Laboratory Assignment 2
Direct Discrete-Time Frequency-Domain-Based Design

2.171 Analysis and Design of Digital Control Systems
Department of Mechanical Engineering
Massachusetts Institute of Technology
Cambridge, MA 02139
Due: In checkoffs on Friday, Ocotober 27, 2006

1 Main Topics

- Servomotor control.
- Proportional and Proportional-Integral velocity control via approximation to continuous-time systems.
- Proportional-Integral-Derivative position control via continuous-time approximation.
- Integral anti-windup.
- System identification via frequency response testing.
- Servomotor control via loop shaping (lead and lag compensation).
- Effect of sensor quantization on servo performance.

2 Equipment

- Personal Computer with DS1102 (dSPACE, Inc.) Controller Card
- Matlab, Simulink, Real-Time Workshop, Real-Time Interface
- Digital Storage Oscilloscope
- Breadboarding System with Analog Electronics
- Function Generator
- Servomotor with Tachometer and Potentiometer
- Linear Power Amplifier

3 Lab Overview

In this laboratory, we study digital control of a D.C. servomotor. Each station is equipped with a D.C. motor having an integral tachometer to measure angular velocity, a single turn potentiometer to measure angular position, and a power amplifier to drive the motor. First, we develop a model for the motor. This model is then used to design velocity and position controllers. For the purpose of this lab, controller design is done using discrete time methods and equivalents.

Initial experiments focus on velocity control. We implement a proportional velocity controller first. With this control scheme, we observe steady-state errors. Also, we observe that increasing gain reduces the steady-state errors. Next we implement a PI velocity controller to eliminate these steady-state errors. Problems with integral windup are observed and an anti-windup scheme is implemented. We meet natural frequency and damping ratio specifications by designing a continous-time controller which is mapped into discrete-time.

Finally, we implement lead-lag position control of a servomotor.

Some of the non-idealities that you will see in this experimental hardware include an internal 700 Hz resonance of the servomotor (under velocity control), and the discontinuity that occurs at the ± 10 V transition in the potentiometer (under position control).

4 Power Amplifier Connections

Each station is equipped with a D.C. motor and a power amplifier. As shown in Figure 2, the power amplifier provides a voltage gain of 2 for the signal from the DAC. The power amplifier connections have already been made for you.

Note: The power amplifiers have an absolute maximum of ± 25 volts. Irreversable damage will occur above this level. The supplies in the lab can provide more than ± 30 volts and thus are capable of destroying the amplifier. Therefore, before shutting off or turning on the supply *always* set it to 0 volts on both sides. As connected, the right-hand supply provides $+V_{cc}$ at its positive terminal, and the left-hand supply provides $-V_{cc}$ at its negative terminal. Common is taken from the jumpered center connections (the negative terminal of the right-hand supply and/or the positive terminal of the left-hand supply).

Set the current limits on both supplies to the maximum. Whenever you are about to turn on the supply, first adjust both voltage controls to zero. Set the right-hand meter to monitor voltage. Set the left-hand meter to measure current. Then power up the supply. At this point you may increase the supply voltage up to the desired ± 17 volt level. (The reason for choosing this voltage is that it is comfortably below the amplifier's absolute maximum rating of ± 25 volts.) We have pre-wired the amplifier to the supply for you as shown in Figure 1.

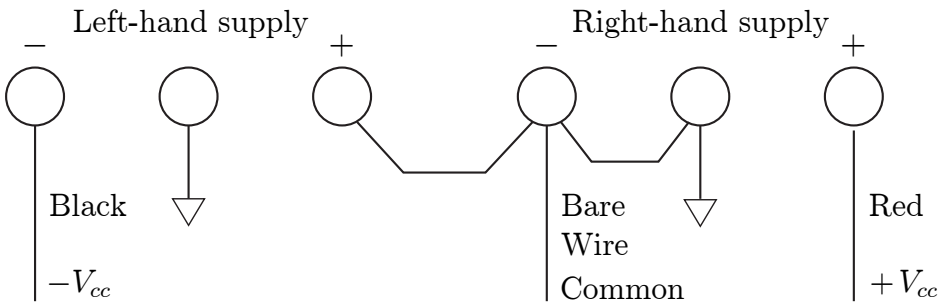

Figure 1: Amplifier/Supply Connections

The negative and positive terminals of the power op-amp are brought out on green and red wires respectively. The board ground plane is also brought out on a black wire for connection on the low-power side. On the high-power side, we have brought out the power supplies with $+V_{cc}$ on a red wire, $-V_{cc}$ on a black wire, and common on an unisulated wire. The amplifier output is brought out on a yellow wire along with another connection to common on a black wire.

5 D.C. Motor model

Each station has a D.C. motor and a power amplifier. Please follow the above precautions before powering up your setup. The D.C. motor has a tachometer to measure velocity and a potentiometer to measure angular position. The motor parameters are:

$$
\begin{aligned}
R &= 7.5\ \Omega \\
L &= 5.55\ \text{mH} \\
\Rightarrow \tau_e &= \frac{L}{R} = 0.7\ \text{ms} \\
K_T &= 0.024\ \text{Nm/A} \\
J &= 1.5 \times 10^{-5}\ \text{Kgm}^2\ \text{(all loads referred to motor shaft)} \\
K_{\text{tach}} &\approx 0.023\ \text{V/rad/sec} \\
K_{\text{pot}} &= \frac{V_{pp}}{2\pi \cdot (\text{gear ratio})} \approx 0.5\ \text{V/rad (of motor)} \\
\text{No. of teeth on pinion} &= 36 \\
\text{No. of teeth on driven gear} &= 216 \\
\text{Gearing ratio, N} &= 6 \\
\Rightarrow \tau_m &= \frac{JR}{K_T^2} \approx 200\ \text{ms} \\
\Rightarrow \frac{\omega_m(s)}{V_m(s)} &= \frac{1}{K_T(\tau_m s + 1)} \qquad (1)
\end{aligned}
$$

In this derivation, we are assuming that the motor inductance is low enough to be ignored, i.e., $L \simeq$

0. Substituting numbers gives the transfer function of this system from applied motor voltage V_m to motor angular velocity ω_m as

$$\frac{\omega_m(s)}{V_m(s)} = \frac{41.7}{0.2s+1}. \tag{2}$$

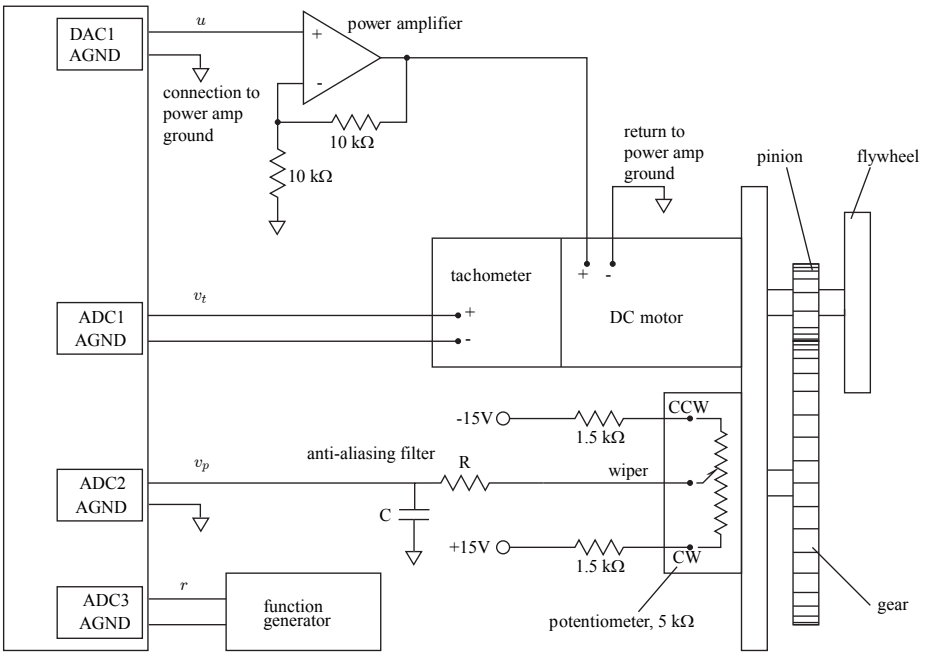

Figure 2: Connections for the D.C. motor setup including an anti-aliasing filter with the values $R = 100\,\mathrm{k}\Omega$ and $C = 0.1\,\mu\mathrm{f}$.

The connection of the motor setup is shown in Figure 2. For the purposes of this lab, the reference signal r is generated by the function generator. In your pre-wired lab kits, the function generator is connected to AD channel 3, the tachometer signal connects to AD channel 1, and the potentiometer signal connects to AD channel 2. The output of DA channel 1 connects to the input of the power amplifier.

In order to match the A/D input range, we connect the potentiometer to operate at a little less than ± 10 volts (± 9.4 V). To do this, we use the ± 15 volt power source from the protoboard, and use a pair of resistors to reduce the voltage to the desired value. The value of the potentiometer resistance is 5 kΩ, thus the pair of voltage dropping resistors are chosen as 1.5 kΩ. These resistors are soldered in place on the motor setup, so you don't need to supply them on your protoboard.

The complete block diagram contains the power amplifier, motor, tachometer, and potentiometer using the parameters given above. Throughout this lab, we will drive the motor with the power amp configured for a voltage gain of 2.

The complete system transfer functions include the sensor gains in addition to the physical con-

stants. Reducing the system block diagram brings us to the to the transfer function from DAC output u to tachometer voltage v_t,

$$\frac{V_t(s)}{U(s)} = \frac{2K_{\text{tach}}/K_T}{\tau_m s + 1} = \frac{1.92}{0.2s + 1} \tag{3}$$

Similarly, we arrive at the relationship between the DAC output u and the potentiometer voltage v_p,

$$\frac{V_p(s)}{U(s)} = \frac{2K_{\text{pot}}/K_T}{s(\tau_m s + 1)} = \frac{41.7}{s(0.2s + 1)} \tag{4}$$

6 Velocity control

Use the motor model developed in the previous section for designing discrete-time velocity control of the D.C. motor. The tachometer output is used as the feedback signal. In this section of the lab, first use a sampling time of T = 1 msec for the implementation on the real-time hardware and complete all indicated sections. This is fast enough that the effects of sampling are negligible, but it makes interesting interactions with a 700 Hz resonance in the motor. Also look at the effect of redesigning the loop for faster sample rates. Once you have seen the effect of a relatively low sample rate, decrease the sample time to T = 0.1 msec. What changes do you observe? Explain.

6.1 Proportional Velocity Control

a) Calculate the ZOH equivalent for the plant represented in Equation 3.

b) Design a proportional velocity controller of the form $G_1(z) = K_p$, which implements a crossover frequency of $\omega_c = 50\frac{rad}{sec}$ and thus $\Omega_c = 50T\frac{rad}{sample}$. Show your calculations. Implement this controller on the real-time hardware. What mechanical impedance (i.e., what equivalent spring/mass/dashpot) is created at the output shaft? How does this impedance vary with changing controller gain K_p? Why? (You can sense this impedance most easily at the potentiometer shaft, since the gearing makes this the higher stiffness point in the drivetrain.) The impedance is most easily felt with the velocity set to zero.

c) What happens as you increase the gain K_p above its nominal design value? What limits the achievable bandwidth?

d) Now please perform system identification via the swept-sine method. Download the dynamic signal analyzer files into your working directory. Create a file like `dsa_demo.mdl`, as shown in Figure 4. This demo file is set up to test the frequency response of a lead compensator. Compile and download this file with `Tools:RTW Build`. Next, go to the MATLAB command window and type the line: `mytf=dsa_tf(logspace(1,3,10),1)`. This starts the operation of the dynamic signal analyzer. You will be asked a set of questions at the command line, and should accept the default answers for all of them. This script file sweeps through a set of input sinusoids, and records the resulting output waveform. It then measures the magnitude and phase shift between them.

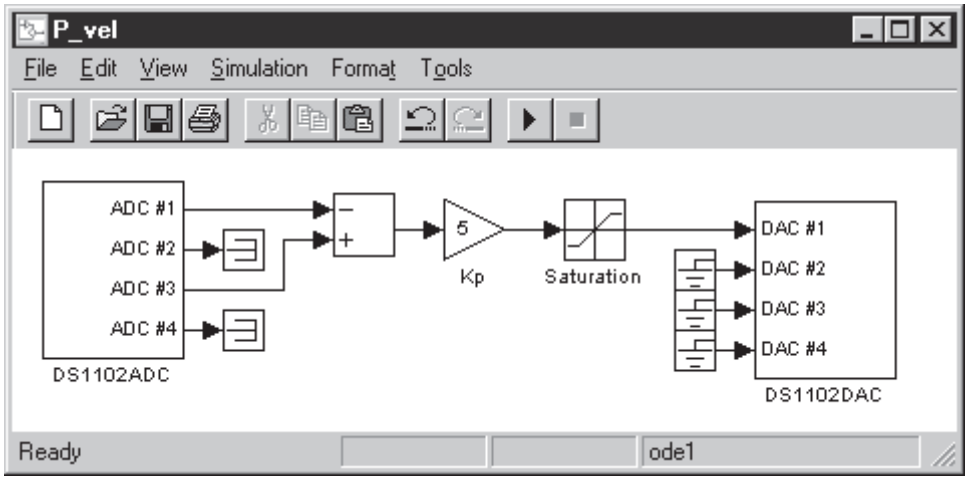

Figure 3: dSpace block diagram used for the implementation of proportional velocity control.

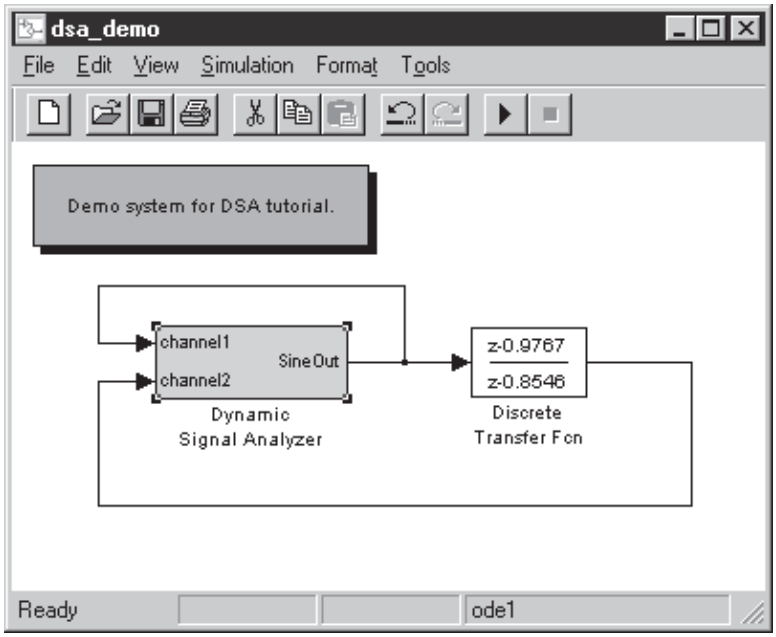

Figure 4: Simulink file `dsa_demo.mdl` used to demonstrate the operation of a dynamic signal analyzer.

The dynamic signal analyzer will create two windows, as shown in Figure 5. One allows you to inspect the input and output signals for saturation or poor signal-to-noise. The other builds an experimental bode plot. When complete, you can examine the data, and type `help dsa_tf` to see the options available.[1] Perform the system identification on the motor velocity plant from Equation 3 using first a sample rate of 1 kHz and then 10 kHz. Explain the results.

e) Repeat your measurements in b) and c) with T = 0.1 msec.

[1]This dynamic signal analyzer was developed by Katie Lilienkamp as part of a Bachelor of Science thesis project in the Department of Mechanical Engineering at MIT.

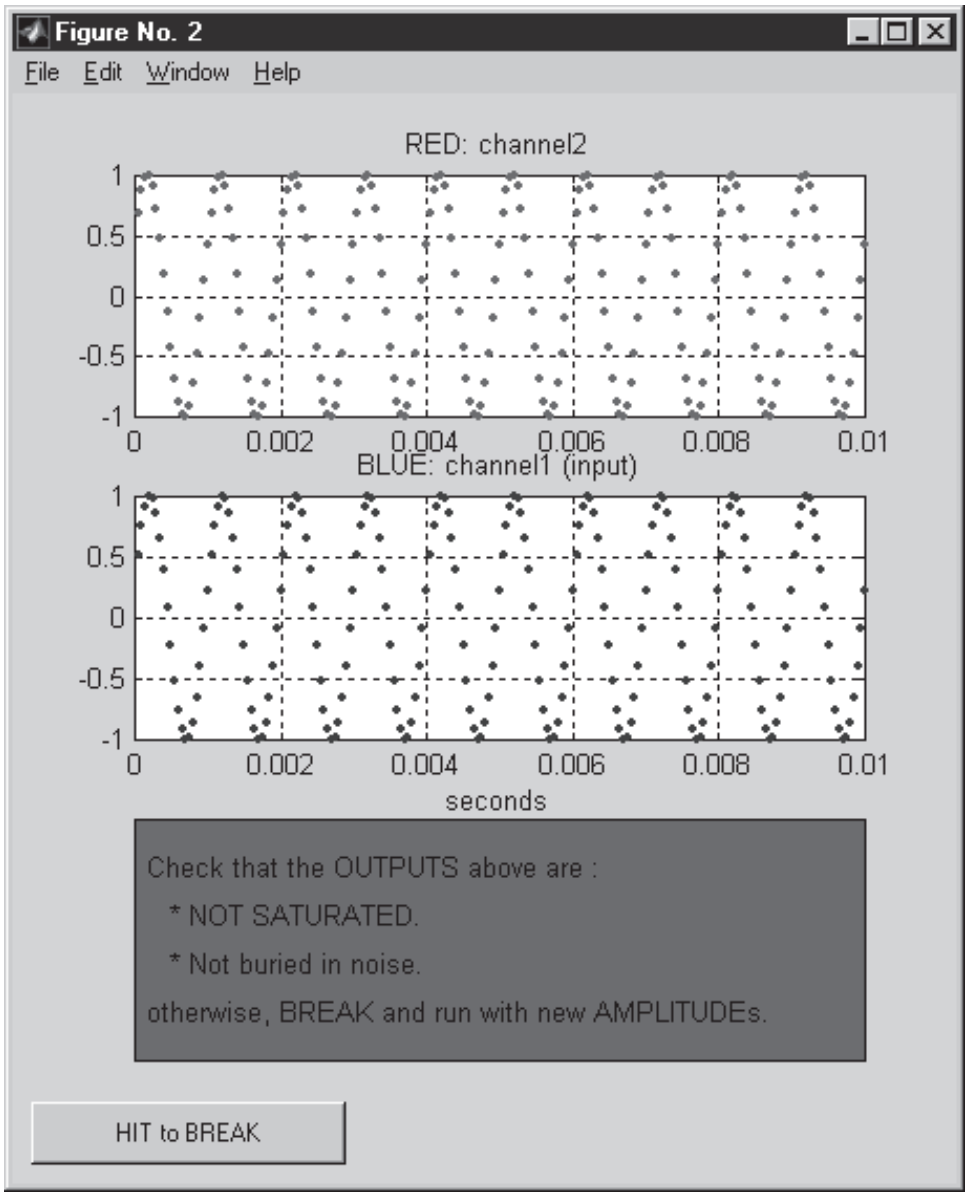

Figure 5: Output windows of the dynamic signal analyzer demonstration program.

6.2 PI velocity control

Note: For this section and the remainder of the lab use T = 0.1 msec.

Next, we implement PI control which removes the constant errors that were present with a proportional controller. Use a PI controller of the form

$$G_2(z) = K_p\left(1 + \frac{K_I z}{z-1}\right)$$

Use a discrete time integrator to implement the integral block. The presence of an integrator in your control law can lead to *integrator windup*. This can be noticed if you cause the error to accumulate in spite of saturation of the actuators. The integrating action of the controller keeps accumulating the error and leads to large overshoots. You can observe this by setting a constant velocity command, and by imposing a velocity error on the flywheel by clamping it with your hand. Be careful not to get yourself hurt while doing this (*i.e., Keep your fingers out of the gear mesh*).

When you release the flywheel, you should be able to see a rather large overshoot. For the purpose of observing windup, use the DC offset of the signal generator to set the velocity to an essentially constant level. Note that the size of the overshoot is dependent on the time for which you hold the flywheel stationary. It is also interesting to conduct this experiment with the velocity command set to zero.

a) Design a PI velocity controller which has a crossover frequency of $\omega_c = 30\frac{rad}{sec}$ and thus $\Omega_c = 30T\frac{rad}{sample}$, and with $\phi_m \geq 45°$. Show your design. Explain how you calculated the necessary values of K_p and K_I. Implement this controller on the real-time hardware.

For a non-zero DC velocity reference, and with the integrator saturation limits set to infinity, clamp the flywheel with your hand for some period of time, and notice the large overshoots when you release it. Modify the integrator saturation limits to prevent windup and re-run the program. Again, impose velocity errors by holding the flywheel with your hand. Notice the greatly improved recovery performance when the integral term is bounded. Explain the difference in the response. You can also most conveniently use the Control Desk interface to set the integrator limits without recompiling.

b) What mechanical impedance (i.e., equivalent spring/mass/dashpot) is created at the output shaft under PI velocity control? Why? Now use the Control Desk interface to set some explicit limits on the integrated output. Where should you set these limits and why? How does the mechanical output impedance change when anti-windup is implemented? Why? (This impedance is best sensed with the velocity command set to zero.)

7 Position control

Position control is in some sense more difficult than velocity control, due to the additional integrator in the transfer function. In position control, feedback is taken from the potentiometer. We use a one-turn 5 kΩ servo potentiometer in this setup. The term servo potentiometer means that it is specified for continuous rotation with a reasonable life time. Conventional pots will have a rotation stop at the extreme of rotation, and also will not endure continuous mechanical operation. Given the selected 1.5 k series resistors and for the ± 15 V supplies the potentiometer voltage output varies from -9.4 to $+9.4$ volts. There is a discontinuity at the end of the range as the voltage transitions from the maximum negative to the maximum positive.

We have used the above information when developing the relationship between potentiometer output voltage and motor shaft angle. This relationship also depends on the gear ratio of the drive

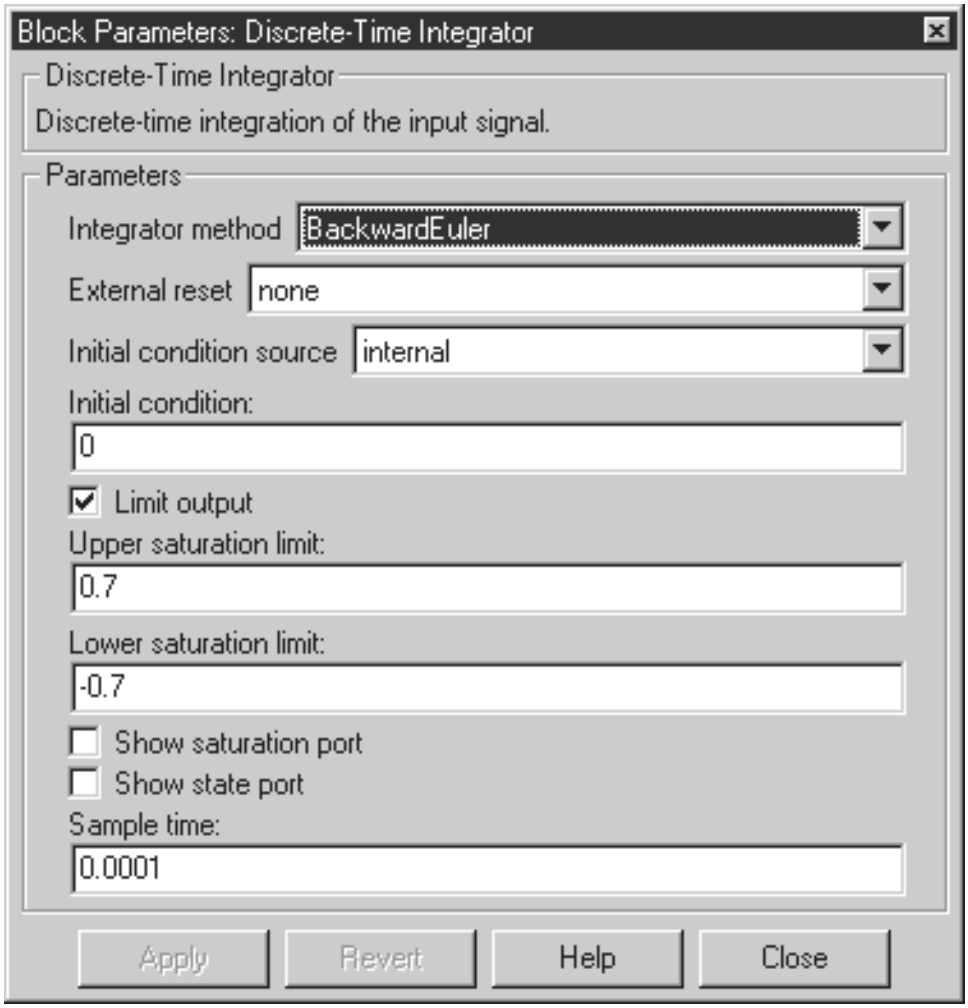

Figure 6: Contents of the discrete-time integrator block allowing the user to limit the integrated value, and to apply anti-windup to the controller.

train. Note that due to the way the potentiometer is connected, the motor angle is 6 times the potentiometer angle. The potentiometer angle is what you measure. However, in the model, we refer all variables to the motor shaft.

We have added an anti-aliasing filter after the potentiometer, and before the A/D input. This anti-aliasing filter consists of a passive RC filter with a time constant of 10 ms. Its phase lag is significant at our target crossover frequencies, and so you should include it in your plant model. The specific values used for the components are $R = 100\,\mathrm{k}\Omega$ and $C = 0.1\,\mu\mathrm{f}$.

a) Create a ZOH equivalent from the plant in Equation 4 and including the anti-aliasing filter.

b) Design a lead-lag compensator using the servomotor plant model to implement a position controller which achieves a crossover frequency of $\omega_c = 50\frac{rad}{s}, \Omega_c = 50T\frac{rad}{sample}$, and a phase margin of 45 degrees. Solve for the poles and zeros of the resulting discrete-time transfer

function. Plot these poles and zeros on the z-plane. Sketch a discrete-time Bode plot of the plant transfer function. Implement your controller on the real-time hardware. Use a discrete-time transfer function block to enter the controller coefficients. Be sure to keep the lag portion of the controller separate so that you can implement integral anti-windup. Separating the lag portion of the controller is a practical detail that should be addressed in all controller designs. Design for a controller update rate of $T = 0.0001$ seconds. Record the system step response and compare with that predicted in your paper analysis. Note that the anti-aliasing filter creates significant phase-shift in the vicinity of crossover and so its dynamics must be accounted for in your design. Label the plot axes with frequency in rad/sec and also in radians/sample so that you can think easily between continuous and discrete-time frequencies.

c) Use the dynamic analyzer to measure the response of the plant and the compensated system. Update your model in b) if necessary.

d) With the controller running, notice the effect of increasing the gain K_p on the servo stiffness. Do you notice any deterioration of position stability with increasing loop bandwidth? What is this due to? What elements in the system limit the ultimate performance?

e) With the controller running, and with a fixed reference, try manually rotating the potentiometer shaft away from a fixed reference point. Notice the restoring torque of the servo increases with time due to the integral control term. Record the transient response of recovery from this induced position error. Can you see the rather large overshoots caused by integral windup? Try this experiment both with and without integral anti-windup and comment on the difference.

8 Quantization

Quantization of the position sensor input often limits the performance of position control systems. In our system, we have quantization due to the analog-to-digital converter. In other systems, the postion sensor is often an encoder or other finite-resolution device. Quantization effects are difficult to see at the 16-bit level with our hardware, so in this section of the lab you will artificially limit the resolution of the A/D converter by adding a quantization block. Lower-resolution A/D converters (such as 8-bit models) are typically less expensive and faster than 12-bit and 16-bit versions. They are also often included with stand-alone microprocessors, and so may be the best converter for a particular application. However, one needs to understand the performance limitations they create.

a) Modify your position control loop to include a quantization block limiting the resolution of the position sensor. Try running the controller at several different quantization levels, compiling and downloading after change. At what point do you notice the position response start to degrade? How would you alter your controller if you were forced to use a low-resolution sensor? Does having a limited sensor resolution affect your ability to take a swept-sine test with the dynamic signal analyzer? What does the position "noise" look like as the quantization is changed? What do the associated control outputs (DAC#1) look like? Why?