February 13, 2023

6.3100 Lecture 3 Notes – Spring 2023

Linearity, time invariance, first order DT system with loss, steady-state error

Dennis Freeman and Kevin Chen

Outline:

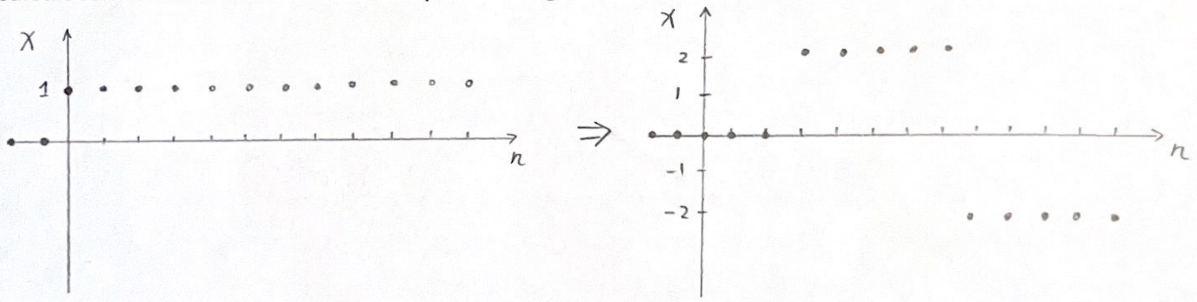
1. First order DT system with arbitrary driving x[n]: linearity, time invariance, and superposition
2. Feedforward control
3. Example: 3D-printing with heat dissipation
4. Steady-state error

1. <u>First order DT system with an arbitrary driving function x[n]: linearity, time invariance, and superposition</u>

In the previous lecture, we introduced first order DT equation:

$$y[n] = \lambda y[n-1] + bx[n-1]$$

We assumed that the input reference function x[n] is 1 for all n≥0, and it is 0 for n<0. This may seem a restrictive condition. In real life, the driving function is arbitrary. For instance, consider we know how to solve the system with the simple driving function on the left, how can we calculate the solution to a more complex driving function on the right?



It turns out we can invoke 2 properties of linear first order difference equations: linearity and time invariance. We state these 2 conditions without proof, but they are intuitive and are easy to prove.

Linearity:

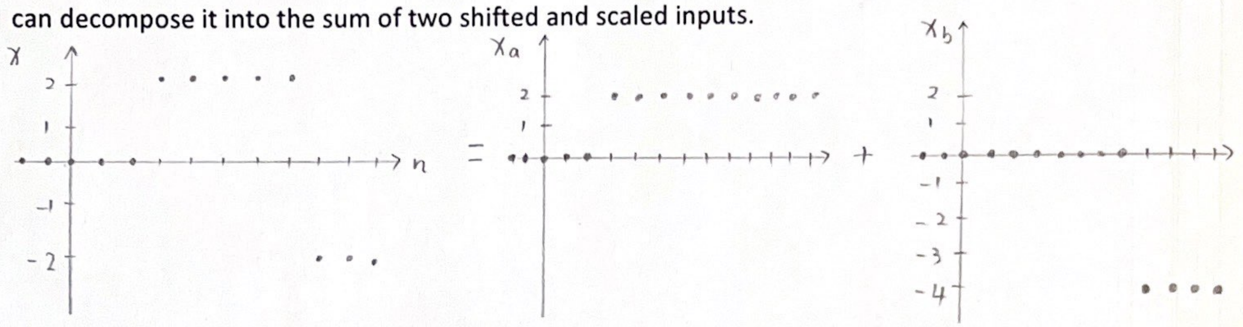
If the reference input $x_a[n]$ leads to the solution $y_a[n]$, and the input $x_b[n]$ leads to the solution $y_b[n]$, then we have:

$$Ax_a[n] + Bx_b[n] \rightarrow Ay_a[n] + By_b[n]$$

where A and B are constants, and the symbol "→" means "leads to". This property can be proved by direct substitution.

Time invariance:
If x[n] → y[n], then x[n-n₀] → y[n-n₀]. This property can also be proved by direct substitution.

These two conditions lead to the superposition principle. Consider the driving example again. We can decompose it into the sum of two shifted and scaled inputs.



For even more complex inputs, we can treat them as an infinite sum of scaled inputs, and use a tool called convolution. It is conceptually simple but requires some more mathematics. We will not cover convolution in this course.

2. Feedforward control

In preparation for lab 1, we are going to quickly introduce a feedforward control example. In this case, we are not going to use any sensor information for feedback. Consider the 3D-printing example, the plant equation is:

$$\frac{T_m[n] - T_m[n-1]}{\Delta T} = \gamma u[n-1]$$

The feedforward controller is given by:

$$u[n] = K_{ff} T_d[n]$$

where Kff is a parameter that we choose. Making a substitution, the system equation becomes:

$$\frac{T_m[n] - T_m[n-1]}{\Delta T} = \gamma K_{ff} T_d[n-1]$$

Rearranging the equation, we have:

$$T_m[n] = T_m[n-1] + \Delta T \gamma K_{ff} T_d[n-1]$$

The system natural frequency is 1, which means that it is unstable! The takeaway is that without any feedback control, the control system is probably not going to perform very well. This concept is useful, and we will return to this at the end of this lecture.

3. Example: 3D-printing with heat dissipation

We are going to make the 3D-printing example a little bit more realistic, and then study how this modeling change will cause a non-zero steady-state error.

The old plant equation is given by:

$$T_m[n] = T_m[n-1] + \Delta T \gamma u[n-1]$$

The change of temperature is only dependent on the input. Realistically, air convection can also cool the printing head. So we can add another term in the equation:

$$T_m[n] = T_m[n-1] + \Delta T \gamma u[n-1] - \Delta T \beta T_m[n-1]$$

Here the parameter β is a constant relating heat loss to the instantaneous temperature $T_m[n]$.

With this system, we can implement the same proportional feedback controller again:

$$u[n] = K_p(T_d[n] - T_m[n])$$

After substitution, the system equation becomes:

$$T_m[n] = \left(1 - \gamma K_p \Delta T - \Delta T \beta\right) T_m[n-1] + \gamma \Delta T K_p T_d[n-1]$$

Note that compared to the last lecture, we now have a new term $-\Delta T \beta T_m[n-1]$ that describes convective cooling. This new term changes our selection of Kp because it influences both stability, convergence rate, and steady-state error.

Stability:

$$-1 < \lambda < 1$$
$$-1 < 1 - \gamma K_p \Delta T - \Delta T \beta < 1$$
$$\frac{2 - \beta \Delta T}{\gamma \Delta T} > K_p > \frac{-\beta}{\gamma}$$

The value of Kp satisfies these limits to guarantee stability.

Convergence:
If we want our system to converge to a steady state value ($T_m[n] = T_m[n-1]$) as quickly as possible, then we need to set the natural frequency to 0. Here, we have:

$$\lambda = \left(1 - \gamma K_p \Delta T - \Delta T \beta\right) = 0$$

Solving for K_p, we obtain

$$K_p = \frac{1 - \Delta T \beta}{\gamma \Delta T}$$

This analysis gives an optimal Kp if we hope the system can converge quickly.

Steady-state error:
However, having a fast convergence rate may not be our only objective. In this example, let's calculate the steady state error. Let's define the error term as:

$$e[n] = T_d[n] - T_m[n]$$

We can rearrange the system equation as:

$$T_m[n] = (1 - \gamma K_p \Delta T - \Delta T \beta) T_m[n-1] + \gamma \Delta T K_p T_d[n-1]$$

$$-e[n] = -(1 - \gamma K_p \Delta T - \Delta T \beta) e[n-1] - \Delta T \beta T_d[n-1]$$

As n approach infinity, the equation becomes:

$$e[\infty] = \lambda e[\infty] + \Delta T \beta T_d[\infty]$$

Solving for the error term, we have:

$$e[\infty] = \frac{\Delta T \beta T_d[\infty]}{1 - \lambda}$$

Note that $e[\infty]$ is the steady-state error. This is an interesting result. First, λ needs to be between -1 and 1 to guarantee stability. So as long as $\beta \neq 0$, our control system will have a steady state error! If having a smaller error is more important to us, then we need to let $\lambda = -1$. This is an important takeaway. In many situations, there is no solution that optimizes every aspect of the control system. We need to consider tradeoffs. Are we prioritizing faster convergence or smaller steady state error? This is a design choice.

4. <u>Steady-state error</u>

Building on the previous example, can I design a controller that removes the steady-state error? This requires us to design a new controller. For instance, we can put together a feedforward-and-proportional controller.

We can set the controller as:

$$u[n] = K_{ff} T_d[n] + K_p (T_d[n] - T_m[n])$$

Now we have 2 numbers to choose: Kff and Kp. The system equation becomes:

$$T_m[n] = (1 - \gamma K_p \Delta T - \Delta T \beta) T_m[n-1] + \gamma \Delta T (K_p + K_{ff}) T_d[n-1]$$

$$e[n] = (1 - \gamma K_p \Delta T - \Delta T \beta) e[n-1] + (-\gamma K_{ff} + \beta) \Delta T T_d[n-1]$$

$$e[n] = \lambda e[n-1] + (-\gamma K_{ff} + \beta) \Delta T T_d[n-1]$$

Now the steady-state error becomes:

$$e[\infty] = \lambda e[\infty] + (-\gamma K_{ff} + \beta) \Delta T T_d[\infty]$$

$$e[\infty] = \frac{(-\gamma K_{ff} + \beta) \Delta T T_d[\infty]}{1 - \lambda}$$

Can we make the steady-state error $e[\infty]$ equal to 0? Yes! We can set

$$K_{ff} = \beta / \gamma$$

In this new controller design, we can choose K_p to optimize the convergence rate, and then choose K_{ff} to eliminate the steady-state error. In the next few weeks, we are going to introduce more complex controllers such as proportional-derivative (PD) and proportional-integral-derivative (PID) controllers.