AN1373
APPLICATION NOTE

SPINDLE MOTOR CONTROL FLL AND DIGITAL FILTER ANALYSIS

by Maurizio Scavazzon

In the disk drive market the need for sinusoidal spindle motor drive is getting more and more important due to factors like acoustic noise, motor wear out, torque ripple, speed control accuracy, etc. In particular, the speed control precision is a key feature and only a flexible and reliable digital solution allows programmability, external components reduction and silicon real estate savings. This invention describes a means to implement the spindle speed control loop in digital way with a fully integrated filter.

It is desirable to design a system capable to include a complete speed control loop in order to alleviate the micro from the task to monitor the speed and correct it. It is also desirable to implement the speed control loop with a digital filter to get flexibility through coefficients setting, reliability (in the case of analog filter, the charge pump pin is a very sensitive node), low cost (measure in terms of silicon real estate), board space and cost reduction with the removal of the passive external components and finally speed accuracy. The implementation described here is a second order IIR filter with setting of three coefficients embedded in a FLL speed control.

In Chapter 1 a mathematical model of the spindle motor will be derived. The mechanical and electrical characteristics will be studied in order to obtain a simplified transfer function of the motor. An example with real spindle parameters will be considered and frequency analysis will be performed.

Chapter 2 talks about designing of an analog compensator. Based on the desired requirements, a second order filter will be derived. At the end an example describes how the filter improve the performances of the overall system.

In Chapter 3, the digital control issue will be analyzed. The technique called "emulation" will be studied and a digital FLL filter will be derived using bilinear transforms.

In Chapter 4 a Mathcad analysis is performed.

In Appendix, the list of symbols used and their units, a Torque conversion table and the External FLL routine (written in C language) can be found.

1 DYNAMIC MODEL OF THE SPINDLE

1.1 Overview

The Spindle is a Brushless DC motor (BLDC). This kind of motor is popular due to the absence of commutators and brushes, as found in ordinary DC motors and to their ability to respond to digital input pulses.

The most important characteristic that makes brushless DC motors different to DC motors, by the mathematical modeling point of view, is the torque-speed curve (linear [non linear] for ordinary DC [BLDC] motors). In the next section, this relationship will be derived.

1.2 Torque-Speed Characteristics

To obtain the Torque-Speed equation, the following assumptions are taken:

• No damping (B=0)

• The motor is running at constant speed ω_m.

The motor is driven with a circuit such that the voltage across each winding (n=0,1,2) is given by

$$V_n = V_s \cdot \sin\left(\theta_e - \frac{2}{3}\pi \cdot n\right) \quad \text{Eq. 1-1}$$

The back-emf voltage ([3]) is

$$e_n = -K_e \omega_m \sin(\theta_e + \alpha_n) \quad \text{Eq. 1-2}$$

where $\alpha_n = -\frac{2}{3}\pi \cdot n$

Since the motor is running at a constant speed ω_m and $\theta_e = \theta_m$, the angular displacement is

$$\theta_e = \omega_m \cdot t \quad \text{Eq. 1-3}$$

To solve this sinusoidal equation, the phasors theory can be used.

A sinusoidal voltage e(t) and its phasor $\overline{E}$ are related by

$$e(t) = Re[\overline{E} \cdot e^{j\omega \cdot t}] = E \cdot \cos(\omega \cdot t + \theta) \quad \text{Eq. 1-4}$$

where E and θ are, respectively, the magnitude and the angle of $\overline{E}$. Using the phasors, the voltage equation becomes ([3])

$$\overline{V}_n + \overline{E}_n = (R + j\omega L)\overline{I} \quad \text{Eq. 1-5}$$

where

$$\overline{V}_n = V_s \angle\left(\alpha_n - \frac{\pi}{2}\right) \quad \text{Eq. 1-6}$$

and

$$\overline{E}_n = -K_e \omega_n \angle\left(\alpha_n - \frac{\pi}{2}\right) \quad \text{Eq. 1-7}$$

Solving for $\overline{I}_n$

$$\overline{I}_n = \frac{V_s - K_e \omega_m}{\sqrt{R^2 + (\omega_m L)^2}} \angle\left(\alpha_n - \frac{\pi}{2} - \tan^{-1}\frac{\omega_m L}{R}\right) \quad \text{Eq. 1-8}$$

The instantaneous current i_n (t) is:

$$i_n(t) = I_o \sin(\omega_m t + \alpha_n - \beta) \quad \text{Eq. 1-9}$$

where

$$I_o = \frac{V_s - K_e\omega_m}{\sqrt{R^2 + (\omega_m L)^2}}$$

and

$$\beta = \tan^{-1}\frac{\omega_m L}{R}$$

Using the above equations in the torque equation

$$T = -K_t \sum_{n=0}^{2} I_n \cdot \sin(\omega_m t + \alpha_n) \qquad \text{Eq. 1-10}$$

the following expression results

$$T = -K_t I_o \sum_{n=0}^{2} \sin(\omega_m t + \alpha_n - \beta)\sin(\omega_m t + \alpha_n) \qquad \text{Eq. 1-11}$$

Using some trigonometric manipulation to the above equation, the following expression for the torque is derived

$$T = -\frac{3}{2}K_t I_o \cos\beta \qquad \text{Eq. 1-12}$$

Now

$$\cos\beta = \cos\left(\tan^{-1}\frac{\omega_m L}{R}\right) = \frac{R}{\sqrt{R^2 + (\omega_m L)^2}}$$

Substituting the expression for I_o and $\cos\beta$ in Eq. 1-12, the final torque equation results

$$T = \frac{3K_t R(V_S - K_e\omega_m)}{2[R^2 + (\omega_m L)^2]} \qquad \text{Eq. 1-13}$$

The above equation give the torque-speed curve and it's obviously non-linear. However, if the inductive term in the Eq. 1-13 is small enough to be neglected, the torque-speed becomes linear just like the DC motor.

1.3 Dynamic Model of a DC Motor

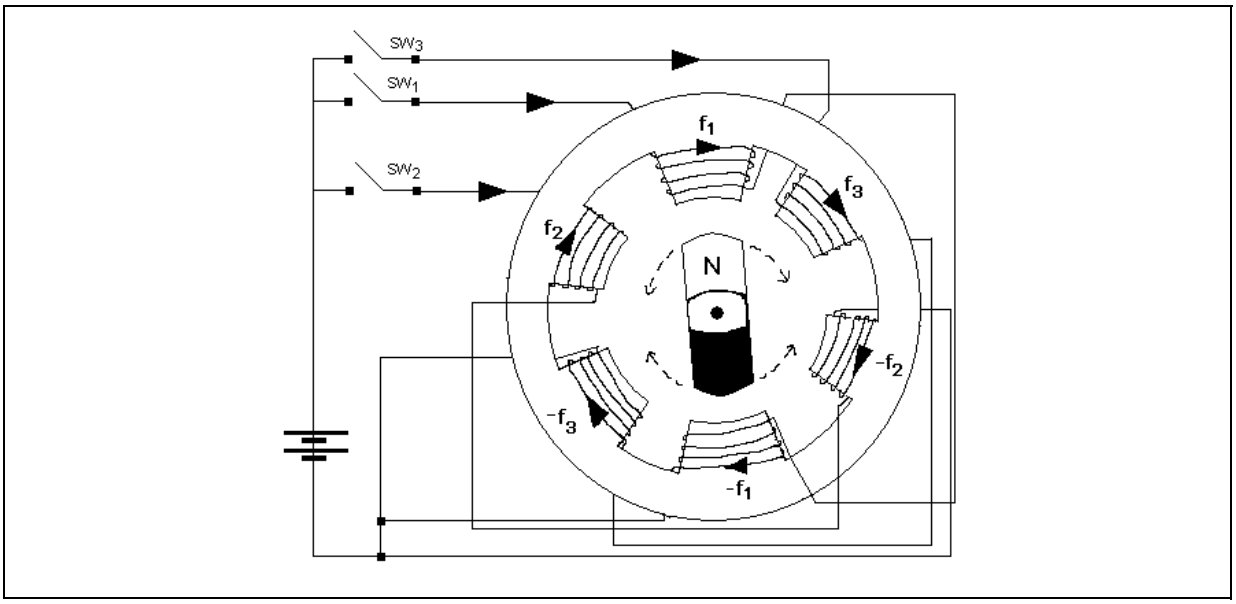
1.3.1 Physical Description

A brushless dc motor is a mechanism that converts electrical power to mechanical power via magnetic coupling. The electrical power is provided by a voltage source, while a spinning rotor provides the mechanical power.

A brushless dc motor consists of three main components: stator, rotor and sensors. Figure 1.1 shows a cross-sectional view of an idealized three-phase brushless dc motor. It has six stator poles with opposite poles being the same phases around which coils are wound. The winding in each pole is called phase winding, signifying that current in each winding is independently excited and offset by an appropriate electrical angular displacement. The idealized motor is a three-phase motor since it has three independent windings. The rotor is a permanent magnet with opposite poles facing the gap.

Classical brushless dc motors have also sensors, required to monitor the position of the rotor. In the Spindle motors, in place of the sensors, position information is obtained by processing the back emf.

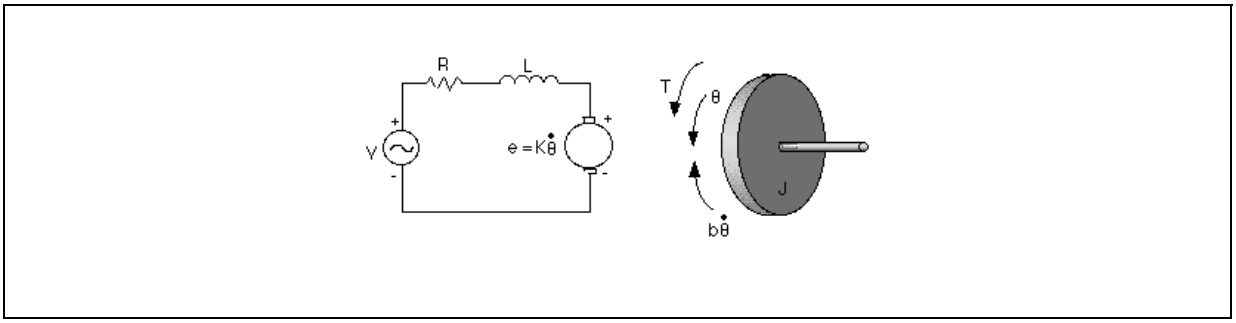
Figure 1.1. Cross section of an ideal brushless DC Motor.



1.3.2 Electrical Characteristics

The equivalent electrical circuit of a dc motor is illustrated in Figure 1.2. It can be represented by a voltage source (V) across the coil of the armature. The electrical equivalent of the armature coil can be described by an inductance (L) in series with a resistance (R) in series with an induced voltage (e), which opposes the voltage source. The induced voltage is generated by the rotation of the permanent magnets through the fixed flux lines of the electrical coil. This voltage is often referred to as the back emf (electromotive force).

Figure 1.2. Electrical representation of a DC motor



Using Kirchoff's voltage law around the electrical loop can derive a differential equation for the equivalent circuit. Kirchoff's voltage law states that the sum of all voltages around a loop must equal zero, or

$$V - V_R - V_L - e = 0 \qquad \text{Eq. 1-14}$$

According to Ohm's law, the voltage across the resistor can be represented as

$$V_R = R \cdot i \qquad \text{Eq. 1-15}$$

where i is the armature current. The voltage across the inductor is proportional to the change of current through

the coil with respect to time and can be written as

$$V_L = L \cdot \frac{di}{dt} \quad \text{Eq. 1-16}$$

where L is the inductance of the armature coil. Finally, the back emf can be written as

$$e = K_e \cdot \omega \quad \text{Eq. 1-17}$$

where K_e is the velocity constant determined by the flux density of the permanent magnets, the reluctance of the iron core of the armature, and the number of turns of the armature winding. $\omega = \theta$ is the rotational velocity of the armature.

Substituting Eq. 1-15, Eq. 1-16 and Eq. 1-17 into Eq. 1-14 gives the following differential equation:

$$V - R \cdot i - L \cdot \frac{di}{dt} - K_e \cdot \omega = 0 \quad \text{Eq. 1-18}$$

1.3.3 Mechanical Characteristics

Performing an energy balance on the system, the sum of the torques of the motor must equal zero. Therefore,

$$T_e - T_{\omega'} - T_\omega - T_L = 0 \quad \text{Eq. 1-19}$$

where T_e is the electromagnetic torque, $T_{\omega'}$ is the torque due to rotational acceleration of the rotor, T_ω is the torque produced from the velocity of the rotor, and T_L is the torque of the mechanical load. The electromagnetic torque is proportional to the current through the armature winding and can be written as

$$T_e = K_t \cdot i \quad \text{Eq. 1-20}$$

where K_t is the torque constant and like the velocity constant is dependent on the flux density of the fixed magnets, the reluctance of the iron core, and the number of turns in the armature winding. $T_{\omega'}$ can be written as

$$T_{\omega'} = J \cdot \frac{d\omega}{dt} \quad \text{Eq. 1-21}$$

where J is the inertia of the rotor and the equivalent mechanical load. The torque associated with the velocity is written as

$$T_\omega = B \cdot \omega \quad \text{Eq. 1-22}$$

where B is the damping coefficient associated with the mechanical rotational system of the machine.

Substituting Eq. 1-20, Eq. 1-21 and Eq. 1-22 into Eq. 1-19 gives the following differential equation:

$$K_t \cdot i - J \cdot \frac{d\omega}{dt} - B \cdot \omega - T_L = 0 \quad \text{Eq. 1-23}$$

1.3.4 Transfer Function Block Diagram

The differential equations given in Eq. 1-18 and Eq. 1-23 for the armature current and angular velocity can be written as

$$\frac{di}{dt} = -\frac{R}{L} \cdot i - \frac{K_e}{L} \cdot \omega + \frac{V}{L}$$ Eq. 1-24

$$\frac{d\omega}{dt} = \frac{K_t}{J} \cdot i - \frac{B}{j} \cdot \omega - \frac{T_L}{J}$$ Eq. 1-25

Applying the Laplace transform to each of the equations results

$$s \cdot I(s) - i(0) = -\frac{R}{L} \cdot I(s) - \frac{K_e}{L} \cdot \Omega(s) + \frac{1}{L} \cdot V(s)$$ Eq. 1-26

$$s \cdot \Omega(s) - \omega(0) = -\frac{K_t}{J} \cdot I(s) - \frac{B}{J} \cdot \Omega(s) - \frac{1}{J} \cdot T_L(s)$$ Eq. 1-27

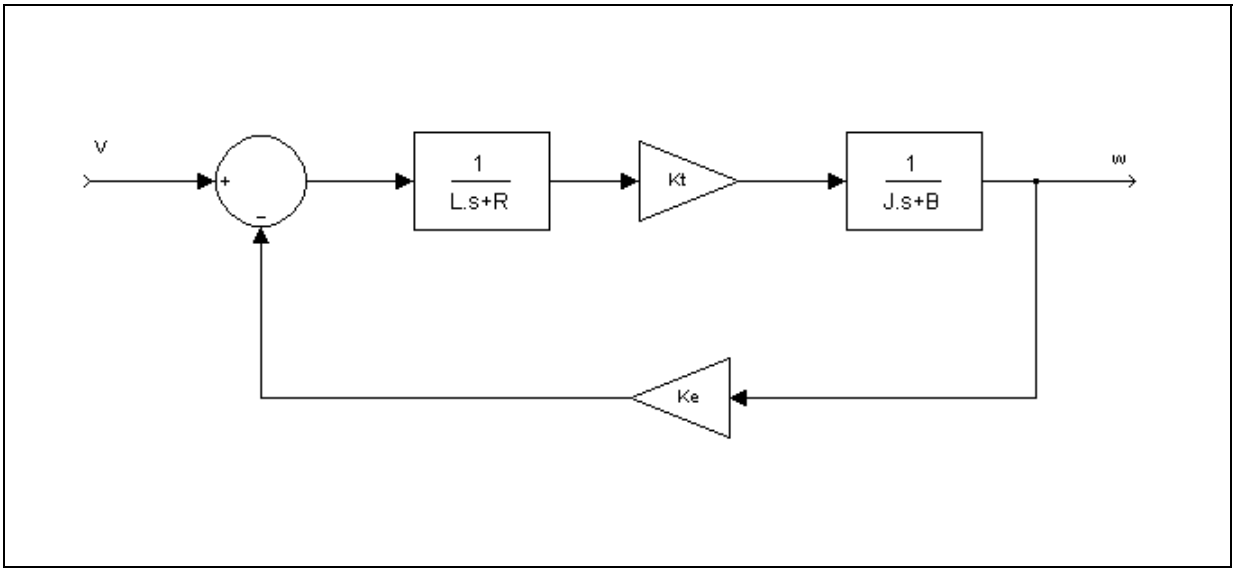
If perturbations around some steady state value are considered, initial conditions go to zero and all the variables become some change around a reference state and the equations can be expressed as follows:

$$I(s) = \frac{-K_e \cdot \Omega(s) + V(s)}{L \cdot s + R}$$ Eq. 1-28

$$\Omega(s) = \frac{-K_t \cdot I(s) - T_L(s)}{J \cdot s + B}$$ Eq. 1-29

In the case of a sun tracking servo system, the only load torque to be concerned with is the friction in the system, which is relatively constant while the motor is moving. Since the change in T_L is zero, it does not need to appear in the block diagram. Also, if one only focuses on the angular velocity as the response of interest, the block diagram becomes as shown in Figure 1.3.

Figure 1.3. Block diagram of the DC motor (driven in voltage mode) as modeled in this study



This block diagram is then reduced, using block diagram algebra, to an overall transfer function between the output angular velocity and input applied voltage. The equation can be expressed as follows:

$$\frac{\Omega(s)}{V(s)} = \frac{K_t}{(J \cdot L) \cdot s^2 + (J \cdot R + L \cdot B) \cdot s + (R \cdot B + K_t \cdot K_e)} \quad \text{Eq. 1-30}$$

1.4 Frequency Analysis of the System

Before starting to design any kind of speed control scheme, it's a good rule to perform a frequency analysis of the plant to better understand the kind of controller needed.

For this purpose an example in which the parameters are picked up from the technical manual of a 5400rpm hard disk drive is presented.

EXAMPLE 1.1

Draw the Bode plot of the open loop transfer function of a spindle motor with the following physical parameters:

Back emf constant K_e = 0.091 V · s

Torque constant Kt = 147.86 g · cm/A

Winding inductance L = 500µH

Winding resistance R = 1.8Ω

Inertia J = 0.732 g · cm · s^2

Damping coefficient B = 0.36 g · cm · s

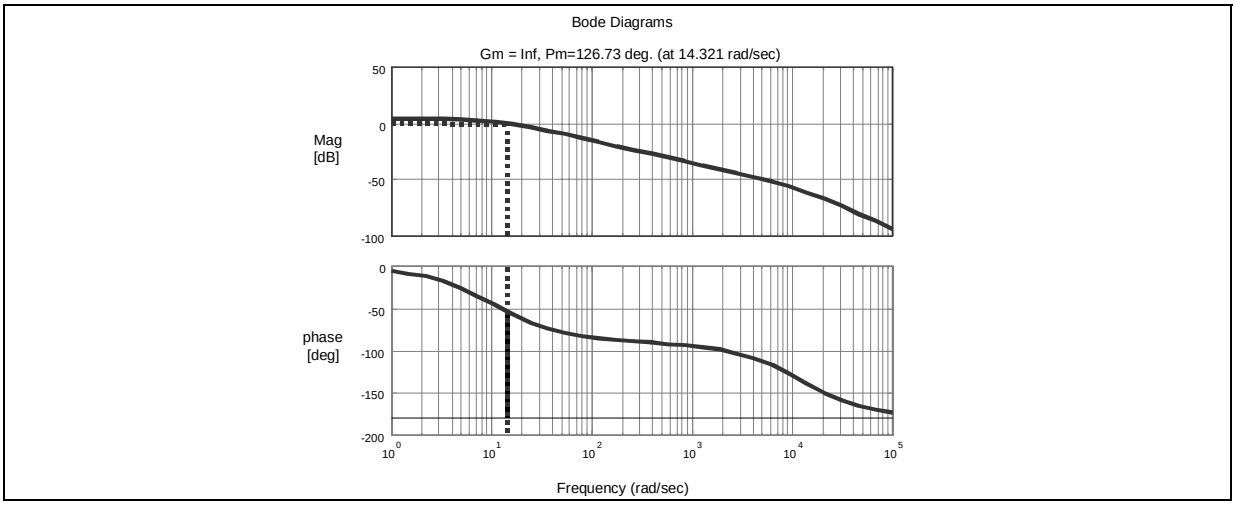
and determine the stability margin.

Solution. Substituting the above values in the Eq. 1-30, the following transfer function results:

$$G(s) = \frac{147.86}{0.0001098 \cdot s^2 + 1.318 \cdot s + 14.1}$$

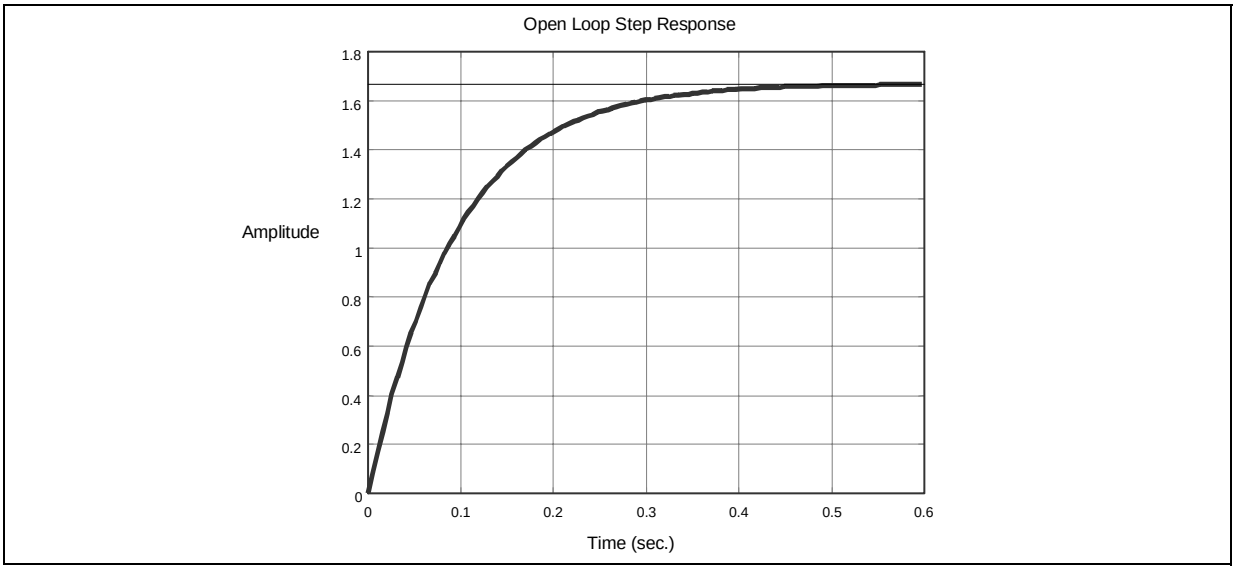
The Bode diagrams are the following

Figure 1.4. Bode Diagrams



The magnitude plot shows there is a pole at very low frequency and another one at very high frequency (there is a 40db/decade attenuation). The phase margin sketch in the phase plot is very good. Below the open loop step response is plotted

Figure 1.5. Open-Loop step response



The above graphic shows that applying a unity step input, the output present a quite large steady state error.

1.5 Conclusions

The mathematical equations of the spindle's model have been derived in the previous section.

The example presented above is very important because it is represent the priori knowledge about the plant that has to be controlled.

Figure 1.5. shows that a regulator is needed in order to make the system's output track the desired input without steady state errors. In the next chapter some techniques to control the above plant will be studied.

2 DESIGN OF THE ANALOG FLL FREQUENCY LOCKED LOOP

2.1 Overview

In this chapter some techniques to accomplish speed control of the spindle motor will be presented. As the spindle is a brushless motor without output sensors, the output signal is processed starting from the back emf.

Figure 2.1. shows the Frequency Locked Loop used to control the spindle's speed.

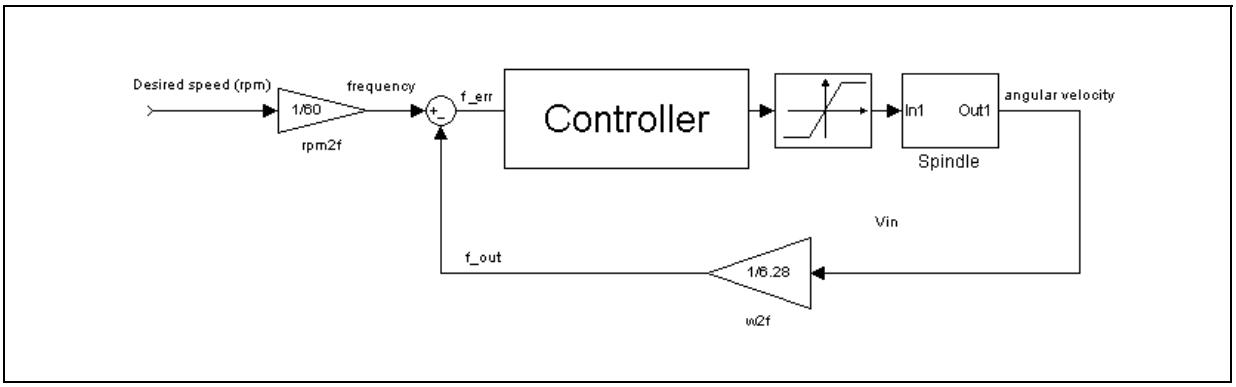
Since the most basic requirement of a spindle motor is that it should rotate at the desired speed, the steady-state error of the motor should be very low (less than 0.1%). One more requirement is that the motor must accelerate to its steady-state speed as soon as it turns on. In this case it is necessary to have the settling time less than 5 seconds. Since a speed faster than the reference may damage the equipment, it is also necessary the overshoot is less than 5%.

The desired requirement can be summarized as follows:

1. Steady-state error of the motor speed less than 0.1%
2. Settling time lass than 5 seconds
3. Overshoot less than 5%

In order to satisfying the first requirement, a pure integrator will be insert in the controller, moreover a proportional gain will be inserted in order to reduce the rise time. This first configuration will be analyzed in section 2.2, while in section 2.2.1 a more complicated controller including a lead-lag filter (to improve system stability) will be presented.

Figure 2.1. Frequency Locked Loop block diagram



2.2 Controller Design

The meaning of the controller is to make the system fast react to speed perturbation maintaining the stability. As discuss in previous section, the requirement to make the steady-state error very low is the why an integrator is needed.

EXAMPLE 2.1

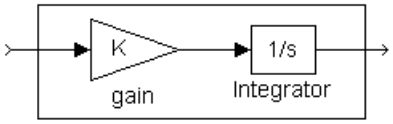
Based on the control scheme sketched in Figure 2.1. and the system parameters introduced in Example 1.1, design a controller that satisfy the following requirements:

- Steady-state error of the motor speed less than 0.1%
- Settling time lass than 5 seconds
- Overshoot less than 5%

Plot the step response of the closed loop system and verify how the system reacts to Torque perturbances.

Solution. The first controller presented is sketched below:

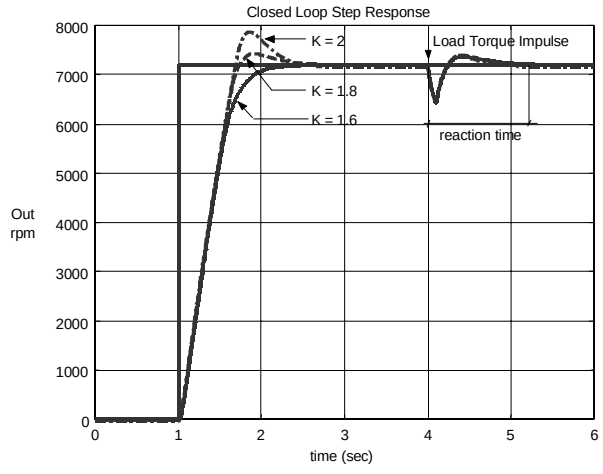
Figure 2.2. Gain-Integrator controller



The above controller allows the user to change the gain K, in order to change the system dynamics.

Next Figure shows the closed loop step response respect to K variations.

Figure 2.3. Step response of the system with the PI controller.



The above plot shows that the gain modifies the system's characteristics (especially rise time and overshoot).

At time 4 seconds, an impulsive disturbance in the load torque has been generated.

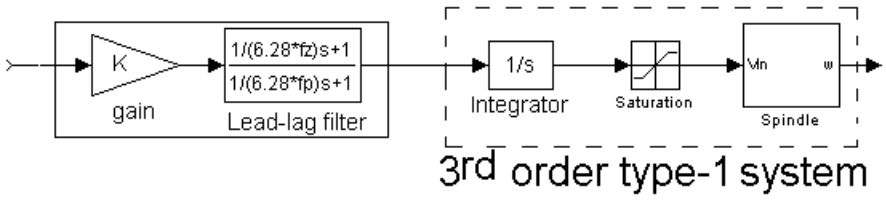
The controller described in the previous example satisfies the requirements. However it is present weak in noise rejection. In many applications it should be desirable to make the recovery time (time the controller take to make the output equal to the input, after a perturbation has fired) as small as possible.

In order to reduce the recovery time, a more complicated controller will be designed.

The new controller is made by the series connection of the previous controller with a filter called Lead-Lag filter.

To analyze the behavior of this filter, the integrator is considered part of the plant. In such a way the plant becomes a 3rd order type-1 system (three poles, one of them in the origin), as shown below:

Figure 2.4.



2.2.1 Lead Compensation

The main purpose of the compensator is to increase the phase (or lead) still occurs, keeping the amplification, at high frequencies, limited. The resulting lead compensation has a the following transfer function

$$C(s) = K \cdot \frac{\tau_z \cdot s + 1}{\tau_p \cdot s + 1} \qquad \text{Eq. 2-1}$$

where $\tau_p < \tau_z$. It is often useful to fix the zero-pole displacement and express the Eq. 2-1 in the following way

$$C(s) = K \cdot \frac{\tau_z \cdot s + 1}{\alpha \cdot \tau_z \cdot s + 1} \qquad \text{Eq. 2-2}$$

where $\alpha=\tau_p/\tau_z$ and $t = 1/2\pi f$. Figure 2.5. shows the frequency response of this lead compensation.

Figure 2.5. Lead compensation frequency response with $1/\alpha = 10$

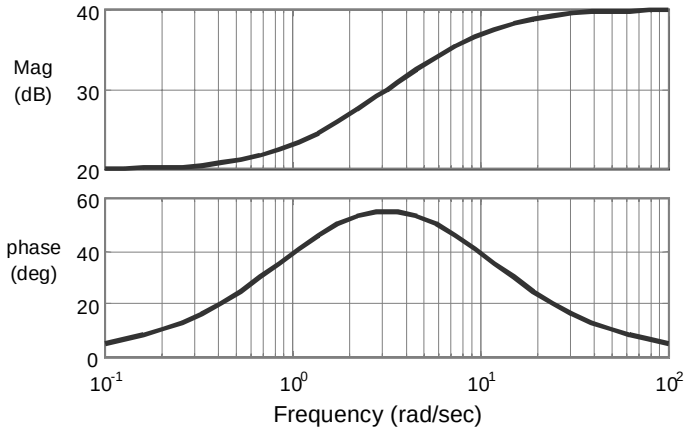


Figure 2.5. shows that a significant amount of phase lead is provided with less amplification at high frequencies.

To understand how the compensator will improve the overall system performances, the system introduced in the example 1.1 will be considered with the addiction of a pole in the origin making the system a 3rd order type I system.

EXAMPLE 2.2

Consider the system introduced at page 7. A pure integrator is connected in series, giving the following system

$$G(s) = \frac{147.86}{0.0001098 \cdot s^3 + 1.318 \cdot s^2 + 14.1 \cdot s}$$

Design a lead compensator for the above system.

Compare the noise rejection performances of the designed control system respect to the system designed in Example 2.1.

Solution. The plant is a 3rd order type I system that means there are three poles with one of them in the origin.

The Eq. 2-2 gives the transfer function of the lead filter.

To understand where to place the pole ($f_p = 1/2\pi\tau_z$) and the zero ($f_z = 1/2\pi\tau_z$) of the compensator, the following graphs can be useful.

Figure 2.6. Open Loop TF of the system (Filter+Plant) Vs changing in K (fz and fp constants)

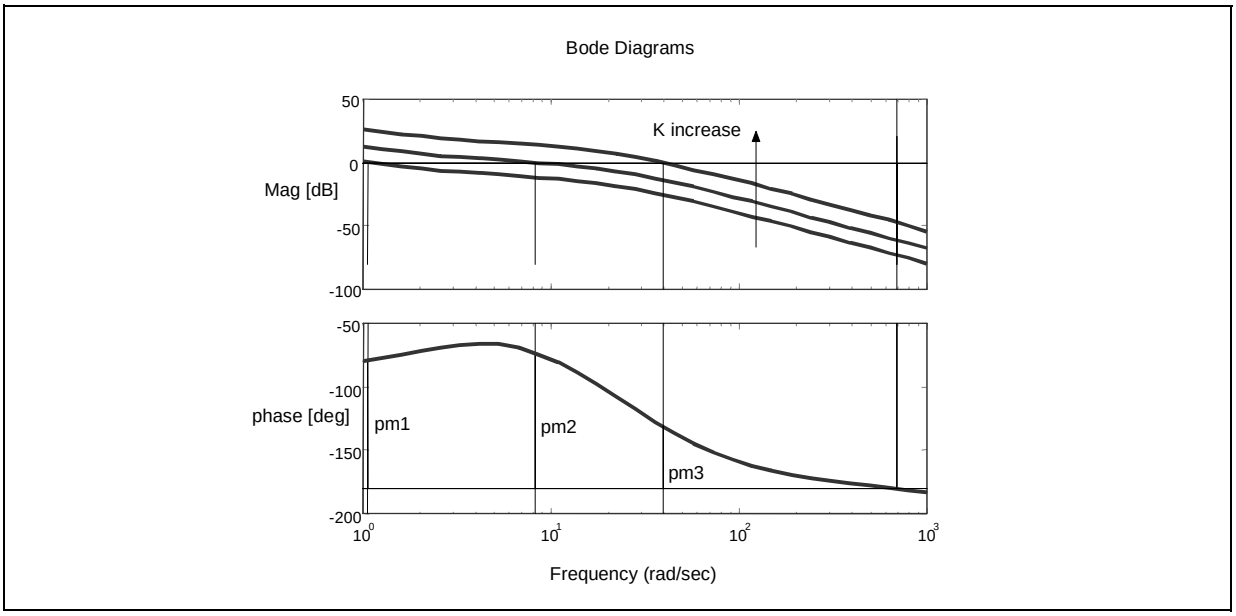


Figure 2.7. Filter TF Vs changing in fz (K and fz/fp constants)

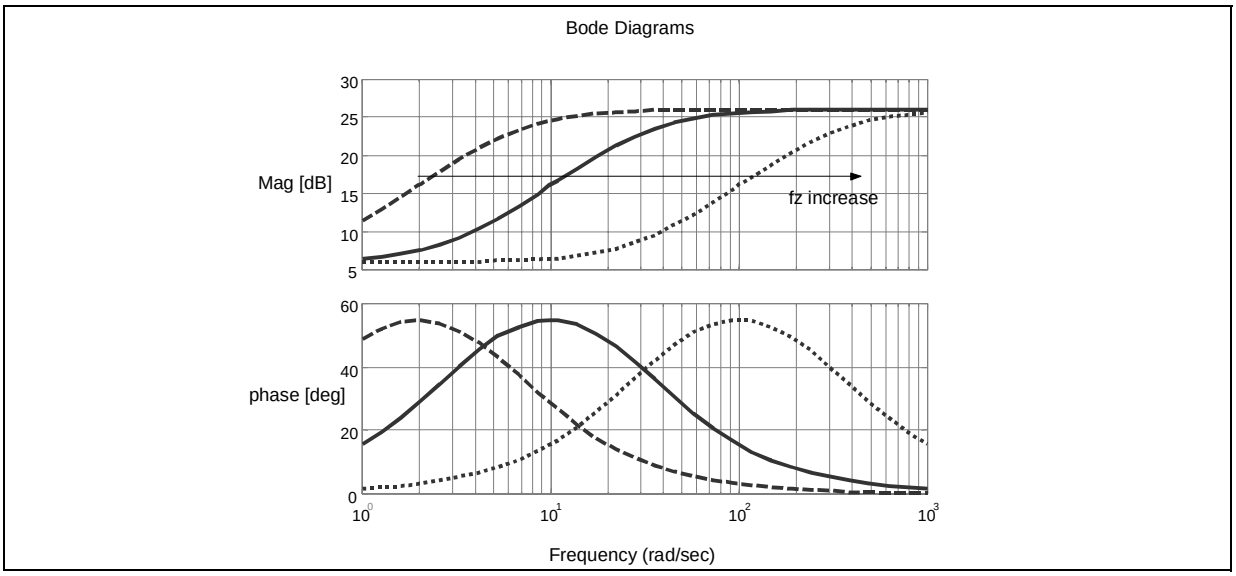


Figure 2.8. Open Loop TF of the system (Filter+Plant) Vs changing in fz (K and fz/fp constants)

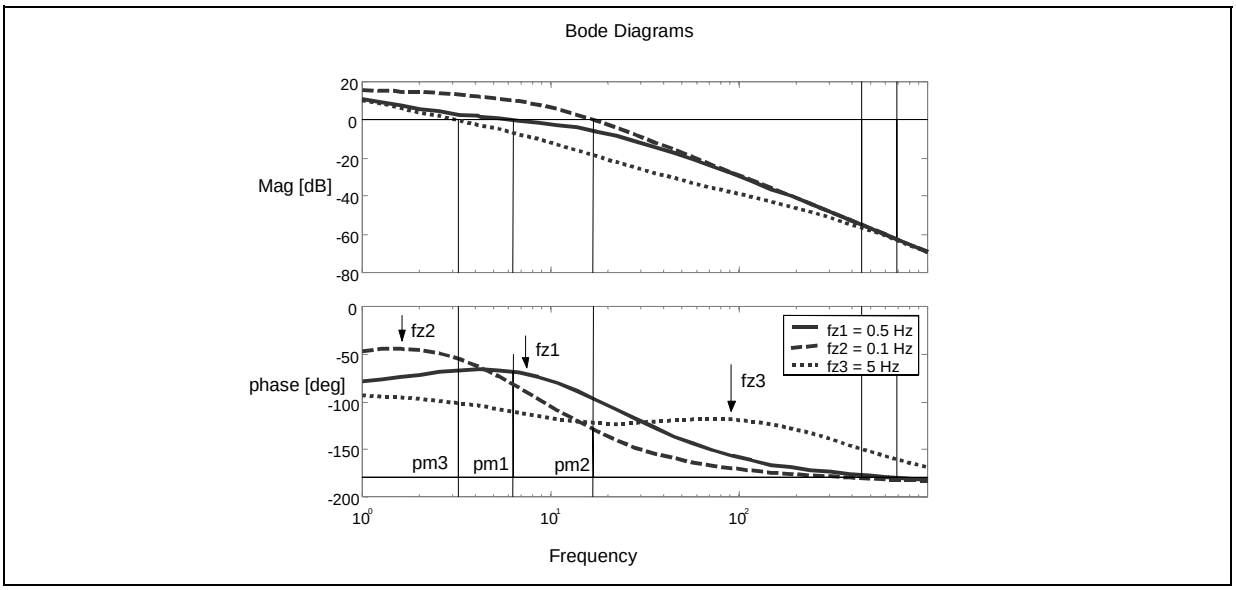


Figure 2.9. Closed loop step response comparison.

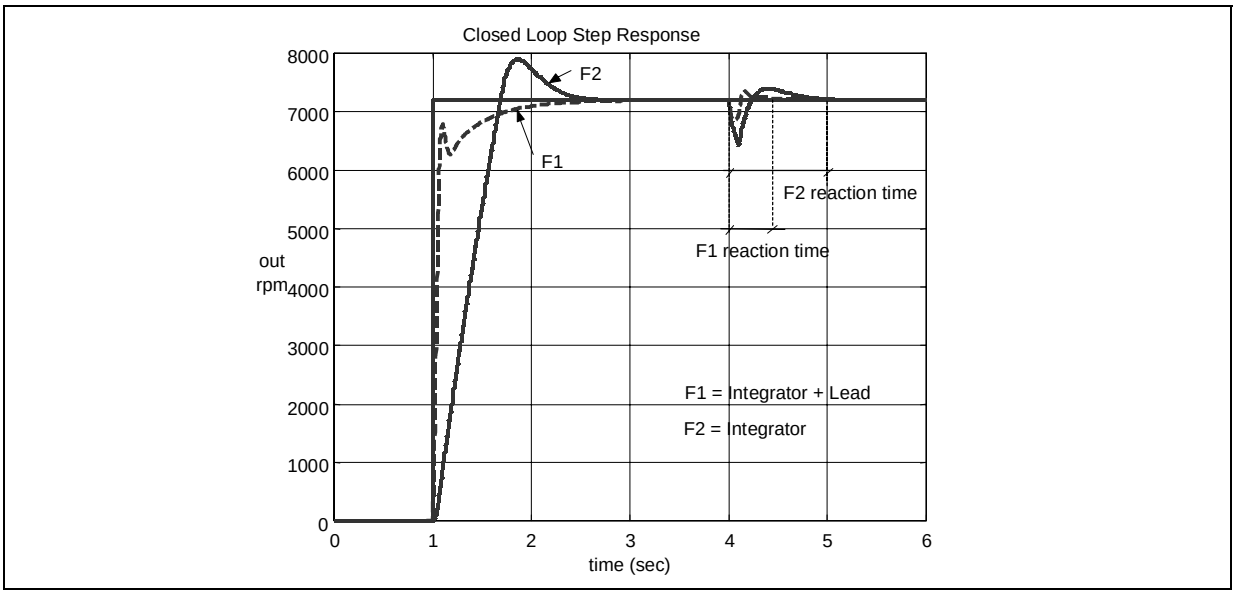


Figure 2.6. shows that the gain of the controller does not affect the phase plot of the system. Increasing the gain results in moving up the Magnitude plot. Consequently the crossover frequency moves to the right and the phase margin too (solid line on the phase plot).

The phase plot of Figure 2.7. shows that increasing or decreasing the frequency of the zero in the filter, keeping the zero-pole displacement and the gain constants, corresponds to move the phase shape respectively, to the right or to the left.

In order to get more stability, the center of the phase shape should be placed where the crossover frequency of the system is. In such a way the overall phase margin grow up due to the lead effect of the filter.

Figure 2.8. shows how the zero frequency of the filter change the phase shape of the overall open loop transfer function. Frequency F1 should be kept to get more stability. After some manual adjustments the following FLL filter's parameters results:

$$K = 0.25$$

$$f_z = 0.5\text{Hz}$$

$$f_p = 5\text{Hz}$$

Figure 2.9. compares the step responses of the system with and without Lead Filter. An impulse torque load noise has been added (at time t = 4 sec) to compares the noise rejection performances.

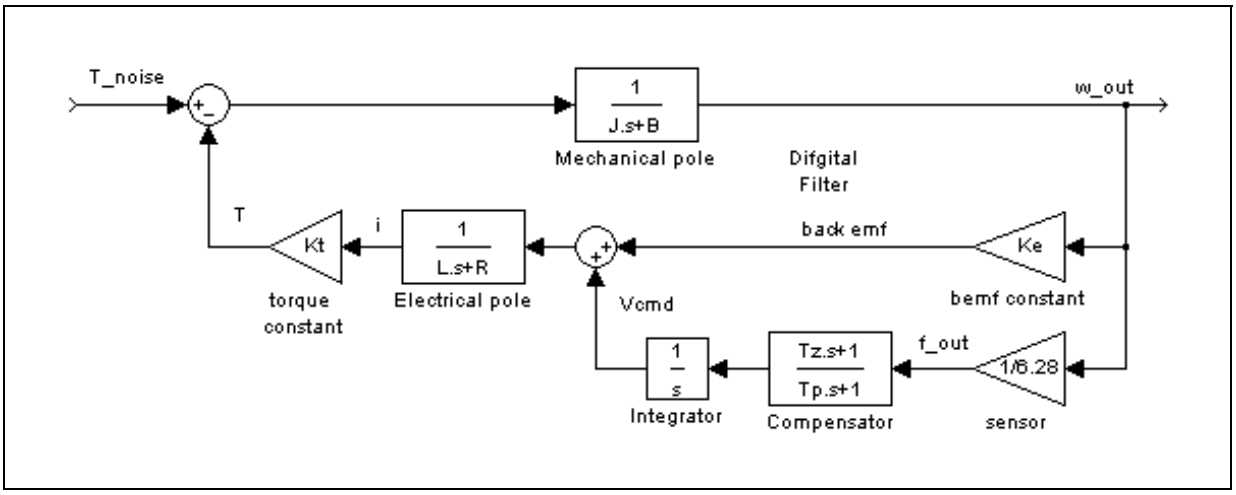
2.3 Noise Rejection

An important issue in control design is the ability of the overall compensated system to reject some kind of noise that gets into the system (e.g. noise due to the sensor or to the actuator).

In the hard disk spindle speed control it is important to avoid the disturbances on the torque, for example due to the oscillations of the hard disk's head. By the way, if it is not possible to avoid them, the compensator has to be able to reject them as fast as possible avoiding degradations of the dynamic performances.

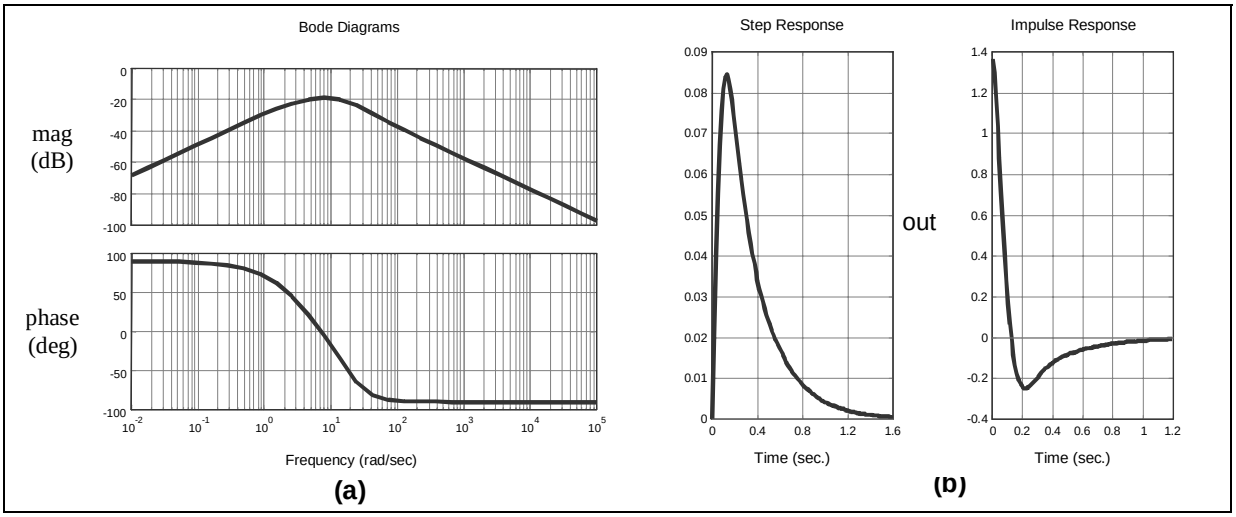
To view the noise rejection performances of the FLL control scheme presented in previous examples, the Figure 2.1. is now redraw focusing the relationship between the input torque noise and the output speed.

Figure 2.10. Block diagram of the compensated system focusing the relationship between input torque noise and output speed.



From the above block scheme it is possible to derive the bode plots of the transfer function between the input noise and the output speed (Figure 2.11.a). Figure 2.11.b shows the step and impulse responses. It can be seen that both the step and impulse noises in the load torque are suppressed as desired.

Figure 2.11. Input torque noise to output speed Transfer Function. (a) Bode plots, (b) step and impulse responses



2.4 Conclusions

Figure 2.9. shows that the Lead Filter is better than the pure integrator in term of recovery time.

The requirement of fast recovery time is becoming one of the main requirements of the last hard disk applications. Due to increasing of DPI (track density per inch), using of faster spindle combined with hard disk supported software (for example using HD as virtual memory in Windows operating system) it is important to recover the desired speed as fast as possible if some perturbations arise. In some applications like laptop computers or smart cards for portable devices (e.g. digital camera), perturbations of the load or suddenly accelerations are common noises that a well-designed controller has to reject quickly.

For the above reasons the FLL filter described in Example 2.2 should be used.

3 DIGITIZATION - DESIGN OF THE DIGITAL CONTROL FLL

3.1 Overview

In the previous chapter an analog controller to allow speed spindle control has been designed. The next step is to derive a digital filter that allows the same control.

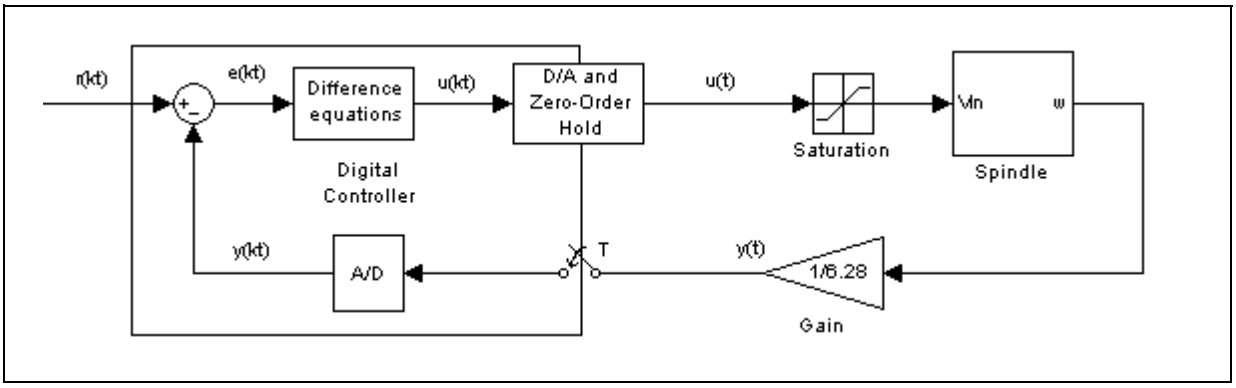
Many techniques can be used, but the one presented it is called emulation. This technique procedes through the following steps:

- Designing of a continuous compensator (as described above)
- Digitizing the continuous compensation

Figure 2.1. at page 9 shows the topology of a typical continuous control system. The computation of the error signal and the dynamic compensation can all be accomplished in a digital computer as shown in Figure 3.1.

The most important differences between the two implementations are that the digital system operates on samples of the sensed spindle output rather than on the continuous signal and that the dynamics represented by the analog controller must be approximated by algebric recursive equations.

Figure 3.1. Block diagram for a basic control system with a digital computer



It is fundamental to know that the most important impact of implementing a control system digitally is the delay associated with the Hold[1]. This issue will be explained further.

The analog controller has been studied in the previous chapter. Now some issues in digital control design will be presented.

Note: 1. The result of the difference equations is a discrete u(kT) at each sample instant. This signal is converted to a continuos u(t) by the digital-to-analog (D/A) converter and the hold: the D/A changes the binary number to an analog voltage and a zero-order hold (ZOH) maintains that same voltage throughout the sample period.

3.2 Bilinear Transform

This technique approaches the problem as one of numerical integration. Suppose

$$\frac{U(s)}{E(s)} = D(s) = \frac{1}{s} \quad \text{Eq. 3-1}$$

which is integration. Therefore,

$$(kT) = \int_{0}^{kT-T} e(t)dt + \int_{kT-T}^{kT} e(t)dt \quad \text{Eq. 3-2}$$

which can be written as

$$u(kT) = u(kT - T) + \text{area under } e(t) \text{ over last } T \quad \text{Eq. 3-3}$$

where T is the sample period.

For bilinear approximation (also called Tustin's method), the task at each step is to use trapezoidal integration that is, to approximate e(t) by a straight line between the two samples (Figure 3.2.).

Writing u(kT) as u(k) and u(kT – T) as u(k – 1) for short, Eq. 3.3 become

$$u(k) = u(k-1) + \frac{T}{2}[e(k-1) - e(k)], \quad \text{Eq. 3-4}$$

Or, taking the z-transform,

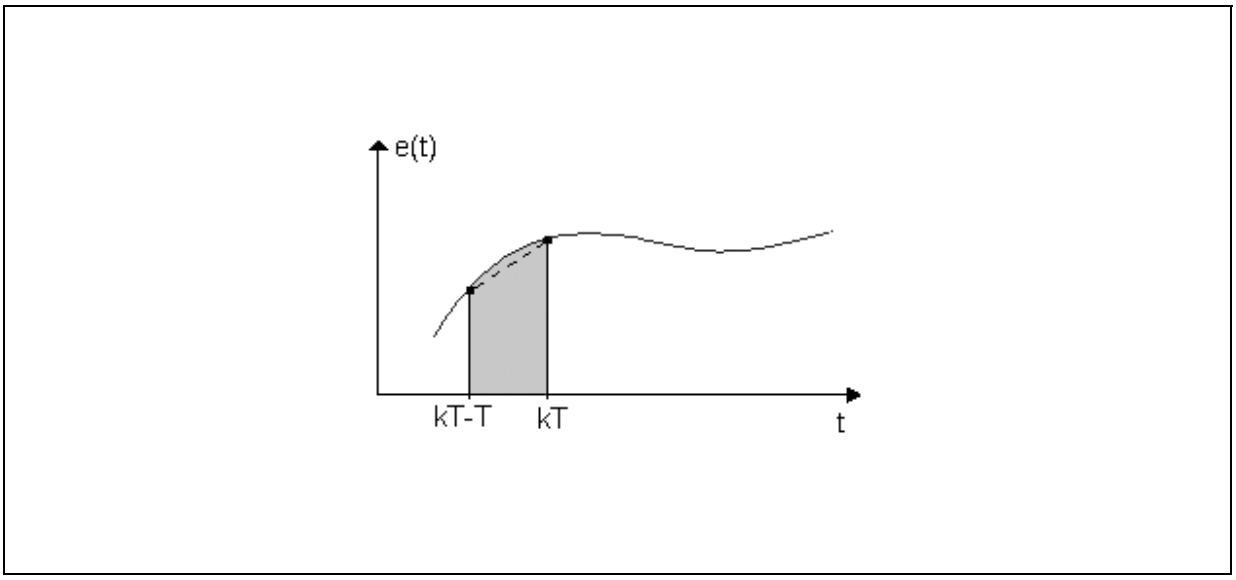
$$\frac{U(z)}{E(z)} = \frac{T}{2} \cdot \frac{1 + z^{-1}}{1 - z^{-1}} \quad \text{Eq. 3-5}$$

In fact, given a continuous transfer function D(s) and substituting

$$s = \frac{2}{T} \cdot \frac{1 - z^{-1}}{1 + z^{-1}} \quad \text{Eq. 3-6}$$

for every occurrence of s yields a D(z) based on the trapezoidal integration formula.

Figure 3.2. Trapezoidal integration



EXAMPLE 3.1

Use the bilinear transform to obtain a first order IIR filter from the following analog filter,

$$C(s) = K \cdot \frac{\tau_Z s + 1}{\tau_P s + 1} \qquad \text{Eq. 3-7}$$

The sampling period for the digital filter is T seconds. Express the digital filter's coefficients as functions of the analog filter's parameters.

Solution. $\tau_x = 1/2\pi f_x$ is the time constant. Eq. 3-7 can be written, for convenience, as follows

$$C(s) = K\frac{f_P s + 2\pi f_Z}{f_Z s + 2\pi f_P} \qquad \text{Eq. 3-8}$$

Substituting Eq. 3-6 into the Eq. 3-8

$$C(z) = C(s)\Big|_{s = \frac{T}{2}\cdot\frac{1-z^{-1}}{1+z^{-1}}} = K\frac{f_P(1+\pi f_Z T)-(1-\pi f_Z T)z^{-1}}{f_Z(1+\pi f_P T)-(1-\pi f_P T)z^{-1}} =$$

$$= K\frac{f_P}{f_Z(1+\pi f_P T)}\frac{(1+\pi f_Z T)-(1-\pi f_Z T)z^{-1}}{1-\frac{(1-\pi f_P T)}{(1+\pi f_P T)}z^{-1}} = \frac{b_0 - b_1 z^{-1}}{1 - a_1 z^{-1}}$$

where

$$a_1 = \frac{1-\pi f_P T}{1+\pi f_P T}$$

$$b_0 = K\frac{f_P}{f_Z}\frac{1-\pi f_Z T}{1+\pi f_P T}$$

$$b_1 = K\frac{f_P}{f_Z}\frac{1+\pi f_Z T}{1+\pi f_P T}$$

are the digital filter's coefficients.

The above digital filter can be implemented using the following difference equation:

$$\begin{cases} y(k) = a_1 \cdot y(k-1) + b_0 \cdot x(k) - b_1 \cdot x(k-1) \\ y(-1) = 0 \qquad\qquad k = 0,1...,n \\ x(-1) = 0 \end{cases}$$

Sometimes (if few bits are available to store the filter's coefficients) a scaling factor λ can be applied having a better resolution. This issue is explained in the next section.

Below are the results

$$A_0 = \lambda \qquad A_1 = \lambda \cdot a_1$$

$$B_0 = \lambda \cdot b_0 \qquad B_1 = \lambda \cdot b_1$$

In Mozart™ internal FLL digital filter (in which the coefficients have ten bits length) λ has been set to 16.

3.3 Scaling

Some applications have the FLL integrated in the microchip. In this case the internal representation of the digital filter's coefficients is a fixed-point integer representation. To better understand this issue the Mozart internal FLL is considered.

The internal representation of the digital filter's coefficients is a 10-bit fixed-point integer representation. This means that only integer values can be stored and for decimal numbers, the ROUND function will be applied.

Below, the generic digital filter formula is presented

$$D(z) = \frac{b_0 + b_1 z^{-1}}{1 + a_1 z^{-1}} \qquad \text{Eq. 3-9}$$

The results do not change if the Eq. 3.9 is multiplied and divided by the same factor:

$$D(z) = \frac{K}{K}\frac{b_0 + b_1 z^{-1}}{1 + a_1 z^{-1}} = \frac{\bar{b}_0 + \bar{b}_1 z^{-1}}{K + \bar{a}_1 z^{-1}}$$

Consider the effect of prescaling the b_0 coefficient. Suppose the real value for that coefficient is 2.49.

If no prescaling is performed, the value stored on the coefficient register is

$$\tilde{b}_0 = ROUND(b_0) = ROUND(2.49) = 2$$

and the percentage error is

$$Error_{\%} = \frac{b_0 - \tilde{b}_0}{b_0} \cdot 100 = 19.68\%$$

Now a prescaling with a factor K=10 is applied.

$$\bar{b}_0 = 10 \cdot b = 24.9$$

With this scaling, the value stored on the coefficient register is

$$\tilde{b}'_0 = ROUND(\bar{b}_0) = ROUND(24.9) = 24$$

After all the coefficients have been scaled, everything goes like the coefficient used in the filter equations was

$$\hat{b}_0 = \frac{\tilde{b}'}{10} = 2.4$$

and the error is

$$Error'_{\%} = \frac{b_0 - \hat{b}_0}{b_0} = 3.61\%$$

It can be seen that a prescaling factor of ten allows a better resolution on the filter coefficients.

Often, it is preferable to scale the coefficients with a factor that is a power of 2, to easily compute the multiplication (the factor 16 is better than 10, because it is correspond to a 4 bit shift left).

3.4 Applicability Limits

Performing an exact discrete analysis or a simulation of a system and determining the digitization for a wide range of sample rates, it can be seen that the system would typically be unstable for rates slower than approximately $5\omega_n$ (five times the cross-over frequency) and the damping would be substantially degraded for rates slower than $10\omega_n$. At sample rates on the order of $20\omega_n$, design by emulation yields reasonable results and can

be used with confidence for sample rates of 30 times the bandwidth.

The errors come about the technique ignores the lagging effect of the Zero-Order Holder, which, on the average, is T/2 sec. A method to account for this is to approximate the T/2 delay including a transfer function approximation for the ZOH:

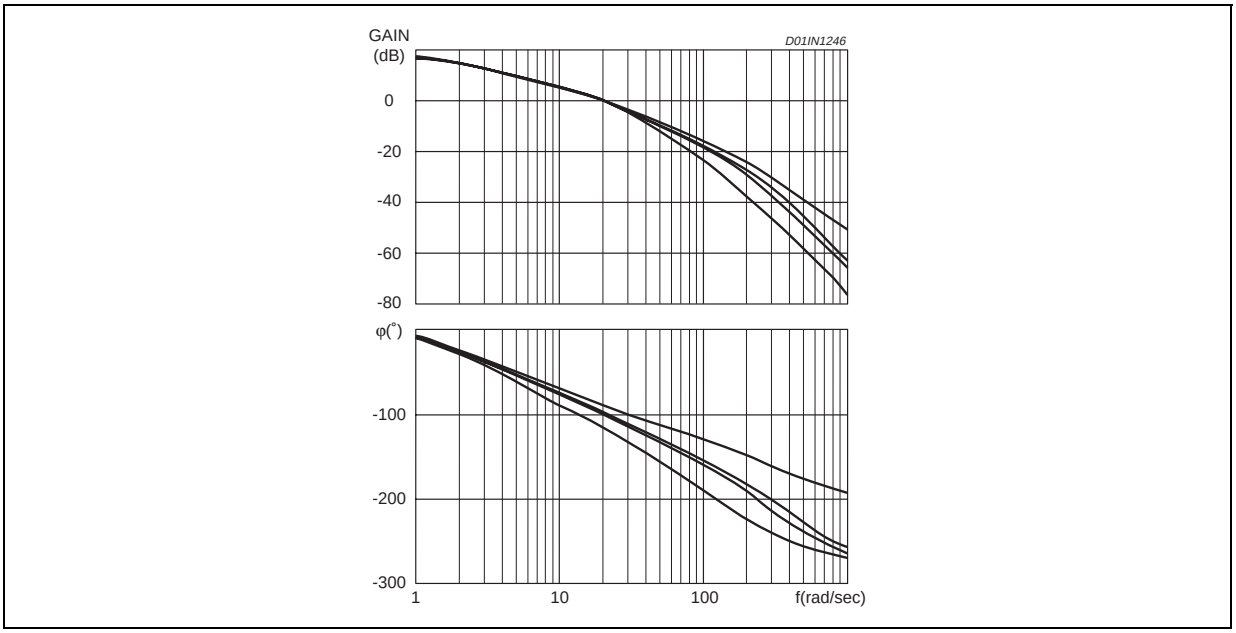
$$G_{ZOH}(s) = \frac{2/T}{s + 2/T} \qquad \text{Eq. 3-10}$$

It could be therefore improved the original discrete design by inserting Eq. 3.10 into the original plant model and finding the compensator that yields a satisfactory response. However, one of the advantages of using the emulation design method is that the sample rate does not need to be selected until the basic feedback design is completed.

In the examples introduced in the previous sections, the bandwidth was very small compared to the sample frequency used in the most common applications.

Figure 3.3. shows the bode plots of the controlled system. The solid line represents the continuous controlled system without the holder has been considered. The other lines represent the same system but with a zero-order holder considered in the plant. Due to the small bandwidth of this application (less than 4 Hz) it can be seen that for the most common applications (desired speed greater than 5400 rpm) the lag effects of the holder are not relevant.

Figure 3.3. Closed Loop bode plots of the controlled system, without holder model (solid), and with zero-order holder and a sample time of 30 ms (line), 11 ms (dot), 8.3 ms (dot line).



3.5 Conclusions

To recapitulate, the process of designing by emulation consists of the following steps:

1. Assume that the system and compensation are continuous and use the design method discussed in the previous chapters to meet all specifications.
2. Use digitization procedure to approximate D(s) with D(z).
3. Implement D(z) directly with a difference equation in the control computer.

4 MATHCAD ANALYSIS OF THE SPINDLE FLL CONTROL

4.1 Overview

The results derived in this Mathcad analysis are referred to the following FLL scheme:

Figure 4.1. Block scheme of the analog FLL spindle control.

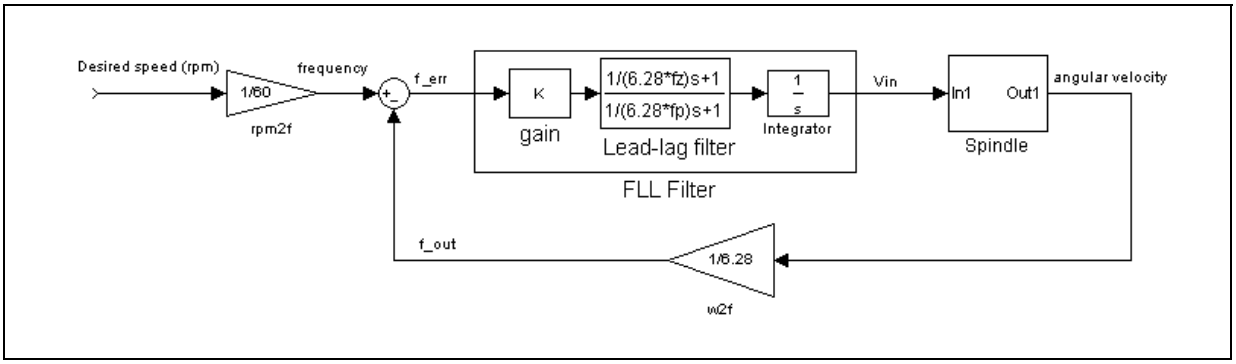
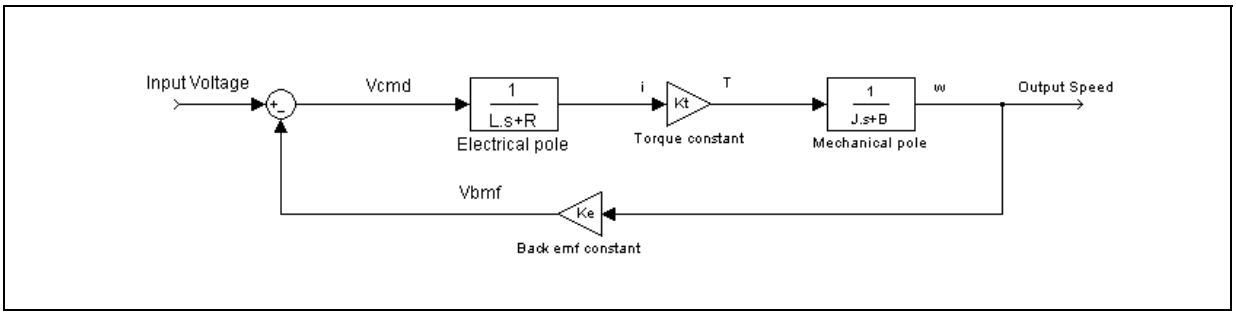


Figure 4.2. Block scheme of the used spindle model



4.2 Spindle Parameters

The mechanical and electrical characteristics of the spindle motor are defined below:

$L := 150 \cdot 10^{-6}$	[H]	Winding Inductance (coil to coil)
$R := 1.8$	[Ω]	Winding Resistance
$J := 0.732$	[g cm s^2]	Moment of Inertia
$B := 0.36$	[g cm rad s^{-1}]	Damping Coefficient
$k_e := 0.091$	[V s / rev]	Back emf Constant
$\lambda := \frac{50000}{9.8 \cdot \pi}$	conversion constant from [V s / rev] to [N m / A]	
$K_t := Ke \cdot \lambda$	[g cm / A]	Torque Constant

The definitions of the frequency and s complex variable are as follows. These are used to perform the frequency analysis:

f:= 0.1,0.2..100 Frequency Range [Hz]

w(f):= 2 · π ·f requency Range [rad/sec]

s(w):= i · w Complex variable used for the Bode plot

Now, the Open Loop transfer function of the Spindle model will be defined

Spindle Transfer Function: $G(s) = \frac{1}{2\pi}\frac{K_t}{JLs^2 + (JR + LB)s + (RB + K_tK_e)}$

Integrator Transfer Function: $I(s) = \frac{1}{s}$

Spindle plus Integrator T.F.: $G_i(s) = I(s) \cdot G(s)$

Spindle poles location

$$\omega_m = \frac{(RJ + BL) - \sqrt{(JR + BL)^2 - 4JL(RB + K_tK_e)}}{2JL}$$

$$\omega_e = \frac{(RJ + BL) + \sqrt{(JR + BL)^2 - 4JL(RB + K_tK_e)}}{2JL}$$

$$f_m = \frac{\omega_m}{2\pi} \qquad f_e = \frac{\omega_e}{2\pi}$$

Mechanical pole location $f_m = 1.704\text{Hz}$

Electrical pole location $f_e = 1.908 \cdot 10^3\text{Hz}$

Analog Filter's Parameters

The analog filter will be designed first. Afterwards, using bilinear transforms, the coefficients of the FLL digital filter can be derived.

Trick: place the zero before the mechanical pole and let the pole frequency be located a decade after the zero (fp=10*fz).

K: = 0.2 Gain:

fz: = 0.5 Zero Frequency (Hz)

fp: = 5 Pole Frequency (Hz)

a: = $\frac{fz}{fp}$ Zero-Pole displacement

Below is the Filter's transfer function:

Filter Transfer Function $$F(s) = K \cdot a \cdot \frac{1 + \frac{s}{2\pi \cdot f_Z}}{1 + \frac{s}{2\pi \cdot f_P}}$$

The overall open loop transfer function (Filter + Integrator + Spindle) becomes

Controlled Spindle TF $$C(s) = F(s) \cdot I(s) \cdot G(s)$$

Digitization - FLL Digital Filter's Coefficients Calculation

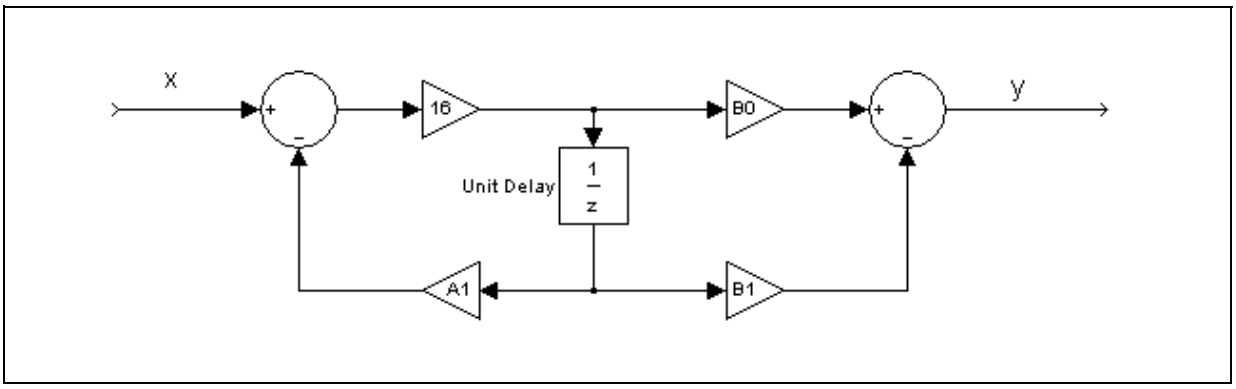
Applying the bilinear transform to the analog filter F(s) defined above, the following digital filter can be obtained

For better resolution of the filter's coefficient, a scaling factor of 16 will be applied to the derived coefficients (equivalent to a 4 bit left shift). The resulting filter transfer function is:

Digital FLL filter $$C(z) = \frac{B_0 - B_1 \cdot z^{-1}}{16 - A_1 \cdot z^{-1}}$$

It is shown in Figure 3 the block diagram for the above IIR filter using a direct form realization:

Figure 4.3. Direct form II of a first order IIR digital filter



This discrete transfer function can be implemented via software using the following difference equations:

$$16 \cdot y(n) = B_0 \cdot x(n) - B_1 \cdot x(n - 1) + A_1 \cdot y(n - 1)$$

Design parameters:

rpm:= 5400	Desired Speed[rpm]
npoles:= 6	Number of motor poles
cycle: = 1	Use 1, if Mechanical cycle is selected, npoles otherwise.
$T_s = \frac{60}{rpm \cdot cycle}$	Ts = 0.011 Period of the updating frequency [s]

Now the FLL digital filter's coefficient will be obtained using bilinear transform.

$A_0 = 16$

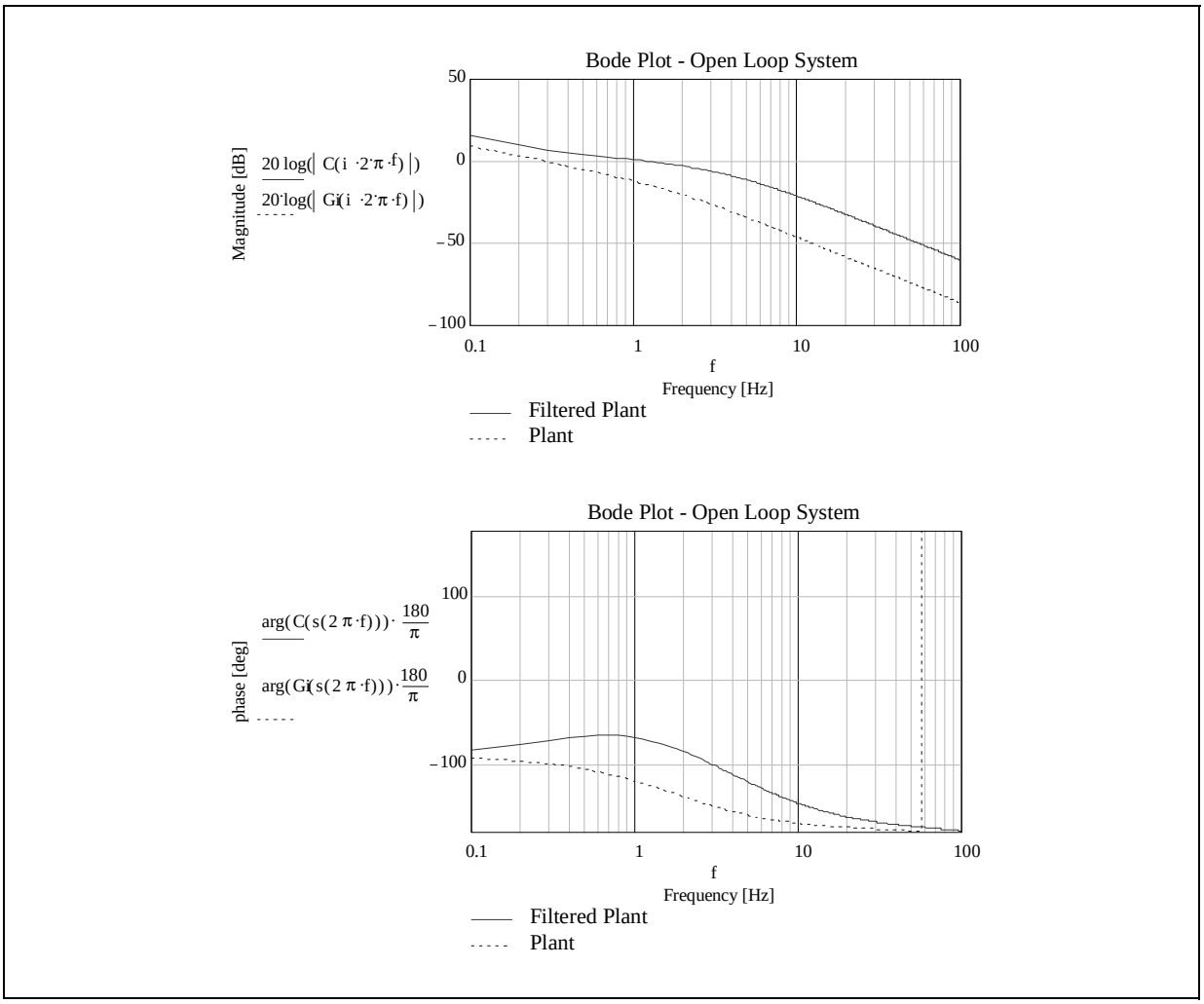
$$A_1 = \text{round}\left(16\frac{1}{1-2\pi\cdot f_Z\cdot T_S}\right) = 17$$

$$B_0 = \text{round}\left(16K\frac{f_P}{f_Z}\frac{1+2\pi\cdot f_Z\cdot T_S}{1+2\pi\cdot f_P\cdot T_S}\right) = 25$$

$$B_0 = \text{round}\left(16K\frac{f_P}{f_Z}\frac{1}{1+2\pi\cdot f_P\cdot T_S}\right) = 24$$

4.3 Bode Plots

Figure 4.4.



5 REFERENCES

5.1 Books

[1] Franklin G.F., J.D. Powell and A. Amami-Naeini, Feedback control of dynamic systems" 3rd ed. Addison Wesley, 1994

[2] Rorabaugh C.B, DSP Primer, McGrawHill, 1998

[3] Chai Hi-Dong, Electromechanical Motion Devices, Prentice Hall PTR, 1998

5.2 Internet

University of Massachusetts Lowell:

http://www.eng.uml.edu/Dept/Chemical/onlinec/white/sdyn/s6/s6fintro/s6fintro.html

Carnegie Mellon University:

http://www.engin.umich.edu/group/ctm/examples/motor/motor.html

6 APPENDIX

6.1 Symbols and their Units

Symbol	Quantity	Unit	Abbreviation
B	damping coefficient	Newton meter sec	N m s
Vn	voltage across winding n	Volt	V
Vs	voltage supply	Volt	V
θ	electrical angle	Radians	rad
K_t	torque constant	Newton meter/Ampere	N m / A
K_e	back emf constant	Volt sec	V sec
T	torque	Newton meter	N m
R	resistance	ohm	Ω
L	inductance	Henry	H
J	inertia	kilogram meter^2	Kg m^2
ω_m	angular velocity of the motor	radians/sec	rad / sec

6.2 External FLL routine

The Frequency Locked Loop control scheme introduced in the previous sections can be implemented both in internal scheme, using a microcontroller on board or using an external PC connected properly via serial or parallel port. Below the C language code routine used for the external FLL with Mozart is presented.

```
long int __export run_fll(long int init, fll_struct *fll_par_ptr,
                                        long int *rcc_log, long int *out_log)
{
/* Closes the spindle speed control loop externally:
        - Waits for FCOM signal to arrive.
        - Reads zero-cross timer result from FPGA
        - Calculates KFLL for the next spindle revolution.
        - Writes the KFLL register.
        Resturns the number of revolutions processed since the last FLL state
initialization.
Revolution clock count and FLL outputs are stored in the arrays designated by
the pointers err_log and out_log.*/


       static fll_struct fll;     /* FLL parameters */
       static long int cyc_count[12];/* past cycle count values */
       static long int err_bad;   /* error value after a glitch, 0 otherwise
*/
       static long int err_km1;   /* mechanical timing error in previous cycle
*/
       static long int flt_km1;   /* filter ouotput in previous cycle */
       static long int acc_km1;   /* accumulator value in previous cycle */
       static long int out_km1;   /* FLL output in previous cycle */
       static long int *rcc_ptr;  /* pointer to array for error storage */
       static long int *out_ptr;  /* pointer to array for FLL output storage
*/
       static long int necpr;     /* electrical cycles per revolutions */
       static long int krev;      /* revolutions since last initialization */
       long int rev_count;        /* clock cycle counts in one revolution */
       float rev_time;            /* revolution time in microseconds */
       long int err_k, flt_k, acc_k;/* error, filter and accumultaor outputs */
       long int out_min, out_k;   /* minimum and current FLL outputs */
       long int c, ec, mc;        /* cycle indices */
       long int zcross, rev;      /* zero cross and revolution indices */
       char msg[128];


       if (init>0)
       /* copy FLL parameters */
       {     fll = *fll_par_ptr;
       /* Initialize FLL state if specified. */
             if (init>1)
             { necprt = spin.npole / 2;
                   if (spin.simulate == 0)
                   {     set_timer(TM_IDLE, 0 );
             /* Setup timer for mechanical or electrical cycle count. */
                         if ((fll.cyc_type == 0) || (fll.cyc_type == 2))
                               set_timer(TM_MCYC,60000L);
                         else
                               set_timer(TM_ECYC,60000L);
                   }
```

```
                else
                      sim_spindle(-fll.out_init);
                for (c=0; c<spin.npole;c++)
                cyc_count[c] = 10 * fll.ref_time / necpr;
                err_bad = 0;
                err_km1 = 0;
                flt_km1 = 0;
                acc_km1 = fll.out_init << fll.acc_shift;
                out_km1 = fll.out_init;
                rcc_ptr = rcc_log;
                out_ptr = out_log;
                krev = 0;
          } /* end if (init > 1) */
          /* Initialize pointers always if the calling routine passes pointers
                to single variables instead of arrays. */
          rcc_ptr = rcc_log;
          out_ptr = out_log;
    } /* end if (init >0) */


    rev = 0;
    zcross = 0;
    while (rev < fll.nrev)
          /* Shift past cycle time count values. */
    {     if (fll.cyc_type > 3)
                for (c = 2*necpr -1; c >= necpr; c--)
                      cyc_count[c] = cyc_count[c-1];
                if (fll.cyc_type > 2)
                {     for (c = necpr -1; c >= 1; c--)
                      cyc_count[c] = cyc_count[c-1];
                /* Get the timer count of current cycle. */
                      if (spin.simulate == 0)
                            cyc_count[0] = set_timer(TM_MCYC, 40000L);
                      else cyc_count[0] = set_timer(TM_ECYC, 40000L);
                }
          else
          /* sim_spindle(.) returns the electrical cycle count. */
          {   cyc_count[0] = sim_spindle(out,km1);
              if ((fll.cyc_type == 0) || (fll.cyc_type ==2)) /* needs mech.
cycle */
                      for (c=1; c<necpr; c++)
                            cyc_count[0] += sim_spindle(out_km1);
          }
          /* Calculate one revolution time according to FLL cycle type. */
          switch (fll.cyc_type)
    {
        case 0: /* open loop, get mechanical cycle */
                   rev_count = cyc_count[0];
                   zcross += necpr;
        break;
        case 1: /* electrical pole */
```

```
                rev_count = necpr * cyc_count[0];
                zcross++;
        break;
        case 2: /* mechanical pole */
                rev_count = cyc_count[0];
                zcross += necpr;
        break;
        case 3: /* interleaved: sum of electrical cycles in last revolution */
                rev_count = 0;
         /*  Add up last necpr (npole/2) number of electrical cycle counts. */
                for (c=0; c<necpr; c++)
                 rev_count += cyc_count[c];
                zcross++;
         break;
         case 4: /* average of mechanical cycles ended in last revolution */
                rev_count = 0;
                for (mc = 0; mc < necpr; mc++)
                 for (ec = 0; ec < necpr; ec++)
                         rev_count += cyc_count[mc + ec];
                rev_count /= 4;
                zcross++;
         break;
         default:
                rev_count = cyc_count[0];
    } /* end switch */

     rev_time = (float)rev_count / 10.0;
     /* Add 1us to the error to compensate for acquisition delays. */
     err_k = (long int)rev_time - fll.ref_time + 1;
     if (err_bad == 0)
     /* Normal history. Keep going. */
     {       if (labs(err_k) > fll.err_glitch)
     /* Unexpected error. See if it will be the same in next revolution. */
             {       err_bad = err_k;
                     err_k = 0;
             }
     }
     else
             /* There was a jump in error in the previous revolution. */
     {       if (labs(err_k) > fll.err_glitch)
     /* Error is still too bad. Process it this time. */
                     err_bad = err_k;
             else
     /* It was a glitch or detection fault. Resume normal operation. */
                      err_bad = 0;
     }
     /* Clip error value. */
     if (err_k > fll.err_clip) err_k = fll.err_clip;
     if (err_k < -(fll.err_clip)) err_k = -(fll.err_clip);
     /* Calculate compensation filter response. */
```

```
        flt_k = ((fll.a1*flt_km1) + (fll.b0*err_k) -fll.b1*err_km1)) / fll.a0;
        err_km1 = err_k;
        flt_km1 = flt_k;
        /* Upsate accumulator. */
        acc_k = acc_km1 + flt_k;
        if (acc_k > fll.acc_clip) acc_k = fll.acc_clip;
        if (acc_k < 0) acc_k = 0;
        acc_km1 = acc_k;
        /* Find output to be sent to the device. */
        out_k = acc_k >> fll.acc_shift;
       /* Minimum setting depends on the current spindle speed. fll.out_clamp
               gives the maximum allowed drop as percentage of the rated
               spindle speed.
       */
        out_min = (long int)((60.0R+6 / rev_time -
(float)spin.speed*(float)fll.out_clamp/100.0)/spin.Kt);
        switch (Device)
        {
        case L7200:
               if (out_min < 50) out_min = 50;
               if (out_k < out_min) out_k = out_min;
               if (out_k > 511) out_k=511;
               /* Check if the bits other than the 8 LSBs need to be modified. */
               if ((0xFF00L & (out_k ^ out_km1)) != 0)
                      write_reg(0x8EL, (out_k >> 8) & 0xFFL);
               write_reg(0x7EL,out_k & 0xFFL);
               break;
        case L72XX:
               if (out_min < 50) out_min = 50;
               if (out_k < out_min) out_k = out_min;
               if (out_k > 255) out_k = 255;
               write_reg(0x06L, out_k);
               break;
        } /* end switch*/
        out_km1 = out_k;


        if (zcross == necpr)
        {      zcross = 0;
               rev++;
               krev++;
               *rcc_ptr = rev_count;
               rcc_ptr++;
               *out_ptr = out_k;
               out_ptr++;
        }
       }

       return (krev-1);
}
```

6.3 Torque Conversion Table

To convert from A to B, multiply by entry in Table

B / A	dyne·cm	g·cm	oz·in	Kg·cm	lb·in	Newton·m	lb·ft	Kg·m
dyne·cm	1	1.01972×10^{-3}	1.41612×10^{-5}	1.01972×10^{-5}	8.850731×10^{-7}	10^{-7}	7.37561×10^{-8}	1.01972×10^{-8}
g·cm	980.665	1	1.38874×10^{-2}	10^{-3}	8.67960×10^{-4}	9.80665×10^{-5}	7.233×10^{-5}	10^{-5}
oz·in	7.06157×10^{4}	72.0079	1	7.20079×10^{-2}	6.25×10^{-2}	7.06157×10^{-3}	5.20833×10^{-3}	7.20079×10^{-4}
Kg·cm	9.80665×10^{5}	1000	13.8874	1	0.86796	9.80665×10^{-2}	7.23300×10^{-2}	10^{-2}
lb·in	1.12985×10^{6}	1.15213×10^{3}	16	1.15213	1	0.112985	8.33333×10^{-2}	1.15213×10^{-2}
Newton·m	10^{7}	1.01972×10^{4}	141.612	10.1972	8.85073	1	0.737561	0.101972
lb·ft	1.35582×10^{7}	1.38255×10^{4}	192	13.8255	12	1.35582	1	0.138255
Kg·m	9.80665×10^{7}	10^{5}	1.38874×10^{3}	100	86.7960	9.80665	7.23300	1

6.4 Speed-Frequency Conversion Table

To convert from A to B, multiply by entry in Table

B / A	rad/sec	Hz	rpm
rad/sec	1	$1/2\pi$	$60/2\pi$
Hz	2π	1	60
rpm	$2\pi/60$	1/60	1

TABLE OF CONTENTS