

Lecture 3

Digital Control

- Part 1: Sampling
- Part 2: User interface
- Part 3: Digital time & frequency domain

G1600726

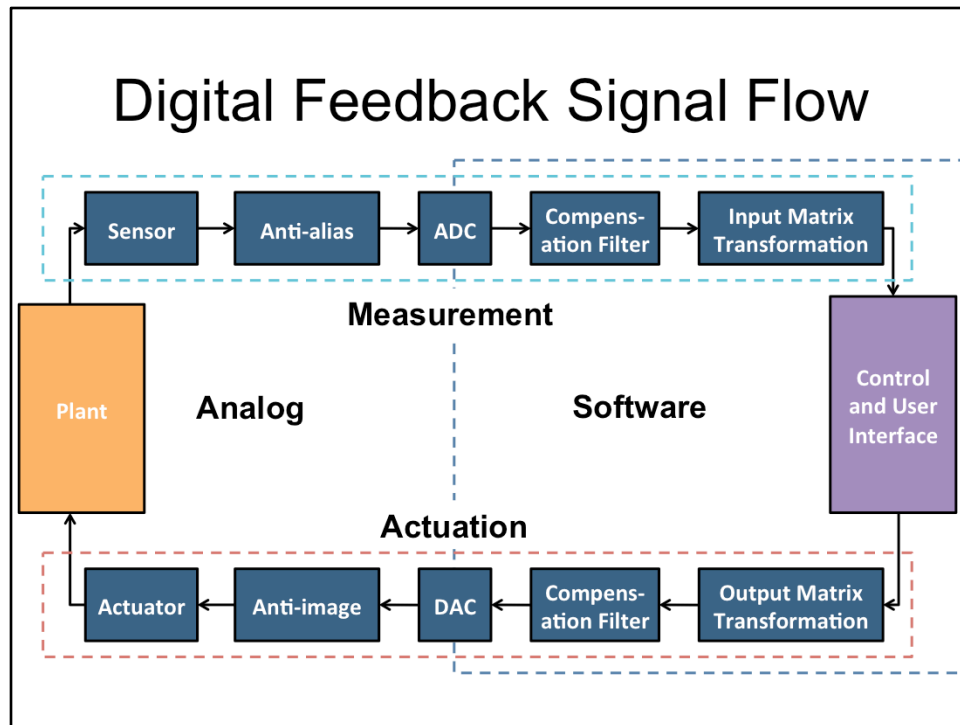
136

Lecture 3

Digital Control
- Part 1: Sampling

G1600726

137



This is a typical block diagram for a feedback loop made with a digital control system. Note the symmetry of the diagram. The left is all analog components, the right software. The top is the sensing path, the bottom the actuation path. Many of the blocks are what we have seen before, the plant, sensor, actuator, and controller. The compensation filters and matrix transformations are optional, and not unique to digital control systems. These are simply used to put signals into some desired units (e.g. meters) and coordinate system (e.g. x-y-z Cartesian coordinates). The main new thing here are the ADC and DAC, which move signals into and out of the computer; and the Anti-alias and Anti-image filters which smooth over the transitions through the ADC and DAC.

Sampling Hardware

- **ADC – Analog to Digital Converter.** Samples the data and brings it into the computer.
- **Anti-Alias filter** – filters noise close to and above the sampling frequency.
- **DAC – Digital to Analog Converter.** Outputs the computer signals.
- **Anti-Image filter** – filters harmonics close to and above the sampling frequency.

139

The Anti-alias filter removes signals close to and above the sampling frequency, before the ADC brings the signals into the computer. The anti-image filter removes harmonics of the computer output before sending those signals to the actuators. We'll see examples of this in the upcoming slides.

Optional Software Blocks

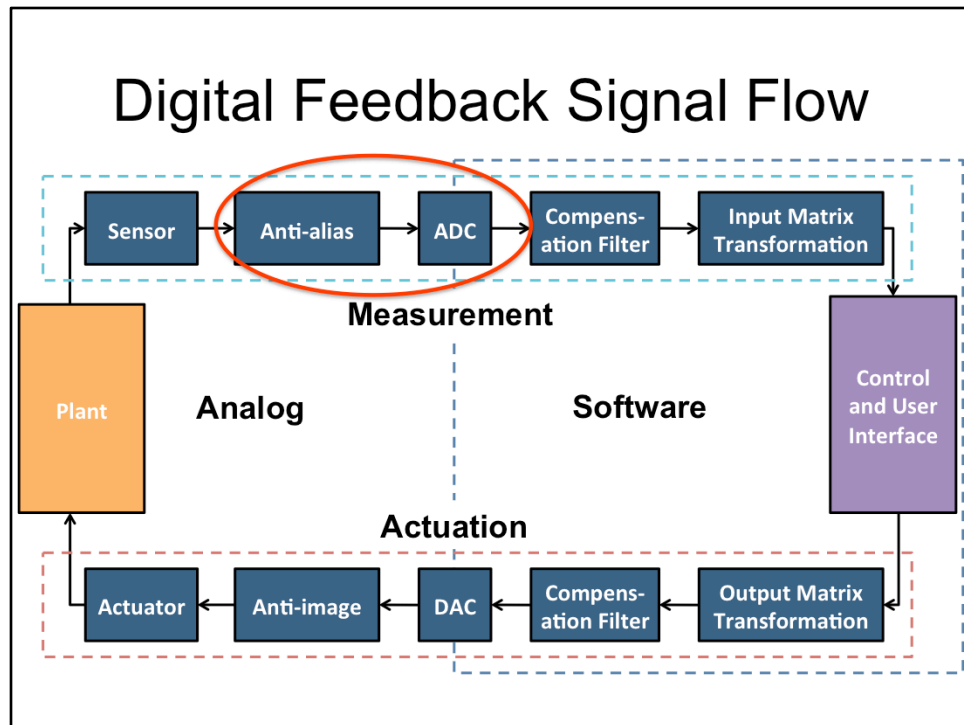
- **Compensation blocks** – Can also be called 'calibration' blocks. These are optional, and are used to cancel out the analog frequency response of sensors and actuators and/or put them into meaningful units.
- **Matrix transformations** – Also optional. Are used to put sensor and actuator signals into useful coordinate systems. For example, 2 vertical sensors next to each other can be combined to give you a vertical (Z) and a rotation (roll) signal.

140

These blocks are optional, but make life easier.

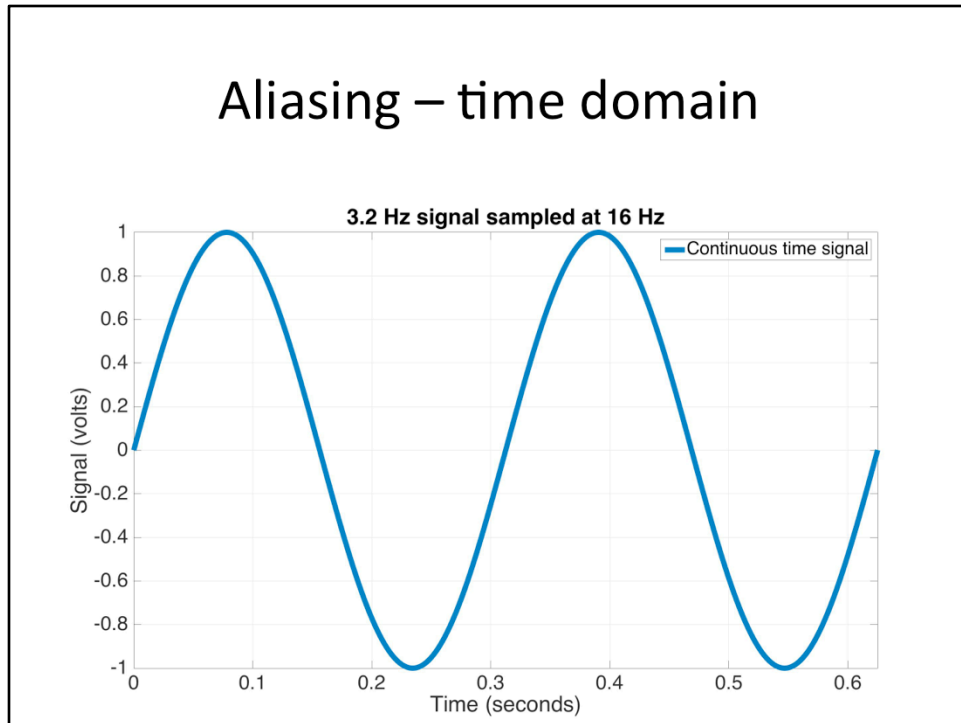
Software Blocks

- **Control and user interface** – Where the control filters and control logic goes. Also, the software that allows the user to interact with the control lives here.



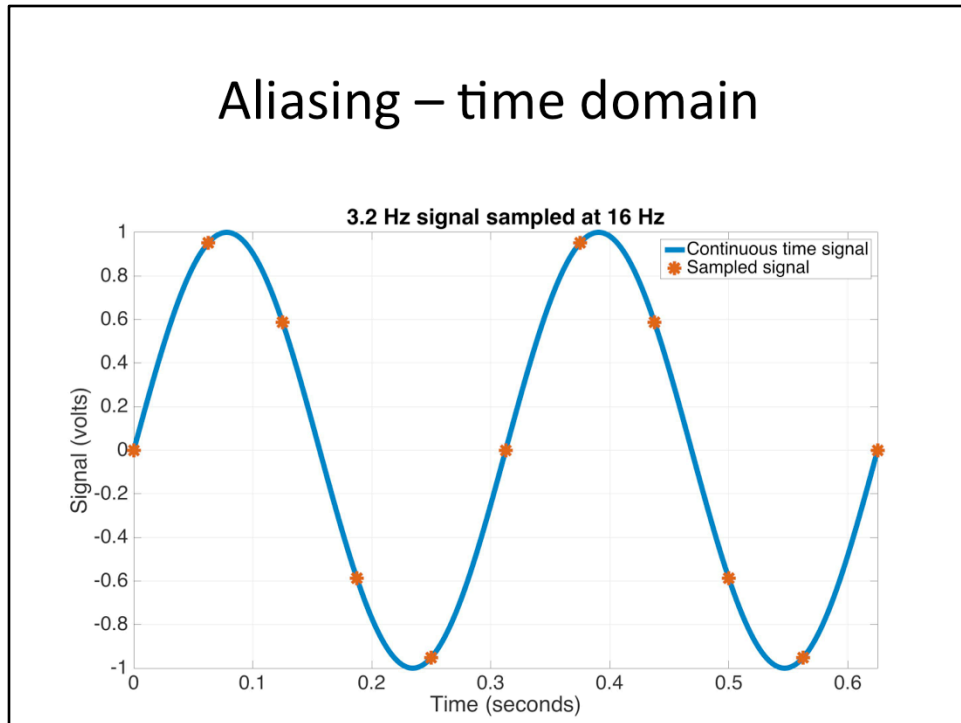
Let's take a look next at what the Anti-alias filter does for us.

Aliasing – time domain



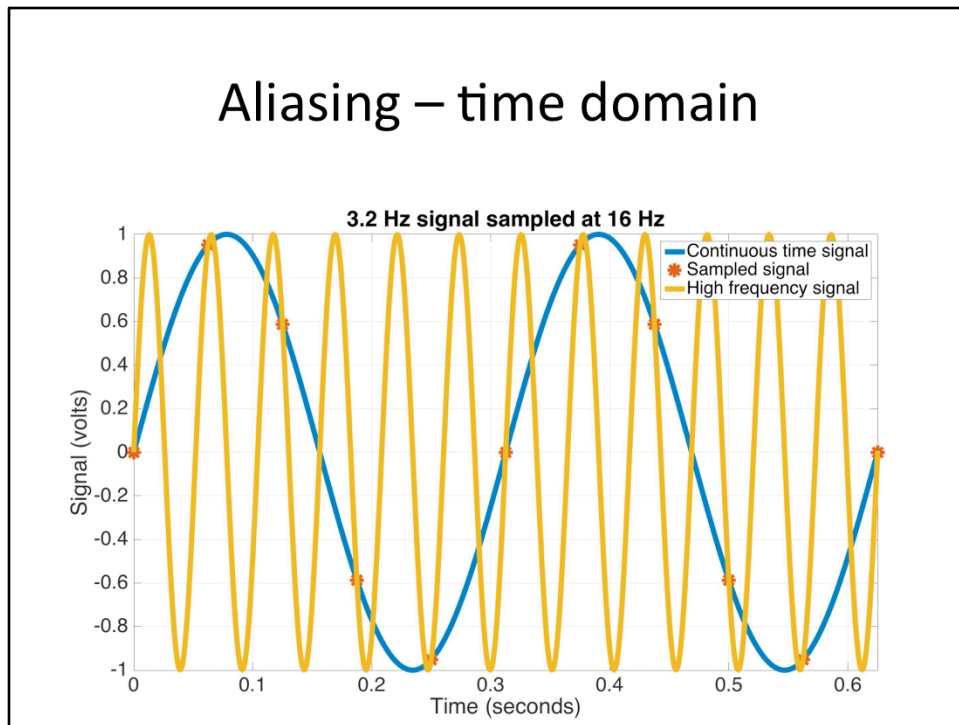
Here is an example of a continuous sine wave, at 3.2 Hz.

Aliasing – time domain



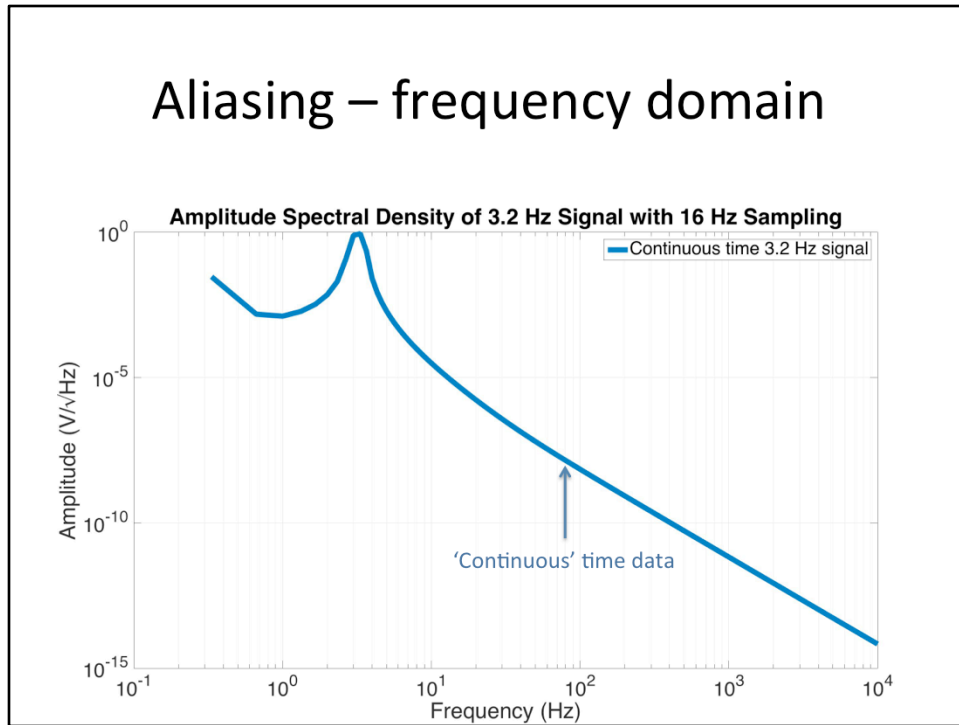
We now sample this wave at 16 Hz with the ADC. The continuous wave passes cleanly through each of the sampled points.

Aliasing – time domain



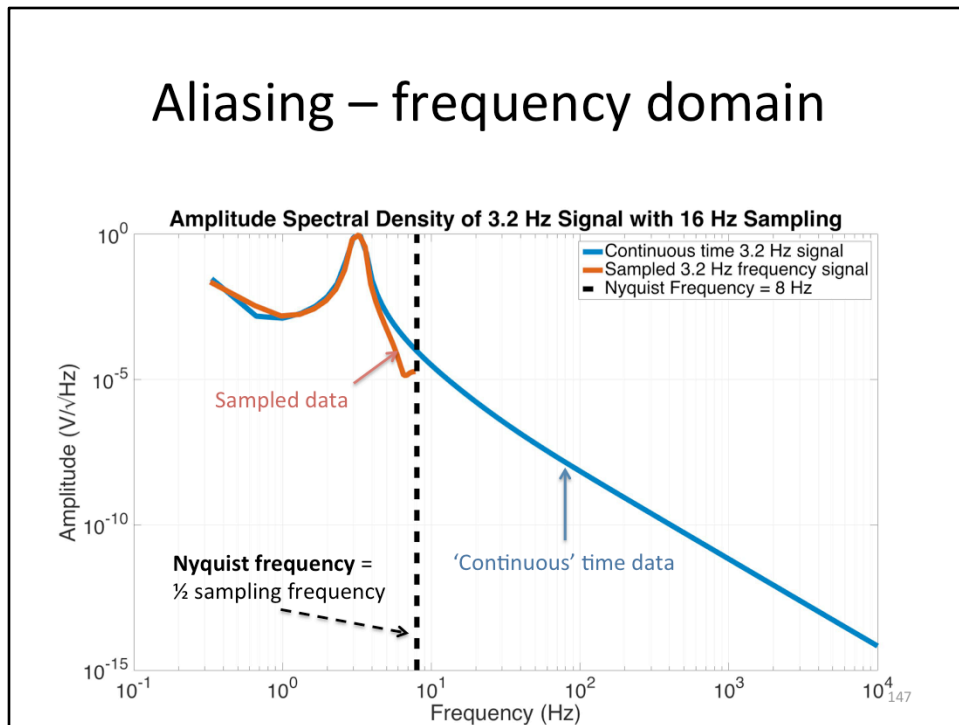
But, we could also draw a higher frequency sine wave through these points. This yellow line is at $3.2+16=19.2$ Hz. So, when we sample with the ADC, we don't know which one we have. Thus, we make the assumption that our sampled points are passing through the lowest possible frequency sine wave.

Aliasing – frequency domain



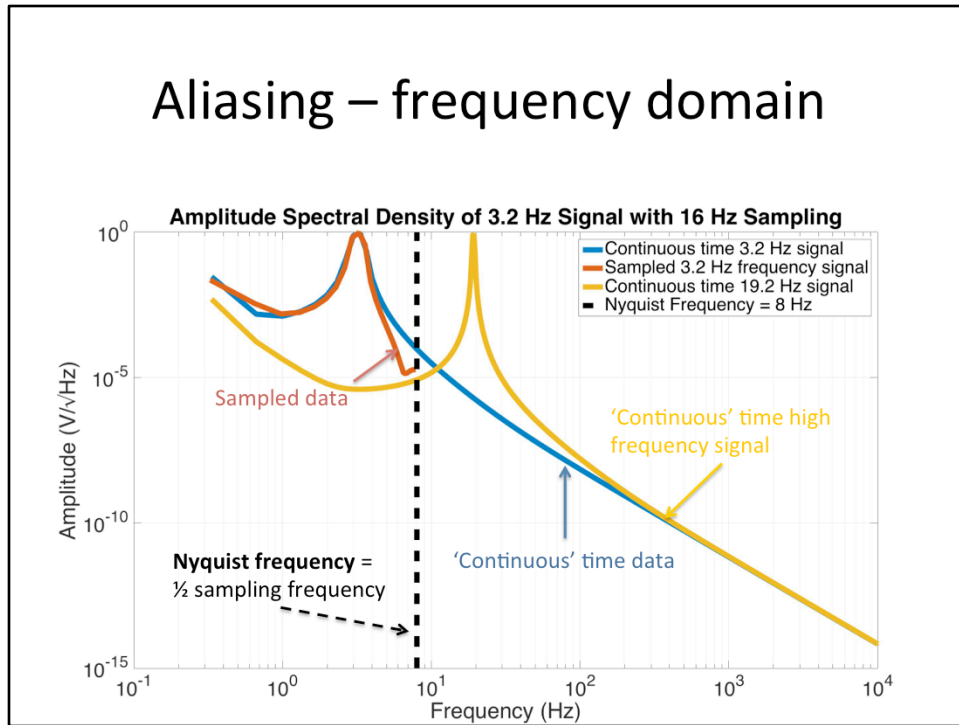
Moving from the time domain to the frequency domain, here is an amplitude spectral density (ASD) of the 3.2 Hz continuous wave.

Aliasing – frequency domain



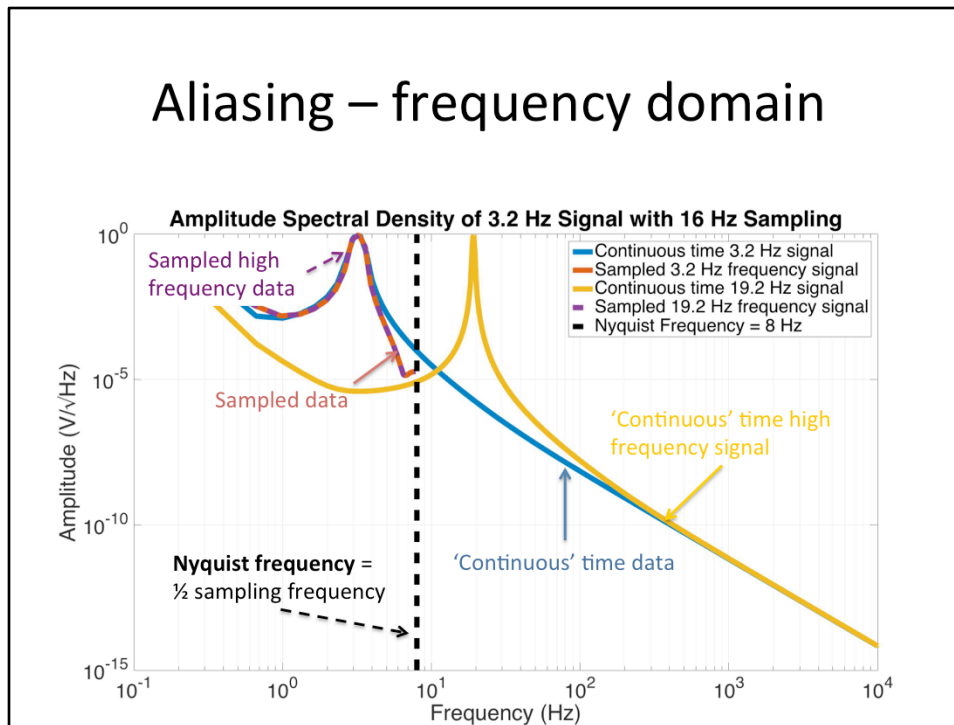
Adding the ASD of the sampled data, we get the red curve (because we assume the sampled points represent the lowest possible frequency). I have also added the vertical dashed line at what is known as the Nyquist frequency. This is half the sampling frequency. It turns out that you can only recover signals that are $<$ half the sampling frequency. Basically, the Nyquist sampling rule says that you need at least 2 samples per sine wave to recover that sine wave. This is why the red line stops at this vertical Nyquist frequency line, we can't recover anything beyond it.

Aliasing – frequency domain



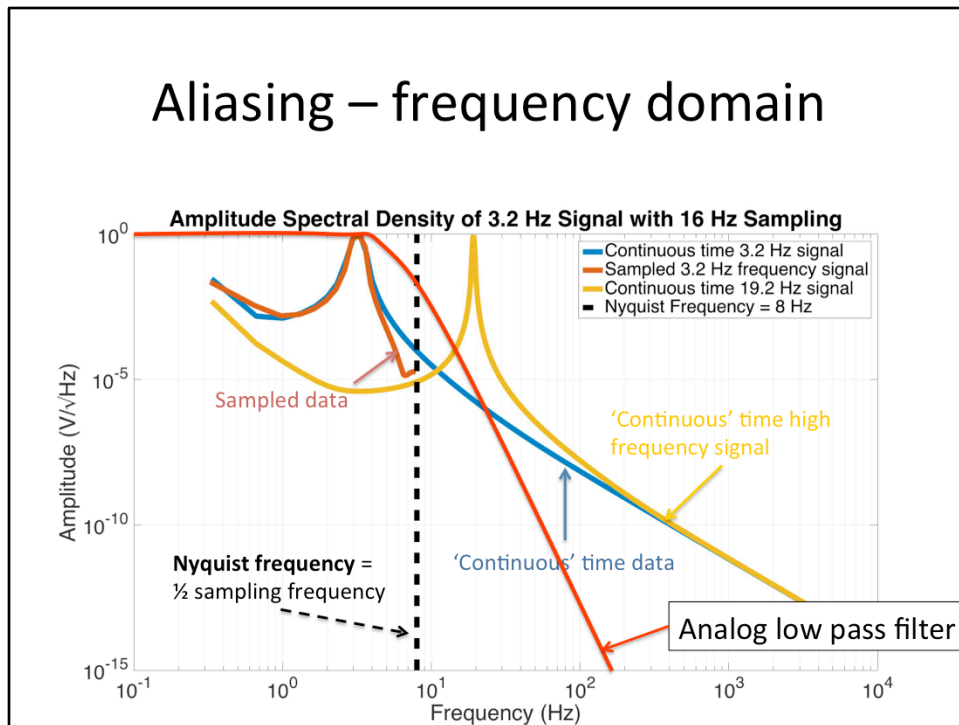
Now, let's look at the ASD of the yellow line. It's at 19.2 Hz.

Aliasing – frequency domain

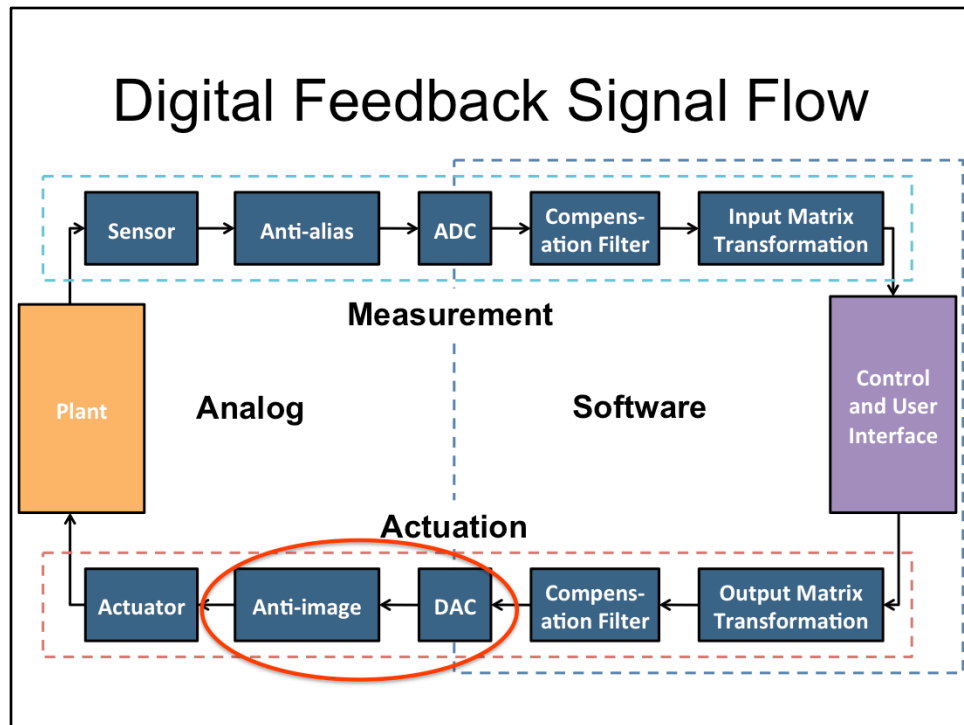


If we were to sample that line, unsurprisingly our sampled ASD would look exactly like our sampled ASD for the 3.2 Hz wave, because we assume the samples represent the lowest possible frequency. This down-converting of signals from above the Nyquist to below is called 'Aliasing'. The aliased frequency is found by subtracting the sampling frequency from the true frequency as many times as necessary until it is less than the Nyquist (and taking the absolute value if necessary). **Aliased frequency = $\text{abs}(\text{true frequency} - n * \text{SamplingFrequency})$, where n is the integer required to make the aliased frequency less than the Nyquist.**

Aliasing – frequency domain



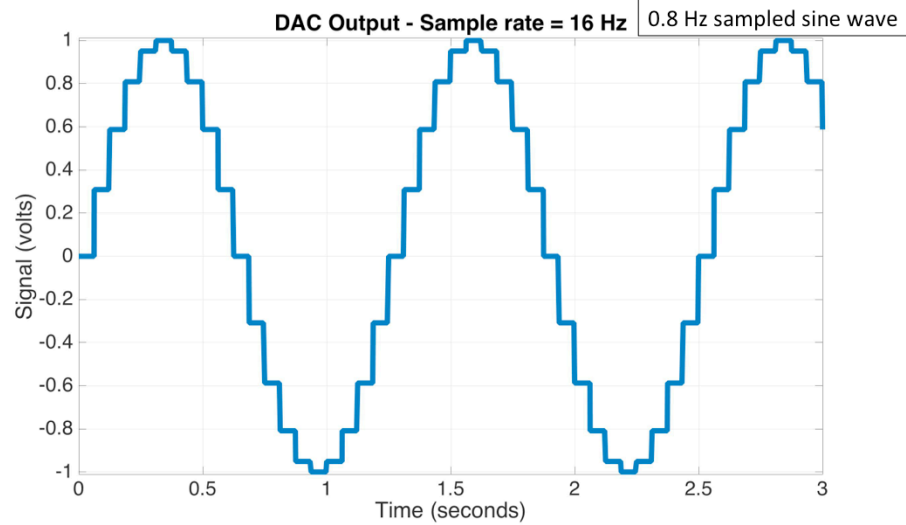
To avoid aliasing, we apply a low pass filter that cuts off any signals above the Nyquist. This way, when we assume our samples pass through signals below the Nyquist, our assumption is not a bad one.



Next, let's look at the anti-image filter. It turns out, this is very similar to the aliasing problem.

Anti-Image filtering

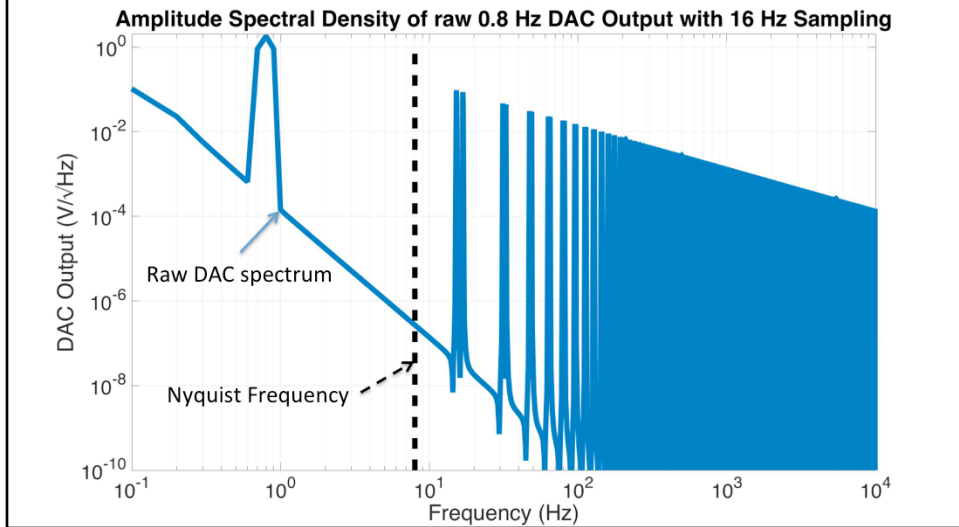
Raw DAC output, before AI filtering



Here is an example of a 0.8 Hz sine wave that the DAC puts out, with 16 Hz sampling. Note all the discrete steps. This happens because the DAC holds its value between samples (known as a zero-order-hold). The problem with these steps is that all those little corners generate higher order harmonics. If we take the ASD of this curve, we see..

Anti-Image filtering

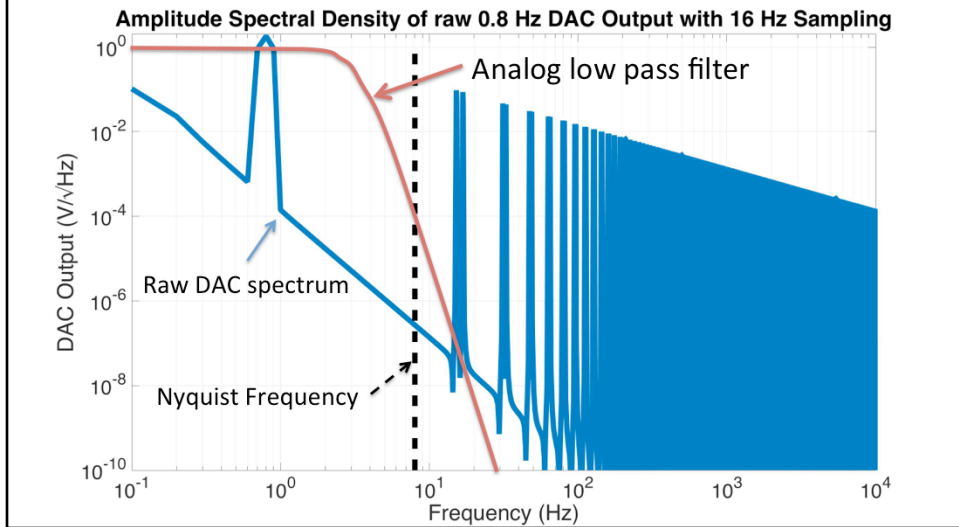
Raw DAC output, before AI filtering



...this. Note, we do get a nice peak the desired 0.8 Hz. However, we also get an infinite series of peaks at all the harmonics.

Anti-Image filtering

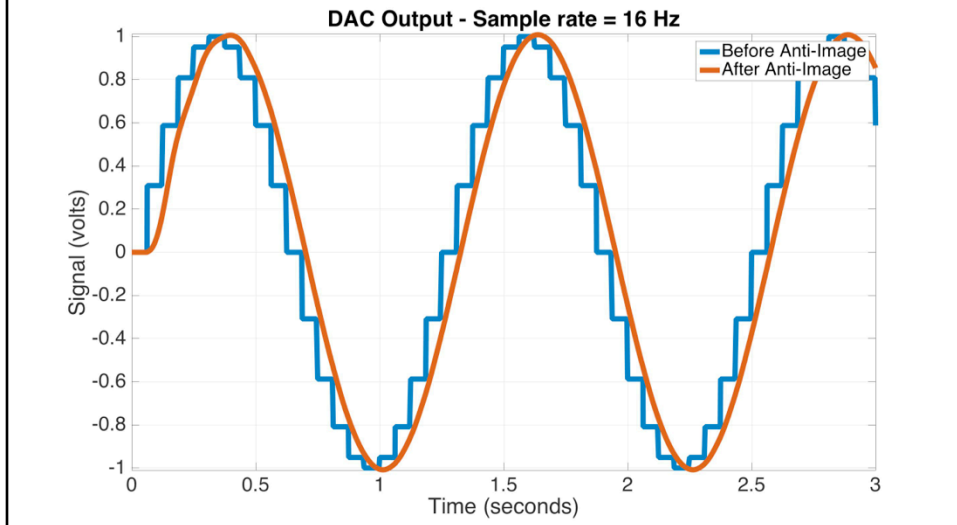
Raw DAC output, before AI filtering



Like in the aliasing case, we apply a low pass filter to remove all those harmonics. This is called the anti-image filter.

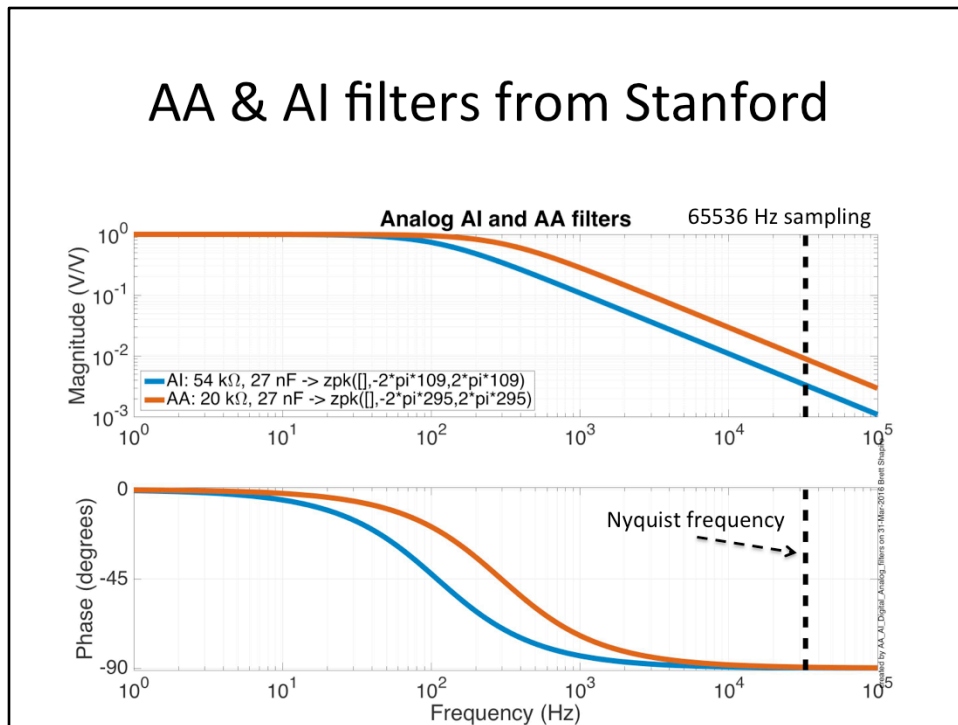
Anti-Image filtering

DAC output, after AI filtering

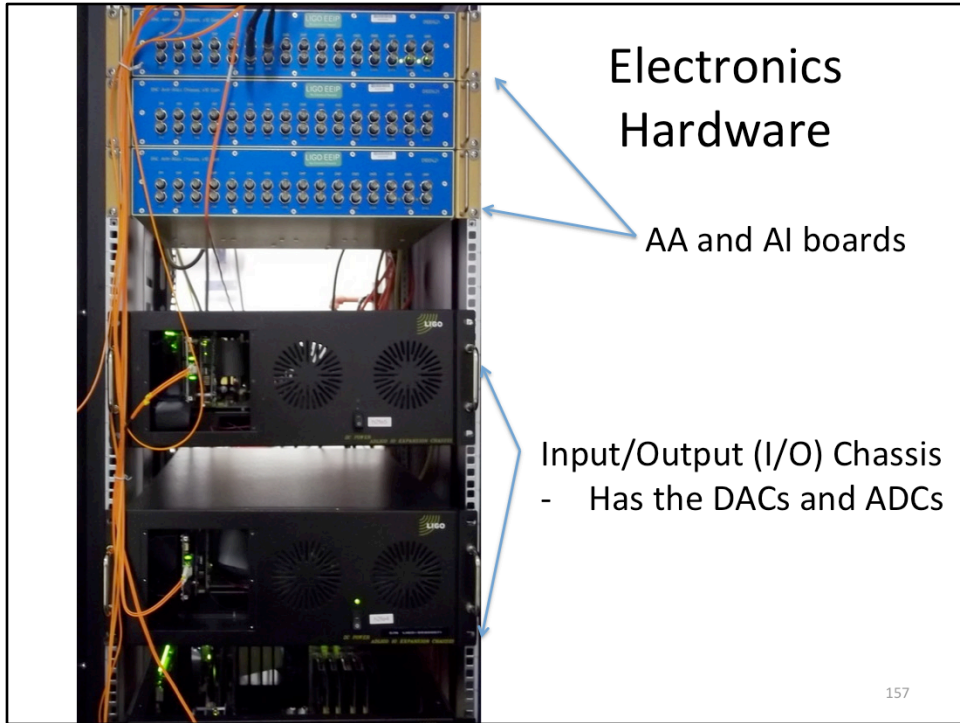


After applying the anti-image filter, we now have a smooth 0.8 Hz sine wave. Note, there is a small phase delay, resulting from the phase loss of the anti-image filter.

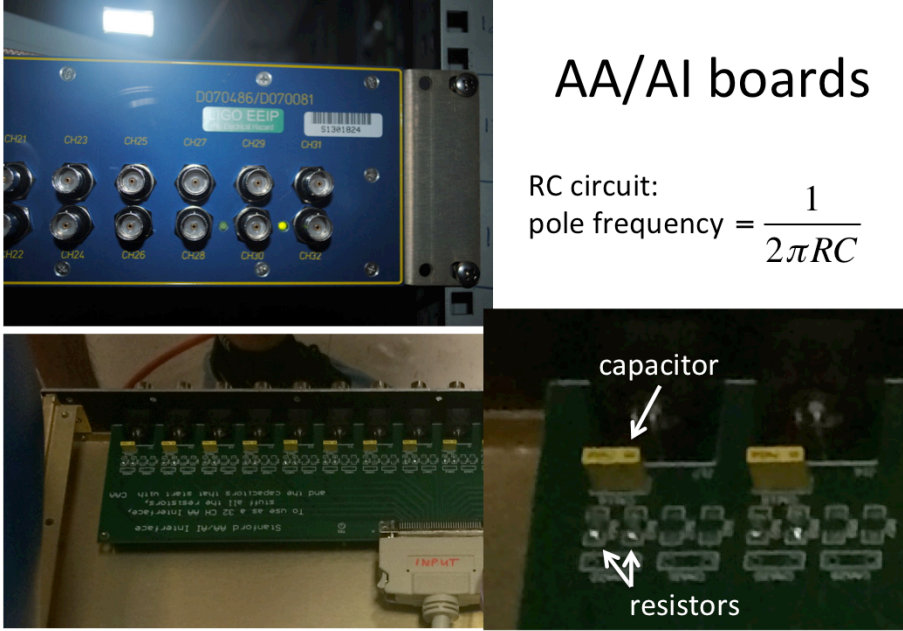
AA & AI filters from Stanford



Here are example anti-alias and anti-image filters from Stanford. The sample rate is about 65 kHz. Likely, there is no particular reason why the two filters aren't the same. They just as well could be. Probably it is because our system is a prototype system and the components were installed at different times. These filters have just a single pole, at 109 Hz or 295 Hz. With only a single pole, the pole frequency must be much less than the sampling rate. See the backup slides describing how we improve on this with oversampling and the use of additional digital AA and AI filters.



Here is what some of these components look like at the sites, in their electronics rack.



AA/AI boards

RC circuit:
pole frequency = $\frac{1}{2\pi RC}$

capacitor

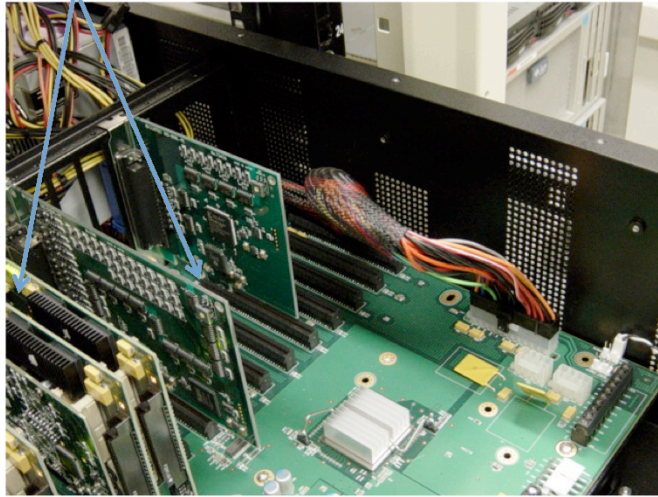
resistors

158

If you look inside an AA or AI board, you'll see some resistors and capacitors that give you a single pole low pass filter.

I/O Chassis

DAC and ADC cards



From T1000422

159

If you look inside the IO chassis, you'll see some ADC cards and DAC cards.

Computers

Front-end computer

Runs the real-time control system
Receives signals from ADC
Sends signals to DAC

Workstation

Runs the user interface



160

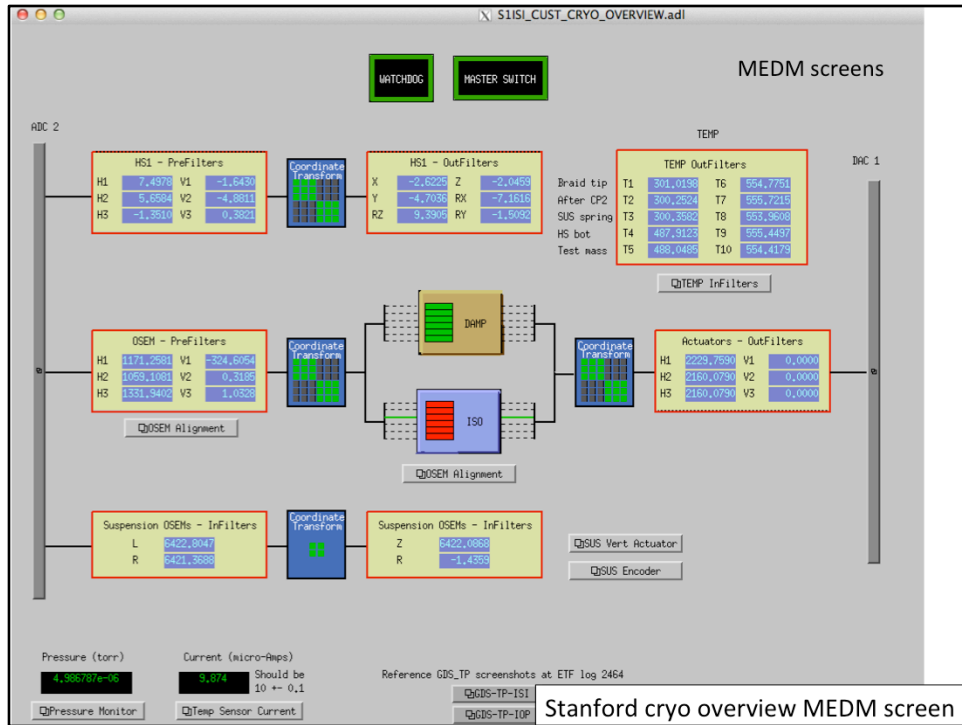
For computing, there are two main types of computers. The front-end computer is the one that runs the controller, and receives signals from the ADC, and sends signals to the DAC. The workstation is the user interface, which has all the associated user interface software and diagnostic tools.

Lecture 3

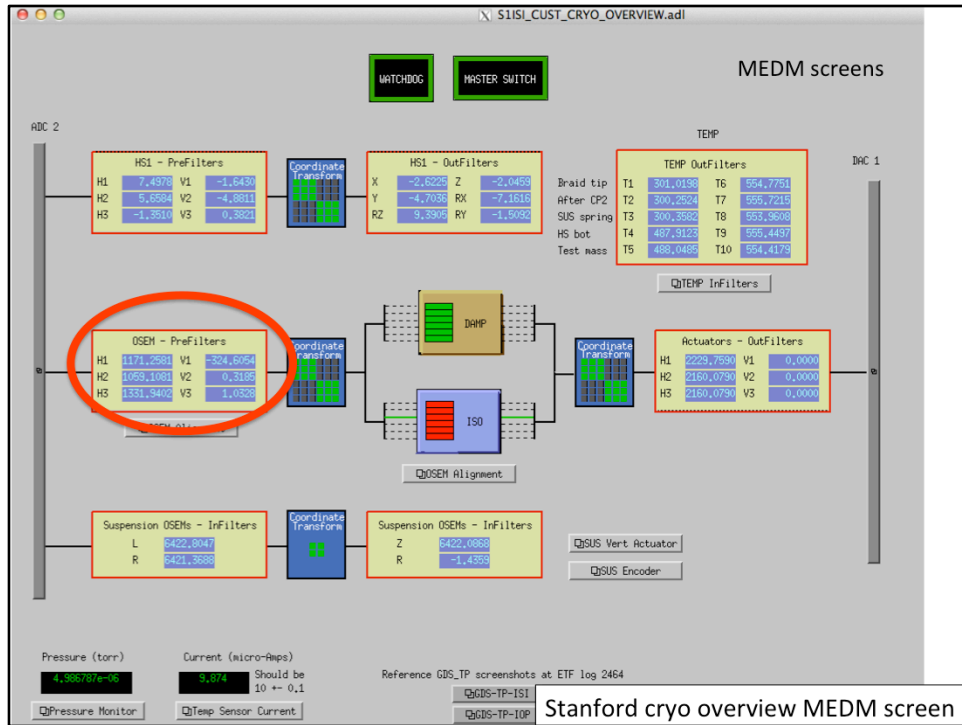
Digital Control
- Part 2: User interface

G1600726

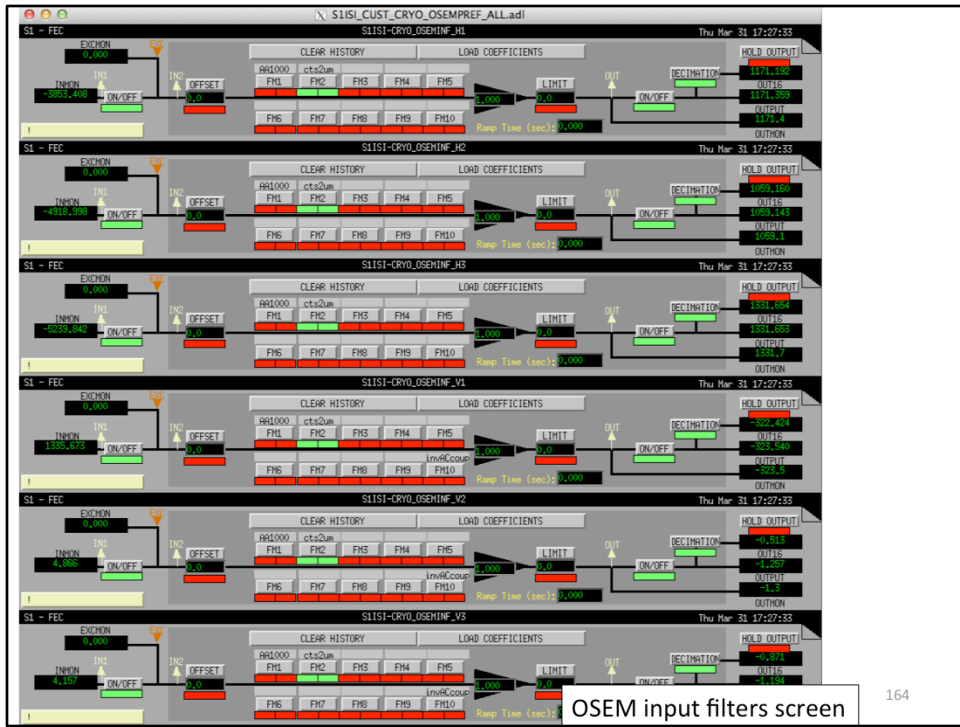
161



This is an example of the main user interface screens. These screens are custom made by the user. In this case, it is for the cryogenics platform at Stanford. The software is called MEDM. The signal flow follows the block diagram we saw earlier, with the ADC on the left, the DAC on the right. In between, the sensor signals go through some compensation filters and matrices. Depending on the sensor group, those signals are then sent through some control filters, then through some more matrices and compensation filters, before going to the DAC.

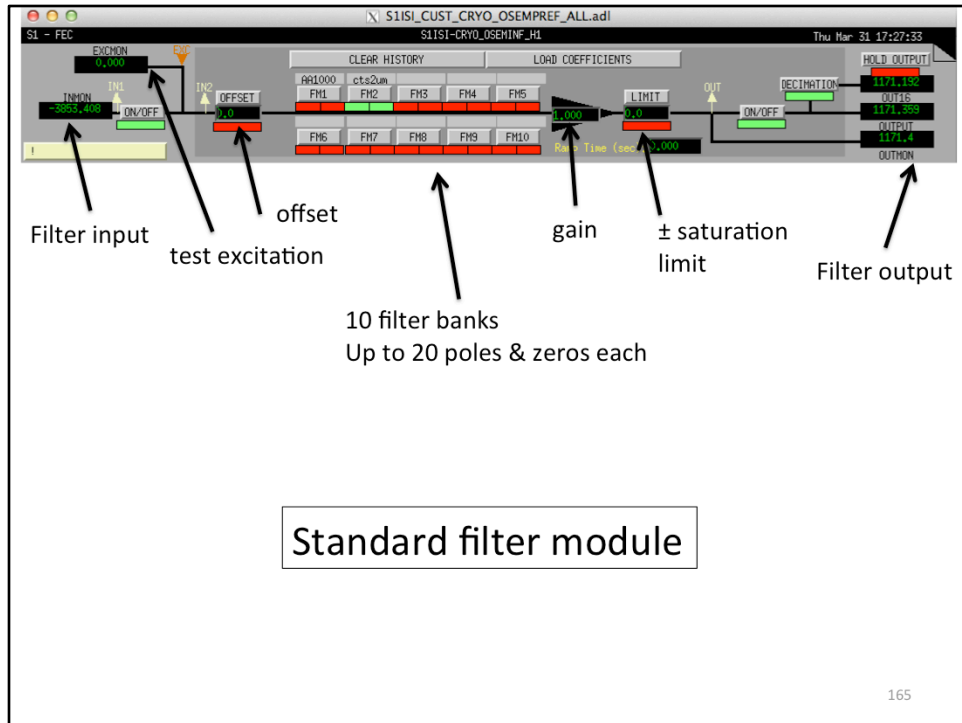


Each of these boxes you can click on to get more detail. Let's take a look inside the OSEM prefilters (or compensation filters).

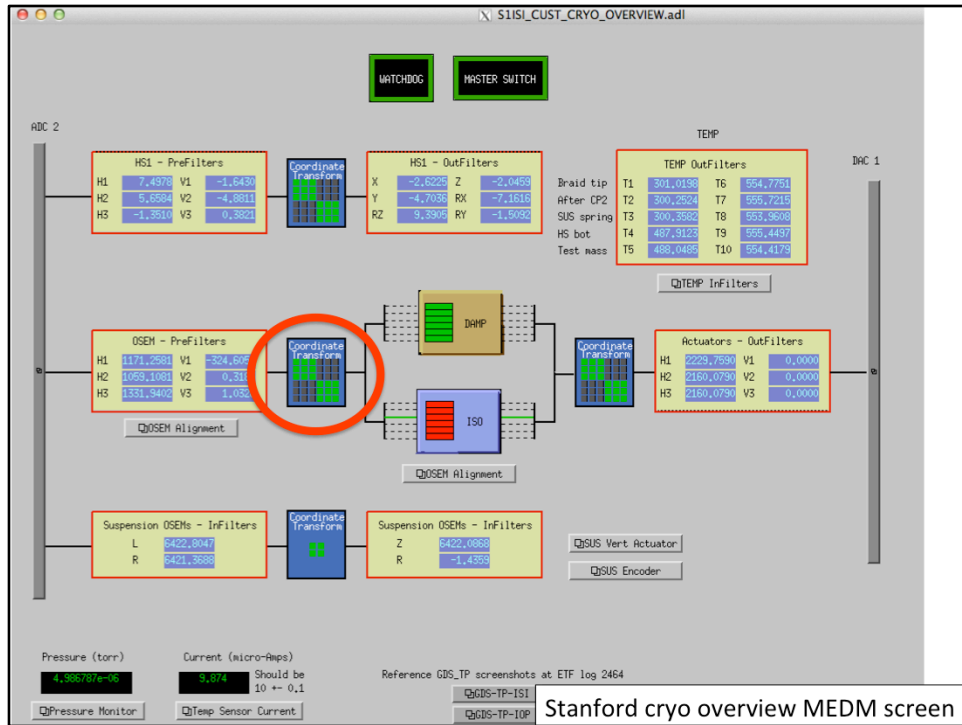


OSEM input filters screen

Here you see a list of 6 filter modules, 1 for each of the 6 OSEMs.



Let's focus on just one. This is the standard LIGO filter module medm screen, which is automatically generated for each realtime filter in the control system. Signals come in on the left, go through a selection of 10 possible filters banks, and then output on the right. You can also apply test excitations, offsets, gains, saturation limits, and flip various switches on and off. Here, since this is a compensation (or calibration) filter, the 2nd filter bank is engaged which converts the raw OSEM signal from units of counts to microns.



Let's now move along the signal path and look inside the OSEM sensor matrix transformation.

OSEM Sensor Input Matrix

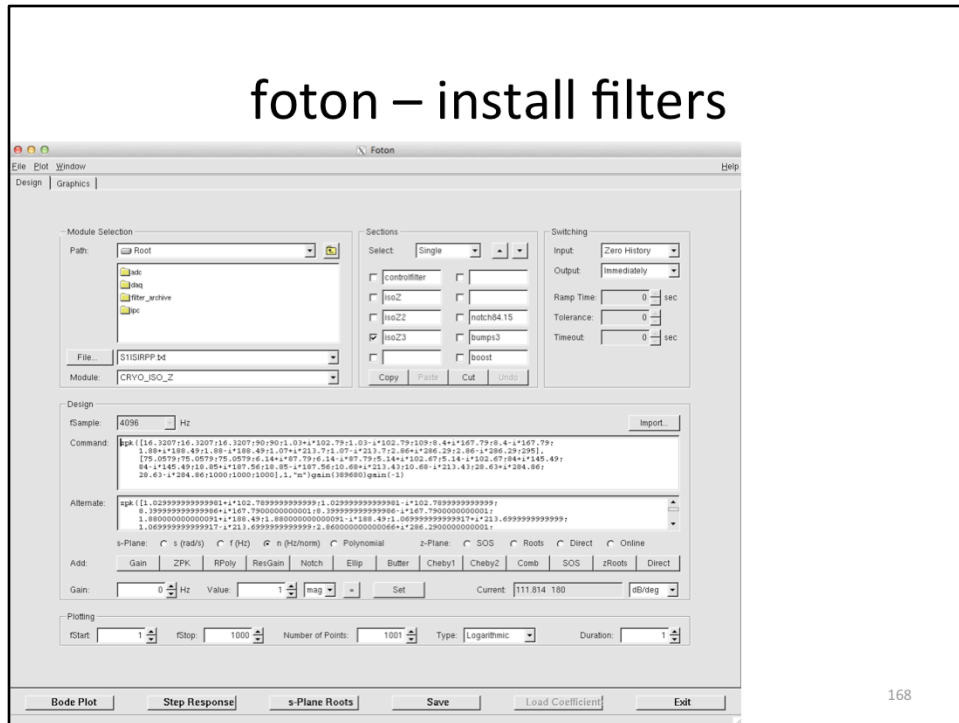


	H1	H2	H3	V1	V2	V3
X	0.22230	-0.6555	0.43320	0.00000	0.00000	0.00000
Y	-0.6285	0.12170	0.50680	0.00000	0.00000	0.00000
RZ	0.64430	0.64430	0.64430	0.00000	0.00000	0.00000
Z	0.00000	0.00000	0.00000	0.33333	0.33333	0.33333
RX	0.00000	0.00000	0.00000	-0.4315	1.27110	-0.8396
RY	0.00000	0.00000	0.00000	1.21860	-0.2356	-0.9830

167

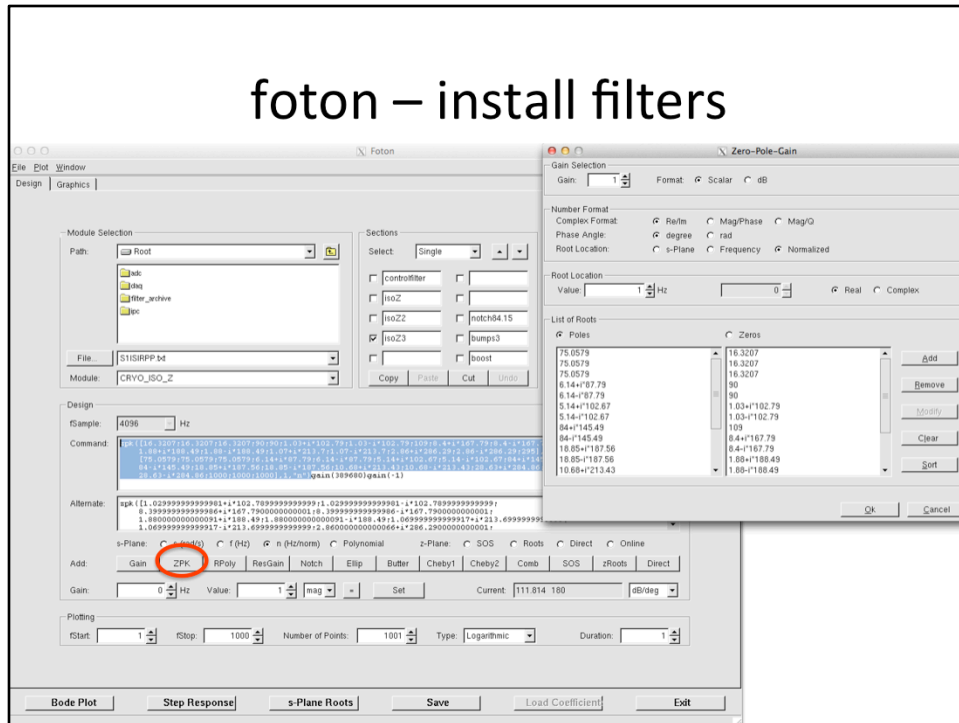
We see a 6 by 6 matrix, that converts the OSEM signals into an X, Y, Z coordinate from around the platform's center of mass, from the original locations of the OSEMs. 3 of the OSEMs measure the horizontal displacement of the 3 corners of the platform; the other 3 measure the vertical motion of those 3 corners. For example, to get Z, or vertical motion of the center of mass, we simply take the average of the 3 verticals: $(V1+V2+V3)/3$.

foton – install filters

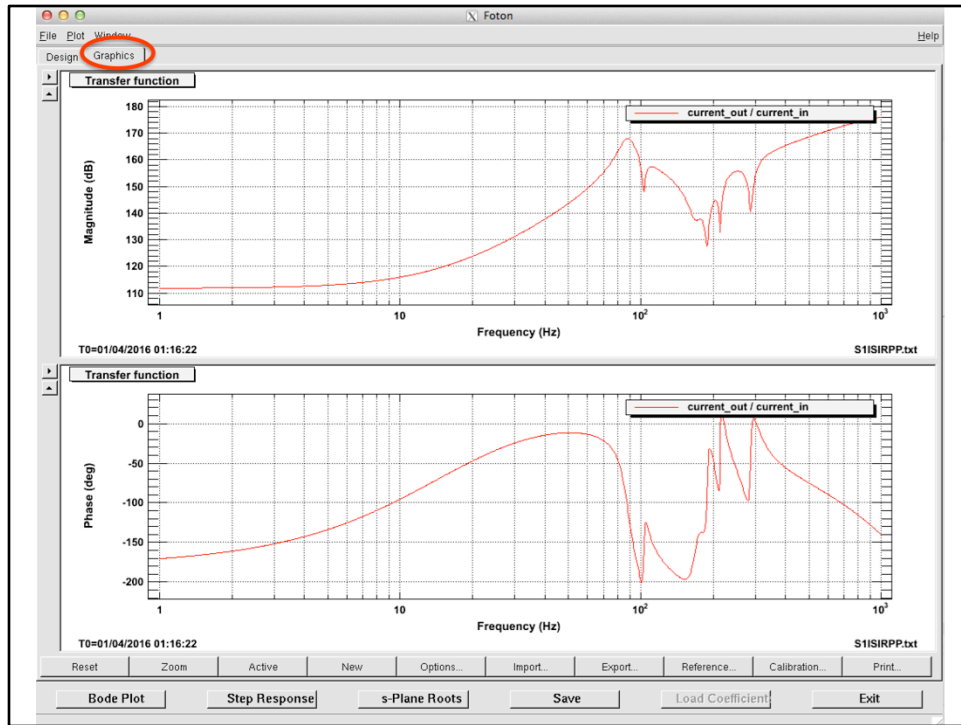


To load filters into the filter banks of the filter module screen, we use an interface called foton.

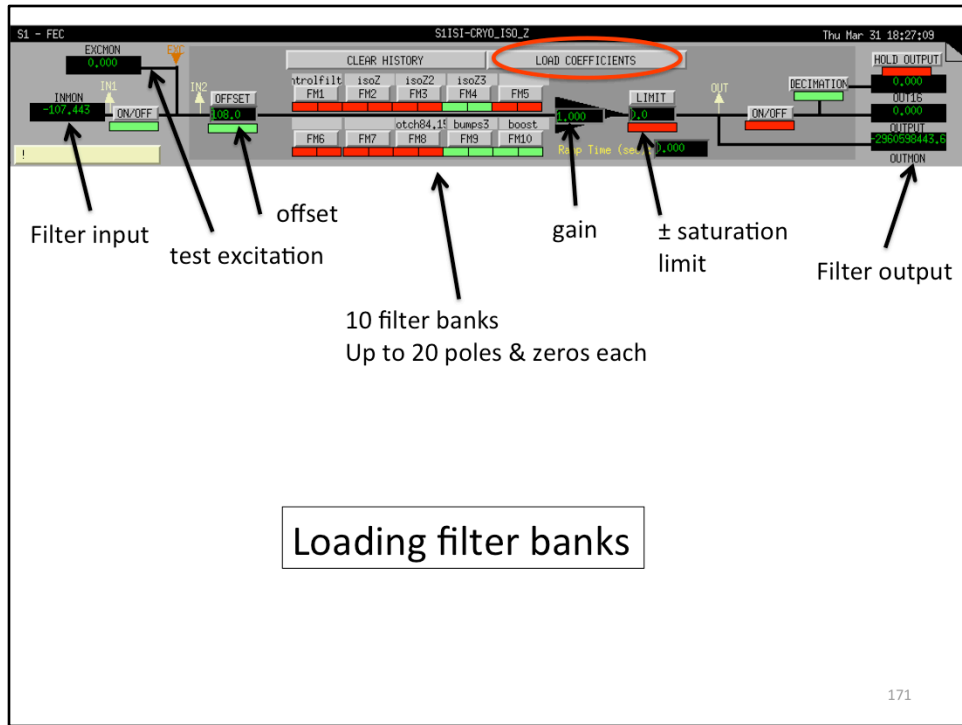
foton – install filters



There are various options for entering filters. A simple one is the ZPK option, which stands for zero, pole, and gain (the name zpk is consistent with matlab notation). With this zpk option, you simply type in the values for all the zeros and poles for each of the 10 possible filter banks. Each bank can hold up to 20 zeros and poles.

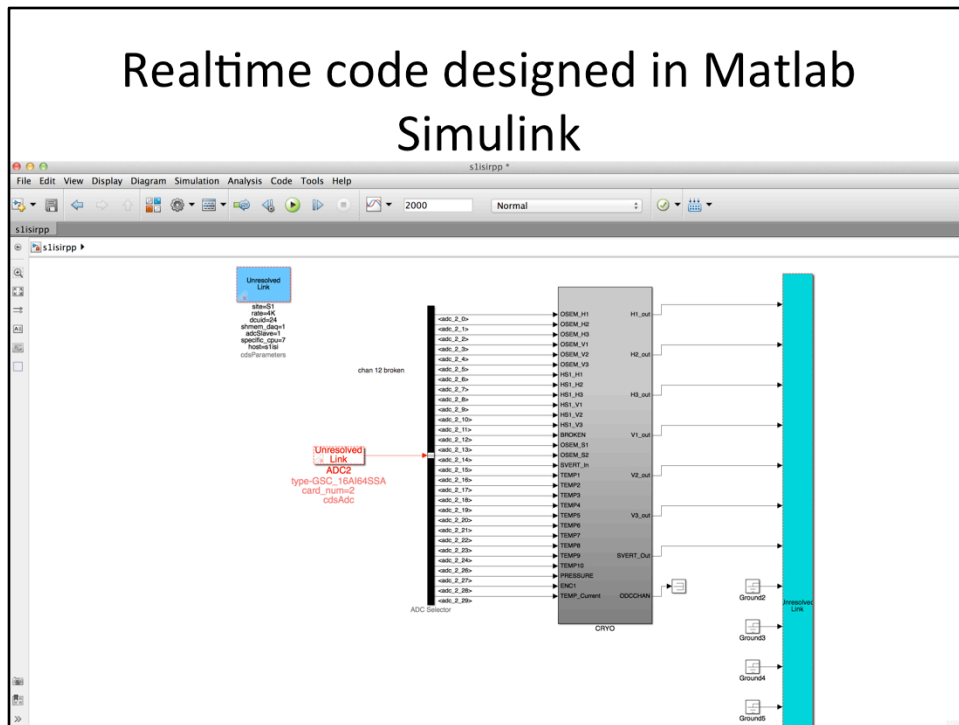


Once you have created a filter, you can plot it to make sure it looks the way you expect. This particular example, is rather complicated. However, this is the beauty of digital controls, it is easy to make an arbitrary filter just by typing in the values. Analog filter like this would require a very complicated circuit, and be very difficult to modify.



After the filter is loaded into foton, just hit the load coefficients button on the filter's medm screen, and this updates the screen.

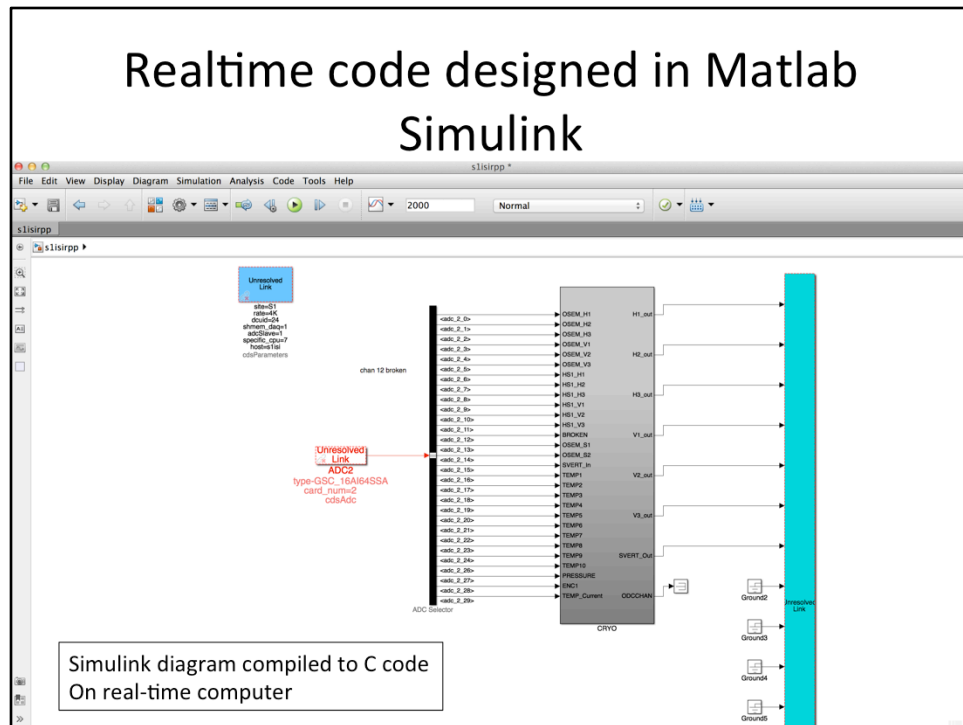
Realtime code designed in Matlab Simulink



All the medm screens are just user interfaces for the realtime code running on the front-end computer. This front-end code is generated by simulink diagrams (through matlab), which define everything about the control system's signal flow. It is here where you decide where the filters and matrices go. You can also setup any kind of logic or install C code to do what is infeasible in simulink alone.

Here we see the same basic signal flow again. The ADC is represented on the left, the DAC on the right. There is a CRYO block in the middle that contains all the real-time software for the stanford cryo platform.

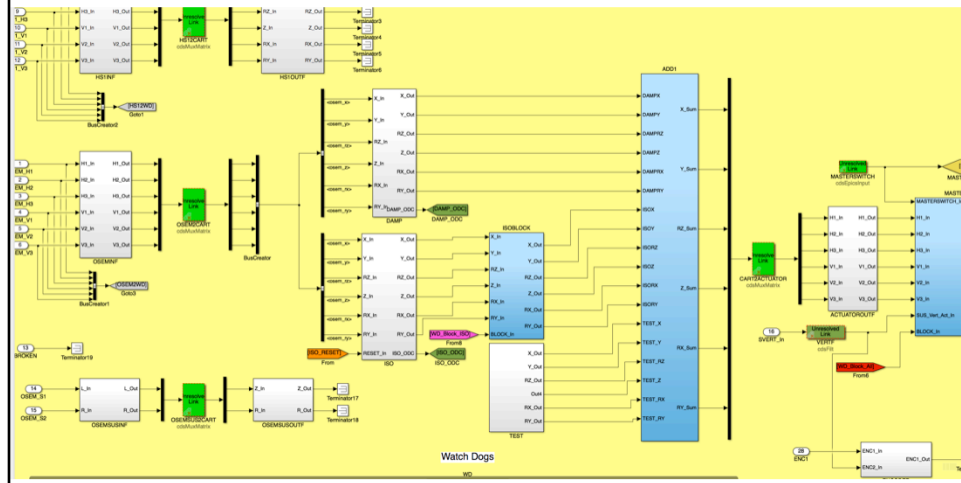
Realtime code designed in Matlab Simulink



Once you have the simulink diagram the way you like it, you compile it to C code, which is what actually runs on the front-end computer.

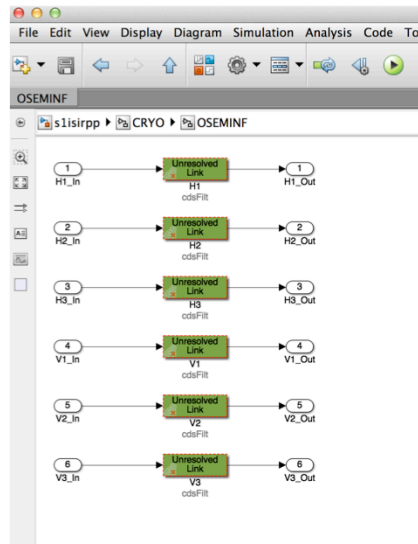
Realtime code designed in Matlab Simulink

CRYO block



If you look inside the CRYO block, you see many more sub-blocks. Each block we saw on the main MEDM screen has a corresponding block here.

Realtime code designed in Matlab Simulink



CRYO/OSEMINF block

175

If you go into one of the sub-blocks, for example the OSEM sensor compensation filters, you'll see a set of filter modules. Each of these filter modules has a corresponding medm filter module medm screen, which we saw earlier.

Making Measurements

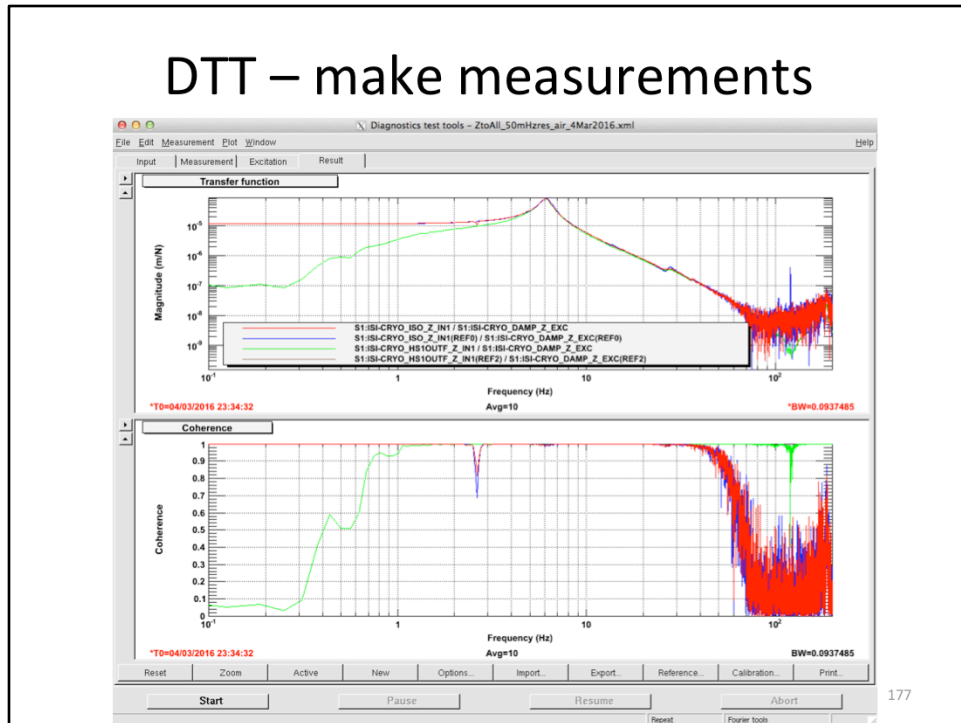
Diagnostic Test Tools (DTT) - Measure TFs and ASDs, etc

Dataviewer – time data plots in real time

176

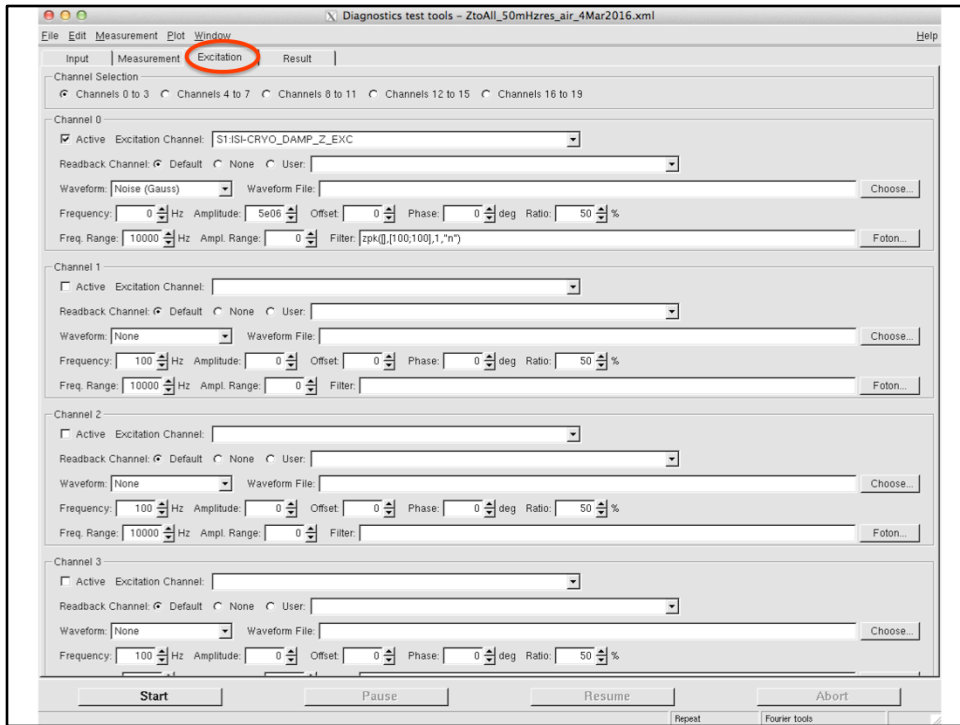
For making measurements, 2 of the most common (though not only) software tools are DTT and Dataviewer.

DTT – make measurements

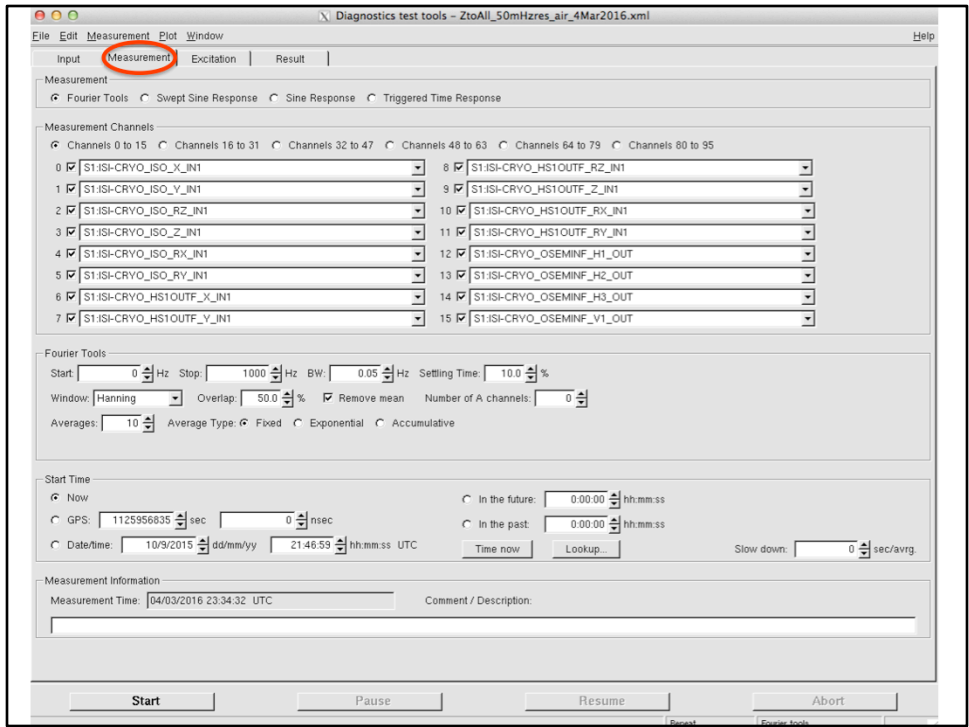


177

DTT is capable of both passive and active (send excitations) measurements. Here is an example of a transfer function measured on the cryo platform by driving the actuators and observing the sensor signals.

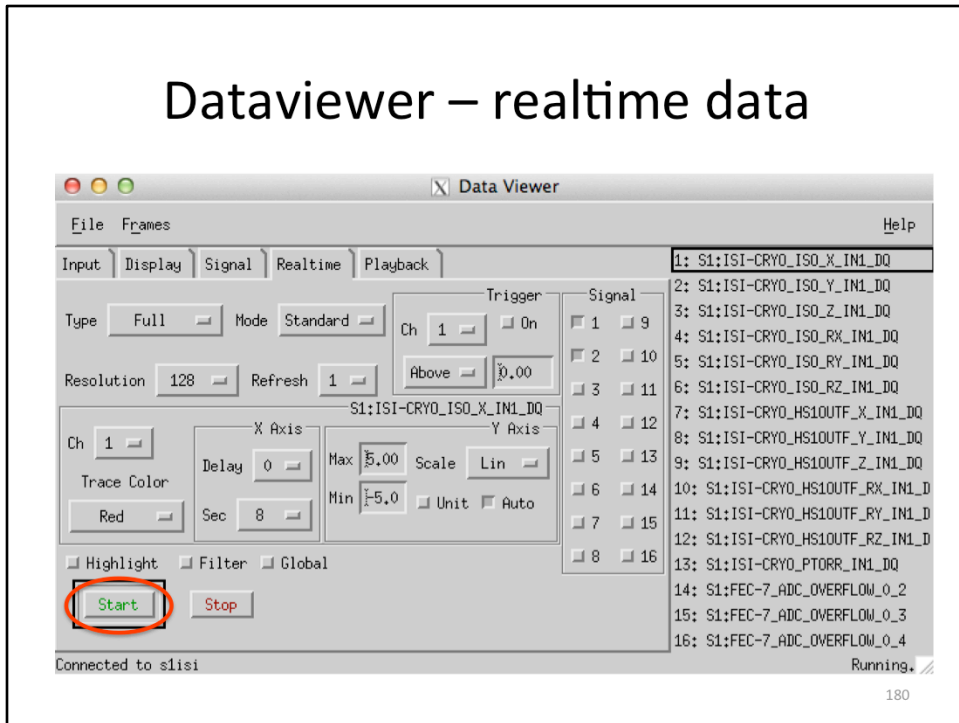


Excitations are sent by setting them up in the excitation tab. Here we are sending white noise, filtered by a 100 Hz low pass filter (2 poles at 100 Hz), to the excitation field of the CRYO_DAMP_Z filter module. This drives the actuators vertically up to 100 Hz.



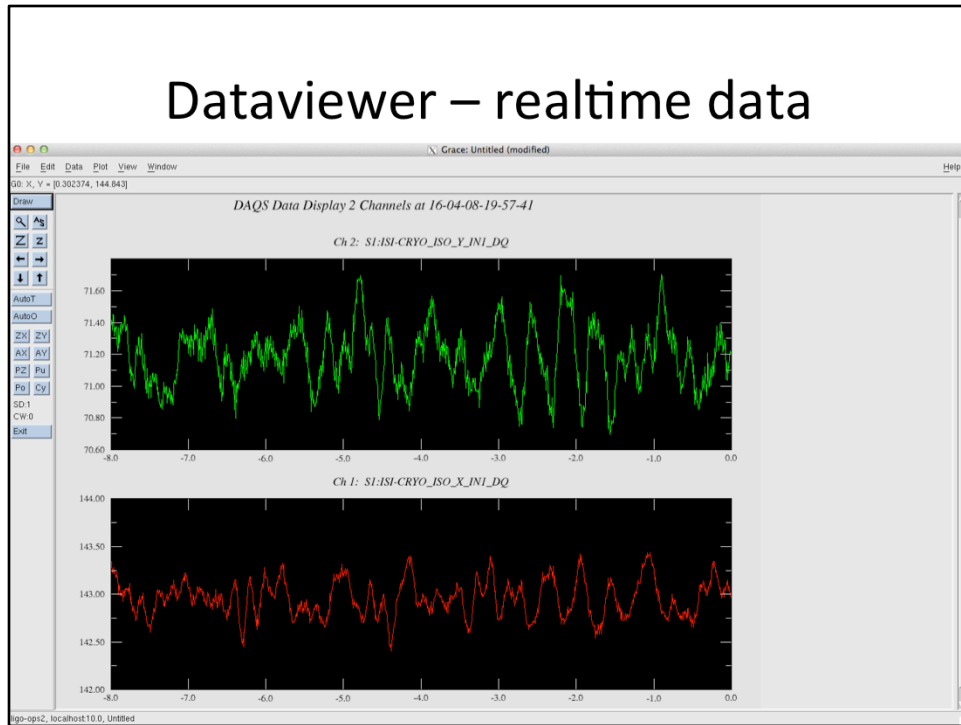
You can choose many, many channels to observe. Here the data is being collected up to 1000 Hz, with 0.05 Hz resolution (BW=frequency bin-width).

Dataviewer – realtime data



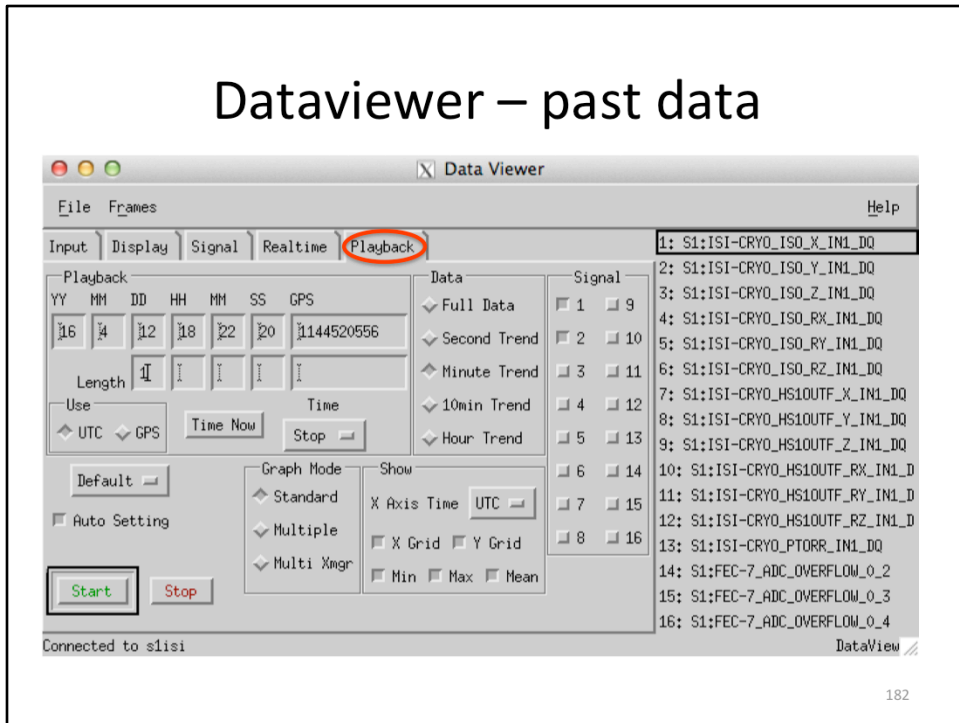
Dataviewer is sort of like an oscilloscope, and is useful for watching the realtime, time domain signals.

Dataviewer – realtime data



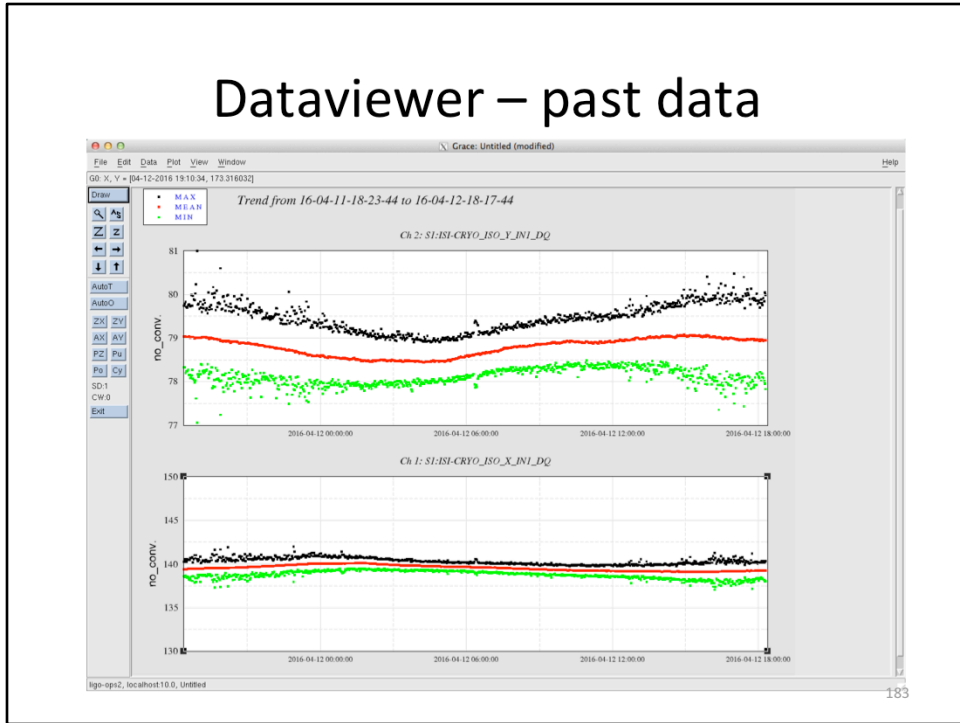
Here is an example of 2 signals being monitored in realtime with Dataviewer, with the settings shown on the previous slide.

Dataviewer – past data



Dataviewer can also plot trends from past data.

Dataviewer – past data



Here is the trend data plotted from the setting shown on the previous slide. It includes 1 day of data, with 1 minute trends. For each minute there are 3 data points displayed, the average in red, the max value in black, and the min value in green. You can do trends with 1 second resolution, 1 minute, 10 minute, and 1 hour. You can also look back at the raw data.

Lecture 3

Digital Control - Part 3: Digital time & frequency domain

G1600726

184

This part of the lecture discusses digital control theory. All the control theory we spoke of up to now is for continuous linear systems. Strictly speaking, sampled digital systems are not linear, so the same control theory does not apply. In most cases, the sample rates we use are fast enough that linear control theory is a very good approximation, and may be used. Thus, in most cases you won't need what is shown here. You may need it if you're pushing your control close to the Nyquist frequency. In any case, it is good to be familiar with it because this represents what is actually going on in the realtime computer.

Filters: Continuous to Digital Conversion

$$\dot{y} + ay = x$$

Differential equation: continuous time

$$\dot{y} \approx \frac{y(k+1) - y(k)}{dt}$$

Approximation of derivative,
where k is the current sample

$$\frac{y(k+1) - y(k)}{dt} + ay(k) = x(k)$$

Approximation of EOM

$$y(k+1) = dt[x(k) - ay(k)] + y(k)$$

Difference equation: digital

185

Continuous time filters can be converted to digital time. This is done by taking a continuous time differential equation and converting it to a digital difference equation, the digital time analog for an equation of motion. To do this, you make a sample based approximation for the derivative.

Z-Transform

Analogous to the Laplace s-transform for continuous systems

$d/dt \rightarrow s$ for continuous systems

$k+1 \rightarrow z$ for digital systems

186

To generate the frequency domain filter, we use the digital time analog for the Laplace transform, the z-transform. The Laplace transform does not apply because the system is no longer linear. The Laplace transform gives you an s for every derivate (1/s for integrals); the z-transform gives you a z for every future sample (z^{-1} for past samples).

Z-Transform

Analogous to the Laplace s-transform for continuous systems

$d/dt \rightarrow s$ for continuous systems

$k+1 \rightarrow z$ for digital systems

Digital	Continuous
Difference equation $y(k+1) = dt[x(k) - ay(k)] + y(k)$	Differential equation $\dot{y} + ay = x$

187

Here is a table comparing digital time to continuous time. In digital, we have difference equations. In continuous, we have differential equations.

Z-Transform

Analogous to the Laplace s-transform for continuous systems

$d/dt \rightarrow s$ for continuous systems

$k+1 \rightarrow z$ for digital systems

Digital	Continuous
Difference equation $y(k+1) = dt[x(k) - ay(k)] + y(k)$	Differential equation $\dot{y} + ay = x$
z-transform $yz = dt(x - ay) + y$	s transform $ys + ay = x$

188

Digital has the z-transform for the frequency domain, continuous has the Laplace transform for the frequency domain.

Z-Transform

Analogous to the Laplace s-transform for continuous systems

$d/dt \rightarrow s$ for continuous systems

$k+1 \rightarrow z$ for digital systems

Digital	Continuous
Difference equation $y(k+1) = dt[x(k) - ay(k)] + y(k)$	Differential equation $\dot{y} + ay = x$
z-transform $yz = dt(x - ay) + y$	s transform $ys + ay = x$
Transfer function $y = \frac{dt}{z + dt * a - 1} x$	Transfer function $y = \frac{1}{s + a} x$

189

Transfer functions have the z variable in the digital domain, and the s variable in the continuous domain.

Z-Transform

Digital	Continuous
Difference equation $y(k+1) = dt[x(k) - ay(k)] + y(k)$	Differential equation $\dot{y} + ay = x$
z-transform $yz = dt(x - ay) + y$	s transform $ys + ay = x$
Transfer function $y = \frac{dt}{z + dt * a - 1} x$	Transfer function $y = \frac{1}{s + a} x$
Frequency domain interpretation $z = e^{i \frac{2\pi}{f_s} f}$	Frequency domain interpretation $s = i2\pi f$

190

Adding another row, while in the continuous domain, s relates to frequency by $s = i * 2 * \pi * \text{frequency}$, in the digital domain, $z = \exp(i * 2 * \pi * \text{frequency} / \text{SamplingFrequency})$.

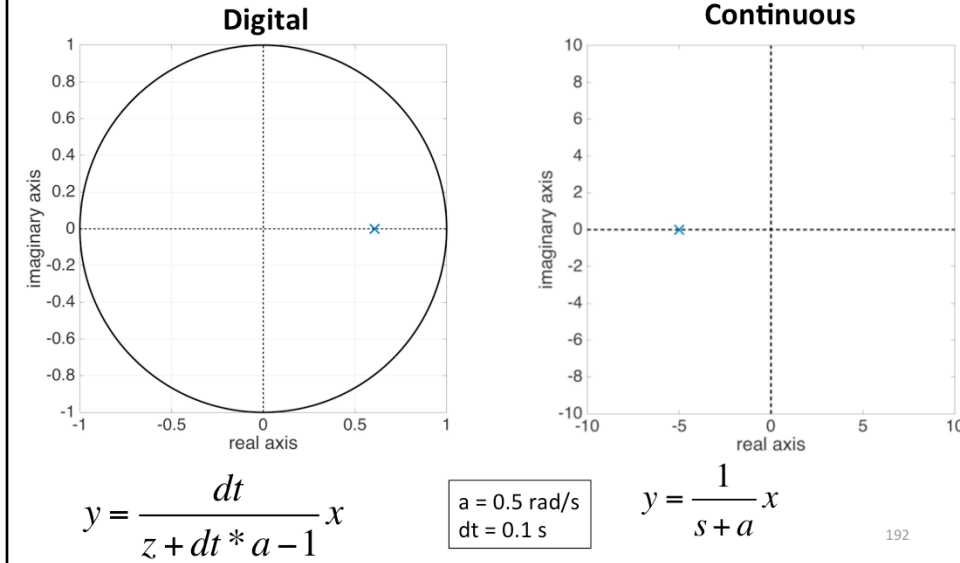
Z-Transform

Digital	Continuous
Difference equation $y(k+1) = dt[x(k) - ay(k)] + y(k)$	Differential equation $\dot{y} + ay = x$
z-transform $yz = dt(x - ay) + y$	s transform $ys + ay = x$
Transfer function $y = \frac{dt}{z + dt * a - 1} x$	Transfer function $y = \frac{1}{s + a} x$
Frequency domain interpretation $z = e^{i\frac{2\pi}{f_s} f} \approx \frac{1}{f_s} s$ where $f_s \gg f$	Frequency domain interpretation $s = i2\pi f$

191

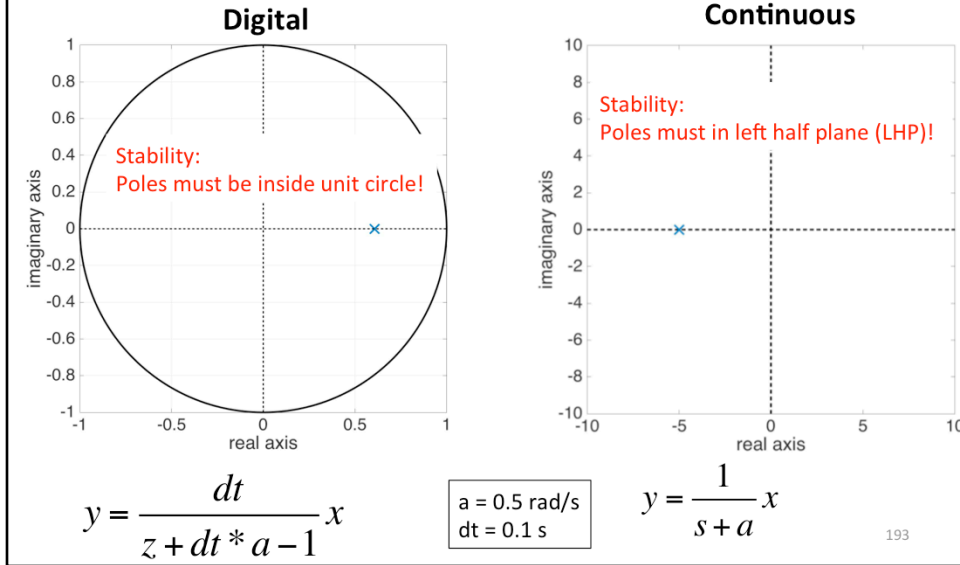
Note, that as the sampling frequency becomes very large, z starts to look more like s (normalized by the sampling frequency). In general, as the sampling frequency becomes large, the digital and continuous domains converge to the same solution (modulo some scale factors).

Complex plane pole-zero map



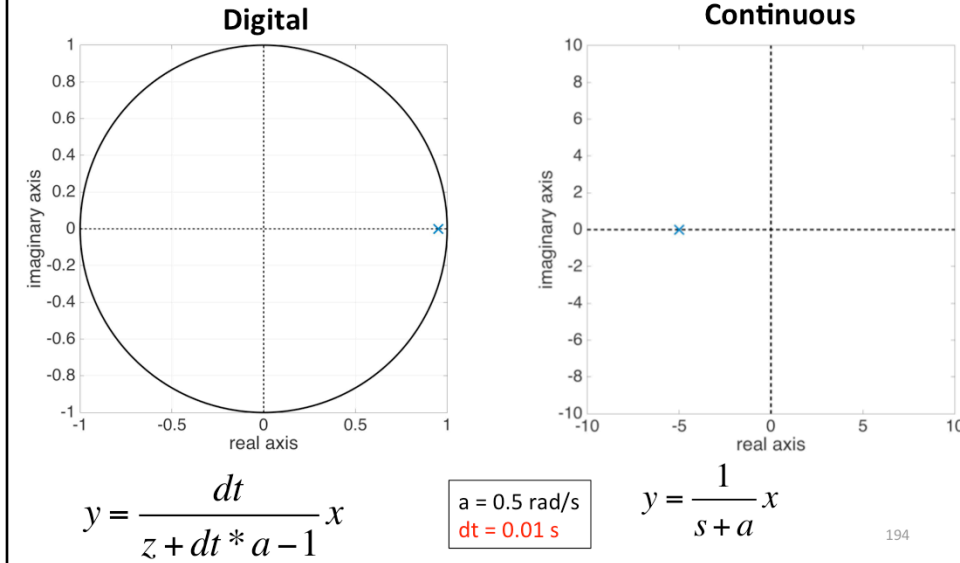
In the continuous domain, we have the complex plane, in the digital domain the complex unit circle. This slides shows the same low pass filter in its digital and continuous time equivalents.

Complex plane pole-zero map

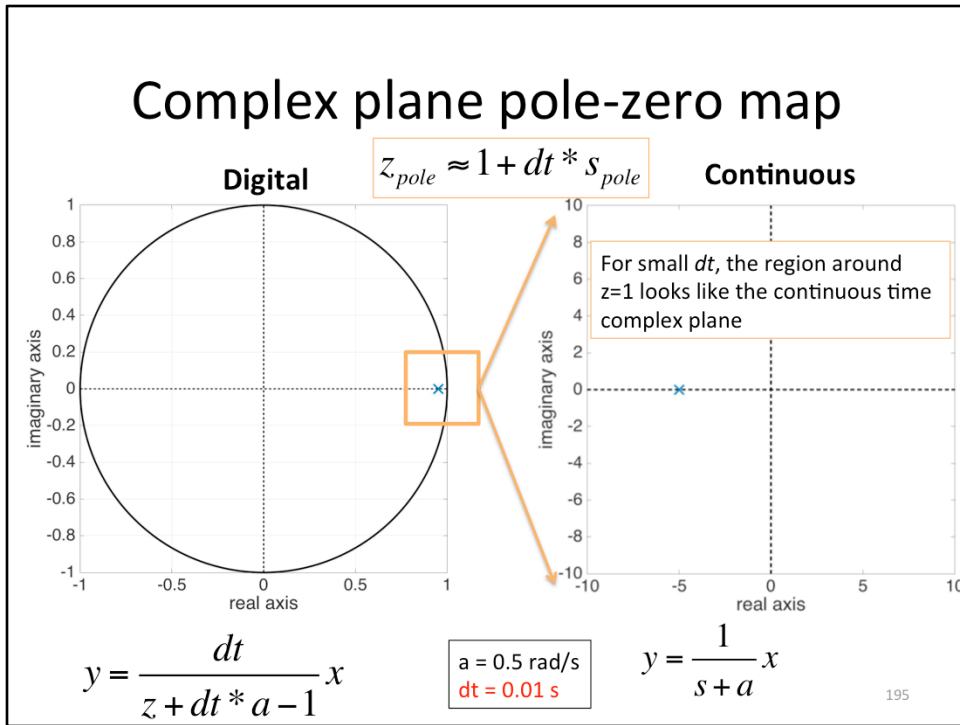


For stable continuous systems, poles must be in the left half plane (LHP). For digital systems, the poles must be inside the unit circle.

Complex plane pole-zero map



If we increase the sample rate, we see the digital pole moves closer to the $z = 1$ point. I have increased the sample rate a factor of 10 here.



As the sample rate approaches infinite, it turns out that the region around $z = 1$ approximates the continuous time complex plane (modulo some scale factors). This visualizes why continuous control theory may be used when we have sufficiently high sample rates. In general, a rule of thumb for a sufficiently high sample rate is one that is 2 orders of magnitude higher than the frequency of your poles and zeros. Beyond that you have to start taking into account the phase loss from the AA and AI filters, as well as the phase loss from the sampling itself (there is a delay between each sample, resulting in a frequency dependent phase loss, see this lecture's backup slides).

Matlab has various conversions from Laplace to Z

```

Continuous time TF HAM_ISI = 
$$\frac{1}{1900 s^2 + 3092 s + 1.258e05}$$

Digital_filter = c2d(HAM_ISI,0.001,'ZOH')           First order hold
                                                    0.001 = sample time (s)

$$\frac{2.63e-10 z + 2.629e-10}{z^2 - 1.998 z + 0.9984}$$


Digital_filter = c2d(HAM_ISI,0.001,'tustin')

$$\frac{1.315e-10 z^2 + 2.629e-10 z + 1.315e-10}{z^2 - 1.998 z + 0.9984}$$


Digital_filter = c2d(HAM_ISI,0.001,'matched')

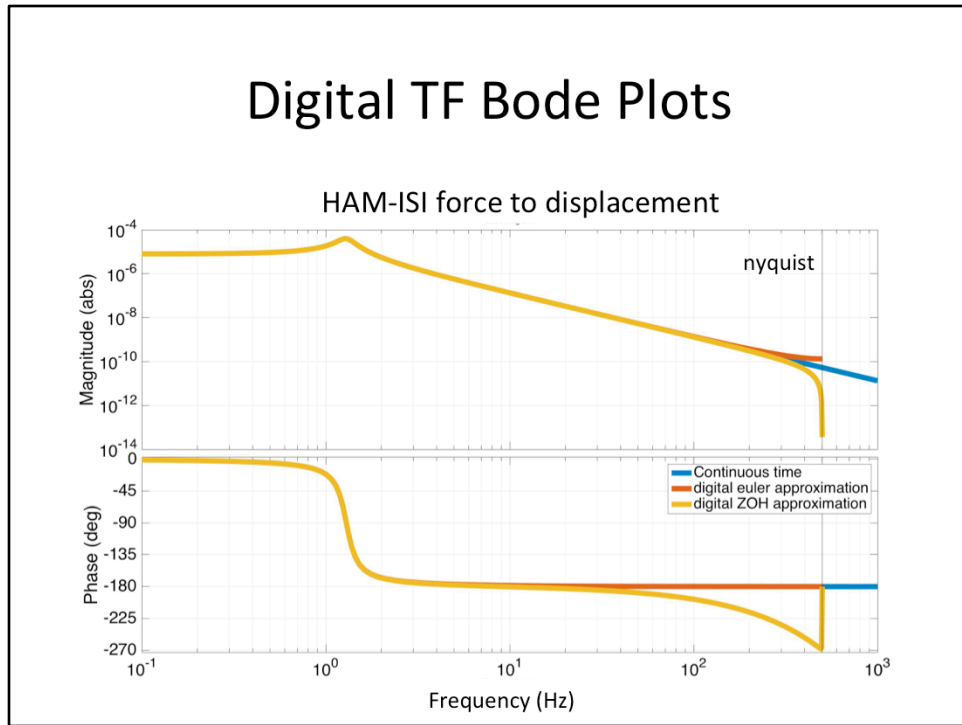
$$\frac{2.629e-10 z + 2.629e-10}{z^2 - 1.998 z + 0.9984}$$


```

196

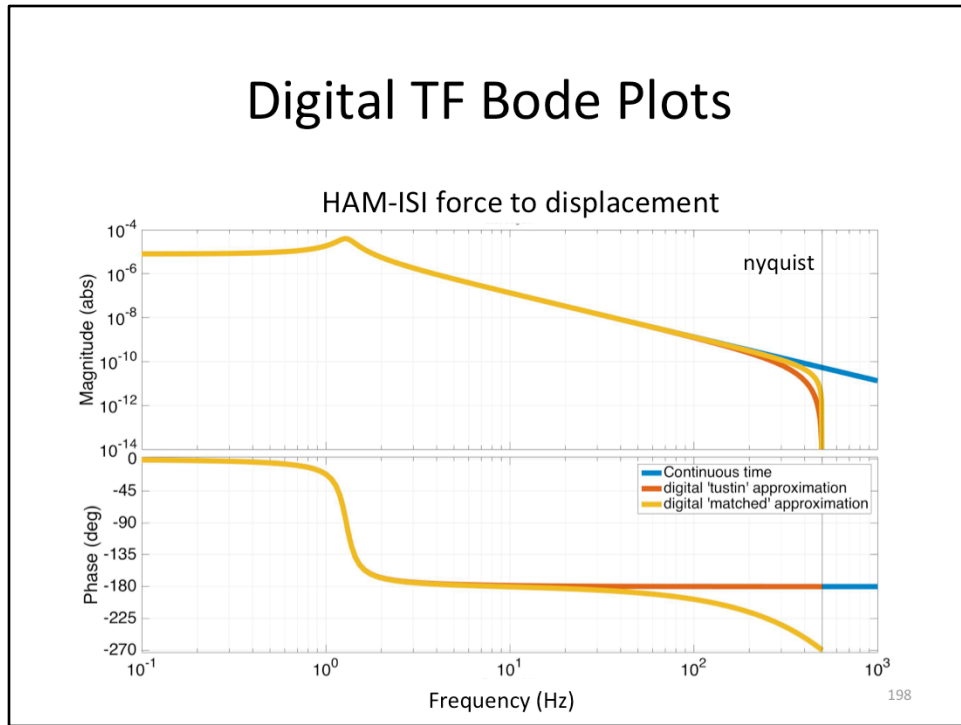
I showed a very simple conversion from continuous time to digital time when I introduced the difference equations (the simple derivative approximation). Matlab has a number of better conversions (in fact the one I showed is so naïve matlab doesn't even list it as an option). These conversions are done in Matlab with the c2d command (continuous-to-digital). The command inputs are the continuous time system, the sample time (in seconds), and the type of conversion. The third input is optional, it defaults to 'ZOH' (zero-order-hold) if not included.

Digital TF Bode Plots



Here are some bode plots of those c2d conversions, plotted against the continuous filter in blue. The red one is the naïve basic derivative approximation one I introduced with the difference equations (that matlab doesn't even include).

Digital TF Bode Plots



Here are some more bode plots of those c2d conversions, plotted against the continuous filter in blue.

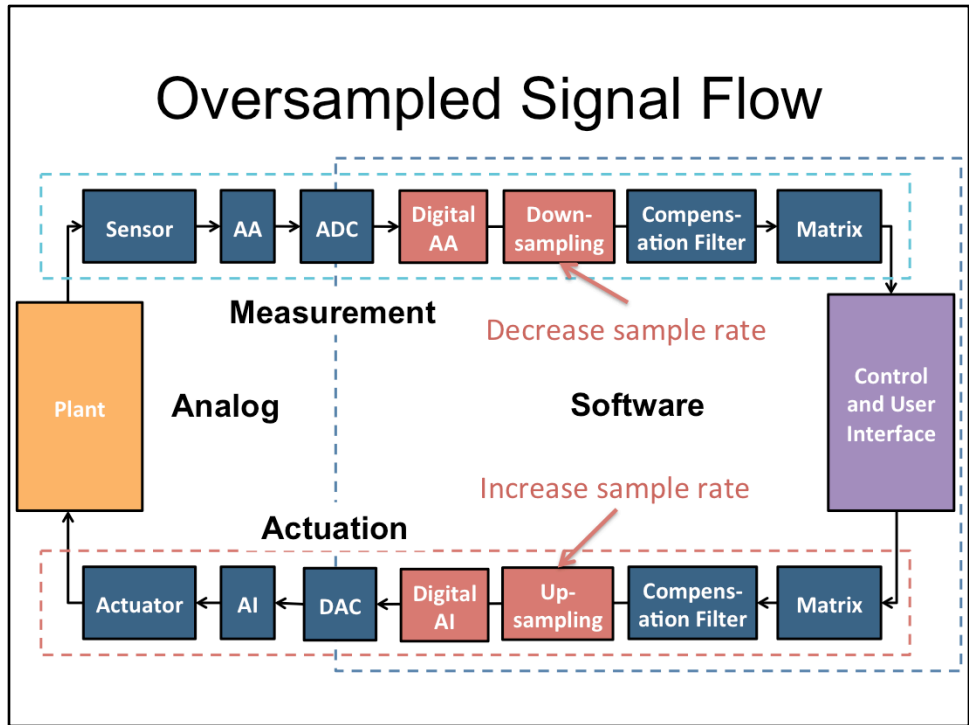
Lecture 3 Summary

- ADCs and DACs are used to sample signals and move them into and out of the computer.
- Anti-alias and Anti-image filters smooth over the transitions through the ADC and DAC to remove unwanted high frequency content.
- LIGO has various software tools for controlling and interfacing with realtime systems.
- The z-transform is the digital equivalent to the Laplace s-transform (needed since sampling is nonlinear).

Lecture 3 – Backups

G1600726

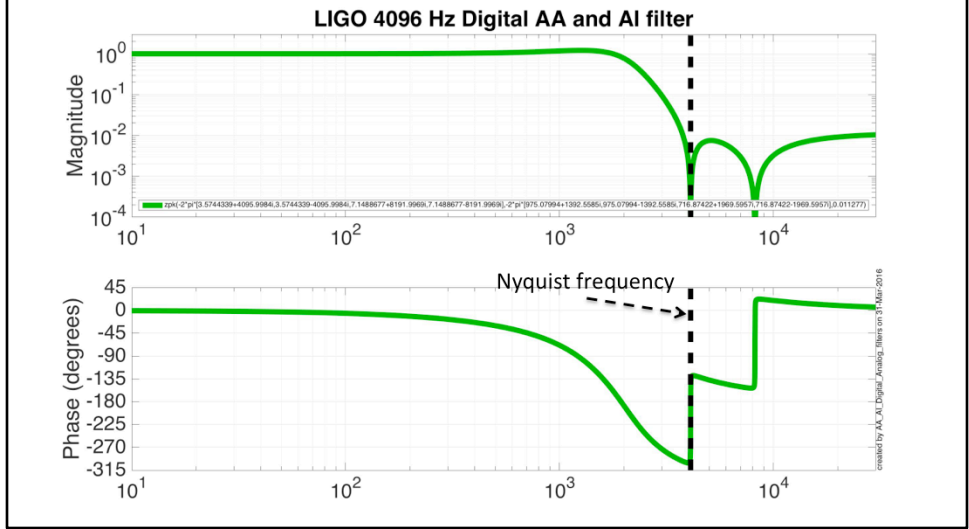
200



This is a more realistic representation of how the sampling actually occurs in LIGO systems. Because analog AA and AI filters are very simple (typically a single pole), their phase loss extends to very low frequencies. So what we do is we sample at a much higher frequency than what we actually need, so we can push the poles of the analog filters higher in frequency. This minimizes the phase loss at the frequencies we care about. At the sites, all systems are sampled at 65536 Hz, while the control systems run at 4096 Hz for the ISIs and 16384 Hz for the suspensions. The control systems need to run at the slower rate because they need sufficient time to do all the necessary computation in each clock cycle. To bridge the gap between the fast sampling of the ADC and slow sampling of the control system, we use digital AA and AI filters. Since these are digital, it is easy to make these with many poles and zeros in order to optimize their phase loss and filtering properties as much as possible. The next slide shows the digital AA and AI filter used for the ISIs.

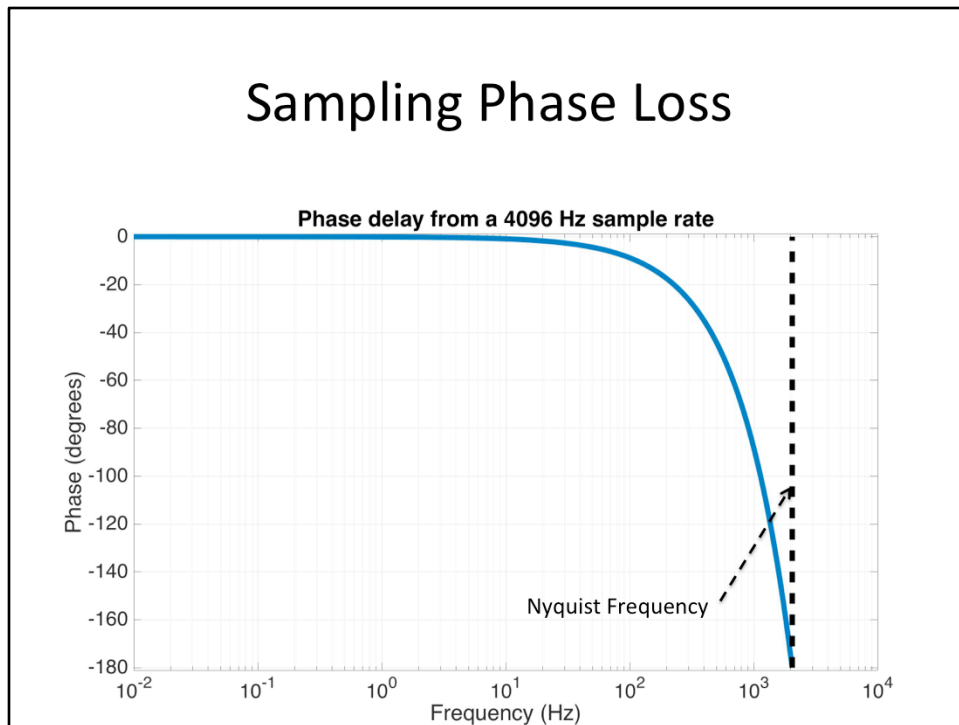
Oversampled Signal Flow

The ADC and DAC sample at 65536 Hz, the controller samples at 4096 Hz

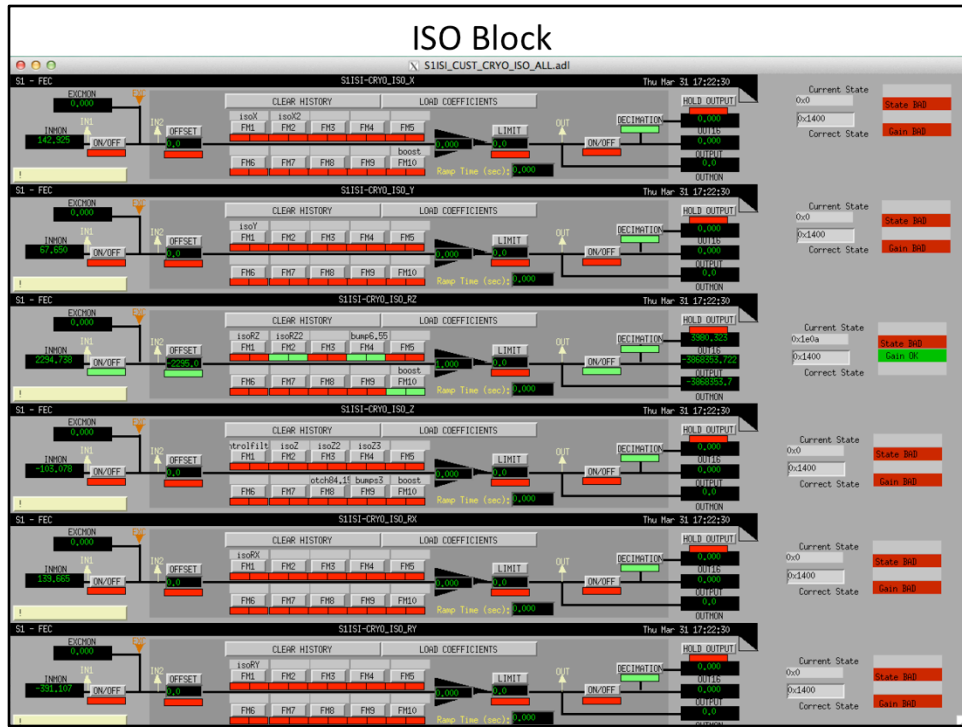


This is the digital AA and AI filter used for the ISIs (same filter for both). It bridges the gap between the 65536 Hz sampling of the ADC and DAC and the 4096 Hz sampling of the control system. This would require a relatively complex circuit design to realize with analog components.

Sampling Phase Loss



Sampling causes a phase delay because you must wait for the next sample time before the control system updates. This phase, in degrees, is $-360 \cdot \text{frequency} / (\text{sample frequency})$; that is just the time between samples converted into phase for a given frequency. Note, you will have additional phase loss from both the anti-alias and anti-imaging filters.



Example medm screen for an isolation block. Note the guardian states on the right.



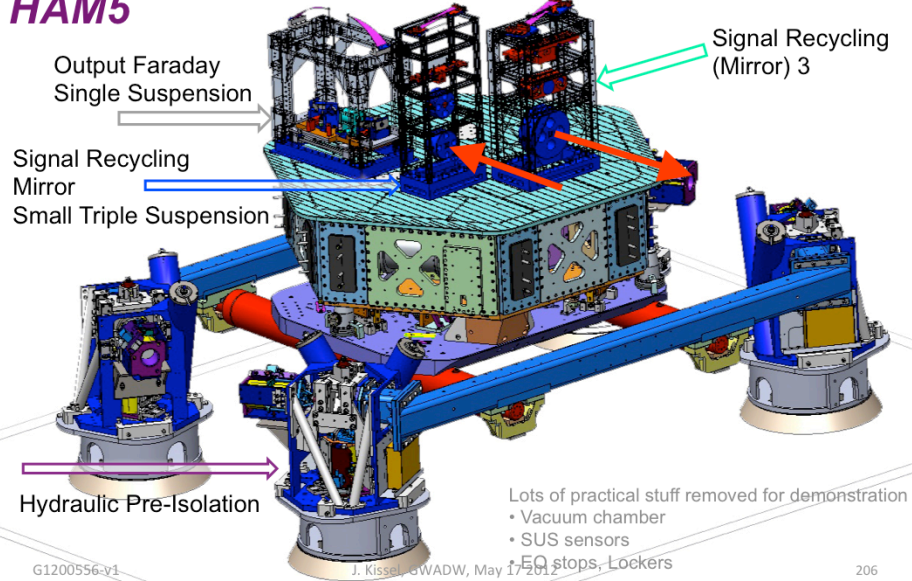
LIGO

General BackUps

Advanced LIGO

A single output chamber is complicated!

HAM5

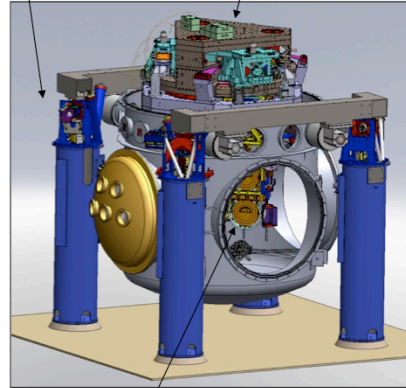


BSCs – core optics



hydraulic external pre-isolator (HEPI) (one stage of isolation)

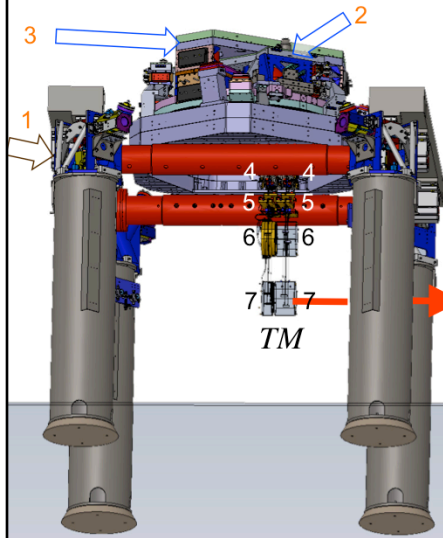
active isolation platform (2 stages of isolation)



quadruple pendulum (four stages of isolation) with monolithic silica final stage

Hybrid Systems

Advanced LIGO - The Design



- 7 Stages of Isolation
 - Hydraulic Preisolation
 - Blade spring and wire flexures
 - Monolithic Final Stage
- 6 DOF sensing on stages 1 – 4, 3 DOF on 5 – 6
 - Inertial and displacement on stages 1-3
 - Displacement only on stages 4 - 6
- 6 DOF DC - 1kHz actuation on Stages 1 – 4, 3 DOF on 5 - 7
- $(6+6+6+[3*6+4]) = 40$ out of 42 Trans./Rot. resonant modes sensed and controlled
- Many-control-loop system
 - Sensor blending, Feed back, Feed forward, Sensor Correction, Heirarchical control
- Versatile 800 kg payload
- Stage 1 – 3 “Performance limited by sensor noise,”
- Stage 4 – 7 “Performance limited by direct transmission of platform motion”

G1200556-v1

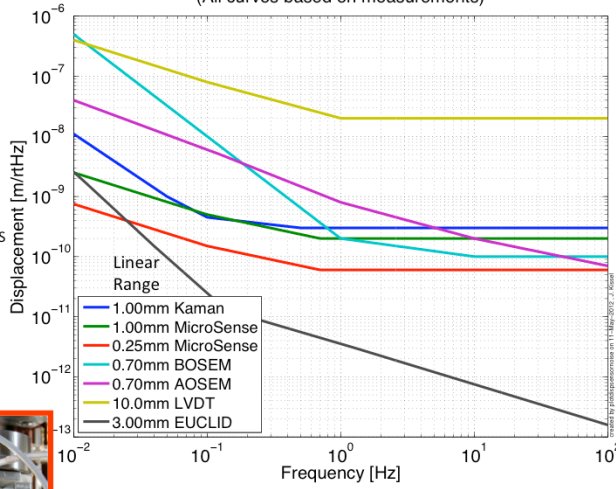
J. Kissel, GWADW, May 17 2012

208

Sensor Noise

Displacement Sensors

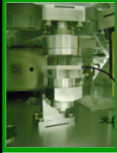
Current and Future Displacement Sensor Noise
(All curves based on measurements)



Inductive
aLIGO Pre-isolator
Stage



Inductive
VIRGO / GEO / KAGRA SAS



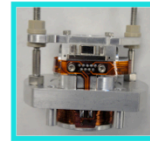
Capacitive
aLIGO Stage 1
ISIs



Capacitive
aLIGO Stage
2 ISIs

G1200556-v1

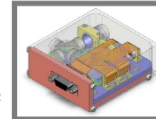
J. Kissel, GWADW, May 17 2012



Shadow Sensor
aLIGO SUS Top Stages
AEI 10m SUS Top Stages



Shadow Sensor
aLIGO SUS Lower
Stages



Interferometric,
U of Birmingham
Prototype (As yet
Non-UHV Comp.)

Sensor Noise Inertial Sensors

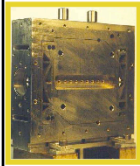
Current Inertial Sensor Sensor Noise
(Some Specs, Some Measurements)



aLIGO Pre-isolator Stage
aLIGO Stage 0 & 1 ISIs
AEI-SAS Vert. Witness

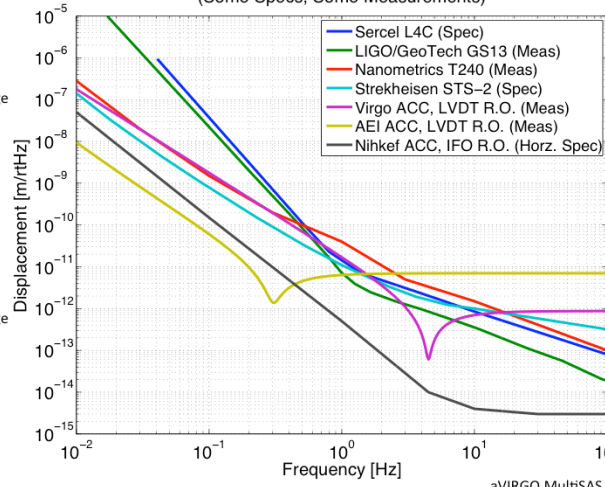


aLIGO Pre-isolator Stage



AEI-SAS Horz. Witness
(Watt Linkage)

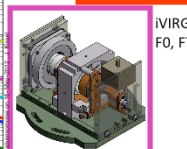
G1200556-v1



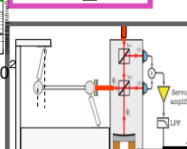
aLIGO
Stage 1
& 2 ISIs



aLIGO
Stage 1
ISIs



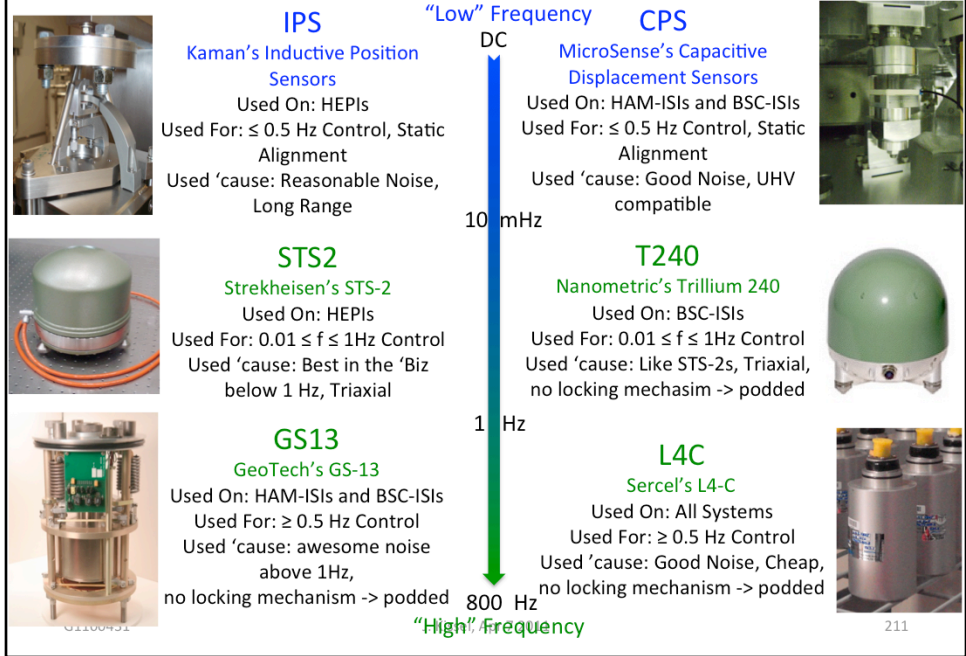
iVIRGO
F0, F7



aVIRGO MultiSAS
Horz. Witness
(Watt Linkage)

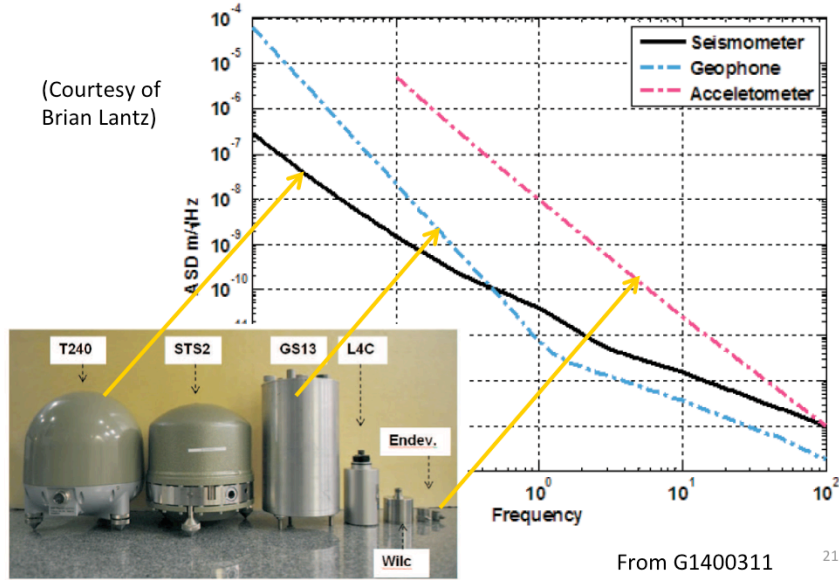
J. Kissel, GWADW, May 17 2012

SEI Sensors and Their Noise



Sensor size and noise

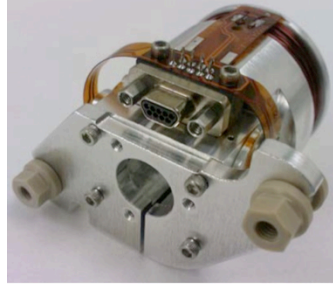
(Courtesy of Brian Lantz)



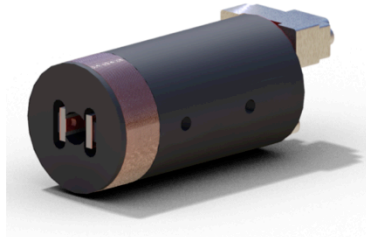
From G1400311

212

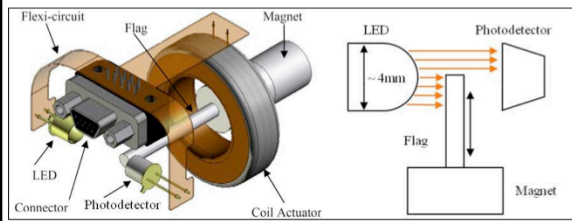
Optical Sensor ElectroMagnet (OSEM)



Birmingham OSEM (BOSEM)



Advanced LIGO OSEM (AOSEM)
- modified iLIGO OSEM



BOSEM Schematic

24 Aug 2014 - Stanford - G1400964

Magnet Types (M0900034)

- BOSEM – 10 X 10 mm, NdFeB, SmCo
10 X 5 mm, NdFeB, SmCo
- AOSEM – 2 X 3 mm, SmCo
2 X 6 mm, SmCo
2 X 0.5 mm, SmCo

213

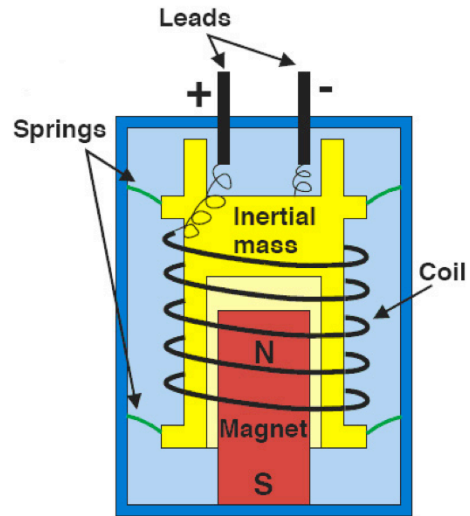
HS1 Geophones



<http://www.geospace.com/hs-1-industrial-geophone/>

214

Geophone Schematic



Source: http://newsline.linearcollider.org/readmore_20070809_ftr1.html

215