LINEAR CONTROL SYSTEM PITFALLS

Do-While Jones
Ridgecrest, Calif.

Do-While Jones began his career in analog circuit design but switched to digital circuit design when he discovered that digital circuits were easier to design. He switched to software design when he discovered that programming computers was easier than building them, then transferred to an organization that was planning a large software project because thinking about programming is easier than programming. Now he lectures on Ada because talking is even easier than thinking. He has published articles on Ada in *Computing Language, Embedded Systems Programming*, and *Dr Dobb's Journal*. His first book, *Ada in Action*, was published in 1989.

Designers are often unpleasantly surprised when they convert an existing analog linear control system to the equivalent digital control system. The digital system is likely to be unstable if you simply replace Laplace Transforms with the corresponding ZTransforms. This paper reveals some common pitfalls and tells how to avoid them.

Design methods for analog, closedloop control systems are well known. Engineers know how to determine stability in terms of gain margin and phase margin. They can accurately predict the response of a linear system to unit step inputs and can find the steady state error. As long as the problem remains in the linear, timecontinuous, analog world, there won't be any surprises.

Not all problems fit into that category any more. Completely analog systems are becoming rare. Digital sensors and microprocessor controllers are finding their way into closedloop control systems. Hybrid systems can perform better than purely analog systems, but designing the hybrid system isn't as simple as it appears at first.

LINEAR SYSTEMS REVIEW

One of the most important, and most difficult courses an engineer takes in college is Linear Systems (sometimes called Modern Control Theory, ClosedLoop Servo Systems, or something like that). There isn't enough time in one semester to adequately cover the subject, so this review is necessarily oversimplified.

Systems are composed of components. Each component has a transfer function. The transfer function tells how to determine the output from the input. For example, an electric motor could be a component in a cassette tape player. (The tape player is the system.) The transfer function of the electric motor tells how fast the motor spins when various input voltages are applied (assuming a constant load). The electric motor converts voltage to angular speed, and the transfer function tells how the motor does it.

In the simplest case, the transfer function is independent of time. That is, if you tell me what the input is right now, I can tell you what the output is right now. In practice, however, it takes components time to respond to an input. If you turn on an electric motor, it takes some time to come up to speed. So, knowing that 2 volts is being applied to an electric motor is not sufficient information to tell you how fast the motor is spinning. You need to know how long the voltage was applied, and how fast the motor was spinning when you first applied the voltage. So, in general, a transfer function is a complex function of time. Fortunately, the calculations can be simplified by working in the frequency domain instead of the time domain.

The frequency domain is easier to work in than the time domain because there are fewer variables needed to characterize the waveform. For example, it takes an infinite number of samples to completely represent a continuous sine wave in the time domain. That is, given an infinitely large table of measured voltages at frequent intervals, one could compute the

value of the voltage at any time by curvefitting to a few voltage points in the neighborhood of interest. On the other hand, a continuous sine wave can be perfectly characterized in the frequency domain by three parameters. Those parameters are its peak amplitude, its frequency, and its phase at zero time.

Therefore, if you measure the amplitude and phase of a sine wave input to a component, and measure the amplitude and phase of the sine wave output, you can divide the output amplitude by the input amplitude to get the gain of the component at that frequency. You can subtract the phase of the output from the phase of the input to determine the phase shift at that frequency. You can repeat these measurements at various frequencies and compute the gain and phase shift of the component as a function of frequency. These two functions (the gain function and the phase shift function) tell you everything you need to know about the component.

These two (real) functions can be expressed in terms of a single function of complex frequency. (That is, the frequency has a "real" and an "imaginary" part.) This function is generally expressed as the ratio of two polynomials. These two polynomials can be factored to find their roots. The roots of the numerator polynomial are called the zeros of the function because when you apply an input at that frequency, you get no output. The roots of the denominator polynomial are called the poles of the function because when you apply an input at that frequency, you get an infinitely large output.

The poles and zeros of the transfer function of the component tell you everything you need to know about the component's response to a sine wave input. (This should answer the two questions you certainly asked your high school math teacher. "What good are complex numbers?" and "When am I ever going to need to know how to find the roots of a polynomial?") Given the poles and zeros you can reconstruct the numerator and denominator polynomials of the transfer function. Given the ratio of the polynomials you can compute the magnitude and phase of any (real or imaginary) frequency.

This only tells you how the component responds to one special case. (The special case when the input is a sine wave.) In general you need to know how the component will respond to any arbitrary input waveform.

One important property of a linear system is that the output produced by two input signals applied simultaneously is equal to the sum of the outputs when the two signals are applied independently. It has been proven that any periodic waveform can be adequately approximated by adding a number of carefully chosen sine waves of the proper frequency, phase, and amplitude. Therefore, knowing how a linear component responses to sinusoidal inputs of various frequencies allows us to compute how it responses to arbitrary periodic waveforms, by adding the responses to each sinewave component.

This allows us, for example, to compute the sound a bell will make when periodically struck with a hammer. It does not, however, let us compute the sound a bell will make when struck once with a hammer because the input isn't periodic it only happens once. However, we can

compute the sound a bell will make when struck once per hour, and the first few seconds of that response is all that we care about. So, we find that the nonperiodic case is just a special case of an input with a very long period.

So, the key to knowing how a component will respond to any arbitrary input is to know how it will respond to sine wave inputs. We can synthesize the input signal by adding sine waves of the proper frequency, amplitude, and phase, and easily determine the amplitude and phase of each output frequency. Then we can add sine waves with those frequencies, amplitudes, and phases to get the output response.

Suppose a system consists of two components. Knowing the individual transfer functions of two system components, we can compute the performance of the total system by multiplying the two transfer functions together. (They are just ratios of polynomials. They are easily multiplied, especially if they are expressed in factored form). Any number of components can be connected in this way to form an "openloop" system.

Open loop systems make poor control systems because they have no way of adjusting to small changes. Try writing two or three sentences on a piece of paper with your eyes closed. Your brain is in complete control of your hand, but it has a difficult time forming the letters properly because it has no way of knowing exactly where your hand is on the paper. It has no way of knowing if the letters are on a straight line or not. Your brain needs feedback to tell it what the output looks like.

Figure 1 shows a "closedloop" system. A portion of the output is combined with the input. When the feedback is exactly equal to the desired input, it cancels out the input. The error signal is zero and the output remains at the proper value. Negative feedback makes a system

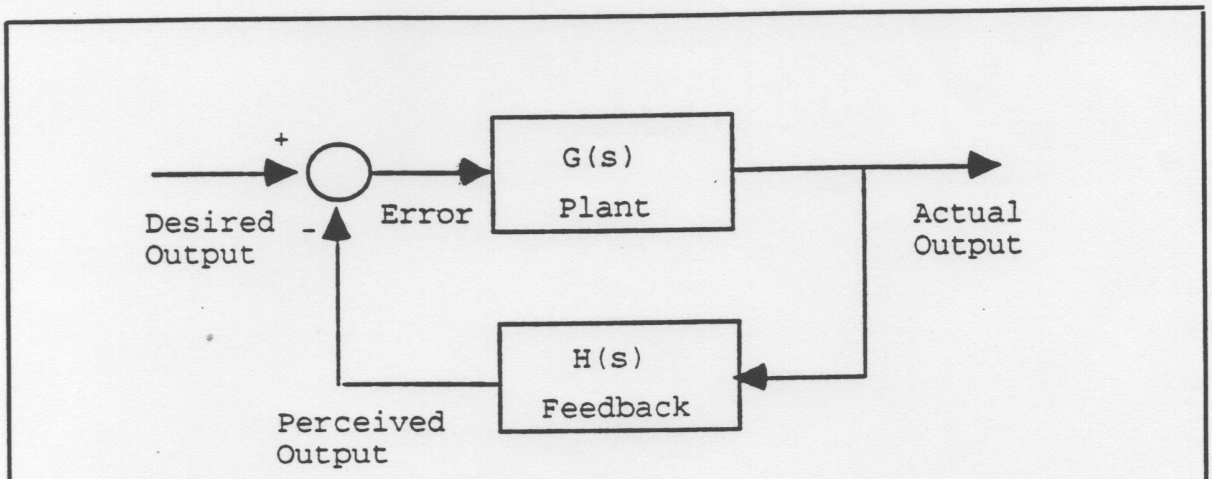

Figure 1. A canonical "closedloop" system derives an error signal from the difference between the actual output and the desired output.

stable because it automatically adjusts the error signal to keep the output at the desired value. When the measurement of the output is added to the control signal (instead of subtracted from it), it is referred to as "positive feedback." Positive feedback makes a system unstable.

For example, a properly designed climate control system compares the current temperature to the desired temperature. If it is too hot, it turns on the air conditioner. If it is too cold, it turns on the heater. Negative feedback keeps the temperature at the desired level. But if the

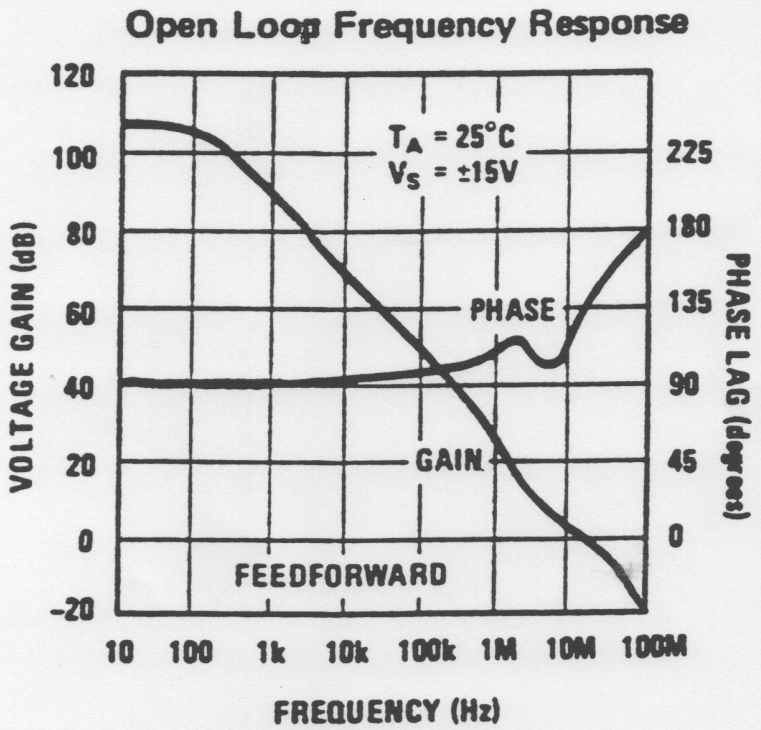

Figure 2. The Bode plot of gain and phase for a National Semiconductors LM101A operational amplifier shows 20 db gain margin and 45 db phase margin.

heating contractor hooks up the wires backwards, the system will go unstable. As soon as the temperature gets just a little bit too warm, it will turn on the heater making it even warmer. Or, if it gets a little bit too cold, the air conditioner will come on making it colder. The positive feedback will make it impossible to control the temperature.
Remember that the transfer function has a phase shift associated with it. If the phase shift is 0, and the wires are hooked up correctly, the feedback will be negative and the system will be stable. But the time it takes a signal to get through a component tends to be relatively

constant, and this fixed time represents more and more phase shift as frequency gets higher. (A 1 Hz signal has a 1 second period. A 0.005 second delay represents a 1.8 degree phase shift at 1 Hz. A 10 Hz signal has a 0.1 second period, so 0.005 seconds represents an 18 degree phase shift at 10 Hz. A 100 Hz signal has a 0.01 second period, so a 0.005 second delay is equivalent to 180 degrees of phase shift.) When the phase shift reaches 180 degrees,

the feedback at that frequency becomes positive, and the system could be unstable.

A closedloop system will be unstable if the transfer function around the loop has a gain greater than 1 and a phase shift of 180 degrees at any frequency. Gain is usually expressed in decibels. Decibels are computed by taking 20 * Log10(GAIN). Since Log10(1) = 0, it is equivalent to say that the system is unstable if the Gain is more than 0 db when the phase shift is 180 degrees.

Loop gain and phase can be plotted on a Bode Plot like the one in Figure 2. This particular Bode plot describes the performance of an LM101A operational amplifier. At 10 Hz it has a gain of 110 db, and the gain decreases to 20 db at 100 MHz. The phase shift is 90 degrees at 10Hz, an wiggles its way up to 180 degrees at 100 MHz.

The amount that the openloop amplitude is below 0 db at the frequency where the phase shift is +/ 180 degrees is called the Gain Margin. That tells you how much more you can increase the gain before the system will be unstable. In Figure 2, the phase shift is 180 degrees at 100 MHz. At 100 MHz the gain is 20, so the LM101A has 20 db gain margin. (The gain can be increased 20 db before the amplifier goes unstable at 100 MHz.)

The smallest difference between +/ 180 degrees and the openloop phase at the 0 db frequency is called the phase margin. That tells how much more phase shift the system can tolerate before it is unstable. In Figure 2 the gain is 0 db near 20 MHz. At that frequency the phase shift is 135 degrees. The phase margin is 45 degrees. (180 degrees 135 degrees.) If an additional 45 degrees of phase lag were added at 20 MHz, the amplifier would go unstable.

Linear system designers add compensation elements to the loop to make sure that the openloop gain is less than 0 db at the frequency where the phase shift is 180 degrees. This requires some skill because things that lower high frequency gain tend to add phase shift as well.

CLOSEDLOOP DESIGN

In general, closedloop control system design requires you to measure the response of the physical components in the systems. For a representative set of frequencies (in the band of frequencies that are likely to occur) you must determine both the amplitude of the response of the system to that frequency, and the phase of the response at that frequency. You can do this directly with a swept sine wave input, or you can determine it indirectly by measuring the step response or impulse response of the system. Knowing this, you can model the physical components as having poles and zeros at particular locations in the complex frequency plane.

After determining the response of each of the components, you can use a computer program

(or a pencil and graph paper) to determine how the system will perform when all the elements are connected together. Usually the analysis is bad news. This forces you to add at least one more component (a compensation filter) with a transfer function that has poles and zeros that will cancel out the poles and zeros that are causing trouble. Once you have done this, the design will give acceptable performance.

The problem is that these filters sometimes need large, expensive inductors. Sometimes the filters need closelymatched components. That means you can't just produce these things on an assembly line. You have to "tweak" each item individually by hand as it is made. Furthermore, the component values may drift with temperature or age, so the system may be stable at room temperature, but not work at 85 degrees. Or it may be stable when it left the factory, but be unstable three months later.

Digital filters don't drift with time or temperature. Digital filters are easily mass produced. They don't need to be tweaked. They don't need inductors or capacitors with tight tolerances. That's why it is so desirable to replace the analog filters in a closedloop control system with digital filters.

There are cookbook methods for designing digital filters with the same characteristics as analog filters. BUT IF YOU REPLACE AN ANALOG FILTER WITH AN IDENTICAL DIGITAL FILTER AND MAKE NO OTHER CHANGES TO THE DESIGN, THE CLOSEDLOOP SYSTEM IS LIKELY TO BE UNSTABLE. Even if by some miracle it is stable, it probably won't have the same characteristics as the analog system.

When the hybrid system doesn't perform as well as the purely analog system, you may be tempted to waste many hours trying to figure out what's wrong with the digital filter. You can check and recheck your digital filter design and you won't find your error because you haven't made an error. The digital filter really is exactly like the analog one. That's not the problem, even though it APPEARS to be the only thing you have changed.

Usually, there is a deadline that has to be met. The product has to be delivered. So, you randomly change poles and zeros until the system seems to be stable. You ship the product, but you don't feel very good about it because you don't really know what the gain and phase margins are, and it could be very close to being unstable. But what else can you do?

THINGS THEY DIDN'T TELL YOU IN SCHOOL

Suppose you break the signal path any place where it is in analog electrical form. Insert a perfect analogtodigital (A/D) converter connected to a perfect digitaltoanalog (D/A) converter at that point. You haven't added any filtering. You haven't changed anything at all. Nothing is different. You may be surprised to discover that the system is no longer stable. That's because you have done four things, any one of which can drive the system over the edge. All four in combination will almost certainly cause the control loop to

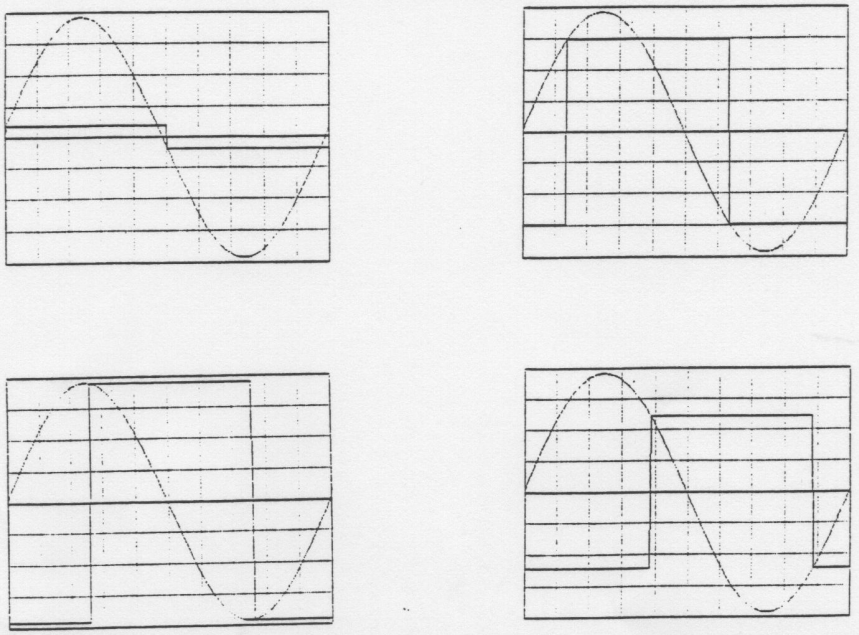

Figure 3. Oscilloscope pictures of sine waves sampled twice at various phases show variable phase and amplitude distortion.

malfunction. Without realizing it you have changed the phase response, changed the amplitude response, introduced harmonic distortion above the Nyquist frequency, and added dither. If you don't compensate for these things the system won't work.

DIGITIZING CHANGES THE PHASE RESPONSE

Figure 3 shows four pictures of sampled sine waves as they appear on a twochannel oscilloscope. In each case you are looking at a sine wave compared with a sampled version of the sine wave. In each case the sine wave is sampled twice per cycle (which is sufficient, according to the Nyquist frequency fable). The difference is that the samples are taken at different phases of the cycle. In general, the sampling frequency isn't phaselocked to the data frequency, so the point at which the data will be sampled is randomly determined.

In the upperleft picture, the sine wave is first sampled at 5 degrees. Notice that the sampled wave isn't quite in phase with the sine wave. In fact, it actually lags the sine wave by 5 degrees.

In the upperright picture, the first sample happens 50 degrees after the sine wave begins. The square wave is definitely lagging the sine wave. If you measure the zero crossing carefully, you will find the square wave lags the sine wave by 49.92 degrees. (It should lag

by 50 degrees, but the oscilloscope has limited bandwidth. The rising edge is not perfectly vertical.)

The two lower pictures show that the square wave lags by 95 degrees when the sine wave is sampled at 95 degrees, and lags by 140 degrees when sampled at 140 degrees.

Most systems will stand a degree or two of added phase, but 140 degrees? That could definitely be a problem. And, you no doubt have noticed, the delay is variable. Since the sampling phase is generally uncorrelated with the data, the amount of phase introduced unpredictable.

But it gets worse. All these figures assume a "perfect" A/DD/A combination. One part of perfection is that the devices are instantaneous. Suppose the A/DD/A conversion process takes 1 millisecond. That delays the output by 1 millisecond. Delaying a 100 Hz signal by 1 millisecond introduces 36 degrees of phase shift. A 1 millisecond delay represents 72 degrees of phase shift for a 200 Hz signal.

In a practical digital closedloop control system, it takes time for the A/D converter to digitize the signal. It takes more time for the microcontroller to read the digitized data into memory. Then it takes some time to do the digital filtering calculations. Then it takes more time to output the result to the D/A converter. None of that time was accounted for in the original analog system analysis.

So, one of the things you MUST do when converting an analog control system to a digital one is to add a delay component to the model that represents the processing delay.

DIGITIZING CHANGES THE AMPLITUDE RESPONSE

You no doubt noticed in Figure 3 that sampling the data twice per cycle introduced substantial amplitude modulation. One reason I picked 5 degrees for the first sample in the upperleft drawing was that if I had picked 0.001 degrees the amplitude would have been so small that you could not have distinguished the signal from the coordinate axis.

This amplitude modulation is greatest when the number of samples per cycle is the lowest, but it is always present to some degree or another. We will talk more about this later.

DIGITIZING CREATES HARMONIC NOISE

Fourier analysis tells us that square waves consist of the fundamental frequency plus healthy doses of odd harmonics. Therefore, the sampled signals in Figure 3 contain not only the fundamental, but substantial third, fifth, and seventh harmonics. In fact, there are

measurable harmonics all the way out past the 127th harmonic. These harmonics can hurt in two different ways.

Suppose the system has some high resonant frequencies. The sampleandhold process moves some of the energy from the fundamental frequency to higher harmonics, which could cause the system to ring if the harmonics match the system's resonant frequencies.

The second way these harmonics can hurt is through the aliasing process. When the number of samples per cycle is low, the harmonic distortion is well above the Nyquist sampling

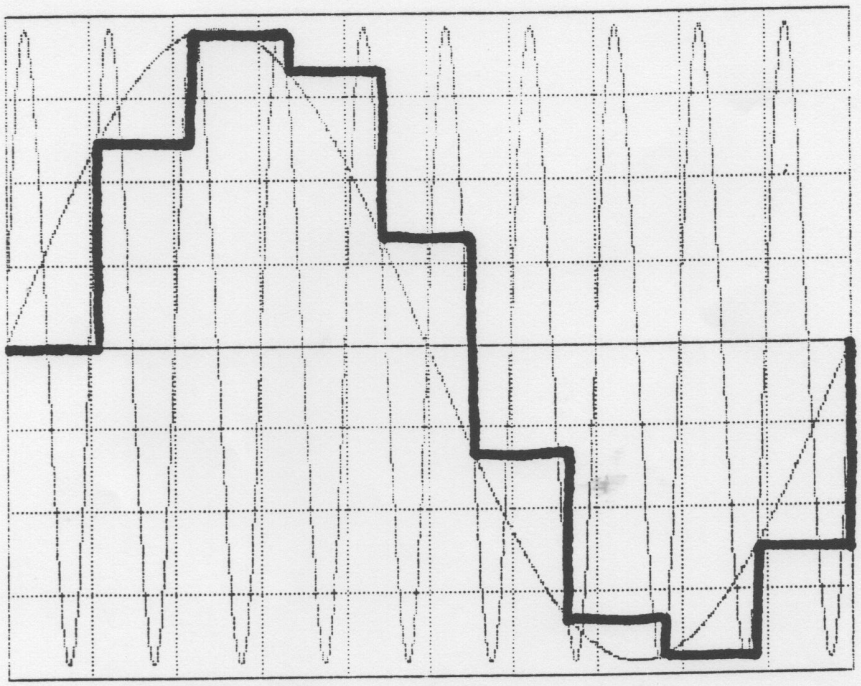

Figure 4. Aliasing causes a 10 Hz signal sampled at an 8 Hz rate to appear to be a 2 Hz signal.

frequency. If this distortion is not filtered out, these high frequencies will be undersampled, folding them down to a lower frequency. Who knows what these bogus signals will do to system performance?

THE NYQUIST FREQUENCY FABLE

The one thing most people think they know about sampled data is wrong. Most people think Nyquist said that two samples per cycle are sufficient to accurately reproduce a sampled waveform. That's not true. What he actually said was that less than two samples is definitely insufficient. If you sample a signal less than twice per cycle, no amount of signal processing, not a perfect "brick wall" low pass filter, not a Cray taking years to analyze

Figure 4 shows how aliasing occurs. Although there appear to be only 3 signals in Figure 4, there are actually 4. The first signal is a 10 Hz sine wave. Since the horizontal scale is 1 second from edge to edge, there are 10 complete cycles of the 10 Hz sine wave. The second signal is a 1 Hz sine wave. The period of a 1 Hz wave is 1 second, so one complete cycle fills the oscilloscope display. The third signal is the 10 Hz sine wave sampled at a 9 Hz rate. This is the bold trace that changes value every 0.11 seconds. The fourth signal is the 1 Hz sine wave sampled at a 9 Hz rate. It exactly coincides with the third trace because the amplitudes of the 10 Hz wave and the 1 Hz wave are identical at those sample times. The sampled 10 Hz wave appears to be a sampled 1 Hz wave. That's why nothing short of psychic processing will correctly reproduce the waveform. (Some very high frequency oscilloscopes are psychic. They take advantage of aliasing to convert signals down to a frequency low enough to be displayed on the screen, but that's another story.)

So, Nyquist said the situation is hopeless if you sample less than twice a cycle. That's a far cry from saying that everything will work perfectly if you sample more than twice per cycle. Sampling twice per cycle only moves the problem from the impossible category to the extremely difficult category.

You have already heard the bad news that sampling twice per cycle can introduce a random phase shift of nearly 0 to 180 degrees. Here is some more bad news: The amplitude of the fundamental component of the sampled waveform can be 0 to +/ 1.3 times the amplitude of the input signal. The amplitude of the third harmonic distortion is 1/3 the amplitude of the fundamental of the output, and the amplitudes of the 5th, 7th, and 9th harmonics are 1/5, 1/7, and 1/9 respectively.

The good news is that things get a lot better when you sample 4 or more times per cycle. Figure 5 shows what happens when a sine wave is sampled 4 times per cycle at four different phases. Fourier analysis of the four sampled waveforms in this figure reveals that the amplitude of the fundamental component of the output waveform is 90% of the amplitude of the input waveform regardless of the phase of the sampling. Furthermore, the phase shift is 45 degrees (half the sampling period) regardless of the phase of the samples. The 3rd, 5th, 7th, and 9th harmonics are 10.45 db, 14.89 db, 17.80 db, and 19.98 db respectively. The only differences in the harmonic analysis of the four pictures in Figure 5 are the phases of the harmonic distortion components. The amplitudes of the harmonics do not depend on sample phase.

Figure 6 shows what happens when a sine wave is sampled 8 times per cycle at four different phases. Fourier analysis of the four sampled waveform in this figure reveals that the amplitude of the fundamental is 97% of the input regardless of the phase of the sampling. Furthermore, the phase shift is 22.5 degrees (half the sampling period) regardless of the phase of the samples. The 3rd, 5th, 7th, and 9th harmonics are 17.12 db, 19.29 db, 23.70 db, and 24.77 db respectively.

Sampling at higher data rates shows dramatic reduction of harmonic distortion at the lower

trillions of trillions of data samples can reproduce the signal correctly. The reason why is that the signal will appear to be another, lower frequency signal.

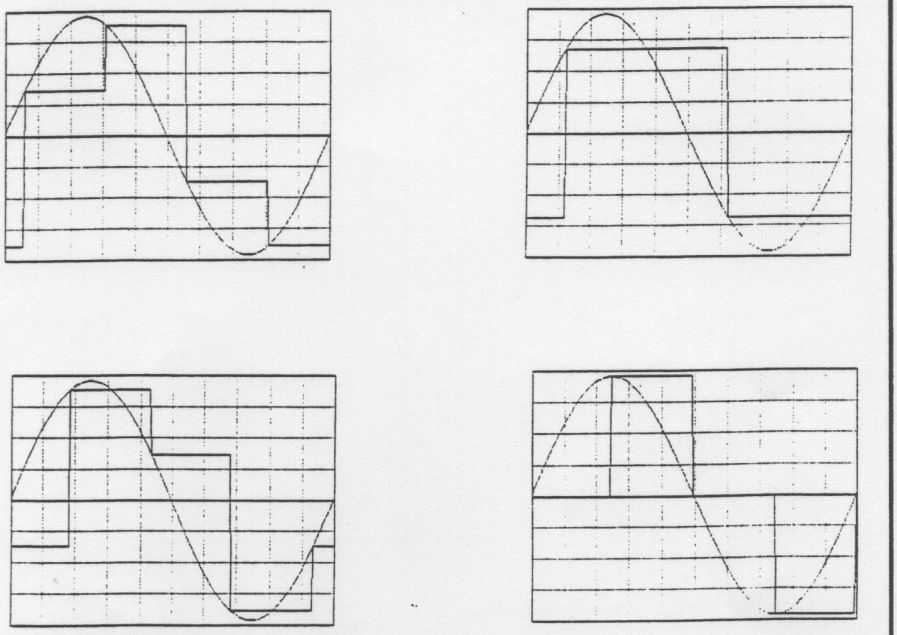

Figure 5. A sine wave is sampled 4 times per cycle at four different phases has constant phase and amplitude distortion.

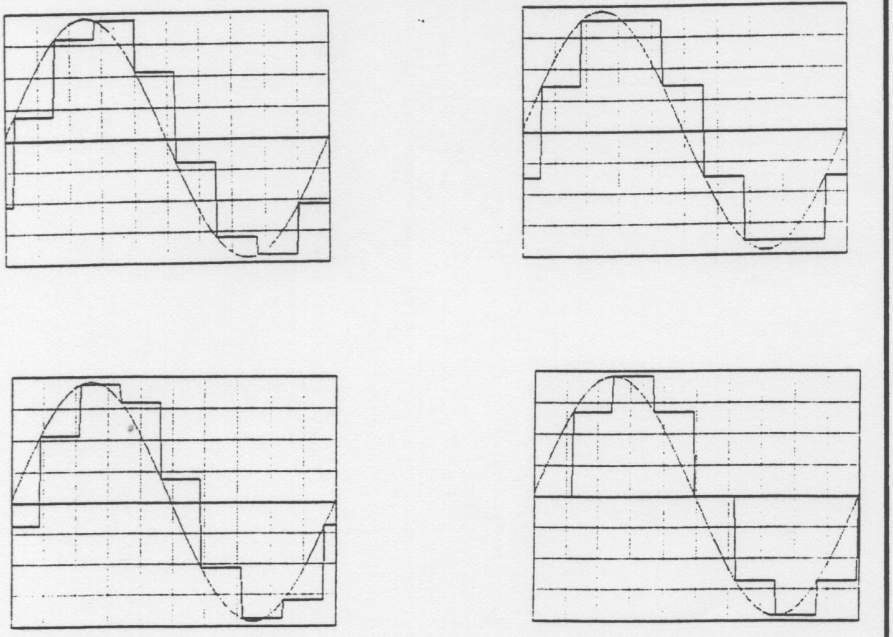

Figure 6. A sine wave is sampled 8 times per cycle at four different phases has less distortion.

harmonics. There is also some reduction of the higher harmonic distortion, but it isn't great. Harmonic distortion for frequencies more than ten times the fundamental tend to fall in the 30 db to 40 db range regardless of sampling frequency.

So, if you sample every signal at least 4 times, the phase shift will be predictable and the

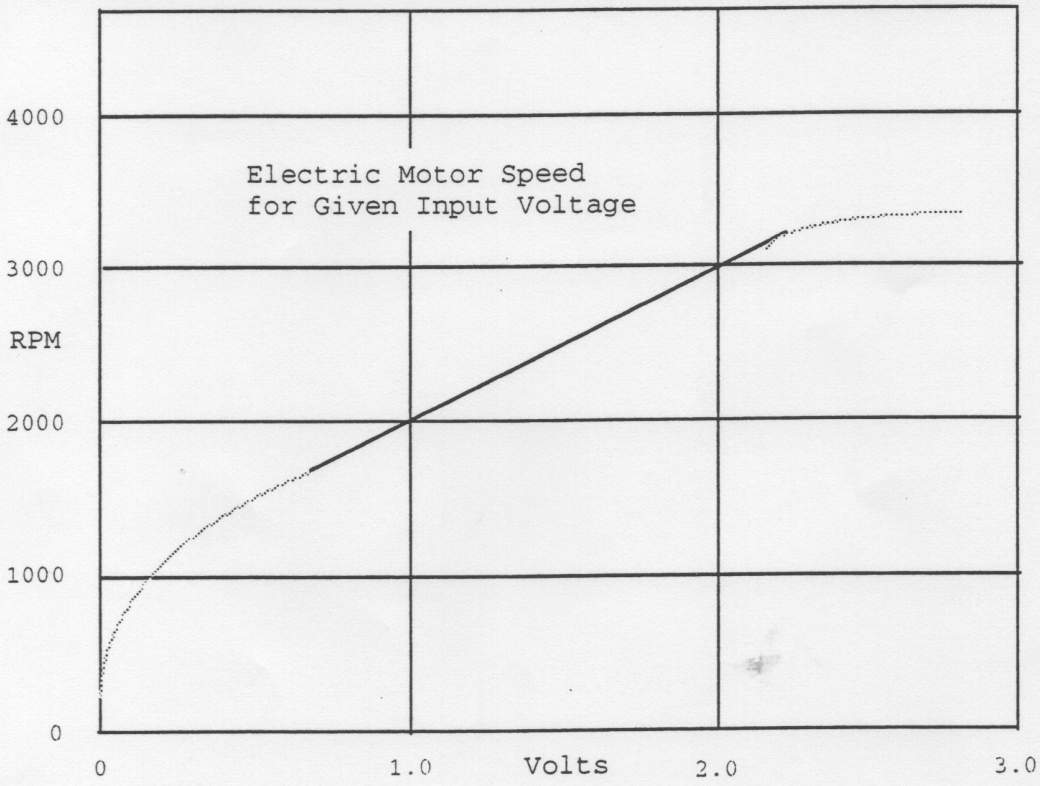

Figure 7. The relationship between steadystate output speed and input voltage for a fictitious electric motor.

amplitude loss will be less than 1 db. (The gain of analog components can easily vary 1 db, so 1 db should be inconsequential to the stability of the closed loop system.) The amplitude of the third harmonic will be 10 db or less, and all higher harmonics will be even lower.

If a signal is sampled 4 times per cycle, the third harmonic will be above the Nyquist frequency. You must drive the third harmonic distortion down into the noise floor to keep it from causing trouble. (We will talk about the noise floor in detail later.) Assuming a 40 db noise floor, that means you will need a filter with 30 db attenuation at the third harmonic of the highest frequency signal.

If you sample every signal at least 8 times, then the 7th harmonic will be 17 db down. You need only 23 db attenuation at this much higher frequency to drive the harmonics down to 40 db. Clearly it gets easier to remove harmonic distortion as the sample rate gets higher because less harmonic distortion is introduced by higher sampling rates.

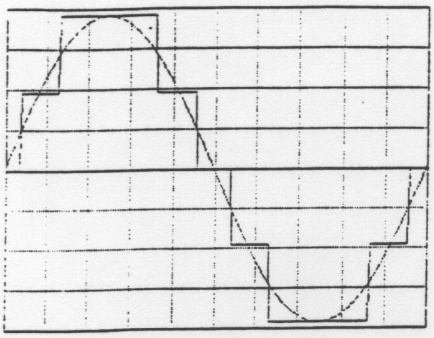

Figure 8. Harmonic distortion created by rounding to a limited number of A/D converter levels also add harmonic distortion, just as the sampleandhold process did.

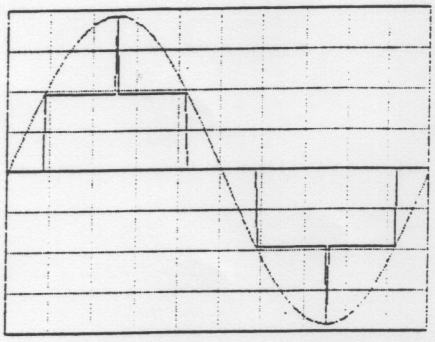

Figure 9. Harmonic distortion caused by truncating toward zero.

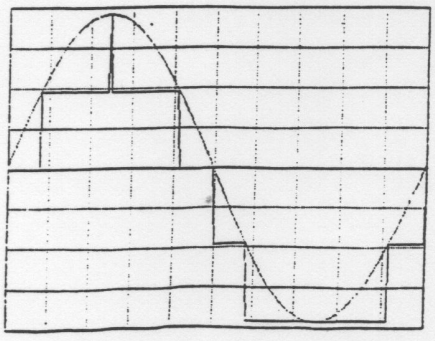

Figure 10. Harmonic distortion typical of a very low resolution A/D converter is bottom heavy

DIGITIZING INTRODUCES DITHER

All the problems discussed so far have been due to the effects of going from continuous time to discrete time samples. There are also problems that arise from going from continuous voltage levels to discrete voltage levels.

Consider the steadystate characteristics of the mythical electric motor shown in Figure 7. For a particular load, the speed is linearly related to the voltage applied (over a limited range of input voltages). Apply 1 volt to the motor and it spins at 2,000 RPM. Apply 2 volts and it spins 3,000 RPM. Any voltage between 1 and 2 volts produces a speed that can be computed from linear interpolation between those two points.

Suppose the output of the D/A converter has a resolution of 0.1 volts. It can produce voltages of 1.0 volts, 1.1 volts, 1.2 volts, etc. That means it can spin the motor at 2,000 RPM, 2,100 RPM, 2,200 RPM, etc.

Suppose the closedloop control system is trying to maintain the motor speed at 2,030 RPM. An analog system would apply a constant 1.03 volts to the motor and hold it there. The digital system can't do that. If it applies 1.0 volts the motor will be too slow, the feedback will detect that, and increase the voltage to 1.1 volts. Then the motor will run too fast, so the system will reduce the voltage back to 1.0 volts (or maybe lower if it isn't stable). The output of the D/A will dither between 1.0 and 1.1 volts, and the motor will always be speeding up or slowing down.

You can reduce the dither by increasing the resolution of the D/A converter. This makes the difference between each voltage level smaller. The dither will still be there, but it might be small enough to be acceptable.

There is always a tradeoff between dynamic range and resolution for a given number of bits. For example, an 8bit D/A with a +/ 15 volt output will have 0.117 volts between output steps. But if the output only needs to

go from 0.5 to 2.5 volts to control the motor from Figure 7, then one could bias and scale the output of the D/A converter such that 15 volts becomes +0.5 volts and +15 volts becomes +2.5 volts. If you do that, there will be 0.0078 volts between output voltage steps. This gives you much better motor control.

Everything we have said about the D/A goes for the A/D as well. If the resolution of the A/D is too coarse, important information gets thrown away. It is important to carefully scale and bias the input to the A/D to take full advantage of its resolution.

Of course it would not make much sense to use a 12bit A/D and assign the value to an 8bit variable, discarding 4 bits. The precision of the arithmetic used needs to be comparable to the resolution of the I/O devices. If the arithmetic precision is low, it is just like using a low resolution A/D or D/A.

The limited number of A/D converter levels also add harmonic distortion, just as the sampleandhold process did. Consider Figure 8. At first glance is looks a lot like a sine wave sampled 8 times per cycle, but it isn't. It was sampled 256 times per cycle, but the sampled value was constrained to the nearest half volt. (In all the oscilloscope output figures the vertical scale has been uncalibrated slightly so that grid lines don't obscure the signal too much.) Therefore, many readings are identical to the previous reading, making it appear that it has been sampled much less often than it actually has.

Careful examination of Figure 8 shows that the changes in voltage levels are not evenly spaced in time. Furthermore, there is no phase lag associated with the sampling because the rounding process anticipates changes in level. Fourier analysis of the waveform shows odd harmonics comparable to a sine wave sampled 8 times, but the phase of each harmonic is either 0 or 180 degrees.

The waveform shown in Figure 8 represents the kind of distortion introduced by floatingpoint calculations when the floating point values are rounded to integers for output. If the floating point values are truncated toward zero, then the output will look like Figure 9. Fourier analysis shows that there is a substantial reduction in the amplitude of the fundamental, and more 5th harmonic distortion.

A/D converters tend to live in an offset world. They commonly consider the negative supply voltage to be 0 and the positive supply voltage to be full scale. Measurements are truncated to the next lower level. Therefore, a signal measured by a very low resolution A/D converter will look like Figure 10. Notice that it is "bottom heavy." This asymmetry adds some even harmonics to the odd harmonics we have seen in all the other examples.

The good news is that all the harmonic frequencies created by quantizing are at least 20 db below the fundamental, and often 30 to 40 db down. In other words, the peak amplitudes of the harmonics of the digitized 1 volt signal are 0.1 volts or less. Since the digitizer that created these harmonics has a resolution of 0.5 volts, these harmonics aren't big enough to flip one bit. They are below the noise floor.

THE NOISE FLOOR

The noise floor limits the dynamic range of an analog system. Suppose for example that an analog circuit is powered by +/ 15 volt power supplies. That means that the biggest signal it can handle, without distortion, is about 14.5 volts peak. (It could be more or less, depending on how much distortion you can tolerate, so it is rather subjective measurement.) Suppose further that when no signals are present there is 10 millivolts (peak) noise. This limits the smallest signal you can detect. The actual value of the smallest signal you can detect depends on how sophisticated you signal processing it. Some applications can detect signals that are less than the noise floor, but lets say that in this case the signal must be at least twice as big as the noise. Therefore, the dynamic range of the system is 20 * Log10(14.5 / 0.020) = 57 db. Therefore, we say that the noise floor is 57 db down from the peak signal level.

(The two questions, "What is the biggest signal the system can handle without distortion?" and "What is the smallest signal the system can detect?" are somewhat subjective. Therefore, the exact values used for the largest and smallest signals may differ, depending on whose definition you use. This will cause a difference in the ratio, which will result in a few decibels of difference in the calculated noise floor.)

If a signal is digitized by an 8bit A/D converter, its negative peak can be as low as 0 and its highest as high as 255. The smallest signal may dither between 127 and 128 (only 1 count from peak to peak). Therefore the dynamic range of 8 bits could be as much as 20 * Log10(255 / 1) = 48 db. (More conservative people might say that the converter should be biased symmetrically around 0, so the biggest signal can only be 254 counts, and the smallest 3 counts, so the dynamic range is only 38 db.) A 10bit A/D converter gives a dynamic range between 50 and 60 db. 12 bits give you 62 to 72 db. 16 bits give you 82 to 90 db of dynamic range.

If you are using 6digit floating point arithmetic, the largest signal could be +/ 999,999 volts and the smallest signal +/ 1 volt, so the dynamic range is 120 db.

You must be sure that you have enough bits of resolution in the A/D and D/A to handle the dynamic range of the system. Furthermore, you must be sure that any harmonic distortion introduced by sampling must be attenuated sufficiently that it won't cause trouble.

HOW TO AVOID THE PITFALLS

(1) Use a high sampling rate to reduce phase modulation, amplitude modulation, and harmonic distortion. I like to set the sampling frequency to be 8 to 12 times the highest frequency above the noise floor. Higher sampling frequencies are better still, but may not be enough better to justify the extra cost. Never sample any signal of interest less than 4 times per cycle.

(2) Add a low pass filter to the D/A output to reduce harmonic distortion. The D/A converter adds harmonic noise that can cause nothing but trouble to the rest of the system. Nip it in the bud. The higher the sampling frequency, the easier it is to build this low pass filter because the harmonics will be lower amplitude and at higher frequencies.

(3) Put an antialiasing filter in front of your A/D converter. Even though you add a low pass filter to the D/A to cut down the harmonic distortion, that doesn't mean you don't have to worry about aliasing. There may be other sources of high frequency noise in the system that an analog system would ignore, but will cause aliasing to introduce false control signals into the loop.

(4) Use adequate resolution. Eight bits will be good enough for some applications, provided you scale the signals to use all eight bits. Just be sure to calculate how much resolution you need, and be sure you have that much resolution.

(5) Keep the processing time short and constant. The more delay you have in a closedloop system, the harder it is to compensate for it. Don't make life difficult for yourself by adding long processing delays. If the processing delay is variable, that makes analysis difficult. Try to keep the processing delay constant so you can account for it.

(6) Completely repeat the linear system analysis and design. You have added a low pass filter on the output of the D/A, added an antialiasing filter in front of the A/D, and introduced a fixed processing delay into the loop. This has changed your polezero constellation considerably. That means you will almost certainly have to change your compensation network. (If you are clever, you may be able to use your added filters as part of the compensation network.)