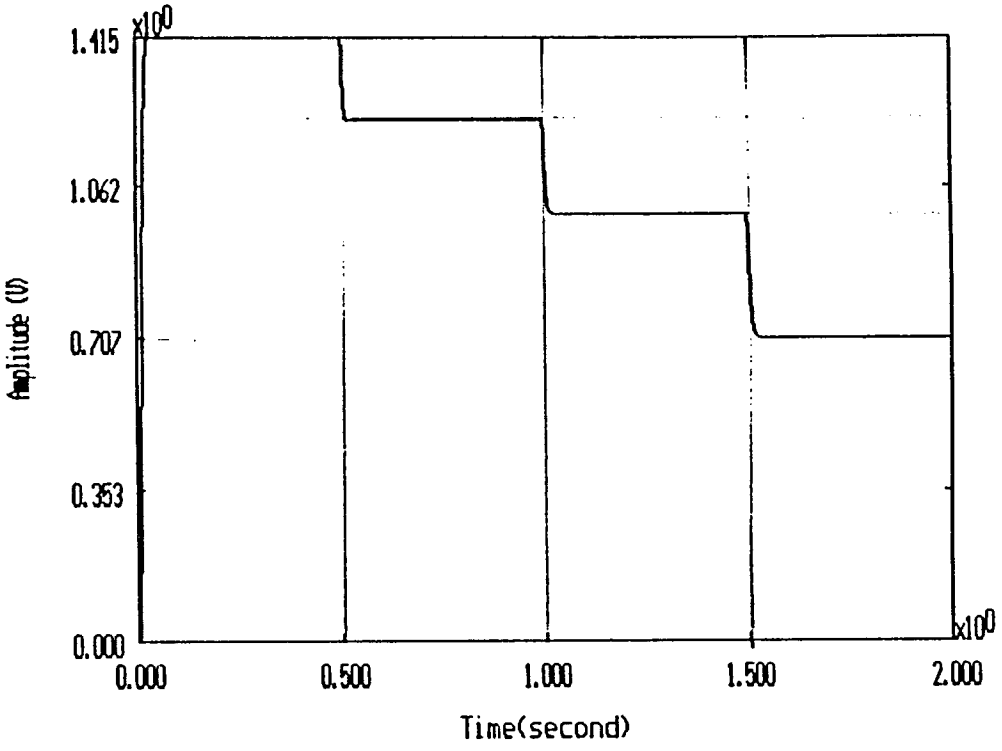


Figure D.7: Simulation for the Single Loop Amplitude Control with Gain $K_I = 0.025$ (case 2)

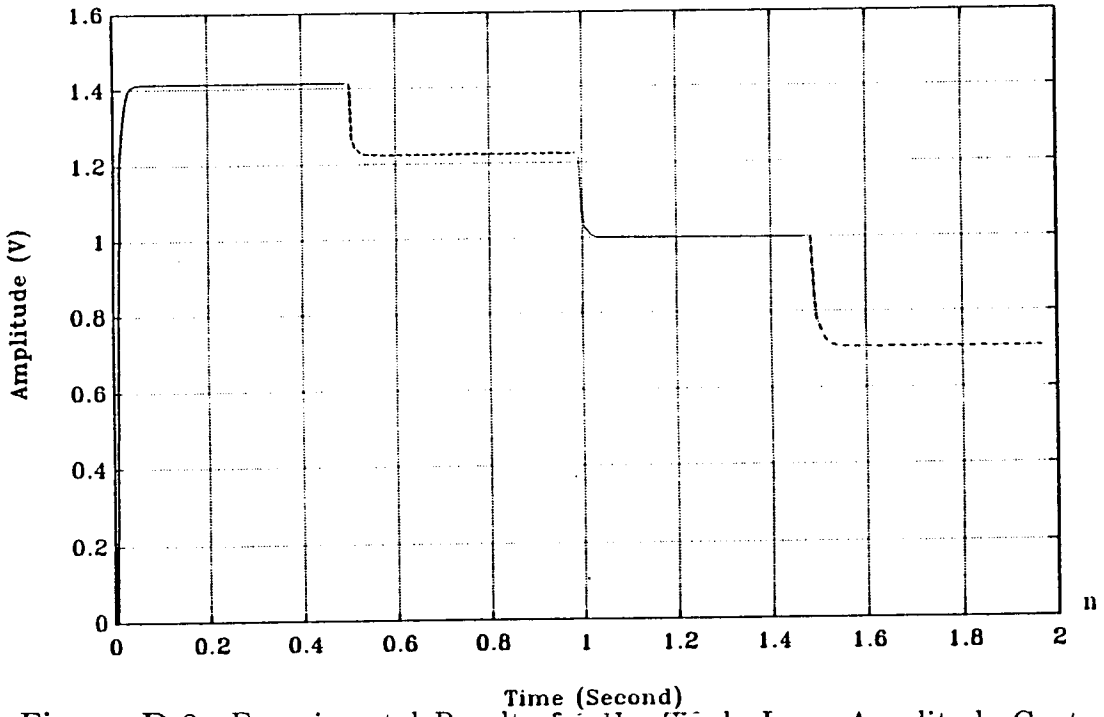


Figure D.8: Experimental Results for the Single Loop Amplitude Control with Gain $K_I = 0.025$ (case 2)

D.2 RESULTS FOR TWO LOOP AMPLITUDE CONTROL

In this section, simulation results for two loop amplitude control with two different gains are shown along with the corresponding experimental results using same gain(s).

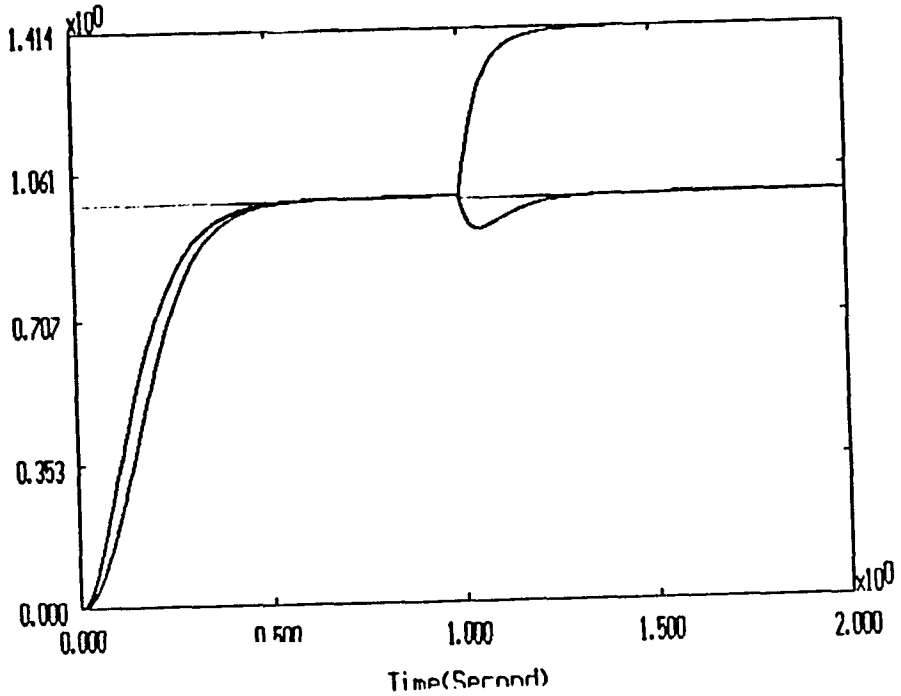


Figure D.9: Simulation for the 2-Loop Amplitude Control with Gain $K_I = 0.0025$ (case 1)

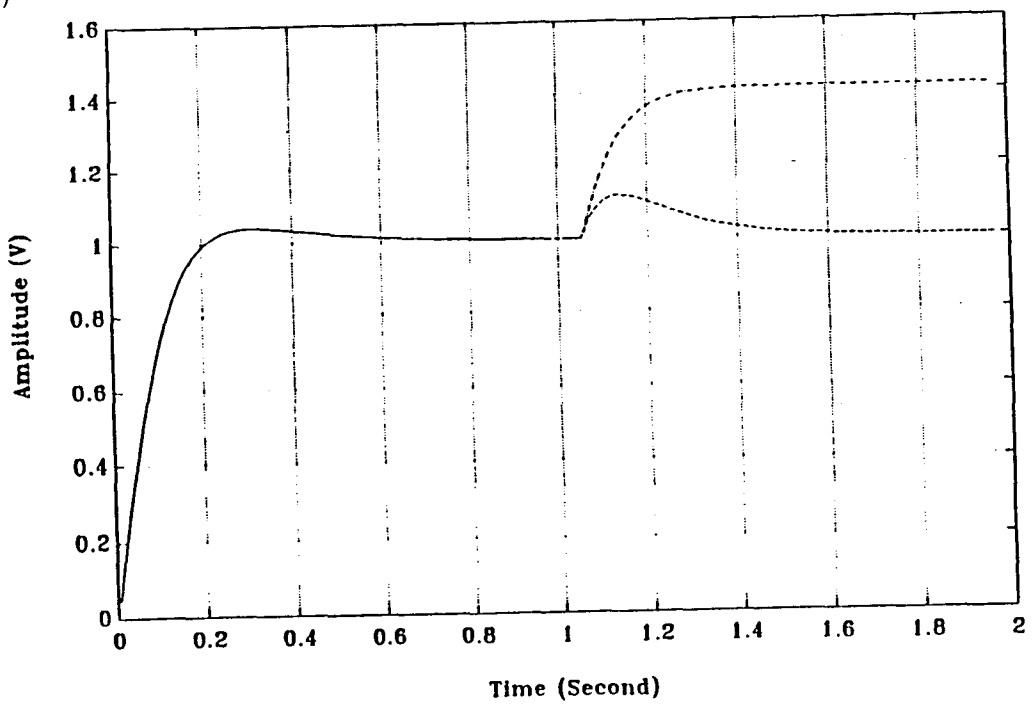


Figure D.10: Experimental Result for the 2-Loop Amplitude Control with Gain $K_I = 0.0025$ (case 1)

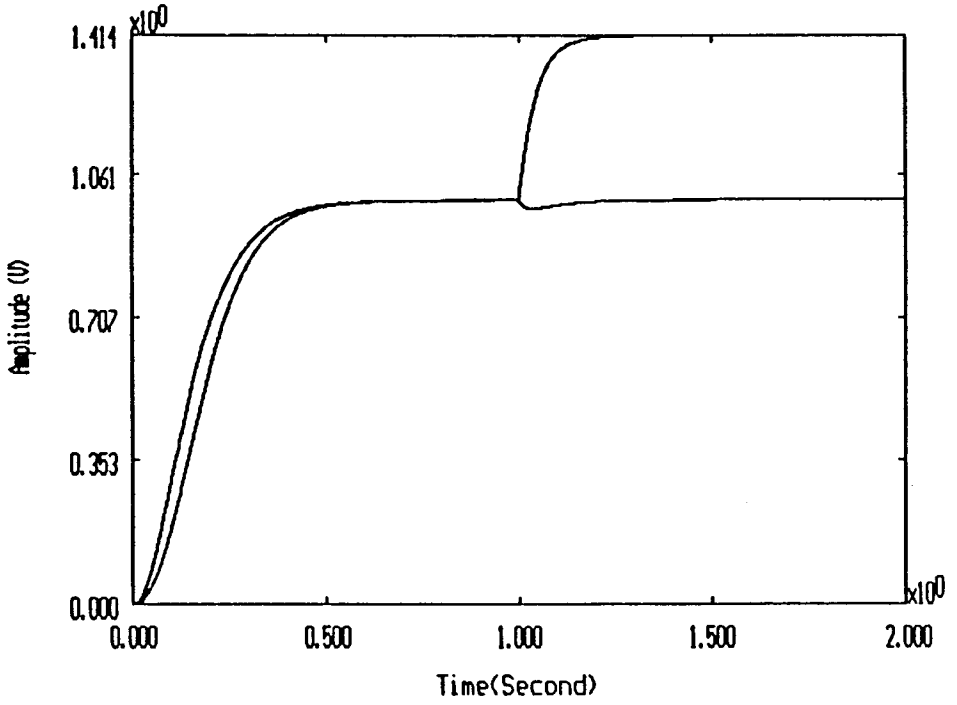


Figure D.11: Simulation for the 2-Loop Amplitude Control with Gain $K_I = 0.0025$ (case 2)

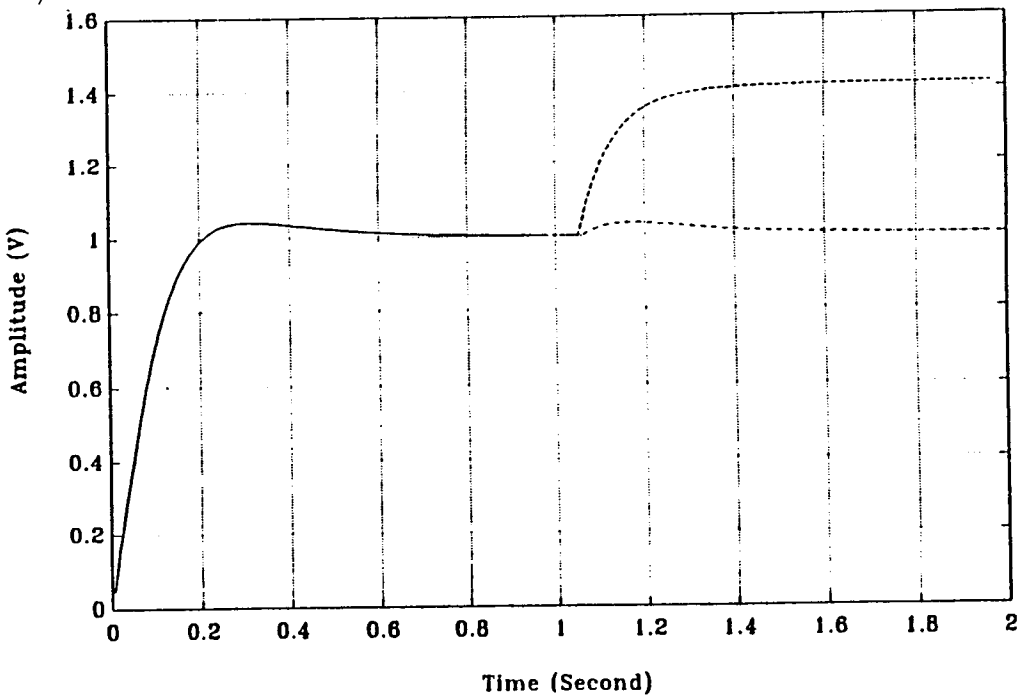


Figure D.12: Experimental Result for the 2-Loop Amplitude Control with Gain $K_I = 0.0025$ (case 2)

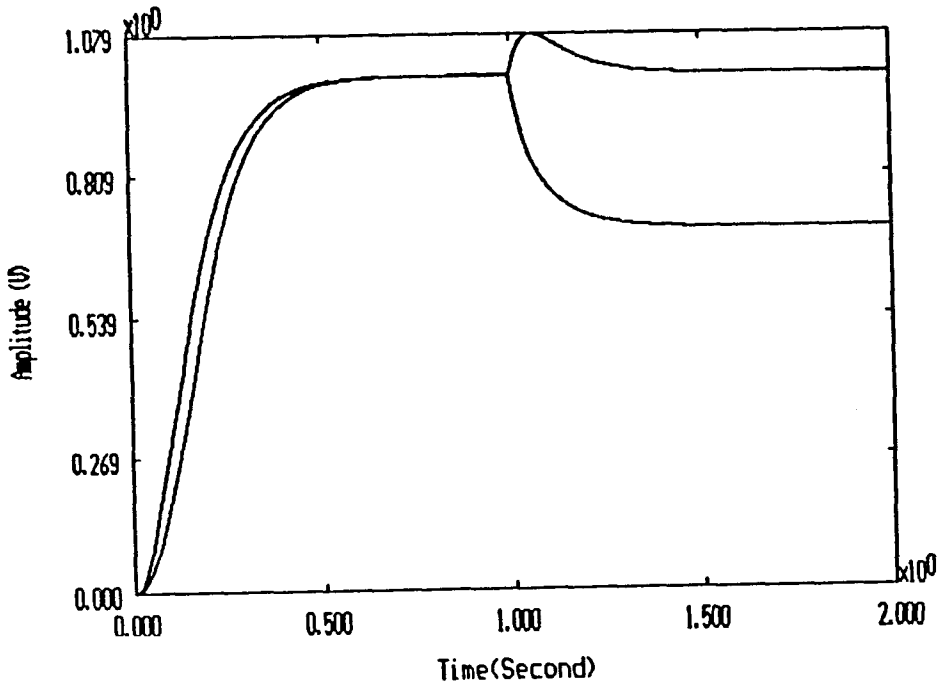


Figure D.13: Simulation for the 2-Loop Amplitude Control with Gain $K_I = 0.0025$ (case 3)

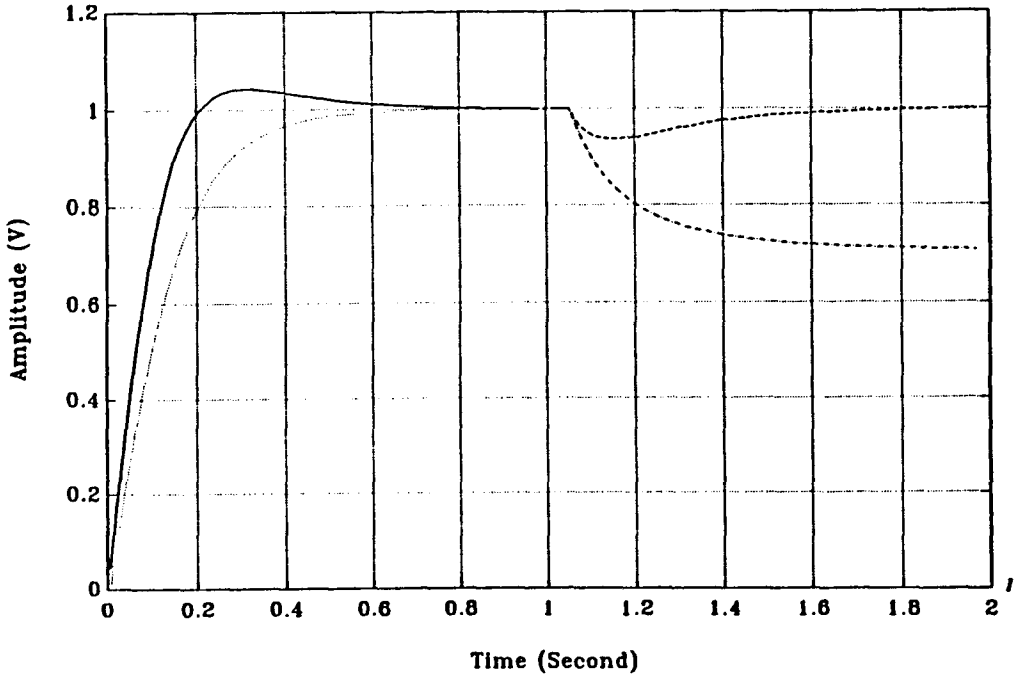


Figure D.14: Experimental Result for the 2-Loop Amplitude Control with Gain $K_I = 0.0025$ (case 3)

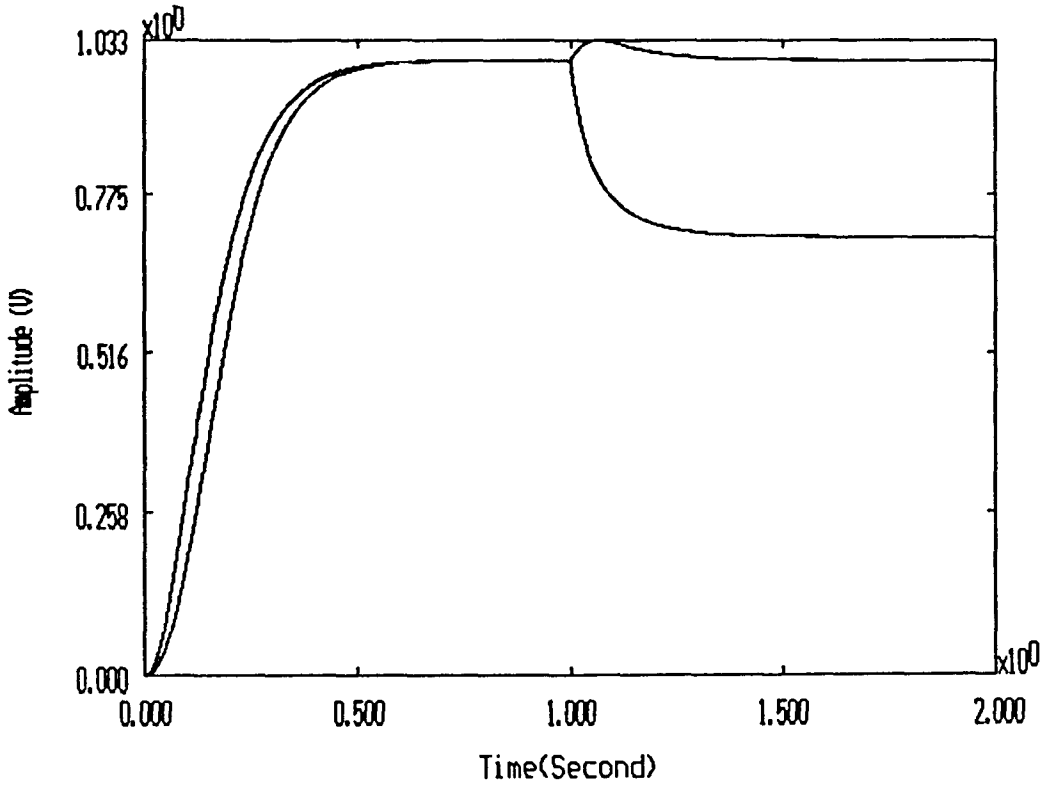


Figure D.15: Simulation for the 2-Loop Amplitude Control with Gain $K_I = 0.0025$ (case 4)

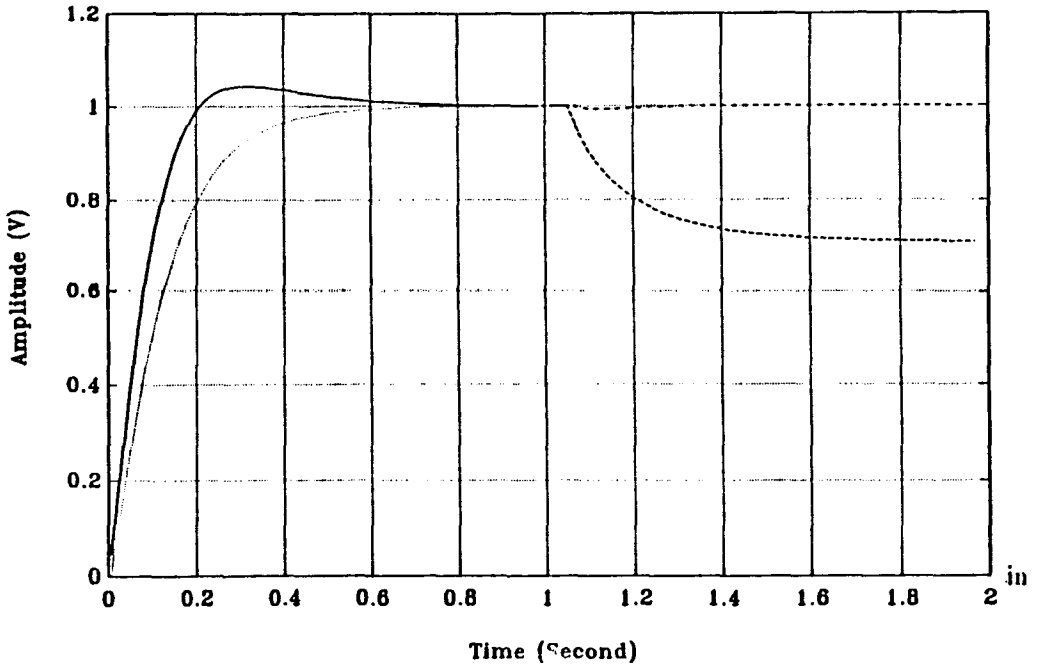


Figure D.16: Experimental Results for the 2-Loop Amplitude Control with Gain $K_I = 0.0025$ (case 4)

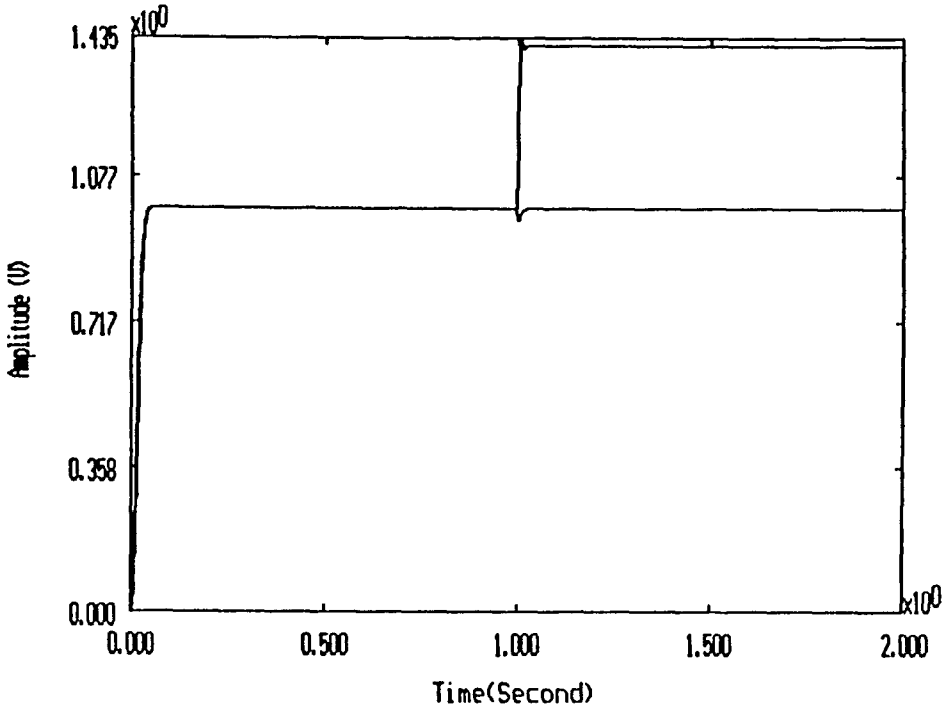


Figure D.17: Simulation for the 2-Loop Amplitude Control with Gain $K_I = 0.025$ (case 1)

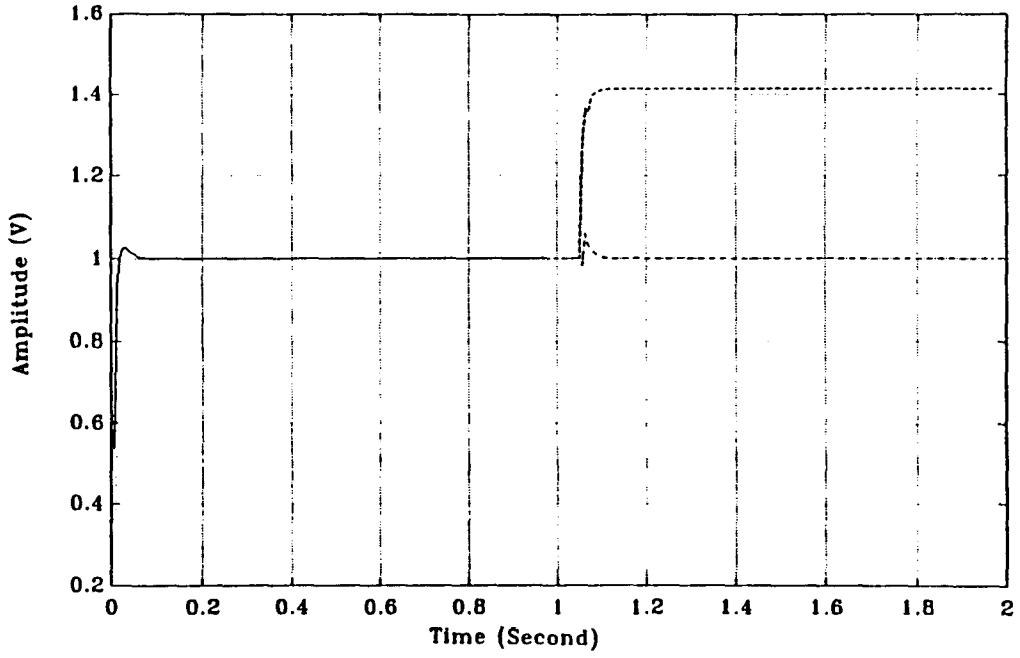


Figure D.18: Experimental Results for the 2-Loop Amplitude Control with Gain $K_I = 0.025$ (case 1)

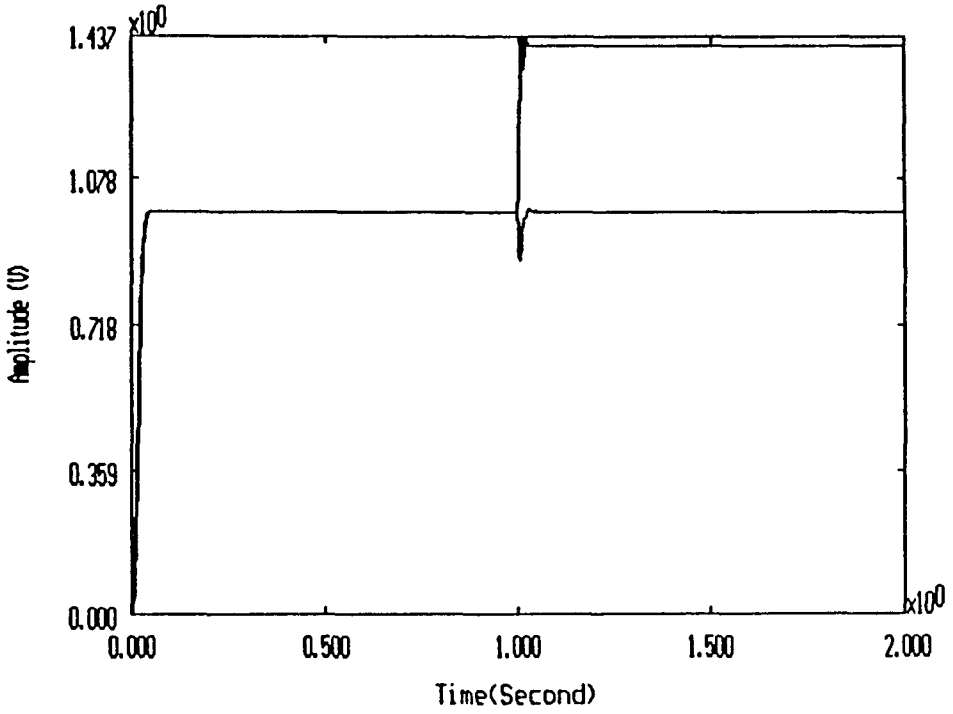


Figure D.19: Simulation for the 2-Loop Amplitude Control with Gain $K_I = 0.025$ (case 2)

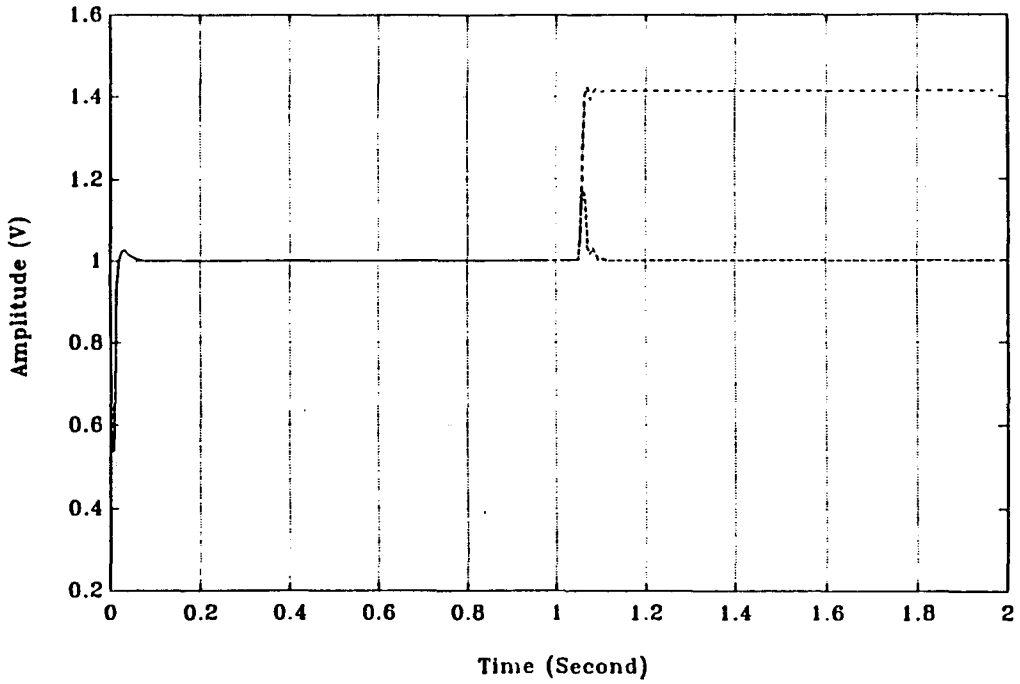


Figure D.20: Experimental Results for the 2- Loop Amplitude Control with Gain $K_I = 0.025$ (case 2)

REFERENCES

1. Timothy N. Chang *Decentralized Robust Control of Interconnected Resonators.* Proceedings to the 1992 IEEE International Conference on Control Application, pp 484 - 489. Dayton, OH

2. Texas Instrument's *TMS320C3x USER'S GUIDE*, 1992.

3. Sonitech International, Inc. *SAIB USER'S GUIDE version 1.0*, 1992.

4. Sonitech International, Inc. *SPIRIT-30 SYSTEM TECHNICAL REFERENCE MANUAL version 2.0*, 1992.

5. Timothy N. Chang *Course Note of Real Time Control System*, 1992.