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

Trying to Keep it Real: 25 Years of Trying to Get the Stuff I Learned in Grad School to Work on Mechatronic Systems

Daniel Y. Abramovitch*

Abstract— **This paper is about the difficulty of making well known and widely accepted advanced textbook control techniques work in an industrial environment, particularly with mechatronic systems that have large numbers of flexible modes. I will go through the methods that fail if done the standard way and the adjustments I have learned to make over the years which get a lot of them to work. I will also go over the methods that seem to work robustly and without much thought in the industrial environment, explaining why they do work. Finally, I will try to show that understanding the differences and commonalities in these two world views allows us to use the principles of one to improve the other.**

I. Introduction and Overview

This paper will present a personal and technical odyssey that I made from graduate school to today, with the underlying theme of trying to get advanced control to work on mechatronic systems. The journey is far from finished, I hope, but I am far enough along to share some insight that might help others. Although this has been a personal journey and I will use a lot of first person to relay a sense of time in the presentation, the discussion is based on techniques and methodologies, as well as those "simple tricks and nonsense" [1] that seem to work when applied with the mindset of advanced control.

A mechatronic system is defined as one in which mechanics and electronics are tightly coupled. Generally, the mechatronic systems that present difficulties are ones that have resonances and anti-resonances with low damping factors (i.e., high Q). I will simply call these *high-Q* systems. This paper will consider small-scale mechatronic systems that need to be manufactured in large numbers, such as disk drives, optical disks, scanning probe microscopes, inexpensive robotics, scientific and electronic instruments, consumer products, and the like. This scale of system cannot be individually tuned in the way that a group of engineers might tune the control system for a multi-million dollar fighter aircraft, nuclear reactor, or cargo ship. The controllers either have to calibrate themselves or be robust enough to operate without repeated calibration.

There will be some basic themes/thoughts/ideas that emerge. In graduate school we moved quickly from classical control to state space, where the real theory was being done. System identification was done in the time domain, which usually meant doing identification on discrete-time models. We will see how that becomes exceedingly difficult on systems with high Q resonances.

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

This leads to the second theme, which is that state space is model-based control, and in order to do model-based control, *one must have a good model*. This seems absurdly obvious, but it turns out that the difficulties in getting a good model for high-Q systems severely limit the applicability of state space methods. Time-domain system identification has difficulties, and most frequency-domain methods focus on using measurements made with Fast Fourier Transforms (FFTs). However, FFT based frequency-domain measurements – having been extracted from time-domain data – manifest a dual problem to that of the time-domain measurements when trying to resolve the sharp frequency-domain features of high Q dynamics. (In this discussion FFT will refer to methods that use broadband input to generate a block of sampled time-domain data which is subsequently transformed using an FFT or related – e.g., periodogram, Power Spectral Density (PSD), etc. – method.) Even the most accurate frequency-domain measurement method, stepped-sine or sine-dwell (known in industry as swept-sine [2]) only produces a frequency response function (FRF) which must be fit to poles, zeros, and gain to get to a parametric model and this turns out to be a tremendous challenge. Progress on this series of problems has required a lot of interdependent pieces of technology.

Before those problems could be solved, I had to come to terms with the third theme, that is that many, many practical control problems are solved quite simply with very little deep consideration of unmodeled dynamics. From the ubiquitous use of Proportional-Integral-Derivative (PID) controllers in industry, to the circuit designers of phase-locked loops (PLLs), to the proponents and users of fuzzy logic and fuzzy control, to the explosion of simple "control" applications on the Raspberry Pi and its cousins, there are a lot of control systems that work without a lot of analysis.

The fourth theme is that if you have done everything else right, then you get to do the advanced methods and see them work. There are times that the physical problem practically begs for a multirate or an adaptive solution, but an adaptive-multirate version of a control design that produces poor results at its tuned state is still a lousy controller.

The fifth theme was best stated by Olivia Newton-John, when she told the world, "Let's get physical." Real control systems are far easier to understand and debug when the signals and states are close to the physical model of the system. In order to preserve the physicality of our models through discretization, we need to understand that many of the seemingly simple industrial control applications using a PID or lead-lag and some filters preserve essential physical

978-1-4799-7786-4/15/$31.00 ©2015 IEEE

intuition. To apply our advanced methods, we must learn to mimic this "preservation of physicality."

The sixth theme, which couples to the fifth, is that we should not remove ourselves too far from the implementation details. ADCs, DACs, choices of processor, analog circuits, actuators and sensors, floating point versus fixed point calculations, all matter. The control designer who gets too far away from these puts large chunks of their design in the hands of someone who likely lacks a system theory perspective. In other words, it is useful to know how to implement controllers beyond the Matlab or Simulink simulation.

The seventh theme is more of an axiom, and that is that advanced methods should gracefully degrade to simple methods if the problem is made trivial enough. In fact, I would argue that this is a fundamental sanity check for any advanced methodology: if one simplifies the problem enough, does one see one of the basic methods that works so robustly? If not, the advanced solution is almost certainly wrong, or at least not optimizing the correct model. *If it is true in general, then it should work in a simple example.*

The eighth theme/principle/observation is that people like to have knobs to turn, as long as there are not too many of them. As much as anything else, this can explain the continued popularity of PID controllers, fuzzy logic, and Gaussian filters. All of these present the user with a relatively small set of digital "knobs" to turn, which give the impression (real or imagined) that the user is using some sort of know-how to optimize the behavior of the system in a way that they can see on an oscilloscope.

The ninth theme/principle/observation is that when measurements are tedious to make and record, people avoid them, or try to find some shortcut way of doing them. In either case, this results in models that are far too distant from physical reality. I will argue then, that doing the grunt work of connecting measurements to models *in a way that makes it trivial to iterate*, is one of the key missing ingredients in mechatronics research. The dual of this is the connection of the design tool to the real-time controller implementation. It should take only a few keystrokes, mouse clicks, or screen touches to implement a new controller parameterization in highly efficient, low latency real-time firmware that preserves most of the hardware platform's raw speed.

Perhaps one more theme can be expressed in the mantra I have been using for a few years now,

Optimality sucks.[1].

Let me explain: Optimality, as we think of it, is based on a cost criterion applied to a limited model of the system. This limited model is *by definition always flawed* when describing the real world. The net effect of pushing for optimality then is often to cause the design to fail when applied to anything real. On the other hand, knowing optimal conditions for a model, and knowing how closely that model describes reality allows us to get excellent control. This is why on real physical systems, optimality conditions are, as Captain Barbossa would say, "more of what you'd call 'guidelines' than actual rules."[3] Thus, the above mantra, Version 2.0, would read

Optimality sucks – but excellence rocks[2].

Still, as Bill and Ted might say [4], "Being excellent is not bad."

II. What Different Perspectives Want

I am going to jump ahead here and offer a view of what two different control perspectives are looking for and then I will try to back that up in the sections that follow. This will be somewhat akin to a two-pass compiler, first reserving variable references, and then coming back to tie them into reality. At least, that is the intent. The reader should understand that in order to present these two perspectives I will make very broad generalizations that won't apply to any one individual exactly. Call it fuzzy reasoning. The two perspectives I will consider here are textbook and practical mechatronic control.

Broadly stated, textbook mechatronic control wants to:

- Take a system model that describes something in the real world.
- Find properties of that model that allow for a new and improved method of control.
- Analyze the new method on the model and show it is "optimal" on some metric.
- Simulate the model in Matlab/Simulink (or something equivalent) and make it "real" by adding in Additive White Gaussian Noise (AWGN) and/or some sort of bounded uncertainty.
- Say it will work in practice.

Broadly stated, practical mechatronic control wants to:

- Hook some real-time controller box into a test system or prototype.
- Push a button that generates *excellent* measurements of the system and produce model options.
- Push another button that generates an accurate, robust, parametric model for controller design, and gives the user a design choices menu.
- Push the design button that produces a high performance, robust control design and projects its behavior against the original measurement.
- Push one more button to transfer the design into a low overhead, high sample rate, real-time control system.

This is where I want them to meet:

- Every button push results in a step that is both physically intuitive and mathematically smart (although not necessarily "optimal").
- Models are heavily measurement driven and measurements can be rapidly iterated with data being passed easily to and from control systems Computer Aided Design (CAD) software. Parametric models for control design can be rapidly and reliably extracted from measurements.

[1] ©that.

[2] ©that, too.

- Implementation choices and trade-offs are reflected back into the system and design model.
- Measurement, model, and design improvements are easily iterated on the experimental system. Designs are easily transferred to the real-time experimental system. Experimental and measurement data are easily transferred back into the CAD program.
- Experiment and design results easily compared to "optimal" model-based projections to see how close implementation is to theoretical best.
- The ability to reflect "non-control" design choices into the model in an intuitive way for co-design with other fields.
- Experiments, models, and designs easily saved in a form that is easy to retrieve, easy to display, and easy to export to many other formats.

The unified approach does not see superiority in either theoretical or practical results, but works to iterate between the two to achieve a practical system that may not be *optimal* but is *excellent*. Much of the work to connect these two worlds involves a lot of grunt work in programming, but this programming cannot be done without a system view that stretches from the physics to the web page. Not only that, but the programming has to take the structure of the physical system.

III. Fresh Out of Grad School (FOOGS)

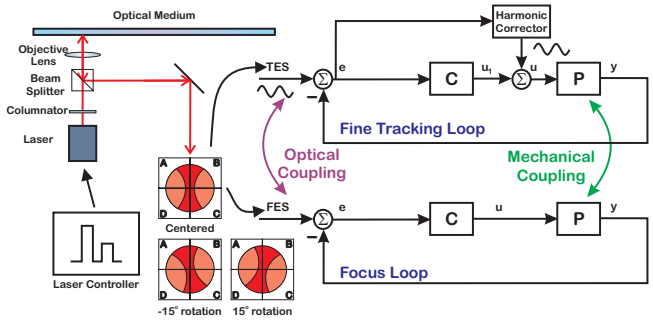

Fig. 1. Optical disks control problem. When actuator is rotary, tracking pattern rotates in the optical detector causing cross talk between channels. There is also mechanical cross coupling in the mechanism.

After graduate school, and after a brief stint at Ford Aerospace in which Dan Witmer introduced me to phase-locked loops [5], [6] (Section VI), I joined Hewlett-Packard Laboratories (HP Labs or HPL). The job at HPL was to work on an optical drive (Figure 1) as a replacement for magnetic disk drives (hard disks), which at the time were using the 5 1/4" format, spinning at 3600 RPM, and holding between 100 MB and 500 MB in a full height drive. The areal density of optical drives at that point was much higher, largely owing to the square aspect ratio of the bits, whereas the magnetic bits had a roughly 20 to 1 ratio of the cross track dimension to the down the track dimension. The issues involved in rewritability (achieved by using magneto-optic recording) and the shrinkage of the relatively large optical access mechanism, and one of the mechanical aspects of this was to move to a rotary actuator as disk drives had [7], [8]. This increased the cross coupling between focus and tracking loops [9], [10].

While my graduate school work was filled with trying to come up with some new theory or algorithm, at HPL I was faced with a situation of needing to do research on MIMO control systems, for which only a laboratory demonstration would do. However, when I started at HPL, the laboratory system for doing control experiments consisted of a system based on a TI TMS-320C25 fixed point DSP chip. The routines were written in assembly language and ran off of an old HP 9836 technical desktop computer. The real-time software on the DSP implemented a simple digital lead controller with some extra filtering. Changing parameters on the system required a reassembly of the assembly language code. The host computer, while hosting its own set of tools, made it impossible to connect the DSP system or any of the real-time data to a CAD program such as Matlab. Furthermore, even though there were large numbers of instruments in the lab, including digital oscilloscopes, spectrum analyzers, and dynamics analyzers, such as the HP 3562A Dynamic Signal Analyzer (DSA), none of these were tied into any computer systems. This meant that measurements were monolithic, not tied into any of the design programs, and most likely to be saved by printing them out on a pen plotter. Furthermore, the difficultly of taking a measurement into something like Matlab made it something that one would do rarely. To me, this seemed like a tool set with no chance of producing good results, largely because of the difficulty in sharing data made rapid, measurement driven design iteration next to impossible.

One positive feature of the system was that the DSA was capable of unwrapping closed-loop frequency response function measurements into open-loop quantities. That is, if one measured the frequency response function (FRF)

$$T = \frac{PC}{1 + PC}, \tag{1}$$

then with a push of a button it would compute

$$\frac{T}{1 - T} = PC. \tag{2}$$

This was very helpful, so long as your system was SISO and your measurement indeed was of T and not $-T$ and not $\frac{P}{1+PC}$ or $\frac{C}{1+PC}$. In these cases, the fixed structure and programming of the instrument became a real limitation.

At the time, there were several possible solutions. I could pay about \$50K–\$100K for an RealSim AC-100 System made by Integrated Systems, Incorporated during that era [11]. I could wait 2–4 years for dSpace to come out with their first floating point DSP based control tool. Instead, I chose the option of working for 16 months to create my own tool using a TMS-320C30 floating point DSP chip on a Banshee DSP board produced by Atlanta Signal Processing (ASPI) at the time. The system I created (Figure 2) [12] to be able to test Multi-Input, Multi-Output (MIMO) controllers for optical disks [10], [13] had the following characteristics:

- The DSP board resided in a standard MS-DOS PC, with an Intel-80386 processor and an 80387 IEEE floating

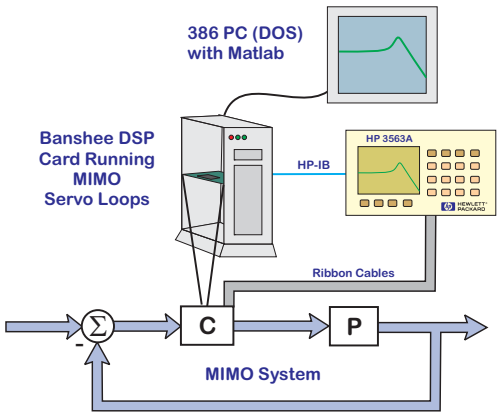

Fig. 2. The Banshee Multivariable Workstation did everything it was supposed to do and not enough.

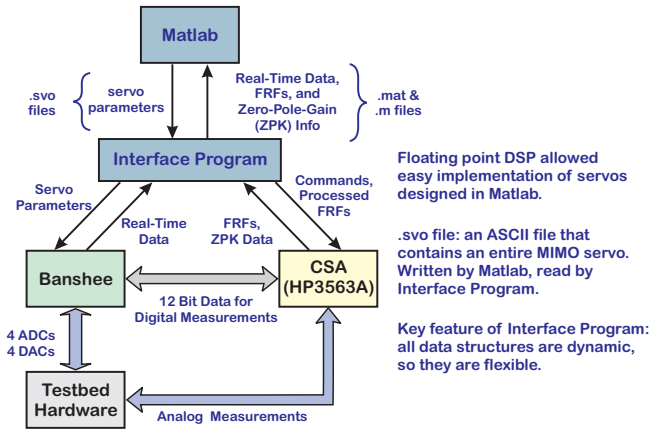

Fig. 3. The Banshee Multivariable Workstation: a functional block diagram.

point co-processor. It was the first full 32-bit computer in the lab, and I used the 32-bit registers to translate between the C30's floating point format and the IEEE floating point format on the PC. Using a standard PC allowed me to run PC-Matlab on the lab system and which meant that if I wrote the glue software (in Borland's Turbo-C) I could tie the real-time system into my Matlab CAD tools.

- The system read multivariable servo parameters from ASCII files written by Matlab in a predefined format, called the .svo format. It would read the number of inputs and outputs from the file and then the type of controller and dynamically allocate memory, read in the multivariable parameters, and dump these parameters to the running DSP between sample instants (Figure 3).
- Hooks were in place for state-space, but the only DSP structures that I had implemented were polynomial form discrete-time filters for controllers.
- There were digital "knobs" to allow the gains and offsets from any of the MIMO filters to be adjusted in real-time. They could also all be switched on and off individually.
- Any signal in the DSP code could be designated as one to be tracked and with every sample instant, the contents of that signal or parameter would be saved on the host computer via dual port memory.
- With the initial work of Carl Taussig, it was interfaced to the HP 3563A Control Systems Analyzer (CSA), an augmented version of HP's 3562A DSA, that was capable of both analog and digital frequency response function measurements and curve fits [14], [15], [16], [17]. The CSA had a superset of the features of the DSA, specifically enhanced to work with discrete-time systems. Significantly, measurements could be coordinated from the host computer and completed measurements and/or parametric curve fits were uploaded to the host computer to be used in Matlab.

I called the research system, the Banshee Multivariable Workstation (BMW) [12], borrowing from a car advertisement to refer to it as, "The Ultimate Servo Machine", because I was young, foolish, and arrogant.[3] The BMW did some things extremely well:

- It separated the problem into three levels of computational latency: hard real-time, near real-time, and non real-time.
  In hard real-time, computations have to be completed between sample instances. Here, the number of clock cycles for multiplications and additions are counted, and division and trigonometric functions are avoided. This is the playground of the efficient IIR filter and the PID, where latency is most critical. Here, DSP chips and FPGAs (Field Programmable Gate Arrays) are used for high speed applications, while slower systems are the realm of small micro-controller chips.
  On the other end is non-real-time computation, which is the land of the CAD tools (Matlab, Mathematica, Maple, Python, Octave), in which designs are formulated and measurement data is post processed. The acceptable latency here is on human time scales. No safety critical computations get done here, but designs for the real-time block are created. This is the world of computing that most people are acquainted with, where problem solving is multiplexed in time with email, chat boxes, and web sites.
  In between is the middle ground of near real-time. This level has to keep up with the hard real-time in an average sense, but does not need to be completed on every sample. On the other end, it needs to be able to interact with the non-real-time level and translate data back and forth for that level. It is characterized by some block calculations and buffers between the different levels. This area is typically inhabited by fast processor chips running lightweight operating systems, such as a stripped down version of Linux.
- The BMW translated between Matlab designs and the low level implementation with a few keystrokes. Because the DSP chip was floating point, I could download transfer function controllers, each in an IIR filter form

[3]No longer young.

with the numerators and denominators expressed as polynomials in z^{-1}.
- It connected real-time measurements and frequency response measurements to control design CAD tools, in this case PC Matlab. In a time when most people printed out their measurements from the screen on a pen plotter, this tool provided seamless integration to upload measurements into design software.

In fact, it did everything that I had promised my managers it would do, yet it did not do nearly enough. I had overcome some basic bottlenecks in applying advanced control design to high speed real-time systems, and yet, for the mechatronic system I was working with, this was wholly insufficient. Issues with the BMW system included:

- The optical disk fine actuator had flexure resonances at around 11 kHz. The 25 kHz sampling rate that was the upper limit of the system barely allowed these resonances to be seen, as they were so close to the Nyquist frequency.
- The original ASPI ADC/DAC card had two ADCs and two DACs. Each ADC fed into a register which also fed a DAC, such that one would write the DAC output to the register, and when the timer fired for the ADC conversion, one would read the ADC value from that same register. What this meant was that it was impossible to operate a feedback loop without at least a full sample of delay due to the ADC/DAC structure. There was no way to implement a current mode estimator [18] or minimal latency control until a new ADC/DAC board was built for me by a co-worker, Lennie Kiyama. Once Lennie's new ADC/DAC board was in place, I could clock data to the DACs without waiting for the next ADC clock. This allowed me to precalculate most of the filter computations and have minimal latency control (Section XII).
- Connecting the digital CSA to the MIMO control loop required a huge amount of work (much of it done by Carl Taussig). Not only did digital logic pods need to be connected onto the DSP board, but an entire bus architecture and sequencing method had to be devised to route the appropriate signals from the MIMO loop in and out of the CSA.
- Extracting pole-zero-gain models from FRFs failed repeatedly. The HP 3563A allowed one to make fully digital measurements, but then these measurements are farther removed from physical dynamics. This made the fit of discrete-time dynamics very susceptible to imperfections in the FRF (Section XVI).

Some engineers in the product division were using fixed point DSPs sampled at 50 kHz – twice as fast as the BMW. Their designs were simple concatenations of multiple digital lead circuits with a notch at 11 kHz, and they were able to achieve better results on the individual SISO focus and tracking loops than I had.

It took me years to understand the full swath of things that had happened on this project. Yes, I had created a great step forward for our lab system computation, and this would pay off when I worked on accelerometer feedforward for hard disk drives [19], [20], [21]. While all of the problems I solved were present in most real-time control research systems, I had to learn a lot more about how things worked in industrial products and more importantly, what were the root causes of success or failure.

IV. What the Textbooks Tell Us

In 2008 I was giving a talk at Karl Åström's advanced control class at UCSB, describing all the things that affect mechatronic control systems and using optical disks, magnetic disks, and atomic force microscopes (AFMs) to compare and contrast [9]. As I wound down the hour-long talk, a student near the front was looking more and more sour. When it was time for questions, his hand went up. "So, you're telling me all this state-space stuff we've been learning all these years is useless?" *Umm. Inside my head, a voice is screaming, "Think, Danny! Think!"* Slowly, I started to say, "Well, state-space control is model-based control, and model-based control depends on having a good model." Picking up steam, I continued, "Face it, most of the time, the models you guys are working with have little to do with reality." In my desperation to not crush what turned out to be a Ph.D. candidate about to graduate, I had summarized the right answer: To apply model-based control in the real world, we need to extract models from measurement data of the physical system. However, for high Q mechatronic systems, identifying the time-domain methods that perform some cost minimization between sampled measurements and discrete-time models [22], [23], [24] is hard because the inputs driving the model do not focus signal energy in a narrow enough range to identify high Q features. The phenomena is easy to see when this data is passed through an FFT to produce a frequency-domain representation. In this case, frequency bins are often wider then the high Q features, and the accuracy of the representation is very weak when the signal level is low, such as at high frequency when the physical system response rolls off. These are the most common methods in the academic texts [25], [26], [27], [28], [29]. Furthermore, lots of parameters are adjusted at once, whether it is for system ID or adaptive control [24], [30], [31], [32].

In the texts, there is a disconnect between methods that try to solve the entire problem, taking on every energy storage mechanism in the system (the states) and the tried and true methods such as PID control. We are told that when we really want to optimize system performance, we need to work in state-space, but the above difficulties in generating accurate models from measurement data are often glossed over.

Likewise, any digital control system depends upon the real-time system that will host it, and yet there is only cursory discussion of the topics of data conversion, fixed point formats, and real-time programming.

Elegant theory is often done in continuous time. (Some of it is in pure discrete time, but then the connection to the physical world is often abandoned.) The examples are almost

invariably low Q, or if there is a high Q resonance, there are only a few and they are distinct.

Discretization, when used, has been handed to Matlab's *c2d* function, with little thought about the consequences. While *c2d* has many options, the Zero-Order Hold (ZOH) approximation [18], [33] is most often used. However, for anything other than a double integrator, relating analog states to the digital states, is next to impossible. Physical intuition has been lost, but this is not given much weight.

In the sections that follow, I will try to show how this contrasts sharply with how most successful practicing engineers I encountered did their work. I will go through a few of these example methodologies, and then try to pull together some common observations.

V. Industrial Mechatronic Control

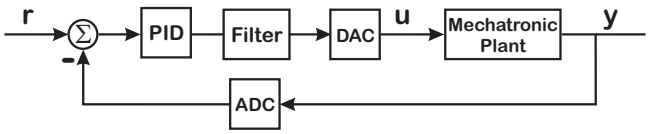

Fig. 4. A practical digital control loop for a mechatronic system. The digital controller is often implemented as a PID like controller in series with filtering to lower the effect of high frequency resonances. Recall from Section I that we are focusing on small mechatronic control systems – of a size and cost that then cannot be adjusted by a human engineer for each device.

One can argue that what are thought of as human built control systems are increasingly digital. Those that are not are either operating at frequencies that are too high for computer control, are using some built in regulation mechanism in the device or process, or are small analog circuits where the feedback loop aspect is almost taken for granted. Anything that folks recognize as a feedback controller is done digitally, probably with a digital PID and perhaps a few notch filters. However, that digital PID is usually a three term controller based on analog PID theory, with the effects of discretization largely ignored. The last two decades have also seen the rise of fuzzy logic controller implementations, where discretization issues are largely handled heuristically.

Why are these systems digital? The simple answer is that it is hard to reprogram a circuit. A more complete answer is that it is hard to do much analysis, design, or schedule any signal changes without some sort of computer technology. Furthermore, conditional logic, the *if-then-else* or *case* switching in combination with filtering and control is much easier using digital logic and processors.

Digital controllers require some skill in real-time programming, and yet, most people skilled in control theory often have far less real-time programming skills. Real-time programming involves picking ADCs and DACs. It involves choosing processors with minimal latency for the job. It involves deciding between fixed point and floating point operations. These skills are not typically taught in engineering schools (apart from "that *one class* in EE") and programming constrained by the evolution of physical responses is certainly are not emphasized in computer science curricula.

There are many good real-time programmers in industry, but like their analog circuit designer cousins, much of their skills are informally obtained over time. As a rule, real-time programmers do not have a lot of elegant theory to work with. Many have come from a signal processing background and have some knowledge of filtering and coding of filter equations. Typically, they do not have a strong understanding of latency. This is a major issue because it means that DSP and FPGA architectures are largely oriented around signal processing – where latency does not matter – as opposed to feedback control – where latency does. I once commented to a Xilinx support engineer that they had lots of design examples and built in hardware constructs for FIR filters, but none for IIR filters. His response was essentially, "Systems with feedback are 5% of our business, so we can't spend much effort on it."

In industrial control, it is always about latency, sensors and actuators, circuits, filters, and programming. Furthermore, far more value is placed in getting anything to work than is placed on the mathematical elegance or optimality of the solution. A 5% or even 20% improvement does not warrant a whole new set of designs, unless there is significant profit advantage in such a small percentage change. Many managers trying to meet product schedules will push for the minimal system that meets the marketing specifications, while there are some that take the long view and push for methods that enable a class of products. Finally, experienced designers will break the system down into separate components that can be designed and tested individually, much as experienced programmers break long programs into functions and subroutines.

We will torture this analogy more later in the paper when we argue that, just as programs need to retain more structure than a hodgepodge of subroutines and functions, our control designs must retain some structure to allow them to be debugged. Furthermore, just as good programs draw their structure from the type of data on which they operate, good control designs must draw their structure from the physical systems to be controlled. For the remainder of this discussion, we will consider the final implementation of any controller to be in a digital circuit, unless otherwise noted. There may be analog portions, e.g., op amp circuits and filters, but the "main" controller will be digital. As such, how we handle discretization is important. If we lose connection with the physics of the system, we generally have a problem.

VI. Phase-Locked Loops: So Much Feedback, Such Simple Analysis

Early in my career, a fellow engineer at Ford Aerospace named Dan Witmer walked into my cubicle and asked me how I would do nonlinear analysis for a phase-locked loop. "What's a phase-locked loop?" Once he patiently explained it to me, I blindly stated that I would simply use Lyapunov redesign [6], [31]. While nonlinear analysis of phase-locked loops was an interesting subject, it was not the main learning point of these devices. I have claimed that PLLs are the most ubiquitous feedback loops built by humans [34], showing

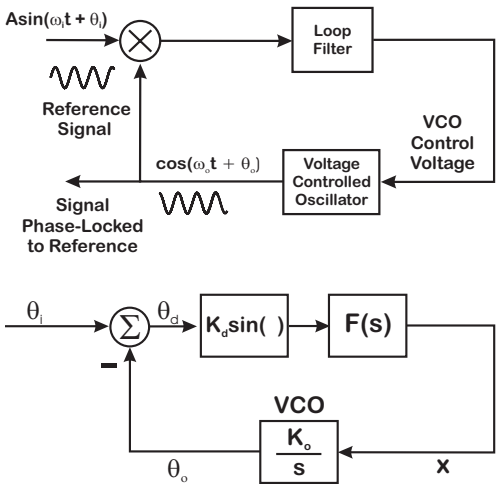

Fig. 5. A simple analog PLL and its baseband model. On top is the basic loop model, where a sinusoidal signal of known frequency but unknown phase enters the system, and a voltage controlled oscillator locks to that input (reference) signal. On the bottom is the simplified nonlinear model of the baseband system, that is the one arrived at by looking at only how the phases of the oscillatory signals, and not the signals themselves, behave. One final approximation, of linearizing the sinusoidal phase detector, allows the phase behavior to be analyzed as a simple analog feedback loop. The VCO is the "plant" and it is modeled as a simple integrator.

up in all of our smart phones, digital watches, and every other computational device we have, and yet the feedback analysis done in textbooks is only of the simplest type [35], [36], [37]. Discussions with PLL experts at several companies also showed that while they knew intricacies of the circuits and the envelope behavior of the phase detector, they generally used only very simple linear feedback analysis in their designs. They paid surprisingly little attention to the stability of the PLL.

It took a while to understand that since most PLLs were first or second order, and for most of these one could show that even the most basic rules of filter design led to closed-loop responses for the phase-space where the parameters that made the linear model stable also made the nonlinear model stable [6], [38]. (In this case phase space refers to the modulation domain or the baseband or envelope behavior of the PLL.) These loops were simple and stable, because the "plant" to be controlled was *always* an integrator, as shown in the lower drawing of Figure 5, and the loop filter was either a gain or a first-order lag, which resulted in stable closed-loop behavior of the system. The Lyapunov analysis showed that in these cases the parameters that made the second-order linear model stable also made the second-order nonlinear model stable. For the circuit designers creating PLLs, they knew from experience that even the simplest, dumbest controller (a.k.a. loop filter) would produce a stable response and so they gave it no mind. Higher order PLLs failed this, and thus were harder to analyze [5].

If our open-loop system is adequately modeled by an integrator, then feedback control of that open-loop system becomes trivial (Section XII). It is easy to show that the first and second-order analog PLLs are always stable, even with the sinusoidal phase-detector nonlinearity [6]. Furthermore, with the right discretization, even the classical discrete-time PLL was stable [38]. What about harmonics? Didn't the designers need to throw in some filters to get rid of signal at the carrier frequency and its harmonics? Why were these not in the textbooks? The answer came from HP/Agilent PLL expert, Rick Karlquist, who laughed and said (more or less), "Well, no RF engineer worth their salt wouldn't know to put in those filters! It goes without saying. That's why it doesn't have to be put in the textbooks." In this moment I realized that any PLL that was not second order was likely to be beaten into a form that looked second order. All those analog RF filters were there, but they were never considered part of the analysis. A lot of work was done to make the system look like an integrator, and then control was done from there. What kind of control was done? *Both kinds, lag filter and PI control.*[4] The designers were left to work out the more taxing PLL design problems of having a good phase detector and an oscillator with minimal phase noise.

VII. "We'll just call you Bruce," or Why Industrial Controllers Are PIDs

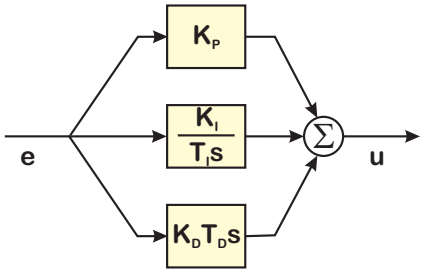

Fig. 6. A simple analog PID model.

PID controllers are the standard of comparison for graduate students, the metric against which their algorithms invariably show improvement. The are ubiquitous, to the point that common language guides exist for individuals skilled in real-time programming, but with no actual knowledge of feedback or filter theory. Tim Wescott's excellent article, *PID Without a Ph.D.* [39], is a shining example of a practical and useful guide that makes the programming of digital PID controllers accessible without explaining the math behind the work. If one peruses available books on or with chapters on PID control [18], [40], [41], [42], [43], one finds that they are either described in continuous time or they are described in discrete time with little explanation about the discretization choices. However, these texts show that the discretization is almost always a backwards rule equivalent. This seems to be ignored or accepted without comment. Why?

One reason might be that the analog PID of Figure 6, which has the analog model of

$$C(s) = K_P + \frac{K_I}{T_I s} + K_D T_D s \tag{3}$$

[4]A tip of the hat to *The Blues Brothers* film.

can, if we set $T_I = T_D = T$, where T is the sampling period, be discretized with a backwards rectangular rule to yield [44]

$$C(z) = K_P + K_I \frac{1}{1 - z^{-1}} + K_D(1 - z^{-1}). \qquad (4)$$

This means that the analog and digital PID coefficients are trivially related through the sample period. Equation 4 lends itself to easy implementation in software, and can be easily modified to add integrator anti-windup.

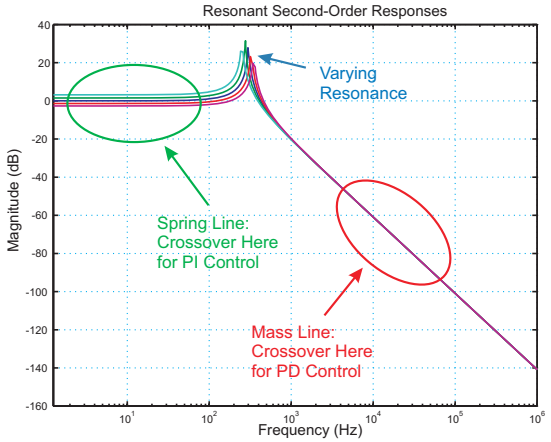

Fig. 7. The regions of PID control for a second-order, resonant plant. Below the resonance, where the magnitude is flat and the phase is near 0° is called the spring line. This references behavior in that frequency zone being dominated by the spring force. Above the resonance, the region which is asymptotic to a magnitude slope of -2 or -40 dB/decade (as with a double integrator) is often called the mass line.

Another aspect of PID controllers that isn't often discussed is the region of operation, relative to something like a second-order mechatronic plant, such as the one diagrammed in Figure 7. In many cases, the control action is taken and removed well below the resonance, and in this case the proportional plus integral (PI) part of the controller is used, to make the open-loop response look like an integrator near gain crossover, and then remove the phase effects before the $-180°$ effect of the resonance. In these cases, the derivative term is seldom used. Similarly, when the resonance is well below the crossover region, the system can be controlled as if it were a double integrator, and in this case the proportional plus derivative (PD) action is used. In these cases, there may still be a motivation to use integral action at low frequency and thus a PID with derivative filtering resembles separate lead and lag controllers. In neither case is precise knowledge of the resonance needed. It is only when the crossover region is relatively close to the resonance that more complex methods must be used, such as those described in [45], [46], [47] become important. The formulas that follow concentrate on the latter, more complex case, while emphasizing the effects of digital implementation as described in [44], [45]. In the case of first-order time delay models, such as those found in process control applications [48], full PID control is used, where the PI portion provides low-frequency gain and the PD portion adds some lead to compensate for the phase due to the delay, the requirements of matching a high Q resonance are not present.

All of this smacks of the *robustness to imprecision* that Lotfi Zadeh often spoke about when describing problems well suited to being solved with fuzzy control [49].

VIII. Can I Talk to You About . . . Filters?

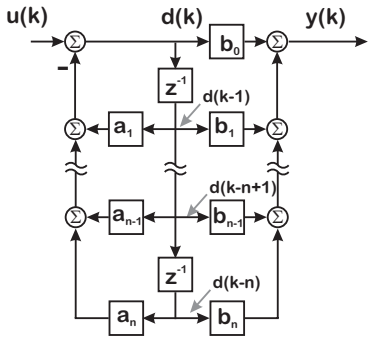

Fig. 8. Direct Form II configuration of an n^{th} order polynomial filter [50].

Just as the RF engineers do with PLLs, many scientists and engineers add independent filters to the signal path of their system. For an analog circuit designer, these filters are typically first or second-order op amp circuits dropped into the signal path to "clean things up". While the designer will note how the filter improves the signals they see on the scope, few tie any phase effects into things they might see in the overall loop response. When filtering is done on the digital side, it can range anywhere from an average of the last N samples to a cascade of M first-order digital filters to an n^{th} order discrete filter based on an analog Butterworth filter prototype [50], [51]. Depending upon the point in the signal path, the phase effects of these filters may or may not be significant, but it is often the case that this is not considered from an overall system view.

FIR filters are rarely used to shape loop dynamics, because the number of delays (taps) needed to produce the same response as an IIR filter is much larger, and this impacts latency. That being said, programmers and scientists often make the mistake of adding in functions that average the last N ADC samples or stringing together M first-order low-pass filters (which they easily understand), rather than generating a more effective L^{th} order filter, where $L < M$.

The advent of high speed floating point on DSP chips has made it possible to implement high-order polynomial filters of the form shown in Figure 8 in real-time. This allowed the BMW system of Section III to take controller designs from Matlab and implement them with little modification in real-time DSP on the TMS-320C30. The controllers are implemented as IIR filters of the form in Figure 8, which are written as transfer functions in the z transform operator, z:

$$\frac{Y(z)}{U(z)} = \frac{b_0 z^n + b_1 z^{n-1} + b_2 z^{n-2} + \ldots + b_n}{z^n + a_1 z^{n-1} + a_2 z^{n-2} + \ldots + a_n} \qquad (5)$$

or equivalently in the unit delay operator, z^{-1}, which lends itself readily to real-time implementation in assembly [52]

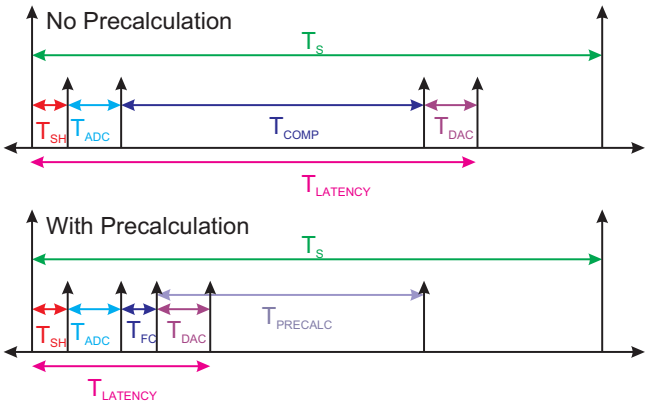

Fig. 9. Input and output timing in a digital control system. The top drawing is without precalculation; the bottom drawing is with. Note that precalculation can be started as soon as the output has been sent to the DAC and therefore is in parallel with the DAC conversion time. The computation time, T_{COMP}, of the top diagram is now split into $T_{PRECALC} + T_{FC}$ where $T_{PRECALC}$ is the computation time needed for the precalculation and T_{FC} is the time needed for the final calculation after the input sample. Modulo some small programming overhead, the split time should equal the total computation time. Here T_{SH}, T_{ADC}, and T_{DAC} represent the sample and hold, ADC conversion, and DAC conversion times, respectively.

or a high-level language:

$$\frac{Y(z^{-1})}{U(z^{-1})} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_n z^{-n}}{1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_n z^{-n}} \quad (6)$$

The transfer function in (6) has an advantage in that the coefficient of the current output term, $y(k)$, is 1, so the filter implementation is:

$$\begin{aligned} y(k) &= -a_1 y(k-1) - a_2 y(k-2) - \ldots - a_n y(k-n) \\ &\quad + b_0 u(k) + b_1 u(k-1) + \ldots + b_n u(k-n). \end{aligned} \quad (7)$$

This form allows the designer to minimize the computational latency diagrammed in Figure 9 by rewriting (7), since $y(k)$ depends mostly on previous inputs and outputs. The only current value needed is $u(k)$ and this is only multiplied by b_0. So we can break this up into [18]:

$$y(k) = b_0 u(k) + \text{prec}(k), \text{ where} \quad (8)$$

$$\begin{aligned} \text{prec}(k) &= -a_1 y(k-1) - \ldots - a_n y(k-n) \\ &\quad + b_1 u(k-1) + \ldots + b_n u(k-n), \end{aligned} \quad (9)$$

and $\text{prec}(k)$ depends only on previous values of $y(k)$ and $u(k)$. This means that $\text{prec}(k)$ can be computed for step k immediately after the filter has produced the output for time index $k-1$ [42]. When the input at time step k, $u(k)$, comes into the filter, it needs merely be multiplied by b_0 and added to $\text{prec}(k)$ to produce the filter output. Thus, the delay between the input of $u(k)$ and the output of $y(k)$ is small and independent of the filter length.

Although this form of controller easily admits the use of precalculation, there are two large drawbacks. The first is that the higher the order of the digital filter, the harder it is to relate any of the digital coefficients to anything physical. Second-order digital polynomial filters still can maintain some ties to a second-order analog transfer function, but

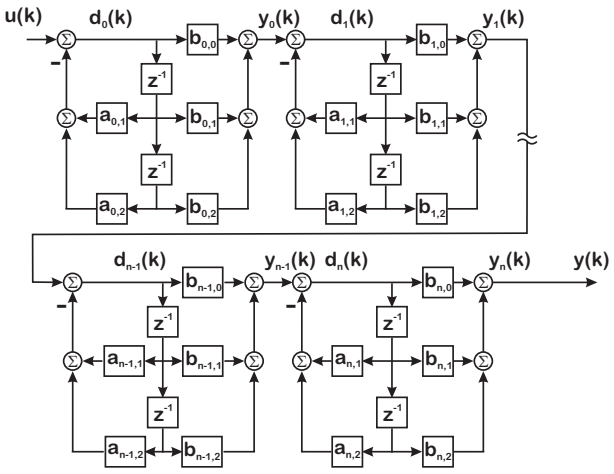

Fig. 10. A high-order filter structured as a serial chain of biquad filters.

beyond that it gets much harder. The second issue is that the elements that generate discrete filter coefficients can become extremely sensitive, particularly when they relate to high Q elements near the unit circle. As these get multiplied together to form the polynomial coefficients, not only is physical intuition completely lost, but the coefficients become even more susceptible. This is particularly true with fixed point arithmetic used in many DSP and FPGA implementations. This is why it is common practice to implement higher-order digital filters (6) for high-Q systems using a series of second-order filters, known as biquads, as diagrammed in Figure 10.

In Figure 10 the i^{th} biquad would have equations:

$$\frac{Y_i(z)}{U_i(z)} = N_i(z) = \frac{b_{i,0} + b_{i,1} z^{-1} + b_{i,2} z^{-2}}{1 + a_{i,1} z^{-1} + a_{i,2} z^{-2}} \quad (10)$$

which gets implemented in the time domain as:

$$\begin{aligned} y_i(k) &= -a_{i,1} y_i(k-1) - a_{i,2} y_i(k-2) + b_{i,0} u_i(k) \\ &\quad + b_{i,1} u_i(k-1) + b_{i,2} u_i(k-2) \end{aligned} \quad (11)$$

It turns out that it is easier to implement this using the delay format [52] which resembles a controller canonical form [53] in control or a direct form II IIR filter [50], [54], [55]:

$$d_i(k) = -a_{i,1} d_0(k-1) - a_{i,2} d_i(k-2) + u_i(k) \quad (12)$$

$$y_i(k) = b_{i,0} d_i(k) + b_{i,1} d_i(k-1) + b_{i,2} d_i(k-2) \quad (13)$$

Biquads are nice because the growth in values can be limited by the short nature of the filter. Thus, finite word length problems are minimized as the sums from the numerator and denominator can balance each other out for a well designed filter [52].

The issue is maintaining the ability to do final calculations in the form of (8) [52]. To my knowledge, this has not been possible until the work I discuss in [56], [57]. I will present an overview of that work in Section XV.

IX. Implementation Details Aren't Small

One of the difficulties in industrial control is the constant tug-of-war between the elegant solution one would like to conceive and the realities of trying to make something work in a real product. It is why – even though the value of simulation is appreciated – nothing convinces the product engineers like seeing something in moving metal. The implementation issues can be so prevalent as to completely remove engineers from thinking about the original elegant theory that they started with.

Even a SISO digital control system will have a convoluted signal path which involves all the sensors, transducers, amplifiers, and the plant itself. While our textbooks like to consider much of this to be lumped into the plant, they are subject to design. It is often the case that the control engineer is the one person on the team with a system level view of the signal path, but it is also often the case that engineers will optimize their part of the problem without considering the other aspects. The buzzword for actually considering more than one area is *co-design*, but what it really means is that one needs to take a step back and view the overall impact that each piece of the system has on the overall response and not limit themselves to one piece.

A classic example is anti-alias filtering, ordained in digital control books [18], [42] as necessary to keep signals at frequencies above the Nyquist rate from entering into the digital loop. In practice, though, these filters never come for free. A Butterworth filter [50], [51] may have a flat magnitude response in the pass band, but has pretty severe phase attenuation that can affect the achievable bandwidth of the system. Signal paths from sensors into ADCs may include lots of voltage changes and various filters to eliminate board level noise sources. Analog designers, working without interaction with control engineers, will often severely cut achievable bandwidth because they did not appreciate the impact of their filter choices on the loop.

The amount of processing available for real-time implementation is limited by the cost of the product. It is a sad but true fact that one cannot put a $200 FPGA into a $50 disk drive. At the same time, the decision is often between a $6 FPGA which theoretically could implement a digital controller using fixed point math at a 100 kHz sample rate, and a $10 ARM processor, which is far easier to program but cannot guarantee precise sampling at 10 kHz because of interrupt timing, and cache misses in the memory. It depends on the cost, power dissipation, and physical interface of the circuitry and what the system requirements can support.

The choice of computational hardware and software has a big effect on achievable bandwidth. Signal processing applications are far more prevalent and generally far "safer" than control applications. However, at the heart of many of these signal processing applications is a feedback loop, often treated as an afterthought. This is akin to those large European clocks with all the automata that operated at different times [58]. The automata were open loop, but their operation depended upon the proper operation of the clock escapement which implemented the timing feedback loop [59], [60], [61].

Most digital hardware is built with signal processing in mind, not feedback control. The result is a set of converters with extra delay, signal processing chips that do not minimize latency, algorithms that may be efficient from a total computation point of view, but are not efficient minimizing the time delay between the time of the current sample and the controller's output to that data. Often, the idea of control filter precalculation, as discussed in Section VIII is lost.

The generation of usable models from measured data on physical systems is one of the great disconnects. Many of the published methods emphasize time domain and FFT based measurements [22], [25] for extraction of parametric models and yet these have little success with high-Q systems. The relay based tuning is very popular for PID controllers [40], but this method cannot be applied to systems with multiple resonances. This is a major breaking point in the modern methods as applied to high-Q mechatronic systems: we cannot apply modern control without parametric models and yet we are rarely able to extract high fidelity parametric models from high-Q systems. In the end, we are left with system models that are often described as "mostly double integrator," but this begs the question: if it is just a double integrator, why do we need all that model-based control? The answer is to reexamine our measurements and model extraction methods, and these will be discussed later.

The signals from which one generates an error signal is often buried in some high frequency signal. This is clearly the case for PLLs [34], but also the case in the AC mode of an atomic force microscope (AFM) [62]. Likewise, the position signals are encoded in high frequency bursts in hard disks [63]. In these cases, the standard methods of demodulating error signals are often envelope methods that are not coherent and therefore not only let a lot of unnecessary noise into the error signal [64], but often have long delays to detect a change in value. Coherent methods can do a lot to address both of these issues, but it requires delving into the non-sexy world of demodulation, with the eye towards producing an error signal that is far cleaner and far faster [13], [65], [66], [67], [68].

In many practical applications, the work around is to do low bandwidth control, e.g., PI control, throwing in some broad (i.e., robust) notches to damp particularly bad high frequency regions, and run tests whose results (consciously or not) make sure that the noise amplification of the feedback controller's disturbance rejection does not fall in a frequency band where there is a lot of broadband noise.

The obvious problem with this approach is that it severely limits the achievable bandwidth. To do differently, we have to model the system far more closely at high frequency and we need to understand noise sources and their effects on the loop (Section XVIII).

X. Summary of the Disconnect

Putting this all together one would surmise that while academic controls research is filled with state-space and

optimality, most industrial control of mechatronic systems is filled with hardware choices, PID controllers, a few filters, and some simplistic programming. Even a system that allowed one to directly implement designs from Matlab in real-time floating point DSP (Section III) under performed a system that implemented a handful of leads and sampled fast. It is against this backdrop that I received a call from a Ph.D. student at a school in the middle Atlantic region of the United States back in 1994, in which the student started by saying how they had read some of my papers and found them quite interesting, and then proceeded to say, "I am very interested in working on optimal control." Before I could stop myself, the words reflexively jumped out of my mouth, "Me, too, pal." To that former student, my apologies, however, I would have *loved* a chance to work on optimal control, or on adaptive control, or on any advanced methodology. Instead, all of my work was driven by the physics and business models [13] of the problem, not by any preferred methodology.

It seems fairly universal that practicing engineers and hobbyists alike take an approach of "divide and conquer" for all of these control problems. Consider the humble operational amplifier (op-amp) [51], [69]. Designing complex circuits without op-amps is difficult, because one part of the circuit will load all the other parts. The op-amp provides separation of the different sections of the circuit, allowing the designer to focus on one isolated problem at at a time. Likewise, the analog filters that notch out harmonics generated by the phase detector in a PLL reduce the problem to one of looking at the phase – a feedback problem for which the plant has been reduced to an integrator [34]. At that point, the control design is PI control. Likewise, in mechatronic systems, a practicing engineer or hobbyist will add in filters (analog or digital) to remove the effects of resonances until the problem can be dealt with using a PID controller (analog or digital) or a fuzzy control block. The plant might not have started out as second-order, but the tweaks went in until the needed "controller" is very simple and physical. In fact, this idea of isolation, of divide-and-conquer, is pretty fundamental and clarifies the work because the individual fixes have a close tie to the physical phenomenon they are addressing. Even a series of biquad filters to deal with a large number of resonances and anti-resonances in a flexible system, is easy to understand, as each second-order numerator and denominator can be assigned to a particular feature of a Bode plot. It reminded me of something my first controls professor, Bob Snelsire at Clemson University had said, "All problems are second-order and no problems are second-order." While Bob was talking about looking at the large scale, baseband behavior of the system, it seemed that practicing engineers beat the problem until it became "second-order" and then were able to apply simple control methods to the new problem.

On the other hand, our optimization tools, seem to be based on giving up structure to a polynomial or canonical form, so as to let the math work more easily. In doing so, in handing the problem over to our optimization tools, we have given up on the physical intuition that many of the classical methods gave us. Automatic curve fits [70], [71] give us 16^{th} order models for what physics and our FRF measurements tell us are 6^{th} order dynamics [72]. Even then, the coefficients of our models may only show a few LSB changes for hundreds of Hertz worth of frequency difference. Our tools tell us to trust the math, but experienced practicing engineers never trust an answer that does not simplify to something physically observable. Is it any wonder that explanations of optimal control and Kalman filtering are almost always done with first and second-order examples? The difficult/advanced examples might include a double integrator and a single resonance, but after that, the relationship of the states to the physics is often lost – particularly after a ZOH equivalent [18] discretization.

This brings to mind another disconnect: discretization. Early in our control education, we learn about the different methods to discretize a dynamic response, and then we promptly ignore most of them. It seems that much of the textbook work involves discretizing a plant system model and then applying digital control to that, while industrial mechatronics work is more likely to have analog plant models and analog controllers which then get implemented with a discrete equivalent. Furthermore, separate discrete equivalents are implemented for different parts of the controller, the PID, the filters, etc. which means that each controller component is discretized independently. While this may overestimate the computational delay, it does keep the controller very physical, very understandable. Compare this to the single discretization of a large plant model where the relationship between discrete plant coefficients and their physical origin is obscured, along with the high-order discrete controller which is far from physical. While one can argue that working with such a discretized plant is more mathematically accurate, if model reduction is applied, this claim is much weaker.

Taking a step back, this dichotomy comes from a feeling that optimization and physical structure require different models, that they are an either-or proposition. However, I believe the problems can be rethought, that we can design control structures that preserve physical intuition, but allow us to use our math tools. The rest of this paper will list a series of methodology adjustments – which, while useful on their own – start becoming incredibly powerful when put together. I hope to convince the reader that these adjustments do much to push advanced methods in a direction that makes them practical.

XI. Principles for Mechatronic Control

In a plenary lecture at the 1998 American Control Conference, Babatunde Ogunnaike was describing large scale chemical process control problems at DuPont. After showing the layers of industrial processes and management buy in from different groups, he finally got to the "control" part, saying something to the effect of, "Process control is 90% process and 10% control. Only once you have done all of these things do you get to do your $\dot{x} = Ax + bu$." This rang true to everything I experienced before and since that

talk. However, I found that many folks working in industrial control problems got so tied up in that 90% that they forgot to apply some basic guiding principles of control to their overall problems. Once I started doing that, I found that I was able to make a fair amount of progress.

Putting all of the above together, a pattern emerges that one can exploit to make mechatronic control far more practical than the standard academic state-space models and yet far more exciting than a "PID and some notches" approach that are so common. The principles will be stated here, and they will be followed by sections describing the implementation of these principles.

- Integrators are really, really easy to control. Therefore, if we can make the open-loop response look like an integrator, we have a system that is very easy to control. If we can properly identify and compensate for all our high-Q dynamics, i.e., *if we can beat our open-loop response into the shape of an integrator response, then our limits become noise and latency*.
- Most mechatronic systems are second-order plus some crap. A smart PID will take care of the baseline second-order (Section XIII), while equalizing the crap out (Section XV) will give us our integrator like open-loop response.
- This depends upon carefully identifying the complex dynamics of the system. This involves precise frequency response measurements (Section XIV) and smart curve fitting (Section XVII).
- Latency, latency, latency! When everything else is done, the first main limitation is latency (Section XII), since as Yogi Bera says, "It's hard to make predictions, especially about the future." An appendix to this limitation is uncertainty or jitter in that latency. In many problems today, the digital systems are so fast relative to the time constants of the devices being controlled that latency can be (and often is) ignored. However, for any problem where bandwidth will be pushed to the achievable limit, that limit is imposed by latency. Furthermore, one cannot simply have a fast DSP chip and ignore the latency of tenth-order Butterworth filters on the inputs to the ADCs and the outputs of the DACs. Instead, the entire loop has to be considered in the latency calculation.
- The other main limitation is noise. We might count other disturbances into this, but these can often be detected with a separate sensor. Injection of broadband noise into the loop must be minimized at the source so that we can minimize moving around Gunter Stein's dirt subject to Bode's Integral Theorem. Put extra effort into detection schemes (Section XVIII).
- Once the dynamics have been identified and the overall loop shaping has been done the gain can be set subject to gain and phase margin constraints as well as closed-loop constraints (Section XII).
- High bandwidth feedback control amplifies sensor noise. High bandwidth feedforward control does not. When applying the same modeling principles above allows us to apply feedforward control to the predictable parts of the system, it should be done (Section XXII).
- If you can expose physical parameters that are changing or unknown to your discrete-time system, you can adapt for those with excellent results (Section XIX).
- You can even create state-space models that are accurate and numerically robust (Section XX).
- You can't have any of this without models based on frequent and accurate measurements, and you won't make frequent and accurate measurements if they involve a lot of grunt work. Ergo, connect your measurement system, your physical system, your real-time system, and your CAD system together in a way that makes it trivial to pass measurements, models, and designs amongst these tools. Time spent on this aspect almost always is paid back by an order of magnitude or more return, in the speed and the quantity of measurements.
  The problem with this is usually not the technology to do it; but the will of the engineer and their managers to put up with the busy work needed to make this happen. Engineers hate spending time away from their main area of contribution, fearing it will make them look like they are wasting time. Programmers hate the inelegant interfaces to instruments and the classless nature of real-time programming. Managers consider such projects out of the main line of contribution and not easily accounted for in Microsoft Project.
  However, the low overhead, self consistent connection of time and frequency measurements with CAD programs such as Matlab, Octave, Maple, Mathematica, or Python, and with the real-time processing system open up whole new vistas for co-measurement and design. Consider the humble frequency response measurement. To do such a measurement, one must connect to analog or digital test points, set up the measurement, run the measurement (with appropriate levels of repetition for averaging and statistical confidence), and then transfer the data back to Matlab or its cousins. It is always amazing how many good engineers are willing to do all these steps manually and repeatedly over the course of months, rather than spending a couple of weeks to make it all happen with the push of a button (modulo the wiring, which we deal with in Section XIV).
  On the other hand, programming real-time digital controllers is hard enough without trying to add in sophisticated FRF generation code, and so much is done from time domain measurements or FFTs of time-domain measurements. Breaking down the barrier allows measurement decisions to be made based on what is best for modeling. It also allows measurements to be done repeatedly and quickly so that modeling is based on the best of many measurements rather than "that one time we actually got some lab data." Spend the time to make the data path trivial and consistent, and everything in the control design gets better.
- Although engineers like to tackle these problems in

isolation, and managers like having an individual who "owns" one aspect of the project, all of the above need to be tackled from a *systems* point of view.

XII. Feedback Control of an Integrator

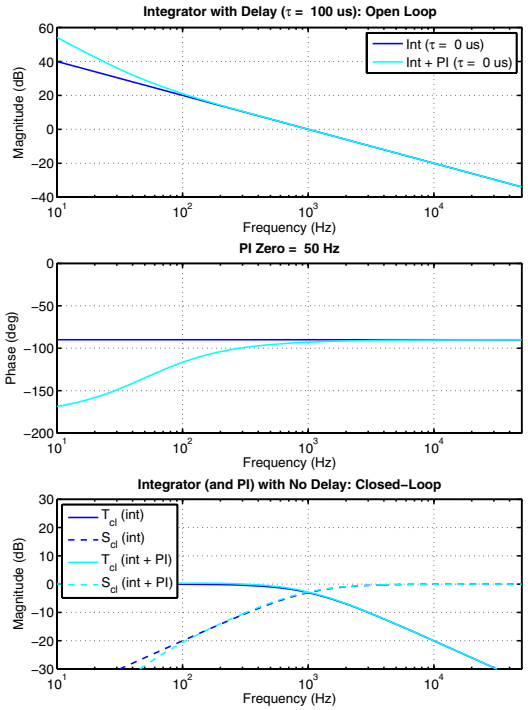

Fig. 11. Plot of open-loop integrator + PI control, without time delay. Essentially, if the PI action is far below crossover, the effects on stability can be ignored, and they usually are. Note the well behaved closed-loop sensitivity and complimentary sensitivity, owing to the 90° phase margin.

Perhaps the easiest open-loop system to control is an analog integrator, K/s. In the absence of delay, it has infinite gain margin and 90° phase margin (PM). As plotted in Figure 11, we see that the sensitivity function, S, has no peaking and the complimentary sensitivity function, T, is an ideal low-pass filter, also with no peaking. Even the addition of a control filter is simple. With the desire for high phase margin any filter action would be removed long before gain crossover. The most likely controller is a lag filter where the pole may or may not be an integrator (PI control). Such a system is usually stable even when discretized and saturated subject to any extra delay in the discretization [38]. This explains the seeming lack of analysis done in PLL work. Even PID control of second-order systems can be viewed as an attempt to close the loop on an open-loop integrator [45], as we will see in Section XIII. However that example also illustrates many of the difficulties involved in "turning the open loop into an integrator".

As Figure 12 shows us that delay is our primary culprit here, it is worth the effort to look at simple delay calculations for a system whose open loop is an integrator. One would imagine that the chief source of pure time delay in a digital controller would be the sampling action. In a discretized system with sample period, T_S, one can assume an average

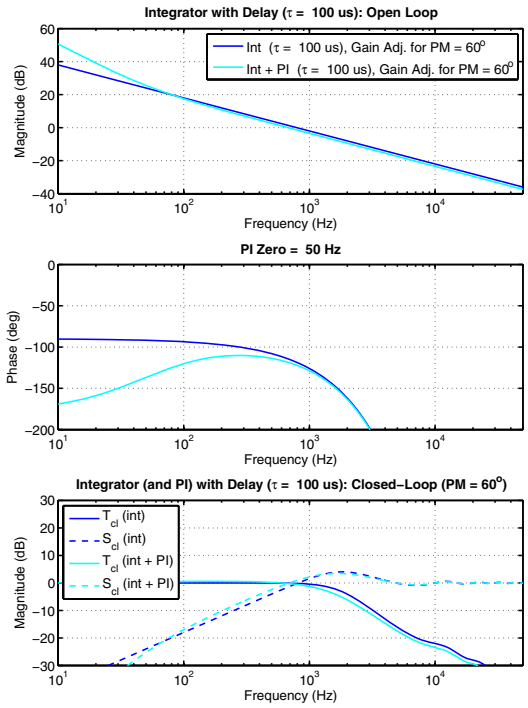

Fig. 12. Plot of open-loop integrator + PI control, with time delay. The effects of pure time delay limit the bandwidth that can be achieved with 60° phase margin and add some peaking in the sensitivity function.

latency due to sampling, $\frac{T_S}{2}$, or a worst-case latency, T_S. However, the full delay of the digital component would include any extra delay in the ADCs, DACs, computation, and communication. Thus we really have, T_D, defined as:

$$T_D = \frac{T_S}{2} + T_{SH} + T_{ADC} + T_{DAC} + T_{comp}. \tag{14}$$

The average delay of seeing events due to sampling has been augmented by the amount of time it takes the digital system to be able to respond to the data when it sees it (Figure 9).

From this, pure delay, one can add the negative phase effects of delay as:

$$D(j\omega) = e^{-j\omega T_D} \text{ with angle } \angle D(j\omega) = -\omega T_D. \tag{15}$$

With phase margin, PM, in degrees, our open-loop phase looks like:

$$-\omega T_D + \angle \frac{K}{s} \geq (-180 + PM)\frac{\pi}{180} \text{ or} \tag{16}$$

$$\omega T_D = 2\pi f T_D \leq (90 - PM)\frac{\pi}{180}, \text{ so} \tag{17}$$

$$f \leq \frac{90 - PM}{360 T_D}. \tag{18}$$

In particular, for a desired phase margin, Equation 18 gives the highest crossover frequency for the open-loop gain plot of the integrator, using the delay of T_D. For the conservative phase margin of 60° we have the simple formula of

$$f \leq \frac{1}{12 T_D}. \tag{19}$$

In an ideal world, where the data conversions, computation, and communication are instantaneous, if we use the average sampling delay of $T_D = \frac{T_S}{2}$, and the conservative phase margin of 60° we have the simple formula of

$$f \leq \frac{1}{6T_S} = \frac{f_S}{6}. \tag{20}$$

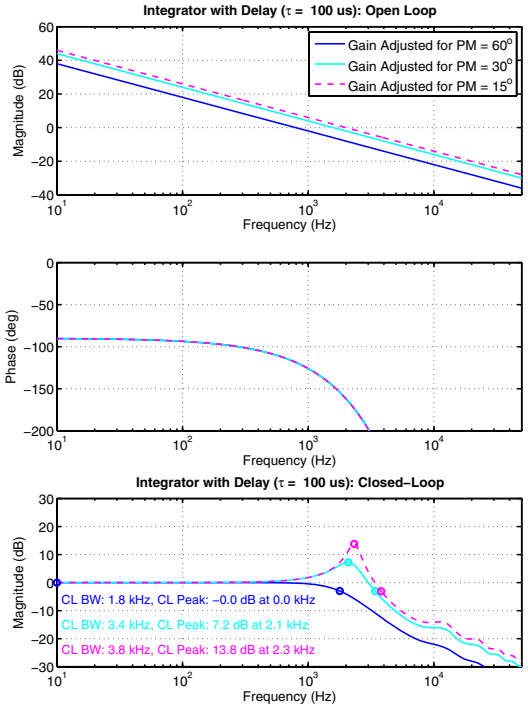

Fig. 13. Plot of open-loop integrator, with fixed time delay. This plot shows the consequence of adjusting the open-loop gain crossover to achieve phase margins of 15°, 30°, and 60°. Note the closed-loop peaking that results from pushing the open-loop gain crossover to the point of such low phase margin.

Why spend much time on such a trivial plant? First it is a ubiquitous plant (see Section VI). Furthermore it gives us a representation of the best case plant that we can control and what limits it. No matter what else we do in our controller, it will be hard to improve on the open-loop crossover frequency limit in Equation 19. We can see that even with such a simple open loop, with a given T_D, if we choose bandwidth over phase margin, we are subject to the closed-loop peaking shown in Figure 13. It seems unlikely that any more complicated open-loop system will do any better. Thus, if we pick the conservative phase margin of 60°, we then get an open-loop crossover limit based on that, and this limits our closed-loop bandwidth, as shown in Figure 14.

Finally, seeing the simplicity of controlling an integrator, an effective control strategy might be to equalize the open-loop response until it looks like an integrator (allowing for some extra integral action at low frequency and some extra roll off at high frequency) and then to set the gains to adjust the crossover frequency.

Even though the equalized system in this step cannot be analyzed as simply as the pure integrator, it is straightforward (albeit tedious) to design Matlab/Octave/etc. scripts to perform this optimization on designs applied directly to FRF measurements. All of the plots in Figures 11–14 were generated this way. Examples of this will be shown in Sections XIII and XVII. However, while this is philosophically easy, the results in Section XIII show that there are major technological pieces that have to be in place to make the control system that simple.

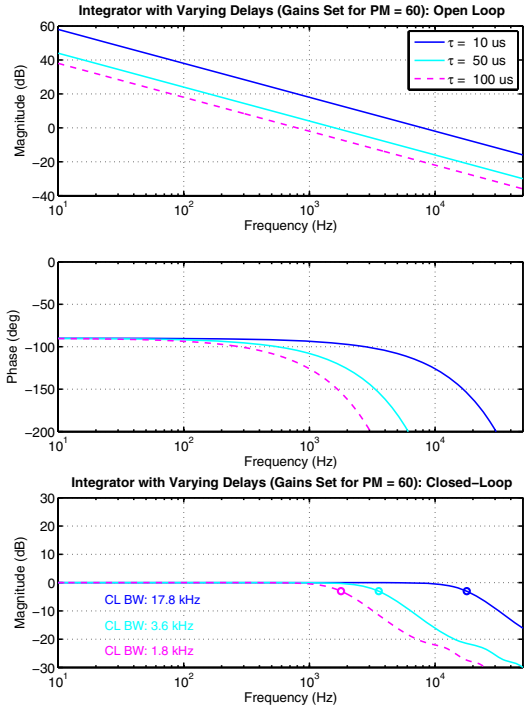

Fig. 14. Plot of open-loop integrator, with varying time delay. Gain is adjusted for maximum open-loop crossover that achieves 60° phase margin. The seemingly obvious result shown is that for an open-loop response that looks like an integrator with time delay. It is the time delay that limits the closed-loop bandwidth for a desired phase margin.

XIII. PID WITH A PH.D.

We return to the question of why PIDs are so ubiquitous. Why is it that if one refers to a controller as a controller, non-engineers will be confused, but if one calls it a PID, they will know what function the block accomplishes. How is it that so many industrial control engineers working on systems both mechatronic and otherwise can simply drop in a PID and get reasonable results. One can argue that they are simply naive or unthinking, but I believe that:

> *If something keeps working over and over again in widely varying situations, it is probably not complete nonsense. There is probably a fundamental reason for this. It is worth the effort to understand that fundamental reason and how far it can be applied, i.e., what limits it.*

In retrospect, this would seem obvious, yet it is often ignored in practice. Making a gross oversimplification, I will say that my professor friends generally ignore such a simple algorithm as being uninteresting for research, while my

industrial friends generally never ask themselves why this thing keeps working. The questions of why these simple things work so well have stayed with me for a long time and I will try to give insight here.

This section's name is a takeoff on Tim Wescott's excellent trade article, "PID Without a PhD," [39] in which he describes in layman's terms the components and programming of a computer based PID controller. In making his article accessible to anyone with programming skills, he glosses over some readily available intuition that we will discuss here.

A digital PID is often programmed as three parallel terms, which are then programatically summed together. The discretization is almost always using a backwards rule equivalent on the individual terms [18], [40], [42]. This should surprise most control engineers since the backwards rectangular rule is considered less accurate than a trapezoidal rule equivalent or a Zero-Order-Hold equivalent. Furthermore, it is far more conservative, mapping the $j\omega$ axis into a smaller circle within the unit circle on the Z-plane. However, the discretization is being applied to the controller, not the plant model The overly conservative approach is actually a savior, because the pole of the discrete D term in the PID ends up at $z = 0$, rather than $z = -1$ as it would for a trapezoidal rule equivalent without filtering [44].

It seems that in practice, a lot of systems are largely modeled as being second-order. Put another way: since second-order systems are so much more tractable than higher-order ones, a lot of engineering goes into making the system to be controlled "second-order". This includes redesign of mechanics and/or addition of analog filters so that the system the controller sees is largely second-order. Given that, mechatronic systems can often be considered to be a second-order part plus resonances. In the case of a hard disk or a wafer stage, that second-order part is usually a double integrator. In the case of an optical disk or an AFM, that second-order part is a spring-mass-damper. In a perfect world we could operate above the second-order resonance and simply add lead to the mass line of Figure 7, making the Bode plot beyond the resonance look like an integrator. In a less perfect world, one might try to equalize the resonance of the second-order section and then proceed towards our integrator shape.

It became clear that the linear PID form could be analyzed as a second-order filter with an integrator and low pass in the denominator and a pair of possibly complex zeros [44]. From there, it seemed that using that filter to equalize the response of a mechatronic AFM actuator was the next step [45]. A measurement of the response was made in closed-loop using an existing controller, the loop was opened and the controller model factored out. This is the blue curve of Figure 15. In wanting to avoid problems I had previously encountered in curve fits, I adapted the HP 3562A curve fitter [71] by restricting the numerator to a constant and the denominator to a simple second-order section over the frequency range of the main actuator resonance. A successful fit is shown in the green curve of Figure 15. The resonance

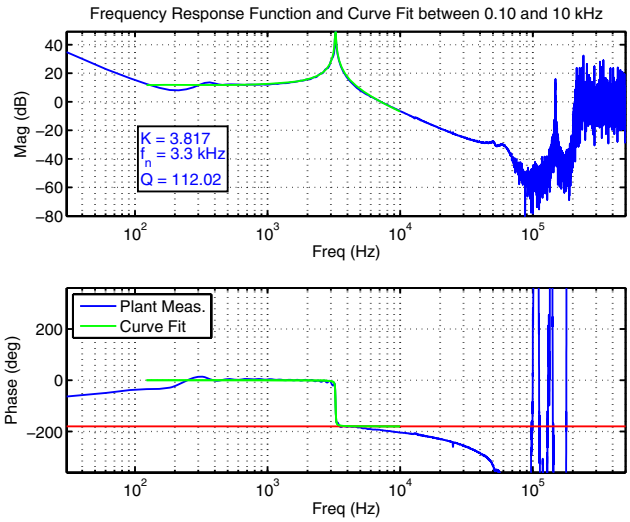

Fig. 15. Plant response and second-order curve fit model of an AFM actuator. From the curve fit, the parameters of a simple resonance can be immediately extracted: $K = 2.817$, $f_n = 3.3$ kHz, and $Q = \frac{1}{2\zeta} = 112.02$. Note that the resonance around 150 kHz is due to the AFM cantilever. It is notched with a separate notch that is not part of the PID design in [45].

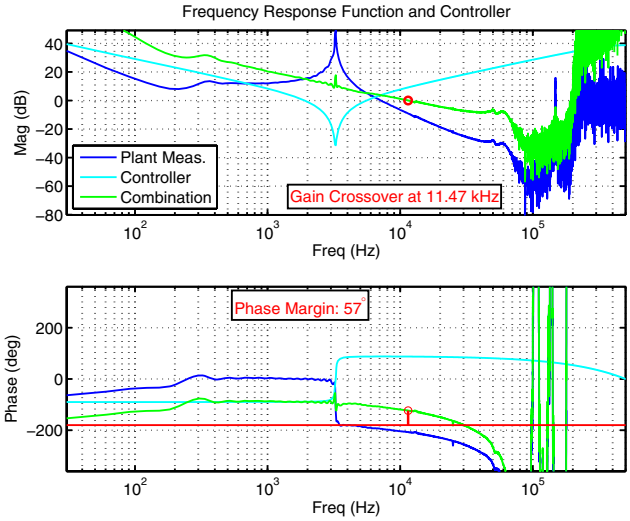

Fig. 16. PID controller generated from resonance/notch parameters and new open-loop response.

parameters were then used to design a second-order notch which was programmed as the numerator of the PID. With the terms of the numerator matched with the digital K_P, K_I, and K_D blocks the resulting filter acted to equalize out the resonance, making the open loop response look much like an integrator, as advocated in Section XII. At that time, a selection of the gain based on phase-margin and closed-loop peaking constraints resulted in the PID controller (cyan curve) and open-loop response (green curve) of Figure 16, and the projected closed-loop response of Figure 17.

This solution had all the elements needed to create a self tuning high performance mechatronic control loop, but they were all flawed. First of all the FRF measurement was done by computing FFTs of time domain data captured on the real-

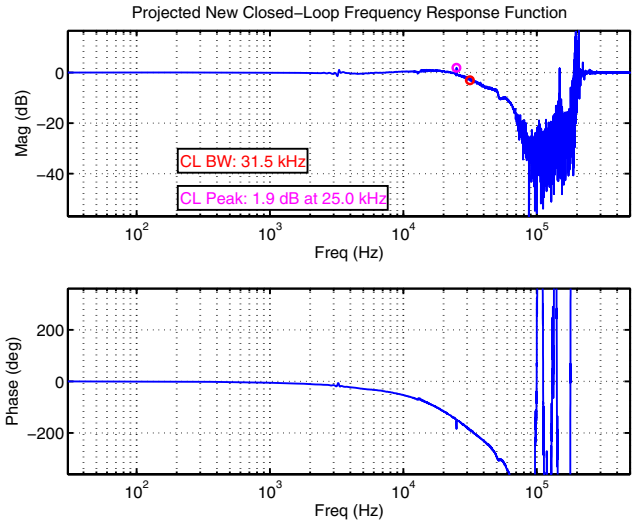

Fig. 17. Projected closed-loop response of AFM system with loop closed on the mass line.

time system. This accounts for the "hash" at high frequency seen in the measurements of Figures 15 and 17. To fix this, I would need to get back to a stepped-sine measurement, and the only way to connect this into the AFM controller was to build it in (Section XIV).

Second, although this actuator was a MEMS actuator which had only the main motor resonance and another flexible resonance, it was next to impossible to manufacture. Thus, we moved to a small piezo actuator with many resonances. A single notch would not cover them all, and stringing together a serial set of notches implemented as independent biquads gave too much computational latency. Thus, we needed a low latency filter that still maintained the numerical accuracy of biquads (Section XV).

Next, the problem of curve fitting had to be solved so as to match the FRF measurements to models that could then be used in compensator design (Section XVI). Finally, the automated loop shaping had to be expanded to take the "turn the open-loop into an integrator" philosophy and make it work for a much more complex system (Section XVII).

Only by solving each of these problems could a general open loop automatically be turned into something that looks like an integrator. The sections that follow will describe how each of these pieces of the puzzle was solved and put into place and what that means for overall mechatronic controller design.

XIV. Built In Stepped-Sine Measurements

A generic control system is shown in Figure 18. Identifying the many component blocks in the system involves injecting and extracting signals at multiple locations. Furthermore, the majority of access points are buried in the digital controller itself.

Using an instrument through analog test points involves not only creating those test points with circuitry, but also forcing many signals that were digital to be converted to analog signals before being measured with our external

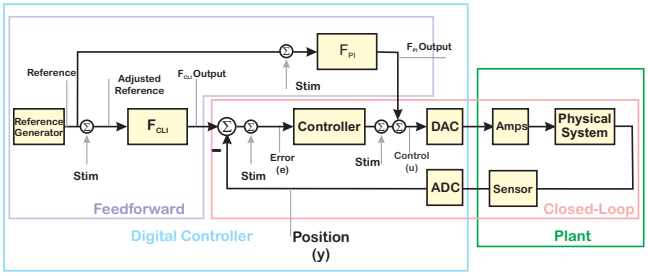

Fig. 18. Block diagram for controlled system showing stimulus points and response measurement points.

instrument. Using digital test points would be preferable, but as described in Section III, my original efforts to use an external digital instrument also failed in part because of the difficulty of connecting that instrument into the test system. In retrospect, it became clear that the FRF measurement system had to be built into the controller so that the digital connections could be made programatically, rather than with test pods.

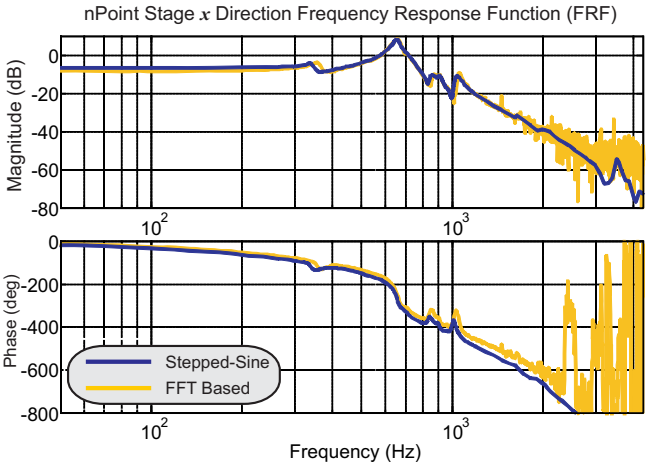

Fig. 19. A comparison of FFT and stepped-sine based FRFs on an nPoint NPXY100 [73] stage. (Courtesy: Jeff Butterworth.)

Likewise Fast Fourier Transform (FFT) based methods do not focus signals in any one feature area, instead relying on broadband excitation (pseudo-random, noise-like signals) or on a chirped sine signal. FFTs are computationally fast, but this speed comes at a price. Besides the lack of frequency isolation on the input, the frequency bins are fixed for a given sample rate and number of samples. For this reason, stepped-sine responses typically produce much "cleaner" measurements, especially at higher frequencies where the mechatronic system response is low as clearly seen in the example of Figure 19. This is described in much greater detail in [74], in which I describe a method of building the complete real-time stepped-sine integration into the digital controller as an extra FPGA block. An earlier comparison was done in [75], focusing on the effects of SNR.

Another method that is quite popular is the use of multi-sines [25] in which harmonically related sinusoids are used as the stimulus into the system. These have the advantage over

random and pseudo-random inputs in that they are periodic and so have no leakage. However, the analysis of signals stimulated by multisine inputs is still done using FFTs and this suffers in signal to noise immunity compared to the use of coherent demodulation associated with the stepped-sine. Furthermore, while the input signals are free from leakage, avoiding leakage in the integrated Fourier terms is difficult because while the frequencies of the input sines can be controlled, the sample frequency of the digital measurement system is usually fixed. This means that while one set of multisine frequencies might tile into the sample frequency of the digital system, the next set over most likely will not. The method of [74] adjusts the input sine frequencies to always be an integer divisor of the sample frequency so as to minimize integration artifacts.

Proponents of FFT based methods have noted that the former are much faster than stepped-sine. Certainly, the computation is much simpler, but in the 50 years since Cooley and Tukey first described the FFT [76], computational power has changed dramatically. We should be able to apply the massive computational advantage to save engineering time, by taking advantage of the improved SNR of stepped-sine. Certainly, it is understood to practitioners that while FRFs based on FFT methods often require large numbers of averages (e.g., 20–100 are not uncommon), the same system measured with stepped-sine methods will require far fewer passes (e.g., 3–10). Furthermore, the advantages shown in Figure 19 mean that far less engineering time is spent trying to guess at unmeasured high frequency dynamics.

One final reason to have built-in stepped-sine is that the method continues to work reasonably in the presence of nonlinearity. In fact, recognizing that stepped-sine produces a measurement of a nonlinear system's describing function [78] one can compare stepped-sine measurements to simulated stepped-sine measurements when the system exhibits a nonlinearity to help identify that nonlinearity, as shown in Figure 20 [77]. While it might be possible to adapt the method in [74] to be composed of multiple harmonically related sine waves inspired by [25], the method would need separate mixers and integrators for each harmonic frequency to preserve the signal quality. This is entirely doable if one wishes to devote the chip area on an FPGA, but makes autogain of the signal and identification of nonlinearities as discussed in [78] and [77] more difficult.

XV. Low Latency Filtering for Mechatronic Systems

It is well understood that the phase lag due to latency in a feedback loop erodes stability and performance characteristics. For a causal filter, the latency is in part determined by the filter length. That is, if a filter has N taps and a sample period of T_S, then the average latency through the filter will be $\frac{N}{2}T_S$. IIR filters are usually favored over FIR filters in feedback loops, as they can achieve similar bandwidth shaping with considerably smaller average delay.

The second source of latency is simply the time required to compute the filter equations between the time that a new

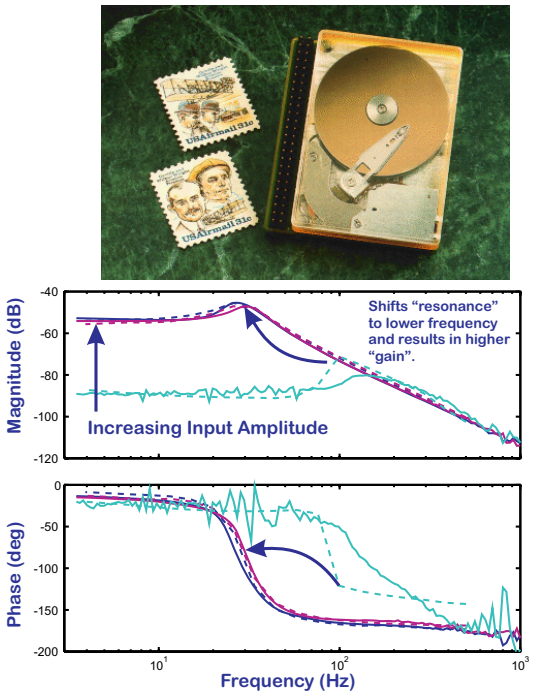

Fig. 20. The top picture shows HP's 1.3" KittyHawk disk drive, 1994 [77]. The bottom two plots show Bode plots of the response from the voice coil motor to the head. The solid curves are laboratory measurements made with the HP 3562A in stepped-sine mode. The dashed curves are simulated measurements made by simulating the stepped-sine algorithm in Simulink [78] with a model of the drive actuator that included nonlinear feedback elements. The different colors represent different input amplitudes. Notice that the frequency response functions change with varying input amplitude, a sure sign of nonlinearity.

sample comes into the filter and the filter produces its output. This delay is generally - but not necessarily - less than one sample period, but it is complicated by the fact that it can change with the number of taps in a filter. That is, a second-order filter obviously takes fewer computation steps than a tenth-order filter. This variable latency can cause unexpected problems with the control loop. A standard technique to minimize this variable latency is to compute everything that does not depend upon the most recent sample ahead of time in a precalculation [42], diagrammed in Figure 9. Once the most recent sample arrives, the last few calculations are performed and the filter output is produced. This has the benefit of not only minimizing the computational latency, but also of making it independent of filter length.

The structure of the multinotch allows it to minimize the computational latency using precalculation, while still preserving the numerical properties needed for finite word length arithmetic.

Restructuring the polynomial filter as a cascade of biquads improves both the physical intuition and the numerical properties of the filter. However, it made precalculation in order to minimize filter latency impossible until the multinotch reformulation described in [56].

$$\frac{Y(z)}{U(z)} = N_n(z)N_{n-1}(z)\cdots N_1(z)N_0(z) \qquad (21)$$

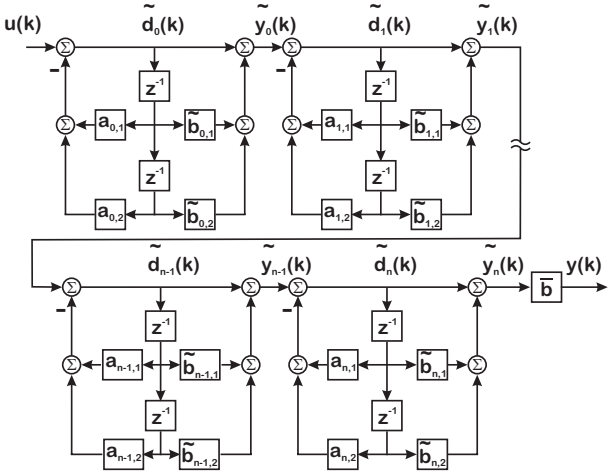

Fig. 21. The updated biquad cascade, with factored out b_0 terms.

$$= b_{n,0} \cdots b_{1,0} b_{0,0} \tilde{N}_n(z) \cdots \tilde{N}_1(z) \tilde{N}_0(z) \quad (22)$$

where

$$\tilde{N}_i(z) = \frac{1 + \tilde{b}_{i,1} z^{-1} + \tilde{b}_{i,2} z^{-2}}{1 + a_{i,1} z^{-1} + a_{i,2} z^{-2}} \quad (23)$$

$$\tilde{b}_{i,1} = \frac{b_{i,1}}{b_{i,0}}, \text{ and } \tilde{b}_{i,2} = \frac{b_{i,2}}{b_{i,0}}. \quad (24)$$

The direct feedthrough gains are concatenated together as:

$$\bar{b} = b_{n,0} b_{n-1,0} \cdots b_{1,0} b_{0,0}. \quad (25)$$

This simple change allows precalculation to be done on a biquad by biquad basis. with precalcs implemented as:

$$\tilde{d}_i(k) = \tilde{\text{prec}}_{i,1}(k) + \tilde{u}_i(k) \text{ and} \quad (26)$$

$$\tilde{y}_i(k) = \tilde{b}_{i,0} \tilde{d}_i(k) + \tilde{\text{prec}}_{i,2}(k) \text{ where} \quad (27)$$

$$\tilde{\text{prec}}_{i,1}(k) = -a_{i,1} \tilde{d}_i(k-1) - a_{i,2} \tilde{d}_i(k-2) \text{ and} \quad (28)$$

$$\tilde{\text{prec}}_{i,2}(k) = \tilde{b}_{i,1} \tilde{d}_i(k-1) + \tilde{b}_{i,2} \tilde{d}_i(k-2). \quad (29)$$

Finally,

$$\tilde{u}_0(k) = u(k), \quad (30)$$

$$\tilde{u}_i(k) = \tilde{y}_{i-1}(k) \text{ for } i = 1 \ldots n, \text{ and} \quad (31)$$

$$y(k) = \tilde{y}_n(k). \quad (32)$$

As described in detail in [56], this allows every downstream section to be precalculated using it's previous values and previous values of the upstream sections. The current input, $u(k)$, is simply added in to every section in parallel. In particular, $\tilde{d}_n(k)$ can be completed as soon as $u(k)$ is available with one multiply and one addition, allowing $\tilde{y}_n(k)$ to be completed with one more addition, and finally $y(k)$ with one more multiplication. Latency is low and does not increase with increasing filter length, but the numerical properties of the biquad are preserved.

At this point, precalculation has been restored and the numerical properties have been preserved, as demonstrated in Figure 22, with notch parameters described in Table I. However, some of these numerical properties can disappear when the sample frequency is significantly higher than the high Q dynamics being filtered. In this case, a simple switch to the same biquad structure but with Δ coefficients solves most of the issues [57].

Biquad #	$f_{N,n}$ (Hz)	Q_n	$f_{N,d}$ (Hz)	Q_d
1	200	10	400	10
2	1000	5	2000	5
3	10,000	10	20,000	10
4	8000	10	4000	10

TABLE I
FILTER PARAMETERS FOR MULTINOTCH EXAMPLE.

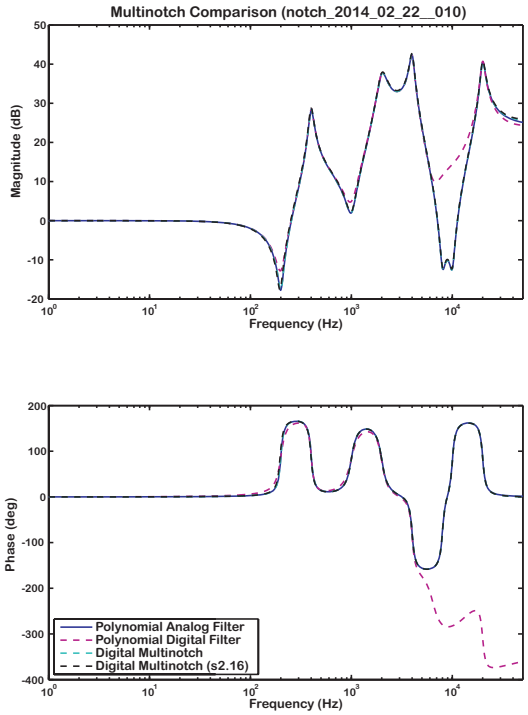

Fig. 22. A comparison of the quantized multinotch to unquantized filters in the example of Table I. Note that even without quantization, the discrete polynomial filter no longer matches the analog response. The multinotch, with even the coarsest quantization considered in these tests, s2.16, still matches the analog response.

XVI. Improved Curve Fitting for Mechatronic Systems

Working on optical drives (Section III), my desire was to move to MIMO models, which required model-based control, which required parametric models of the physical system. Extracting parametric models from the Frequency Response Functions (FRFs) produced by the CSA required curve fits, [15], [16], [17], [70], [71]. The CSA and DSA had curve fitting algorithms that worked well on measurements of analog circuits, but failed repeatedly on those of the drive mechanism. Instead of a second or fourth order model that physical intuition would have suggested, the models were of high order, and contained unstable poles and non-minimum phase zeros. A more mature version of myself would have worked to improve the measurements from the start, but it was sufficiently confusing for me at the time to realize

that I could not use the existing tools to get parametric models from which to work. While I should have recognized that discrete-time FRFs needed to be sanity checked against continuous time FRFs, the discrete-time representation of high Q systems had issues.

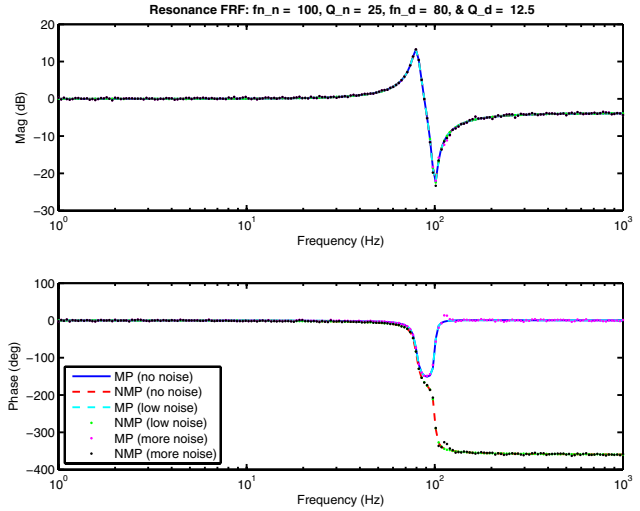

Fig. 23. FRF of biquad filter with $f_{n,n} = 100$ Hz, $Q_n = 25$, $f_{n,d} = 80$ Hz, $Q_d = 12.5$. Additive white Gaussian noise with $\sigma = \{0, 0.002, 0.02\}$ is added to the real and imaginary responses.

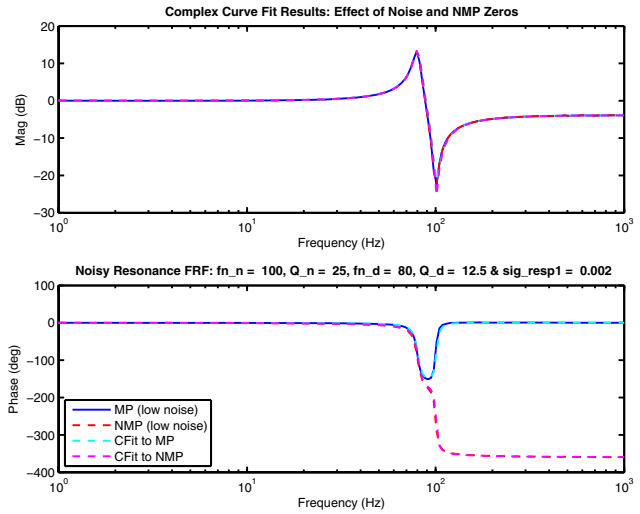

Fig. 24. Complex curve fit applied to simple resonance/anti-resonance with FRF noise $\sigma = 0.002$. At this point, the curve fit still seems to work, and can match both minimum phase (MP) and non-minimum phase (NMP) responses. (Cyan overlays blue, magenta overlays red.)

Consider the example in Figure 23. This is a fairly simple second order section where the FRF has been corrupted by adding various levels of additive white Gaussian noise to the complex response. Two versions of the dynamics are plotted, one with a complex pair of minimum phase zeros and a second where the zeros have been flipped over the $j\omega$ axis to make them non-minimum phase. A simple version of the complex curve fit is applied to this system, in that the order of the complex fit is limited to be second order. For very small

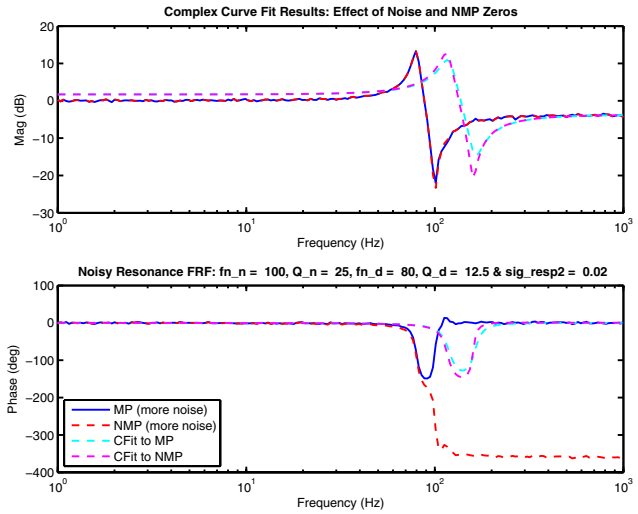

Fig. 25. Complex curve fit applied to simple resonance/anti-resonance with FRF noise $\sigma = 0.02$. Even with this low level of noise, the complex curve fit has failed badly.

amounts of noise in the FRF measurement, the complex curve fit still works, as shown in Figure 24. However, increasing the noise level slightly causes the complex fit to badly miss the parameter locations, as shown in Figure 25. Furthermore, the fit has missed the sense of the NMP zeros as well.

One possible explanation is as follows: One of the problems with curve fitting results from small bumps in the FRF magnitude that are not accompanied by matching phase variations. Thinking about Bode's gain-phase relationship [79], we realize that if the magnitude variation is not accompanied by phase variation, the only way for the curve fitter to explain it is by adding a pole-zero combination that result in a magnitude blip and a net 360 degree phase jump. Unfortunately, this is well suited to matching a high Q resonance with a high Q pair of NMP zeroes. This kind of issue with the linear fit makes the normal curve fit method largely unusable. The examples here are very simple. For responses with many resonance, anti-resonance features, it only gets worse.

One major source of discrete-time non-minimum phase zeros is the unwitting fitting of poles and zeros to pure time delay, as discussed in [72]. One of the ways to approximate delay is with a Padé approximation, and even at low order this maps time delay to NMP zeros. Accounting for the delay directly means that the compensator is only trying to account for the part of the system it can do something about.

So, even if the FRF measurement is good, that is, even if it is done using sine-dwell and has high coherence as described in Section XIV, we are a long way from a usable state-space model or a usable model for any sort of control design. At this point, the missing piece was the ability to extract reasonable models from good FRF measurements.

The situation in Figure 25 is telling, since even a simple biquad with fairly low levels of noise in the FRF measurement cause a pretty dramatic miss in the fit response. The

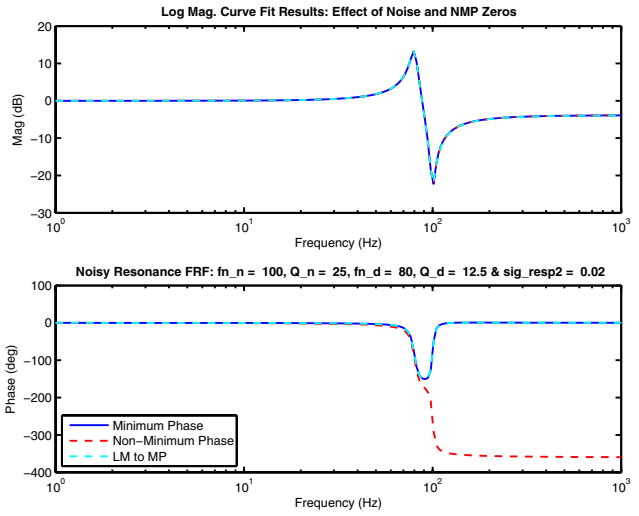

Fig. 26. Least squares fit assuming a biquad filter structure and using only log magnitude (LM) measurement data. The fit done with the higher level of noise, $\sigma = 0.02$, still matches the no noise response extremely well.

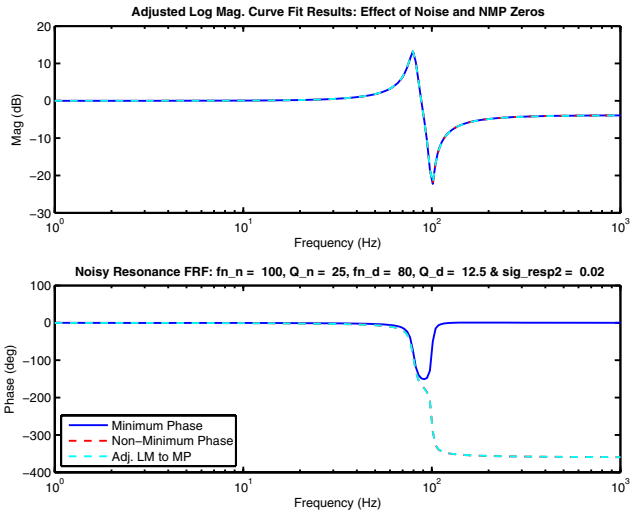

Fig. 27. Adjusted least squares fit to the LM curve, where the fit from Figure 26 is subsequently checked for phase jumps near dynamic features.

inspiration for a solution came in an old paper by Sidman et. al. [80] in which they suggested two fixes: to work with the log magnitude response (which would force an assumption of a minimum phase system, but would de-emphasize noise in the amplitude) and to do the fit by cycling through a series of fixed dynamic models to see which produced the lowest residual error. I guessed that if I could fit low-order dynamics and remove them from the response – something I called successive dynamic removal – then a series of low-order fits would result in an eventual fit to the entire response. I was asked to hand off my notes that detailed this strategy [81] (as well as the Matlab scripts for automatic PID tuning [45], [82] and multinotch parameterization [56], [57] to a new member of the Agilent Labs AFM team, Chris Moon. Using this approach, he found that if he used the multinotch of Section XV, the built-in stepped-sine of Section XIV, Sidman et. al.'s log magnitude fit, and Matlab's *lsqfit* routine, he could use successive dynamic removal to eventually turn the open-loop response into an integrator. The filter that was fit to do this was a combination of PID controller and multinotch, with parameters automatically arrived at by the algorithm. He made the assumption that resonances and anti-resonances would be interlaced and added scripts to roughly approximate these peaks and troughs to give a starting point for fitting individual sections [83].

Applying this method to our earlier example results in the very accurate match of the minimum phase dynamics as shown in Figure 26, which shows none of the issues from the complex fit. However, the assumptions of the fit are that the system is minimum phase. To get around this, the fit can be adjusted by a method that checks the phase residuals in the area of an identified feature (resonance, anti-resonance, or pair) and then makes an adjustment and checks the phase residuals again. An example of this is shown in Figure 27 and this method will be discussed in [84].

XVII. Loop Shaping

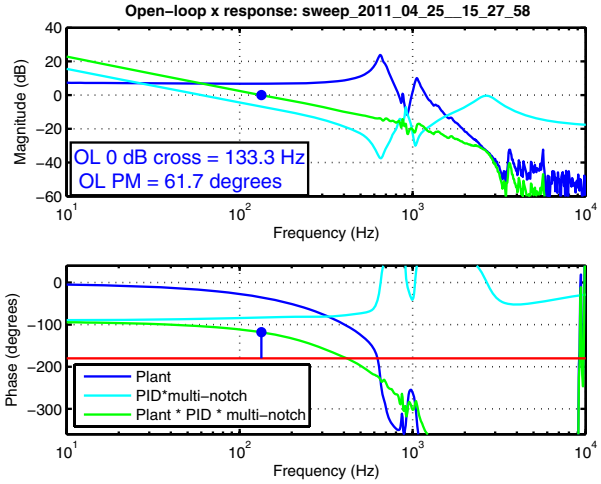

Fig. 28. Measured plant, compensator (PID+multinotch), and generated open-loop frequency response for AFM x-axis actuator. The PID and multinotch are automatically tuned to generate an open-loop response that looks like an integrator over the frequency range of interest, allowing the open-loop crossover to be set subject to phase-margin constraints.

As discussed in Section XII control of an integrator can easily be optimized by hand in closed-form, but on real systems which have to be shaped to resemble an integrator generally require evaluation by computer. We can use the integrator guidance to build software that searches real measured responses (and projected designs on those measured responses) to find our limits. This was shown on a second-order response in Section XIII. For a mechatronic system with many high Q resonances and anti-resonances, the combination of PID and multinotch, guided by the tuning described in Section XVI, yields results such as those shown in Figures 28 and 29. Note in Figure 28 that the open loop has indeed been shaped into the form of an integrator. The limiting factor for bandwidth is the phase margin requirement

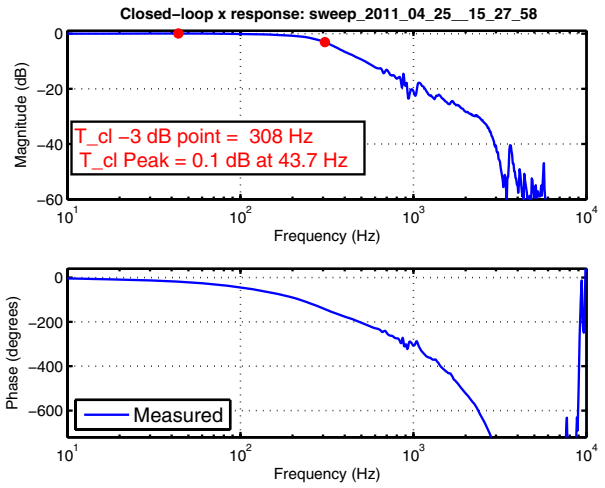

Fig. 29. The measured closed-loop response from the design in Figure 28. Note the nice, low-pass look of the closed-loop response and the minimal peaking. The closed-loop bandwidth can be adjusted from open-loop constraints or closed-loop peaking constraints.

of 60°. This does result in the closed-loop response of Figure 29 which has 308 Hz bandwidth and virtually no peaking.

Beating the open-loop response into an integrator is certainly not a unique concept, but a brilliant example of this is described in [85] for adjusting the dynamics of the room sized NASA Vertical Motion Simulator.

XVIII. Identifying and Limiting the Sources and Effects of Noise

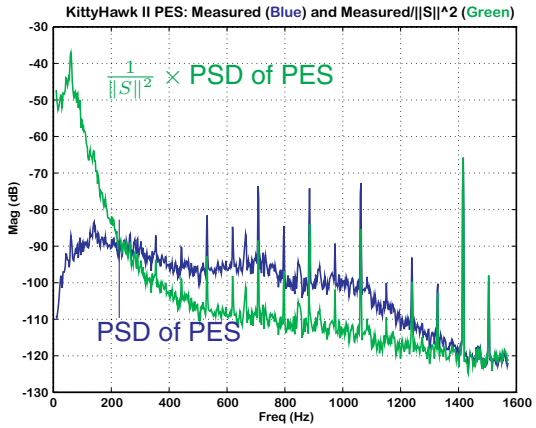

Fig. 30. PSD of PES, and PSD of PES filtered by $\frac{1}{\|S\|^2}$.

As discussed in Section XI doing high fidelity control on mechatronic systems involves equalizing out the parasitic resonances, limiting latency, and limiting noise. Why minimize noise at its source before it gets into the loop? As Gunter Stein's legendary 1989 Bode Lecture [86], [87] pointed out to us, Bode's Integral Theorem [88], [89] tells us that any noise that enters the feedback loop can not be eliminated (at least not by linear filtering) but only moved around.

For example, since disk drive actuators are essentially double integrators plus resonances, the Position Error Signal

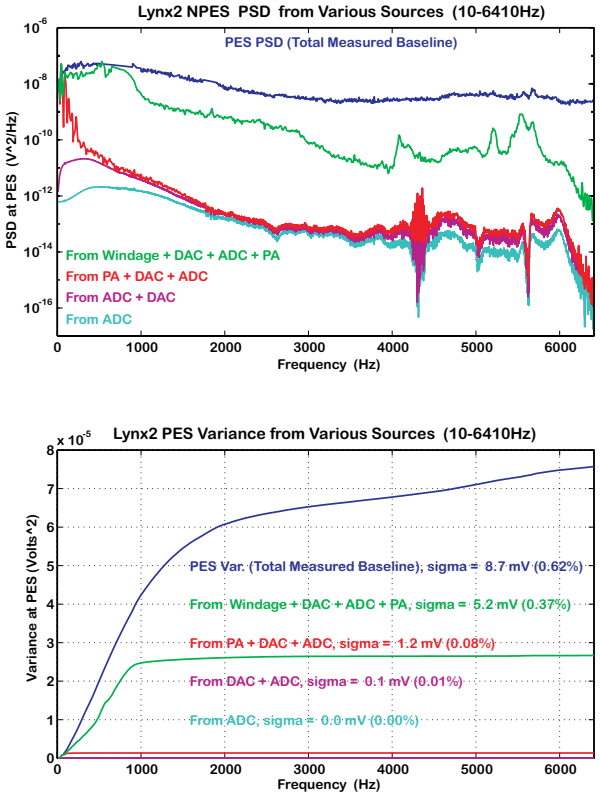

Fig. 31. Decomposition of baseline noise sources in a hard disk.

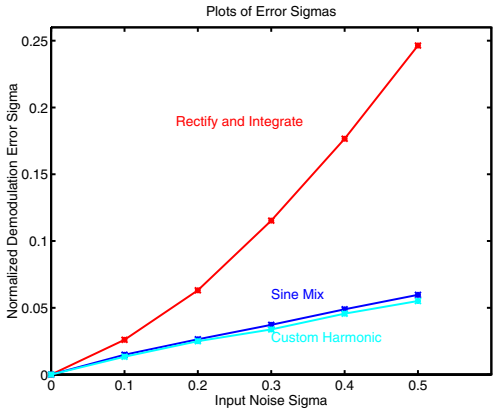

Fig. 32. Using coherent demodulation made up of specific harmonics of servo burst (modeled on HP Cougar I servo signals) dramatically diminishes effects of noise.

(PES) can only be measured in the presence of a feedback loop. The normal "flat" PES spectrum is a closed-loop signal and to get an "input noise" from measured PES, one needs to back filter the power spectral density (PSD) by $1/\|S\|^2$. Inspired by and recalling Gunter's talk 5 years later, I did the back filtering on the PES signal from a KittyHawk drive, and produced the green curve of Figure 30, which was shocking at the time. This drew reactions from the division servo engineers akin to, "That just ain't right." They would look at the plot, start to speak, stop, try to start again, and then shake their heads.

This one "not right" plot led to a systematic way of

evaluating many noise sources on disk drives, and building up the noise strata. This became known as the PES Pareto Method [64], [90], [91], [92], [93], [94], which we published after HP exited the disk drive business. The results, shown in Figure 31, surprising at the time, though not in retrospect, that Position Sensing Noise (PSN) (generated reading position from magnetic domains) and windage (air battering the actuator) were the main components affecting PES, directly led to work on improved airflow and position sensing methods, including coherent demodulation [65]. An illustrative result of this is shown in Figure 32, where coherent demodulation is far less susceptible to white noise than "rectify and integrate" methods. This work also led directly to high frequency wobbles [95], [96] and coherent AC mode demodulation for atomic force microscopes [13], [67], [68].

The common thread leading back to this discussion is that understanding the sources and system effects of noise in the loop and then finding detection methods that minimize these before they enter the loop should not be ceded to non-control engineers. It is the systems view, the realization that bandwidth cannot be pushed unless noise is attacked, that makes these efforts worthwhile. Attacking noise in our signal detection methods should be seen as a major key to high bandwidth control.

XIX. When You've Done Everything Else Right, You Can Adapt

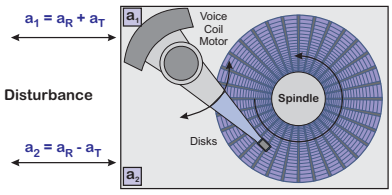

Fig. 33. Head disk assembly under effects of external acceleration. The individual accelerometers can be thought of as detecting the sum and difference of rotational acceleration, a_R, and translational acceleration, a_T. If the accelerometer gains are equal, then their difference gives $2a_R$.

The HP KittyHawk 1.3" disk drive was slated for mobile applications where shock and vibration would be an issue. While translational disturbances were assumed to be decoupled because of the balanced rotary actuator, rotational disturbances entered directly into the tracking loop. Furthermore, the sectored servo [9], [63] employed by hard disks limited the sensor bandwidth of the PES signal. However, there was no limit on accelerometer sample rates. This allowed for multi-rate feedforward cancellation using the accelerometer, building on the development of [97], [98].

At a poster session at the 1996 IFAC World Congress in San Francisco [19], [21], Matt White was presenting work on rejecting rotational disturbances in full size (5 1/4" at the time) disk drives [32], [99]. Matt's work was a careful study with multiple adaptive parameters. At some point when the session got quiet he turned to me and asked something to the effect of, "How the heck does your work look so simple?"

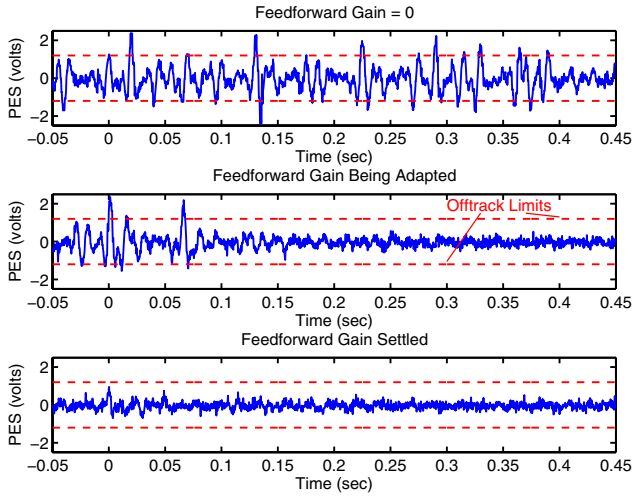

Fig. 34. Adaptive feedforward accelerometer compensation of rotary vibration on a KittyHawk disk drive. The top plot shows the effects of rotary disturbance with no feedforward. The red dashed lines are the offtrack limits. When adaptive feedforward is turned on in the middle plot, the effects of the disturbance are quickly minimized. In the lower plot, the drive stays within the offtrack limits, despite the external rotary disturbance.

As we talked, a couple of differences emerged: First, the smaller KittyHawk disk had simpler dynamics and I was able to formulate the problem into a single parameter adaptation of the unknown rotary accelerometer gain. This allowed a simple Least Mean Squares (LMS) algorithm that switched on only when there were sufficient levels of detected rotary disturbance for the algorithm to have guaranteed persistence of excitation [22] (Figure 34). The other difference was that I had programmed the BMW (Section III) to sample the accelerometer 4 times as fast as the PES. This simple exploitation of a physical feature of the system dramatically improved my rejection bandwidth.

Years later this "physically inspired approach" would pay off again when I was doing some consulting work on disk drives. Again the problem was rotary disturbance rejection, this time by differencing two inexpensive linear accelerometers that might not be balanced, as diagrammed in Figure 33. The mismatch between the two, which could be in the range of $\pm 15\%$ limited the benefits of such a feedforward system [100]. Eric Miller had built a shaker system that inadvertently shook the drive with both rotation and translation. He was chagrined by this, as the translational disturbances, a_T, showed up in PES – which should not have happened with a balanced actuator. The "Aha!" moment was when we realized that the only way that a_T showed up in PES was parasitically through mismatched accelerometers, and this provided the key to an augmented adaptation algorithm. In this, the input from translational disturbance, a_T, to PES was used to equalize the accelerometer gains while the input from rotational disturbance, a_R, to PES could then minimize the effects of rotational disturbances [101], as shown in the plots of Figure 35. *While these experiences are by no means exhaustive, they do point once again to the*

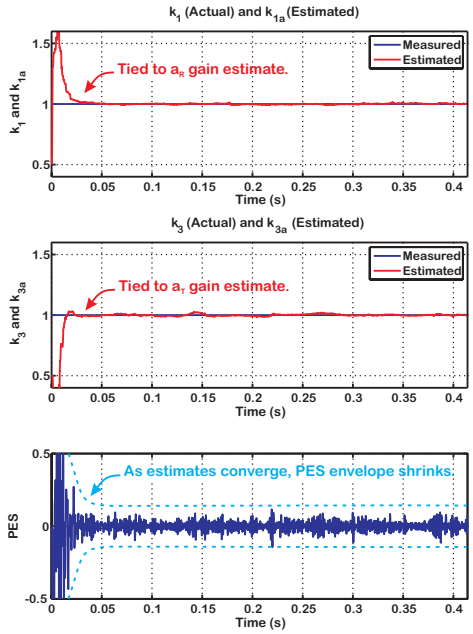

Fig. 35. Adaptation simulation using noise driven a_R and a_T, i.e., a_R and a_T are driven by noise only. The parameters adjust to minimize PES. Note the rapid convergence of k_{3a} to k_3 (balancing accelerometer gains) and k_{1a} to k_1 (adjusting feedforward gain from a_R). The effects of disturbance on PES rapidly goes away.

retention of physical parameters, this time for adaptation, as a means of dramatically simplifying control problems.

XX. State-Space Models for Mechatronic Systems

At this point, the reader might assume that I have avoided state space for much of my career, given all the problems that present themselves. However, for the Quintessential Phase (QP) Project Eric Johnstone needed to build a state estimator, essentially an augmented state Extended Kalman Filter to estimate turbulence in optical beam paths for interferometers and compensate it [102], [103]. The mechatronic wafer stage system would have a high number of high Q resonances as had shown up in AFM actuators, but the estimator demanded an accurate model. It turns out that the numerical accuracy of the multinotch provided an insight: What if the multinotch structure [56], [57] could be turned into a state-space structure with excellent numerical properties. The realization of that structure is the biquad state-space (BSS) structure [104] in discrete time, as described below.

Each of these biquad sections has a state-space realization, taking the form of:

$$\begin{bmatrix} x_{i,k+1} \\ x_{i,k} \end{bmatrix} \begin{bmatrix} -a_{i1} & -a_{i2} \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_{i,k} \\ x_{i,k-1} \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u_{i,k} \quad (33)$$

while the state output equation is given by:

$$\begin{bmatrix} \tilde{y}_{i,k+1} \end{bmatrix} = \begin{bmatrix} \tilde{b}_{i1} - a_{i1} & \tilde{b}_{i2} - a_{i2} \end{bmatrix} \begin{bmatrix} x_{i,k} \\ x_{i,k-1} \end{bmatrix} + \begin{bmatrix} 1 \end{bmatrix} u_{i,k} \quad (34)$$

The properly scaled output is generated via:

$$\begin{bmatrix} y_{i,k+1} \end{bmatrix} = \begin{bmatrix} b_{i0} \end{bmatrix} \begin{bmatrix} \tilde{y}_{i,k+1} \end{bmatrix}. \quad (35)$$

The indexing of $\tilde{y}_{i,k+1}$ and $y_{i,k+1}$ are a bit odd since we have direct feedthrough in our structure and $\tilde{y}_{i,k+1}$ depends on $x_{i,k+1}$ as well as $x_{i,k}$, $x_{i,k-1}$, and $u_{i,k}$. Thus, it's cleaner in what follows to call the biquad outputs, $\tilde{y}_{i,k+1}$ and $y_{i,k+1}$, respectively. We chain these together by noting that:

$$\begin{aligned} u_{i+1,k} &= \tilde{y}_{i,k+1}, && \text{for } 0 \le i < n, \\ u_{0,k} &= u_k, && \text{and} \\ \tilde{y}_{n,k+1} &= \tilde{y}_{k+1}. \end{aligned} \quad (36)$$

If one is willing to go through the algebraic pain and suffering of applying Equation 36 to each biquad structure, a very regular state-space structure results. For a 3-biquad model, we get the state equation of 37. The unscaled output is in Equation 38, both displayed in Figure 36 due to their size. Finally, the properly scaled outputs are generated via:

$$\begin{bmatrix} y_{2,k+1} \\ y_{1,k+1} \\ y_{0,k+1} \end{bmatrix} = \begin{bmatrix} b_{20}b_{10}b_{00} & 0 & 0 \\ 0 & b_{10}b_{00} & 0 \\ 0 & 0 & b_{00} \end{bmatrix} \begin{bmatrix} \tilde{y}_{2,k+1} \\ \tilde{y}_{1,k+1} \\ \tilde{y}_{0,k+1} \end{bmatrix}. \quad (39)$$

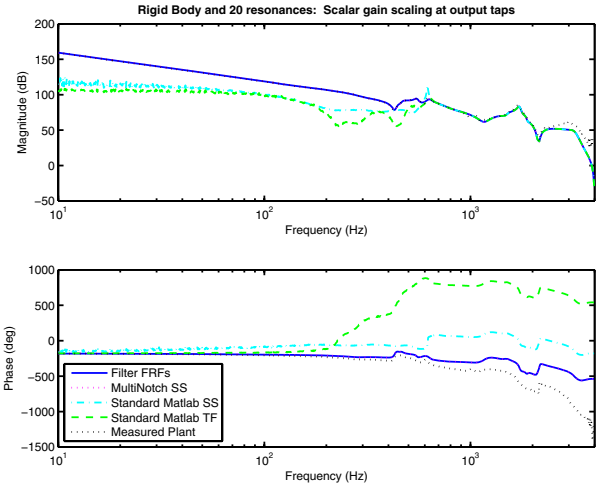

Fig. 37. Comparing state-space forms to Aerotech stage frequency response. Modeling the system with first 20 biquads and a rigid body, there is a massive difference between the conventional methods and the biquad state-space method.

The results in Figure 37 are pretty dramatic. There is another less obvious property of the BSS, which is that each biquad block has been discretized individually. This means that one could start with an analog version of the BSS, as described in [105], and discretize each biquad block separately. The result is that the biquad structure of the digital version is the same as the biquad structure of the analog version, although the internal behavior is different. This means that taken two at a time, the states of the digital version map directly back to the states of the analog version. This allows the designer to generate a physical model, transform this into an analog BSS model, discretize this, and relate the measurements made using the discrete model directly back to the physical model. I believe that this is unique for higher-order discrete-time models.

XXI. Crossing the Disconnect

Section XX and the results of [104], [105] show that after all the wandering, we are back to state space, but armed with

$$\begin{bmatrix} x_{2,k+1} \\ x_{2,k} \\ x_{1,k+1} \\ x_{1,k} \\ x_{0,k+1} \\ x_{0,k} \end{bmatrix} = \begin{bmatrix} -a_{21} & -a_{22} & \tilde{b}_{11}-a_{11} & \tilde{b}_{12}-a_{12} & \tilde{b}_{01}-a_{01} & \tilde{b}_{02}-a_{02} \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -a_{11} & -a_{12} & \tilde{b}_{01}-a_{01} & \tilde{b}_{02}-a_{02} \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -a_{01} & -a_{02} \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_{2,k} \\ x_{2,k-1} \\ x_{1,k} \\ x_{1,k-1} \\ x_{0,k} \\ x_{0,k-1} \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} u_k \quad (37)$$

$$\begin{bmatrix} \tilde{y}_{2,k+1} \\ \tilde{y}_{1,k+1} \\ \tilde{y}_{0,k+1} \end{bmatrix} = \begin{bmatrix} \tilde{b}_{21}-a_{21} & \tilde{b}_{22}-a_{22} & \tilde{b}_{11}-a_{11} & \tilde{b}_{12}-a_{12} & \tilde{b}_{01}-a_{01} & \tilde{b}_{02}-a_{02} \\ 0 & 0 & \tilde{b}_{11}-a_{11} & \tilde{b}_{12}-a_{12} & \tilde{b}_{01}-a_{01} & \tilde{b}_{02}-a_{02} \\ 0 & 0 & 0 & 0 & \tilde{b}_{01}-a_{01} & \tilde{b}_{02}-a_{02} \end{bmatrix} \begin{bmatrix} x_{2,k} \\ x_{2,k-1} \\ x_{1,k} \\ x_{1,k-1} \\ x_{0,k} \\ x_{0,k-1} \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} u_k \quad (38)$$

Fig. 36. State equations for discrete-time biquad state-space [104] with scalar output scaling. Note the regular and intuitive structure which is very similar to that of the analog version [105].

a new set of tools. The built-in stepped-sine and improved curve fit results have allowed us to identify the system model for high-Q plants in a mostly automated way. The "turn the open loop into an integrator" design criterion has provided – in conjunction with the structures to cleanly implement this in real-time – a fairly robust, high performance design. Finally, the translation of these measurement derived models into numerically robust state-space forms with intuitive mappings between continuous and discrete-time representations mean that real-time measurement and implementation of these controllers is completely reasonable.

The objectives of the measurement and computation tools are not that far different from those in [106] where the integration of identification [107] with a design framework for discrete-time control [108] is used to automate MIMO robust control design of mechatronic systems, using identification uncertainty to guide control design. The approaches both are pushing the strong link between the accuracy of the measurement derived model and the control performance asked of the design. Much of the difference is in the specific form of the controller filters. In fact, it should be clear that the identification presented here and the Biquad State-Space structure are perfectly applicable to H_2 optimal control design. Finding a link to robust control design such as H_∞ is a matter of finding a link between the highly structured model uncertainty that would show up in the BSS and the Δ uncertainty model used in H_∞.

XXII. Feedforward

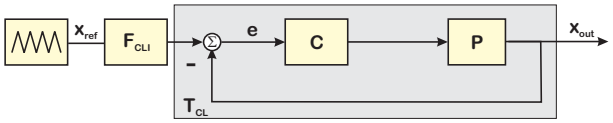

Fig. 38. Combined feedback-feedforward control using the F_{CLI} input.

One more section shows how these methods can be used to simplify, augment, and parallel model-based work. There is a fair body of work on feedforward control, including the Zero Phase Error Tracking Controller (ZPETC) of Tomizuka [109]

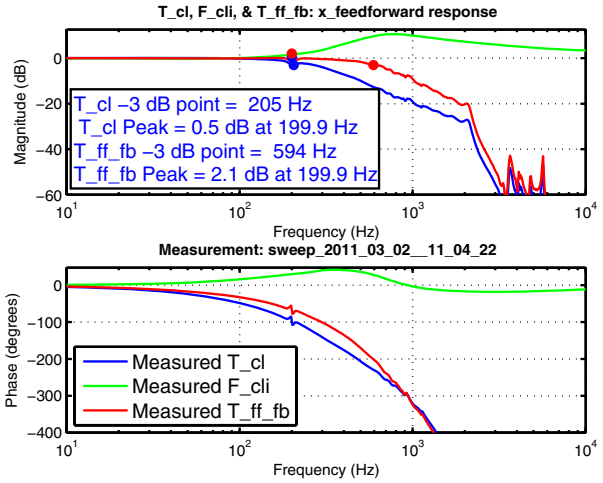

Fig. 39. Measured T_{CL}, F_{CLI}, and $T_{FF,FB}$ on nPoint NPXY100 stage. The x axis feedback controller was autotuned to produce the T_{CL} response. A new measurement was done, and from this a feedforward tune was done to generate F_{CLI}. The measurement was repeated to generate the combined $T_{FF,FB} = F_{CLI}T_{CL}$ response.

and the Zero Magnitude Error Tracking Controller described but not called such in [110], [40] and discussed in Rigney et al. [111]. Feedforward has also been a driving effort in the X-Y control of AFMs [112], [113], [114], [115]. The series of work on combined feedforward-feedback control for mechatronic systems such as X-Y positioners for atomic force microscopes [116], [117], [118], [119], [120], [121], [122], [123], [124] point very strongly to the advantages of using feedforward when the system is presented with a reference signal. These methods largely depend upon first generating an effective feedback controller based on a plant model and then designing a feedforward controller based on the closed-loop model generated from the plant and controller models. The work in [120] and [124] did not assume knowledge of the plant model, but only of the closed-loop response, as diagrammed in Figure 38. Still, as described in [118], [119] the objective was described as perfect tracking, in which $F_{CLI} = T_{CL}^{-1}$. While this might seem reasonable to

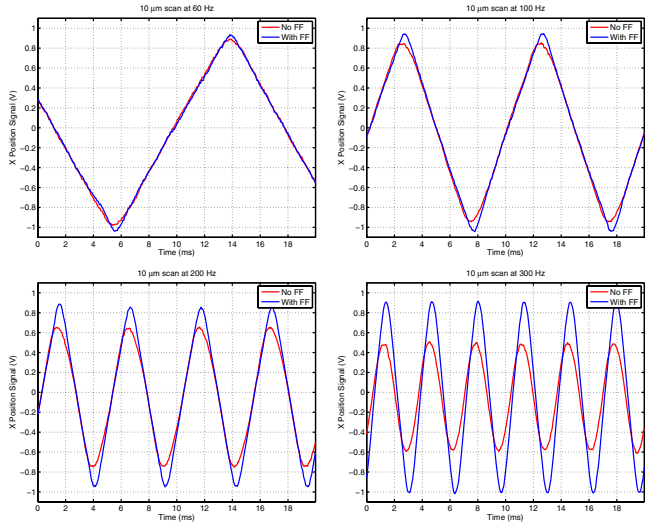

Fig. 40. Time response data measured using an Agilent Infiniium 54831M digital scope on the physical system shows the effect of the feedforward. From upper left to lower right, the reference input is a triangle wave of 60 Hz, 100 Hz, 200 Hz, and 300 Hz. At 60 Hz, both pure feedback and combined feedforward/feedback controllers can match the first and third harmonics constituting the triangle wave, but as the reference frequency goes up, only the combined controller can keep up. The combined controller can match the fundamental of the 300 Hz reference, but the third harmonic is about 10 dB down, leading to rounding of the response.

do if T_{CL} was minimum phase, most engineers would realize that this result requires infinite bandwidth from $F_{CLI}T_{CL}$, which would not only violate the Nyquist Criterion, but also cause the actuators to operate at high speed on amplified quantization noise. A more practical look at this yields a much simpler and more practical design method.

Realizing that the tools described earlier, particularly the built in stepped-sine of Section XIV, with a digital patch panel that allows us to make measurements from wherever we want in Figure 18, we can easily make a stepped-sine measurement of our closed-loop system. If we have adhered to the design approach of making the open loop look like an integrator and preserving 60° of phase margin, then our closed-loop FRF, $T_{CL}(f)$ is likely to have the response of a low-pass filter.

From here, we can measure T_{CL} and generate our desired transfer function shape (say a low-pass filter with more bandwidth). Our fitting routines can then adjust a multinotch to provide what ends up being a combination of lead-lag filters and notch/bump filters. An example of this is shown in Figure 39, where an nPoint NPXY100 stage was measured in the X axis. A feedback controller was generated using a combination of PID and multinotch filters as described in Section XVII. From there, the feedforward controller was generated as described above to almost triple the input-output bandwidth. Note that unlike the perfect tracking filter, we do not try to for infinite bandwidth. We can see this in the plots of Figures 40, where the improvement of reference tracking is very clear beyond the bandwidth of T_{CL}. The use of a double lead also means that the requested increased bandwidth can be limited to something reasonable for the physical system to achieve. Returning to our model-based approach, we see that this is equivalent to asking for $F_{CLI}T_{CL}$ to have a shape of a new low-pass filter with approximately triple the reference to position bandwidth of T_{CL} alone.

XXIII. Conclusions and Future Work

If there is anything that I have learned on this journey it is that lack of analysis and optimality doesn't stop most people from working on "control systems." In fact, the explosion of cheap sensors and actuators, as well as inexpensive computation platforms such as the Raspberry Pi and the MicroZed mean that we have witnessed only the tip of the iceberg. Guidance from the control community would be welcomed, but it has to be in clear, physically intuitive terms which are extensions of what they are doing now, not wholesale replacement.

I have never been a fan of the KISS (Keep It Simple, Stupid) acronym because it implied the inability to do complex things, either from the speaker or the listener. It also implies that one or both lack intelligence. However, it is my belief that we often make complex things overly intimidating and even the brightest of us are keenly aware of the number of times we have done boneheaded things. Hence, I would like to propose the KICK acronym, which stands for Keep It Clear, Knucklehead. We need to keep our explanations clear and physical, as the folks listening are our customers. And we need to make sure that the advanced methods provide at least as much bang for the computational buck as the simple methods.

That being said, methods that kept working in practice did so for one or more fundamental reasons. The advanced control background that caused me to look for those underlying factors revealed a lot that, in retrospect, seems quite intuitive. Furthermore, if there is one absolute take away from this work, it is that putting in the work up front to make high fidelity measurements something that is repeatable and easy, whether in the design stage or in the operation of the system, pays dividends far beyond many of our most sophisticated optimization methods. A long time ago, Gene Franklin quipped to me, "Well, you can only control as well as you can measure." Likewise, you can only model what you can accurately measure.

There are tremendous benefits of model-based methods. They promise a close tie to the physical dynamic equations, something that has been restored with the connection between analog and digital BSS models. They appeared to give a systematic way to handle MIMO systems. It seems, though that the intuition preserving approach may yield benefits there. Measuring a 2×2 mechatronic system still requires input-output FRFs of the four SISO systems and only after those individual systems have been curve fit, can one really talk about combining them into a more compact model. In such systems, how close do dynamics have to be to be considered common? By what metric will these be evaluated? If the BSS preserves the model precision of a SISO system, how do we choose between close biquad pairs to reduce the

model of our MIMO system? These seem like worthwhile directions to pursue. The goal is not to ignore model-based or optimization methods, but to provide a rapprochement between them and the physical intuition of classical methods. The goal is not to ignore practicing engineers and hobbyists for dealing with a too trivial set of mathematics, but to have a set of tools that starts with their intuitive understanding and can be iteratively improved to take care of more and more dynamic features. We might call this approach *Optimization Inspired Classical Control*.

American football coaching legend Lou Holtz was once asked if the small town he was working in was the end of the world. His response was, "No, but you can see it from here." Are we at the point where a few button pushes lead to measurement based, mathematically excellent designs? No, but we can see it from here. This understanding is merely a starting point. There is still much work to do on tying these types of methods into MIMO work. I hope to flesh out the curve fitting methodology that works so in Section XVI with the ability to detect NMP zeros [84]. Likewise, understanding more of the trade-offs between Δ coefficients and the δ transform is a key piece in automatically generating "safe to implement" filter designs for use in high speed digital hardware, which is often fixed point [125]. Finally, how do we generalize adaptation so that it is on physically recognizable quantities? Redefining advanced methods in such a way as to make use of the more intuitive structures used by long time practicing engineers not only garners more buy in from those engineers, but seems to dramatically improve the ability of those methods to be safely implemented.

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] R. C. Blackham, J. A. Vasil, E. S. Atkinson, and R. W. Potter, "Measurement modes and digital demodulation for a low-frequency analyzer," *Hewlett-Packard Journal*, vol. 38, pp. 17–25, January 1987.

[3] G. Verbinski, *Pirates of the Caribbean: The Curse of the Black Pearl*. Walt Disney Pictures, 2003.

[4] S. Herek, *Bill & Ted's Excellent Adventure*. De Laurentiis Entertainment Group (DEG), 1989.

[5] D. Y. Abramovitch, "Analysis and design of a third order phase-lock loop," in *Proceedings of the 1988 IEEE Military Communications Conference (MILCOM)*, (San Diego, CA), pp. 455–459, IEEE, IEEE, October 1988.

[6] D. Y. Abramovitch, "Lyapunov Redesign of analog phase-lock loops," *The IEEE Transactions on Communication*, vol. 38, pp. 2197–2202, December 1990.

[7] J. Hagopian, "Data storage apparatus," U.S. Patent 3,007,144, IBM, October 31 1961.

[8] J. Porter. Personal correspondence, July 20, 2000.

[9] 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.

[10] D. Y. Abramovitch, "Magnetic and optical disk control: Parallels and contrasts," on-line full draft of paper, Agilent Labs, Palo Alto, CA 94304, June 2001. http://www.labs.agilent.com/personal/-Danny_Abramovitch/pubs/hd_vs_od_servo.pdf.

[11] T. F. Library, "Integrated systems' $matrix_x$ family extends its rapid prototyping and real-time product line; realsim AC-100 model c40 with PC front end adds a single-box, low-cost, high-performance desktop solution," 1995. [Online; accessed May 24, 2015].

[12] D. Abramovitch, "The Banshee Multivariable Workstation: A tool for disk drive servo research," in *Proceedings of the ASME Winter Annual Meeting*, (Anaheim, CA), ASME, ASME, November 1992.

[13] D. Y. Abramovitch, "Coherent demodulation with reduced latency adapted for use in scanning probe microscopes," United States Patent 7,843,627, Agilent Technologies, Santa Clara, CA USA, November 30 2010.

[14] Hewlett-Packard, *HP 3563A Control Systems Analyzer*, 1990.

[15] Hewlett-Packard, *z-Domain Curve Fitting in the HP 3563A Analyzer*, hp 3563a-1 product note ed., 1989.

[16] Hewlett-Packard, *Control System Development Using Dynamic Signal Analyzers: Application Note 243-2*, 1984.

[17] Hewlett-Packard, *Curve Fitting in the HP 3562A*, product note hp 3562a-3 ed., 1989.

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

[19] D. Abramovitch, "Rejecting rotational disturbances on small disk drives using rotational accelerometers," in *Proceedings of the 1996 IFAC World Congress*, (San Francisco, CA), pp. 483–488 (Volume O), IFAC, IEEE, July 1996.

[20] D. Abramovitch, "Rejecting rotational disturbances on small disk drives using rotational accelerometers," *Control Engineering Practice*, vol. 5, November 1997.

[21] D. Y. Abramovitch, "Rejection of disturbances on a disk drive by use of an accelerometer," United States Patent 5,663,847, Hewlett-Packard, Palo Alto, CA USA, September 2 1997.

[22] L. Ljung, *System Identification: Theory for the User*. Prentice-Hall Information and System Sciences Series, Englewood Cliffs, New Jersey 07632: Prentice-Hall, 1987.

[23] L. Ljung and T. Söderström, *Theory and Practice of Recursive Identification*. MIT Press Series in Signal Processing, Optimization, and Control, Cambridge, MA 02142: MIT Press, 1983.

[24] G. C. Goodwin and K. S. Sin, *Adaptive Filtering Prediction and Control*. Information and Systems Science Series, Englewood Cliffs, N.J. 07632: Prentice-Hall, 1984.

[25] R. Pintelon and J. Schoukens, *System Identification: A Frequency Domain Approach*. Piscataway, NJ: IEEE Press, first ed., 2001.

[26] L. Ljung and T. Glad, *Modeling of Dynamic Systems*. Upper Saddle River, NJ: Prentice Hall, 1994.

[27] J.-N. Juang, *Applied System Identification*. Upper Saddle River, NJ: Prentice Hall, 1994.

[28] J. S. Bendat and A. G. Piersol, *Random Data: Analysis and Measurement Procedures*. Wiley Series on Probability and Statistics, New York, NY: John Wiley & Sons, third ed., 2000.

[29] J. S. Bendat and A. G. Piersol, *Engineering Applications of Correlation and Spectral Analysis*. New York, NY: John Wiley & Sons, second ed., 1993.

[30] B. D. O. Anderson, R. R. Bitmead, J. C. R. Johnson, P. V. Kokotovic, R. L. Kosut, I. M. Y. Mareels, L. Praly, and B. D. Riedle, *Stability of Adaptive Systems: Passivity and Averaging Analysis*. Cambridge, MA and London, England: MIT Press, 1986.

[31] P. C. Parks, "Liapunov redesign of model reference adaptive control systems," *IEEE Trans. on Automatic Control*, vol. AC-11, July 1966.

[32] M. White and M. Tomizuka, "Increased disturbance rejection in magnetic disk drives by acceleration feedforward control," *Control Engineering Practice*, vol. 5, no. 6, pp. 741–751, 1997.

[33] K. J. Åström and B. Wittenmark, "Practical issues in the implementation of self-tuning control," *Automatica*, vol. 20, pp. 595–605, 1984.

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

[35] R. E. Best, *Phase-Locked Loops: Design, Simulation, and Applications*. New York: McGraw-Hill, fourth ed., 1999.

[36] D. H. Wolaver, *Phase-Locked Loop Circuit Design*. Advanced Reference Series & Biophysics and Bioengineering Series, Englewood Cliffs, New Jersey 07632: Prentice Hall, 1991.

[37] F. M. Gardner, *Phaselock Techniques*. New York, NY: John Wiley & Sons, second ed., 1979. ISBN 0-471-04294-3.

[38] D. Abramovitch, "Lyapunov Redesign of classical digital phase-lock loops," in *Proceedings of the 2003 American Control Conference*, (Denver, CO), pp. 2401–2406, AACC, IEEE, June 2003.

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

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

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

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

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

[44] D. Y. Abramovitch, "A unified framework for analog and digital PID controllers," in *Prodeedings of the 2015 Multi-Conference on Systems and Control*, (Sydney, Australia), IEEE, IEEE, September 2015.

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

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

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

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

[49] T. Williams, "Fuzzy Logic is anything but fuzzy," *Computer Design*, pp. 113–127, April 1992.

[50] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing*. Englewood Cliffs, N. J.: Prentice Hall, 1975.

[51] P. Horowitz and W. Hill, *The Art of Electronics*. Cambridge University Press, first ed., October 31 1980.

[52] Texas Instruments, *TMS320C4x User's Guide*, 1993.

[53] T. Kailath, *Linear Systems*. Englewood Cliffs, N.J. 07632: Prentice-Hall, 1980.

[54] J. O. Smith, *Introduction to Digital Filters with Audio Applications*. http://www.w3k.org/books/: W3K Publishing, 2007.

[55] P. M. Embree, *C Algorithms for Real-Time DSP*. Upper Saddle River, NJ 07458: Prentice Hall PTR, 1995.

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

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

[58] O. Mayr, *Authority, Liberty & Automatic Machinery in Early Modern Europe*. Baltimore and London: The Johns Hopkins University Press, 1986.

[59] A. V. Roup and D. S. Bernstein, "On the dynamics of the escapement mechanism of a mechanical clock," in *Proceedings of the 38th IEEE Conference on Decision and Control.*, vol. 3, (Phoenix, Az), pp. 2599–2604, IEEE, IEEE, December 7-10 1999.

[60] A. V. Roup, D. S. Bernstein, S. G. Nersesov, W. M. Haddad, and V. Chellaboina, "Limit cycle analysis of the verge and foliot clock escapement using impulsive differential equations and Poincaré maps," *International Journal of Control*, vol. 76, no. 17, pp. 1685–1698, 2003.

[61] J. Wagner, C. Huey, K. Knaub, E. Volk, and A. Jagarwal, "Modeling and analysis of a weight driven mechanical tower clock," in *Proceedings of the 2010 American Control Conference*, (Baltimore, MD), pp. 634–639, AACC, IEEE, June 2010.

[62] 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.

[63] D. Y. Abramovitch, "Magnetic and optical disk control: Parallels and contrasts," in *Proceedings of the 2001 American Control Conference*, (Arlington, VA), pp. 421–428, AACC, IEEE, June 2001.

[64] 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.

[65] D. Abramovitch, "Customizable coherent servo demodulation for disk drives," *IEEE/ASME Transactions on Mechatronics*, vol. 3, pp. 184–193, September 1998.

[66] D. Y. Abramovitch, "Disk drive servo demodulation system which supresses noise on the position error signal," United States Patent 5,801,895, Hewlett-Packard, Palo Alto, CA USA, September 1 1998.

[67] D. Y. Abramovitch, "Low latency demodulation for atomic force microscopes, Part I: Efficient real-time integration," in *Proceedings of the 2011 American Control Conference*, (San Francisco, CA), AACC, IEEE, June 29–July 1 2011.

[68] D. Y. Abramovitch, "Low latency demodulation for atomic force microscopes, Part II: Efficient calculation of magnitude and phase," in *Proceedings of the IFAC 18th World Congress*, (Milan, Italy), IFAC, IFAC, August 28–September 2 2011.

[69] D. Lancaster, *Active Filter Cookbook*. Newnes, second ed., 1996.

[70] E. Levy, "Complex-curve fitting," *IRE Transactions on Automatic Control*, vol. AC-4, pp. 37–43, 1959.

[71] J. L. Adcock, "Curve fitter for pole-zero analysis," *Hewlett-Packard Journal*, vol. 38, pp. 33–37, January 1987.

[72] J. A. Butterworth, L. Y. Pao, and D. Y. Abramovitch, "Fitting discrete-time models to frequency responses for systems with transport delay," in *ASME 2011 International Mechanical Engineering Congress and Exposition*, ASME, ASME, 2011.

[73] nPoint, "Multi-axis stages," 2015. [Online; accessed February 5, 2015].

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

[75] J. Schoukens, R. M. Pintelon, and Y. J. Rolain, "Broadband versus stepped sine FRF measurements," *IEEE Transactions on Instrumentation and Measurement*, vol. 49, pp. 275–278, April 2000.

[76] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Math. Comput.*, vol. 19, pp. 297–301, April 1965. Reprinted in *Digital Signal Processing*, ed. L. R. Rabiner and C. M. Rader, pp. 223-227, New York: IEEE Press, 1972.

[77] D. Abramovitch, F. Wang, and G. Franklin, "Disk drive pivot nonlinearity modeling Part I: Frequency Domain," in *Proceedings of the 1994 American Control Conference*, (Baltimore, MD), pp. 2600–2603, AACC, IEEE, June 1994.

[78] F. Wang, D. Abramovitch, and G. Franklin, "A method for verifying measurements and models of linear and nonlinear systems," in *Proceedings of the 1993 American Control Conference*, (San Francisco, CA), pp. 93–97, AACC, IEEE, June 1993.

[79] H. Bode, "Relations between attenuation and phase in feedback amplifier design," *Bell System Technical Journal*, vol. 19, pp. 421–454, 1940.

[80] M. D. Sidman, F. E. DeAngelis, and G. C. Verghese, "Parametric system identification on logarithmic frequency response data," *IEEE Transactions on Automatic Control*, vol. 36, pp. 1065–1070, September 1991.

[81] D. Y. Abramovitch, "Task list for fully automated PID tuning," internal report/memo, Agilent Laboratories, September 21 2007.

[82] D. Y. Abramovitch, S. T. Hoen, and R. K. Workman, "Automatic generation of PID parameters for a scanning probe microscope," United States Patent 7,987,006, Agilent Technolgies, Inc., Santa Clara, CA USA, July 26 2011.

[83] D. Y. Abramovitch and C. R. Moon, "Cascaded digital filters with reduced latency," International Application Published Under the Patent Cooperation Treaty WO 2012/118483, World Intellectual Property Organization, September 9 2012.

[84] D. Y. Abramovitch, "Curve fits on high-Q mechatronic systems in the presence of noise," in *To be submitted to the 2016 American Control Conference*, (Boston, MA), AACC, IEEE, July 2016.

[85] R. A. Mueller, "Optimizing the performance of the pilot control loaders at the NASA vertical motion simulator," *American Institute of Aeronautics and Astronautics (AIAA) Journal of Aircraft*, vol. 47, pp. 682–693, March-April 2010.

[86] G. Stein, "Respect the unstable." Bode Lecture presented at the 1989

IEEE Conference on Decision and Control, Tampa FL, December 1989.

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

[88] H. W. Bode, *Network Analysis and Feedback Amplifier Design*. Bell Telephone Laboratories Series, New York: D. Van Nostrand Company, first ed., 1945.

[89] C. Mohtadi, "Bode's integral theorem for discrete-time systems," *Proceedings of the IEE*, vol. 137, pp. 57–66, March 1990.

[90] 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.

[91] 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.

[92] 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.

[93] 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," in *Digests of The Magnetic Recording Conference 1997*, (Minneapolis, MN), IEEE Magnetics Society, IEEE, September 1997.

[94] D. Y. Abramovitch, T. N. Hurst, and R. H. Henze, "Method and apparatus for decomposing drive error signal noise sources," United States Patent 5,909,661, Hewlett-Packard, Palo Alto, CA USA, June 1 1999.

[95] D. Y. Abramovitch, "Turning the tracking problem sideways: Servo tricks for DVD+RW clock generation," in *Proceedings of the 2000 American Control Conference*, (Chicago, IL), pp. 2615–2620, AACC, IEEE, June 2000.

[96] D. Y. Abramovitch and D. K. Towner, "A re-writable optical disk having reference clock information permanently formed on the disk," United States Patent 6,046,968, Hewlett-Packard Co., Palo Alto, CA USA, April 4 2000.

[97] D. B. Davies and M. D. Sidman, "Active compensation of shock, vibration, and wind-up in disk drives," in *Advances in Information Storage Systems, Vol. 5* (B. Bhushan, ed.), pp. 5–20, New York, NY: ASME Press, 1993.

[98] V. L. Knowles and D. M. Hanks, "Shock and vibration disturbance compensation system for disc drives," European Patent Application 871065555.3, Hewlett-Packard Co., Corporate Patent Department, M/S 20B-O, 3000 Hanover Street, Palo Alto, CA 94304 USA, June 1987.

[99] M. White and M. Tomizuka, "Increased disturbance rejection in magnetic disk drives by acceleration feedforward control," in *Proceedings of the 1996 IFAC World Congress*, (San Francisco, CA), pp. 489–494 (Volume O), IFAC, IEEE, July 1996.

[100] A. Jinzenji, T. Sasamoto, K. Aikawa, S. Yoshida, and K. Aruga, "Acceleration feedforward control against rotational disturbance in hard disk drives," *IEEE Transactions on Magnetics*, vol. 37, pp. 888–893, March 2001.

[101] D. Y. Abramovitch and G. Hsu, "Mitigating the effects of disturbances of a disk drive," in *Proceedings of the 2015 Multi-Conference on Systems and Control*, (Sydney, Australia), IEEE, IEEE, September 21-23 2015.

[102] E. Johnstone and D. Y. Abramovitch, "Quintessential Phase: A method of mitigating turbulence effects in interferometer measurements of precision motion," in *Proceedings of the 2013 American Control Conference*, (Washington, DC), AACC, IEEE, June 17–19 2013.

[103] R. Loughridge and D. Y. Abramovitch, "A tutorial on laser interferometry for precision measurements," in *Proceedings of the 2013 American Control Conference*, (Washington, DC), AACC, IEEE, June 17–19 2013.

[104] 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.

[105] 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.

[106] T. Oomen, R. van Herpen, S. Quist, M. van de Wal, O. Bosgra, and M. Steinbuch, "Connecting system identification and robust control for next-generation motion control of a wafer stage," *IEEE Control Systems Magazine*, vol. 22, pp. 102–118, January 2014.

[107] T. Oomen and O. Bosgra, "System identification for achieving robust performance," *Automatica*, vol. 48, pp. 1975–1987, June 2012.

[108] T. Oomen, M. van de Wal, and O. Bosgra, "Design framework for high-performance optimal sampled-data control with application to a wafer stage," *International Journal of Control*, vol. 80, pp. 919–934, June 2007.

[109] M. Tomizuka, "Zero phase error tracking algorithm for digital control," *ASME Journal of Dynamic Systems, Measurement, and Control*, March 1987.

[110] J. Wen and B. Potsaid, "An experimental study of a high performance motion control system," in *Proceedings of the 2004 American Control Conference*, (Boston, MA), pp. 5158–5163, AACC, IEEE, June 2004.

[111] B. P. Rigney, L. Y. Pao, and D. A. Lawrence, "Nonminimum phase dynamic inversion for settle time applications," *IEEE Transactions on Control Systems Technology*, vol. 17, no. 5, pp. 989–1005, 2009.

[112] D. Croft, G. Shed, and S. Devasia, "Creep, hysteresis, and vibration compensation for piezoactuators: Atomic force microscopy application," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 128, pp. 35–43, March 2001.

[113] D. Croft and S. Devasia, "Vibration compensation for high speed scanning tunneling microscopy," *Review of Scientific Instruments*, vol. 70, pp. 4600–4605, December 1999.

[114] K. K. Leang, Q. Zou, and S. Devasia, "Feedforward control of piezoactuators in atomic force microscope systems: Inversion-based compensation for dynamics and hysteresis," *IEEE Control Systems Magazine*, vol. 19, pp. 70–82, 2009.

[115] K. K. Leang and S. Devasia, "Feedback-linearized inverse feedforward for creep, hysteresis, and vibration compensation in AFM piezoactuators," *IEEE Transactions on Control Systems Technology*, vol. 15, no. 5, pp. 927–935, 2007.

[116] J. A. Butterworth, L. Y. Pao, and D. Y. Abramovitch, "Combined feedforward/feedback control of atomic force microscopes," in *Proceedings of the American Control Conference*, (New York, NY), AACC, IEEE, June 2007.

[117] J. A. Butterworth, L. Y. Pao, and D. Y. Abramovitch, "The effect of nonminimum-phase zero locations on the performance of feedforward model-inverse control techniques in discrete-time systems," in *Proceedings of the American Control Conference*, (Seattle, WA), AACC, IEEE, June 2008.

[118] J. A. Butterworth, L. Y. Pao, and D. Y. Abramovitch, "A comparison of control architectures for atomic force microscopes," in *Proceedings of the 2008 IFAC Triennial World Congress*, (Seoul, Korea), AACC, IEEE, July 2008.

[119] J. A. Butterworth, L. Y. Pao, and D. Y. Abramovitch, "A comparison of control architectures for atomic force microscopes," *Asian Journal of Control*, vol. 11, pp. 175–181, March 2009.

[120] J. A. Butterworth, L. Y. Pao, and D. Y. Abramovitch, "Adaptive-delay combined feedforward/feedback control for raster tracking with applications to AFMs," in *Proceedings of the American Control Conference*, (Baltimore, MD), AACC, IEEE, June 2010.

[121] J. A. Butterworth, L. Y. Pao, and D. Y. Abramovitch, "Analysis and comparison of three discrete-time feedforward model-inverse control techniques for nonminimum-phase systems," *Mechatronics*, vol. 22, pp. 577–587, August 2012.

[122] J. A. Butterworth, L. Y. Pao, and D. Y. Abramovitch, "A discrete-time single-parameter combined feedforward/feedback adaptive-delay algorithm with applications to piezo-based raster tracking," *IEEE Transactions on Control Systems Technology*, vol. 20, pp. 416–423, March 2012.

[123] J. A. Butterworth, L. Y. Pao, and D. Y. Abramovitch, "A comparison of ILC architectures for nanopositioners with applications to AFM raster tracking," in *Proceedings of the American Control Conference*, (San Francisco, CA), AACC, IEEE, June 2011.

[124] J. A. Butterworth, L. Y. Pao, and D. Y. Abramovitch, "Dual-adaptive feedforward control for raster tracking with applications to AFMs," in *Proceedings of the IEEE International Conference on Control Applications, CCA 2011*, (Denver, CO, USA), IEEE, IEEE, September 28-30 2011.

[125] D. Y. Abramovitch, "A comparison of Δ coefficients and the δ parameterization," in *To be submitted to the 2016 American Control Conference*, (Boston, MA), AACC, IEEE, July 2016.