Dynamic System Modeling and Control Design

Experimental Characterization of First Order Systems

September 16, 2024

Outline

Recap: First Order Systems

In the previous lectures, we discussed how to solve first order systems:

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

- We saw how the natural frequency, λ, determines the stability, steady-state, and convergence of our system.
- We analyzed the zero state response and saw how linearity and time-invariance allow us to study arbitrary driving signals.

Recap: First Order with Loss

We made progress towards designing a "realistic" system which includes some loss:

$$y[n] = (1 + \Delta T \beta) y[n-1] + \gamma \Delta T u[n-1].$$

- $y[n]$ is the output of our system, e.g., the measured temperature,
- $u[n]$ is the control signal we design,
- ΔT relates to the sampling rate of the microcontroller,
- The parameters β and γ are system properties that we want to measure.

Method for Measuring β, γ

We can design a reasonable control signal $u[n]$ with the goal of measuring β, γ.

- In particular, with this goal we don't care about stability, following some trajectory, etc.

$$y[n] = (1 + \Delta T \beta) y[n-1] + \gamma u[n-1].$$

Let's try two choices for $u[n]$:

- Feedback: $u[n] = K_p(x[n] - y[n])$,
- Feedforward: $u[n] = K_{ff} x[n]$.

Feedback Control, β, and γ

Given the feedback controller $u[n] = K_p(x[n] - y[n])$, the system equation becomes:

$$
\begin{aligned}
y[n] &= (1 + \Delta T\beta)y[n-1] + \gamma\Delta T u[n-1], \\
y[n] &= (1 + \Delta T\beta)y[n-1] + \gamma\Delta T K_p(x[n-1] - y[n-1]), \\
y[n] &= \underbrace{(1 + \Delta T\beta - \gamma\Delta T K_p)}_{\lambda} y[n-1] + \gamma\Delta T K_p x[n-1].
\end{aligned}
$$

A bit problematic; the natural frequency changes as we change K_p.

Feedforward Control, β, and γ

Given the feedforward controller $u[n] = K_{ff}x[n]$, the system equation becomes:

$$y[n] = (1 + \Delta T\beta)y[n-1] + \gamma\Delta T u[n-1],$$
$$y[n] = \underbrace{(1 + \Delta T\beta)}_{\lambda} y[n-1] + \gamma\Delta T K_{ff}x[n-1].$$

Much better! Now, we can estimate λ to back-calculate β.

Deriving β from Computing λ

We need find two relationships between β and γ. One common approach is to look at the step response of the system.

- In particular, we can analyze the output of our system when the input function $x[n] = 1$ for all $n \geq 0$ and 0 otherwise.

We can calculate λ by measuring the time–denoted n^*–required for $y[n]$ to reach half of its steady-state $y[\infty]$,

$$\lambda^{n^*} = 0.5 \Rightarrow n^* \log_e \lambda = \log_e 0.5 \Rightarrow \lambda = \exp\left(\frac{1}{n^*} \log_e 0.5\right).$$

Using the Steady-State Equation

Then, we can back-calculate for β:

$$\beta = \frac{\exp\left(\frac{1}{n^*}\log_e 0.5\right) - 1}{\Delta T}.$$

Next, we need to solve for γ. Let's look at the steady state condition:

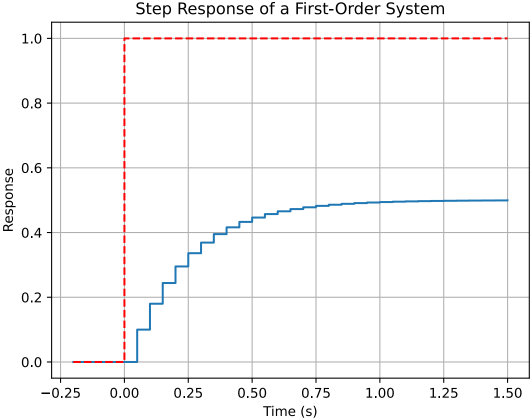
$$y[\infty] \approx y[\infty](1 + \Delta T\beta) + \gamma \Delta T K_{ff}.$$

Solving for γ, we find:

$$\gamma = -\frac{y[\infty]\beta}{K_{ff}} = -\frac{y[\infty]\left(\exp\left(\frac{1}{n^*}\log_e 0.5\right) - 1\right)}{K_{ff}\Delta T}.$$

Check Yourself: Deriving β, γ for Unknown System

Consider the following plot of the step-response of a first-order system:

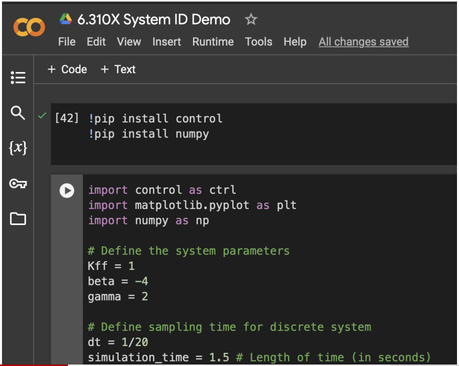


Here, a feedforward controller is used with $K_{ff} = 1$. Use the method described to measure the system parameters γ and β.

Numerical Tools for Analyzing Control Systems

First order systems are simple enough to solve manually. However, the algebra becomes increasingly tedious for higher order systems.

Python has a `control` library which is useful for modeling systems. A Google Colab notebook for the following code is available (here).



A (Condensed) Look at the Code..

```
# Define the system parameters
Kff = 1
beta = -4
gamma = 2
dt = 1/20

# Define the transfer function
num = np.array([0, dt*gamma*Kff])
den = np.array([1, -(1+dt*beta)])

# Define our first-order system
system = ctrl.TransferFunction(num,den,dt=dt)

# Get step response
time = np.arange(0, 1.5, dt)  # Create the time vector
_, response = ctrl.step_response(system, T=time)
```

Obtaining the Transfer Function

In the coming weeks, we'll explain the transfer function. For now, let's see how we obtained it from "pattern matching".

Rearranging our system function, we obtain:

$$y[n] - (1 + \Delta T\beta)y[n-1] = \gamma\Delta T K_{ff} x[n-1].$$

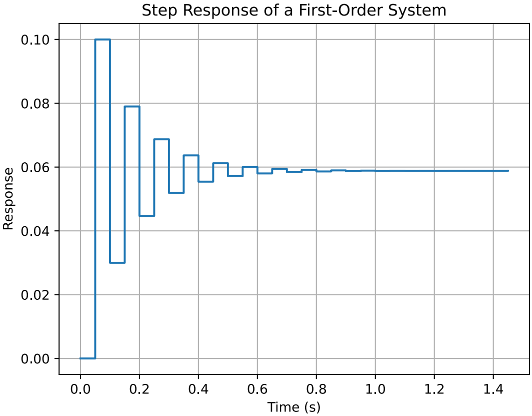
The denominator contains the coefficients in front of $y[n]$ and $y[n-1]$; the numerator contains the coefficients in front of $x[n]$ and $x[n-1]$

Modifying Code for Feedback Controller

```
# Define the system parameters
Kff = 1
Kp = 15
beta = -4
gamma = 2
dt = 1/20
# Define the transfer function
num = np.array([0, dt*gamma*Kff])
den = np.array([1, -(1+dt*beta) + dt*gamma*Kp])
# Define our first-order system
system = ctrl.TransferFunction(num,den,dt=dt)
# Set up timing variables
time = np.arange(0, 1.5, dt)  # Create the time vector
# Get step response
_, response = ctrl.step_response(system, T=time)
```

Simulating a First Order System with Feedback

Running the code, we obtain the following step response:



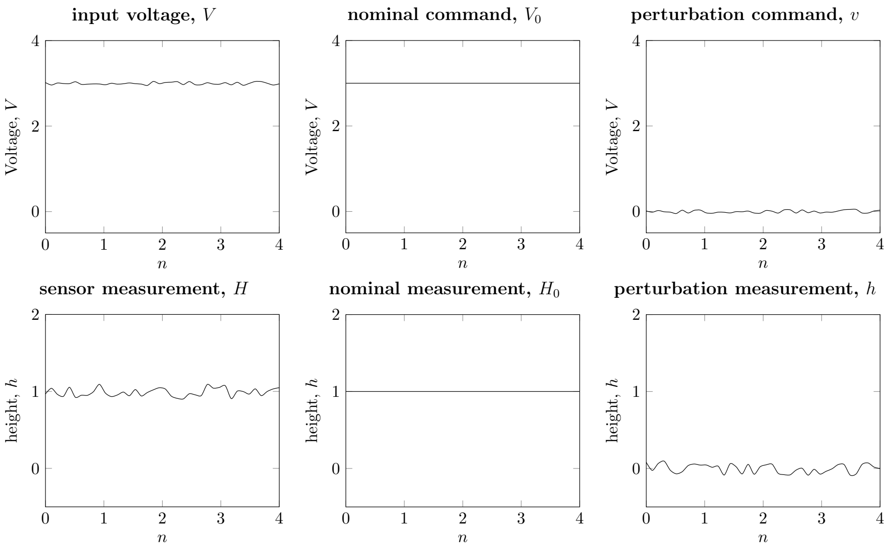
Nominal and Perturbation Control Signals

Often time we control a system relative to an equilibrium state. Consider modeling a quadrotor that is hovering at a set point.

There is a large nominal input command (some driving voltage) and a large nominal altitude set point (height).

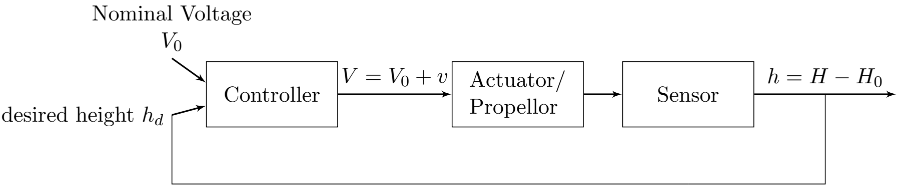
- These nominal commands do not require feedback control.
- We are interested in the “perturbation” control signals and sensor output.

Visualizing Nominal vs. Perturbation Signals



Block Diagrams with Nominal and Perturbation Quantities

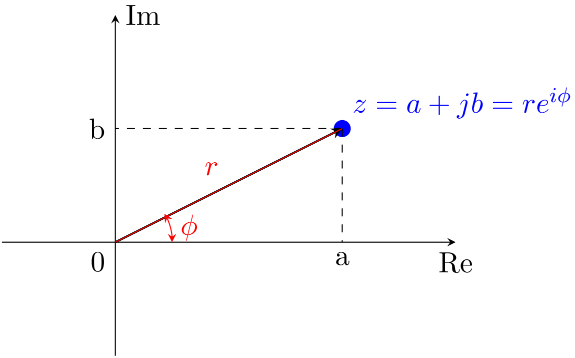
When we draw block diagrams, we need to specify which are the nominal quantities and which are the perturbation quantities we control:



Here, V is the perturbed voltage (nominal + perturbation); h is the perturbed height.

Complex Number Definitions

Complex numbers are critical when analyzing higher order systems. We will use j to denote the imaginary number, $j = \sqrt{-1}$.



$$r = \sqrt{a^2 + b^2}, \phi = \tan^{-1}\left(b/a\right), a = r\cos\phi, b = r\sin\phi.$$

Complex Numbers and Analyzing Stability

Two important relations for j:

$$j^2 = -1, \quad \frac{1}{j} = \frac{j}{j^2} = -j$$

We can use the polar form to evaluate the stability of our system:

$$\lambda^n = (re^{j\phi})^n = r^n e^{jn\phi}.$$

The phase $e^{jn\phi}$ has an amplitude of 1, and the r^n term determines whether the system is stable. Importantly, the amplitude of natural frequencies must always be less than 1.

Natural Frequencies on the Unit Circle