Following reference input matrix

The feedback control law above makes the open-loop system behave like $\mathbf{A}_{ref}$, but the input dynamics are still that of the original system. Here's how to make the input dynamics behave like $\mathbf{B}_{ref}$. We want to find the $\mathbf{u}_{imf,k}$ that makes the real model follow the reference model.

$$\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_{imf,k}\\ \mathbf{z}_{k+1} &= \mathbf{A}_{ref}\mathbf{z}_k + \mathbf{B}_{ref}\mathbf{u}_k\end{aligned}$$

Let $\mathbf{x} = \mathbf{z}$.

$$\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{z}_{k+1}\\ \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_{imf,k} &= \mathbf{A}_{ref}\mathbf{x}_k + \mathbf{B}_{ref}\mathbf{u}_k\\ \mathbf{B}\mathbf{u}_{imf,k} &= \mathbf{A}_{ref}\mathbf{x}_k - \mathbf{A}\mathbf{x}_k + \mathbf{B}_{ref}\mathbf{u}_k\\ \mathbf{B}\mathbf{u}_{imf,k} &= (\mathbf{A}_{ref} - \mathbf{A})\mathbf{x}_k + \mathbf{B}_{ref}\mathbf{u}_k\\ \mathbf{u}_{imf,k} &= \mathbf{B}^{\dagger}((\mathbf{A}_{ref} - \mathbf{A})\mathbf{x}_k + \mathbf{B}_{ref}\mathbf{u}_k)\\ \mathbf{u}_{imf,k} &= -\mathbf{B}^{\dagger}(\mathbf{A} - \mathbf{A}_{ref})\mathbf{x}_k + \mathbf{B}^{\dagger}\mathbf{B}_{ref}\mathbf{u}_k\end{aligned}$$

The first term makes the open-loop poles match that of the reference model, and the second term makes the input behave like that of the reference model.

C.4 Time delay compensation

Linear-Quadratic regulator controller gains tend to be aggressive. If sensor measurements are time-delayed too long, the LQR may be unstable (see figure C.1). However, if we know the amount of delay, we can compute the control based on where the system will be after the time delay.

We can compensate for the time delay if we know the control law we're applying in future timesteps ($\mathbf{u} = -\mathbf{K}\mathbf{x}$) and the duration of the time delay. To get the true state at the current time for control purposes, we project our delayed state forward by the time delay using our model and the aforementioned control law. Figure C.2 shows improved control with the predicted state.[2]

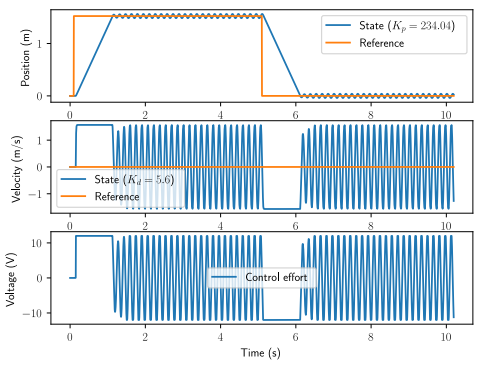

Figure C.1: Elevator response at 5 ms sample period with 50 ms of output lag

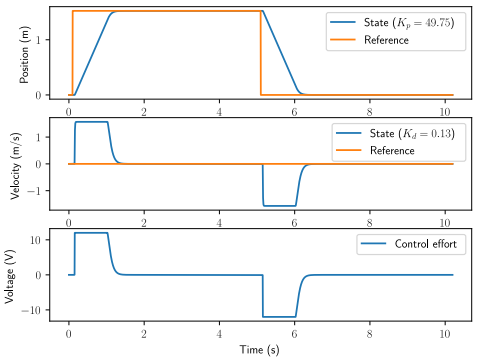

Figure C.2: Elevator response at 5 ms sample period with 50 ms of output lag (delay-compensated)

[2] Input delay and output delay have the same effect on the system, so the time delay can be simulated with either an input delay buffer or a measurement delay buffer.

For steady-state controller gains, this method of delay compensation seems to work better for second-order systems than first-order systems. Figures C.3 and C.5 show time delay for a drivetrain velocity system and flywheel system respectively. Figures C.4 and C.6 show that compensating the controller gain significantly reduces the feedback gain. For systems with fast dynamics and a long time delay, the delay-compensated controller has an almost open-loop response because only feedforward has a significant effect; this has poor disturbance rejection. Fixing the source of the time delay is always preferred for these systems.

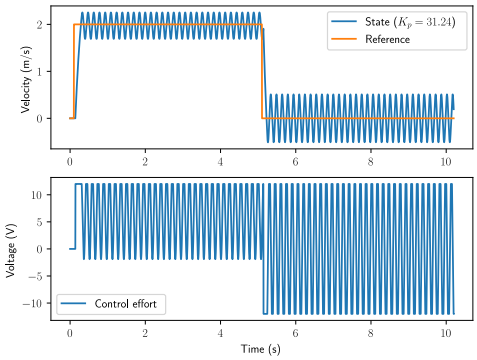

Figure C.3: Drivetrain velocity response at 1 ms sample period with 40 ms of output lag

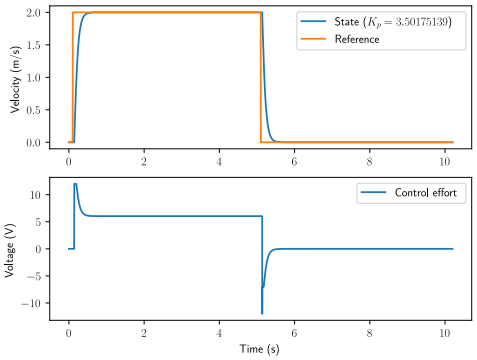

Figure C.4: Drivetrain velocity response at 1 ms sample period with 40 ms of output lag (delay-compensated)

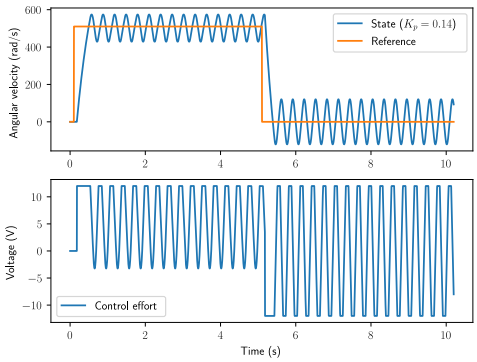

Figure C.5: Flywheel response at 1 ms sample period with 100 ms of output lag

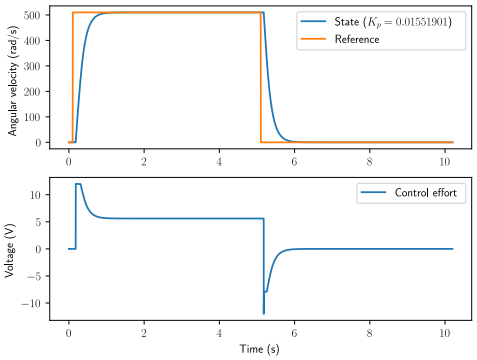

Figure C.6: Flywheel response at 1 ms sample period with 100 ms of output lag (delay-compensated)

Since we are computing the control based on future states and the state exponentially converges to zero over time, the control action we apply at the current timestep also converges to zero for longer time delays. During startup, the inputs we use to predict the future state are zero because there's initially no input history. This means the initial inputs are larger to give the system a kick in the right direction. As the input delay buffer fills up, the controller gain converges to a smaller steady-state value. If one uses the steady-state controller gain during startup, the transient response may be slow.

All figures shown here use the steady-state control law (the second case in equation (C.15)).

We'll show how to derive this controller gain compensation for continuous and discrete systems.

C.4.1 Continuous case

The undelayed continuous linear system is defined as $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t)$ with the controller $\mathbf{u}(t) = -\mathbf{K}\mathbf{x}(t)$. Let L be the amount of time delay in seconds. We can avoid the time delay if we compute the control based on the plant L seconds in the future.

$$\mathbf{u}(t) = -\mathbf{K}\mathbf{x}(t + L)$$

We need to find $\mathbf{x}(t + L)$ given $\mathbf{x}(t)$. Since we know $\mathbf{u}(t) = -\mathbf{K}\mathbf{x}(t)$ will be applied over the time interval $[t, t + L)$, substitute it into the continuous model.

$$\begin{aligned}
\dot{\mathbf{x}} &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \\
\dot{\mathbf{x}} &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}(-\mathbf{K}\mathbf{x}(t)) \\
\dot{\mathbf{x}} &= \mathbf{A}\mathbf{x}(t) - \mathbf{B}\mathbf{K}\mathbf{x}(t) \\
\dot{\mathbf{x}} &= (\mathbf{A} - \mathbf{B}\mathbf{K})\mathbf{x}(t)
\end{aligned}$$

We now have a differential equation for the closed-loop system dynamics. Take the matrix exponential from the current time t to L in the future to obtain $\mathbf{x}(t + L)$.

$$\mathbf{x}(t + L) = e^{(\mathbf{A}-\mathbf{B}\mathbf{K})L}\mathbf{x}(t) \tag{C.10}$$

This works for $t \geq L$, but if $t < L$, we have no input history for the time interval $[t, L)$. If we assume the inputs for $[t, L)$ are zero, the state prediction for that interval is

$$\mathbf{x}(L) = e^{\mathbf{A}(L-t)}\mathbf{x}(t)$$

The time interval $[0, t)$ has nonzero inputs since it's in the past and was using the normal control law.

$$\begin{aligned}
\mathbf{x}(t + L) &= e^{(\mathbf{A}-\mathbf{B}\mathbf{K})t}\mathbf{x}(L) \\
\mathbf{x}(t + L) &= e^{(\mathbf{A}-\mathbf{B}\mathbf{K})t}e^{\mathbf{A}(L-t)}\mathbf{x}(t)
\end{aligned} \tag{C.11}$$

Therefore, equations (C.10) and (C.11) give the latency-compensated control law for all $t \geq 0$.

$$\begin{aligned}
\mathbf{u}(t) &= -\mathbf{K}\mathbf{x}(t + L) \\
\mathbf{u}(t) &= \begin{cases} -\mathbf{K}e^{(\mathbf{A}-\mathbf{B}\mathbf{K})t}e^{\mathbf{A}(L-t)}\mathbf{x}(t) & \text{if } 0 \leq t < L \\ -\mathbf{K}e^{(\mathbf{A}-\mathbf{B}\mathbf{K})L}\mathbf{x}(t) & \text{if } t \geq L \end{cases}
\end{aligned} \tag{C.12}$$

C.4.2 Discrete case

The undelayed discrete linear system is defined as $\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k$ with the controller $\mathbf{u}_k = -\mathbf{K}\mathbf{x}_k$. Let L be the amount of time delay in seconds. We can avoid the time delay if we compute the control based on the plant L seconds in the future.

$$\mathbf{u}_k = -\mathbf{K}\mathbf{x}_{k+L/T}$$

We need to find $\mathbf{x}_{k+L/T}$ given $\mathbf{x}_k$. Since we know $\mathbf{u}_k = -\mathbf{K}\mathbf{x}_k$ will be applied for the timesteps k through $k + \frac{L}{T}$, substitute it into the discrete model.

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k$$

$$\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{A}\mathbf{x}_k + \mathbf{B}(-\mathbf{K}\mathbf{x}_k)\\ \mathbf{x}_{k+1} &= \mathbf{A}\mathbf{x}_k - \mathbf{B}\mathbf{K}\mathbf{x}_k\\ \mathbf{x}_{k+1} &= (\mathbf{A} - \mathbf{B}\mathbf{K})\mathbf{x}_k\end{aligned}$$

Let T be the duration between timesteps in seconds and L be the amount of time delay in seconds. $\frac{L}{T}$ gives the number of timesteps represented by L.

$$\mathbf{x}_{k+L/T} = (\mathbf{A} - \mathbf{B}\mathbf{K})^{\frac{L}{T}}\mathbf{x}_k \tag{C.13}$$

This works for $kT \geq L$ where kT is the current time, but if $kT < L$, we have no input history for the time interval $[kT, L)$. If we assume the inputs for $[kT, L)$ are zero, the state prediction for that interval is

$$\begin{aligned}\mathbf{x}_{L/T} &= \mathbf{A}^{\frac{L-kT}{T}}\mathbf{x}_k\\ \mathbf{x}_{L/T} &= \mathbf{A}^{\frac{L}{T}-k}\mathbf{x}_k\end{aligned}$$

The time interval $[0, kT)$ has nonzero inputs since it's in the past and was using the normal control law.

$$\begin{aligned}\mathbf{x}_{k+L/T} &= (\mathbf{A} - \mathbf{B}\mathbf{K})^k\mathbf{x}_{L/T}\\ \mathbf{x}_{k+L/T} &= (\mathbf{A} - \mathbf{B}\mathbf{K})^k\mathbf{A}^{\frac{L}{T}-k}\mathbf{x}_k\end{aligned} \tag{C.14}$$

Therefore, equations (C.13) and (C.14) give the latency-compensated control law for all $t \geq 0$.

$$\begin{aligned}\mathbf{u}_k &= -\mathbf{K}\mathbf{x}_{k+L/T}\\ \mathbf{u}_k &= \begin{cases}-\mathbf{K}(\mathbf{A} - \mathbf{B}\mathbf{K})^k\mathbf{A}^{\frac{L}{T}-k}\mathbf{x}_k & \text{if } 0 \leq k < \frac{L}{T}\\ -\mathbf{K}(\mathbf{A} - \mathbf{B}\mathbf{K})^{\frac{L}{T}}\mathbf{x}_k & \text{if } k \geq \frac{L}{T}\end{cases}\end{aligned} \tag{C.15}$$

If the delay L isn't a multiple of the sample period T in equation (C.15), we have to evaluate fractional matrix powers, which can be tricky. Eigen (a C++ library) supports fractional powers with the `pow()` member function provided by `<unsupported/Eigen/MatrixFunctions>`. scipy (a Python library) supports fractional powers with the free function `scipy.linalg.fractional_matrix_power()`. If the language you're using doesn't provide such a function, you can try the following approach instead.

Let there be a matrix $\mathbf{M}$ raised to a fractional power n. If $\mathbf{M}$ is diagonalizable, we can obtain an exact answer for $\mathbf{M}^n$ by decomposing $\mathbf{M}$ into $\mathbf{P}\mathbf{D}\mathbf{P}^{-1}$ where $\mathbf{D}$ is a diagonal matrix, computing $\mathbf{D}^n$ as each diagonal element raised to n, then recomposing $\mathbf{M}^n$ as $\mathbf{P}\mathbf{D}^n\mathbf{P}^{-1}$.

If a matrix raised to a fractional power in equation (C.15) isn't diagonalizable, we have to approximate by rounding $\frac{L}{T}$ to the nearest integer. This approximation gets worse as $L \bmod T$ approaches $\frac{T}{2}$.