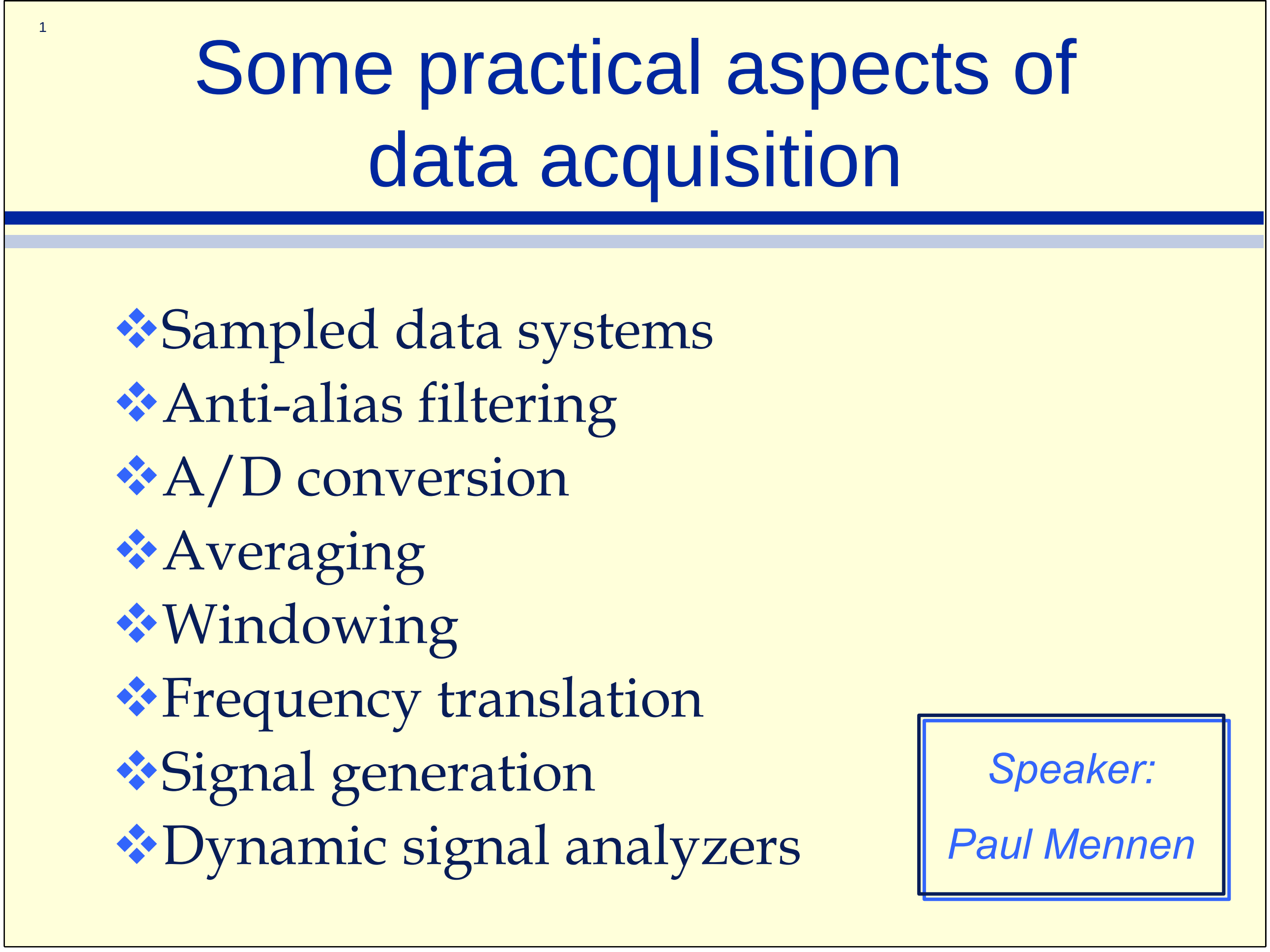


1

Some practical aspects of data acquisition

- ❖Sampled data systems
- ❖Anti-alias filtering
- ❖A/D conversion
- ❖Averaging
- ❖Windowing
- ❖Frequency translation
- ❖Signal generation
- ❖Dynamic signal analyzers

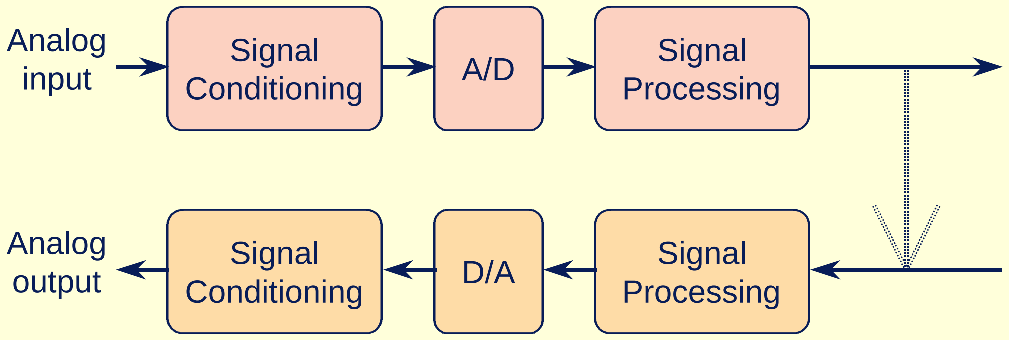
Speaker:

Paul Mennen

1st I'll talk about sampled data systems in general, then more detail on these six topics.

Then I'll talk about the analyzers that employ these ideas. Since I have built and supported these analyzers, I been exposed to many of the practical concerns and problems that are not often found in books. So I would like to share some of those with you today.

Sampled Data Systems



- ❖ Examples of sampled data systems:
  - ❑ CD player
  - ❑ Disk Drives
  - ❑ Fax/Modem
  - ❑ Instrumentation

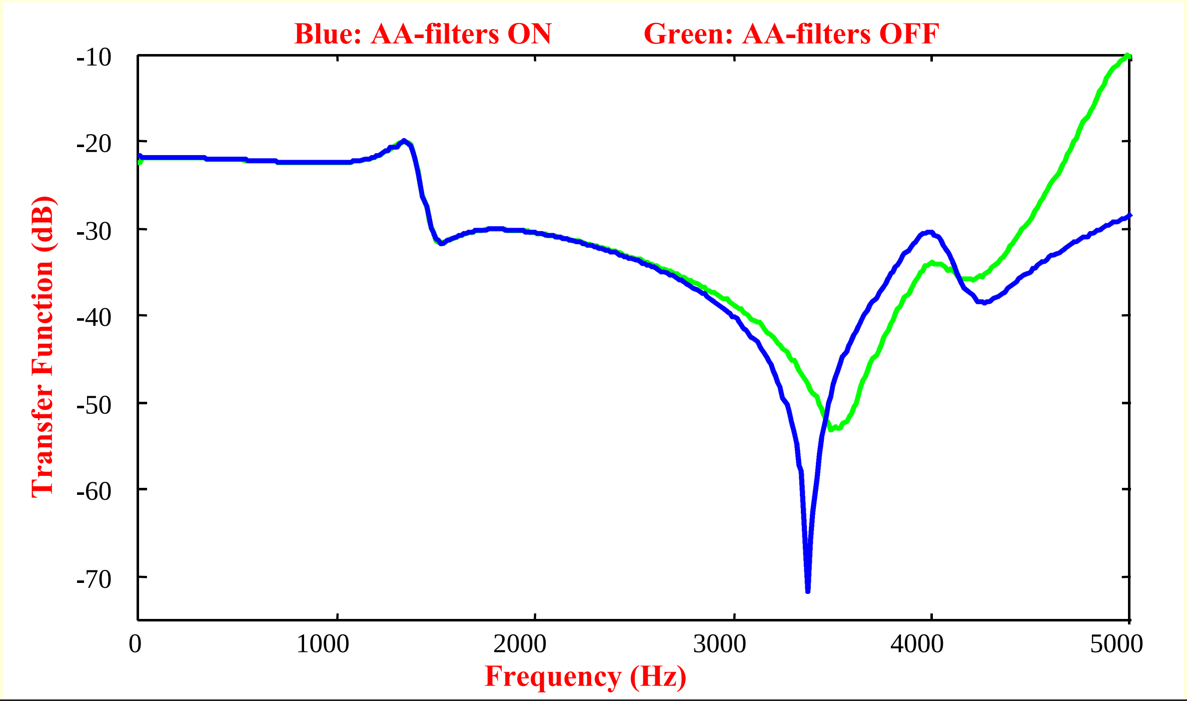
A sampled data system has either this acquisition section or this signal generation section or both.

Acquisition
- Signal conditioning: Preamps, bias, AA filters, etc.
- A/D: Discretize signal in both time & amplitude
- Signal Processing: DFT and more

Generation
- Signal Processing: data from external source or from acquisition
- D/A: Convert signal to analog form (zero order hold)
- Signal conditioning: Smoothing filters, buffers, power amps, etc

Examples
- CD player: only bottom (generation) half
- Fax modem: send & receives data to/from analog phone lines
- Disk drives: 1.) RW head (data)
  2.) RW head positioning control system.
- Instruments: Sampled data systems are replacing many older style analog instruments as digital technology advances in cost and performance.

Anti-aliasing: Do you believe?



I've heard people say that they will be able to tell an alias when they see one, so it's not worth the expense to have AA filters.

Of course it is not possible to tell. But if you are not convinced, look at this transfer function of an electrical network:

<u>Blue trace</u>: with AA filters (accurate). Substantial notch at 3.3kHz.

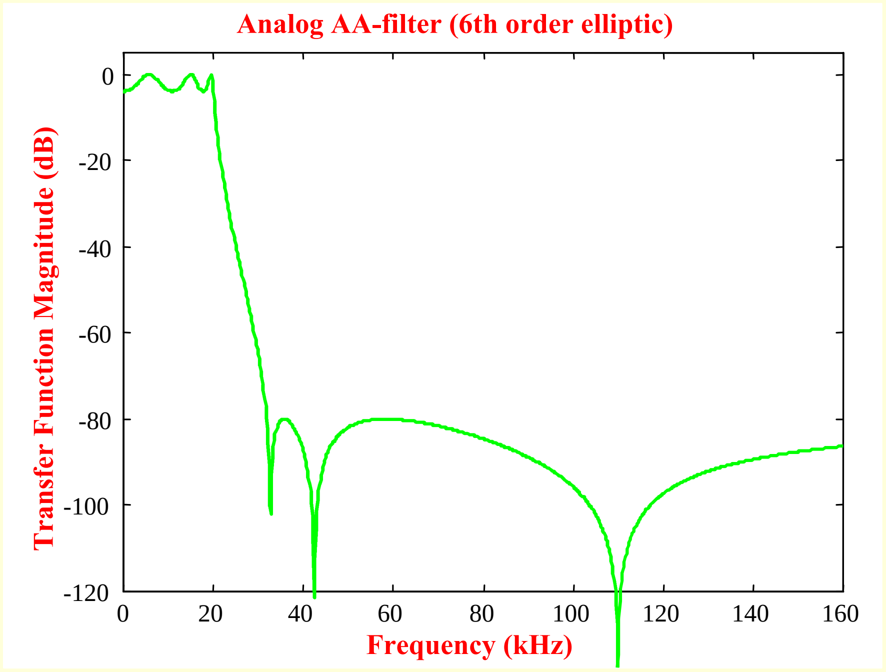
<u>Red trace</u>: without AA filters. Not only is the notch not nearly as deep, but it shifted frequency as well. The error at the high end is nearly 20dB. Obviously without AA filters, you would have no way to know that this transfer function estimate was bogus.

Often the lack of AA filtering will cause a jagged appearance to the transfer function and/or low coherence. Not always, however. In this case, averaging smoothed out the erroneous estimate and coherence was near 1.

If audience isn't familiar with Xfer: Xfer is the ratio of the power of the system output to the power of the system excitation measured at each frequency.

Anti-aliasing: Filter specifications

- Passband ripple
- Stopband rejection
- Efficiency



Here is an example of a typical analog AA filter (for use with a conventional A/D converter, not a sigma-delta as in SigLab).

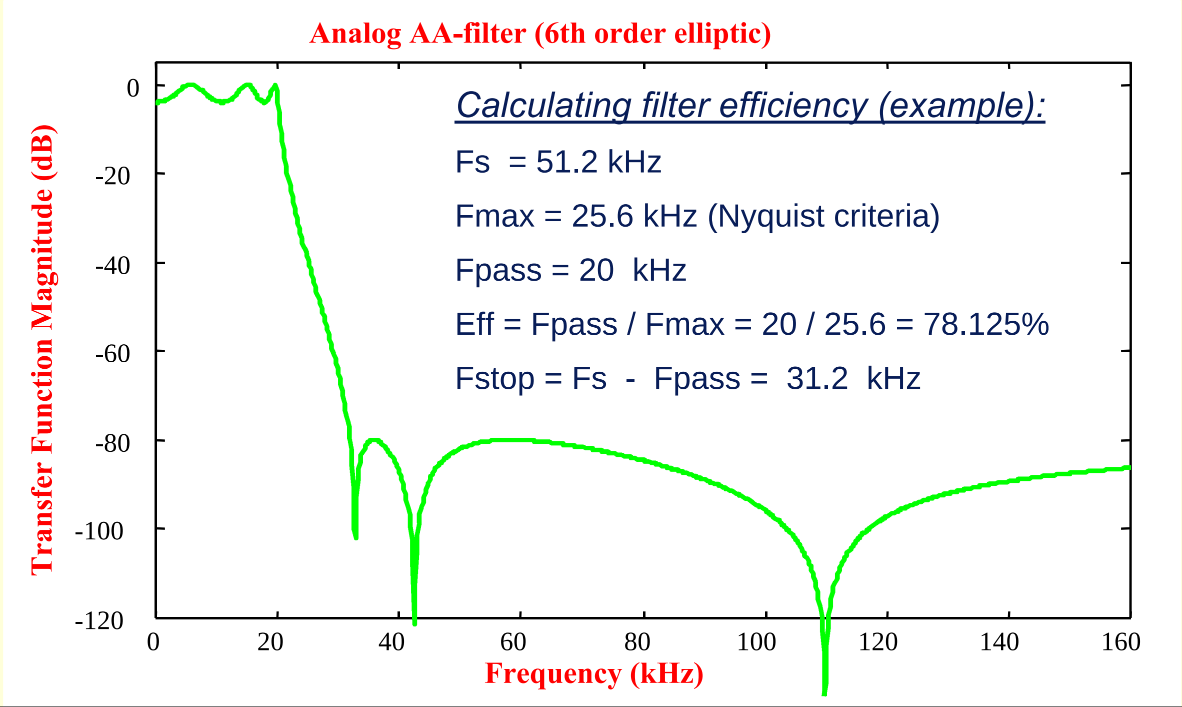
These are the important characteristics of AA-filters.

Passband ripple: This example shows several dB of ripple, which is larger than you would normally want in a high quality front-end. The ripple in the passband degrades the amplitude accuracy spec. To do this (all other things being equal) we would need to use a larger order filter.

Stopband rejection: About -80dB in this example. More is better. Dynamic range can never be better than the amount of stopband rejection.

Efficiency: The 3rd and last important parameter relates to how steeply the response transitions from the pass band to the stop band. Usually specified in terms of rolloff in dB/octave or dB/decade. A more useful way of specifying this parameter I will call “efficiency”. Now, overlay the next slide on top of this one.

Filter efficiency



Calculating filter efficiency (example):

Fs = 51.2 kHz

Fmax = 25.6 kHz (Nyquist criteria)

Fpass = 20 kHz

Eff = Fpass / Fmax = 20 / 25.6 = 78.125%

Fstop = Fs - Fpass = 31.2 kHz

In this example, the sample rate was chosen to be 51.2 kHz.

[This may seem like a strange choice, but it is common. This is because 512 is a power of 2. Since the frame size is usually chosen to also be a power of two (for the FFT), the resulting frame times and frequency bins turn out to be nice even numbers.]

Nyquist tells us that we must sample twice as fast as the highest frequency contained in the signal. Thus the passband may not go higher than 25.6 kHz. In this example, Fpass = 20kHz, so about 78% of this available bandwidth is used, and the efficiency is 78%.

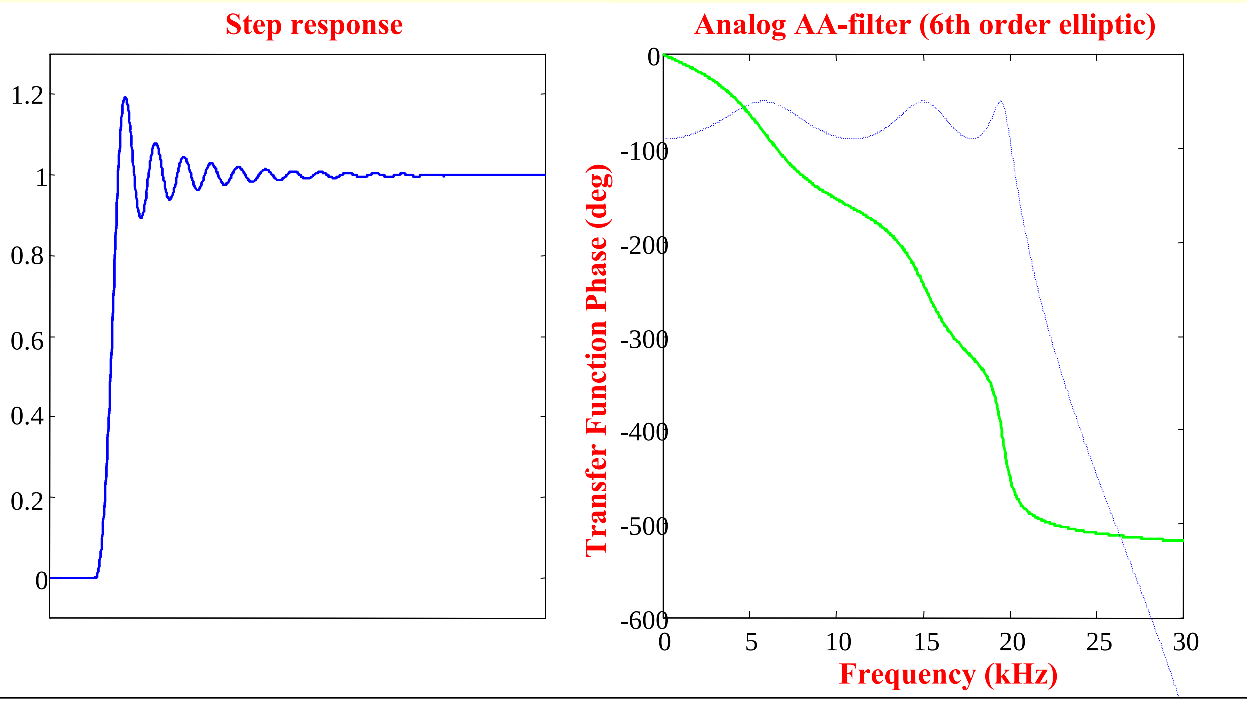
Now it's useful to calculate where the stopband must begin to ensure the aliases are filtered out: Fstop = Fs - Fpass = 51.2 - 20 = 31.2 kHz.

So the filter attenuation must be equal to or greater than the required alias protection at all frequencies greater than 31.2 kHz.

What happens if we try to use all the available bandwidth:

Fpass = 25.6kHz, Fstop = 51.2 - 25.6 = 25.6 kHz = Fpass. Thus we would need a brick wall filter which of course is impossible to build.

Anti-aliasing: Do you always want it?



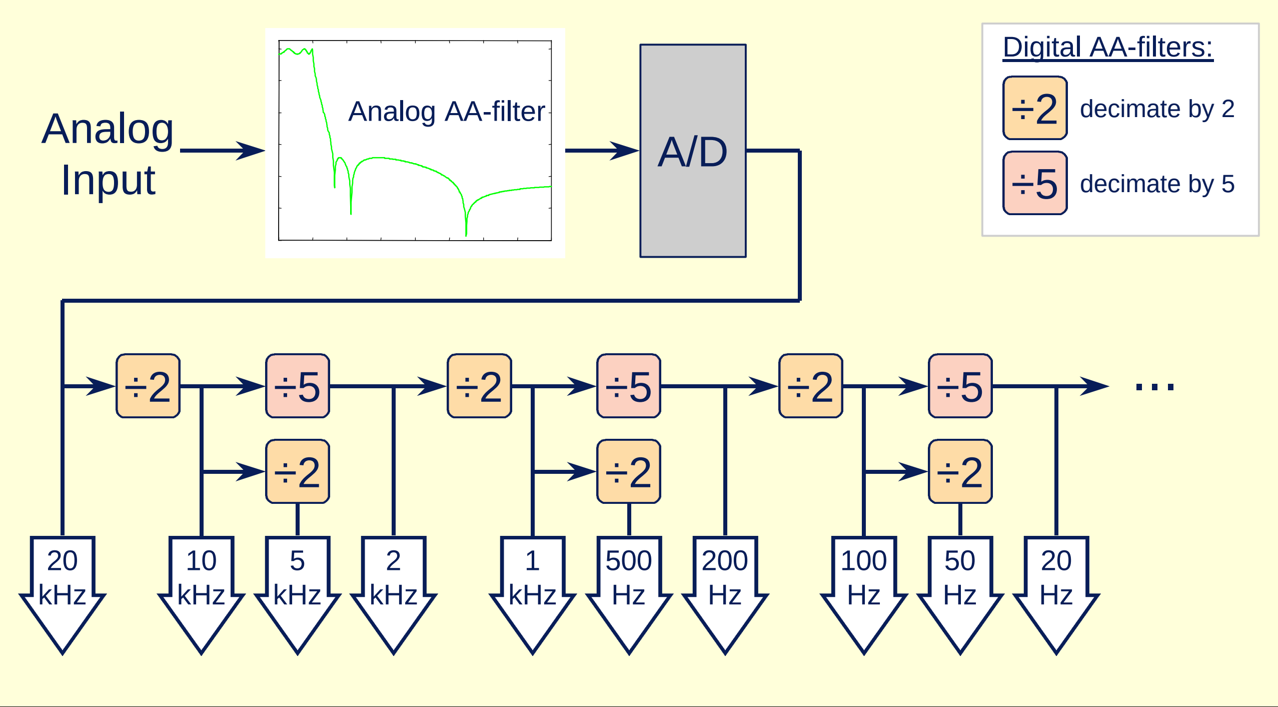
AA-filtering has its drawbacks as well. If you are making purely time domain measurements, such as estimating time delay or overshoot, then the AA-filter makes the job more difficult.

Here is the step response of the AA-filter shown previously. The filter overshoot would make it difficult to estimate the overshoot of the system being measured.

For reference, the blue line is the passband magnitude response of the AA-filter. The green line is the filter's phase response. Notice that it is not a straight line. This non-linear phase which is characteristic of efficient AA-filters also distorts the time history of the signal being measured (but does not affect spectral measurements).

For these reasons, many dynamic signal analyzers provide the ability to bypass the AA-filtering. But when you do, beware. As many of you know from using digital oscilloscopes, occasionally you may completely misinterpret the display as an unsuspected alias comes home to roost.

AA-filtering: Fixed sample rate method



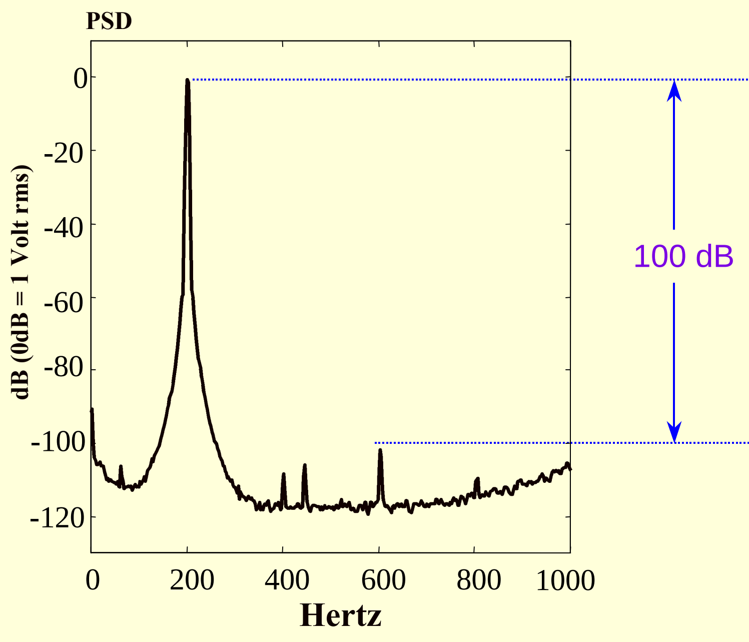
One problem for an instrument desinger is that the AA-filter passband cutoff frequency must change depending on the bandwidth required for each specific measurement.

Traditionally this was solved by building the filter with components that can be switched in value using relays or FET switches. For high performance AA-filters this is a difficult design job and an even more difficult manufacturing problem.

Another approach is called a switched capacitor filter, where the passband frequency can be adjusted by controlling a clock rate. However SCFs are usually limited to lower quality SDSs because of their high noise and modest achievable levels of stop band rejection.

The more modern approach, used when high performance is required, is called a fixed sample rate system. An analog AA-filter is used with a fixed cutoff frequency at the highest sample rate allowed by the A/D. If lower samples rates are required, digital decimating low pass AA-filters are used to reduce the sample rate. Usually this filtering is designed using multiple stages of decimate by 2 and decimate by 5 filters. This is more efficient computationally than computing the entire filter in a single stage. Each filter down the chain gets easier because the sample rates get lower and lower.

A/D conversion: What is dynamic range?



The ratio of the maximum signal to the largest spurious response.

Moving one step down the chain: A/D conversion.

Leads to an important spec of data acquisition systems: Dynamic range

Assume we connect up an ultra-pure sine wave to the input. Theoretically the power spectrum should show only one peak. Actually we will see other spurious signals as well. The size of these spurs determine the dynamic range of the acquisition system.

There is no one commonly accepted definition of dynamic range, but this is the one I find most useful.

A/D conversion: Quantization noise

- ❖Quantization noise amplitude (σ) is proportional to A/D step size (2^{-b})
  - ❑SNR $\propto 2^b$
  - ❑$\text{SNR}_{\text{dB}} \propto 20 \log_{10}(2^b)$
  - ❑$\text{SNR}_{\text{dB}} = 6.02\, b + C$
- ❖Often stated as 6 dB of dynamic range per A/D bit (myth!)

Quantization noise is a measure of the amplitude error introduced by the A/D because its output can take on only a finite number of values.

Suppose the A/D has “b” bits. Then the A/D can take on 2^b different values. Thus the step size and the quantization noise (in terms of its standard deviation) is proportional to 2^{-b}.

So if the noise is proportional to 2^{-b}, the SNR is proportional to 2^b.

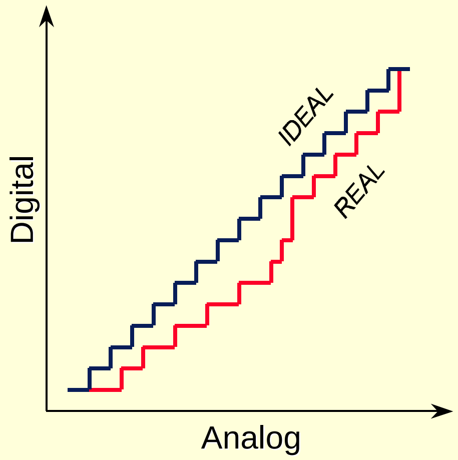
Taking the log of both sides we get: SNR = about 6 dB per bit plus this constant C to account for the proportionality.

Most books take many pages to derive this formula. Most of the complications have to do with figuring out what the constant C is. However most people ignore C, since it is negligible once we are up to a handful of bits. Also different answers are obtained depending on the assumptions made.

Most people interpret this as meaning that the acquisition dynamic range is simply 6 dB times the number of A/D bits. Wrong! The dynamic range can be much worse because of other errors in the system. Also the dynamic range can be much better since the averaging effects of the FFT and the digital filters reduce the effects of the quantization noise.

Limitations to instrumentation dynamic range

- A/D linearity



- Signal conditioning linearity
- AA-filter stop band
- Digital and power supply noise
- External noise
- Sample clock jitter
- A/D aperture error

So if quantization noise is not the primary limit to dynamic range, what is? The two most likely limitations to the dynamic range are the A/D linearity and the linearity of the signal conditioning circuits.

A/D linearity: Curve in black is for an ideal 4 bit converter. Curve in red shows a realizable converter (exaggerated). Each step of the ideal transfer function is exactly the same size. Such an ideal converter (if you could build such a thing) would pose no limitation to dynamic range. The waviness of the red curve represents the nonlinear behavior. Note in this example there are also two missing codes.

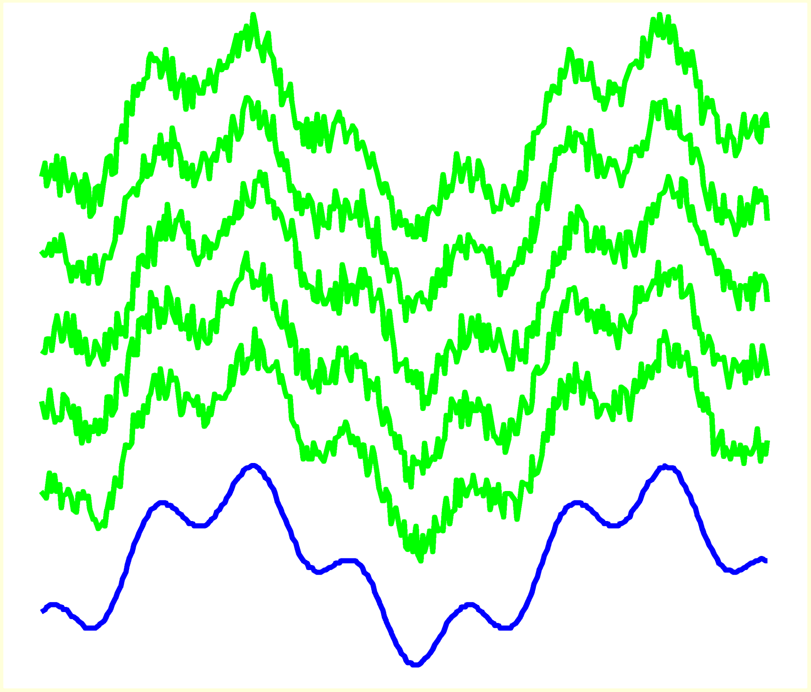
The nonlinearities of the pre-amplifiers or analog AA-filters are just as likely to limit the dynamic range. Also the dynamic range of the measurement can't be considered greater than the amount of rejection in the AA-filter stopband.

Noise from the power supply or digital circuits can leak into the signal conditioning circuits causing spurs which limit dynamic range. Similarly noise terms can come from external sources such as a CRT monitor sitting on top of the equipment.

The assumption of the FFT is that the signal is uniformly sampled. Clock jitter (usually a problem only with external sampling) violates this assumption and can be viewed as a non-linearity.

With modern sample and hold circuits, the A/D aperture error is usually small enough not to be a factor.

Averaging: Time domain



- ❖ Must have a stable trigger
- ❖ Also known as synchronous time averaging

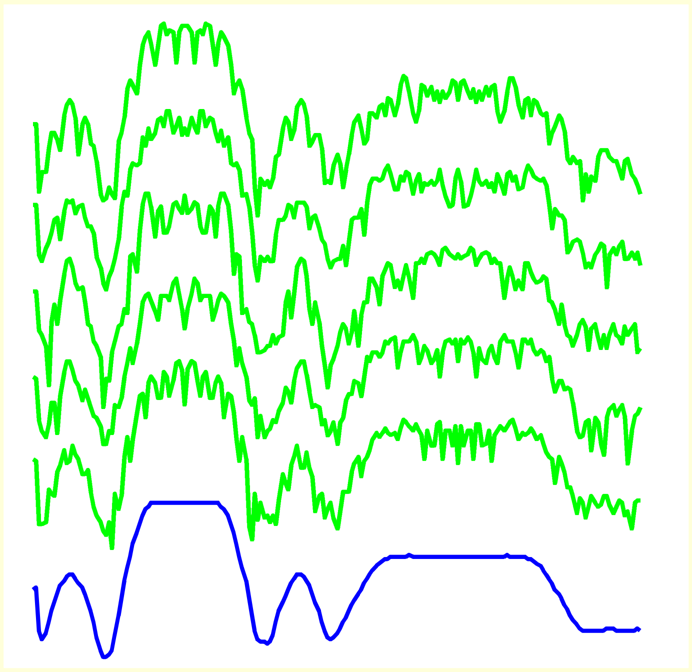
Moving down the chain again: Averaging.

There are two types of averaging used in a dynamic signal analyzer: Time and frequency domain.

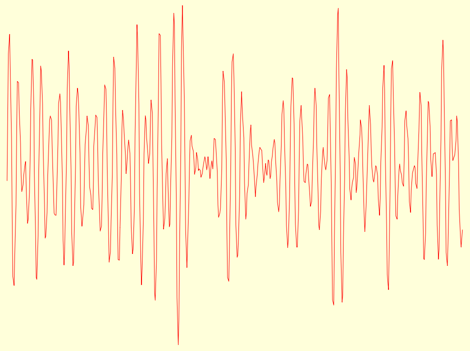
Here is an example of time domain averaging. These five traces, are instantaneous (i.e. not averaged) time histories of a repeating signal. When we add them up point-by-point (i.e. column-wise) the result is seen to be much smother.

For this to work, we must have a stable trigger so that the picture doesn’t jump around in the time axis. Without a trigger, the time domain average (the signal and the noise) will eventually average out to zero.

Averaging: Frequency domain



- PSDs of a single frame show high variance.
- PSD variance is reduced with frequency domain averaging.

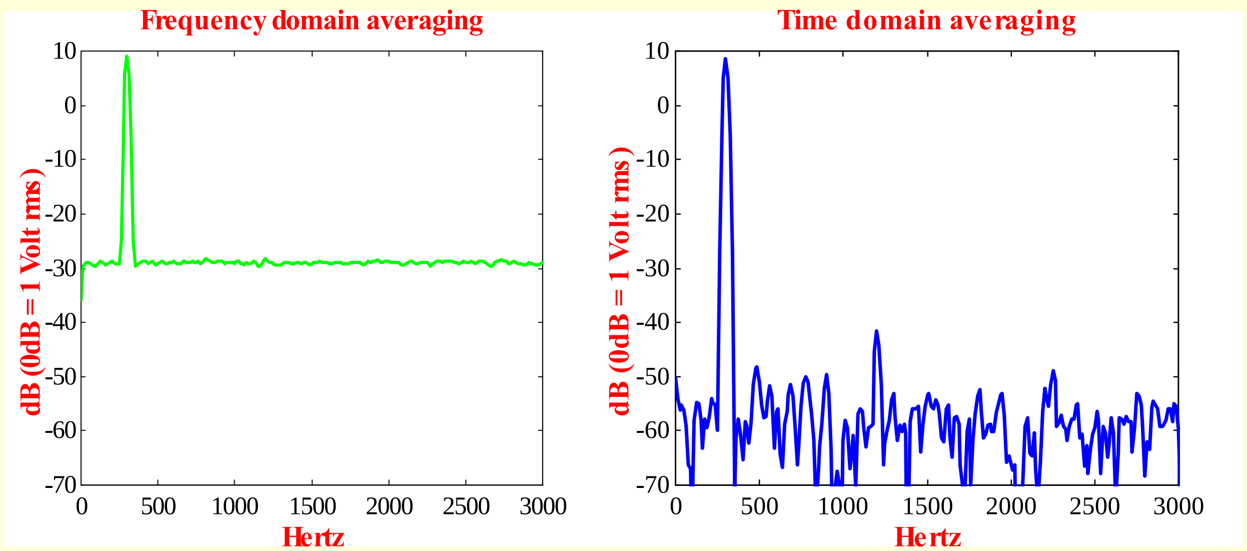


- A stable trigger is not needed.

For frequency domain averaging, the idea is similar except that for each time history, we first compute an instantaneous PSD. These five traces are such PSDs from 5 consecutive frames. Adding them up column-wise as before yields the averaged trace below. Note that the variance of the average result is much less than the instantaneous PSDs. (I actually used 50 averages to get this degree of variance reduction.)

Triggering is not required. Note from the look of the time history, that a stable trigger would be impossible with this type of data.

Compare time & frequency domain averaging



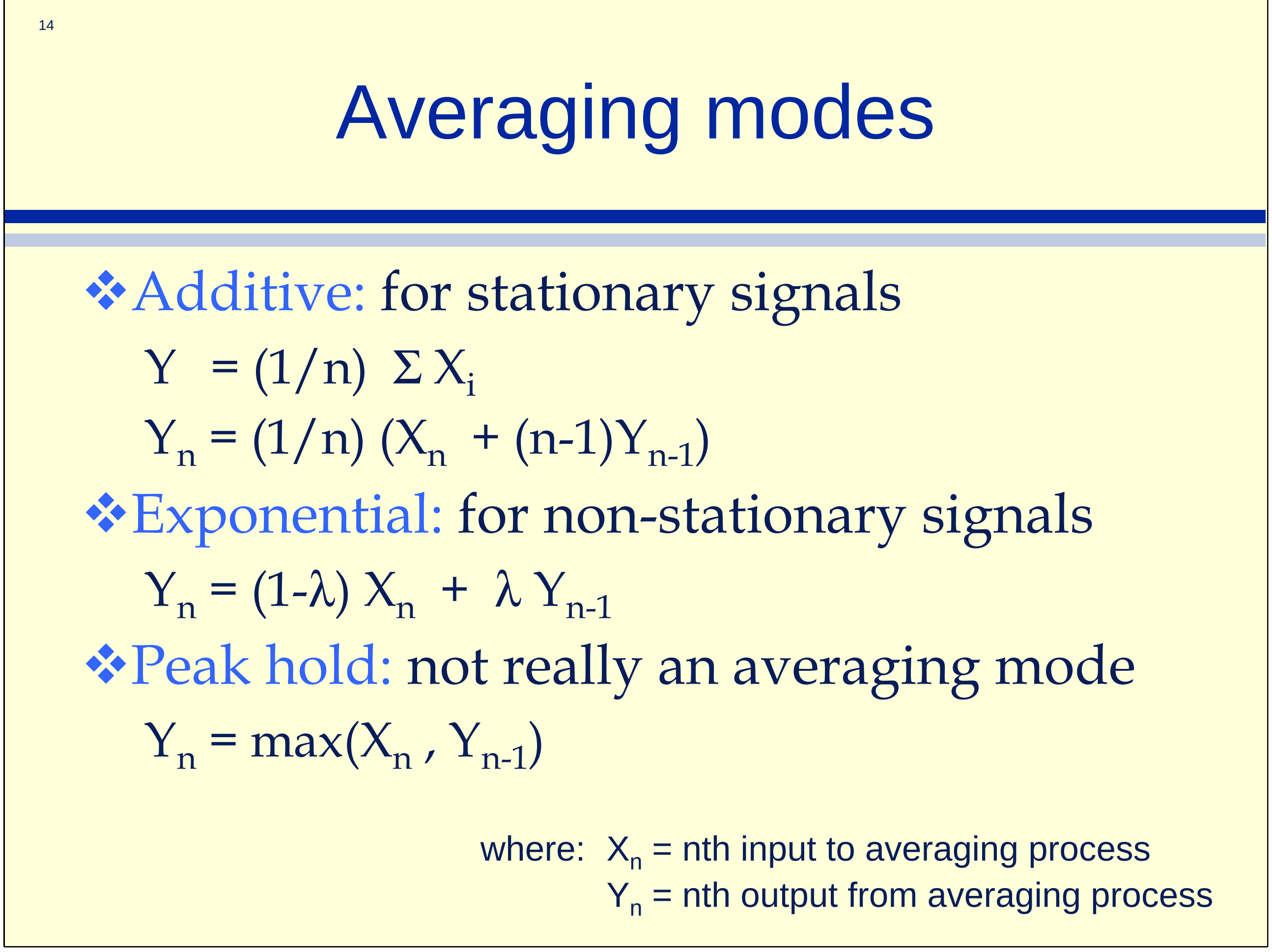
- Time domain averaging can eliminate uncorrelated noise

Here is an example that starkly shows the contrast between the two averaging types.

The left side is a PSD computed using frequency domain averaging. It shows a single tone buried in a high level of noise.

The right side is a PSD computed from the result of a time domain average (on the same signal as before). Note that the noise level is much lower since it has been averaged out. This reveals the existence of another tone in the signal (which is the 3rd harmonic). Remember this only works if you have a stable trigger. Also signal components that are not harmonically related to the fundamental may also average out to zero since they are not synchronously related to the trigger event.

So, more averaging in the frequency domain does not tend to reduce the noise level. It just measures the amount of noise more accurately. However with time domain averaging, the more averaging, the lower the noise.



Averaging modes

- ❖ Additive: for stationary signals

  $Y = (1/n) \ \Sigma X_i$

  $Y_n = (1/n) (X_n + (n-1)Y_{n-1})$

- ❖ Exponential: for non-stationary signals

  $Y_n = (1-\lambda) X_n + \lambda Y_{n-1}$

- ❖ Peak hold: not really an averaging mode

  $Y_n = \max(X_n , Y_{n-1})$

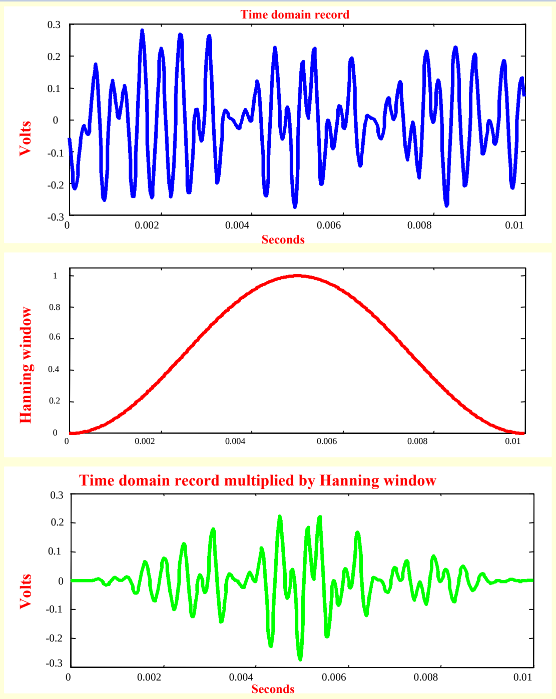
where: X_n = nth input to averaging process
Y_n = nth output from averaging process

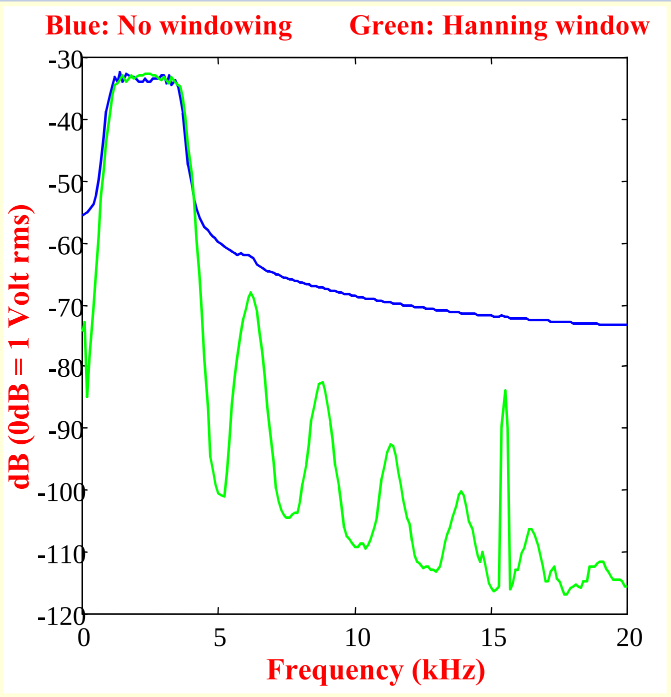
For both time and frequency averaging types, I have been implicitly talking about additive averaging. That is, as shown in this formula, we add the *i*th PSD, as *i* goes from 1 to n (the number of averages) and then divide by n to normalize the result. The second equation is a recursive way of computing the same thing, with the advantage being that the result of each step is correctly normalized so we can view the average as it is being computed. Additive averaging is primarily useful for stationary signals, i.e. signals that don't change their frequency or amplitude characteristics during the time we are viewing them.

If the signal is non-stationary, then additive averaging is not very useful. However if we still need some degree of variance reduction, we can use exponential averaging to accomplish this while still being able to track changes in the input signal. In this formula, λ (from 0 to 1) controls the tradeoff between the variance reduction provided and the ability to track changes in the input. As λ approaches 0, the old data is weighted less, and the tracking ability approaches that of an instantaneous (non averaged) measurement. As λ approaches 1, the old data is weighted more resulting in better variance reduction and a longer time constant. [note: the time constant is $1/(1-\lambda)$ frames]

Peak hold does not really fit the conventional idea of averaging, but it is convenient to classify it here. Basically the maximum of the input and previous average is saved, with this determination made separately at each frequency bin. This is most useful when you are trying to determine whether a certain event has occurred over a long measurement time.

Windowing: Solves the leakage problem





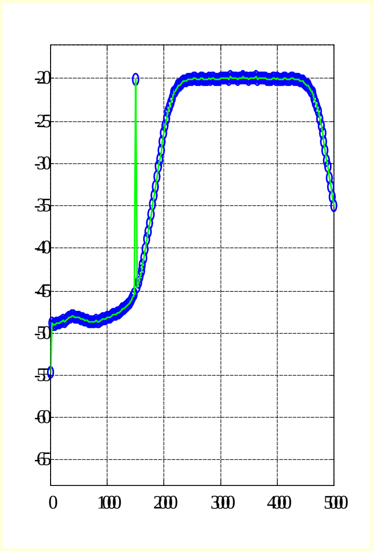
A time history is shown at the upper left. The blue trace on the plot to the right shows the magnitude of the DFT of this time history. The green trace on the same plot is actually a more accurate representation of the signals frequency content. The tone near 16 kHz and the other lumps in the PSD are masked in the DFT by a problem referred to as leakage. The reason is that the blue trace represents the frequency content of the time history repeated an infinite number of times (the periodicity assumptions). Note that the time history begins at a negative voltage but ends at a positive voltage. So when the frame is repeated an placed immediately following, there is a sharp transition from this positive value to the negative one. This sharp transition spreads (or leaks) energy into every bin of the DFT, and obscures the true nature of the frequency content of the signal if we could have sampled it for longer.

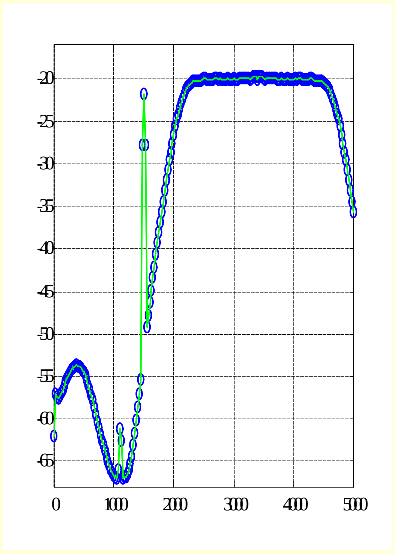
Windowing helps solve this problem. Below the time history is the shape of perhaps the most popular window, called Hanning. It is simply a raised cosine function. It is one in the middle and tapers to zero at both ends. If we multiply the time history by this Hanning window, we get the signal in green. Now when we repeat this new signal, no transitions occur at the boundary. The green frequency trace is the magnitude of the DFT of this windowed signal.

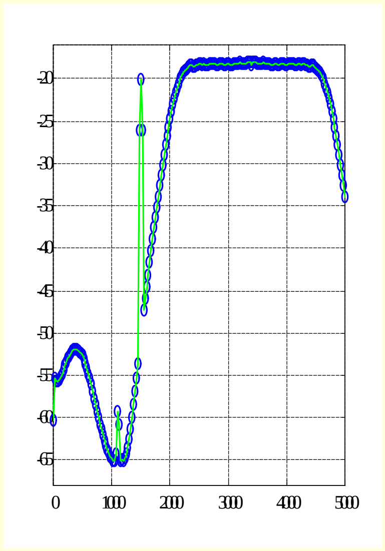
One problem with windowing is that it reduces the total energy in the signal which, as we will see, is difficult to account for.

Windowing: A blessing and a curse

Boxcar		Hanning (Power correction)		Hanning (Amplitude correction)	
1.5 kHz:	-20 dB	1.5 kHz:	-21.7 dB *(1.7dB error)*	1.5 kHz:	-20 dB
3 kHz:	-20 dB	3 kHz:	-20 dB	3 kHz:	-18.3 dB *(1.7dB error)*







To illustrate a drawback of windowing, I fabricated a signal consisting of a 1.5 kHz sine wave (at -20dBV) superimposed with a narrow band noise source. The PSD is computed using a boxcar window (i.e. no windowing) and displayed on the left. Notice that the sine wave and the noise source show up at the same amplitude (-20dBV).

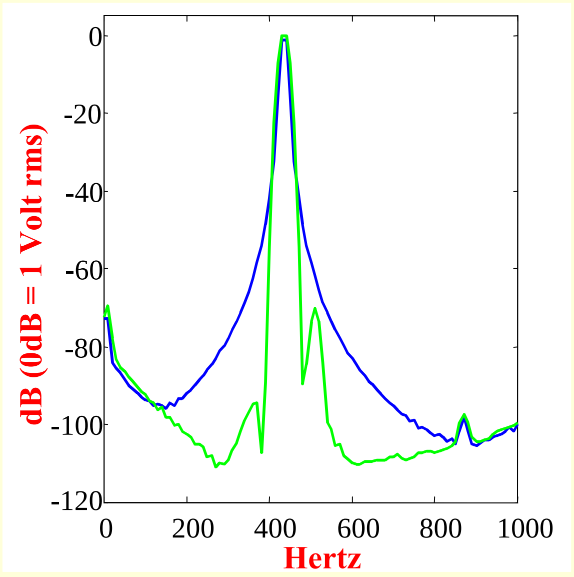
Now we apply the Hanning window (in the frequency domain this can be implemented as a convolution with [-.25 .5 -.25]). The result is shown in the middle figure. Now that the leakage has been reduced by the windowing, we can see that the signal also contains a small 1.1kHz tone. (With the boxcar window, that tone was buried by leakage). The power loss due to this window is 3/8 (sum the squares of kernel), so we have corrected the plot for this power loss by multiplying by 8/3. Notice that the amplitude of the noise source is the same as before. However, when we cursor the peak at the sine wave, we see an amplitude of -21.7 dB. (1.7 dB smaller than before.)

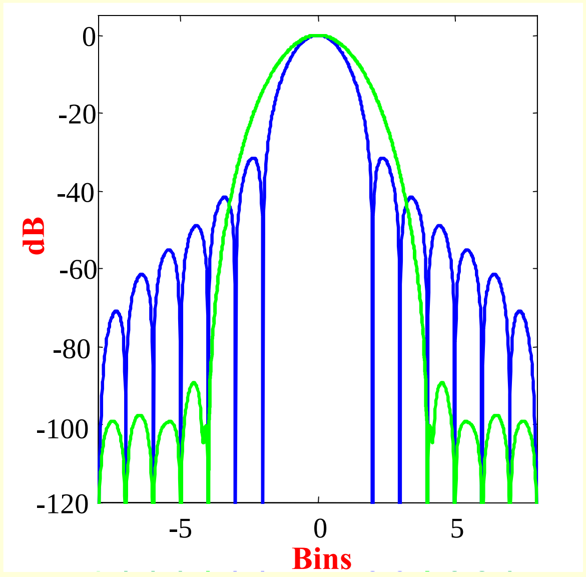
So, what if we wanted to measure the amplitudes of sinusoidal data. Well then we can apply the amplitude correction factor instead (the reciprocal of the square of the center term = 4). With this correction factor we see the display on the right. Now when we cursor the sine wave peak we get the correct amplitude of -20 dBV. But then the amplitude of the noise reads -18.3 dB (again a 1.7 dB error).

Obviously, one correction factor cannot make both numbers right. The windowing has distorted the data in a non-linear way causing the DFT to change shape. (Explain intuitively by showing how Hanning spreads the power of the sine wave out into the neighboring bins.) This same effect is different for the random portion since the neighboring bins are already at roughly the same level.

Hanning is not the only useful window

Purple: Hanning   Green: 4-term 92dB Blackman-Harris





Detecting a signal 70dB down and 7.5 bins away is not possible with Hanning.

Cover right side picture:

This is a PSD of a signal containing 2 sine waves separated by 75 Hz or 7.5 FFT lines (Δf = 10Hz).

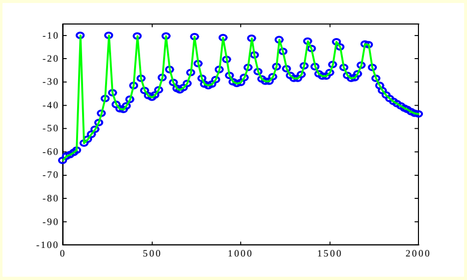
Draw in the shape for the rectangular window. See that the Hanning window provides considerable leakage protection but not enough to detect the smaller sine wave. The Blackman-Harris window easily shows the second sine wave. (Blackman-Harris is actually a family of windows, this one known as the minimum 4-term Blackman-Harris).

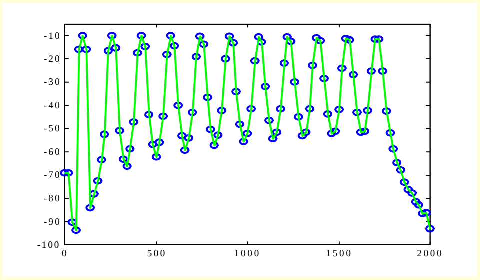
Uncover the right picture:

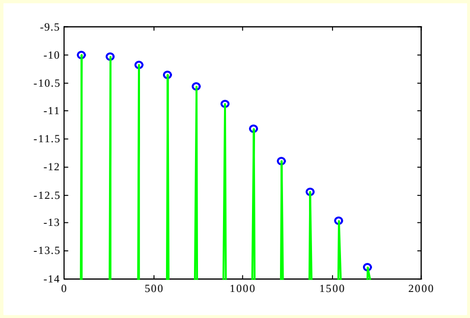
We can see how much leakage protection we get by viewing the Fourier transform of the window shape. The smaller the side lobes, the smaller the leakage. For example, on the Hanning window curve, at 7.5 bins we see that the side lobe is at about -70dB. But this is the same amplitude as the smaller sine wave, so it is still masked by the leakage. The Blackman-Harris window on the other hand shows about 100 dB of leakage suppression at that position.

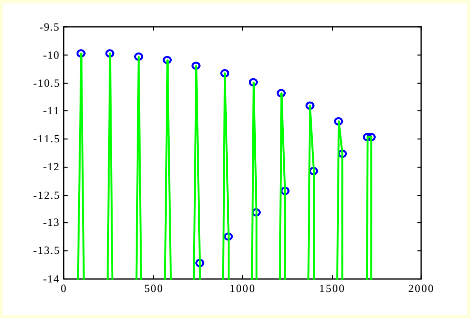
The price you pay for the smaller side lobes is the wider main lobe which decreases the frequency resolution of the measurement.

Hanning also doesn't help much for scalloping loss









Frame=256 points
Δf = 20 Hz

f (Hz)	f / Δf
100	05.00
261	13.05
422	21.10
583	29.15
744	37.20
905	45.25
1066	53.30
1227	61.35
1388	69.40
1549	77.45
1710	85.50

Boxcar: Scallop loss = 3.9 dB
Highest side lobe = 13 dB

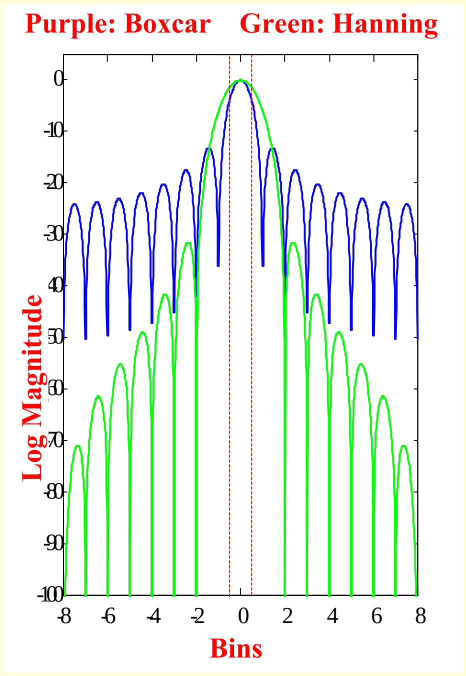
Hanning: Scallop loss = 1.4 dB
Highest side lobe = 32 dB

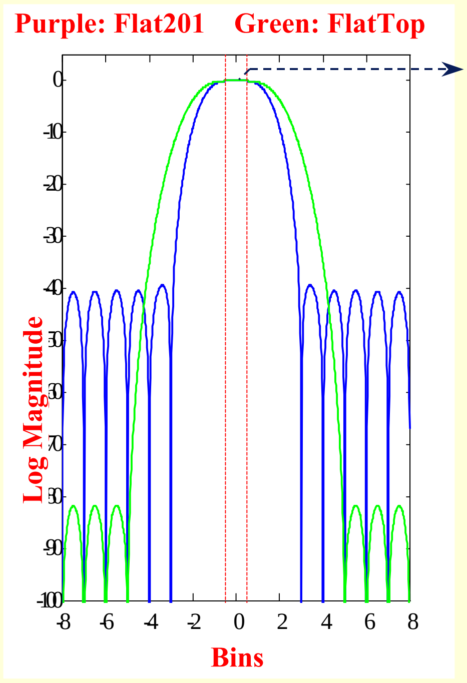
Another problem with windowing (and of FFT analysis in general) is scalloping loss. In my struggle to demonstrate this phenomenon, I fabricated a signal consisting of the sum of 11 equal amplitude sine waves at the frequencies listed here (from 100 Hz up to 1710 Hz). If we sample 256 points of this signal at a 5120 Hz sample rate, our FFT line spacing is Δf = 20 Hz. So dividing by Δf we get the bin numbers shown here. The important part is the fractional component shown in blue. Note that the first component, 100 Hz is exactly on a line (bin 5) and the last component, 1710 is exactly between two lines (bin 85.5), and all the components in between slowly shift from these two extremes as can be seen by looking at the fractional part of the bin number.

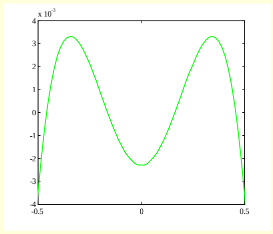
Now the PSD (no windowing) is displayed in the upper left. The high leakage is especially noticeable near 2 kHz. Otherwise the display looks like a good representation of the signal, with one peak for each sine wave component. However if we expand the amplitude scale (near -10 dB) we see that the amplitudes are not the same for all components as you might expect. The first term (100Hz) is actually correct. The others are in error (up to 3.9 dB) due to scalloping loss.]

When we repeat the experiment with the Hanning window, we see that the leakage is much reduced, but the scalloping loss is still significant (a bit less at 1.4 dB).

FlatTop windows to the rescue







Expansion of center bin of FlatTop window. Vertical scale is only +/- 0.004 dB

Cover right side:

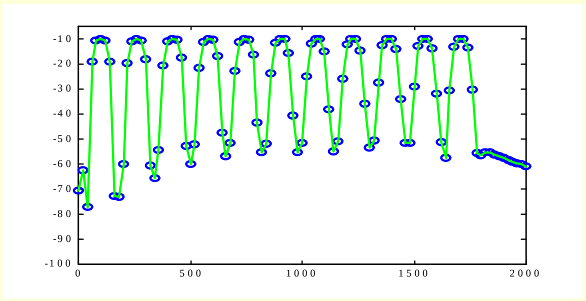
Why do we get this scalloping loss? The answer is easily seen from the Fourier transform of the window shapes (shown here for rectangular and Hanning). The red dotted lines show the extend of the center bin. The amount of droop within these dotted lines represents the scalloping loss. You can see that Hanning shows less droop than the boxcar.

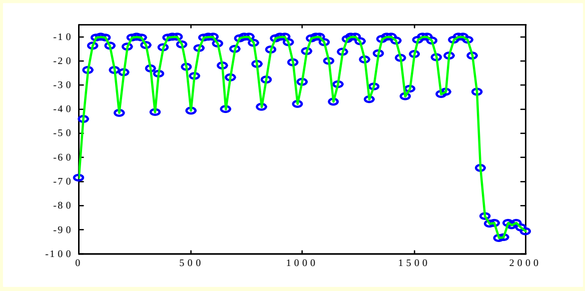
Uncover right side:

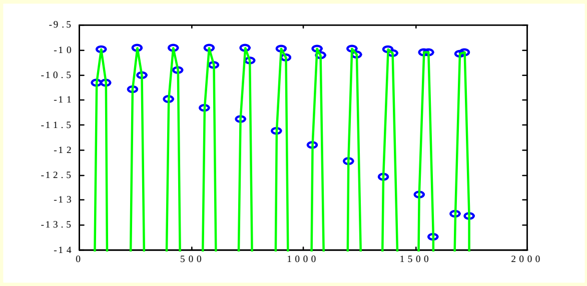
Now lets look at the transform of two windows that do not have appreciable scalloping loss. The first one (blue) I'll call the "Flat201" window (stolen from the Potter 201) and the second one (green) I'll just call "FlatTop". (Note: FlatTop refers to a generic window style, and not to specific window coefficients. So one instrument's FlatTop may be different than the next.)

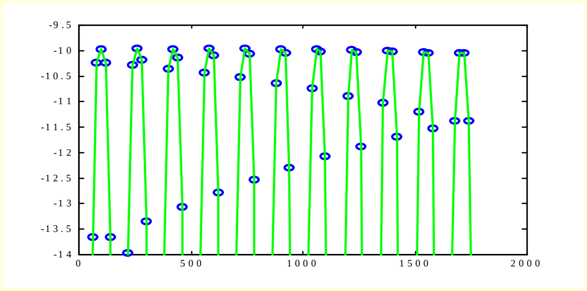
The main difference between these two windows is in how they play the tradeoff between leakage suppression and the width of the main lobe. Both reduce scalloping loss to negligible levels (.01 and .0035 dB respectively). Here is an expanded view of the center bin for the FlatTop window (Flat201 looks similar). Not that the vertical scale used here is very small (+/- .004 dB).

FlatTop windows: The proof









Flat201: Scallop loss = ±0.01 dB
Highest side lobe = 40 dB

FlatTop: Scallop loss = ±0.0035 dB
Highest side lobe = 82 dB

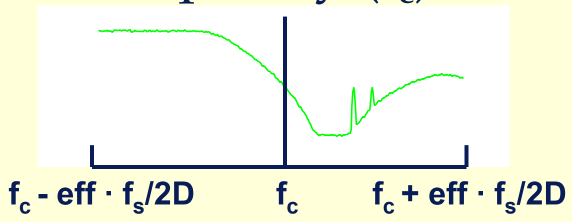
Now here is the proof.

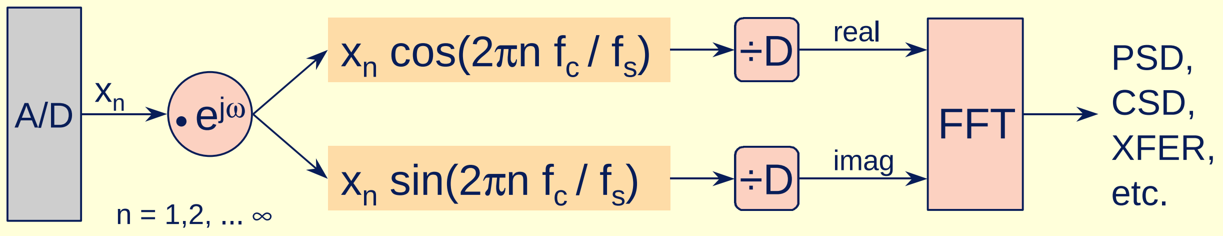
We again compute the PSD of the signal containing the 11 sine waves, but now using these two FlatTop windows. The lower plots show the same expanded view as before, and there is no scalloping loss visible.

We can see that the leakage suppression on the right side (FlatTop) is much better than on the left (Flat201). But note how the width of the main lobe affects the result. In the display using the FlatTop, the 11 peaks are more smeared together due to the wider main lobe.

Frequency translation (zoom)

- Centers the FFT analysis power in a narrow band around some center frequency (f_c)
- Not the same thing as display expansion





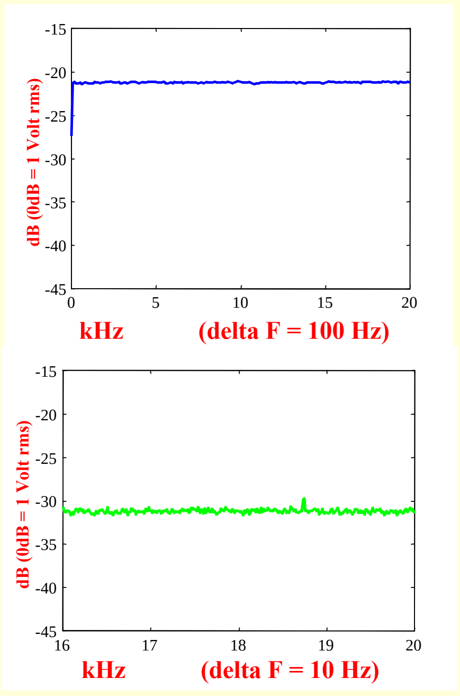
Frequency translation is an important data acquisition technique. It's also often called zoom. (In the graphics world, zoom refers to display expansion. But frequency translation is not a display expansion.)

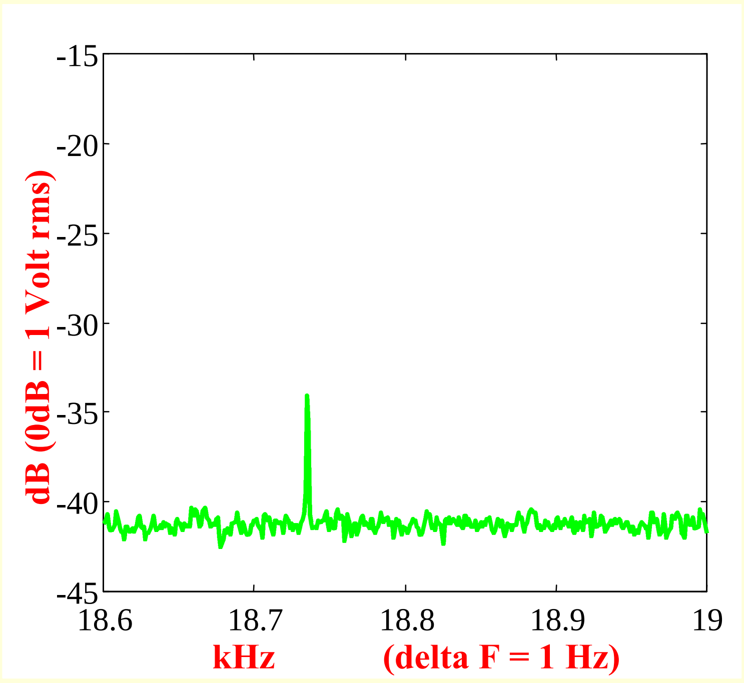
To those of you familiar with radio theory, it is similar to heterodyning. The samples from the A/D converter (x_n) are multiplied by a complex phasor ($e^{j\omega}$) resulting in these two sequences (real and imaginary parts). Each of these sequences is then fed through a decimate by D low pass filter. The zoom factor (or expansion factor) is D/2. Each of these ÷D blocks represent the whole cascaded chain of decimate by 2 and 5 blocks shown before. Since two of these blocks are required, the filtering operation for frequency translation requires twice the processing horsepower as for baseband.

Since the FFT can handle complex valued inputs, we can then feed the result of the filtering directly to the FFT and we can compute all the usual FFT based functions (auto & cross spectra, transfer functions, etc.)

Multiplying by the complex exponential essentially twists or frequency shifts the data, so that the center frequency moves down to DC. (The twisted signal is complex valued since it is no longer symmetric around DC.) Then the filtering increases the frequency resolution by discarding the frequencies outside the band of interest.

Zoom can pull a signal out of noise





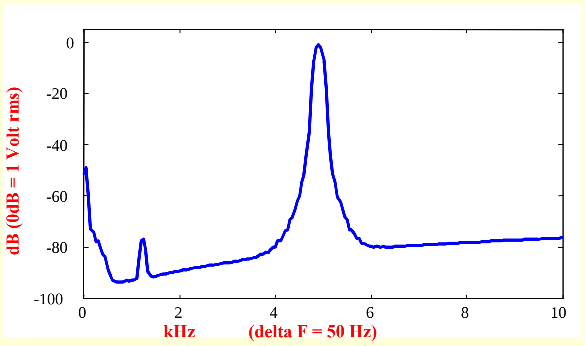
Because of the averaging effect of the digital filters, we can often detect signals using zoom that otherwise would be buried in the noise.

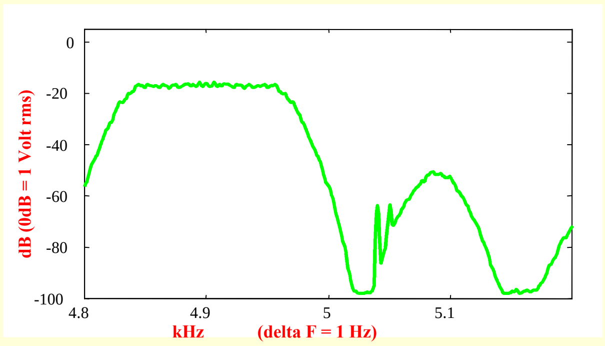
For example, the upper left plot is a baseband PSD of what looks like pure random noise.

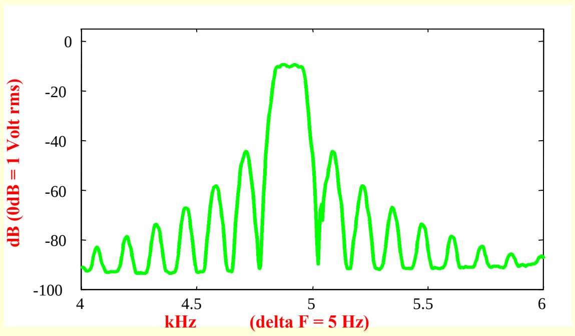
In the lower left plot we have zoomed around 18 kHz by a factor of 10 (the resolution is improved from 100 Hz to 10 Hz). Now we can barely see that there is a sine wave buried in the noise.

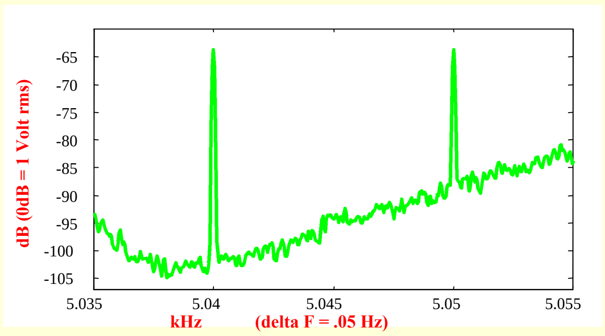
Now zooming by another factor of 10 (resolution of 1 Hz) we can more clearly see the sine wave.

Baseband can’t match zoom’s resolving power







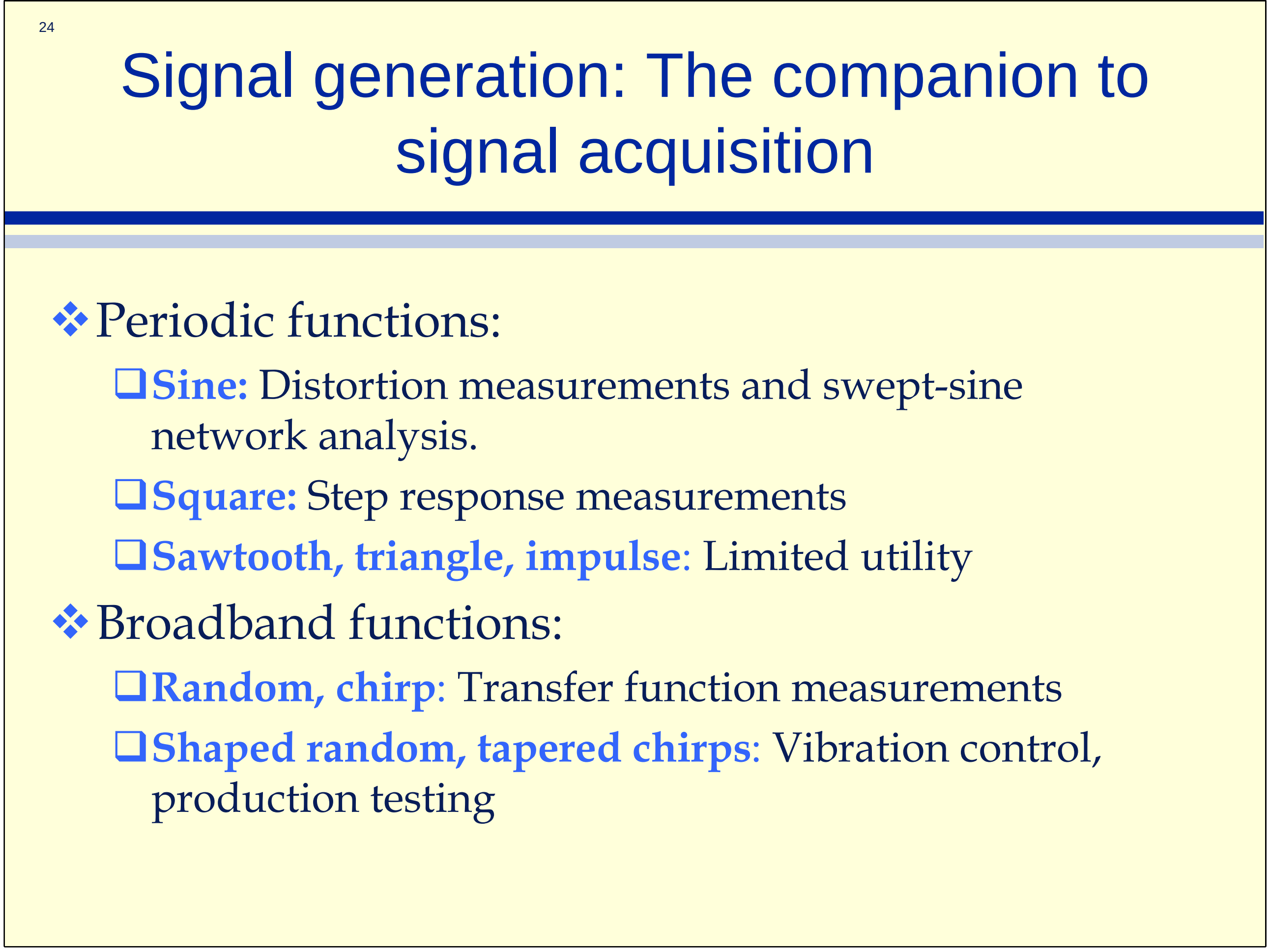


In the upper left is a baseband measurement with a resolution of 50 Hz. The signal looks like it has primarily one component near 5 kHz. The shape is a little broad to be a pure sine wave, but it is difficult to tell whether it is sinusoidal or random.

An increase in resolution by a factor of 10 (lower left) shows that the peak is actually a band of noise. Also we see side bands that were not evident from the baseband display. We can also see a small glitch near 5 kHz that indicates there might be something irregular there.

So we increase the resolution again by a factor of 5 (upper right). Now we can see that this little glitch actually was two sine waves near 5.05 kHz. If we zoom by another factor of 20 (lower right) we can now see these two tones clearly.

Note that this extra resolution comes at a price. The price is acquisition time. The acquisition time is equal to the reciprocal of the resolution. So for the baseband measurement (res = 50Hz) the acquisition time is 20mS. Thus we could do 50 averages in only 1 second. The lower right plot has Δf=.05Hz, so the acquisition time is 20 seconds and our 50 averages would take almost 17 minutes. This is a fact of physics; the fastest analyzer in the world can’t do it any faster.



Signal generation: The companion to signal acquisition

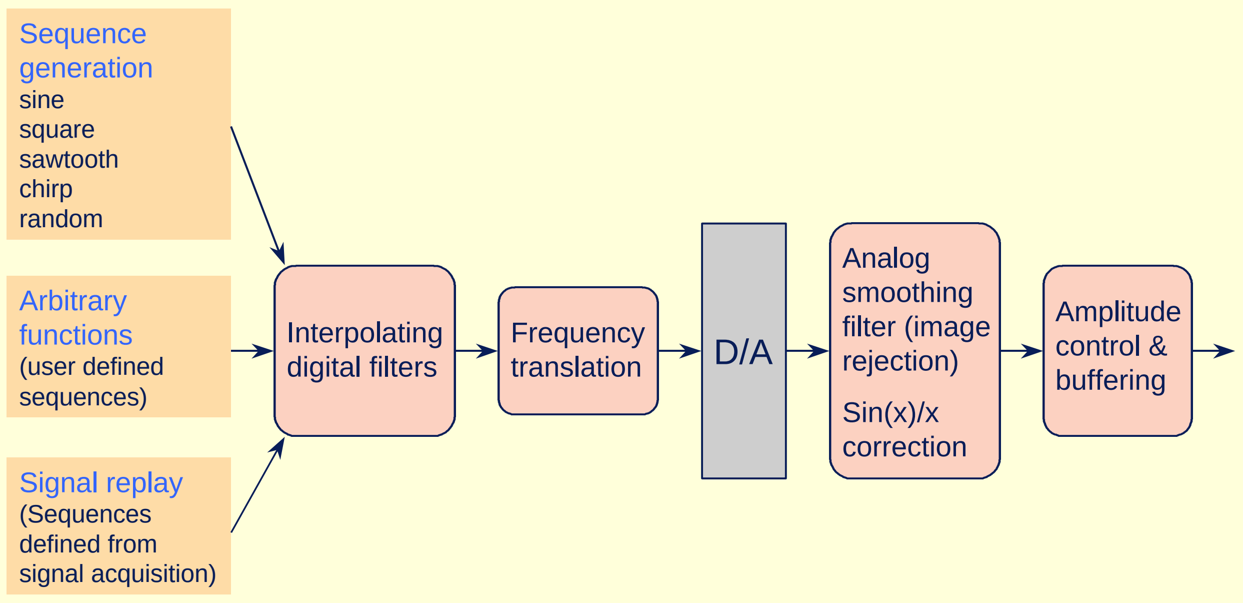
- Periodic functions:
  - **Sine:** Distortion measurements and swept-sine network analysis.
  - **Square:** Step response measurements
  - **Sawtooth, triangle, impulse**: Limited utility
- Broadband functions:
  - **Random, chirp**: Transfer function measurements
  - **Shaped random, tapered chirps**: Vibration control, production testing

Signal generation often gets slighted in discussions of sampled data systems. However it is just as important and just as difficult to do well. Many of the technical challenges are similar, although some are unique (e.g. the sequence generation for random noise and chirps)

The most useful output signal is the ubiquitous sine wave, the basis function for all spectral analysis. For example, to measure the distortion of a network at a specific frequency, we need to excite the network with a sine wave that is as pure as possible. The purity of the sine wave limits the levels of distortion that can be measured. In general it is desirable for the dynamic range of the signal generation system to be similar to the dynamic range of the acquisition system. The sine wave can also be stepped in frequency to make transfer function measurements using the classic swept-sine technique. The other periodic functions have more occasional uses, usually for time domain measurements.

Broadband functions, the other class of output functions, are used to excite a whole band of frequencies at once. These functions allow network measurements to be made much faster than using sine waves.

Signal generation: The nuts & bolts



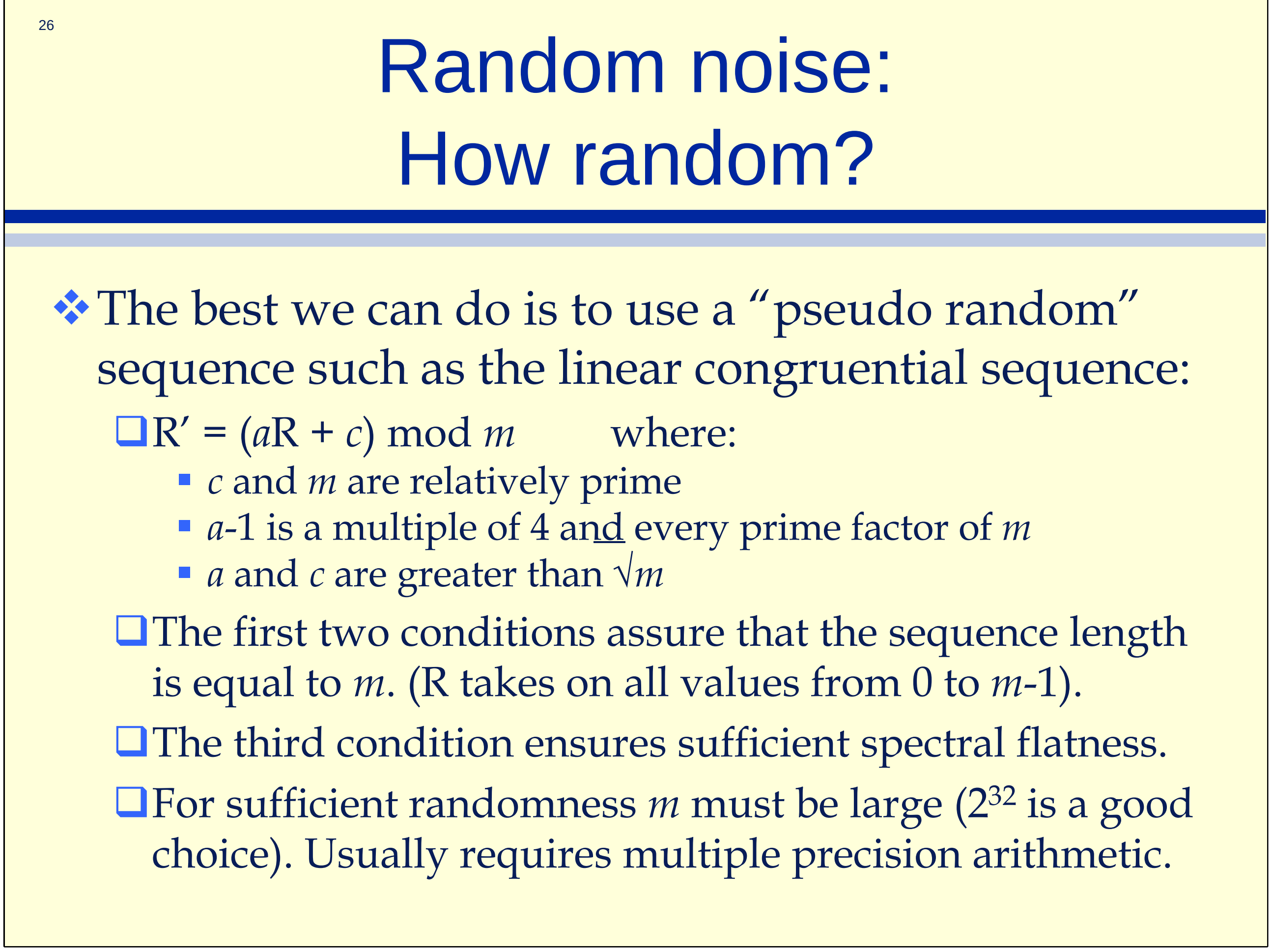
Other than the sequence generation, the components of a signal generation system are similar to a data acquisition system. The interpolating digital filters are used so that the D/A and its smoothing filter can run at one fixed sample rate.

The interpolated signal can then be frequency translated to concentrate the energy of the output in a narrow band. It is common to frequency translate the output when the acquisition system is also frequency translating.

The D/A converts the digital numbers into a series of analog steps, each step holding its value constant for the sample period (called a zero-order hold).

An analog smoothing filter is used to smooth out these steps. The characteristics and quality measures of this filter are identical to that of the AA-filters. Often the exact same filter design can be used. The D/A zero order hold introduces a sin(x)/x frequency error which is usually compensated for with a simple one-pole filter.

Finally the filtered output is amplified or attenuated to the desired output amplitude and buffered to yield the desired driving characteristics.



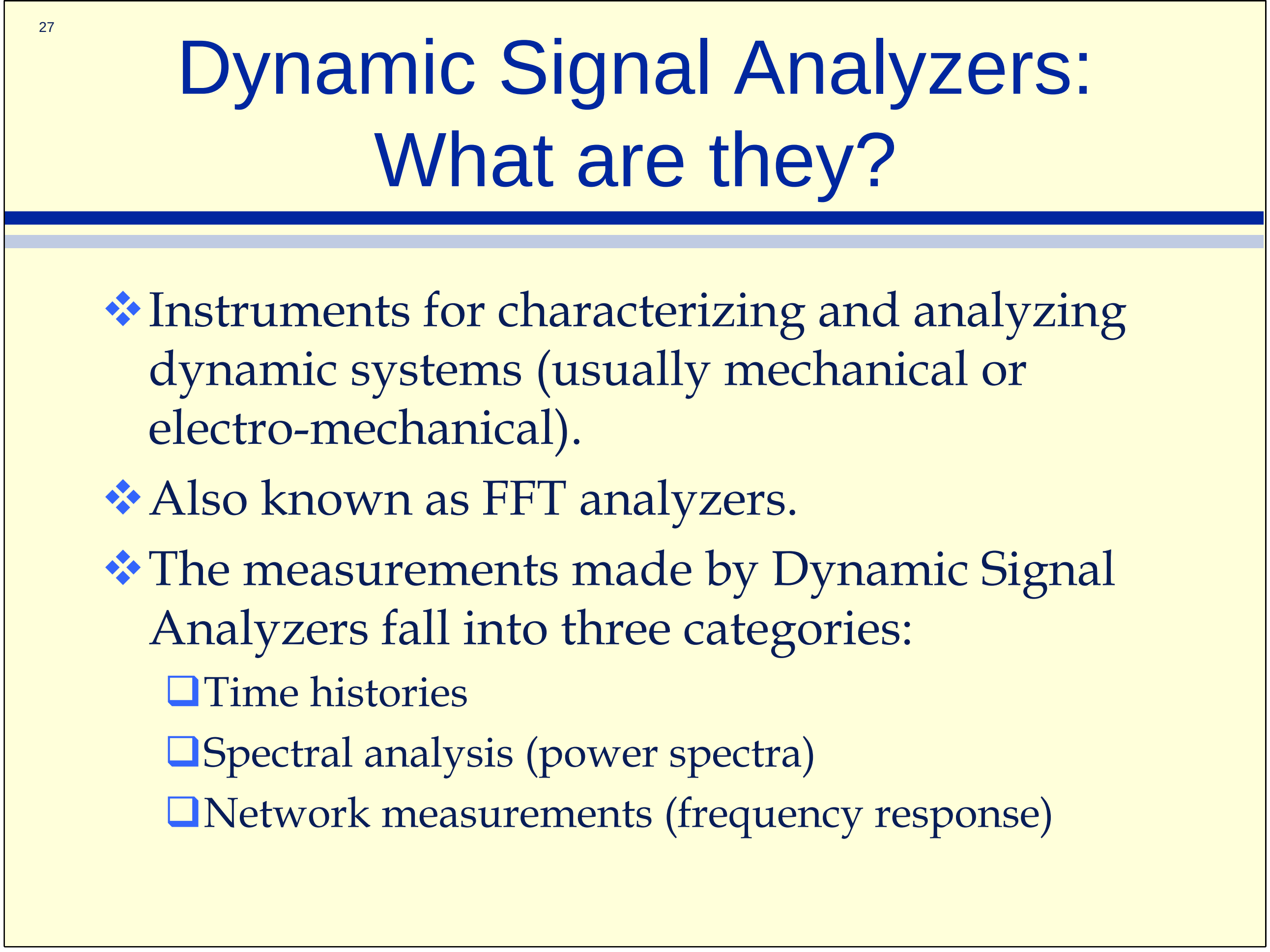
Random noise: How random?

- The best we can do is to use a “pseudo random” sequence such as the linear congruential sequence:
  - $R' = (aR + c) \bmod m$ where:
    - c and m are relatively prime
    - a-1 is a multiple of 4 and every prime factor of m
    - a and c are greater than $\sqrt{m}$
  - The first two conditions assure that the sequence length is equal to m. (R takes on all values from 0 to m-1).
  - The third condition ensures sufficient spectral flatness.
  - For sufficient randomness m must be large (2^{32} is a good choice). Usually requires multiple precision arithmetic.

Of the sequence generators show in the previous diagram, the most challenging one is random noise. Can a deterministic device such as a DSP chip generate a purely random number sequence (i.e. a sequence that does not repeat) ?

No it can’t. However we can generate sequences that at least over a short period of time look random. These are called pseudo random sequences. There are many algorithms for computing pseudo random sequences, both simple and complex. Complexity does not ensure a long sequence however. The literature is full of examples of extraordinarily complex algorithms that yield extremely poor results.

My favorite because of its simplicity and well researched properties in known as the linear congruential sequence. We start with an arbitrary number (R) known as a random seed. We multiply by constant *a*, add another constant *c* and then divide by a third constant *m* and save the remainder from this division (R’) as the next number in our random sequence. Since the remainder will always be between 0 and *m*, the random numbers generated are also in this range. If the sequence covers all such values, then it is called a maximal length sequence. Choosing the three constants according to these two simple constraints has been proven to be sufficient to ensure a maximal length sequence. Thus we can make our sequence as long as desired by picking a large value for *m*. Choosing *m* to be a power of 2 simplifies the arithmetic. $m = 2^{16}$ is good enough for some applications. 2^{32} is much better, but usually requires multiple precision arithmetic (32 by 32 bit multiply).



Dynamic Signal Analyzers: What are they?

- ❖ Instruments for characterizing and analyzing dynamic systems (usually mechanical or electro-mechanical).
- ❖ Also known as FFT analyzers.
- ❖ The measurements made by Dynamic Signal Analyzers fall into three categories:
  - ❑ Time histories
  - ❑ Spectral analysis (power spectra)
  - ❑ Network measurements (frequency response)

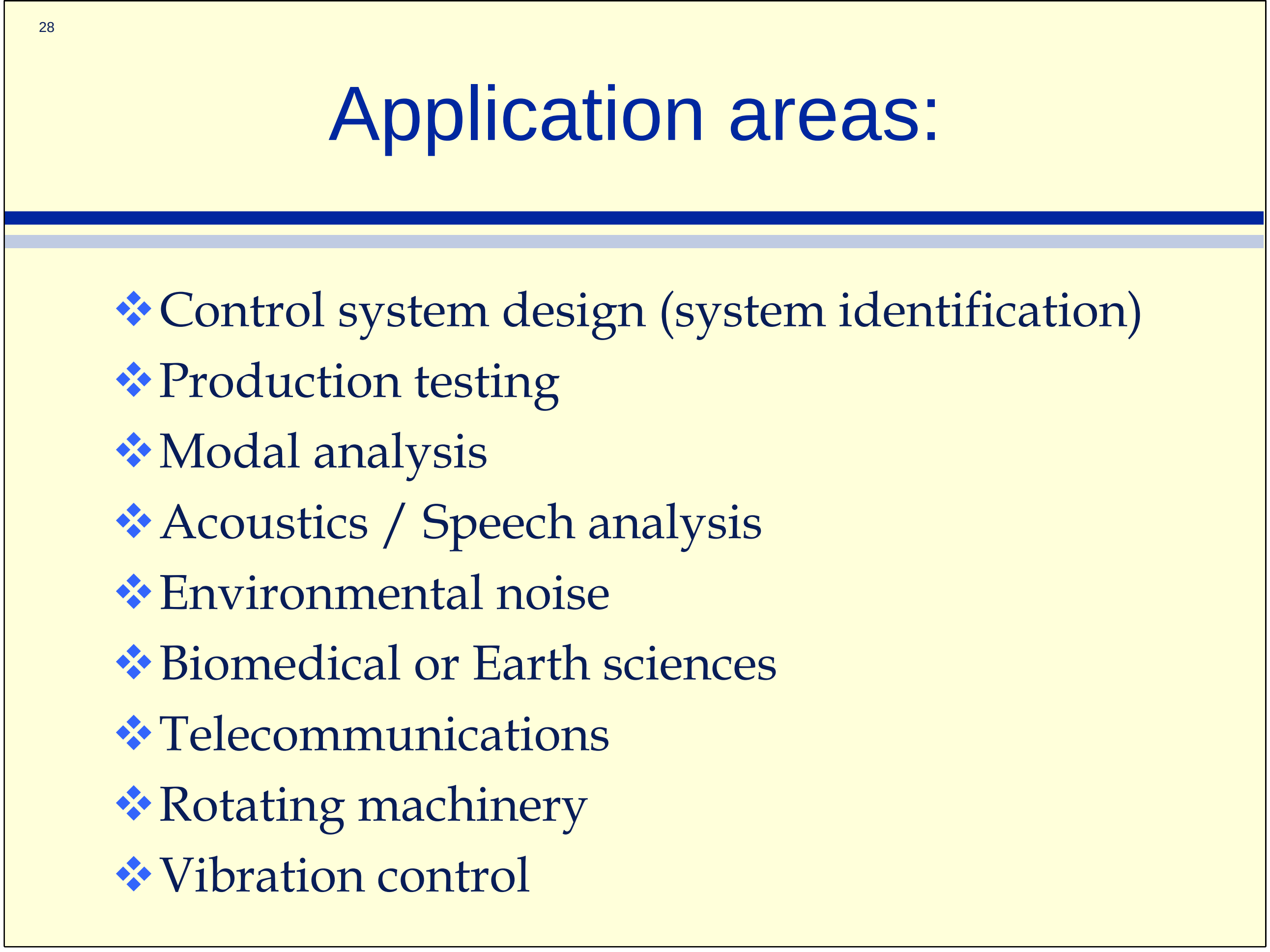
The name Dynamic Signal Analyzers was coined by HP, one of the early makers of such equipment. Other manufactures have sometimes used this fairly descriptive name, although FFT analyzers or Fourier analyzers is also common. They are designed to analyze signals and systems covering a broad range of fields and applications, usually involving something that moves (mechanical) with some applications involving purely electrical systems (e.g. telecommunications).

The measurements are usually divided into these three categories.

For time domain measurements, the sophistication of a dynamic signal analyzer is not usually required. A digital scope would suffice. However these functions are useful, since it often means that you can leave the scope behind.

Most spectral analysis problems can be accomplished with a single input channel, and draw on the power of the FFT to display the time histories in the frequency domain.

Network measurements refer to measuring the input/output characteristics of a system usually to compute a frequency response function (FRF) or a transfer function (mathematical system model). Network measurements usually require a signal generator to excite the system under test, and always require two input channels, one to measure the excitation and one to measure the system response.

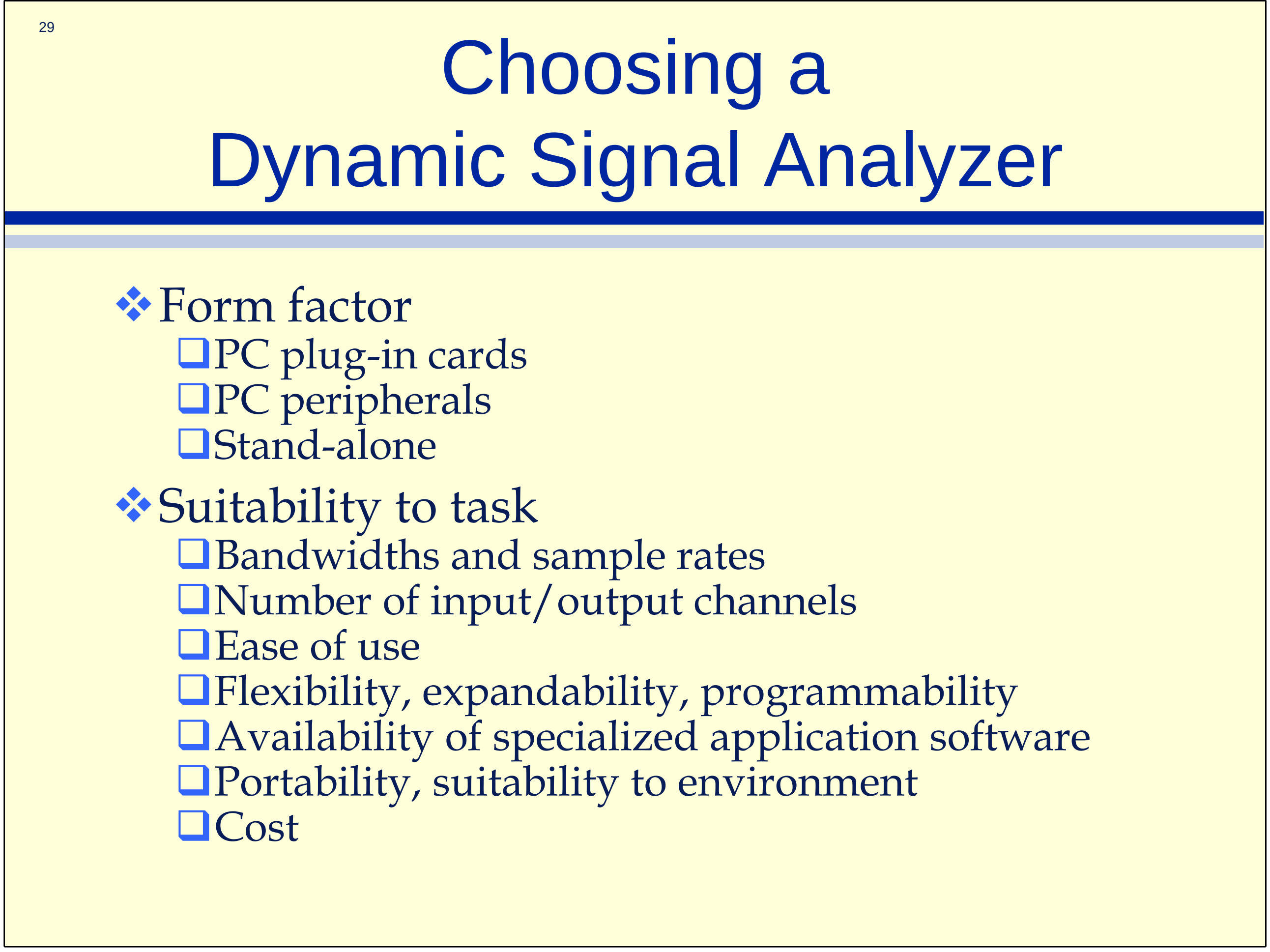


28

Application areas:

- ❖ Control system design (system identification)
- ❖ Production testing
- ❖ Modal analysis
- ❖ Acoustics / Speech analysis
- ❖ Environmental noise
- ❖ Biomedical or Earth sciences
- ❖ Telecommunications
- ❖ Rotating machinery
- ❖ Vibration control

Dynamic signal analyzers cover such a diverse range of applications that it is somewhat difficult to categorize them. However at least 90% of the time I find that the customers application can fit into one of the following categories.



Choosing a Dynamic Signal Analyzer

- Form factor
  - PC plug-in cards
  - PC peripherals
  - Stand-alone
- Suitability to task
  - Bandwidths and sample rates
  - Number of input/output channels
  - Ease of use
  - Flexibility, expandability, programmability
  - Availability of specialized application software
  - Portability, suitability to environment
  - Cost

Dynamic signal analyzers come in three flavors. The first type are built on cards that plug in to your personal computer (usually IBM compatibles). Although this style is the least expensive the measurement performance is usually quite limited. They are so constrained by space that they often omit such essentials as anti-aliasing filters or built-in signal generators. Also the dynamic range capabilities are often severely hampered by the harsh noise environment found inside the typical PC. The second type are built in a separate box but still connect to a PC to perform all the human interface functions to which they are so well adapted. Since the noise and space problems can be more easily controlled, the performance of the PC peripherals are much better. With the continued advancement of notebook computers, the PC peripherals are the more capable mobile performers since the plug-in cards rarely work with notebook computers. The third type is the traditional stand-alone instrument which includes all the knobs, switches, and display functions all within one box. The stand-alone analyzers tend to be very expensive since they include custom display, mass storage, keyboard, and pointing device functions. Plus they can't take advantage of the monthly advances in price/performance of personal computers nor the ever widening array of special purpose computer languages and 3rd party application specific software.

Here are the seven main points to consider before purchasing a dynamic signal analyzer. Use the first two criterion to quickly eliminate unsuitable contenders. Ease of use should be high on the priority list, since all the power and capability in the world won't be used if the user interface is cumbersome, inconsistent, or overly complex. If your application is fixed and not likely to change a few years down the road, cost should be your next most important concern. Otherwise flexibility, expandability, and programmability will be they key to the equipment's continued utility.