Intro for DDDAS 2020 Industry Panel:
("Who am I and why am I here?")

Daniel Y. Abramovitch

Mass Spectrometry Division
Agilent Technologies
Santa Clara, CA 95051

- Born in Canada, grew up in Alabama, college at Clemson, grad school at Stanford. (Yes, I'm missing CFB.)
- Spent 30+ years working in industry on problems related to control, signal processing, and instrumentation.
- I don't have any magic answers, but as Liam Neeson says in *Taken*, *"What I do have are a very particular set of skills, skills I have acquired over a very long career."*
- Many of the DDDAS examples look like "big iron" with large budgets and many engineers per system.
- My experience is with "small iron", mechatronic systems with highly flexible dynamics, which blow up many of the standard assumptions.
- If I can help here, it is in showing what big iron DDDAS problems (*state-space on steroids*) can learn from the small iron problems.

Big Iron vs. Small Iron Problems

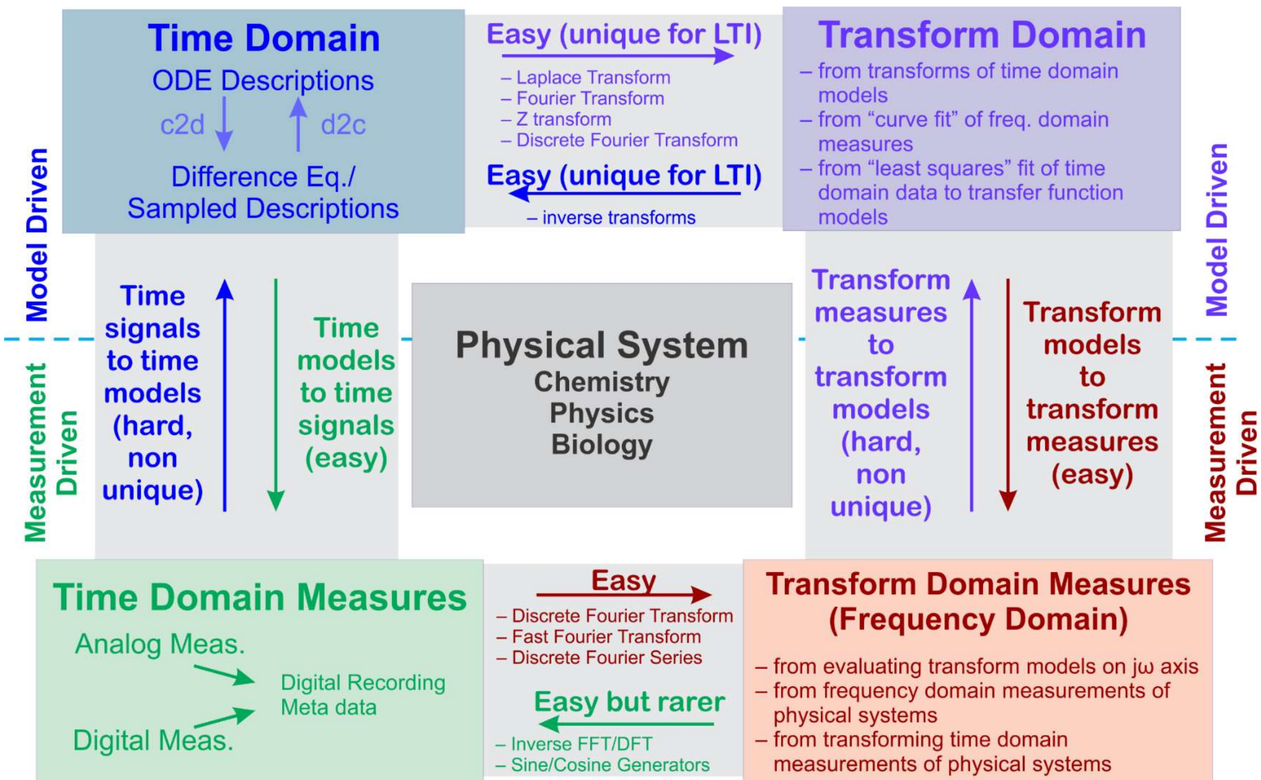
"Big Iron" problems:

- Individual device is very expensive (think process reactor or fighter plane or spacecraft)
- Mission of that device is even more expensive (think spacecraft)
  - Cost of device failure on a mission is catastrophic.
- So, many engineers and scientists can work on tuning each individual device.
  - Models for estimators and controllers may have common structure, but individual parameters are adjusted by skilled engineers for each individual device.

"Small Iron" problems:

- Individual device is relatively inexpensive, consumer scale, \$50 (HDD) – \$100,000 (Tesla)
- We can spend money on engineering, product design, and manufacturing line.
  - But the incremental cost to build any device has to be very small.
  - Want to avoid repairs. Low end ones are disposable.
- Individual engineers don't tune any one device.
  - Devices are either robust (generally lower performance) or
  - Self-tuning and self-diagnostic (which have to be done with product hardware, a.k.a. edge computing)

The Difficulty of Getting Model Parameters from Measurements: A Picture of the Domains

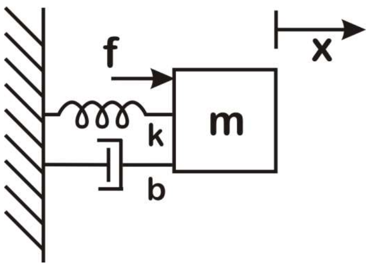


Top domains model driven.

Bottom domains measurement driven.

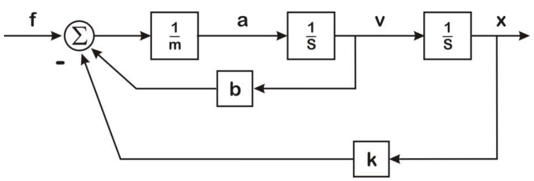
- Easy to go from top to bottom (evaluate model at time/frequency points)
- Very hard to go from bottom to top (need to fit lots of measurement data to a small set of parameters).
- Akin to password encryption/decryption
- A lot of folks give up.
- But the key step is going from measurement driven frameworks to model driven frameworks, and that step is the hard one.

Why Online ID of Discrete-Time Linear Models Usually Fails with Low Damping: The Fate of Physical Parameters Under Discretization



Even simple discretization makes coefficients very complicated

$$\frac{X(s)}{F(s)} = \frac{1/m}{s^2 + \frac{b}{m}s + \frac{k}{m}} \quad \Longrightarrow \quad \frac{Y(z)}{U(z)} = \frac{b_{0,D} + b_{1,D}z^{-1} + b_{2,D}z^{-2}}{1 + a_{1,D}z^{-1} + a_{2,D}z^{-2}}$$



Trapezoidal Rule:

$$\Delta = 1 + \frac{b}{m}\frac{T}{2} + \frac{k}{m}\frac{T^2}{4}$$

$$b_{0,D} = \frac{1}{\Delta}\left(\frac{1}{m}\frac{T^2}{4}\right) \qquad a_{0,D} = 1$$

$$b_{1,D} = \frac{2}{\Delta}\left(\frac{2}{m}\frac{T^2}{4}\right) \qquad a_{1,D} = \frac{2}{\Delta}\left(\frac{k}{m}\frac{T^2}{4} - 1\right)$$

$$b_{2,D} = \frac{1}{\Delta}\left(\frac{1}{m}\frac{T^2}{4}\right) \qquad a_{2,D} = \frac{1}{\Delta}\left(1 - \frac{b}{m}\frac{T}{2} + \frac{k}{m}\frac{T^2}{4}\right)$$

Moral: We need discrete time model, but it obscures physical meaning.

The Fate of Physical Parameters Under Discretization

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

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

where

$$\Delta = 1 + \frac{b}{m}\frac{T}{2} + \frac{k}{m}\frac{T^2}{4}$$

$$b_{0,D} = \frac{1}{\Delta}\left(\frac{1}{m}\frac{T^2}{4}\right) \qquad a_{0,D} = 1$$

$$b_{1,D} = \frac{2}{\Delta}\left(\frac{2}{m}\frac{T^2}{4}\right) \qquad a_{1,D} = \frac{2}{\Delta}\left(\frac{k}{m}\frac{T^2}{4} - 1\right)$$

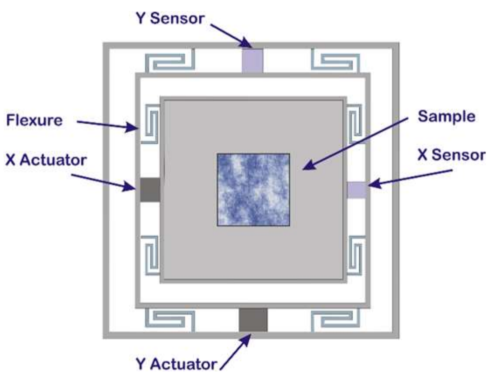
$$b_{2,D} = \frac{1}{\Delta}\left(\frac{1}{m}\frac{T^2}{4}\right) \qquad a_{2,D} = \frac{1}{\Delta}\left(1 - \frac{b}{m}\frac{T}{2} + \frac{k}{m}\frac{T^2}{4}\right)$$

- This is just a simple, common, second order model.
  - For higher order models, the obscuration is much worse.
  - For system ID or trying to create a state space model, the physical meaning is lost.
  - Consider the SNR needed to back these out from any set of measured signals.
  - Put another way, even Neo isn’t figuring out what’s in this matrix.
- Systems for which this online, discrete-time model regression ID usually works are usually well behaved, i.e.
  - Stable and well damped
  - Systems where one doesn’t care so much about not recovering the physical parameters.
  - Coincidentally, the same types of systems were ML/AI has had success. (Maybe not coincidental).

The Case for Connected Measurements

For a lot of reasons, a typical control lab often still has a bunch of beautiful and disconnected pieces of technology. On their own, they are good but limited.

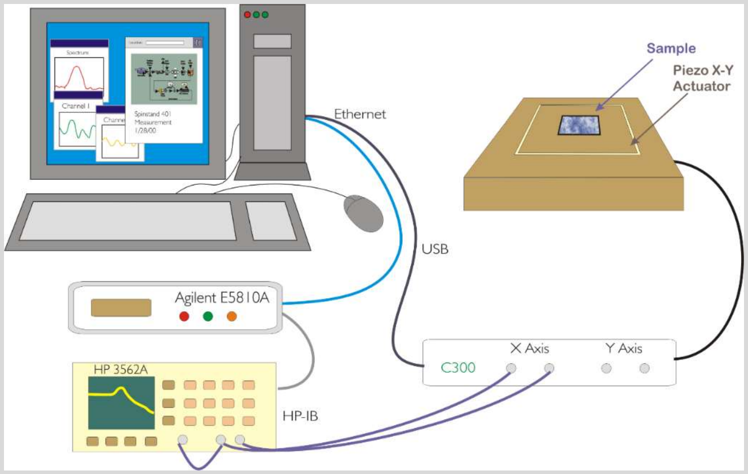
- They are from different vendors, with different interfaces, built by folks who would rather not be doing all that “not real engineering” programming.
- Some wonderful boxed instruments are both expensive and old. There are no inexpensive replacements *and* their communication interfaces are archaic.



“But the point of a measurement is lost, if you keep it a secret. Why didn't you record it so you could tell the world, eh?” – Dr. Strangelove, evangelizing on connected measurements

- What happens when you do the grunt work to put these heterogeneous pieces together?

Connecting Measurements Leads to Rapid Iteration



- DSA tied to stage controller, and through network hub to PC.
- PC runs measurement, aggregates data, and does waveform math (MATLAB).
- Data saved as web pages.

- Run DSA measurements on closed-loop system, controller.
- Use MATLAB to model existing controller or measure the existing controller by opening the loop on the system (disconnecting the wires from the controller to the X-Y stage).
- Open the loop with waveform math, divide out *C* and get *P*.
- With measured *P* design new *C*.
- With new *C* rewrap the loop (waveform math) to project T_{CL}.
- When projected T_{CL} looks good, download new *C* to stage controller.

You Can’t Do Big Data If You Mess Up the Little Data

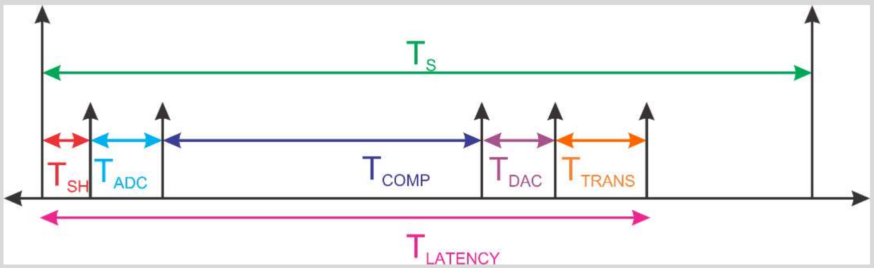
How the data is gathered matters

- The sensor, the instrumentation amplifiers, the ADC, the data collection path.
- But many “algorithm folks” display little interest in the “hardware”, the filtering, the sampling, the delay, the quantization, the relative accuracy of different data streams, their synchronization.
- There is often also a corresponding lack of interest in the physical process being measured.
- The hardware folks (scientists, digital and analog circuit designers) are often unaware of algorithm needs and how relatively simple adjustments that they could make might improve the AI.
- Why spend more on an ADC that samples 10, 100,1000, times faster than the physical system needs?
  - The answer, as George Carlin would say: *Because we can.*
  - The follow up is: What do we do with the extra samples?

Management understands engineering workstations and servers more than microcontrollers, FPGAs, and DSPs.

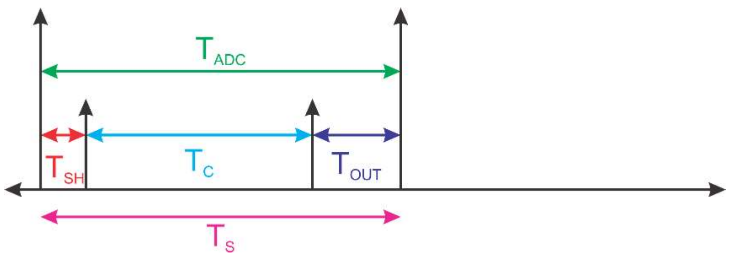
- Product cost constraints make it hard to build in computational head room.
- The long term benefits of building infrastructure that enables some possible benefit of ML/AI are harder to define than the immediate costs of a more powerful embedded computing platform.

Systems Folks Need to Get Involved with Component Selection: Ex. ADCs



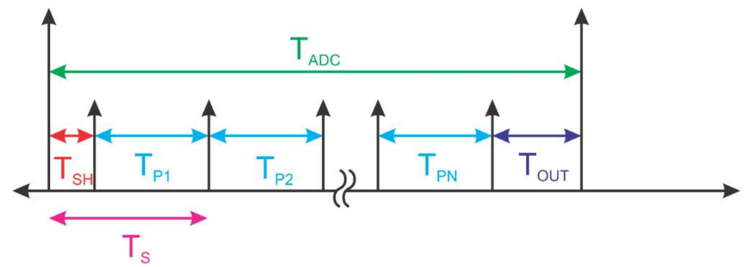
Sample period has many components:

– sample & hold, ADC time
– computation time
– DAC time, transmission time
– all have to finish before T_S



ADC time also has components:

– sample & hold, conversion, output/transmit time



Conversion calcs often broken up into stages

- Can allow smaller T_S
- But latency often much bigger (DSP don't care)

When someone not attuned to latency makes choice, they can blow 90% of the phase margin & bandwidth.

– These "Oops!" moves cannot be fixed by any algorithm.

What Can Small Iron Problems Teach Us About DDDAS

“I can’t change the laws of physics.” – Scotty

- Even in small problems, understanding the underlying process saves a lot of computation by *cutting the number of tuning parameters by several orders of magnitude*.
- Going from empirical “data-driven” fits to parametric models makes system understanding and prediction more efficient and far less brittle. *And, yes, it’s really hard.*
- Systems/control engineers have a chance to be the great integrators of these problems.
  - Only someone who understands ADCs and latency can convince a circuit designer why they should change their chosen circuit.

Building hardware with an awareness to the machine intelligence algorithms preempts a lot of problems.

- If we are smart about how we use Moore’s Law and all the inexpensive redundancy it affords, we can hand or sophisticated algorithms much better data.
- “It’s not who has the best algorithm that wins, It’s who has the most data.” – Andrew Ng
- Maybe make that “the most *good* data”.

If it’s true in general, it should be true in a single case.

- These same principles might be useful in these “Big Iron” systems.