KOLLMORGEN

www.DanaherMotion.com

SERVO**STAR**® S- and CD-series Velocity Tuning Algorithms

This document describes the velocity and current algorithms, auto and manual tuning features, and the filtering capabilities of the SERVO**STAR**® S and SERVO**STAR**® CD product line. These products offer a choice of four velocity controller algorithms (filters):

- **PI** controller (**P**roportional & **I**ntegral) (COMPMODE = 0)
- **PDFF** controller (**P**seudo **D**erivative **F**eedback + **F**eedForward) (COMPMODE = 1)
- **PP** controller (**P**ole-**P**lacement) (COMPMODE = 2)
- **APP** controller (**A**dvanced **P**ole-**P**lacement) (COMPMODE = 3)

Each controller may be selected for use in the velocity loop (OPMODE = 0 or 1). The position loop has an additional proportional gain controller with feed-forward. Since this is a digital control system, a discrete system representation approach is taken.

Structure of Controllers

The controller structure for the four controller types is shown below:

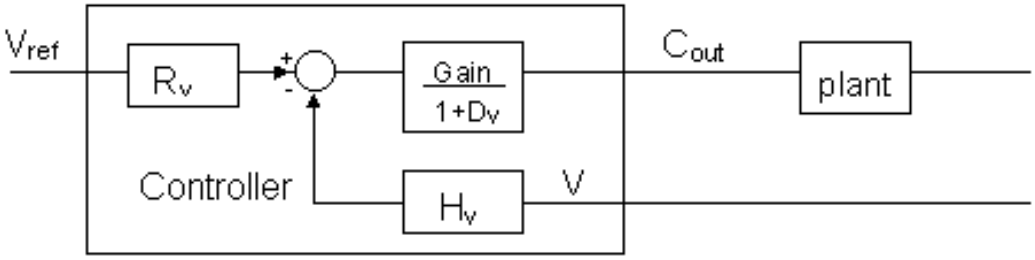


Figure 1. Controller Structure

The controller is realized as polynomial in a 32-bit accuracy. It has three main blocks:

- R_v -The pre-filter polynomial.
- H_v - The feedback polynomial.
- $1+ D_v$ - The forward polynomial.

In addition, it has saturation blocks and an anti-windup mechanism which prevents overshoot and oscillation in the case of command saturation.

TYPES OF CONTROLLERS

PI Controller (COMPMODE = 0)

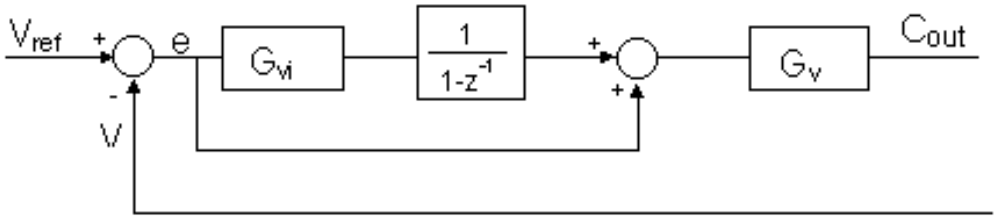


Figure 2. PI Controller Configuration

The PI controller has a unity feedback with no pre-filter. The control algorithm is as follows:

$C_{out} = G_v * [\ 1 + G_{vi} /(1-z^{-1}\ )\ ] * e \qquad e = V_{ref} - V$

The proportional gain term, G_v , stabilizes the system. The integral term, G_{vi} , compensates for the steady state error. The PI controller offers set-point tracking capability as the algorithm is entirely error driven. It also tends to provide middle range bandwidth which permits fair set-point tracking capability.

Both parameters, G_v and G_{vi} , are tuned by trial & error techniques. These terms are represented by the SERVO**STAR**® variables GV and GVI respectively.

PDFF Controller (COMPMODE = 1)

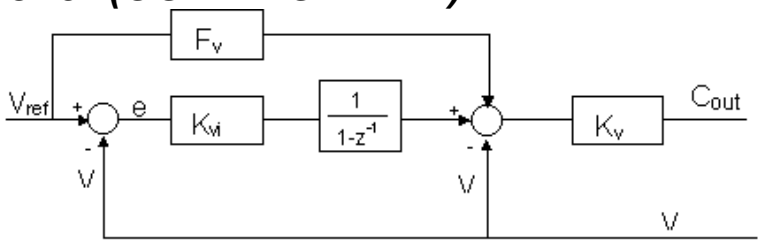


Figure 3. PDFF Controller Configuration

The PDFF controller is the familiar PDF controller with a feed-forward term added. The differences between the PDFF and the PI control scheme are listed below:

- The feedback term is inserted between the integral and proportional blocks (typical approach to "pseudo-derivative" effect). This feedback is superior from a load-noise rejection point of view.
- The feed-forward block gives the control system the flexibility to change its characteristics from a basic PDF controller (where the feed-forward term $F_v = 0$) to a PI controller ($F_v = 1$). F_v may vary between 0 and 1 which gives this controller structure its advantage of flexibility relative to PDF and PI controllers. Actually, the PDFF is a generalized controller. The PI and PDF controllers are simply two cases of the PDFF controller encountered at the limits of F_v.

The controller parameters - K_v , K_{vi} , and F_v are tuned by trial & error techniques. These terms are represented by the variables KV, KVI, and KVFR respectively in the SERVO**STAR** system.

Standard Pole-Placement Controller (PP; COMPMODE = 2)

The PP controller parameters are calculated from the system inertia and the desired closed-loop bandwidth by using the Pole-Placement Design Method (PP). The calculation is based on the following equation:

$$\frac{V}{V_{ref}} = \frac{Gain * R_v * B}{(1+A)*(1+D_v) + Gain * B * H_v} = \frac{Gain * B}{(1+C)}$$

where R_v , H_v , and D_v are the controller polynomials and the term B/(1+C) represents the plant model, in which B is proportional to the inverse of the system inertia. (1+C) is the characteristic equation constructed from the closed-loop bandwidth (or poles) of the system. It also includes other predefined poles that are automatically calculated by the drive to give maximum robustness and torque disturbances rejection. By specifying the moment of inertia (≅ 1/B) and the closed-loop bandwidth (dominant poles of 1+C), the controller polynomial coefficients are calculated automatically. Therefore, only two parameters have to be specified for controller tuning:

- The **Load Inertia Ratio** parameter (represented by the variable LMJR)
- The **Closed-loop System Bandwidth** (represented by the variable BW)

It is not necessary to know the load inertia for the design. A very easy way to tune this parameter is described in this document.

The main difference between the PP controller and the PDFF controller is the order of the polynomials:

- The D_v polynomial is not just an integrator as in the PDFF, but a higher-order polynomial.
- The structure presents a non-unity feedback.
- R_v and H_v are higher-order polynomials that give a larger number of degrees of freedom. This leads to high performance.

Practically, those differences mean that the controller uses data from previous samplings and can better compensate for delays which occur normally in discrete systems. In addition, the motor and the load are controlled in the same matter. There is no "disconnection" between the load and the motor. This property of the controller leads to the main advantages of PP controllers in relation to the PDFF and PI controllers. These advantages are:

- **Higher closed-loop bandwidth**
  Since the controller polynomials compensate better for the delay of the system, higher bandwidth is attainable.
- **Better stability & robustness**
  Due to the high order of the Dv polynomial compared to the integrator of the PI and PDFF controllers, the system stability and robustness are improved.
- **No overshoot design**
  The structure of the PP controller provides a non-overshoot design, whereas the PI controller provides an overshoot by its very nature.

- **Separation between bandwidth and gain**
  The PP controller uses the higher-order polynomials to attain high bandwidth using the dynamics of the controller poles. This is an advantage over the higher gains required by PI and PDFF to achieve high bandwidth. It improves the noise rejection, robustness, and stability of the system.
- **The load and the motor are controlled together**
  The pole placement method for controller design based on the motor position sensor is able to control the motor and the load at the same time, while the PI controller may "disconnect" the load from the motor. When the gain of the PI increases, the resonance "seen" by the motor moves to the anti-resonance. On the motor side, the system is more stiff (the resonance and the anti-resonance on the motor side are nearly similar). On the load side, the resonance of the system can be excited (at the anti-resonance frequency).
- **Easier to tune**
  Since the calculation of the controller parameters is done mathematically, according to the user request of closed-loop bandwidth, the tuning procedure does not involve trial and error, so it is more convenient and straight-forward.

The PP structure provides another advantage that makes the controller tuning easier. When an underestimation of the load inertia occurs, the closed velocity loop time response presents an overshoot. An overestimation of the load inertia leads to undershoot and oscillations. The PP type of controller can be tuned simply by adjusting one parameter - the load inertia ratio. Once the proper load inertia ratio is selected, changing the bandwidth parameter either improves or degrades performance.

Advanced Pole-Placement Controller (APP; COMPMODE = 3)

You can use a higher-order polynomials controller (the APP controller), which is preferable for flexible (resonant) systems with considerably high delay. This controller is also referred to as the "Advanced Pole-Placement" controller.

The calculation of the coefficients of the higher-order polynomials cannot be done by the drive, but has to be done by external software. For example, Kollmorgen's CONCAD motion analyzer, a control and mechanical diagnostic integration tool, can design this type of controller.

Similar to the PP method, the CONCAD system finds the controller polynomials Rv, Hv and Dv by solving the equation:

$$\frac{Gain * Rv * B}{(1+A)(1+Dv)+Gain * B * Hv} = \frac{Gain * B}{(1+C)}$$

Unlike the PP, it uses a higher-order plant model which includes a pole, an anti-resonance, and a resonance, as shown in the following equation:

$$\approx \frac{G(S^2 + 2\xi_{ar}\omega_{ar}S + \omega_{ar}^2)}{(S+P)(S^2 + 2\xi_r\omega_r S + \omega_r^2)}$$

Taking the Z transform of this continuous velocity model equation changes the denominator and the numerator order to 3 and 5, respectively.

After solving the equation above, and obtaining Rv, Hv and Dv, the coefficients of the polynomials are down-loaded to the drive by **MOTIONLINK**. These terms are represented by the vector variables R, H, and D respectively in the SERVO**STAR** system.

LOW PASS FILTERS

For applications with difficult vibration and/or torsional resonance problems, two independent low-pass filters (LPF) are provided in the SERVO**STAR**. When activated, each low-pass filter attenuates all the frequency components above its cutoff frequency.

A number of vibration sources exist in servo motor applications. Some of these are: torsional resonance, lateral resonance, power ripple (60/120... Hz), and other structural vibrations. The effects of these undesirable vibrations include audible noise, weakening of structures and mounting failures, reduction in the system bandwidth, excessive motor current, and control loop oscillations. Many of these are effectively filtered out using one or two low-pass filters. However, these filters cannot compensate for a poorly designed mechanical system.

Both low-pass filters are implemented digitally through software in the microprocessor. Software filters are easy to tune and are just as effective as hardware filters. Each low-pass filter is a first-order filter. The filter cutoff frequencies (LPFHZ1 and LPFHZ2 for the first and second filters, respectively) are in the range of 20 to 800 Hz, in steps of 20 Hz. A second-order filter can be implemented by activating the two filters with the same cutoff frequencies.

The filter is placed in the velocity-error signal just before the torque command is clamped by current limits. The filters are available only in the PI, PDFF, and simple pole placement controllers (COMPMODE's 0-2). To activate a filter, use the FILTMODE command. Setting FILTMODE = 0, turns off both low-pass filters. Setting FILTMODE = 1, turns on the first low-pass filter, for which the cutoff frequency is LPFHZ1. FILTMODE = 2 turns on both filters (the cutoff frequency for the second filter is LPFHZ2).

AutoTuning, using the TUNE command, sets FILTMODE to 0.

If the filter cutoff is specified at a low frequency, the bandwidth of the system is low. Setting the cutoff frequency too low may cause the system to exhibit total instability. If the system is tuned to have high bandwidth and the filter cutoff frequency is low, the motor may oscillate. In such a case, either the filter cutoff should be increased or the bandwidth of the system decreased. As a general rule of thumb, the filter frequency should not be less than four times the desired system bandwidth.

All filters add phase-lag delays. In applications where the SERVO**STAR** is being controlled by another position loop controller, this lessens the system's phase margins.

AutoTuning the Velocity Controller

The AutoTune function (using the TUNE command in the SERVO**STAR**) is an automatic procedure which does the following:

- Estimates the system load inertia. (LMJR variable).
- Designs a controller based on the estimated load inertia and the requested bandwidth (BW variable) by using the optimal Pole-Placement Design Method (PP).

Once the PP controller is designed (values are assigned to LMJR and BW), the designs for the PI and PDFF controller types are derived from the PP controller, which results in values being assigned to the variables GV, GVI, KV, KVI, and KVFR.

The TUNE command syntax and usage is described in the variables section of this document.

The Concept

The idea of the AutoTune procedure is that since the PP controller design is based on the system inertia, the better the load inertia is estimated, the better the system performance of the controller.

One of the PP controller qualities is that if the system model is exact, it responds with no overshoot to a velocity-step command. Usually, the inertia of the load is unknown or cannot be accurately determined. During the Auto-Tune process, the drive estimates the load inertia by assigning an initial value to the load inertia variable (LMJR). It then issues a velocity step command, analyzes the system response to this command, and adjusts the value of LMJR based on that response. The following guidelines apply:

- If the assigned load inertia value is less than the actual load inertia, the system response has an overshoot.
- If the assigned load inertia value is greater than the actual load inertia, an undershoot may occur in the step response. This may even lead to oscillations and controller instability.
- Once the system response has no or minimum overshoot, it can be assumed that the load inertia is well estimated. The drive ceases the tuning process at this point and retains the value it has assigned to LMJR.

Once the load inertia is well estimated, the controller bandwidth may be increased (or decreased) to give better performance. A well-tuned controller is one in which the response to a velocity step command has no overshoot, undershoot, or oscillations.

The Procedure

The AutoTuning procedure is activated by the TUNE command. The system has its default values for the bandwidth, direction, and speed, but you may choose other optional values. The low-pass filters must be turned off (FILTMODE = 0) before the TUNE command is used. The following is the description of the AutoTune procedure with selection of optional values (refer to the Variable/Command section of this document for syntax explanation):

Specify the command: TUNE < optional parameters: bandwidth, direction, and velocity >

A command example: **TUNE 25 0 600**

(Tune the controller for a BW of 25 Hz, bi-directional movement, and step command velocity of 600 rpm)

The AutoTune operation will not be executed if:

- The operation mode in not Serial Velocity (OPMODE 0).
- The compensation mode is Advanced Pole-Placement (COMPMODE = 3).
- The drive is in Hold Position mode (HOLD = 1).
- The drive is in Zeroing mode (ZERO = 1).
- The Recording process is active.
- The TUNE parameters are outside the permitted ranges.

Manual Tuning of the Velocity Controller

Where the results from the AutoTune do not yield satisfactory performance, the manual tuning procedure may be performed. Manual tuning includes the initial steps of load inertia estimation, and the subsequent steps to design the optimum controller.

The following is a manual-tuning procedure (it is similar to the procedure that is performed automatically by the drive during the AutoTune process). The procedure described is based on using a dumb terminal interface utilizing the standard ASCII protocol. The same procedure may be followed much more conveniently using the Tune Screen in **MOTIONLINK**. Go to the Tune Screen and press F1 to access context-sensitive help that explains how to tune the drive.

If tuning from a dumb terminal, use the following procedure:

1. Set the operation mode to serial velocity, using **OPMODE 0**.
2. Set the velocity compensator to standard pole-placement, or PP, using **COMPMODE 2**.
3. Enable the drive using **EN**.
4. Set the desired bandwidth using **BW** command. It is recommended to set the initial value low (10 Hz), **BW 10** and then work up.
5. Set the **load inertia ratio** to zero using **LMJR 0**. The load inertia value LMJR is expressed in percent of the rotor inertia.

The range for LMJR is 0 to 10,000. It represents a range of load:rotor inertia ratios ranging from 0:1 to 100:1 (For example, to set a 10:1 ratio between the load inertia and the motor inertia, use the command LMJR 1000). At this point the step response should have the maximum overshoot, but the control loop should be stable (low gains).

6. Activate the recording mechanism to record the **velocity command**, the **current command**, and the **actual (feedback) velocity**, using **RECORD 1 1024 VCMD ICMD V**.
7. Jog the motor in constant low speed, using **J value1** command (For example, start with the command, **J 150)**
8. Activate the recording trigger, using **RECTRIG VCMD value2 20 1**. Value2 should be greater than value1 in the previous stage (For example, **RECTRIG VCMD 200 20 1).**
9. Jog the motor to a higher value than specified in the **RECTRIG** command. This is actually the step command. The higher the step, the better are the results. The step should not be below 300 [RPM] (For example, **J 500).**
10. The recording takes about half a second. After this period of time, reduce the motor speed.
11. Verify that the recording process is terminated using **RECDONE** command.
12. Dump the recorded data to the serial port using **GET** command.
13. Analyze the results.
    If the current command is saturated (gets absolute values of **ILIM**) during the process, reduce the step or reduce the bandwidth, and return to step 6.
14. Inspect the feedback velocity:
    - If there is an undershoot or oscillations, reduce **LMJR** to the mean value of the present **LMJR** and the last **LMJR** where an overshoot exists, and return to step 6.
    - If there is an overshoot, increase **LMJR** by 100 and return to step 6.
    - Repeat this process until the overshoot is minimum or nearly in a "Non overshoot" situation. (The estimated load inertia may be queried by **LMJR**.)

Advanced Tuning of the Velocity Controller

The automatic and manual tuning procedures use the Pole-Placement design method based on a simple system model. Of course, this is not always the case. When a load is connected to the motor and the coupling/gearing is very stiff, the simple model is accurate enough (meaning it does not change). However, in the case where there is a lot of flexibility in the system (less stiffness between the gear and the load), the model changes to include both anti-resonance and resonance.

The following figure shows the Bode plot of an amplitude only for simple and resonant plant models (frequency domain):

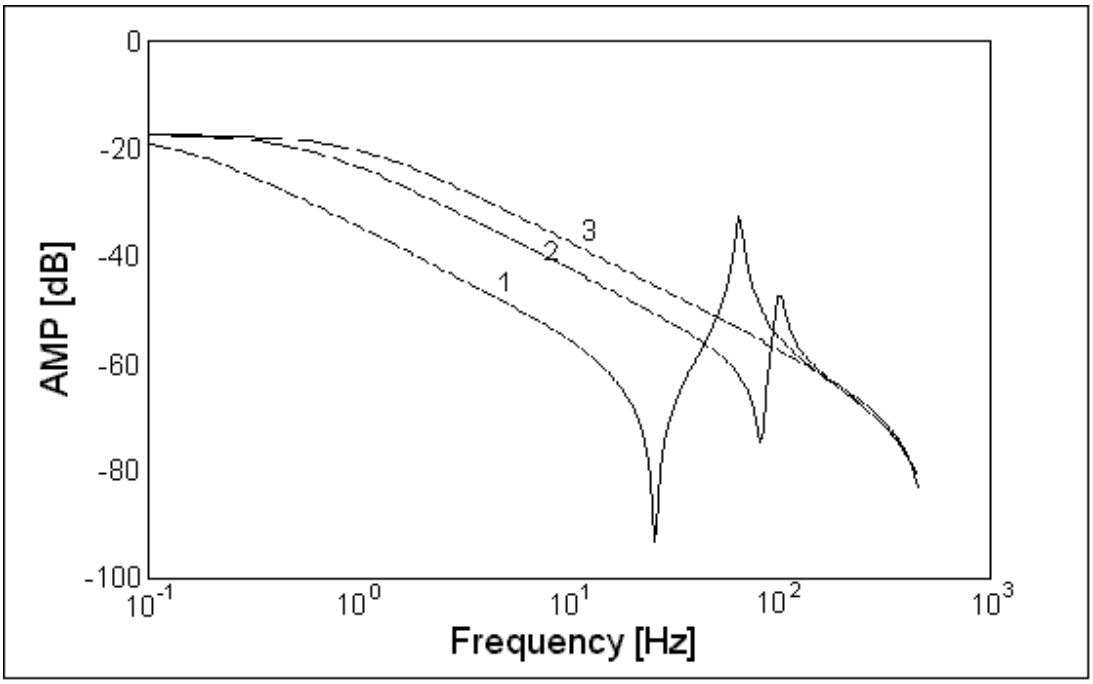


Figure 4. Bode Plots for an Electromechanical System

- Plot **3** represents the simple model.
- Plot **1** (high inertia load) and plot **2** (low inertia load) represent the situation where the load is connected to the motor with a non-rigid connection (both anti-resonance and resonance exists).

Notice that the three lines meet at the high frequencies range where the lines represent the inertia of the motor and the part of the load which is rigidly connected to the motor. Usually, the flexibility of a system is within the gear. Therefore, the equivalent model at high frequencies is based on the motor inertia with a small effect of the load inertia.

In the case of a resonant model, the closed-loop bandwidth that the AutoTune process uses, is limited by the anti-resonance frequency. In order to receive higher bandwidth, lower load inertia ratio has to be selected (even though it may cause an overshoot). Therefore, we can define two cases:

1. **Stiff system / Small (typical) load**
   The simple model is accurate and the AutoTune procedure is suitable for the load inertia estimation and controller parameters.
2. **The system is flexible**
   Use the AutoTune procedure to analyze the load inertia in the low frequency range. The resultant bandwidth should be defined lower than the anti-resonance frequency (select the minimum of 10 Hz in case of flexible system).
   - Increase the bandwidth in 5 Hz increments until the desired performance is achieved or oscillation occurs. Choose the last bandwidth that gave the desired performances or stable responses.

- To have a higher bandwidth, define zero load inertia and a bandwidth higher than the resonance frequency. The resonance frequency may be estimated from the system response oscillations.
- Analyze the system response with respect to the overshoot and the oscillations. If the overshoot is too high, increase the load inertia until the desired response is received (compromise).

Advanced Topics

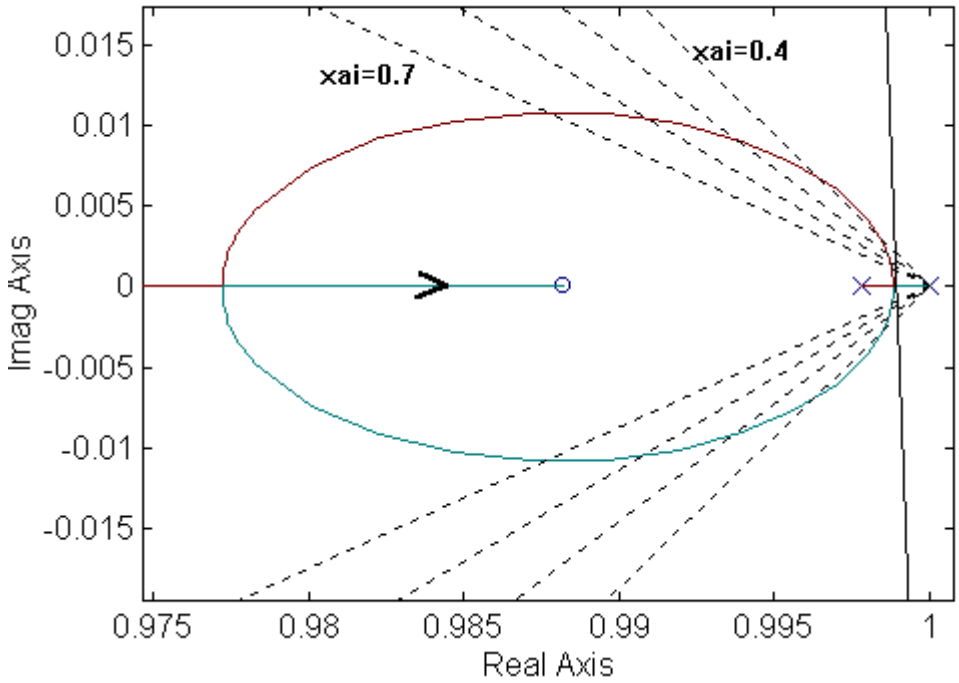
PI Controller

The closed-loop transfer function of the system with plant model B/(1+A) and PI controller is:

$$\frac{V}{V_{ref}} = \frac{BG_v\left[(G_{vi}+1)-z^{-1}\right]}{(1+A)(1-z^{-1})+BG_v(1+G_{vi}-z^{-1})}$$

The equation shows that the closed-loop has a zero due to the integrator of the PI controller. The zero in the system causes an overshoot of the step response.

The following root locus diagram shows the behavior of the closed-loop poles when the gain, G_v ,is changing for a very stiff system. This assumes that the frequency of the zero in the system is greater than the mechanical pole of the system.



Only when the gain, G_v, is near the infinity, is the zero canceled with the closed-loop pole. In most cases, the closed-loop poles are in the circle as shown above. The poles are either complex, with damping factor lower than 0.7 or located in higher frequencies than the zero. In both cases, the effect presents an overshoot in the step response.

PDFF Controller

In order to understand the "derivative" effect of the special feedback in the PDFF controller, Figure 5 shows an equivalent control loop with a "derivative" term in its main feedback instead of the feedback signal of the PDFF. This explains the "pseudo derivative" effect of the PDFF controller:

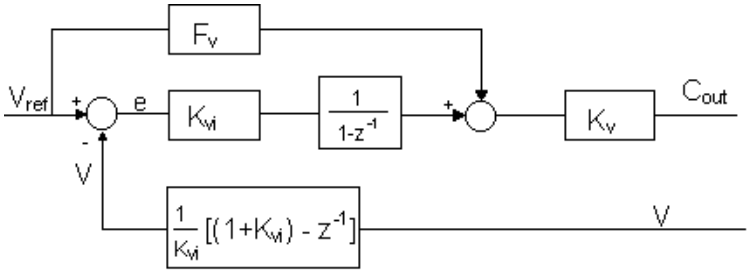


Figure 5. Equivalent PDFF Controller configuration

The feed-forward term, F_v, gives the control system the flexibility to change its characteristics from a simple PDF controller to a PI controller. This may be seen by introducing a pre-filter term "R", which is equivalent to F_v:

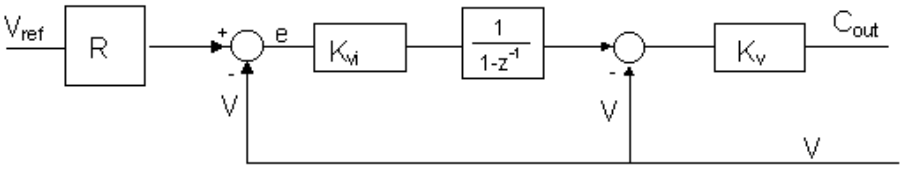


Figure 6. PDF Controller configuration with pre-filter, equivalent to PDFF

In general, $R = 1 + F_v / K_{vi} * (1 - z^{-1})$. It can be seen that when F = 0, then R = 1 , and the control loop is a simple PDF controller. When F = 1, then $R = 1 + (1 - z^{-1})/ K_{vi} = 1/ K_{vi} [(1 + K_{vi}) - z^{-1}]$, which gives a simple PI controller. To understand this, a pre-filter block is introduced into the control loop of Figure 2 (shown in Figure 7):

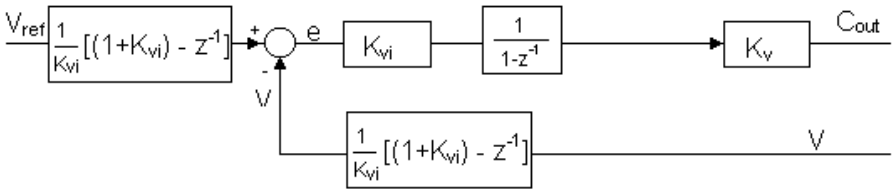


Figure 7. PI Controller configuration generated from a PDFF controller

It can be seen that the terms in the feedback and in the pre-filter are identical. By moving both terms inside the control loop, an equivalent control loop scheme is obtained as in Figure 1, where $K_{vi} = G_{vi}$, which is a PI controller. It means that the PDFF is a generalized controller where the PI controller is simply a particular case of it, F=1.

There is, of course, the option to choose the value of F somewhere between 0 and 1, which gives this controller its advantage of flexibility in relation to PDF and PI controllers.

PP Controller

The Pole-Placement Design Method is a direct method for controller design. Its algorithm evaluates the controller parameters based on the plant model, the controller structure, and specific requirements of the closed-loop transfer function.

Referring to Figure 6, the closed-loop transfer function of the velocity loop is:

$$\frac{V}{V_{ref}} = \frac{Gain * R_v * B}{(1+A)*(1+D_v) + Gain * B * H_v}$$

where the velocity plant is defined as B/(1 + A).

A, B, R_v, H_v and D_v are polynomials which are defined as:

$$1 + A = 1 + a_1 z^{-1} + \ldots\ldots\ldots + a_n z^{-n}$$
$$B = b_0 + b_1 z^{-1} + \ldots\ldots\ldots + b_m z^{-m}$$
$$1 + D = 1 + d_1 z^{-1} + d_2 z^{-2} + d_3 z^{-3}$$
$$H_v = h_0 + h_1 z^{-1}$$
$$R_v = r_0 + r_1 z^{-1}$$

In the Pole-Placement method, the desired closed-loop poles and R_v poles are determined. The polynomial coefficients H_v and D_v are calculated accordingly. In order to receive the desired closed-loop poles without changing the system zeroes, the closed-loop transfer function is defined in the following way:

$$\frac{V}{V_{ref}} = \frac{Gain * R_v * B}{(1+A)*(1+D_v) + Gain * B * H_v} = \frac{Gain * R_v * B}{(1+C) * R_v}$$

where (1 + C) includes the dominant closed-loop poles (user-defined bandwidth) and other poles automatically calculated in the drive for maximum robustness and torque disturbances rejection. R_v consists of the pre-filter poles. Since R_v exists in the numerator and the denominator of the right side of the equation, it is obvious that R_v poles do not affect the closed-loop behavior.

If n is the degree of A and m is the degree of B , the number of poles of R_v and (1 + C) is limited to (n+m). For example, if the degrees of B and A are 5 and 3, respectively, the maximum number of closed-loop and observer poles is 8. (A complex pole is considered as 2 poles).

The following equations are examples of physical continuous system model descriptions:

Systems that are considered Stiff:

$$\approx \qquad \frac{K_a K_T / J_T}{S + B_m / J_T}$$

where J_T is the total moment of inertia of the motor and the load : $J_T = J_{m+} J_l$

Systems that are considered as flexible system-resonant models:

$$\approx \quad \frac{K_a K_T / J_T * (S^2 + \omega_{ar} \xi_{ar} S + \omega_{ar}{}^2)}{(S + B_m / J_T)(S^2 + \omega_r \xi_r S + \omega_r{}^2)}$$

where the resonance and anti-resonance are functions of the physical parameters (Moment inertia [load and motor], molecular damping, stiffness, viscosity friction damping [load and motor], etc.)

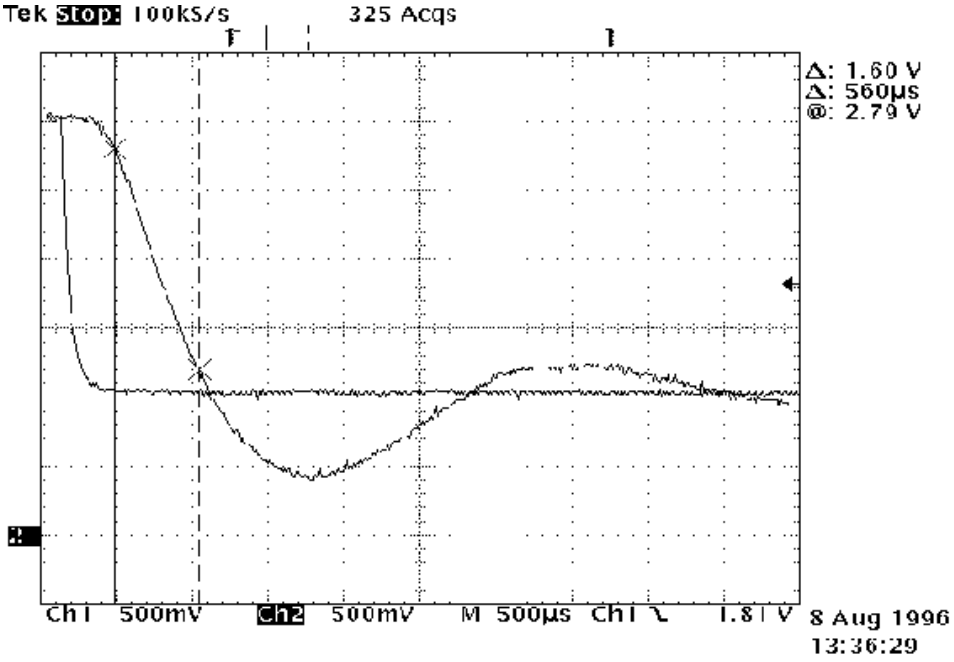
Since the PP controller design is based on a simple model, you must decide on which plot the controller will be designed: the highest one (load inertia = 0), the lowest one (maximum load inertia of the system, before the anti-resonance frequency), or somewhere in between. The selection of the load inertia in this range implies also the selection of the system bandwidth. If the highest value of the load inertia is selected, the maximum bandwidth that can be selected is lower than the anti-resonance bandwidth. If the lowest value of the load inertia is selected, the bandwdth that can be selected may be higher than the anti-resonance frequency. In general, the selected value is somewhere between these extreme values.

Examples: Comparison of PI and PP / APP

Current Loop Controller

The following graphs show the real behavior (step response) of the phase current loop controller of the SERVO**STAR** on a brushless motor (Kollmorgen **GOLD**LINE family) using a PP or a PI controller.

PI Controller - Current Loop

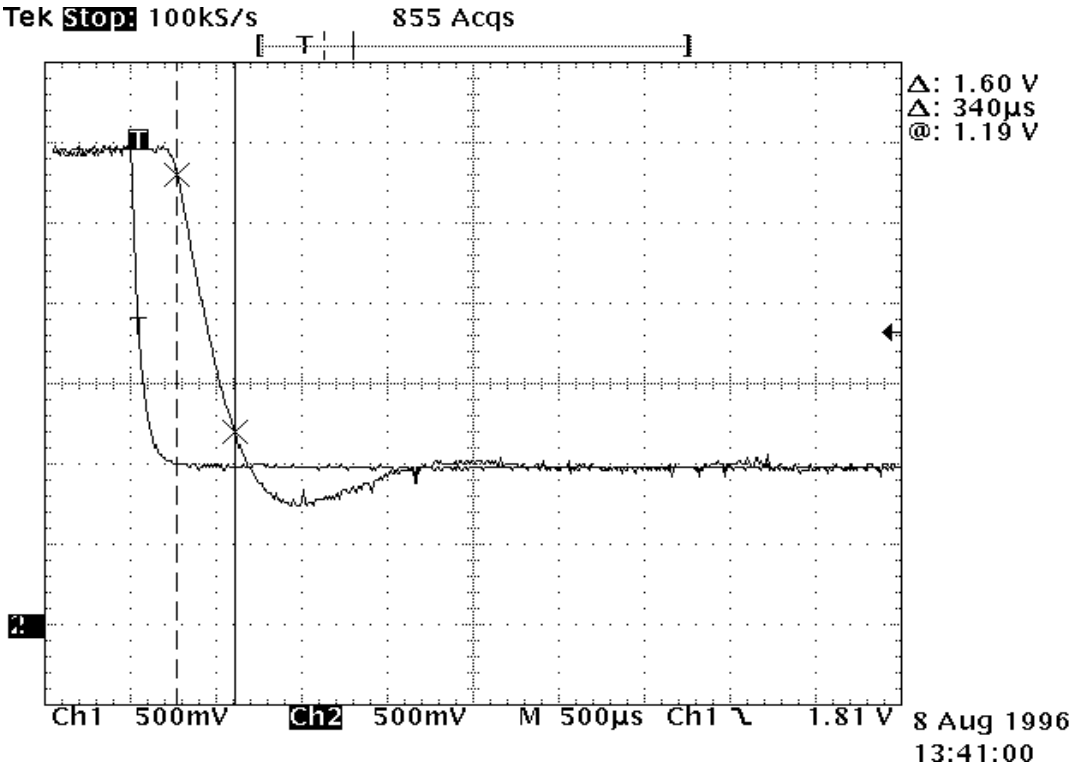


Overshoot : 30 %

Maximum Bandwidth achievable: 850 Hz where

Bandwidth $= f_n = \frac{2.5}{2\pi\, tr}$ **Hz**

Pole Placement Controller - Current Loop



Overshoot : 10 %
Maximum Bandwidth achievable: 1,404 Hz where

Bandwidth = $f_n = \frac{2.5}{2\pi\, tr}$ **Hz**

Velocity Controller

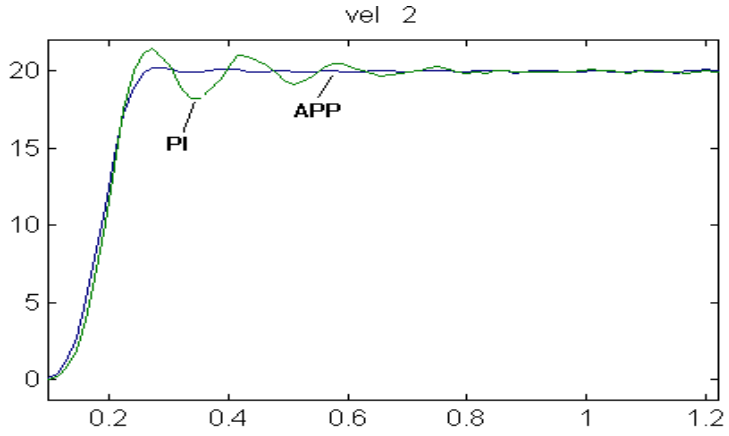
The following graphs show the real behavior (step response) of the velocity of axis 2 of a six degree industrial robot. This electro-mechanical axis presents the following characteristics:

- Low frequency Resonance
- Extreme low damping factor
- Large change in moment of Inertia depending on robot movement
- Large cross-coupling torque disturbances
- Large unbalance torque

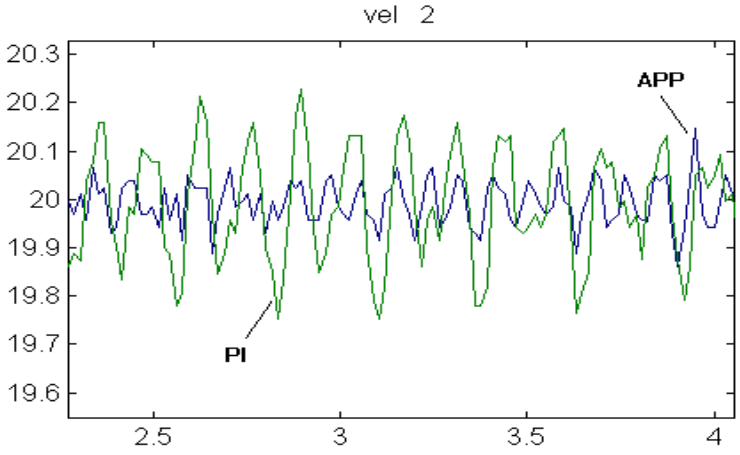
Application requirements by priorities :

1) Rise time
2) Ripple less than 1.5%
3) No overshoot

The Results (Velocity Graph) Of Two Controllers PI And APP



The Steady State Of The Two Controllers



	PI	APP
Peak to peak ripple	2.2 %	1.1 %
Overshoot	15 %	0 %
Oscillations are presented to achieve required rise time	YES	NO