16-299 Assignment 3: Effects of unmodelled dynamics

Often a control system designer either accidentally or deliberately designs a control system on the basis of a model that is simpler than reality. We call the difference between the model and reality “unmodelled dynamics”.

For this assignment let’s pretend we think a system is a single mass ($m = 1$) that slides along a frictionless horizontal floor, and has a rocket thruster that can provide any desired force ($f = u$) at any time ($f = u(t)$). The dynamics of this system is given by $f = m * acceleration$.

A simple linear feedback controller with position and velocity gains:

$$u = -pos\_gain * (x - x_d(t)) - vel\_gain * \dot{x}. \quad (1)$$

is used to move the mass to a desired position (x_d) at any particular time ($x_d(t)$). x is the current position, and $\dot{x}$ is the current velocity. If the simple model of $f = m * acceleration$ were correct, the position and velocity gains could be as large as you like, and the controlled system could be as fast and stiff as you like.

In reality, the system is more complicated in ways you don’t know about or don’t want to model. I will describe several ways it can be more complicated. In this assignment you will pick one of the forms of unmodelled dynamics (A, B, or C) and explore its effect on how the system behaves, and how it might limit feedback gains of the simple linear feedback controller.

Side note: Another limiting factor for feedback control is when the actuator output saturates (a maximum force or torque magnitude) or a maximum value of a state variable (maximum velocity or joint range).

Pick one of these forms of unmodelled dynamics (A, B, or C)

A) Simple actuator dynamics: The rocket thruster generates a force f, but it takes time to change the force. So instead of $f = u$,

$$\dot{f} = (u - f)/t_c. \quad (2)$$

[Some terminology: This is a first-order linear time invariant system and t_c is known as the time constant. (`https://en.wikipedia.org/wiki/Time_constant`)]. When $f = u$, the rate of change of force with time is zero ($\dot{f} = 0$), but if u is different than f, the rate of change of force is non-zero changing the force towards u. For this assignment, use $t_c = 1$.

To simulate this system, we need to incorporate the force f into the state: $(x, \dot{x}, f)$. The action is now the rate of change of force $u = \dot{f}$. Simulation requires the derivatives of each component of the state. The derivative of the position x is the velocity $\dot{x}$ which is part of the state. The derivative of the velocity is the acceleration $\ddot{x} = f/m$ and thus depends on the force which is the third element of the state vector. The derivative of the force f depends on the action: $\dot{f} = u$. See the matlab code in the directory A-actuator-dynamics for how this can be simulated.

B) A structural mode: In this case the mass m can deform with one vibratory mode. We can model that mode with two masses connected by a spring. For this assignment use a spring constant k of 10 and a damping b of 3. In the figure below $m_1 = m_2 = m/2$. The feedback controller measures x_1 as x and the force u acts on m_2.

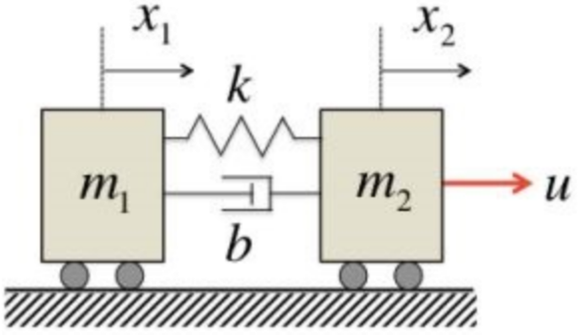

Figure 1: Unmodelled dynamics B

To simulate this system, the state vector becomes $(x_1, x_2, \dot{x}_1, \dot{x}_2)$. Simulation requires the derivatives of each component of the state. The derivatives of the positions x_j are the velocities $\dot{x}_j$ which are part of the state. The derivative of the velocity x_1 (an acceleration) is the spring force divided by m_1. Let's use the convention that when the spring is stretched its force is positive and given by $k * (x_2 - x_1)$. In this case a positive spring force positively accelerates m_1 to the right: $\ddot{x}_1 = k * (x_2 - x_1)/m_1$. The derivative of the velocity x_2 is the total force divided by m_2. The total force is the action u combined with the spring force, which pulls m_2 to the left when it is positive, so: $\ddot{x}_2 = (u - k * (x_2 - x_1))/m_2$. See the matlab code in the directory B-structural-mode for how this can be simulated.

C) Time delay: Humans have about a 100 millisecond (0.1 second) time delay in our control loop. We will model this as $f(t) = u(t - 0.1)$. This can be implemented in a simulator by having an array of N u values. At each "tick", the last (rightmost) value in the array is used to set the control, all the values in the array are shifted in time (right), and the current command u is put in the first (leftmost) position in the array. This is known as a "delay line". Simulate a time delay of 100ms. With a sampling period of 0.001s, you need an delay line array that is 100 samples long. See the matlab code in the directory C-delay, for how I implement a delay line.

What am I supposed to do?

1) Choose a position gain and a velocity gain such that the step response of the simple model (idealized system) is critically damped, and takes 300ms to get and stay within 0.95 of the way to the goal (if the system starts at zero, and moves to one, when does it reach 0.95 and stay within a 0.05 distance of the goal (between 0.95 and 1.05).

2) Pick one of the systems A, B, or C.

3) Simulate the same step response, except applying the gains you chose in part 1 to the system A, B, or C.

4) How fast can you get your chosen system to within the ± 0.05 band around the goal, by setting the position gain and the velocity gain? This is one “limit” on how high you can set the gains.

5) Keeping the velocity gain at the same value as in part 4), increase the position gain until the system goes unstable (it has a growing oscillation or an exponential increase in the position). What is that value? This is another “limit” on how high you can set the gains.

6) Keeping the position gain at the same value as in part 4), increase the velocity gain until the system goes unstable. What is that value?

7) Use optimization to match or beat your best speed in part 4, by optimizing the position and velocity gains.

Extra Credit: You don't have to do this, but it is fun!

E1) Do parts 3, 4, 5, and 6 for all of the three systems (A, B, and C).

E2) How do your answers to parts 4, 5, and 6 depend on the parameters of the systems: the time constant t_c in A, the spring constant k in B, and the amount of delay d in C?

E3) We will eventually talk about state space controllers, which (in theory only, since there are always more unmodelled dynamics that limit gains) can make a system as fast or as stiff as you like. The first step is to define a state of the more complex system. In A, the actuator output would become another state variable and be added to the state vector. In B, the position and velocity of both masses (x_j and $\dot{x}_j$) would be in the state vector. In C, the contents of the delay line array are added to the state variable.

Side note: In general, anything a simulator has to remember from the previous simulation step should be part of the state vector.

Can you manually choose gains for the full state vector, and beat your best speed in part 4?

E4) Use optimization to choose the "full state" feedback gains in part E3.

E5) Do parts E2, E3, and E4 for all the three systems (A, B, and C).