STORED WAVEFORM ADAPTIVE MOTOR CONTROL

by

Jeffery C. Beall

Thesis submitted to the Faculty of

the Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE

in

MECHANICAL ENGINEERING

APPROVED:

Robert G. Leonard, Ph.D. Chairman

Harry H. Robertshaw, Ph.D.

Arvid Myklebust, Ph.D

March, 1986

Blacksburg, Virginia

Stored Waveform Adaptive Motor Control

by

Jeffery C. Beall

ABSTRACT

This study investigates an adaptive control scheme designed to maintain accurate motor speed control in spite of high-frequency periodic variations in load torque, load inertia, and motor parameters. The controller adapts, stores and replays a schedule of torques to be delivered at discrete points throughout the periodic load cycle. The controller also adapts to non-periodic changes in load conditions which occur over several load cycles and contains inherent integrator control action to drive speed error to zero. Using computer simulations, the control method was successfully applied to a 3ϕ synchronous motor and a permanent magnet D.C. motor. The D.C. motor (or A.C. servo-motor) controller has superior characteristics and this system performance was compared to P, PI and PID control for two severe load cases -- a periodic step load and a four-bar linkage load. Simulation studies showed the schedule control method to be stable and in comparison to the PID controller to have 1) nearly the same speed of response but without the overshoot found in PID control, 2) nearly the same mean speed error (≈ 0), 3) 12-50 times better reduction in speed fluctuation, and 4) the schedule controller gains were much easier to find than PID gains for this low-order, highly responsive system.

ACKNOWLEDGMENTS

I would like to thank the members of my committee, Drs. Myklebust and Robertshaw for their contributions to my undergraduate and graduate education.

I would like to thank for her enduring patience with my questions - over the past five years.

Most of all, I would like to extend my sincerest appreciation to Dr. Robert G. Leonard who made this possible for me. I believe the greatest benefit of graduate school is obtaining the priviledge to work closely with the wisest and most capable of individuals. Certainly the most valuable part of my education here has come from working with Dr. Leonard.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF FIGURES (continued)

LIST OF FIGURES (continued)

INTRODUCTION

This study investigates a simple adaptive control scheme which uses stored waveforms to improve speed control and torque production in a variety of applications. Though this study addresses several specific problems the method developed is not limited to these situations.

The primary purpose of this study was to develop a practical adaptive speed controller for applications where the load inertia and load torque requirements vary periodically as a function of rotor position. Electric-motor-driven four-bar linkages and slider-crank mechanisms are examples of this type of load. The varying inertia and torque requirements of these mechanical linkages may cause large, high frequency, periodic fluctuations in motor speed.

The controller developed here for speed control of the mechanical linkages above adaptively creates and stores in memory a schedule representing the torque values the motor must deliver at each rotor position in the load cycle. Since the load cycle repeats itself this torque schedule can be replayed as a function of rotor position. The schedule is updated continuously to compensate for slowly varying parameters in the motor or load. The values in the torque schedule represent either a voltage or current, depending on the amplifier design. This controller has the advantage of requiring very little a-priori information about the load since only speed and position are used to adapt the schedule.

The results of this work suggests other applications for adaptive torque scheduling. Some of the possible applications are discussed in the Recommendations.

This study has two parts. Chapter 1 examines torque schedule control of a three-phase permanent magnet synchronous motor whose torque angle, δ, is not inherently fixed by controller action. The torque angle, δ, is the angle between the stator magnetic axis and the rotor magnetic axis and is ideally equal to 90°. The work in Chapter 1 suggests motor control would be much simplier if δ is fixed at 90° by the controller. The controller design can easily be modified to accommodate this change and the result is a simpler implementation of torque angle control than is currently in the literature [1]. Fixing δ essentially transforms the three-phase synchronous motor into a brushless AC servo motor. The results of Chapter 2, which examine torque schedule control of a permanent magnet d.c. motor, also apply to the three-phase AC servo motor. Conclusions and recommendations follow in Chapter 3.

LITERATURE REVIEW

There are a great number of control strategies available for motor speed control. Probably the most popular methods for d.c. motor control are the linear classical control methods which use some combination of proportional (P), integral (I) or derivative (D) control or derivative feedback. These same principles are often used in control systems which are not linear. The low cost and high efficiency of solid-state switching devices have led to the popularity of non-linear motor controllers and motor drives. Chopper drives using pulse frequency and pulse width modulation (PWM) and half-wave and full-wave thyristor drives are popular for d.c. motor applications. Square wave, PWM, and other inverter configurations are used in brushless d.c. and synchronous motor applications. Non-linear motor controllers (excluding adaptive techniques) commonly use on-off feedback (with or without a deadband) or error-squared feedback.

Pfaff [1], for example, uses several new techniques in controlling a brushless AC servo drive. A coordinate transformation combined with rotor position information is used to construct the stator flux ninety electrical degrees ahead of the rotor flux at all times - effectively fixing the torque delta, δ. The coordinate transformation matrix is implemented with summing amplifiers and stored sine and cosine functions. This logic generates the desired current level for each phase winding as a function of rotor position. Pfaff used a new technique for controlling the voltage source inverter which drives the windings. The current level in each winding is maintained within a deadband region around the reference value using on-off control. This

technique effectively minimizes the switching rate needed (at any given operating point) for constructing a sinusoidal approximation by PWM. (PWM typically uses a fixed switching rate with a variable duty-cycle to approximate the sinusoidal waveform.) The AC servo had two other loops also, an analog P.I. loop measuring speed for speed control and a digital P feedback loop for position control.

Linear state feedback may be used for pole placement if the system (the A matrix) is well known and is essentially time invariant. The load cases of interest here, however, have widely varying parameters and load conditions and preclude the use of such methods.

Adaptive control, depending on the method, offers some relief. A relatively simple scheme described by Mounfield [2] uses signal synthesis to alter the dynamic characteristics (damping ratio, natural frequency) of any system which can be approximated as second-order. The adaptation rates of this method are too slow for load conditions which vary at twice the motor speed. Carlson [3] studied several adaptive control schemes as applied to controlling the input link speed of a four-bar linkage. Carlson was able to reduce the speed fluctuations to less than ±2% using perturbation adaptive control [3]. This method models the linkage kinematic influence-coefficients, calculates the required torque at each point in the load cycle and delivers the calculated torque. This torque is only an approximation to the actual torque requirement (depending on the accuracy of the model) and the error or perturbation is controlled by a secondary controller. This method requires knowledge of the linkage and is somewhat calculation intensive.

Carlson also simulated a non-linear adaptive control scheme that was not mechanism dependent. This control scheme utilized a generalized linkage model to estimate torque and inertia. The coefficients of this generalized model were found by a least squares estimator. Carlson concluded that this approach still requires some modification to obtain an acceptable transient response.

An alternative to the solution of kinematic equations for manipulator control has been suggested by Albus [4]. Albus proposes an adaptive system - Cerebellar Model Articulation Controller or CMAC, which computes control functions by referring to a table. System reference commands and feedback signals are used to address a memory location where the control functions are stored. The control functions are created by one of several iterative training procedures. The torque schedule controller proposed here is much simpler, but operates in a similar manner.

Many of the control methods discussed here are obviously not suitable for driving high frequency periodic loads with changing parameters. Others could do the job perhaps but are difficult to implement. This work compares the performance of the three linear classical combinations of P, PI, and PID control with the simple torque scheduling controller.

1.0 SYNCHRONOUS MOTOR CONTROL

Introduction

The purpose of this chapter is to present a new method for speed control of three-phase synchronous motors. A permanent magnet rotor machine is used to describe the method but the method could be applied to wound rotor machines was well.

A preview of each section of this chapter is given below. These give an overview of the material to be presented and allow the more knowledgeable reader to skip over some of the sections.

1.1 Synchronous Motor Operation - explains synchronous motor fundamentals and develops concepts and vocabulary used in later sections. The main points of this section are summarized for quick review on p. 17.

1.2 Balanced Current Motor Model - contains the derivation of the equations which relate the electrical terminal variables, current amplitude and frequency, to the motor shaft torque and speed. The equations developed here provide the basis for the model used in the computer simulation.

1.3 Overview of Control Strategies - describes in the most general sense what must be done to control speed and torque of the three phase synchronous motor.

1.4 Overview of Synchronous Motor Drives - describes in a general way the manner in which speed control is now accomplished in three phase synchronous motors.

1.5 Torque Scheduling Controller - describes a new method of speed

and torque control which can accomodate the high frequency periodic loads discussed in the literature review.

1.6 Performance Evaluation and Discussion - presents the results of simulation studies which compare the performance of the torque scheduling controller to the most common synchronous motor configuration. Four-bar linkage and viscous damping loads are considered here.

1.1 Synchronous Motor Operation

In this section, synchronous motor operation is explained for the simplest 3-phase machine. A schematic drawing of this motor is shown in Fig. 1-1. The armature windings for each phase (A, B and C) are shown as discrete bundles of conductors housed in the stator slots. The rotor is a simple 2-pole permanent magnet with a single magnetic axis as shown in Fig. 1-1.

Figure 1-2 shows how a current loop can generate a magnetic flux whose polarity is given by a right-hand rule. If the fingers of the right hand curl in the direction of current flow around the coil, then the thumb will be pointing in the direction of flux flow from south to north on the interior of the coil.

Figure 1-3a shows in better detail how one phase winding of the motor forms a current loop. The phase winding generates a flux as shown in Fig. 1-3b. The polarity of the flux depends on the direction of the current flowing in the windings, also shown in Fig. 1-3b.

If a permanent magnet rotor is placed in the stator of Fig. 1-3a (see also Fig. 1-4a), the rotor magnetic axis will tend to align with the phase A magnetic axis as shown in Fig. 1-4b. The rotor experiences

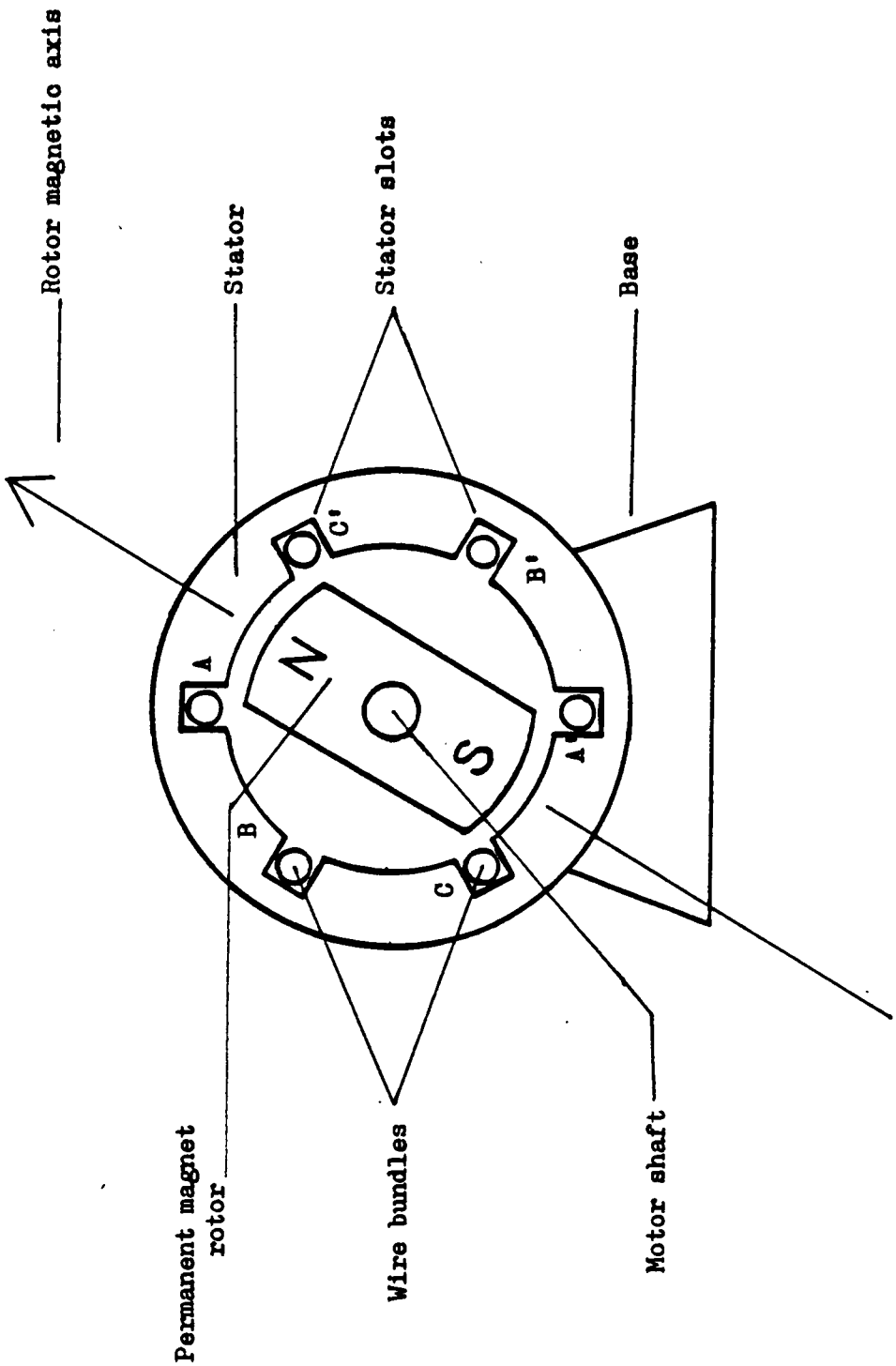


Fig. 1-1 Simplified 2-pole, three-phase synchronous motor with permanent magnet rotor (end view).

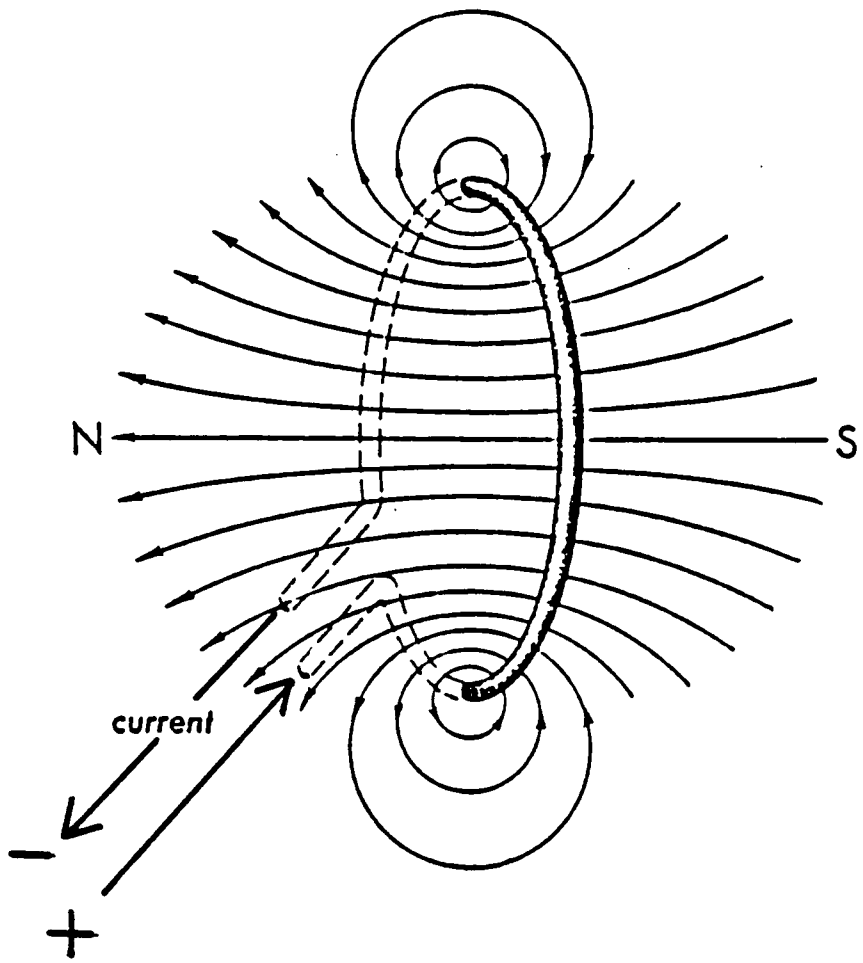


Fig. 1-2 Diagram of the magnetic field through and around a single loop of wire carrying an electric current.

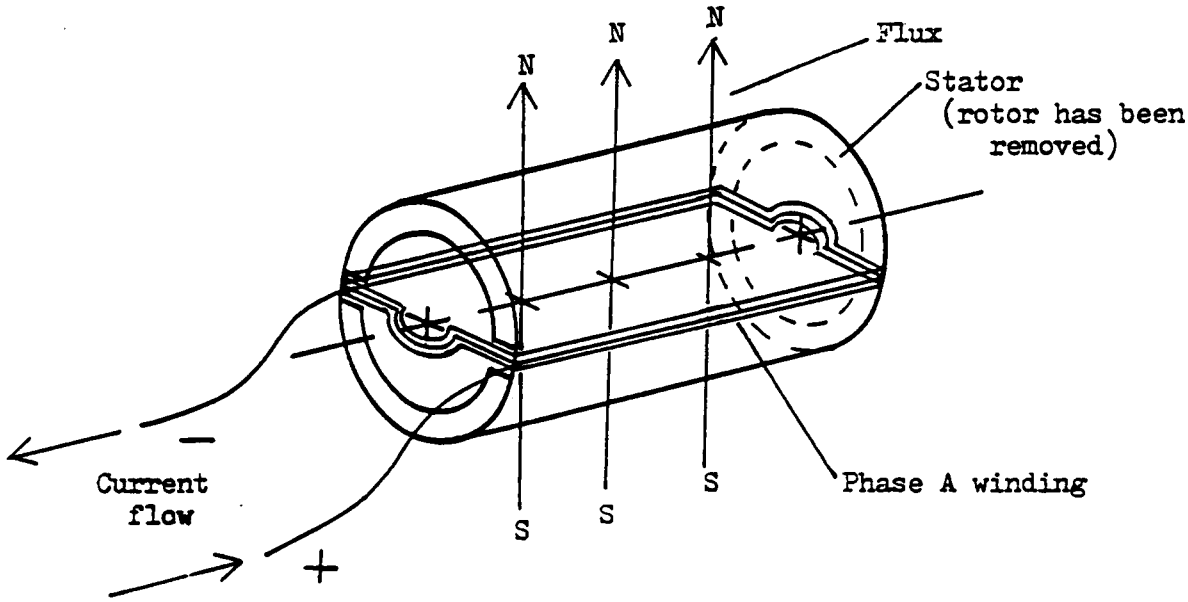


Fig. 1-3a Current loop formed by a single phase winding.

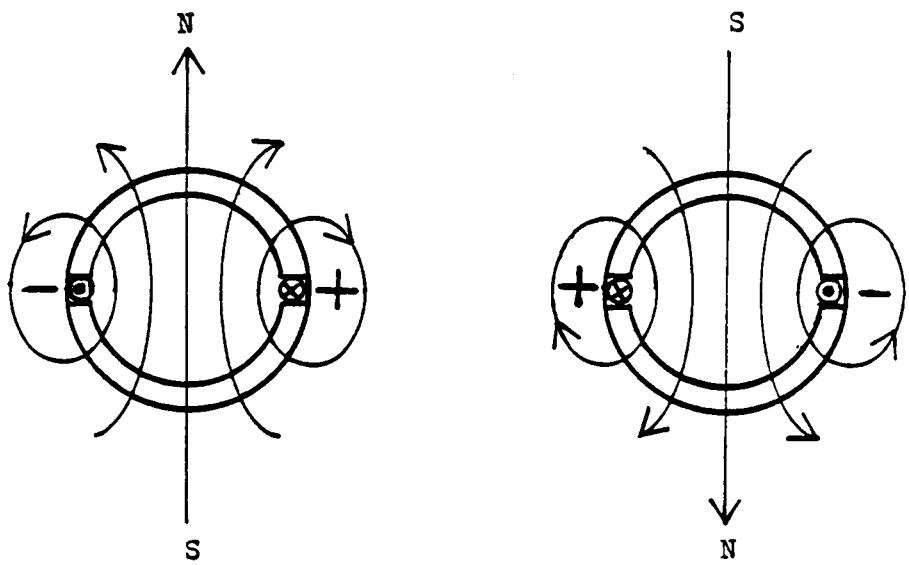


Fig. 1-3b End view showing change of flux polarity when current direction is reversed. Figure 1-3b (left) corresponds to Fig. 1-3a.

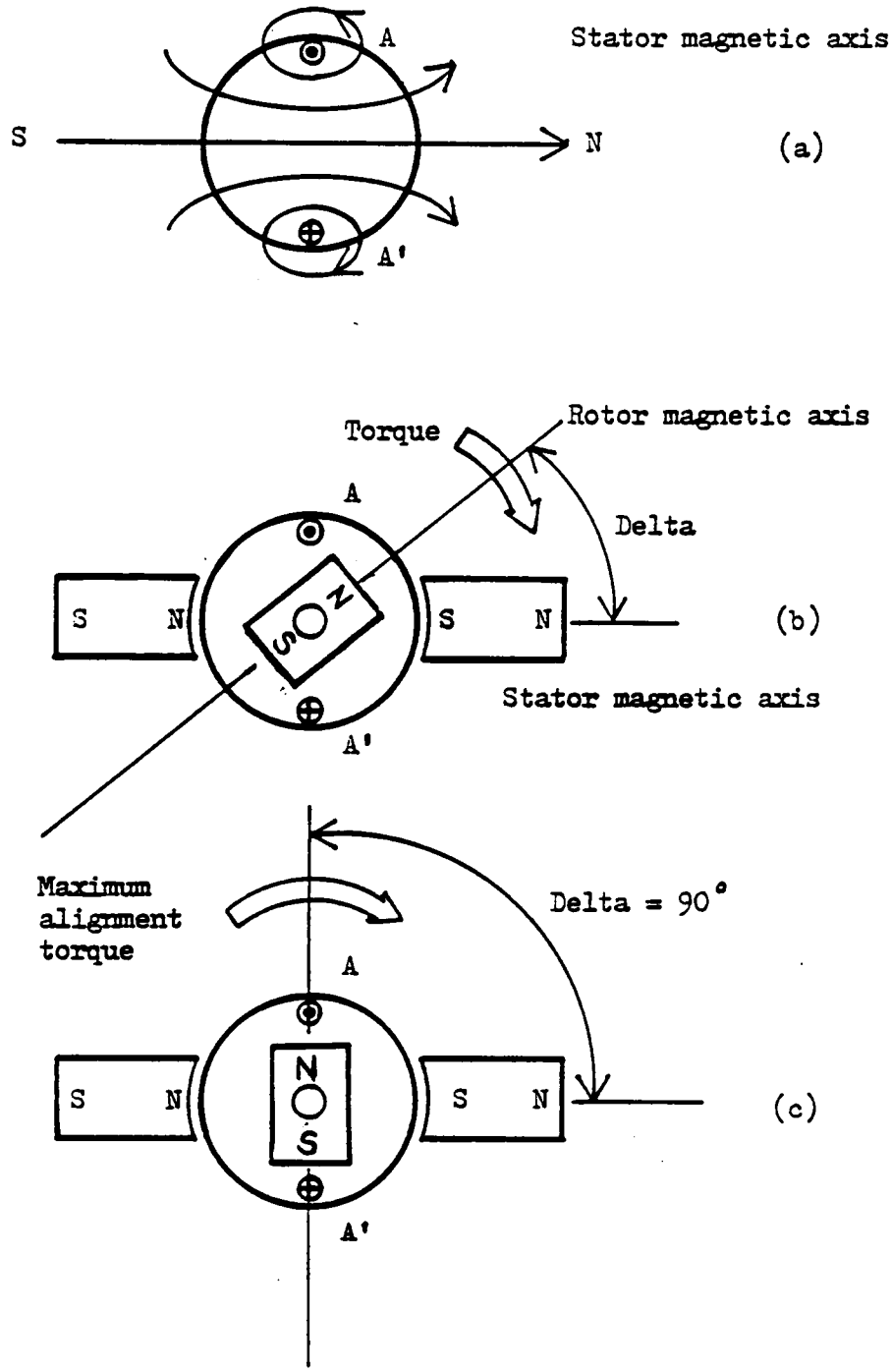


Fig. 1-4 Diagram of rotor alignment with phase A flux.

an alignment torque given by

$$T = F \sin(\delta)$$

where F is a measure of the rotor and stator magnetic field strength and δ is the angle between the rotor and stator magnetic axes. The maximum value of alignment torque occurs when δ equals 90° and $\sin(\delta)$ equals one (see Fig. 1-4c).

By adding two more windings, B and C, two more magnetic axes are added to the stator. The three phase windings and their respective magnetic axes are shown in Fig. 1-5.

If current is driven through successive phase windings, then the rotor will "step" around to align with successive magnetic axes, which are 60° apart as shown in Fig. 1-6. The rotor will align on each phase axis twice during each revolution of the rotor. Note that the current through a winding must be reversed when the rotor approaches the axis for the second time in that revolution. If the current and hence polarity of the magnetic flux is not reversed, the rotor will be repulsed rather than attracted to the axis.

If the rotor is stepped around by switching the phase currents in a square wave fashion, the motor torque and speed will be irregular. To smooth the torque production, each winding is driven with a sinusoidal current. The three current waveforms, which are identical and 120° out of phase from each other, are plotted against the electrical angle ($\omega_e t$) in Fig 1-7. This plot allows us to find the value of the current in any phase winding at any point in time. The magnetic flux

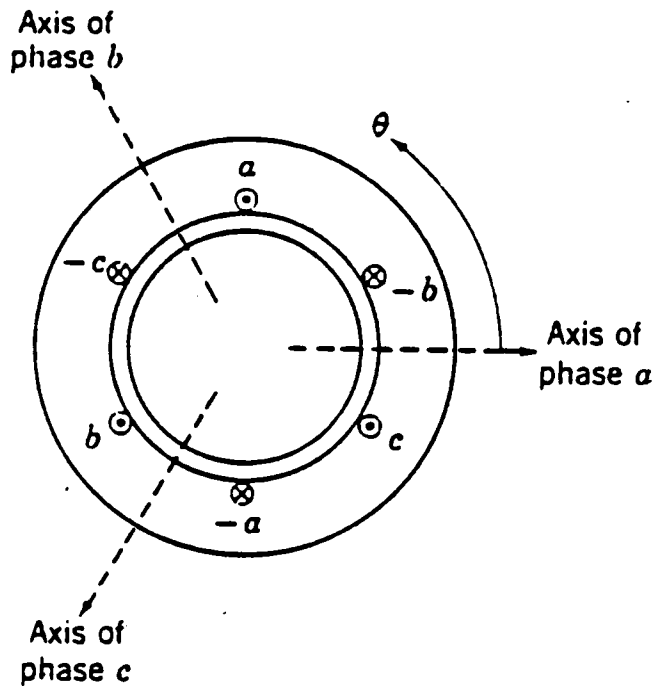


Fig. 1-5 Phase windings and their respective magnetic axes. Adapted from Fitzgerald [5], p. 146.

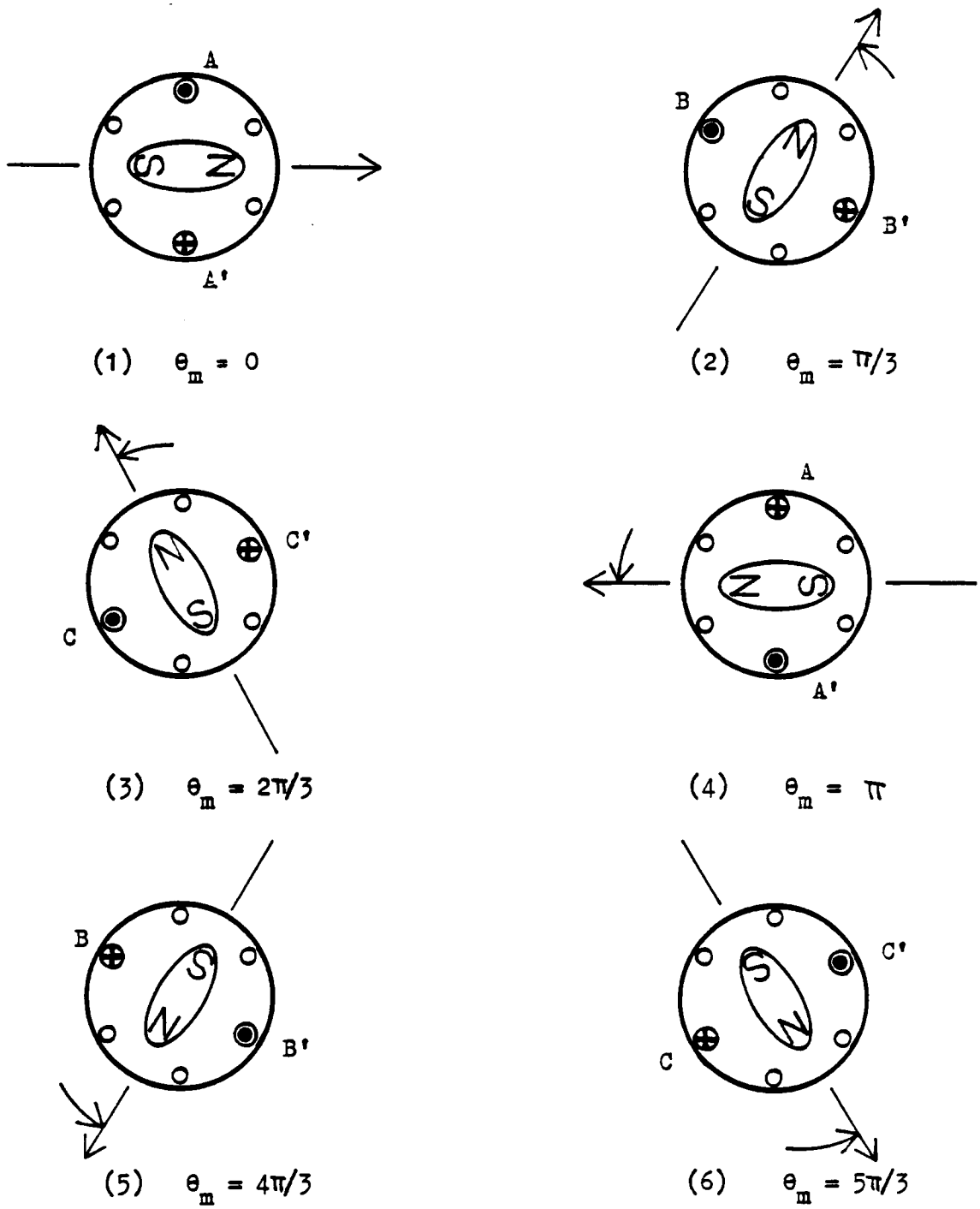


Fig. 1-6 Rotor is stepped around by driving current through successive phase windings. Current is reversed on the second alignment with a particular axis.

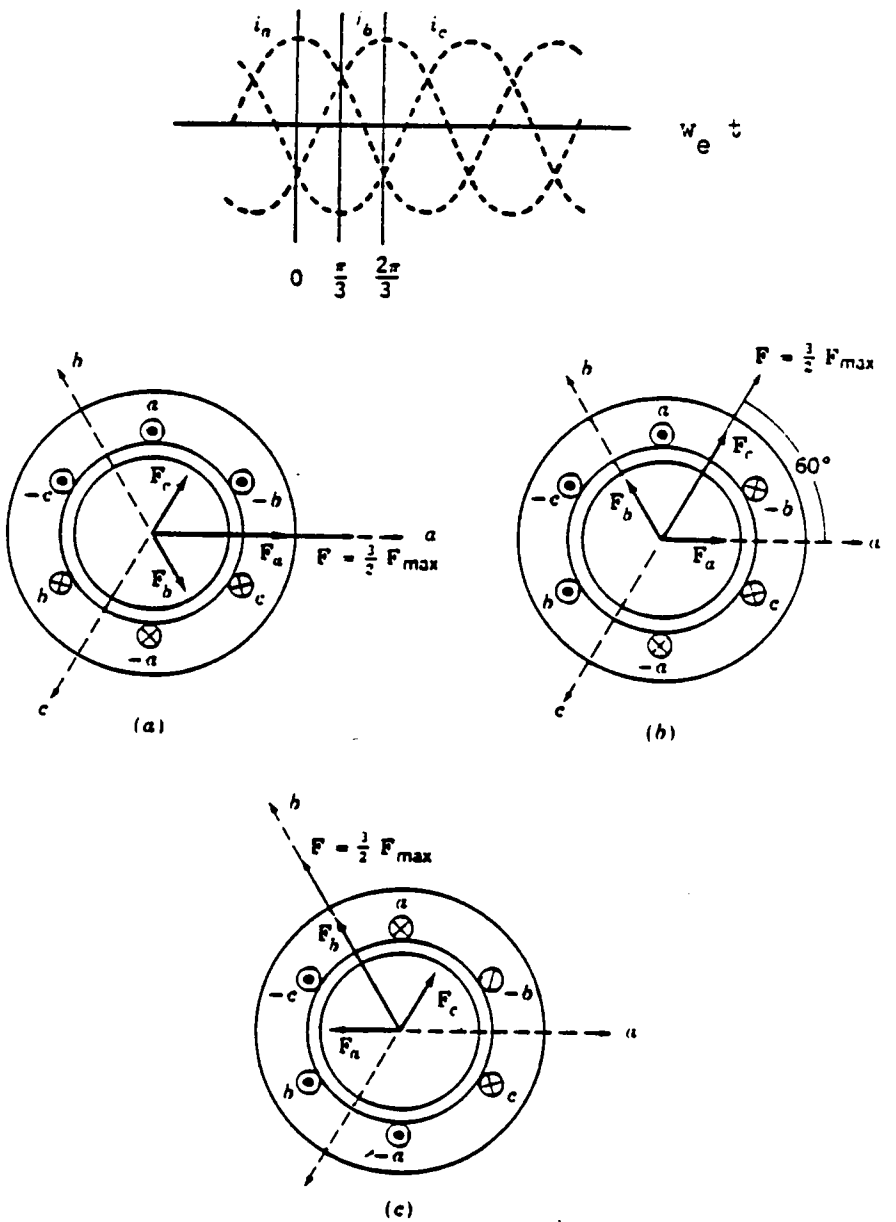


Fig. 1-7 Instantaneous phase currents plotted against $w_e t$ (top) and the rotating field they produce. Adapted from Fitzgerald [5], p. 149.

produced by a given phase winding is proportional to the current in that phase. The flux can be visualized as a vector along that particular phase axis. The length and polarity of the flux vector is determined by the amplitude and direction of the current flowing in the winding. Figure 1-7a shows the summation of the three flux vectors F_a, F_b, and F_c which are generated by the instantaneous currents i_a, i_b, i_c in the windings at $\omega_e t = 0$ (see top of Fig. 1-7). Note that the current in the phase A winding is at the maximum positive value - which corresponds to the maximum flux vector, F_{max}, in the direction shown. Figures 1-7a,b show flux vector summations for $\omega_e t = \pi/3$ and $2\pi/3$, respectively.

The net flux vector formed from the vector sum of the individual phase vectors is called the net stator flux vector.

It can be shown that the net stator flux vector has a constant length or amplitude so long as the three sinusoidal stator currents have fixed and equal amplitudes. It can also be shown that the net stator flux vector rotates at the same rate as the electrical angle $\omega_e t$. This can be seen in Fig 1-7 where a 60° step in the direction of the net stator flux vector corresponds to a $\pi/3$ change in $\omega_e t$. This is the case for the 2-pole motor; the net stator flux vector rotates at the same frequency as the driving currents.

In other words, driving the stator windings with the currents described results in a rotating magnetic flux in the interior of the stator. The flux intensity is proportional (up to a point) to the amplitude of the sinusoidal stator currents and the flux rotates at the same frequency as the sinusoidal stator currents.

When the rotor is placed in the rotating flux, it will tend to

align its magnetic axis with the rotating net stator flux vector. Of course, the rotor must turn to maintain alignment. If no torque is applied to the shaft (i.e., no load and no friction), then the rotor flux vector will align exactly with the rotating stator flux vector and the rotor is said to be turning at synchronous speed with torque angle δ equal to zero. If the load torque on the shaft is increased, the angle δ between the net stator flux vector and rotor flux vector (or magnetic axes of each) increases until the alignment torque balances the load torque. If the load torque varies periodically, then δ will change periodically as will the instantaneous motor speed. The average speed of the motor however, will still be the synchronous speed.

Several of the concepts developed in this section are most important in later discussions. These are:

1. The rotor magnetic flux is generated by permanent magnets and is therefore a fixed quantity.
2. The net stator flux is generated by three equal amplitude sinusoidal currents in the stator windings.
3. The net stator flux can be visualized as a vector which rotates at the electrical frequency of the phase currents and whose amplitude is proportional to the amplitude of the phase currents.
4. The rotating net stator flux pulls the rotor at synchronous speed (same as electrical frequency) with a shaft torque proportional to $\sin(\delta)$ where δ is the angle between the stator flux vector and the rotor flux vector.
5. The average speed of the motor is determined solely by the

frequency of the driving currents.

6. Motor torque output is a function of torque angle δ and the amplitude of the stator currents. Motor torque is independent of motor speed.

1.2 Balanced Current Motor Model

The simplest mathematical model that preserves synchronous motor torque production dynamics assumes that the stator windings are driven with balanced (equal amplitude) sinusoidal currents. This approach ignores the inductance of the stator windings and assumes that the driving amplifier can achieve the voltage waveforms needed to create the winding currents given below. (A method for creating these waveforms is given in a later section.)

$$i_a = I_s \cos(\omega_e t) \tag{1-1}$$

$$i_b = I_s \cos(\omega_e t + 2\pi/3) \tag{1-2}$$

$$i_c = I_s \cos(\omega_e t - 2\pi/3) \tag{1-3}$$

Following the usual convention, the lowercase "i's" represent the instantaneous current in each of the phase windings (A, B, and C). The upper case "I_s" is the amplitude of the current waveforms and must be the same for each phase though I_s is not necessarily a constant itself. It can be shown by setting $t = 0$ in Eqs. (1-1,2,3) that the net stator flux vector initially lies along the phase A magnetic axis. This position will be considered the reference point from which rotation of the net stator flux vector is measured. The angle of rotation of the

net stator flux vector is also called the electrical phase angle θ_e. θ_e equals zero when the net stator flux vector is aligned with the phase A magnetic axis. Since $\theta_e = \omega_e t$, it will be convenient later on to find θ_e from the method used to generate the phase A current waveform, given by Eq. (1-1). The electrical angle θ_e is needed in the control algorithm.

As stated earlier, the three stator currents produce a rotating magnetic field on the interior of the stator. The field direction and intensity is conveniently represented by the net stator flux vector. The net stator flux vector has a constant length for a given current amplitude and rotates at synchronous speed. The rotor is pulled around by the rotating field at this same speed. The load torque on the shaft determines the angle δ between the stator flux vector and the rotor flux vector. The maximum torque available for a given current level (I_s) is achieved when the rotor flux vector lags the net stator flux vector by 90°. The maximum torque output of the motor is achieved when $\delta = 90^o$ and I_s is large enough to cause total magnetic saturation. Total magnetic saturation means that further increases in stator current will not increase the amplitude of the net stator flux vector or increase the torque output. Saturation effects less severe than total saturation do not effect the operation of the controller and are ignored.

Fitzgerald's [5] derivation for torque production in a synchronous motor is based on a conservation of energy relation:

Net Energy in (excluding I^2R losses)	=	Energy Stored in the Magnetic Field	+	Mechanical Energy Out

This derivation ignores energy losses from flux leakage and iron reluctance. Neglecting iron reluctance is a good approximation for permanent magnet motors since the air-gap reluctance is very large. (The reluctance of the new high flux density magnets is about the same as air - which doubles or triples the effective air gap.) It is assumed here that flux leakage losses do not significantly change the synchronous motor torque production characteristics.

Fitzgerald derives the equation below:

$$T_{gen} = \frac{P}{2} * i_s * i_r * L_{sr} * \sin \frac{P}{2} \theta_m \qquad (1\text{-}4)$$

T_{gen} = torque output from one phase of the motor

P = number of poles on the rotor

i_s = current in one phase winding of the stator (proportional to stator flux vector)

i_r = current in the rotor (proportional to rotor flux vector)

$L_{sr} \sin P/2\ \theta_m$ = mutual inductance between rotor and stator as a function of θ_m.

θ_m = mechanical displacement of the rotor in radians. See Fig. 1-6.

This equation is simplified by setting P = 2 and lumping i_r and L_{sr} into a single term K_T. L_{sr} is a constant for any given motor and i_r must also be constant since it is proportional to the rotor flux which is constant (being generated by a permanent magnet).

The torques on the rotor arising from each phase are given by:

$$T_A = K_T\, i_a \sin(\theta_m) \tag{1-5}$$

$$T_B = K_T\, i_b \sin(\theta_m + 2\pi/3) \tag{1-6}$$

$$T_C = K_T\, i_c \sin(\theta_m - 2\pi/3) \tag{1-7}$$

where i_a, i_b, i_c are given in Eqs. (1-1,2,3). The total magnetic torque on the rotor is therefore:

$$T_{TOTAL} = T_A + T_B + T_C \tag{1-8}$$

1.3 Overview of Control Strategies

Given the torque equations just developed and the previous description of synchronous motor fundamentals, it is possible to describe the conditions necessary for speed control. It is helpful to visualize motor operation in terms of a revolving stator flux which pulls the rotor flux vector with a torque output that is dependent on (1) the length of the stator flux vector (or stator current I_s) and (2) the angle δ between the two flux vectors. Perfect speed control has been accomplished when the rotor (also rotor flux vector) turns at exactly the desired speed. In order for the rotor to turn at a constant speed,

the motor torque must always equal the load torque. Under these conditions, the net acceleration of the rotor is zero.

Several methods are possible for regulating the torque output of the motor. All of the methods involve varying the length of the stator flux vector or changing the torque angle δ or some combination of the two. Though stator flux magnitude can be varied only by altering the stator current I_s, the torque angle δ can be changed several different ways. These will be discussed next.

Synchronous motors frequently don't have a speed controller but are driven directly from AC line voltage. Stator flux magnitude and speed of rotation are fixed by the line voltage and frequency. In this case, δ is determined entirely by the load torque on the rotor. A large load torque tends to slow the rotor and increase δ until the alignment torque equals the load torque. If the load torque gets smaller, the rotor tends to accelerate to make δ smaller and balance the torques. When δ is changed by a variation in the load torque, the rotor will experience a speed transient while it is "hunting" for the new equilibrium value of δ. The magnitude of the transient speed fluctuation depends on the total inertia of the motor/load system and the amount of damping in the system.

At this point, it is possible to see that varying the stator current I_s can also effect δ. For example, if the load torque is constant and I_s is increased, then the stator flux vector gets longer or equivalently the stator flux field is stronger which pulls the rotor into closer alignment thereby decreasing δ. This may be a very slow method of varying δ if system inertia is high. It takes time for the

rotor to accelerate to the new value of δ.

A third and faster way to change δ is to electrically move the stator flux vector with respect to the rotor flux vector. The stator flux vector can be accelerated by varying the frequency of the stator currents. For example, increasing the frequency of the stator sinusoidal currents will cause the stator flux vector to accelerate up to the higher speed. As the stator flux vector begins to move away from the rotor flux vector, δ is increased. The frequency must be stepped back to its original value to prevent overspeeding the rotor. This method will be referred to as frequency control.

Generally speaking, torque regulation and speed control can be accomplished using one or some combination of the methods discussed.

δ could also be regulated by varying the stator current amplitude alone. For example, given a constant load torque, the current amplitude I_s could be adjusted to achieve any desired steady-state value of δ between 0 and $\pi/2$. The minimum value of I_s possible corresponds to $\delta = \pi/2$. (It is desirable to operate near the minimum value of current if possible so as to reduce heating losses.) Given that the desired value of δ has been established, it can be maintained during a load transient if I_s is varied to meet the changing load. Note that frequency control is not used here. The sinusoidal currents are being supplied at the desired frequency and only the waveform amplitude I_s is being varied to maintain δ and meet the load torque requirements.

To summarize, only the stator current frequency and amplitude can be varied to achieve speed control of the permanent magnet type synchronous motor. Rotor speed is generally dependent on turning the

stator flux vector at the desired speed though not necessarily at all times as in the case of frequency control. Rotor torque may be varied by controlling δ using frequency control, current amplitude, or the load itself. Torque also varies with stator current if the torque angle δ is held fixed. In most applications, δ is controlled by the load. This method is inadequate for the high frequency loads of interest here.

1.4 Overview of Synchronous Motor Drives

This section is intended to give an overview of the methods presently used to drive three phase synchronous motors. Motor drives are distinguished from motor controllers in that motor drives provide the voltage or current waveforms for driving the motor windings while the controller dictates the frequency and amplitude of those waveforms.

Ideally, the three-phase synchronous motor drive would provide three sinusoidal waveforms 120° out of phase at any frequency and amplitude desired. In actual practice, the motor drive outputs are not sinusoidal and different motor drives use different sinusoidal approximations. Four general types of motor drives will be briefly discussed: voltage-fed inverters, current-fed inverters, cycloconverters, and stored waveform drives. Emphasis will be on the stored waveform drive since this is the type used later in the computer simulation.

Voltage-fed inverters create approximations of sinusoidal voltage waveforms by switching silicon controlled rectifiers. Waveforms for two common methods of approximation are shown in Fig. 1-8a,b.

Current-fed inverters also use silicon controlled rectifiers. A

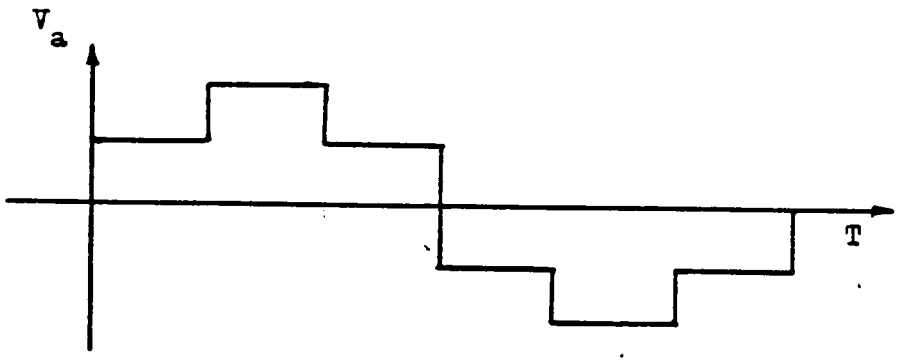


Fig. 1-8a Phase A voltage, three - phase bridge inverter output waveform.

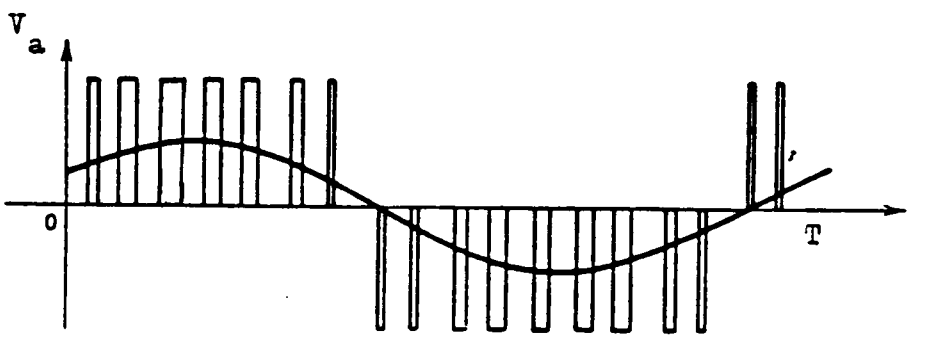


Fig. 1-8b Phase A voltage. Sinusoidal approximation by the pulse - modulation method. Adapted from Dewan [6], p.223.

waveform generated by a current-fed inverter drive is shown in Fig. 1-9a.

The cycloconverter creates lower frequency sinusoidal approximations by "piecing together" portions of three-phase line voltage as shown in Fig. 1-9b. Filtering is used to remove some of the high frequency components.

The stored waveform drive produces a very close approximation to sinusoidal waveforms. In this method, one cycle of a sinusoidal waveform is stored in memory for each of the phase windings. The stored waveforms, which are shown in Fig. 1-10, are phase shifted 120° as required by the motor. The waveforms are replayed from memory and amplified to drive the motor windings. The amplitude can be varied by multiplying the stored value by a gain before amplification. This method will be referred to as "scaling" the amplitude of the sinusoidal waveform. The frequency of the waveforms can be varied by altering the frequency of the clock used to latch the values out of memory. Note that θ_e, the angle of the stator flux vector equals θ_A, the angle of the phase A waveform being replayed from memory. (See Fig. 1-10.) θ_A (and hence θ_e) is known to some accuracy depending on the number of values stored in memory for the phase A waveform. For example, if the phase A waveform were stored in 64 discrete values then θ_e would be known to an accuracy of one part in 64.

This motor drive can deliver either voltage or current waveforms proportional to the stored waveform depending on the type of amplifier used. This motor drive is more expensive but does have the advantage of very smooth torque production and less I^2R losses associated with

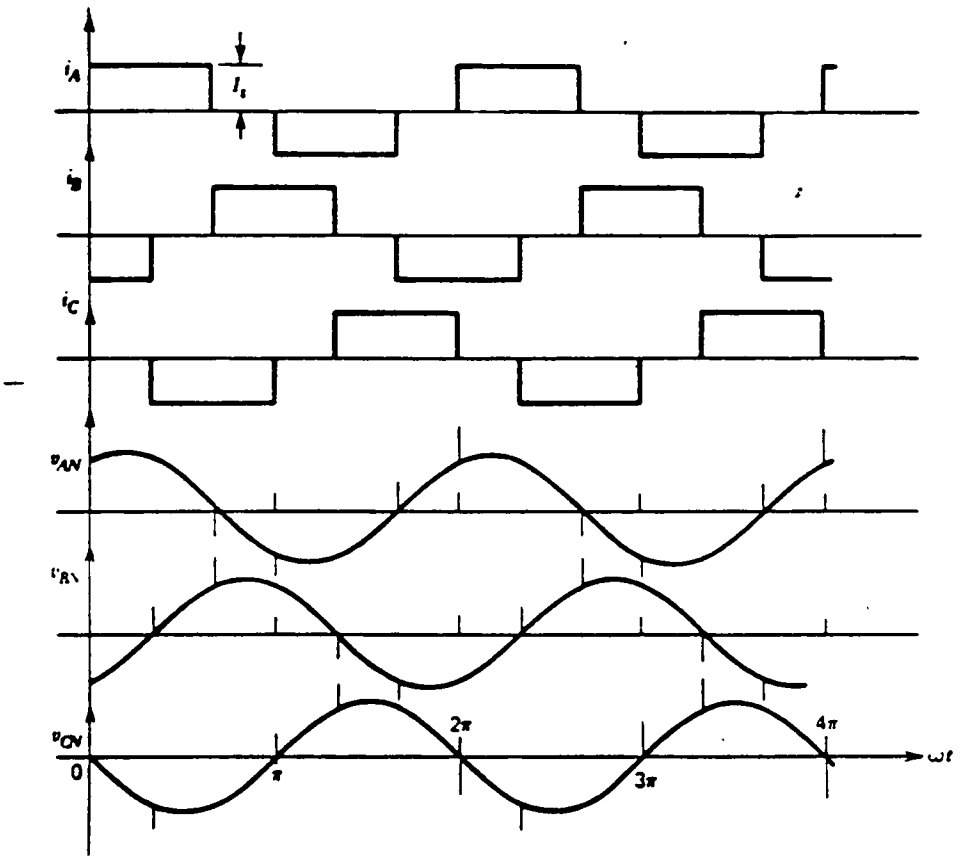


Fig. 1-9a Three phase current source inverter waveforms. Adapted from Dewan [6], p.264.

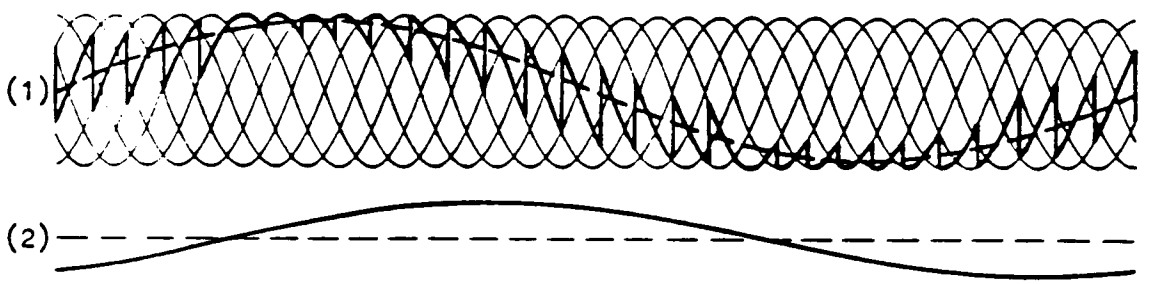


Fig. 1-9b Cycloconverter output waveforms (1) unfiltered output voltage (2) output current. Adapted from Dewan [6], p.327.

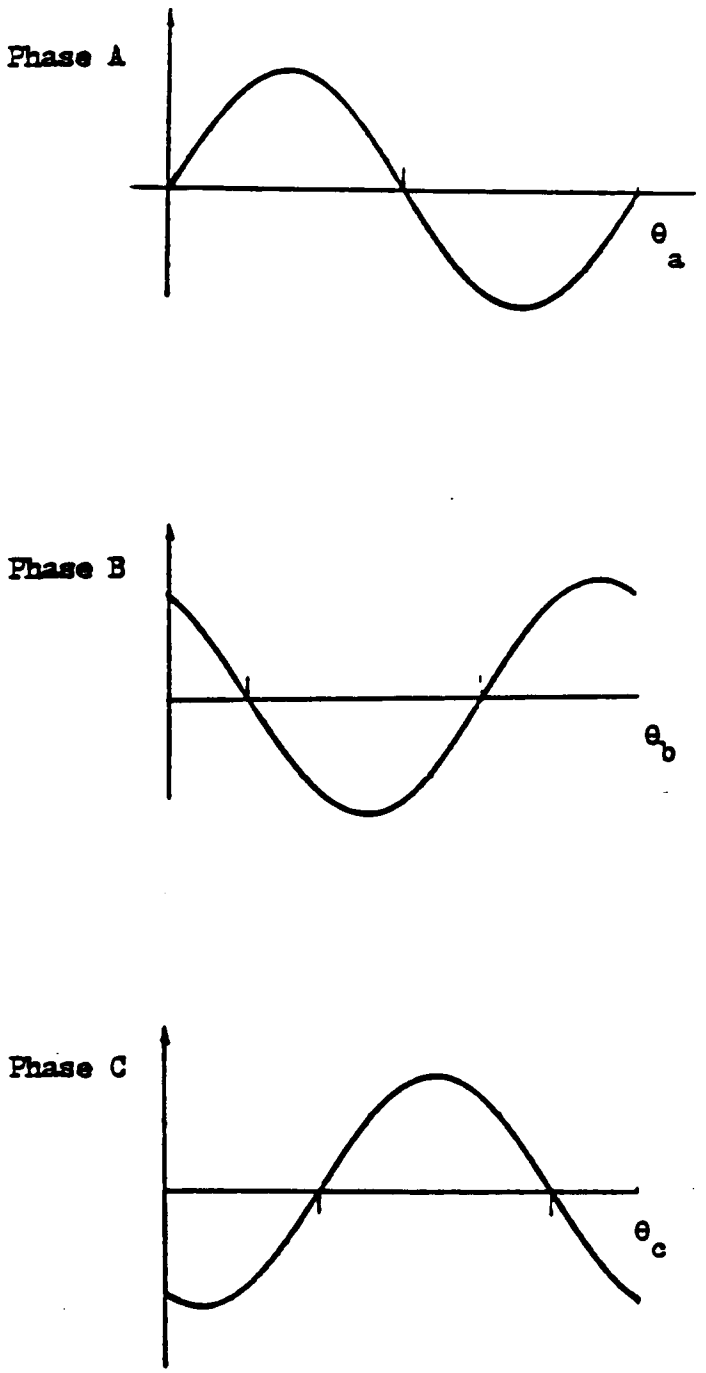


Fig. 1-10 Waveforms stored by a three - phase stored waveform motor drive. Actual curves are stored digitally.

current square waves. It has other advantages, which will be discussed later.

It is important to note again that the motor drive circuit is separate from the motor controller logic. For the inverter drives discussed, the motor controller would determine the timing of the SCR gating signals (i.e., frequency) and the amplitude of the voltage or current fed to the switching network. Likewise, a controller for the stored waveform drive would scale the waveform amplitude and control clock speed for frequency control.

Many different control circuits or algorithms are in use. The literature indicates that the control schemes are typically concerned with controlling frequency, power factor correction, maintaining synchronization by limiting the rate of change of frequency, and occasionally some simple control over the angle δ. In some AC servo applications δ is held fixed so that motor dynamics are very similar to a d.c. motor (see Chapter 2). In most cases, the controller provides the proper amplitude and frequency waveform for power factor and speed control; δ is determined by the load as discussed earlier. For many applications, the speed transients associated with a varying load are small or slowly changing and the present control methods are adequate.

High frequency periodic loads however, tend to drive the motor into large periodic speed fluctuations; i.e., a condition of continual "hunting" for equilibrium. Present control methods are not adequate for this type of load.

1.5 Adaptive Torque Scheduling Controller

This section describes a new method of adaptive speed control as applied to a three-phase permanent magnet synchronous motor. This method extends very accurate speed control to four-bar linkages, slider cranks and other applications where the load torque has high frequency periodic components and to variations in motor parameters, mild saturation effects and thermal effects. This method also adapts to non-periodic slowly varying load components. This section is divided into three parts, the first part describes how the torque schedule controls the motor, the second part describes how the torque schedule is adapted and the third part describes how the schedule may be adapted further to control the torque angle and compensate for slowly varying non-periodic loads.

The Torque Schedule Method

As stated earlier, the controller takes advantage of the fact that the high frequency components of the load torque are periodic. Essentially, the controller creates in memory a schedule of the load torque requirements over one cycle (load cycle) of the lowest frequency periodic load component. For example, a compressor torque load may be approximated by a Fourier series expressing the load torque in terms of rotor position which contains about four or five terms as shown below.

$$T = T_o + T_1 \sin(\theta_m + \phi_1) + T_2 \sin(2\theta_m + \phi_2) + T_3 \sin(3\theta_m + \phi_3) + T_4 \sin(4\theta_m + \phi_4) \ldots \quad (1\text{-}9)$$

The fundamental is the the lowest frequency term and corresponds to one revolution of the compressor drive shaft. The torque schedule must be long enough to store values for this fundamental period. All higher frequency components will then be stored also. The torque schedule must also contain an "offset" term which corresponds to the constant or slowly varying term T_o in Eq. (1-9). A block diagram of the controller, motor and load is shown in Fig. 1-11.

The torque schedule controller considered here drives the stator windings to control the motor speed and torque. The current waveforms output by the controller are:

$$i_a = I_s(\theta_m) \cos(\omega_e t) \quad (1\text{-}10)$$

$$i_b = I_s(\theta_m) \cos(\omega_e t + 2\pi/3) \quad (1\text{-}11)$$

$$i_c = I_s(\theta_m) \cos(\omega_e t - 2\pi/3) \quad (1\text{-}12)$$

These relatively smooth current waveforms are generated using the stored waveform motor drive discussed in Section 1.4. These waveforms allow smooth torque production and accurate speed control down to very low speeds. Figure 1-11 shows how the stored sinusoids for each phase are multiplied by $I_s(\theta_m)$ before amplification. A value of $I_s(\theta_m)$ which corresponds to each point in the load cycle is stored in the torque schedule. In this case, the torque schedule doesn't store torque per se but actually contains values of current which are proportional to torque

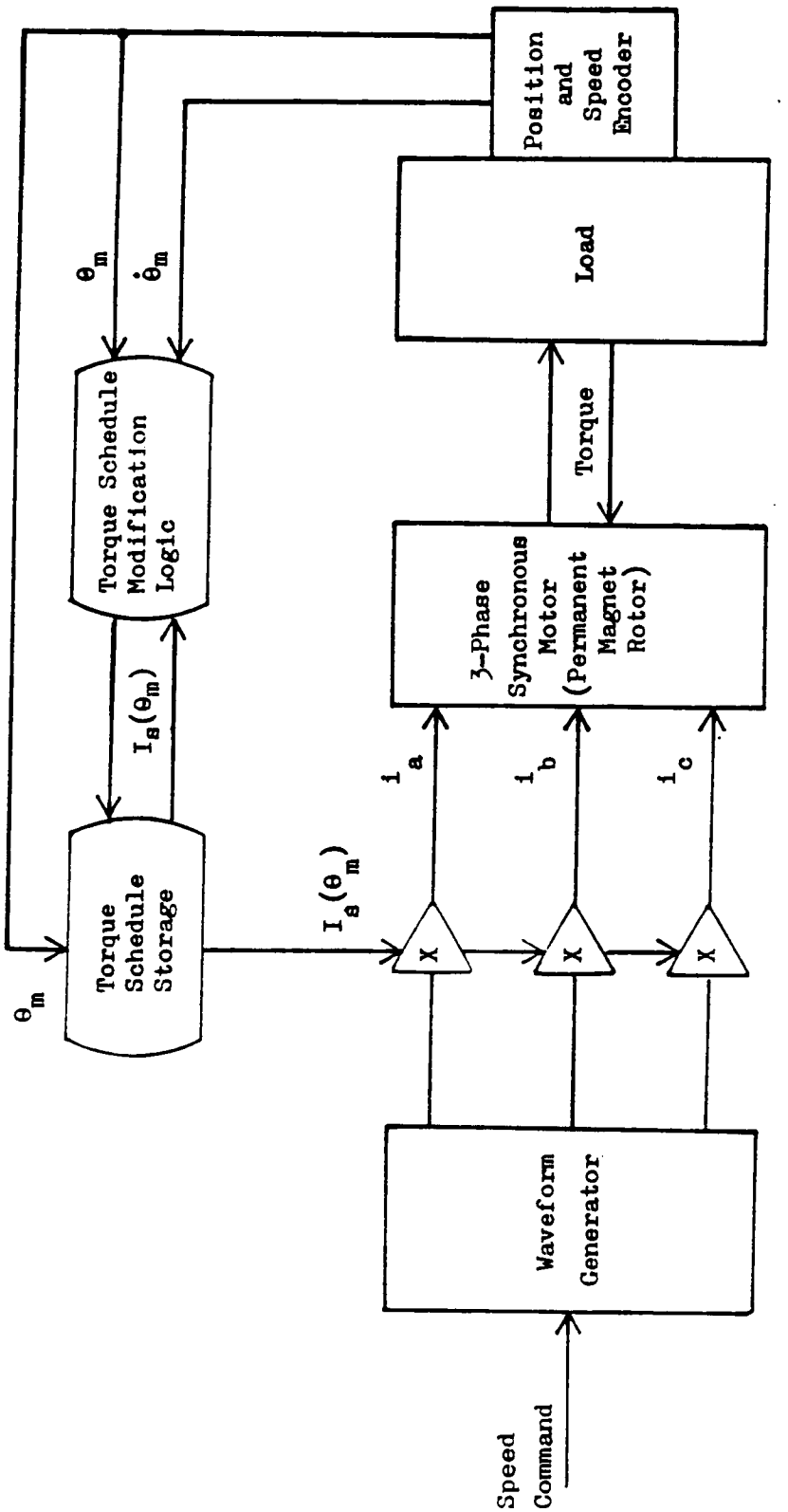


Fig.1-11 Block diagram of Torque Schedule controller and 3-Phase synchronous motor with load.

if δ is fixed. Note that varying $I_s(\theta_m)$ changes only the length of the net stator flux vector. The speed of rotation of the net stator flux vector is fixed by the frequency ω_e. Recall that ω_e is controlled by varying the frequency of the clock used to latch the stored waveforms out of memory.

The most important point here is that <u>if</u> $I_s(\theta_m)$ can be adapted to match the load torque requirements, then the net acceleration on the rotor will be zero. The rotor will be pulled at the synchronous speed by the net stator flux vector and the angle δ between them will be constant. Only the length of the net stator flux vector will be varied to meet the changing torque requirements.

Adapting the Torque Schedule

This section describes a method for adapting a schedule of $I_s(\theta_m)$ values to meet the periodic load torque requirements.

The length of the schedule to be created depends on the load. It is assumed here for convenience that one load cycle occurs over one revolution of the motor. In general, the schedule must match the load cycle regardless of the motor revolutions necessary to drive the load through one complete cycle.

Adapting the torque schedule requires rotor position and speed information. In this case, it is assumed that the rotor is fitted with a 100 increment position encoder and that speed is available in digital form. The encoder must also have a separate index signal which marks the beginning of each revolution.

The encoder divides a motor revolution into 100 increments of

0.06283 ($2\pi/100$) radians each. For each increment, there is a corresponding current value, $I_s(i)$, in the torque schedule used by the stored waveform motor drive to generate the instantaneous currents given in Eqs. (1-10,11,12). In this way, the currents in the stator are weighted at each increment in the revolution by a distinct value stored in memory.

The stored current values can be adapted using the equation below.

$$I_s(i) = I_s(i)\ [1 - K_1 * (S_i - S_{i-1})] \qquad (1\text{-}13)$$

where $I_s(i)$ = i^{th} value of current in memory, discrete equivalent of $I_s(\theta)$.

i = i^{th} increment, $1 \leq i \leq N = 100$ here.

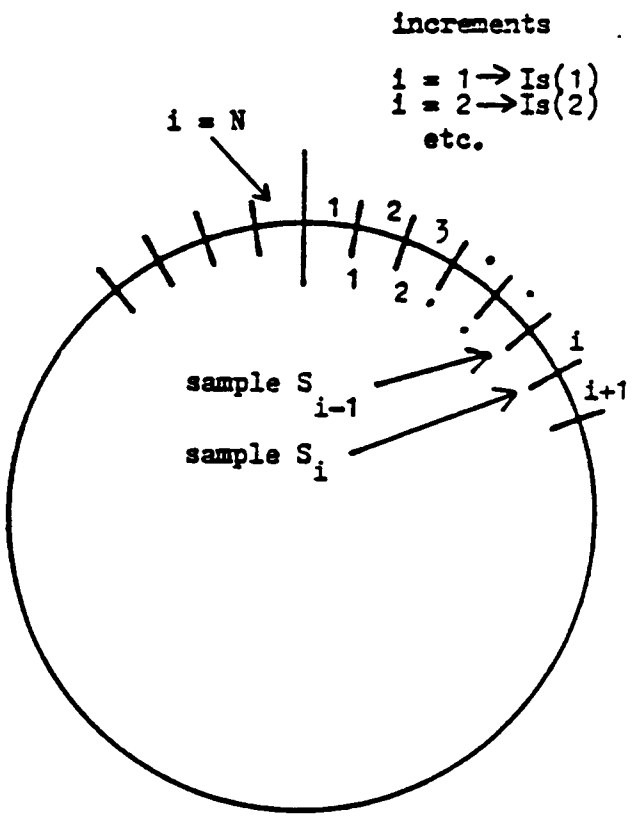
K_1 = adaptation or schedule gain.

S_i = Speed sampled at the end of the i^{th} increment. (See Fig. 1-12.)

S_{i-1} = Speed sampled at the beginning of the i^{th} increment. (See Fig. 1-12.)

Figure 1-12 shows how the increments are labeled and where the speed terms in Eq. (1-13) are sampled.

Equation (1-13) is evaluated at the completion of each increment, when S_i has been sampled. The modified value of $I_s(i)$ is then returned to memory for use during this increment in the next load cycle. The index signal from the encoder marks the start of the next revolution so that the controller can replay the schedule from the beginning.



$\Delta S_i = S_i - S_{i-1}$

ΔS_i = The change in speed over the i^{th} increment.

Fig. 1-12 Diagram of one load cycle. Method of labeling increments and speed sampling points shown.

The values of $I_s(i)$ in the schedule before it adapts must be chosen by the user. The initial values of $I_s(i)$ can be any constant that is large enough to drive the load open-loop. The controller will replay the constant initial value and drive the motor open-loop, until Eq. (1-13) is implemented. With this initial value of $I_s(i)$, the motor should attain average synchronous speed with steady-state periodic speed fluctuation (if the load has periodic torque components). It will be shown next that as the torque schedule adapts, motor speed will approach a constant (synchronous speed) and δ will approach some constant value between 0 and $\pi/2$.

The manner in which the torque schedule controls speed and torque angle δ can be seen by considering a simple example. Given that the rotor is loaded with a sinusoidally varying torque, the rotor speed will fluctuate at the load frequency as shown in Fig. 1-13.

By examining eq. 1-13 it can be seen that the largest adaptive control action is taken when the difference in speed over an increment, $S_i - S_{i-1}$, is maximum. It is apparent from Fig. 1-13 that the largest control action is taken in the increment where motor speed slope is maximum (increment #1 in Fig. 1-13). Note also that almost no control action is taken over increment #2 in Fig. 1-13 since $S_i - S_{i-1}$ is nearly zero.

Increment #1 can be examined more closely in Fig. 1-14 where the change in velocity is shown to be linear. The constant net torque that produced this change in speed is shown in Fig. 1-14. Since $S_i - S_{i-1}$ is large in this increment, a large control action will be taken to reduce the net torque. (In this example, $I_s(i)$ is increased to counter the

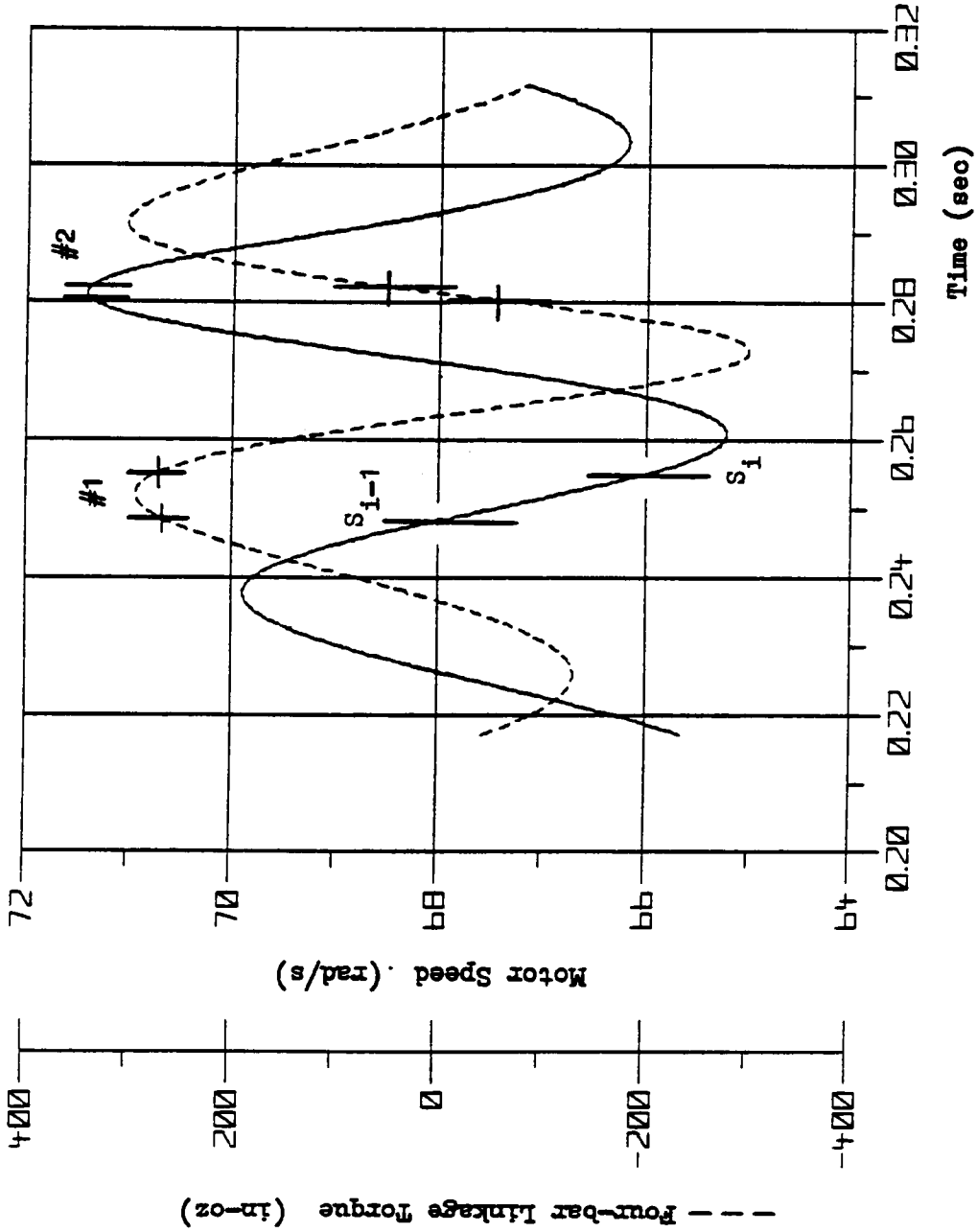


Fig. 1-13. Plot of sinusoidal load torque with corresponding speed response.

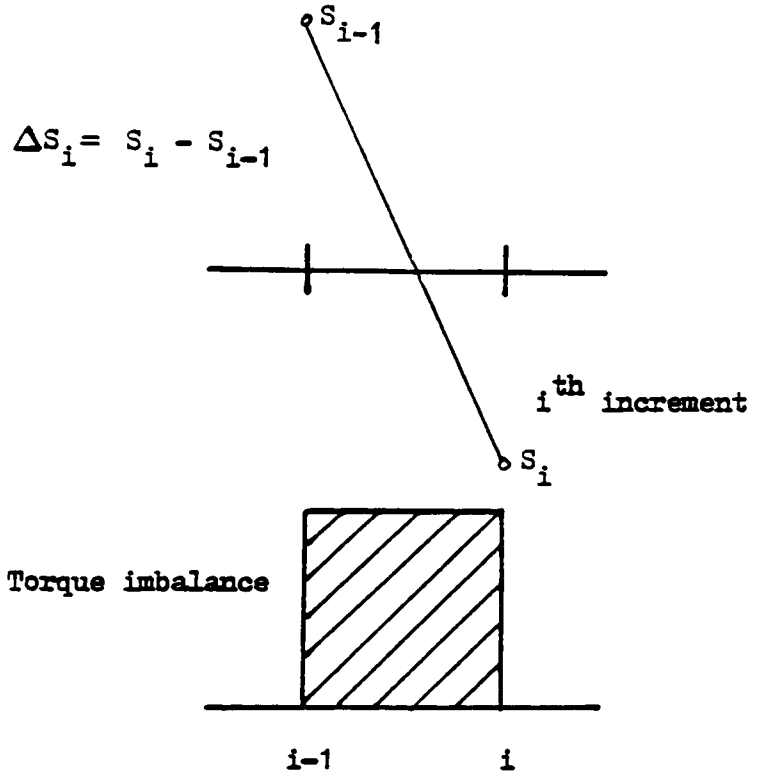


Fig.1-14a Torque imbalance with corresponding ΔS_i for the i^{th} increment.

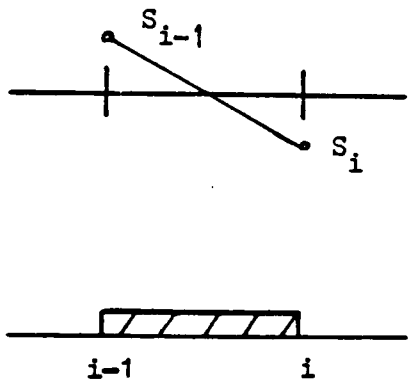


Fig.1-14b Torque imbalance and ΔS_i reduced by lowering motor current in this increment.

load torque.) Reducing the net torque reduces the change in speed over the increment. Therefore, Eq. (1-13) adjusts the current in every increment to reduce the slope or rate of change of speed (if it is non-zero). The net effect of reducing the slope is that the sinusoidal variations in speed and δ collapse onto their respective mean values. Equation (1-13) does not affect the synchronous speed or the mean value of δ. Equation (1-13) acts only to reduce changes in speed and hence δ is fixed in the process.

Controlling the Steady-State Value of δ

This section describes how the torque schedule can be further adapted to control the steady-state torque angle δ and compensate for slowly-varying non-periodic loads.

Recall that δ is the angle between the rotating stator flux vector and the rotor magnetic axis:

$$\delta = \theta_e - \theta_m$$

where θ_e = the electrical angle of the net stator flux vector - found as described on pages 18,19.

θ_m = mechanical or rotor angle as measured by the position encoder.

The relationship between rotor and stator flux is analogous to a nonlinear spring-mass system. The rotating stator flux pulls the rotor mass via a magnetic spring. The spring stiffness or spring constant

corresponds to field intensity (i.e., I_s or net stator flux vector length). From the mechanical engineers perspective, controlling δ is achieved by varying the "spring constant" or stiffness of the magnetic field to maintain the rotor in its desired position relative to the stator regardless of the load placed on the rotor. The desired position (or angle δ) is $\delta = \pi/2$ since maximum torque is achieved using the minimum current (or minimum spring constant). It is essential that δ not exceed $\pi/2$ since this corresponds to "breaking the spring" or loss of synchronization.

The purpose here is to maintain some explicit control over δ to help maintain synchronization, minimize current consumption, and prevent δ from drifting due to slowly-varying non-periodic loads.

The steady-state value of δ is proportional to the average value of current stored in the schedule. Increasing the mean level of current decreases δ. Decreasing the mean level of current increases δ. Care must be taken that the control action for correcting the steady-state value of δ does not inhibit the control action of Eq. (1-13), which regulates the instantaneous value of δ. For example, there may be some increment in the cycle which, because of the load, requires more current but at the same time the instantaneous value of δ may be too small which indicates the current must be reduced. The net effect may be to reduce the current and cause the net acceleration over the increment to increase rather than decrease. This can be eliminated several different ways.

One way is to implement Eq. (1-13) until δ converges on some steady value and then implement a second stage of control which then varies the

mean level of the schedule already adapted.

Another method, the one used here, is to average δ over each load cycle and alter the mean value of the schedule based on this average value of δ. This allows δ to be controlled once per motor revolution without affecting the control action of Eq. (1-13). The modified equation for adapting the torque schedule is given below.

$$I_s(i) = I_s(i) \; [1 - K1 * (S_i - S_{i-1})] + I_{OFFSET} \tag{1-14}$$

where

$$I_{OFFSET} = - K2 * AVGDE \tag{1-15}$$

$$AVGDE = \frac{\sum_{i=1}^{N} (\delta_i - \text{desired } \delta)}{N}$$

N = number of times δ is sampled per load cycle

Note that AVGDE is the average error in δ over one load cycle based on N measurements of δ over the load cycle.

In summary, the torque schedule adaptation equation has two parts. The first term, corresponding to Eq. (1-13) is used to eliminate high frequency fluctuations in speed and δ and contains no explicit control of the steady-state value of δ. The second term, I_{OFFSET}, is used to adjust the mean of the current values stored in memory to control the steady-state value of δ and compensate for slowly-varying non-periodic changes in the load torque.

1.6 Performance Evaluation-Discussion

In this section, the performance of the torque schedule controller as applied to a three-phase synchronous motor is evaluated. The motor, controller and load were simulated using a fixed step, fourth-order, Runga-Kutta integration routine. A commented version of the simulation program may be found in the Appendix.

Two load cases will be examined to compare the performance of the torque schedule controlled motor with a synchronous motor driven open-loop (where δ is determined by the load).

Load Case I - Viscous Damping

Load Case I is a viscous damping load which contains no high frequency components. This type of load is easily driven at synchronous speed and constant δ by either motor configuration. The open-loop motor can only drive this load at $\delta = 0.8$ with a stator current of 10A. The schedule values of the controlled motor however, are modified (reduced) once per revolution by the I_{OFFSET} term to reduce current and increase δ. The first step reduction in current can be seen at $t = 0.3$ sec. in Fig. 1-15. This step change in current produces the "hunting" transient discussed earlier. The schedule adapts to damp out these transients. The net effect is a reasonably smooth transition from $\delta = 0.73$ to $\delta = 1.25$ and from $I_s = 10$ A to $I_s = 7$ A.

Figure 1-16 shows the system response when the I_{OFFSET} gain K2 is increased from K2 = 0.7 to K2 = 1.0. Overshoot and settling time have increased as might be expected.

The control action of the I_{OFFSET} term is essentially that of an

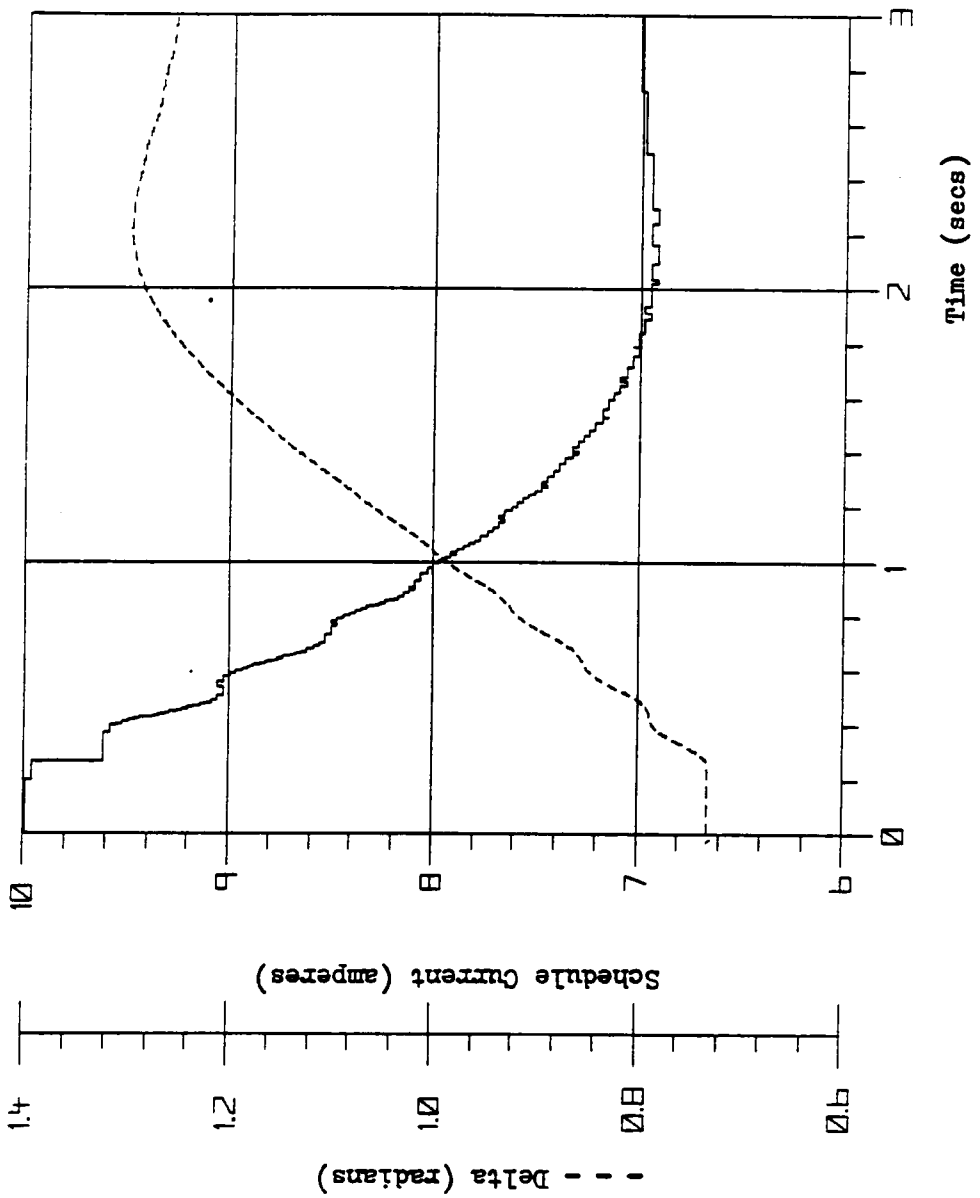


Fig. 1-15 Load Case I. K1=2, K2=0.7 and NPC=100. Plot of reduction in schedule current to drive delta to 1.25 rad.

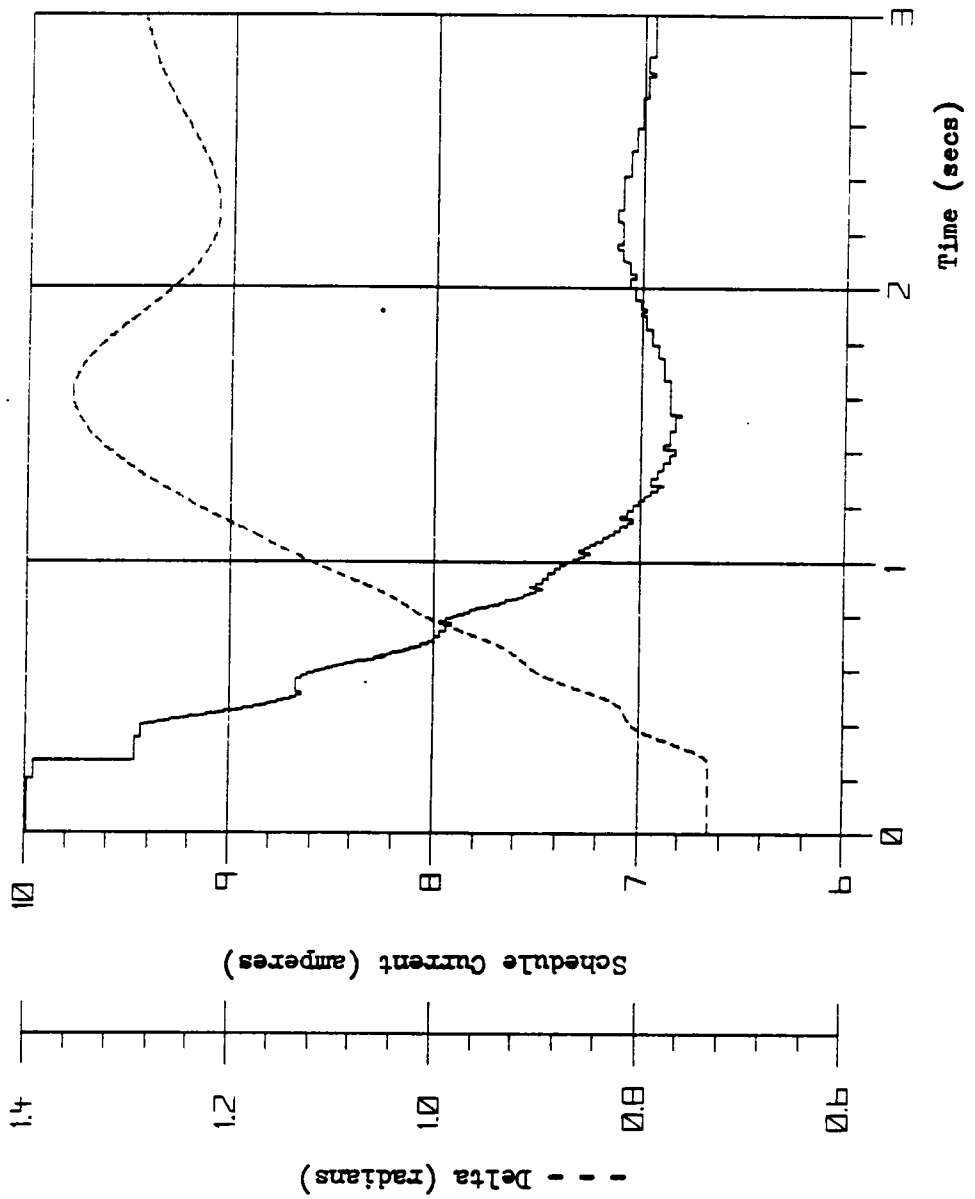


Fig. 1-16 Load Case I. K1=2, K2=1.0, and NPC=100. Plot of reduction in schedule current to drive delta to 1.25 rad.

error integrator. Gain K2 regulates the rate of integration. Unfortunately, the optimum value of K2 is a function of δ. It can be shown that when δ is small, $\Delta\delta/\Delta I_s$ is small. When δ is large, $\Delta\delta/\Delta I_s$ is large. For example, if K2 is chosen too large then the rate of integration will be adequate until δ gets large and then the rate of integration will be much too great; resulting in overshoot and perhaps loss of synchronization. If K2 is chosen small enough to ensure small overshoot then controller response is slow initially.

Load Case II - Four-bar Linkage

Load Case II is a four-bar linkage. The periodically varying torque requirements and inertia are calculated from an influence coefficient model used by S. Carlson [3]. The dominant features of the load are the centrifugal force term which is proportional to motor speed squared and varies at twice the fundamental frequency and the load torque variations from very large positive values to large negative values in each motor revolution. A plot of the theoretical torque requirements plus a 200 in-oz (1.41 N-m) friction load are plotted in Fig. 1-17. (A breakdown of the components of the four-bar linkage load is given in Table 1, Chapter 2.)

In this load case, the individual stored increment values and the I_{OFFSET} term simultaneously reduce the periodic speed fluctuations and drive the torque angle δ to 1.25 radians.

Speed and δ for the open loop motor are shown in Figs. 1-18,19. The steady state speed error is ±6% and the mean value of δ is 0.87 radians with appoximately ±0.07 rad. fluctuation.

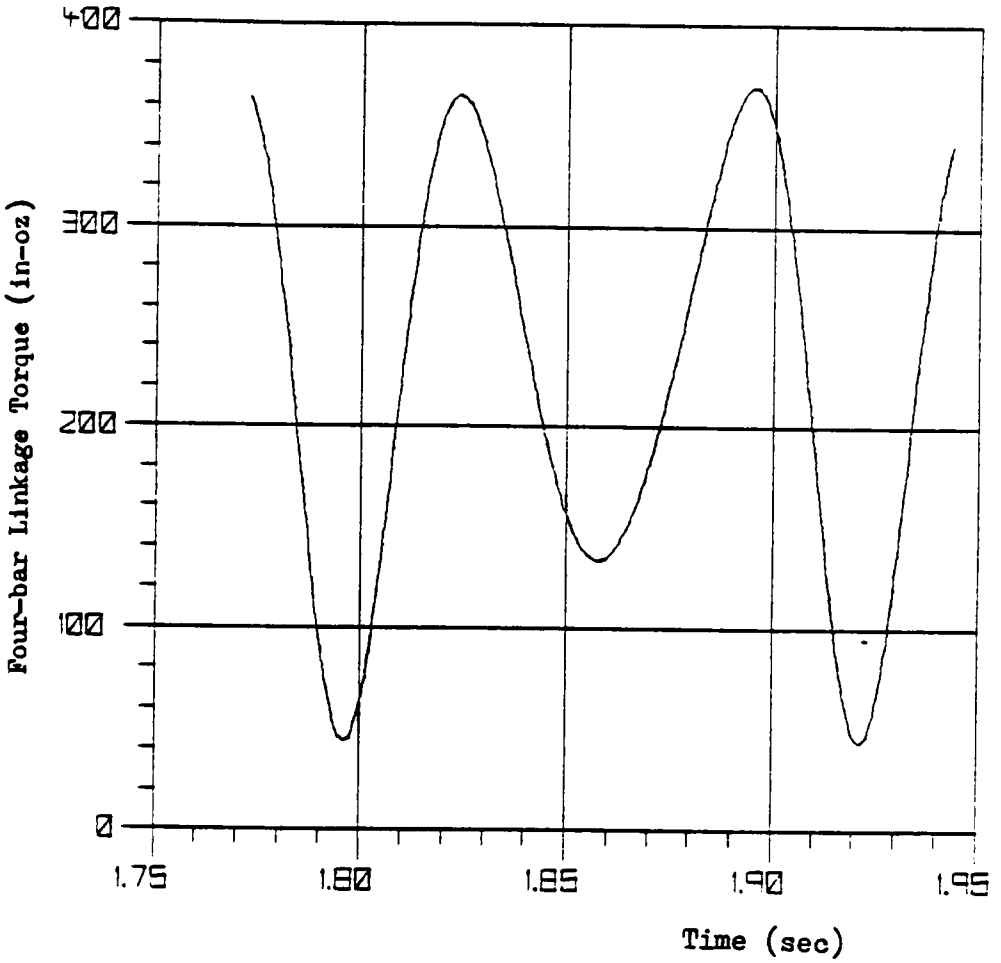


Fig. 1-17 Theoretical four-bar linkage load torque for a constant speed of 50 r/s (includes friction torque).

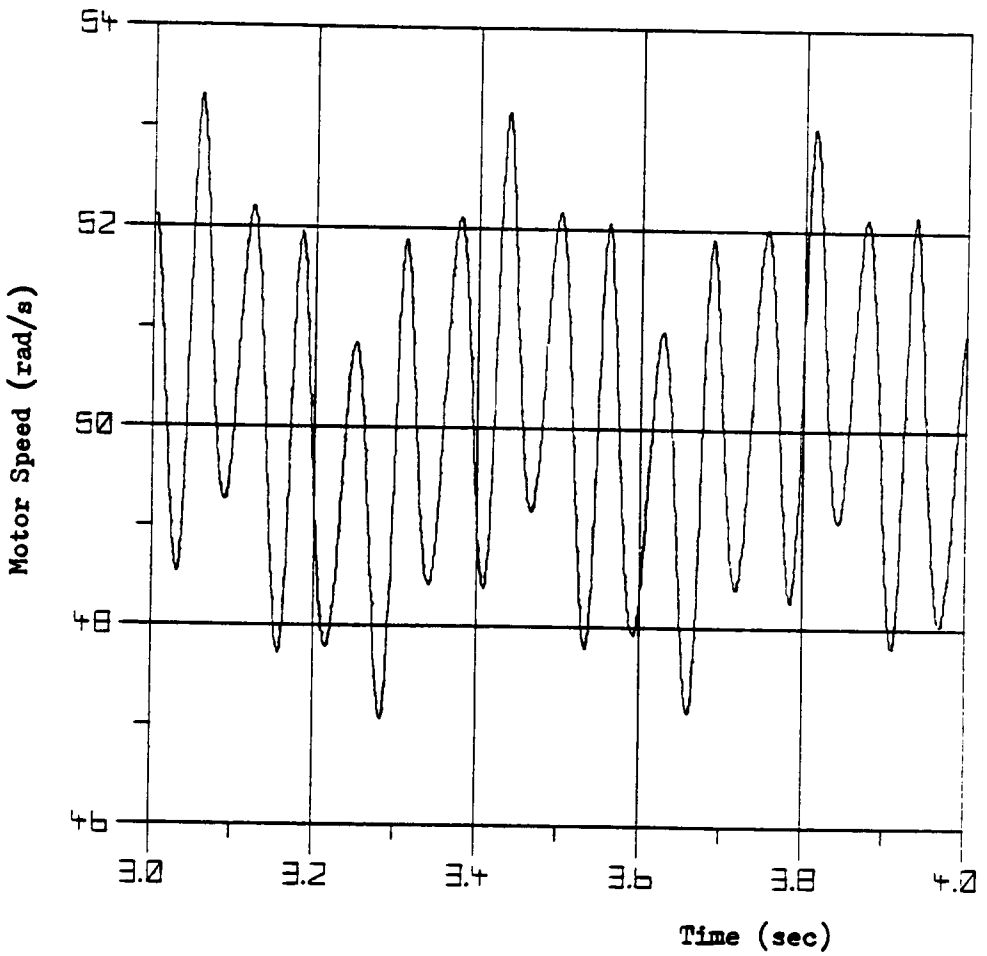


Fig. 1-18 Steady-state speed response of open-loop three-phase synchronous motor to four-bar linkage load.

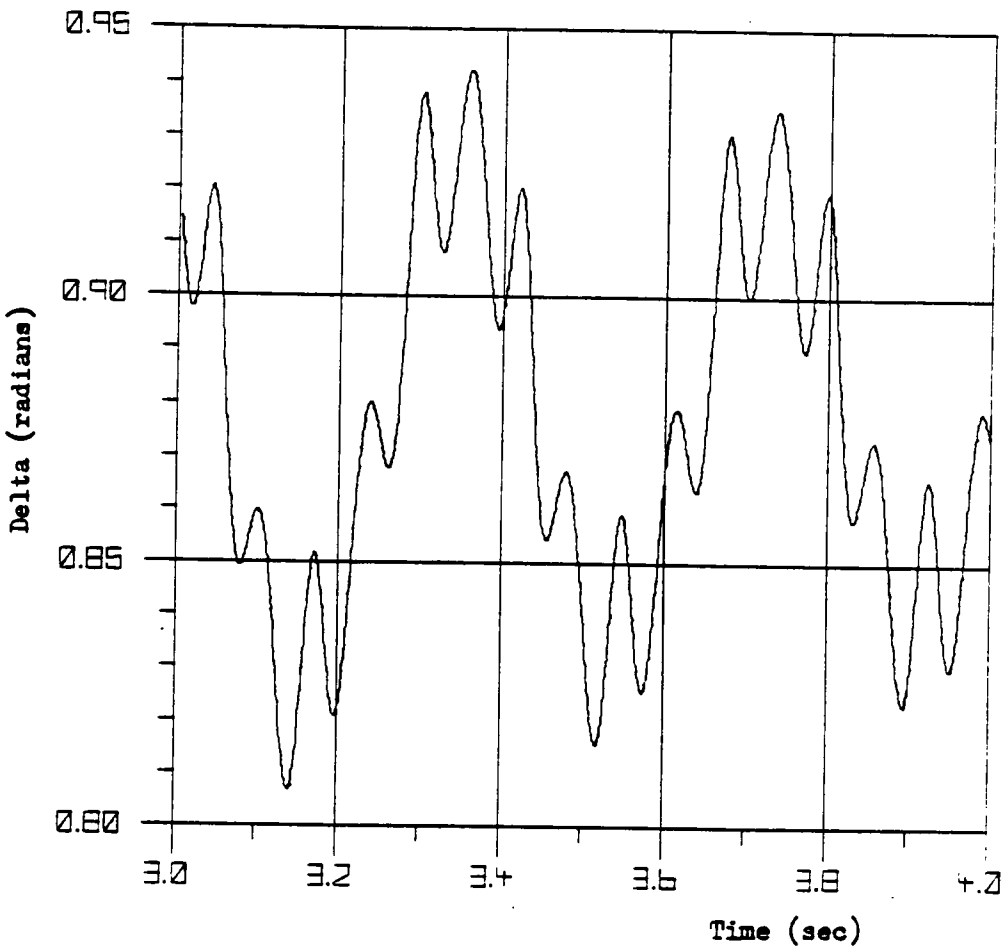


Fig. 1-19 Steady-state delta response of open-loop three-phase synchronous motor to four-bar linkage load.

Figure 1-20 shows the speed response while the schedule is adapting. The schedule currents were able to reduce the periodic speed error to .01% after 8 adaptation cycles. Speed transients from controlling δ with I_{OFFSET} are less than 0.06% at t = 3 secs. Figure 1-21 shows the response of δ over the same time period. Figure 1-22 shows the progression of the schedule as it adapts. Figure 1-23 shows a close-up of the adapted schedule. Note the similarity between Fig. 1-17 and Fig. 1-23.

The torque scheduling controller and current amplifier deliver a current for each increment of the load cycle. This arrangement is simpler than the voltage schedule (which is examined in the D.C. motor chapter) because here the torque is modified only within the desired increment; i.e., there is no current lag or "overflow" into the next increment. Therefore, selecting the number of increments per cycle (NPC) is less critical. If NPC is relatively large (greater than 100 for this case), the adaptation rates for each increment are slowed, assuming K1 is fixed, since the change in speed (ΔS_i) is smaller.

Quantization error in the measured speed and in the schedule of stored values was reflected in the speed accuracy obtainable. Cases were run for schedules with 8-bit storage corresponding to a ±15A range (30/256 = 0.1172 A/step resolution) and for 12-bit storage for a ±15A range (30/2048 = 0.0146 A/step resolution). The speed accuracy for the 8-bit storage was ±0.1% and ±0.01% for the 12-bit storage.

The work in this chapter has demonstrated one way in which the torque schedule controller might be used to drive a synchronous motor. Later in the D.C. motor chapter, there is a discussion of the simple

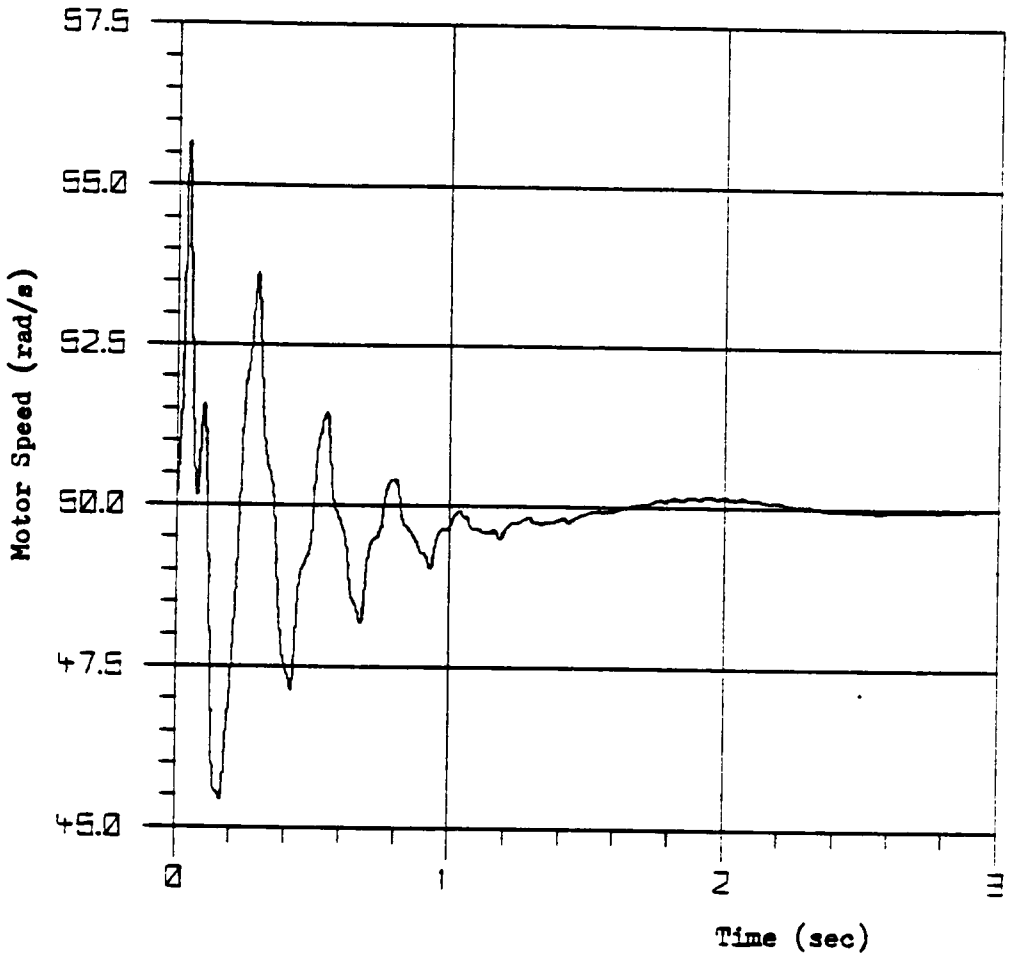


Fig. 1-20 Speed response of torque schedule controlled three-phase synchronous motor (K1=2, K2=0.7, NPC=100). Load Case II.

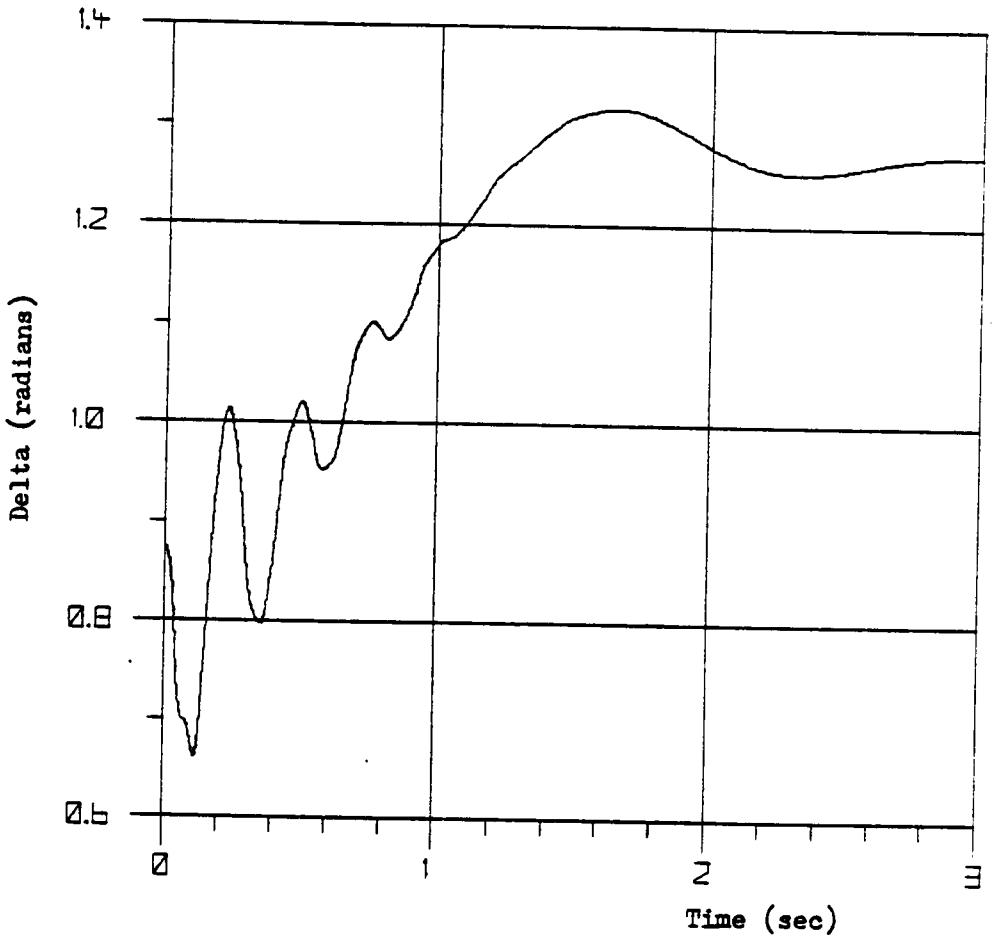


Fig. 1-21 Delta response of torque schedule controlled three-phase synchronous motor (K1=2, K2=0.7, NPC=100). Load Case II.

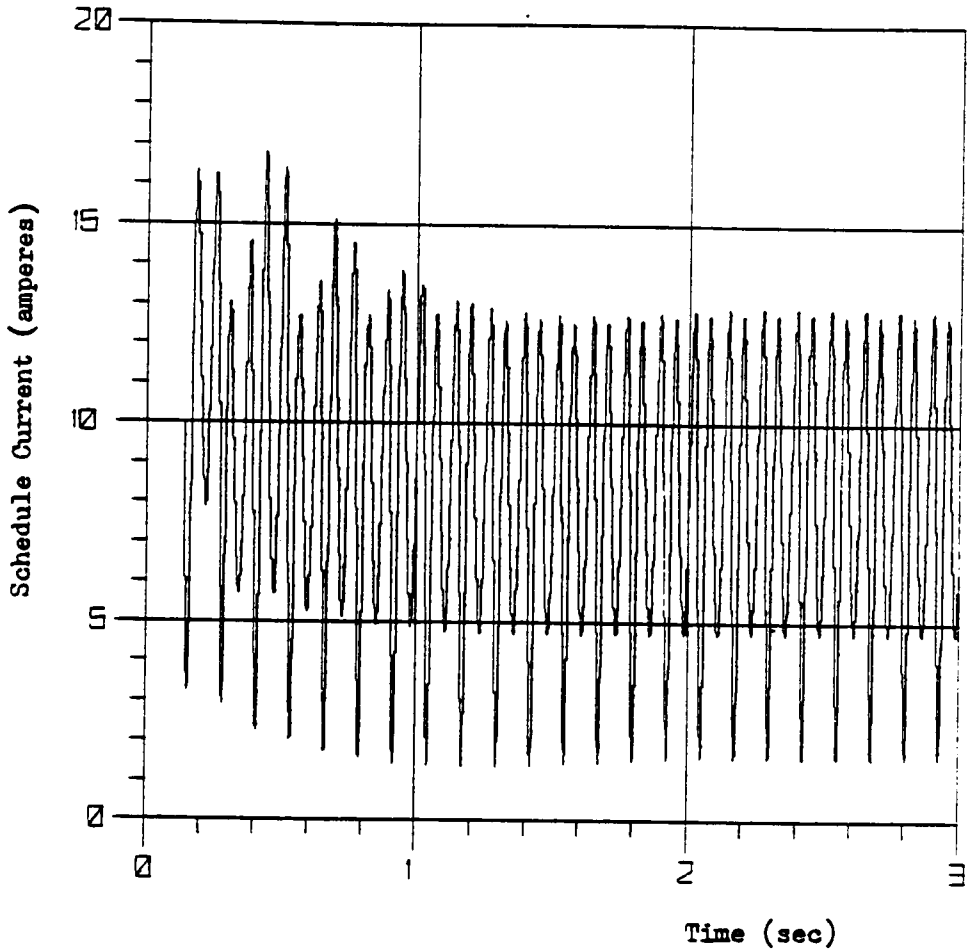


Fig. 1-22 Plot of the torque schedule adapting to the four-bar linkage load (Load Case II) with K1=2, K2=0.7 and NPC=100.

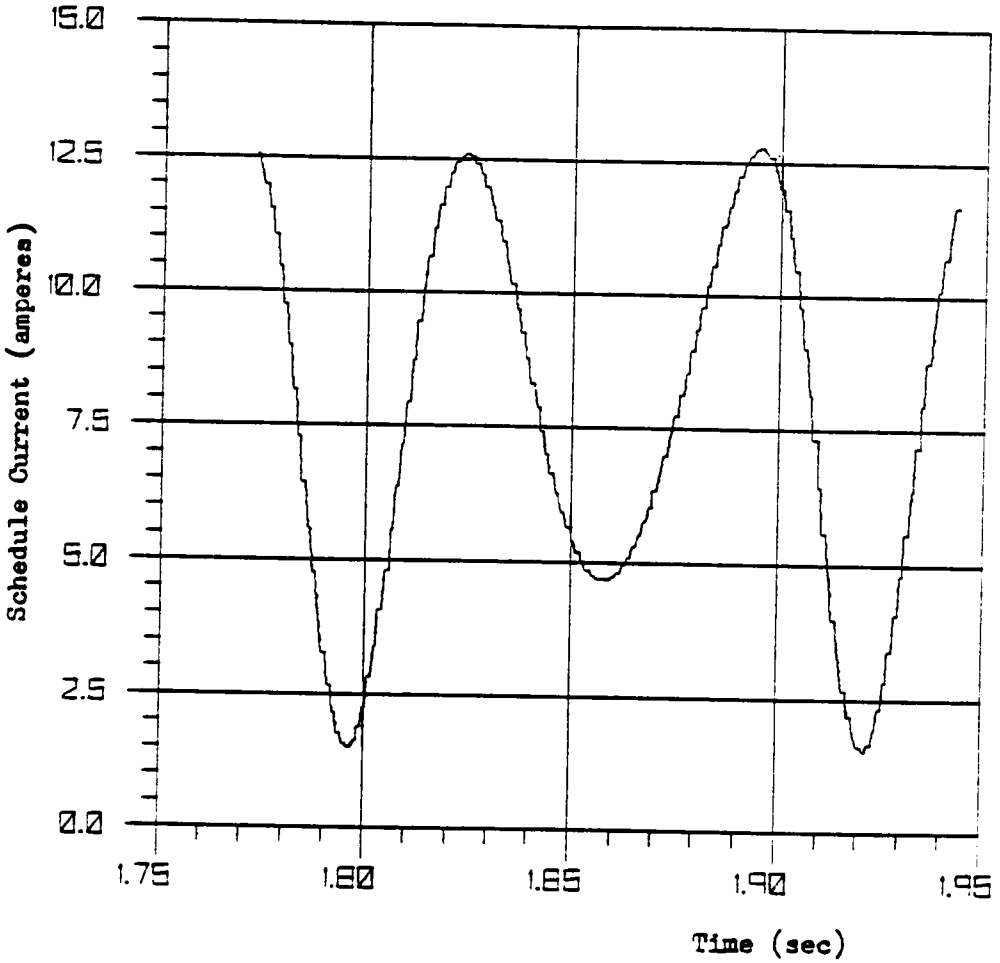


Fig. 1-23 Plot of the adapted torque schedule. Load cycle spans t=1.78 to t=1.91 sec. Schedule values are 12-bit.

changes that would likely be made in order to drive the three-phase synchronous motor (using the same hardware) as a brushless D.C. motor and eliminate transient "hunting" and the possibility of loss of synchronization. When the synchronous motor is driven this way, it is dynamically similar to the permanent magnet D.C. motor, which is to be discussed next.

2.0 INTRODUCTION

This chapter describes the application of the torque scheduling control system to a permanent magnet d.c. motor. The controller varies armature voltage to achieve speed control. The torque schedule controller could also be used to control speed via the field flux in d.c. motors with separately excited field windings.

A brief preview of the material in this chapter is given below.

2.1 D.C. Motor Model Equations - contains the derivation of the electrical and mechanical equations which describe the motor and load.

2.2 General Methods of Speed Control - presents an overview of feedback speed control techniques and explains why these methods are inadequate for controlling the high frequency loads of particular interest here.

2.3 D.C. Torque Schedule Controller - explains how the torque schedule controller can be used with proportional or error squared feedback control to reduce periodic speed fluctuations.

2.4 Performance Evaluation and Discussion - presents the results of simulation studies which compare the performance of P, PI, and PID controllers with the torque schedule controller for two load cases. A discussion of conditions for stable controller operation and gain selection completes this chapter.

2.1 D.C. Motor Model Equations

A permanent magnet d.c. motor can be modeled by a pair of coupled differential equations.

The equation which describes the electrical characteristics is

usually derived using Kirchoff's voltage law on the motor equivalent circuit, see Fig. 2-1. A summation of the voltages in the circuit yields:

$$V_T = i_a R_a + L_a \frac{di_a}{dt} + K_e \dot{\theta}_m \quad . \tag{2-1}$$

where V_T - terminal voltage

i_a - instantaneous armature current

R_a - armature resistance (includes terminal resistance)

L_a - armature inductance

K_e - motor back-emf constant

$\dot{\theta}_m$ - rotor speed

The equation describing the mechanical characteristics of the motor and load are found from a summation of the torques applied to the rotor.

$$\Sigma T_{rotor} = J_T \ddot{\theta}_m$$

$$J_T \ddot{\theta}_m + B \dot{\theta}_m = K_T i_a - T_L \quad . \tag{2-2}$$

where J_T - lumped inertia of the rotor and load

$\ddot{\theta}_m$ - rotor acceleration

B - lumped viscous damping coefficient for the motor and load

$\dot{\theta}_m$ - rotor speed

K_T - motor torque constant

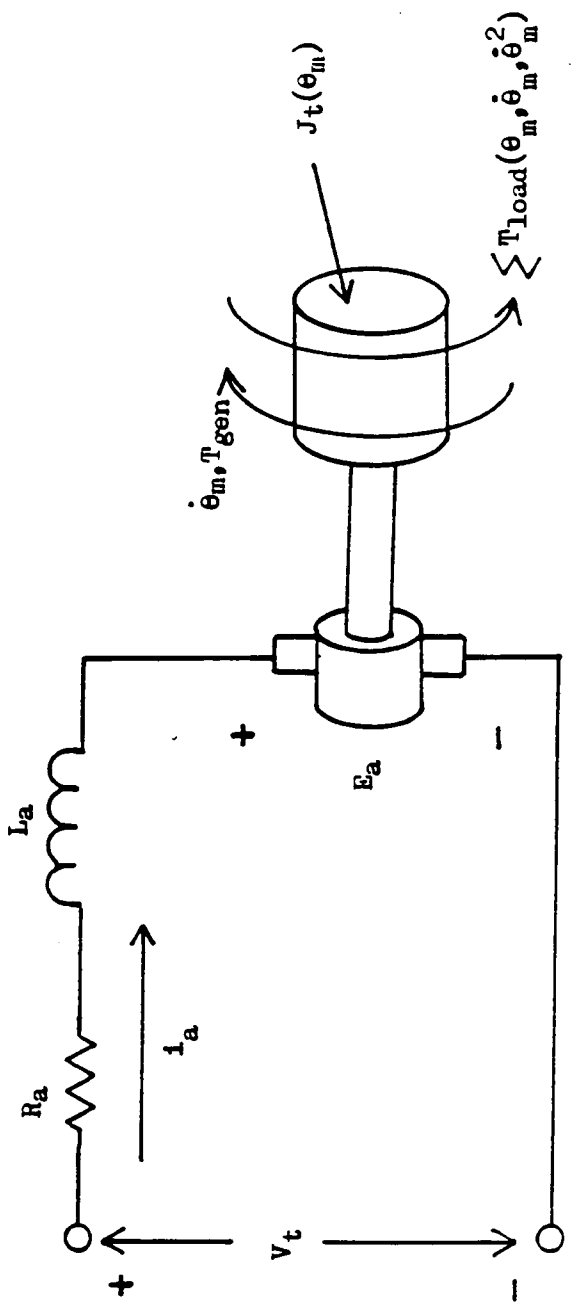


Fig. 2-1. Simplified schematic of permanent magnet d.c. motor and load.

i_a - armature current

T_L - this term includes any other torque load that might be applied to the motor.

2.2 General Methods of Speed Control

A variety of speed control methods were discussed briefly in the literature review. The P, PI and PID control methods have been chosen for further discussion because of their popularity, simplicity and because most non-adaptive controllers operate on similar principles. These controllers can therefore demonstrate the advantages and limitations of each control principle in dealing with the loads of interest here.

Proportional control is simple but requires very high gains to reduce speed droop and satisfactorily compensate for large fluctuations in load torque. Proportional control with high loop gain is susceptible to noise, instability in higher order systems and cannot compensate for changes in load level, speed level or motor parameters. Error squared feedback and bang-bang control are similar in that neither can adapt to changing conditions.

Proportional-Integral (PI) control eliminates speed droop without resorting to high loop gain but the integrator cannot reduce periodic speed fluctuations. This can be seen by considering a sinusoidal speed fluctuation around the speed command value. The integrator sums speed error over the negative half cycle of the sinusoid and delivers a large signal to increase motor speed. The load, however, is decreasing and the motor is about to overspeed on its half cycle excursion above the

command speed. The positive speed error is integrated and cancels exactly the error from the negative half cycle. The integration signal is therefore reset to zero at the end of each load cycle. The net effect is a reduction and centering of the speed fluctuation on the speed command level but no elimination of the periodic speed error.

The PI plus derivative or PID controller can reduce the speed fluctuations. The derivative of speed (or speed error) is acceleration (negative acceleration) and derivative feedback tends to limit system acceleration. This control action introduces a compromise between rapid transient response and reduction of periodic speed fluctuations. As with all fixed gain non-adaptive controllers the PID controller offers no compensation for large changes in load level, speed level or motor parameters. PID control requires the measurement or derivation of three state quantities and is very similar to state-feedback in this sense.

The performance evaluation section compares the performance of the above control schemes to the adaptive torque scheduling controller. Some of the controller limitations discussed will become more apparent from simulation data.

2.3 D.C. Torque Schedule Controller

Speed control of the permanent magnet d.c. motor is much simpler than for the three-phase synchronous motor. The torque angle δ is fixed near 90° by commutator action and only one power amplifier is needed to drive the armature winding.

The torque schedule controller operates together with an analog proportional or an error squared (e^2) feedback loop to control the

motor. Figure 2-2 shows a block diagram of the controlled system. The analog feedback loop allows for fast transient response to changes in the command speed or non-periodic changes in the load. Integration feedback is included in the torque schedule control action and drives the average speed error to zero.

The torque schedule operates in a similar manner to that discussed in Chapter 1. The adaptation equation is slightly different and the initial values in the schedule are zeroes. The equation below was used to adapt a voltage schedule, though could be used to adapt either a current schedule or a voltage schedule.

$$VML(i) = (VML(i) - K_s * \Delta S_i) - K_v * V_{OFFSET} \qquad (2\text{-}3)$$

where
- $VML(i)$ - stored voltage for the i^{th} increment of the load cycle
- K_s - schedule gain which regulates the rate of schedule adaptation
- ΔS_i - same as before, the change in speed over the i^{th} increment of the load cycle.
- K_v - integrator gain
- V_{OFFSET} - speed error integrator term defined below

The V_{OFFSET} term is defined as

$$V_{OFFSET} = \frac{1}{2\pi} \int_0^{2\pi} (S_{pdcom} - \dot{\theta}_m)\, d\theta_m \qquad (2\text{-}4)$$

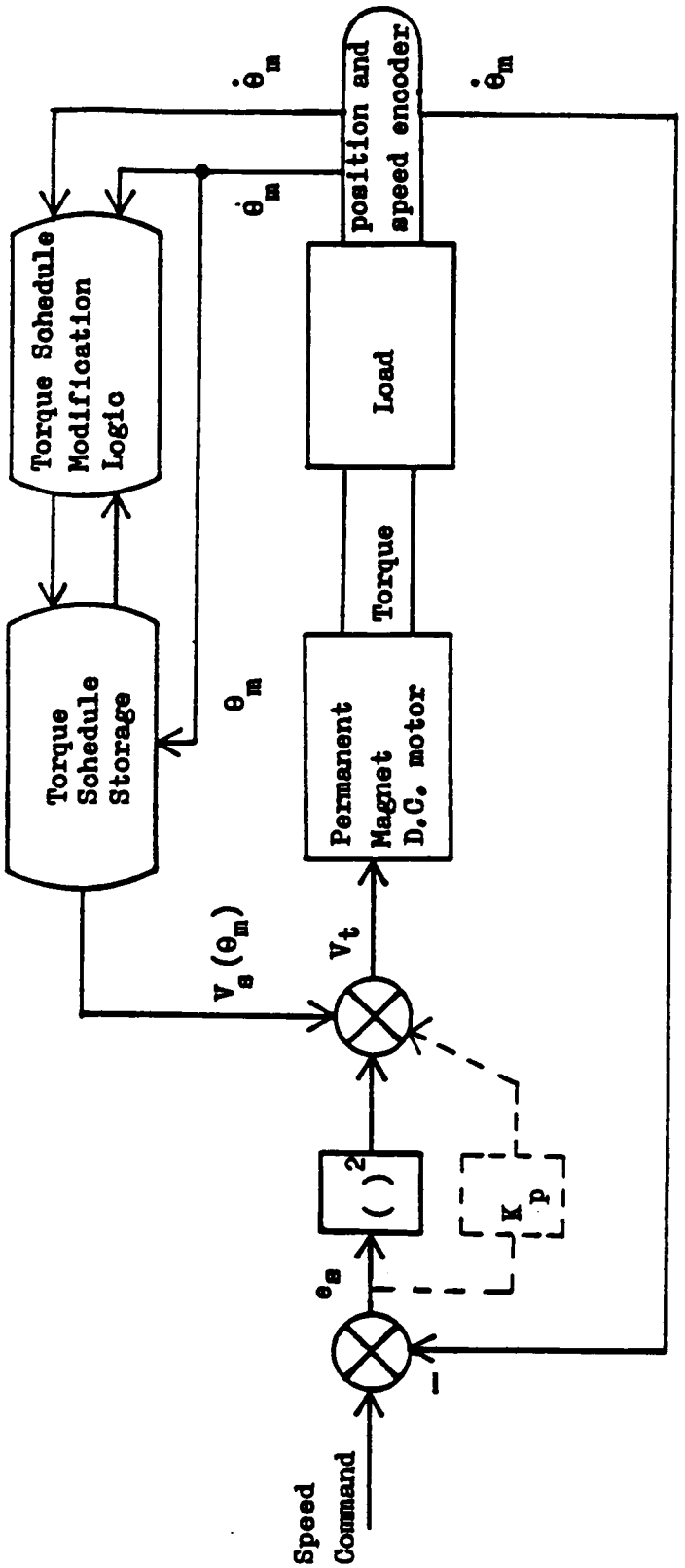


Fig. 2-2 Block diagram of the torque schedule controller and permanent magnet d.c. motor. Error squared or proportional control can be used in the forward loop.

$\dot{\theta}_m$ - motor speed

S_{pdcom} - speed command

V_{OFFSET} integrates the speed error over each load cycle and therefore represents the mean speed error of the cycle. V_{OFFSET} modifies the schedule to drive the mean speed error to zero. Precautions must be taken to insure stable integrator action and these will be examined in the Discussion part of the following section.

2.4 Performance Evaluation and Discussion

In this section, the performance of the adaptive torque scheduling controller as applied to a permanent magnet d.c. motor is evaluated. The motor, controller and load were simulated by numerical integration of the governing differential equations using a fixed-step fourth-order Runga-Kutta integration routine. A commented version of the simulation program is in the Appendix.

The torque schedule controller performance is compared to P, PI and PID controller configurations for two load cases. Load Case I is a four-bar linkage. The mechanism dynamics and torque requirements are calculated from an influence-coefficient model used by S. Carlson [3]. A plot of the load torque is shown in Fig. 2-3. Load Case II is a periodic step in load torque. Though this type of loading may not be frequently encountered in real applications, it is used here as a "worst case" for evaluating controller performance. A plot of the step load function is shown in Fig. 2-4. Both load cases contain small amounts of viscous damping on the order of 10% of the full load torque capacity

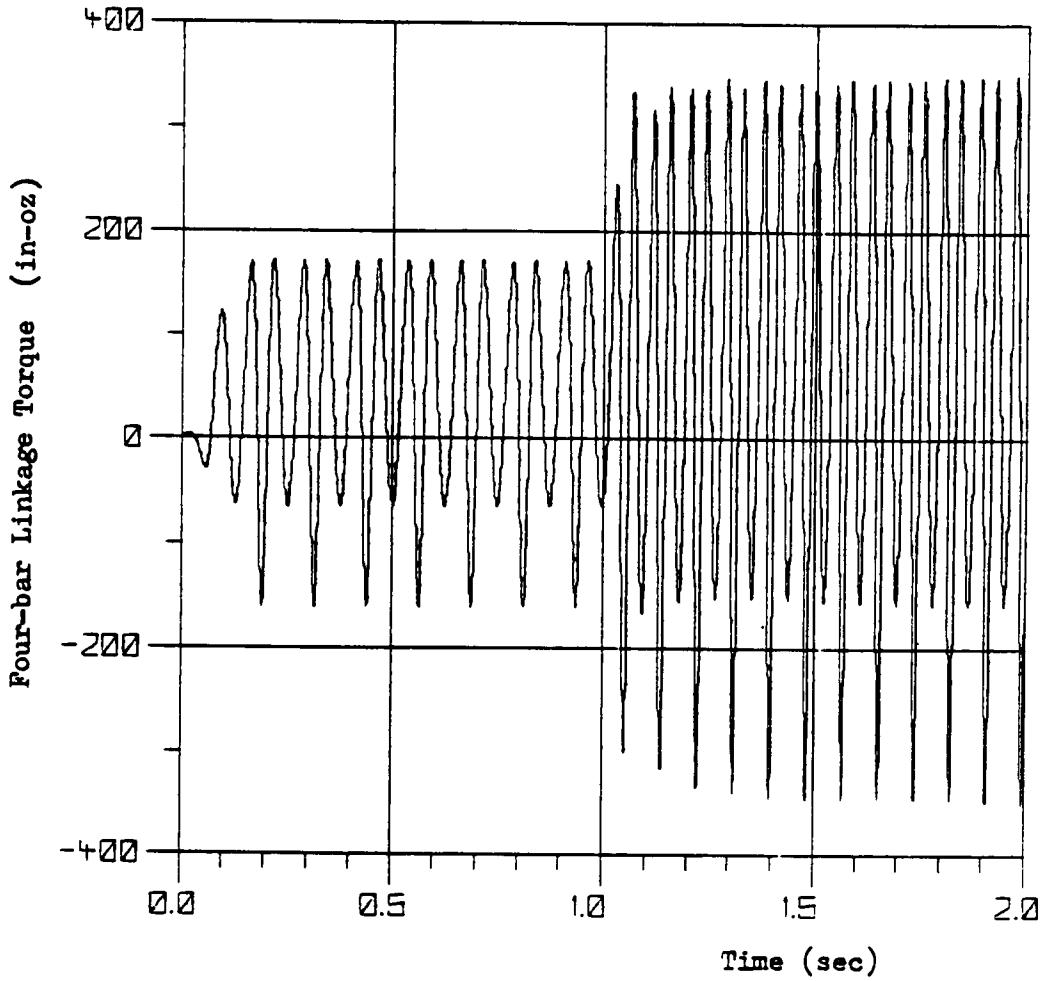


Fig. 2-3. Load case I. Four-bar linkage load torque requirements for speeds of 50 and 75 r/s.

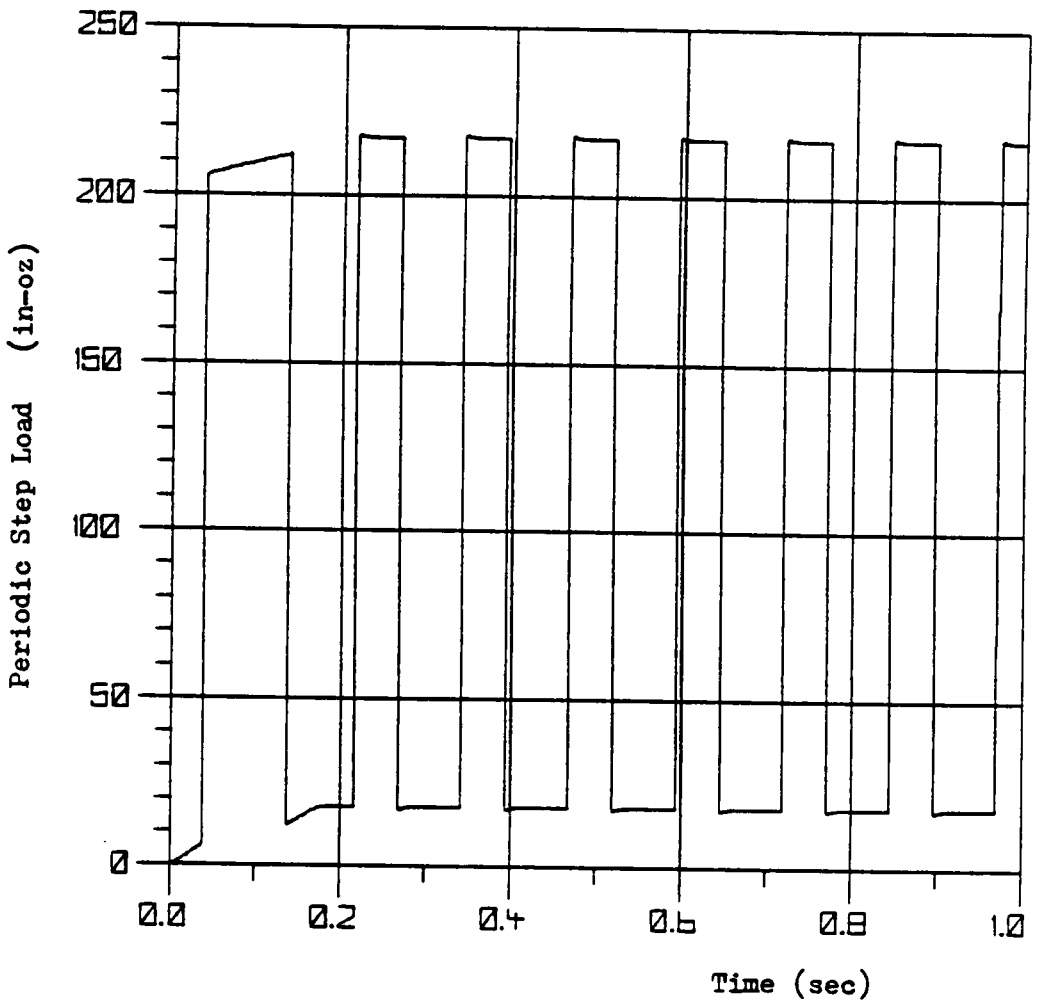


Fig. 2-4. Load case II. 200 in-oz (1.41 N-m) periodic step load. Viscous load is 17.5 in-oz (0.124 N-m) at 50 r/s.

of the motor. Table 2-1 summarizes the motor and load specifications for each load case.

Table 2-1 Motor and Load Specifications

Motor Specification	Customary	S.I.
armature resistance	7.0 ohms	7.0 ohms
armature inductance	19.43×10^{-3} henries	19.43×10^{-3} henries
voltage constant	21.9 V/KRPM	0.209 V • s
torque constant	29.63 in-oz/A	0.209 N • m/A
maximum current	12A	12A
maximum power	0.24 horsepower	0.179 kW

LOAD CASE I - Four Bar Linkage
(Approximate magnitude of the load parameters are given below, the actual linkage dynamics were calculated from a model)

Inertia	$0.7 + 0.07 \sin(\theta_m + \phi_1)$	in-oz-s^2
	$(0.005 + 0.0005 \sin(\theta_m + \phi_1)$	N-m-s^2)
Centripetal Load Component	$0.06\ \dot{\theta}_m^{\,2} \sin(2\theta_m + \phi_2)$	in-oz
	$(0.00042\ \dot{\theta}_m^{\,2} \sin(2\theta_m + \phi_2)$	N-m)
Gravity Load Component	$6.0 \sin(\theta_m + \phi_3)$	in-oz
	$(0.042 \sin(\theta_m + \phi_3)$	N-m)
Viscous Damping Coef.	0.35 in-oz-s	(0.0025 N-m-s)

LOAD CASE II - Periodic Step Load

Inertia	0.7 in-oz-s^2	(0.005 N-m-s^2)
Viscous Damping Coef.	0.35 in-oz-s	(0.0025 N-m-s)
Step Load Torque	200 in-oz	(1.41 N-m)

CASE I - Four-Bar Linkage Load

Proportional Controller Performance

This simple control arrangement results in a closed loop system which is modeled here as second order. This system is stable for any gain setting, however the damping ratio falls with increasing gain. Speed error, the magnitude of the fluctuations and the mean value error, both decrease as forward loop gain K_p is increased.

System response for three gain settings and two speed levels are plotted in Figs. 2-5,6,7.

Figure 2-5 corresponds to K_p = 14.64 which yields a damping ratio of ξ = 1.0. The roots of the closed loop system are -177 and -184. The roots were calculated using the mean load inertia since this quantity varies periodically with rotor position. (The roots or eigenvalues given are for comparison only, the actual transient response is significantly limited by saturation effects.) Mean speed error was -2% and speed fluctuations were ±4% for a speed command of 50 r/s (S_{pdcom} = 50 r/s). Mean speed error was -2.0% and speed fluctuations were ±5% for S_{pdcom} = 75 r/s. Armature current, which has been limited to 12A, is also shown in Fig. 2-5.

Figure 2-6 corresponds to K_p = 41.2 which yields a damping ratio of ξ = 0.6. The roots for the closed loop system are -180 ± 240j. Mean speed error was -0.4% and speed fluctuations were ±2% for S_{pdcom} = 50 r/s. Mean speed error was -0.7% and speed fluctuations were ±3.4% for S_{pdcom} = 75 r/s.

Figure 2-7 corresponds to K_p = 500 which yields a damping ratio

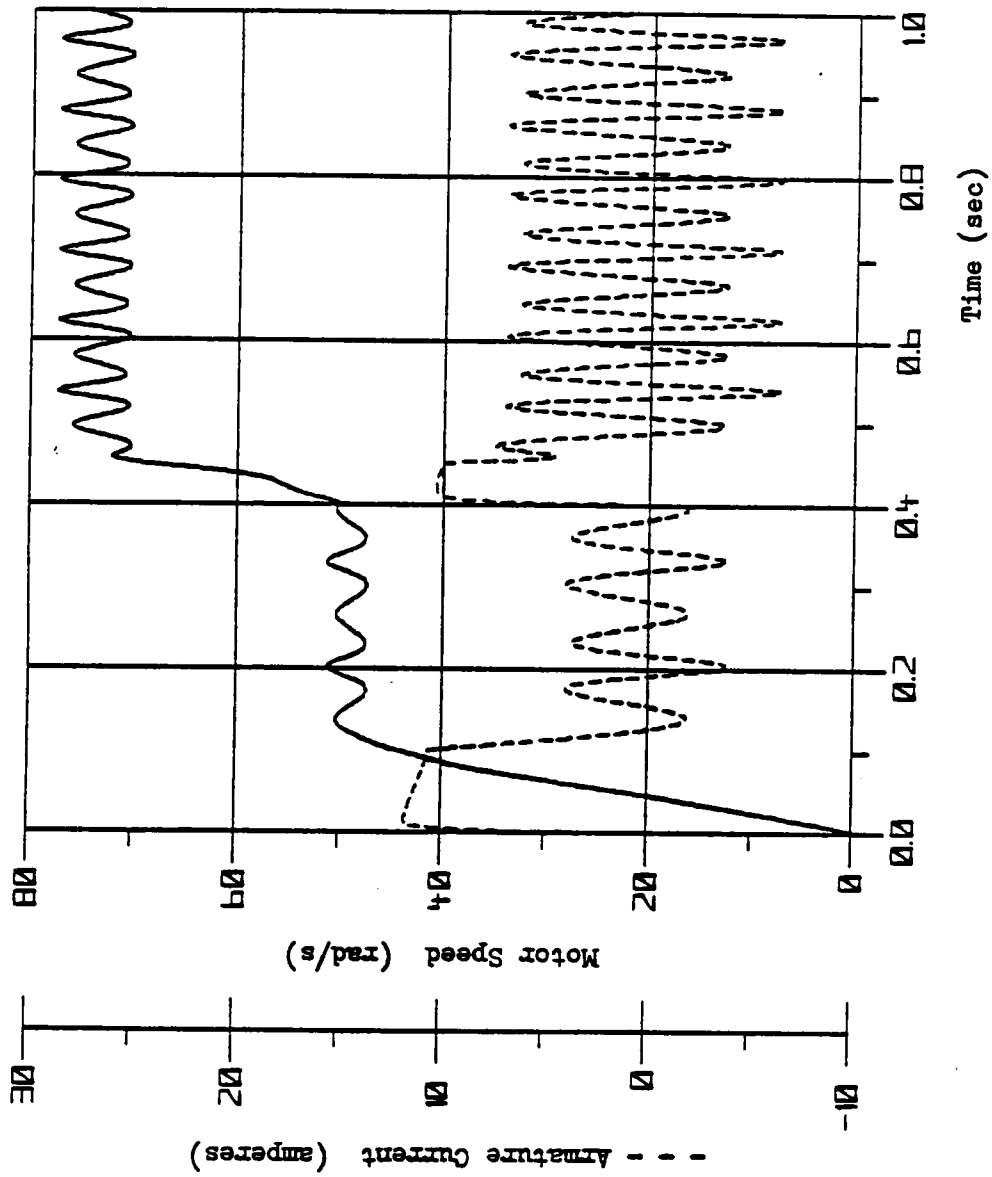


Fig. 2-5. P controlled system speed response to four-bar linkage load (K_p=14.6).

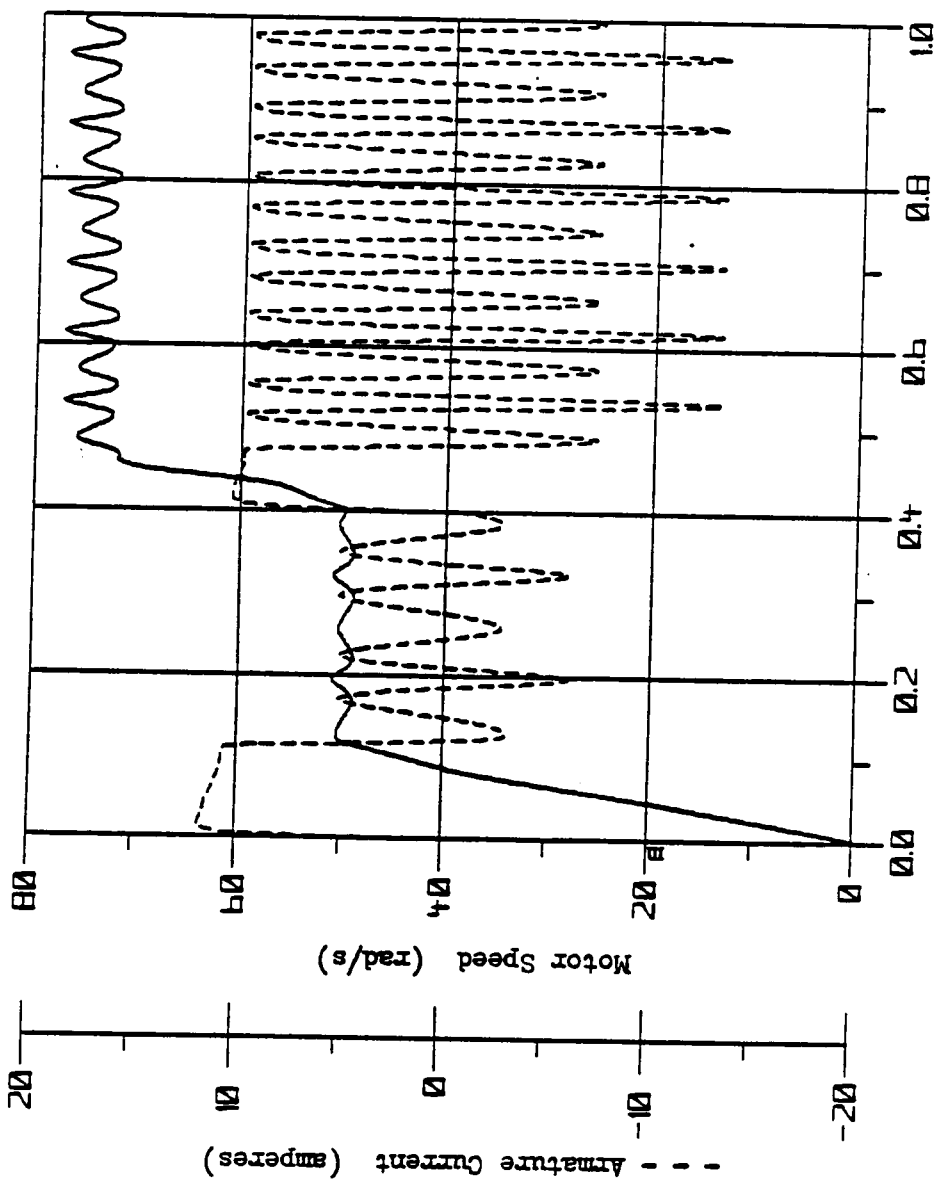


Fig. 2-6. P controlled system speed response to four-bar linkage load (K_p=41.2).

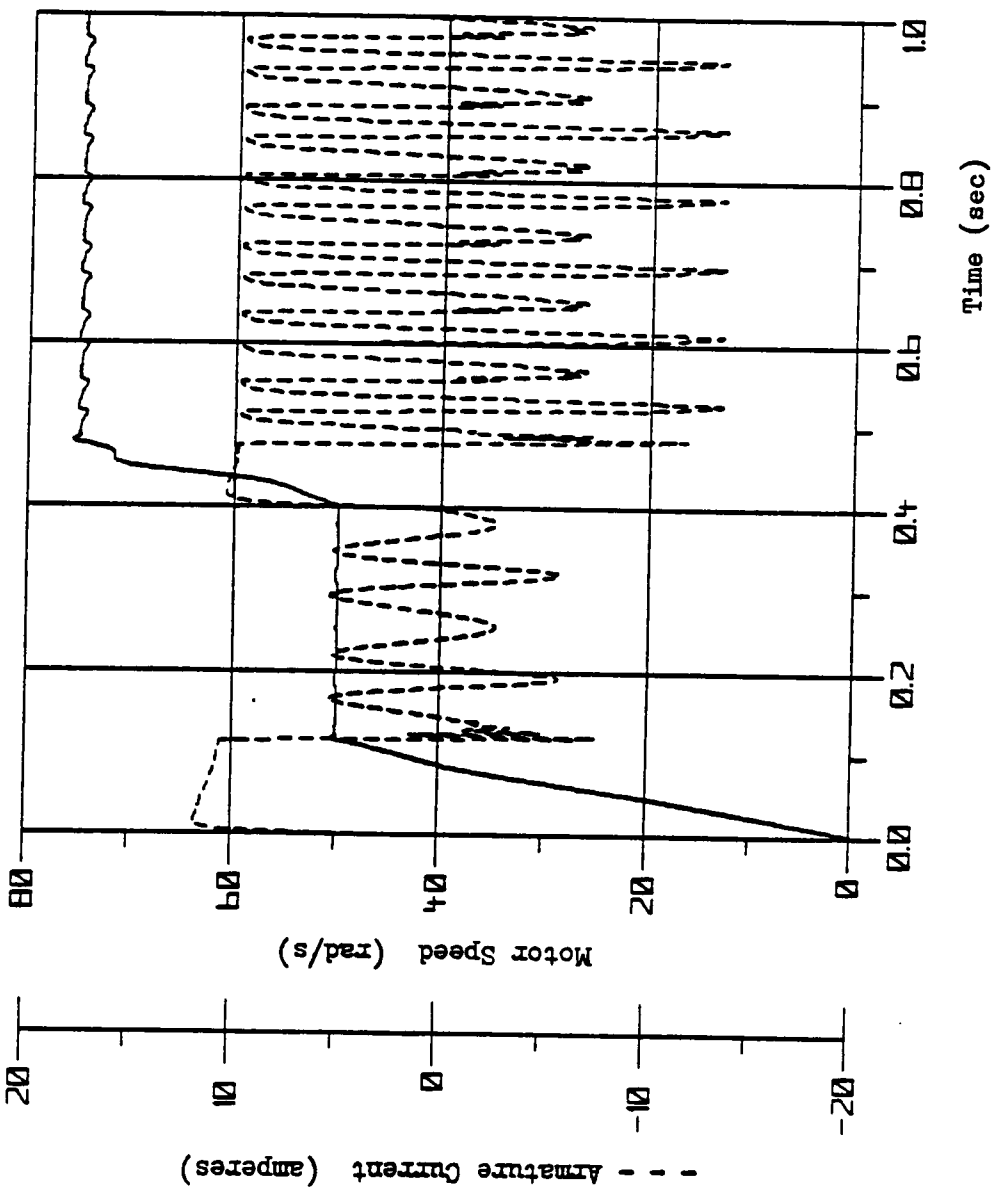


Fig. 2-7. P controlled system speed response to the four-bar linkage load (K_p=500).

ξ = .17. The roots of the closed loop system are -180 ± 1028j. Mean speed error was -0.05% and speed fluctuations were ±0.2% for S_{pdcom} = 50 r/s. Mean speed error was -0.27% and speed fluctuations were ±0.67% for S_{pdcom} = 75 r/s.

System performance as shown for a very large forward loop gain (K_p = 500) appears to be excellent. However, the system probably could not be realized in practice for the reasons stated in Section 2.2. System overshoot is small even with ξ = .17 because the amplifier saturates at 84 V. Without saturation the initial terminal voltage would have been 25,000 volts.

Proportional - Integral Controller Performance

The PI controller offers one advantage over the P controller in that speed droop is eliminated without using a large K_p. The PI controller gains were found by first selecting K_p = 14.64 and then examining the root locus of the system as K_i was varied. The system with PI control is third-order and the roots of the closed loop system remain real until K_i = 400. The roots for K_i = 400 were approximately -59, -61 and -240. System response for K_i = 400 was fast but the overshoot exceeded 35%.

Figure 2-8 corresponds to K_p = 14.64, K_i = 8 and closed-loop roots of -0.53, -169, and -190. The mean speed error or speed droop for either speed level was very close to zero. The speed fluctuations were ±4% for S_{pdcom} = 50 r/s and ±5% for S_{pdcom} = 75 r/s and cannot be eliminated by any K_i gain setting (as discussed in Section 2.2). Speed fluctuations actually increase slightly as K_i is increased.

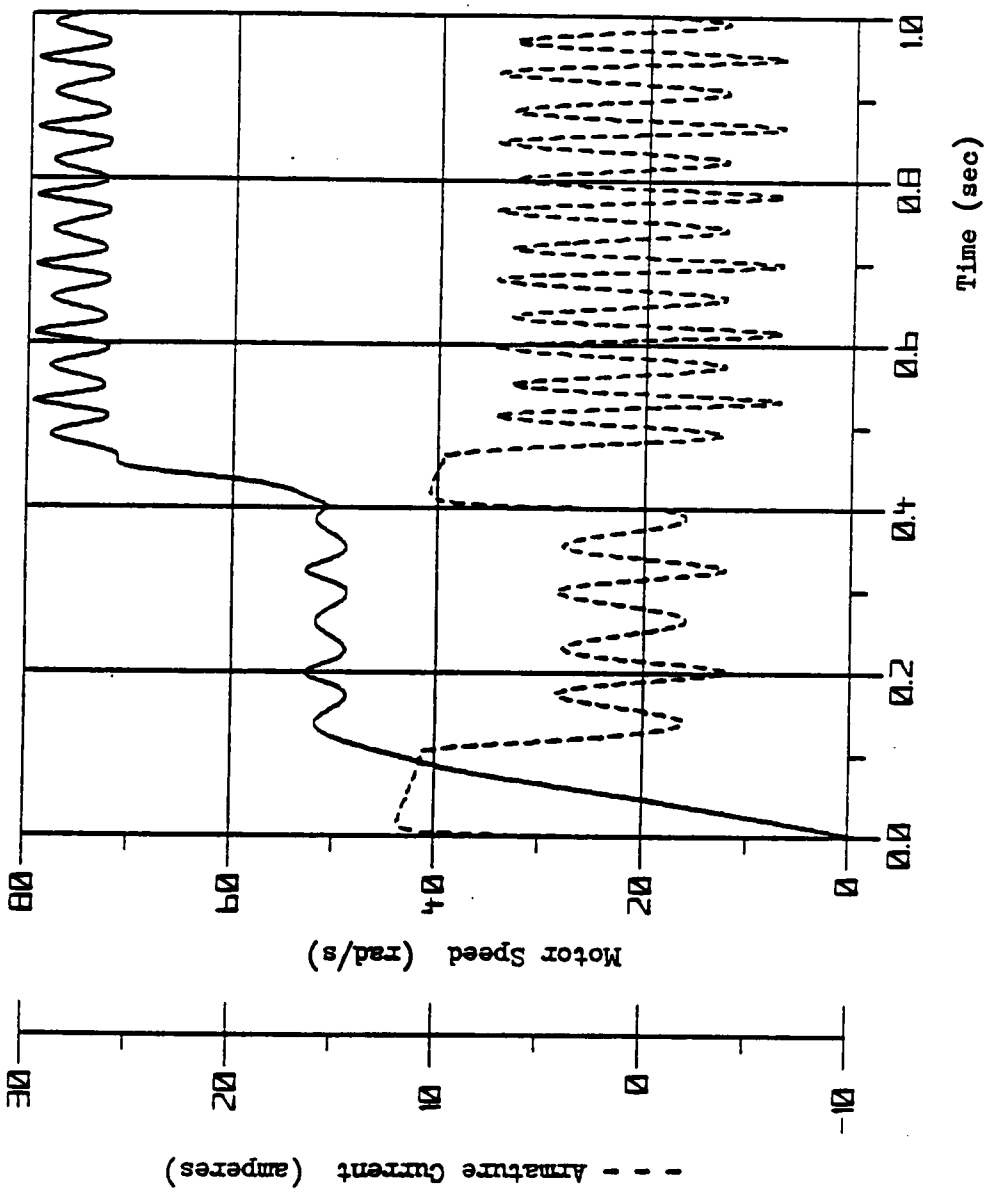


Fig. 2-8. PI controlled system speed response (K_p=14.6, K_i=8). Mean speed error is zero.

Proportional-Integral-Derivative Controller Performance

Derivative feedback limits the rate of change of speed error and helps reduce the periodic speed fluctuations. The derivative feedback gain K_d was found by choosing K_p and K_i and then examining the roots of the characteristic equation as K_d is varied. Results from the simulation runs were also used to select K_d. Initially gain settings were K_p = 14.6, K_i = 8 and K_d was varied. The root locus showed a root near -0.54 which was affected very little by K_d as it was varied from 0.0004 to 2.0. The root at -0.54 gave good performance with PI control because the system converged directly onto the desired speed with no overshoot. Derivative feedback slowed the system down allowing the integration term to get much too large and causing a large overshoot. The root at -0.54 was moved to about -5.7 by increasing K_i to 80. The integrator then dominated the initial response to the speed command which also allowed the derivative feedback to be larger and more effective in reducing speed fluctuations. Best performance, i.e., fast response, reasonable overshoot and best reduction in speed fluctuations was found with K_p = 14.6, K_i = 80 and K_d = 1.32. The system roots were -5 ± 5.2j and -3226. (Note that "best" performance here was not quantified by a standard criterion, e.g., IAE, ITAE, etc. Instead, best performance was judged primarily on the reduction of steady state speed error.)

Figure 2-9 shows the system response with the gain settings K_p = 14.64, K_i = 80, and K_d = 1.32.

Overshoot was about 10%, mean speed error near zero and fluctuations ±0.4% for S_{pdcom} = 50 r/s. Overshoot was about 5%, mean

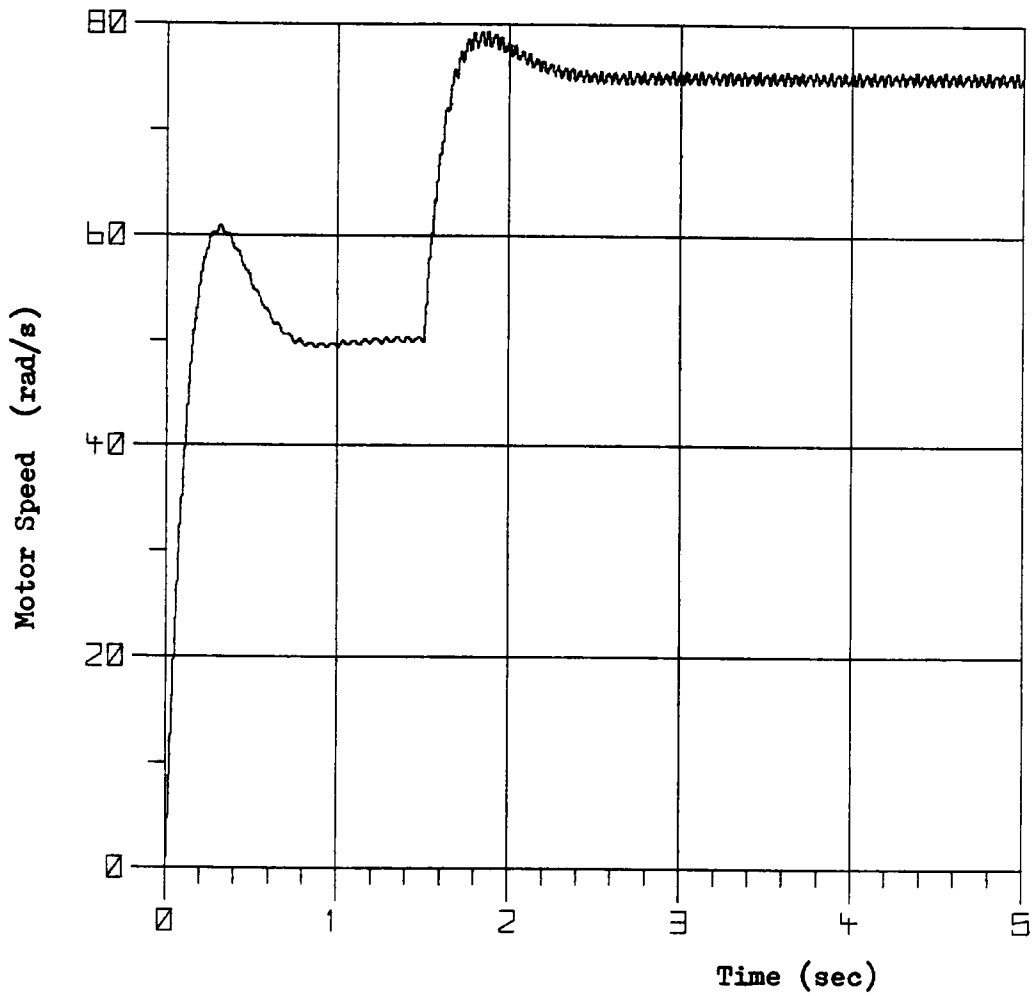


Fig. 2-9. PID controlled system speed response (K_p=14.6, K_i=80, and K_d=1.32).

speed error near zero and speed fluctuations ±0.6% for S_{pdcom} = 75 r/s.

PID control offers reasonably good speed control for the linkage modeled. Figure 2-9 shows the difference in controller response at different speed levels - both in speed error and overshoot. The PID controller has an optimum set of gains for each operating point of the system. Changes in speed, load or motor characteristics cannot be compensated for.

The simulation results show that the PID control scheme is the best of those considered so far. The performance of the torque schedule controller will be evaluated in terms of the PID simulation results.

Torque Schedule Controller Performance

Figure 2-10 shows the system speed response to speed commands of 50 and 75 r/s for Load Case I.

The schedule adapted in 5 load cycles to drive speed fluctuations from ±4% to ±0.015% for S_{pdcom} = 50 r/s. The schedule integrator drove mean speed error to less than -0.015% after 8 load cycles.

Following the step change in the speed command to 75 r/s, speed fluctuations were driven to less than 0.014% in 6 load cycles and mean speed error was reduced to less than 0.004% after 14 load cycles.

Figure 2-11 shows a plot of the integrator voltage $K_v * V_{OFFSET}$. The schedule values were modified once per revolution by the voltage levels shown. The $K_v * V_{OFFSET}$ term is limited to 10 V.

Figure 2-12 shows a plot of the armature current generated by the scheduled voltages. This current waveform corresponds very closely to the load torque plotted in Fig. 2-3.

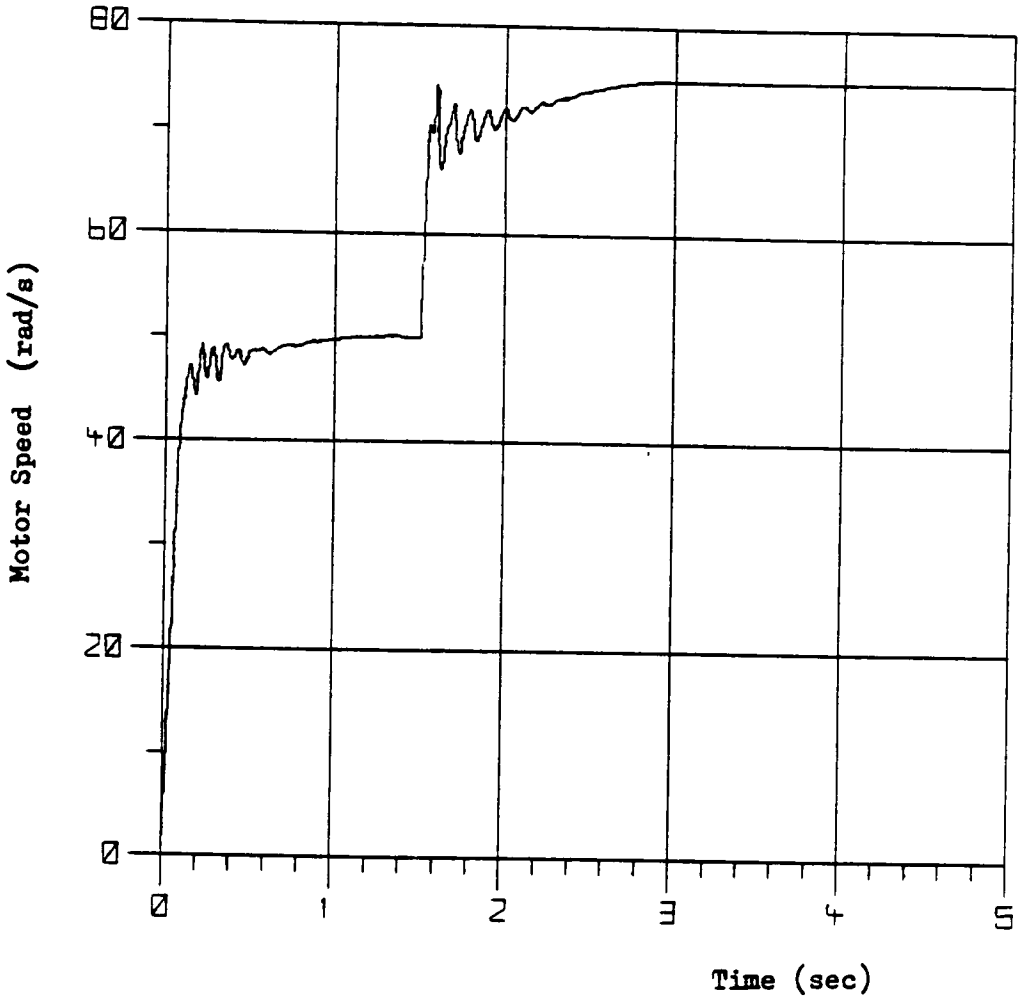


Fig. 2-10. Torque schedule controlled system response to the four-bar linkage load.

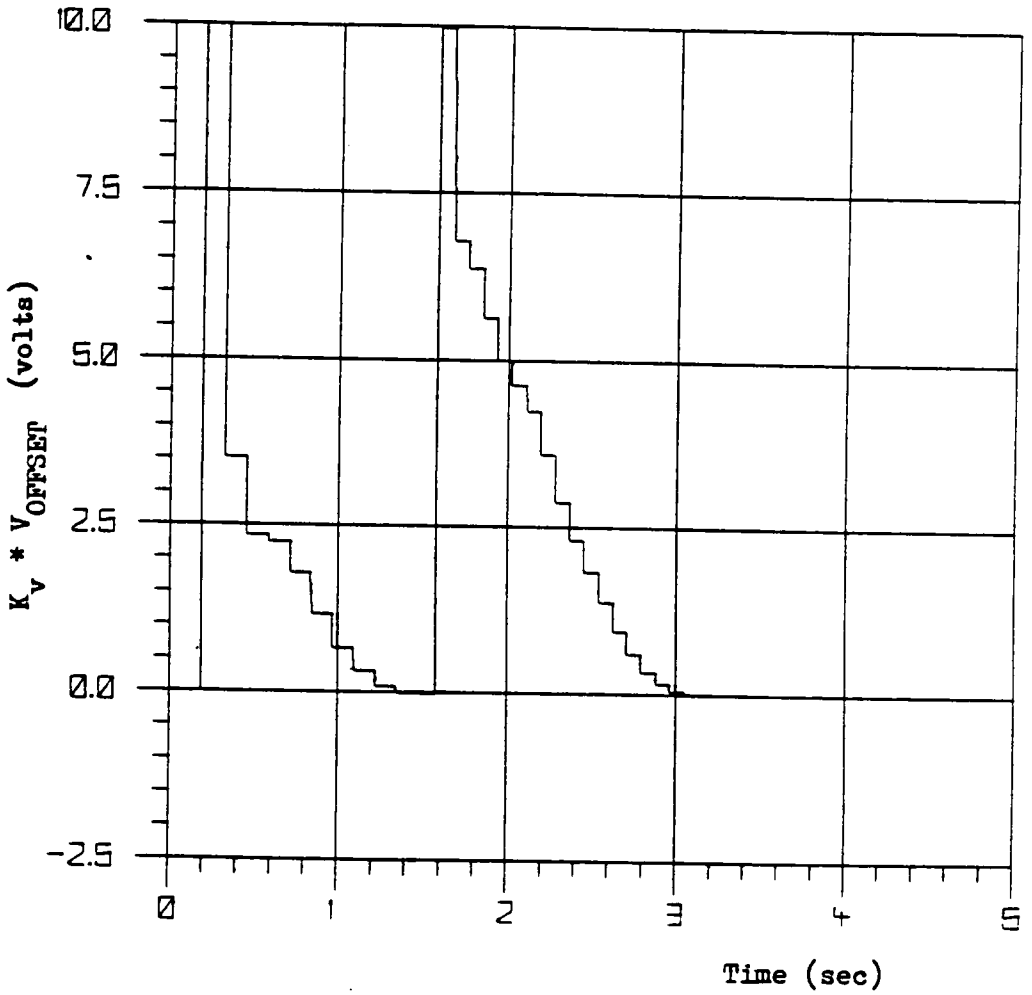


Fig. 2-11. Plot of the voltage added to the schedule output at the completion of each load cycle.

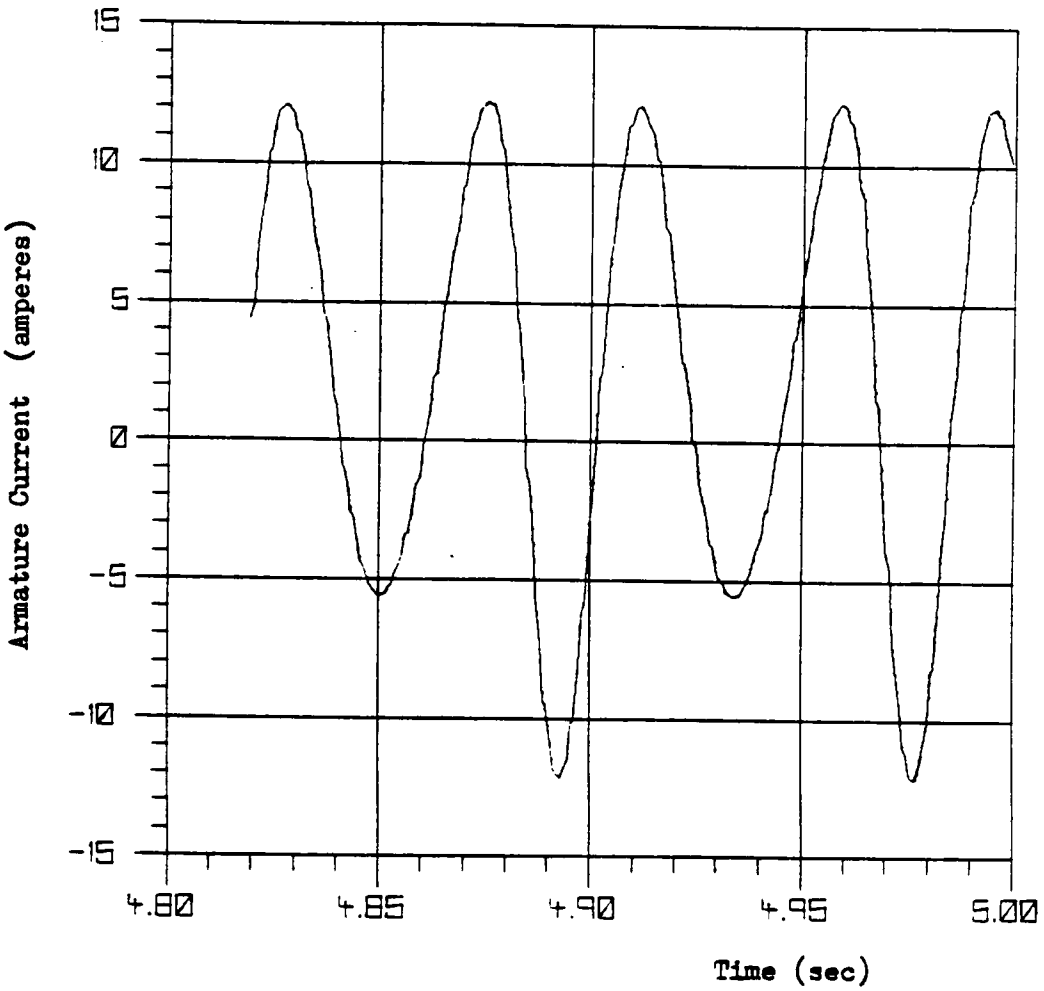


Fig. 2-12. Motor armature current generated by the schedule voltages Linkage speed is 75 r/s.

The adaptation equation gains for the simulation were K_v = 1.26 and K_s = 84/126 (for S_{pdcom} = 50/75 r/s). A method for choosing these gains is discussed later. All of the measured quantities - speed, speed error, position and voltage values in the schedule were assumed to be of 12-bit accuracy. The system response followed the sequence below.

1. zero initial conditions
2. step input, S_{pdcom} = 50 r/s, set K_s = 84
3. e^2 feedback loop saturates and increases motor speed rapidly
4. the voltage schedule begins adaptation when motor speed exceeds (.8) (S_{pdcom})
5. e^2 feedback approaches zero as the integration term K_v * V_{OFFSET} adjusts the schedule voltage to drive speed error to zero
6. step input, S_{pdcom} = 75 r/s, set K_s = 126
7. steps 3-5 are repeated
8. simulation ends.

The e^2 feedback loop gives better performance in this application for reasons which will be discussed later. The voltage schedule does not begin adaptation until motor speed exceeds 80% of the desired level so as to not interfere with a rapid response to changes in the speed command.

The torque schedule controller reduced speed error fluctuations better than the other controllers considered. Speed fluctuations were reduced from ±0.45% to 0.015% (factor of 30) for S_{pdcom} = 50 r/s and from ±0.7% to ±0.014% (factor of 50) for S_{pdcom} = 75 as compared to PID control. Note that the speed error for the schedule controller motor at

the two speed levels differs by 7%. The PID error differs by 50%.

It is apparent from Fig. 2-10 that the schedule controlled system experience on undesirable speed transient during the initial schedule adaptation. (The speed control during adaptation however, is no worse than that provided by the P or e^2 feedback loop alone.) The relatively large speed transient is necessary for rapid schedule adaptation. Speed transients at a particular operating point can be minimized by saving a previously adapted waveform for future system operation.

The mean speed error is very small for the schedule controlled system but not driven to zero because of a small interference between the $K_v * V_{OFFSET}$ control action and the $K_s * \Delta S_i$ term in Eq. (2-3). Essentially the $K_v * V_{OFFSET}$ gets so small that the $K_s * \Delta S_i$ term (which is also small) is about the same size. The terms are opposite in sign and their difference is lost during quantization so adaptation ceases.

It is important to note that speed controller performance in a simulation such as this may not be obtainable in a real system. The purpose here is to compare relative performance using models which account for the major dynamic characteristics of the systems involved.

CASE II - Periodic Step Load

The motor parameters, damping coefficient, mean inertia and controller gains throughout Case II are identical to those used in Case I. Therefore, a discussion of the methods of gain selection and system eigenvalues will not be repeated here.

Proportional Controller Performance

System speed response and armature current for three gain settings at S_{pdcom} = 50 r/s are plotted in Figs. 2-13,14,15.

Figure 2-13 shows speed response for K_p = 14.6 (ξ = 1.0). Mean speed error was -5% and speed fluctuations were ±3.2%.

Figure 2-14 shows speed response for K_p = 41.2 (ξ = 0.6). Mean speed error was -2% and speed fluctuations were ±1.6%.

Figure 2-15 shows speed response for K_p = 500 (ξ = 0.17). Mean speed error was less than -0.4% and speed fluctuations were ±0.7%. The expected large overshoot for a system with ξ = 0.17 is not present because of amplifier saturation.

As expected, the highest loop gain yielded the best performance in the simulation. For reasons stated earlier (Section 2.2), perhaps a more reasonable estimate of P controller performance corresponds to K_p = 41.2 and ξ = 0.6. The current waveforms in Figs. 2-13,14,15 show that a true current square wave is (obviously) not possible in an inductive circuit. Figure 2-15 shows what a close approximation (obtainable in a real system) looks like. Current oscillation (or "ringing") occurs at each step change in the load. It will be shown later that the torque schedule controller can sometimes create a very similar current waveform without using a large forward loop gain.

Proportional - Integral Controller Performance

Figure 2-16 shows speed response for the PI controlled system using the gains K_p = 14.64 and K_i = 8. The integrator gain happens to match the load conditions and there is no overshoot present. The speed fluc-

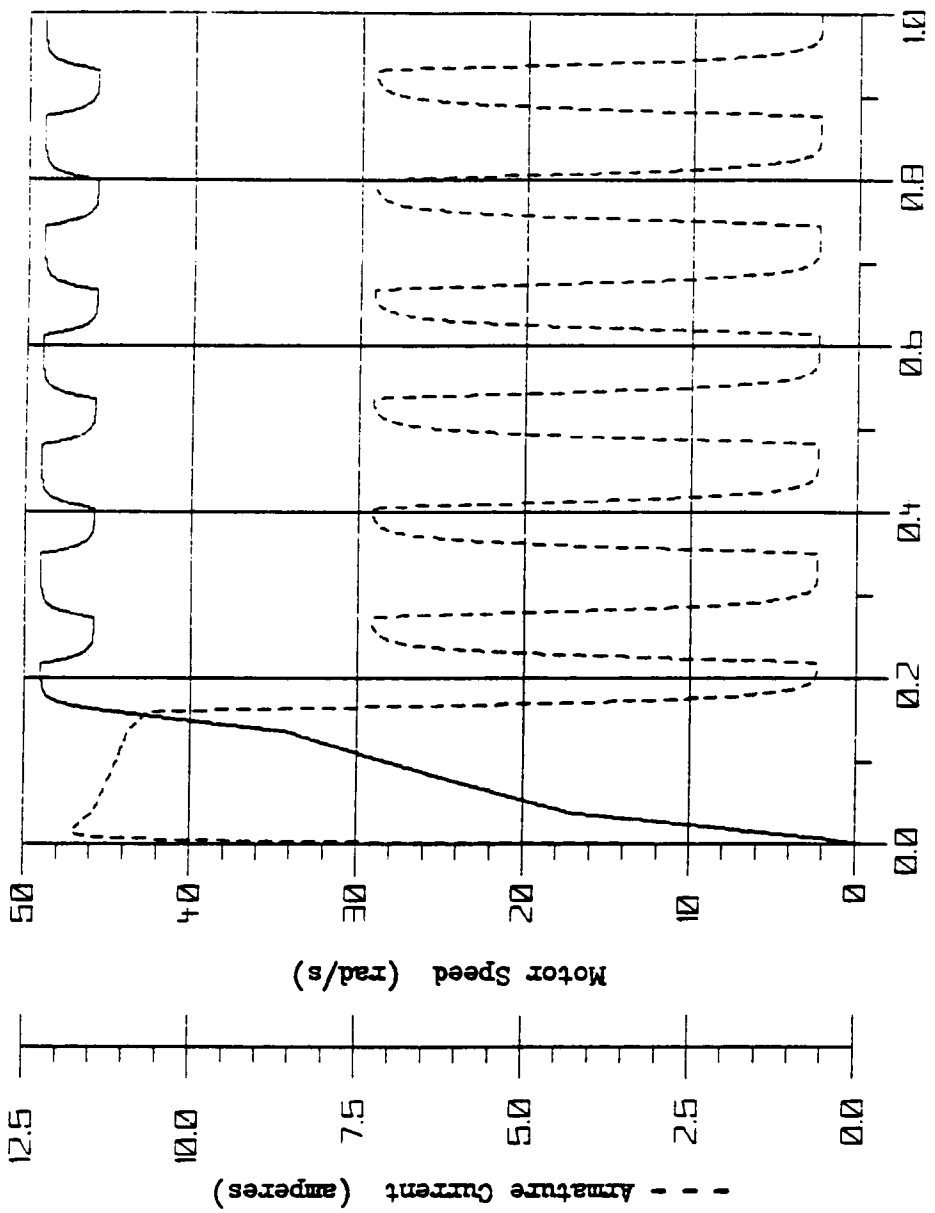


Fig. 2-13. P controlled system speed response to the periodic step load (K_p=14.6).

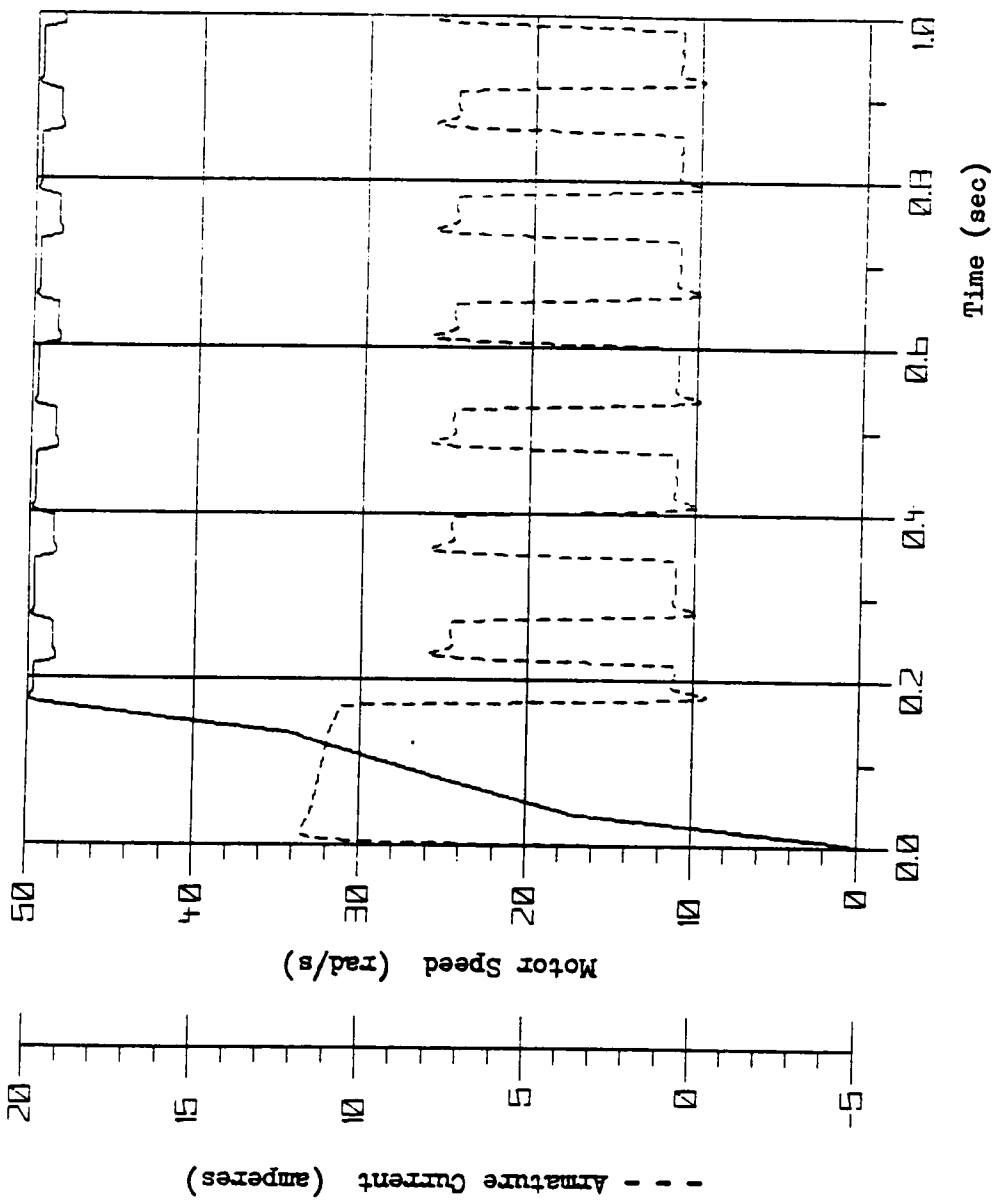


Fig. 2-14. P controlled system speed response to the periodic step load (K_p=41.2)

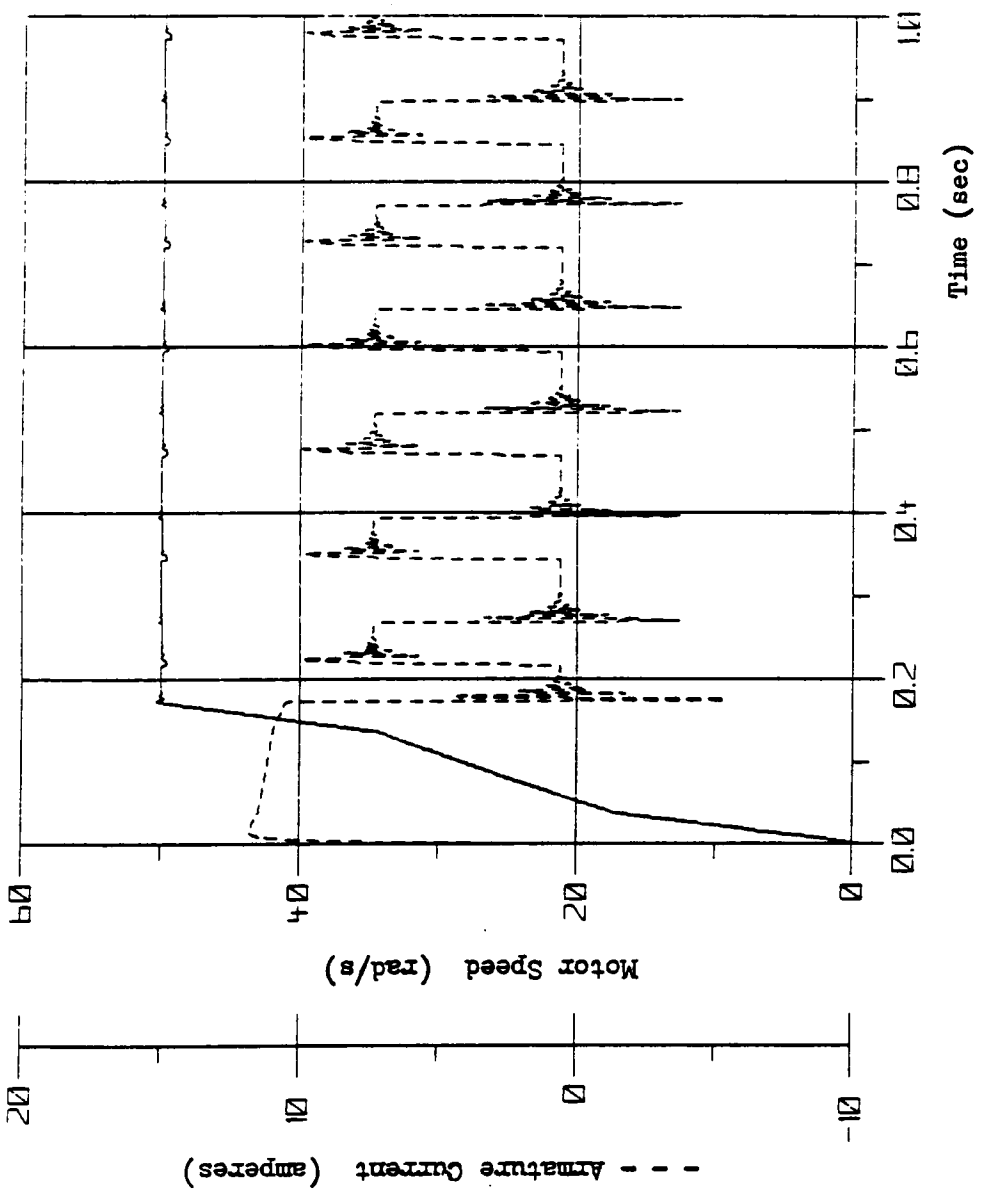


Fig. 2-15. P controlled system speed response to periodic step load (K_p=500). Note current ripple during step change in load.

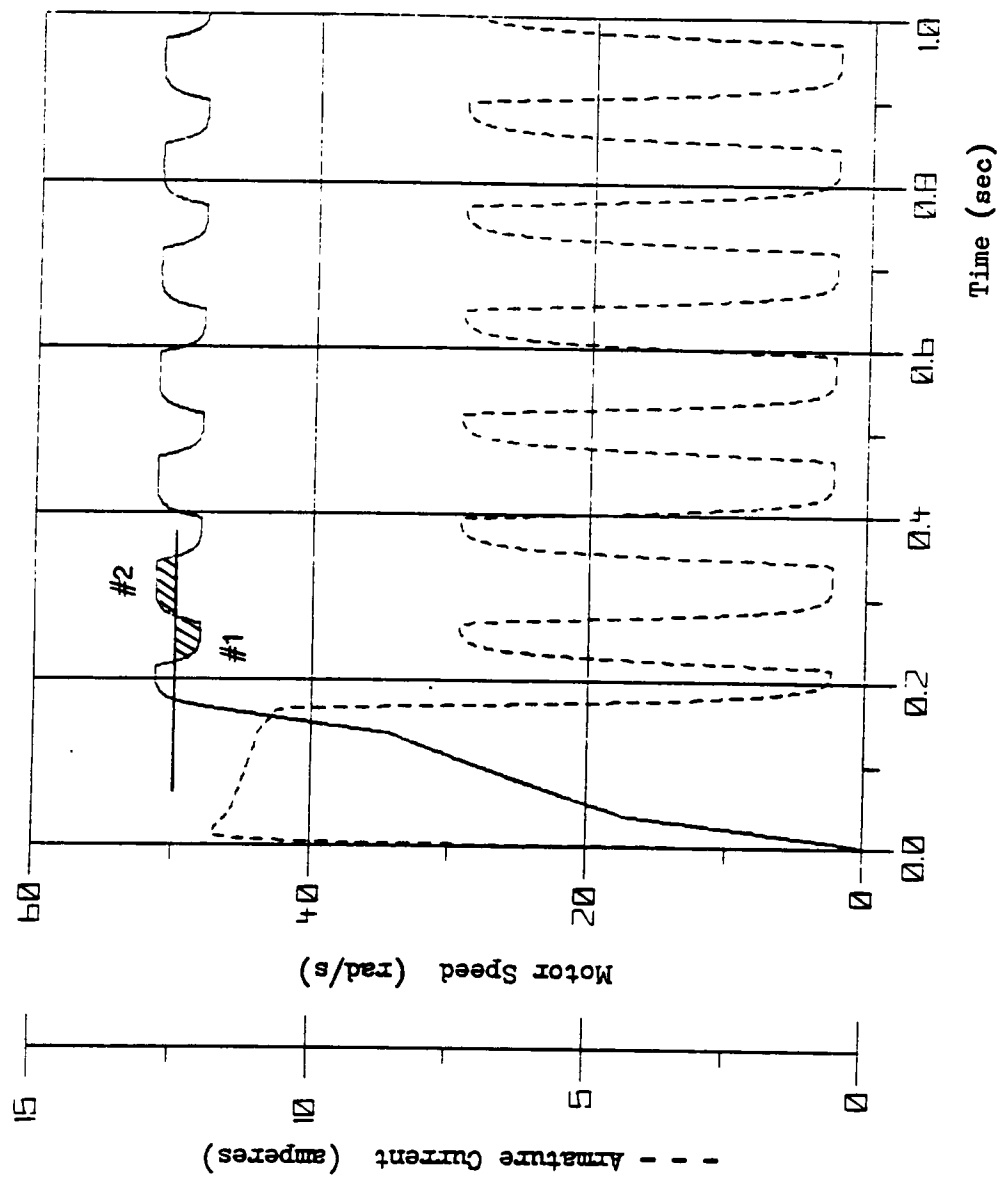


Fig. 2-16. PI controlled system speed response to periodic step load (K_p=14.6, K_I=8). Mean speed error is zero.

tuations are not exactly centered on the speed command (50 r/s) because the load cycle is not symmetrical. The shaded areas 1 and 2 of one load cycle in Fig. 2-16 are nearly equal so the average speed error is very close to zero for each cycle. As before, increasing the integrator gain will not reduce the speed fluctuations which are ±5%.

Proportional - Integral - Derivative Controller Performance

The system response for gain settings $K_p = 14.6$, $K_i = 80$ and $K_d = 1.32$ at $S_{pdcom} = 50$ r/s is shown in Fig. 2-17. Mean speed error was very close to zero after 1.25 secs and speed fluctuations were ±1.0%. Overshoot was approximately 30%.

The current waveform is close to a true square waveform. The derivative feedback greatly improves system response and eliminates the "ringing" associated with a large forward loop gain. Derivative feedback can cause the current "ringing" if K_d is too large. In this simulation, K_d is well matched to the load conditions.

Again, PID control offers the best overall performance and the results here will be used to evaluate the torque scheduling controller.

Torque Schedule Controller Performance

Figure 2-18 shows the system speed response to a command speed of 50 r/s. The schedule adapted in 9 load cycles to drive speed fluctuations from ±4% to 0.08%. The schedule integrator drove mean speed error to 0.01% in 12 load cycles. Figure 2-19 shows a plot of the integrator output voltage $K_v * V_{OFFSET}$. The schedule values were modified once per revolution by the voltage levels shown in Fig. 2-19. The $K_v * V_{OFFSET}$

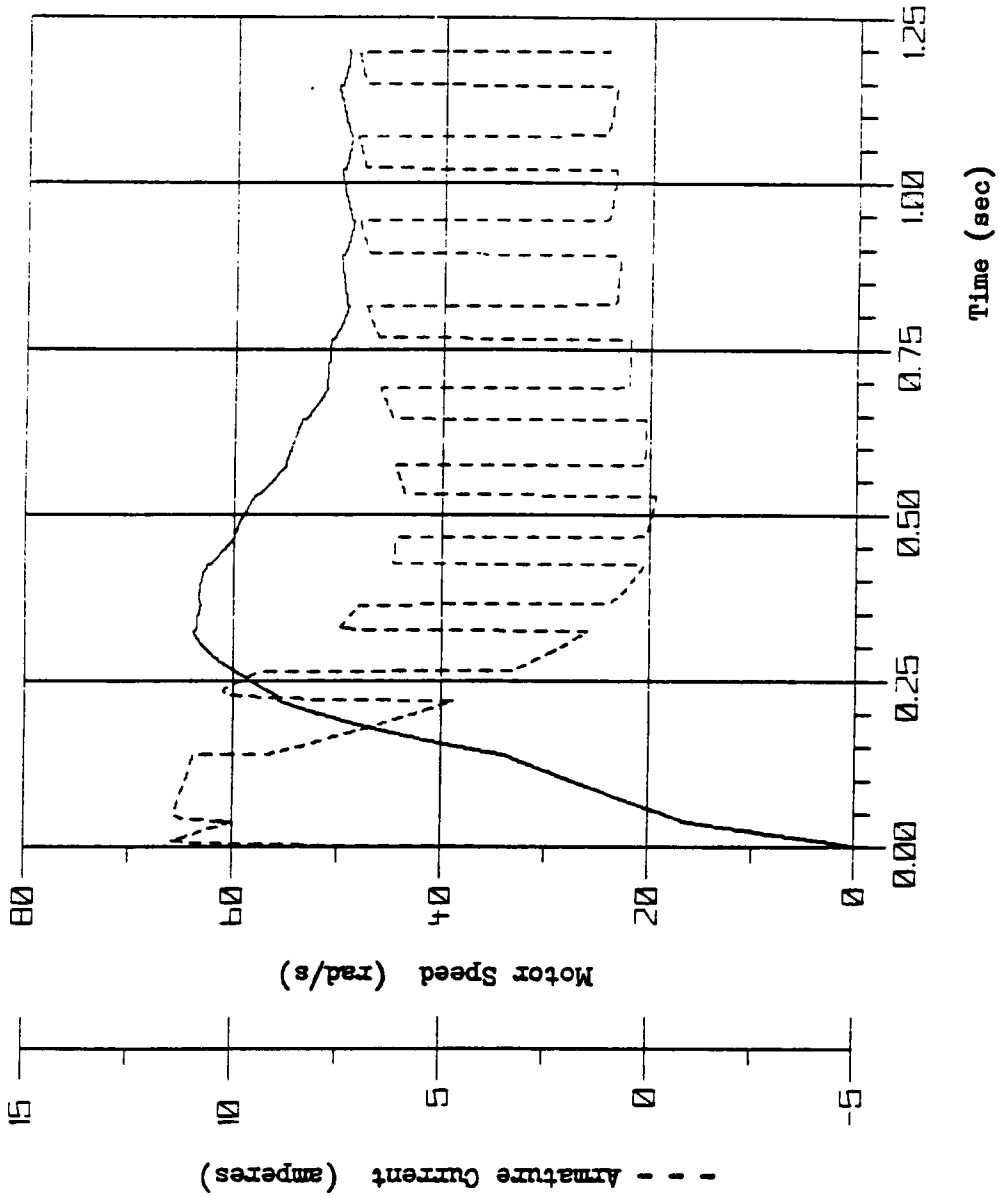


Fig. 2-17. PID controlled system response to the periodic step load (K_p=14.6, K_i=80, K_d=1.32)

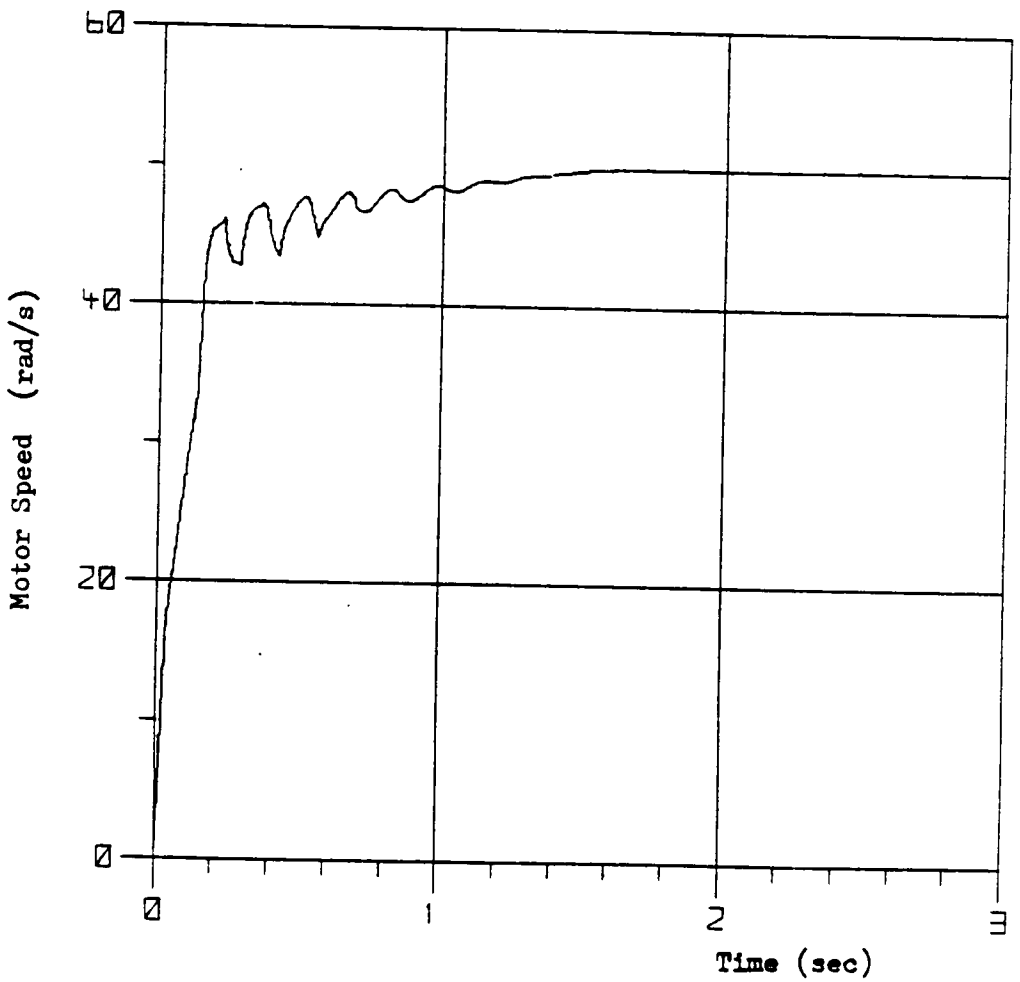


Fig. 2-18. Torque schedule controlled system response to periodic step load.

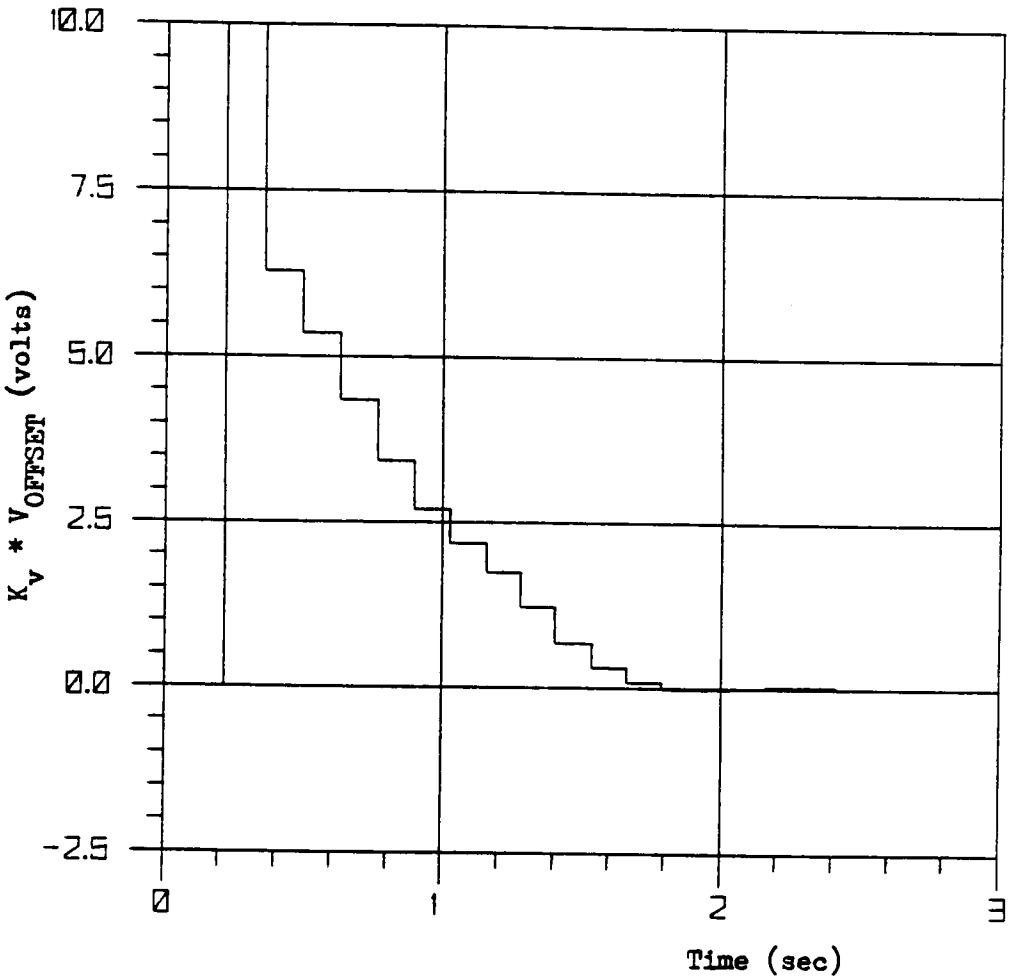


Fig. 2-19. Plot of the voltage added to the schedule output at the completion of each load cycle.

term is limited to 10 V. Figure 2-20 shows a plot of the schedule as it adapts and Fig. 2-21 shows the armature voltage. Figure 2-22 shows the corresponding current waveform for several load cycles.

The adaptation equation gains for the simulation shown where K_v = 1.26 and K_s = 84 as before. All other aspects of the controller operation were also the same as discussed earlier except steps 6 and 7 are eliminated from the system response sequence on p. 79.

The torque schedule controller performance was slightly better than the PID controller performance. Speed fluctuations were reduced from ±1.0% (PID) to ±0.08% (factor of 12). The schedule controlled system overshoot was very small (0.1%) compared to PID overshoot of 30%. The PID controlled system was also slightly faster, reaching steady state in 1.25 secs versus 1.5 secs for the schedule controlled system.

Mean speed error for the schedule controlled system was not driven to zero because of the interference between the $K_V * V_{OFFSET}$ and ΔS_i terms discussed earlier.

Figure 2-23 shows system response to a change in the speed command from 50 r/s to 75 r/s. The periodic step load does not change with the increased speed, only the viscous damping component of the load increases. Therefore, only the mean voltage of the schedule must be adjusted to meet the new load. Some speed fluctuations do appear during the speed change but these are adapted to in about 3 cycles. Mean speed error was reduced to less than 0.013% after 14 cycles. The steady state speed fluctuations were about the same magnitude as for the 50 r/s operating point ±0.04 r/s (±0.05%) since the load requirements have only changed a small percentage.

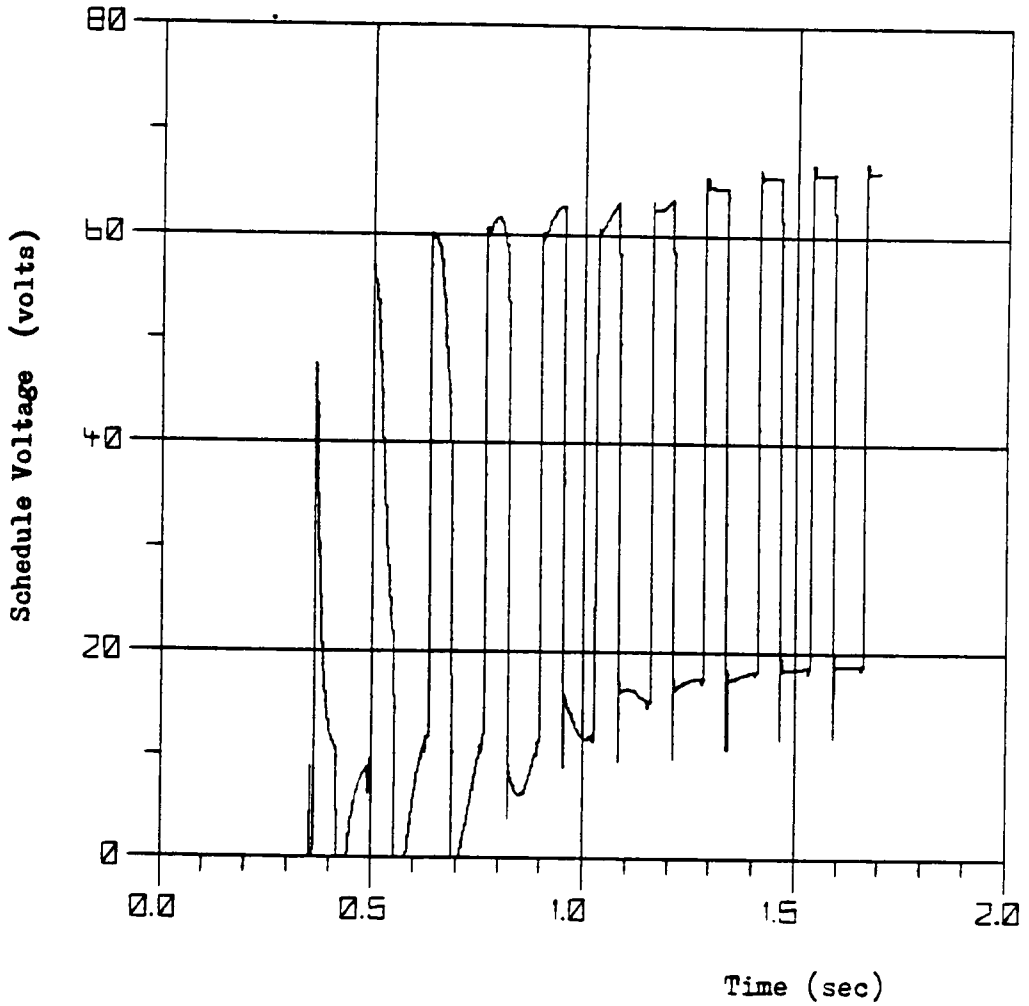


Fig. 2-20. Plot of the stored schedule voltage as it adapts to the periodic step load. The schedule contains 64 values (NPC=64) over the load cycle with 12-bit accuracy.

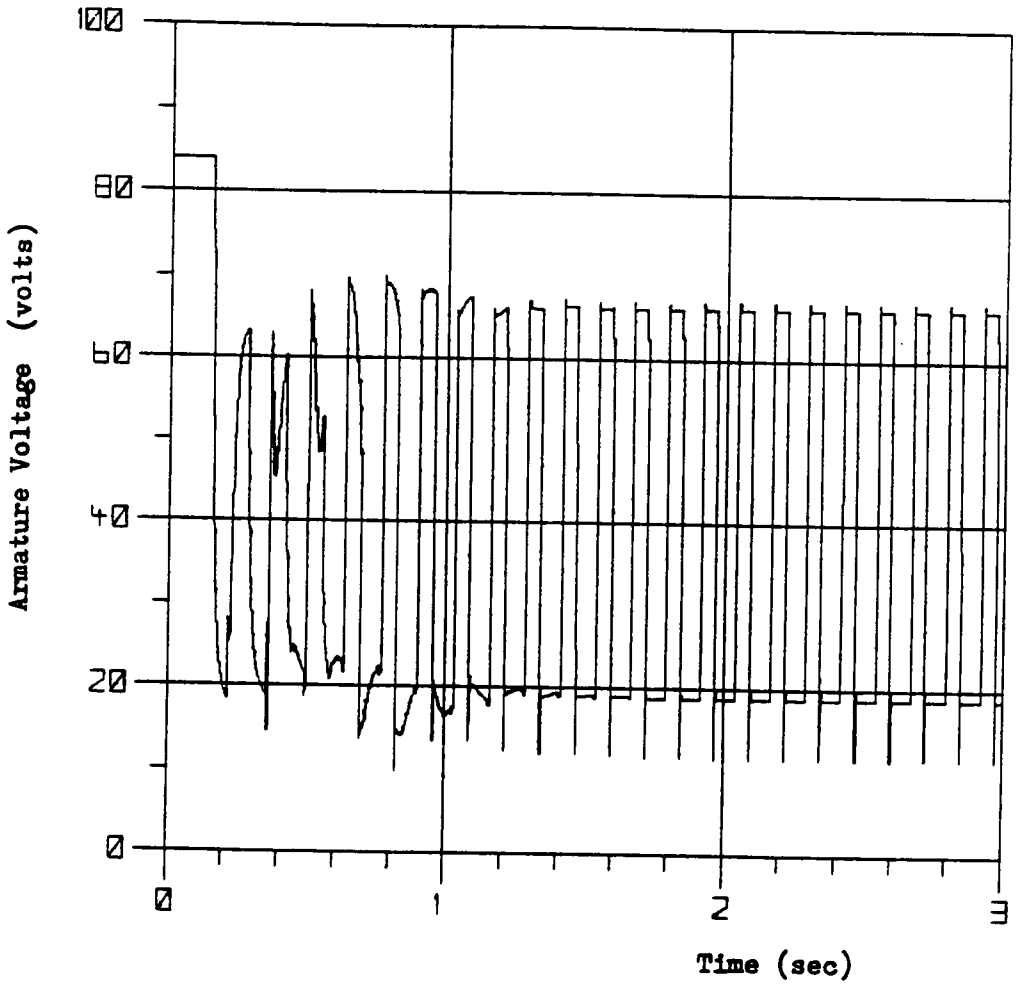


Fig. 2-21 Motor armature voltage as the schedule adapts. Voltage amplifier saturates at 84 volts.

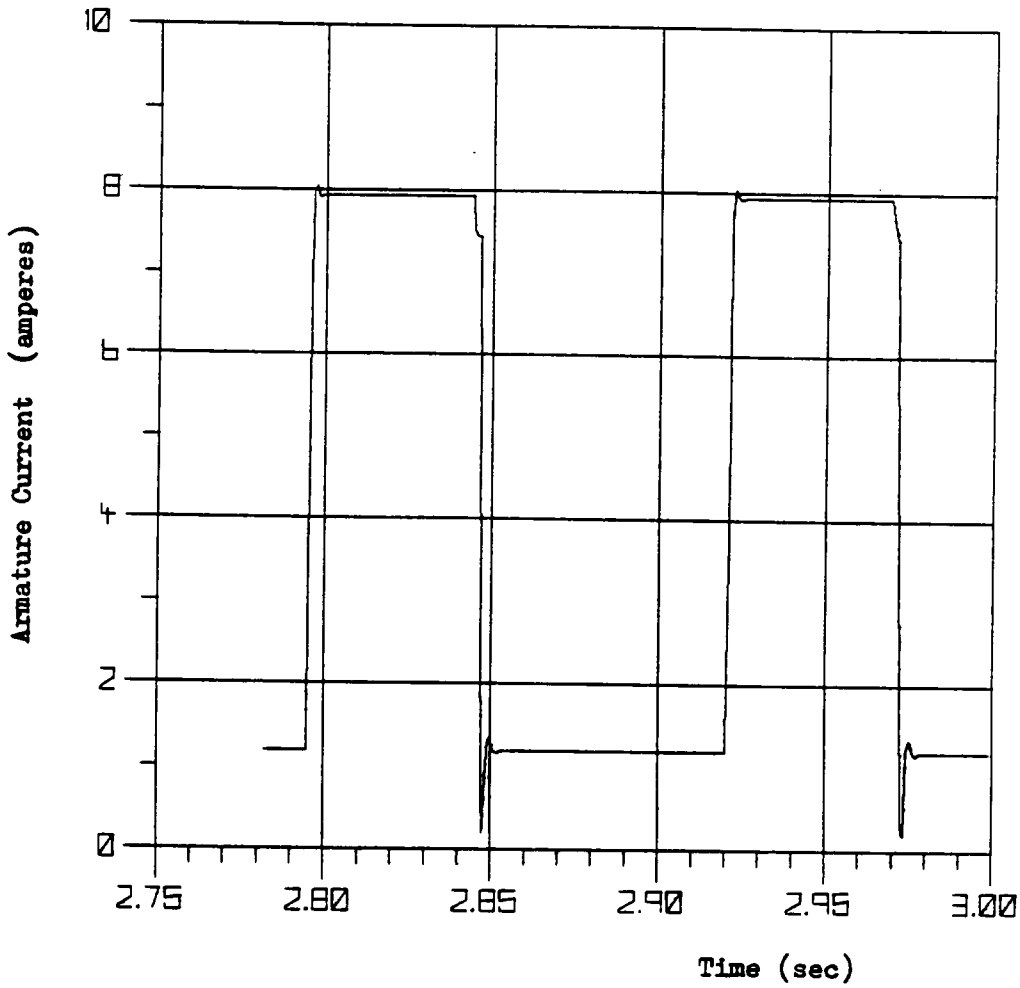


Fig. 2-22. Motor armature current generated by the schedule voltages for the periodic step load. Linkage speed is 50 r/s.

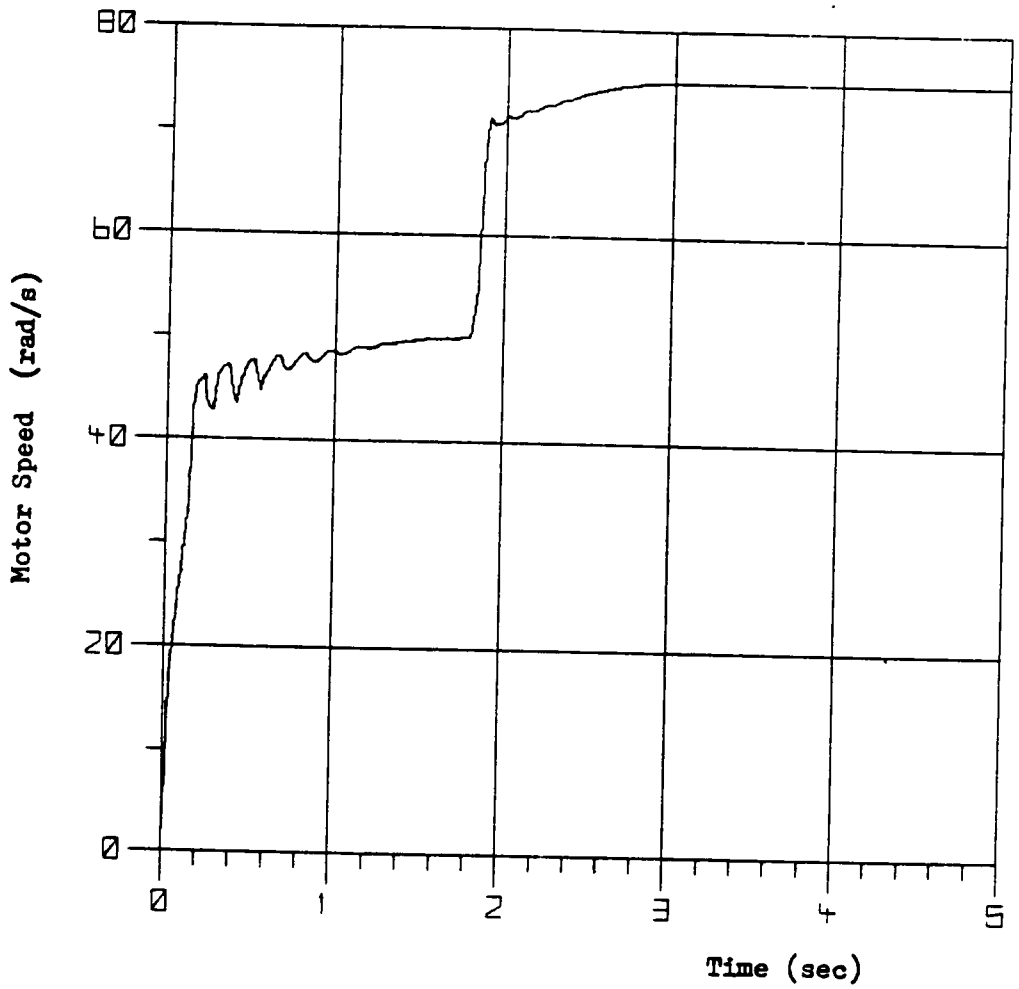


Fig. 2-23. Torque schedule controlled system response for the periodic step load with step change in speed command from 50 to 75 r/s.

2.5 DISCUSSION

This section explains the conditions necessary for stable controller performance and rapid adaptation.

The single most important stability consideration concerns the fundamental assumption of the adaptation algorithm Eq. (2-3). This equation assumes that ΔS_i, the change in speed over an increment i, is caused by the load torque. When the net torque over the increment is in the direction of the load torque then the schedule voltage is increased in this increment to counter the load torque. If the schedule delivers a voltage which is too large, then the change in speed over the increment will indicate that the schedule voltage should be reduced. A problem can arise if the P or e^2 control action creates a net torque in the opposite direction of the load torque. In this case, the schedule adaptation equation responds in opposition to the P or e^2 control action and not to the load torque itself. To eliminate this possibility, the plant plus the P or e^2 controller should have effectively dominant real eigenvalues.

Controller stability was tested by simulating an overloaded motor and a rapid loss of load. Figure 2-24 shows the adapted voltage schedule of an overloaded motor. The schedule voltages have saturated here at +84 V while the load actually required 100 V to achieve best performance. Speed control was degraded by this saturation but the controller operation remained stable.

A rapid reduction in load was also simulated to determine if the schedule could adapt quickly to the new load conditions. As will be discussed later, a step change (reduction here) in the load can cause a

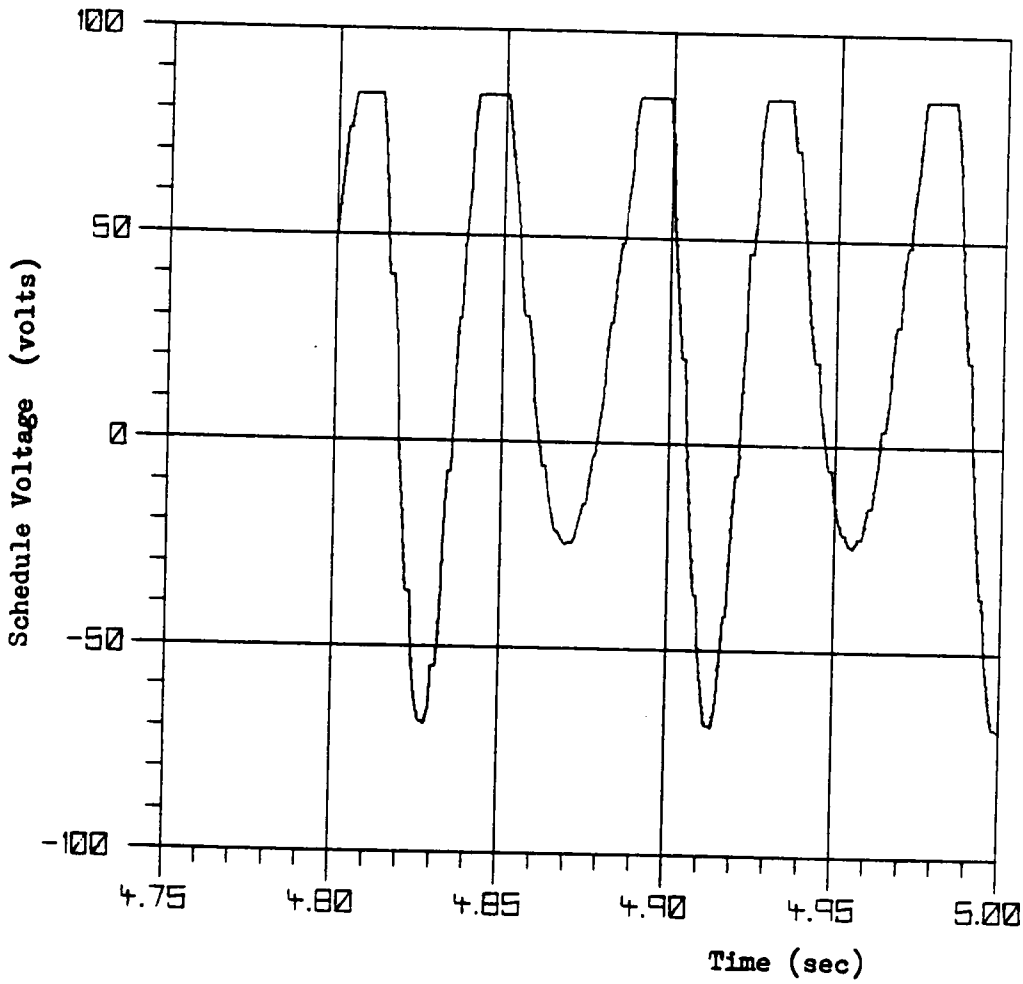


Fig. 2-24. Plot of the adapted voltage schedule of an overloaded motor. Schedule values are limited to 84 volts.

"ringing" phenomena in the schedule which damps out slowly but doesn't effect speed control significantly. A smooth reduction in the load as shown in Fig. 2-25 is easily tracked by the schedule adaptation scheme. Figure 2-26 shows the reduction of the schedule within the envelope of the decreasing load. The schedule values adapt quickly enough to achieve a mean speed error of 0.014% and fluctuation error of less than ±0.0001% within 3 cycles after the loss of the load. Speed error did not exceed 1% at any point during the reduction of the load as shown in Fig. 2-27 and Fig. 2-28.

Adaptation rate is affected by several factors which will be discussed next. It is clear from examining the adaptation equation (2-3) that the schedule adapts more quickly when ΔS_i is large and corresponds to a net load torque. This leads to a situation where it is desirable to have a small K_p (proportional control gain) which allows the schedule to adapt more quickly. A small K_p, however reduces system response to large errors especially during start-up and changes in the speed command. The error-squared feedback allows for faster adaptation and good response to large errors. The effective gain factor is very large for large errors and small in the neighborhood of the desired speed level. The schedule adapted 30% faster when the e^2 method was used over P feedback with $K_p = 14.6$ ($\xi = 1.0$).

The number of increments stored per cycle (NPC) also effects the adaptation rate. Faster adaptation was obtained with NPC = 64 than NPC = 100 (for the same K_s) because ΔS_i is larger over each increment as NPC is reduced. The lower limit on NPC is dictated by the expected frequency components of the load.

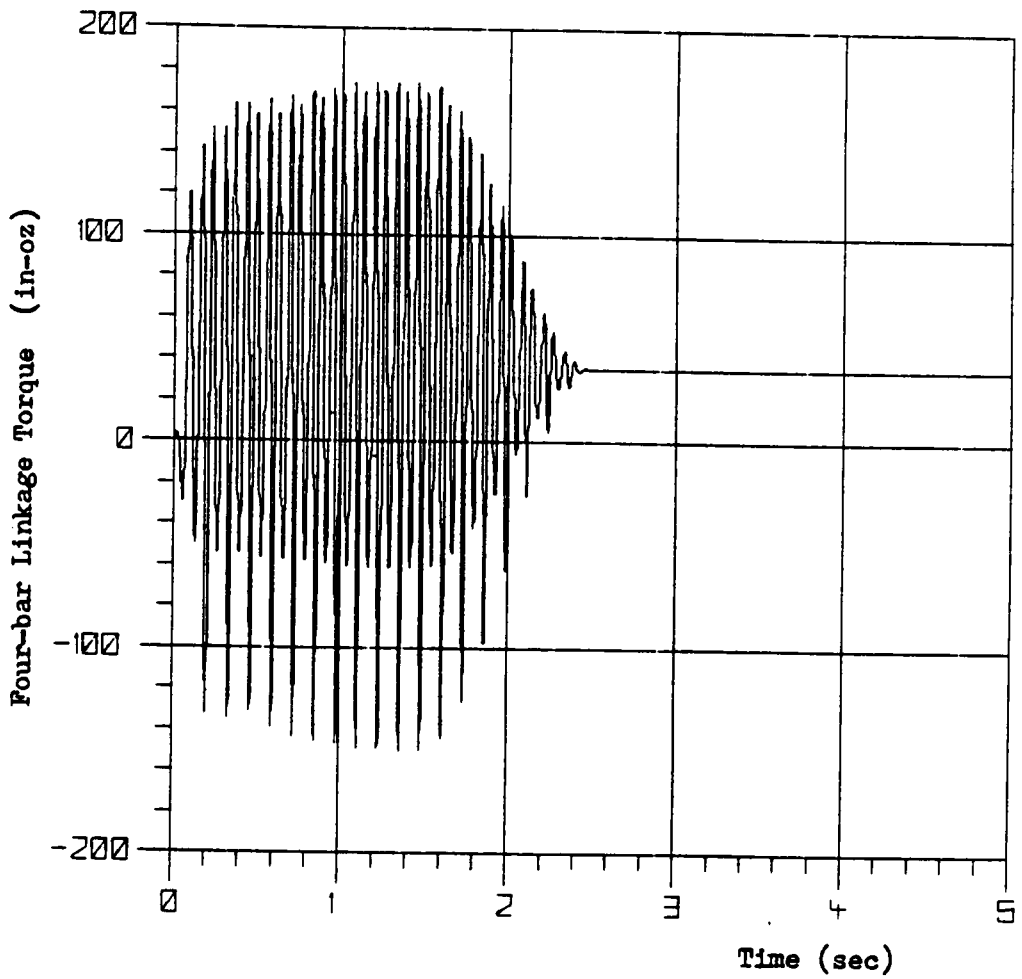


Fig. 2-25. Reduction in four-bar linkage load torque in 1 sec. Viscous load component is still present.

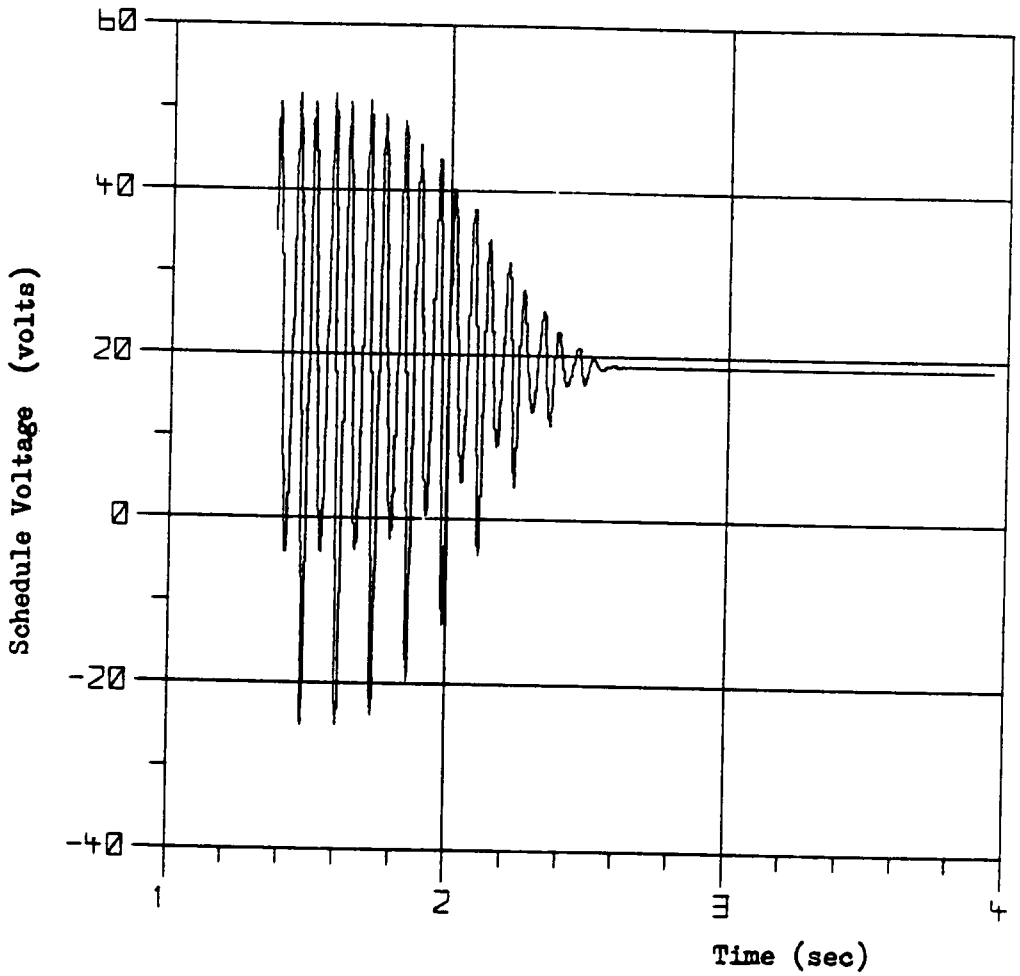


Fig. 2-26. Plot of the reduction in stored schedule values to track the decreasing load torque.

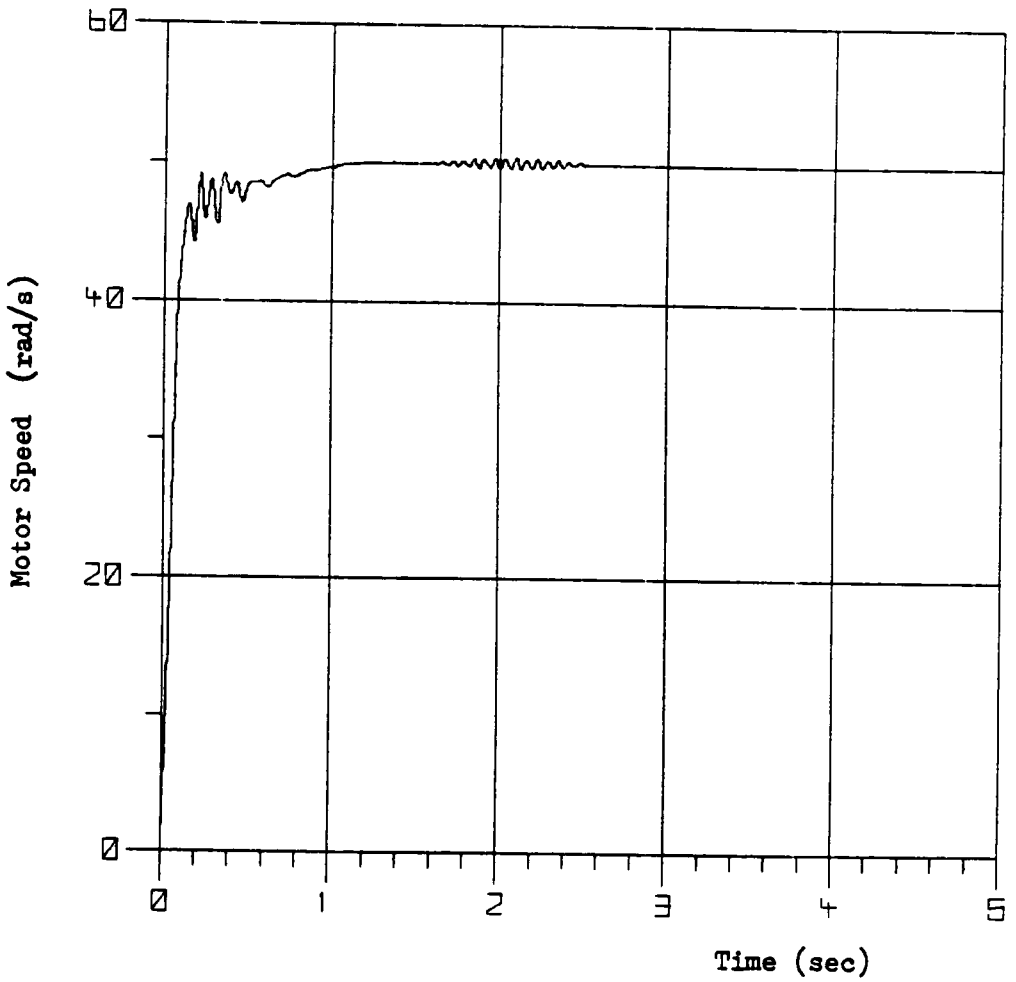


Fig. 2-27. Schedule controlled system speed response during smooth reduction in load torque between t=1.5 sec and t=2.5 sec.

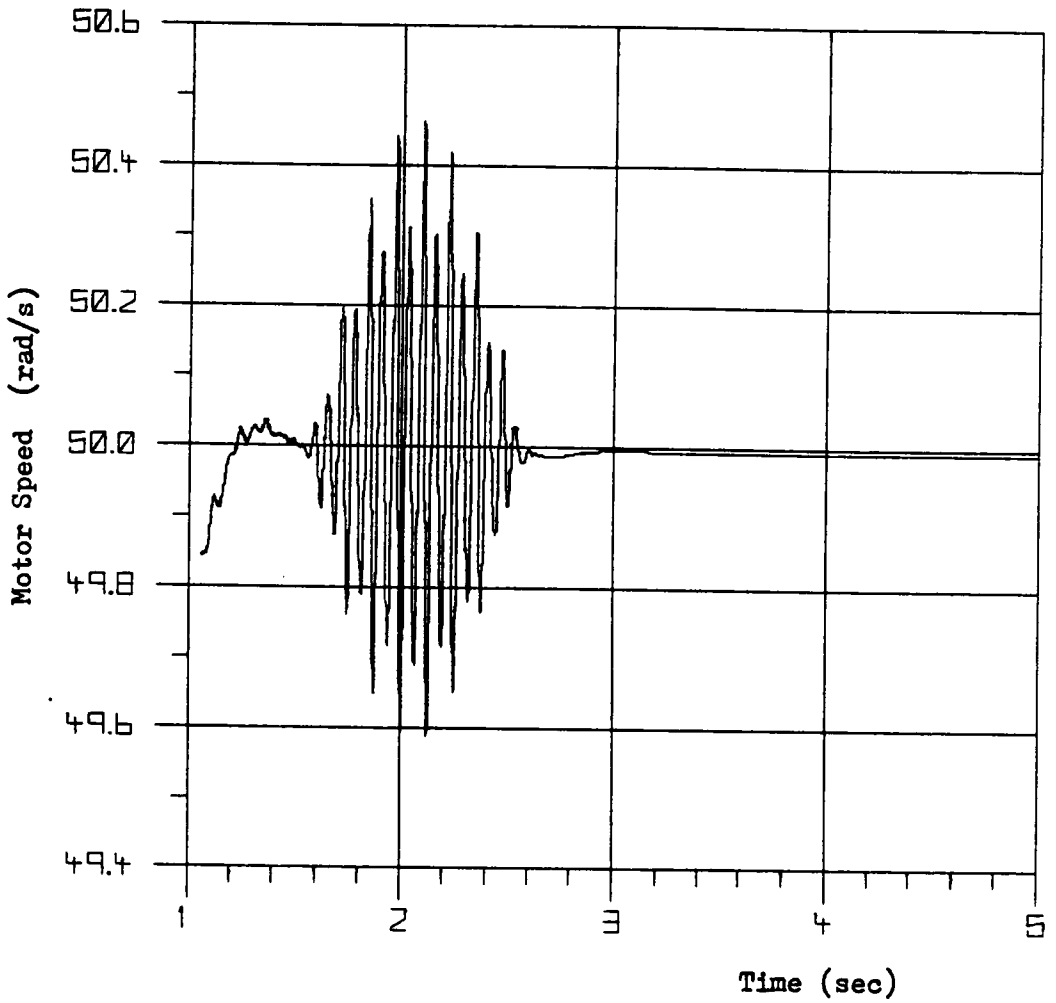


Fig. 2-28. Detail view of speed response during load reduction. The schedule adapts to hold speed error below 1%.

One additional consideration is the real time of each increment which increases with decreasing motor speed or NPC. The objective has been to change the motor torque in a given increment by changing the voltage of this increment. Motor torque is proportional to current which lags behind the voltage because of armature inductance. A square voltage pulse results in an exponential rise in current within the increment and an exponential fall in current outside the increment. This current "overflow" increases the torque in the next increment where it may not be needed by the load. The adaptation equation inherently tends to compensate for the undesirable current overflow by decreasing the voltage of the next increment. In smoothly changing voltage schedules, this compensation is not noticeable. In the periodic step or impact load, a ripple or "ringing" effect in the schedule values can be seen. Figure 2-29 shows a plot of voltage schedule values (dashed line plot) and the corresponding armature current with the ripple effect present. The step load (not shown in Fig. 2-29) occurs at about t = 2.322 s and causes the schedule voltage level to saturate at 110 V in a hopeless attempt to compensate for the step load. The current lags the voltage and does not rise fast enough to meet the load. The slope of the current response is easily calculated from the first order response of the system. The initial slope is $\Delta V_T/L_a$ (ΔV_t is the step change in terminal voltage, L_a is the armature inductance). A linear voltage amplifier is used here so the slope $\Delta V_T/L_a$ will vary with the step changes in voltage called for by the schedule in each increment. A switching amplifier or so called current amplifier will have a fixed V_T (and L_a) and therefore a fixed slope of response. Generally, the

current amplifier will achieve the desired level sooner than the voltage amplifier because the current amplifier always uses the maximum supply voltage available. However, there will always be some current lag present and therefore also some current overflow. Figure 2-29 shows how the second (#2) schedule voltage has decreased below the actual voltage required by the load to reduce the current flow to the proper level of about 7.5 A. This ringing effect is radically reduced when the schedule voltage is limited to a value which prevents the current in the first increment from exceeding 7.5 A.

Current overflow has a relative reduced effect when the real time of the increment is longer than five electrical time constants (L_a/R_a). That is, the current "overflow" into the next increment is small compared to the current flow within the desired increment.

To summarize, current "overflow" creates a slight mismatch between the scheduled torque and the load torque. The relative effect of this mismatch can be reduced by minimizing NPC, decreasing motor speed, using a current or switching amplifier with a high voltage supply and/or using a motor with a small armature inductance.

There is another consequence of the current "overflow" problem which deserves special attention. Figure 2-30 shows the effect of a non-periodic impact load (Fig. 2-30a) on the system. A random noise spike in the digital speed feedback loop could cause a similar situation but is not as likely. In either case, the schedule adapts to the ΔS_i and on the next revolution delivers voltage and current waveforms as shown in Fig. 2-30b.

Two problems now exist. The current is not needed in the first

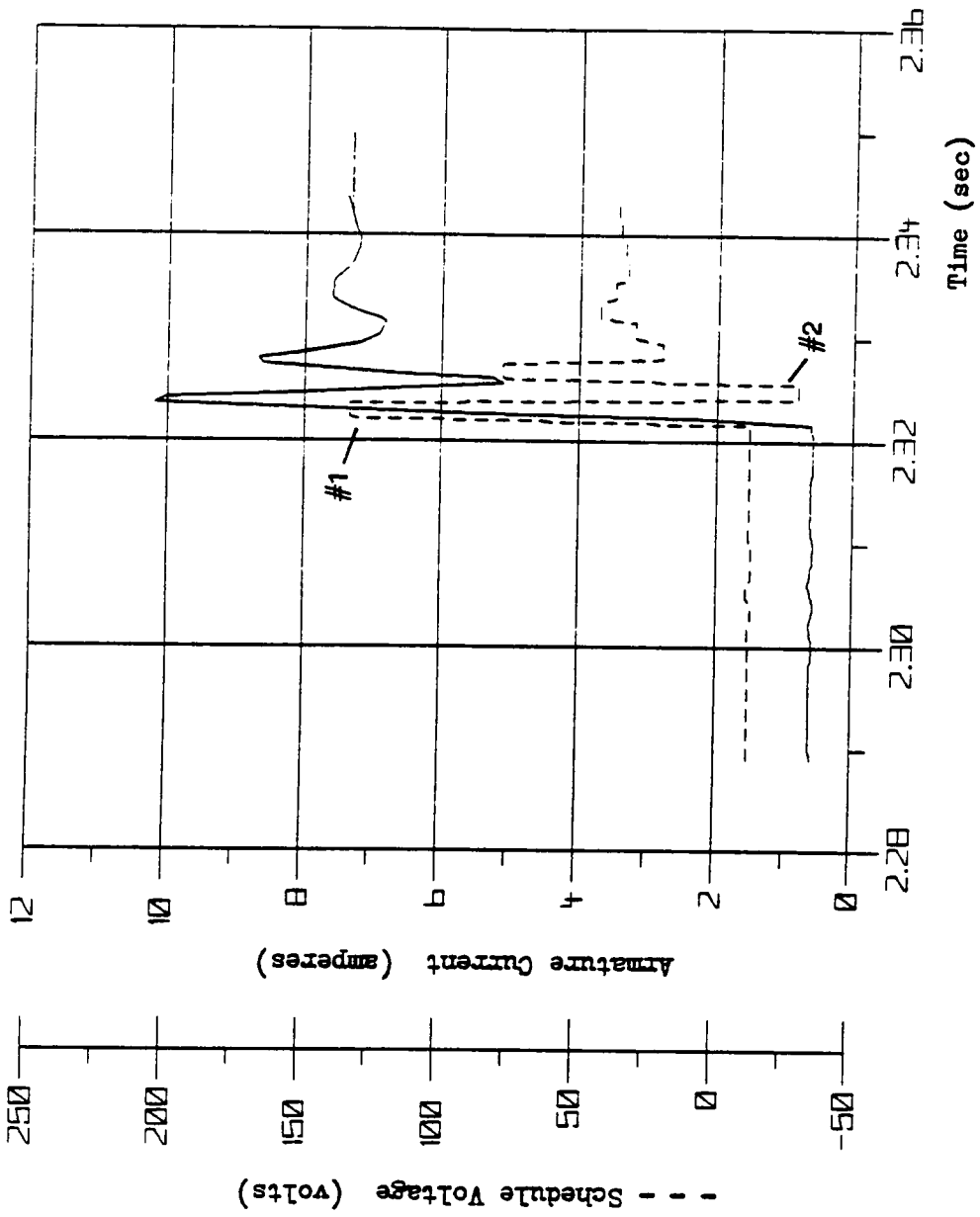


Fig. 2-29. Schedule voltage and corresponding armature current during a step increase in load torque.

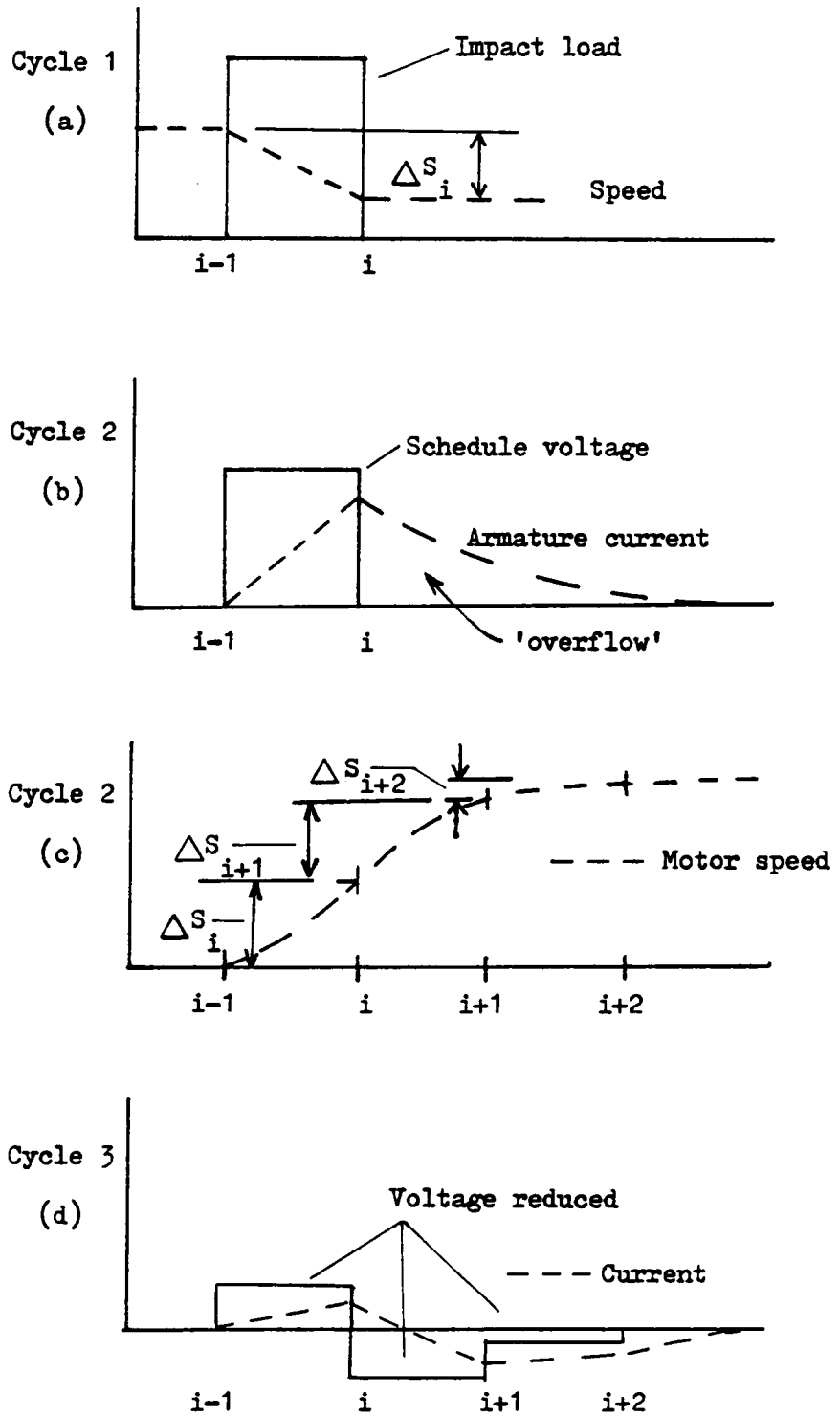


Fig. 2-30. Events following a random impact load.

increment or in the second or perhaps third. The algorithm will now respond to the ΔS_i measured in the three increments (see Fig. 2-30c). The algorithm responds to the change in speed by decreasing the voltage in each increment where ΔS_i is positive. If the adaptation equation gain K_s is not too large, the increment (i) voltage in the schedule will return to its initial value after several cycles. Increment (i+1) voltage must go negative to reduce the current flow in that increment back to its original value. Figure 2-30d shows the rippling effect "downstream" from the impact as successive increments adapt to compensate for the "overflow".

Simulation studies show this process to be slow - especially for small values of ΔS_i. Speed control was not greatly affected because the schedule could adapt rapidly (to large ΔS_i) and keep speed error within acceptable limits. Random impact load transients in the schedule could be avoided by not adapting the schedule unless a particular event occurs several times.

Gain Selection

In order to estimate the adaptation equation gain K_s, consider the following development.

The net torque over an increment (T_{net}) can be approximated:

$$T_{net} \approx J \frac{\Delta S_{imax}}{\Delta t} \qquad (2\text{-}5)$$

J = estimate of system inertia

ΔS_{imax} = maximum change in speed over an increment in the

load cycle

Δt = estimate of the real time of the increment.

T_{net} can then be used to estimate the minimum voltage which must be delivered in that increment.

$$V_{min} = \frac{T_{net} * R_a}{K_T} \tag{2-6}$$

where R_a = armature resistance

K_T = motor torque constant

Then, from Eq. (2-3), K_s is simply:

$$K_s = \frac{V_{min}}{\Delta S_{imax}} \tag{2-7}$$

By combining Eqs. (2-5,6,7), it's possible to show that K_s is not a function of ΔS_{imax}.

$$K_s = \frac{J R_a}{\Delta t K_T} \tag{2-8}$$

A speed command of 50 and NPC = 64 yields:

$$\Delta t \approx \frac{2\pi}{(NPC)(S_{pdcom})} \approx 0.002 \text{ secs}$$

and with J = 0.7, R_a = 7.0 and K_T = 29.6, the gain K_s equals 84, which was the value used in the simulations presented here. The gain K_s can easily be written as function of the command speeds for operation over a

wider speed range.

$$K_s = \frac{(S_{pdcom})(NPC)(J)(R_a)}{2\pi K_T} \tag{2-9}$$

If a fixed value of K_s is used in Eq. (2-3), then it should be calculated using the minimum speed command desired for the system.

A method for estimating the integrator gain K_v is based on Eq. (2-4). This

$$V_{OFFSET} = K_v * \frac{\int_0^{2\pi} (\text{speed error})\, d\theta_m}{2\pi} \tag{2-4}$$

equation shows that V_{OFFSET} is proportional to the average speed error integrated over one load cycle - not integrated over time. For example, if the speed command is 50 r/s and the average speed over the cycle is 49 r/s then $V_{OFFSET} = K_V * 1.0$. If an e^2 feedback loop is being used, then an error of 1 r/s corresponds to a terminal voltage of 1^2 or 1 V. Therefore, the schedule must be increased by at least 1 V and perhaps a little more if the load increases with speed. $K_v = 1$ should then bring the schedule value very close to the desired voltage. Simulation runs showed that for $K_v = 0.63$ to $K_v = 1.26$, the integration response was consistent and well behaved even at lower speeds. The number of cycles to convergence on a steady state speed varied with the load conditions. As an example, for the four-bar load and $K_v = 1.26$, steady state conditions took 4 cycles at $S_{pdcom} = 25$ r/s, 8 cycles at $S_{pdcom} = 50$ r/s and 14 cycles at $S_{pdcom} = 75$ r/s. Integrator convergence took twice as

many cycles for each load level at $K_v = 0.63$.

P control requires a different K_v gain setting. For example, if $K_p = 10$ then an error of 1 r/s corresponds to a motor voltage of 10 volts. In this case, a value of $K_v = 10$, or to be conservative, $K_v = 5.0$ might be used. V_{OFFSET} was modeled so as to saturate at 10 volts in the simulations. This was considered a realistic constraint and also helped limit the severity of response to very large errors.

CONCLUSIONS

An adaptive motor control scheme was investigated by computer simulation. The controller adapts and stores a schedule of torques to be delivered at discrete points throughout a periodic load cycle. A proportional (P) or error squared (e^2) feedback loop is included in the d.c. motor controller for rapid response to large speed errors. The schedule controller was also applied to a 3ϕ synchronous motor (without P or e^2 feedback).

The performance of the scheduling controller was compared to open-loop synchronous motor performance and to P, PI and PID controlled d.c. motor performance.

Simulation studies for the synchronous motor revealed:

1. Good speed control is possible but the variable torque angle δ makes this control method difficult and loss of synchronization is likely.
2. It is easy to fix the torque angle using the hardware already necessary for the scheduling controller. This configuration is superior.

When the torque angle δ is fixed, the resulting system is dynamically identical to the mechanically commutated d.c. motor. Two severe load cases - a periodic step load and a four-bar linkage load - were used to compare relative controller performance.

The simulation studies for the d.c. motor (which also apply to the 3ϕ AC servo-motor, i.e., fixed δ) revealed the following:

1. The PID controller gave best overall performance of the P, PI and PID configurations, though optimal gain selection for this

low order system was difficult.

2. The scheduling controller reduced steady state speed error fluctuations 12-50 times better than the best PID performance.
3. The PID controller response was only slightly faster than the scheduling controller response. However, overshoot of 10-30% seemed unavoidable in obtaining fast PID response.
4. The schedule controller integrator responded consistently over a wide range of speeds, converging on ≈ 0 speed error without overshoot. PID overshoot varied with the load.
5. The scheduling controller can adapt to high frequency periodic variations which occur in the forward loop of the system, e.g., periodic variations in load torque, load inertia and motor torque constant (due to thermal effects, variation in air gap or mild saturation).
6. The scheduling controller can also adapt to fairly rapid, non-periodic, smooth changes in load conditions.
7. The scheduling controller is simple to implement, very little knowledge of the load is required. The controller gains were much easier to find than for the PID controller.

The simulation results also showed:

8. Speed control during the initial adaptation of the schedule is not better than the P or e^2 feedback loop provides. Relatively large speed transients are necessary for the schedule to adapt quickly but can be avoided in subsequent operations by saving the adapted waveform for use at that operating point.

9. The net torque of the system including the schedule feedback loop (i.e., the e^2 or P feedback loop plus the motor and load) must be in the direction of the load torque for the schedule to adapt to the load.

10. Random impact loads and random noise do not significantly affect speed control in the short term but do cause an unnecessary and undesirable "ripple" in the stored waveform.

The simulation results did not address the long term effects (beyond 12 secs) of noise and/or random impact loads on stability and speed regulation of the schedule controlled system.

Overall, the torque scheduling controller is simple to implement, requires very little knowledge of the load, can adapt to a wide variety of load conditions and appears to give excellent speed control. The long term effect of noise and/or random impact loading can only be determined by building and testing an actual controller.

RECOMMENDATIONS

Results from the simulation studies indicate that this control scheme has potential. The next step is to build a controller and verify the performance indicated by the simulation studies. The control scheme should first be implemented on a simple d.c. motor driven system. A microcomputer with some external hardware could be used to adapt the schedule. If the microcomputer is too slow for real-time implementation, the schedule could be adapted on every other load cycle. Equation (2-4), the speed error integrator for each load cycle, could easily be constructed using a clock and a counter. The real time of each revolution could be compared to the ideal real time at a given speed command level. The difference in real time is proportional to the average speed error over the load cycle.

It would also be interesting to compare the initial adapted waveform to later waveforms after operating temperatures have reached steady state or after several hours of exposure to system noise and/or impact loading. Waveforms might also be stored for replay at another time to avoid the start-up transients which occur when the schedule adapts from zero initial conditions.

The control scheme might be more easily applied to the field winding of a separately excited motor to achieve speed control without having to regulate armature power which is typically 10 to 100 times larger than field power.

This control method shows some promise for application in 3ϕ AC servo-motor drives also. These drives have many good characteristics: brushless construction, better heat dissipation geometry, smooth torque

production and maximum energy conversion capability. A stored waveform drive used with a position encoder can be used to construct the stator flux vector ninety electrical degrees ahead of the rotor at all times. An up-down counter with separate power supply can be used with the position encoder to insure that the rotor position is always known. Each rotor position would correspond to three memory locations which contain the current values to be delivered to each of the three phase windings. This method is simpler than implementation of matrix equations in hardware as discussed in the literature review. The three values recalled from memory are proportioned to construct a stator flux vector at the proper angle. The magnitude of the stator flux vector can be "scaled" by multiplying the three values by a gain factor. This gain factor would come from the torque schedule. A convenient way to carry out this scaling is as follows. The three values from memory are sent to digital-to-analog converters (DAC) before amplification. The reference voltage supplied to the three DAC's determines the magnitudes of the voltages supplied to the amplifiers. The torque schedule value may be converted in another DAC and this output used to vary the reference voltage of the three DAC's just discussed. In this way, the torque schedule values can "scale" the magnitude of the stator flux at each increment in the rotor revolution. Synchronization is maintained at all times and the system dynamic characteristics are identical to those of a mechanically commutated d.c. motor.

REFERENCES

1. Pfaff, G., A. Weschta, and A. F. Wick, "Design and Experimental Results of a Brushless AC Servo Drive," IEEE Transactions on Industry Applications, Vol. IA-20, No. 4, July/August 1984.

2. Mounfield, W. P., "Adaptive Positioning Control by Signal Synthesis," Thesis (M.S.), Virginia Polytechnic Institute and State University, 1980.

3. Carlson, S. O., "Adaptive Control of a Four-Bar Linkage," Thesis (M.S.), Virginia Polytechnic Institute and State University, 1985.

4. Albus, J. S., "A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC)," Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME, September 1975.

5. Fitzgerald, A. E., C. Kingsley, Jr., and S. D. Umans, Electric Machinery, McGraw-Hill, 1983.

6. Dewan, S. B., G. R. Slemon, and A. Straughen, Power Semiconductor Drives, Wiley, 1984.

APPENDIX

```
C       THIS IS THE MAIN PROGRAM OF THE SYNCHRONOUS MOTOR SIMULATION.
C       3-PHASE 2-POLE SYNCH.MOTOR MODEL
C       CURRENT DRIVEN

        IMPLICIT REAL*8 (A-H,O-Z)
        REAL*8 IAT,IBT,ICT,KT,KJ,IM,IMA,IML,IOFSET
        REAL*8 IA,IB,IC,IAL,IBL,ICL,NDT,K1,K2,KJI

C       THESE ARE THE ARRAYS USED IN THE MATRIX EQ.
        DIMENSION X1(4),X2(4),C(4,9)

C       THESE ARRAYS ARE SENT TO THE PLOTTING ROUTINE.

        DIMENSION TH(6000),THD(6000),THDD(6000),TP(6000),TQF(6000)
        DIMENSION DELT(6000),IA(6000),IB(6000),IC(6000),IML(501)
        DIMENSION A(4,4),B(4,5),X(4),DX(4),U(4),Q(4)
        COMMON A,B,U
C*************************************************************************
C       THIS SECTION REQUESTS DATA FOR THE SIMULATION FROM A
C       PARAMETER FILE - - e.g. FOR010.DAT
C*************************************************************************

50      WRITE(6,*) ' ENTER PARAMETER FILE NUMBER'
        READ(5,*) M
52      READ(M,*) DT,TSTOP,IPLOT,BD,W,IM,KT,KJI,X(1),X(2)
        WRITE(6,55) DT,TSTOP,IPLOT,BD,X(2)
55      FORMAT(5X,'DT=',F6.5,4X,'TSTOP=',F4.2,4X,'IPLOT=',I3,4X,'BD='
     *,F4.2,4X,'THIC=',F7.3)
60      WRITE(6,65)
65      FORMAT(' 1 TO RUN THIS FILE',/,' 2 TO CHANGE FILE #',/,' 3 TO EX
     *IT')
        READ(5,*) J
        GO TO(80,50,1000), J

C*************************************************************************
C       THE PARAMETERS BELOW WILL BE DESCRIBED HERE.  'THE' IS THE
C       ELECTRICAL ANGLE, DSUM IS THE SUM OF SAMPLED DELTA VALUES
C       OVER THE LOAD CYCLE, P=SECS/REV ,IPB IS THE INCREMENT IN
C       THE LOAD CYCLE, THIC = INITIAL THETA, NPC=NUMBER OF INCRE-
C       MENTS PER LOAD CYCLE, TOL = ALLOWED ERROR IN POSITION
C       ENCODER(Z), NCS=THE # OF CYCLES SAVED, ODT=ORIGINAL TIME
C       STEP, DTI=TEMPORARY TIME STEP, PH1=120 DEGREES IN RADS.
C       Q() IS A MATRIX USED BY THE RUNGE-KUTTA ROUTINE.
C*************************************************************************

80      THE=0.0
        DSUM=0
```

```
      P=(2.*PI)/W
      IPB=1
      THIC=X(2)
      IOFSET=0.0
      TSUM=0
      NPC=100
      TOL=.01/DFLOAT(NPC)
      NCS=1
      PI=3.141592654
      T=0.0
      ODT=DT
      DTI=DT
      N=1

      ITIME=0
      PH1=2.094395102
      DO 100 I=1,4
100   Q(I)=0.0

C*************************************************************************
C     INPUT GAINS AND LOAD PARAMETERS FROM THE SCREEN.
C     T1=COEF OF CENTRIFUGAL LOAD COMPONENT(.06)
C     T2=COEF OF THE GRAVITY LOAD COMPONENT(10-50IN-OZ)
C     T3=COEF OF ARBITRARY SINUSOIDAL LOAD.(50 IN-OZ)
C     K1=ADAPTATION GAIN,K2=IOFSET CONTROL GAIN.
C     SLP=SLOPE OF THE SPEED CHANGE.
C*************************************************************************

      WRITE(6,*) ' T1? T2? T3?  K1?  K2?  SLP? '
      READ(5,*) T1,T2,T3,K1,K2,SLP
C*************************************************************************
C     LOAD THE INITIAL CURRENT VALUES INTO THE SCHEDULE.

      DO 110 J=1,NPC
      IML(J)=IM
110   CONTINUE

C*************************************************************************
C     THIS BLOCK OF LINES DEFINES THE A AND B MATRICES.
C*************************************************************************
120   SX=X(2)
      IF(SX.LT.0) SX=0
      NSS=1
      KJ=KJI
      A(1,1)=-KJ*BD
      A(1,2)=0.0
      A(2,1)=1.0
      A(2,2)=0.0
```

```
      B(1,1)=KJ
      B(1,2)=-KJ
      B(2,1)=0.0
      B(2,2)=0.0

C*********************************************************************
C     THIS BLOCK DEFINES A SPEED CHANGE OVER T=1.75 TO 2.75.
C*********************************************************************


150   IF(T.LT.1.75) GO TO 175
      IF(T.GT.2.75) GO TO 175
      W=50 + SLP*(T-1.75)

C*********************************************************************


C     THIS BLOCK DEFINES A BALANCED CURRENT MODEL FOR THE
C     THREE PHASE SYNCH. MOTOR.
C*********************************************************************


175   THE=W*T
      DELTA=THE-X(2)

      FBA=DSIN(X(2))
      FBB=DSIN(X(2) + PH1)
      FBC=DSIN(X(2) - PH1)

      IAT=IML(IPB)*DCOS(THE)
      IBT=IML(IPB)*DCOS(THE+PH1)
      ICT=IML(IPB)*DCOS(THE-PH1)

      AA=IAT*FBA
      AB=IBT*FBB
      AC=ICT*FBC


C*********************************************************************

C     THIS BLOCK MODIFIES THE SCHEDULE, QUANTIZES AND LIMITS
C     THE VALUES STORED.  MODIFY IS SET =1 AT THE END OF EACH
C     INCREMENT, DELTASPEED IS CALCULATED, THE SCHEDULE INCR.
C     (IPB-1) CURRENT IS MODIFIED AND THE MODIFY SWITCH SET
C     BACK TO 0.

C*********************************************************************
      X1R=X(1)
```

```
      IF(MODIFY.NE.1) GO TO 250
      DELTASPD=X1R-X1RB
      IP=IPB-1
      IF(IP.EQ.0) IP=NPC
      IML(IP)=IML(IP)*(1.-K1*DELTASPD) + IOFSET
      JCB=IML(IP)*(1024./30.)
      IML(IP)=JCB*(30./1024.)
      IF(IML(IP).GT.30) IML(IP)=30
      IF(IML(IP).LT.-30) IML(IP)=-30
      MODIFY=0


C*************************************************************************

C     TT IS THE TOTAL MOTOR OUTPUT TORQUE, TL IS THE SUM OF THE
C     LOAD TORQUE COMPONENTS APPLIED, TSUMD IS THE TOTAL LOAD
C     (INCLUDING VISCOUS DAMPING) CONVENIENT FOR PLOTTING.
C     U1 AND U2 ARE THE INPUTS TO THE SYSTEM.

C*************************************************************************
250   TT=-KT*(AA+AB+AC)
      ARG=X(2)
      IF(ARG.LT.0) ARG=0
      TL=T1*TLCC + T2*TLGG + T3*SIN(X(2))
      TSUMD=TL + X(1)*BD

      U(1)= TT
      U(2)= TL
      IF(N.EQ.1) GO TO 500

C*************************************************************************

C     THE INITIAL VALUES OF THE STATE BEFORE THE NEXT TIME STEP
C     ARE SAVED IN CASE THE STEP MUST BE TAKEN AGAIN WITH A

C     MODIFIED TIME STEP.

C*************************************************************************
      X1T=X(1)
      X2T=X(2)
      TTM=T
      DET=DE

C*************************************************************************
```

```
C       RKG IS CALLED TO MAKE THE STEP IN TIME.  RKG IS A 4TH ORDER
C       RUNGE - KUTTA ROUTINE. IF DISPLACEMENT (X(2)) IS NEG. WE
C       GO AROUND THE POSITION ENCODER ROUTINE.

C*****************************************************************
300     CALL RKG(2,DT,T,X,DX,Q)
        IF(X(2).LT.0.0) GO TO 350
        IF(NSW.EQ.1) GO TO 320
        SN=X(1)
        NSW=1
C*****************************************************************
C       THE ONLY PURPOSE OF THIS BLOCK OF FILES IS TO SIMULATE A
C       POSITION ENCODER.  AFTER EACH STEP IN TIME (BY RKG) WE
C       CHECK TO SEE IF WE ARE NEAR THE END OF AN INCREMENT,
C       IF WE OVER STEP THE TOLERANCE ZONE THE TIME STEP IS
C       MODIFIED UNTIL THE STEP IS MADE WITHIN TOLERANCES
C       (SPECIFIED BY TOL).  THE MODIFIED TIME STEP IS LABELED
C       NDT.  AT THE COMPLETION OF EACH INCREMENT WE GO TO 340,
C       FOR TIME STEPS TAKEN WITHIN AN INCREMENT, WE GO TO 350.

C*****************************************************************
320     TV1N=X(2)/(2.*PI*NCS)
        TV2N= TV1N - AINT(TV1N)
        PIPB=TV2N*NCS*NPC + 1
        IPT=PIPB
        TGT=DFLOAT(IPB)/DFLOAT(NPC)

        IF(TV2N.LT..1.AND.IPB.EQ.NPC) TGT=0

        IF(TV2N.LT.TGT+TOL.AND.TV2N.GT.TGT-TOL) GO TO 340
        IF(IPT.EQ.IPB) GO TO 350


        TV10=X2T/(2.*PI*NCS)
        TV20=TV10- AINT(TV10)
        IF(TGT.EQ.0) TGT=1
        NDT=DABS(TV20-TGT)/DABS(X(1))

        T=TTM
        X(1)=X1T
        X(2)=X2T
        DT = NDT
        DTI=NDT
        GO TO 300
C*****************************************************************
C       THE BLOCK BELOW IS EXECUTED AT THE END OF EACH INCREMENT
C       IPB IS THE # OF THE INCREMENT (THEREFORE INCREASE BY ONE)
C       RETURN TO THE ORIGINAL(ODT) TIME STEP.  SN SAVES THE SPEED
C       AT THIS TIME TO BE USED IN THE NEXT DELATSPD CALCULATION.
```

```
C       DSUM INTEGRATES ANOTHER DELTA, MODIFY IS SWITCHED ON, IF WE
C       ARE ALSO AT THE COMPLETION OF A LOAD CYCLE THE LAST GROUP OF
C       LINES FIND THE AVERAGE DELTA ERROR OVER THE LOAD CYCLE,
C       WRITE IT TO THE SCREEN, RESET IPB FROM NPC+1 TO 1, AND
C       CALCULATES IOFSET.

C*************************************************************************


340     IPB=IPB+1
        DT=ODT
        X1RR=SN
        SN=X(1)
        DSUM=DELTA+DSUM
        MODIFY=1
        IF(IPB.NE.NCS*NPC+1) GO TO 350
        AVGDE=DSUM/NPC
        DE=1.25-AVGDE
        WRITE(6,*) AVGDE
        DSUM=0
        IPB=1
        IOFSET=-K2*DE
C*************************************************************************
C       THE LINES BELOW CHECK TO SEE IF THE SIMULATION SHOULD
C       BE TERMINATED(TSTOP), CHECK THE PLOT INTERVAL AND FILL
C       THE ARRAYS DEFINED BETWEEN LINE 500 AND N=N+1. THE ARRAYS
C       ALONG WITH N (THE # OF POINTS TO BE PLOTTED) ARE SENT TO
C       OUTPUT.FOR TO PREPARE FOR SCREEN OR PAPER PLOTS.

C*************************************************************************
350     ITIME=ITIME+1
        IF(T.GT.TSTOP) GO TO 999
        IF(ITIME.EQ.IPLOT) GO TO 400
        GO TO 120
400     ITIME=0
        TSUM=0.0
500     IA(N)=IAT
        IB(N)=IML(IPB)
        IC(N)=IOFSET
        DELT(N)=DELTA
        THDD(N)=DX(1)
        TP(N)=T
        TQF(N)=TSUMD
        THD(N)=X(1)
        TH(N)=IPB
        N=N+1
        IF(N.EQ.2) GO TO 300
```

```
      GO TO 120
999   N=N-1
      CALL OUTPUT(TSTOP,N,TP,THD,THDD,DELT,TQF,IA,IB,IC,TH)
      GO TO 60
1000  STOP
      END
```

```
C       THIS SUBROUTINE IS FOR SIMULATING THE 4-BAR LINKAGE
C       AS WAS DONE BY S.O.CARLSON P.184 OF HIS THESIS.

        SUBROUTINE SOC(Q1D,Q1,TLC,TLG,RJI,NSW)

        IMPLICIT REAL*8 (A-H,O-Z)
        REAL*8 I1,I2,I3,M1,M2,M3
C       CONSTANTS

        G=386.0
        M1=0.0827
        M2=0.007
        M3=0.104
        I1=0.648
        I2=0.00635
        I3=5.0
        R1=0.375
        R2=1.625
        R3=2.06
        TH1=3.1415
        TH2=0.0
        TH3=0.0
        AA1=0.75
        AA2=3.25
        AA3=5.5
        AA4=6.0

C       CALCULATE THE INITIAL POSITION OF THE MECHANISM

        IF(NSW.GT.0) GO TO 50

        SGN3=-1
        SGN2=1

        CC3=(AA1**2+AA3**2+AA4**2-AA2**2)/2./AA1/AA3
        CC2=(AA1**2+AA2**2+AA4**2-AA3**2)/2./AA1/AA2



C       CALCULATE THE CURRENT POSITION OF THE MECHANISM

50      A=DSIN(Q1)
        B=DCOS(Q1)-AA4/AA1
        C=CC2-AA4/AA2*DCOS(Q1)
        FF2=(-A+SGN2*DSQRT(A*A+B*B+C*C))/(C-B)
        Q2=DATAN(FF2)*2
        B3=DCOS(Q1)-AA4/AA1
```

```
      C3=CC3-AA4/AA3*DCOS(Q1)
      FF3=(A+SGN3*DSQRT(A*A+R3**2-C3**2))/(B3+C3)
      Q3=2.*DATAN(FF3)

C     CALCULATE THE INFLUENCE COEFFICIENTS

C     ROTATIONAL

      G3=AA1/AA3*DSIN(Q1-Q2)/DSIN(Q3-Q2)
      G2=AA1/AA2*DSIN(Q1-Q3)/DSIN(Q3-Q2)

C     ACCELERATION

      H2=AA1/AA2*(DSIN(Q3-Q2)*DCOS(Q1-Q3)*(1.-G3)-
     $ DSIN(Q1-Q3)*DCOS(Q3-Q2)*(G3-G2))/DSIN(Q3-Q2)**2

      H3=AA1/AA3*(DSIN(Q3-Q2)*DCOS(Q1-Q2)*(1.-G2)-
     $ DSIN(Q1-Q2)*DCOS(Q3-Q2)*(G3-G2))/(DSIN(Q3-Q2)**2)

C     LINK 2 TRANSLATIONAL INFLUENCE COEF.

      G2R=AA1**2+R2*R2*G2*G2+2.*AA1*R2*G2*DCOS(-Q1+Q2+TH2)
      GBHR=R2*R2*G2*H2+AA1*R2*(H2*DCOS(-Q1+Q2+TH2)-
     $ G2*DSIN(-Q1+Q2+TH2)*(G2-1.))

C     GRAVITATIONAL INFLUENCE COEF.

      X1=R1*DCOS(Q1+TH1)
      X2=AA1*DCOS(Q1)+G2*R2*DCOS(Q2+TH2)
      X3=R3*DCOS(Q3+TH3)*G3

C     CALCULATE TLG AND TLC

      TLG=G*(M1*X1+M2*X2+M3*X3)
      TLC=Q1D*Q1D*(I2*G2*H2+I3*G3*H3+M2*GBHB)

C     CALCULATE RJI

      RJI=1./(I1+I2*G2*G2+I3*G3*G3+M2*G2B)

      RETURN
      END
```

```
C       THIS IS A STANDARD 4TH ORDER RUNGE-KUTTA INTEGRATION
C       ROUTINE. NEQ IS THE ORDER OF THE SYSTEM, THE A AND
C       B MATRICES IN THE EQ. XDOT=AX+BU ARE IN COMMON
C       WITH THIS ROUTINE, THEY ARE FOUND IN THE MAIN PROGRAM.
C       H IS THE TIME STEP, X IS TIME, Y IS THE STATE, DY IS
C       THE RATE OF CHANGE OF THE STATE OVER H.
C       THATS ENOUGH HINTS -- YOU CAN TAKE IT FROM HERE!
C
C
C
        SUBROUTINE RKG (NEQ,H,X,Y,DY,Q)
        IMPLICIT REAL*8 (A-H,O-Z)
        DIMENSION A(2),Y(NEQ),Q(NEQ),DY(NEQ)
        A(1)=0.29289321881345247 5
        A(2)=1.7071067811865475 2
        H2=0.5*H
        CALL DERIV(Y,DY,X)
        DO 13 I=1,NEQ
        B=H2*DY(I)-Q(I)
        Y(I)=Y(I)+B
13      Q(I)=Q(I)+3.0*B-H2*DY(I)
        X=X+H2
        DO 20 J=1,2
        CALL DERIV(Y,DY,X)
        DO 20 I=1,NEQ
        B=A(J)*(H*DY(I)-Q(I))
        Y(I)=Y(I)+B
20      Q(I)=Q(I)+3.0*B-A(J)*H*DY(I)
        X=X+H2
        CALL DERIV(Y,DY,X)
        DO 26 I=1,NEQ
        B=(H*DY(I)-2.0*Q(I))/6.0
        Y(I)=Y(I)+B
26      Q(I)=Q(I)+3.0*B-H2*DY(I)
        RETURN
        END

        SUBROUTINE DERIV(X,DX,T)
        IMPLICIT REAL*8 (A-H,O-Z)
        DIMENSION X(4),DX(4),A(4,4),B(4,5),U(5)
        DIMENSION X1(4),X2(4)
        COMMON A,B,U
        CALL MATMPY(A,4,4,X,4,1,X1)
        CALL MATMPY(B,4,5,U,5,1,X2)
        CALL MATADD(X1,4,1,X2,DX)
        RETURN
        END
```

```
      SUBROUTINE MATMPY(A,MA,NA,B,MB,NB,C)
      IMPLICIT REAL*8 (A-H,O-Z)
C  C=A*B
C  A IS MA BY NA
C  B IS MB BY NB
C  C IS MA BY NB
C  NA MUST EQUAL MB
      DIMENSION A(MA,NA),B(MB,NB),C(MA,NB)
      DO 2 J=1,MA
      DO 2 I=1,NB
      P=0.0
      DO 1 K=1,NA


1     P=P+A(J,K)*B(K,I)
2     C(J,I)=P
      RETURN
      END
      SUBROUTINE MATADD(A,MA,NA,B,C)
      IMPLICIT REAL*8 (A-H,O-Z)
C  C=A+B
C  A,B AND C ARE MA BY NA
      DIMENSION A(MA,NA),B(MA,NA),C(MA,NA)
      DO 1 I=1,MA
      DO 1 J=1,NA
1     C(I,J)=A(I,J) + B(I,J)
      RETURN
      END
```

```
C     OUTPUT. FOR
C     THIS SUBROUTINE IS USED TO PREPARE THE PLOT DATA INTO
C     THE FORMAT USED BY THE PLOTTING ROUTINE FASTPLT2.FOR.
C
C     WE SEND TSTOP, N (# OF POINTS TO PLOT), TP (TIME ARRAY) ,
C     THD (SPEED ARRAY), THDD (ACCEL ARRAY) , ETC.  THE OTHER
C     ARRAYS WERE DEFINED AS NEEDED.
C
      SUBROUTINE OUTPUT(TSTOP,N,TP,THD,THDD,DELT,TT,IA,IB,IC,TH)
      IMPLICIT REAL*8 (A-H,O-Z)
C
C     ONLY 4 ARRAYS ARE LOADED AT ONE TIME TO BE SENT TO THE
C     FASTPLT2.  THE 3 'Y' AXIS VARIABLES PLT1-PLT3 AND THE
C     'X'-AXIS WAS ALWAYS TIME.
C
      DIMENSION PLT1(6000),PLT2(6000),PLT3(6000),TPLOT(6000)
C
C
C     THE LINES BELOW ALLOW THE OPTION OF ECSAPE FROM
C     THE PLOTTING ROUTINE, THEN BY CALL 'LIST' A CHOICE
C     OF VARIABLES FOR PLOTTING IS PRESENTED.  THREE MUST
C     BE SELECTED. THEN CALL ZOOM.  ZOOM ALLOWS SELECTED
C     VIEWING BETWEEN ANY TWO POINTS ON THE TIME AXIS
C     FOR A DETAILED LOOK AT THE PLOT.  THEN CALL LOADUP.
C     THIS LOADS THE ARRAYS PTL1-PTL3 AND TPLOT WITH THE RANGE
C     OF VALUES CHOSEN BY THE ZOOM FUNCTION.  THEN CALL
C     FASTPLT2 AND DELIVER THE CHOSEN ARRAYS FOR DISPLAY
C     ON THE SCEEN, OR TO BE PLOTTED ON THE VERSAJUNK PLOTTER.
C
 5     WRITE(6,10)
10    FORMAT(' TYPE',/,' 1 TO EXIT',/,' 2 TO CONTINUE')
      READ(5,*) I
      GO TO(900,200), I
200   CALL LIST(I1,I2,I3)
      K=0
      L=0
      CALL ZOOM(N,TSTOP,K,L)
220   NC=1

      GO TO(5,300,320,340,360,380,400,420,440), I1

240   NC=2

      GO TO(650,300,320,340,360,380,400,420,440), I2

260   NC=3
```

```
      GO TO(650,300,320,340,360,380,400,420,440), I3

300   CALL LOADUP(THD,K,L,TP,TPLOT,PLT1,PLT2,PLT3,NC)
      GO TO 500
320   CALL LOADUP(DELT,K,L,TP,TPLOT,PLT1,PLT2,PLT3,NC)
      GO TO 500
340   CALL LOADUP(THDD,K,L,TP,TPLOT,PLT1,PLT2,PLT3,NC)
      GO TO 500
360   CALL LOADUP(IA,K,L,TP,TPLOT,PLT1,PLT2,PLT3,NC)
      GO TO 500
380   CALL LOADUP(IB,K,L,TP,TPLOT,PLT1,PLT2,PLT3,NC)
      GO TO 500
400   CALL LOADUP(IC,K,L,TP,TPLOT,PLT1,PLT2,PLT3,NC)

      GO TO 500
420   CALL LOADUP(TT,K,L,TP,TPLOT,PLT1,PLT2,PLT3,NC)
      GO TO 500
440   CALL LOADUP(TH,K,L,TP,TPLOT,PLT1,PLT2,PLT3,NC)

500   GO TO(240,260,650), NC
600   WRITE(6,610)
610   FORMAT(' 1 PLOT TO SCREEN',/,' 2 PRINT TO SCREEN')
      WRITE(6,620)
620   FORMAT(' 3 PLOT TO FILE',/,' 4 PRINT TO FILE')

      READ(5,*) M
      GO TO(650,660,670,680), M

650   M=L-K+1
      CALL FASTPLT2(PLT1,PLT2,PLT3,TPLOT,I1,I2,I3)


      GO TO 5
660   GO TO 600
670   GO TO 600
680   GO TO 600
900   RETURN
      END
      SUBROUTINE LIST(I1,I2,I3)
      WRITE(6,*) 'VARIABLE LIST - PLEASE SELECT ONE'
      WRITE(6,10)
10    FORMAT(' 1 THD',/,' 2 DELTA',/,' 3 THDD',/,' 4 IA',/,' 5 IR',/
     *  ,' 6 IC',/,' 7 TT',/,' 8 TH')
C2345678
      READ(5,*) I1,I2,I3
      I1=I1+1
      I2=I2+1
      I3=I3+1
      RETURN
      END
```

```
      SUBROUTINE ZOOM(NP,TSTOP,K,L)
      IMPLICIT REAL*8 (T)
5     WRITE(6,10)
10    FORMAT(' ZOOM?',/,' 1 FOR YES',/,' 2 FOR NO')
      READ(5,*) I
      GO TO(20,30), I
20    WRITE(6,*) 'STARTING TIME?'
      READ(5,*) ST
      IF(ST.GT.TSTOP) GO TO 20
25    WRITE(6,*) 'FINAL TIME?'
      READ(5,*) FT
      IF(FT.LT.ST) GO TO 25
      IF(FT.GT.TSTOP) FT=TSTOP
      CK=(ST/TSTOP)*NP + 1.0
      CL=(FT/TSTOP)*NP
      K=CK
      L=CL
      GO TO 50
30    K=1
      L=NP
50    RETURN
      END

      SUBROUTINE LOADUP(PLDR,K,L,TTP,TPLOT,PLT1,PLT2,PLT3,NC)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*4 TPLOT,PLT1,PLT2,PLT3
      DIMENSION PLDR(1),TTP(1),TPLOT(1),PLT1(1),PLT2(1),PLT3(1)
      JS=L-K+1
      GO TO(5,15,25), NC
5     TPLOT(1)=JS
      PLT1(1)=JS
      DO 10 I=2,JS+1
      TPLOT(I)=TTP(I+K-2)
      PLT1(I)=PLDR(I+K-2)
10    CONTINUE
      GO TO 50
15    PLT2(1)=JS
      DO 20 I=2,JS+1
      PLT2(I)=PLDR(I+K-2)
20    CONTINUE
      GO TO 50

25    PLT3(1)=JS
      DO 30 I=2,JS+1
      PLT3(I)=PLDR(I+K-2)
30    CONTINUE

50    RETURN
      END
```

```
C       THIS ROUTINE IS USED ONLY TO LABEL THE 'Y'-AXIS OF THE
C       PLOTS.  EIGHT LABELS ARE AVAILABLE (I=1-8).  SEE THE
C       PLOT 10 OWNERS MANUAL FOR DETAILS.
C
        SUBROUTINE YLABEL(I)
        DIMENSION L1(4),L2(6),L3(5),L4(3),L5(7),L6(3),L7(3),L8(3)

        DATA L1/3,84,72,68/
        DATA L2/5,68,69,76,84,65/
        DATA L3/4,84,72,68,68/
        DATA L4/2,73,65/
        DATA L5/6,73,77,40,84,72,41/
        DATA L6/2,73,67/
        DATA L7/2,84,84/
        DATA L8/2,84,76/
        I=I-1
        GO TO(10,20,30,40,50,60,70,80), I
10      CALL VSTRIN(L1)
        GO TO 90
20      CALL VSTRIN(L2)
        GO TO 90
30      CALL VSTRIN(L3)
        GO TO 90
40      CALL VSTRIN(L4)
        GO TO 90
50      CALL VSTRIN(L5)
        GO TO 90
60      CALL VSTRIN(L6)
        GO TO 90
70      CALL VSTRIN(L7)
        GO TO 90
80      CALL VSTRIN(L8)

90      RETURN
        END
```

```
C       THIS IS THE MAIN PROGRAM OF THE D.C. MOTOR SIMULATION.
C       THIS PROGRAM IS USED FOR P,PI,PID AND SCHEDULE CONTROLLER
C       SIMULATIONS.
C

        IMPLICIT REAL*8 (A-H,O-Z)
        REAL*8  LA,KT,KJ,JT,JM,JL,KF,KE,KJI,KS,KV,KD,NDT
        REAL*8  IA,IB,IC,IAL,IBL,ICL,LND,KP,KI,KA,ODT
        DIMENSION X1(4),X2(4),C(4,9)

C       THESE ARRAYS ARE SENT TO THE PLOTTING ROUTINE.

        DIMENSION TH(6000),THD(6000),THDD(6000),TP(6000),TQF(6000)
        DIMENSION DELT(6000),IA(6000),IB(6000),IC(6000),VAL(501)
C       THESE ARRAYS ARE USED IN THE MATRIX EQ. XDOT=AX+BU FOR
C       SOLUTION IN RKG.FOR... 4TH ORDER RUNGA-KUTTA ROUTINE.


        DIMENSION A(4,4),B(4,5),X(4),DX(4),U(4),Q(4)
        COMMON A,B,U
C*************************************************************************


C       THIS SECTION REQUESTS DATA FOR THE SIMULATION FROM A
C       PARAMETER FILE.
C
C*************************************************************************
50      WRITE(6,*) ' ENTER PARAMETER FILE NUMBER'
        READ(5,*) M
52      READ(M,*) DT,TSTOP,IPLOT,SP,T2,T3,T4,T5,AL,BD
        WRITE(6,55) DT,TSTOP,IPLOT
55      FORMAT(5X,'DT=',F6.5,4X,'TSTOP=',F4.2,4X,'IPLOT=',I3,4X)
        WRITE(6,56) SP,T2,T3,T4,T5,AL
56      FORMAT(2X,'SP=',F6.2,3X,'T2=',F6.2,3X,'T3=',F6.2,3X,'T4=',F6.2,
     *3X,'T5=',F4.3,3X,'AL=',F8.6)
60      WRITE(6,65)
65      FORMAT(' 1 TO RUN THIS FILE',/,' 2 TO CHANGE FILE #',/,' 3 TO EX
     *IT')
        READ(5,*) J
        GO TO(80,50,1000), J

C*************************************************************************

C       THE PARAMETERS BELOW WILL BE DESCRIBED HERE.  NPC IS THE # OF
C       INCREMENTS IN ONE LOAD CYCLE.  KS IS THE SCHEDULE GAIN AS A
C       FUNCTION OF THE SPEED COMMAND.  R,KT,KE,AND LA ARE THE MOTOR
C       PARAMETERS RESISTANCE,TORQUE CONSTANT,BACK-EMF CONSTANT, AND INDUCTANCE.
```

```
C       DP IS VISCOUS DAMPING COEF.  PH1= 120 DEGREES IN RADIANS.
C       N AND IPB WILL BE DEFINED LATER.  ODT IS THE ORIGINAL
C       DELTA T OR TIME STEP.   DTI IS AN INTERMEDIATE TIME STEP.
C       TOL IS THE PERCENT ERROR ALLOWED IN THE POSITION ENCODER
C       FOR EACH INCREMENT.  W IS THE PERCENT OF THE SPEED COMMAND
C       AT WHICH THE SCHEDULE IS TURNED ON. THE REMAINING VARIABLES
C       WILL BE EXPLAINED LATER.

C*************************************************************************
80      NPC=64
        KS=1.68*SP
        NCS=1
        R=7.03
        KT=29.63
        KE=.209
        LA=AL
        DP=.35
        PH1=2.094395102
        PI=3.141592654
        N=1
        IPB=1
        ODT=DT
        DTI=DT
        TOL=.01/DFLOAT(NPC)
        W=.8
        T=0.0
        TSUM=0
        SN=0
        X1RB=0
        ITIME=0
        VM=0.0
        VINT=0
        DO 100 I=1,4
100     Q(I)=0.0

C*************************************************************************


C       INPUT GAINS FROM THE SCREEN FOR THIS RUN.  KA IS THE POWER
C       AMPLIFIER GAIN, KP IS THE PROPORTIONAL LOOP GAIN,  KI IS THE
C       INTEGRATOR GAIN, KV IS THE INTEGRATOR GAIN IN THE SCHEDULE
C       CONTROLLER,  KS IS THE SCHEDULE ADAPTATION GAIN
C       VLTH = VOLTAGE LIMIT OF THE SCHEDULE 'HIGH'
C       VLTL = VOLTAGE LIMIT OF THE SCHEDULE 'LOW'
C
        WRITE(6,*)' KA? KP? KI? KV?   KS?  VLTH?  VLTL?'
        READ(5,*) KA,KP,KI,KV,KS,VLTH,VLTL
```

```
C       SPDCOM IS THE SPEED COMMAND INPUT FROM THE PARAMETER FILE.
C       DO 110 LOADS THE SCHEDULE WITH ITS INITIAL (VM) VALUES.

        SPDCOM=SP
        VLT=(VLTH-VLTL)/2.0
        DO 110 J=1,NPC
        VML(J)=VM
110     CONTINUE

C*************************************************************************
C        INITIAL VALUES OF THE STATES

        DX(1)=0.0
        DX(2)=0.0
        DX(3)=0.0

        X(1)=0.0
        X(2)=0.0
        X(3)=0.0



C*************************************************************************
C       THIS CALL IS TO A PROGRAM BY S.O. CARLSON WHICH

C       CALCULATES THE TORQUE REQUIREMENTS OF THE FOUR-BAR
C       LINKAGE  (LOAD CASE I).
C       THE SUBROUTINE USES CURRENT SPEED(X(1)) AND POSITION (X(2))
C       TO CALCULATE CENTRIFUGAL TORQUE COMPONENT(TLCC), GRAVITY
C       TORQUE COMP.(TLGG) AND THE INVERSE INERTIA RJI(RJI=1/J).
C       NSS IS A SWITCH TO INITIALIZE A SUBROUTINE IN SOC.


150     CALL SOC(X(1),X(2),TLCC,TLGG,RJI,NSS)
        NSS=1
        KJ=RJI
C       ENTRIES IN THE 'A' MATRIX AND 'B' MATRIX ARE DEFINED


        A(1,1)= -KJ*(BD+DP)
        A(1,2)=0.0
        A(1,3)=KT*KJ
        A(2,1)=1.0
        A(2,2)=0.0
        A(2,3)=0.0
        A(3,1)=-KE/LA
        A(3,2)=0.0
        A(3,3)=-R/LA
```

```
      B(1,1)=-KJ
      B(1,2)=0.0
      B(2,1)=0.0
      B(2,2)=0.0
      B(3,1)=0.0
      B(3,2)=1./LA
C*********************************************************************

      X1R=X(1)
C     THE NEXT FOUR LINES DECIDE WHEN THE SCHEDULE IS TO BE
C     ADAPTED.  'MODIFY' IS SET EQUAL TO 1 AT THE COMPLETION
C     OF EACH LOAD CYCLE.  'NSW'=1 ANY TIME AFTER MOTOR SPEED
C     EXCEEDS 80% OF SPDCOM.

C*********************************************************************
      IF(MODIFY.NE.1) GO TO 160
      IF(NSW.EQ.1) GO TO 152
      IF(X(1).LT.W*SP) GO TO 155
      NSW=1
C
C     IN THE NEXT FEW LINES THE CHANGE IN SPEED OVER THE
C     INCREMENT, DELTASPEED, IS CALCULATED. THE SCHEDULE
C     IS STARTED ONLY IF DELTASPEED IS LESS THAN .1.
C     THE SS=1 MEANS THE SCHEDULE HAS BEEN STARTED SO
C     CONTINUE THE MODIFICATION OF THE SCHEDULE.
C     DELTASPEED IS TRUNCATED TO 12-BIT ACCURACY.
C     LINES 153-155 UPDATE THE LAST SCHEDULE VALUE,TRUNCATE
C     TO 12-BIT,AND BOUND THE SCHEDULE BY VLTH AND VLTL.
C
152   DELTASPD=X1R-X1RB
      IP=IPB-1
      IF(IP.EQ.0) IP=NPC
      IF(SS.EQ.1) GO TO 153
      CK=DABS(DELTASPD)
      IF(CK.GT..1) GO TO 155
      MBW=DELTASPD*2048

      DELTASPD=MBW/2048.
      SS=1
153   VML(IP)=(VML(IP) - KS*DELTASPD) + VOFFSET
      JCB=VML(IP)*(2048./VLT)
      VML(IP)=JCB*(VLT/2048.)
      IF(VML(IP).GT.VLTH) VML(IP)=VLTH
      IF(VML(IP).LT.VLTL) VML(IP)=VLTL
155   MODIFY=0
C*********************************************************************
```

```
C       THIS BLOCK OF FILES SWITCHES TLP=1 ON AND OFF TO CREATE
C       THE PERIODIC STEP LOAD IF NEEDED.  THE LOAD APPLIED IS SUMMED
C       INTO ONE TERM TSUM, TSUMD IS A SUM OF ALL THE LOAD COMPONENTS
C       SENT TO THE PLOTTER i.e. INCLUDES THE VISCOUS COMPONENTS.
C
160     IF(DSIN(X(2)).LT..3) GO TO 162
        TLP=1.0
        GO TO 163
162     TLP=0.0
163     TSUM= TLGG + T5*TLCC
        TSUMD=T2*TLG+TLGG+(BD+DP)*X(1)+TLCC+T4*TLP
C***********************************************************************
C        THIS BLOCK OF FILES SWITCHES THE SPEED COMMAND AND UPDATES
C       THE SCHEDULE GAIN KS.
C
        IF(T.LT.T2) GO TO 170
        IF(NS.EQ.1) GO TO 170
        SPDCOM=SPDCOM + 25
        KS=1.68*SPDCOM
        NS=1
C***********************************************************************
C       THIS BLOCK OF FILES CONTAINS THE P,I AND D CONTROL
C       EQUATIONS.  THE INTEGRATOR VINT BELOW DOES NOT
C       INTEGRATE ERROR OVER TIME BUT OVER DISPLACEMENT.
C       THE DERIVATIVE TERM IS ACTUALLY DERIVATIVE FEEDBACK,
C       NOT THE DERIVATIVE OF THE SPEED ERROR.
C       VINT DOES INTEGRATE OVER TIME IN PI OR PID CONTROL
C       VINT INTEGRATES OVER DISPLACEMENT FOR THE SCHEDULE CONTROLLER


170     SPDERR=SPDCOM-X1R
        ASPDERR=((X1T+X(1))/2.) - SPDCOM
        VINT=VINT-ASPDERR*(X(2)-X2T)
        DTI=DT
        VDER=-KD*DX(1)
        VPROP=KP*SPDERR
        VTERM=VPROP + KI*VINT + VDER
C***************************************************************************
C       THE BLOCK OF LINES BELOW DEFINE THE INPUTS TO THE SYSTEM
C       U(1) AND U(2).  VTERM IS THE SUM OF THE PID FEEDBACK FED
C       TO THE POWER AMPLIFIER.  U(2) IS LIMITED AS SHOWN.
C


C***************************************************************************


        U(1)=TSUM
        CAT=KA*VTERM
        IF(CAT.GT.84) CAT=84
```

```
      IF(CAT.LT.-84) CAT=-84
      U(2)=CAT + T3*VHL(IPB) + VOFFSET
      IF(U(2).GT.VLTH) U(2)=VLTH
      IF(U(2).LT.VLTL) U(2)=VLTL
      IF(N.EQ.1) GO TO 500
C*************************************************************************
C     THE STATES ARE SAVED IN CASE THE TIME STEP IS MODIFIED
C     FOR ANOTHER STEP.
200   X1T=X(1)
      X2T=X(2)
      X3T=X(3)
      TSAVE=T
C*************************************************************************
C     AT LINE 300 RKG IS CALLED TO MAKE THE NEXT STEP IN
C     TIME.  DERIV IS A ROUTINE WITHIN RKG--IT CALCULATES
C     THE PRESENT DERIVATIVE TO BE USED IN THE DERIVATIVE
C     FEEDBACK (D).
C*************************************************************************
300   CALL RKG(3,DT,T,X,DX,0)
      CALL DERIV(X,DX,T)
C*************************************************************************
C     THE ONLY PURPOSE OF THE NEXT BLOCK OF FILES IS TO
C     SIMULATE A POSITION ENCODER.  RKG MAKES A STEP IN

C     TIME AND DISPLACEMENT X(2), IF THE TIME STEP IS TOO
C     LONG i.e. X(2) STEPS BEYOND THE END OF THE INCREMENT
C     (ALSO BEYOND THE TOLERANCE ALLOWED) THE TIME STEP IS
C     SHORTENED--A NEW DTI IS DEFINED AND RKG IS CALLED AGAIN
C     THE PROCESS CONTINUES UNTIL X(2) IS WITHIN TOLERANCES.
C*************************************************************************
      IF(X(2).LT.0) GO TO 345
320   TV1N=X(2)/(2.*PI*NCS)
      TV2N= TV1N - AINT(TV1N)
      PIPB=TV2N*NCS*NPC + 1
      IPT=PIPB
      TGT=DFLOAT(IPB)/DFLOAT(NPC)
      IF(TV2N.LT..1.AND.IPB.EQ.NPC) TGT=0

      IF(TV2N.LT.TGT+TOL.AND.TV2N.GT.TGT-TOL) GO TO 340
      IF(IPT.EQ.IPB) GO TO 345

325   TV10=X2T/(2.*PI*NCS)
      TV20=TV10- AINT(TV10)
      IF(TGT.EQ.0) TGT=1
      NDT=DABS(TV20-TGT)/DABS(X(1))
```

```
      T=TSAVE
      X(1)=X1T
      X(2)=X2T
      X(3)=X3T
      DT=NDT
      DTI=NDT
      GO TO 300

C*************************************************************************
C     THE PROGRAM STEPS TO 340 AT THE END OF EACH ENCODER INCREMENT
C     AND THE INCREMENT NUMBER (IPB) OF THE LOAD CYCLE IS INCREASED
C     BY 1.  THE TIME STEP IS RETURNED TO ITS ORIGINAL VALUE ODT.

C     SN SAVES THE SPEED AT THE END OF THE INCREMENT, X1RB MOVES
C     THE PREVIOUS SN TO THE DELTASPEED EQU. ABOVE.  MODIFY =1 IS THE
C     SWITCH WHICH TELLS THE ALGORITHM TO MODIFY THE SCHEDULE.
C     THE NEXT LINE CHECKS TO SEE IF THE LOAD CYCLE HAS BEEN
C     COMPLETED.  IF IT HAS NOT WE GO TO THE PLOTTING ROUTINE (345).
C     IF THE CYCLE IS COMPLETE THEN CALCULATE VOFFSET AND LIMIT
C     IT TO 10 VOLTS,  THEN RESET VINT=0 AND IPB TO 1 i.e. THE
C     FIRST INCREMENT IN THE NEXT LOAD CYCLE.
C


340   IPB=IPB+1
      DT=ODT
      X1RB=SN
      SN=X(1)
      MODIFY=1
      IF(IPB.NE.NCS*NPC+1) GO TO 345
      VOFFSET=KV*VINT
      IF(VOFFSET.GT.10) VOFFSET=10
      IF(VOFFSET.LT.-10) VOFFSET=-10
      VINT=0
      IF(IPB.EQ.NPC+1) IPB=1
C*************************************************************************
C     THE BLOCK OF LINES WHICH FOLLOW PRINT THE SIMULATION TIME TO
C     SCREEN EVERY 300 TIME STEPS,  CHECKS TO SEE IF THE TIME HAS
C     EXCEEDED TSTOP (END OF THE SIMULATION),  AND ON THE PLOT
C     INTERVAL IT LOADS THE ARRAYS WITH THE SIMULATION VALUES SHOWN
C     BELOW.  AT THE COMPLETION OF THE RUN THE PROGRAM CALLS
C     'OUTPUT'  WHICH IS A PLOTING ROUTINE.
C
C*************************************************************************
345   ITIME=ITIME+1
      IWRITE=IWRITE+1
      IF(IWRITE.NE.300) GO TO 350
      WRITE(6,*) T
      IWRITE=0
```

```
350     IF(T.GT.TSTOP) GO TO 999
        IF(ITIME.EQ.IPLOT) GO TO 400
        GO TO 150
400     ITIME=0
500     IA(N)=X(3)
        IB(N)=VML(IPB)
        IC(N)=VTERM
        DELT(N)=U(2)
        THDD(N)=DX(1)
        TP(N)=T
        TQF(N)=TSUMD
        THD(N)=X(1)
        TH(N)=VOFFSET
C       WRITE(6,*) N,TP(N),THDD(N),TQF(N),TH(N)
        N=N+1
        IF(N.EQ.2) GO TO 200
        GO TO 150
999     N=N-1
        CALL OUTPUT(TSTOP,N,TP,THD,THDD,DELT,TQF,IA,IB,IC,TH)
        GO TO 60
1000    STOP
        END
```

The vita has been removed from the scanned document