Mini Project 1: Vibration Suppression for the Cart Pendulum in the Down Position

Requirements

For mini project one, your task is to design a few back control system that suppresses the vibrations of the cart pendulum system with the pendulum in the down position. For the final demonstration, the instructor will flick the pendulum, and your Arduino must record at least one second of data that shows the pendulum swinging before your vibration suppression controller turns on. You may start your Arduino code before or after the instructor flicks the pendulum. Your goal is to minimize the settling time of the pendulum as calculated by the script `mp1_settling_time_calc.py`. Your performance grade will be based on this settling time. Note that the script assumes your encoder is the output of a plant block called `G` and that you have one `if_then` block whose output becomes non-zero when the vibration suppression kicks on.

Additional testing requirements

- Your vibration suppression must switch on before the encoder peaks fall below 75 counts.
- The cart cannot move until the vibration suppression control turns on.

Required First Step

- You must verify the sign of your encoder before you can start this project

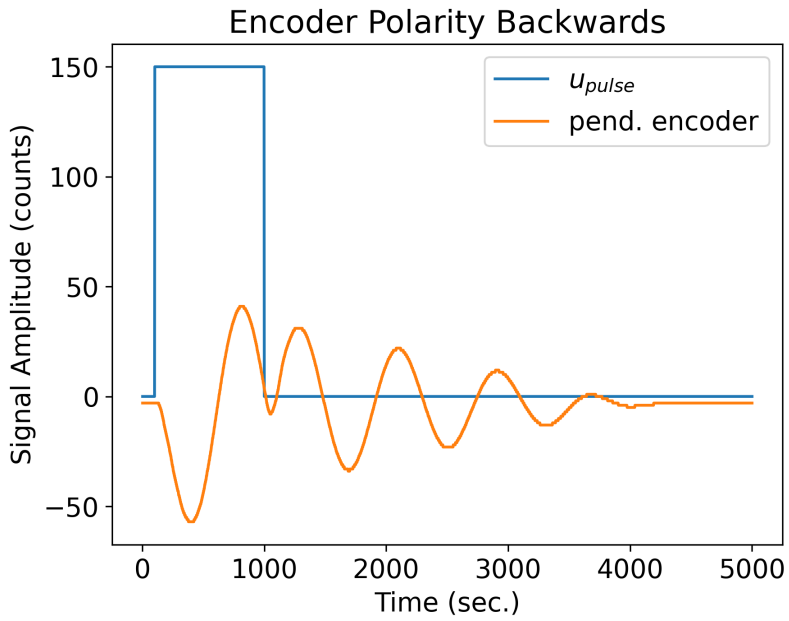
Open-Loop Pulse Test to Check the Sign

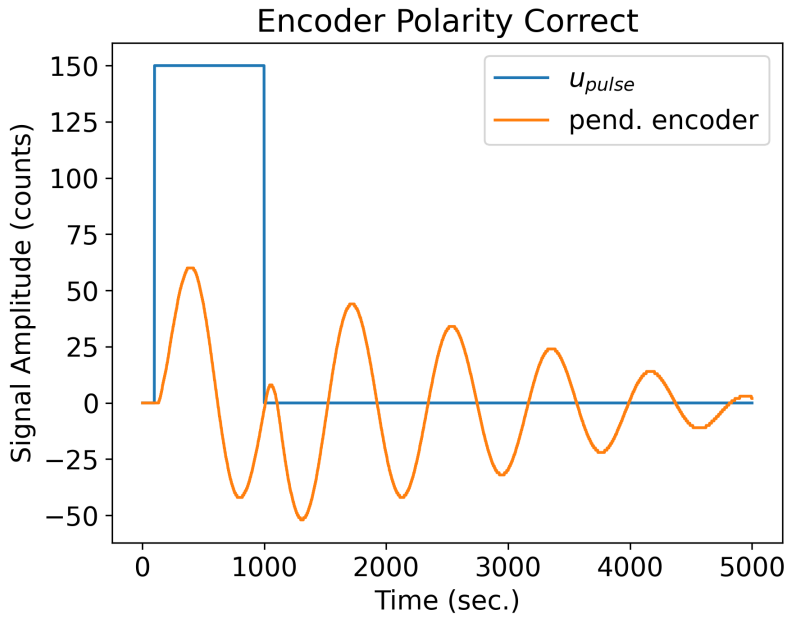
Trying to perform feedback control will go very badly if the signs are wrong. You need to verify that a forward pulse cause the encoder reading to increase. If this is not the case, swap the green and white wires on pins 2 and 3. In order to perform the test, create a positive pulse that is large enough to over come friction (probably 150-200 counts) and send it to both wheels.

In order to perform the open-loop pulse test, you need only two blocks: a pulse input and a `plant_with_double_actuator`. The `plant_with_double_actuator` has a custom actuator whose Arduino class is `cartmotors` and the sensor is an encoder with sensitivity of 4. The two inputs to the plant represent the two motors (right and left). Since we are not line following this year, you probably want them to have the same input (you may discover that a small correction factor is needed to drive straight).

Make sure you use the latest cart Arduino template. The current version is `arduino_template_cart_rev_3.ino`.

Here are examples of incorrect and correct encoder readings:





Note: The core issue is the whether the first hump on the encoder reading goes up or down. What happens after the pulse turns off will depend on the exact width of the pulse.

Additional required steps

- You will need a transfer function for the system in order to perform root locus control design. In order to find the transfer function, you need to perform an open-loop swept sign test. Because the system is open-loop stable, you do not need feedback for your

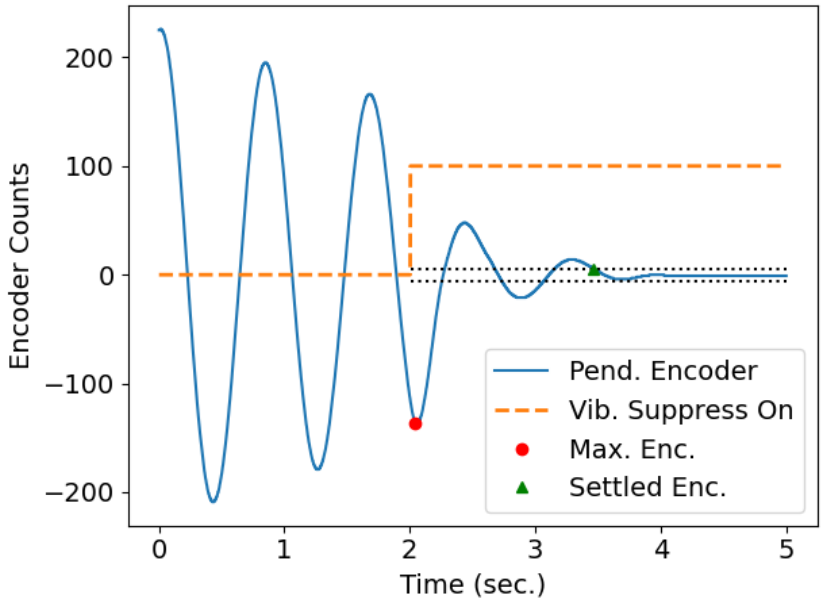
swept sine test. You should only have two blocks for the swept sine test: the swept sine input block and the `plant_with_double_actuaor` plant block.

- Once you have a transfer function for the system, you are ready to begin root locus control design. Keep in mind that your goal is to minimize settling time.

Recommended additional steps

- You will likely want to attempt PD tuning to see if you can get a good settling time that way. You might do This for two reasons:
  - You are not yet ready to do root locus control design
  - You want to see if manual tuning improves the Settling time after you have completed root locus control design
- You may find that you can improve the repeatability of your settling time if you include additional conditions for turning on your vibration suppression beyond just waiting for a set period of time

Example Vibration Suppression Settling Time Graph



Delaying when the vibration suppression kicks on

Your pendulum must swing for at least one second before your vibration suppression turns on. In order to make this happen, you need to use an `if_block` to send zero to your motors until a condition is met. Use a `loop_count` block to measure time. There are 500 loop counts per second. There is a variable on the Arduino that gets incremented each time the timer interrupt occurs. If you combine the `loop_count` with a greater than block and

the `if_block`, you can make the Arduino wait to turn on the vibration suppression until a condition is satisfied.