MDP LABORATORY

PID Control of 2-wheel drive Robot

Step 1: Initial Setup

Ensure robot motor driver, Arduino board, encoder is setup, as you would have done in the previous MDP lab sessions.

Step 2: Test Motor

To test the motor,

a. Setup the Arduino code to drive the motor at a set speed (-400 to 400). You will use the motor driver library `DualVNH5019MotorShield.h` for this (Refer to motor driver documentation for more details).

b. Ensure both wheels are rotating as expected, proportional to the set speed.

Step 3: Test Encoder

You need to be able to convert the encoder square-wave to wheel rpm.

a. First, read the encoder signal of each motor into any unused digital input pin of Arduino. You may use either output A or output B of each encoder.

b. Compute the time-width of the square-wave. pulseIn(pin,HIGH) is a Arduino function that can help you with this. Note that the width of the square wave is twice the time the pulse is high. (pulseIn function will limit the execution time of the loop. You may consider variants using interrupt driven calculation of the time-width)

c. Convert time-width to wheel rpm using the fact that there are 562.25 square waves for every revolution of wheel

d. Test the calculated rpm for different motor speeds.

Step 4: Perform step test

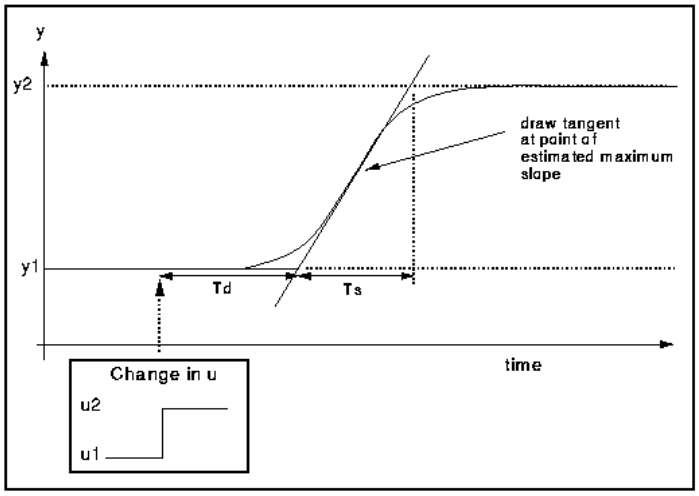
To model the motor, you need to perform a step test to determine the model characteristics.

a. Change the motor speed from 300 to 250 (close to average operating speed). Note the step change 'M' is -50.

b. Tabulate the recorded rpm from encoder, before and after the step change

c. Plot the rpm in excel or MATLAB

d. Approximately fit the graph to an ideal first-order response shown below. Compute K, τ_s, τ_d from the data (K = (y2-y1)/(u2-u1)) :



Step 5: Compute Kp,Ki,Kd for PID controller

From the model constants above, refer to the Ziegler-Nichols Open Loop tuning table below to compute kp,ki,kd for the PID controller :

Note: $K_p = K_c$, $K_i = K_c/T_i$, $K_d = K_c * T_d$

Controller	Parameters		
	K_c	T_i	T_d
P	$\frac{\tau_s}{K\tau_d}$		
PI	$\frac{0.9\tau_s}{K\tau_d}$	$\frac{\tau_d}{0.3}$	
PID	$\frac{1.2\tau_s}{K\tau_d}$	$2\tau_d$	$0.5\tau_d$

Seborg, et al, Process Dynamics and Control, Second Edition..

If you are not satisfied with the PID performance, you may slightly tweak the parameters or try PI controller instead by referring back to this table.

Step 6: Code the PID controller in Arduino

Follow the algorithm below to code the digital implementation of PID controller in Arduino (for each motor) –

A. Set required wheel speed ‘set_rpm’ (0 to 130)
B. Calculate k1,k2,k3
   Where
   K1 = Kp+Ki+Kd
   K2 = -Kp-2*Kd
   K3 = Kd
C. Read encoder rpm into ‘y’ (Data Acquisition)
D. Calculate error between ‘set_rpm’ and ‘y’
E. Compute ‘u’ using digital PID control law

$$u[k] = u[k-1] + K_1 * e[k] + K_2 * e[k-1] + K_3 * e[k-2]$$

F. Send 'u' to output for control as the motor speed
G. Wait for 'xx' msec (sampling time) before continuing
H. Repeat B to G

Step 7: Test straight line motion

Setting the same speed set points for both the wheels, test the 2-wheel drive motion against a straight line. If significant deviation is observed, try different variations of the model parameters and the tuning gains computed in steps 4 and 5 respectively.