Ph 77 - Advanced Physics Laboratory

Department of Physics, California Institute of Technology

- Electronics Track -

Feedback Control

One common example of a feedback control system (also called a *servo* system) is that used to maintain a constant room temperature, as illustrated in Figure 1. Here the sun heats the room, a thermometer measures the room temperature, and a controller (called a thermostat in this application) drives an air conditioner to counteract heating from the sun.

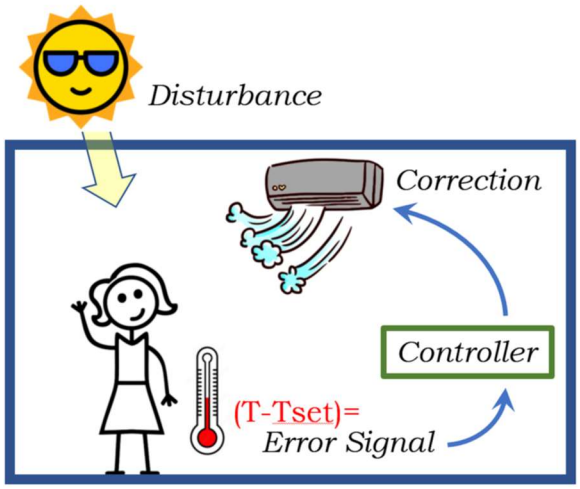

Figure 1. Maintaining a constant temperature in a room is an everyday example of a feedback control system.

If you search for images of "feedback control block diagram" online, you will find all sorts of schematic diagrams that describe a situation like that in Figure 1, with a great variety of different notations. Every field seems to use a different language, but the basic elements of most feedback control systems include:

1) Some physical object you want to control (often called the *plant*). (In our air-conditioning example, the plant is simply the room.)
2) Some means of measuring the quantity you want to control. (The thermometer.)
3) Some means of actuating on the physical object you want to control (The air conditioner.)
4) And a controller (usually electronic) that connects (3) to (2).

Feedback control systems appear everywhere in experimental sciences and technology, and there is a great deal of scientific literature describing their design and optimization. Numerous books on control theory delve into the mathematical complexities at great length, and engineers often spend years mastering this subject. In the Ph77 Electronics Track we will limit ourselves to a summary of the fundamentals that you may find quite useful if you ever find yourself in a situation where you need to control something.

The best way to develop and optimize any feedback control system is by first creating a detailed computational model, and there are software tools for this (for example in Matlab). A good model will allow you to predict the behavior of even complex systems, and thus design accordingly. However,

good models require accurate inputs, especially regarding how quickly the actuator can change the physical system. For the example in Figure 1, the temperature sensor is accurate and essentially instantaneous, but it can take a long time for the air conditioner to cool the room, and it probably does not do so uniformly. A serious computational model would need comprehensive knowledge of the thermal dynamics inside the room, which is a nontrivial problem.

Learning how to do this kind of modeling can be quite involved and time-consuming, and this level of care if often not necessary. You can frequently build a perfectly adequate servo system if you just understand basic feedback-control behaviors. We will take a hands-on approach in this lab by building some rudimentary servo systems and observing how they behave. This will give you an overarching view of basic feedback-control dynamics and the underlying theory, plus you will gain some experience with servo operation. Finally, this introduction provides a reasonable jumping-off point for further expanding your understanding of this subject.

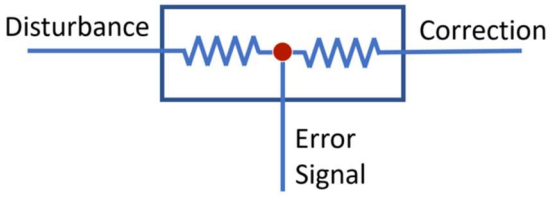

Figure 2. A simple two-resistor network provides an excellent demonstration of basic feedback-control operation and behaviors. In Box52, each resistor is 1kOhm.

A Basic Servo System

Our first step is to consider the super-simple, purely electronic servo system shown in Figure 2 (Box52 in the lab). Here the plant is a small metal box containing two equal-size resistors connected to three BNC connectors, and our (totally contrived) goal is to control the voltage at the red dot shown in the Figure. The voltage at this point is called the *Error Signal*, and the goal of our servo system is to bring that voltage to zero. Obviously, this is not a particularly useful goal, but the simplicity of this system makes it especially easy to model and understand. You can think of it as being like the "hydrogen atom" of electronic servo systems.

Thwarting our goal is the fact that there is some *Disturbance* that tends to make the Error Signal voltage non-zero. The nature of this Disturbance is normally unknown, being caused by some outside influence we cannot measure. All we can do is look at the Error Signal and then apply a *Correction* voltage using the remaining BNC port.

A basic servo system – theory

Figure 3 shows one way to use feedback to accomplish the stated goal of reducing the error signal. Here we simply send the error signal into an amplifier with gain $-G$ and then use the amplifier output as the correction signal. We will assume (initially) that G is a fixed positive constant.

To understand the math in Figure 3, note that the error signal is just the average of D and C. [Why? Because the amplifier has a high-impedance input. Thus, no current flows through the Error-Signal wire, so the two-resistor network behaves as if that wire did not exist.] The correction signal is simply $C = -GE$, so doing the math gives

$$E = \frac{D}{2+G} \tag{1}$$

Looking at some limits, note that setting $G = 0$ gives you $C = 0$ and $E = D/2$, which makes sense for a resistor divider with zero volts on one side. However, taking $G \to \infty$ sends $E \to 0$, which is the goal

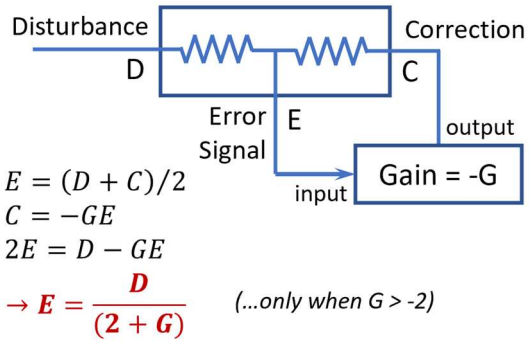

Figure 3. Using feedback to reduce the error signal for the two-resistor servo system shown in Figure 2.

of the servo system. Of course, infinity is a big number, so a practical servo system with this circuit diagram cannot reach the ideal goal of $E = 0$. More on getting around that problem below.

Also, note that the math in Figure 3 assumes that there exists a finite, stable solution to this problem, which is not the case when $G = -2$. Typically we assume $G > 0$, so the feedback circuit reduces the size of E, as desired.

A basic servo system – in the lab

In the Ph77 lab, you can build a feedback control system around this two-resistor network using the layout shown in Figure 4. The function generator creates a known *Disturbance* signal, which sends the *Error Signal* to some nonzero value. The error signal is then amplified and sent into the *Correction* port, with the intention of reducing the size of the error signal voltage. This is called a *Proportional Gain* servo because the correction signal is simply proportional to the error signal. This is not always the best feedback approach, but it is the simplest.

Note that this layout sends the Error Signal to input B of the SR560, with the input set to A-B. This gives a negative gain, as desired.

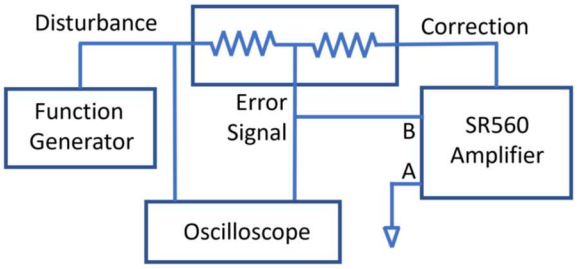

Figure 4. Constructing a feedback-control system around the two-resistor network shown in Figure 2.. The SR560 is set so its input equals A-B.

Exercise 1. Set up this servo system in the lab, starting with G=1. Use a small disturbance signal – say a 100Hz, 100mVrms sine wave, so you don't overload the SR560. Also, include a 1kHz low-pass filter in the SR560 to help with servo stability. Put both the D and E signals on the oscilloscope, as shown in Figure 4, and tidy up the display until you get something that looks like Figure 5. Adjust the SR560 gain and the low-pass filter rolloff frequency to see what happens, then turn the gain up to G=50 to produce a result like that in Figure 5. Add a copy of your screenshot to your e-notebook.

Notice that the servo will begin to oscillate wildly if you turn the gain up too high. When you see that happen, turn the gain back down. You can usually turn the gain higher without oscillating if you include more low-pass filtering. All servo systems are susceptible to feedback oscillations, and we will discuss that later.

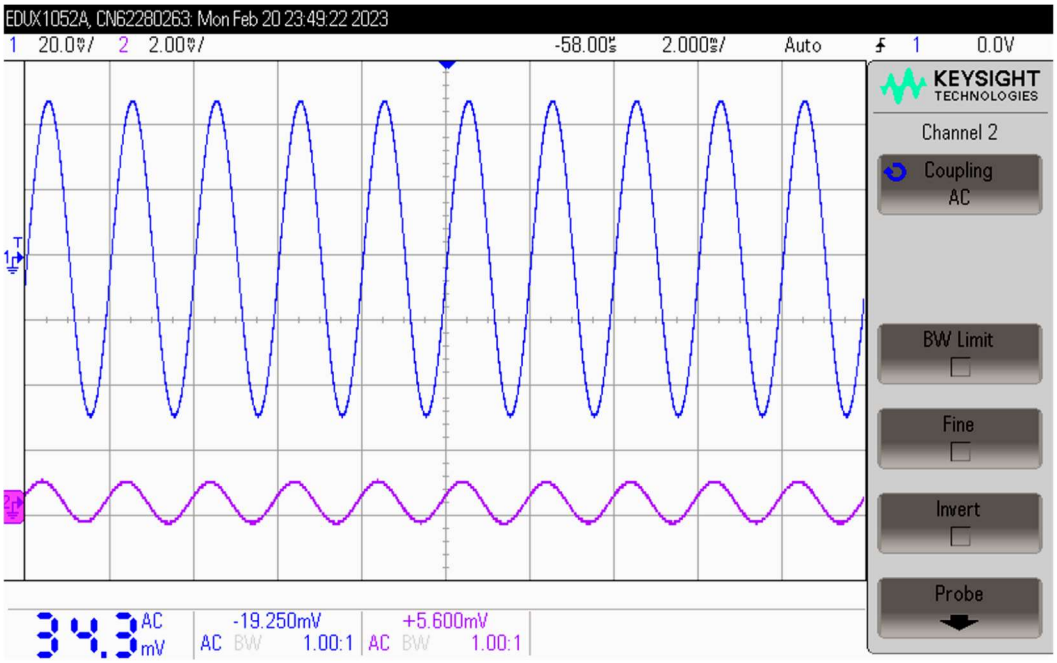

Figure 5. A screenshot showing the servo in operation, with G=50. The top trace shows the Disturbance signal, while the bottom trace shows the Error signa. As expected, the error signal is about 100x smaller than the disturbance (note the different vertical scales in the upper left of the screen).

A basic servo system – comparing experiment and theory

Because this two-resistor-network servo system is both simple and well-defined, theory fits experiment quite well as long as the servo is not oscillating.

Exercise 2. Measure the servo response by using your oscilloscope to measure the D and E signals as you change the SR560 gain G (using the other settings listed in Exercise 1). Start your data-taking at the highest gain because that point will be the most difficult to acquire. On a good day you might get to G=1000, but you want to avoid the servo oscillating at high gains. Stopping at G=100 is fine, but something is wrong if you cannot turn the gain that high. Also, make sure that your SR560 filtering does not greatly reduce the effective amplifier gain at the disturbance frequency.

Feel free to use all your oscilloscope tricks to measure the small E signal – mostly by signal averaging. Also remember that the *Measure* feature yields better results measuring Vrms, because measurements of Vpp are less accurate with noisy signals.

Once you have a high-gain data point, do not change any other parameters (other than G) to take the other data points. Plot the ratio E/D as a function of G on a log-log graph and add a theory curve (with no fit parameters). If all went well, the theory line should nicely go through your data points.

While you have this feedback system set up, try sending in a square-wave disturbance signal. Depending on your gain and rolloff-frequency settings, you should see something like that in Figure 6.

Note that the math in Figure 3 is essentially the same for sinusoidal or DC signals. If D has some constant offset, then E will have a constant offset as well, described by the same theory curve. In

Figure 6, the latter part of each square-wave half-cycle behaves like constant-voltage signal, so you can see this offset effect. In a proportional-gain servo, E will never go to zero if D is nonzero.

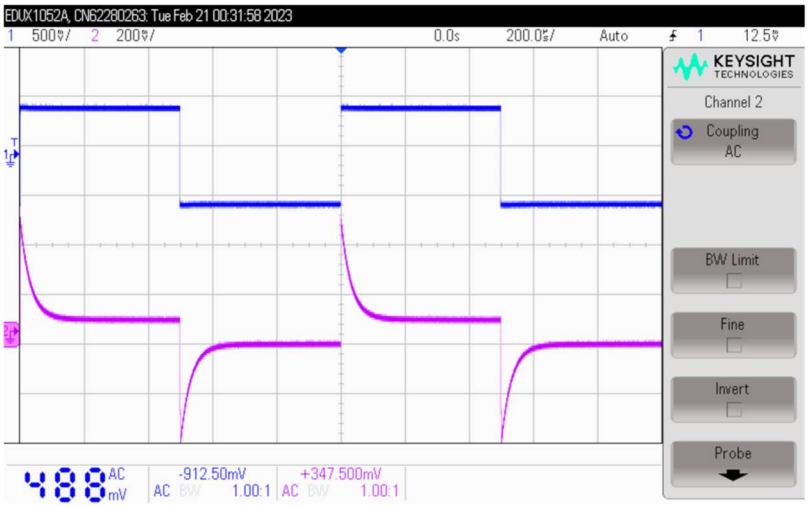

Figure 6. Another screenshot showing the two-resistor network servo in operation. Here a square-wave disturbance signal (top trace) yields the error signal seen in the bottom trace. Note that the error signal (bottom trace) does not go to zero in the constant-voltage regions.

The spikes happen because the servo is not able to suppress the high-frequency signal components that make up the sharp transitions. Put another way, the low-pass filtering in the SR560 means that G falls off at high frequencies, so the servo is not able to suppress those abrupt transitions.

Exercise 3. Using a 100 Hz square wave and an amplifier gain of G=100, produce screenshots like Figure 6 for low-pass filter frequency cut offs of f0 = 10 Hz, 100 Hz, and at the highest f0 you can obtain before the servo oscillates. At the highest f0, zoom in on an initial servo spike and document its approximate FWHM with another screenshot. The SR560 can only respond up to frequencies of about 1 MHz, so this limits the response time of your servo to about 1 μsec. If you push the servo any harder than that, the time lag in the SR560 will make the servo oscillate.

Modeling these square-wave response traces is possible, but tedious, so we will not go into that. Making detailed computational models is often necessary to design and optimize complex servo systems, but that task is beyond the scope of this lab. Our goal here is to develop some intuition by observing and measuring basic servo behaviors.

Integral Gain

When considering the math in Figure 3, we pointed out something of a philosophical flaw in this servo design: If D is nonzero and G cannot be infinite, then you can never achieve your goal of achieving E=0. Thus, in most realistic scenarios, the proportional-only servo geometry inevitably results in some nonzero offset in E. And we can do better.

The standard way to avoid this "offset problem" is to incorporate something called *integral gain* in your servo. Instead of using

$$C(t) = -G_P E(t) \tag{2}$$

in Figure 3, replace this with

$$C(t) = -G_I \int_0^t E(t')dt' \tag{3}$$

where G_P and G_I are both gain constants. This change makes the correction signal C depend on the entire *time history* of E. So, the plot thickens….

If we start out with C=0 at t=0 (by building this into the ON switch of our servo), then C changes with time from that point forward. Assuming some nonzero E at t=0, C will slowly grow in amplitude as the integral does its thing. The correction signal then feeds back through the servo, and this will reduce E (provided the sign of the gain is correct). If all goes well, the servo will drive the system to E=0, and once that happens C will no longer change. You can see this is the case by considering the integral expression.

Integral gain is useful because it can yield a nonzero C when E=0, which is just what you want. Moreover, as the disturbance D drifts around slowly, the servo will automatically adjust C to keep E close to zero. Servo magic at its best…. If this discussion is confusing, read it again, ponder how C changes with time, or ask someone. Integral gain is an important concept in feedback control.

In the real world, one often uses servos with both proportional and integral gain, yielding a combined "PI" servo with

$$C = -G_P E(t) - G_I \int_0^t E(t')dt' \tag{4}$$

In many instances, an additional derivative term (with a gain times dE/dt) is included as well, yielding a "PID" servo. The derivative term can improve the servo performance at high frequencies, better suppressing small disturbance "kicks". But we will not discuss that topic further here and consider only PI servos for now.

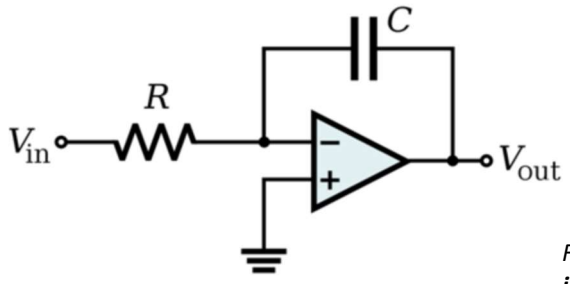

*Figure 7. Here an op-amp is used to make an **integrator** circuit that is often used in feedback-control systems.*

One way to produce a complex servo feedback with any functional form you want is by digital means, and modern thermostats contain a microprocessor that does this. And you can buy "smart" thermostats that try to optimize the feedback algorithm for your specific home, and perhaps even your specific living habits. But while digital feedback wins on flexibility, it can be too slow for some blisteringly fast electronic applications. Then you need fast analog circuits, and Figure 7 shows how you can produce integral gain using a single op-amp (which can have gain-bandwidth products in the GHz range).

In this circuit, the output voltage continues to change for any finite input voltage. Once Vin = 0, however, then Vout remains constant, equal to whatever voltage it happened to have when the zero-input condition was met.

Exercise 4. Analyze the op-amp circuit in Figure 7 in the time domain and show that it yields the expression in the Figure. [Hint: Use Ohm's law for the resistor, $V_R = I_R R$, and add the capacitor-charging law $dV_C/dt = I_C/C$. Then plug those into the op-amp rules we covered previously (see the Amplifiers handout) and integrate.]

Exercise 5. Analyze the same circuit in frequency space using the op-amp rules with the complex impedance for a capacitor to give

$$V_{out} = -\frac{1}{i\omega RC} V_{in} \tag{5}$$

for the case of a sinusoidal input signal. This expression shows:

1) The integral gain is proportional to ω^{-1}, which goes to infinity at zero frequency. This is why integral gain completely suppresses DC offsets. It gives infinite gain at low frequencies only, which avoids servo oscillations.

2) Integral gain is accompanied by a 90-degree phase shift, which plays a role in servo stability, as we will discuss later.

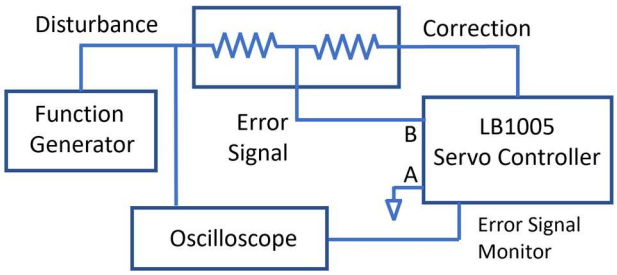

Figure 8. The wiring diagram for setting up a two-resistor network servo using the LB1005 controller.

The LB1005 Servo Controller

Your next task is to observe integral gain in the lab using the Newport LB1005 Servo Controller and the same two-resistor network shown in Figure 2. The LB1005 is a general-purpose instrument that incorporates PI feedback with plenty of adjustable parameters, designed for use in a broad variety of laboratory settings.

Build your servo using essentially the same connection diagram you used before, except replacing the SR560 with the LB1005, as shown in Figure 8. Note also:

1) Set the LB1005 Input Offset dial to 5.0, so no offset is applied to the error signal (more about this later).

2) Instead of sending the error signal directly to the 'scope, as shown in Figure 4, send the LB1005 Error Monitor output to the 'scope instead.

3) Have a look at the LB1005 Operating Manual (available in the lab or online) to get an overview of how the LB1005 works.

Hook everything up and apply a 100Hz, 1Vpp square wave for the disturbance signal and view both D and the Error Monitor on the 'scope. Make sure both channels on the 'scope are DC coupled. To begin, set the P-I Corner to INT (pure integral gain) and set the Gain to 5. You can ignore the LF Gain and the two Sweep knobs, as we will not be using these yet. To turn the servo on, toggle from Lock OFF to Lock ON. You should see a signal that looks roughly like that in Figure 9.

Comparing Figure 9 with what you had previously in Figure 6, notice one important difference. In Figure 6, the error signal did not go to zero in the constant-voltage regions, because previously we only used proportional-gain feedback, which is all you can do with the SR560. In contrast, the LB1005 is set to use integral gain, so now the error signal goes nicely to zero as desired. (If you look closely, you will see that the error signal remains a few millivolts away from zero, and this comes from various voltage offsets in the LB1005. This instrument was not designed to be precise in this way, as eliminating voltage offsets at the millivolt level is a nontrivial task.)

Now change the circuit a bit by sending the error-signal directly to the 'scope, rather than using the LB1005 Error Monitor (see Figure 4). You will now see that the Input Offset knob changes the DC offset of the error signal. This is because the LB1005 subtracts the Input Offset to create a new error signal, and this new error signal is what the controller sets to zero. The offset can be either positive or negative, and the offset is zero when the dial is set to 5.0.

Mathematically, the LB1005 creates a new error signal E' = E+offset, and the servo then drives E' to zero. This feature is useful if you want to set a signal to a finite voltage, rather than to zero volts. Set the Input Offset so the direct error signal goes to (nearly) zero volts, and then go back to looking at the LB1005 error monitor signal, as shown in Figure 8.

If this seems complicated, remember that the LB1005 is designed with users in mind, and users want lots of features… like an input offset knob.

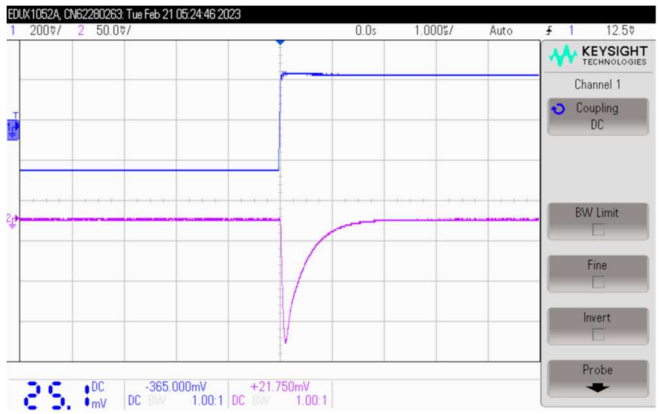

Figure 9. This screenshot is like that in Figure 6, except here the servo uses pure integral gain (I), while Figure 6 used pure proportional gain (P). Here the error signal goes all the way to zero (away from the transition region), while the error signal never went fully to zero in Figure 6. Because the entire goal of a servo is to drive E to zero, this is an important distinction!

Exercise 6. Have a closer look at the high-frequency servo response by zooming in on the transition region, as illustrated in Figure 9. Then set the PI corner frequency to 1MHz and use an input signal amplitude of 500 mVpp. You should be able to turn up the gain and narrow the transition spike to less than 1μsec in width. See how narrow you can make this spike (measure an approximate FWHM, like before), and show a screenshot of your measurement in your e-notebook.

Turn the gain up a bit too high and observe the damped-oscillation signal just before the servo oscillates. This is common in servo systems – they behave well at low gain, oscillate wildly above some gain threshold, and start to show some kind of damped ringing behavior just below threshold. (This happens whenever you give the servo a jolt, as a square-wave transition does). Measure the oscillation frequency and you should get something around 5 MHz. Full-scale oscillations (at higher gain) happen at roughly the same frequency. This frequency is likely determined by delays the LB1005 electronics.

The internal phase shifts become substantial at this frequency, suggesting amplifier time delays of around 0.2μsec.

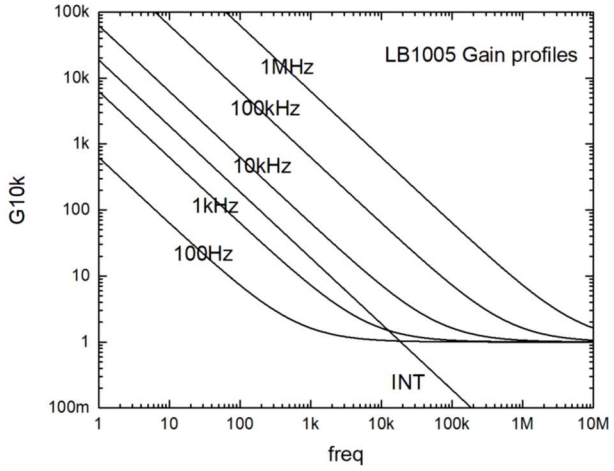

Figure 10. The feedback gain profiles for the different P-I Corner settings in the LB1005 servo controller.

Next steps

Now that you have had a brief introduction to feedback control systems, your next goal is to characterize their behavior in more detail and look at how you can optimize a servo control system for maximum performance.

Begin by setting up your two-resistor network servo as you did before, using the wiring diagram shown in Figure 8. Use a sine-wave disturbance signal with an initial 10kHz frequency and set the P-I corner frequency to 1MHz. Set the Input Offset to 5.0, which means no DC offset is added to the Error Signal inside the LB1005.

Figure 10 shows the LB1005 Gain as a function of frequency for the different P-I corner frequency settings (from the LB1005 user manual). These curves all move up and down in unison as the Gain setting is changed. Note that the INT gain is equal to the 3kHz PI setting at low frequency. Note also that the 1MHz corner provides what is nearly the same as pure integral gain, except the overall gain is higher than the INT curve. One thing to note from this graph is that you cannot get very high gain using pure integral gain (i.e. the INT curve) – this is just the way the LB1005 is configured internally. If you want to have high gain at low frequencies (which is needed in what follows), then you need to use the 1MHz setting.

Exercise 7. Once you have your servo set up, set the P-I corner to 1MHz and turn the gain up until the servo is about to oscillate, but not so high that is actually oscillating. Then measure D and E to obtain the servo gain $G = D/E - 2$ at 10 kHz. Without changing the LB1005 Gain setting, repeat this G measurement as you change the sine-wave frequency over a broad range, giving a plot that looks something like that shown in Figure 11. Reproduce this graph for yourself and add it to your e-notebook.

It can be a challenge to obtain the low-frequency points on this graph, as you need to use your oscilloscope quite effectively when the signal amplitude becomes very low. If you are up for this challenge, go for it. But don't spend a lot of time chasing after these last few low-frequency points, as they add little to the story.

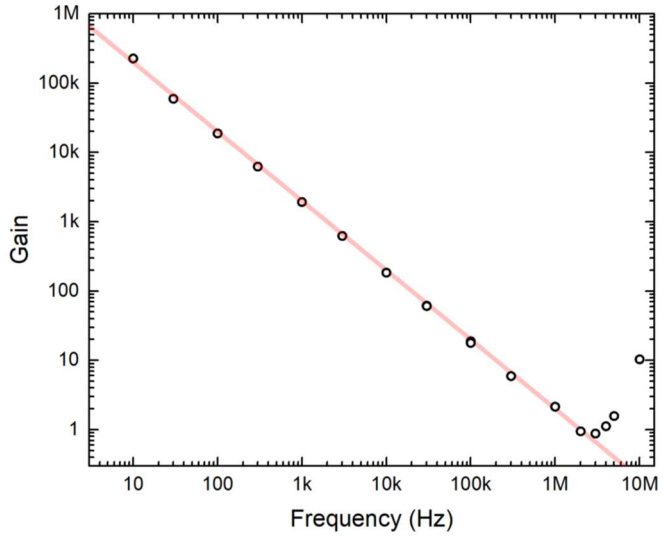

Figure 11. A plot of the servo gain as a function of frequency, as described in Exercise 1. The dots are data points, while the fit curve show a simple 1/f dependence.

We can learn a lot by interpreting this curve, so consider the following points.

>> This is the maximum gain curve you can get for this servo set-up because you cannot turn the gain knob any higher without the servo oscillating. You can turn the gain lower, however, and doing so would yield the same curve, just shifted down. This maximum gain setting is determined mainly by the LB1005 internal phase shifts, so it does not depend on the input Disturbance signal.

>> The basic $1/f$ dependence follows because the 1MHz curve in Figure 10 is essentially pure integral gain below about 1MHz. In our initial investigation of the two-resister network, we assumed that G was constant, independent of frequency. With integral gain, we see that $G \sim 1/f$, and this appears in the direct measurements of G in Figure 11.

>> If you look at the phase information on your oscilloscope, you can observe a 90-degree phase shift between the D and E signals except at the highest frequencies. (Try it.) Again, this follows from the above discussion of integral gain.

>> The behavior above 1MHz is a bit ill-defined from a theoretical standpoint because all the electronic components in this servo (including in the LB1005) do not behave ideally at such high frequencies. There are unwanted low-pass filters everywhere in this regime, even in the BNC cables. Moreover, the LB1005 was not designed to work above 1MHz, and we have no model for what is happening there. As a result, we cannot say much about the detailed servo behavior above 1MHz. However, this high-frequency region plays an important role in bringing about the servo oscillations, and this indirectly affects the servo performance at lower frequencies. And we can understand the overall oscillation behavior even when we cannot model it accurately.

Servo Oscillations

It is no accident that this barely stable servo begins to act oddly at about the same frequency where G=1 in Figure 11. This is called the *unity-gain frequency*, and you can see why it is important by considering the basic servo formula

$$E = \frac{D}{2 + G} \tag{6}$$

that we described above. We assumed that G was a positive constant in that discussion, but the math becomes much more interesting when you realize that G has a phase as well as an amplitude.

For pure proportional gain, the phase is essentially zero (ideally), and the phase is 90 degrees with pure integral gain (again, ideally). In the non-ideal world, however, the phase tends to increase with frequency because of small time delays throughout the servo system. Delays can be introduced anywhere, including in the sensor, the actuator, the controller, the cables, or other places in the plant. For simplicity, we can lump all these delays into an effective phase for G.

For some overall signal time delay Δt, the resulting phase shift will be roughly $\Delta\varphi \approx f\Delta t$, where f is the signal frequency. As one turns up f, eventually the phase shift will increase to 180 degrees, reversing the sign of G in Equation 6. And, following the theory discussion above, this means the servo will become unstable if the magnitude of G is greater than 2.

At the beginning of this exercise, you turned up the servo gain until it was just about to oscillate. And oscillations happen when the overall phase shift reaches 180 degrees while the gain is greater than unity. What Figure 11 tells us is that the phase shift reaches 180 degrees just above 1 MHz, so avoiding oscillations means keeping $|G| < 2$ at that frequency. Because $G \sim 1/f$ is provided by using integral gain in the controller, this means that you cannot turn the gain any higher than what you see displayed in Figure 11.

Another way of looking at this is to note that a proper servo will supply *negative feedback* that reduces the disturbance. However, depending on the amplitude and phase of the gain, the servo might also supply *positive feedback* that will create large oscillations even with no disturbance signal. The transition between these two cases occurs when the phase shift equals 180 degrees at the unity-gain frequency.

There are two especially important take-away messages here:

1) You always want a servo system with high gain, because that is how you reduce the error signal. The highest possible gain, however, is limited by the phase shifts inherent in the whole servo system. Interestingly, even the gain at low frequencies is limited by what happens at high frequencies. If you want to build a better servo, then you need to reduce the delays, thus reducing the phase shifts, as only this allows a higher gain at all frequencies. In other words, to produce the best servo gains at low frequencies, you need to improve performance at the highest frequencies.

2) The phase shifts in a servo are related to how quickly G changes with frequency. If $G \sim 1/f$, then the phase shift cannot be lower than 90 degrees. If $G \sim 1/f^2$, then the phase shift cannot be lower than 180 degrees (meaning the servo will always oscillate). And these are fundamental results that you cannot change. You cannot arbitrarily adjust $G(f)$ without also changing the resulting phase shifts. The full relation between gain and phase is given in the *Kramers-Kronig relation*, which is a general theoretical result beyond the scope of this handout.

Bode Plots

It is customary in servo design to make what is called a *Bode plot*, or *Bode diagram*, in which you plot the servo gain and phase as a function of frequency, in a log-log format. Figure 11 is a rudimentary Bode plot, here showing just the amplitude of the gain. The phase plot is fairly simple in this case, as the phase is basically 90 degrees until it goes off the rails above the unity-gain frequency.

Bode plots are often used simply for displaying and managing servo phase shifts. By engineering smaller time delays in the servo system (which means correspondingly small phase shifts), one can push the unity-gain frequency higher. And, as shown in Figure 11, this usually means better servo performance at lower frequencies as well. As a general rule, the faster a servo can operate, the better its overall performance will be.

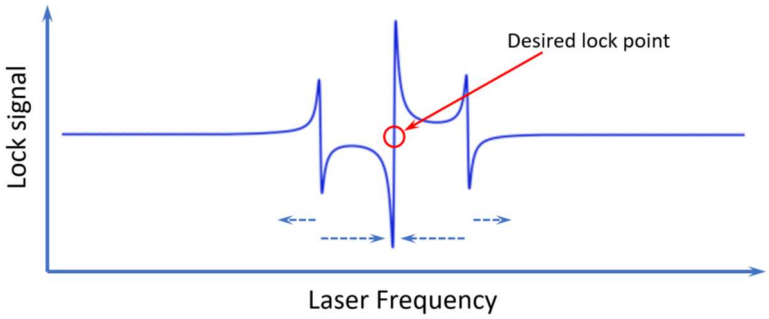

Figure 12. This plot shows a typical "lock signal" that changes with laser frequency. The goal of a laser-locking servo in this case is to "lock" the frequency at the red dot.

The Need for Speed

You may be thinking, why are we focusing so much on servo speed? What difference does it make in practice? To see why speed is often so important in servo design, consider that many servo systems are not nearly as robust as your two-resistor network servo. One example is laser frequency locking, where the goal is to take a frequency-tunable laser and "lock" the laser frequency to a specific fixed value.

If the desired laser frequency comes from a stable optical cavity or an atomic reference line (for example), the lock signal might look something like that shown in Figure 12. This signal is a voltage that depends on the laser frequency, which is derived from the optical reference being used. Ignoring where this signal comes from (for now), this is what you have to work with, and the red circle defines the desired laser frequency.

If you look closely at Figure 12, you can see that there is only a narrow frequency range where a servo system might operate. Immediately to the right of the red dot, a positive lock signal will push the laser to lower frequencies (assuming the controller is set up properly), while a negative lock signal will push the laser to higher frequencies. Over the narrow range shown by the long-dashed arrows, the servo will be stable. Outside this narrow frequency range, however, the controller will actually push the laser *away* from the desired lock point (shown by the short-dashed arrows).

The usual problem with this situation is that the laser frequency is often highly susceptible to outside disturbances. If the laser is locked and you clap your hands, acoustic coupling could push the laser out of the lock region, at which point the laser will lose lock, and it will not come back into lock on its own.

There are two ways to help this situation:

1) Make the laser less susceptible to disturbances (an excellent solution, but this often this requires a lot of effort to achieve), or

2) Increase the speed of the servo. If the servo can react quickly enough, and the gain is high enough, then the servo can remain in the lockable region even if it experiences a jolt.

Designing robust servo systems
Because feedback control systems are so ubiquitous in all branches of science and technology, you may well find yourself faced with the challenge of creating a robust servo system. If so, try to remember these tips:

1) The most important aspect of any servo system is the *plant* (see the discussion associated with Figure 1), so always consider that first. If the plant (including the sensors and actuators) is designed poorly, then do not try to fix this by implementing some kind of fancy controller. That approach may be a waste of time. Once you have substantial phase delays in the plant, this problem likely cannot be fixed with a better controller.

2) When designing your plant, reduce time delays and generally build sensors and actuators that can act quickly. A high-speed (a.k.a. high bandwidth) servo allows higher gains at all frequencies, and thus better servo performance.

3) When designing your plant, also try to reduce the disturbances as best you can. If the disturbances are low enough, then even a poor servo may be good enough.

A Light-intensity Servo

Now for some servo fun. While the two-resistor network is pedagogically useful, it is not something with any practical applications. To move our discussion into the real world, Figure 13 shows an example of something you might actually want to accomplish in the lab using a feedback control system. Here the goal is to produce a constant light intensity in some experiment.

In this sketch, the photodiode measures the quantity to be controlled (the light intensity), and this is monitored using a digital voltmeter (DVM). Your goal is to set up a servo system that maintains a constant DVM output of 1.000 volts (to an accuracy of a few percent at least). If the ambient light varies, for example, the DVM should remain at 1.000 volts. And you accomplish this by shining light from an LED onto the photodiode.

Note that these pieces contain most of what you need in a feedback control system. You have something you want to control (the light intensity), a device that measures what you want to control (the photodiode), and a means of changing the light intensity (the LED). (The DVM just provides a visual monitor for human observers). All that is missing is the controller.

Figure 13. In this servo example, an LED shines light onto a photodetector, and our goal is to "lock" the photodiode output at 1.000 volts.

Exercise 8. Your first task is to design a servo system using the LB1005 that can accomplish this goal. Specifically, this takes the form of a connection diagram that shows how the various parts are used. Naturally, you need to do this before you can begin assembling the system in the lab. Servo design can be tricky when you are new to the game, so here are some instructions and explanations:

>> Figure 13 is the first step, as these are the core elements in your light-stabilization system. We have provided a light-stabilization "kit" for you, shown in Figure 14. This has most of the components

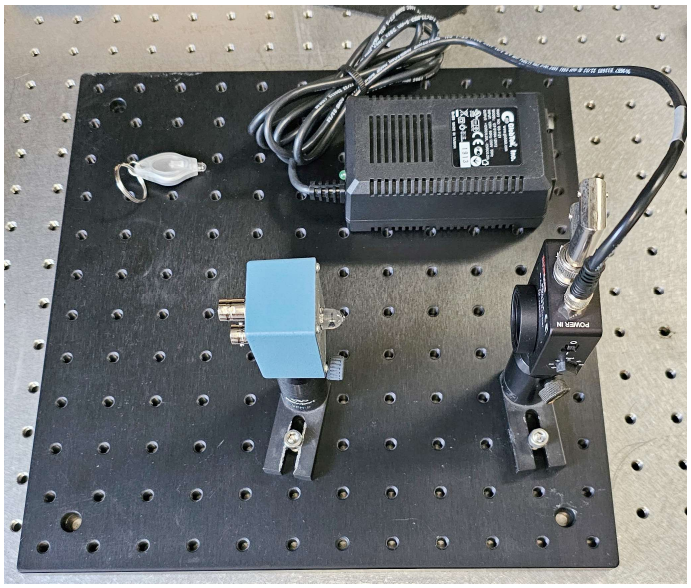

Figure 14. The "kit" of parts for building a light-intensity servo, including an LED, photodiode (PD), PD power supply, and a DVM. Please do not scatter these parts around the lab; leave this kit intact for the next users.

you need, so you do not have to spend all afternoon scrounging around the lab to find things (although that would be the normal situation in most research labs).

>> In addition to the DVM, the photodiode output also needs to go to the LB1005 input, so you first need to choose input A or B. It is often difficult to get this right, because doing so requires that you don't make any sign errors in your design. Often it is easiest to just ignore the sign and try it both ways to see what works. But we will help you out this time: the photodiode output should plug into input B. Input A is grounded by default, so you do not need to plug anything into A.

>> The LED plugs into the LB1005 output, and Figure 15 shows what is inside the LED box. The resistors are in place to limit the LED current so it will not burn out, and you can drive the LED with either (or both) of the BNC inputs. When the servo is operating, the LB1005 output is sufficient to drive the LED and change its output light intensity.

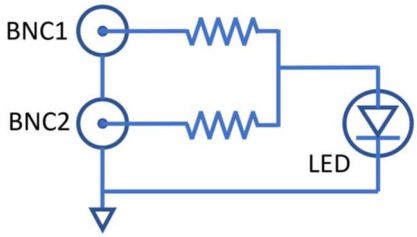

Figure 15. Schematic of the LED box. Both current-limiting resistors are 150Ω.

>> Connect the LB1005 Error Monitor signal to your oscilloscope. When the servo is working, this signal should go to zero. The 'scope will also show you if the servo is oscillating.

When you finish your connection diagram, show it to your TA for confirmation, and correct it as necessary. Make sure you understand the various pieces and what they do. Then put a copy in your e-notebook.

Exercise 9. Once you have a satisfactory wiring diagram, set up and lock your servo. This is a multi-step process, and again it helps to have someone walk you through it the first time. So here are additional instructions and explanations:

>> When the LB1005 is in the Lock Off position, the feedback loop is turned off. Instead, the LED is driven by the Sweep Output, and this allows you to get the optical elements set up properly. Leave the Sweep In BNC open, and use the Output Offset knob to vary the LED brightness. (This knob has a 10-turn range, so you may have to dial it up quite a bit.) You can see the LED brightness change directly, and you should see the PD signal vary as well. Adjust the PD gain so ambient light alone (no LED) gives a signal <1V. Then adjust the Output Offset so you can manually drive the DVM output to the desired 1.00 volts.

When doing this, you are acting as a human servo controller. You watch the DVM and set the LED brightness to give 1.000 volts. If you cannot make this happen, then neither can the LB1005 controller. Make sure to give the servo a good "dynamic range" – you should be able to vary the DVM signal from <0.5 volts to >2 volts at least. When this looks good, set the Output Offset to give a DVM signal close to 1 volt.

>> You next need to establish a "set point" for the servo. Note that the LB1005 error signal is equal to $E = A - B$ - *Offset* (it says so right on the front panel), and *Offset* is set using the Input Offset dial.

Remember that the servo will try to bring the error signal E to zero – that is its whole job. With the LB1005 still in the Lock Off position, send the Error Monitor signal to your oscilloscope while the DVM reading is 1.00 volts. Make sure the 'scope is DC coupled, and then adjust the *Offset* until the Error Monitor signal goes to zero. When you turn the lock on, the servo should send the Error Monitor signal to zero. You want to make sure that this zero-error-signal state corresponds to 1.00 volts of light output.

>> With all this preparation, you are now ready to hit the switch to "lock" the servo. The error signal on the 'scope should go to zero, and you can tweak the Input Offset so the DVM is locked at or near the desired 1.00 volt.

>> Try moving the LED around (by a small amount) to see what happens to the servo, specifically the DVM signal. You should see a *huge* difference between the Lock On and Lock Off states. With Lock Off, the DMV varies quite a bit as you move the LED. With Lock On, the DMV is rock solid at 1.00 volts. However, if you exceed the servo range, then lock will be lost (the green light turns red on the LB1005) telling you that the servo can no longer maintain 1.00V. But if you go back to a favorable configuration, the servo will lock in place once more. Try different PD gain settings as well.

>> When you are done, take a photo of your apparatus, showing the servo locked and the DVM at 1.00V, and add the photo to your e-notebook. Congratulations! You have now set up a perfectly useful light-intensity servo system.

Exercise 10. Now that your light-stabilization servo is working, send a square-wave disturbance signal into the second BNC on the LED box and observe the error signal. As you did above, measure the narrowest spikes you can produce (FWHM) and add a screenshot to your e-notebook. Once again, if you turn the servo gain up a bit more, you will start to see a "ringing" behavior at the square-wave transition. With the gain near its maximum, try shining your phone's flashlight on the photodetector; the DMV should hold solid at 1.00V.

What all this demonstrates is that a ready-made controller like the LB1005 is a versatile servo controller with lots of adjustments you might find useful in the lab. You can apply a sweep signal to view the error signal, change the set point, and change the sign as needed. This controller was designed mainly for locking lasers – either the laser intensity or laser frequency, depending on circumstances – but it can be used for other purposes as well.

Temperature control is another common servo goal, and many different classes of commercial temperature controllers are available. High-temperature applications typically use thermocouple sensors and resistive heaters, and controllers often include an "autotune" capability, where the controller goes into a setup mode that ramps the temperature up and down, automatically finding optimal PID gains for your application. Closer to room temperature, controllers can accept thermistors or other temperature sensors while driving thermoelectric modules that provide both heating and cooling.

We have not gone into much servo theory in this lab, but that is quite a large subject to master. If you need to stabilize a laser intensity in the lab, it is often efficient to ignore theory and just pull out your LB1005 (or something similar). Set it up, adjust the various knobs, and maybe that is good enough for what you need.

Of course, if you are designing a servo system for a spacecraft or some other large project, this try-it-and-see *modus operandi* will not do at all. When you really need to get everything right, that usually means building a detailed numerical model. There are software tools to help you here, but you need to know all the transfer functions and other properties of all the servo elements, so be ready for a long drawn-out process. Big projects are like that.

The take-home message: if you need a servo system in a research lab, it is usually best to buy an appropriate controller already designed for your application. You often need not know what is inside the box; you just need to know how to hook it up, like you just did with the LB1005. If you have an especially demanding application (e.g. a spacecraft), get ready for some serious design work.