Mini Project Simulation & Control

The following code was written for the Arduino microcontroller. The code sends a PWM to the motor for one second and displays motor voltage, current time, and velocity of the wheel. This data was then copied into MATLAB to graph the open loop step response.

Arduino Code:

// Authors Luke Henke and Gabe Alcantar-Lopez

// Arduino step response experiment

//Define constants and import Encoder.h library

#define ENCODER\_OPTIMIZE\_INTERRUPTS

#include <Encoder.h>

#define PERIOD 5

#define encoderOneAPin 2

#define encoderOneBPin 3

#define MOTOR\_ONE\_PWM 9

#define ENABLE\_PIN 4

#define MOTOR\_ONE\_DIRECTION 7

#define COUNTS\_PER\_ROTATION 64

//Set encoder pins

Encoder wheel(encoderOneAPin, encoderOneBPin);

int motorVoltage = 0;

void setup() {

Serial.begin(74880);

//Set motor PWM and Motor direction pins as outputs

pinMode(MOTOR\_ONE\_PWM, OUTPUT);

pinMode(MOTOR\_ONE\_DIRECTION, OUTPUT);

//Assign PWM as analog value from Arduino

analogWrite(MOTOR\_ONE\_PWM, motorVoltage);

//Assign direction as digital input from Arduino

digitalWrite(MOTOR\_ONE\_DIRECTION, 1);

pinMode(ENABLE\_PIN, OUTPUT);

digitalWrite(ENABLE\_PIN, 1);

}

void loop() {

//Variable to read current time

unsigned long time\_now = millis();

static bool startedStep = false;

//IF statement to ensure 1 second has elapsed before starting motor

if(time\_now > 1000 && startedStep == false){

//Set motor voltage to full value

motorVoltage = 255;

analogWrite(MOTOR\_ONE\_PWM, motorVoltage);

startedStep = true;

}

//IF statement to ensure data is read between 1 and 2 second

if (time\_now > 1000 && time\_now < 2000){

static double oldPosition = 0;

double newPosition = wheel.read()\*2\*PI / COUNTS\_PER\_ROTATION;

double velocity = (newPosition - oldPosition) / PERIOD \* 1e3;

//Print values of interest

Serial.print(time\_now);

Serial.print("\t");

Serial.print(motorVoltage);

Serial.print("\t");

Serial.print(velocity);

Serial.print("\t");

Serial.print(wheel.read());

Serial.println();

oldPosition = newPosition;

}

unsigned long currentTime = millis();

if( currentTime - time\_now > PERIOD){

Serial.println("ERROR - Main takes too long");

}

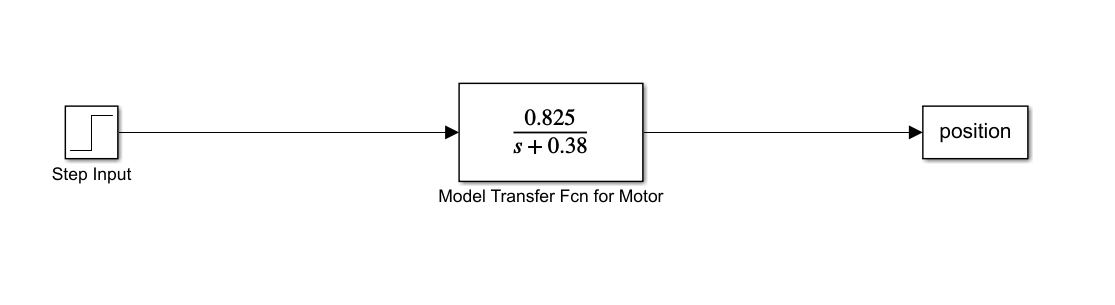
while(millis() < time\_now + PERIOD){

//wait approx. [period] ms

}

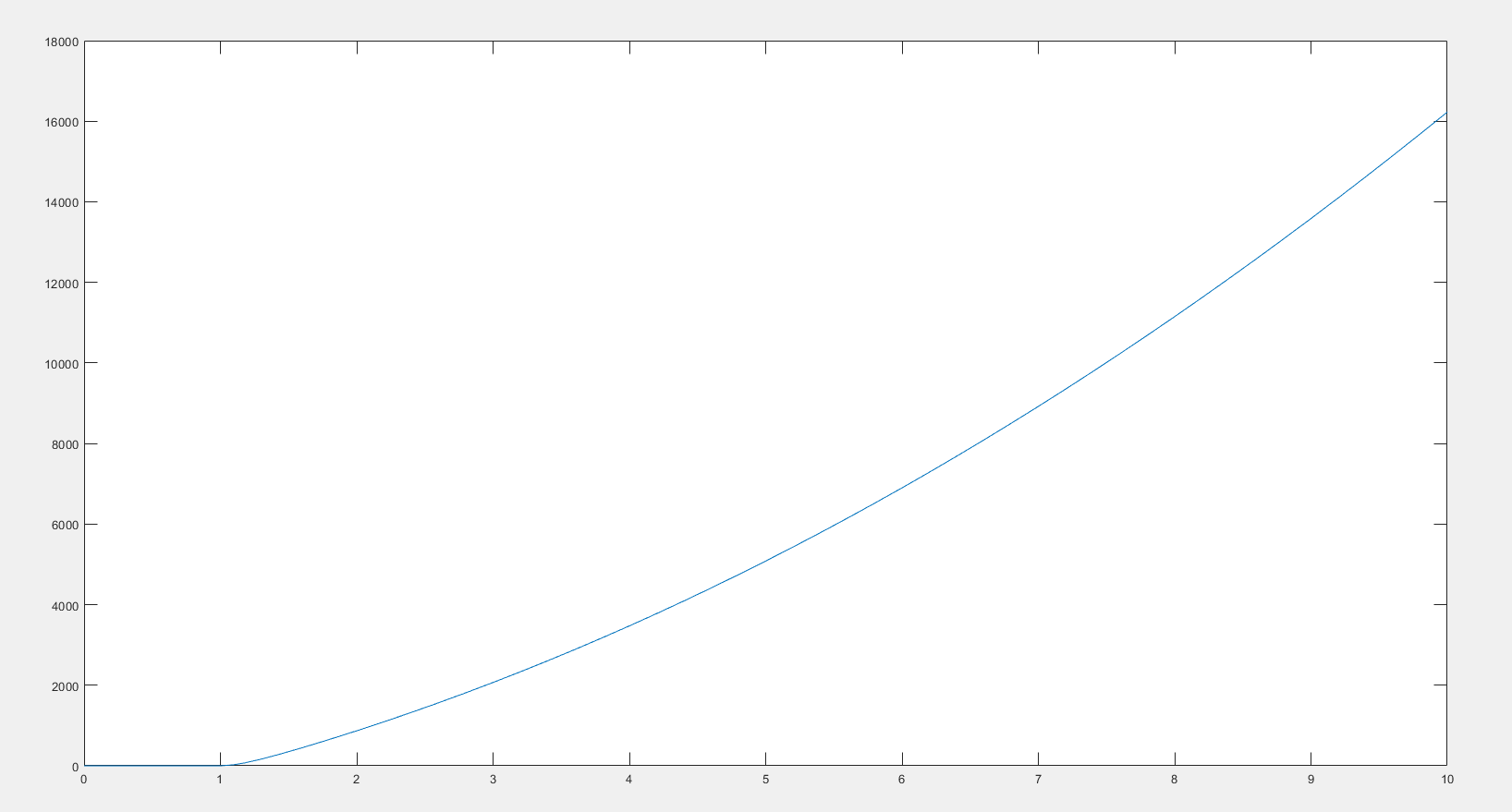
}

The data was then logged into MATLAB and used to create a simulated block diagram as shown in Figure 1. The block diagram was modeled after the EENG307 notes which discuss identifying a system and modeling the system as a transfer function. The transfer function model in this instance is shown in Equation 1.



**Figure 1**: Block diagram for simulated open loop step response.

The open loop response from the MATLAB code is shown in Figure 2. As one can see, the open loop response of the system is not very useful since the response is unbounded and therefore provides no control of the system.



**Figure 2**: Graph of simulated open loop step response.

The model transfer function was compared against the data from the Arduino when running step response of the motor. The following MATLAB code was used to plot the actual motor response with the simulated step response on the same graph as shown in Figure 3.

MATLAB Code:

%% Setup transfer function variable S

s=tf('s');

%% Set time equal to column1 of array from Arduino step experiment

time = results(:,1)-1310;

%% Set velocity equal to column3 of array from Arduino step experiment

velocity = results(:,3);

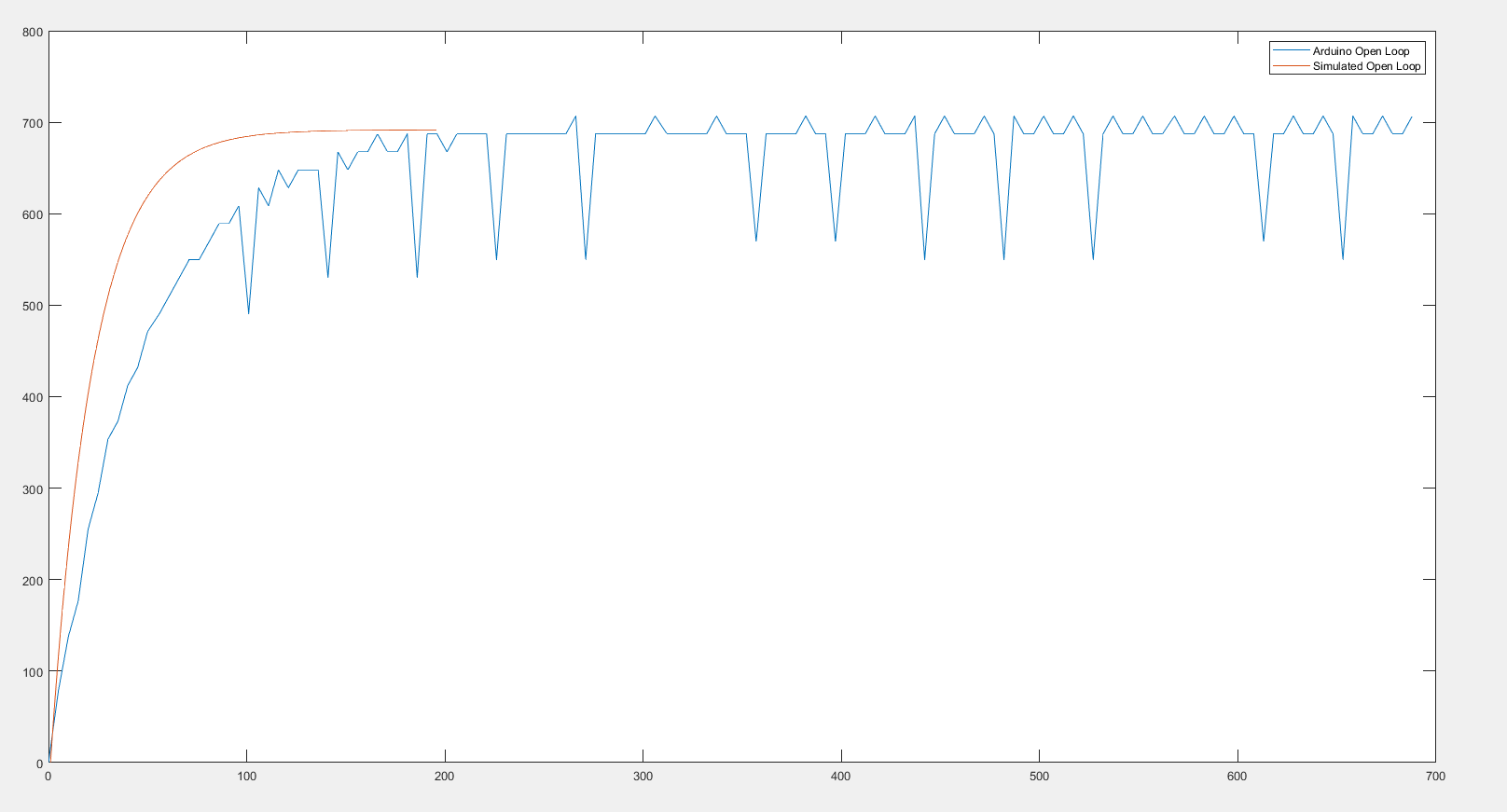
%% Model transfer function from EENG307 notes

sys = tf(83,[1 0.12]);

%% Plot Arduino data and modeled data on same graph

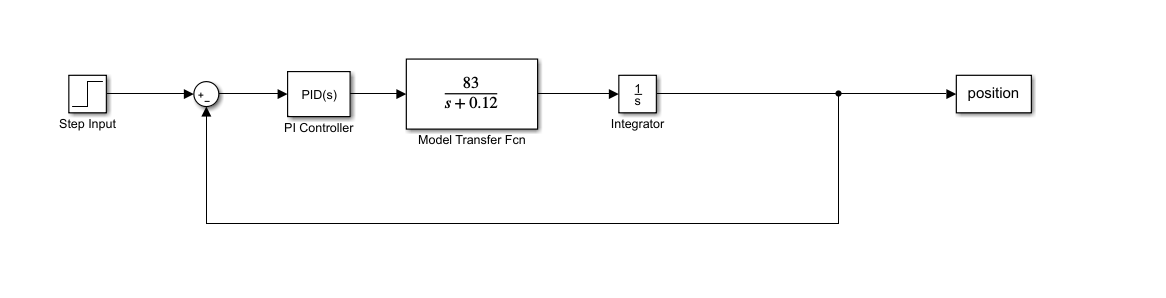
figure (1)

plot(time,velocity);



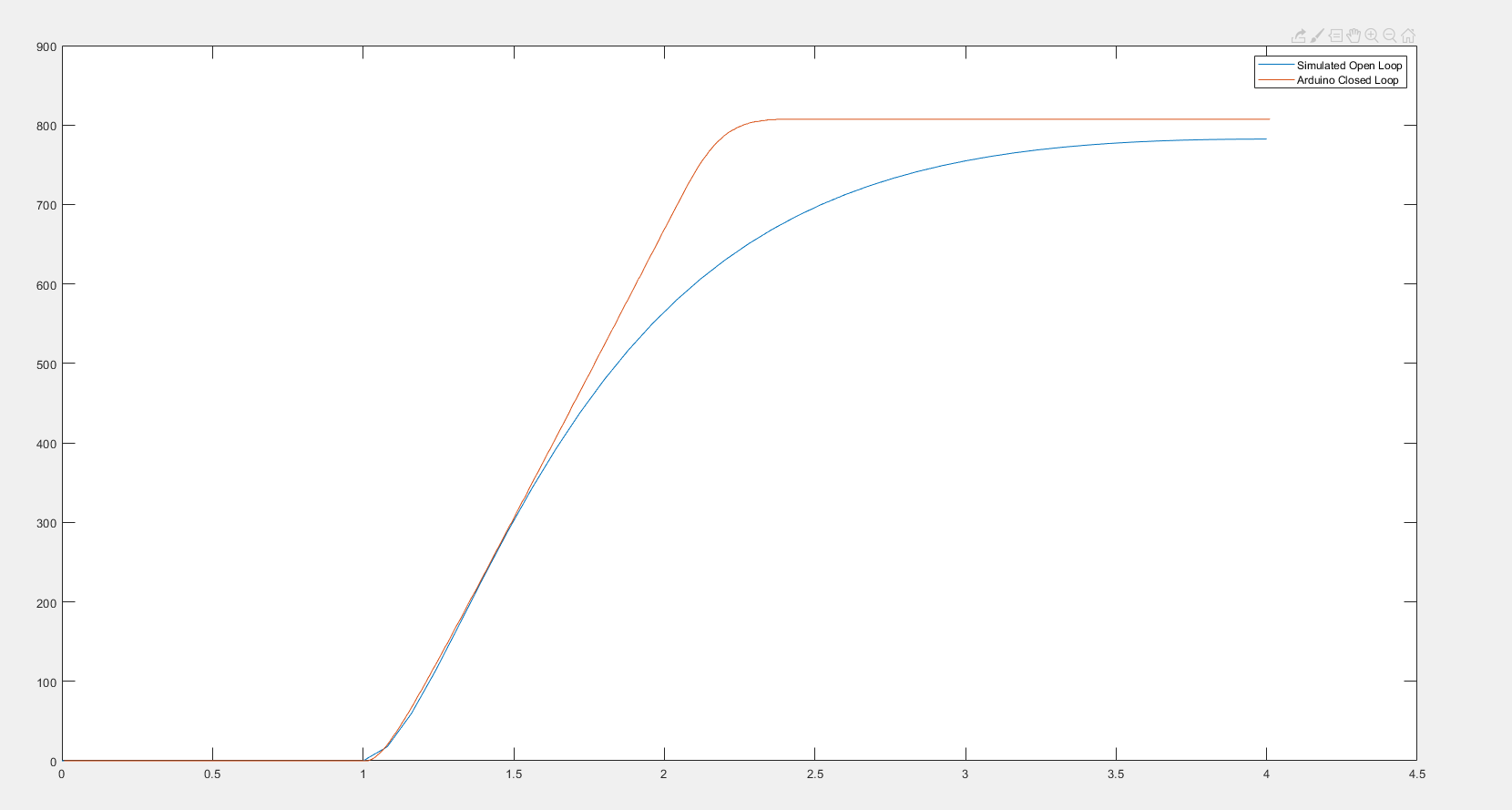
**Figure 3**: Graph comparing the simulated and actual motor step response.

After running the open loop response, a closed loop response was added to the block diagram as shown in Figure 4. Additionally, a PI controller was added to the block diagram in order to tune the model to the requested requirements.



**Figure 3**: Block diagram for simulated closed loop step response.

After the closed loop step response experiment, the block diagram was tuned to have a step response of 1 second and an overshoot of 12% or better. The simulated and actual motor closed loop response was plotted on the same graph as shown in Figure 5.



**Figure 5**: Graph comparing the simulated and actual motor step response closed loop system.