Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 52-9 (2019) 85–90

Thoughts on Furthering the Control Education of Practicing Engineers ⋆

Daniel Y. Abramovitch *

* *Mass Spec Division, Agilent Technologies, 5301 Stevens Creek Blvd., M/S: 3U-WT, Santa Clara, CA 95051 USA (e-mail: danny@agilent.com).*

Abstract: This paper presents some insights on teaching practical improvements to control methods to engineers already practicing in the field. Virtually all of these engineers have taken an introductory controls class (perhaps many years ago) and have some experience with some circuitry, programming, and actual control implementations. We argue that the lessons that have the most impact on these engineers are those that tie theoretical insight to methods that they can adapt almost immediately to make their jobs easier. We discuss both the environment of a typical practicing engineer and some of the lessons that can immediately make them more effective.

Keywords: Continuing control education in industry, challenges for control engineering curricula, pedagogy in control engineering.

1. INTRODUCTION

Part of the proverbial "gap" between theory and practice in control is that most of the instructors in control are professors who are teaching the proper theory behind implementing control systems, and most of the students – if they end up practicing control – will end up dealing with a plethora of issues not encapsulated by a one semester class on control theory. The students that go out into industry may or may not encounter a feedback control problem in their first few years of work. Those that do encounter such problems will often find that they are not working on designing a controller from scratch, but instead working to adapt a controller that has previously been in use at their company to some new project. The controller most likely is some form of PID, although the specific parameterization is not known. Perhaps there is documentation, but it most likely does not tie well to anything they learned in their control class. It may be implemented in analog form with op amps, but is most likely inside a block of embedded software (ESW) with circuitry and converters to connect it to the actual device. Again, the documentation here may mostly lie in the brain of the original designer. Perhaps in conjunction with this project, perhaps on a different project, the engineer needs to add some filtering to a system. What help is there to allow them to bridge the gap from a classic DSP text (Oppenheim and Schafer (1970)) to a working piece of code? Even if they are using the Matlab Filter Design Toolbox, how do they get the system measurements into Matlab and the coefficients provided by the design into a functional and debuggable piece of embedded software (ESW)?

These are the types of problems that confront practicing engineers, and before we can teach them about advanced methods from current research, we have to tie the types of problems they encounter in their daily work to the larger body of control theory. We have to be willing to frame advanced methods in a context that does more than tell them how optimal something is: it must also tell them when it is actually useful.

We need to do this not so much for the sake of the practicing control engineer, but for our own. With agents, networks, robots, self driving cars, smart grids, swarms of drones, etc. the prevalence of human-built feedback systems in our lives is increasing exponentially. In parallel, the growing qualitative understanding of the role of feedback in biological and environmental systems – and the need to model and quantify these – cries out for a public understanding of the need for, uses of, and pitfalls in feedback systems. The prevalence of cheap real-time computing platforms and hobbyist accessible programmable cars, houses, robotics, and drones means that there will be a proliferation of computer controlled devices, with most of the code written by people with not even a basic understanding of feedback systems. If we wish to play a role in guiding these efforts, it is up to us to make control theory relevant to all these people "practicing control without a license."

This paper provides some insight based on the author's personal experience in industrial research, often involving control systems, but always involving getting something physical to work. At the same time, the author has kept close enough ties with the academic controls community to be able to draw from their methods in his work. The hope is to demonstrate a thought process that makes advanced methods more accessible and practical in real world environments to practicing engineers.

⋆ Daniel Y. Abramovitch is a system architect in the Mass Spec Division at Agilent Technologies.

10.1016/j.ifacol.2019.08.129

2. UNDERSTANDING THE AUDIENCE

The worlds of academic research and industry are obviously different, but the way in which they are different is often misperceived by audiences in both camps. While the academic controls world seems ever focused on finding elegant algorithms, the world of industry has as its research goal a product (or a family of products) that can be successfully commercialized. The days of Bell Labs and IBM Research in which an industrial researcher could have a good career producing only demonstrations and research papers are long gone. Even in the heyday of Hewlett-Packard Labs, there was a focus on getting to a product in 5–10 years. The time lines have only gotten shorter for what remains of these industrial research labs. Industry is about products.

What this means is that for members of the controls research community (who are mostly academics) to see their work turn into something that enters the real world, they must be able to communicate the methods, techniques, and results to an audience that spends much of their time dealing with things that do not show up in a linear, time-invariant (LTI) model.

With some broad generalizations, the typical practicing engineer involved in control work is someone who has a Bachelors or Masters Degree in Electrical, Mechanical, or Chemical Engineering. It used to be rare for someone with a degree in Computer Engineering, Software Engineering, or Computer Science to work on control systems, but the prevalence of inexpensive hardware modules, and a growth in the popularity of Data Science and Machine Learning, combined with the expansion of control and optimization methods beyond moving chunks of metal have made this more common.

In this article, we will not stray too far into the latter group. Instead we will mostly focus on the former, a group that has had at least one course in control theory, may have some experience with signal processing, may have had a circuits, dynamics, mass transport, or thermodynamics class in college, and has had to write some programs, both in college and in industry. Depending upon the size of their organization, this "practicing controls engineer" may spend most of their time working on feedback control problems (say in the hard disk drive (HDD) or aerospace industry) or they may delve into it every few months when the product needs some loops to be tuned.

With the exception of the aerospace industry – where sophisticated methods and large teams of control engineers are common – it is more likely that the practicing control engineer is working on their own, or with only a small number of colleagues with which they can discuss their work. They are judged on their ability to contribute to a profitable product, and so they are highly unlikely to write papers on their work. More the point, they are also unlikely to attend many controls conferences or read many theoretical papers. Their needs, therefore, are not for something that is 2 % better than the previous year's result, but for something that allows them to be more effective in the tasks before them. No matter how elegant a solution one has, if the practicing engineer cannot see a way to use it, they will revert to what they known (until that fails).

3. A TYPICAL INDUSTRIAL CONTROL ENVIRONMENT

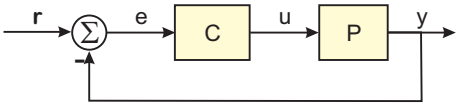

Fig. 1. The classic SISO feedback control block diagram often used for analysis.

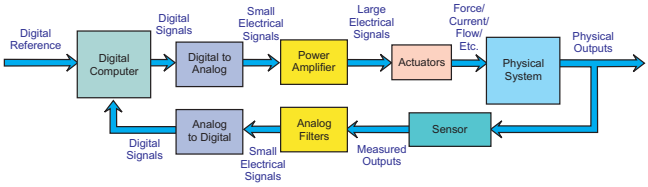

Fig. 2. A more complete SISO feedback control block diagram which shows many of the pieces critical to implementation.

One way to understand the fundamental differences between a practical view of control systems and a theoretical view is to consider the block diagrams of Figures 1 and 2. In Figure 1, we have the purest of feedback loops where all the messiness of actually building a control system is swept into the model block of P. We do not even consider sampling here even though it is most likely that any advanced control method must be implemented digitally (except for certain rare super high speed applications). On the other hand, Figure 2, while still being SISO includes a lot of blocks, each with their own design and implementation characteristics. The C block from the former picture now must take into account the conversion and computation environment. The P block must take into account sensors, actuators, and all sorts of conversion electronics. It is far cleaner to come up with advanced versions of C in the former than it is to realize that in the real world, C is the last thing one gets to work on. All the other blocks must be taken care of first, hopefully in a way that still allows one design freedom to do something useful in the controller block. This latter world is the one that the practicing engineer inhabits. To make advanced methods relevant to them, we must first show that we have an understanding of their entire workspace.

Again, we must generalize to teach anything on this subject, so we beg the reader's indulgence. There will be many variations, but a fairly typical control design environment in which a typical practicing engineer finds themselves has:

- A PID (Proportional-Integral-Derivative) controller. With exceptions in the aerospace industry, the hard-disk drive (HDD) industry, and other advanced mechatronic problems, most industrial controllers are PIDs. They are the bane of the second year graduate student whose thesis often compares their model based method to the lowly PID, but if that student goes into industry, the odds favor them working with these devices. We will spend some time on that in

Section 4, since to not explain this and provide useful PID guidance immediate disconnects the controls researcher from their audience.

- The practicing engineer often starts with an existing control system in their environment. They may choose to start from scratch, but this is a much larger undertaking that has to be weighed against the costs and time to market. This extra cost/risk must be justified and so more often than not, the decision is made to tweak an existing controller.
- Even though the controller is a PID controller, it is not "standard." That is, the relationship of the coefficients to any sort of standard form (Abramovitch (2015c)). Instead, they are in whatever favorite form the original engineer used, most likely with little written documentation, no control equations written down, no discussion of the discretization, and few comments in the actual code.
- Someone got it to work, but it's entirely possible that this person has moved on, either to another job in the company, another company, or – more often than one might expect – the netherworld.
- This last point often means that folks in their company have been tweaking the system for years.

Practicing control engineers generally do not care about being on the bleeding or cutting edge of control performance. Instead they want a way to understand what they have been working on for years in the context of some extension of the control theory they learned years ago. They want to learn about pitfalls and gotchas they may run into (or have already): what causes them, how to spot and avoid them, and how to fix them when they happen.

At the same time, in practical control systems, the extra blocks of Figure 2, the amplifiers, data converters, sampling, computation available, noise, time delay, nonlinearities, etc. all potentially interfere with the ability of an an advanced control method to work. The expert in advanced methods might wonder why they should be discussing these side topics. The answer is that if they are not handled, then their methods are likely to not work – or at the very least not provide the promised advantages. When that happens, their methods seem impractical, not applicable, and in other ways not useful. This also costs the researcher credibility and when this becomes the norm of interaction between experts in advanced control techniques and practicing control engineers, the proverbial theory-practice gap gets only wider.

Section 4 will discuss ways to help practicing engineers improve on their existing PID environment. The sections that follow will discuss how many of these other issues disrupt the application of high performance control. This gives the researcher an opportunity to put these issues in context and offer methods for minimizing their negative impact. In the end, most SISO feedback controllers have an equivalent form that is a concatenation of a PID controller and equalizing filters. In either case, the performance of the controller is determined in large part by how closely the underlying model describes the physical system.

4. HELP FOR THE PID

As PID controllers dominate the industrial landscape and as it is very hard to get buy-in from practicing engineers to scrap everything they have that is working for something new without first having earned some credibility that one understands their environment, a sensible starting point for interacting with practicing engineers is to help them improve their PID controllers.

While PID controllers are the most standard controller used in practice, the PIDs themselves suffer from a lack of standardization. Virtually any control structure that has a proportional gain, an integrator with its own gain, and a differentiator with its own gain can be considered a PID controller, but the specification of gains can have a multitude of styles. Consider the following three PID controller equations. The center term for each is the typical "three parameter" form, while the rightmost term puts the PID in the form of an analog filter. For brevity, we have omitted the cases when a low pass filter is added to the differentiator term, although this would be necessary if the PID were to be discretized with a Trapezoidal Rule Equivalent (Abramovitch (2015c)). Still, these three forms are close to the style of specification of most continuous time, unfiltered PID controllers that may be encountered, either in analysis or in commercial controllers. Recognizing one of these forms, we should be able to easily translate the control parameters into one of the other forms. This is extremely useful in getting an apples to apples comparison of different commercial PID controllers.

$$C(s) = K\left(1 + \frac{1}{T_I s} + T_D s\right) = K\left(\frac{1 + T_I s + T_I T_D s^2}{T_I s}\right) \quad (1)$$

$$C(s) = K_P + \frac{K_I}{s} + K_D s = \frac{K_P s + K_I + K_D s^2}{s} \quad (2)$$

$$\begin{aligned} C(s) &= K_P + \frac{K_I}{T_I s} + K_D T_D s \\ &= \frac{K_P T_I s + K_I + K_D T_I T_D s^2}{T_I s} \end{aligned} \quad (3)$$

The form of Equation 1 is typically – but not always – associated with process control applications, temperature, and pressure control. There is an overall controller gain, but the relative gains of the three parts are adjusted via the terms T_I and T_D, which nominally are meant to refer to the integration time (time over which the integral takes place) and differentiation time (time over which the derivative takes place), respectively. Even here, the terminology is confusing, since as written the integral and differentiator take place over all time. These terms take the place of the integrator and differentiator gains. Still, they are a bit confusing since one would naturally assume that integration and differentiation times would be characteristics of the device or a measurement parameter, rather than a control gain.

The form of Equation 2 has three independent PID "knobs" with no hint of relation to the integration and differentiation time. The form of Equation 3 breaks the integration and differentiation times out from the integrator and differentiator gains. While this last form may seem a bit tedious due to the extra coefficients, it has a distinct

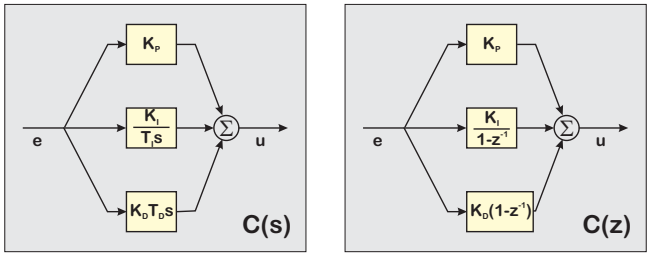

Fig. 3. A parallel form topology of a simple analog PID controller (top). The digital PID controller (bottom) shows much of the same structure.

advantage when discretized using a backwards rectangular rule equivalent where $s \longrightarrow \frac{z-1}{T_S z}$. If one sets $T_S = T_I = T_D$, which aligns with integration and differentiation over a single sample period, we get:

$$C_B(z) = K_P + \frac{K_I}{1-z^{-1}} + K_D(1-z^{-1}) = \frac{(K_P+K_I+K_D)z^2 - (2K_D+K_P)z + K_D}{(z-1)z} \quad (4)$$

What is most revealing about this form is the similarity between the structures of Equations 3 and 4 as demonstrated in Figure 3. The continuous and discrete parameters are related via $T_I = T_D = T_S$, while the structure remains essentially the same. Furthermore, the backwards rectangular rule discretization has taken a non-proper analog PID form and made it proper and therefore directly implementable.

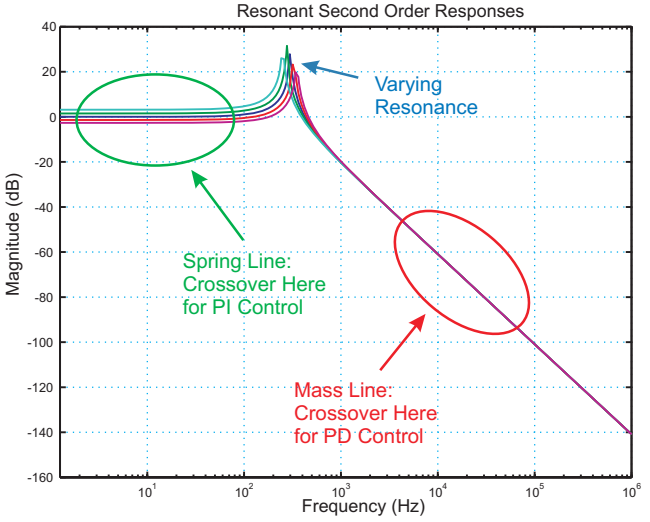

Fig. 4. The typical ranges in a mechatronic system and how they relate to PID control.

While the denominator of Equation 4 does not change with PID parameters, the numerator can produce anything from a lag filter (set $K_D = 0$) to a lead filter (set $K_I = 0$) to a notch filter (Abramovitch et al. (2008); Abramovitch (2015c)). Figure 4 shows the ranges of a typical mechatronic control system. If the loop is closed well below the resonance, a PI controller may be used. If the loop is closed far above the resonance, a PD controller may be employed. It is when we try to close the loop in the vicinity of the resonance that we need to carefully use all the PID coefficients, and so we need a far more accurate model than the previous two cases.

Showing these types of insights to practicing engineers moves the PID from something separate from the rest of control design to simply being a particularly useful substructure of control design. It ties the work on their current control methods with a path to more advanced methods, even if that starts simply with improved modeling of their system so that they can obtain better PID parameters. Finally, in a longer version of this paper, we would include a discussion of integrator anti-windup, which should be present in any practical PID controller.

Another bridge between PID controllers an advanced methods is to realize that almost any linear SISO state space controller can be formulated as a PID plus a group of matched filters. Looking at it this way, the matched filters correspond to the estimator model and neither of them works well if the modeling is inaccurate. Generally, for systems beyond a double integrator, successful modeling involves a deep dive into measurements.

5. ACTUATORS, SENSORS, AND CIRCUITS . . . OH MY!

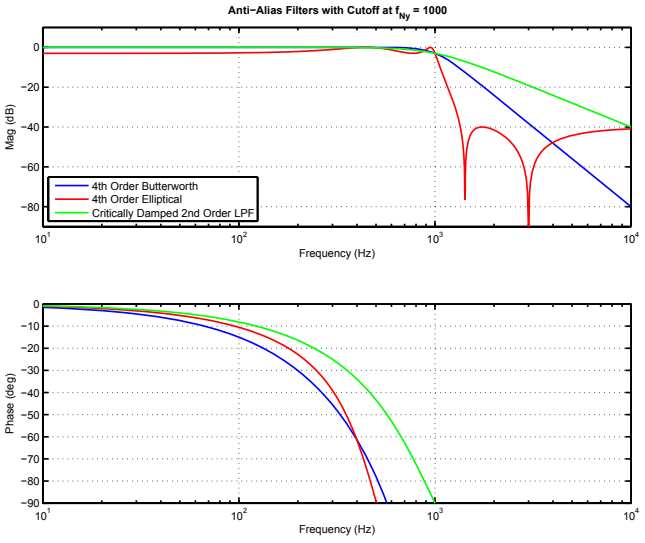

Fig. 5. Frequency responses of various anti-alias filters. All filters have a DC gain of 1, with the passband ending at the Nyquist Frequency (f_{Ny} = 1kHz here). Under an assumption that the Nyquist frequency is 5 or 10× the open loop crossover frequency, we can examine the filter phase response, which can significantly degrade the phase margin of the system, as documented in Table 1.

Filter	Phase at $f_{Ny}/10$	Phase at $f_{Ny}/5$
4^{th} Order Butterworth	−15.0276°	−30.1223°
4^{th} Order Elliptical	−10.5523°	−22.8086°
2^{nd} Order Butterworth	−8.1486°	−16.4211°

Table 1. Phase penalty of representative anti-alias filters.

As we have discussed, before a practicing engineer can design a better C block, they must first be aware of the designs and limitations of the other blocks in Figure 2. While this may seem overwhelming at first, it is important to note that the typical practicing engineer is not expected

to be the team expert on these other blocks. By the same token, they cannot remain ignorant of them. Their success in improving their control designs involve having enough of an understanding of the other blocks to realize when and where those blocks are limiting their abilities. They simply need to know enough of the language of the experts to be able to communicate the design requirements to them.

A perfect example is in the use of anti-alias filters. Virtually every book on digital control will mention that designers should use an anti-alias filter to limit signals at frequencies above the Nyquist frequency into the feedback loop. That is, they recommend an analog filter applied to signals before the ADC that cuts off signals above the Nyquist rate.

What is left out of these discussions is any talk of the phase effects of anti-alias filters, as demonstrated in Figure 5 and Table 1. One might desire maximum flatness in the pass band (frequencies below Nyquist) and a sharp drop after Nyquist, but filters with such sharp shapes are either not causal or have substantial effects on the system phase even a decade below the Nyquist frequency. The 4^{th} Order Butterworth filter does not achieve 40 dB of attenuation until the frequency is at $3\times$ the Nyquist Frequency. If all critical frequencies of the feedback loop are below $f_{Ny}/10$ then the phase penalty is only about 15 degrees, but in systems that can only sample at $10\times$ the critical frequencies, this phase penalty doubles to 30 degrees. A 2^{nd} Order Butterworth halves this phase effect, but does not get to 40 dB of attenuation until a full decade above the Nyquist Frequency. One can apply a 4^{th} Order Elliptical filter to get better stop band attenuation and less phase effects, but this comes at the cost of inaccuracies in the magnitude of the pass band. In this case, the design limit was 3 dB, but that is still a gain variation of $\sqrt{2} \approx 1.4142$. Many control systems cannot tolerate a gain uncertainty of 40% in the passband.

At this point, we are affecting the basic performance of the system. If one chooses an anti-alias filter with higher bandwidth to combat this problem, signals above the Nyquist rate are allowed into the feedback loop. A similar thing happens if one chooses a less aggressive anti-alias filter. Moves made to decrease the phase effects result in more aliased signals. Eliminating more aliased signals severely affects the phase margin.

Similar issues can be found with converter blocks, such as ADCs and DACs. While most control engineers have some understanding of quantization modeling as a zero mean, uniform density, white noise (Widrow (1956)), few discuss the differences between ADC converter types. $\Delta-\Sigma$ ADCs are quite popular in low cost audio applications, but they do not sample at a precise moment, and so are less suitable for feedback systems. Successive Approximation Register (SAR) ADCs tend not to have the same speed as $\Delta-\Sigma$ ADCs, but their phase response is better. Even then, different ADCs can have anywhere from one to tens of samples in pipeline delays through the device. The delay has no affect on signal processing applications, but has the effect of wiping out phase margin. Many of these ADCs and DACs, created for a signal processing market, have serial interfaces to the digital processing systems. Again, this may be perfectly fine so long as the sampling is 20X the desired close-loop bandwidth or open-loop crossover, but when this oversampling guarantee is violated we are left with a severe latency issue.

These are the types of real world implementation issues that limit the performance of any control system. They are particularly glaring when they swamp the 10% improvement due to adding a lot of mathematical machinery. This is yet another area that instructors need to raise the consciousness of practicing engineers so that they understand that mitigating them is another step towards enabling higher performance control and advanced methods.

6. CAN I TALK TO YOU ABOUT . . . FILTERS?

Most books focus on theory rather than how to write code or create circuits. Practicing engineer left with gotchas and other surprises that hit them when they try to implement these designs. When filters are in analog circuitry, they are usually modeled as IIR filters. The relevant questions are usually ones of how to write code or implement a circuit that implements the desired math in a way that can be verified. In the case of analog circuitry, component mismatches may lead to significant errors. Even matched components can drift over time.

In the case of digital filters there are a wide variety of issues. The filter design is likely specified and parameterized for a particular sample rate. The actual implementation is susceptible to timing jitter for a variety of issues, including clock jitter and multi-tasking operating systems. Even when the filter routine is run at a consistent clock, the structure of the filter program itself determines the time delay to an output. Unless the calculation is being done in a parallel architecture, such as a Field Programmable Gate Array (FPGA) and programmed properly, the delay in a filter's output is proportional to the length of the filter. When this delay is small compared to the sample period and system time constants, it can be ignored. When the computational delay is a significant portion of the sample period, then a delay that varies with the filter length is another source of controller uncertainty. A partial solution to the variable latency delay is to precompute most of the filter at the previous time step and then only do the last bit of computation that depends on the last input when the filter routine runs (Franklin et al. (1998); Abramovitch (2015a)). Here is the main point of this section: there is very little discussion in control design texts about how to actually write this code, or to create the FPGA designs. There are a few exceptions, such as those in (Auslander et al. (2002); Embree (1995)), but they are far from mainstream.

Another common pitfall is that of the Finite Impulse Response (FIR) filter, which is enormously popular in signal processing due to its simplicity, its inherent stability, the lack of buildup of quantization errors, and the possibility of having a linear phase response (for certain FIRs). FIR filters are not usually well suited for feedback control due to their inherent long length for a given frequency domain shape compared to Infinite Impulse Response (IIR) filters. However, the practicing engineer, arriving on the job, will often find systems with FIR filters in the code of the feedback loop. Even blocks as innocuous as an input averaging block are generating an FIR filter where the

average phase delay is half the number of the FIR delays. Advanced control researchers must be able to clear such topics before any of our methods can be applied in a real world system.

7. WHAT CAN DIGITAL DO FOR YOU?

One of the ideas that gets little discussion in a lot of controls texts are all the things beyond the implementation of a control law that one gets out of digital control methods. Digital methods give us more than a chance to do control implementation in embedded software; they allow us to do things that don't lend themselves easily to circuits and continuous time analysis. Sanity checking results, switching modes depending upon the inputs from auxiliary sensors, updating parameters, applying repetitive control methods or Iterative Learning Control, are all things that really can only be done in the digital world. Thus, the bureaucracy of data handling needed to do these things needs to be accepted as part of enabling the vastly superior performance – not due to an improved implementation of a differential equation, but due to the ability to switch between multiple methods depending upon what the measurements are saying. This gets far too little discussion in standard controls texts.

8. DEMYSTIFYING STATE SPACE

One of the largest differences between the academic view of control and that in practice is the prevalence or lack thereof of state space methods. State space methods dominate academic algorithms while they are far less common in industry. One of the main keys to understanding this dichotomy is that state space methods are model based methods, and model based methods require ... a good model (Abramovitch (2015b)). A good model of a physical system requires extensive and repeated measurements to expose the structure and determine parameters. One of the ignored prerequisites for such a process is tight coupling between the measurement systems and the control system design software. This is a tedious step in the work, and neither results in many new papers nor in new products. Yet, this often ignored connection allows engineers to do what they do best: iterate to find a better answer. Without these tight connections, the iteration loop of making measurements, transferring the results to design software, generating a new design, testing that design, and generating new measurements becomes so painful that the typical engineer will stop after the first functional set of parameters. The tedium makes it painful to get an accurate model, and without an accurate model any sort of performance optimization – whether by loop shaping or state-space design – is limited to the point of being futile.

In the author's experience, this explains the great divide: obtaining representative, accurate, workable models of physical systems is a difficult task often not addressed in a practical way in the research and teaching literature. While system identification needs to be done in discrete time, the most popular time-domain, discrete time identification methods fall short, especially in systems with a lot of lightly damped dynamics. Unable to generate an accurate model, practicing engineers quickly find that their state space estimators are only accurate for the lowest frequency and/or most highly damped states. They are either stuck with a lot of machinery that performs no better than classical methods or they give up and use classical methods.

One way to relate this to practicing engineers often involves the concept of matched filters, which are usually designed in the frequency domain. The quality of the matched filter, and it's ability to shape the response is directly related to how well the filter "matches" the system to be compensated. Again, this depends upon the quality of the modeling. The key difference is that matched filter measurements and modeling are often done in the frequency domain (except for highly damped systems). The point of emphasis is that these are what Gene Franklin pointed out as simply different ways of looking at the same problem.

9. CONCLUSIONS

This paper has attempted to cover some of the topics the author feels are important for interactions between researchers in advanced control methods and practicing control engineers. While certainly not exhaustive, it is hoped that these ideas give insight into what limits practical control systems, how those limits might be addressed, and the importance of discussing them in the context of trying to impart knowledge of any advanced methods to practicing engineers.

REFERENCES

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

Abramovitch, D.Y. (2015b). 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*, 223–250. IEEE, IEEE, Sydney, Australia.

Abramovitch, D.Y. (2015c). A unified framework for analog and digital PID controllers. In *Proceedings of the 2015 Multi-Conference on Systems and Control*, 1492–1497. IEEE, IEEE, Sydney, Australia.

Abramovitch, D.Y., Hoen, S., and Workman, R. (2008). Semi-automatic tuning of PID gains for atomic force microscopes. In *Proceedings of the 2008 American Control Conference*. AACC, IEEE, Seattle, WA.

Auslander, D.M., Ridgely, J., and Ringgenberg, J. (2002). *Control Software for Mechanical Systems: Object-Oriented Design in a Real-Time World*. Prentice Hall.

Embree, P.M. (1995). *C Algorithms for Real-Time DSP*. Prentice Hall PTR, Upper Saddle River, NJ 07458.

Franklin, G.F., Powell, J.D., and Workman, M.L. (1998). *Digital Control of Dynamic Systems*. Addison Wesley Longman, Menlo Park, California, third edition.

Oppenheim, A.V. and Schafer, R.W. (1970). *Digital Signal Processing*. Prentice Hall, Englewood Cliffs, N. J.

Widrow, B. (1956). A study of rough amplitude quantization by means of Nyquist sampling theory. *IRE Transactions on Circuit Theory*, 3, 266–276.