Modeling a DC Motor in Simulink - Part 2

Lab 3, ESDM

Objective

Using the Simulink/Simscape DC Motor model in simple simulations.

Theoretical aspects

TBD.

Design patterns

- Design pattern = a reusable generic pattern

Start-stop with S-R Flip Flop

- When you need a boolean condition (e.g. like On/Off) where:
  - the On condition depends on some conditions happening
  - the Off condition depends on some totally different conditions happening
- Then you may need to use the S-R Flip Flop block

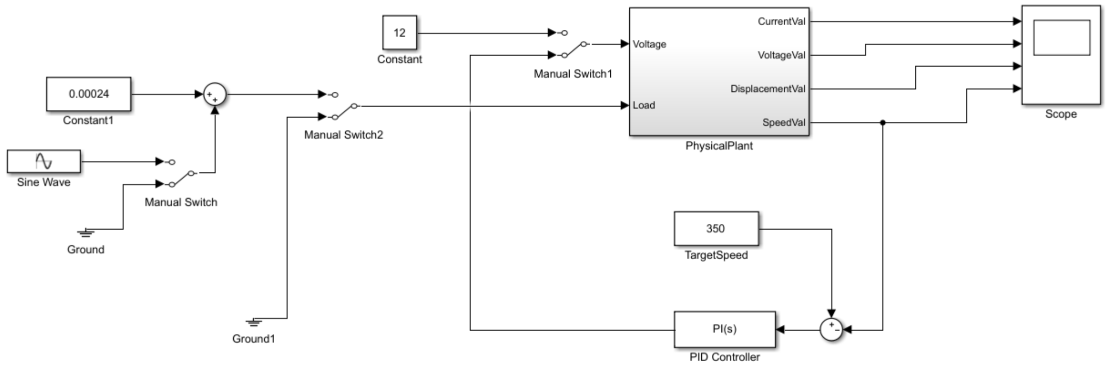
Exercises

1. Load the DC Motor Simulink model file that you created at the last lab. Alternatively, load the Simulink model file `Lab3_Start.slx`, provided in the current lab.

2. Simulate a variable load.

   Add to the constant load input a 10% slow sinusoidal variation (amplitude = 10% of the constant value). Observe the outputs. What happens to the speed?

3. Stabilize the speed with a PID block in a feedback connection, replacing the constant voltage input, as in the figure below. The desired target speed is a constant 350 rad/s.



   In the PID block parameters, use a 0 value for the I and D constant (only the P constant is non-zero). Choose an appropriate P value so that the motor works reasonably well.

   a. What happens to the motor speed?
   b. What happens at the beginning of the simulation?
   c. Is the target speed actually reached?
   d. What is the influence of the P value? Modify it and observe the differences.

4. Set a small non-zero value for the I value as well.

   a. What is the effect? You may need to change the value manually until finding a good value.
   b. Once you found a good value, what is the effect of decreasing the value of I? What if we increase it (a little)?

5. Use the `Tune...` button in the PID block parameters to access Matlab's PID tuning tool.

   a. Play around and observe the effects.
   b. How is the behavior if you use only a PI-type controller? (D component is 0)

6. Update the motor operation in case hard-stop detection from the previous lab, as follows:

    - Initially the motor is stopped and remains stopped
    - The motor is started with a TRUE value on a new boolean input called `StartMotor`.
    - Once a motor is started, the input `StartMotor` can become false, without stopping the motor
    - A hard-stop is detected as it was implemented last time (speed < threshold AND current > threshold)
    - When hard-stop is detected, the motor supply is shut down
    - The motor remains stopped until another pulse on the `StartMotor` input arrives

7. Update the previous design with the following requirement:

    - Once a motor is stopped due to hard-stop, it shall be started only after at least 1 second has passed. If the `StartMotor` command is received during this 1 second, the start is delayed until the 1 second expires.