2015 IEEE Conference on Control Applications (CCA)
Part of 2015 IEEE Multi-Conference on Systems and Control
September 21-23, 2015. Sydney, Australia

A Unified Framework for Analog and Digital PID Controllers

Daniel Y. Abramovitch*

Abstract— **This paper presents a unified framework for considering analog and digital PID controllers by tying them to second order linear models. While no one item in this is novel, the unified treatment allows one to more rationally select PID parameters from loop shaping desires. Furthermore, the treatment leads to some simplified understanding of popular implementation choices for PID controllers.**

I. Introduction

One cannot examine most industrial control environments without tripping over simple controllers of all sorts of specifications broadly labeled as PIDs (Proportional-Integral-Derivative Controllers). This paper attempts to place these slightly different forms into a common framework. Issues of discretization and PID design from filter design are addressed in a coherent way. Furthermore, PID implementations are increasingly digital, since the explosion of incredible processing capabilities in low power and low cost packages such as the Raspberry Pi [1] and the Xilinx Zynq [2] means that real-time processing is available for even the cheapest application. Nevertheless, virtually all descriptions of PID controllers are in continuous time [3], [4]. Even in digital control texts [5], [6], [7], the only discretization used is a backwards rectangular rule, without explanation of why this was chosen. This is curious, since backwards rectangular rule equivalents are so absent of discussions of control as to only be available in Matlab when using one of its PID specific tools. This may result from many applications such as process control [8] being slow enough – compared to the digital hardware – that almost any discretization will work, but this author has worked mostly in mechatronics in which the dynamics are significantly more in frequencies affected by the discretization phase hit [9], [10], [11]. This and the fact that discrete PIDs often maintain the structure in firmware of three separate blocks rather than a second order filter, has caused a further examination of these aspects from a practical view.

Furthermore, the specification of PID controllers follows several different conventions, both in the literature [3], [6], [7], [12], [13], [14] and in the commercial specifications. For the purchaser of a commercial PID system, this can make modeling of the system difficult, as the controller can't be modeled without knowing the specification used.

While loop shaping descriptions of PID controllers have been used in many references [8], [12], [15], [16] (among others), the philosophy here is to take the loop shaping specifications (alternately described as filter needs [9]) and return them to the three distinct PID terms. This is because in practice, most PID controllers are accessed via the three terms, allowing for practical nonlinearities such as anti-windup to be instantiated. Thus, while knowing how to go from analog PID coefficients, K_P, K_I, and K_D to a loop shaping filter is useful, knowing the reverse path from the loop shaping filter to the analog and digital PID coefficients is far more practical. Finally, these relationships can easily be encoded into design tools such as Matlab, to allow rapid translation from the analysis to the implementation.

The rest of this paper is organized as follows. Section II introduces a framework for different analog PID controllers. Section III discusses different regions of action for a PID controller on a characteristic mechatronic response. Section IV discusses relationships between a classic PID controller specification (without derivative filtering) and some second order filters. Section V repeats the analysis in Section IV, but with a low pass filter on the derivative section. Section VI discusses the effects of various discretization schemes on the PID.

II. PID Control

Four basic versions of analog PID control equations show up in the control literature and in commercial PID controllers. In the time domain representation those forms are:

$$\begin{aligned} u(t) &= K_P e(t) + \frac{K_I}{T_I}\int_0^t e(\tau)d\tau + K_D T_D \dot{e}(t), & (1)\\ u(t) &= K_P e(t) + K_{I,i}\int_0^t e(\tau)d\tau + K_{D,i}\dot{e}(t), & (2)\\ u(t) &= K_P e(t) + \frac{K_I}{T_I}\int_0^t e(\tau)d\tau + K_D T_D \dot{x_1}(t), & (3)\\ u(t) &= K_P e(t) + K_{I,i}\int_0^t e(\tau)d\tau + K_{D,i}\dot{x_2}(t), & (4)\end{aligned}$$

where $e(t)$ error input to the controller, $u(t)$ is the controller output, and

$$\dot{x_1} = \dot{e} - \frac{a_1}{T_D}x_1 \quad \text{and} \quad \dot{x_2} = \dot{e} - a_1 x_2. \qquad (5)$$

In the frequency domain the four forms for $C(s) = \frac{U(s)}{E(s)}$ are:

$$\begin{aligned} C(s) &= K_P + \frac{K_I}{T_I s} + K_D T_D s, & (6)\\ C(s) &= K_P + \frac{K_{I,i}}{s} + K_{D,i}s, & (7)\\ C(s) &= K_P + \frac{K_I}{T_I s} + K_D\frac{T_D s}{T_D s + a_1}, & (8)\\ C(s) &= K_P + \frac{K_I,i}{s} + K_{D,i}\frac{s}{s+a_1}. & (9)\end{aligned}$$

*Daniel Y. Abramovitch is a system architect in the Mass Spectrometry Division, 5301 Stevens Creek Blvd., M/S: 3U-DG, Santa Clara, CA 95051 USA, danny@agilent.com

For ease of explanation, we will keep to the frequency domain forms. In 8 and 9 we have chosen the derivative filter gain so that – in combination with the derivative – it has a high frequency gain of 1. We could also have chosen to a filter with DC gain of 1. The four forms are chosen by picking two options:

- explicit time specification and
- differentiator filtering.

Explicit time specification simply refers to whether the T_I and T_D terms are present, or whether they are absorbed into K_I and K_D, respectively. It is perfectly legitimate to have

$$K_{I,i} = \frac{K_I}{T_I} \quad \text{and} \quad K_{D,i} = K_D T_D, \tag{10}$$

where $K_{I,i}$ and $K_{D,i}$ can be considered "implicit time" versions of the integral and differential gains. Alternately, the designer can easily go from explicit to implicit time simply by setting $T_D = T_I = 1$. However, leaving the T_I and T_D terms in the equation give the designer some flexibility *and* also allow these terms to drop out when the discrete-time PID is generated. In particular, for the backward rule equivalent of an ideal PID controller with the sample period, $T = T_I = T_D$, the time terms drop out of the equation, making it appear much simpler.

The second option is differentiator filtering. We know that any practical analog differentiator will eventually roll off. It should make sense to explicitly include this in the controller design, but this is not common. Perhaps designers are expecting the plant dynamics and/or circuits to provide low pass behavior. Still, one might wonder why use of a low-pass derivative filter is not a standard practice. This author's best guess is that the most common implementation of a PID controller is a backward rule discrete equivalent approximation. As we will see in Section VI-A, this equivalent puts in its own low pass filter on the differentiator. The typical over-conservatism of the backwards rule equivalent saves the casual designer the trouble and will tend to behave well, especially at low frequencies.

Understanding these four basic forms are useful to a user that has purchased a system that includes a PID controller *e.g.,* the controller of a motion control system. Invariably, the user trying to model these systems will find that one of these forms has been used without it being documented in the product literature. Likewise, technical papers on PID controllers will often default to one of these forms without any discussion about the particular choice Because of this, it is pretty common to see PID gain ranges that vary all over the place, even for the same basic controller. These 4 forms are summarized in Table I.

It should be obvious that we can put these separate terms into one transfer function. What may not be obvious is how this will look once it is combined. Sections IV –VI discuss this.

Equations (6)–(9) can all be related to second order sections and these can be used for loop shaping. We will focus on (6) and (8) since setting $T_I = T_D = 1$ gets these to (7) and (9). Thus, by setting the parameters of the PID, we can set the parameters of the notch. Note that the parameterization will change quite a bit depending upon if we use (6) or (8), and this is because the filter in (8) is acting only on the differentiator.

Form	Time Dom. Equation	Freq. Dom. Equation
Explicit Time, No Filtering	(1)	(6)
Implicit Time, No Filtering	(2)	(7)
Explicit Time, Deriv. Filtering	(3)	(8)
Implicit Time, Deriv. Filtering	(4)	(9)

TABLE I
A SUMMARY OF THE FOUR BASIC FORMS OF ANALOG PID CONTROL WITH REFERENCES TO THEIR ASSOCIATED TIME DOMAIN AND FREQUENCY DOMAIN EQUATIONS.

III. PID REGIONS

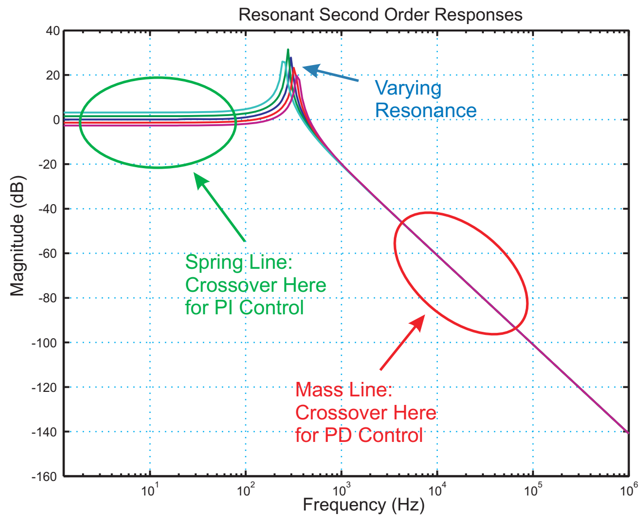

Fig. 1. The regions of PID control for a second order, resonant plant.

In many practical uses, one can consider a PID controller to be operating on a second order or lower plant, such as the one diagrammed in Figure 1. In many cases, the control action is taken and removed well below the resonance, and in this case the proportional plus integral (PI) part of the controller is used, to make the open loop response look like an integrator near gain crossover, and then remove the phase effects before the $-180°$ effect of the resonance. In these cases, the derivative term is seldom used. Similarly, when the resonance is well below the crossover region, the system can be controlled as if it were a double integrator, and in this case the proportional plus derivative (PD) action is used. In these cases, there may still be a motivation to use integral action at low frequency and thus a PID with derivative filtering resembles separate lead and lag controllers. In neither case is precise knowledge of the resonance needed. It is only when the crossover region is relatively close to the resonance that more complex methods, such as those described in [9], [15], [16] become important. The formulas that follow concentrate on the latter, more complex case, while emphasizing the effects of digital implementation as described in [9]. In the case of first order time delay models, such as those found in process control applications [8], full PID control is used, where the PI portion provides low frequency gain and the PD portion adds some lead to compensate for the phase due to the delay, the the requirements of matching a high Q resonance are not present.

IV. Unfiltered Analog PID and Second Order Sections

Starting with (6), let's set $T_D = T_I = T$ and put everything over a common denominator:

$$C(s) = \frac{K_D T}{s}\left[s^2 + \frac{K_P}{K_D}\frac{s}{T} + \frac{K_I}{K_D T^2}\right], \quad (11)$$
$$= \frac{K_D T s^2 + K_P s + \frac{K_I}{T}}{s}. \quad (12)$$

While (11) allows us to solve for the numerator parameters as a second order section, (12) is a standard numerator/denominator form that we might use in Matlab. This form of the PID is not proper, but this is typically mitigated by the way PID controllers are implemented.

The numerator of (11) also has the form of a second order section i.e.,

$$N(s) = \frac{K}{\omega_n^2}\left(s^2 + 2\zeta\omega_n s + \omega_n^2\right), \quad (13)$$

so we should be able to set

$$\frac{K_D T}{s}\left[s^2 + \frac{K_P}{K_D}\frac{s}{T} + \frac{K_I}{K_D T^2}\right] = \frac{K}{\omega_n^2 s}\left[s^2 + 2\zeta\omega_n s + \omega_n^2\right]. \quad (14)$$

If we were simply to try to match $N(s)$ in (13) then we might choose to match the DC gain to some prespecified value, K. However, the form that we want our PID to match has infinite DC gain, so we need to pick a frequency and gain that we wish to match and then evaluate the right side of (14). If:

$$\frac{N(s)}{s} = \frac{K}{s\omega_n^2}\left(s^2 + 2\zeta\omega_n s + \omega_n^2\right), \quad (15)$$

then

$$\frac{N(j\omega_0)}{j\omega_0} = \frac{K}{j\omega_0\omega_n^2}\left(\omega_n^2 - \omega_0^2 + j\frac{\omega_n\omega_0}{Q}\right), \quad (16)$$

where $Q = \frac{1}{2\zeta}$. If we pick our desired gain, K_0, at a certain frequency, $\omega_0 = 2\pi f_0$, then we get

$$K_0 = \left|\frac{N(j\omega_0)}{j\omega_0}\right| = \frac{K}{\omega_0\omega_n^2}\sqrt{(\omega_n^2 - \omega_0^2)^2 + \left(\frac{\omega_n\omega_0}{Q}\right)^2}. \quad (17)$$

This can be solved for K via

$$K = \frac{K_0\omega_0\omega_n^2}{\sqrt{(\omega_n^2 - \omega_0^2)^2 + \left(\frac{\omega_n\omega_0}{Q}\right)^2}}. \quad (18)$$

Using (18) to pick K allows us to equate terms in (14) allows us to solve for ω_n, ζ, and Q:

$$\omega_n = \frac{1}{T}\sqrt{\frac{K_I}{K_D}},\ \zeta = \frac{K_P}{2\sqrt{K_I K_D}},\ \text{and} \quad (19)$$
$$Q = \frac{1}{2\zeta} = \frac{\sqrt{K_I K_D}}{K_P}. \quad (20)$$

However, for design, we might want to specify ω_n and ζ or Q and then re-derive the PID gains as a function of those parameters. This should give us a better way of picking K_P, K_I, and K_D, if we know which center frequency and damping we want the controller numerator to achieve.

Let's set ω_n. We also know T, which is our integration and differentiation time, but will also be our sample time. Finally, we set Q (which is equivalent to setting ζ). From (14) we have

$$\frac{K_I}{K_D} = (\omega_n T)^2 \text{ and } \frac{K_P}{K_D} = \frac{\omega_n T}{Q} = 2\zeta\omega_n T. \quad (21)$$

If we now let K_D be our overall controller gain, scaling K_D means scaling K_P and K_I in the same proportions to maintain the desired shape of the compensator. In summary pick K from (18) to set the controller gain. Then

$$K_D = \frac{K}{T\omega_n^2},\ K_I = KT, \text{ and } K_P = \frac{K}{Q\omega_n}. \quad (22)$$

An example of generating PID gains from notch filter parameters is shown in Section VII. An example using a PID to form a notch filter for loop shaping of an AFM control loop is shown in [9].

V. Filtered Differentiator PID and Second Order Sections

Let's set $T_D = T_I = T$ in (8), put everything over a common denominator, and then relate it to the second order section from (13):

$$C_1(s) = (K_P + K_D) \times \left[\frac{s^2 + \frac{K_P a_1 + K_I}{(K_P+K_D)T}s + \frac{K_I a_1}{(K_P+K_D)T^2}}{s(s+\frac{a_1}{T})}\right] \quad (23)$$
$$= \frac{K}{\omega_n^2}\left[\frac{s^2 + 2\zeta\omega_n s + \omega_n^2}{s(s+\frac{a_1}{T})}\right], \quad (24)$$

Since the PID has infinite DC gain, we need to pick a frequency and gain that we wish to match.

$$\frac{N(s)}{s} = \frac{K}{s\omega_n^2}\left(\frac{s^2 + 2\zeta\omega_n s + \omega_n^2}{s + \frac{a_1}{T}}\right), \quad (25)$$

then

$$\frac{N(j\omega_0)}{j\omega_0} = \frac{K}{j\omega_0\omega_n^2}\left(\frac{\omega_n^2 - \omega_0^2 + j\frac{\omega_n\omega_0}{Q}}{j\omega_0 + \frac{a_1}{T}}\right), \quad (26)$$

where $Q = \frac{1}{2\zeta}$. If we pick our desired gain, K_0, at a certain frequency, $\omega_0 = 2\pi f_0$, then we get

$$K_0 = \left|\frac{N(j\omega_0)}{j\omega_0}\right| = \frac{K}{\omega_0\omega_n^2}\frac{\sqrt{(\omega_n^2-\omega_0^2)^2+\left(\frac{\omega_n\omega_0}{Q}\right)^2}}{\sqrt{\omega_0^2+\left(\frac{a_1}{T}\right)^2}}. \quad (27)$$

This can be solved for K via

$$K = \frac{K_0\omega_0\omega_n^2\sqrt{\omega_0^2+\left(\frac{a_1}{T}\right)^2}}{\sqrt{(\omega_n^2-\omega_0^2)^2+\left(\frac{\omega_n\omega_0}{Q}\right)^2}}. \quad (28)$$

Using (28) to pick K allows us to equate terms in (14) allows us to solve for ω_n, ζ, and Q:

$$\begin{aligned}\omega_n &= \frac{\sqrt{b_1b_2}}{T} = \frac{\sqrt{\frac{K_Ia_1}{K_P+K_D}}}{T} \quad \text{and} && (29)\\ \zeta &= \frac{1}{2}\frac{(b_1+b_2)}{\sqrt{b_1b_2}} = \frac{1}{2}\frac{K_Pa_1+K_I}{\sqrt{(K_P+K_D)K_Ia_1}}. && (30)\end{aligned}$$

From the damping factor, ζ, we can compute the quality factor, $Q = \frac{1}{2\zeta}$, of the anti-resonance:

$$Q = \frac{\sqrt{b_1b_2}}{b_1+b_2} = \frac{\sqrt{(K_P+K_D)K_Ia_1}}{K_Pa_1+K_I} \quad (31)$$

We can use (29), (30), and (31) to back out PID gains based on ω_n and ζ or Q. First from (23) and (29) we see that our overall controller gain K is given by

$$\frac{K}{\omega_n^2} = K_P + K_D = \frac{K_Ia_1}{(\omega_nT)^2}. \quad (32)$$

From (24)

$$\frac{K}{\omega_n^2} = K_P + K_D = \frac{K_Pa_1+K_I}{\omega_nT}Q. \quad (33)$$

Now, we can pick the frequency, ω_n, at which to match gains. T will be our integration, differentiation, and eventual sample time, so we will know that. Finally we can pick ζ or Q. If we want to vary $K = K_P + K_D$ then (32) tells us that we need to scale K_Ia_1 by the same amount to keep a constant ω_n. Likewise, (33) tells us that we need to scale $K_Pa_1 + K_I$ in a similar fashion to keep Q constant.

One possibility for doing this is to pick a_1 from physical limitations of the analog hardware, noise issues, or our desired denominator for (23). With a_1 fixed, (32) sets K_I and (33) sets K_P and therefore K_D. However, this is basing things solely on the desire to set the numerator of the controller. We will see in Sections VI-A and VI-B that our discretization method also limits the choice of a_1.

In summary, pick a_1. Then let:

$$K_D + K_P = \frac{K}{\omega_n^2}. \text{ Then} \quad (34)$$

$$K_I = \frac{KT^2}{a_1} \text{ and } K_P = \frac{KT^2}{a_1}\left(\frac{1}{Q\omega_nT} - \frac{1}{a_1}\right). \quad (35)$$

VI. Discrete PID

An ideal PID without differentiator filtering (from (6)) can be discretized using a backward rectangular rule but not the trapezoidal rule. The usually conservative backward rule equivalent allows the use of an unfiltered derivative PID design, since the continuous derivative maps to a zero at $z = 1$ and a pole at $z = 0$. We can't apply the trapezoidal rule to the unfiltered differentiator of (6), because the $z+1$ term in the denominator results in a pole at $z = -1$, which will be an internal oscillatory pole in the compensator. So, while the closed-loop system might be stable, we wouldn't have internal stability.

The filtered differentiator of (8) can be implemented using either backward rectangular or trapezoidal rule as will be shown later.

A. Backward Rectangular Discrete PID

Applying the backward rectangular rule to (6) yields

$$C(z) = K_P + \frac{K_ITz}{T_I(z-1)} + K_DT_D\frac{z-1}{Tz}, \quad (36)$$

and setting $T = T_I = T_D$ we get

$$C(z) = K_P + K_I\frac{z}{z-1} + K_D\frac{z-1}{z}. \quad (37)$$

In terms of z^{-1} this is

$$C(z) = K_P + K_I\frac{1}{1-z^{-1}} + K_D(1-z^{-1}). \quad (38)$$

Note that (37) and (38) have discrete PID gains that are trivially related to the analog PID gains through the sample period, T. Equation (38) is useful for generating the time domain difference equation in 3 separate units, proportional, integral, and derivative. It is the equation from which we would program this controller, as it would make it easy to add an anti-windup piece to just the integral portion despite being harder to analyze in the z domain. We can rewrite (37), though, as:

$$\begin{aligned}C(z) &= \frac{K_P(z-1)(z)+K_Iz^2+K_D(z-1)^2}{z(z-1)}, && (39)\\ C(z) &= \frac{b_0z^2-b_1z+b_2}{z^2-z}, \text{ where} && (40)\\ b_0 &= K_P+K_I+K_D, && (41)\\ b_1 &= 2K_D+K_P, \text{ and } b_2 = K_D. && (42)\end{aligned}$$

Using (40), we can examine the discrete-time properties of the linear model of this PID in Matlab.

Applying the backward rectangular rule to (8) (with the filtered differentiator) yields

$$C(z) = K_P + K_I\frac{Tz}{T_I(z-1)} + K_D\frac{T_D\frac{z-1}{Tz}}{T_D\frac{z-1}{Tz}+a_1}. \quad (43)$$

Again, setting $T = T_I = T_D$ we get

$$C(z) = \frac{U(z)}{E(z)} = K_P + K_I\frac{z}{z-1} + K_D\frac{z-1}{z(1+a_1)-1}, \quad (44)$$

or equivalently

$$C(z) = K_P + K_I\frac{z}{z-1} + K_D\frac{\frac{1}{1+a_1}(z-1)}{z-\frac{1}{1+a_1}}, \tag{45}$$

or in terms of z^{-1}

$$C(z^{-1}) = K_P + K_I\frac{1}{1-z^{-1}} + K_D\frac{\frac{1}{1+a_1}(1-z^{-1})}{1-\frac{1}{1+a_1}z^{-1}}. \tag{46}$$

Equation (46) is useful for generating the time domain difference equation in 3 separate units. For linear analysis, if we set

$$\alpha = \frac{1}{1+a_1} \tag{47}$$

we can combine the elements of (45) as:

$$\begin{aligned} C(z) &= K_P + K_I\frac{z}{z-1} + K_D\frac{\alpha(z-1)}{z-\alpha}, && (48)\\ C(z) &= \frac{b_0z^2 - b_1z + b_2}{(z^2-(1+\alpha)z+\alpha)}, \text{ where} && (49)\\ b_0 &= K_P + K_I + K_D\alpha && (50)\\ b_1 &= K_P(1+\alpha) + K_I\alpha + 2K_D\alpha && (51)\\ b_2 &= K_P + K_D\alpha && (52) \end{aligned}$$

Equation (49) is the one that we can use for linear analysis in the z domain. The controller will have stable roots for all $a_1 > 0$, so we have changed the character of the discrete differential term. Without the filtering, it had a pole at $z = 0$. With the filter, the pole is at $z = \frac{1}{1+a_1}$ which moves from being close to $z = 1$ with a_1 small to close to $z = 0$ with a_1 very large.

B. Trapezoidal Discrete PID

Applying the trapezoidal rule to (8) works as well because the filtering of the differentiator removes the possibility of a ringing pole in the compensator:

$$C(z) = K_P + K_IT\frac{z+1}{T_I2(z-1)} + K_D\frac{T_D\frac{2}{T}\frac{z-1}{z+1}}{T_D\frac{2}{T}\frac{z-1}{z+1}+a_1}. \tag{53}$$

Setting $T = T_I = T_D$ we get

$$C(z) = K_P + \frac{K_I}{2}\frac{z+1}{z-1} + K_D\frac{2\frac{z-1}{z+1}}{2\frac{z-1}{z+1}+a_1}. \tag{54}$$

This reduces to

$$C(z) = K_P + \frac{K_I}{2}\frac{z+1}{z-1} + K_D\left(\frac{2}{2+a_1}\right)\frac{z-1}{z-\frac{2-a_1}{2+a_1}}. \tag{55}$$

which has a stable differentiator for all $a_1 \geq 0$. In terms of z^{-1} this is:

$$C(z) = K_P + \frac{K_I}{2}\frac{1+z^{-1}}{1-z^{-1}} + K_D\left(\frac{2}{2+a_1}\right)\frac{1-z^{-1}}{1-\frac{2-a_1}{2+a_1}z^{-1}}. \tag{56}$$

Equation (56) can be used to implement the PID in the time domain. However, it is more common to implement each of the sections independently and vary the gains as if it were an analog controller. For analysis purposes, if we set

$$\beta = \frac{2}{2+a_1} \quad \text{and} \quad \gamma = \frac{2-a_1}{2+a_1} \tag{57}$$

then (55) can be reduced to a single fraction to allow one to evaluate the lag-lead behavior of the PID in the z domain.

$$\begin{aligned} C(z) &= K_P + \frac{K_I}{2}\frac{z+1}{z-1} + K_D\beta\frac{z-1}{z-\gamma}, && (58)\\ C(z) &= \frac{b_0z^2 - b_1z + b_2}{z^2-(1+\gamma)z+\gamma} \text{ where} && (59)\\ b_0 &= K_P + \frac{K_I}{2} + K_D\beta, && (60)\\ b_1 &= K_P(1+\gamma) + \frac{K_I}{2}(\gamma-1) + 2K_D\beta, \text{ and} && (61)\\ b_2 &= K_P\gamma - \frac{K_I}{2}\gamma + K_D\beta.. && (62) \end{aligned}$$

Equation (59) is the one we would use for linear analysis of the controller. The poles of (59) are easy to determine at $z = 1$ and $z = \gamma$, but the zeros of this controller are much more complicated, as seen from (60) – (62). The use of the trapezoidal rule and the derivative filtering makes the relationship of the physical gains to the discrete equation gets more convoluted than with the backward rule equivalent that has no derivative filtering, as seen in (39)–(42).

VII. Design Example

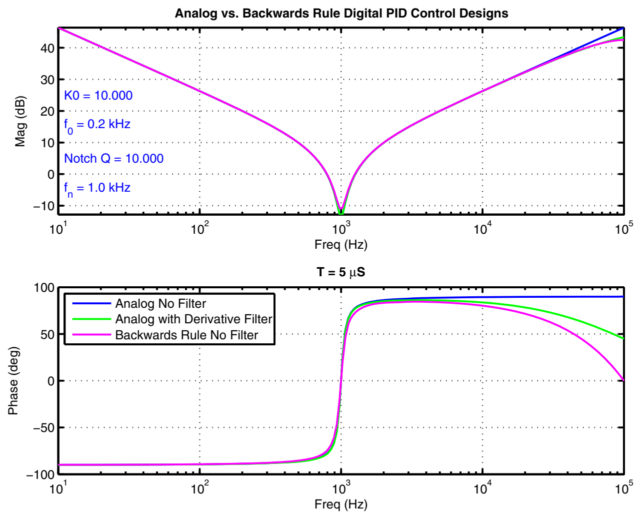

Fig. 2. Continuous and discrete PID controller response derived from notch filter model. The backwards rule version is based on the unfiltered continuous design.

This section shows how the methods in this paper can be applied to a particular loop design. The parameters of the notch filter model are $f_n = 1$ kHz and $Q = \frac{1}{2\zeta} = 10$. The parameters $K_0 = 10$ at $f_0 = 200$ Hz, mean that the parameters are adjusted so that the filter has a gain of 10 at 200 Hz. The integration time, T_I, differentiation time, T_D, and eventual sampling time T_S are all set equal to $T = 5\mu S$. In the case of design based upon a filtered differentiator, the corner frequency is chosen to be at 50 kHz.

We can see from Figure 2 that the net effect of the backwards rule is to place a low pass filter on the differentiator portion. (The pole at $z = 0$.) The main issue is that we do not have design freedom with choice of this

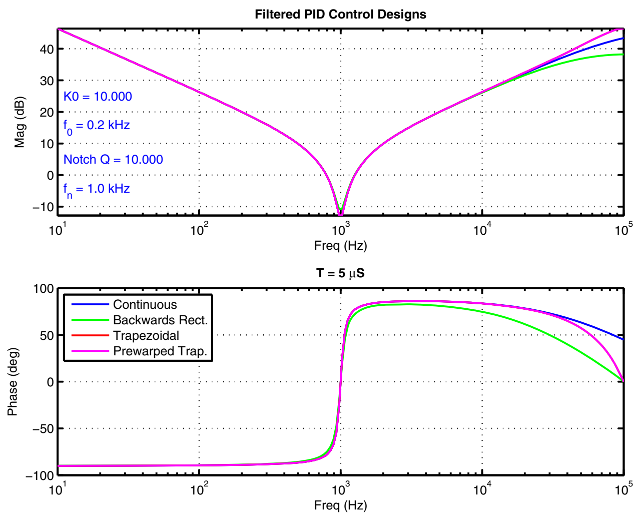

Fig. 3. PID controller response derived from notch filter model. This version has a low pass filter on the derivative. Note that the prewarped trapezoidal and the trapezoidal rule are virtually identical, as the prewarp frequency is at 1 kHz, far below the 200 kHz sample frequency.

filter, but for many the simplicity of a simple translation between analog PID parameters and digital PID parameters is worth this. The comparison in Figure 3 shows that with the trapezoidal rule equivalent of a filtered analog design, we can minimize the phase penalty. This is most useful when dynamics at higher frequency require equalization, such as with a multinotch [11] and we do not want to be limited by our PID implementation.

VIII. Conclusions

This paper presented a unified framework for analog and digital PID controllers based on a set of second order relationship and some assumptions about filtering. The uniform treatment allows a designer to read different PID specifications in the literature and easily relate them in a common way. These relationships also allow

- better design of continuous time PID controllers by extracting the gains from standard controller forms,
- better implementation of discrete PID controllers through a better understanding of their relationships to the continuous forms, and
- simplified linear analysis of PID controllers.

These relationships seem absent from the standard discussions of PID controllers [6], [7], [12], [13]. However, by using these relationships, the design of PID controllers can be done using the standard filter design methods most common in control engineering.

Furthermore, the framework leads one to understanding why so many practical PID implementations are in the forms of Equations 37 or 38, since the gains translate trivially from analog to digital and the typical excessive conservativeness of the backwards rectangular rule allows less careful designers to ignore any filtering issues. A more complete discussion of filtering was avoided due to length issues for the paper, but it is clear that one can use higher order filtering on the derivative section or on the entire PID [12].

The formulas for translation between different forms are especially useful in creating design scripts in Matlab or Octave, as was applied to AFM controller design in [9]. In this paper, an AFM actuator is measured and fit to a second order model which is then controlled using a PID with gains picked so as to notch out the main resonant behavior. Moreover, the understanding here can be viewed as part of a larger context for mechatronic control system design [17], in which the PID design is coupled with filter design [11], [18] and frequency response measurements [19], [17].

References

[1] E. Upton and G. H. and, *Raspberry Pi User Guide*. John Wiley & Sons, third ed., 2014.

[2] L. H. Crockett, R. A. Elliot, M. A. Enderwitz, and R. W. Stewart, *The Zynq Book: Embedded Processing with the Arm Cortex-A9 on the Xilinx Zynq-7000 All Programmable Soc*. Strathclyde Academic Media, 2014.

[3] M. A. Johnson and M. H. Moradi, eds., *PID Control: New Identification and Design Methods*. London: Springer-Verlag, 2005.

[4] K. Ogata, *Modern Control Engineering*. Prentice-Hall Instrumentation and Controls Series, Englewood Cliffs, New Jersey: Prentice-Hall, fourth ed., 2001.

[5] K. M. Moudgalya, *Digital Control*. John Wiley & Sons, 2007.

[6] G. F. Franklin, J. D. Powell, and M. L. Workman, *Digital Control of Dynamic Systems*. Menlo Park, California: Addison Wesley Longman, third ed., 1998.

[7] K. J. Åström and B. Wittenmark, *Computer Controlled Systems, Theory and Design*. Englewood Cliffs, N.J. 07632: Prentice Hall, second ed., 1990.

[8] B. W. Bequette, *Process Control: Modeling, Design, and Simulation*. Prentice Hall, 2006.

[9] D. Y. Abramovitch, S. Hoen, and R. Workman, "Semi-automatic tuning of PID gains for atomic force microscopes," in *Proceedings of the 2008 American Control Conference*, (Seattle, WA), AACC, IEEE, June 11–13 2008.

[10] D. Y. Abramovitch, S. B. Andersson, L. Y. Pao, and G. Schitter, "A tutorial on the mechanisms, dynamics, and control of atomic force microscopes," in *Proceedings of the 2007 American Control Conference*, (New York, NY), pp. 3488–3502, AACC, IEEE, July 11–13 2007.

[11] D. Y. Abramovitch, "The Multinotch, Part I: A low latency, high numerical fidelity filter for mechatronic control systems," in *Proceedings of the 2015 American Control Conference*, (Chicago, IL), pp. 2161–2166, AACC, IEEE, July 2015.

[12] K. J. Åström and T. Hägglund, *Advanced PID Control*. Oxford Series on Optical and Imaging Sciences, ISA Press, August 15 2005.

[13] T. Wescott, "PID without a PhD," *Embedded Systems Programming*, pp. 86–108, October 2000.

[14] G. F. Franklin, J. D. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*. Upper Saddle River, New Jersey: Prentice Hall, fifth ed., 2006.

[15] W. C. Messner, M. D. Bedillion, L. Xia, and D. C. Karns, "Lead and lag compensators with complex poles and zeros," *IEEE Control Systems Magazine*, vol. 27, pp. 44–54, February 2007.

[16] W. Messner, "Formulas for asymmetric lead and lag compensators," in *Proceeding of the 2009 American Control Conference*, (St. Louis, MO), pp. 3769–3774, AACC, IEEE, June 2009.

[17] D. Y. Abramovitch, "Trying to keep it real: 25 years of trying to get the stuff I learned in grad school to work on mechatronic systems," in *Proceedings of the 2015 Multi-Conference on Systems and Control*, (Sydney, Australia), IEEE, IEEE, September 2015.

[18] D. Y. Abramovitch, "The Multinotch, Part II: Extra precision via Δ coefficients," in *Proceedings of the 2015 American Control Conference*, (Chicago, IL), pp. 4137–4142, AACC, IEEE, July 2015.

[19] D. Y. Abramovitch, "Built-in stepped-sine measurements for digital control systems," in *Prodeedings of the 2015 Multi-Conference on Systems and Control*, (Sydney, Australia), IEEE, IEEE, September 2015.