PID Control of a Robot Via Position Feedback

Nick Pham

Setup and Procedure

Please refer to the lab handout for this information.

Prelab

(a) The simulink model is a typical feedback controller. The plant is the state space model described in the lab specification:

$$\dot{s} = v \tag{1}$$
$$\dot{v} = -av + bu \tag{2}$$

The velocity plus noise is measured at the A-D converter, which is then added to the desired reference signal and input into the controller. This is a PID controller, which uses a linear combination of the proportional, integral, and derivative of the error to determine the control input signal to the plant.

(b) The initial values for the coefficients were $K_p = 100$, $K_i = 20$, and $K_d = 0.3$. This produced an underdamped output for a few seconds before settling to the desired speed (see Figure 1).

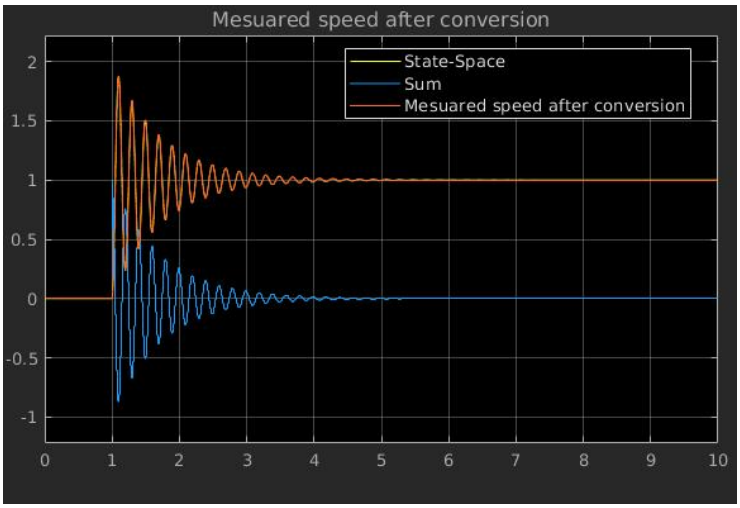

Fig. 1: Underdamped response for $K_p = 100$, $K_i = 20$, and $K_d = 0.3$.

Increasing the derivative component K_d to 10 quickly reduces the oscillations (see Figure 2). Because the "strength" of the derivative part of the control was increased, there was a strong corrective action during the high acceleration phase, resulting in much less overshoot and stability in less than a second.

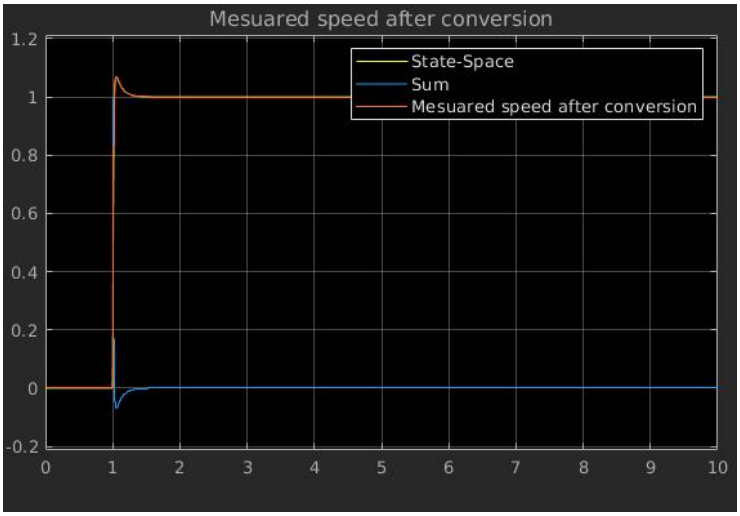

Fig. 2: System response for $K_p = 100$, $K_i = 20$, and $K_d = 10$.

Not all combinations of coefficients result in a successful and stable system. For intance, removing the proportional control coefficient altogether results

in wild and increasing oscillations (see Figure 3). Notice that after 20 seconds, the oscillations grow on the order of 10^{16} times the desired speed.

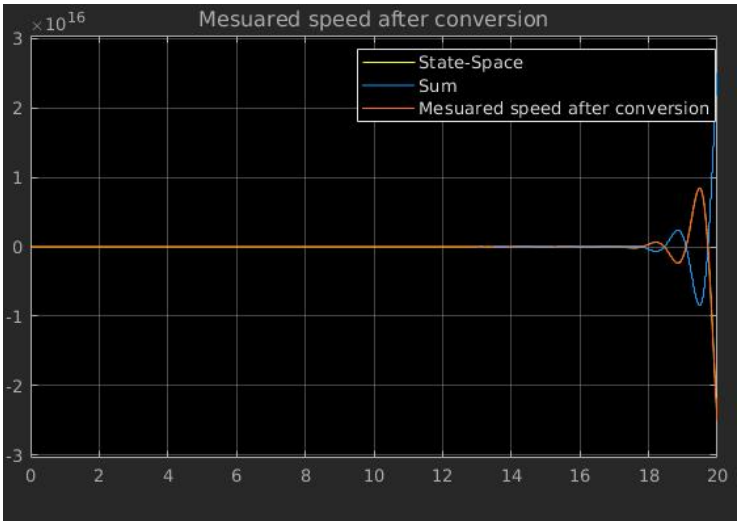

Fig. 3: Oscillatory response for $K_p = 0$, $K_i = 20$, and $K_d = 0.3$.

(c) To choose these parameters, first find a good k_p value for minimal disturbance. Then, choose a k_d and k_i values to help find a faster convergence without too much overshoot or oscillations.

(d) The PID auto tuner can be used to set the PID coeffcients for various response times and transient behaviors. For high performance (response time set to 0.01 seconds and transient behavior set to 0.9), the tuner gave parameters $K_p = 140$, $K_i = 245$, and $K_d = 20$. This behavior has only a 3.5% overshoot. This gave fairly similar performance characteristics to the set of coefficients used in Figure 2, fast setting with some small overshoot.

(e) Adding noise to the measurement resulted in similar noise in the output when using the settings from the auto tuner (see Figure 4). This probably resulted from the very fast response time, so the output was following the noise accurately. To improve the quality of the output, the PID controller should filter the signal to remove the noise. Because the mean of the noise is still 0, a low pass filter could perform this task, which would be implemented by increasing the relative importance of the integration coefficient K_i.

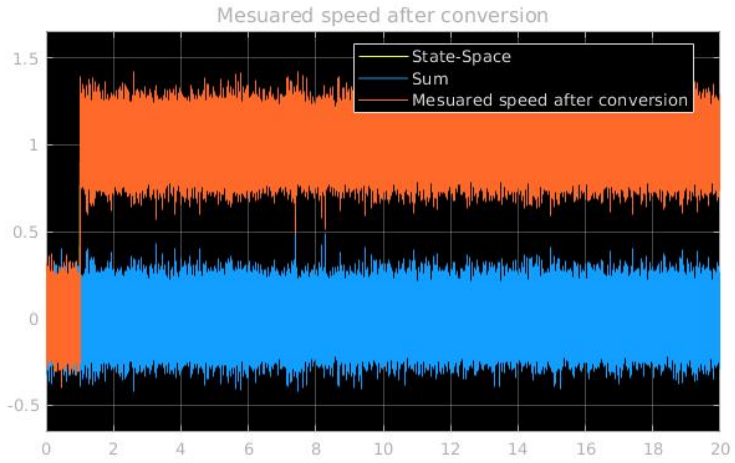

Fig. 4: System response with auto tuned parameters $K_p = 140$, $K_i = 245$, and $K_d = 20$ to noisy input with $var = 0.01$

(f) The sample rate of the measurement also has an important effect on the response of the system. Decreasing the sample time from 0.0001 to 0.005 results in larger overshoot (see Figure 5). Decreasing the sample time further to 0.01 results in unstable oscillations (see Figure 6).

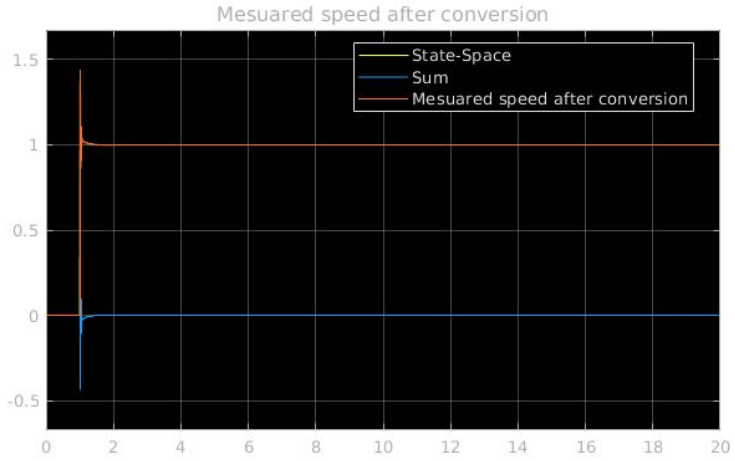

Fig. 5: System response with auto tuned parameters $K_p = 140$, $K_i = 245$, and $K_d = 20$ with sample rate limited to 0.005. Note the large overshoot.

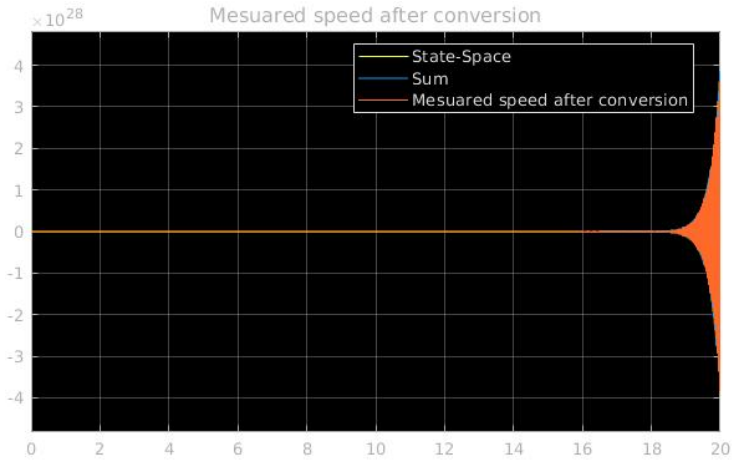

Fig. 6: Unstable system response with auto tuned parameters $K_p = 140$, $K_i = 245$, and $K_d = 20$ with sample rate limited to 0.01

(g) To further illustrate the effects of response time on the performance of the system, a sinusoidal input can be used in place of the step input. Reseting the sample time back to 0.0001, at relatively low frequencies such as $6.3rad/sec = 1Hz$, the output tracks the input wave closely, with error on the order of 10^{-2} (see Figure 7). However, increasing the frequency of the reference sine wave to $100Hz$ resulted in error on the order of 10^0, which is about the same amplitude as the reference sine wave itself (see Figure 8.

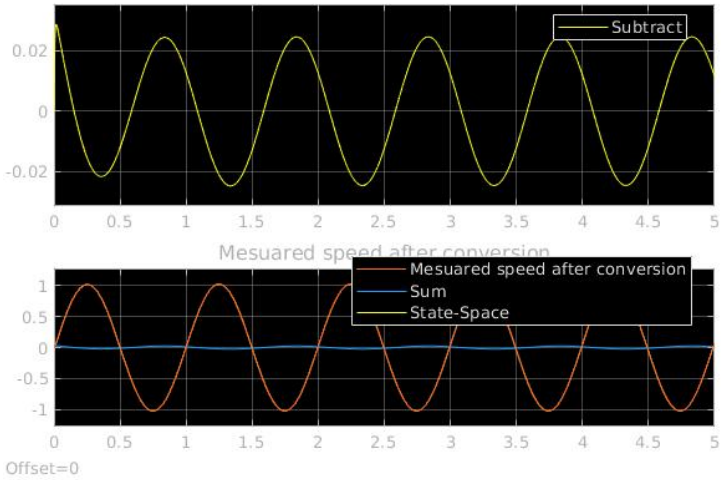

Fig. 7: System response with auto tuned parameters $K_p = 140$, $K_i = 245$, and $K_d = 20$ to sinusoidal reference input with frequency $1Hz$. The uppper plot shows the error signal, on the order of 10^{-2}.

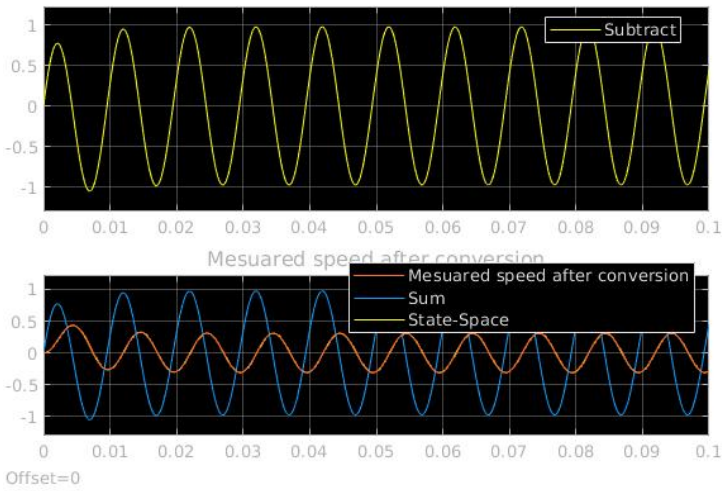

Fig. 8: System response with auto tuned parameters $K_p = 140$, $K_i = 245$, and $K_d = 20$ to sinusoidal reference input with frequency $1Hz$. Note the much larger error signal.

Lab Results and Discussion

Tuning the PID started with finding a suitable k_p coefficient. On the suggestion that the range [2, 5] would be a good place to start, the first test was for

$(k_p, k_i, k_d) = (2, 0, 0)$. Figure 9 shows the system measurements for these values. Note that the initial condition of the robot was below 0, so the graph is shifted in time somewhat. With this value, there was some overshoot, but little oscillations. The PWM of the motor did clip though.

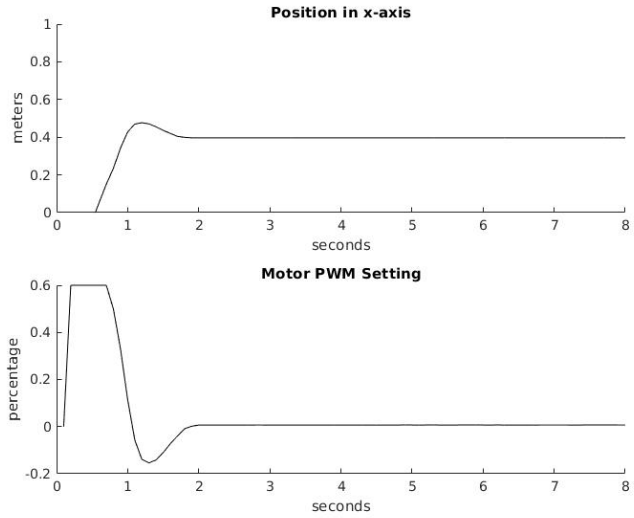

Fig. 9: System response to $(k_p, k_i, k_d) = (2, 0, 0)$.

Increasing k_p to 3.5 increased the overshoot and hence oscillations, as seen in Figure **??**. The graph shows that there may have been some error, as there were some blips a few seconds after the system seemed to settle.

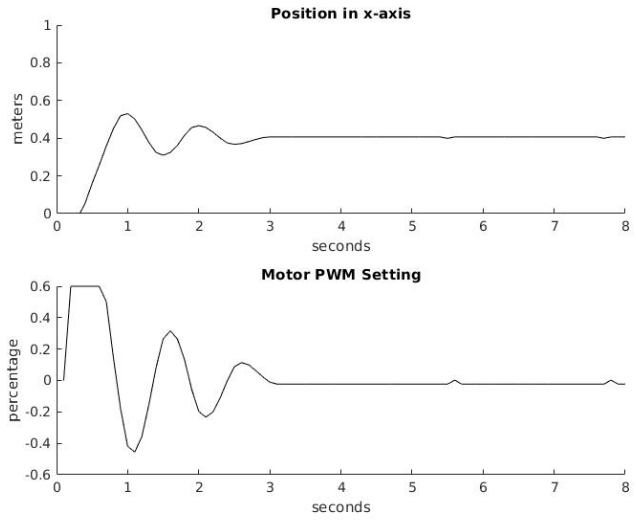

Fig. 10: System response to $(k_p, k_i, k_d) = (3.5, 0, 0)$.

To reduce constant error, the k_i parameter was increased to 1. Figure **??** shows the result, which does not converge as well as the just proportional controller. This might be because the integral part is outweighing the proportional control. For best results, it is usually good to keep the integral and proportional coefficients much less than the proportional control.

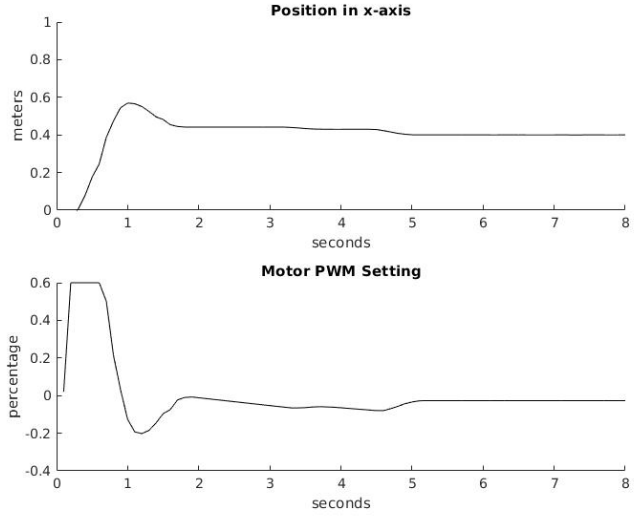

Fig. 11: System response to $(k_p, k_i, k_d) = (2, 1, 0)$.

Decreasing the k_i parameter to 0.05 and increase the proportional control to 4 allowed the system to converge, but still with some oscillations, as seen in Figure **??**.

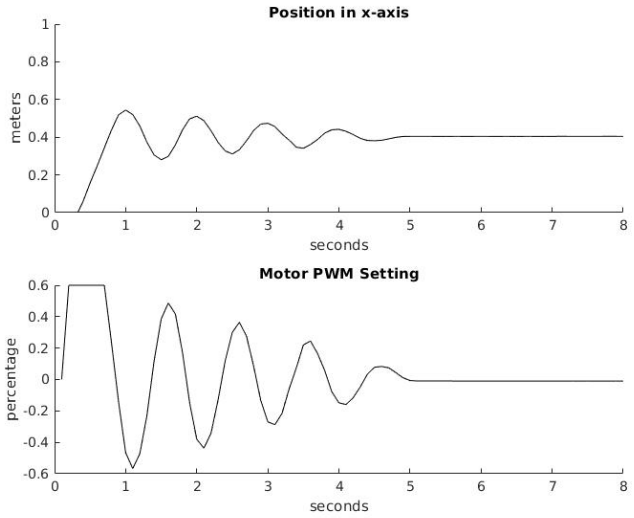

Fig. 12: System response to $(k_p, k_i, k_d) = (4, 0.05, 0)$.

Finally, a derivative control was also included to encourage faster convergence, as seen in Figure **??**. However, this was not much if at all better than just the proportional control. Though in idealized simulations PID controllers do work better than just proportional controls, in the real world with noisy and imprecise measures and non ideal actuators (the motors likely required a certain threshold power to make the vehicle move), a proportional only controller can be sufficient in many cases.

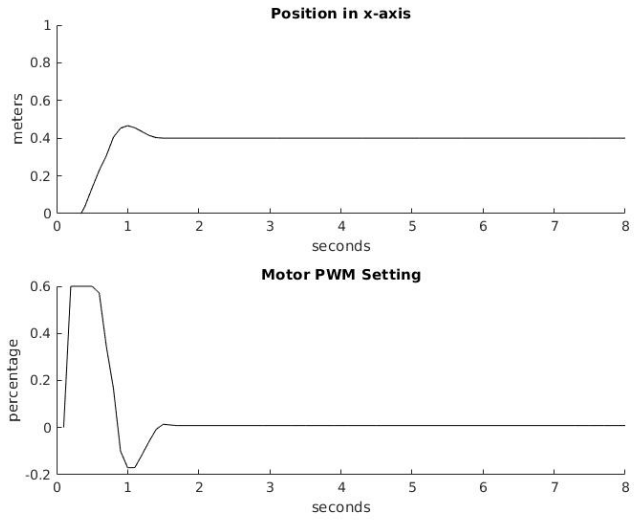

Fig. 13: System response to $(k_p, k_i, k_d) = (2.5, 0.05, 0.1)$.

In summary, the k_p parameter increases the feedback control input when the error between the system output and the desired output is large. This should be the main component of the control. Large k_p values likely require large resources, so there is an upper bound on this value in practice. However, larger values of k_p will lead to reaching the target faster, but at the cost of higher overshoot. Lower values will have less overshoot but take longer to reach the target.

The k_i parameter accounts for the integral for the integral of the error, and is used to correct for long term accuracy error offsets. It can correct for calibration errors in measurement and actuation devices, but should not be too large compared to the proportional parameter.

The k_d parameter can be used to decrease the response time of the system, because at larger rates of change the control input will be increased. Like the k_i parameter, it should not be too large compared to k_p. Setting k_i and k_p requires iterative trial and error for fine tuning, and the two must be balanced with each other.