STABILIZATION OF AN ARBITRARY ORDER TRANSFER FUNCTION WITH TIME DELAY USING PI, PD AND PID CONTROLLERS

A Thesis by

Sead Sujoldžić

B. S. Electrical Engineering, Wichita State University, 2001

Submitted to the College of Engineering
and the faculty of the Graduate School of
Wichita State University
in partial fulfillment of
the requirements for the degree of
Master of Science

December 2005

STABILIZATION OF AN ARBITRARY ORDER TRANSFER FUNCTION WITH TIME DELAY USING PI, PD AND PID CONTROLLERS

I have examined the final copy of this Thesis for form and content and recommend that it be accepted in partial fulfillment of the requirement for the degree of Master of Science with a major in Electrical Engineering

_____
John M. Watkins, Committee Chair

We have read this Thesis
And recommend its acceptance:

_____
M. Edwin Sawan, Committee Member

_____
Christian Wolf, Committee Member

DEDICATION

To my family, Hivzija, Jasmina and Selena Sujoldžić, and relatives, especially the Memić family, for their never-ending care and attention, to Branimir Gajić and Đorđe Stojšić (in no particular order) for being true friends, to Kuzmanović, Gubić and Vujaklija families for treating me like a family member and to Danelle M. Phillips, whose persistence to succeed in life in spite of difficult circumstances provided me with a constant inspiration throughout the course of writing the thesis

Eppur si muove

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. John Watkins, for providing guidance, support and motivation. Also, I would like to thank Dr. Edwin Sawan for recommending Dr. Watkins as my advisor and for never being to busy to answer questions. Thanks are also due to Dr. Christian Wolf for believing in my ability to pursue mathematics and teaching me that a mathematical proof is either correct or incorrect. I would also like to express my gratitude to Dr. Kenneth Miller for giving me the opportunity to teach mathematics and for providing financial assistance for my graduate studies. In addition, I would like to thank Dr. Phillip Parker for his strict requirements and rigorous approach to mathematics, which made me understand engineering so much more. Finally, I would like to express my appreciation to one of the most significant professor in my life, Dr. Stephen Brady, with whom I have had the pleasure of taking four classes at the beginning of my college career and whose teaching methods have inspired me since the first day I entered a classroom as an instructor.

ABSTRACT

This thesis is concerned with developing a procedure for stabilizing a linear time-invariant plant of an arbitrary order with time delay utilizing proportional-integral (PI), proportional-derivative (PD) and proportional-integral-derivative (PID) controllers. The method presented here is based on computing the stability boundary in terms of the proportional (K_p) and integral gain (K_i) for the PI case, and similarly, proportional and derivative gain (K_d) for the PD case. The two variables are then plotted on the same coordinate system, thus obtaining the stability region for each controller used. For the PID case, the stability bounds are derived by observing the three planes (K_p, K_i), (K_p, K_d) and (K_i, K_d).

The advantage of this procedure is the fact that it does not require the knowledge of the plant transfer function parameters, but only its frequency response. If the plant function is known, the procedure may also be used to analytically obtain the stabilizing controllers. We also present the tuning rules for user specified gain and phase margins.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION

1.1 PID Controller

More than 60 years after the introduction of proportional-integral-derivative (PID) controllers, they remain the standard component of industrial process control. They have been used to control temperature, pressure, flow rate, chemical composition, force, speed (vehicle cruise control) and a number of other variables. Let us now consider the diagram in Fig. 1 to see how an actual PID controller is implemented.

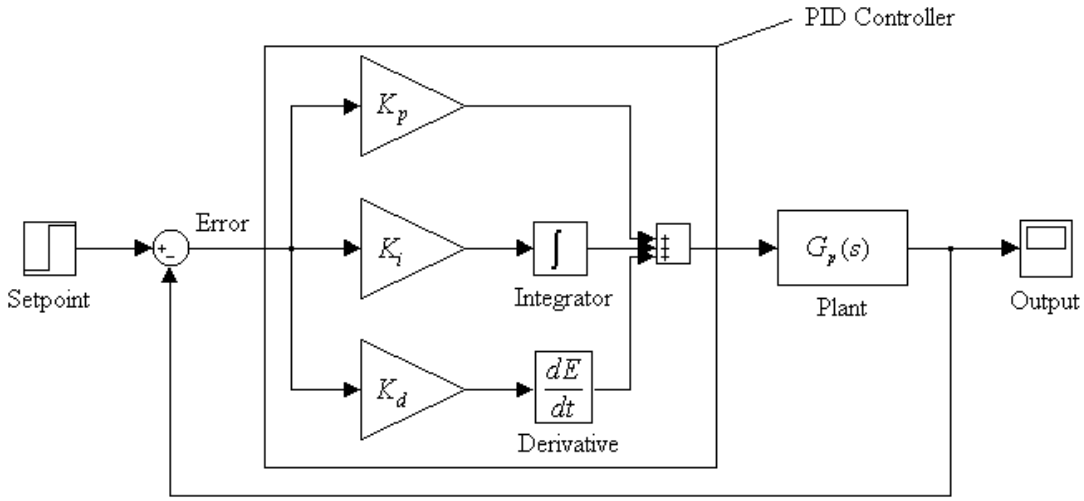

Fig. 1. A basic negative feedback control system with a PID element.

The controller measures an output of a process (plant) and controls an input, with a goal of maintaining the output at a desired value called the setpoint. Then it subtracts the measurement from the setpoint to determine an error. The error is then processed in three different ways simultaneously [13]:

(1) To handle the present, the error is multiplied by a proportional constant K_p.

(2) To handle the past, the error is integrated over a period of time, and then multiplied by a constant K_i.

(3) To handle the future, the first derivative of the error is calculated with respect to time, and multiplied by another constant K_d.

The sum of the above is added as a final output of the PID loop. Mathematically, the controller operation can be described in the time domain as

$$C_0(t) = K_p\left[E(t) + \frac{1}{K_i}\cdot\left(\int E(t)dt\right) + K_d\cdot\left(\frac{d}{dt}E(t)\right)\right], \tag{1.5.1}$$

where

$$E(t) = SP(t) - PV(t) \tag{1.5.2}$$

is the error signal associated with the setpoint $SP(t)$ and the process variable $PV(t)$ [8]. The idea is to eliminate the error by increasing the output if the error is positive or by decreasing it if the error is negative.

The most difficult part of controller design is to determine out just how much of a corrective effort the controller should apply to the process in each case. A proportional controller simply multiplies the error by a constant to compute its next output. Unfortunately, a proportional controller tends to quit working once the process variable approaches close enough to the setpoint. It will settle on a fixed output that leaves a small steady-state error.

1.2 Integral Operation

Soon after the limitation of the proportional controller was realized, engineers discovered that the steady-state error could be avoided by automatically resetting the

setpoint to an artificially high value [9]. The idea was to let the proportional controller pursue the artificial setpoint so that the actual error would be zero by the time the controller quit working. This was achieved by slowly raising (or lowering) the artificial setpoint as long as the actual error remained nonzero. As it happens, the automatic reset operation turned out to be mathematically equivalent to integrating the error and adding that total to the output of the controller's proportional term. The result is a proportional-integral (PI) controller that will continue to generate an ever-increasing output until the error has been eliminated.

Unfortunately, integral action does not guarantee perfect feedback control. A PI controller can cause closed-loop instability if the integral action is too aggressive. The controller may over-correct for an error and create a new one of even greater magnitude in the opposite direction. When that happens, the controller will eventually start driving the output back and forth between fully on and fully off, a phenomenon known as hunting [16].

1.3 Derivative Operation

Adding derivative action to the configuration can sometimes prevent hunting. The derivative term in a full PID controller is active only when the error is changing. And if the setpoint happens to be constant, the error changes only when the process variable starts moving away from or towards the setpoint. This is particularly useful if the controller's previous attempts have caused the output to approach the setpoint too rapidly. The deceleration provided by the derivative reduces the possibility of overshoot and hunting.

However, if the derivative acts too aggressively, it causes hunting all by itself. This effect is particularly pronounced in processes that react quickly to the controller's efforts, such as motors and robots.

Derivative action also tends to add a dramatic spike to the controller's output in the case of an abrupt change in the error due to a new setpoint. This forces the controller to start taking corrective action immediately without waiting for the integral or proportional action to take effect. Compared to a two-term PI or proportional-derivative (PD) controller, a full PID controller can even appear to anticipate the level of effort that will ultimately be required to maintain the process variable at the new setpoint.

1.4 Derivative Drawbacks

Dramatic swings in the control effort can be troublesome in applications (such as room temperature control) that require slow and steady changes in the controller's output. A blast of hot air following every adjustment to the thermostat would not only be uncomfortable for the occupants of the room but hard on the furnace as well.

For such applications it is advantageous to abandon derivative altogether or calculate the derivative alternatively. Derivative action is also a problem for applications that involve noisy measurements. The derivative term will contribute to the controller's output every time the process variable appears to change. The controller could end up taking corrective actions even if the actual process variable has already reached the setpoint. Most of the modern controllers include a filter to provide a much smoother input to the derivative term.

1.5 Time-Delay Systems

In the mathematical description of a physical or biological process, it is quite common to assume that the future behavior of the process depends only on the present state, and therefore can be described by a finite set of ordinary differential equations. This is satisfactory for a large class of practical systems. However, such description does not account for important behaviors of many systems that include time delays. Indeed, due largely to the current lack of effective methodology for analysis and control design for such systems, the time-delay elements are often either neglected or poorly estimated, which frequently results in analysis and simulation of insufficient accuracy, which in turn leads to unsatisfactory performance of the system designed [14]. Indeed, it has been demonstrated in the area of automatic control that a relatively small delay may lead to instability or significantly deteriorated performances for the corresponding closed-loop system.

In order to reliably analyze and design feedback control for such systems, it is necessary to consider the fact that the system's future behaviors depend not only on the current value of the state variables, but also some past history of the state variables. These systems are called time-delay systems. Examples of the systems that cause time delays include: the measurement of system variables (engineering process), the physical nature of some system's components (hydroelectric power systems; biological and ecological systems, such as population dynamics) or signal transmissions (power and communications systems).

To effectively deal with time-delay systems, the engineers in the control fields are faced with three issues: 1) How to mathematically describe such systems in a form which is convenient for analysis and design? 2) How to analyze a given system to extract some fundamental properties? And 3) How to design a PID (PD, PI) controller to achieve stability and satisfy desired performance requirements?

1.6 Previous Work

There are several tuning techniques used today: Ziegler-Nichols, frequency domain tuning, relay based tuning, tuning using optimization, internal model control tuning and other methods [11].

Traditionally, the controllers have been tuned experimentally by the method of Ziegler and Nichols [6]. This method called "continuous cycling method' has been known as a fairly accurate procedure to determine good settings of PID controllers for a wide range of common industrial processes. The controller parameters were obtained in the following way: set $K_i = K_d = 0$ and increase K_p until a periodic oscillation occurs in the output. The value of K_p at this point is called the ultimate gain (K_u) and the oscillation period is called the ultimate period (T_u). The final step is to compute the controller parameters based on the following three formulas:

$$K_p = 0.6K_u$$

$$K_i = \frac{1.2K_u}{T_u}$$

$$K_d = 0.075K_uT_u. \qquad (1.5.3)$$

However, very little emphasis is given to measurement noise, sensitivity to process variation and setpoint response that can cause the closed-loop system to be poorly damped and to have poor stability margins.

Without bringing the system close to instability a new experimental method was introduced by Aström and Hagglund using a relay to generate an oscillation for measuring the ultimate gain and ultimate period [7]. This was done by employing the configuration shown in Fig. 2, where the relay is adjusted to induce a self-sustaining oscillation in the loop [12].

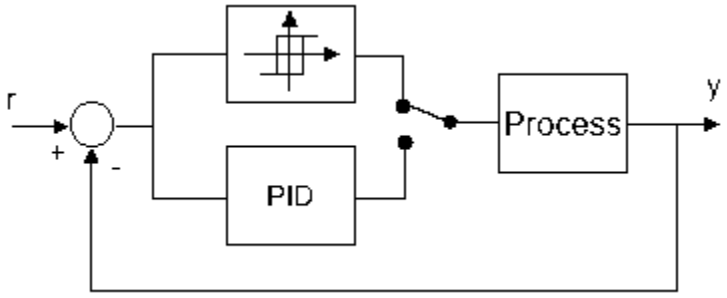

Fig. 2. Block diagram of relay feedback.

In earlier work based on the Buckingham's pi-theorem, the application of dimensional analysis in tuning of PI controllers for first order systems with delay was introduced [4]. This analytical method was based on defining K_p and K_i gains in terms of the three parameters of the plant function: plant gain, time delay and the coefficient of the first order term in the denominator. The goal was to then to apply the pi-theorem to simplify these functions. The proposed method proved to be advantageous over Ziegler-Nichols method.

In [2], state-feedback and state-feedback with integral feedback controllers were

used to control double integrator systems with time delay. The authors were able to find a relationship between the design requirements and the following parameters: gain crossover-frequency and phase margin, and also, phase crossover-frequency and gain margin.

In [3], an extension of the Hermite-Biehler theorem for quasi-polynomials was used to derive all stabilizing PID controllers. Datta et al. first described the closed-loop characteristic equation as a quasi-polynomial

$$\delta^*(j\omega) = \delta_r(\omega) + j\delta_i(\omega), \tag{1.5.4}$$

where $\delta_r(\omega)$ and $\delta_i(\omega)$ represent, respectively, the real and imaginary parts of $\delta^*(j\omega)$. Then, due to Pontryagin's results, they were able to find the necessary and sufficient condition that defines the range of K_p such that zeros of $\delta_r(\omega)$ and $\delta_i(\omega)$ are simple and real. Through further investigation they also showed that for a fixed value of K_p the region defining the stabilizing parameters K_d and K_d is a convex polygon. Similar results were also used to obtain the two-dimensional stabilizing regions in case of PI and PD controllers.

The method introduced in [1] considers decomposing the numerator and denominator of the plant transfer function into their even and odd parts and then obtaining solutions for the stabilizing parameters in terms of such decomposition. Since the solutions are functions of frequency, the author also considers the calculation of the frequency range over which the parameters need to be evaluated.

All the analytical methods in this section are limited to the first or second order plants and assume that the plant parameters are known. A new procedure is required that can be applied to a wider class of systems.

1.7 Proposed Method

We propose a new solution to the problem of stabilizing an arbitrary order transfer function with time delay through application of PI, PD and PID controllers. This method involves decomposing the plant frequency response into real and imaginary parts and computing the stabilizing parameters for each controller based exclusively on such a decomposition. Compared to the results obtained within the last few years, no complex mathematical derivations are required to find all stabilizing PID (PI, PD) controllers, but only the numerical frequency response of the plant transfer function. If the plant parameters are given, we can apply the same procedure to provide a complete analytical solution of all stabilizing PI, PD and PID gain values. We can also find all controllers that achieve a desired gain and phase margin.

CHAPTER II

STABILIZATION WITH A PI CONTROLLER

2.1 Stability Bounds

Consider the single-input single-output (SISO) system in Fig. 3.

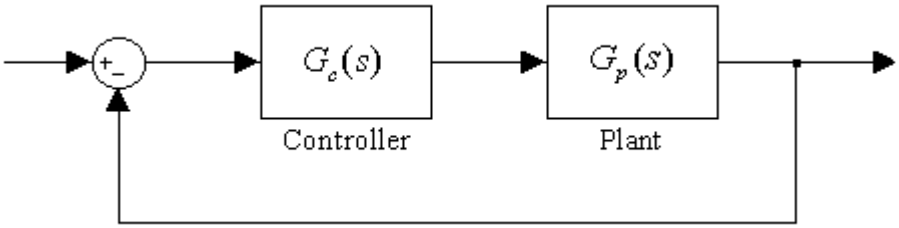

Fig. 3. A basic control system with unity feedback.

The stabilizing techniques used in this paper are based on characterizing the plant to be controlled by

$$G_p(s) = G(s)e^{-\tau s}, \tag{2.1.1}$$

where τ is the time delay. The controller being used is of the PI type, so $G_c(s)$ is given by

$$G_c(s) = K_p + \frac{K_i}{s} = \frac{K_p s + K_i}{s}. \tag{2.1.2}$$

The problem is to determine the values of K_p and K_i for which the closed–loop characteristic polynomial $\Delta(s)$ of the system in Fig. 4 is Hurwitz stable.

The characteristic equation can be written as

$$\Delta(s) = 1 + G_c(s)G_p(s). \tag{2.1.3}$$

Letting $s = j\omega$ and decomposing $G_p(j\omega)$ as

$$G_p(j\omega) = R_p(\omega) + jI_p(\omega), \tag{2.1.4}$$

results in

$$\Delta(j\omega) = 1 + \left(\frac{K_p j\omega + K_i}{j\omega} \right) \left(R_p(\omega) + jI_p(\omega) \right)$$

$$= 1 + \left(K_p - \frac{K_i j}{\omega} \right) \left(R_p(\omega) + jI_p(\omega) \right). \tag{2.1.5}$$

Expanding $\Delta(j\omega)$ and setting it to zero produces

$$R_\Delta(\omega) + I_\Delta(\omega) = 0, \tag{2.1.6}$$

where

$$R_\Delta(\omega) = 1 + K_p R_p(\omega) + \frac{K_i I_p(\omega)}{\omega} \tag{2.1.7}$$

and

$$I_\Delta(\omega) = K_p I_p(\omega) - \frac{K_i R_p(\omega)}{\omega}. \tag{2.1.8}$$

Setting the real part equal to zero gives

$$K_p R_p(\omega) + K_i \frac{I_p(\omega)}{\omega} = -1, \tag{2.1.9}$$

or equivalently,

$$K_p \left(\omega R_p(\omega) \right) + K_i \left(I_p(\omega) \right) = -\omega. \tag{2.1.10}$$

For the imaginary part, we get

$$K_p \left(\omega I_p(\omega) \right) + K_i \left(-R_p(\omega) \right) = 0. \tag{2.1.11}$$

Hence, to obtain the values of K_p and K_i which define the stability boundary, we need to solve the following system

$$\begin{bmatrix} \omega R_p(\omega) & I_p(\omega) \\ \omega I_p(\omega) & -R_p(\omega) \end{bmatrix} \begin{bmatrix} K_p \\ K_i \end{bmatrix} = \begin{bmatrix} -\omega \\ 0 \end{bmatrix} \tag{2.1.12}$$

Solving (2.1.12) for $\omega \neq 0$, we obtain

$$K_p(\omega) = -\frac{R_p(\omega)}{\left|G_p(j\omega)\right|^2} \tag{2.1.13}$$

and

$$K_i(\omega) = -\frac{\omega I_p(\omega)}{\left|G_p(j\omega)\right|^2}, \tag{2.1.14}$$

where

$$\left|G_p(j\omega)\right|^2 = \left(I_p(\omega)\right)^2 + \left(R_p(\omega)\right)^2. \tag{2.1.15}$$

At $\omega = 0$, (2.1.12) is given by

$$\begin{bmatrix} 0 & I_p(0) \\ 0 & -R_p(0) \end{bmatrix} \begin{bmatrix} K_p \\ K_i \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \tag{2.1.16}$$

Solving (2.1.16), we find that K_p is arbitrary and $K_i = 0$, unless $I_p(0) = R_p(0) = 0$. This condition holds only when $G_p(s)$ has a zero at the origin. In such case, a PI compensator should be avoided as the zero at the origin cancels the PI

pole at the origin and the system becomes internally unstable. Thus $K_i = 0$ provides an additional stability bound.

Equations (2.1.13) and (2.1.14) can be used to solve for K_p and K_i either analytically or numerically. Note that if we have the frequency response of $G_p(s)$, we can easily obtain the values of K_p and K_i by extracting the real and imaginary parts of $G_p(s)$ at each frequency and substituting them into the corresponding equations. This will be illustrated in the examples that follow.

We also see that the two parameters depend on $\omega \in [0, \infty)$. However, since the PI controllers cannot add phase to the system, we only need to consider the closed interval $[0, \omega_c]$. To find the general solution for ω_c, we use (2.1.4) and the fact that at $\omega = \omega_c$ the phase of $G_p(j\omega)$ is equal to $-\pi$, or equivalently,

$$\tan^{-1}\left(\frac{I_p(\omega_c)}{R_p(\omega_c)}\right) = -\pi. \tag{2.1.17}$$

Hence, $I_p(\omega_c) = 0$ and $R_p(\omega_c) < 0$.

2.2 Bounds for Desired Gain and Phase Margins

It is sometimes required to design a controller that will satisfy specific gain and phase margins. That can be achieved by inserting a test function $G_{gp} = Ae^{-j\theta}$ in the feed forward path as shown Fig. 4 [1].

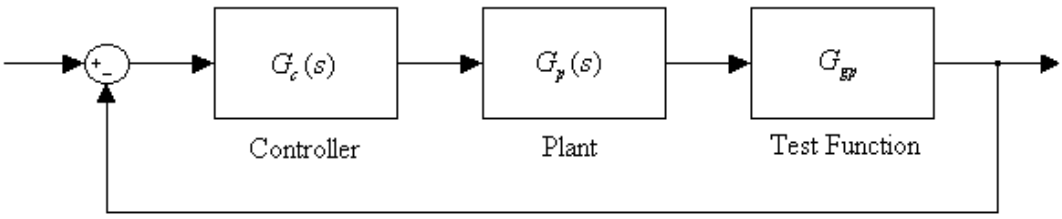

Fig. 4. A basic control system with a test function.

Then, $G_p(j\omega)$ becomes

$$G_p(j\omega) = G(j\omega)e^{-j\omega\tau}Ae^{-j\theta}$$

$$= \tilde{G}(j\omega)e^{-j\tilde{\tau}}, \tag{2.2.1}$$

where

$$\tilde{G}(j\omega) = AG(j\omega) \tag{2.2.2}$$

and

$$\tilde{\tau} = \omega\tau + \theta. \tag{2.2.3}$$

To design the controller that satisfies the gain margin condition, we first set A to the required gain margin and $\theta = 0$ in (2.2.1). We then compute the corresponding set of values for K_p and K_i, by decomposing (2.2.1) into real and imaginary parts and substituting them into (2.1.13) and (2.1.14). To satisfy the criterion for the phase margin, we repeat the procedure, but with θ set to the required phase and $A = 1$. Next, we plot the two results on the same coordinate system. The solution is the region where the two graphs intersect.

CHAPTER III

STABILIZATION WITH A PD CONTROLLER

3.1 Stability Bounds

Now we consider the system in Fig. 3 with the same plant function as before, but with a PD controller represented as

$$G_c(s) = K_p + K_d s. \tag{3.1.1}$$

We can apply the same procedure as in the case of a PI controller to find the stabilizing values of K_p and K_d. The characteristic equation in terms of $j\omega$ is

$$\Delta(j\omega) = 1 + \left(K_p + (K_d\omega)j\right)\left(R_p(\omega) + jI_p(\omega)\right). \tag{3.1.2}$$

Expanding $\Delta(j\omega)$ and setting it to zero produces

$$R_\Delta(\omega) + I_\Delta(\omega) = 0, \tag{3.1.3}$$

where

$$R_\Delta(\omega) = 1 + K_p R_p(\omega) - K_d \omega I_p(\omega) \tag{3.1.4}$$

and

$$I_\Delta(\omega) = K_p I_p(\omega) + K_d \omega R_p(\omega). \tag{3.1.5}$$

Setting the real and imaginary parts equal to zero gives

$$K_p\left(R_p(\omega)\right) + K_d\left(-\omega I_p(\omega)\right) = -1, \tag{3.1.6}$$

and

$$K_p\left(I_p(\omega)\right) + K_d\left(\omega R_p(\omega)\right) = 0. \tag{3.1.7}$$

Hence, we need to solve the following system

$$\begin{bmatrix} R_p(\omega) & -\omega I_p(\omega) \\ I_p(\omega) & \omega R_p(\omega) \end{bmatrix} \begin{bmatrix} K_p \\ K_d \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \end{bmatrix}. \tag{3.1.8}$$

Solving (3.1.8) for $\omega \neq 0$, we obtain

$$K_p(\omega) = -\frac{R_p(\omega)}{\left|G_p(j\omega)\right|^2} \tag{3.1.9}$$

and

$$K_d(\omega) = \frac{I_p(\omega)}{\omega\left|G_p(j\omega)\right|^2}. \tag{3.1.10}$$

Note here that

$$K_d(\omega) = -\frac{1}{\omega^2} K_i(\omega). \tag{3.1.11}$$

Solving (3.1.8) for $\omega = 0$, we find that K_d is arbitrary while

$$K_p = -\frac{1}{R_p(0)} \tag{3.1.12}$$

and

$$I_p(0) = 0. \tag{3.1.13}$$

The lower bound in (3.1.12) will be non-zero only for type-zero plants. Otherwise, $R_p(0) = \infty$ and

$$K_p = 0. \tag{3.1.14}$$

Also note that for all real plants, (3.1.13) will hold and $R_p(0)$ will be equal to the DC gain of the plant.

Again, we need to find the upper bound ω_c, that is, the point where

$K_p(\omega_c) = K_p(0).$

Consider the two diagrams in Fig. 5.

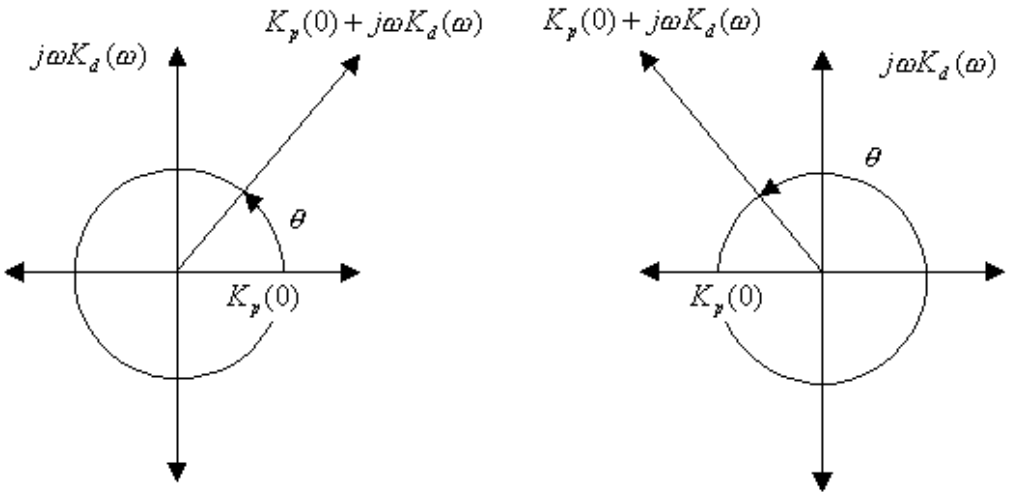

Fig. 5. Vector diagrams for a PD controller.

The diagram on the left describes a PD controller function when $K_p(0) > 0$. Note that the amount of phase that this particular controller will add at the critical frequency to the loop transfer function is anywhere from zero to $\frac{\pi}{2}$ radians which leaves a range of $\left[-\pi, -\frac{3\pi}{2}\right)$ for the phase of the plant transfer function. Similarly, the diagram on the right shows a vector composition of a PD controller where $K_p(0) < 0$. In this case the controller will add anywhere from $\frac{\pi}{2}$ to π radians at the critical frequency, which means that the phase of the plant transfer function at ω_c can be anywhere in the set $\left(-\frac{3\pi}{2}, -2\pi\right]$. In the case where the plant transfer function is a type-one system or higher,

$K_p(0) = 0$ will hold, resulting in the controller phase angle of $\frac{\pi}{2}$ radians at ω_c and the plant transfer function phase of $-\frac{3\pi}{2}$ radians at the critical frequency.

Therefore, at the critical frequency, the system will satisfy

$$\angle G_p(j\omega_c) \in \begin{Bmatrix} \left[-\pi, -\frac{3\pi}{2}\right), \text{ if } K_p(0) > 0 \\ \left[-\frac{3\pi}{2}\right], \text{ if } K_p(0) = 0 \\ \left(-\frac{3\pi}{2}, -2\pi\right], \text{ if } K_p(0) < 0 \end{Bmatrix}. \tag{3.1.15}$$

3.2 Bounds for Desired Gain and Phase Margins

We proceed the same way as in the PI case, this time using (3.1.1) as the controller function. To obtain the values for K_p and K_d, we decompose (2.2.1) into real and imaginary parts and substitute them into (3.1.9) and (3.1.9) where $\omega \in [0, \omega_c]$.

CHAPTER IV

STABILIZATION WITH A PID CONTROLLER

4.1 Stability Bounds

Finally, we attempt to stabilize the system in Fig. 3 with the PID controller that is defined as

$$G_c(s) = K_p + \frac{K_i}{s} + K_d s = \frac{K_p s + K_i + K_d s^2}{s}. \tag{4.1.1}$$

This time, the goal is to find the values of K_p, K_i and K_d such that the zeros of the closed-loop characteristic equation of the system in Fig. 3 are in the left half of the complex plane.

The characteristic polynomial in terms of $j\omega$ can be written as

$$\Delta(j\omega) = 1 + \left(\frac{K_p j\omega + K_i + K_d \omega^2}{j\omega} \right) \left(R_p(\omega) + jI_p(\omega) \right). \tag{4.1.2}$$

After expanding and equating $\Delta(j\omega)$ to zero, we obtain

$$R_\Delta(\omega) + I_\Delta(\omega) = 0, \tag{4.1.3}$$

where

$$R_\Delta(\omega) = \left(\omega R_p(\omega) \right) K_p + \left(I_p(\omega) \right) K_i - \left(I_p(\omega)\omega^2 \right) K_d + \omega \tag{4.1.4}$$

and

$$I_\Delta(\omega) = \left(\omega I_p(\omega) \right) K_p - \left(R_p(\omega) \right) K_i + \left(R_p(\omega)\omega^2 \right) K_d. \tag{4.1.5}$$

Setting the real and imaginary parts to zero yields

$$\left(\omega R_p(\omega)\right)K_p + \left(I_p(\omega)\right)K_i = K_d\left(I_p(\omega)\omega^2\right) - \omega \tag{4.1.6}$$

and

$$\left(\omega I_p(\omega)\right)K_p - \left(R_p(\omega)\right)K_i = -K_d\left(R_p(\omega)\omega^2\right). \tag{4.1.7}$$

Applying the methods from chapters II and III, we can obtain the stability boundary locus in the $\left(K_p, K_i\right)$ plane for a fixed value of K_d and, similarly, the locus in the $\left(K_p, K_d\right)$ plane for a fixed value of K_i.

Let us first consider the solution in the $\left(K_p, K_i\right)$ plane. Here, we need to solve the following system

$$\begin{bmatrix} \omega R_p(\omega) & I_p(\omega) \\ \omega I_p(\omega) & -R_p(\omega) \end{bmatrix}\begin{bmatrix} K_p \\ K_i \end{bmatrix} = \begin{bmatrix} K_d\left(I_p(\omega)\omega^2\right) - \omega \\ -K_d\left(R_p(\omega)\omega^2\right) \end{bmatrix}. \tag{4.1.8}$$

Solving (4.1.8) for $\omega \neq 0$, we obtain

$$K_p(\omega) = -\frac{R_p(\omega)}{\left|G_p(j\omega)\right|^2} \tag{4.1.9}$$

and

$$K_i(\omega) = \omega^2 K_d(\omega) - \frac{\omega I_p(\omega)}{\left|G_p(j\omega)\right|^2}. \tag{4.1.10}$$

In the $\left(K_p, K_d\right)$ plane the system becomes

$$\begin{bmatrix} -\omega R_p(\omega) & \omega^2 I_p(\omega) \\ \omega I_p(\omega) & \omega^2 R_p(\omega) \end{bmatrix}\begin{bmatrix} K_p \\ K_d \end{bmatrix} = \begin{bmatrix} \omega + K_i I_p(\omega) \\ K_i R_p(\omega) \end{bmatrix}, \tag{4.1.11}$$

which means that $K_p(\omega)$ is the same as in (4.1.9) and for $\omega \neq 0$,

$$K_d(\omega) = \frac{K_i(\omega)}{\omega^2} + \frac{I_p(\omega)}{\omega\left|G_p(j\omega)\right|^2}. \tag{4.1.12}$$

Solving (4.1.8) for $\omega = 0$ we obtain the equation identical to (2.1.16) and, therefore, we conclude that that K_p is arbitrary and $K_i = 0$, unless $I_p(0) = R_p(0) = 0$ which holds only when $G_p(s)$ has a zero at the origin. The same results follow when we solve (4.1.13) for $\omega = 0$.

Note that since the PID solution depends on both $\left(K_p, K_i\right)$ and $\left(K_p, K_d\right)$ planes, we can use equations (2.1.17) and (3.1.15), respectively, to find the critical frequencies.

The general solution in the $\left(K_i, K_d\right)$ plane for a fixed K_p is not possible since the system that needs to be solved is

$$\begin{bmatrix} -I_p(\omega) & \omega^2 I_p(\omega) \\ -R_p(\omega) & \omega^2 R_p(\omega) \end{bmatrix} \begin{bmatrix} K_i \\ K_d \end{bmatrix} = \begin{bmatrix} \left(\omega R_p(\omega)\right)K_p + \omega \\ -\left(\omega I_p(\omega)\right)K_p \end{bmatrix} \tag{4.1.14}$$

and

$$\left\| \begin{bmatrix} -I_p(\omega) & \omega^2 I_p(\omega) \\ -R_p(\omega) & \omega^2 R_p(\omega) \end{bmatrix} \right\| = 0. \tag{4.1.15}$$

However, in [3], applying the Hermite-Biehler theorem to quasi-polynomials, Silva et al. have proved that for a fixed value of K_p, the solution set in the $\left(K_i, K_d\right)$ plane is a convex polygon. Such polygon can be obtained by the following procedure:

(1) Define two stability boundaries in the $\left(K_p, K_i\right)$ plane for a fixed K_d.

(2) Define two stability boundaries in the $\left(K_p, K_d\right)$ planes for a fixed K_i.

(3) Extract the four lines that define the sides of the polygon.

This will be illustrated in examples that follow.

4.2 Bounds for Desired Gain and Phase Margins

Using (4.1.1) as the controller function and fixing K_p, we can obtain the values for K_i and K_d by decomposing (2.2.1) into real and imaginary parts and substituting them into (4.1.9), (4.1.10) and (4.1.12) with the appropriate critical frequency for $\left(K_p, K_i\right)$ and $\left(K_p, K_d\right)$ plane.

CHAPTER V

EXAMPLES

5.1 Example 1:

In this example we will consider the problem of finding the analytical solution for the set of all stabilizing PI controllers for the second order transfer function

$$G_p(s) = \frac{K_f}{s(s+a)} e^{-s\tau}, \tag{5.1.1}$$

where $a > 0$ and $K_f > 0$ is the plant gain. We will also solve the problem for a specific a and K_f. Applying Euler's formula and expanding, we get

$$G_p(j\omega) = -\frac{K_f \cos(\omega\tau)}{\omega^2 + (a\omega)j} + \frac{K_f \sin(\omega\tau)j}{\omega^2 + (a\omega)j}. \tag{5.1.2}$$

Now we multiply both the numerator and the denominator of $G_p(j\omega)$ by the complex conjugate $\omega^2 - (a\omega)j$ and rearrange such that $G_p(j\omega) = R_p(\omega) + jI_p(\omega),$ where

$$R_p(\omega) = -\frac{K_f}{\omega}\left(\frac{\omega\cos(\omega\tau) + a\sin(\omega\tau)}{\omega^2 + a^2}\right) \tag{5.1.3}$$

and

$$I_p(\omega) = \frac{K_f}{\omega}\left(\frac{\omega\sin(\omega\tau) - a\cos(\omega\tau)}{\omega^2 + a^2}\right). \tag{5.1.4}$$

Substituting (5.1.3) and (5.1.4) into (2.1.13) and (2.1.14), respectively, we achieve the boundary defined by

$$K_p(\omega) = -\frac{\omega^2\cos(\omega\tau) + a\omega\sin(\omega\tau)}{K_f} \tag{5.1.5}$$

and

$$K_i(\omega) = \frac{a\omega^2 \cos(\omega\tau) - \omega^3 \sin(\omega\tau)}{K_f}. \tag{5.1.6}$$

To find the critical frequency of this particular system, we set the phase of the transfer function equal to $-\pi$, or equivalently,

$$-\omega_c \tau - \frac{\pi}{2} - \tan^{-1}\left(\frac{\omega_c}{a}\right) = -\pi. \tag{5.1.7}$$

Simplifying and taking tangent of both sides results in

$$\tan(\omega_c \tau) = \frac{a}{\omega_c}. \tag{5.1.8}$$

Now suppose that $K_f = 1000$, $a = 2$ and $\tau = 0.7$. Equations (5.1.5) and (5.1.6) give

$$K_p(\omega) = -\frac{\omega^2 \cos(\omega\tau) + 2\omega \sin(\omega\tau)}{1000} \tag{5.1.9}$$

and

$$K_i(\omega) = \frac{2\omega^2 \cos(\omega\tau) - \omega^3 \sin(\omega\tau)}{1000} \tag{5.1.10}$$

The critical frequency can be found by solving (5.1.8) graphically.

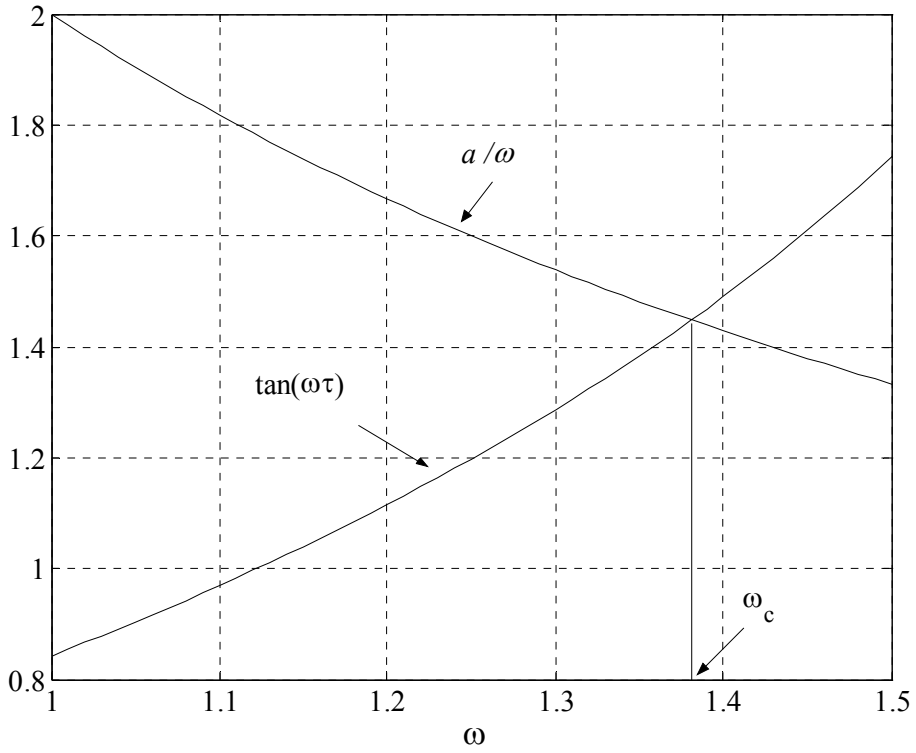

Fig. 6. Plot of the terms involved in (5.1.8).

From Fig. 6, $\omega_c = 1.3808$ rad/s and the stability region in the (K_p, K_i) plane is displayed in Fig. 7. A detailed Matlab script for each graph is provided in the appendix.

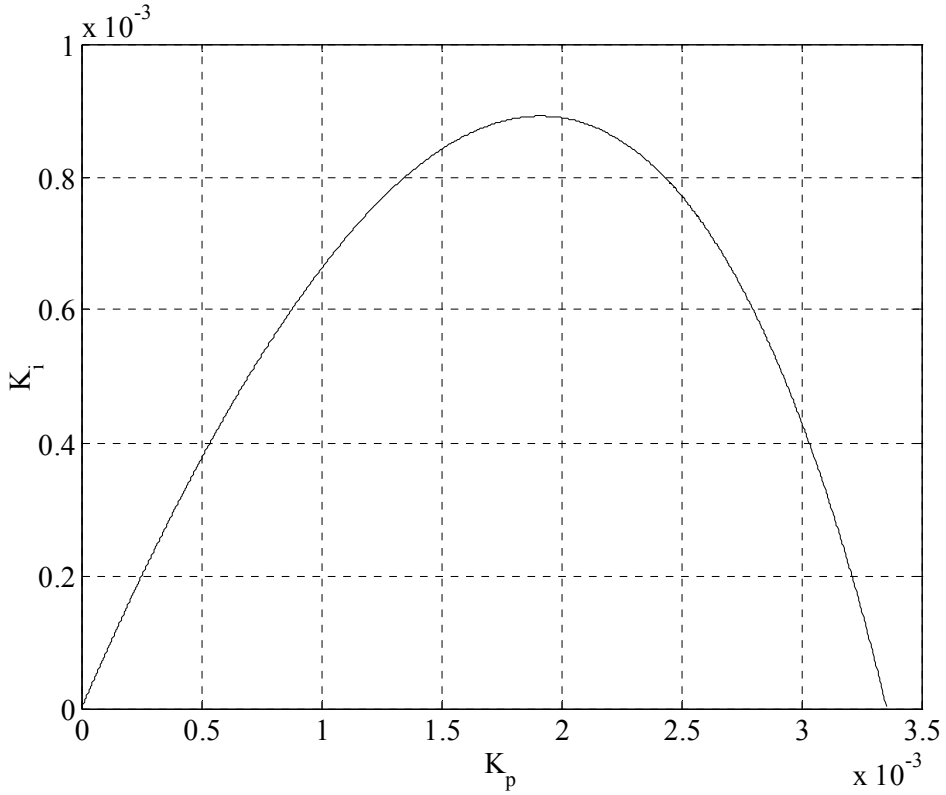

Fig. 7. Stability region for Ex. 1.

A plot of the time response of the closed-loop system to a unit step input for $K_p = 1.0 \times 10^{-3}$ and $K_i = 1.0 \times 10^{-4}$ is shown in Fig. 8, indicating the stability.

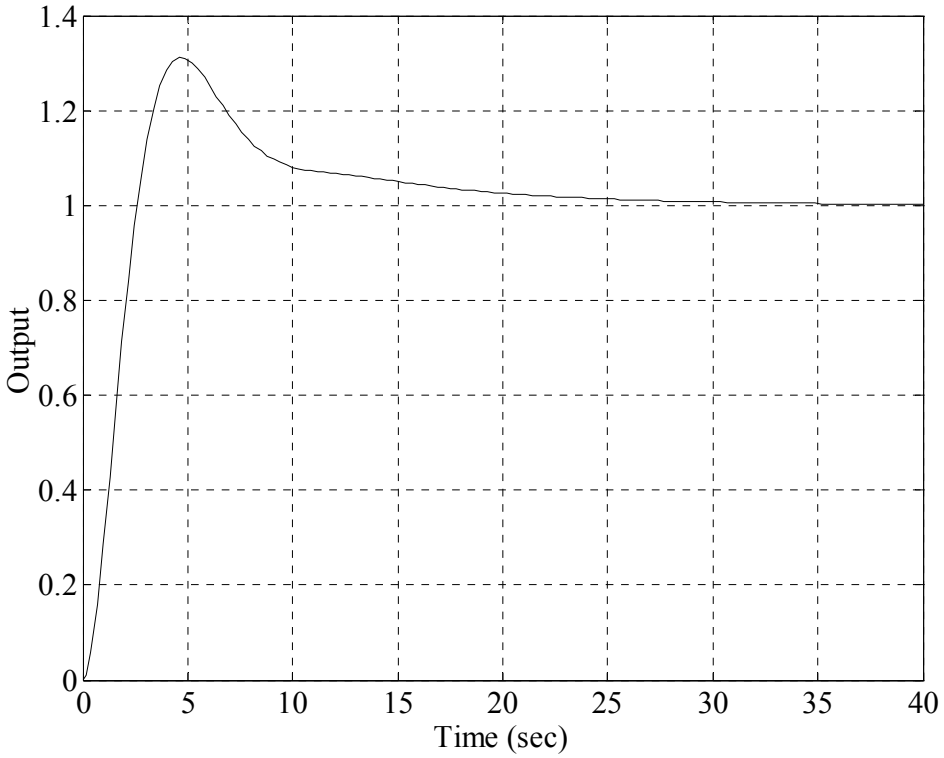

Fig. 8. Time response of the closed-loop system for Ex. 1.

5.2 Example 2:

In this example, we will take an experimentally measured frequency response of a Quanser DC motor and numerically find the set of all stabilizing PD controllers such that the gain margin is greater than or equal to 2 and the phase margin is greater than or equal to $\frac{\pi}{4}$. The virtual dynamic signal analyzer discussed in [5] was used to measure the frequency response of a Quanser SRV02-ET DC motor with a 0.1 sec. measurement

delay. The frequency response was measured for the motor input voltage to the shaft angular position. The magnitude and phase plots of the motor are displayed in Fig. 9.

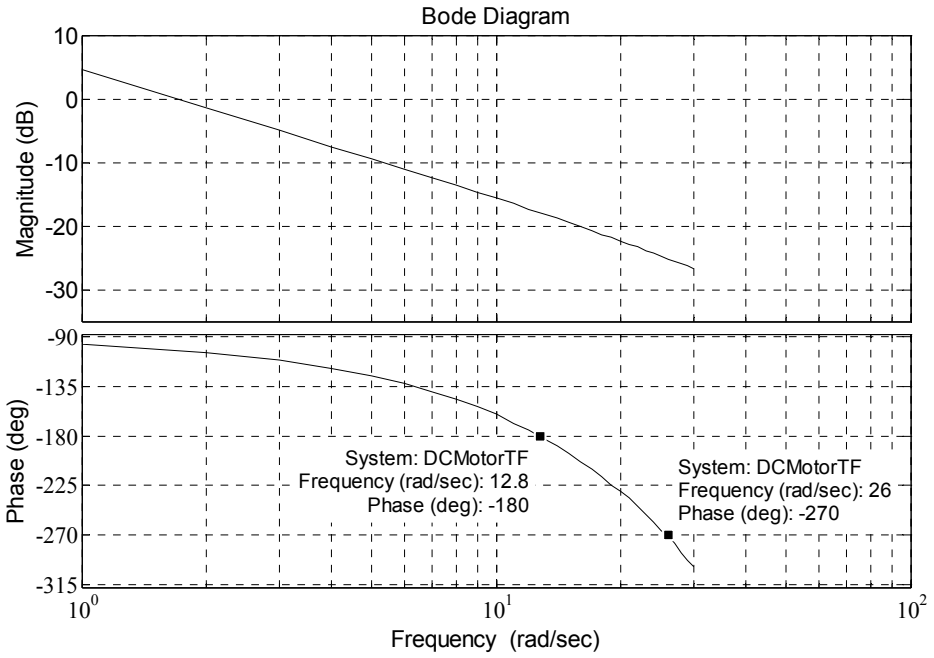

Fig. 9. Magnitude and phase plots of a Quanser DC servo.

We first compute the general stability boundary locus, *i.e.*, the locus when $A = 1$ and $\theta = 0$. To design the controllers that satisfy the gain margin condition, we first set $A = 2$ and $\theta = 0$ in (2.2.1). Then, from the given frequency response, we can extract the real and imaginary value at each ω and substitute it into (3.1.9) and (3.1.10) to obtain the stability boundary. To satisfy the criterion for the phase margin, we repeat the procedure, but with $A = 1$ and $\theta = \frac{\pi}{4}$. The results are shown in Fig. 10.

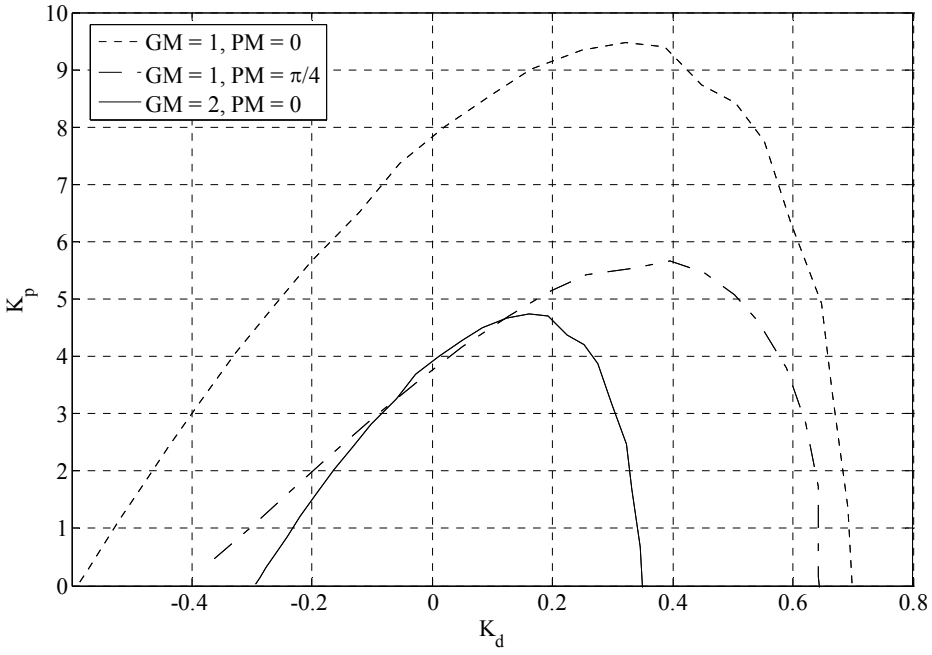

Fig. 10. Stability region for Ex. 2.

The region obtained from the intersection of the areas under the three graphs contains all stabilizing values of K_p and K_d that satisfy the desired gain and phase margin requirements.

To verify our results, we chose $K_d = 0.2$ and $K_p = 4.0$ and tested the response of the motor to a step input of 45°. From Fig. 11, we see that the closed-loop system is indeed stable.

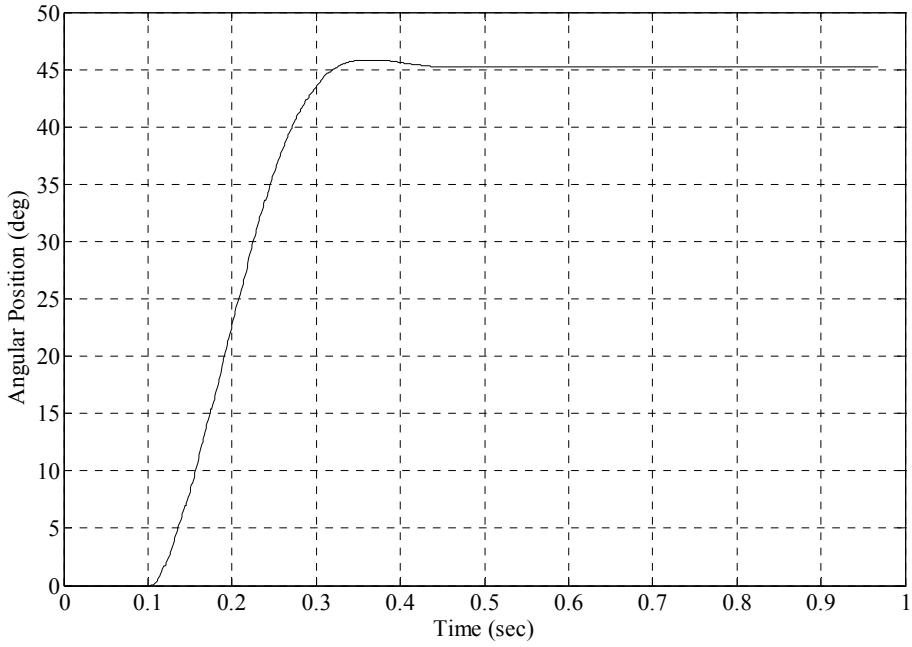

Fig. 11. Step response of a Quanser DC servo with GM ≥ 2 and PM $\geq \frac{\pi}{4}$.

5.3 Example 3:

In this example, we will take a known fourth order unstable transfer function

$$G_p(s) = \frac{27}{(s-1)(s+28)^3} e^{-0.5s} \tag{5.3.1}$$

that was analyzed in [1], and find the set of all stabilizing PI controllers numerically. From the Bode diagram in Fig. 12, it is clear that the critical frequency is 0.95 rad/s.

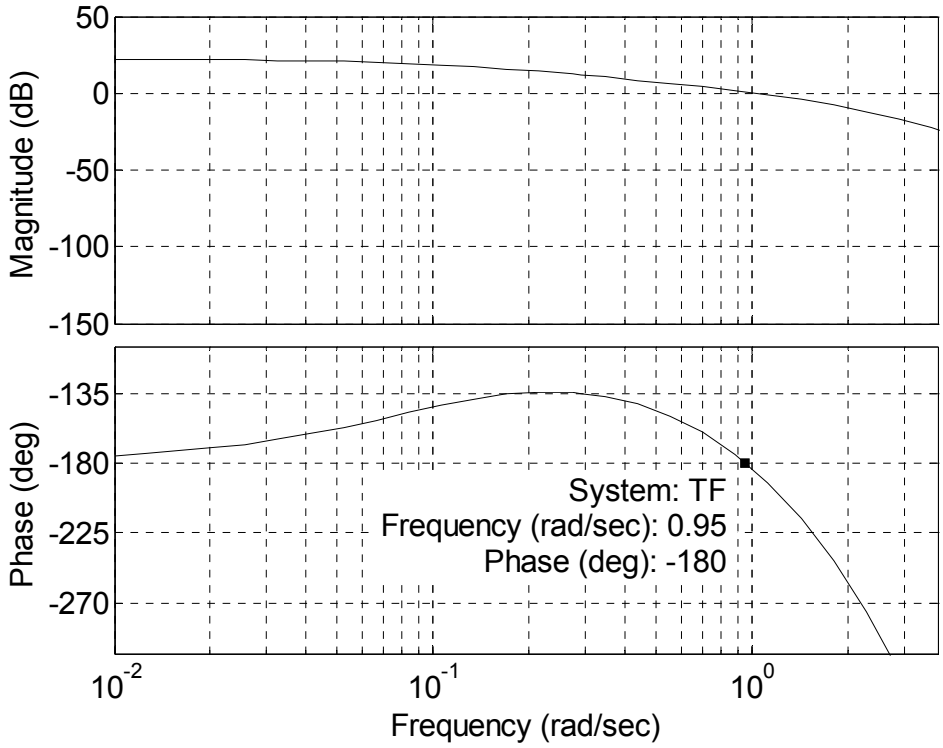

Fig. 12. Magnitude and phase plots of the transfer function in Ex. 3.

Extracting real and imaginary parts and solving (2.1.13) and (2.1.14) for $\omega \in [0, 0.95]$, we obtain the stability region in Fig. 13, which is identical to that presented in [1].

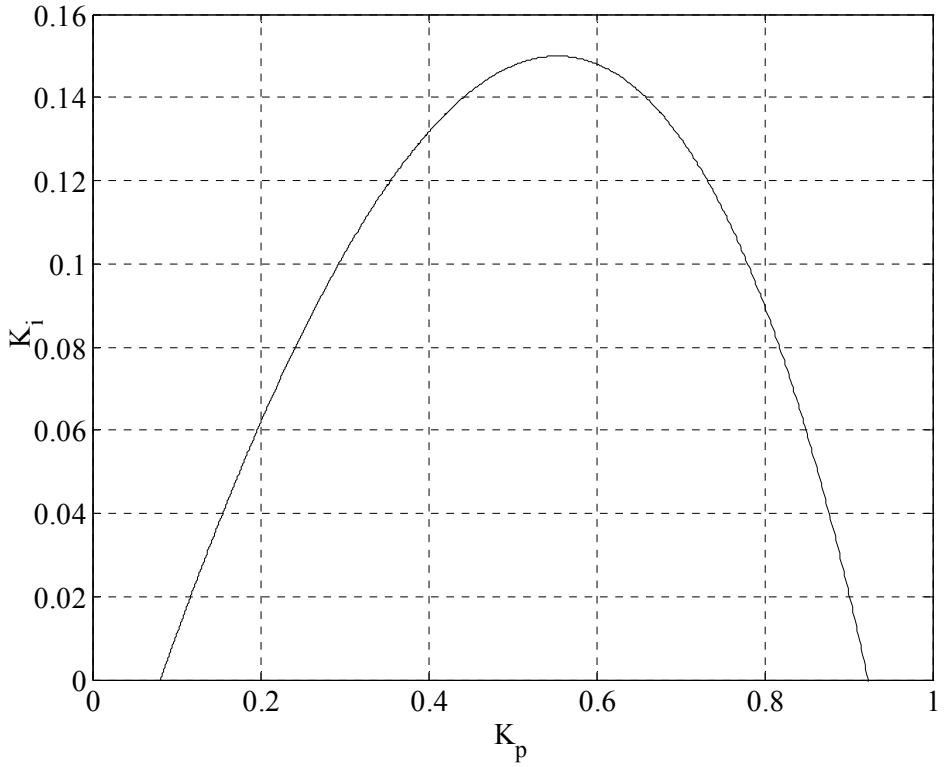

Fig. 13. Stability region for Ex. 3.

5.4 Example 4:

Consider a plant with a second order transfer function

$$G_p(s) = \frac{-0.5s+1}{(s+1)(2s+1)} e^{-0.6s}, \tag{5.4.1}$$

that has been analyzed in [1]. The goal is to find all stabilizing values of a PID controller such that the closed loop system is stable. The phase and magnitude are shown in Fig. 14.

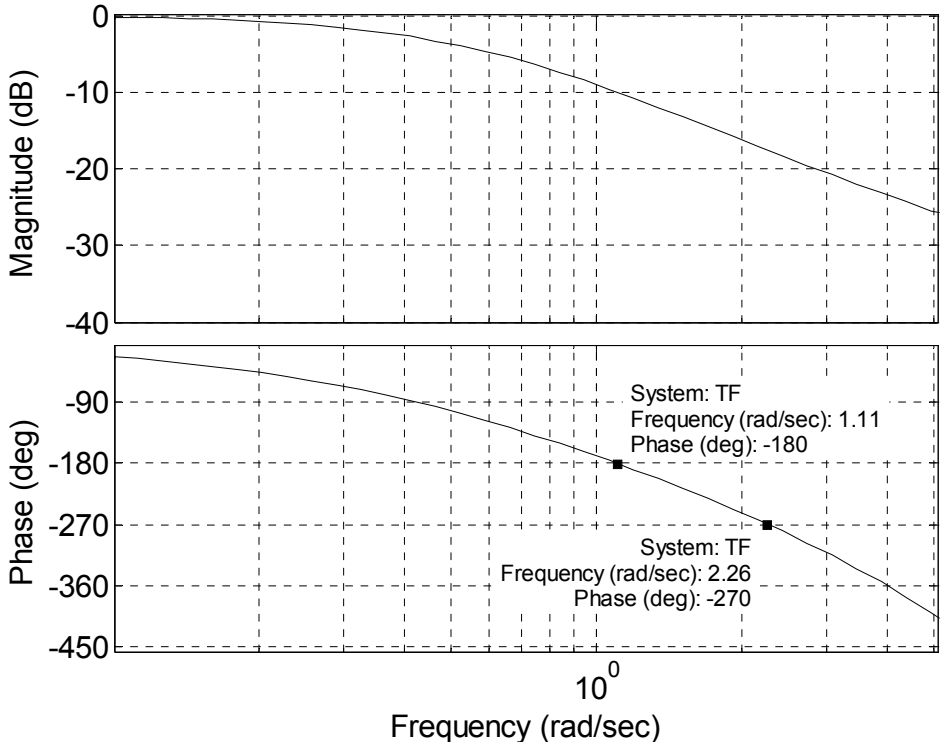

Fig. 14. Magnitude and phase plots of the transfer function in Ex. 4.

Using (4.1.9) and (4.1.10) we obtain the two stability regions in the $\left(K_p, K_i\right)$ plane for $K_d = 0$ and $K_d = 0.4$, shown in Fig. 15. Similarly, using (4.1.9) and (4.1.12), in Fig. 16 we graph the two stability boundaries in the $\left(K_p, K_d\right)$ plane for $K_i = 0.7$ and $K_i = 1$.

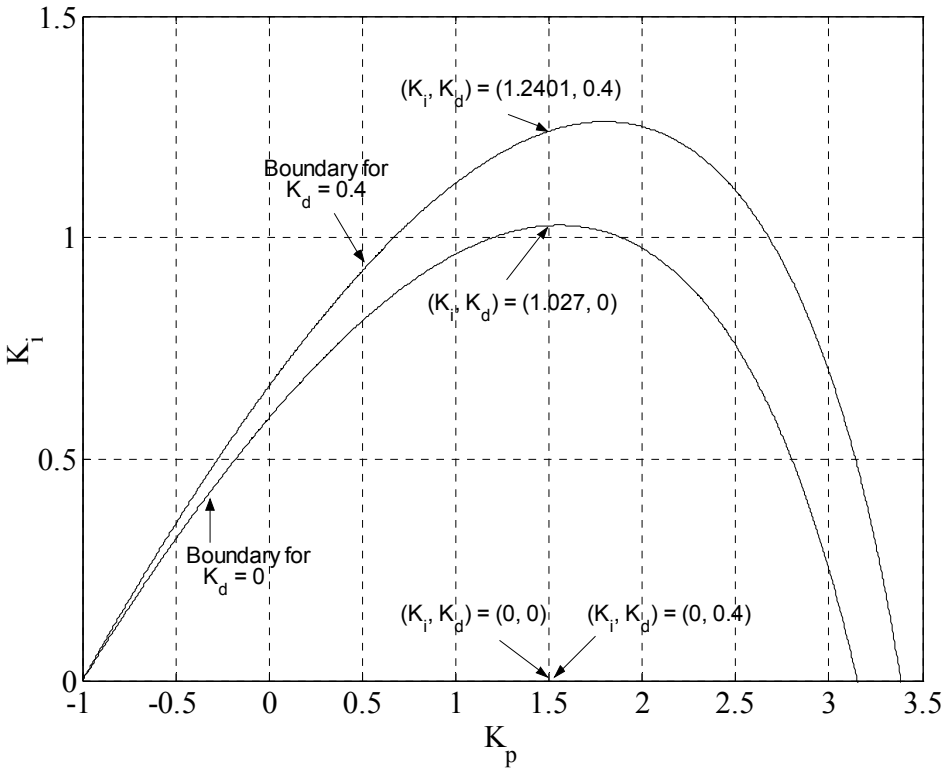

Fig. 15. Stability region in the $\left(K_p, K_i\right)$ plane for $K_d = 0$ and $K_d = 0.4$.

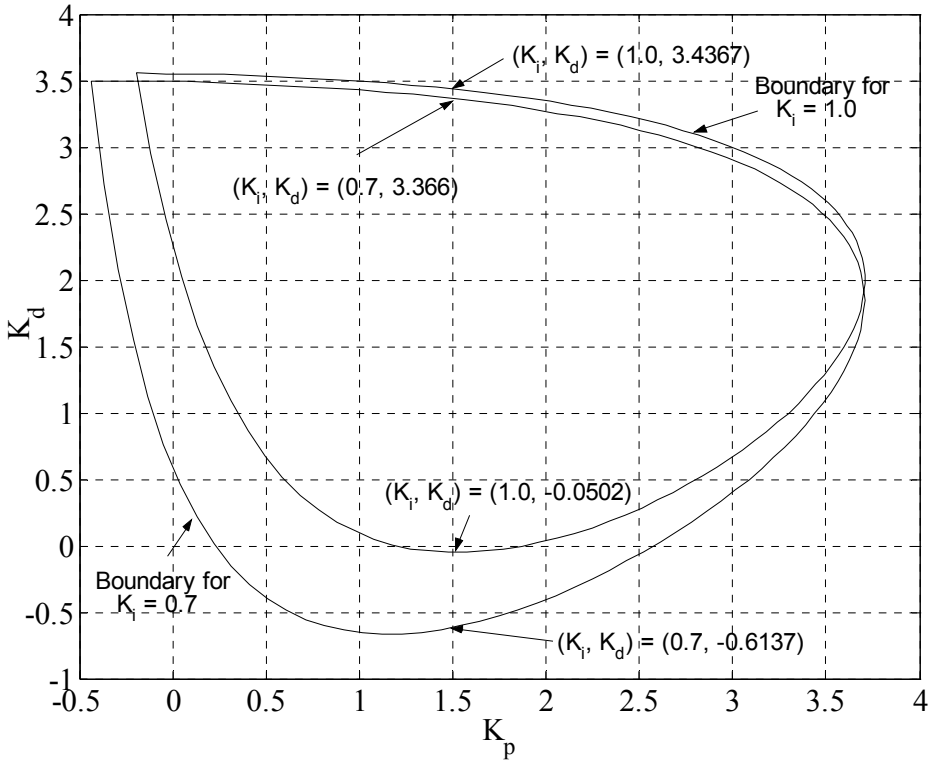

Fig. 16. Stability region in the $\left(K_p, K_d\right)$ plane for $K_i = 0.7$ and $K_i = 1$.

Fig. 17 shows the behavior of stability regions in the $\left(K_p, K_d\right)$ plane for various values of K_i.

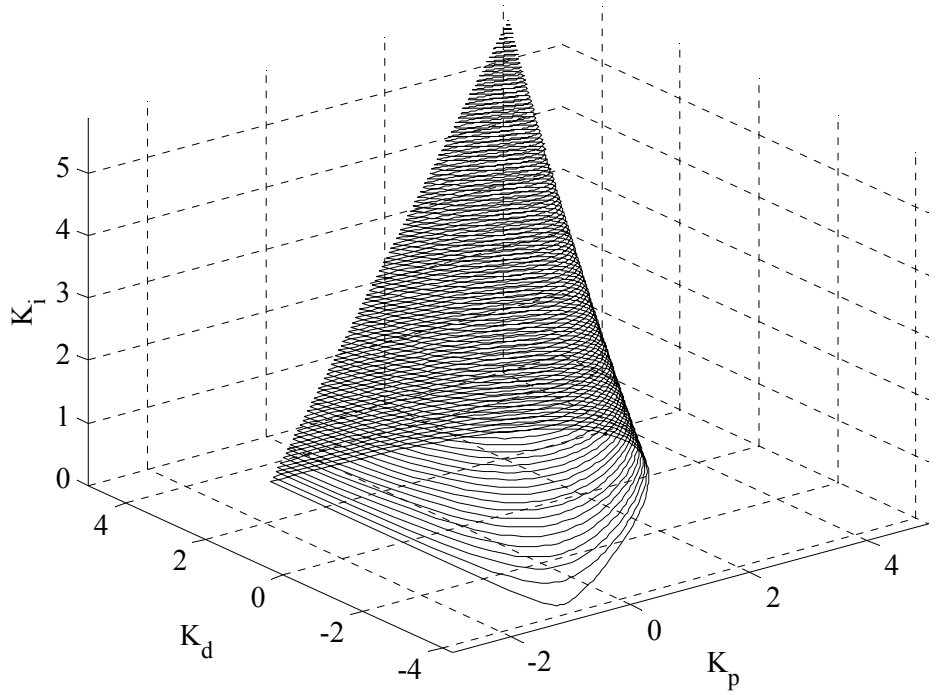

Fig. 17. Stability regions in the $\left(K_p, K_d\right)$ plane for $K_i = [0, 5.8]$.

To obtain the four lines that define the polygon, we fix $K_p = 1.5$. From Fig. 15 we see that the line $K_p = 1.5$ intersects the boundary of the stability region for $K_d = 0.4$ at $K_i = 0$ and $K_i = 1.2401$. Similarly, the line $K_p = 1.5$ intersects the boundary of the stability region for $K_d = 0$ at $K_i = 0$ and $K_i = 1.027$.

Therefore, the first line passes through the points $(K_i, K_d) = (0,0)$ and $(K_i, K_d) = (0,0.4)$, while the second line passes through $(K_i, K_d) = (1.027,0)$ and $(K_i, K_d) = (1.2401,0.4)$. The equations for these two lines are

$$l_1 : K_i = 0 \tag{5.4.2}$$

and

$$l_2 : K_d = 0.533K_i - 0.547\,. \tag{5.4.3}$$

From Fig. 16 we see that one line passes through $(K_i, K_d) = (0.7,-0.6137)$ and $(K_i, K_d) = (1.0,-0.0502)$ while the other one includes the points $(K_i, K_d) = (0.7,3.336)$ and $(K_i, K_d) = (1.0,3.4367)$. Hence, the equations for these two lines are

$$l_3 : K_d = 1.88K_i - 1.93 \tag{5.4.4}$$

and

$$l_4 : K_d = 0.236K_i + 3.2\,. \tag{5.4.5}$$

Graphing these four lines on the same coordinate system, we obtain the convex polygon in Fig. 18 whose interior is the stability region in the (K_i, K_d) plane for $K_p = 1.5$.

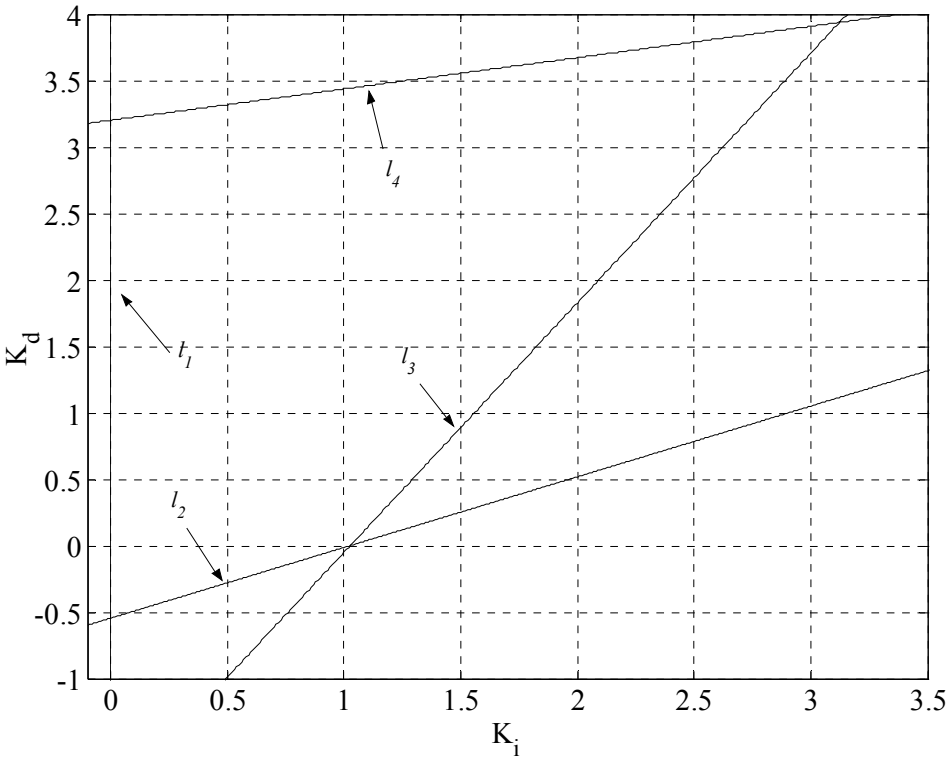

Fig. 18. Stability region for $K_p = 1.5$.

For verification we chose $K_p = 1.5$ with several values from Fig. 18 and tested the step response of the closed loop system. From Fig. 19 we see that the closed-loop system is stable.

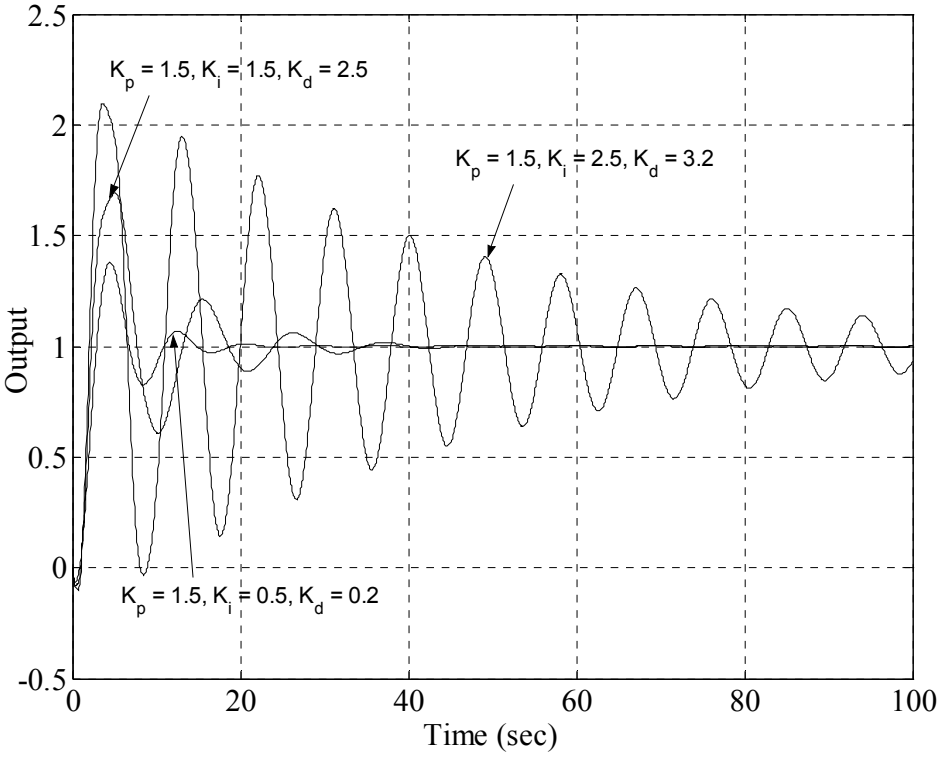

Fig. 19. Step responses for $K_p = 1.5$ and various values of K_i and K_d.

5.5 Example 5:

Here we consider an event operated in space, but controlled from the station on Earth [10]. The goal is to position a telescope driven by a low power actuator to point at a planet. However the telescope receives the command signal from Earth with a delay of $\pi/16$ seconds. Then a sensor measures the direction of the telescope and relays this information back to Earth with a delay of $\pi/16$ seconds. Suppose that it is required to stabilize this experiment by using a PID controller. The system is described in Fig. 20.

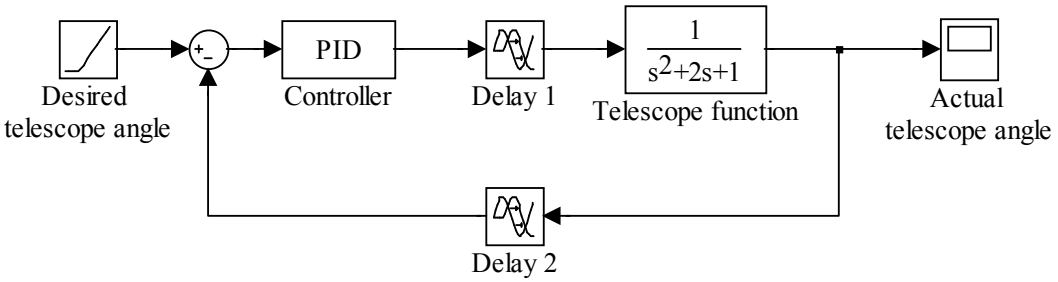

Fig. 20. Feedback control system for the telescope experiment.

So the total transfer function of the telescope, actuator, sensor and round-trip delay is

$$G_p(s) = \frac{1}{(s+1)^2} e^{-\frac{\pi}{8}s}. \tag{5.5.1}$$

The phase and magnitude are shown in Fig. 21.

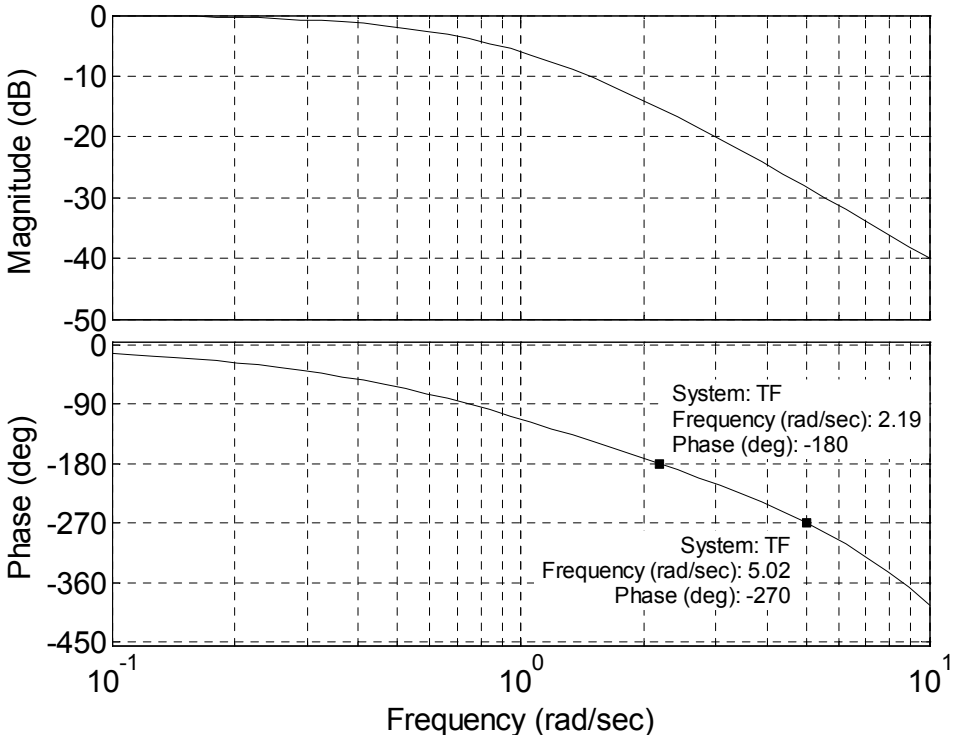

Fig. 21. Magnitude and phase plots of the transfer function in Ex. 5.

Using (4.1.9) and (4.1.10) we obtain the two stability regions in the $\left(K_p, K_i\right)$ plane for $K_d = 0$ and $K_d = 0.07$, shown in Fig. 22. Similarly, using (4.1.9) and (4.1.12), in Fig. 23 we graph the two stability boundaries in the $\left(K_p, K_d\right)$ plane for $K_i = 0.5$ and $K_i = 2.5$.

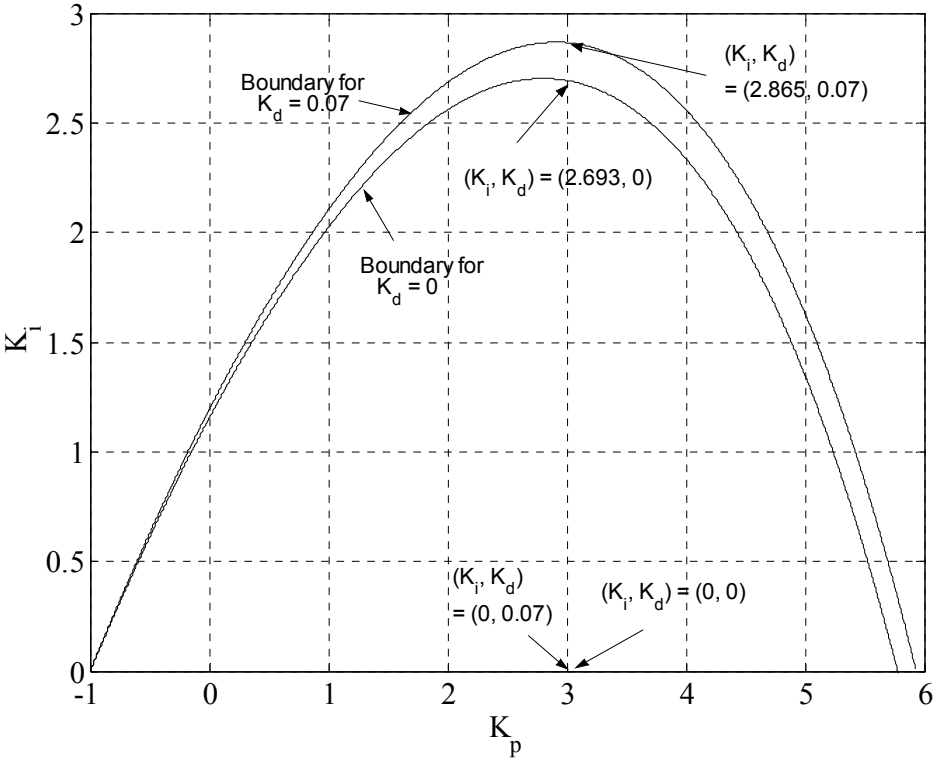

Fig. 22. Stability region in the $\left(K_p, K_i\right)$ plane for $K_d = 0$ and $K_d = 0.07$.

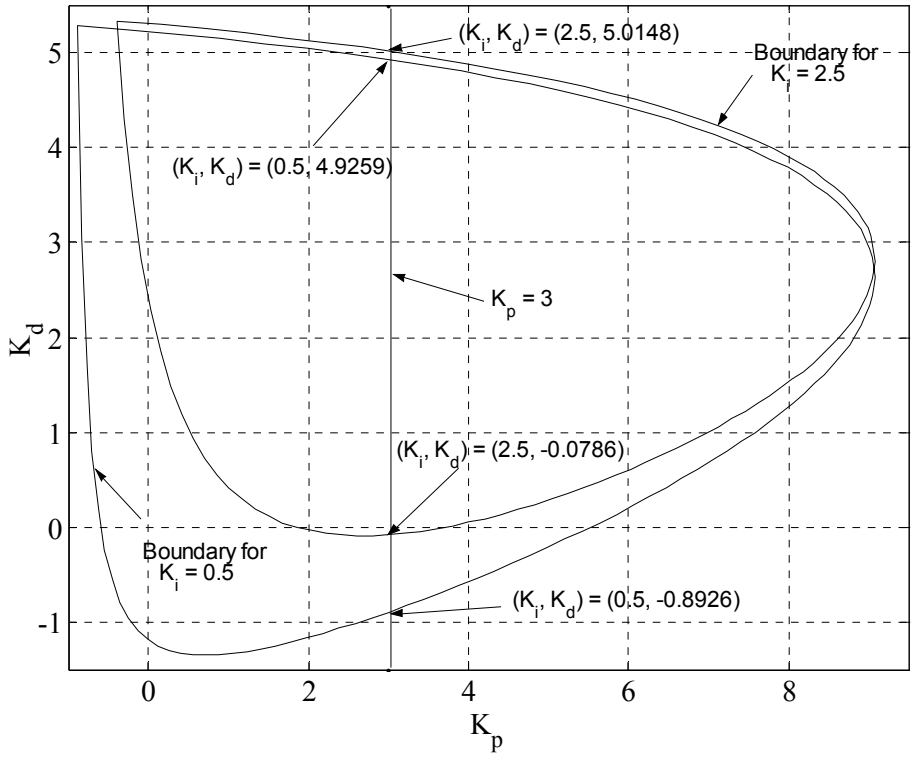

Fig. 23. Stability region in the $\left(K_p, K_d\right)$ plane for $K_i = 0.5$ and $K_i = 2.5$.

Fig. 24 shows the behavior of stability regions in the $\left(K_p, K_d\right)$ plane for various values of K_i.

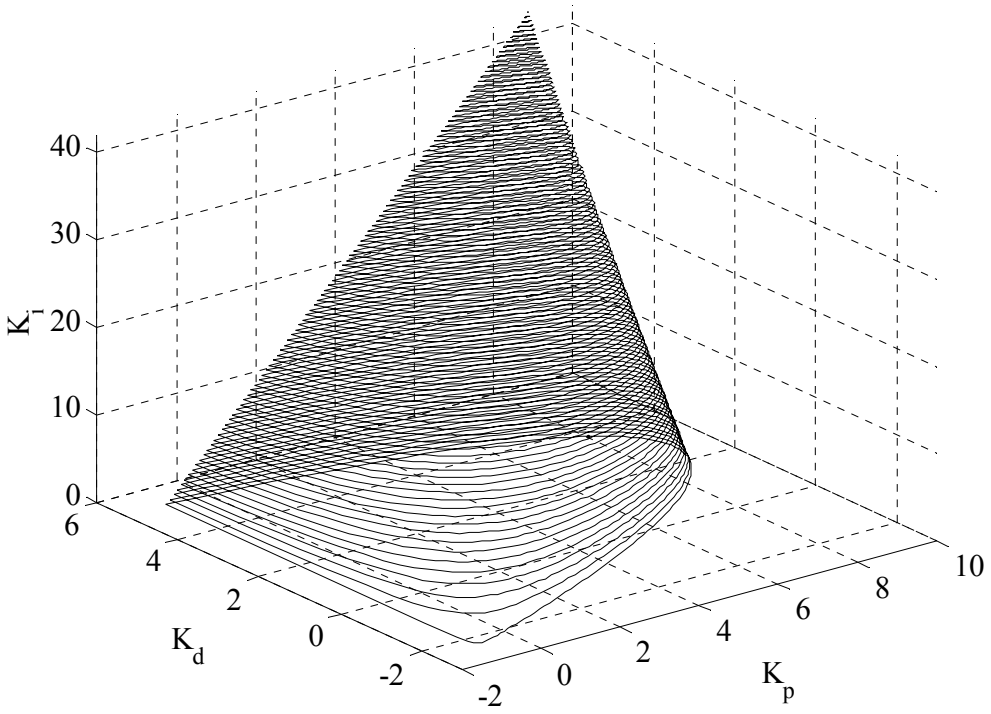

Fig. 24. Stability regions in the $\left(K_p, K_d\right)$ plane for $K_i = [0, 42.0]$.

To obtain the four lines that define the polygon, we fix $K_p = 3.0$. From Fig. 22 we see that the line $K_p = 3.0$ intersects the boundary of the stability region for $K_d = 0.07$ at $K_i = 0$ and $K_i = 2.865$. Similarly, the line $K_p = 3.0$ intersects the boundary of the stability region for $K_d = 0$ at $K_i = 0$ and $K_i = 2.693$.

Therefore, the first line passes through the points $(K_i, K_d) = (0,0)$ and $(K_i, K_d) = (0, 0.07)$, while the second line passes through $(K_i, K_d) = (2.693, 0)$ and $(K_i, K_d) = (2.865, 0.07)$. The equations for these two lines are

$$l_1 : K_i = 0 \tag{5.5.2}$$

and

$$l_2 : K_d = 0.407 K_i - 1.096 . \tag{5.5.3}$$

From Fig. 23 it is clear that one line passes through $(K_i, K_d) = (0.5, -0.8926)$ and $(K_i, K_d) = (2.5, -0.0786)$ and the other one goes through $(K_i, K_d) = (0.5, 4.9259)$ and $(K_i, K_d) = (2.5, 5.015)$. Therefore, the equations for these two lines are

$$l_3 : K_d = 0.407 K_i - 1.096 = l_2 \tag{5.5.4}$$

and

$$l_4 : K_d = 0.044 K_i + 4.9 . \tag{5.5.5}$$

Graphing these four lines on the same coordinate system, we obtain the convex polygon in Fig. 25 whose interior is the stability region in the (K_i, K_d) plane for $K_p = 3.0$.

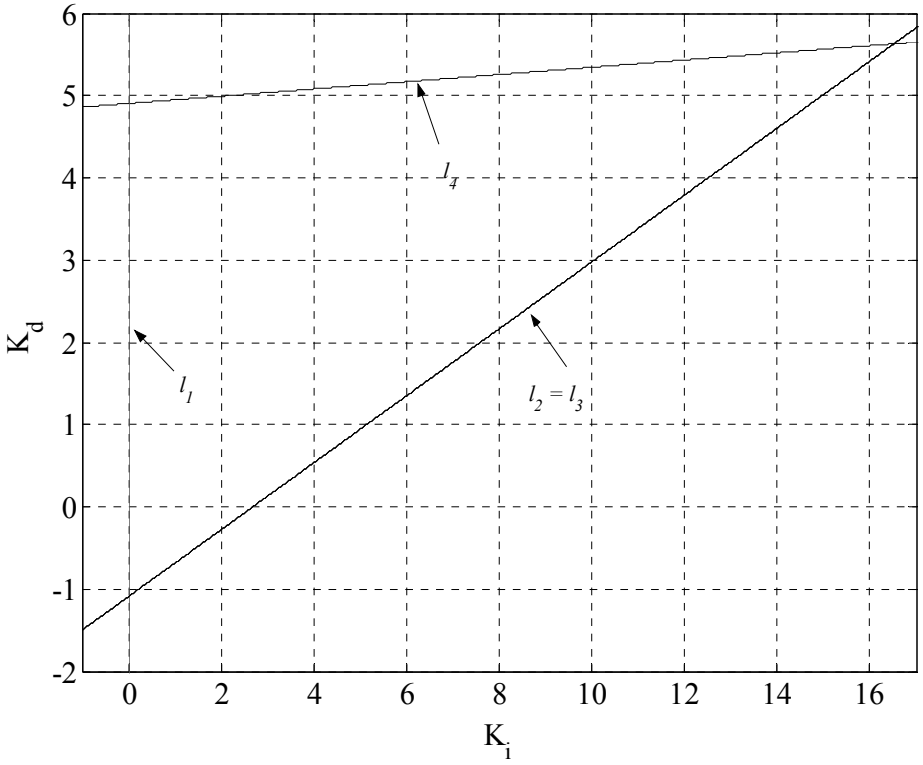

Fig. 25. Stability region for $K_p = 3.0$.

For verification we chose a few values of K_i and K_d from the convex set in Fig. 25 and tested the step response of the closed loop system. From Fig. 26 we see that the closed-loop system is indeed stable.

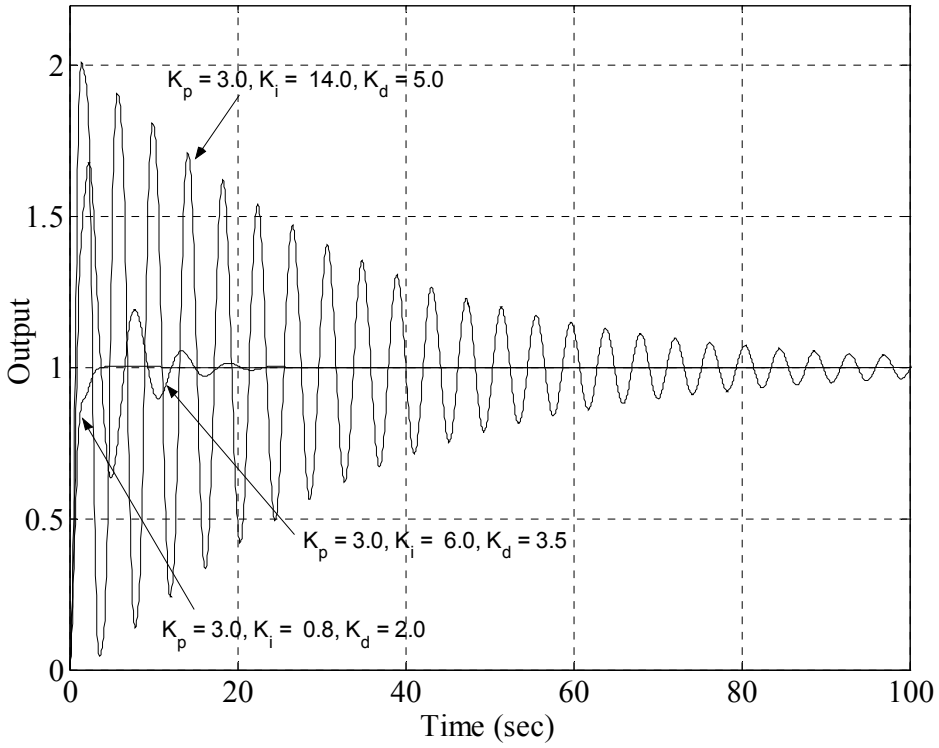

Fig. 26. Step responses for different values of K_i and K_d in Ex. 5.

CHAPTER VI

CONCLUSION

This thesis provides an efficient and straightforward method for stabilization of PI, PD and PID loops. The procedure is based on the frequency response of the plant, whose parameters may or may not be known, and does not make any restriction on the order of the system. Compared to previously developed methods, this approach has advantages in the sense that it is faster, no complex mathematical derivations are necessary and it can be directly applied to plants with only a measured frequency response. We also extend the algorithm to systems that require a specific gain and phase margin. Our methods are shown to be successful in the several examples.

LIST OF REFERENCES

REFERENCES

[1] N. Tan. "Computation of stabilizing PI and PID controllers for processes with time delay," *ISA Trans.*, vol. 44, 2005, pp. 213-223.

[2] J. Watkins and G. Piper. "On the impact of Cross-Link Delays on Spacecraft Formation Control," *Journal of the Astronautical Science*, to be published.

[3] G. J. Silva, A. Datta, and S. P. Bhattacharyya. PID Controllers for Time-Delay Systems. Boston: Birkhäuser, 2005.

[4] S. Tavakoli and P. Fleming. "Optimal Tuning of PI Controllers for First Order plus Dead Time/Long Dead Time Models Using Dimensional Analysis," *European Control Conference*, 2003.

[5] Watkins, J. M. and O'Brien Jr., R.T. "A novel approach to a control systems laboratory," in *Proc. of the IMECE,* Washington, 2003.

[6] Nichols, N. B. and Ziegler, J. G., "Optimum Settings for Automatic Controllers", Trans. ASME, 64, pp. 759-768, (1942).

[7] Aström, K. J. and Hagglund, T. "Automatic Tuning of Simple Regulators with Specifications on Phase and Amplitude Margins", Automatica, Vol. 20, 1984, pp 645-651.

[8] Ender, D.B. *Control Engineering* (September 1993), 180 –190.

[9] Reis, Ronald A and Webb John W. *Programmable Logic Controllers: Principles and Applications (4th Edition).* Prentice Hall, 1998.

[10] Bishop, Robert H. and Dorf, Richard C. *Modern Control Systems*. Prentice Hall, Inc. Upper Saddle River, New Jersey, 2001.

[11] Lelic, Muhidin, "PID Controllers in Nineties," 1999 <www.ece.rutgers.edu/~gajic/IEEETalk.pdf>.

[12] "*PID controller*," <http://en.wikipedia.org/wiki/PID_controller>.

[13] Van Doren, Vance J., "Understanding PID Control," *Control Engineering,* June, 2000 <http://lamspeople.epfl.ch/kirrmann/Slides/PIDControl.htm>.

[14] Gu, Keqin. and Kharitonov, Vladimir L. *Stability of Time-Delay Systems.* Birkhauser, 2003.

APPENDIX

APPENDIX

The following is the collection of the Matlab programs used to generate the specified figures from chapter V.

Matlab code used to generate Fig. 6:

```
clear all

w = 0.001:0.01:pi-0.001;
T = 0.3;
a  = 3.5;

plot(w, atan(w./a))
hold on
plot(w, (pi/2 - (T).*w),'k')
```

Code used to generate Fig. 7:

```
clear all

w = 0:0.001:2.9189;
T = 0.3;
a = 3.5;
Kf = 5;

Kp = ( (w.^2).*cos(w.*T)+a.*w.*sin(w.*T) ) * (1/Kf);
Ki = ( (w.^2)*a.*cos(w.*T)-(w.^3).*sin(w.*T) ) * (1/Kf);

plot (Kp, Ki, 'k')
grid

xlabel('K_p');
ylabel('K_i');
```

Simulink diagram used to generate Fig. 8:

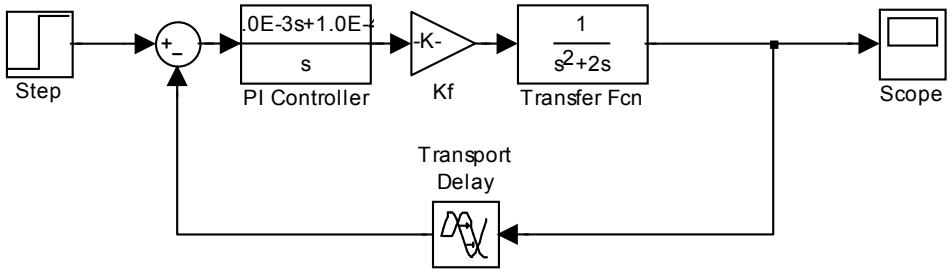

Fig. 27. Simulation diagram used to obtain the step response in Fig. 8.

Code used to generate Fig. 9 and Fig. 10:

```
load encpdpi  % where encpdpi contains the frequency response of the motor

Gfd = Gf;
[Gc,w] = frdata(Gfd,'v');

GM = 1;
PM = 0/180*pi;

Gc = GM*squeeze(freqresp(Gfd,w))*exp(-j*PM);

Rp = real(Gc);
Ip = imag(Gc);
magGc = abs(Gc);

Kp = -Rp./(magGc.^2);
Ki = -w.*Ip./(magGc.^2);
Kd = -Ki./(w.^2);

KpLB = -1/Rp(0);

figure (9)
bode (Gc)

figure(10)
plot(Kd,Kp,'k:')
xlabel('K_d')
```

```
ylabel('K_p')

axis([-.6 .8 KpLB 10])
grid on
hold on

GM = 1;
PM = 45/180*pi;

Gc = GM*squeeze(freqresp(Gfd,w))*exp(-j*PM);

Rp = real(Gc);
Ip = imag(Gc);
magGc = abs(Gc);

Kp = -Rp./(magGc.^2);
Ki  = -w.*Ip./(magGc.^2);
Kd = -Ki./(w.^2);

plot(Kd,Kp,'k-.')

GM = 2;
PM = 0/180*pi;

Gc = GM*squeeze(freqresp(Gfd,w))*exp(-j*PM);

Rp = real(Gc);
Ip = imag(Gc);
magGc = abs(Gc);

Kp = -Rp./(magGc.^2);
Ki  = -w.*Ip./(magGc.^2);
Kd = -Ki./(w.^2);

plot(Kd,Kp,'k-')

legend('GM = 1; PM = 0', 'GM = 1, PM = \pi/4','GM = 2, PM = 0')
```

Script used to generate Fig. 12 and Fig. 13:

```
clear all

TF = tf(27,[1.0000    8.3000   22.6800   19.6000   -2.1952]);
get(TF);
set(TF,'iodelay',0.5);

w = 0:0.001:1;

Gp = squeeze(freqresp(TF,w));

Rp = real(Gp);
Ip = imag(Gp);
magGp = abs(Gp);

Kp = -Rp./(magGp.^2);
Ki = -w'.*Ip./(magGp.^2);

figure(12)
bode(TF)

figure(13)
plot(Kp,Ki)

grid on
xlabel('K_p')
ylabel('K_i')
axis([0 1 0 0.16)
```

Program used to generate Fig. 14 and Fig. 15:

```
clear all

TF=tf([-0.5 1],[2 3 1])
get(TF);
set(TF,'iodelay',0.6);

w = 0.001:0.001:1.22;

Gp=squeeze(freqresp(TF,w));
```

```
Rp=real(Gp);
Ip=imag(Gp);
magGp=abs(Gp);

Kdc1 = 0;
Kdc2 = 0.4;


Kp = -Rp./(magGp.^2);

Ki1= (Kdc1).*(w'.^2) - w'.*Ip./(magGp.^2);

Ki2= (Kdc2).*(w'.^2) - w'.*Ip./(magGp.^2);
figure(14)
bode(TF)

figure(15)
plot(Kp, Ki1,'r', Kp, Ki2,'b')

grid on
xlabel('K_p')
ylabel('K_i')

axis([-1 3.5 0 1.5]);
```

Code used to generate Fig. 16:

```
clear all

TF=tf([-0.5 1],[2 3 1])
get(TF);
set(TF,'iodelay',0.6);

w = 0.001:0.001:4;

Ki1 = 0.7;
Ki2 = 1.8;

GP = squeeze(freqresp(TF,w));

[Kd1, Kp1, wmin1, wmax1] = intlinePID(TF, Ki1);
[Kd2, Kp2, wmin2, wmax2] = intlinePID(TF, Ki2);
```

```
figure(16)
plot(Kp1, Kd1,'r', Kp2, Kd2,'b')

grid on

xlabel('K_p')
ylabel('K_d'),
```

where ***intlinePID*** is the following function:

```
function [Kd, Kp, wmin, wmax] = intlinePID(TF, Ki, PM, GM)

n = 100;
wrange = fsolve(@myfunPID,[0.3 2.2], optimset('display','off'), Ki, TF);
wmax = max(wrange);
wmin = min(wrange);
range = wmin:(wmax-wmin)/(n-1):wmax;

Gp=squeeze(freqresp(TF,range));

Rp = real(Gp);
Ip = imag(Gp);
MagGp = abs(Gp);

Kp = -Rp./(magGp.^2);
Kd = (Ki)./(range'.^2) + Ip./ (range'.*(magGp.^2));
```

and ***myfunPID*** is the function defined as :

```
function F = myfunPID(W, Ki, TF)

GPw1 = evalfr(TF, j*W(1));
GPw2 = evalfr(TF, j*W(2));

Rp1=real(GPw1);
Ip1=imag(GPw1);
magGp1=abs(GPw1);

Rp2=real(GPw2);
Ip2=imag(GPw2);
magGp2=abs(GPw2);
```

```
Kd1= (Ki)./(W(1).^2) + Ip1./ (W(1).*(magGp1.^2));
Kd2= (Ki)./(W(2).^2) + Ip2./ (W(2).*(magGp2.^2));

Kp1 = -Rp1./(magGp1.^2);
Kp2 = -Rp2./(magGp2.^2);

F = [Kd1 - Kd2; Kp1 - Kp2];
```

Code used to generate Fig. 17:

```
clear all

TF = tf([-0.5 1],[2 3 1])
get(TF);
set(TF,'iodelay',0.6);

n = 100;

wmax = 2.26;
w = 0.0001:(wmax-0.0001)/(n-1):wmax;

GP = squeeze(freqresp(TF,w));

Rp = real(GP);
Ip = imag(GP);
magGp = abs(GP);

Ki = 0;

[Kd, Kp, wmin1, wmax1] = intlinePID(TF, Ki);

Kid = Ki*ones(size(Kd));

figure(17)
plot3(Kp, Kd, Kid)

Kds = Kd;
Kps = Kp;
Kis = Kid;

dk3=.01;

Kimax = 5.9;

for Ki = dk3:(Kimax-dk3)/(n-1):Kimax,
```

```
        zrange=fsolve(@myfunPID,[wmax 0.05],optimset('display','off'), Ki, TF);
        zmax=max(zrange);
        zmin=min(zrange);
        z=zmin:(zmax-zmin)/(n-1):zmax;

        GP2 = squeeze(freqresp(TF,z));

        Rp = real(GP2);
        Ip = imag(GP2);
        magGP2 = abs(GP2);

        Kd = (Ki)./(z'.^2) + Ip./ (z'.*(magGP2.^2));
        Kp = -Rp./(magGP2.^2);
        Kid = Ki*ones(size(Kd));

        line(Kp,Kd,Kid)
        Kds = [Kds; Kd];
        Kps = [Kps; Kp];
        Kis  =  [Kis; Kid];

end

axis([-3 5 -4.3 5 0 Kimax])
grid on
xlabel('K_p')
ylabel('K_d')
zlabel('K_i')
```

Script for Fig. 18:

```
clear all

Ki = -5:0.01:20;

Kd1 = Ki*0;
Kd2 = (0.53275)*Ki - 0.547134;

Kd3 = (1.8783)*Ki - 1.9285;
Kd4 = (0.2357)*Ki + 3.201;

figure(18)
plot(Kd1, Ki,'k', Ki, Kd2,'b')
hold on
```

```
plot(Ki, Kd3,'r', Ki, Kd4,'k')

grid
xlabel('K_i')
ylabel('K_d')

axis([-0.1,3.5, -1, 4])
```

Simulink diagram used to generate the responses in Fig. 19:

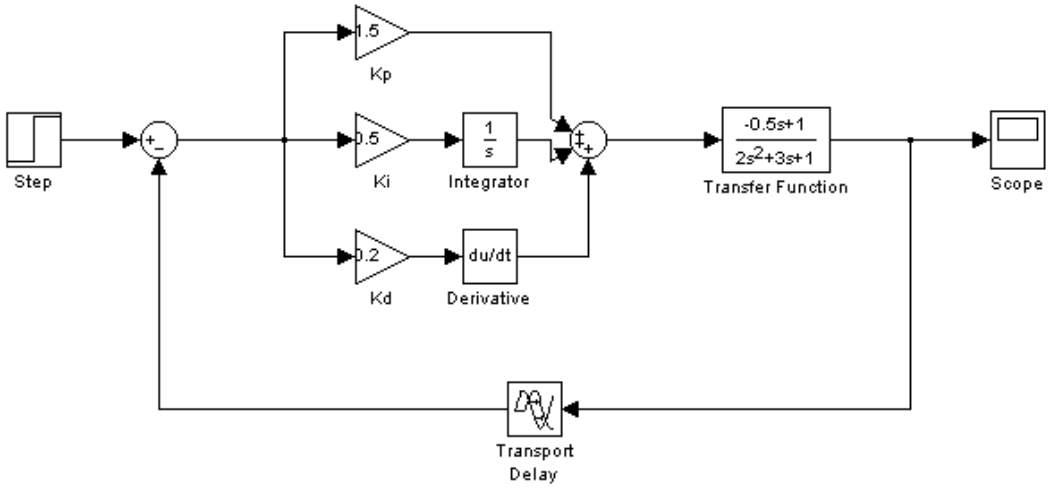

Fig. 28. Diagram used to obtain the step response in Fig. 19.

Codes for the remaining graphs:

Changing the necessary variables such as the transfer function, Kdc1, Kdc2 etc., the same code from Fig. 14 and Fig. 15 can be used to generate the plots in Fig. 21 and Fig. 22.

With minor modifications, the script used to generate Fig. 16 can be used to obtain the plots in Fig. 23.

With minor changes in the transfer function and the frequency ranges, the code for Fig. 17 can be used to generate the graphs in Fig. 24.

Changing the expressions for Kd1 through Kd4 and the axis command in the code used for. Fig. 18, we can obtain the region in Fig. 25.

The schematic diagram from Fig. 28, with the appropriate changes of the controller parameters, the transfer function and the delay can be used to obtain the step responses in Fig. 26.