To the Graduate Council:

I am submitting herewith a thesis written by Nan Ren Huang entitled "Disturbance Observer Based Motion Controller Using Digital Signal Processor". I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

James C. Hung, Major Professor

We have read this thesis
and recommend its acceptance:

Accepted for the Council:

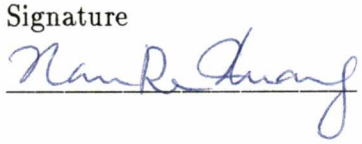
Vice Provost
and Dean of The Graduate School

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a Master's degree at The University of Tennessee, Knoxville, I agree that the Library shall make it available to borrowers under rules of the Library. Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of the source is made.

Permission for extensive quotation from or reproduction of this thesis may be granted by my major professor, or in his absence, by the Head of Interlibrary services when, in the opinion of either, the proposed use of the material is for scholarly purposes. Any copying or use of the material in this thesis for financial gain shall not be allowed without my written permission.

Signature



Date

12/31/90

DISTURBANCE OBSERVER BASED MOTION CONTROLLER USING DIGITAL SIGNAL PROCESSOR

A Thesis
Presented for the
Master of Science
Degree

The University of Tennessee, Knoxville

Nan Ren Huang

May 1991

ACKNOWLEDGMENTS

The author wishes to express sincere appreciation and gratitude to Dr. James C. Hung for his invaluable guidance and consultation. Without his help and encouragement this thesis would not have been written.

The author also wishes to thank Dr. J. M. Bailey and Dr. D. Rosenberg for their assistance and for review of this work. He is also grateful to his wife Hsiang Huei Huang for her patience and moral support. Special acknowledgement goes to Dr. Yasuhiko Dote of Muroan Institute of Technology. His knowledge and efforts were a major contribution to this thesis.

This research was performed at University of Tennessee under contract from Texas Instrument DSP Group, Houston.

DEDICATED TO

my parents Mr. and Mrs.

Shun-Ann Huang

for your encouragement and help
in token of the love I have for you

Abstract

Accurate performance of servomotor drives is required in many industrial applications on today's automatically controlled machines. This thesis presents the robust control technique applied to a motion controller. A disturbance observer was designed first. For quick response to a reference speed command, a predictive controller was then designed. A disturbance observer and a predictive controller were combined to form a robust system for a motor controller. This control arrangement makes it possible to have a wide range of motor parameter variation, and also provides stable operation.

The control system was then simulated using PC-SIMNON. After computer simulation verfication, the speed control system was implemented using two microprocessors: TMS320C14 for computation and INTEL 80286 for monitoring and user interface. Both simulation and experimental results show superior operation of the system with the disturbance observer.

Contents

List of Figures

Chapter 1

Introduction

The primary motivation for this work is the question of robustness. A robust feedback system continues to yield satisfactory performance in the event of a disturbance, approximation, or modeling uncertainty.

Many industrial applications require robust or parameter insensitive speed control. This work combines an equivalent disturbance observer with a predictive controller to form a robust system for motor control. This control arrangement makes it possible to have a wide range of motor parameter variation, and also provides stable operation. Moreover, it also provides a means to suppress disturbances.

This investigation will first review basic disturbance rejection theory, and then the control strategy as applied to a motor controller will be described in detail. Following that, two linear control methods will be compared that are used to obtain a speed control system design for a motor controller. The two design methods used are : 1) Proportional plus predictive control and 2) Proportional, predictive plus equivalent disturbance canceller control. In this investigation, the proportional, predictive plus equivalent disturbance canceller control will be used to obtain a robust control system. After the control strategy has been planned and designed, the analysis follows.

Then the stability and parameter sensitivity problem will be answered by using direct calculation and computer simulation. The computer simulation programs used in this simulation are MATHCAD, MATLAB and SIMNON. Following is a chapter concerning digitization of the analog control system by using a direct z transformation algorithm and bilinear transformation.

After computer simulation verification, an analog model of a motor was constructed and tested with an applied disturbance. The speed control system is microcomputer based consisting of a digital signal processor (DSP) TMS320C14 with a two channel A/D and D/A data acquisition board and a 80286 based IBM PC-AT compatible computer. The DSP is used for fast computation, and the PC is used for speed monitoring and command input. The experimental results not only show good agreement with simulation results, they also show superior operation of the proportional plus disturbances canceller controller.

Chapter 2

Principle of Disturbance Observer

2.1 Introduction

The presence of a disturbance is one of the reasons for using control. Obviously every care should be taken by the designer to eliminate such disturbance sources where possible, or at least to minimize their influences. The character of the disturbances imposes fundamental limitations on the performance of a control system.

An overview of different ways to eliminate disturbances is given first. This includes feedforward and direct computation. Different ways to observe the disturbance and methods to suppress the disturbance for motor control are discussed last.

2.2 Disturbance rejection

It is often the case that the plant is influenced by extraneous events that cause the controlled output of the system to differ from the reference input. In some situations the disturbances are those which are to be expected during normal operation of the plant. The problem of disturbance rejection from a system is considered next.

If the disturbance and all states of the system can be measured then it may be possible to use these measurements as an extra input to the system to counteract the effect of the disturbance. This technique is often referred to as 'disturbance feed-forward' [2] and it is illustrated in Fig. 2.1. In this figure, the disturbance is filtered prior to its application to the forward-path network. If the compensator is known, a filter may be designed with a transfer function that is equal to the inverse of the compensator transfer function, and therefore the effect of the disturbance is totally eliminated. If the disturbance can be measured, its effect on the output of the system can be computed directly or indirectly. By the superposition principle, a compensating plant input can be generated to counter not only the disturbance effect at the output but also each state variable.

2.3 Motor control system

In this section, a motor controller system based on the disturbance observer is introduced. A disturbance canceller is designed with the idea of trying to eliminate the disturbance. In order to increase the speed of response, a predictive controller is combined with a disturbance observer to form a total control system.

Disturbance observer based controller

In Fig. 2.2, a simplified model for a motor is given. The following equation [1] describes the motor's dynamics:

$$J\dot{w} + Bw + T_l = K_t I, \tag{2.1}$$

where J is the moment of inertia, B is the viscous friction coefficient, K_t is the torque

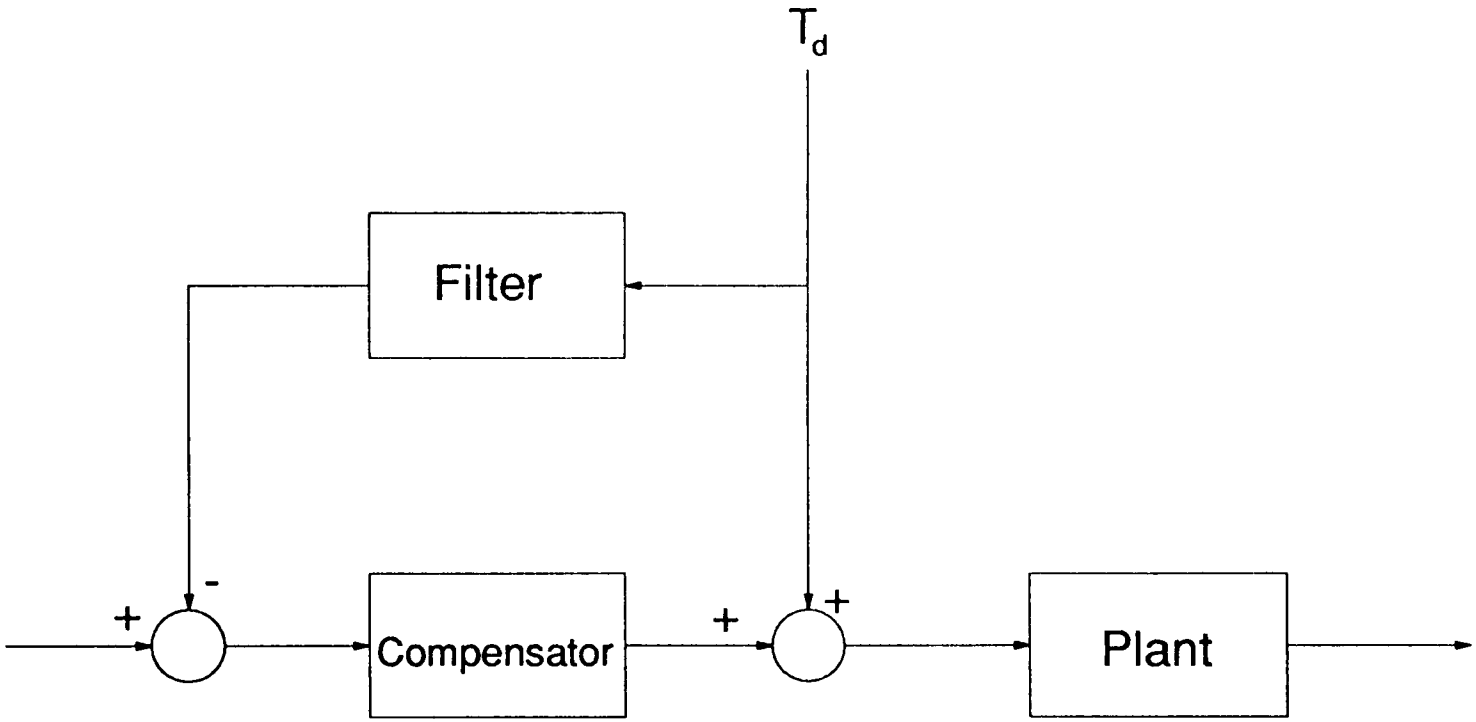


Fig. 2.1 Disturbance feedforward.

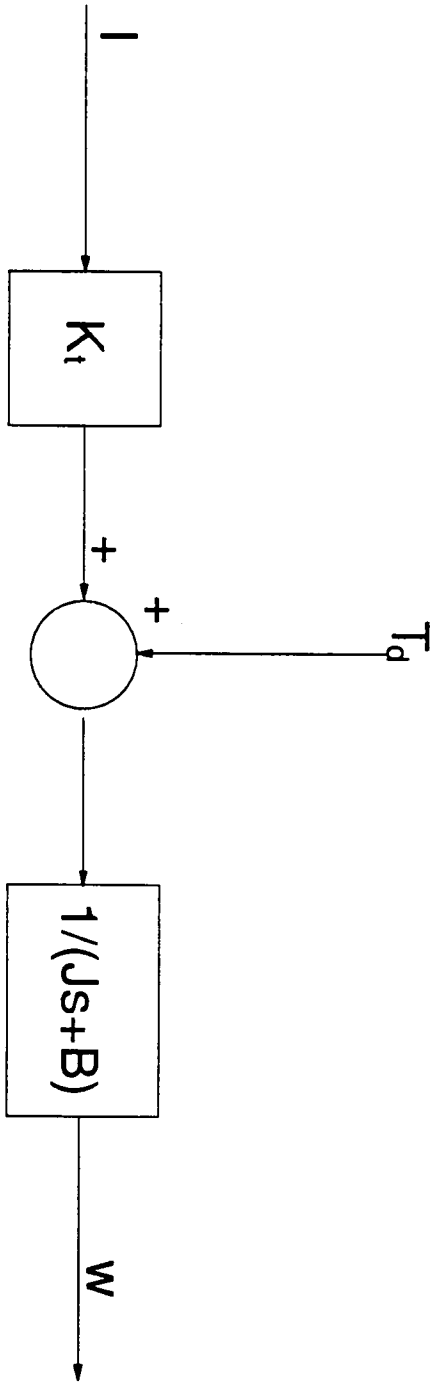


Fig. 2.2 Simplified model for motor.

constant, T_l is the external load torque, w is the motor speed, and I is the motor current. Let

$$J = \hat{J} + \Delta J$$

and

$$K_t = \hat{K}_t + \Delta K_t,$$

where $\hat{}$ denotes nominal value and Δ represents variation . An equivalent disturbance T_d is defined as

$$T_d(s) = T_l + BW(s) + \Delta JsW(s) - \Delta K_t I(s), \tag{2.2}$$

where $T_l(s)$ is the external load torque, BW(s) is the viscous friction torque, $\Delta JsW(s)$ is the torque due to parameter variation, $\Delta K_t I(s)$ is the torque ripple, and s is the Laplace transform variable.

The disturbance can be obtained from equation 2.1 and equation 2.2 as follows:

$$T_d(s) = \hat{K}_t I(s) - s\hat{J}W(s). \tag{2.3}$$

The control objectives are to determine inputs to a physical process in order to achieve the desired goals, such as obtaining a required speed for the motor. The problem in a motor controller is how to suppress the effect of a disturbance torque. The answer can be given in two steps.

The first step is to identify the disturbance torque. There are two approaches that can be used to identify the disturbance torque in the motor controller. As in Fig. 2.3, an observer can be constructed to estimate disturbance torque. The output is feed back to the inverse of the plant transfer function, $(\hat{J}s + \hat{B})$, and then the electromagnetic torque (T_e) is subtracted, yielding the estimated disturbance torque. Or, one can construct an

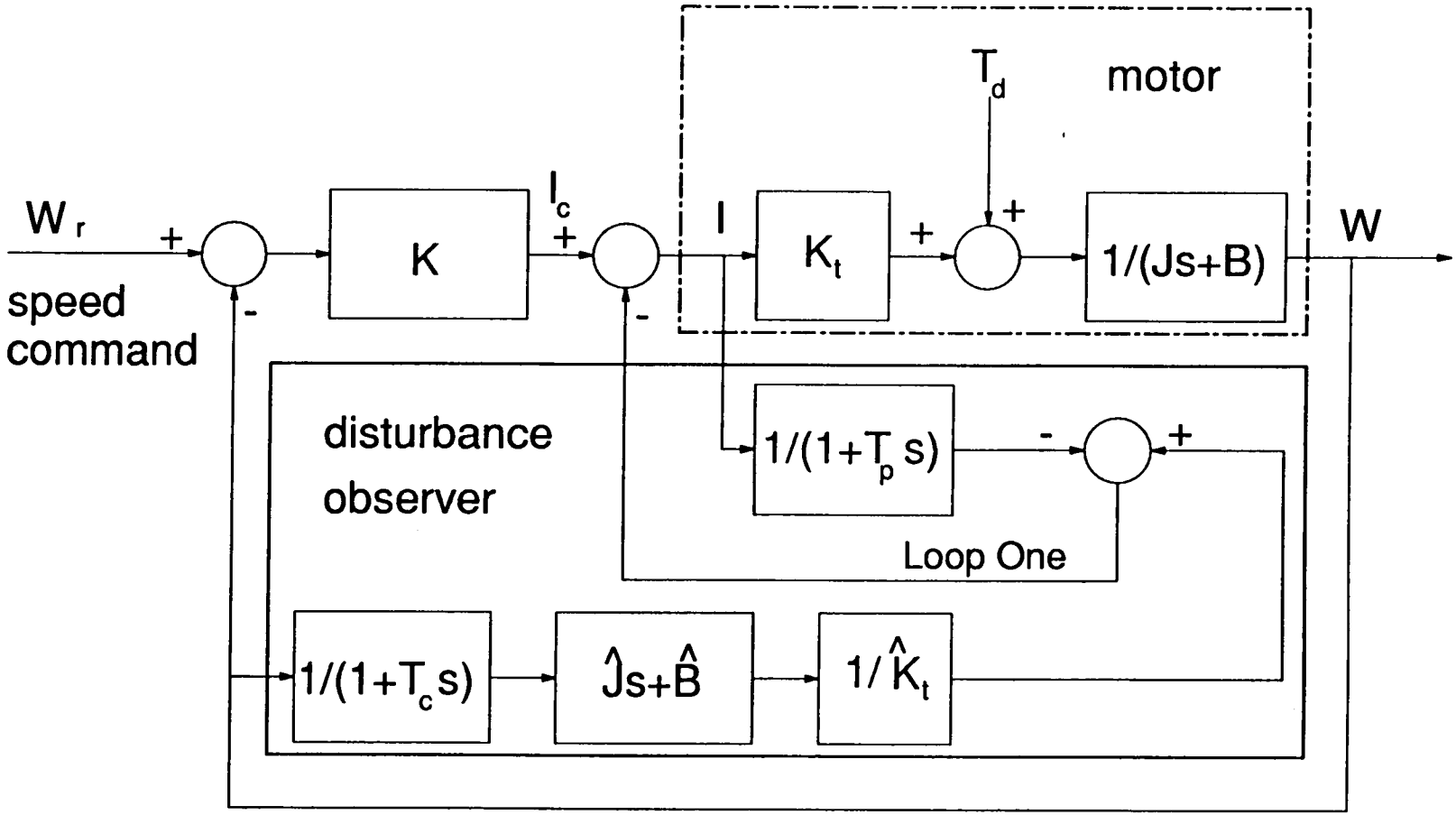


Fig. 2.3 Compensated system by the disturbance observer.

observer to estimate the equivalent disturbance current. If output is feed back to the inverse of the plant transfer function and torque constant and then the control current is subtracted, one has the estimated disturbance current. Fig. 2.3 shows the second approach of constructing an observer to estimate the equivalent disturbance current.

Before proceeding any further, it needs to be pointed out that the dynamical equation of the inverse plant and torque constant can not be realized. Consequently, a low-pass filter, $1/(1 + T_c s)$, is added to help realize such a design. Another low-pass filter ,$1/(1+T_p s)$, is added to avoid the algebraic loop in digital computation. However, once the low-pass filters are added in the control system, they have an effect on system response. The discussion of this aspect will be made in the next chapter.

The second step is to eliminate the disturbance torque. When the estimated disturbance torque or disturbance current has been identified, one can adjust the plant input or the state variables inputs to counter the disturbance effect. In order to obtain a quick command response, a proportional gain controller K is added.

Predictive controller

To obtain good transient response and to eliminate steady-state error to speed command, a predictive controller is employed. The design of the predictive controller is in essence a calculation of the inverse of the motor dynamic. In order to realize the inverse of the motor dynamic, a low pass filter, $1/(1 + T_c s)$, is added. With this predictive controller added into the control system, one has eliminated the steady-state error that the system had. Fig 2.4 shows the overall control block diagram which is the combined observer canceller and predictive controller. Adding a predictive controller results in fast response and eliminates the steady state error, as shown in Fig. 2.5.

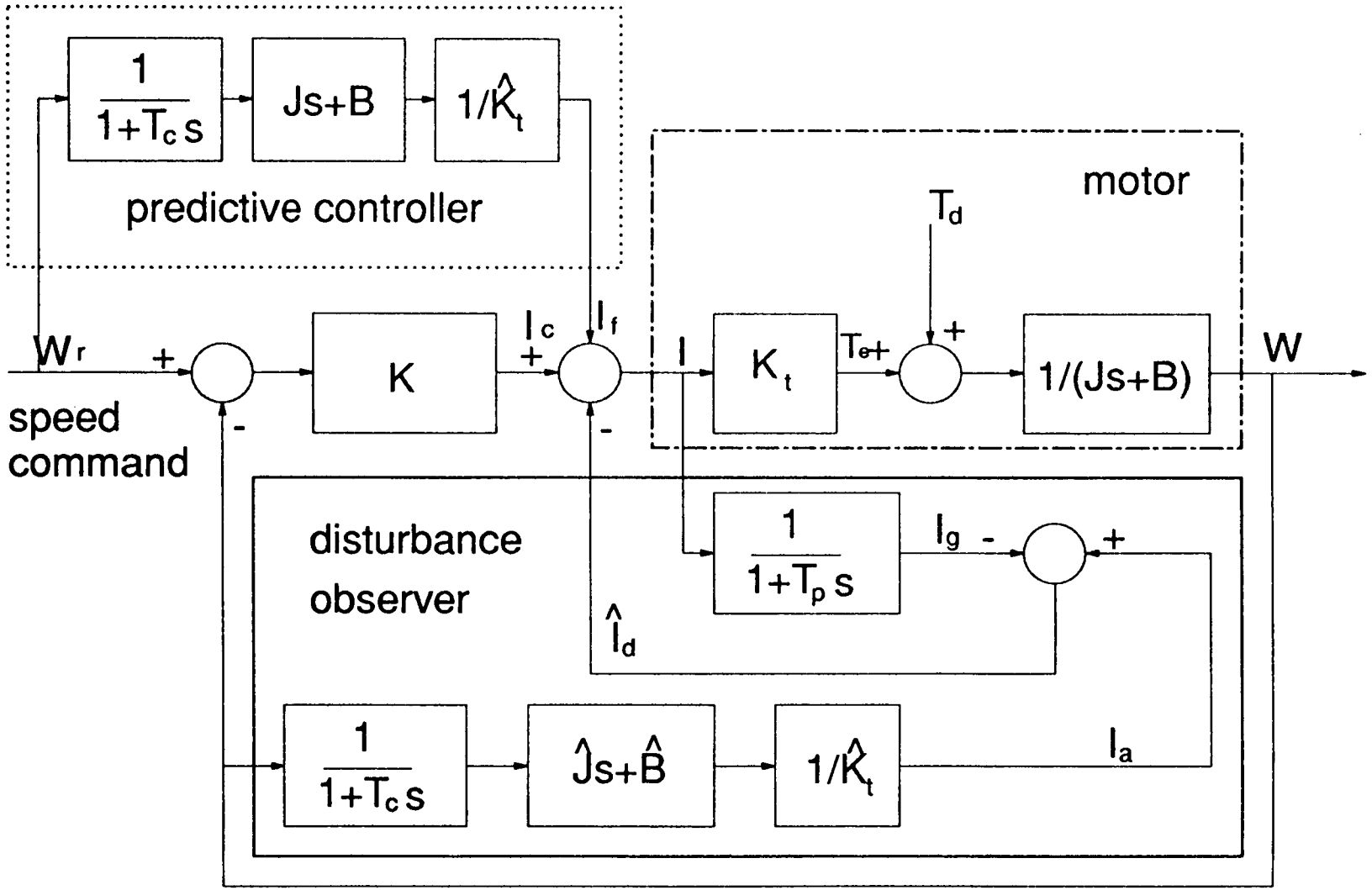


Fig. 2.4 Predictive plus disturbance canceller.

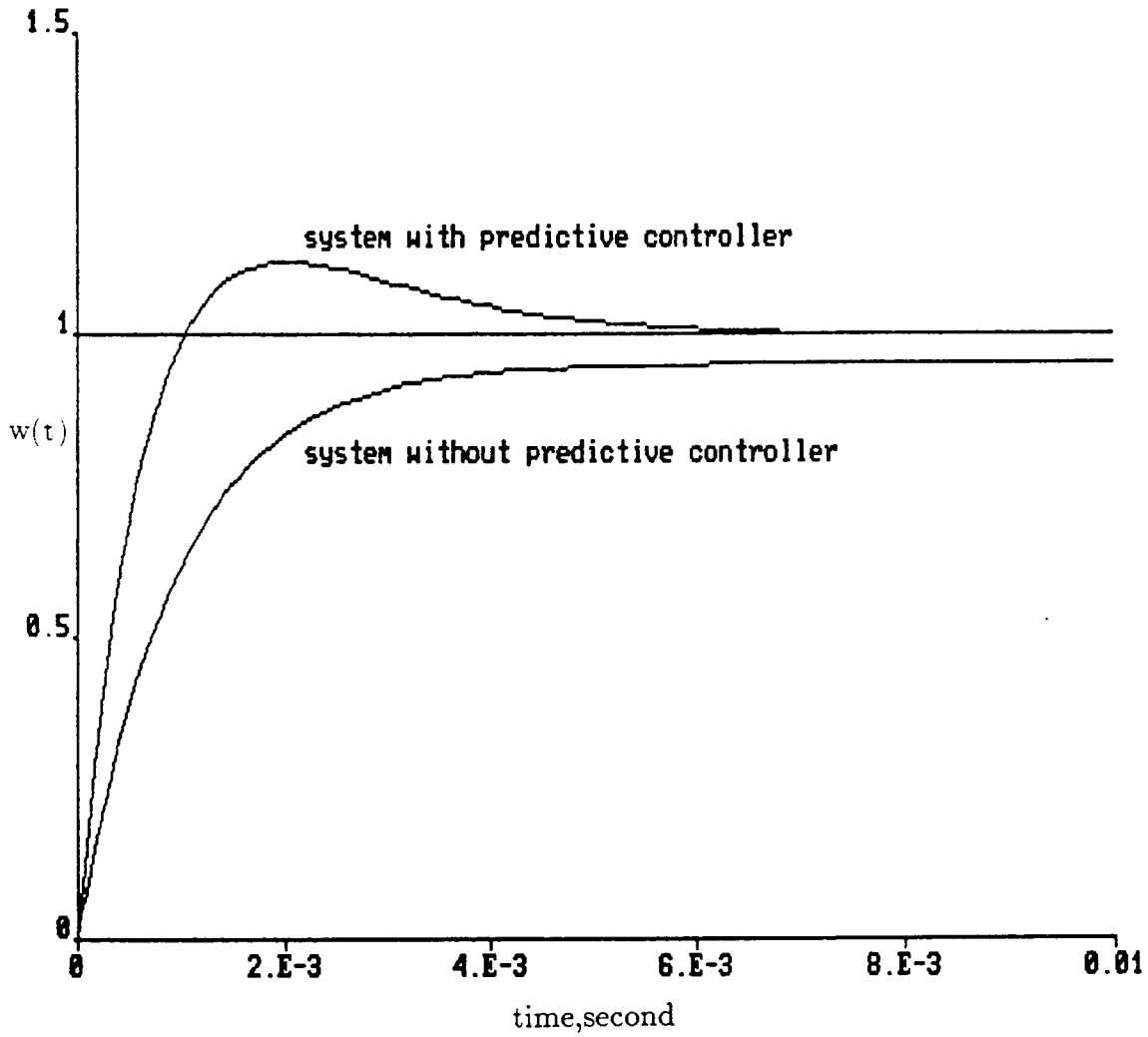


Fig. 2.5 Steady-state response comparison.

It should be mentioned that the scheme represented by Fig. 2.4 was suggested by Professor Yasuhiko Dote of Muroran Institute of Technology.

2.4 Conclusion

The main purpose of this chapter is to develop methods to suppress the effect of disturbances. A method to eliminate the disturbances has also been presented. A disturbance observer for a motor controller is shown in detail. By eliminating the estimated disturbances, a disturbance canceller for the motor controller has been designed. A motion controller is designed combining a predictive controller and a disturbance canceller.

Chapter 3

Control System Analysis

3.1 Introduction

When designing a control system, one needs to keep in mind that the most important performance criterion is the over-all system stability. If a system is unstable, then to speak of its steady-state response would be meaningless, since the system is useless unless it is first stabilized by some means of compensation. In the beginning of this chapter, the transfer function for a motor control system will be derived. Following that, the question of stability will be addressed. When the real system is in operation, the performance may be somewhat different from what is expected. With this in mind, the parameter sensitivity of the motor control system will be investigated. Two control systems, shown in Fig. 3.1 and Fig. 3.2, will be used in this chapter to analyze stability and sensitivity.

3.2 Stability analysis

A system is defined as stable if a bounded input always produces a bounded output. But before investigating the stability of the motor control system, the transfer function

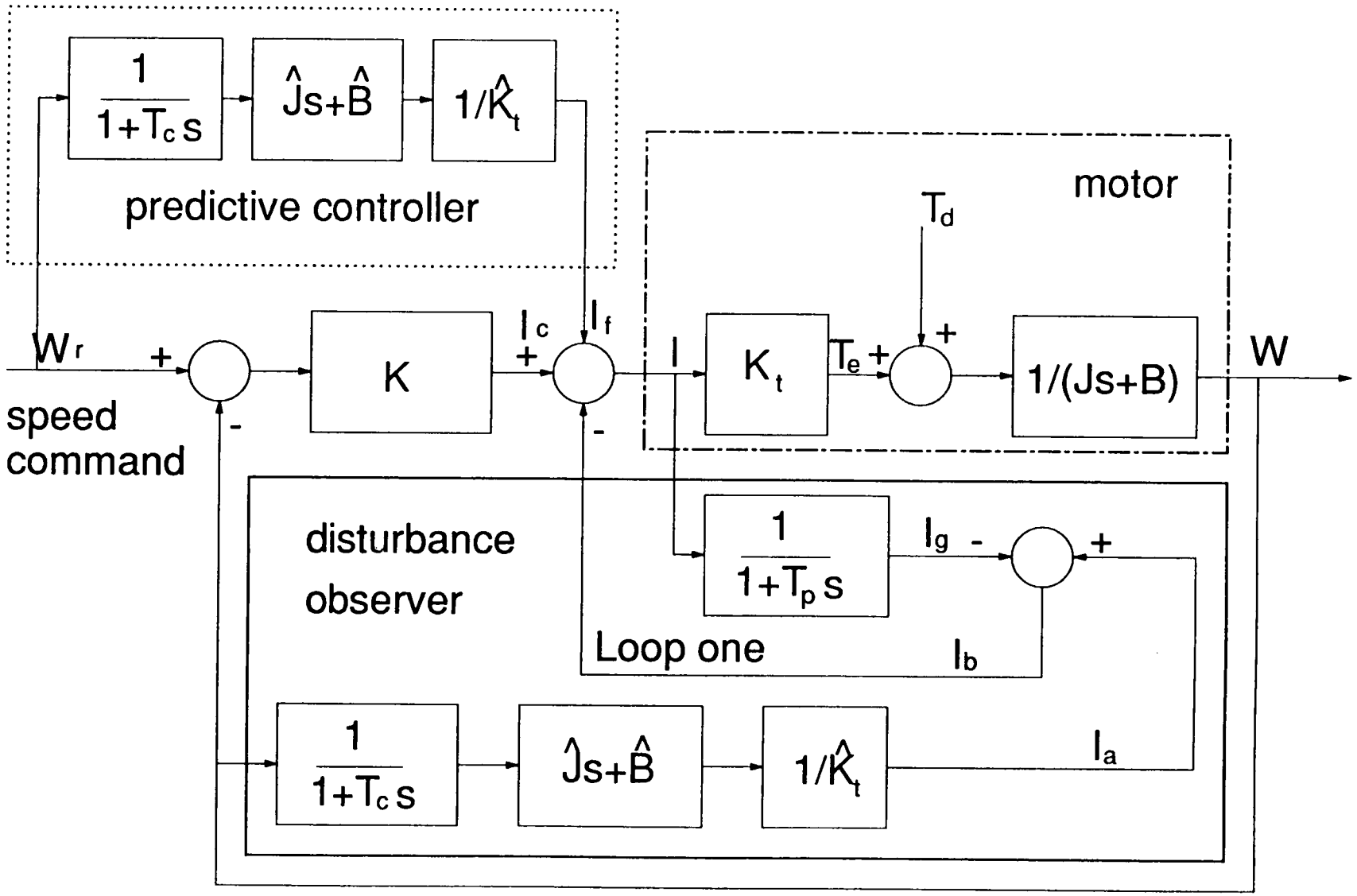


Fig. 3.1 Overall control block diagram.

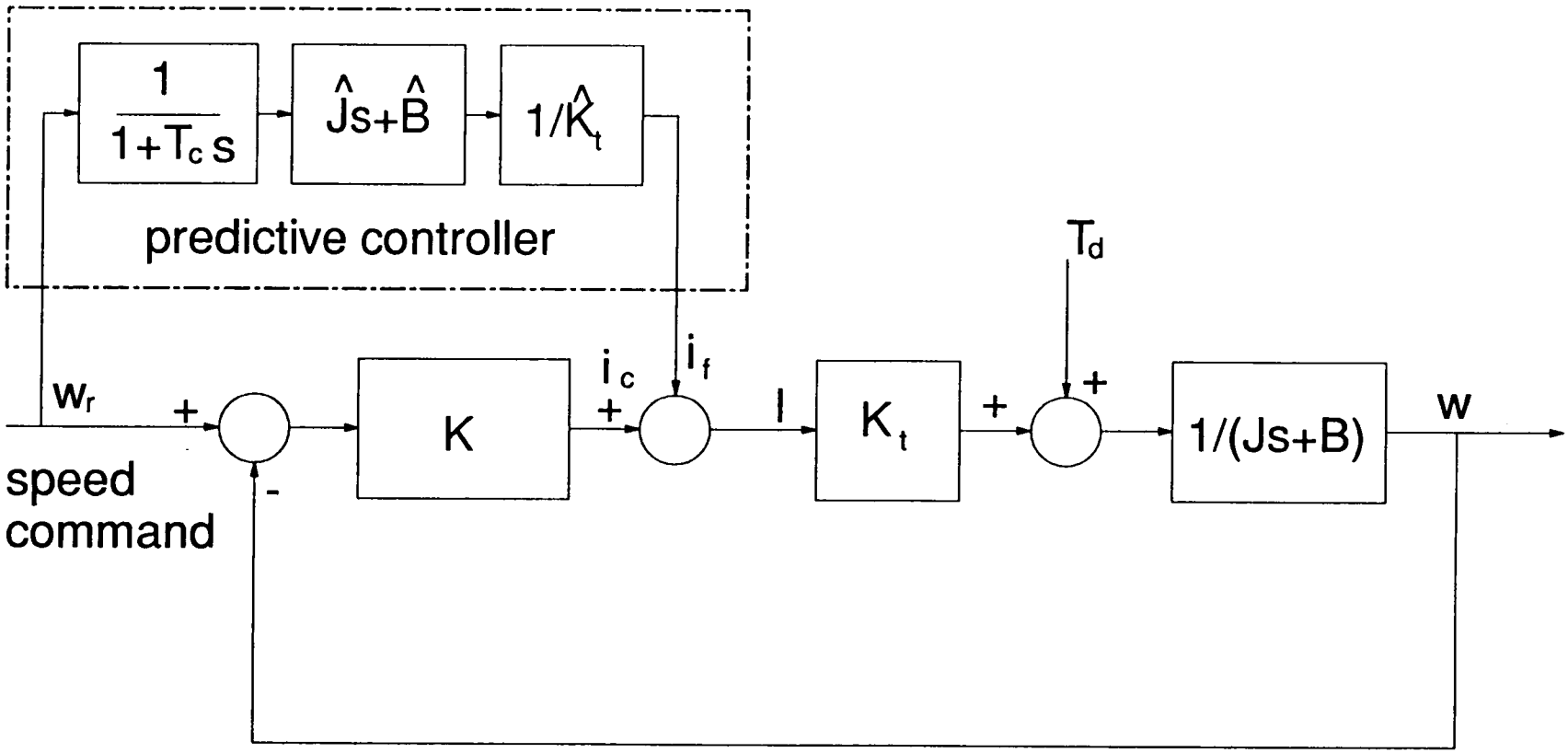


Fig. 3.2 Proportional plus predictive control.

needs to be derived. In this section, four transfer functions for the motor control system will be derived and the step response analysis for each transfer function will follow. The transfer functions $H_1(s)$ and $H_2(s)$ are the control ratios between the input and output of the control system with and without a disturbance observer. This is shown in Fig. 3.1 and Fig. 3.2, where W_r is the system reference input and W is the system output. The transfer functions $H_3(s)$ and $H_4(s)$ are the disturbance transmissions between the disturbance input and the system output as shown in Fig. 3.1 and Fig. 3.2, respectively, where W is the system output and T_d is the disturbance torque.

3.2.1 Control ratio for motor control system

In this section, two systems control ratios will be derived first, then the steady state response of the output from both systems will be compared. From the general gain formula, the control ratio $H_1(s)$ for the system with disturbance observer is

$$H(s) = \frac{1}{\Delta}(T_1\Delta_1 + T_2\Delta_2) \tag{3.1}$$

$$H_1(s) = \frac{[KK_t(1 + T_cs) + (Js + B)]}{(1 + T_cs)(Js + B) - \frac{(1+T_cs)(Js+B)}{1+T_ps} + (Js + B) + KK_t(1 + T_cs)} \tag{3.2}$$

and the control ratio $H_2(s)$ for the system without disturbance observer is

$$H_2(s) = \frac{KK_tT_cs + Js + KK_t + B}{(Js + B + KK_t)(1 + T_cs)}. \tag{3.3}$$

However, if the design values for $K = 10$, $T_c = .001$, $T_p = .001$ and the nominal values for $J = .02$, $K_t = 2$, $B = 1$, then the transfer function $H_1(s)$ equals $H_2(s)$ and can be written as

$$H_1(s) = H_2(s) = \frac{2000s + 1050000}{s^2 + 2050s + 1050000}. \tag{3.4}$$

Two poles of the system are located at -1000 and -1050. There are no roots in the right half side of the s-plane.

Next, the system control ratio will be analyzed with a step response, and the results will be shown.

If the speed command for both systems is a step input, ($W_r = 1/s$) , and the nominal and the design parameters remain the same, then the output can be written as

$$W(s) = H(s)U(s) = \frac{2000s + 1050000}{s(s^2 + 2050s + 1050000)}. \tag{3.5}$$

By the initial value theorem, the initial output can be found for both control systems with a step input,

$$\lim_{s\to\infty} sW(s) = \lim_{t\to 0} w(t) = 0. \tag{3.6}$$

By the final value theorem, the final output value for both control systems with a step input is

$$\lim_{s\to 0} sW(s) = \lim_{t\to\infty} w(t) = 1. \tag{3.7}$$

By taking the inverse of the Laplace transform of equation 3.5, the output is

$$w(t) = U(t) + 19e^{-1000t} - 20e^{-1050t}. \tag{3.8}$$

The step response of the system output is plotted in Fig. 3.3, where the 2 percent settling time is about .006 s and peak time is about .002 s.

In the next section, the disturbance transmission for both systems will be examined, and the step response for both disturbance transfer functions will be investigated.

3.2.2 Disturbance transfer function for motor control system

The main advantage of the disturbance canceller will now be considered. That is its ability to suppress the disturbance. In this section, the disturbance transmission for the system with and without a disturbance observer will be investigated and compared.

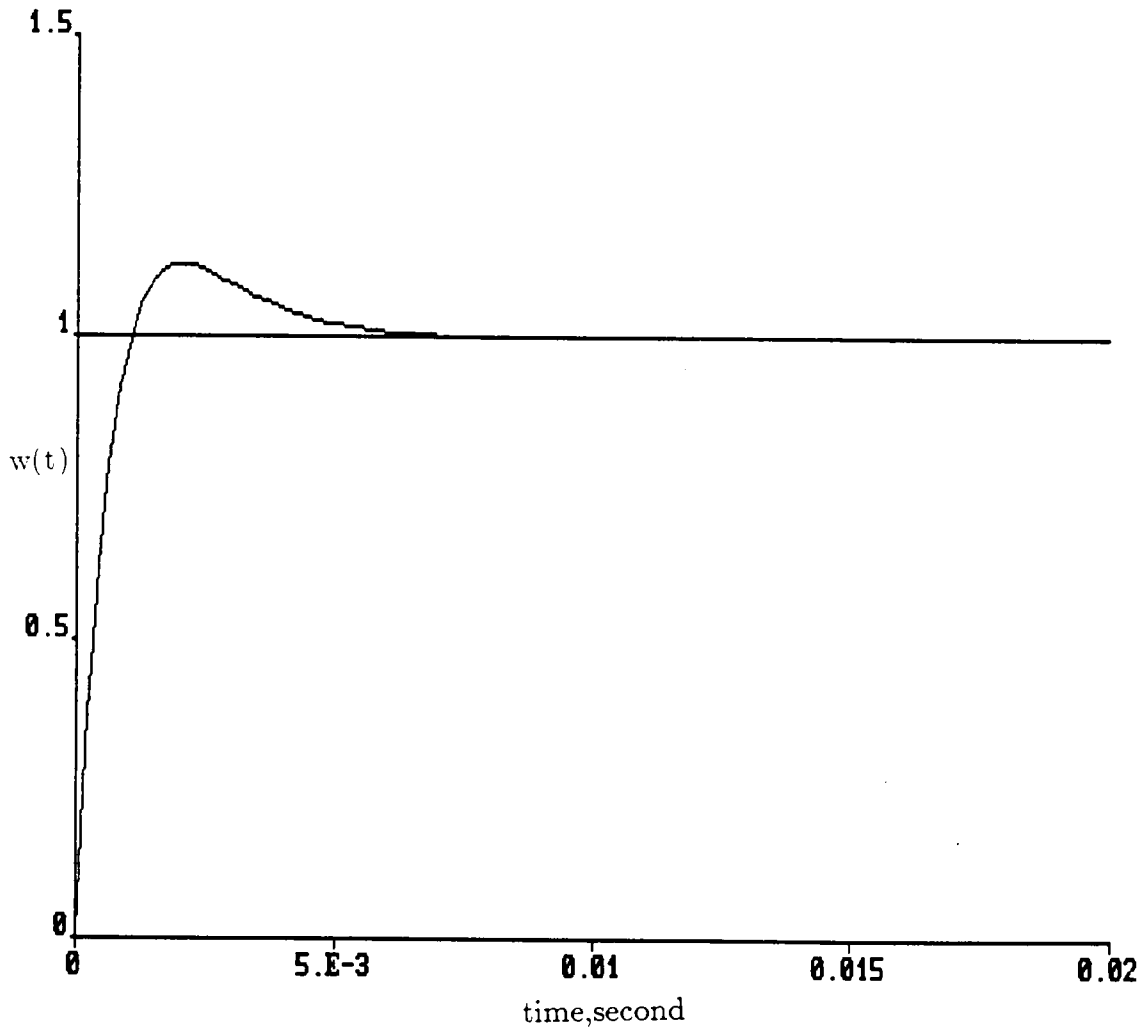


Fig. 3.3 Step response of system output.

First, the disturbance transfer function for both systems will be derived. Following that, by applying the initial value theorem and final value theorem, both control system responses to the applied disturbance will be examined. By the general gain formula, the transfer function for the system with a disturbance observer can be written as

$$H_3(s) = \frac{T_p s}{(1 + T_p s)(Js + B + KK_t)}. \tag{3.9}$$

If the nominal system parameters remain the same, the transfer function can be written as

$$H_3(s) = \frac{50s}{(s + 1000)(s + 1050)}. \tag{3.10}$$

If the disturbance equals to a step input and the nominal values remain the same, then the output is

$$W(s) = H_3(s)U(s) = \frac{T_p s}{s(1 + T_p s)(Js + B + KK_t)}. \tag{3.11}$$

Again, when the input T_d is a step input, the initial value can be found by the initial value theorem,

$$w(0) = sW|_{s\to\infty} = 0. \tag{3.12}$$

The final value can be found by the final value theorem,

$$w(\infty) = sW|_{s\to 0} = 0. \tag{3.13}$$

The final value equal to zero means that the system has the capability to eliminate the disturbance effect. That is one of the goals for this design. From the final value, the designer will know if the disturbance will be eventually canceled. By the inverse of the Laplace transform, one can find the step response of the output w(t) when the nominal

system parameters remain the same,

$$w(t) = -e^{-1000t} + e^{-1050t}. \tag{3.14}$$

The disturbance transfer function for the system without the disturbance observer can be written as

$$H_4(s) = \frac{1}{Js + B + KK_t}. \tag{3.15}$$

Next, the control system without the observer will tested by applying a step disturbance to the system. If the disturbance is equal to a step input, then the output is

$$W(s) = H_4(s)U(s) = \frac{1}{s(Js + B + KK_t)}. \tag{3.16}$$

If the nominal value and the design value remains the same, the transfer function can be written as

$$W(s) = \frac{50}{s(s + 1050)}. \tag{3.17}$$

By the initial value theorem, the initial value of the step disturbance transmission is

$$w(0) = sW|_{s\to\infty} = 0. \tag{3.18}$$

This is the same as the system with the disturbance observer. Also, by the final value theorem, the final value of the step disturbance transmission for the system without the disturbance observer is

$$w(\infty) = sW|_{s\to 0} = \frac{50}{1050}. \tag{3.19}$$

The final value is not equal to zero. This means the effect of the disturbance can not be eliminated. In contrast to a control system with a disturbance observer with final value equal to zero, a system without an observer will not be able to take a disturbance without producing some steady state error.

Consequently, by only considering the final value of a system with a step disturbance, the superiority of the system with the disturbance observer is proven. Fig. 3.4 shows the disturbance transmission step response of both control systems. Comparison of each system's peak value in response to a step disturbance shows that the system with the disturbance observer has an edge of superiority.

3.3 Choice of low-pass filter time constant

In the following only the system with the disturbance observer will be considered. The effect of the time constant on the low-pass filter will be discussed. As mentioned before, once the low pass filter is added in the system, an effect is seen in system response. In Fig. 3.1, there are three low-pass filters. The filter between I_g and I has time constant equal to T_p. The filter between I_f and W_r has time constant equal to T_c. The filter between I_a and W also has time constant equal to T_c. The filter between I_g and I is used to break the loop and provides a time delay. The other two filters are used to realize the equation between I_a and W and between I_f and W_r. The choice of these two time constants has an effect on the system response and also the ability to suppress the disturbance. In the following, a discussion on how to choose the time constant will be presented with two different points of view.

3.3.1 Choice of low-pass filter time constant by control ratio step response

If all of the parameters are of nominal value, the effect of changing the low-pass filter time constant T_p is shown in Fig. 3.5. In Fig. 3.5, T_c is held constant .001s and T_p is varied to give a family of curves. From the plot, one can see that the lower the

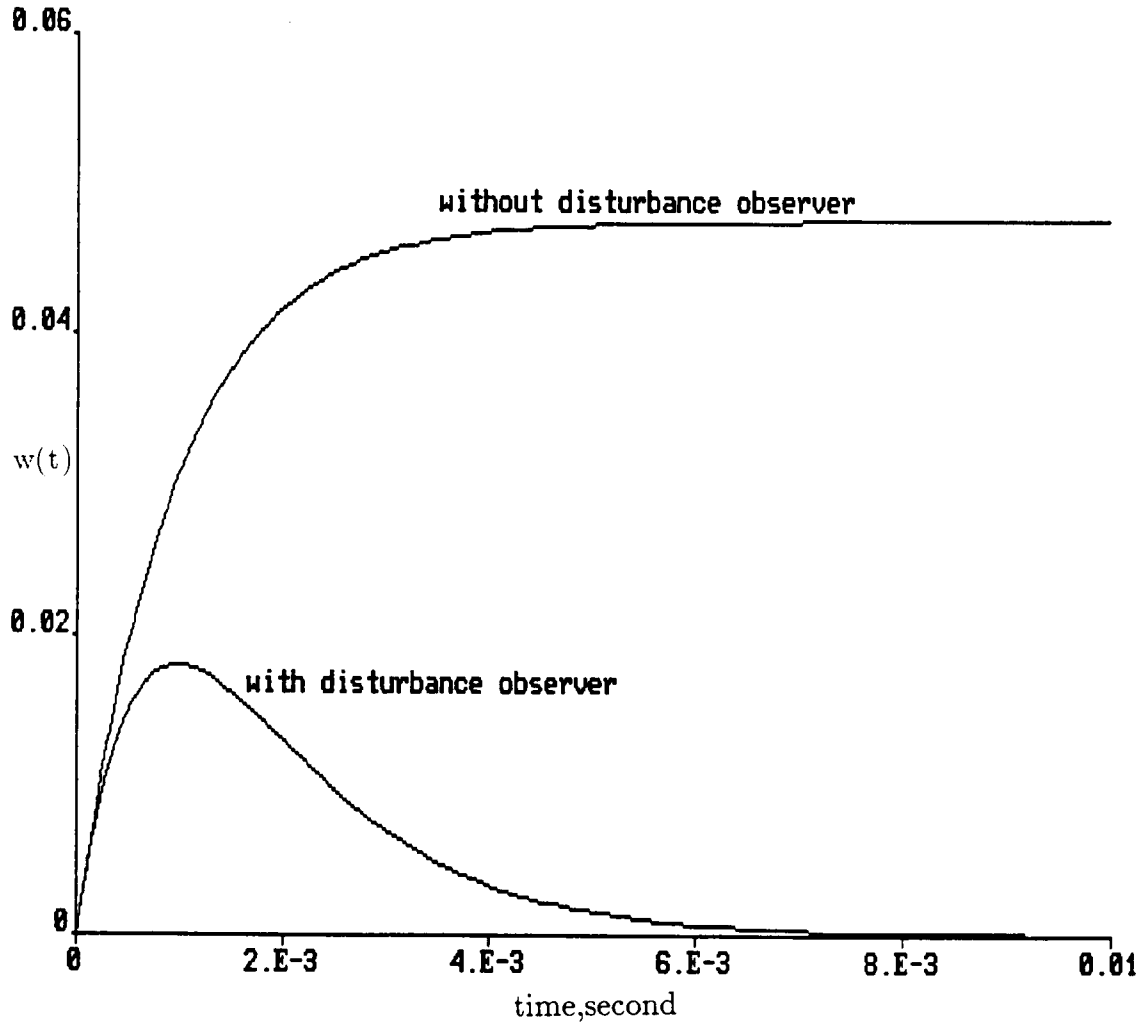


Fig. 3.4 System disturbance transmission comparison.

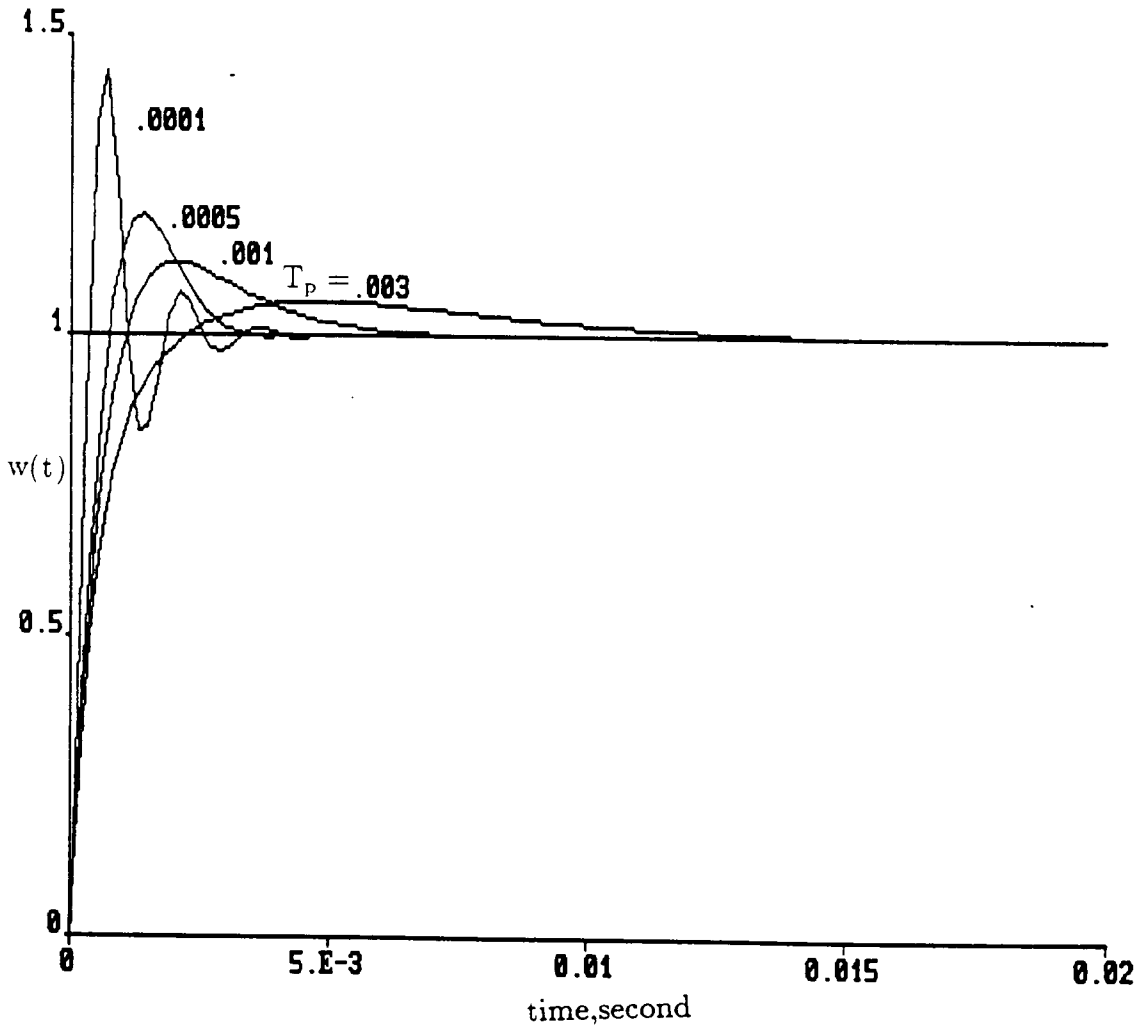


Figure 3.5 Steady state response of system with disturbance observer varied T_p and a constant $T_c = .001s$

time constant, the shorter the time to maximum overshoot t_p. Also the lower the time constant, the higher the overshoot.

In contrast, the effect of changing the time constant T_c on the observer's low pass filter and the predictive controller's low-pass filter is shown in Fig. 3.6. In Fig. 3.6, T_p is held constant at .001s and T_c is allowed to vary. From the plot shown, the higher the time constant, the longer the time to maximum overshoot t_p, and the higher the time constant, the higher the overshoot. Consequently, the choice of the time constant is a compromise.

The effect of changing both of T_p and T_c simultaneously is shown on Fig. 3.7. It needs to be pointed out that the above discussion did not take the disturbance input into consideration. If the system computer capability and the need of the process is understood, it is fairly easy to choose the time constant for the low-pass filter with the help of computer simulation. A program using the SIMNON language for this part of the simulation is listed in Appendix A.

3.3.2 Choice of low-pass filter time constant by disturbance transmission step response

With the idea of eliminating the disturbance effect as quickly as possible, one takes the low-pass filter time constant effect into consideration again. The following discussion assumes no reference input and a step disturbance.

In Fig. 3.8, the time constant T_c is held at a constant of .001s and T_p is allowed to vary, showing the effect of adjusting the time constant T_p. One can see that choosing a smaller T_p improves the canceling time. In Fig. 3.9, T_p is held at .001s and T_c is allowed to vary, showing that the smaller T_c is, the smaller the overshoot. Fig. 3.10

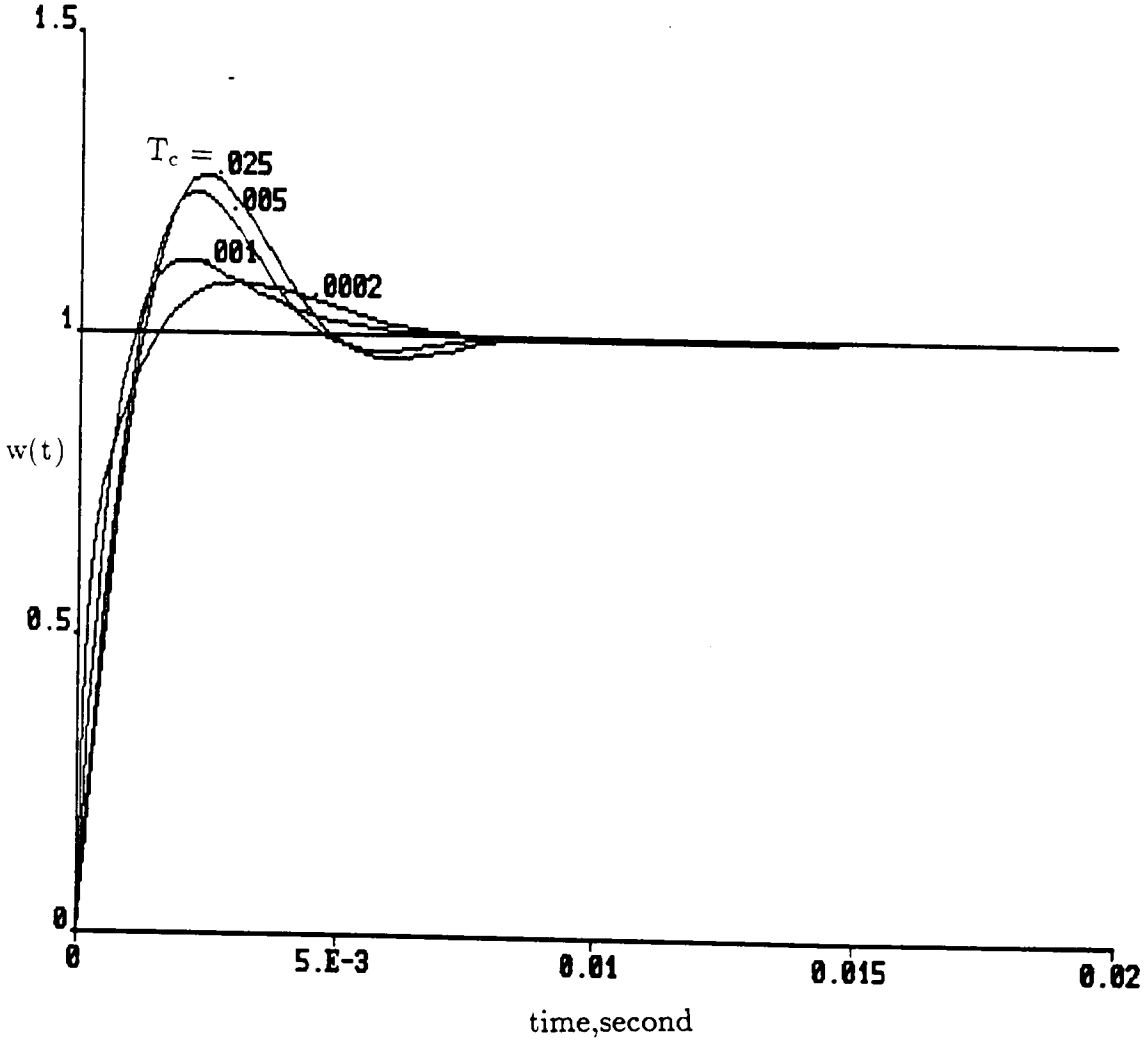


Figure 3.6 Steady state response of system with disturbance observer varied T_c and a constant T_p = .001s

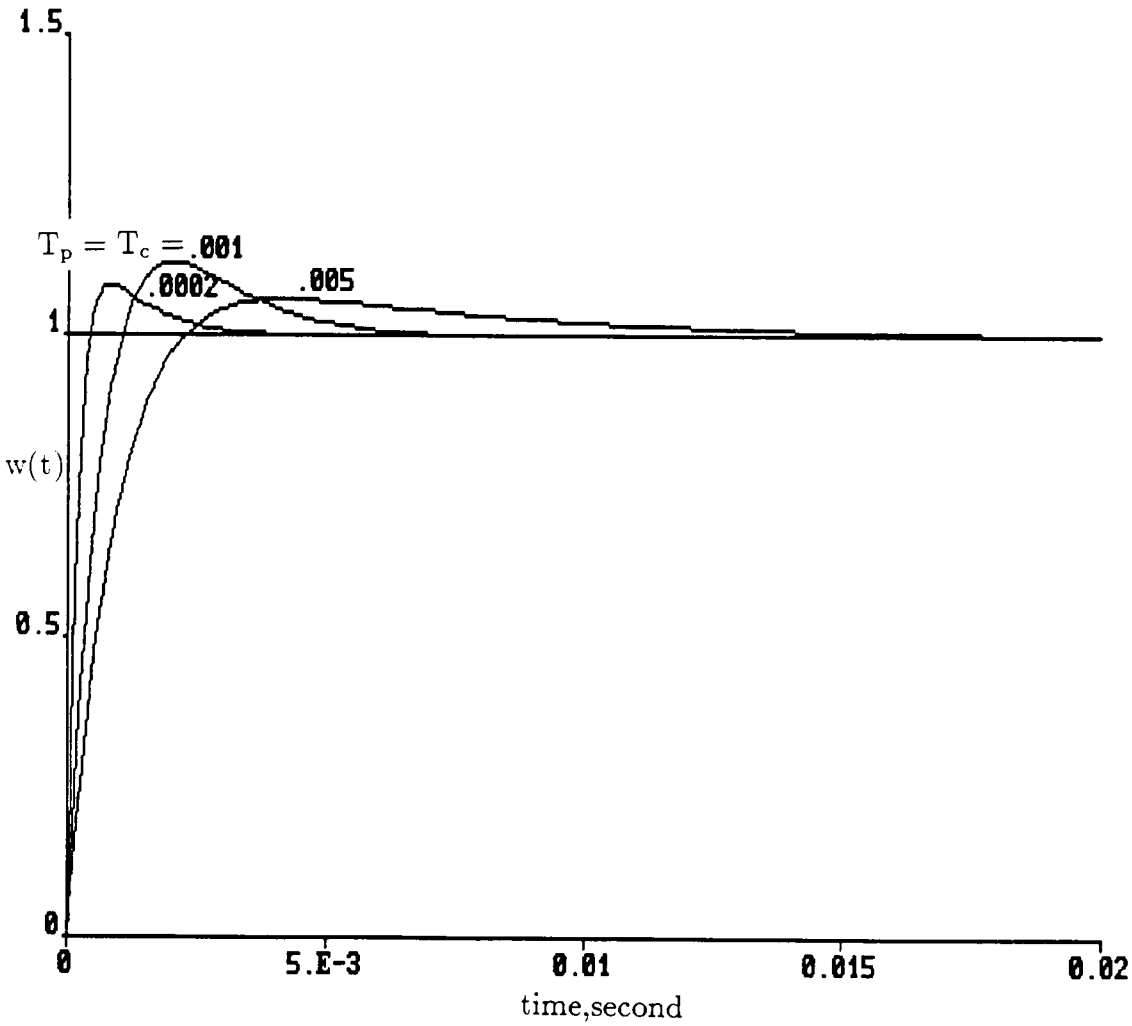


Fig. 3.7 Steady state response of system with disturbance observer with varied T_p and T_c.

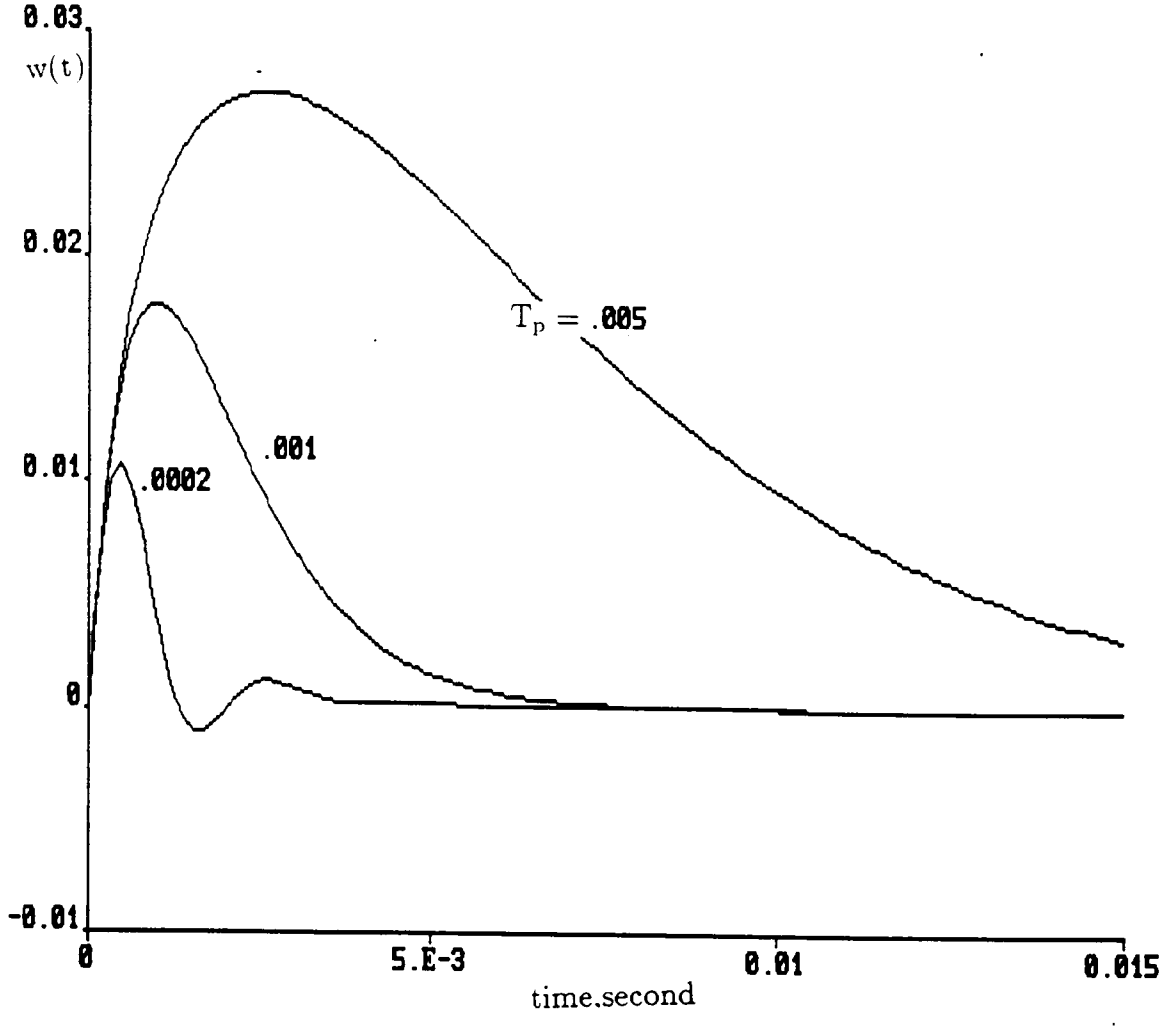


Figure 3.8 Step disturbance transmission for the system with disturbance observer with varied T_p and a constant T_c = .001s

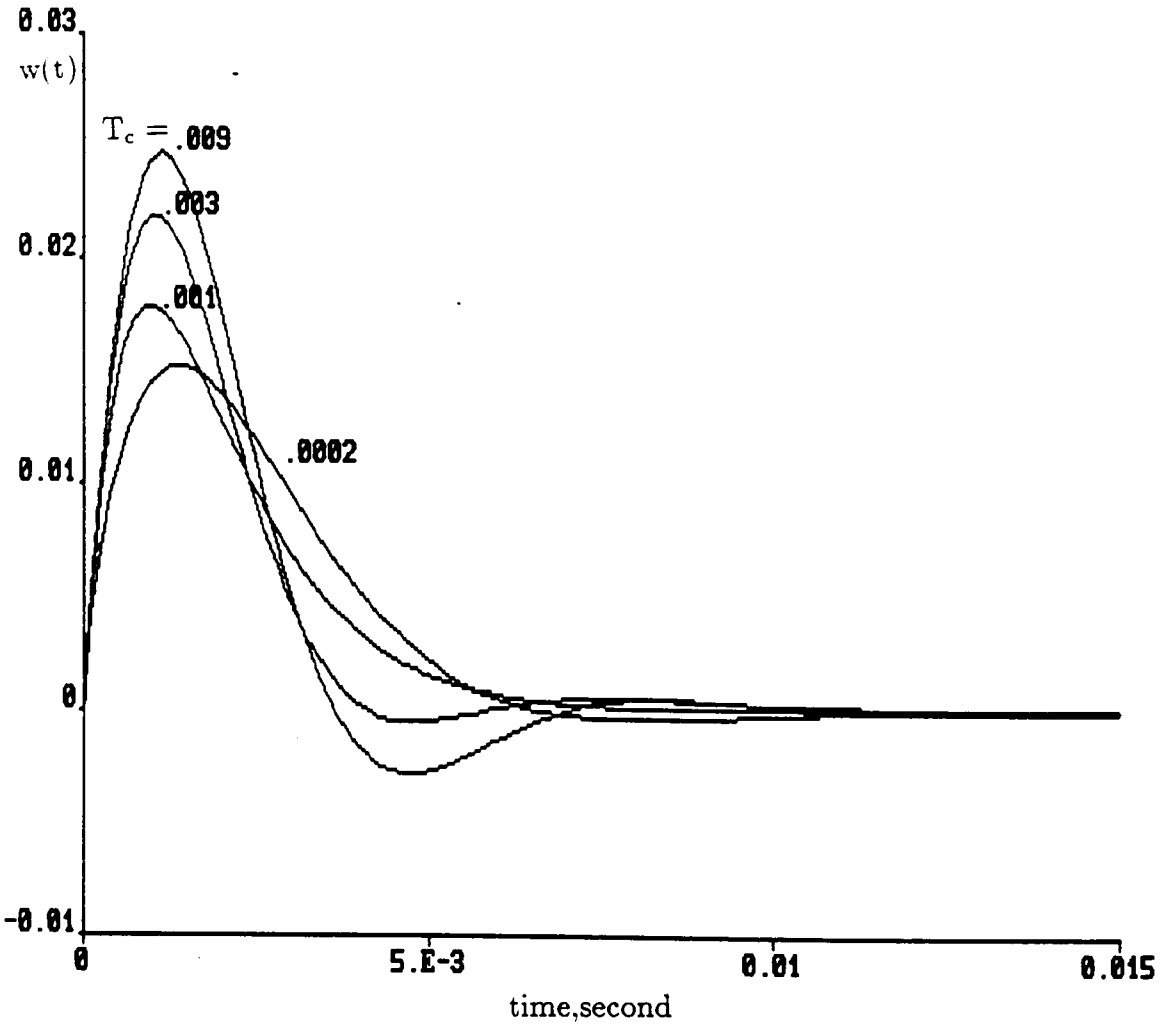


Figure 3.9 Step disturbance transmission for the system with disturbance observer with varied T_c and a constant T_p = .001s

shows the effect of changing T_p and T_c simultaneously. This shows that the smaller the time constant, the quicker the system can eliminate the disturbance.

With an understanding of the process requirements such as settling time or overshoot, it is fairly easy to choose the time constant. A program using the SIMNON language for this part of the simulation is listed in Appendix A.

3.4 System sensitivity analysis

3.4.1 Introduction

The performance characteristics of most components are affected by their environment. Thus any change in component characteristics causes a change in the transfer function and therefore the controlled quantity. The effect of a parameter change on system performance can be expressed in terms of a *sensitivity function* .

The sensitivity function is introduced as a means of deducing the variation which occurs in one dependent variable as a consequence of the changes occurring in some other independent variables. For instance, if for any reason, the transfer function of a plant P(s) should change, then the transfer function of the closed loop system H(s) will also change. The amount of change depends on the total loop function L(s) since

$$H(s) = \frac{G(s)P(s)}{1 + G(s)P(s)} = \frac{L(s)}{1 + L(s)}, \tag{3.20}$$

where G(s) is the compensator function.

Let us assume that the plant changes by an amount δp, then expanding the resulting system as a Taylor's series leads us to

$$H(s, P + \delta P) = H(s, P) + \frac{\partial H(s)}{\partial P(s)}\delta P + ... \tag{3.21}$$

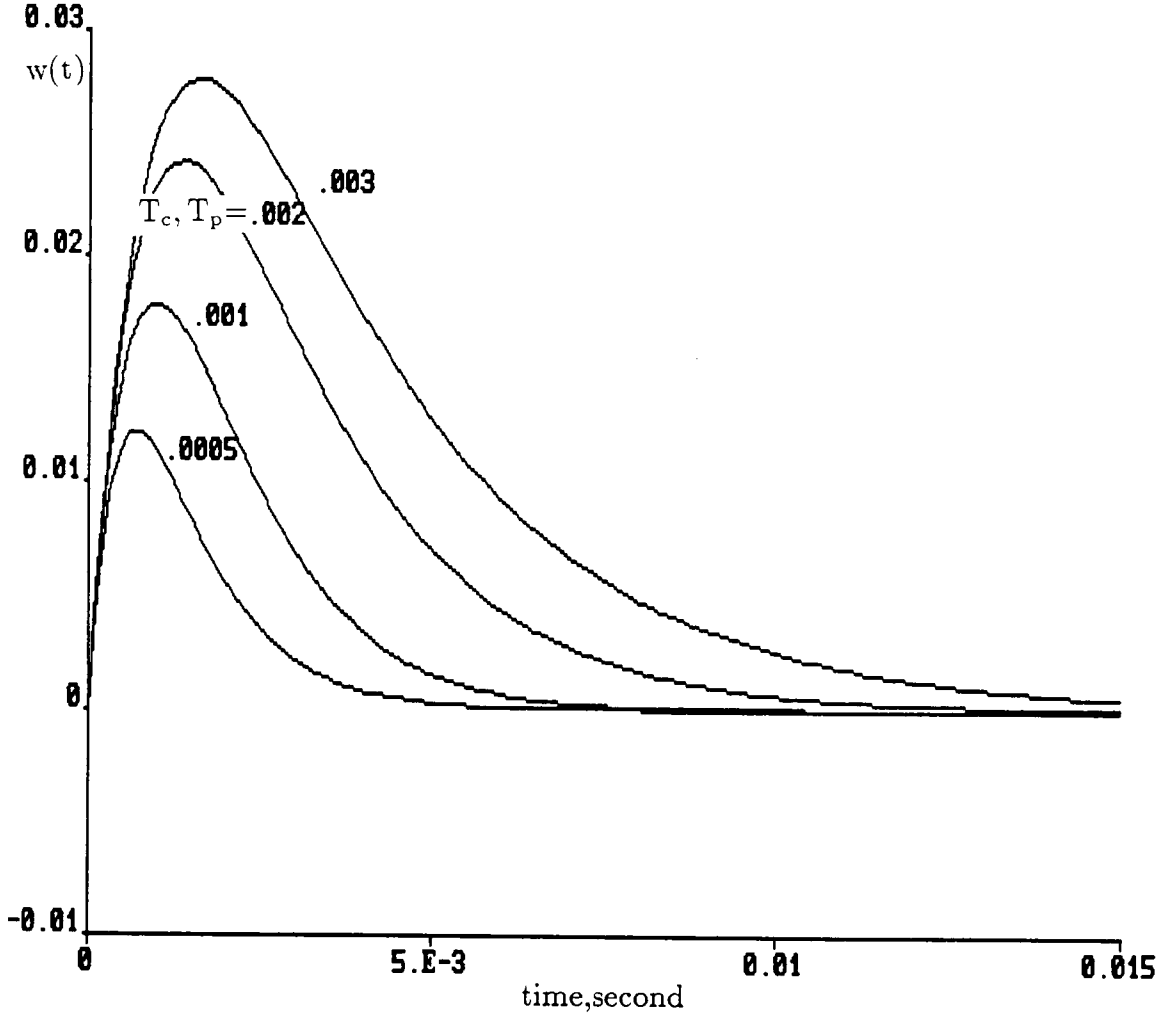


Fig. 3.10 Step disturbance transmission for the system with disturbance observer with varied $T_c = T_p$.

The higher-order terms of the equation may be discarded because of little contribution. Then one has

$$\delta H = \frac{\partial H(s)}{\partial P(s)} \delta P. \tag{3.22}$$

By re-arranging:

$$\frac{\delta H}{H} = \frac{\frac{\partial H}{H}}{\frac{\partial P}{P}} \frac{\delta P}{P}, \tag{3.23}$$

where

$$\frac{\frac{\partial H}{H}}{\frac{\partial P}{P}} = S_P^H, \tag{3.24}$$

where S is the classical sensitivity function. Of course, P is a function of perhaps a number of variable quantities, such as gain, poles, and zeros. The analysis here has considered variation in the transfer functions P(s). It reveals that S_P^H never exceeds a magnitude of one. Also, the smaller the value of S_P^H, the less sensitive the system is to a variation in the transfer function. A positive value of the sensitivity function means that the output increases from its nominal response. Similarly, a negative value of the sensitivity function means that the output decreases from its nominal response. However, if the system is represented by the frequency transfer function H(jw), then the magnitude S_P^H does not have limits of 0 to 1, but can vary from 0 to any magnitude. Two control system sensitivities to parameter variation will be investigated in detail in the following section.

3.5 Sensitivity analysis

Sensitivity analysis is concerned with the effectiveness of parameter variation to an existing control system. In this section, two approaches will be considered. Frequency

response sensitivity [6] will be investigated first. Following that, the transient response sensitivity will be discussed. Again, the two control systems in Fig. 3.1 and Fig. 3.2 will be analyzed and compared.

3.5.1 Frequency Response Sensitivity

The variation of the system function as related to the variable parameters of the plant has been defined in the previous section. The equivalent frequency response variation is obtained by simply replacing the Laplacian operator s by jw. In this section, the moment of inertia will be used as the parameter which is allowed to vary.

(a) Frequency response sensitivity on control ratio

If all of the nominal parameter values remain the same, except that the moment of inertia (J) is allowed to vary, then the control ratio for the system with the disturbance observer can be written as

$$H_1(s) = \frac{40s + 21000}{Js^2 + 41s + 21000}. \tag{3.25}$$

The derivative of the transfer function with respect to J is

$$\frac{\partial H}{\partial J} = \frac{-40s^3 - 21000s^2}{(Js^2 + 41s + 21000)^2}. \tag{3.26}$$

If the system is now represented by the frequency transfer function H(jw), the formula developed above is the same in form, but the argument is jw instead of s. As the moment of inertia varies within H(jw), the magnitude of the sensitivity function can be plotted as a function of frequency. One can analyze the system at any magnitude of the transfer function as long as the transfer function reaches at least that magnitude. The analysis for the control ratio on both systems is based on $|H_0(jw_p)| = H_0 = .707$ for the nominal system values of the control systems. If one substitutes $|H_0| = .707$ and s =

jw into equation 3.25, one finds that the frequency w is equal to 2475. By substituting s = 2475j into equation 3.26, one finds $\partial H/\partial J$ equal to 30.119. Substituting the above values into equation 3.24 and letting the variable parameter be the moment of inertia (J), then the sensitivity at a frequency of 2475 is

$$S_J^H|_{H=H_0 \text{ and } J=J_0} = \frac{J}{H}\frac{\partial H}{\partial J} = \frac{.02}{.707}(30.119) = .843. \tag{3.27}$$

If all of the nominal values remain the same except that the moment of inertia (J) is allowed to vary, then the transfer function for the system without a disturbance observer can be written as

$$H_2(s) = \frac{40s + 21000}{Js^2 + 21s + 1000Js + 21000}. \tag{3.28}$$

The derivative of the transfer function with respect to J is

$$\frac{\partial H}{\partial J} = \frac{-40s^3 - 61000s^2 + 21000s}{(Js^2 + 21s + 1000Js + 21000)^2}. \tag{3.29}$$

The analysis for the control ratio on this system is also based on $|H_0(jw_p)| = H_0 = .707$ for the nominal system values of the control system. If one substitutes $|H_0| = .707$ and s = jw into equation 3.28, one finds that the frequency w is equal to 2475. By substituting s = 2475j into equation 3.29, $\partial H/\partial J$ is equal to 34.621. Substituting the above values into equation 3.24 and varying the moment of inertial (J), shows that the sensitivity at a frequency of 2475 is

$$S_J^H|_{H=H_0 \text{ and } J=J_0} = \frac{J}{H}\frac{\partial H}{\partial J} = \frac{.02}{.707}(34.621) = .9728. \tag{3.30}$$

From the above comparison, at $|H_0| = .707$, the sensitivity of the system without the disturbance observer is higher. Fig. 3.11 shows that the system without a disturbance observer has higher sensitivity than the system with the disturbance observer at the

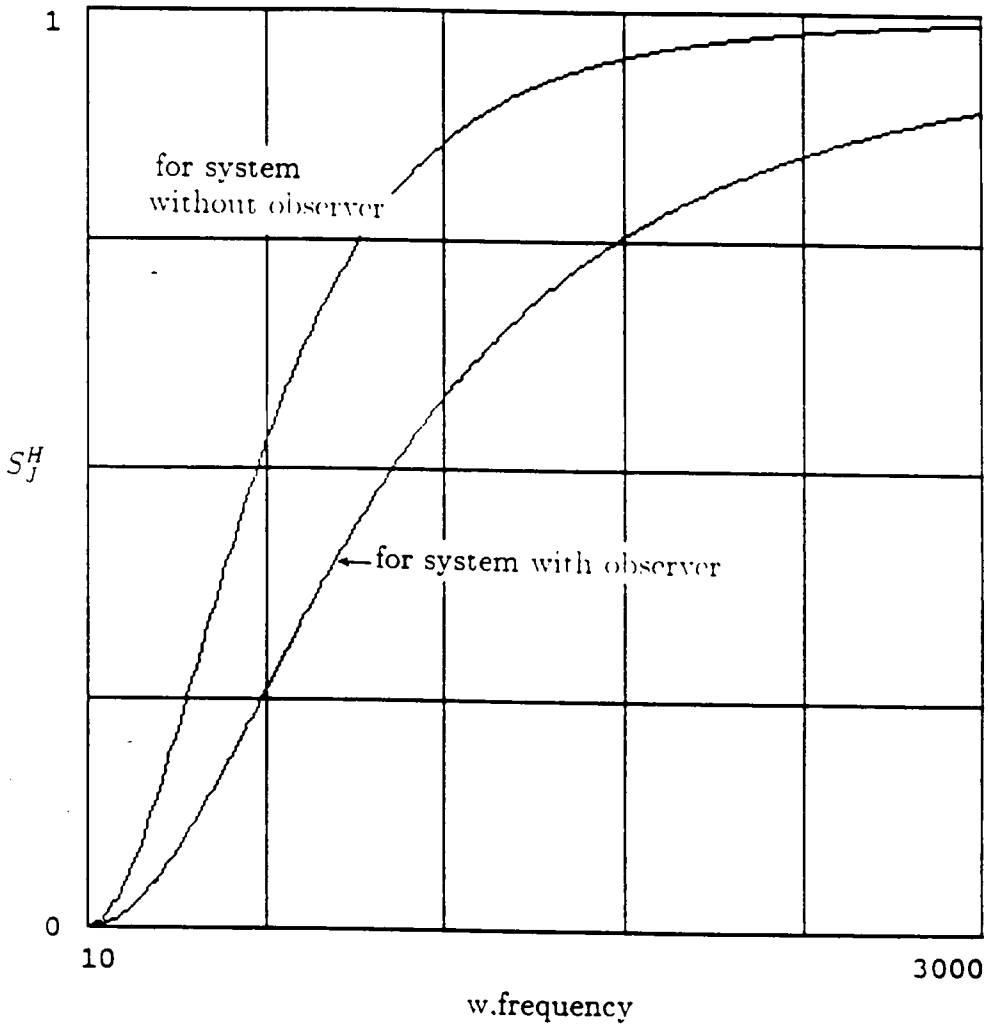


Figure 3.11 Sensitivity due to moment of inertia variation for control ratio

same frequency. Fig. 3.11 also shows that the higher the frequency, the higher the sensitivity.

(b) Frequency response sensitivity on the disturbance transmission function

If all of the nominal values are to remain the same except the moment of inertia (J), which is allowed to vary, then the transfer function for the system with disturbance observer $H_3(s)$ can be written as

$$H_3(s) = \frac{s}{Js^2 + 41s + 21000}. \tag{3.31}$$

The derivative of the transfer function with respect to J is

$$\frac{\partial H}{\partial J} = \frac{-s^3}{(Js^2 + 41s + 21000)^2}. \tag{3.32}$$

If the analysis is based on determining sensitivity at $|H_0(jw_p)| = H_0 = .02$ for the nominal values of the control system, then the frequency is found to be 2040 and $\partial H/\partial J$ equals .781. The sensitivity at a frequency of 2040 is

$$S_J^H|_{H=H_0 \text{ and } J=J_0} = \frac{J}{H}\frac{\partial H}{\partial J} = \frac{.02}{.02}(.781) = .781. \tag{3.33}$$

Fig. 3.12 shows the sensitivity (due to the moment of inertia variation) for the disturbance transmission function of the system with an observer.

If all of the nominal values and design values remain the same except for the moment of inertia (J), which is allow to vary, then the disturbance transmission function $H_4(s)$ for the system without a disturbance observer can be written as

$$H_4(s) = \frac{1}{Js + 21}. \tag{3.34}$$

Taking the derivative with respect to J yields

$$\frac{\partial H}{\partial J} = \frac{-s}{(Js + 21)^2}. \tag{3.35}$$

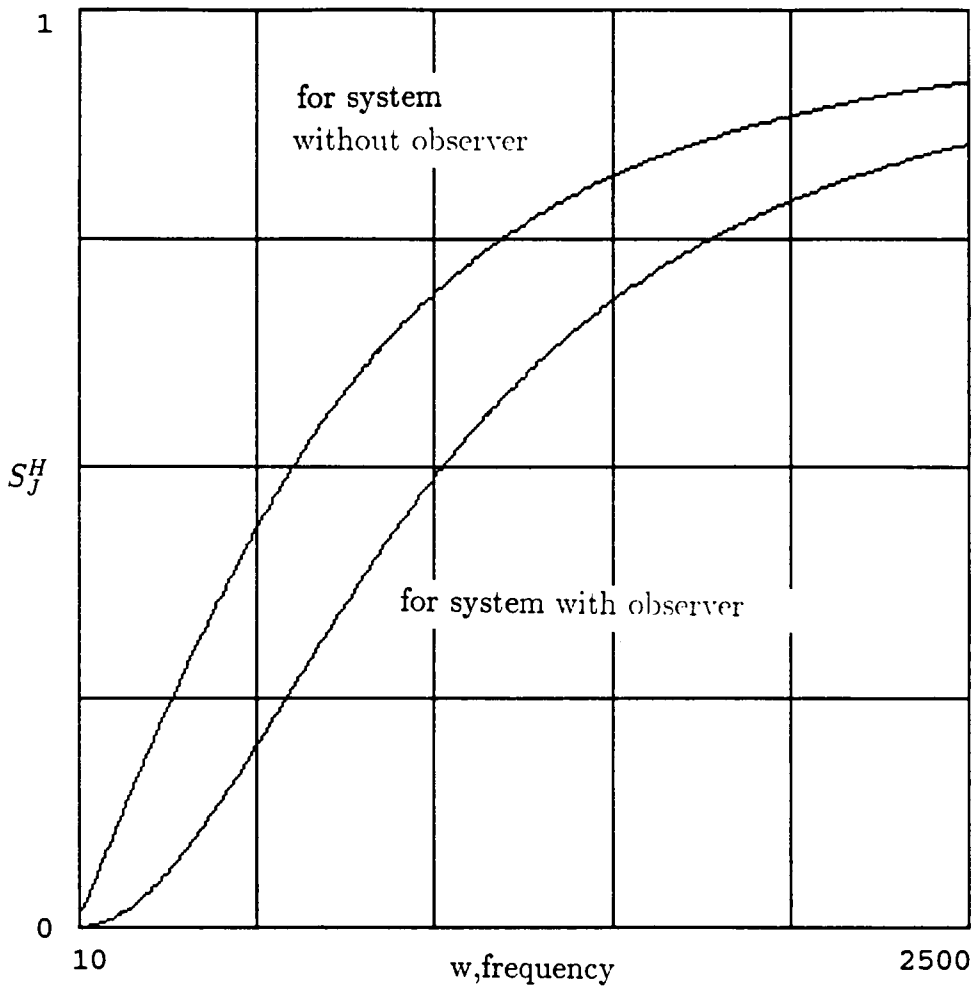


Fig. 3.12 Sensitivity due to moment of inertia variation for disturbance transmission.

If the analysis is based on determining the sensitivity at $|H_0(jw_p)| = H_0 = .02$ for the nominal values of the control systems, then the frequency is found to equal 2240 and $\partial H/\partial J = .915$. The sensitivity at a frequency of 2240 is

$$S_J^H|_{H=H_0 \text{and} J=J_0} = \frac{.02}{.02}(.915) = .915. \tag{3.36}$$

From the above comparison at $|H_0| = .02$, the sensitivity of the system without the disturbance observer is higher. Fig. 3.12 shows the sensitivity due to moment of inertia variation for the disturbance transmission function. From Fig. 3.12, at the same frequency, the system without a disturbance observer has higher sensitivity than the system with a disturbance canceller. It also shows that the sensitivity is increased with increasing frequency for both systems.

3.7.2 Transient Response Sensitivity

As in the frequency response sensitivity, transient response sensitivity can be used as another tool to analyze the variation of the system response. If this tool is to be of any use , one needs to be aware of how the nominal response will be modified as a consequence of inevitable parameter variation.

Fig. 3.13 shows the step response of the control system with the moment of inertia varied 100 percent for the system with and without a disturbance observer. Also, the nominal system step response is plotted in the same figure. Fig. 3.14 gives a comparison of step response between the system with a disturbance observer and the system without an observer. Because the step response of the system with a disturbance observer follows the nominal plant step response closer than the system without an observer, one can say that the system with a disturbance observer is less sensitive to moment of inertia

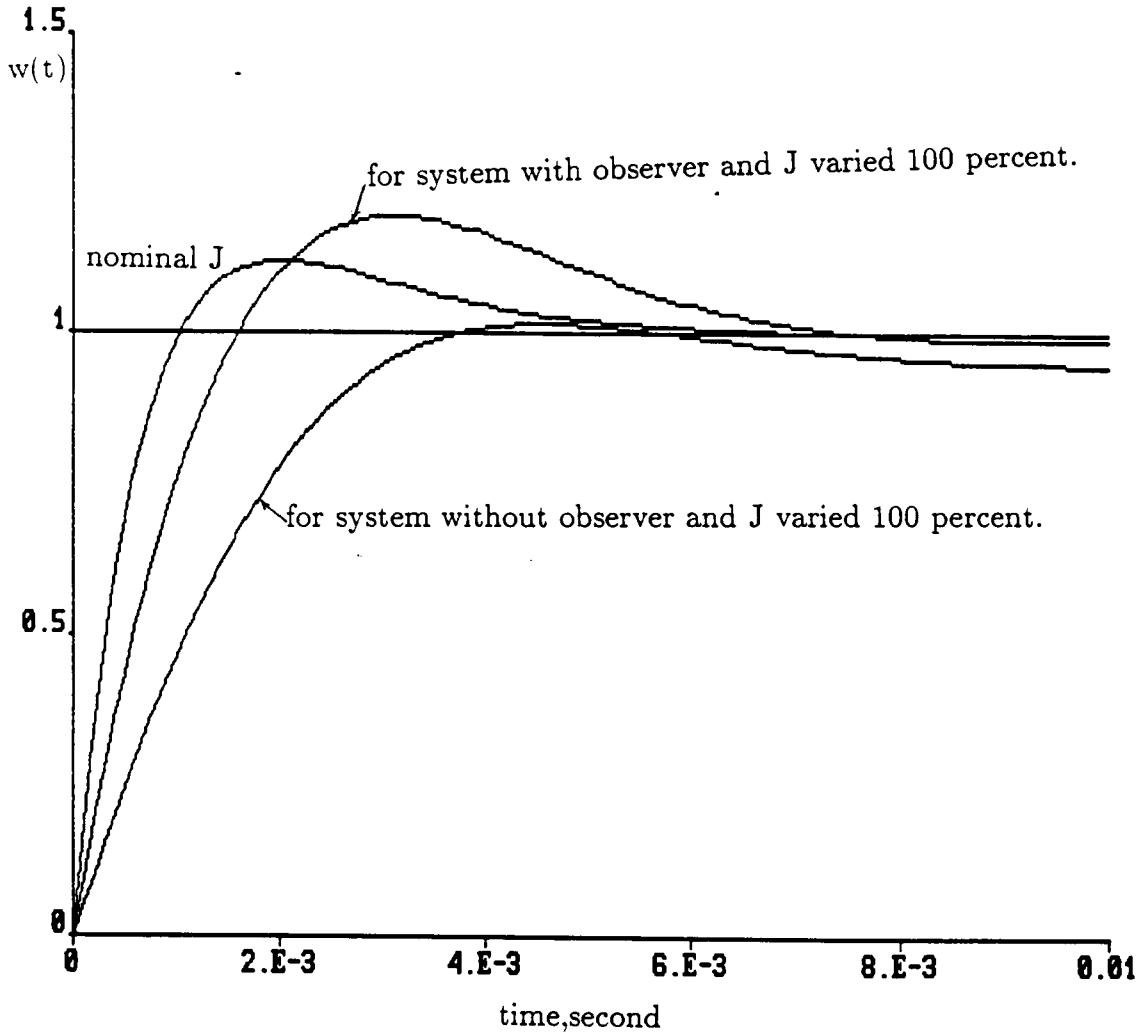


Fig. 3.13 Step response of system output with J varied 100 percent

step response of system output			
	100% variation of J		nominal J
time (second)	with observer w(t)	without observer w(t)	nominal w(t)
.001	0.860	0.610	1.00
.002	1.100	0.885	1.13
.003	1.120	0.938	1.10
.004	1.100	0.950	1.05
.005	1.075	0.970	1.02
.006	1.045	0.995	1.01

Figure 3.14 Step response data comparison

variation.

Fig. 3.15 shows the step response of the control system with the moment of inertia varied 100 percent for the system with and without a disturbance observer but with the predictive controller removed. One sees that the existence of a predictive controller may make the system less robust. Fig. 3.16 gives a comparison of the step response of the system with a disturbance observer and the system without an observer. From the above comparison, one can say that the system with a disturbance observer is less sensitive to moment of inertia variation.

3.6 Conclusion

In this chapter, the stability of the motor control system has been investigated. With consideration of system response requirements and computer capability, ways of choosing the low-pass filter time constant have been discussed. The effect of the time constant was also investigated. More importantly, the benefit of having a disturbance canceller has been discussed in detail. Without the disturbance observer, the system can not eliminate the disturbance. Consequently, a steady-state error will always exist. With the understanding that the control system model involves many approximations, the designer needs to take these approximations into consideration. In this chapter, the sensitivity function has been introduced which relates either the system frequency or transient response to a change of the system parameters. Also, sensitivity analysis has been performed for two control systems in frequency and transient response.

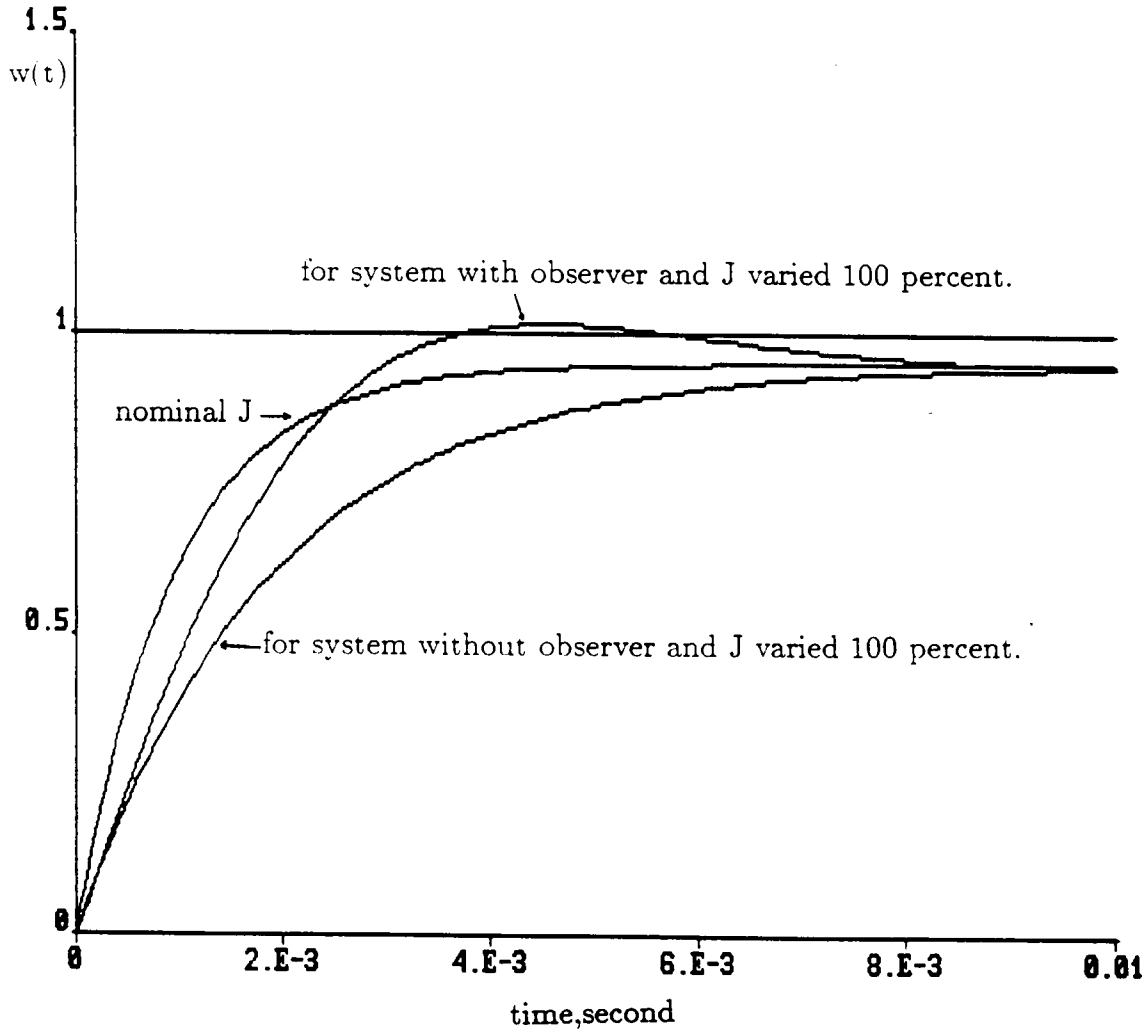


Figure 3.15 Step response of system without predictive controller system output with J varied 100 percent

Step response of system without predictive controller output comparison			
	100% variation of J		nominal J
time (second)	with observer w(t)	without observer w(t)	nominal w(t)
.001	0.469	0.375	0.619
.002	0.800	0.625	0.836
.003	0.938	0.750	0.912
.004	1.010	0.875	0.938
.005	1.010	0.913	0.947
.006	1.000	0.931	0.951

Figure 3.16 Step response data comparison for system without predictive controller

Chapter 4

System Digitization

4.1 Introduction

In this chapter, the designed system will be digitized. Two methods of transformation will be used. In the beginning of the chapter, the Tustin transformation method will be used to transform the predictive controller and one branch of the disturbance observer. Then a direct z-transform will be used to transform another branch of the disturbance observer. The problem of only using the Tustin transformation to digitize the system will be discussed. Following that, simulations of a continuous system and a digital system will be given. Finally, the simulations of the continuous and digital systems will be compared.

4.2 Tustin transformation

Several methods have been developed for obtaining the approximate z-domain transfer function from the s-domain transfer function representation. The transformations from s to z domain are accomplished by substituting functions of z for the s terms in the s-domain transfer function. One of the most popular of these approximation methods

is the Tustin transformation, which is

$$s = \frac{2}{T}\frac{1 - z^{-1}}{1 + z^{-1}} \tag{4.1}$$

and

$$z = \frac{1 + s\frac{T}{2}}{1 - s\frac{T}{2}}. \tag{4.2}$$

4.3 Digitization

Before proceeding with the digitization, several variables in Fig. 4.1 will be defined. Let I_f represent the output of the predictive controller, I_c is the output of the proportional controller, I is the controller output current, I_g is the current output from the low pass filter, I_a is the equivalent disturbance plus control current, and I_b is the equivalent disturbance current. In the following, the predictive controller will be digitized first and then the disturbance canceller.

4.3.1 Predictive controller digitization

From Fig. 4.1, the relationship between I_f and W_r can be written as

$$\frac{I_f}{W_r} = \frac{\hat{J}s + \hat{B}}{(1 + T_c s)\hat{K}_t} = \frac{\hat{J}s + \hat{B}}{T_c\hat{K}_t s + \hat{K}_t}, \tag{4.3}$$

where $\hat{J}$ is the nominal value of the moment of inertia, $\hat{B}$ is the nominal value of viscous friction coefficient, $\hat{K}_t$ is the nominal value of the torque constant and T_c is the time constant of the low pass filter. Substituting equation 4.1 into equation 4.3 yields

$$\frac{I_f}{W_r} = \frac{(2\hat{J} + \hat{B}T) - (2\hat{J} - \hat{B}T)z^{-1}}{(2T_c\hat{K}_t + \hat{K}_t T) - (2T_c\hat{K}_t - \hat{K}_t T)z^{-1}}. \tag{4.4}$$

By dividing equation 4.4's denominator into its numerators and letting

$$C = \frac{2\hat{J} + \hat{B}T}{2\hat{K}_t T_c + \hat{K}_t T},$$

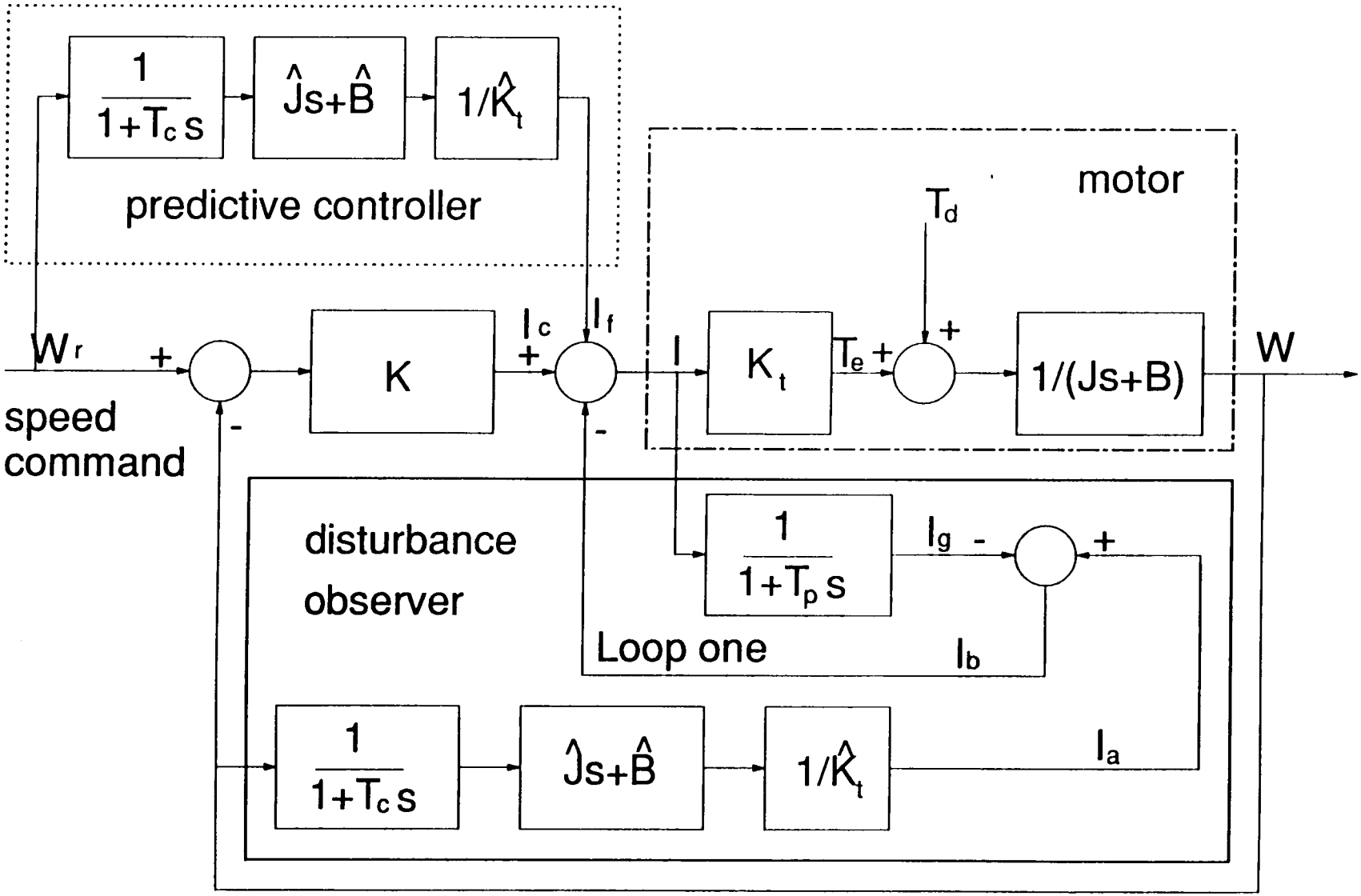


Fig. 4.1 Overall control block diagram.

one has the following equation:

$$I_f = CW_r + \frac{[C(2\hat{K}_t T_c - \hat{K}_t T) - (2\hat{J} - \hat{B}T)]W_r z^{-1}}{(2\hat{K}_t T_c + \hat{K}_t T) - (2\hat{K}_t T_c - \hat{K}_t T)z^{-1}}. \tag{4.5}$$

Let

$$Y = \frac{[C(2\hat{K}_t T_c - \hat{K}_t T) - (2\hat{J} - \hat{B}T)]W_r z^{-1}}{(2\hat{K}_t T_c + \hat{K}_t T) - (2\hat{K}_t T_c - \hat{K}_t T)z^{-1}}.$$

From equation 4.4 and equation 4.5,

$$I_f = CW_r + Y. \tag{4.6}$$

The relationship between Y and W_r can be seen from the following difference equation:

$$y(k) = \frac{(2\hat{K}_t T_c - \hat{K}_t T)}{(2\hat{K}_t T_c + \hat{K}_t T)} y(k-1) + \frac{C(2\hat{K}_t T_c - \hat{K}_t T) - (2\hat{J} - \hat{B}T)]}{(2\hat{K}_t T_c + \hat{K}_t T)} w_r(k-1). \tag{4.7}$$

Consequently, the output of this predictive controller (I_f) can be computed by

$$i_f(k) = Cw_r(k) + y(k). \tag{4.8}$$

4.3.2 Disturbance canceller digitization

The relationship between I_a and W in Fig. 4.1 can be written as

$$\frac{I_a}{W} = \frac{\hat{J}s + \hat{B}}{(1 + T_c s)\hat{K}_t} = \frac{\hat{J}s + \hat{B}}{T_c \hat{K}_t s + \hat{K}_t}. \tag{4.9}$$

Substituting equation 4.1 into equation 4.9 yields

$$\frac{I_a}{W} = \frac{(2\hat{J} + \hat{B}T) - (2\hat{J} - \hat{B}T)z^{-1}}{(2T_c\hat{K}_t + \hat{K}_t T) - (2T_c\hat{K}_t - \hat{K}_t T)z^{-1}}. \tag{4.10}$$

By dividing equation 4.10's denominator into its numerator, one obtains

$$I_a = CW + \frac{[C(2\hat{K}_t T_c - \hat{K}_t T) - (2\hat{J} - \hat{B}T)]Wz^{-1}}{(2\hat{K}_t T_c + \hat{K}_t T) - (2\hat{K}_t T_c - \hat{K}_t T)z^{-1}}, \tag{4.11}$$

where C is defined as last section and Q is defined as

$$Q = \frac{[C(2\hat{K}_t T_c - \hat{K}_t T) - (2\hat{J} - \hat{B}T)]Wz^{-1}}{(2\hat{K}_t T_c + \hat{K}_t T) - (2\hat{K}_t T_c - \hat{K}_t T)z^{-1}}.$$

From equation 4.10 and 4.11,

$$I_a = CW + Q. \tag{4.12}$$

The relationship between Q and W can be seen by the difference equation

$$q(k) = \frac{(2\hat{K}_t T_c - \hat{K}_t T)}{(2\hat{K}_t T_c + \hat{K}_t T)} q(k-1) + \frac{C(2\hat{K}_t T_c - \hat{K}_t T) - (2\hat{J} - \hat{B}T)]}{(2\hat{K}_t T_c + \hat{K}_t T)} w(k-1). \tag{4.13}$$

So, the equivalent disturbance plus control current (I_a) can be computed by

$$i_a(k) = Cw(k) + q(k). \tag{4.14}$$

If one continues to use the Tustin approximation on the filter between I_g and I in Fig. 4.1, the equation between I_g and I has the following form:

$$\frac{I_g}{I} = \frac{A}{s + A}, \tag{4.15}$$

where A is the inverse of the filter time constant. Substituting equation 4.1 into the above equation gives

$$H(z) = \frac{I_g}{I} = \frac{AT + ATz^{-1}}{2 + AT - (2 - AT)z^{-1}}, \tag{4.16}$$

where T is sampling time. Then the input and output are in the same time frame. If only the output depends on the input value, then one can always program to calculate the input first and then pass the information to the output. However, if the input and the output are dependent upon each other, this Tustin transformation method can not be applied to that situation. But, if the sampling time is small enough, one can delay

the input information one sampling period. Then

$$H(z) = \frac{I_g}{I} = \frac{ATz^{-1} + ATz^{-2}}{2 + AT - (2 - AT)z^{-1}}. \tag{4.17}$$

The above equation needs three variables, namely $ATz^{-1}/(2 + AT)$, $ATz^{-2}/(2 + AT)$, and (2-AT)$z^{-1}/(2 + AT)$. The output of low pass filter can be computed by

$$i_g(k) = \frac{2 - AT}{2 + AT} i_g(k - 1) + \frac{AT}{2 + AT} i(k - 1) + \frac{AT}{2 + AT} i(k - 2). \tag{4.18}$$

So, the disturbance current (I_b) can be computed by the following difference equation:

$$i_b(k) = i_a(k) - i_g(k). \tag{4.19}$$

When the Tustin transformation is applied to the disturbance canceller, one is forced to delay the input information one sampling period and use the three variables to realize the equation. Consequently, it is appropriate to try to determine if there is a better solution.

4.4 Another approach to digitize the control system

If one eliminates loop one in Fig. 4.1, and assuming $I_r = I_c + I_f$, then the following equation exists:

$$I = \frac{I_r - I_a}{1 - D}, \tag{4.20}$$

where D represents the low pass filter, $1/(1 + T_p s)$. If the nominal value of $T_p = .001s$, then $D = (1000)/(s + 1000)$. Substituting D into equation 4.20, yields

$$I = \frac{I_r - I_a}{1 - \frac{1000}{s+1000}} = \frac{I_r - I_a}{\frac{s}{s+1000}}. \tag{4.21}$$

Equation 4.21 can be written as

$$I = \frac{I_r(s+1000)}{s} - \frac{I_a(s+1000)}{s}. \tag{4.22}$$

Let

$$I_1 = \frac{I_r(s+1000)}{s} = \frac{I_r s + 1000 I_r}{s} \tag{4.23}$$

and

$$I_2 = \frac{I_a(s+1000)}{s} = \frac{I_a s + 1000 I_a}{s}, \tag{4.24}$$

then one eliminates the dependence of I and I_g. Substituting equation 4.1 into equations 4.23 and equation 4.24, gives

$$\frac{I_1}{I_r} = \frac{(2+1000T) - (2-1000T)z^{-1}}{2-2z^{-1}} \tag{4.25}$$

and

$$\frac{I_2}{I_a} = \frac{(2+1000T) - (2-1000T)z^{-1}}{2-2z^{-1}}. \tag{4.26}$$

Equations 4.25 and 4.26 have the same form as equation 4.4. If one follows the same procedure on equation 4.4, one should be able to digitize the system without delaying one sampling period for the input information.

Fig. 4.2 shows the comparison of the steady system output from a digital system with the loop eliminated with the steady state output of a continuous system without eliminating the loop. The two output responses are almost identical, as shown in Fig. 4.2.

If one only looks at the output without checking each variable inside the controller, then eliminating loop one seems to be a good approach for digitizing the system. However, by using this approach, one introduces two integrals, I_1 and I_2. From the mathe-

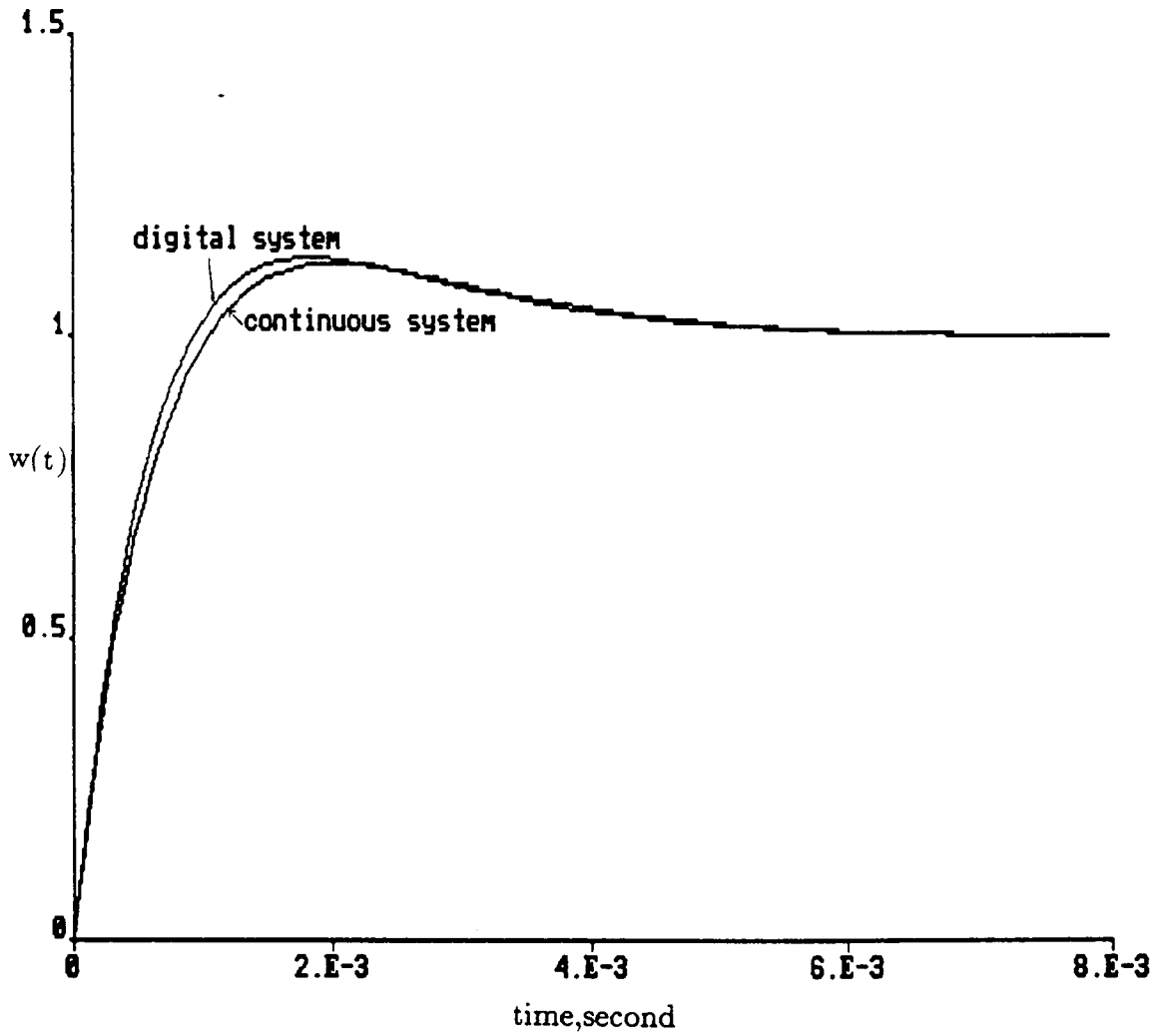


Fig. 4.2 Continuous and digital system output step response comparison.

matical point of view, one does not introduce any problem. I_1 and I_2 can go to infinity as long as the current (I) stay in a reasonable range.

However, when one needs to implement this, one is going to encounter some difficulty [7], because the largest number 12 bits can represent is 2^{12}. If the number is larger than 2^{12}, most microprocessors will report an overflow error or become negative representation. Fig. 4.3 shows the results of the simulation upon elimination of the loop. Variables I_1 and I_2 are getting larger and the output current (I) remains as in the continuous system.

The SIMNON [1] language program for this part of the simulation is listed in Appendix A under file names of Dac7a, Ac1a and conn7, where Dac7a is the digital program, Ac1a is the analog plant program and conn7 is the connecting program.

4.5 Direct z-transform

Eliminating loop one in Fig. 4.1 creates two integrals. Therefore another method to transform the system is needed. The transformation from s to z domain for the low pass filter between I_g and I can be accomplished by the direct z-transform. Also, the Tustin transformation can be used to transfer the inverse of the motor dynamic in the predictive controller and disturbance observer. From the table of z-transforms, the transfer function

$$H(s) = \frac{I_g}{I} = \frac{A}{s + A} \tag{4.27}$$

can be transformed into

$$H(z) = \frac{I_g}{I} = \frac{A}{1 - e^{-AT}z^{-1}} \times T, \tag{4.28}$$

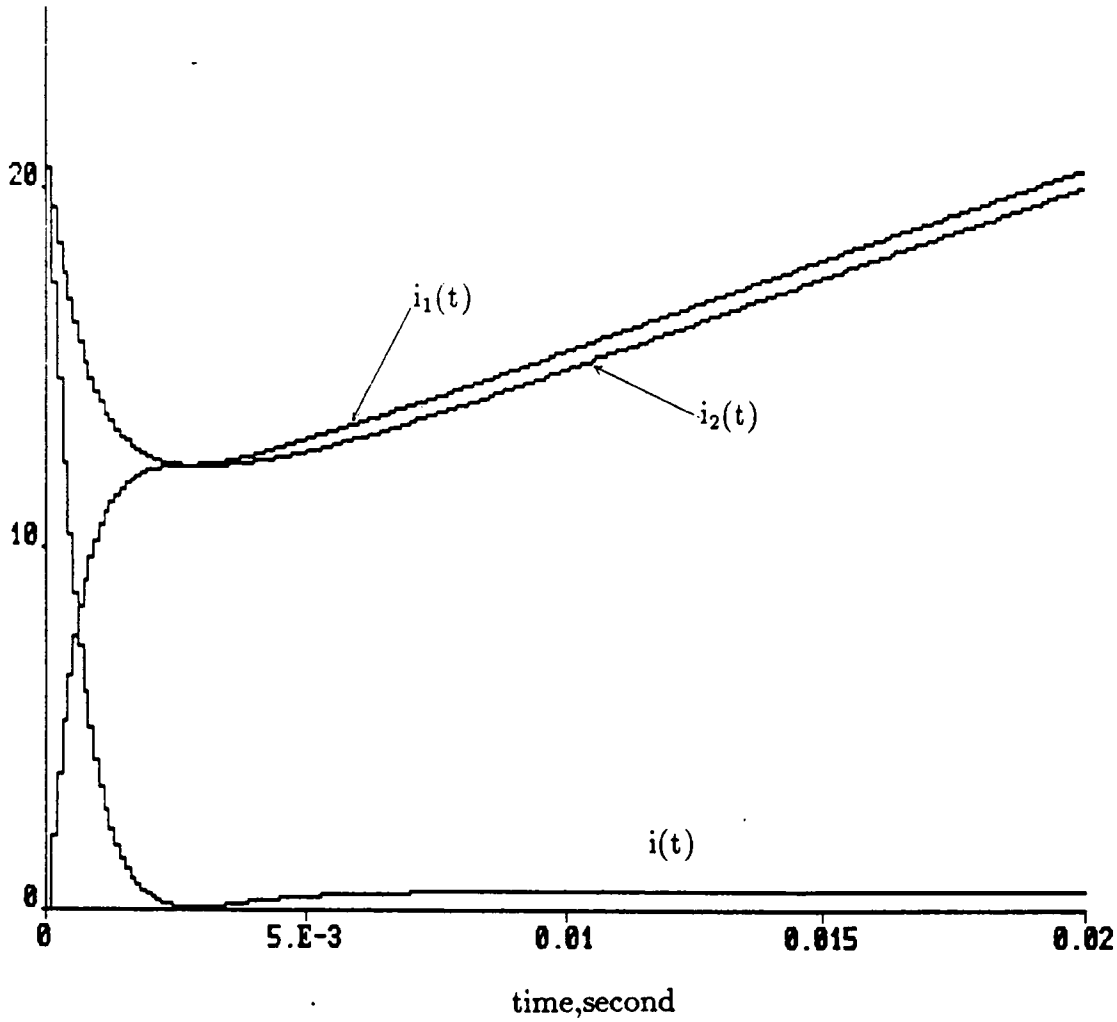


Fig. 4.3 Plot of controller variable when loop eliminated

where T is the sampling time and A is the inverse of the low pass filter time constant. Multiplication by T is to compensate for a gain loss of 1/T due to sampling. Since the sampling time is small, one can delay the input information one sampling period. Then

$$H(z) = \frac{I_g}{I} = \frac{\frac{A}{T}z^{-1}}{1 - e^{-AT}z^{-1}}. \tag{4.29}$$

One advantage of using the direct z-transform is that it is comparatively easy to implement. Also, the accuracy of the response of this z domain transfer function is good. More importantly, using the above transformation method, the transfer function has only 2 variables, A/Tz^{-1} and $(e^{-AT})z^{-1}$. The addition of one variable may not seem significant, but in writing the program for the microprocessor, it does make some difference. One variable may require two RAM spaces and tens of instruction sets to pass the value to the other variable.

4.6 Comparison between continuous and discrete systems

From the above discussion, it would be desirable to digitize the low pass filter between I_g and I with the direct z-transform, compensate with T and then use the Tustin method to transform the inverse of the motor dynamic on the predictive controller and the disturbance observer.

After digitizing the continuous system, one may want to know the performance of the digital system compared with the continuous system. In the following, three comparisons between a digital system and a continuous system will be given.

The first comparison will be of the ability to suppress the disturbance. Fig. 4.4

shows the steady-state response of the output with a different magnitude disturbance applied at .02 s to the continuous system. Fig. 4.5 shows the same plot for the digital system. From these two figures, one can say that the digital system response is almost the same as that of the continuous system and the ability to suppress the disturbance remains.

The second comparison is of the steady state response of all system variables. Fig. 4.6 shows the steady-state response of the continuous system for all of the variables inside the controller. Fig. 4.7 shows the same plot for the digital system. These two figures reveal that with a reasonable input, the variables inside the controller will remain within a reasonable range.

The third comparison is to check if all the variables will remain within a reasonable range when the disturbance is applied. Fig. 4.8 shows the steady-state response of the continuous system for all of the variables inside the controller with a step disturbance applied at .01 s. Fig. 4.9 shows the same plot for the digital system. These two figures show that even with a disturbance , all the variables stay within a reasonable range. From the simulation of the continuous system and the simulation of the digital system, the transformation performed as expected.

The SIMNON language program for this part of the simulation is listed in Appendix A under the file names of nan1, ren1 and huang1. A macro program is also listed in Appendix A. for repeat simulations. This feature is a useful addition to the normally available SIMNON capabilities.

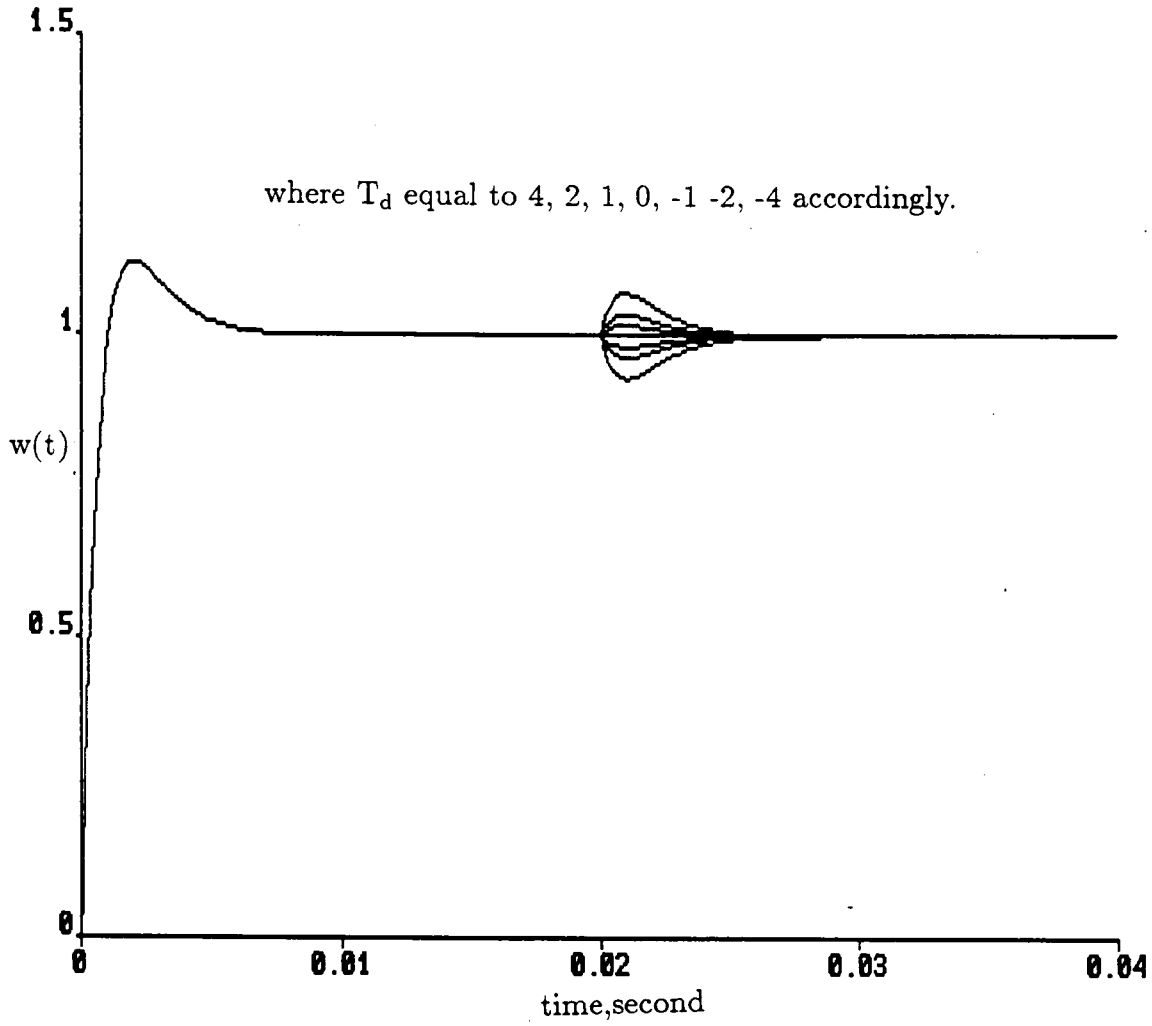


Fig. 4.4 Step response of continuous system output with disturbance applied at .02 second.

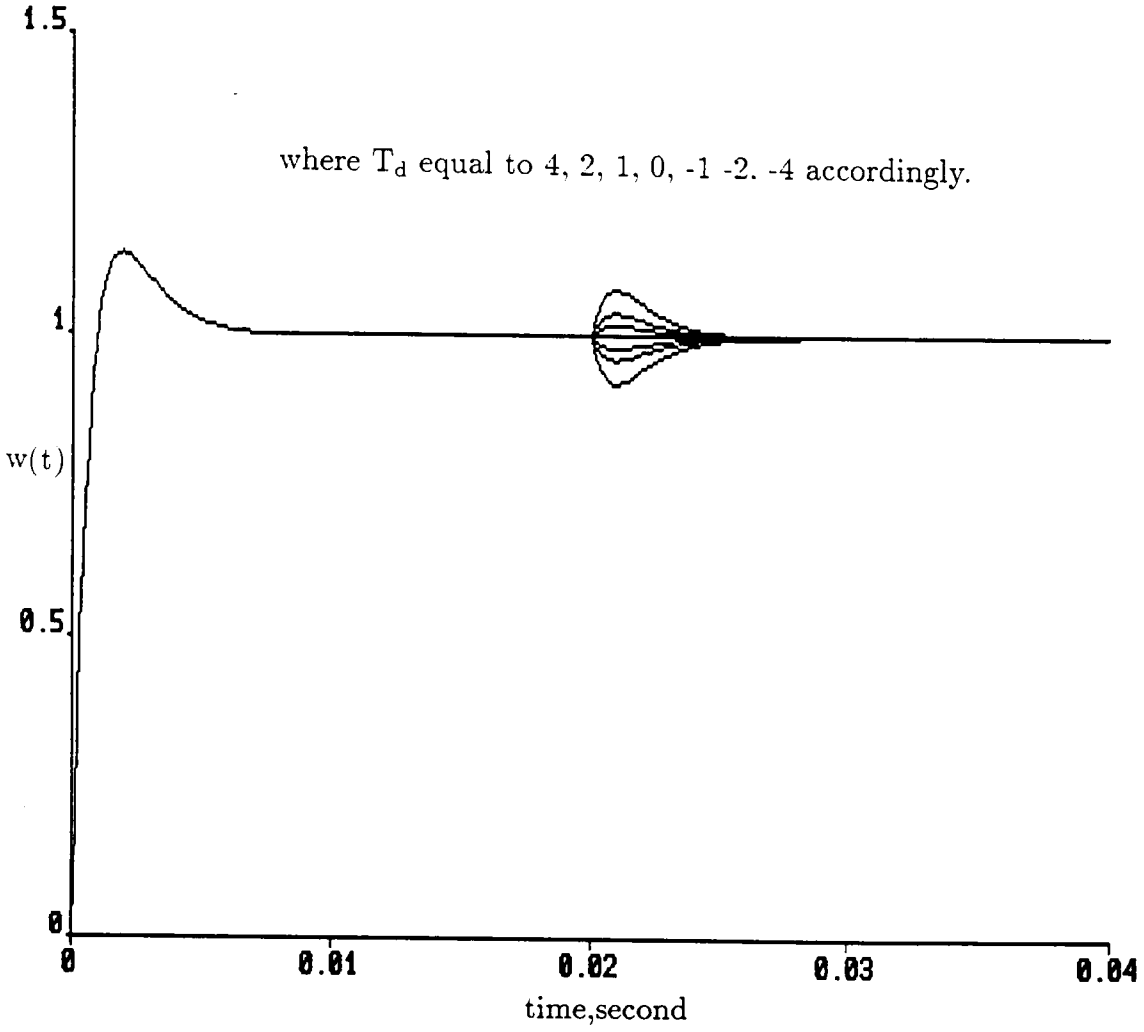


Fig. 4.5 Step response of digital system output with disturbance applied at .02 second.

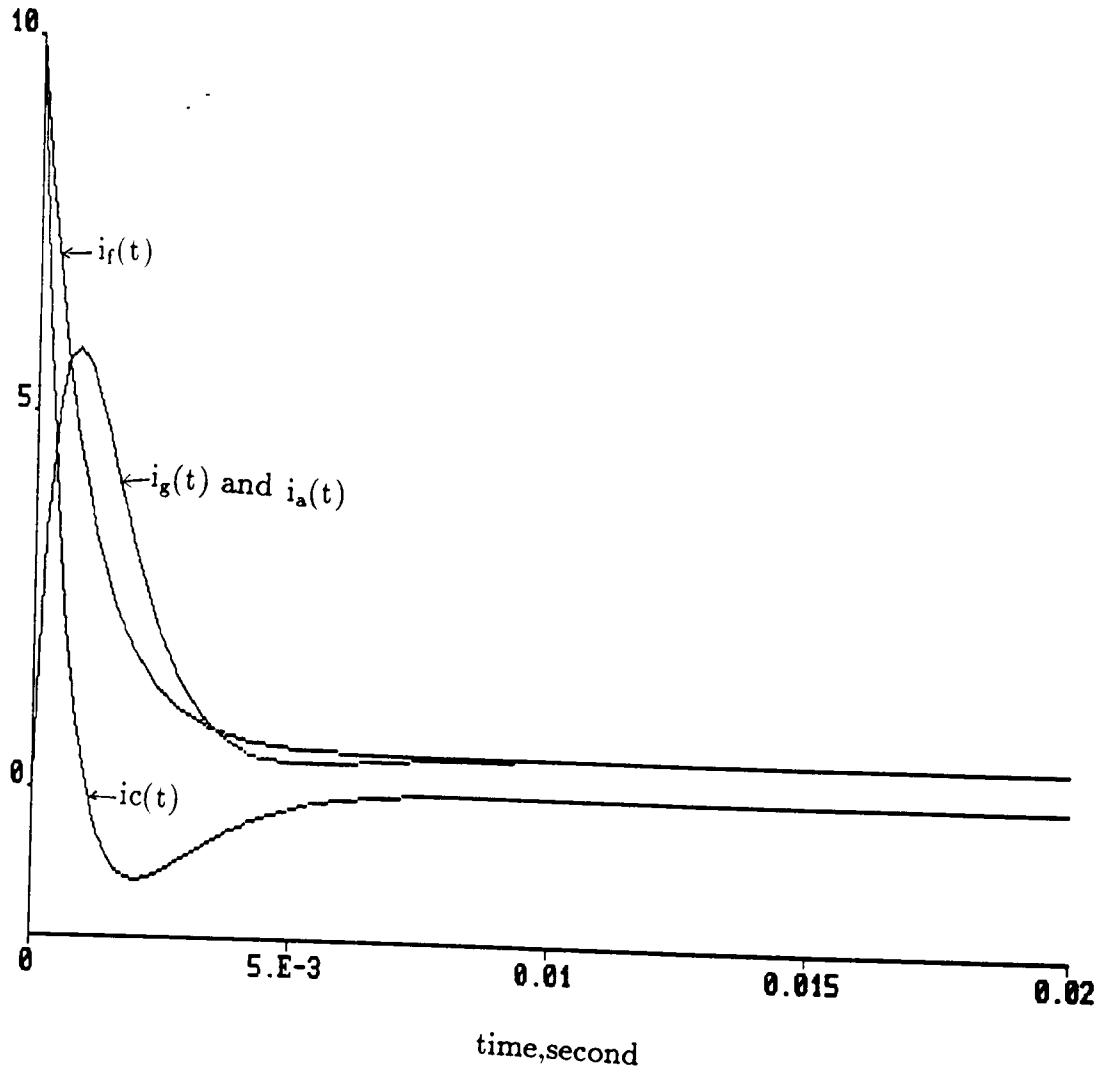


Fig. 4.6 Steady-state response for all variables in continuous system.

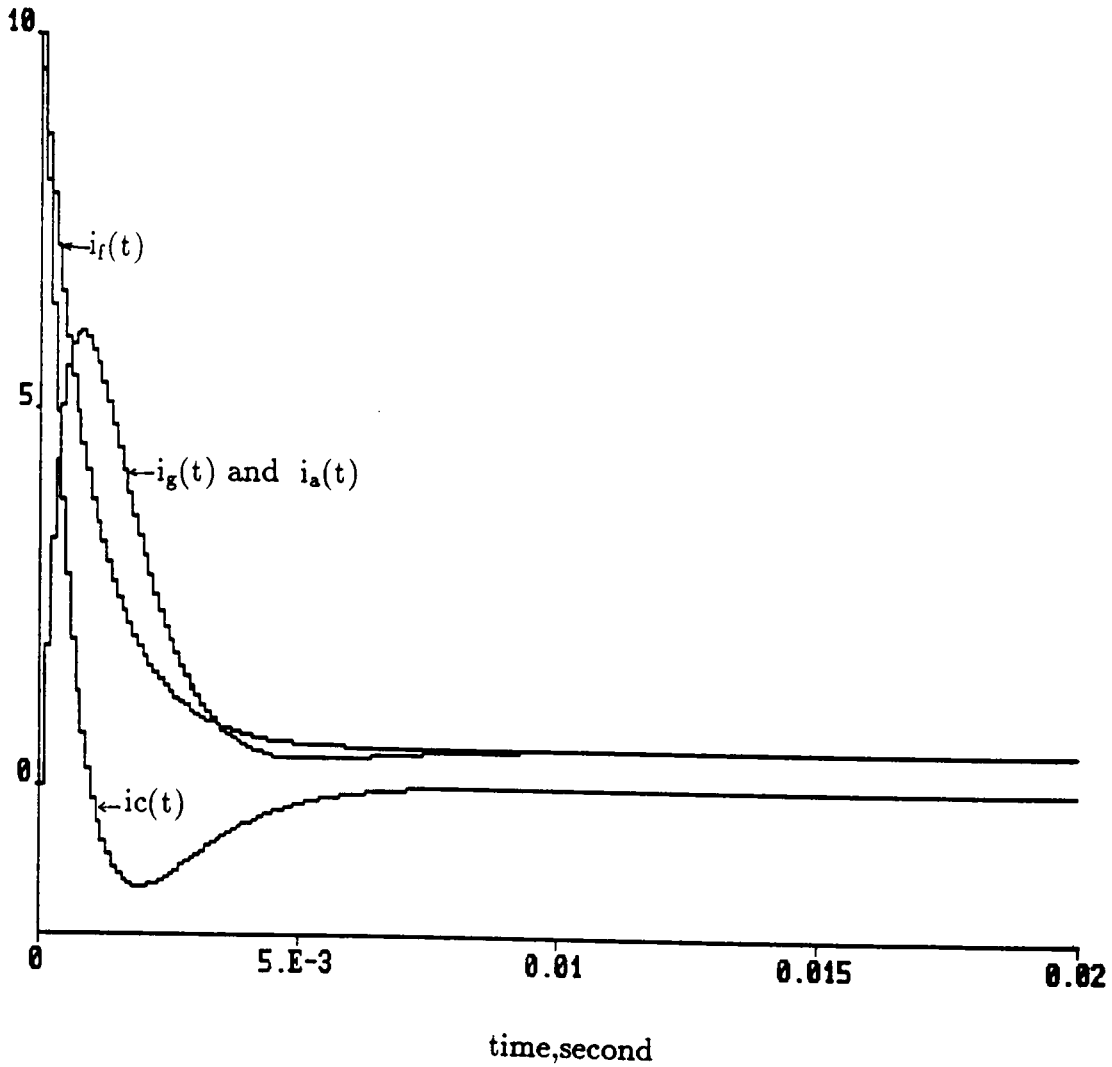


Fig. 4.7 Steady-state response for all variables in digital system.

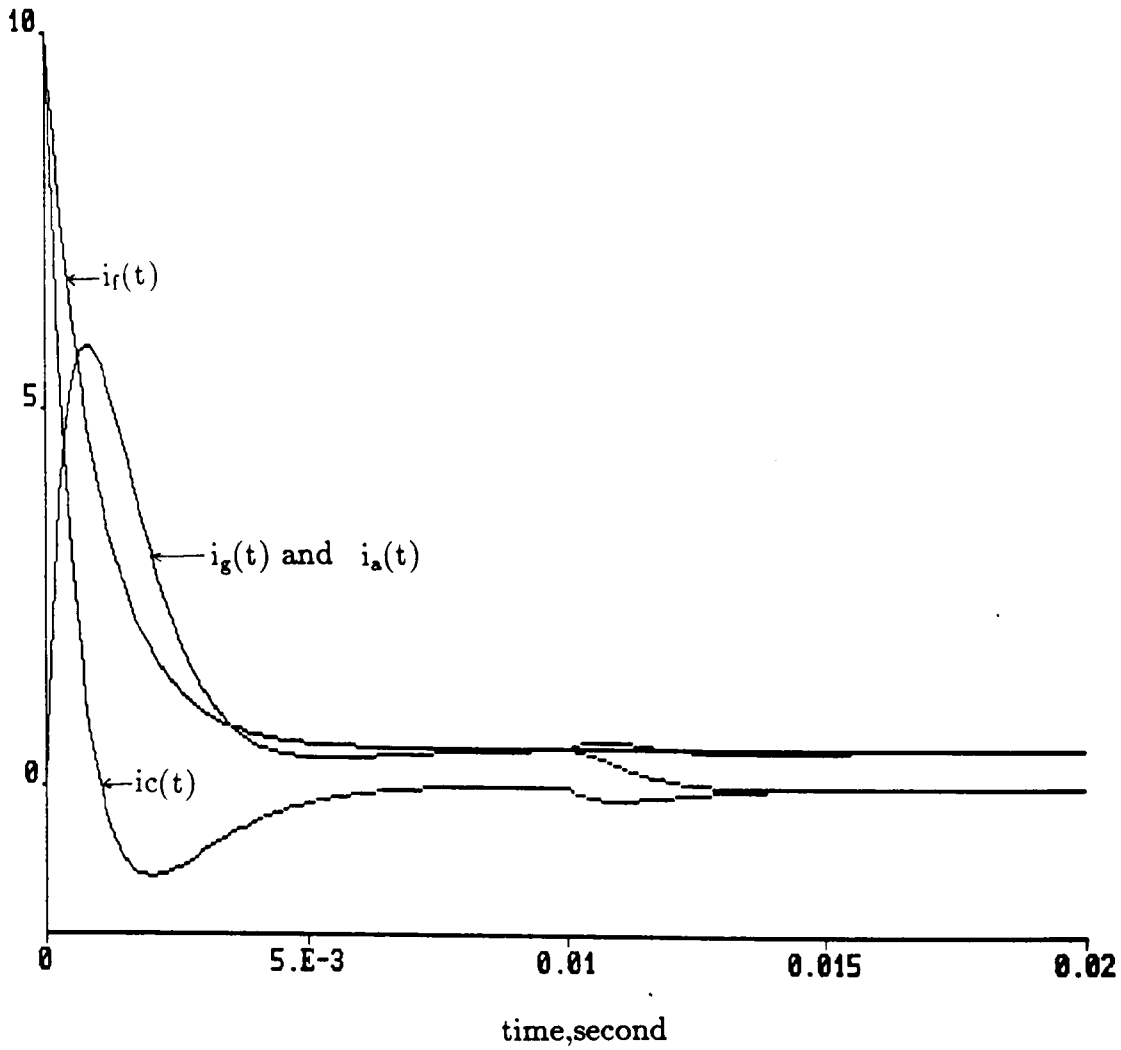


Fig. 4.8 Steady-state response for all variable in continuous system with disturbance applied at .01 second.

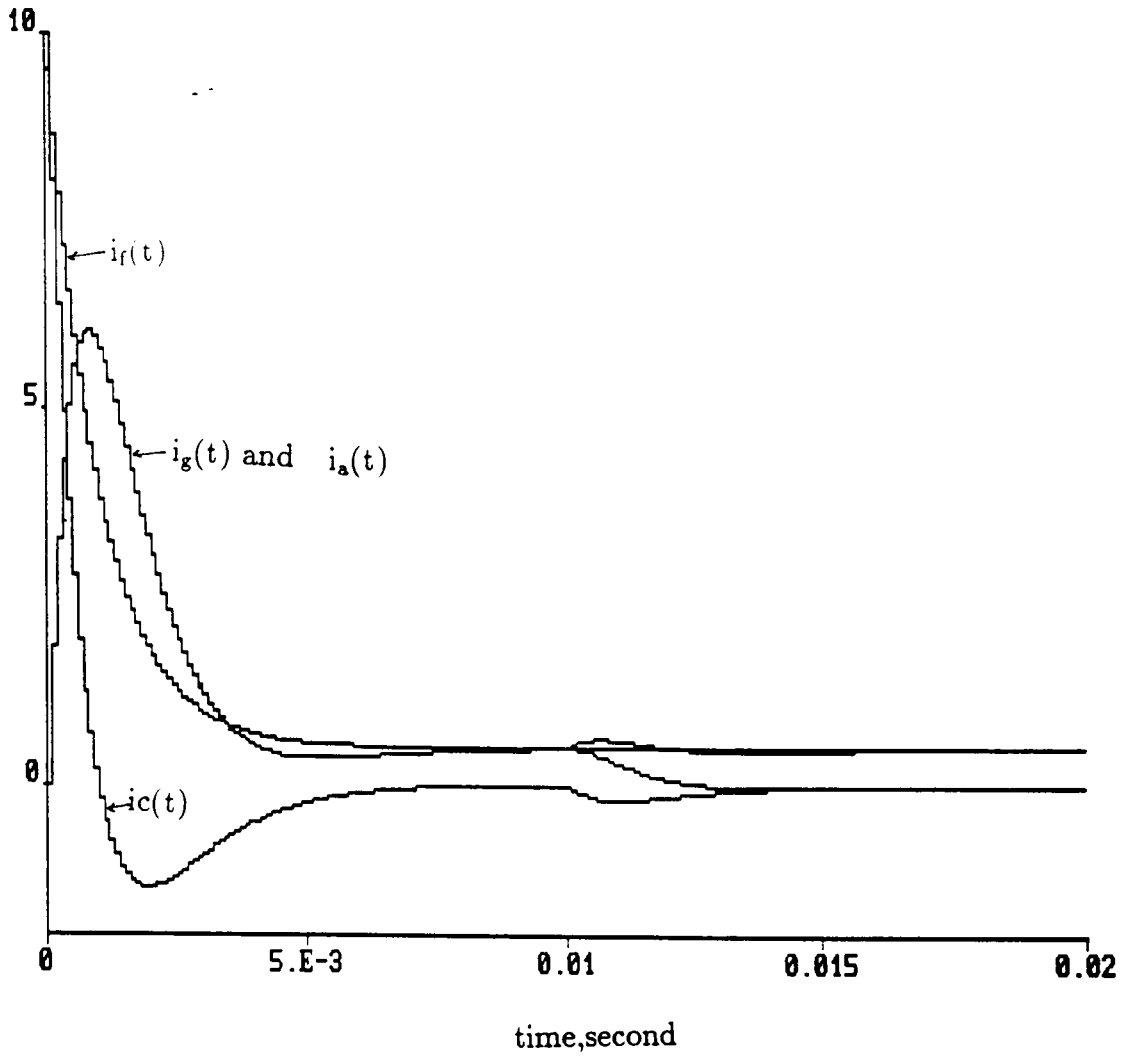


Fig. 4.9 Steady-state response for all variable in digital system with disturbance applied at .01 second.

4.7 Conclusion

From the simulation, it is shown that two transformation methods can simultaneously digitize a continuous system. Ideally, the z-domain transfer function, which is obtained by digitizing an s-domain transfer function, maintains all the properties of the s-domain transfer function.

Thus, while this is not achieved completely, the transformation preserves three particular properties which are of interest to the designer. These three properties are DC gain, stability, and step response. The step response property has been shown in the last section, and the last section also shows that the transformation optimized the allowed sampling range. The transformation maintains the stability of the s-domain control system. That is, the transformation of a stable s-domain control system is stable in z domain. The DC gain for the s-domain and z-domain control system transfer functions are identical, i.e.

$$H(s)|_{s=0} = H(z)|_{z=1}$$

for the predictive controller and one branch of the disturbance observer. Finally, with comparison of the continuous and digital simulations, it is clear that the transformation preserves the most important properties of the motor control system.

Chapter 5

Control System and Software Design

5.1 Introduction

The motor speed is controlled by a digital signal processor (DSP) based system. This system is comprised of a 10 MHz clock 80286 CPU[1] PC-AT compatible personal computer and a 25.6 MHz clock TMS320C14 DSP[2] based control system - Power- 14[3]. The PC-AT provides a user interactive interface and program development while the TMS320C14 DSP does most of the computation and data I/O. The complete computer speed control system diagram is shown in Fig. 5.1.

5.2 Data acquisition system

---

[1] Trademark of Intel Corp.

[2] Trademark of Texas Instrument Inc.

[3] Trademark of Teknic Inc.

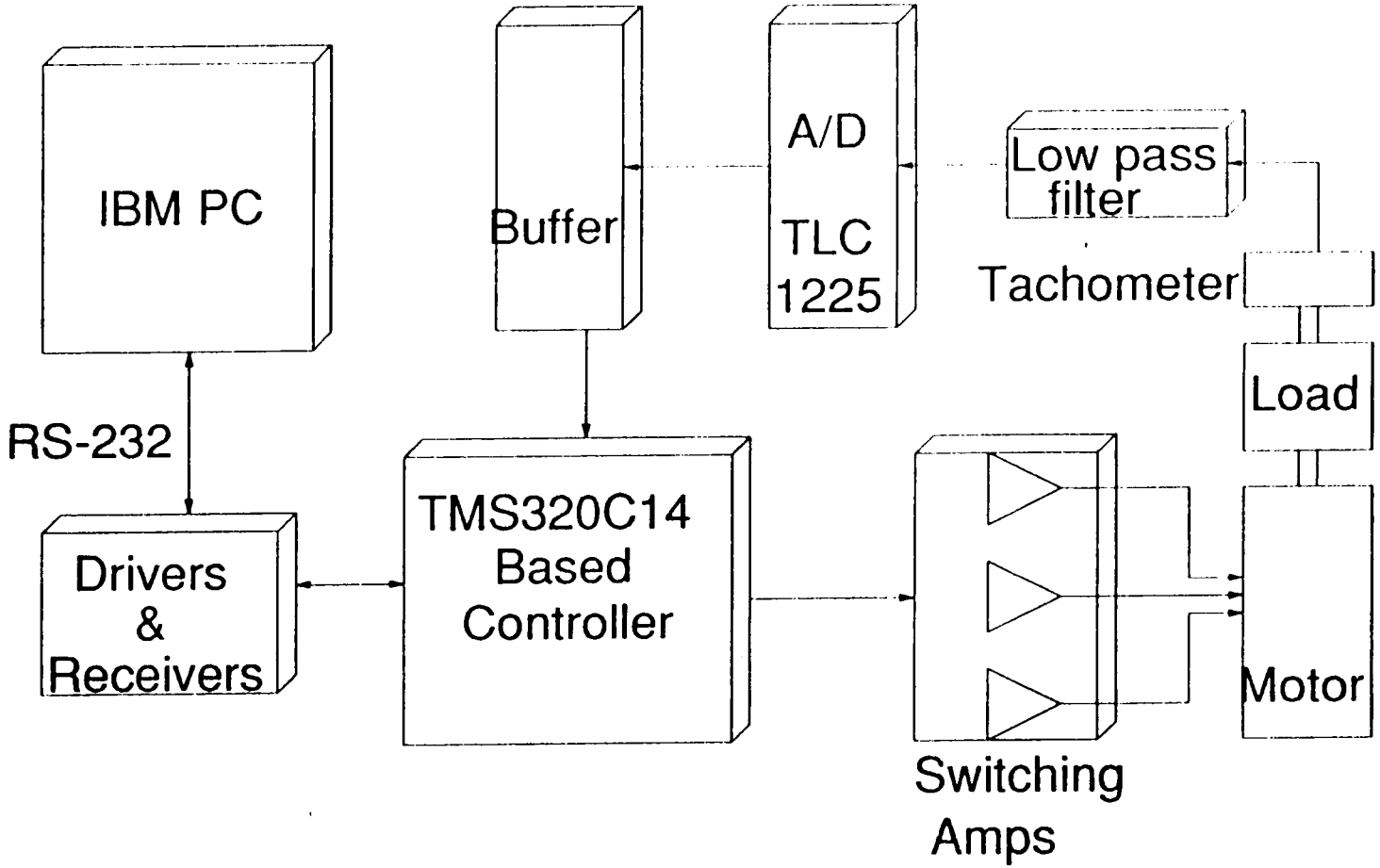


Fig. 5.1 Overall hardware system.

The data acquisition system includes a 2 channel analog to digital signal input [8]. This A/D uses a 50 KHz conversion rate. An on-board stable +5 V reference is used as the upper input reference of the converter. The +5 V reference is inverted to produce a -5 V lower reference. The A/D allows bipolar operation with analog input ranging in amplitude from -5 V to +5 V.

The output mode can be chosen by the output configuration logic. Four different output modes are available. In this experiment, the two channel pass-through mode was chosen. The servo power supply is chosen at 20V and .4A. Using this servo power supply will enable the Power-14 to generate -1V to +1V of output.

Only one input channel is used for speed feedback. The input signal is filtered by a 20 KHz anti-aliasing low pass filter. One output channel feeds the desired current signal into the motor model. The sampling rate is programmable. However, in this experiment, the output sampling rate is set at 10 KHz to allow for more computation and communication time.

5.3 Input/Output scaling

As mentioned earlier, the analog signal in the Power-14 is ranges from -5 V to +5 V, and digital representation is from -32768 to 32767. The input signal flow diagram is shown in Fig. 5.2. If the speed signal is from -1 V to +1V, then output of the A/D is represented by -819 to 818. With software scaling, the -1V to +1V can be represented by -4096 to 4095 in Q12 representation [9]. Q format is a number representation commonly used when performing operations on noninterger numbers. In Q format, the Q number

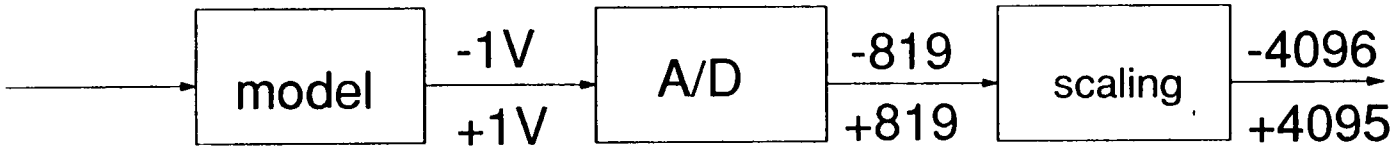


Fig. 5.2 Input signal diagram.

(12 in Q12) denotes how many bits are located to the right of the binary point. A 16-bit number in Q12 format, therefore, has 3 assumed binary points to the right of the most significant bit. Since the most significant bit constitutes the sign of the number, then the number represented in Q12 may take values from +7.9999 to -8.

5.4 Handshaking Design

The communication between the PC and the Power-14 requires handshaking. This means both sides should be available for data transmission. The I/O port 1 on the Power-14 is designed for the PC bus. Bit 13 of port 1 indicates that the PC is going to send data while bit 14 indicates that the PC is ready to receive data. Thus, the handshaking can be simply implemented by checking port 1 and waiting for the availability of the PC or DSP320C14 [14]. The procedure for reading data from the PC can be summarized as:

1. Check bit 13 of port 1.

2. If it is low, go back to check.

3. If it is high, get data from PC.

Similarly, the procedure of sending data to the PC can be summarized as

1. Check bit 14 of port 1.

2. If it is low, go back to check.

3. If it is high, send data to PC.

The RS-232 connector is wired to interface to a DB-9 female connector. As such, the Power- 14 acts as DTE (Data Terminal Equipment) configured for connection to a

MODEM.

5.5 Motor Model

Instead of using a motor, an op-amp model for the motor was constructed. Fig. 5.3 shows the model of the motor transfer function and the torque constant. The model provides a means to apply a disturbance. [10]

5.6 Software Design

Software development process

Software development is a multi-step process for the Power-14 board. The program is first written in assembly language using an ASCII output word processor. This source file is then passed to the TMS320 cross-assembler which assembles the program into a binary format and creates an object file. Object files are joined with the linker. In using the Power-14, one final step must be performed. That is, the output file from the linker must be passed through a conversion program which allows the COFF (common object file format) developed code to run in the TI.TAG environment. The Power-14 was developed using TI.TAG. A utility called DSPROM is available to convert the COFF file to a TI.TAG file [12]. Fig. 5.4 illustrates the software development process, where the Simulator is a software program that simulates operation of the TMS320C14 for software development [13].

The major topics discussed in this section include: (1) Power-14 board initialization. (2) Monitor program design [11,12,13]. (3) Predictive controller program design. (4)

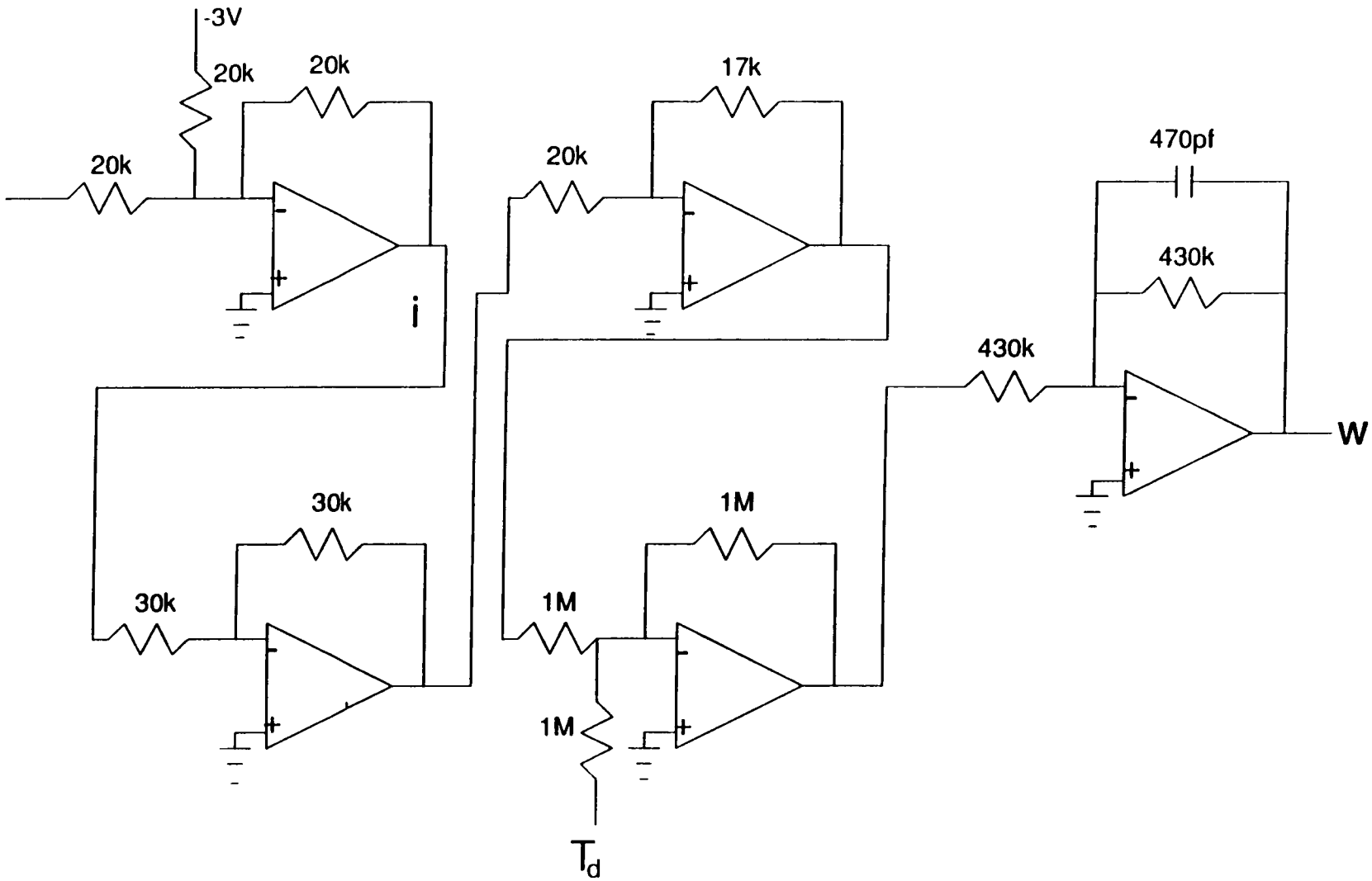


Fig. 5.3 Motor model.

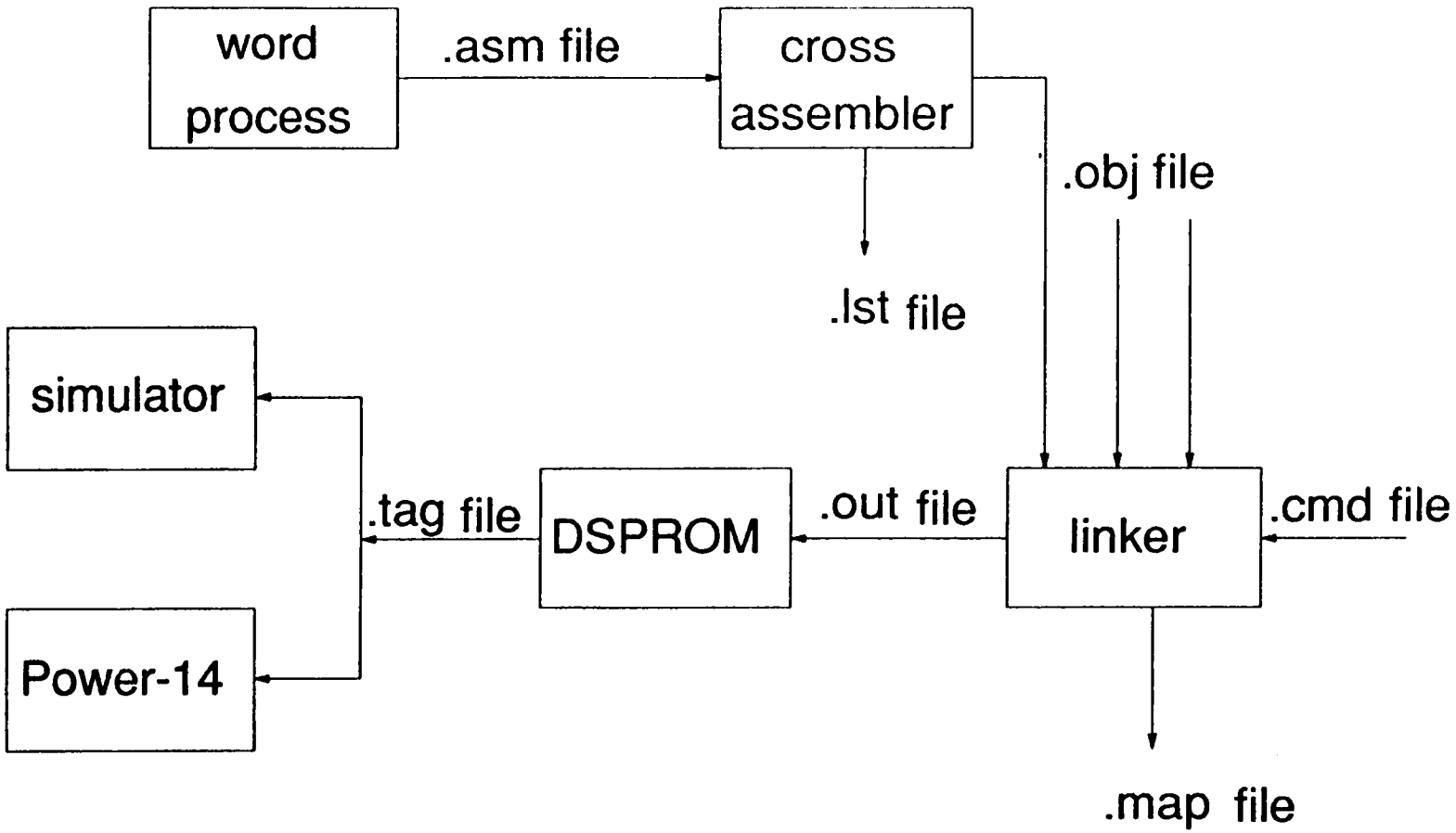


Fig. 5.4 Software development process.

Disturbance canceller program design. (5) Main program design. (6).Overall program design. In each of the topics, a flow diagram will be given to help to explain the program flow. Each program is listed in Appendix B. A list of these programs is as follows: (1) Initialization program, (2) Main disturbance observer based program, (3) Output program, (4) Q12 format program, (5) Proportional control program, (6) Feedforward control program, (7) Disturbance canceller program. (8) Monitoring program, (9) Servo integral gain program, (10) Servo proportional gain program, (11) Servo differential gain program. (12) UTK command program, (13) UTK help program, (14) Change speed program, (15) Change gain program, (16) Change 1st parameter program, (17) Change 2nd parameter program, (18) Change 3rd parameter program, (19) Change 4th parameter program, (20) Change 5th parameter program, (21) Change 6th parameter program, (22) Change 7th parameter program, (23) Change 8th parameter program, (24) Print RAM value.

5.6.1 Power-14 board initialization

It is necessary to initialize the system prior to execution of a DSP algorithm. Initialization is normally done following a reset so that the processor will meet the requirements of the system. In this control system, the following functions should be considered:

(1) Auxiliary register pointer.

(2) Data memory page pointer.

(3) Overflow mode.

(4) Watchdog timer.

(5) Data address.

(6) Detect any error conditions.

The start up and interrupt vector codes are contained in this program. Data constants and data addresses are defined,and each data constant is initialized. A/D conversion is also set up in this program. The support system (such as baud rate setup, watch dog timer period register) to interface with the control program is also initialized. The flow diagram in Fig. 5.5 shows the initialization procedure.

5.6.2 Monitor program design

In order to monitor each individual variable, a monitor program has been developed. This program can simultaneously print 6 variables to the screen in hex format. Also, this program will only print out the variable at the user specified time interval in order to allow more computation. The flow diagram in Fig. 5.6 shows the monitor program procedure.

5.6.3 Predictive controller program design

As mentioned in Chapter 4, after digitization, the output of the predictive controller (I_f) can be computed by

$$i_f(k) = Cw_r(k) + y(k), \tag{5.1}$$

where

$$C = \frac{2\hat{J} + \hat{B}T}{2\hat{K}_t T_c + \hat{K}_t T}$$

and

$$Y = \frac{[C(2\hat{K}_t T_c - \hat{K}_t T) - (2\hat{J} - \hat{B}T)]W_r z^{-1}}{(2\hat{K}_t T_c + \hat{K}_t T) - (2\hat{K}_t T_c - \hat{K}_t T)z^{-1}}.$$

Consequently, one can program to calculate $i_f(k)$,

$$i_f(k) = i_{f1}(k) + i_{f2}(k), \tag{5.2}$$

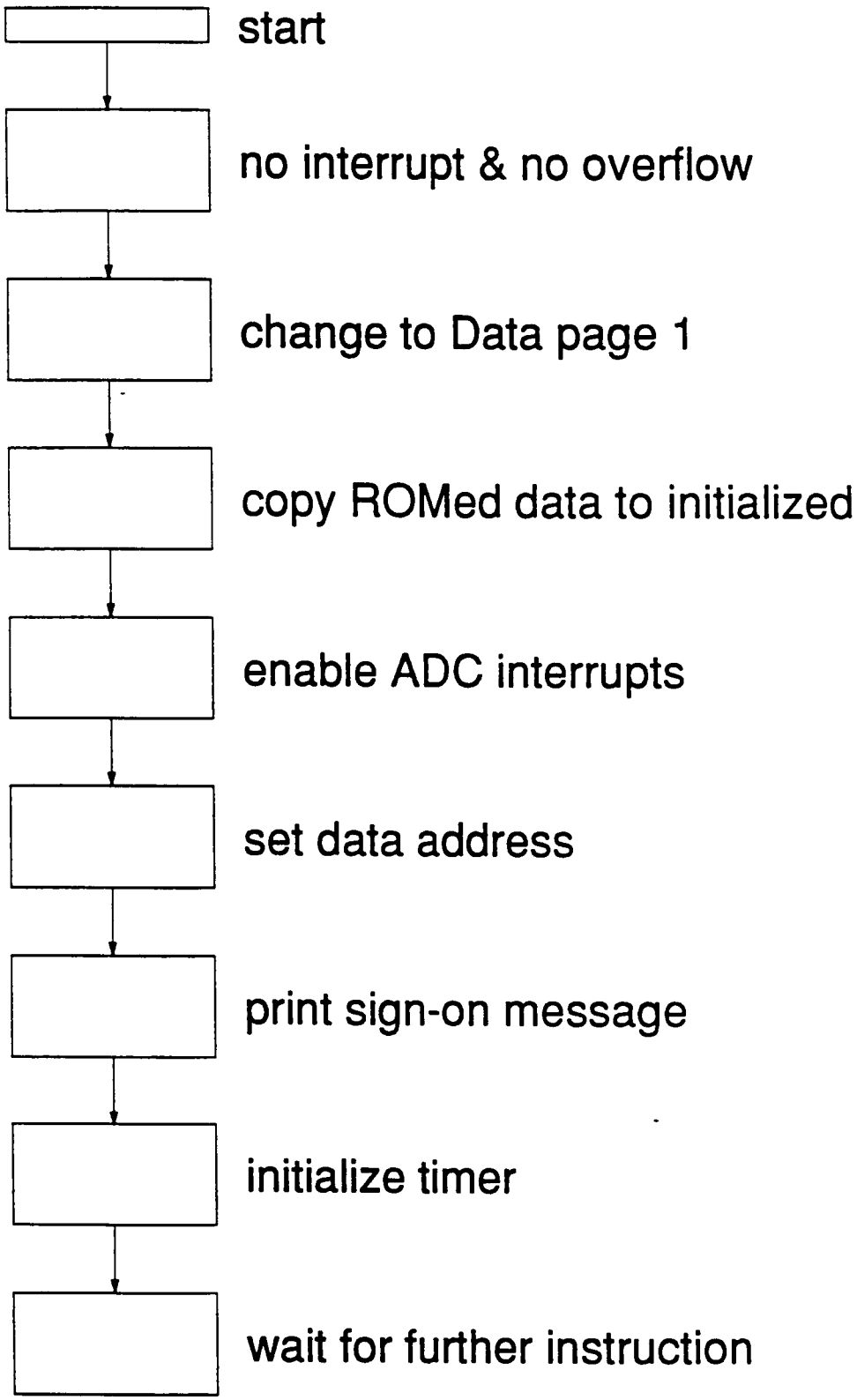


Fig. 5.5 Flow diagram for initialization.

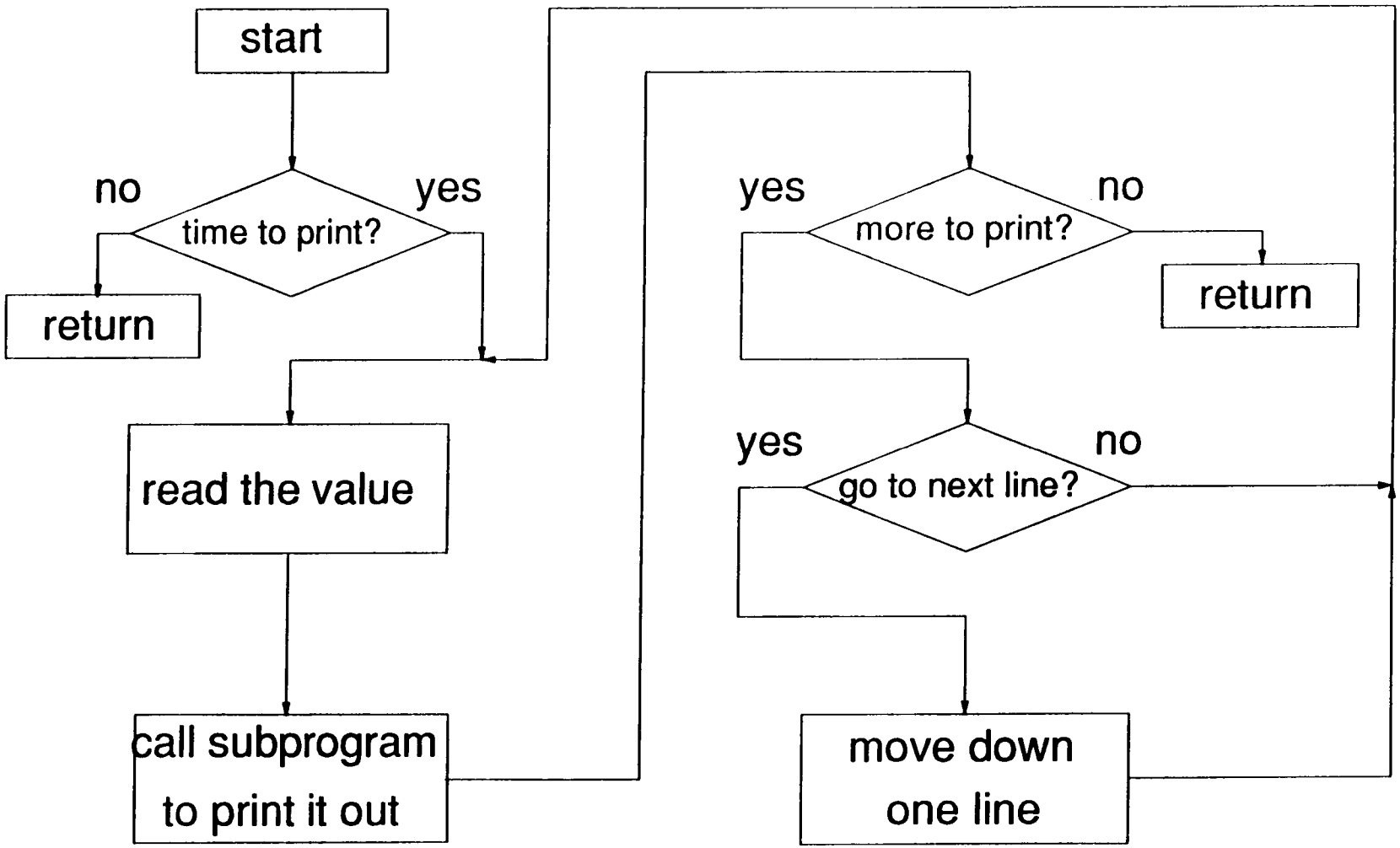


Fig. 5.6 Flow diagram for monitor program.

where $i_{f1}(k)$ equals to $Cw_r(k)$ and

$$i_{f2}(k) = i_{f21}(k) + i_{f22}(k), \tag{5.3}$$

where

$$i_{f21}(k) = \frac{(2\hat{K}_t T_c - \hat{K}_t T)}{(2\hat{K}_t T_c + \hat{K}_t T)} i_{f2}(k-1) \tag{5.4}$$

and

$$i_{f22}(k) = \frac{C(2\hat{K}_t T_c - \hat{K}_t T) - (2\hat{J} - \hat{B}T)]}{(2\hat{K}_t T_c + \hat{K}_t T)} w_r(k-1). \tag{5.5}$$

The procedure to compute the output of the predictive controller is best described in the flow chart of Fig. 5.7.

5.6.4 Disturbance canceller program design

The disturbance canceller program design can be divided in two parts. The first part is to compute I_g. The second part is to compute I_a. The procedure to calculate I_g is described in the following:

$$H(s) = \frac{I_g}{I} = \frac{A}{s+A},$$

where A is the inverse of low-pass filter time constant, and I_g is the output of the low-pass filter. From the discussion of digitization in chapter 4, I_g can be calculated as

$$i_g(k) = e^{-AT} i_g(k-1) + \frac{A}{T} i(k-1).$$

Consequently, one can program the following way to calculate the I_g:

$$i_g(k) = i_{g1}(k) + i_{g2}(k), \tag{5.6}$$

where $i_{g1}(k) = e^{-AT} i_g(k\text{-}1)$ and $i_{g2}(k) = (A/T)i(k\text{-}1)$. The second part is to compute I_a. As mentioned before, I_a is the estimated disturbance current plus control current

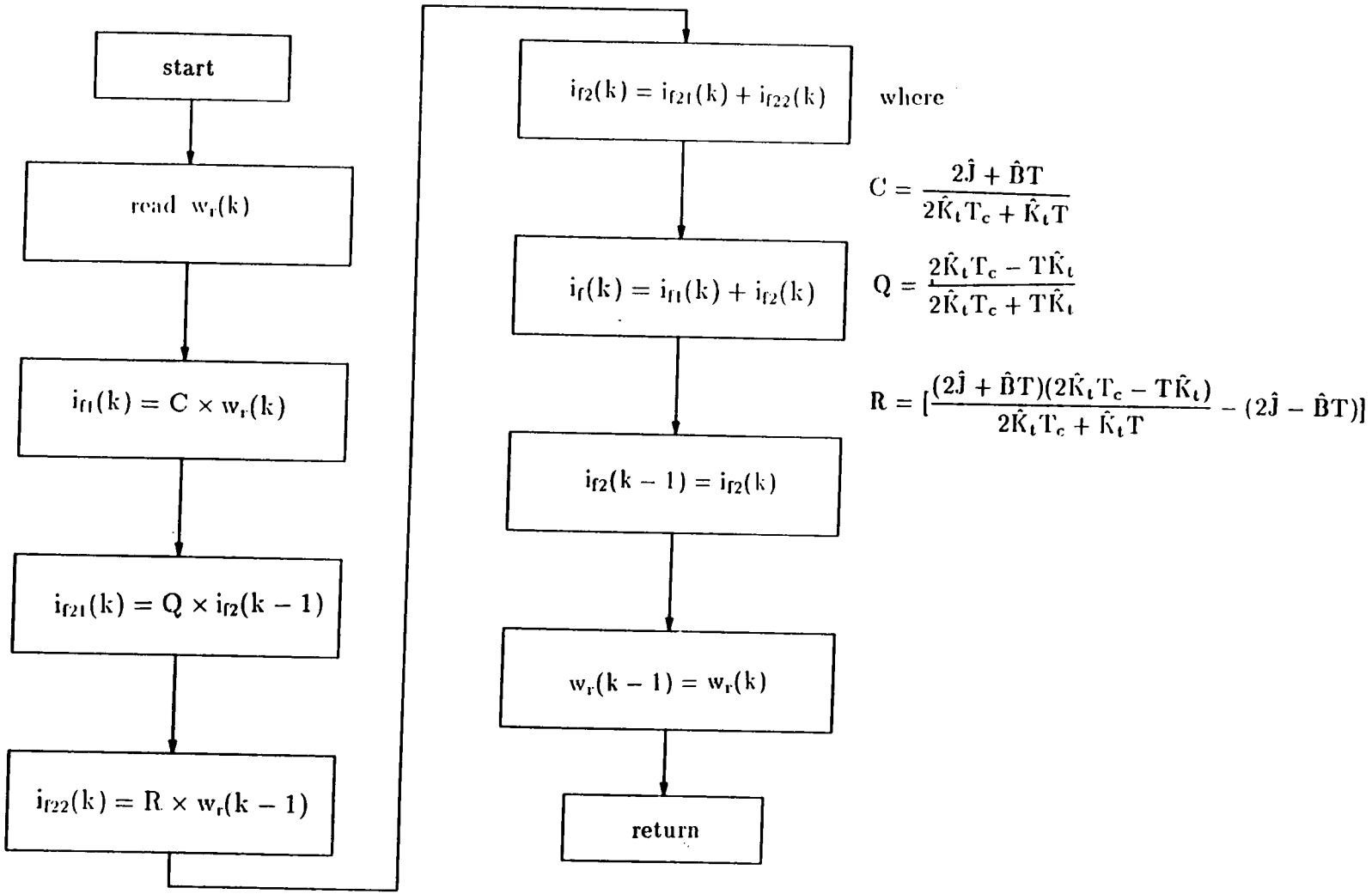


Fig.5.7 Flow diagram for predictive controller

and can be computed by

$$i_a(k) = Cw(k) + q(k), \tag{5.7}$$

where q(k) is equal to

$$q(k) = \frac{(2\hat{K}_t T_c - \hat{K}_t T)}{(2\hat{K}_t T_c + \hat{K}_t T)} q(k-1) + \frac{C(2\hat{K}_t T_c - \hat{K}_t T) - (2\hat{J} - \hat{B}T)]}{(2\hat{K}_t T_c + \hat{K}_t T)} w(k-1). \tag{5.8}$$

Consequently, one can program to calculate $i_a(k)$ as

$$i_a(k) = i_{a1}(k) + i_{a2}(k), \tag{5.9}$$

where $i_{a1}(k)$ equal to Cw(k) and

$$i_{a2}(k) = i_{a21}(k) + i_{a22}(k), \tag{5.10}$$

where

$$i_{a21}(k) = \frac{(2\hat{K}_t T_c - \hat{K}_t T)}{(2\hat{K}_t T_c + \hat{K}_t T)} i_{a2}(k-1) \tag{5.11}$$

and

$$i_{a22}(k) = \frac{C(2\hat{K}_t T_c - \hat{K}_t T) - (2\hat{J} - \hat{B}T)]}{(2\hat{K}_t T_c + \hat{K}_t T)} w(k-1). \tag{5.12}$$

After I_g and I_a have been obtained, I_b can be computed as

$$i_b(k) = i_a(k) - i_g(k). \tag{5.13}$$

The procedure to compute the output of the predictive controller is best described in the flow chart of Fig. 5.8.

5.6.5 Main controller program

The main controller program has two functions. The first is for the controller calculation. The second function is to print out the contents of RAM. This second function is

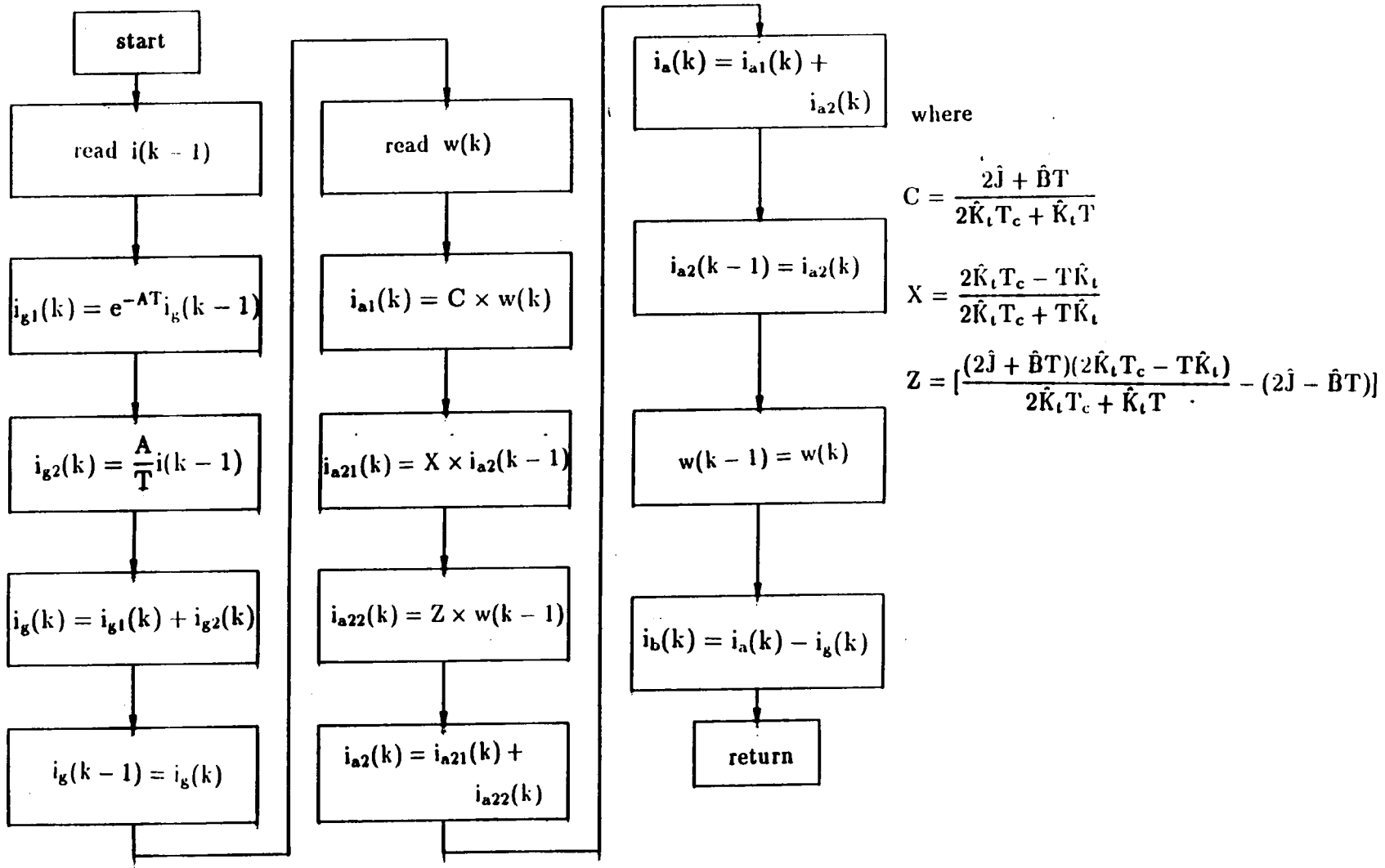


Fig. 5.8 Flow diagram for disturbance canceller

developed for debugging purposes. In the controller program, all of the parameters are calculated in a subprogram. The main program is used to collect all of the parameters and perform the necessary arithmetic for each parameter. After the control current is calculated, the main program calls the subroutine to generate the output. Fig. 5.9 is the flow diagram for this part of the program.

5.6.6 Overall program design

The overall program can be divided into two parts. The first part is the background, which handles the interrupts, A/D conversion, and communication with the PC-AT. The second part is the foreground part which handles the controller calculation, the screen printing, and RAM contents printing. Fig. 5.10 is the flow diagram for the overall program.

5.7 Experimental results of speed responses

After computer simulation verification of the developed algorithm and successful testing of the model, the complete system was tested with a reference command. The system response is shown in Fig. 5.11. When applied with a step disturbance, the system response shows good agreement with the simulation. Fig. 5.12 shows the system response with a step disturbance at .41 to .92 second. Fig. 5.13 shows the step response for the system without a disturbance observer. Fig. 5.14 shows the system response with a step disturbance at .04 to .82 second. Fig. 5.12 and Fig. 5.14, shows that the system with a disturbance observer is better in handling the disturbance.

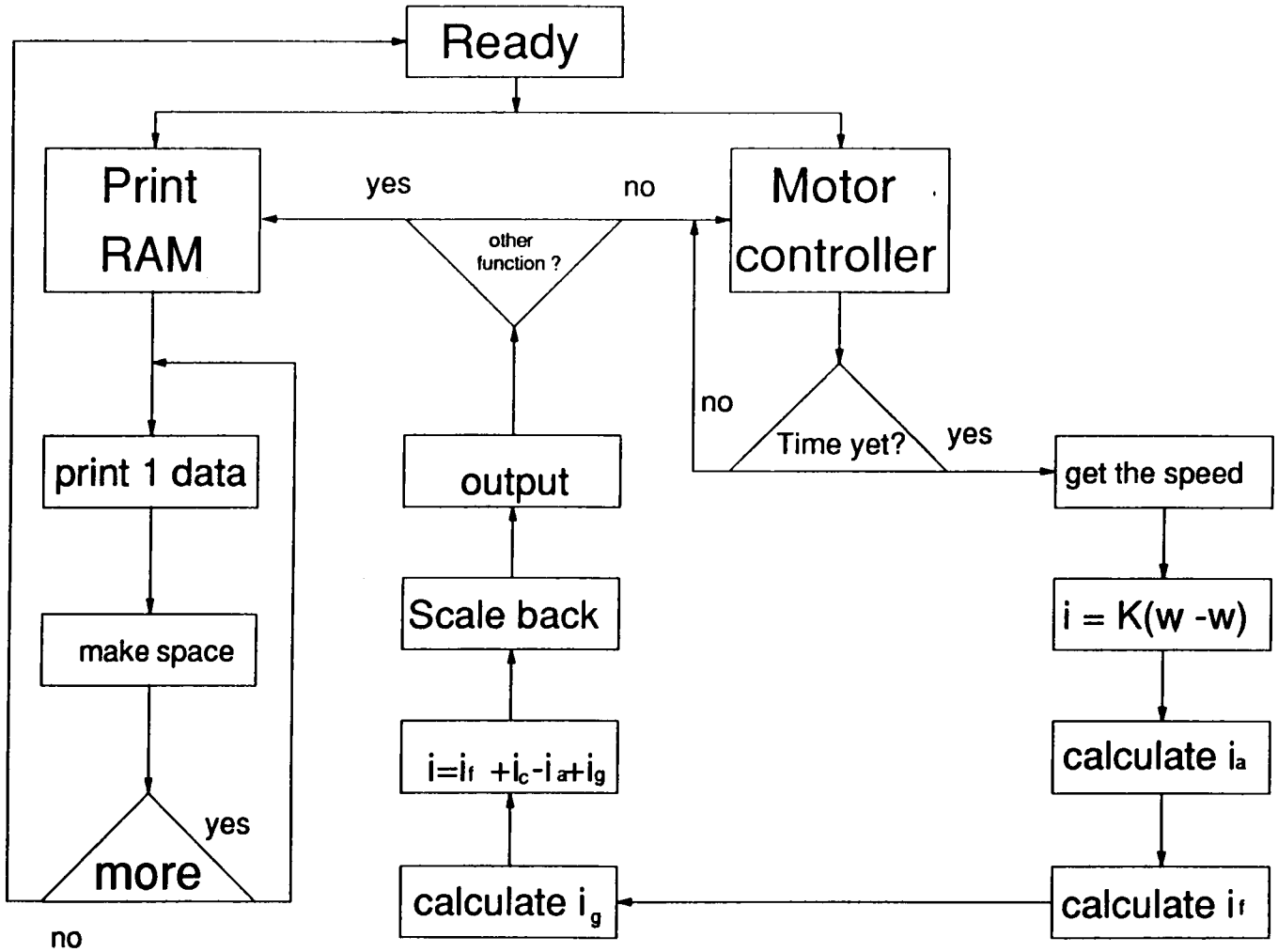


Fig. 5.9 Main controller program.

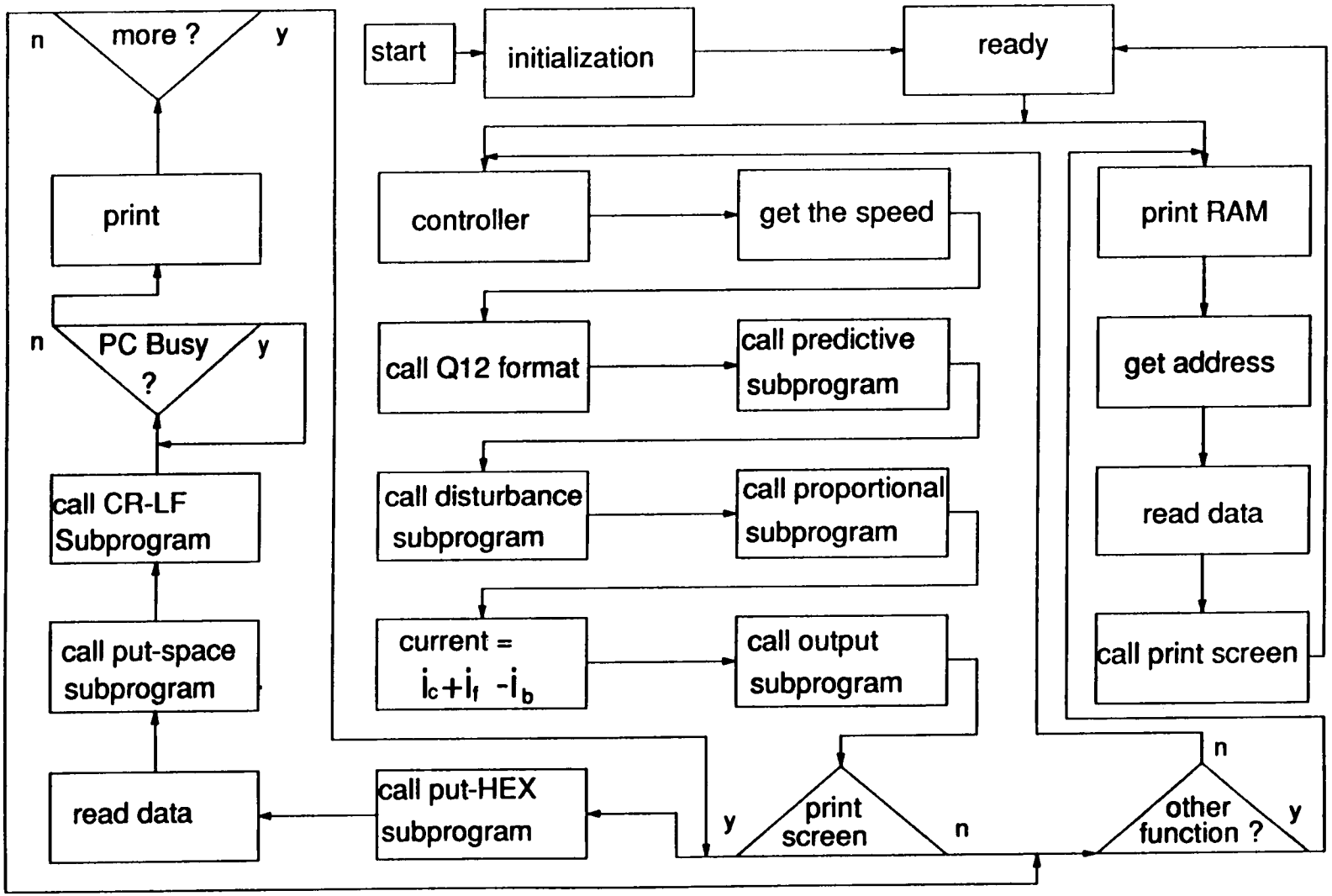


Figure 5.10 Overall control program

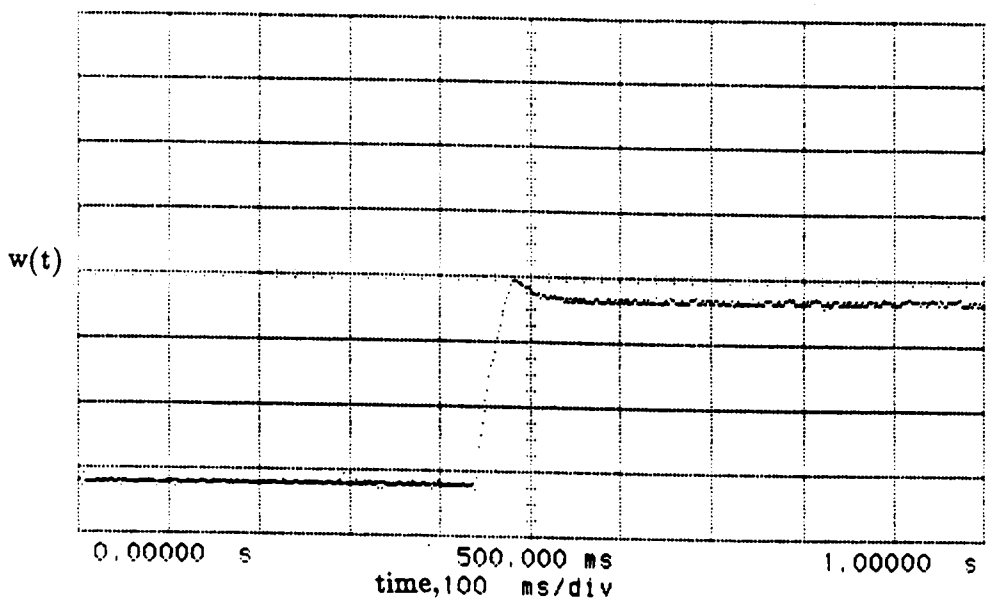


Figure 5.11 Step response for control system with disturbance observer

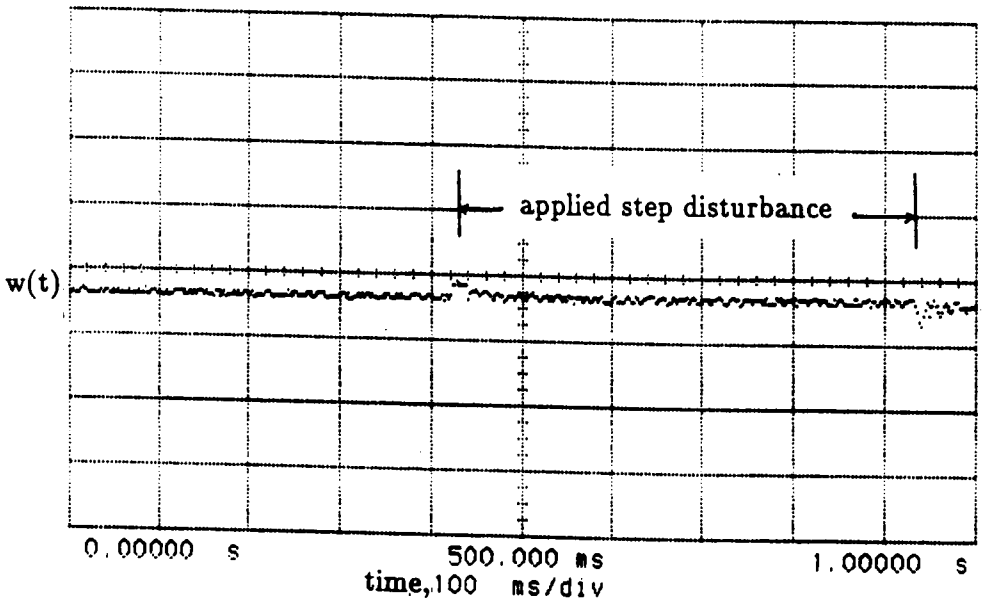


Figure 5.12 Control system with disturbance observer under disturbance

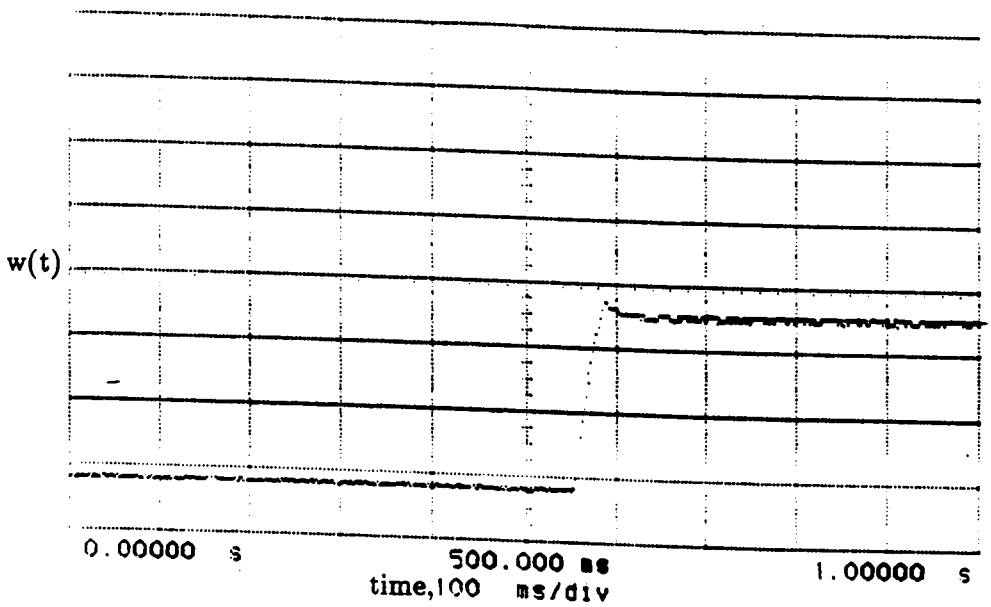


Figure 5.13: Step response for control system without disturbance observer

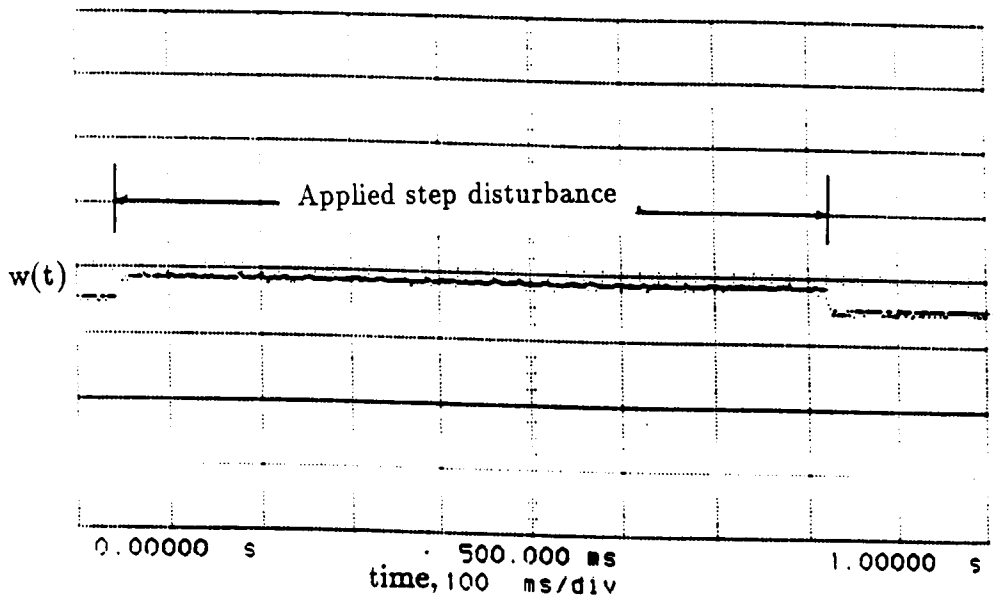


Figure 5.14. Control system without disturbance observer under disturbance

Chapter 6

Conclusion

In this thesis, the design of a digital controller has been discussed. This digital controller design included : 1) A disturbance observer, which served to counter the effect of disturbance during operation. 2) A proportional controller, and 3) A predictive controller used for fast response.

The control system was simulated on an IBM PC using SIMNON language. The simulation helped to iterate the controller parameters for achieving the best performance. The SIMNON language proved to be effective in reducing the design effort for both the continuous and the digital systems.

Of particular concern was that the output of the control system should not be affected by a system disturbance, or at least the effect of the disturbance should be minimized. Moreover, the parameters of the filters in the predictive controller and disturbance observer were iterated and reasonably chosen.

All three of the filter time constants were chosen at .001 s. Also, the affect of the low-pass filter time constant on system settling time and overshoot was discussed. An

intense discussion on how to select the low-pass filter time constant in the designed controller was presented.

Recognizing that the performance characteristics of most components is affected by their environment, the sensitivity of the system was analyzed. A system with a disturbance observer and a system without an observer were analyzed and compared in frequency response and transient response.

A complete current control model for torque constant, disturbance torque, and motor has been constructed. A current control DSP-based program has been written in TMS320C14 assembly language. Using a TMS320C14 based computer control system, hardware and software interfaces were implemented for the controller with and without the disturbance. By using the disturbance observer and predictive controller, the experimental result shows good agreement with the simulation. Experimental results also indicate that these control algorithms effectively cancel external disturbances and produce excellent speed response. The controller with a disturbance observer is superior to the controller without a disturbance observer.

References

References

[1] Krause, P. C. and Wasynczuk, o. ,"Electro mechanical Motion Devices, "*McGraw-Hill Book Company*,1989, 171 -172. Berkeley, CA, 1988.

[2] Ashworth, M. J., "Feedback Design of Systems with Significant Uncertainty," *Research Studies Press, a division of John Wiley and sons LTD.*,N.Y., NY, 1982, 8 - 9.

[3] Elmqvist, H. , Astrom, K. J. and Schonthal, T., "SIMNON, User's Guide for MS-DOS Computer," Department of Automatic Control, Lund Institute of Technology, 1986.

[4] Gupta, S. C. and Hasdorff, L., "Fundamentals of Automatic Control," Robert E. Krieger Publishing Company, Malabar, Florida , 1986, 51 - 82.

[5] Cheng, David K.,"Analysis of Linear Systems," Addison - Wesley Publishing Company, Inc., Reading, Massachusetts, 1959, 275 - 277.

[6] D'Azzo, J. J. and Houpis, C. H.,"Linear Control System Analysis and Design, Conventional and Modern," McGraw-Hill Book Company, 1988,488,504.

[7] Houpis, C. H. and Lamont, G. B., "Digital Control Systems Theory, Hardware, Software," *McGraw-Hill Book Company* 1985, 23 - 72.

[8] Sewhuk, D., Bucella, T., and Battley, T., "The Power-14 User Manual," Teknic Inc.,Rochester, N. Y. 1989, 23-26.

[9] Texas Instruments, " TMS320C14/TMS320E14 User's Guide," Texas Instruments Inc., 1988.

[10] Clayton, C. B.,"Operational Amplifiers," Butterworth Scientific,Boston, Mass., 158 - 160.

[11] Texas Instruments, "Digital Signal Processing Application with the TMS320 Family," Texas Instrument Inc., 1986.

[12] Texas Instruments, "Assemble Language Tools Guide," Texas Instruments Inc., 1988.

[13] Texas Instruments, "TMS320 Family Simulator User's Guide," Texas Instruments Inc., 1988.

[14] Texas Instruments, "First Generation TMS320C1X User's Guide," Texas Instruments Inc., 1989.

APPENDIXES

Appendix A

```
CONTINUOUS SYSTEM sta2
"
"for stability analysis
"
""""""""""""""""""""""""""""""
" 10/20/90
"
" By Nan Ren Huang
""""""""""""""""""""""""""""""
" define state
STATE w ig ia2 ift2
DER dw dig dia2 dift2
" e= error between the desired speed and actur motor speed
e=wr-w
" ic =current after the proportional control
ic=10*e
"
"for predictive controller
ift=ift1+ift2
ift1=wr/(100*Tc)
dift2=(-100*ift2+(50-(1/Tc))*wr)/(100*Tc)

"for disturbance observer
ia=ia1+ia2
ia1=w/(100*Tc)
dia2=(-100*ia2+(50-(1/Tc))*w)/(100*Tc)

"for current feedback loop
dig=(-ig+i)/Tp

"eliminate the estimated disturbance
ib=ia-ig

i=ic+ift-ib

tm=Kt*i
te=tm+td

dw=(-B*w+Te)/J
"
"input equal to step
wr:0
"
"nominal value
"
```

```
td:1 " disturbance torque
J:.02 " moment of inertia
Kt:2 " torque constant
B:1 " viscous friction coefficient
Tp:.001
Tc:.001
END
```

```
    MACRO dplot0
"
"""""""""""""""""""""""""
" 10/24/90
" Nan Ren huang
"""""""""""""""""""""""""
"a macro file for repeat plot on sta2 or sta3
"
plot w
simu 0 .01
par tp:.001
simu 0 .01
par tp:.003
simu 0 .01
par tp:.0001
simu 0 .01
par tp:.0005
simu 0 .01
END
```

```
   DISCRETE SYSTEM nan1
"Disturbance observer without eliminate the loop
INPUT wr wfeedb
OUTPUT current
STATE ig ia22 ift2
NEW nig nia22 nift2
TIME t
TSAMP ts

e=wr-wfeedb
ic=10*e
i=ic-ib+ift
ib=ia-ig

nig=i*.09516281964+ig*.90483718036

ia=ia1-ia2
ia1=10*wfeedb
ia2=ia21+ia22
ia21=.452381*wfeedb
nia22=.9047619*ia22+.8616781*wfeedb


nift2=.90476*ift2+.861666*wr
ift=ift1-ift2
ift1=9.547619*wr
current=i
ts=t+h

h:.0001
END
```

```
    CONTINUOUS SYSTEM ren1
"Plant
INPUT current
OUTPUT wfeedb
STATE w
DER dw

tm=K*current
te=td+tm

dw=-50*w+50*te

wfeedb=w

td:0
K:2
END
```

```
    CONNECTING SYSTEM huang1
"connecting system for plant and disturbance observer
TIME t
wr[nan1]=1
wfeedb[nan1]=wfeedb[ren1]
current[ren1]=current[nan1]

END
```

Appendix B

```
    .title  'Disturbance Observer Based Controller'
;-------------------------------------------------------
;
;
; name :
;       main.s
;
;
; module description:
;       This is the main module.  The startup and interrupt
;       vector code is contained in this module.
;       This application provides for a character oriented
;       user interface, interrupt driven I/O, timer
;       initialization, servo sample rate adjustments,
;       ADC polling and support to interface
;       with the Power-14 monitor for debugging using
;       hardware and software breakpoints.
;
; INCLUDES:
        .include        "c14io.inc"
        .include        "c14const.inc"
        .include        "ascii.inc"
        include         "mon.inc"
; !END!

;-------------------------------------------------------
; name
;       LocalDefs
; definitions
        .bss    servoState,1            ; Servo/ADC poll state
        .bss    curADCchannel,1         ; Current ADC channel
; end


;-------------------------------------------------------
; name
;       GlobalDefs
; definitions
        .bss    ADC_readings,8  ; Table of last ADC readings
        .bss    Position,1      ; Count up and down variable
        .bss    NoServo,1       ; Disable servo
        .bss    BackJoy,1       ; Background Joy stick enable
;                               ; !skip start!
        .def    ADC_readings
        .def    Position,NoServo,BackJoy
;                               ; !skip end!
; end
```

```
;-------------------------------------------------------------
; name
;       Startup vectors
;
;       functions\all
;       functions\startup
;       sections\Int_Vecs
;       modules\main.s
; description:
;       This code overlays the 320 vector area and
;       dispatches the appropriate
;       routine to handle the startup vector and the
;       interrupt vector.
;-------------------------------------------------------------

        .sect   "Int_Vecs"
        B       InitVector              ; Reset vector
        B       InterruptPoll           ; Interrupt vector
;end
        .text
;-------------------------------------------------------------
; name
InitVector:
;
; description:
;    This function initializes the power-source application.
;
;
; external reference
        .ref    ONE
        .ref    InitROMdata             ; Initialized RAM
                                        ; start copy in ROM
        .ref    InitedRAM               ; Initialized RAM
                                        ; start
        .ref    InitCount
        .ref    SharedBegin             ; Boundary to shared
                                        ; memory area
        .ref    BreakReg,ConfigReg      ; Ext. I/O copies
        .ref    $ExtBank                ; I/O constants
        .ref    SOtail,SOhead,SObuf     ; Serial output
                                        ; structure
        .ref    SItail,SIhead,SIbuf     ; Serial input
                                        ; structure
        .ref    TMP
;
; external procedures used
        .ref    PutStr,Cmd_Parse,servo_init
```

```
;-------------------------------------------------------
;!skip start!

DiagIOP15       .set    0       ; Set to one for IOP15 toggle

        DINT                            ; No interrupts yet
        ROVM                            ; No overflow mode
                                        ; in monitor math!
        LDPK    1                       ; Data page 1
;
; Clear data RAM for all data structures
;
        LARK    AR0,SharedBegin-1       ; Avoid shared area
        ZAC
        LARP    AR0
I000:
        SACL    *
        BANZ    I000

        LACK    1                       ; Initialize useful
                                        ; constant 1
        SACL    ONE

;
; Copy ROMed data to initialized globals
;

        LARK    AR0,InitCount-1         ; Load loop counter
                                        ; into AR0
        LARK    AR1,InitedRAM           ; Ptr to RAM globals
                                        ; to AR1
        LACK    InitROMdata             ; INITBL is located
                                        ; in first 255 words
                                        ; so short reference
                                        ; will be OK
I010:
        LARP    AR1
        TBLR    *+,AR0                  ; Get value from ROM,
                                        ; incr pntr, switch
                                        ; to cntr
        ADD     ONE                     ; Incr. ROM pointer
        BANZ    I010

        LAC     ONE,ADCintEnBit         ; Enable ADC
                                        ; interrupts
        OR      BreakReg
        SACL    BreakReg
        OUT     $ExtBank,BSR
        OUT     BreakReg,BRKport
```

```
;
; Initialize I/O ring buffers
;
        LACK    SIbuf
        SACL    SItail
        SACL    SIhead
        LACK    SObuf
        SACL    SOtail
        SACL    SOhead

        .if     DiagIOP15
        .ref    $BitBank
        OUT     $BitBank,BSR
        LAC     ONE,15                  ; ISR timer watcher
                                        ; bit
        IN      TMP,DDR
        OR      TMP
        SACL    TMP
        OUT     TMP,DDR
        .endif

;
;-------------------------------------------------------
; The beginning of application code starts here
;-------------------------------------------------------
        .text
;external procedures used
        .ref    PutStr,Cmd_Parse
;-------------------------------------------------
;for proportional control
;-------------------------------------------------
        .def    ic,wr,wfeedb,kp
;-------------------------------------------------
;for print screen
;-------------------------------------------------
        .def    count1,count1back,count2,count2back
;-------------------------------------------------
;for filter
;-------------------------------------------------
        .def    ig,ig1,ig2,igpass,currentpass,kk1,kk2
;-------------------------------------------------
;observer
;-------------------------------------------------
        .def    ia,ib,ia1,ia2,ia21,ia22,ia2pass
        .def    kk3,kk4,kk5,Truewfeedbpass
;-------------------------------------------------
;feedforward
;-------------------------------------------------
        .def    ift1,ift2,ift22,ift21,ift2pass,ift
        .def    kk6,kk7,kk8
```

```
;-------------------------------------------------
;Q12 format
;-------------------------------------------------
        .def    ANDand,Truewfeedb,ONE5
;-------------------------------------------------
;sampling rate
;-------------------------------------------------
        .def    sample_rate,sample_rateback
;-------------------------------------------------
;output
;-------------------------------------------------
        .def   current,Current1,Current2
;-------------------------------------------------
;motor speed feedback
;-------------------------------------------------
        .def    feedback1,pos_neg_chk
        .def    ONE0
        .def    OutBitMode
        .def    truepwmout
;-------------------------------------------------
;new PWM output constant (pos.)
;-------------------------------------------------
        .def    ONE34CBh,ONE2800h,ONE2000h,ONE1800h
        .def    ONE733h,ONE1000h,ONEC00h,ONE265
        .def    pwmvar1,pwmvar2,pwmvar3,ONE666h
        .def    ONE500,ONE375
;-------------------------------------------------
;new PWM output constant (neg.)
;-------------------------------------------------
        .def    M2p5,M2p,M1p5,M1p,Mp7,Mp4
        .def    ONE400,ONE385,ONE365,ONE340,ONE310
;-------------------------------------------------
;check output overflow constant
;-------------------------------------------------
        .def    M1p6,ONE1999h
        .def    ONE5000h
        .def    ONE3000h,ONEB33h,ONE1,ONE10
;-------------------------------------------------
;       Place init. table in data space
;-------------------------------------------------
        .data
InitTable
        .word   7FFFh           ; for outbitmode 6V
;Q12 format
        .word   0fffh           ; for ANDand
;reference speed
        .word   2048            ; for wr=1 scale factor=2
;proportional
        .word   4096            ;for kp
```

```
;filter
        .word   3706            ;for kk1
        .word   390             ;for kk2
;observer
        .word   4096            ;for kk3
        .word   370             ;for kk4
        .word   3706            ;for kk5
;feedforward
        .word   4096            ;for kk6
        .word   370             ;for kk7
        .word   3706            ;for kk8
;sampling rate
        .word   950             ;sampling rate
;rate to print screen
        .word   80              ;first parameter for print
                                ;screen
        .word   80              ;second parameter for print
                                ;screen
*new PWM output pos
        .word   34CBh           ;ONE34CBh
        .word   2800h           ;ONE2800h
        .word   2000h           ;ONE2000h
        .word   1800h           ;ONE1800h
        .word   1000h           ;ONE1000h
        .word   0C00h           ;ONEC00h
        .word   733h            ;ONE733h
        .word   500             ;ONE500
        .word   375             ;ONE375
        .word   265             ;ONE265

*new PWM output neg
        .word   0D800h          ;M2p5
        .word   0E000h          ;M2p
        .word   0E800h          ;M1p5
        .word   0F000h          ;M1p
        .word   0F4CDh          ;Mp7
        .word   0F99Ah          ;Mp4
        .word   400             ;ONE400
        .word   385             ;ONE385
        .word   365             ;ONE365
        .word   340             ;ONE340
        .word   310             ;ONE310

*check output overflow ?
        .word   0E666h          ;M1p6
        .word   1999h           ;ONE1999h
        .word   5000h           ;ONE5000h
        .word   666h            ;ONE666h
        .word   0D000h          ;pos_neg_chk
        .word   3000h
        .word   0B33h
```

```
UTK:    .string CR,LF,"Nan R. Huang",CR,LF,"DEC. 1990",0
        .text

;-------------------------------------------------
;       Symbol Inside Controller
;-------------------------------------------------
;----------output-----------------
current          .usect  "control",1
Current1         .usect  "control",1
Current2         .usect  "control",1
feedback1        .usect  "control",1
;-----------observer---------------
ia               .usect  "observer",1
ia22             .usect  "observer",1
ia1              .usect  "observer",1
ia2              .usect  "observer",1
ia21             .usect  "observer",1
ia2pass          .usect  "observer",1
Truewfeedbpass   .usect  "observer",1
kk3              .usect  "observer",1
kk4              .usect  "observer",1
kk5              .usect  "observer",1
ib               .usect  "observer",1
;-----------print screen--------------
count1          .usect  "prtscr",1
count1back      .usect  "prtscr",1
count2          .usect  "prtscr",1
count2back      .usect  "prtscr",1
;-----------feedforward---------------
ift              .usect  "control",1
ift1             .usect  "control",1
ift2             .usect  "control",1
ift2pass         .usect  "control",1
ift21            .usect  "control",1
ift22            .usect  "control",1
kk6              .usect  "observer",1
kk7              .usect  "observer",1
kk8              .usect  "observer",1
;------------filter--------------------
kk1              .usect  "filter",1
kk2              .usect  "filter",1
ig               .usect  "filter",1
ig1              .usect  "filter",1
ig2              .usect  "filter",1
igpass           .usect  "filter",1
currentpass      .usect  "filter",1
;-----------proportional control--------
wr      .usect  "prop",1
wfeedb  .usect  "prop",1
kp      .usect  "prop",1
ic      .usect  "prop",1
```

```
;-----------Q12 format -----------------
ANDand          .usect  "q12form",1
Truewfeedb      .usect  "q12form",1
ONE5            .usect  "q12form",1
;-----------sampling rate--------------
sample_rate     .usect  "samplin",1
sample_rateback .usect  "samplin",1
;-------------pwm temp. address---------
truepwmout      .usect  "pwmtemp",1
pwmvar1         .usect  "pwmtemp",1
pwmvar2         .usect  "pwmtemp",1
pwmvar3         .usect  "pwmtemp",1
pos_neg_chk     .usect  "pwmtemp",1
;-------------------------------------------------
;       Constant
;-------------------------------------------------
ONE0    .usect  "control",1
ONE1    .usect  "control",1
ONE2    .usect  "control",1
TEMPR   .usect  "control",1
ONE10   .usect  "control",1
OutBitMode      .usect  "control",1
;--------------new PWM output pos-----------------
ONE34CBh        .usect  "outpos",1
ONE2800h        .usect  "outpos",1
ONE2000h        .usect  "outpos",1
ONE1800h        .usect  "outpos",1
ONE1000h        .usect  "outpos",1
ONEC00h         .usect  "outpos",1
ONE733h         .usect  "outpos",1
ONE500          .usect  "outpos",1
ONE375          .usect  "outpos",1
ONE265          .usect  "outpos",1
;--------------new PWM output neg-----------------
ONE400          .usect  "outneg",1
ONE385          .usect  "outneg",1
ONE365          .usect  "outneg",1
ONE340          .usect  "outneg",1
ONE310          .usect  "outneg",1
M2p5            .usect  "outneg",1
M2p             .usect  "outneg",1
M1p5            .usect  "outneg",1
M1p             .usect  "outneg",1
Mp7             .usect  "outneg",1
Mp4             .usect  "outneg",1
;--------------output overflow ?-----------------
M1p6            .usect  "outover",1
ONE1999h        .usect  "outover",1
ONE5000h        .usect  "outover",1
ONE666h         .usect  "outover",1
```

```
ONE3000h        .usect  "outover",1
ONEB33h         .usect  "outover",1
;---------------------------------------
;       Load 0 1  5
;---------------------------------------
        LDPK    0
        LACK    0               ;ONE0=0
        SACL    ONE0
        LACK    1               ;ONE1=1
        SACL    ONE1
        LACK    2               ;ONE2=2
        SACL    ONE2
        LACK    5               ;ONE5=5
        SACL    ONE5
        LACK    5               ;ONE10=5
        SACL    ONE10
;---------------------------------------
;       Init constant from table
;---------------------------------------
        LT      ONE1
        MPYK    InitTable
        PAC
        TBLR    OutBitMode      ;OutBitMode=7FFFh
        ADD     ONE1
        SACL    TEMPR
        TBLR    ANDand          ;ANDand=0fffh
;for motoer speed reference
        LAC     TEMPR
        ADD     ONE1
        SACL    TEMPR
        TBLR    wr              ;speed command
;for proportional controller parameter
        LAC     TEMPR
        ADD     ONE1
        SACL    TEMPR
        TBLR    kp              ;proportional gain
;for low pass filter time constant parameter
        LAC     TEMPR
        ADD     ONE1
        SACL    TEMPR
        TBLR    kk1             ;1st parameter
        LAC     TEMPR
        ADD     ONE1
        SACL    TEMPR
        TBLR    kk2             ;2nd parameter
```

```
;for disturbance observer controller parameter
        LAC     TEMPR
        ADD     ONE1
        SACL    TEMPR
        TBLR    kk3             ;1st parameter
        LAC     TEMPR
        ADD     ONE1
        SACL    TEMPR
        TBLR    kk4             ;2nd parameter
        LAC     TEMPR
        ADD     ONE1
        SACL    TEMPR
        TBLR    -kk5            ;3rd parameter
;for feedforward controller parameter
        LAC     TEMPR
        ADD     ONE1
        SACL    TEMPR
        TBLR    kk6             ;1st parameter
        LAC     TEMPR
        ADD     ONE1
        SACL    TEMPR
        TBLR    kk7             ;2nd parameter
        LAC     TEMPR
        ADD     ONE1
        SACL    TEMPR
        TBLR    kk8             ;3rd parameter
;sampling rate
        LAC     TEMPR
        ADD     ONE1
        SACL    TEMPR
        TBLR    sample_rate     ;parameter for the sampling
                                ;rate
;rate to print monitor
        LAC     TEMPR
        ADD     ONE1
        SACL    TEMPR
        TBLR    count1          ;first parameter for print
                                ;screen
        LAC     TEMPR
        ADD     ONE1
        SACL    TEMPR
        TBLR    count2          ;second parameter for print
                                ;screen
*new PWM output
        LAC     TEMPR
        ADD     ONE1
        SACL    TEMPR
        TBLR    ONE34CBh        ;for constant 34cbh
        LAC     TEMPR
        ADD     ONE1
```

```
SACL    TEMPR
TBLR    ONE2800h        ;for constant 2800h
LAC     TEMPR
ADD     ONE1
SACL    TEMPR
TBLR    ONE2000h        ;for constant 2000h
LAC     TEMPR
ADD     ONE1
SACL    TEMPR
TBLR    ONE1800h        ;for constant 1800h
LAC     TEMPR
ADD     ONE1
SACL    TEMPR
TBLR    ONE1000h        ;for constant 1000h
LAC     TEMPR
ADD     ONE1
SACL    TEMPR
TBLR    ONEC00h         ;for constant c00h
LAC     TEMPR
ADD     ONE1
SACL    TEMPR
TBLR    ONE733h         ;for constant 733h
LAC     TEMPR
ADD     ONE1
SACL    TEMPR
TBLR    ONE500          ;for constant 500
LAC     TEMPR
ADD     ONE1
SACL    TEMPR
TBLR    ONE375          ;for constant 375
LAC     TEMPR
ADD     ONE1
SACL    TEMPR
TBLR    ONE265          ;for constant 265
LAC     TEMPR
ADD     ONE1
SACL    TEMPR
TBLR    ONE400          ;for constant 400
LAC     TEMPR
ADD     ONE1
SACL    TEMPR
TBLR    ONE385          ;for constant 385
LAC     TEMPR
ADD     ONE1
SACL    TEMPR
TBLR    ONE365          ;for constant 365
LAC     TEMPR
ADD     ONE1
SACL    TEMPR
TBLR    ONE340          ;for constant 340
```

```
LAC     TEMPR
ADD     ONE1
SACL    TEMPR
TBLR    ONE310          ;for constant 310
LAC     TEMPR
ADD     ONE1
SACL    TEMPR
TBLR    M2p5            ;for 2.5 in Q12
LAC     TEMPR
ADD     ONE1
SACL    TEMPR
TBLR    M2p             ;for 2 in Q12
LAC     TEMPR
ADD     ONE1
SACL    TEMPR
TBLR    M1p5            ;for 1.5 in Q12
LAC     TEMPR
ADD     ONE1
SACL    TEMPR
TBLR    M1p             ;for 1 in Q12
LAC     TEMPR
ADD     ONE1
SACL    TEMPR
TBLR    Mp7             ;for .7 in Q12
LAC     TEMPR
ADD     ONE1
SACL    TEMPR
TBLR    Mp4             ;for .4 in Q12
LAC     TEMPR
ADD     ONE1
SACL    TEMPR
TBLR    M1p6            ;for 1.6 in Q12
LAC     TEMPR
ADD     ONE1
SACL    TEMPR
TBLR    ONE1999h        ;for constant 1999h
LAC     TEMPR
ADD     ONE1
SACL    TEMPR
TBLR    ONE5000h        ;for constant 5000h
LAC     TEMPR
ADD     ONE1
SACL    TEMPR
TBLR    ONE666h         ;for constant 666h
LAC     TEMPR
ADD     ONE1
SACL    TEMPR
TBLR    pos_neg_chk     ;for constant d000h
LAC     TEMPR
ADD     ONE1
```

```
        SACL    TEMPR
        TBLR    ONE3000h        ;for constant 3000h
        LAC     TEMPR
        ADD     ONE1
        SACL    TEMPR
        TBLR    ONEB33h         ;for constant b33h
        LAC     count1          ;pass parameter
        SACL    count1back
        LAC     sample_rate     ;pass parameter
        SACL    sample_rateback
        LAC     count2          ;pass parameter
        SACL    count2back
;---------------------------------------
;       Change data page point
;---------------------------------------
        LDPK    1
;---------------------------------------
; Enable interrupt sources
;
INIT_IM
  .set    ~(_TXINT|_CAPINT0|_CAPINT1|_CAPINT2|_TIMINT2|_INT)
  LDIOB   InterruptBank,IM,INIT_IM
  M_EINT                            ; Let interrupts loose!
;
; Ready to roll...
;
        LACK    UTK                 ; short ptr to sign-on
        CALL    PutStr              ; Send sign-on string
        CALL    servo_init+1        ; Run PWM initialization
        B       Cmd_Parse           ; Run Cmd_Parse forever!!
```

```
      .text
;-------------------------------------------------------------
;main disturbance observer based control program start here
;-------------------------------------------------------------
;name of controller
;disturbance observer based controller
;
;description
;       This module calculate the output of the controller
;       by calling the subprogram.
        .def    pwm_control
;external reference
;adc_read=subroutine to read ADC.
        .ref    adc_read
;external reference
;ADC_readings=the storing address for the adc reading value.
        .ref    ADC_readings
;external reference
;ic=the current after the proportional controller.
        .ref    ic
;external reference
;Current1,Current2=temp. address for current
;current=control current
;Current output
       .ref     current,Current1,Current2

;proportional
       .ref     feedback1,pos_neg_chk
       .ref     Truewfeedb
       .ref     ONE0,ONE5,ONE
       .ref     truepwmout,ONE666h
       .ref     Truewfeedbpass
       .ref     OutBitMode
       .ref     ANDand

;external reference
;constant=ONE34CBh,ONE2800h,ONE2000h,ONE1800h,ONE1000h
;constant=ONE265,ONE500,ONE375,M2p5,M2p,M1p5,M1p,Mp7,Mp4
;constant=ONE385,ONE365,ONE340,ONE310,ONE400,ONEC00h,ONE733h
;temp. address for pwm calculation=pwmvar1,pwmvar2,pwmvar3

       .ref     ONE34CBh,ONE2800h,ONE2000h,ONE1800h,ONE1000h
       .ref     pwmvar1,pwmvar2,pwmvar3
       .ref     ONE265,ONE500,ONE375,ONEC00h,ONE733h
```

```
;new PWM output neg
        .ref    M2p5,M2p,M1p5,M1p,Mp7,Mp4
        .ref    ONE385,ONE365,ONE340,ONE310,ONE400
;voltage overflow ?
        .ref    M1p6
        .ref    ONE5000h,ONE1999h
        .ref    ONEB33h,ONE3000h,count1,count1back,ONE10,ONE1
;external reference
;sample_rate=parameter to adjust the sampling rate.
;sample_rateback=sample_rate original value.
        .ref    sample_rate,sample_rateback
pwm_control:
        .word   controlMsg
        POP
        LAC     ONE                     ; make sure no servo
        SACL    NoServo
        CALL    CRLF

DisTurb:

Delay_Time:
        LDPK    0                       ;loop to adjust
                                        ;the sampling rate.
        LAC     sample_rate
        SUB     ONE1
        SACL    sample_rate
        BGEZ    DisTurb
        LAC     sample_rateback
        SACL    sample_rate

Read_Feedback:
        LDPK    1
        LAC     ADC_readings+5          ;read in w value
        LDPK    0
        SACL    feedback1               ;feedback1=temp. register
        LAC     feedback1
        SUB     pos_neg_chk             ;use pos_neg_chk=D000 to
                                        ;check polarity
        BLEZ    Pos_Feedback
;-----------------------------------------------
;this is for neg. feedback only
;-----------------------------------------------
Neg_Feedback:
        LAC     feedback1
        ABS                              ;get the abs value
                                         ;of feedback
        CALL    settoq12
        LAC     ONE0                     ;get to the neg. Q12
                                         ;of Truewfeedb
```

```
        SUB     Truewfeedb
        SACL    Truewfeedb
        CALL    P_control
        B       Observer
;-------------------------------------------------
;this is for pos. feedback only
;-------------------------------------------------
Pos_Feedback:
        LAC     feedback1
        CALL    settoq12
        CALL    P_control

feedforward:
        CALL    Feed_Forward
Obs_Output:
        CALL    Current_back
        CALL    Dis_Obs
;-------------------------------------------------
; controller output
;-------------------------------------------------
                LAC     ic              ;i=ic+ift-ib
                ADD     ift
                SACL    current
                LAC     current
                SUB     ib
                SACL    current
;-------------------------------------------------
; scale back before output
;-------------------------------------------------
SCALEBACK:
                LT      current
                MPY     ONE2000h        ;scale back 2
                PAC
                SACH    Current1,4

                LT      Current1
                MPY     ONE2000h        ;scale back 2
                PAC
                SACH    Current1,4

                LAC     ic
                BLZ     negscaleback

                LAC     ONE1999h        ;make sure current
                SUB     Current1        ;will stay in bound
                BLZ     sethighv

                LT      Current1
                MPY     ONE5000h        ;scale back 5,
```

```
                PAC
                SACH    Current2,4      ;actual i in Q12
                B       THISOUTPUT

sethighv:        LAC     OutBitMode
                SACL    Current2
                NOP
                B       THISOUTPUT
negscaleback:
                LAC     Current1        ;if Current1 lower
                                        ;than  -1.6 V
                SUB     M1p6
                BLZ     setlowv
                LT      Current1
                MPY     ONE5000h
                PAC
                SACH    Current2,4
                B       THISOUTPUT
setlowv:
                LAC     pos_neg_chk     ;set current with -3.0V
                SACL    Current2
```

```
;-------------------------------------------------------------
; output here
;-------------------------------------------------------------

THISOUTPUT:
                LAC     Current2
                SUB     ONE34CBh            ;3.3V
                BLZ     nexta
                LAC     ONE500
                SACL    truepwmout
                B       Adjust_PWM

nexta:          LAC     Current2
                SUB     ONE2800h            ;2.5V
                BLZ     nextb
                SACL    pwmvar1
                LT      pwmvar1
                MPYK    160
                PAC
                SACH    pwmvar2,4
                LAC     pwmvar2
                ADD     ONE375
                SACL    truepwmout
                B       Adjust_PWM

nextb:          LAC     Current2
                SUB     ONE2000h            ;2.0V
                BLZ     nextc
                SACL    pwmvar1
                LT      pwmvar1
                MPYK    220
                PAC
                SACH    pwmvar2,4
                LAC     pwmvar2
                ADD     ONE265
                SACL    truepwmout
                B       Adjust_PWM

nextc:          LAC     Current2
                SUB     ONE1800h            ;1.5V
                BLZ     nextd
                SACL    pwmvar1
                LT      pwmvar1
                MPYK    180
                PAC
                SACH    pwmvar2,4
                LACK    175
```

```
                ADD     pwmvar2
                SACL    truepwmout
                B       Adjust_PWM
nextd:          LAC     Current2
                SUB     ONE1000h            ;1.0V
                BLZ     nexte
                SACL    pwmvar1
                LT      pwmvar1
                MPYK    140
                PAC
                SACH    pwmvar2,4
                LACK    105
                ADD     pwmvar2
                SACL    truepwmout
                B       Adjust_PWM
nexte:
                LAC     Current2
                SUB     ONEC00h             ;0.75V
                BLZ     nextf
                SACL    pwmvar1
                LT      pwmvar1
                MPYK    180
                PAC
                SACH    pwmvar2,4
                LACK    65
                ADD     pwmvar2
                SACL    truepwmout
                B       Adjust_PWM

nextf:          LAC     Current2
                SUB     ONE733h             ;0.45V
                BLZ     nextg
                SACL    pwmvar1
                LT      pwmvar1
                MPYK    217
                PAC
                SACH    pwmvar2,4
                LACK    ONE0
                ADD     pwmvar2
                SACL    truepwmout
                B       Adjust_PWM

nextg:          LAC     Current2
                SUB     ONE0                ;0.0V
                BLZ     nexth
                SACL    pwmvar1
                LT      pwmvar1
```

```
                MPYK    222
                PAC
                SACH    pwmvar2,4
                LACK    100
                SUB     pwmvar2
                SACL    pwmvar3
                LAC     ONE0                    ;negative
                SUB     pwmvar3
                SACL    truepwmout
                B       Adjust_PWM
nexth:          LAC     Current2
                SUB     Mp4                      ;-.4V
                BLZ     nexti
                SACL    pwmvar1
                LT      pwmvar1
                MPYK    238
                PAC
                SACH    pwmvar2,4
                LACK    195
                SUB     pwmvar2
                SACL    pwmvar3
                LAC     ONE0                    ;negative
                SUB     pwmvar3
                SACL    truepwmout
                B       Adjust_PWM

nexti:          LAC     Current2
                SUB     Mp7                      ;-.7V
                BLZ     nextj
                SACL    pwmvar1
                LT      pwmvar1
                MPYK    133
                PAC
                SACH    pwmvar2,4
                LACK    235
                SUB     pwmvar2
                SACL    pwmvar3
                LAC     ONE0                    ;negative
                SUB     pwmvar3
                SACL    truepwmout
                B       Adjust_PWM
nextj:          LAC     Current2
                SUB     M1p                      ;-1 V
                BLZ     nextk
```

```
                SACL    pwmvar1
                LT      pwmvar1
                MPYK    100
                PAC
                SACH    pwmvar2,4
                LAC     ONE265
                SUB     pwmvar2
                SACL    pwmvar3
                LAC     ONE0                    ;negative
                SUB     pwmvar3
                SACL    truepwmout
                B       Adjust_PWM
nextk:          LAC     Current2
                SUB     M1p5                     ;-1.5 V
                BLZ     next1
                SACL    pwmvar1
                LT      pwmvar1
                MPYK    90
                PAC
                SACH    pwmvar2,4
                LAC     ONE310
                SUB     pwmvar2
                SACL    pwmvar3
                LAC     ONE0                    ;negative
                SUB     pwmvar3
                SACL    truepwmout
                B       Adjust_PWM

next1:          LAC     Current2
                SUB     M2p                      ;-2.0 V
                BLZ     nextm
                SACL    pwmvar1
                LT      pwmvar1
                MPYK    60
                PAC
                SACH    pwmvar2,4
                LAC     ONE340
                SUB     pwmvar2
                SACL    pwmvar3
                LAC     ONE0                    ;negative
                SUB     pwmvar3
                SACL    truepwmout
                B       Adjust_PWM

nextm:          LAC     Current2
                SUB     M2p5                     ;-2.5 V
                BLZ     nextn
                SACL    pwmvar1
```

```
                LT      pwmvar1
                MPYK    50
                PAC
                SACH    pwmvar2,4
                LAC     ONE365
                SUB     pwmvar2
                SACL    pwmvar3
                LAC     ONE0                    ;negative
                SUB     pwmvar3
                SACL    truepwmout
                B       Adjust_PWM

nextn:          LAC     Current2
                SUB     pos_neg_chk             ;-3.0 V
                BLZ     nexto
                SACL    pwmvar1
                LT      pwmvar1
                MPYK    40
                PAC
                SACH    pwmvar2,4
                LAC     ONE385
                SUB     pwmvar2
                SACL    pwmvar3
                LAC     ONE0                    ;negative
                SUB     pwmvar3
                SACL    truepwmout
                B       Adjust_PWM

nexto:          LAC     ONE0                    ;output -3.36V
                SUB     ONE400
                SACL    truepwmout

MakeNeg:        LAC     ONE0
                SUB     truepwmout
```

```
Adjust_PWM:

        LDPK    1
        SACL    cmdAddr             ; Save for future
                                    ; use
;
; Program channel 0 value which = input value-2*Period
;
        ZAC                         ; Set to channel 0 first
        CALL    pwm_setChannel
        LAC     cmdAddr             ; Get bipolar output
        ADD     PwmPeriod,1         ;   add offset
        SACL    cmdAddr             ; Save for channel 1
        CALL    pwm_setLength
;
; Set channel 1 to 2*Period - Output
;
        LACK    1
        CALL    pwm_setChannel
        LAC     PwmPeriod,2
        SUB     cmdAddr
        CALL    pwm_setLength
        CALL    Pr_Screen
HitNextDog:
        LDPK    1
        CALL    adc_read
        SACL    adcValue
        CALL    GetChk          ; Any characters from user?
        BNZ     Cmd_Parse
        HitDog
        B       DisTurb
controlMsg:     .string  "ontrol loop ",0   ;command string
```

```
;-------------------------------------------------------------
;Q12 format
;-------------------------------------------------------------
;name
;settoq12
;
;description
;     this program change the motor speed to Q12 format.
;
        .def    settoq12
;
;external reference
;wfeedb=temp. address for the motor speed value.
;Truewfeedb= half of the motor speed in Q12 format.

        .ref    ANDand,wfeedb,Truewfeedb
settoq12:
        AND     ANDand          ;get ride of fist 4 bits
        SACL    wfeedb          ;save the row value of the
                                ;feedback
        LT      wfeedb
        MPYK    2048            ;divided by 2
        PAC
        SACH    wfeedb,4        ;save (not a Q12)
        LT      wfeedb
        MPY     ONE5
        PAC
        SACL    Truewfeedb      ;this is the true w/2 value
                                ;in Q12
        RET                     ;go back to where it coming.
```

```
;-------------------------------------------------------------
; proportional control
;-------------------------------------------------------------
        .text
;name
;P_control
;
;description
;         This module calculate the output of the
;         proportional controller. The kp is allow to
;         change.
        .def    P_control
;
;external reference parameter
; wr=reference speed
;Truewfeedb= half of the motor speed in Q12
;kp=proportional gain
;ic=current after the proportional gain
        .ref    wr,Truewfeedb,kp,ic

P_control:
        LAC     wr              ;ic=(wr-Truewfeedb)*kp
        SUB     Truewfeedb
        SACL    ic
        LT      ic
        MPY     kp
        PAC
        SACH    ic,4
        RET                     ; go back to where is
                                ; coming from.
```

```
;-------------------------------------------------------------
;Feed forward subroutine
;-------------------------------------------------------------
                .text
;name
;Feed_Forward
;
;description
;       This module calculate the output value of the
;       feedforward controller.
                .def    Feed_Forward
;external reference
;wr=motor speed reference
;ift=current after the feedforward controller.
;ift=ift1-ift2
;ift1=wr(k-1)*kk3
;ift2=ift21+ift22
;ift21=kk4*wr(k-1)
;ift22=kk5*ift2(k-1)
;ift2pass=ift2(k-1)
;kk6=1st parameter for the feedforward controller.
;kk7=2nd parameter for the feedforward controller.
;kk8=3rd parameter for the feedforward controller.

                .ref    wr,ift1,ift2,ift22,ift21,ift2pass
                .ref    kk6,kk7,kk8,ift
Feed_Forward:
                LDPK    0
                LT      wr              ;ift1=kk3*wr
                MPY     kk6             ;scale factor = 10
                PAC
                SACH    ift1,4

                LT      wr              ;ift22=kk4*wr
                MPY     kk7             ;scale factor = 10
                PAC
                SACH    ift22,4

                LT      ift2pass        ;ift21=kk5*ift2(k-1)
                MPY     kk8             ;scale factor = 10
                PAC                     ;ift2pass sacle
                                        ;factor=10
                SACH    ift21,4
```

```
LAC     ift21           ;ift2=ift21+ift22
ADD     ift22
SACL    ift2

LAC     ift1            ;ift=ift1-ift2
SUB     ift2
SACL    ift

LAC     ift2            ;pass old data
SACL    ift2pass

RET                     ;return to the orig.
```

```
;-------------------------------------------------------------
;Disturbance canceller
;-------------------------------------------------------------
;name
; disturbance canceller
;
; description:
;        This module contain two sub-module. 1st is the
;        Current_back module which is to calculate the
;        output of the filter 1. 2nd is the Dis_ob module
;        which is to calculate the output of the disturbance
;        observer.
                .text
;
;name
;Current_back
                .def    Current_back
;external reference
;ig=current after the filter.
;ig=ig1+ig2
;ig1=currentpass*(1-kk1)
;ig2=igpass*kk1 , where (1-kk1)=kk2
;kk1=1st adjustable parameter for the low pass filter time
;constant
;kk2=2nd adjustable parameter for the low pass filter time
;constant

                .ref    ig,igpass,current,currentpass
                .ref    kk1,kk2,ig1,ig2,ic,ia,ib
Current_back:
                LAC     ig                ;pass old parameter
                SACL    igpass


                LT      currentpass       ;ig1=i(k-1)*kk2
                MPY     kk2
                PAC
                SACH    ig1,4

                LT      igpass            ;ig2=ig2(k-1)*kk1
                MPY     kk1
                PAC
                SACH    ig2,4

                LAC     ig1               ;ig=ig1+ig2
                ADD     ig2
                SACL    ig
```

```
                LAC     ia              ;ib=ia-ig
                SUB     ig
                SACL    ib


                LAC     current         ;pass old parameter
                SACL    currentpass


;---------------------------------------------------------
;Disturbance_Observer subroutine
;---------------------------------------------------------
                .text
;name
;Dis_Obs
                .def    Dis_Obs
;
;external reference
;Truewfeedb=half of the motor speed
;Truewfeedbpass=old parameter=Truewfeedb(k-1)
;ia=ia1-ia2
;ia1=kk3*motor speed
;ia2=ia21+ia22
;ia21=ia2(k-1)*kk5
;ia22=w(k-1)*kk4
                .ref    Truewfeedb,ia1,Truewfeedbpass,ia22
                .ref    ia21,ia2,ia,kk3,kk4,kk5,ia2pass
Dis_Obs:
                LT      Truewfeedb      ;ia1=kk3*w
                MPY     kk3
                PAC
                SACH    ia1,4

                LT      Truewfeedbpass  ;ia22=kk4*w(k-1)
                MPY     kk4             ;scale factor=10
                PAC
                SACH    ia22,4

                LT      ia2pass         ;ia21=kk5*ia(k-1)
                MPY     kk5             ;scale factor =
                                        ;10 because of
                PAC                     ;iapass sacle factor=10
                SACH    ia21,4

                LAC     ia21            ;ia2=ia21+ia22
                ADD     ia22
                SACL    ia2
```

```
LAC     ia1             ;ia=ia1-ia2
SUB     ia2
SACL    ia


LAC     ia2             ;pass old data
SACL    ia2pass

LAC     Truewfeedb      ;pass old data
SACL    Truewfeedbpass

RET
```

```
;-------------------------------------------------------
;monitoring the variables of the controller
;-------------------------------------------------------

        .text
; name
; Pr_Screen
;
; description:
;       This module monitor the different variable at the
;       user specify time interval. To change the variable,
;       just change LAC ig or LAC current those two variable.
;
        .def    Pr_Screen
;
;external reference
;count1=first loop of the print monitor.
;count2=second loop of the print monitor.
;count1back=address where store the reset number.
;counter3back=address where store the reset number.
;ADC_readings=the row value of the motor speed is stor in
;the 5th number.
;ig=control current branch.
;ia=estimated disturbance current.
;ic=proportional current.
;ift=feedforward control current.
;ONE10,ONE1=constant.

        .ref    count1,count2,count1back,count2back,ADC_readings
        .ref    ONE10,ONE1,ig,ia,ic,ift
Pr_Screen:
        LDPK    0
        LAC     count1              ;print every count1 time
        SUB     ONE1
        SACL    count1
        BGEZ    HitNextRet
        LAC     count2              ;second loop for print screen
        SUB     ONE1
        SACL    count2

        LAC     count1back          ;set back to count1
        SACL    count1
        LAC     count2
        BGEZ    HitNextRet
        LAC     count2back          ;set back to count2
        SACL    count2
```

```
        LDPK    0
        LAC     ig                      ; print what ?
        LDPK    1
        CALL    PutHex                  ; Print it out
        CALL    PrSP                    ; Space out to next value
        LDPK    0
        LAC     current                 ; print what ?
        LDPK    1
        CALL    PutHex                  ; Print it out
        CALL    PrSP                    ; Space out to next value
        LDPK    0
        LAC     ONE10                   ;update every 10th time
        SUB     ONE1
        SACL    ONE10

        BGEZ    HitNextRet
        LACK    5                       ;CRLF every 5th time
        SACL    ONE10

        LDPK    1
        CALL    CRLF
HitNextRet:
        RET                             ;return to where is come
                                        ;from.
```

```
;-------------------------------------------------------
        .text
; !NAME!
servo_IntegralGain:
;
; !PATHS!
;       functions\all
;       functions\cmds
;       subcommand\servo
;       modules\cmds.s
; !0!
; DESCRIPTION:
;       Servo subcommand - display/set PID Integral gain
;
; SYNTAX:
;       {s}ervo {I}ntegral gain <current> [{new}]
;
 ; EXTERNAL DATA REFERENCES:
        .ref    Igain,Kintegrator
;
;-------------------------------------------------------
;!skip start!

        .word   intSmsg
        POP                                 ; Save stack space
        LAC     Igain
        CALL    PutDefault
        SACL    Igain
        ZAC
        SACL    Kintegrator
        B       Cmd_Parse

intSmsg:         .string "ntegral gain ",0

; !skip end!
; !end!
```

```
;-------------------------------------------------------------
        .text
; !NAME!
servo_ProportionalGain:
;
; !PATHS!
;       functions\all
;       functions\cmds
;       subcommand\servo
;       modules\cmds.s
; !0!
; DESCRIPTION:
;       Servo subcommand - display/set PID proportional gain
;
; SYNTAX:
;       {s}ervo {P}roportional gain <current> [{new}]
;
; EXTERNAL DATA REFERENCES:
        .ref    Pgain
;
;-------------------------------------------------------------
;!skip start!

        .word   propSmsg
        POP                             ; Save stack space
        LAC     Pgain
        CALL    PutDefault
        SACL    Pgain
        B       Cmd_Parse

propSmsg:       .string "roportional gain ",0

; !skip end!
; !end!
```

```
;-------------------------------------------------
        .text
; !NAME!
servo_DifferentialGain:
;
; !PATHS!
;       functions\all
;       functions\cmds
;       subcommand\servo
;       modules\cmds.s
; !0!
; DESCRIPTION:
;       Servo subcommand - display/set PID
;       Differential gain
;
; SYNTAX:
;       {s}ervo {D}ifferential gain <current> [{new}]
;
; EXTERNAL DATA REFERENCES:
        .ref    Dgain
;
;-------------------------------------------------------------
;!skip start!

       .word   diffSmsg
        POP                                 ; Save stack space
        LAC     Dgain
        CALL    PutDefault
        SACL    Dgain
        B       Cmd_Parse

diffSmsg:       .string "ifferential gain ",0

; !skip end!
; !end!
```

```
;-------------------------------------------------------------
        .text
; name
Cmd_PWM:
;
; description:
;       UTK command interpreter
;
;-------------------------------------------------------------

        .word   pwmMsg
        POP                     ; Conserve stack, we know
                                ; where to return
        LACK    PWM_list        ; Point at command table
        B       doCommand       ; Does not return

pwmMsg: .string "TK ",0

        .data
PWM_list:
        .word   '?',utk_help      ; Help command
        .word   'i',pwm_init      ; Initialize PWM
        .word   'p',pwm_period    ; PWM period
        .word   'l',pwm_length    ; Set PWM channel's
                                  ; pulse length
        .word   'h',pwm_halt      ; Halts PWM and coasts
                                  ; output
        .word   'm',pwm_mode      ; Sets output mode
        .word   's',pwm_set       ; Sets PWM as a signed input
        .word   'c',pwm_control   ; Sets disturbance control
                                  ; loop
        .word   'w',change_speed  ; change the motor speed
        .word   'g',change_gain   ; change the proportional
                                  ; gain
        .word   '1',change_1st_p  ; change the 1st coefficient
        .word   '2',change_2nd_p  ; change the 2nd coefficient
        .word   '3',change_3rd_p  ; change the 3rd coefficient
        .word   '4',change_4th_p  ; change the 4th coefficient
        .word   '5',change_5th_p  ; change the 5th coefficient
        .word   '6',change_6th_p  ; change the 6th coefficient
        .word   '7',change_7th_p  ; change the 7th coefficient
        .word   '8',change_8th_p  ; change the 8th coefficient
        .word   0                       ; End indicator
```

```
;-------------------------------------------------------------
        .text
; name
utk_help:
;
; description:
;       UTK subcommand - sub-command help
;
; suntax:
;       {U}TK {?} help
;
;-------------------------------------------------------------

        .word   helpMsg
        POP
        LACK    PWM_list
        B       genHelp
```

```
;-------------------------------------------------------------
        .text
; name
change_speed:
;
; description:
;       subcommand - display/set speed
;
; syntax:
;       {U}TK   {w}r_speed  <current> [{new}]
;
; external references:
        .ref    wr
;
;-------------------------------------------------------------
;!skip start!

        .word   speedmsg
        POP                             ; Save stack space
        LDPK    0
        LAC     wr
        LDPK    1
        CALL    PutDefault
        LDPK    0
        SACL    wr
        LDPK    1
        B       Cmd_Parse

speedmsg:       .string "r reference ",0
```

```
;-------------------------------------------------------
        .text
; name
change_gain:
;
; description:
;       subcommand - display/set proporational gain
;
; syntax:
;       {U}TK   {g}ain   <current> [{new}]
;
; EXTERNAL DATA REFERENCES:
        .ref    kp
;
;-------------------------------------------------------
;!skip start!

        .word   gainmsg
        POP                             ; Save stack space
        LDPK    0
        LAC     kp
        LDPK    1
        CALL    PutDefault
        LDPK    0
        SACL    kp
        LDPK    1
        B       Cmd_Parse

gainmsg:        .string "ain",0
```

```
;-------------------------------------------------------------
        .text
; name
change_1st_p:
;
; description:
;       subcommand - display/set 1st parameter
;
; syntax:
;       {U}TK   {1}st_parameter  <current> [{new}]
;
; external references:
        .ref    kk1
;
;-------------------------------------------------------------
;!skip start!

        .word   firstmsg
        POP                                 ; Save stack space
        LDPK    0
        LAC     kk1
        LDPK    1
        CALL    PutDefault
        LDPK    0
        SACL    kk1
        LDPK    1
        B       Cmd_Parse

firstmsg:       .string "st parameter",0
```

```
;-------------------------------------------------------------
        .text
; name
change_2nd_p:
;
; description:
;       subcommand - display/set 2nd parameter
;
; syntax:
;       {U}TK   {2}nd_parameter   <current> [{new}]
;
; external references:
        .ref    kk2
;
;-------------------------------------------------------------
;!skip start!

        .word   secondmsg
        POP                                 ; Save stack space
        LDPK    0
        LAC     kk2
        LDPK    1
        CALL    PutDefault
        LDPK    0
        SACL    kk2
        LDPK    1
        B       Cmd_Parse

secondmsg:       .string "nd parameter",0
```

```
;-------------------------------------------------------------
        .text
; name
change_3rd_p:
;
; description:
;       subcommand - display/set 3rd paramter
;
; syntax:
;       {U}TK   {3}rd_parameter  <current> [{new}]
;
; external references:
        .ref  - kk3
;
;-------------------------------------------------------------
;!skip start!

        .word   thirdmsg
        POP                                 ; Save stack space
        LDPK    0
        LAC     kk3
        LDPK    1
        CALL    PutDefault
        LDPK    0
        SACL    kk3
        LDPK    1
        B       Cmd_Parse

thirdmsg:       .string "rd parameter",0
```

```
;-------------------------------------------------------------
        .text
; name
change_4th_p:
;
; description:
;       subcommand - display/set 4th parameter
;
; syntax:
;       {U}TK   {4}th_parameter  <current> [{new}]
;
; external references:
        .ref  · kk4
;
;-------------------------------------------------------------
;!skip start!

        .word   fourthmsg
        POP                                 ; Save stack space
        LDPK    0
        LAC     kk4
        LDPK    1
        CALL    PutDefault
        LDPK    0
        SACL    kk4
        LDPK    1
        B       Cmd_Parse

fourthmsg:       .string "th parameter",0
```

```
;---------------------------------------------------------
        .text
; name
change_5th_p:
;
; description:
;       subcommand - display/set 5th parameter
;
; syntax:
;       {U}TK   {5}th_parameter  <current> [{new}]
;
; external references:
        .ref    kk5
;
;---------------------------------------------------------
;!skip start!

        .word   fifthmsg
        POP                                 ; Save stack space
        LDPK    0
        LAC     kk5
        LDPK    1
        CALL    PutDefault
        LDPK    0
        SACL    kk5
        LDPK    1
        B       Cmd_Parse

fifthmsg:       .string "th parameter",0
```

```
;-------------------------------------------------------------
        .text
;  name
change_6th_p:
;
; description:
;       subcommand - display/set 6th parameter
;
; syntax:
;       {U}TK   {6}th_parameter  <current> [{new}]
;
; external references:
        .ref  ' kk6
;
;-------------------------------------------------------------
;!skip start!

        .word   sixthmsg
        POP                                 ; Save stack space
        LDPK    0
        LAC     kk6
        LDPK    1
        CALL    PutDefault
        LDPK    0
        SACL    kk6
        LDPK    1
        B       Cmd_Parse

sixthmsg:       .string "th parameter",0
```

```
;-------------------------------------------------------
        .text
; name
change_7th_p:
;
; description:
;       subcommand - display/set 7th parameter
;
; syntax:
;       {U}TK   {7}th_parameter  <current> [{new}]
;
; external references:
        .ref    kk7
;
;-------------------------------------------------------
;!skip start!

        .word   seventhmsg
        POP                                 ; Save stack space
        LDPK    0
        LAC     kk7
        LDPK    1
        CALL    PutDefault
        LDPK    0
        SACL    kk7
        LDPK    1
        B       Cmd_Parse

seventhmsg:       .string "th parameter",0
```

```
;-------------------------------------------------------
        .text
; name
change_8th_p:
;
; description:
;       subcommand - display/set 8th parameter
;
; syntax:
;       {U}TK   {8}th_parameter  <current> [{new}]
;
; external references:
        .ref  - kk8
;
;-------------------------------------------------------
;!skip start!

        .word   eigthmsg
        POP                                 ; Save stack space
        LDPK    0
        LAC     kk8
        LDPK    1
        CALL    PutDefault
        LDPK    0
        SACL    kk8
        LDPK    1
        B       Cmd_Parse

eigthmsg:       .string "th parameter",0
```

```
;-------------------------------------------------------
; print RAM
;-------------------------------------------------------
        .text
; Name
Cmd_RAM:
;
;description:
;  Print values in data memory in hexadecimal notation.
;  User can change the no_of_line to different #
;  User can also change the Length to different #
; external reference
        .ref    ONE
;
; external routines reference:
        .ref    CRLF,PutHex,PrSP
;
;*************************************************************
;!skip start!

Length          .set    50  ; Default # to dsp for dump cmds
no_of_line      .set    4   ; #/line for dump cmds

       .word   ramMsg
       POP                 ;Conserve stack,
       LACK    Length      ; Set default length
       CALL    CmdSetup    ; Get address and length
dR10:
        LACK    no_of_line  ;formatting loop counter set
        SACL    cmdCntr
        CALL    CRLF
        LAC     cmdAddr     ; Print address
        CALL    PutHex
        CALL    AP2         ; Formatting spaces
dR20:
        LAR     AR0,cmdAddr
        LARP    AR0
        LAC     *
        CALL    PutHex      ; Print it out
        CALL    PrSP        ; Space out to next value
        LAC     cmdAddr     ; Update memory pointer
        ADD     ONE
        SACL    cmdAddr
        LACK    0FFh
        AND     cmdAddr
        SACL    cmdAddr
        LAC     cmdLen      ; Update number left
        SUB     ONE
        SACL    cmdLen
```

```
        BZ      Cmd_Parse   ; all done here?
        LAC     cmdCntr     ; Printed a lines worth?
        SUB     ONE
        SACL    cmdCntr
        BNZ     dR20        ; 0=end of line
        B       dR10        ; new lines worth

ramMsg: .string "AM print ",0
```

VITA

Nan Ren Huang was born in Tainan, Taiwan on April 11, 1956. In August, 1989 he received the Bachelor of Science degree in Electrical Engineering from Lamar University, Beaumont, Texas. He entered The Graduate School of The University of Tennessee, Knoxville in September, 1989. During the time, he worked as a graduate research assistant. He received the degree of Master of Science, with a major in Electrical Engineering in May 1991. He is an Engineer-In-Training. He is also certified as a class A engineer by Ministry of Transportation of Taiwan, R. O. C.. He is a student member of The Institute of Electrical and Electronics Engineers.