Practical Methods Workshop: An Introduction

Daniel Y. Abramovitch
System Architect, Agilent Technologies
E-mail: danny@agilent.com

March 8, 2022; 12:07 PM

Abstract

This portion of the workshop is the general overview and introduction. We will present some overall ideas and themes that will recur time and again during the rest of the workshop. Problem areas will be introduced, which we plan to shed some light on later in the day.

1 Introduction

"Practical Methods for Control" refers to the stuff one has to do when one wants to do controls in the real world. It is the crossing of the proverbial gap between theory and practice to get theory that can be used in practice and practical systems that are far more empowered by analysis and design. The difference between idealized analysis and the grubby details of implementation can be seen in comparing Figures 1 and 2. Figure 1 is the type of diagram

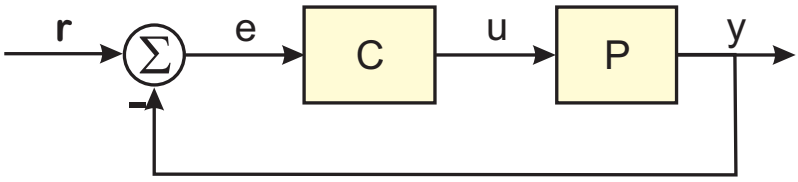


Figure 1: A generic control loop.

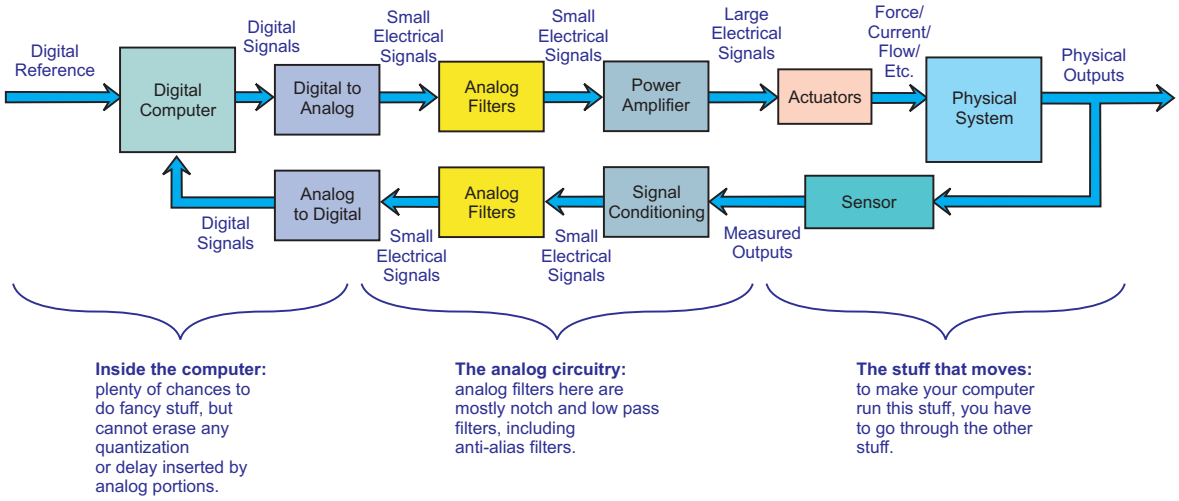


Figure 2: A more detailed, but still generic, digital control loop.

we would associate with analysis, but to implement a control system, we need to consider all the components in Figure 2. The main difference, conceptually, is that there are a lot more pieces between the Physical System and the controller implemented in the Digital Computer. Typically, to get from Figure 2 to Figure 1, a lot of those pieces get either swept into P or C. For eample, C might include the Digital Computer, and the converters. The power electronics, actuators, and sensors may very well be wrapped up in P. The analog filters might go in either direction, depending upon the design style.

That being said, while compressing all those blocks into P and C might help with analysis, that involves an assumption that this simplification doesn't significantly affect the response. If that were always the case, we would not need to worry about practical methods. Another way to look at the two sides of the great divide is to consider them as a form of responsive

reading.

One side is the academic/theoretical (AT) side, where one starts with analytical models, CAD tools (e.g. MATLAB), and no end of advanced algorithms. Here, all the cool kids use state-space methods.

On the other side is the implementation/industrial (II) world, where experience often is considered more important than analysis, where models – if they show up – are first or second order (plus some crap to be eliminated separately) where the kinds of controllers used are "Both kinds: PID and three-term."

Our goal here is not to close this proverbial, 50-year gap, but to use some "simple tricks and nonsense" [1] to get a few rope bridges across it. Let's throw out a few ideas. As we will sometimes have to do, getting the general idea will involve some broad generalizations and simplifications.

- In academic/theoretical (AT) problems, we start with a model.
- In implementation/industrial (II) problems, getting to a reasonable model is 85–95% of the control design.
- In academic/theoretical (AT) problems, we often analyze in continuous time (CT).
- In implementation/industrial (II) problems, we almost always implement using a digital computer. It may – and usually does – connect to the real world via analog circuitry, but the control law, parameters, decision making, and intelligence are almost all digital these days.
- In AT problems, the models are "mostly" linear, time-invariant (LTI), but can be quite large.
- In II problems, the models – if the are written down – are first or second order. When the physical system presents higher order properties, a lot of effort is made to beat down, segment, divid up, etc. the problem into second order chunks.

  There are exceptions, mostly in the high-end "fighter-plane" problems. These problems usually have the characteristic that multiple engineers are needed to tune/adjust a single instance of the design.
- In AT problems, the filtering of extra dynamics is often folded into the control design.

- In II problems, filter blocks are added along the analog/digital path in a piecemeal way to remove imperfections in the signal path close to where they occur, thereby isolating them from the rest of the system.
- In AT problems, a more complex algorithm that produces a slightly better response on some metric leads to a publication.
- In II problems, the goal is to make a reliably working system, a device that operates, a product that sells.
- In AT problems, the nonlinearities are often added in at the end to make the problem "more real".
- In II problems, the nonlinearities usually frame and limit the scope of the control design.
- People working on AT problems would love to get their advanced algorithms into hardware on physical systems, but it's hard to make that stuff work.
- People working on II problems would love to add advanced algorithms to their physical systems, but it's hard to make that stuff work.

We will follow up this set of broad, oversimplified, massively generalized assumptions with a few premises:

- State space filtering and control are model based filtering and control, and model based control and filtering require (wait for it . . .) a good model.
- Extracting models from measurements is imperfect and depends strongly on the device/system being measured.
  - Certain physical systems admit only certain types of measurements.
  - Each measurement class only gives a certain amount of information about a system and no one measurement gives a complete picture.
- All the measurements are digital, i.e. they result in sampled data that gets fed into a computer.
- Most physical systems are engineered to present a low order model – at least in their basic, low frequency behavior.

Therefore, in trying to understand practical control methods:

- We will pick some representative low order models that demonstrate what we see in the physical world (Section 2).
- We will talk about what kinds of measurements we can make to determine the parameters of these models.
- More importantly, we will talk about how to write code (or what type of code needs to be written) to make the most of these measurements.
- We will discuss why for any first or second order model, a well tuned digital PID and some filters have all the needed degrees of freedom to implement excellent (not optimal) control.
  - We will give a common framework for PID controllers.
  - We will discuss the different ways to design PID controllers based upon what model measurements are available.
  - We will discuss the hows and whys of PID discretization.
  - We will talk about what to expect from a closed-loop PID response on one of our canonical physical models.
- We will then move beyond the simple models to discuss the hows and whys of compensating extra dynamics. In particular, much is made of filter design with little thought to how to implement compensating filters. Someone has to go there. It might as well be us.
- A key factor in being able to intelligently design control systems is the ability to rapidly iterate between measurements, modeling, design, and implementation. We will discuss the types of data connections needed between these devices, and give examples of how this can be done.
- We can't do computer control without knowing something about computers, and how to get data in and out of them, so we will discuss computational models and different methods of getting control implemented on digital computers.

It makes no senose to discuss modern control systems without some discussion of the effect of sampling on physical system models. In particular, the obfuscation of physical parameters

created in any sampling scheme should make us reconsider most identification for practical systems.

In Section 2, we will present a set of low order models that can be considered representative of the types of practical systems that get controlled with simple controllers, such as PIDs. These models are helpful because in intially looking here, we are able to frame the discussion of model identification from measurements. In particular, we can see what parameters can be extracted from step responses, and how practical use of step response must be coupled with an assumption of one of these model types [2]. These same models are also useful in understanding possible closed-loop responses of ideal PID controllers [3].

So, having simple, linear models can help us get a "best we can do" idea for our control systems. In our discussions of practical identification, we will see how the best information we can get out of step responses depends on assuming one of these ideal models. In the section on PIDs, we will see what we can say by assuming one of these models and picking one of the PID forms.

Of course, knowing the ideal response would be great except that "stuff happens", and that stuff makes things worse that the "best we can do". We will talk about these in Section 3. This stuff limits our use of simple models and forces us to think about a lot of other issues.

Because discretization has such a strong effect on almost all our control systems, we will then give a useful introduction in Section 3.1.

2 Low Order Models

In this section, we present a handful of low order models that describe basic behaviors we see in practical control systems. These models will be used both in extracting parameters from measurements and in understanding the closed-loop behavior of various PID forms (P, PI, PD, and PID) on these models. Moreover, these models are quite representative of ones seen in practice, and so we can use them to get insights into what can and cannot be extracted from different measurement types to quantify our systems. How do we get from first principles (science) to models that help us do better control designs? What parts of our models can we verify from actual measurements? How do we work knowledge of current

working loops into modeling for improved performance of the same system?

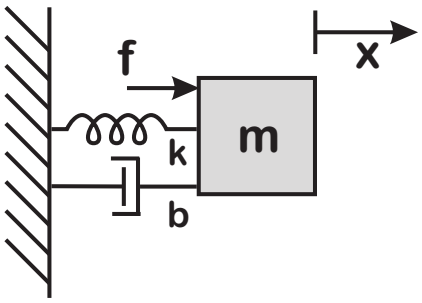


Figure 3: Spring-mass-damper system represents the typical second order mechatronics system.

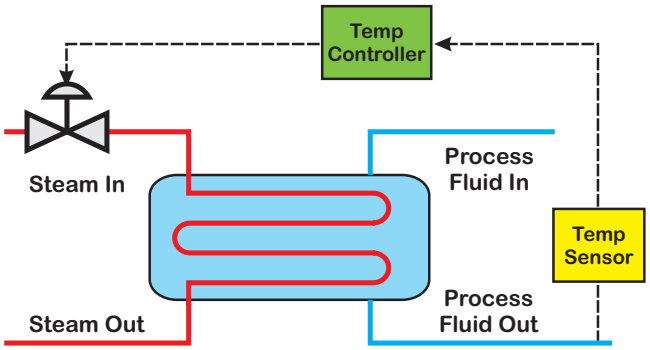


Figure 4: The heat exchanger is a classic first order system with delay.

It turns out that a lot of the simple, practical control problems that are solved using a PID or lead/lag controller can be classified as one of a handful of first or second order models. Of course, a closer look often reveals a lot of extra dynamics, but a vast number of problems end up being modeled this way. For this reason, a lot of practical control designs can be considered to be using one of these or a combination of them.

In particular, we can consider the following simple systems. They can be considered as the "rigid body mode" or baseband portion of more complex systems. When possible, we will discuss where these models arise. We will also show a typical Bode plot and note the key characteristics of that plot for each model

Note that we are giving physical models in continuous time although most controllers that we will discuss and most measurements that we will make are done with computers in discrete

time. For our purposes, the continuous time is far closer to the physical system and therefore gives cleaner understanding. We will spend some time on how to reconcile the continuous time models with discrete time measurements.

2.1 Integrator

$$\frac{X(s)}{F(s)} = H(s) = \frac{K}{s} \tag{1}$$

An integrator may well be the easiest physical system to control [4]. In fact, many controller designs first try to make the open loop look like an integrator before closing the loop on that integrator.

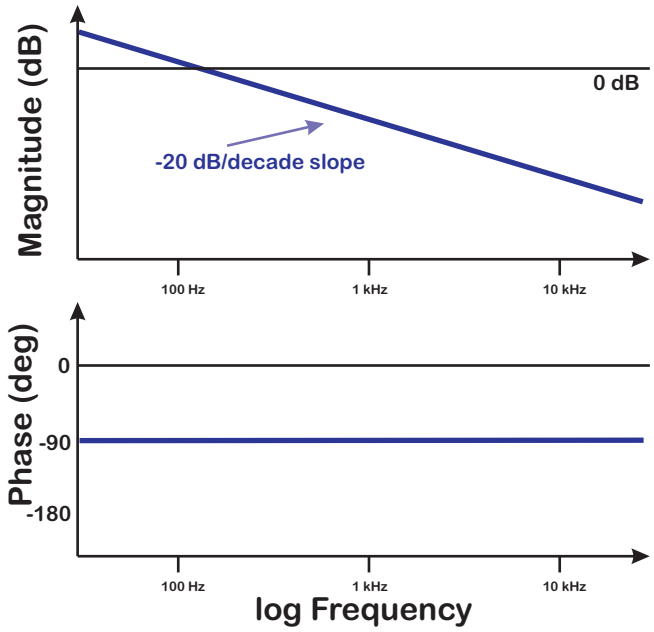


Figure 5: A schematic Bode plot response of a single integrator model. It is characterized by a phase that is flat at -90° and a magnitude that drops off at -20 dB/decade. It is stable. The addition of delay has no effect on the magnitude response but adds "negative phase" that gets more negative as the frequency gets higher.

An integrator may be one of the easiest systems to control. From a PID perspective, any of the combinations, P, PI, PD, and PID, can control this loop with (theoretically) no upper

limit on gain.

2.2 Integrator with Delay

If the transport delay in the system is a significant portion of the response, then we modify Equation 1 to include this time delay factor.

$$\frac{X(s)}{F(s)} = H(s) = \frac{K}{s}e^{-sT_D} \tag{2}$$

If one generates a Bode plot of Equation 2, then the magnitude response is identical to that of Equation 1, but the phase response is significantly different. In time this corresponds to a system with impulse response of

$$\begin{array}{ll} h(t) = K1(t-T_D) & \textbf{for } t - T_D \geq 0 \\ 0 & \textbf{otherwise} \end{array} \tag{3}$$

Includind delay always makes things worse in a system model, because eventually the growing negative phase takes away all the phase margin. Thus, while all PID variants can control this model, there is an upper limit on the gain for any of them.

2.3 First Order Low Pass with Delay

Pure integrators are nice constructs in theory, but hard to find in the physical world. Often they are leaky, which means that rather than the stead response to a step, their impulse response is

$$\begin{array}{ll} h(t) = Kae^{-a(t-T_D)} & \textbf{for } t - T_D \geq 0 \\ 0 & \textbf{otherwise} \end{array} \tag{4}$$

which has the transfer function of

$$\frac{X(s)}{F(s)} = \frac{Ka}{s+a}e^{-sT_D}. \tag{5}$$

This equation in one form or another is typically used to model problems such as flow problems or heat exchangers [5, 6, 7], diagrammed in Figure 4. As such it is a fundamental system in chemical process control. Many systems, especially in heat flow, biology, and chemical process control, are adequately described by this model. As with the integrator with delay, all the PID variants can control this model, but there is an upper bound on the gain determined by the variant used, the delay, and the pole of the filter.

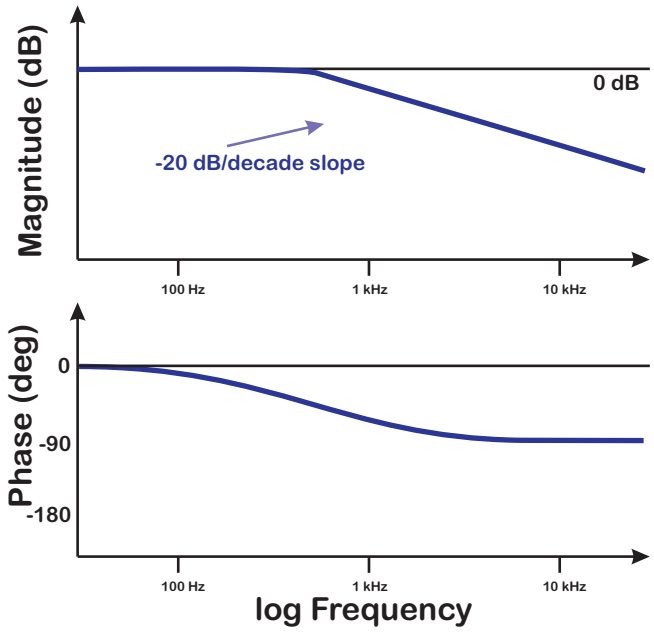


Figure 6: A schematic Bode plot response of a first order model. It is characterized by a phase that goes from 0° down to -90° and a magnitude that is flat and then drops off at -20 dB/decade. It is stable. The addition of delay has no effect on the magnitude response but adds "negative phase" that gets more negative as the frequency gets higher.

2.4 Bilinear Filter

When there is differentiation as well as integration, then there are dynamics in the numerator of the transfer function, as seen in Equation 6:

$$\frac{X(s)}{F(s)} = K\left(\frac{a_{1c}}{b_{1c}}\right)\left(\frac{s+b_{1c}}{s+a_{1c}}\right) \tag{6}$$

This is the form that analog lead circuits, used to stabilize systems by providing band limited differentiation. It is also the form that an analog lag filter takes. The key difference is that for a lead, $b_{1c} < a_{1c}$ and for a lag $b_{1c} > a_{1c}$.

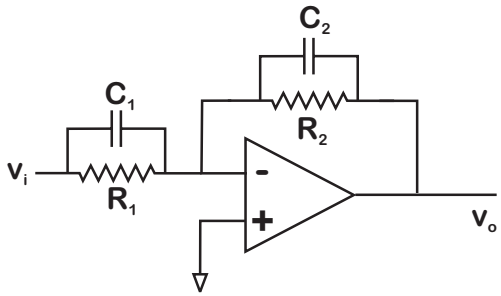


Figure 7: A generic simple circuit diagram of a bilinear filter.

An simple, generic op-amp circuit is shown in Figure 7. If we do standard op amp circuit analysis, we have a single current going from V_o to ground and from ground to V_i so that

$$i = \frac{V_o}{Z_o} = -\frac{V_i}{Z_i}, \text{ or} \tag{7}$$

$$\frac{V_o}{V_i} = -\frac{Z_o}{Z_i}. \tag{8}$$

We have

$$Z_o = \frac{\frac{R_2}{sC_2}}{R_2 + \frac{1}{sC_2}} = \frac{\frac{1}{C_2}}{s + \frac{1}{R_2C_2}}. \tag{9}$$

Likewise

$$Z_i = \frac{\frac{R_1}{sC_1}}{R_1 + \frac{1}{sC_1}} = \frac{\frac{1}{C_1}}{s + \frac{1}{R_1C_1}}, \tag{10}$$

so that

$$\frac{Z_o}{Z_i} = \frac{\frac{\frac{1}{C_2}}{s+\frac{1}{R_2C_2}}}{\frac{\frac{1}{C_1}}{s+\frac{1}{R_1C_1}}} = \left(\frac{C_2}{C_1}\right)\left(\frac{s + \frac{1}{R_1C_1}}{s + \frac{1}{R_2C_2}}\right). \tag{11}$$

Finally,

$$\frac{V_o}{V_i} = -\left(\frac{C_2}{C_1}\right)\left(\frac{s + \frac{1}{R_1C_1}}{s + \frac{1}{R_2C_2}}\right). \tag{12}$$

Comparing Equations 6 and 12, we see that $b_{1c} = \frac{1}{R_1C_1}$ and $a_{1c} = \frac{1}{R_2C_2}$, so we can get a lead if $R_1C_1 < R_2C_2$ and a lag if $R_1C_1 > R_2C_2$. We also have

$$K\left(\frac{a_{1c}}{b_{1c}}\right) = -\left(\frac{C_2}{C_1}\right) \tag{13}$$

although making this exact would depend upon finding the right resistors and capacitors. The simple diagram of Figure 7 is not a very good circuit from an analog design point of view, but it does make for an easy to understand analysis. The other thing is that this simple circuit is inverting, so we would likely follow it with an inverting voltage follower circuit to buffer the signal and change the sign (Figure 8) or use a different configuration going into the non-inverting input of the op-amp.

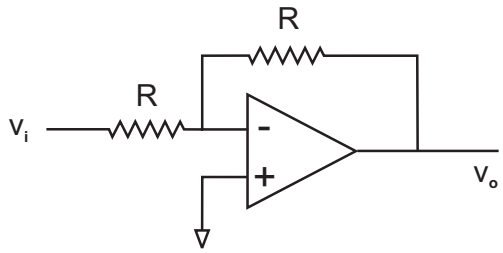


Figure 8: A generic simple circuit diagram of an inverter.

2.5 Double Integrator

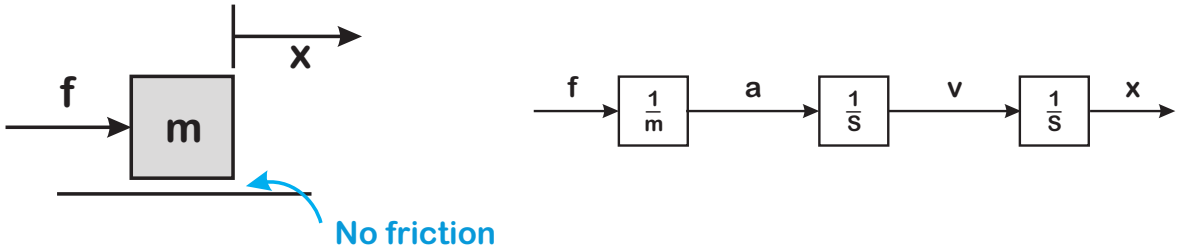


Figure 9: A pure mass on a frictionless plane is one physical system that results in a double integrator model.

Another model that shows up in countless idealized control problems is the double integrator (Figure 9), so named because it is described by the solution to the differential equation,

$$\begin{array}{ll} \ddot{x}(t) = 0 + Kf & \textbf{for } t \geq 0 \\ 0 & \textbf{otherwise} \end{array} \tag{14}$$

where f is the input to the system. This has the transfer function of

$$\frac{X(s)}{F(s)} = \frac{K}{s^2}, \tag{15}$$

and is seen in most physics problems as a mass on a frictionless plane. If $x(t)$ is displacement, then we get back Newton's $f = ma$ where $\ddot{x} = a$ and $K = 1/m$.

This model is the starting point for many simple electro-mechanical systems. In fact, some papers and texts really never get to the point of adding extra dynamics because they are hard to deal with. However, they do represent the behavior of a 3-axis staiblized satellite (where each axis behaves as a double integrator), since there is no friction in space or wind to deal with. However, this model does not describe the more flexible spacecraft parts (such as the robot arms used on the NASA's space shuttle or on the International Space Station.

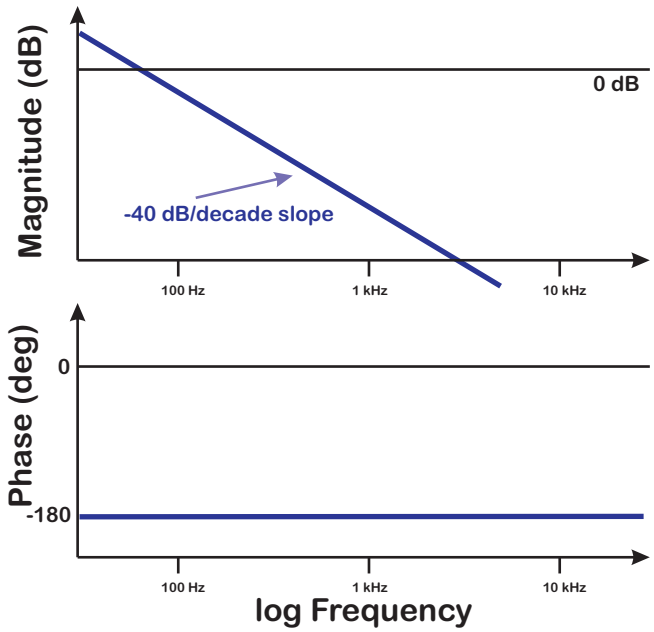


Figure 10: A schematic Bode plot response of a double integrator model. It is characterized by a phase that is flat at -180° and a magnitude that drops off at -40 dB/decade. It is not stable in the sense of bounded output to a bounded input, but it is relatively easy to control.

We will see that even from such a simple model we can predict when each of the forms of PID will work. P and PI will not suffice, but PID can and PD may be the best of all.

2.6 Double Integrator with Delay

If we model the transport delay along with the double integrator, we get:

$$\frac{X(s)}{F(s)} = \frac{K}{s^2} e^{-sT_D} \tag{16}$$

Everything gets worse and more limited with delay, so while our PD controller may work for whatever gain we wish in the double integrator case, the addition of delay will cause us to have to limit the gain based on the negative phase that the delay provides.

2.7 Pure Delay

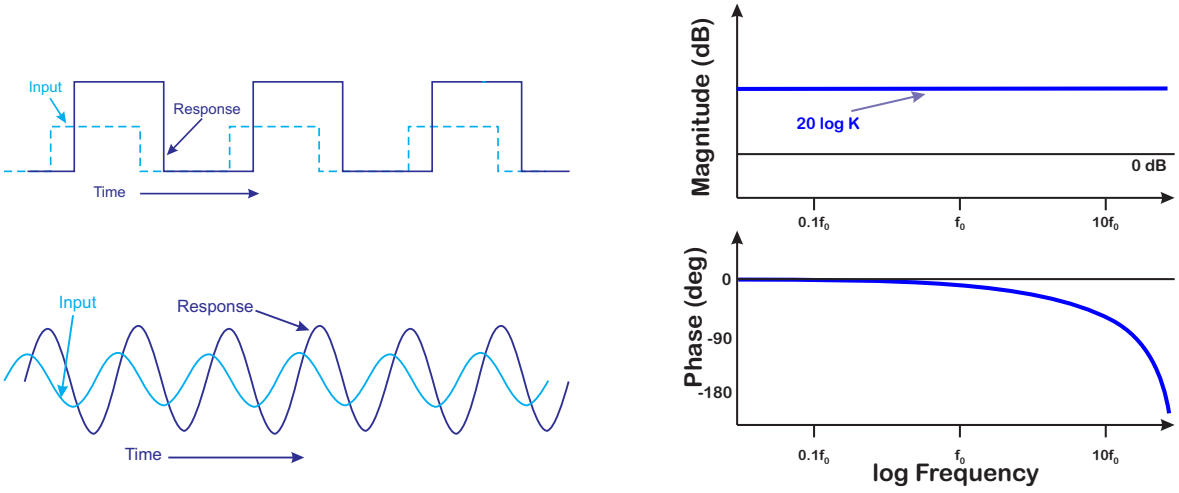


Figure 11: On the left, simple examples of pure time delay in a system. The output reproduces the input, with perhaps some scaling, but no other distortion. On the right is a sketch of the Bode plot this produces: flat gain ($20logK$) and phase going from 0 to increasingly negative with high frequency.

We can also have a model that is a pure time delay, that is, the output reproduces the input but delayed T_D seconds. The transfer function for this is:

$$\frac{X(s)}{F(s)} = Ke^{-sT_D}, \tag{17}$$

and it has a gain of K for all frequencies, but an ever decreasing phase. Signals passsing through lines without attenuation, or simply data passing through a computer system exhibits this kind of behavior. Two examples of such signals are shown on the left of Figure 11. On the right is sketched the resulting Bode plot for the pure delay with gain K.

2.8 Simple Resonance with No Zeros

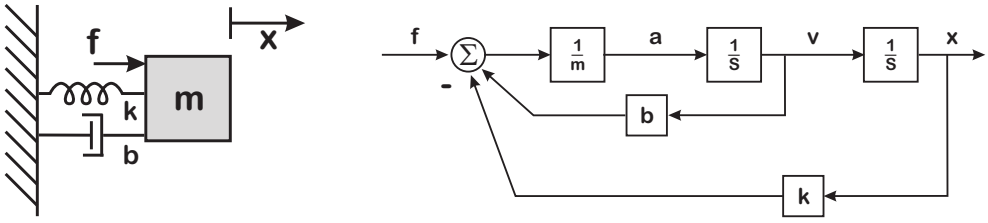


Figure 12: Simple spring-mass-damper problem. On the left is a schematic diagram and on the right is the resulting system block diagram. The schematic diagram can also be obtained by augmenting a double integrator with feedback from both the position and velocity term, resulting in a second order, stable system. If the ratio between the velocity feedback term and the spring feedback term is small enough, we get the complex roots of a resonant system.

The spring-mass-damper system of Figure 3 provides an easy to visualize model for a lot of simple electromechanical systems as well as for a lot of analog circuits. The block diagram of such a physical system is derived from adding position feedback (k, corresponding to the spring term) and velocity feedback (b, corresponding to the damping term) to the diagram of Figure 9 to get Figure 12. The transfer function is:

$$\frac{X(s)}{F(s)} = \frac{\frac{1}{m}}{s^2 + \frac{b}{m}s + \frac{k}{m}}, \tag{18}$$

which can be put into resonance terms:

$$\frac{X(s)}{F(s)} = K\frac{\omega_d^2}{s^2 + 2\zeta_d\omega_d s + \omega_d^2}, \tag{19}$$

by setting $\omega_d^2 = \frac{k}{m}$, $2\zeta_d\omega_d = \frac{b}{m}$, and $K\omega_d^2 = \frac{1}{m}$.

2.9 Simple Resonance with One Zero

$$\frac{X(s)}{F(s)} = K\frac{s + b_{1,c}}{s^2 + 2\zeta_d\omega_d s + \omega_d^2} \tag{20}$$

We will see that even from such a simple model we can predict when each of the forms of PID will work, P and PI if the overall gain is low enough, PD if the gain is high enough, and PID if we have an accurate model.

2.10 Resonance with Anti-Resonance (notch)

We do not usually see this system in isolation since most physical systems exhibit some sort of low pass behavior when we get to high enough frequencies, and some sort of rigid body behavior at the lowest frequencies. Mechatronics engineers are used to seeing these responses out beyond those rigid body behaviors.

$$\frac{X(s)}{F(s)} = K\left(\frac{\omega_d^2}{\omega_n^2}\right)\left(\frac{s^2 + 2\zeta_n\omega_n s + \omega_n^2}{s^2 + 2\zeta_d\omega_d s + \omega_d^2}\right), \tag{21}$$

This model also shows up as a system component. For example, analog conditioning circuits often have notches (anti-resonances in the numerator of (21)) to eliminate problem frequencies. Flexible mechanical systems may have many, many resonance, anti-resonance pairs in them, which would end up as a series of models like Equation 21.

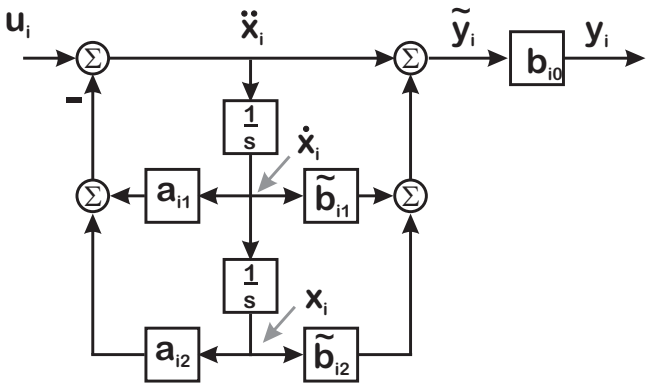


Figure 13: A block diagram of an analog biquad filter, which can implement Equation 21.

Figure 13 shows the biquad as the i^{th} member of a chain of biquads in which $a_{i1} = 2\zeta_{id}\omega_{id}$, $a_{i2} = \omega_{id}^2$, $\tilde{b}_{i1} = 2\zeta_{in}\omega_{in}$, $\tilde{b}_{i2} = \omega_{in}^2$, and $b_{i0} = K\frac{\omega_{id}^2}{\omega_{in}^2}$.

The modeling of such systems in state space using chains of analog (or digital) biquad filters has been described by the author in [8] and [9].

2.11 Some General Ideas

Why does this matter?

1) Because each of these base models has properties that determine what kind of control can be easily and practically be designed and implemented for it.
2) Because we can take a simple parameterization of the most general PID model and apply it to these, we can see the effects of different tuning schemes.

3 Stuff Happens

We will see in later sections how the closed-loop responses of these low order models under the control of various forms of PID controllers (P, PI, PD, PID) and be predicted. However, if linear continuous-time analysis of simple models and simple controllers were sufficient to build great controllers in the real world, this would be a one hour workshop.

Instead, we know that in the real world, stuff happens. Sampling, delay, and noise. Oh my! This section will point out how those turn this workshop into a two-day affair, crammed into one day. Without that stuff, wgood theory would be enough to generate good practical control systems.

3.1 Sampling

As mentioned in Section 1, we will assume that the control law will be implemented in a digital computer (or a computer for anyone born after 1980). By digital computer we may be discussing anything from a Field Programmable Gate Array (FPGA) chip connected to the physical system with Analog to Digital Converters (ADCs) and Digital to Analog Converters (DACs), to MicroControllers, to Digital Signal Processing (DSP) chips, to full processors, to Linux and Windows systems. What we are excluding is the implementation of control laws via analog circuitry or analog computers (the "other" computers). While there are some clear sampling advantages to analog implementation, and these still exist at very high frequencies – where it is hard to sample fast enough and actually do control calculations between the samples, these are increasingly the domain of a small niche. An understanding of analog circuit design is extremely helpful in understanding the interface circuit that connect sensors to the ADCs and DACs to driver electronics and actuators. We hope to touch on this at the end of the workshop, or in a future, extended workshop.

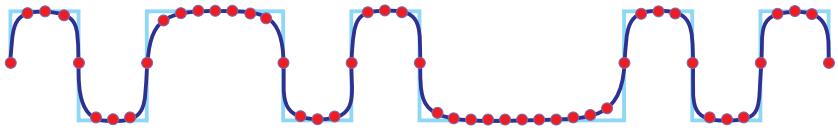


Figure 14: A diagram of real-time sampling of a signal. The ideal signal (cyan) has sharp transitions, but passage through the physical system often has a low-pass filter (LPF) nature to it resulting in a rounded curve (blue). The sampled signal (red) typically (but not always) is produced at regular sample intervals and with a certain level of quantization in the conversion. One can say that the real goal of a sampling system is to recover or infer the behavior of the ideal signal, but this simple diagram points out that even the physical filtering must be taken into account in any attempt to do so.

Realizing that most modern implementations are on digital computers, how do we discuss thinking in analog while implementing in digital? When is "sampling fast" enough and what insights can we gain from paying attention to how we discretize things?

The fact that controllers will be digital means that we should want a very physical, intuitive understanding of how sampling affects our perception of the physical signal (Figure 14), and how it limits what we can do with the controller. In particular, sampling adds delay to the system, delay results in negative phase, and negative phase reduces phase margin. Thus, the

simple act of sampling the data limits our top end bandwidth before anything else can.

The second thing we must understand about sampling is that no one discretization model fully captures our ability to control a system digitally. The workhorse of most discretization in control is the Zero Order Hold (ZOH) Equivalent [10], but even the most cursory look at this method reveals that all physical intuition about any system more complex than a double integrator is lost using this method. We will discuss how using other discretization methods may preserve physicality with a small (and often negligible) loss of mathematical exactness in the model.

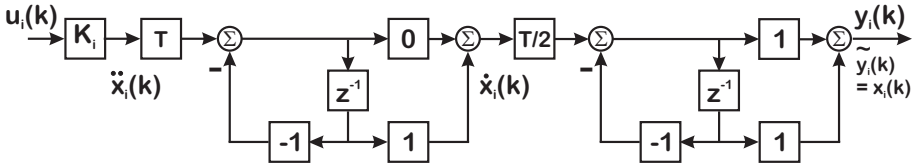


Figure 15: Discrete double integrator BLSS model (ZOH equivalent). In this drawing, we've chosen to make the index, i, as with a biquad stage, but we are explicitly labeling the different integration levels.

A simple example of what is wrong with our understanding of discretization is in discretizing the double integrator of Section 2.5 and Figure 9. In the continuous time drawing on the right, we can easily pick off the velocity term, which makes using both position and velocity measurements available for feedback. Let's remember that for a second order system, access to position and velocity for feedback is full state feedback, and this is the 800 pound gorilla of control theory – it puts the poles anywhere it wants. The situation gets much worse as soon as we discretize. If we use the standard Zero-Order Hold equivalent model and use a Bilinear State-Space form (BLSS) [11] shown in Figure 15, we can access the position, but the intermediate result does not do a good job of representing velocity, since the integrators are not interchangeable. It seems highly illogical to go to the trouble of generating a state-space realization on the basis that state-space gives us access to all the states and somehow lose access to velocity in one of the simplest state-space realizations available [11].

Consider the earlier example of the second order resonance, of Equations 18 and 19:

$$\frac{X(s)}{F(s)} = \frac{\frac{1}{m}}{s^2 + \frac{b}{m}s + \frac{k}{m}} = \frac{K\omega_d^2}{s^2 + 2\zeta_d\omega_d s + \omega_d^2}, \tag{22}$$

or to the more general analog biquad model:

$$\ddot{x} \;\; = \;\; -a_1\dot{x} - a_2 x + u \tag{23}$$

$$y = b_0\ddot{x} + b_1\dot{x} + b_2, \tag{24}$$

with the transfer function description:

$$\begin{aligned}(s^2 + a_1 s + a_2) &= U(s)\\ Y(s) &= (b_0 s^2 + b_1 s + b_2)\,,\end{aligned} \tag{25}$$

yielding

$$\frac{Y(s)}{U(s)} = \frac{b_0 s^2 + b_1 s + b_2}{s^2 + a_1 s + a_2}. \tag{26}$$

A discrete transfer function version of (26)

$$\frac{Y(z)}{U(z)} = \frac{b_{0,D} z^2 + b_{1,D} z + b_{2,D}}{z^2 + a_{1,D} z + a_{2,D}} = \frac{b_{0,D} + b_{1,D} z^{-1} + b_{2,D} z^{-2}}{1 + a_{1,D} z^{-1} + a_{2,D} z^{-2}} \tag{27}$$

The question is what the meaning of the new coefficients in relation to the old ones, and this is entirely related to both the original parameters and the discretization method. Exact discretization methods obscure the coefficient meaning and couple states in a very non-intuitive way. Some other approximations, in which the original transfer function is broken into a cascade of second order sections (biquads) can preserve much physical intuition.

There are lots of discretization methods and even when one does the "exact" math, one doesn't get a satisfying answer. In fact, the exact math can give an answer that is so convoluted as to obscure any hope of physical intuition and this is bad. The Trapezoidal Rule, also known as Tustin's Rule or a bilinear equivalent, substitutes discrete time operators (based on the Z transform) for the continuous time operator (based on the Laplace transform). Using the Trapezoidal Rule, we make the substitution:

$$s \longleftarrow \frac{2}{T}\left(\frac{z-1}{z+1}\right) \tag{28}$$

and if we substitute for s in (26) to get to (27) then we end up with the following mappings:

$$\begin{array}{lcllcl}
\Delta & = & 1 + a_1\frac{T}{2} + a_2\frac{T^2}{4} & & & \\
b_{0,D} & = & \frac{1}{\Delta}\left(b_0 + b_1\frac{T}{2} + b_2\frac{T^2}{4}\right) & a_{0,D} & = & 1 \\
b_{1,D} & = & \frac{2}{\Delta}\left(b_2\frac{T^2}{4} - b_0\right) & a_{1,D} & = & \frac{2}{\Delta}\left(a_2\frac{T^2}{4} - 1\right) \\
b_{2,D} & = & \frac{1}{\Delta}\left(b_0 - b_1\frac{T}{2} + b_2\frac{T^2}{4}\right) & a_{2,D} & = & \frac{1}{\Delta}\left(1 - a_1\frac{T}{2} + a_2\frac{T^2}{4}\right)
\end{array} \tag{29}$$

For the simple spring-mass-damper system of (22), we end up with

$$\begin{array}{rclcrcl} \Delta & = & 1+\frac{b}{m}\frac{T}{2}+\frac{k}{m}\frac{T^2}{4} & & & & \\ b_{0,D} & = & \frac{1}{\Delta}\left(\frac{1}{m}\frac{T^2}{4}\right) & & a_{0,D} & = & 1 \\ b_{1,D} & = & \frac{2}{\Delta}\left(\frac{2}{m}\frac{T^2}{4}\right) & & a_{1,D} & = & \frac{2}{\Delta}\left(\frac{k}{m}\frac{T^2}{4}-1\right) \\ b_{2,D} & = & \frac{1}{\Delta}\left(\frac{1}{m}\frac{T^2}{4}\right) & & a_{2,D} & = & \frac{1}{\Delta}\left(1-\frac{b}{m}\frac{T}{2}+\frac{k}{m}\frac{T^2}{4}\right) \end{array} \tag{30}$$

The point of the discussion is, if it looks complicated, it's supposed to be. The physical parameters get lost in the shuffle a lot and we need to fight to keep them in terms that are meaningful in our discrete time model. The b, k, and m parameters are spread all over Equation 30. This also spells bad news for trying to extract physical parameters from time domain ID with discrete time models. The accuracy to which we need to identify the coefficients in Equation 27 in order to back out the physical coefficients using Equation 29 is tremendous. The problems get worse when the system order gets higher.

What I've come to realize is that breaking the problem down into blocks and discretizing the blocks makes a lot of sense in the sense that each block has it's own discretization error, but it also preserves the physical meaning of the original block (if you do it right, which still isn't trivial).

3.2 Delay

The discussion of sampling brings into focus a discussion of time delay, and an understanding of time delay should be part of any control system implementation. After all, time delay manifests itself as a pure negative phase in the frequency domain. That is for a delay, T_D, the Fourier Transform pair [12] is:

$$f(t-T_D) \supset F(s)e^{-j2\pi f T_D}, \tag{31}$$

which means once again that even if everything else is done perfectly, the time delay will eventually drive the open-loop phase below -180° at some frequency, $f = f_{lim}$.

While we think about time delay from sampling, where typically one assumes an average delay of:

$$T_D \approx \frac{T_S}{2}, \tag{32}$$

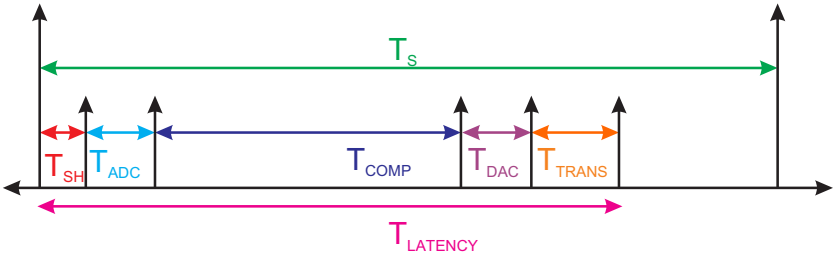


Figure 16: A diagram of delays in computing a control signal for a sampled data system.

and $T_S = 1/f_S$ is the sample period, there are many other components of delay in the system. One can consider the delay from the time that a signal is sampled until a control signal responding to that signal is produced, as diagrammed in Figure 16. Even if the sampling delay was 0 there would be physical transport delays in a system, the time it takes for fluid to flow from an actuator value to the flow sensor, the time that it takes for the effects of a heating coil to be sensed at a thermocouple that is displaced from the heater, the time for ions to travel from the source of a mass spectrometer to the detector, or the time for packets to be transmitted across a network, just to name a few.

As humans, we like to simplify, and this often means that we focus on one dominant delay. For fast mechatronic systems such as atomic force microscopes (AFMs, [13]), the computational delay diagrammed in Figure 16. For chemical process control [5, 14], the physical delays swamp the computational delays allowing for a slower sample rate (longer T_S) and therefore making the relative effect of $T_{LATENCY}$ in the diagram.

3.3 Time Constants

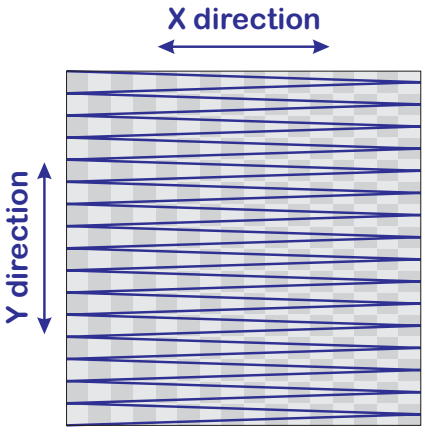
Both discussions of sampling and delay bring up the discussion of time constants. Generally, the term time constant is associated with the solution to a first order, linear differential equation,

$$\dot{t} = at = \frac{1}{\tau}t, \tag{33}$$

where τ is called the time constant and is the time for an initial condition to decay to e^{-1} of it intial condition value. In common usage though, it is generally considered the amount of time it takes for some response to really start rolling off. In this context, time constants can be thought of a how much time it takes for a signal to get through a system, or alternately, how quickly or slowly a system can respond to sudden changes in input. Time constants then

can tell us how relatively important our delay is. After all, if the constants of the system are super slow relative to the sampling delay, then the sampling delay can be ignored. (This is also known as "sampling fast", but really should be fast relative to the time constants in question.) Likewise, transport delay only matters relative to the time constants of the system in question.

One of the reasons that control engineers working in chemical process control can use techniques like Model Predictive Control [15] is that their system time constants are incredibly slow, when compared to high speed electronics or mechatronics. In fast systems, such as Atomic Force Microscopes (AFMs) [13] or Phase-Locked Loops (PLLs) [16] (used in communication, clocking, and synchronization), current computation technology is not fast enough to allow that.



Rate	Definition
sample rate	frequency of data sampling
scan rate	frequency of 1 line scan in each direction
frame rate	scan rate/lines per frame
pixel rate	scan rate · pixels per line
samples per pixel	sample rate/pixel rate
scan speed	scan line width (μm/line) · scan rate (lines/s)

Figure 17: A raster scan generates the initial movements in an AFM. The blue line represents the scan, which is generally faster in one direction – commonly denoted as the X axis. The checkerboard pattern underneath schematically shows pixels that might be formed from such a scan pattern. Each direction of scanning along the X axis is used to form a separate image. For each scan, the sampling of data along the Y axis will be considerably slower than the sampling of data along the X axis. Typically, some sort of decimation or averaging is done to reduce the X data samples to the pixel rate in the X direction.

We often don't consider what is obvious in retrospect: that a lot of our frequencies come from the movements of the system. Consider the movements of an AFM [17], generated by a raster scan to create an image. Figure 17 shows the X-Y scanning of a sample area that is typical in generating an image. On the right of the figure is a table that shows the different rates in the system. One of the areas of potential confusion is that any one of the rates listed can be what any engineer – or more frighteningly, any marketing person – calls

the "rate". If one makes certain assumptions about the control design, including the phase margin, the scan size, the number of pixels, and the width of a line, one can assess the desired closed-loop bandwidth and acceptable system latency. This then allows us to back out the needed sample rate and electronics switching speed.

We will find that it is useful to paraphrase Sun Tzu [18] here, "Know your time constants, and know your dynamics, and you can close 100 loops without disaster."

3.4 Nonlinearities

All of these can create issues for making a system work, but as they say in late night commercials, "Wait! There's more!" This is because every real system has nonlinearities, and these nonlinearities often make the quantization we see in the ADCs and DACs look like small potatoes. Often this describes slew rate limits on flow, or simply a valve being able to only open to 100% and close to 0% flow (rather than towards $+\infty$ or $-\infty$). The thing about nonlinearities is that they break our linear analysis tools and so the question is not about whether a system has nonlinearities or is nonlinear, but really about how nonlinear a system actually is. Does this account for 90% of the response or 0.9%? Is a system governed by different equations in one region of operation than in another? The ability to detect these conditions and change the controller's behavior is one of the great driving forces behind using computer control. Of course, we still have to build that.

The saturation nonlinearity is one of the most common, showing up all along the signal chain, from limits on valves to limits on voltages, to limits on the size of signals in fixed point computation. How much of a signal's nominal response exists in between the limits? What does the rest of the system do once those limits are hit.

- In double integrator problems, the effects of saturation do not affect the reachability of any state and therefore has little effect on closed-loop stability. However, it is the saturation that sets up the time optimal, bang-bang control solution.
- For some systems, the effective gain reduction due to saturation can lead to instability. One of the simplest systems that display this is the triple integrator where for small enough feedback gain, the root locus reveals closed-loop poles in the right half plane. Thus, for a large enough input signal, the gain reduction due to saturation results in

instability., The details depend upon the specific feedback compensator.

- One of the most nefarious and hardest to debug locations for saturation is in notch/anti-notch filters. A notch can be thought of working by creating a canceling signal that matches the input signal at a particular frequency of interest. If that input signal is saturated coming into the notch, the generated canceling signal will also be clipped, resulting in a notch that doesn't really work.
- In fixed point math, not only can signals be clipped, but also filter coefficients. Returning to our notch example, it turns out that for high Q filters, the limits on coefficients can take stable, minimum-phase filters and make them unstable and/or non-minimum phase [19].

Lest we forget ADCs and DACs provide quantization nonlinearities in each digital control system. While we are accustomed to our double precision floating point numbers, ADCs and DACs have fixed quantization with the quantization level inversely related to the sample period and the chip cost. The fastest ADCs are the 8-bit converters used in multi-Gigahertz scopes (cite some Keysight,Techtronix scopes). The highest accuracy in products in broad distribution in the market is around 24 bits. Typical quantization levels in control applications are typically between 12 – 18 bits.

3.5 Noise

Finally, there is noise, which is a term used to describe anything from shot noise to biases due to cables or friction. Noise can be easy to model (e.g additive white Gaussian noise) and not describe what we are observing or may be perfectly descriptive and hard to handle analytically. As with all of the above, it places a limit on what information we can cleanly extract from a system, either in our attempts to model the system or to properly control it.

Less common is the understanding of the effects of noise through the feedback loop, or how to back noise measurements out to their sources. In the legendary "Respect the Unstable" Bode Lecture of 1989 [20] Gunter Stein educated us to the idea that loops do not eliminate noise, they merely move it around, as he brilliantly illustrated with a dirt digging problem, reconstructed from memory on the right side of Figure 18.

If one uses that as a starting point and works backwards, one can establish what the "input

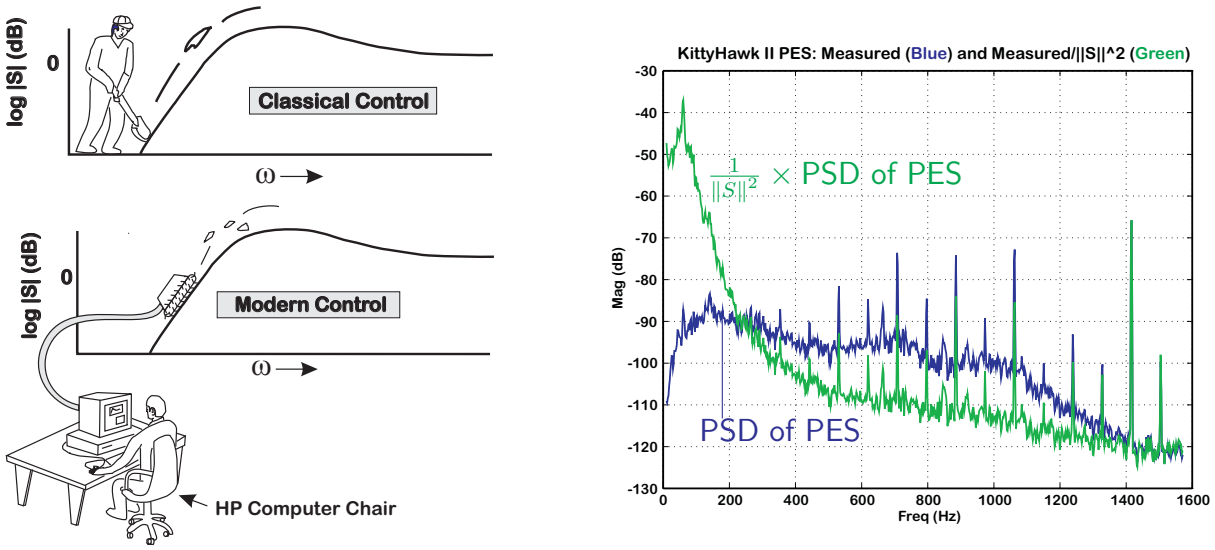


Figure 18: On the left, Gunter Stein's dirt digging analogy, recreated from memory circa 1994. On the right, KittyHawk 1.3" disk drive: PSD of PES, and PSD of PES filtered by $\frac{1}{\|S\|^2}$.

noise" must have looked like, as shown in Figure 18. This became the basis for the PES Pareto methodology of analyzing the effects of noise on a system [21, 22, 23, 24].

Even then, when we are trying to make practical measurements, we must default to some idealizations. The Widrow model [25] of quantization, it assumes quantization error as uniform white noise on the interval, $[-q/2, q/2]$. To analyze noise through a linear filter requires an assumption of additive, Gaussian, white noise. We step away from analytical purity and mathematical exactitude to be able to gain understanding.

3.6 Skill Sets

Beyond a technical risk, there is an organizational risk posed by the necessity to involve multiple designers with varied skill sets int he design. If one walks around a typical digital control loop of Figure 2, one sees that each block brings with it a particular set of skill

requirements.

Digital Computer: In order to do control, we will need to do digital control. Besides the obvious needs to understand some digital control theory, we also need to program the computer to implement the control law. This involves understanding of the physical system time constants and the power of computation we have available. Can we run in a multi-tasking operating system or do we need to be close to a pure processor or FPGA solution? In which language will we program? What levels of abstraction do we need? What do we gain and what do we give up when we let the advanced tool handle all these abstractions?

Physical System: The first step to doing any control is to have some understanding of the physical system. That requires domain expertise in the system to be controlled. It is a sad, but true case that physical systems do not come with their own third order analytical models. Perhaps we can fix that after the next Big Bang, but until then we are stuck trying to model physical systems from a combination of first principles (science), measurements, and simulation. Even measurements require some system knowledge to know what must be measured and what one can hope to understand.

Sensor: The sensors need to interface with the physical system to return an analog of the physical property being measured. As such, there is a lot of science tied to circuitry tied to machine design.

Actuators: On the other side of sensors are the actuators that tie electrical signals into physical action. Again, they involve a combination of machine design, science, and signal knowledge.

Analog Filters: Most conversion circuits require some sort of signal filtering to minimize noise entering the loop. Before the signal can go into an ADC or go from a DAC to the physical system, we want to clean up the edges, remove noise spikes, notch out harmonics, etc. These are done with analog circuits and the art of analog design is almost a rarity these days. What are the phase effects of different low-pass or anti-alias filters? Do power line harmonics need to be removed? All of these require some knowledge of analog design.

ADCs and DACs: The choice of signal converters is rarely simply about picking the number of bits, or we would simply pick the converters with the highest resolution. Instead, we must contend with delays, both in the interface between converter and computer (parallel or a variety of serial formats) as well as delays in the conversion method (SAR, pipelined, etc.).

Power Electronics: Before we can send a DAC signal to actuate a backhoe arm, we need some power electronics to scale things up. Before that engine temperature can be tracked, it must pass through a thermocouple. Signals in kilovolts need to be scaled down to that of the ADC, typically under 5V these days. For the electrical grid to get smart, 3-phase measurements at high voltage levels need to be made. All of these require a knowledge of power electronics, which are often not taught in many collegiate Electrical Engineering curricula anymore.

Control Design: At the end, we come back to control design, but control design in the context of these other factors. In the real world, optimality sucks [4]. That is, optimality, which can only exist in the context of an abstract, usually simplified, mathematical model, cannot exist in a real world system where all that stuff is happening. However, our idealized, simplified models can be used to tell us the best we can do, even in the real world. Guided by understanding how to apply our theory and optimization to real world systems, we can't get to optimal control, but we can get to excellent control. Or to be simplified to a bumper sticker, Optimality sucks . . . but excellence rocks[4].

4 Introduction Summary

To quote Winston Churchill [26], "Now this is not the end. It is not even the beginning of the end. But it is, perhaps, the end of the beginning." This introduction is intended to motivate the dives in the sections to come. The difference between an idealized system in analysis and a real working system involves addressing these issues, extracting a workable system model in the face of sampling, time delays, nonlinearities, and noise, then applying that model in a way that does not violate the limitations imposed by those issues.

In the sections that follow, we will discuss:

- System Models and Characterizing Them with Measurements: We will first delve into that fundamental need for applying advanced control methods to physical systems: how to extract usable models from available measurements. For these, and particularly when only step response methods are available, we will find that we need to assume something akin to one of the models described in Section 2. Using ideas from Section 3.1, we will discuss the difficulties and limitations of trying to use discrete-time, time domain identification on models for which we want to extract physical understanding.

- Simple Controllers for Simple Models (or why so many controllers are PIDs): A dual to extracting a simple model is understanding what can be done with simple control schemes. Thus, we will put PIDs into a unified framework [27] and then describe what we can tell about these controller for our canonical simple models of Section 2.

- These two segments will provide some intuition into what we can expect from simple models and controllers, and they will form a backdrop for the other aspects of practical systems. In particular, they can be seen as the ideal that we are trying to get back to when other system features interfere with our plans.

- Resonances, Anti-Resonances, Filtering, and Equalization: As Maarten Steinbuch writes on the board in his control systems class, mechatronic systems are "-2 plus stuff." (The word, "stuff" is a diplomatic substitution here, just as in the expression, "Stuff happens.") For flexible, mechatronic systems, that stuff consists of one or more high Q resonances and anti-resonances. If one wishes to not be limited by these, we must understand practical application of filters for equalization.

- Practical Loop Design, Or Why Most Open Loops Should Be a Constant or an Integrator, and How to Get There: We will discuss a relatively simple design objective, where it shows up without much discussion in many engineering problems, and what we have to do to apply it in practical problems.

- Signal Detection, Sensors, Sample Rates, and Noise (Oh My): When everything else is said and done, we can't do feedback control without sensing the data and how quickly and cleanly we can do this determines a lot of what any algorithm can do in a control system. (Sensor noise goes right through.) Thus, it's worth understanding sensors, sample rates, time delay, and noise propagation through the system.

- Integrating in Feedforward Control: As the Cheshire Cat pointed out to Alice [28], it's a lot easier to get where we want to go if we actually know where we want to go. In control, that is called feedforward. Sometimes this is not available to us, but how to we make use of it when it is?

- Ask Your Doctor: Is State Space Right for You? State space control is model based control and model based control requires a . . . model. If we've kept up to this point, we may very well believe that the greatest limiting factor in using state space control on physical systems is our ability (or not) to extract accurate models of the actual system and to understand the system based limitations of the accuracy of those models. We will try to develop a mental checklist of the things we need to get to so that we can do what the cool kids do.

- **Pick a Chip, Any Chip: Finally, while Silicon is cheap, finding the right Silicon takes some investment. We can only scratch the surface here, but this section will give the listener a start on understanding a three layer computation model for real-time systems and how to make use of this in the chips we may chose to use for our control systems.**

References

[1] H. Solo, "Musings on mystical energy fields," in Star Wars, Episode IV (G. Lucas, ed.), (A galaxy far away from here), Lucasfilm, 20th Century Fox, 1977.

[2] D. Y. Abramovitch, "Measurements for the design of control systems: Step and frequency response methods," in Presented at Applications Friday during the 2016 American Control Conference, (Boston, MA), July 2016.

[3] D. Y. Abramovitch and S. B. Andersson, "Understanding and tuning PID controllers," in Presented at Applications Friday during the 2016 American Control Conference, (Boston, MA), July 2016.

[4] 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), pp. 223–250, IEEE, IEEE, September 2015.

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

[6] MathWorks, "Temperature control in a heat exchanger," 2016. [Online; accessed June 21, 2016].

[7] S. Padhee, "Controller design for temperature control of heat exchanger system: Simulation studies," WSEAS Transactions on Systems and Control, vol. 9, pp. 485–491, 2014.

[8] D. Y. Abramovitch, "The continuous time biquad state space structure," in Proceedings of the 2015 American Control Conference, (Chicago, IL), pp. 4168–4173, AACC, IEEE, July 2015.

[9] D. Y. Abramovitch, "The discrete time biquad state space structure: Low latency with high numerical fidelity," in Proceedings of the 2015 American Control Conference, (Chicago, IL), pp. 2813–2818, AACC, IEEE, July 2015.

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

[11] D. Y. Abramovitch, "Adding rigid body modes and low-pass filters to the biquad state space and multinotch," in Proceedings of the 2018 American Control Conference, (Milwaukee, WI), AACC, IEEE, June 2018.

[12] R. N. Bracewell, The Fourier Transform and Its Applications. New York: McGraw-Hill, 2 ed., 1978.

[13] 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.

[14] D. E. Seborg, T. F. Edgar, D. A. Mellichamp, and F. J. Doyle, Process Dynamics and Control. Wiley, fourth ed., 2016.

[15] J. B. Rawlings, "Tutorial overview of model predictive control," IEEE Control Systems Magazine, vol. 20, pp. 38–52, June 2000.

[16] D. Y. Abramovitch, "Phase-locked loops: A control centric tutorial," in Proceedings of the 2002 American Control Conference, (Anchorage, AK), AACC, IEEE, May 2002.

[17] D. Y. Abramovitch, "A tale of three actuators: How mechanics, business models and position sensing affect different mechatronic servo problems," in Proceedings of the 2009 American Control Conference, (St. Louis, MO), pp. 3357–3371, AACC, IEEE, June 10-12 2009.

[18] S. Tzu, The Art of War: Translated and with an Introduction by Samuel B. Griffith. London, Oxford, New York: Oxford University Press, 1971. ISBN 0-19-501476-6.

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

[20] G. Stein, "Respect the unstable," IEEE Control Systems Magazine, vol. 23, pp. 12–25, August 2003.

[21] D. Abramovitch, T. Hurst, and D. Henze, "An overview of the PES Pareto Method for decomposing baseline noise sources in hard disk position error signals," IEEE Transactions on Magnetics, vol. 34, pp. 17–23, January 1998.

[22] D. Abramovitch, T. Hurst, and D. Henze, "The PES Pareto Method: Uncovering the strata of position error signals in disk drives," in Proceedings of the 1997 American Control Conference, (Albuquerque, NM), pp. 2888–2895, AACC, IEEE, June 1997.

[23] T. Hurst, D. Abramovitch, and D. Henze, "Measurements for the PES Pareto Method of identifying contributors to disk drive servo system errors," in Proceedings of the 1997 American Control Conference, (Albuquerque, NM), pp. 2896–2900, AACC, IEEE, June 1997.

[24] D. Abramovitch, T. Hurst, and D. Henze, "Decomposition of baseline noise sources in hard disk position error signals using the PES Pareto Method," in Proceedings of the 1997 American Control Conference, (Albuquerque, NM), pp. 2901–2905, AACC, IEEE, June 1997.

[25] B. Widrow, "A study of rough amplitude quantization by means of Nyquist sampling theory," IRE Transactions on Circuit Theory, vol. 3, pp. 266–276, 1956.

[26] Wikipedia, "Winston Churchill," 2018. [Online; accessed March 28, 2018].

[27] D. Y. Abramovitch, "A unified framework for analog and digital PID controllers," in Proceedings of the 2015 Multi-Conference on Systems and Control, (Sydney, Australia), pp. 1492–1497, IEEE, IEEE, September 2015.

[28] L. Carroll, Alice's Adventures in Wonderland. Macmillan, 1898.