Using the Smith Predictor to Control Systems with Long Delays

EE120 Fall 2016, GSI: Phil Sandborn

The forward velocity of a robotic lunar module is modeled by the following differential equation,

$$v(t) = k_{rover}u(t) - \tau_{rover}\frac{dv(t)}{dt}$$

where $\tau_{rover} = 0.3s$ (a system time-constant), $u(t)$ is the input signal, and k_{rover} is the proportional gain associated with the input.

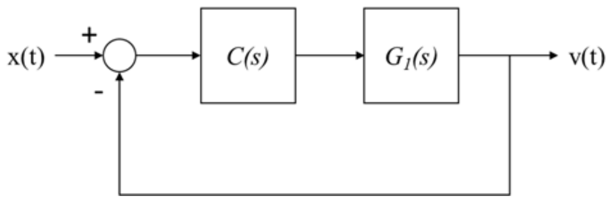
1. Write system function, $G_1(s) = \frac{V(s)}{U(s)}$, for the lunar module's equation of motion.
2. Construct a time-domain model with $k_{rover} = 0.5$, $\tau_{rover} = 0.3s$; plot the step-response.
   - Use discrete time, and select an appropriate sampling frequency, and formulate a matrix multiplication step to transform an input vector, unitStep[n], into an output vector, roverVelocity[n]

$$\begin{bmatrix} v_0 \\ \vdots \\ v_{N-1} \end{bmatrix} = \begin{pmatrix} a^0 & 0 & 0 \\ \vdots & \ddots & 0 \\ a^{N-1} & \cdots & a^0 \end{pmatrix} \begin{bmatrix} u_0 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

3. Specify a P-I controller (system function $C_1(s)$) to maintain the position/velocity of the robotic cart model. Choose K_{p1} and K_{i1} so that the closed-loop system (system function $H_1(s)$) has a specified open-loop bandwidth, $\omega_B = 6.75\ rad/s$ (equivalent to choosing K_{p1} and K_{i1} s.t. $\omega_{unity} = \omega_B$).

$$C_1(s) = K_{p1} + \frac{K_{i1}}{s}$$

$$H_1(s) = \frac{C_1(s)G_1(s)}{1 + C_1(s)G_1(s)}$$

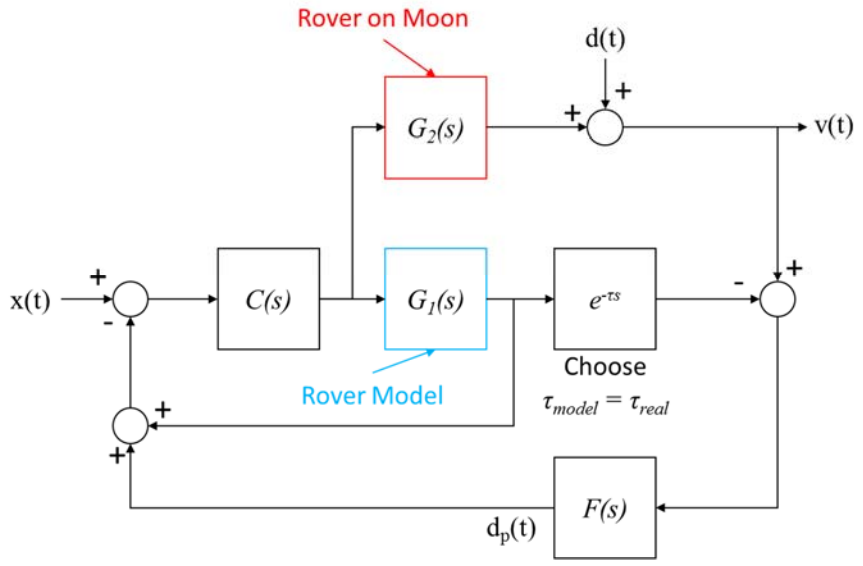


4. Implement the PI-controller helper function in the Python Notebook to build a PI-controller transformation matrix, similar to the formulation in Question 2. Use the helper functions to simulate the step-response of the closed-loop system.
5. Plot the bode diagram of the closed loop transfer function, $H_1(s)$. What is the bandwidth of this transfer function?
6. Introduce a time delay into the system, τ_{moon}, given that the nominal distance from the earth to the moon is 384.4x10^6 meters, and speed of light in vacuum is 3x10^8 m/s. Write the system function of the rover without controller but with delay, $G_2(s)$.
7. Implement the time delay (in the matrixDelay helper function) as the modification of a transformation matrix. Use this helper function to simulate the step-response of the delayed open-loop system, $G_2(s)$.
8. Construct a time-domain model of the closed-loop system using your controller ($C_1(s)$) and the system function with delay ($G_2(s)$). You may need to adjust the gain values of your controller so that the new system is stable! (Hint: it may require a smaller open-loop bandwidth than before.)

$$H_2(s) = \frac{C_1(s)G_2(s)}{1 + C_1(s)G_2(s)}$$

9. Examine the step response from your time-domain model. Why is your closed loop system with delay, $H_2(s)$, different from your closed-loop system without delay, $H_1(s)$?

Now, we'll construct a controller that implements an architecture known as the "Smith predictor," which allows us to increase stability while maintaining a high system function bandwidth. The block diagram for the Smith predictor in our case is shown below. Our new control system requires knowledge about $G_2(s)$ without delay (which we have already modeled as $G_1(s)$), and the delay.



Note: in this diagram, and in the Python code, G₁ does not contain delay, but G₂ does contain delay.

10. Using the system functions found in previous parts, derive a system function for the Smith predictor, $H_{smith}(s) = \frac{V(s)}{X(s)}$. Assume the velocity disturbance, d(t), is zero. Implement this system function as a matrix in the constructSmith helper function, and simulate a step-response. (Hint: you can use the same controller as you designed in Question 3, where the open loop bandwidth was 6.75 rad/s.)

In practice, the model may not be a perfect representation of the real system. Let's imagine a situation where the internal temperature of the rover's electronics varies in such a way that the gain and/or delay of the rover's transfer function drifts from nominal values.

11. Change the actual rover's delay to several values around the nominal value, and comment on the step-responses (use values: $0.95\tau_{moon}$, $0.99\tau_{moon}$, $1.01\tau_{moon}$, $1.05\tau_{moon}$).
12. Change the actual rover's gain to several different values around the nominal value, comment on the step-responses (use values: $0.95k_{rover}$, $0.99k_{rover}$, $1.01k_{rover}$, $1.05k_{rover}$).

We have now seen that when our actual system drifts from our model specification, various artifacts show up in the step responses. We can address these artifacts by examining the gain and phase margins for the Smith predictor block diagram.

13. Plot the bode diagram for the inner loop (H_1). What are the gain and phase margins for this inner loop (plot the frequency response for the open-loop gain)?
14. Plot the bode diagram for the outer loop (x to d_p), for a mismatched $G_1(s)$ and $G_2(s)$. What are the gain and phase margins for this loop (plot the frequency response for the outer-loop opened, but the inner-loop closed)?
15. We've ignored the outer-loop filter to now. How can we change F(s) to change the gain margin so that we can improve the stability of this system? Implement the changes in python.
16. Test the step-responses again for actual rover gains and delays that drift from nominal values, but using your new filter. Have your step-responses improved from your previous results?