MASSACHVSETTS INSTITVTE OF TECHNOLOGY
Department of Electrical Engineering and Computer Science
6.302.0x—Introduction to Feedback System Design
January 2016

Notes Set 2

Where we've been, where we are, and where we are going next.

In each week of this three-week class, we are trying to weave together three concepts: a strategy for constructing mathematical models for a system of interest, a technique for characterizing that model's behavior, and an approach to feedback control based on insights from the modeling and characterization.

In the first week, we characterized homogenous LDE models. We showed that their natural frequencies could be used to determine important model behaviors: growing or decaying, monotonic or oscillatory, and so on. In parallel, we examined how to construct homogenous LDE models for systems like the wall-seeking robot or the mutinous crane. And in the first lab, we used homogenous LDE models and their natural-frequency-based characterization to examine the up- and down-pointing copter-levitated arm. In particular, we derived a simple, and then a more complicated, model and used the model's natural-frequency-based characterization to show that together, gravity and air resistance were responsible for stabilizing the arm position in the down-point case, and that gravity alone was responsible for destablizing the arm in the up-pointing case.

This week, we will be investigating the impact of *proportional feedback* on a first-order system's response to variability and disturbances. We will examine how to model disturbances for systems of interest using first-order LDEs with inputs, and in parallel, we will learn how to characterize the model's behavior using natural frequencies *and* by examining immediate and asymptotic responses to canonical inputs (unit steps and unit samples, see below). In lab, we will need to combine these concepts (and the ones we learned last week) to design an effective feedback/feed-forward approach to controlling propellor speed by adjusting motor current.

Next week, we will learn how to use natural frequencies to design second-order systems, and show how our understanding of higher-order LDEs can aid in the in the design of *proportional plus delta feedback*, (if you have seen the term PID control, delta is the D). In parallel, we will learn how to design the controller by examining the associated natural frequencies. And in lab, you will design a high-performance position controller for the copter-levitated arm.

Feedback and Feed-Forward Control in First Order Systems

What is Feedback Control

Thermostats, cruise control, camera autofocusers, scooter stabilizers, aircraft autopilots, audio amplifiers, maglev; all examples that demonstrate the pervasive role of feedback control in engineering design. But not all controllers use feedback. For example, when you stand on one foot, you use feedback control to stay balanced, by shifting your weight if you start to fall. But when you kick a ball towards a goal with that same foot, you are using predictive, or *feed-forward*, control. The ball arrives at the goal because you determined the right way to kick it, not because you corrected its trajectory in midflight (unless the ball is outfitted with your quadcopter, or the game is quidditch).

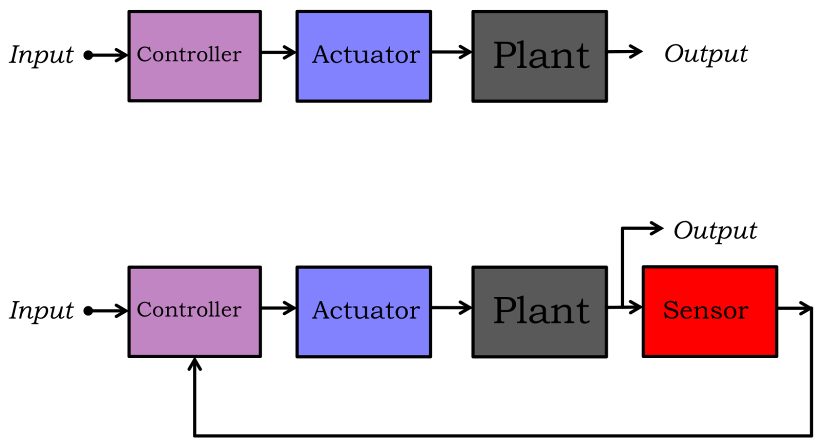

Figure 1: Two Control System Block Diagrams, one with Feedback (bottom) and one Feed-forward (top).

Feedback control has costs and limitations; we will confront some of those in lab this week. In fact, most modern controllers do not rely entirely on feedback, but use combinations of feed-forward and feedback control. But in order to describe those trade-offs, we first need a canonical description of the differences between feedback and feed-foward control. To this end, consider the feedback control (bottom) and feed-forward control (top) block diagrams in 1.

The *plant* block in both diagrams refers to what is being controlled, for example: room temperature, vehicle cruising speed, or robot position. The *actuator* refers to the physical instrument that modifies the plant or its environment in response to instructions from the controller. Actuator examples include the furnace in a temperature control system, the car engine in a cruise control system, or the motor-and-gears in an autofocusing system. The *controller* instructs the actuator, either based on a predetermined recipe of actuator-plus-plant model (feed-forward) or based on measurements of the plant state (feedback).

The *sense* block in the feedback control diagram means that the controller continually receives plant state updates, so if the state is perturbed, the controller can easily adapt its instructions to make corrections. When you use feedback control to balance on one foot, you continually monitor your "state" (probably your tilt angle), and weight-shift in the opposite direction of your tilt. In feed-forward control, there is no monitoring, the the controller sends instructions to the actuator based on a recipe or a model of how the actuator and plant will respond.

Kicking the ball towards a goal is not exactly analogous to feed-forward control the way engineers have classically thought about the problem, though the field is changing rapidly. When kicking a ball towards a goal, your kick is based on your mental model of how to get the ball to heads towards the goal, you can not impact the ball in midflight. But unlike our feed-forward block diagram, you do have the a sensor (you can see the ball), you just lack the in-flight actuator to correct the ball's trajectory.

Now suppose you kick the ball many times, and watch where it goes, and eventually learn how to kick the ball in to the goal. You are observing experiments and using those observations to improve your model, so you are using feedback, but is it feedback control? Not traditionally, but approaches from areas like machine learning are changing our perspectives.

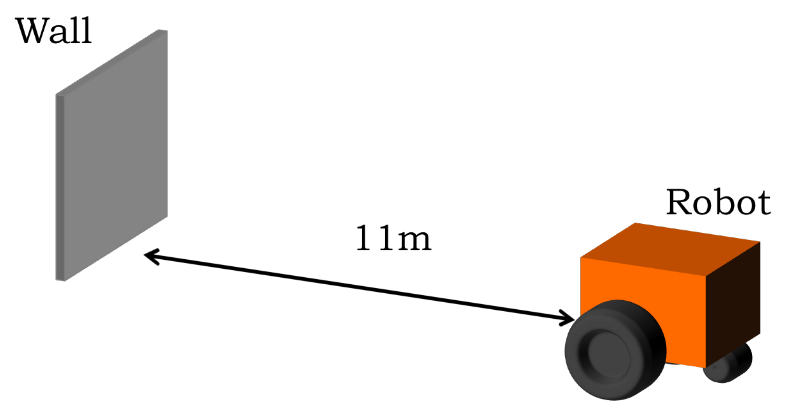

Figure 2: A robot with a light sensor and a timer

Hitting the wall again

Suppose you have a simple robot that moves forward continuously when turned on, and remains stationary otherwise. The robot must perform only one task, to move from an initial position, eleven meters from a wall, to a final position, exactly one meter from the wall (see Figure 2). Consider two strategies for controlling the robot so it performs this task.

Strategy One: The robot has a timer, and you use it to determine T, the interval of time the robot must be turned on to travel ten meters. To have the robot performs its task, your controller (controller 1) first turns the robot on, and then turns it off when the timer reading exceeds T seconds.

Strategy Two: The wall has a lamp, and the robot has a light sensor. You use them to determine L, the sensor reading that corresponds to one meter from the wall. To have the robot perform its task, your controller (controller 2) first turns the robot on, and then turns it off when the light sensor reading exceeds L.

Sensitivity to Actuator Variations

We can best understand the differences between the two strategies by considering the impact of changes in the environment or the robot.

Example 1 *Suppose the robot's initial position is modified, to twenty-one meters from the wall. But there is no opportunity to run new tests and determine new values for T and L.*

Question *1. How would you change the timer-based controller? Do you expect that the robot's final position will still be one meter from the wall?*

For controller one, you could keep the robot on until the timer reading exceeds $2\mathsf{T}$*. But the robot's final position may be inaccurate, doubling the interval ignores robot start-up and slow-down, which do not double.*

Question *2. How would you change the light-sensor-based controller? Do you expect that the robot's final position will still be to one meter from the wall?*

For controller 2, you do not need to change anything. And it will still end up one meter from the wall.

Example 2 *Consider what will happen if on of the wheels on the robot breaks, and is replaced with larger wheels.*

Question *3. Will controller 1 still position the robot accurately? Will controller 2?*

For controller 1, the time calibration will be wrong. The robot will likely travel too far, and might even crash in to the wall. For controller 2, the robot will still end up one meter from the wall.

The timer-based controller and the light-sensor-based controller both rely on initial calibration, and both are making measurements while the robot is moving, but there is a key difference. The timer-based controller is measuring time, and those readings are independent of the robot's distance to the wall. So time measurements can not be used to correct for changes in robot initial position or changes in wheel size. The light-sensor-based controller's measurements are related to the robot's current position, regardless of how the robot got there. *Feeding back* these measurement allow the controller to correct for changes in robot initial position or wheel size.

For these two strategies, we see a key attribute of feedback systems, they reduce the impact of variations or disturbances. The light-sensor-based controller always gets us to the right final position, regardless of wheel size or starting position, as long as the sensor is calibrated correctly. We will investigate these ideas more formally below.

Sensitivity to Sensor Variations

There is one more aspect of feedback control that makes it more a tool for the engineer than the scientist.

Example 3 *Suppose the light on the on the wall is replaced by a brighter light.*

Question *4. Will controller 2 still position the robot accurately?*

No, the robot will end up too far away from the wall. The brighter light will cause the sensor's reading to exceed the L *threshold further away than one meter from the wall, and the robot will be turned off. The problem is that the calibration used to relate sensor reading to distance was based on light sensor readings from a dimmer lamp.*

For controller 1, we calibrated the actuator by measuring the time to travel ten meters. Of course the controller fails when we change the actuator or the distance to the wall. But the same is true of controller 2. We calibrated the light sensor by measuring the light level one meter from the lamp on wall. Of course controller 2 fails when we switch to a brighter wall lamp.

It is true that that by using feedback, we made the robot behavior insensitive to changes in the actuator or the enviroment, but we also made it much more sensitive to changes in the sensor. So why is feedback so important? Because **it is far easier and cheaper to make an accurate sensor than to make an accurate actuator, or to ensure a consistent environment.** The value of feedback that it enables this important engineering trade-off. But it is a trade-off, using feedback has costs and limitations, as we will learn over the next few weeks.

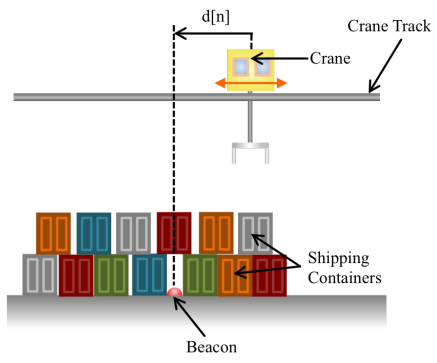

Figure 3: The Crane Redux.

First-Order LDE Models of Proportional Control

When we adjust velocity to control position, or adjust acceleration to control velocity, we say our system is *first order*. The former is equal to the rate of change of the latter. When we adjust acceleration to control position, we say our system is *second order*, because the former is related to the latter by two rates of change in sequence. Acceleration is the rate of change of velocity, and velocity is the rate of change in position. The approach to feedback control, and its ability to reduce the impact of variations and disturbances, is very different in the two cases. We will consider using *proportional feedback* on first-order systems this week, and examine second-order systems next week.

Example 4 *Our ill-fated heavy cargo crane returns, and is shown in Figure 3. It still travels on a track above a buried beacon, and measures its signed distance to the beacon every* δT *seconds. It has been redesigned to self-drive, and can reposition itself automatically at any location along the track. In addition, the crane now has an internal velocity controller, and we must design a new external position controller. The input to our position controller should be* r*, a sequence of desired crane locations, and* d*, the sequence of measured crane locations, and its output should be* v*, a sequence of crane velocities.*

To model how the crane moves along its track, we can relate the measured distance, d, to its velocity, v, as

$$d[n] = d[n-1] + \delta T v[n-1], \tag{1}$$

where $d[n]$ and $d[n-1]$ are the measured distances along the crane's track at sample time n and $n-1$, respectively, and $v[n-1]$ is the crane velocity at the time of sample $n-1$.

A *proportional feedback* approach to controlling the crane would be to make the crane velocity proportional to the difference between the desired and measured crane positions, as diagrammed in Figure 4. Then the velocity is given by

$$v[n] = K_p(r[n] - d[n]). \tag{2}$$

where K_p is the *proportional gain* and r is the desired position.

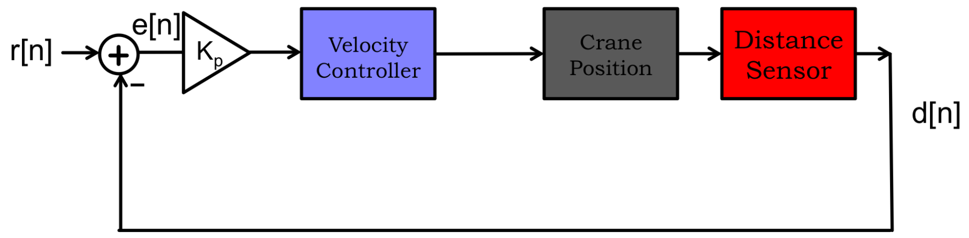

Figure 4: Crane Proportional Controller.

In order to determine the performance of this proportional-feedback approach as a function of gain, we substitute the control formula in to the distance-velocity relation for the crane,

$$d[n] = d[n-1] + \delta T K_p \left(r[n-1] - d[n-1] \right), \tag{3}$$

and then simplify,

$$d[n] = (1 - \delta T K_p) d[n-1] + \delta T K_p r[n-1], \tag{4}$$

to arrive at a first-order LDE with an input, r.

Examining Errors

Since our primary goal is to minimize position error, defined by

$$e[n] \equiv r[n] - d[n], \tag{5}$$

where $e[n]$ is the error at the n^{th} step, we will be able to better understand the relationship between proportional feedback gain and error by deriving a difference equation for the error alone.

Starting with the tautology $r[n] = r[n]$, and subtracting the difference equation for d above from both sides,

$$r[n] - d[n] = r[n] - \left(d[n-1] + \delta T K_p \left(r[n-1] - d[n-1] \right) \right), \tag{6}$$

and simplifying,

$$e[n] = (r[n] - d[n-1]) - \delta T K_p e[n-1], \tag{7}$$

results in an LDE for e, but it still depends on d, a quantity we do not yet know.

We can eliminate d by rewriting $r[n]$ as $r[n-1]$ plus a difference,

$$r[n] = (r[n] - r[n-1]) + r[n-1], \tag{8}$$

and then substitute this alternative representation for $r[n]$ in the equation for error,

$$e[n] = (r[n] - r[n-1]) \left(r[n-1] - d[n-1] \right) - \delta T K_p e[n-1], \tag{9}$$

then substitute $(r[n-1] - d[n-1]) = e[n-1]$,

$$e[n] = (r[n] - r[n-1]) + e[n-1] - \delta T K_p e[n-1], \tag{10}$$

and finally, reorganize,

$$e[n] = (1 - \delta T K_p) e[n-1] + (r[n] - r[n-1]). \tag{11}$$

We now have a first-order LDE for the position error, with desired position as an input. Well, not exactly desired position, but rather the differences in desired position from one sample to the next.

If we assume that the desired position changes only occasionally, and we are interested in how the error behaves *between* those changes, the error equation becomes homogenous, and we already learned how to solve those equations.

That is, between changes in desired position, $r[n] - r[n-1] = 0$, the error is given by

$$e[\tilde{n}] = (1 - \delta T K_p) e[\tilde{n} - 1] \tag{12}$$

where $n_0 < \tilde{n} < n_1$ where $\tilde{n}$ is used to remind us that we are only considering samples between n_0, the time sample of a particular change in r, and n_1, the time sample for the next change in r.

Right after a change in desired position,

$$e[n_0] = r[n_0] - d[n_0], \tag{13}$$

then the solution to the homogenous equation for e is given by

$$e[\tilde{n}] = (1 - \delta T K_p)^{\tilde{n} - n_0} e[n_0]. \tag{14}$$

And now we have insight in to how to select the gain, K_p! If we want the crane to move to the new desired position right after a change, then we want the position error to decrease. And that means we want

$$|(1 - \delta T K_p)| < 1 \tag{15}$$

or

$$0 < K_p < \frac{2}{\delta T}. \tag{16}$$

If we want the error to decrease fast, then we want to select K_p to make

$$|(1 - \delta T K_p)| \approx 0 \tag{17}$$

or

$$K_p \approx \frac{1}{\delta T}. \tag{18}$$

And one more point, if we want the error to decay monotonically (so the crane does not rock back and forth),

$$0 < (1 - \delta T K_p) < 1 \tag{19}$$

or

$$0 < K_p < \frac{1}{\delta T}. \tag{20}$$

Disturbances and Proportional Control

Our crane controller has a curious property. We just saw that if we choose a proportional gain that is positive and less than $\frac{2}{\delta T}$, the crane's position will eventually match the desired position, even if we make our gain very small. Though, in the case of very small proportional gain we will have to be patient and wait a long time before making another change in desired position. The main point

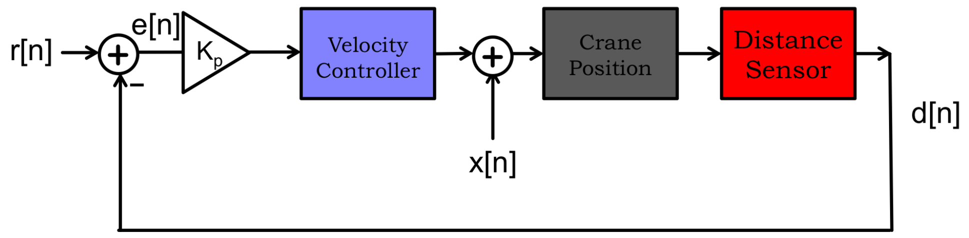

Figure 5: Crane Proportional Control With Disturbances.

is that if we only change the desired position once, then *higher gains do not get us better matches, they only get us faster matches.*

This exactness in the position match disappears once we consider disturbances. For the crane, the major disturbance is wind, and if we model wind as a velocity disturbance, the result will be the modified block diagram of Figure 5. In the figure, $x[n]$ is the velocity impact of the disturbing wind at the n^{th} step.

Including velocity disturbances in the relation between position and velocity

$$d[n] = d[n-1] + \delta T\left(v[n-1] + x[n-1]\right), \tag{21}$$

and then using the proportional feedback formula, $v[n] = K_p e[n]$,

$$d[n] = d[n-1] + \delta T\left(K_p e[n-1] + x[n-1]\right). \tag{22}$$

To derive the equation for the position error in this disturbing case, we again use the "subtract $r[n]$ from both sides" trick (and forgo a step or two), to get

$$e[n] = r[n] - r[n-1] + e[n-1] - \delta T\left(K_p e[n-1] + x[n-1]\right). \tag{23}$$

Reorganizing,

$$e[n] = (1 - \delta T K_p)e[n-1] - (r[n] - r[n-1]) - \delta T x[n-1]. \tag{24}$$

Again assuming that we are interested in the intervals between changes in desired position (and as a reminder that we are restricting ourselves to values of n between n_0 and n_1, we used $\tilde{n}$),

$$e[\tilde{n}] = (1 - \delta T K_p)e[\tilde{n}-1] - \delta T x[\tilde{n}-1]. \tag{25}$$

Note that we have a first-order LDE for e, as before, but now the LDE has an input, the disturbance due to wind, as represented by x.

It is clear that in a heavy wind, our proportional feedback controller will not position the crane in exactly the right position. But what is the relationship between proportional gain, the strength of the wind, and the inaccuracy in crane position. Will using high gain reduce the impact of high winds? For that, we will need to look at solving first-order difference equations with an input.

First-Order LDE's with unit-step and unit-sample inputs.

For any system we are trying to control, we will presumably try to influence its outputs by changing its inputs, so we will need to add an input to the homogenous LDE models we already know. In the first-order case, the general form of an LDE with an input is

$$y[n] = \alpha_1 y[n-1] + \beta_1 x[n-1] \ , \tag{26}$$

where α_1 and β_1 are real coefficients, the sequence x is a known input, and the sequence y is the output we need to determine.

For the systems we will be considering, most often we are interested in one of two types of inputs, a *unit sample* or a *unit step*.

Unit Sample Input

A *delayed unit sample* with delay n_0 is given by

$$x[n] = 0 \quad n \neq n_0 \quad x[n] = 1, \quad n = n_0, \tag{27}$$

for which we use the compact notation $x[n] = \delta[n - n_0]$.

As a formal matter, systems described by LDEs are completely characterized by the output due to a unit sample input, referred to as the *unit sample response*, but that is a digression for us. Formality aside, we can reason about systems by examining their response to a unit sample.

If $y[0] = 0$, and the input is a delayed unit sample, $x[n] = \delta[n-n_0]$, then $y[n]$ has a familiar power series form that can be derived in a few steps.

For $n \leq n_0$, $y[n] = 0$, and for $n = n_0$,

$$y[n_0 + 1] = \beta_1 x[n_0] = \beta_1 \delta[n_0 - n_0] = \beta_1 \tag{28}$$

For $n > n_0 + 1$, $x[n] = 0$ and therefore $y[n] = \alpha_1 y[n-1]$, the generating difference equation for the power series. Iterating yields

$$y[n] = (\alpha_1)^{n-n_0} \frac{\beta_1}{\alpha_1} \quad n > n_0. \tag{29}$$

a form very similar to the solution of the homogenous case with non-zero initial conditions.

In the stable case, that is if $|\alpha_1| < 1$,

$$\lim_{n \to \infty} y[n] = y[n_\infty] = 0. \tag{30}$$

Unit Step Input

A *delayed unit step* with delay n_0 is given by

$$x[n] = 0 \quad 0 \leq n < n_0 \quad x[n] = 1, \quad n \geq n_0, \tag{31}$$

for which we use the compact notation $x[n] = u[n - n_0]$

Given a unit step input, $x[n] = u[n - n_0]$, and $y[0]$ is still zero, then $y[n] = 0 \;\; n \leq n_0$ and $y[n_0 + 1]$ must equal $\beta_1 x[n_0] = \beta_1 u[n_0 - n_0] = \beta_1$. For $n > n_0 + 1$, $x[n] = 1$ and therefore

$$y[n] = \alpha_1 y[n-1] + \beta_1.$$

If we write out the terms, we can see the pattern,

$$\begin{aligned} y[n] &= 0, & n &= n_0 && (32)\\ y[n] &= \beta_1, & n &= n_0 + 1 && (33)\\ y[n] &= \alpha_1\beta_1 + \beta_1, & n &= n_0 + 2 && (34)\\ y[n] &= \alpha_1^2\beta_1 + \alpha_1\beta_1 + \beta_1, & n &= n_0 + 3. && (35) \end{aligned}$$

Iterating yields a representation of y as a power series sum,

$$y[n] = \sum_{m=n_0+1}^{n} (\alpha_1)^{m-n_0} \frac{\beta_1}{\alpha_1} \quad n > n_0. \tag{36}$$

In the stable case, that is $|\alpha_1| < 1$,

$$\lim_{n\to\infty} y[n] \equiv y[n_\infty] = \frac{\beta_1}{1 - \alpha_1} \tag{37}$$

which can be seen by summing the series.

Determining the large n limit

For the stable case, $|\alpha_1| < 1$, if the input eventually becomes constant, such as in the delayed unit step, we are often only interested in behavior for large n. If so, there is an easier approach to determining $y[n_\infty]$, a surprisingly useful trick more generally.

Suppose $x[n] = 1$ for $n > n_0$ and $|\alpha_1| < 1$, In this stable, constant input, case, $y[n]$ and $yn - 1]$ both approach $y[n_\infty]$ for large n, so

$$y[n_\infty] = \alpha_1 y[n_\infty] + \beta_1 \quad , \tag{38}$$

and solving for $y[n_\infty]$,

$$y[n_\infty] = \frac{\beta_1}{1 - \alpha}. \tag{39}$$

There are many ideas hidden in the above simple formula, and they are best exposed by answering questions.

Question 5. Given $\beta_1 > 0$ and $x[n] = u[n]$, if $\alpha_1 = -1$ does $y[n]$ remain bounded?

Question 6. For the $\alpha_1 = -1$ case, does $y[n_\infty]$ exist?

Question 7. Does $y[n]$ remain bounded if $\alpha = 1$?

Question 8. Given a $\beta_1 > 0$, and $x[n] = u[n]$, what value of α_1 minimizes $y[n_\infty]$ when it exists, and what is that minimum value.

The answers, in order of question

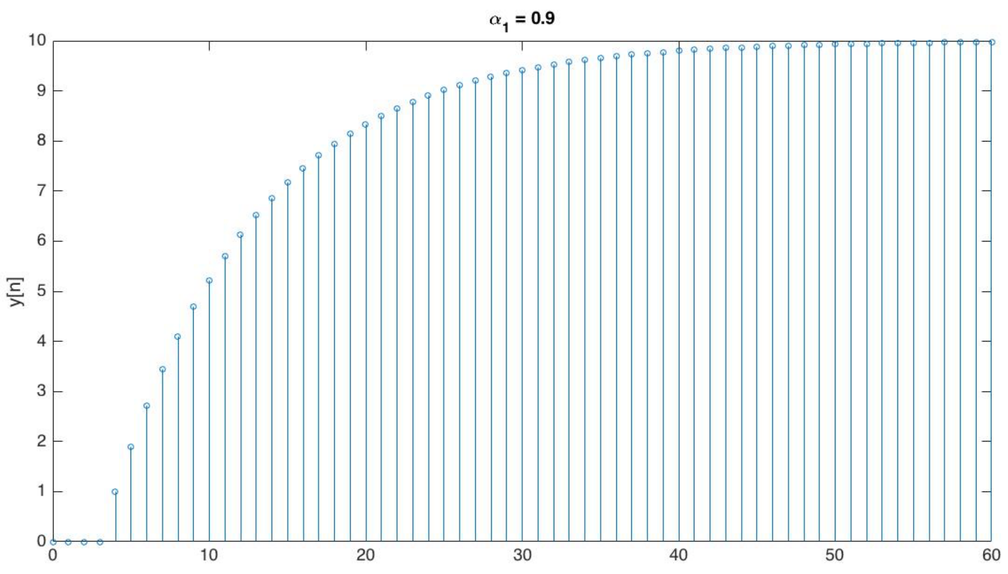

Figure 6: Delayed step response for $\alpha_1 = 0.9$ and $n_0 = 3$.

- Yes, $y[n]$ is bounded.
- No, $y[n_\infty]$ does not exist as $y[n]$ alternates between 0 and β_1 forever.
- If $\alpha_1 = 1$, $y[n]$ is unbounded and $y[n_\infty$ does not exist.
- Setting α_1 very close to -1 minimizes $y[n_\infty]$, which is bounded from below by $\frac{\beta_1}{2}$. But then $y[n]$ oscillates and decays very slowly.

And one last use of $y[n_\infty]$, we use it to construct the entire step response. If $y[0] = 0$, $x[n] = u[n - n_0]$, and $|\alpha_1| < 1$, then $y[n] = 0$ for $n \leq n_0$ and

$$y[n] = \left(1 - \alpha_1^{n-n_0}\right) y[n_\infty] \quad n > n_0. \tag{40}$$

In Figures ,,**??**, we show the behavior of the step response for several different values for α_1. Notice the large n values in the plots. For example, even if $\alpha_1 < 0$, the large n value is still positive.

A summary of key points

There are two general ideas in these notes:

- Proportional feedback controllers can reduce the impact of variations.
- A common first-order application of proportional feedback is controlling position by setting the velocity proportional to position error.

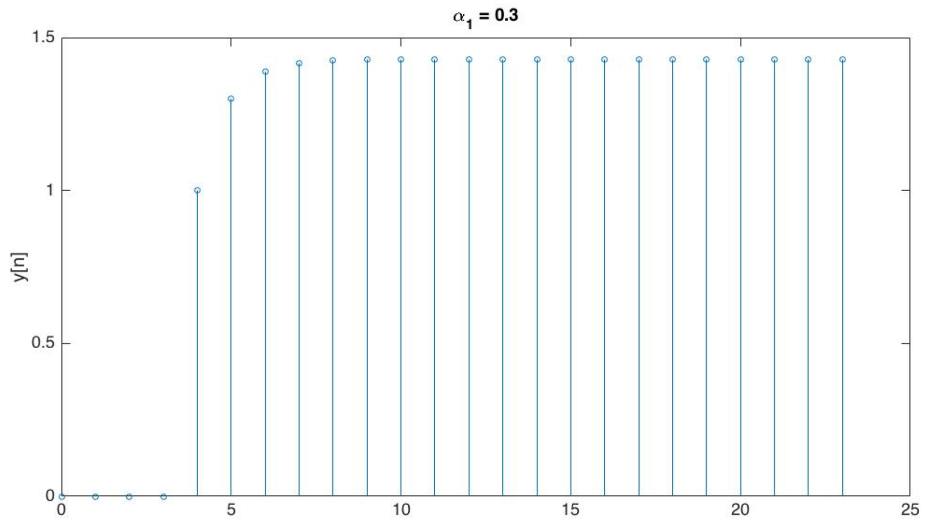

Figure 7: Delayed step response for $\alpha_1 = 0.3$ and $n_0 = 3$.

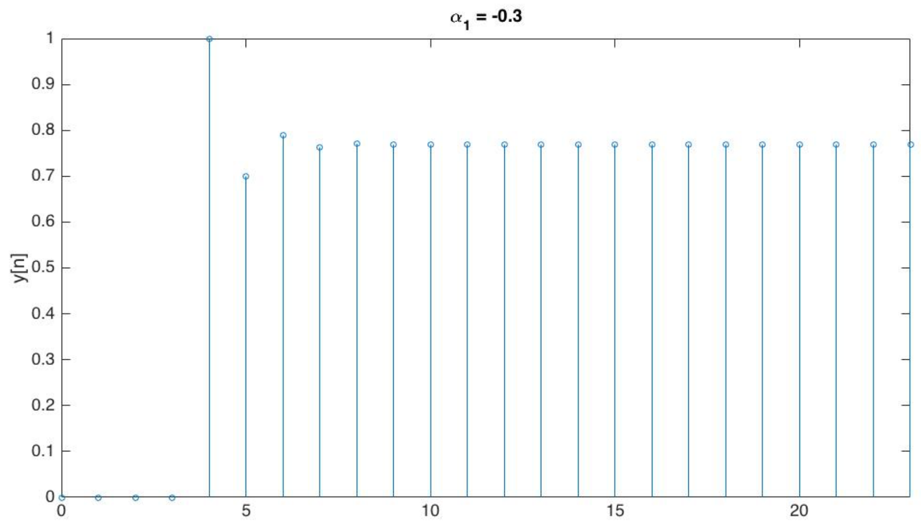

Figure 8: Delayed step response for $\alpha_1 = -0.3$ and $n_0 = 3$.

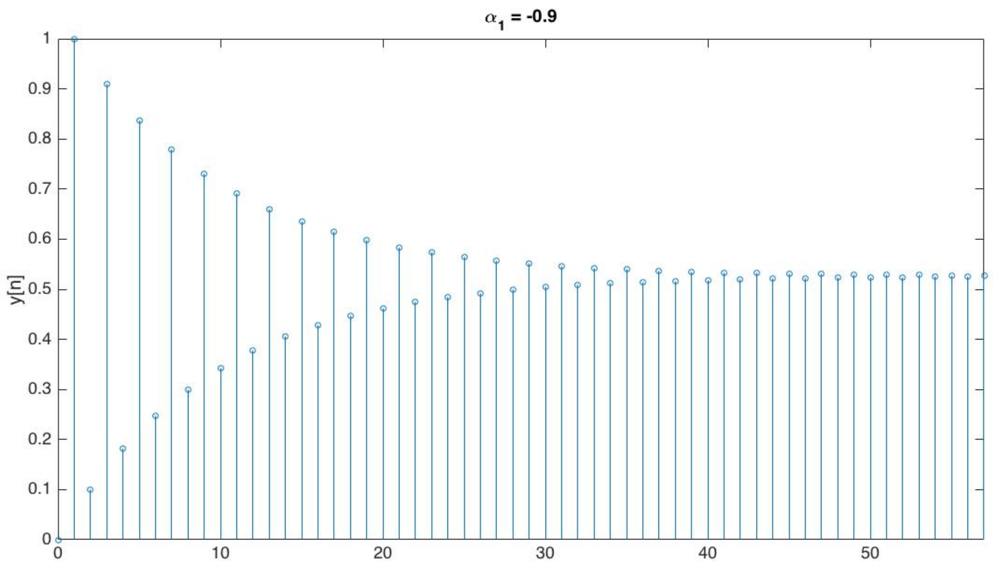

Figure 9: Unit step (no delay) response for $\alpha_1 = -0.9$ and $n_0 = 0$.

And for the application of controlling position by setting the velocity proportional to position error with gain K_p, the key points are:

- To ensure stability, $0 < K_p < \frac{2}{\delta T}$.
- To ensure monotonic error reduction, $0 < K_p < \frac{1}{\delta T}$.
- For fastest error reduction, $K_p \approx \frac{1}{\delta T}$.
- If there are no disturbances, and the desired position is fixed, then the position error decays to zero (unless there are disturbances).
- In the case of a constant disturbance, there is a particular simple formula for the eventual error.

Did you miss that last item in the notes? No, you did not. The formula was not given. We put it here, as it is the most important point of the notes. Recall that the error with disturbance (assuming no change in desired position) is

$$e[n] = (1 - \delta T K_p)e[n-1] - \delta T x[n-1]. \tag{41}$$

where for constant disturbance, $x[n] = x[0]$ for all n. If $0 < K_p < \frac{2}{\delta T}$ (stable case), then using the trick at the end of the previous section.

$$e[n_\infty] = \frac{-1}{1 - (1 - \delta T K_p)} \delta T x[0], \tag{42}$$

or

$$e[n_\infty] = -\frac{x[0]}{K_p}. \tag{43}$$

Higher gains increase disturbance rejection, but stability considerations limit the gain.

The most important point of these notes is: **Higher gains improve disturbance rejection, and faster sample rates allow higher gains.**