ME 586: Project 1 Report
Air-Driven Engine Control

Harsh Savla & TJ Wiegman

2022-12-02

Abstract

In this lab, an STM32F100RB microcontroller programmed in C was used to control an air-driven engine. The overall system was first identified by measuring the step response of both the electronically-controlled valve that allowed air into the engine as well as the step response of the engine itself. From there, optimal K_p and K_i coefficients were calculated, and the system was succesfully run with a PI controller, even stabilizing itself after experiencing external disturbances. From this project, the authors learned the importance of carefully keeping consistent units as they identify and calculate controller coefficients for an unknown system in order to create a succesful controller.

1 System Identification

First and foremost, the system was calibrated across input voltages ranging from 0 to 1 V, which produced output engine speeds from 0 to 1350 RPM at zero load. Based on the information gathered during calibration, system identification was carried out for step inputs of 0 to 0.5 V for the electronically-controlled valve and 0 to 13 PSI for the engine. All experiments were carried out with a supply pressure of 40 PSI.

To mesure the step response of the electronically-controlled valve, the input voltage was held at 0 V for two seconds, and then it was changed to 0.5 V (2176 counts on the microcontroller's DAC). The load on the engine was set such that the engine did not rotate during any part of this process. This experiment was carried out for a total time period of 10 seconds. The pressure output of the valve was recorded at 10 ms resolution and plotted in Figure 1. The shape of that plot reveals that the valve behaved as a first order system, as expected. From that plot, the time constant τ_{valve} and the gain K_{valve} were determined to be 0.6 and -0.1953, respectively.

To measure the step response of the engine, the electronically-controlled valve was given an input voltage such that it output 13 PSI, and allowed to saturate the air input system at that pressure. While that pressure built up, the switch connecting the engine to the air input system was held in the off position. After approximately two seconds, the switch was flipped such that the engine experienced a sudden change in input pressure from 0 to 13 PSI. The experiment was carried out for a total time period of 10 seconds. The engine's output voltage was recorded at 10 ms resolution and plotted in Figure 2. The shape of that plot reveals that the engine behaves as a first order system, as expected. From that plot, the time constant τ_{engine} and gain K_{engine} were determined to be 0.14 and 23.9, respectively.

2 Controller Design

This system had three control requirements. It needed a damping ratio ζ of 0.707, a 2% settling time t_s of 3 seconds, and no steady-state error. In order to achive these goals, a simple PI controller was sufficient. The system, with controller installed, was modeled as shown in Figure 3.

Figure 1: The input air valve's step response to a sudden change in control voltage.

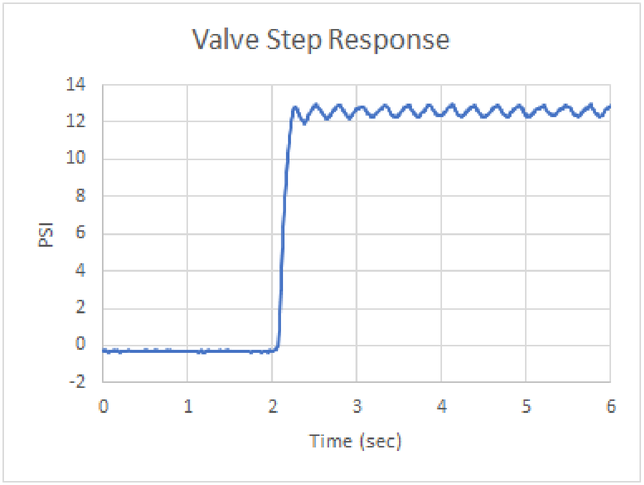


Figure 2: The engine's step response to a sudden change in input pressure.

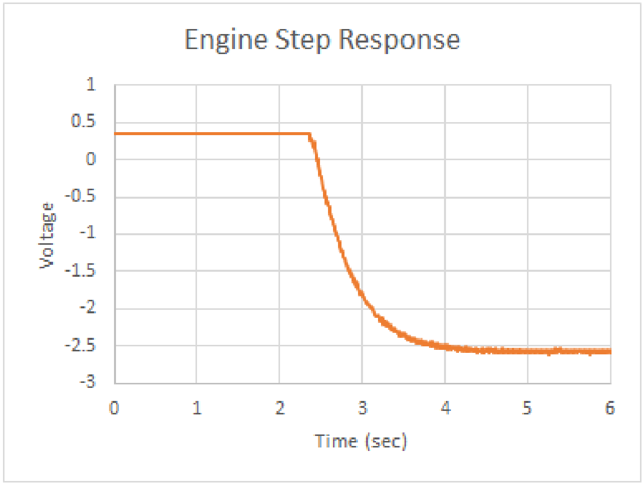
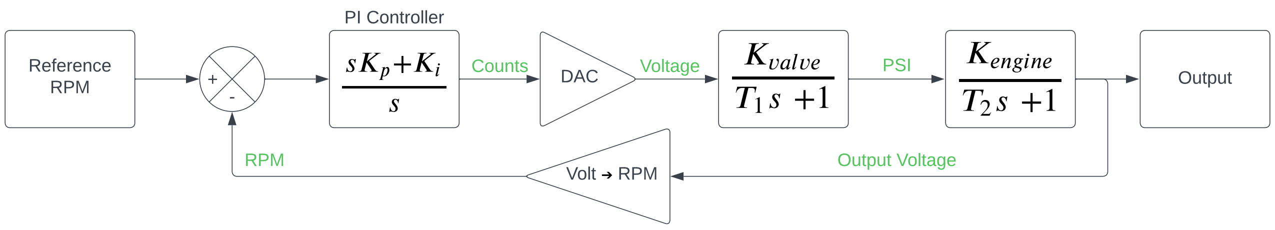


Figure 3: A PI controller gives input to a 1st-order valve, which gives input to a 1st-order engine, which feeds back its output voltage to the PI controller. Units for each signal are shown in green.



3 Simulation and Implementation

A simulink model of the system was created as shown in Appendix A.1. After solving the system transfer equations by hand, optimal controller coefficients K_p and K_i were determined to be -0.1441 and -0.3935, respectively. Using those coefficients, the simulink model was used to predict the system's response to a step input from 0 to 800 RPM. The simulated output is plotted in Figure 4.

To run the controller with physical hardware, it was implemented in C. The code is shown in Appendix A.2, and it was run on an STM32F100RB microcontroller at a sample rate of 10 ms. The system's respose to a step input from 0 to 800 RPM, just as simulated with the simulink model, was recorded and then plotted in Figure 5. To test the system's resilience against outside disturbance, the load was adjusted mid-run and the response recorded and then plotted in Figure 6.

4 Conclusion

This lab gave excellent practice on the common tasks of system identification and controller design for a practical electromechanical system. The most important learning outcome from this project was that the units used in the model absolutely need to be consistent across all the pieces of the system, as otherwise the (incorrectly) calculated coefficients will destabilize the system. Our first attempt to control the system was unsuccessful, but after carefully redoing our calculations with a corrected model architecture, the updated controller worked flawlessly.

Figure 4: Simulated output for an 800 RPM step input as the reference signal.

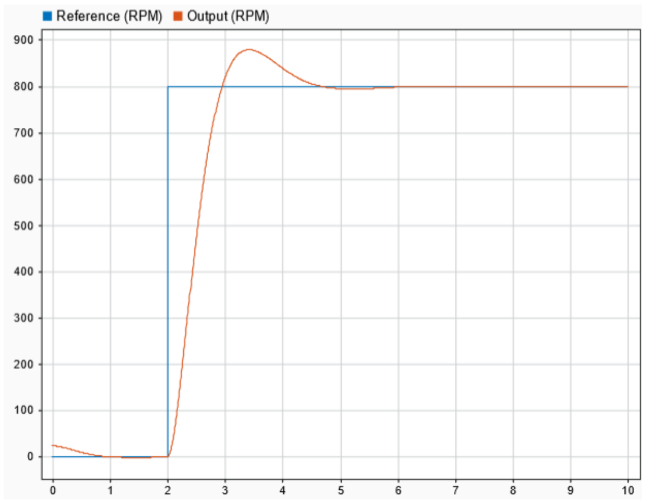


Figure 5: Recorded output for an 800 RPM step input as the reference signal on a physical system.

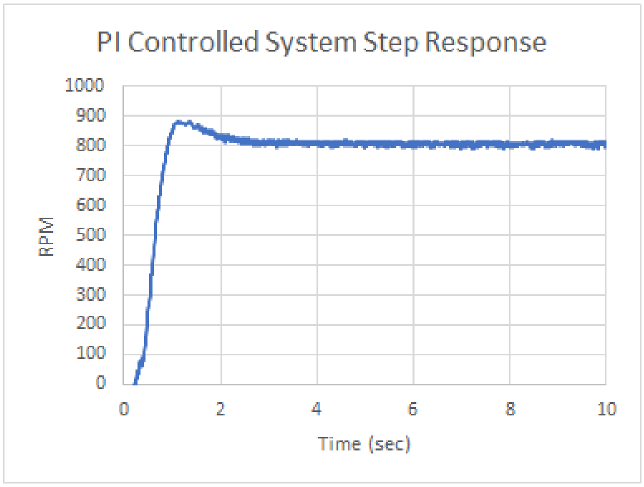
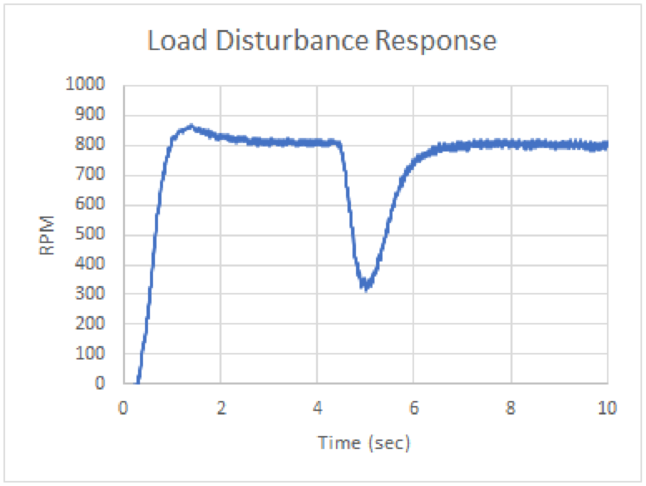
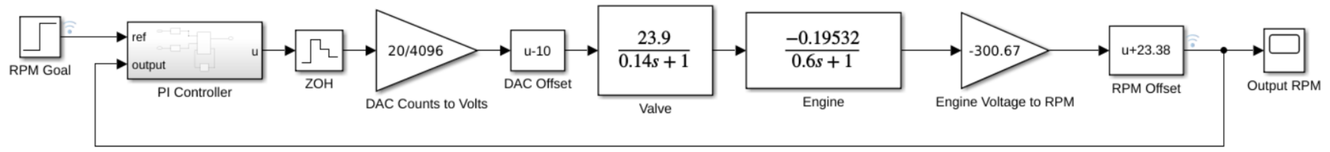


Figure 6: Recorded output for an 800 RPM step input, with reactions to an external disturbance.



A Appendix

A.1 Simulink System Model (With PI Controller)



A.2 PID Controller (Timer Interrupt) Code

```c
// DAC bounds are constants
#define OUTPUT_MAX 4095
#define OUTPUT_MIN 0

// Global variables
int ram_ptr[1000]; // measured data stored in memory
int stored_actual = 0; // number of values actually written
float time_period = 10; // in milliseconds
float setpoint = 0;
float error_prior = 0;
float integral = 0;

// PID Coefficients (D = 0, for PI control)
float Kp = -0.14411;
float Ki = -0.393539;
float Kd = 0.0;

// Timer interrupt routine
void timehand(void) {
    // Setup
    int measured_ADC;
    float measured_engvolt;
    float error;
    float derivative;
    float output;
    float reference;

    // Get position from ADC input
    measured_ADC = a_to_d(1);

    // Convert ADC count to engine voltage
    measured_engvolt = 0.0024414*measured_ADC - 5;

    // Convert ADC count to RPM and save to memory
    if (stored_actual < 1000) {
        ram_ptr[stored_actual] = -0.6984*measured_ADC + 1504;
        stored_actual = stored_actual + 1;
    }

    // PID Controller
    reference = -0.0033*setpoint + 0.0762; // convert setpoint from RPM to engine voltage
    error = reference - measured_engvolt;
    integral = integral + (error * (time_period / 1000)); // T/1000 to convert ms to seconds
    derivative = (error - error_prior) / (time_period / 1000);
    output = (Kp * error) + (Ki * integral) + (Kd * derivative);
    error_prior = error;

    // Convert output voltage to DAC count (within bounds)
    output = 204.8*output + 2048;
    if (output > OUTPUT_MAX) output = OUTPUT_MAX;
    if (output < OUTPUT_MIN) output = OUTPUT_MIN;

    d_to_a(0, output);
}
```