5-1-2022

Investigating Polynominal-Fit Extrapolation for Iterative Gain Updating in a Controller

Malachi Landis

Title: Investigating Polynomial-Fit Extrapolation for Iterative Gain Updating in a Controller

by

Name: Malachi Landis

An Honors Capstone

submitted in partial fulfillment of the requirements

for the Honors Certificate

to

The Honors College

of

The University of Alabama in Huntsville

Honors Capstone Director: Dr. Richard Tantaris

Program Director:

Malachi Landis	4/30/20
Student (signature)	Date
Richard N. Tantaris — Digitally signed by Richard N. Tantaris Date: 2022.05.01 12:24:11 -05'00'	5/1/22
Director (signature)	Date
D. Keith Hollingsworth — Digitally signed by D. Keith Hollingsworth Date: 2022.05.01 12:34:01 -05'00'	5/1/22
Department Chair (signature)	Date
William Wilkerson — Digitally signed by William Wilkerson Date: 2022.05.03 07:34:22 -05'00'	
Honors College Dean (signature)	Date

HONORS COLLEGE
THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Honors College
Frank Franz Hall
+1 (256) 824-6450 (voice)
+1 (256) 824-7339 (fax)
honors@uah.edu

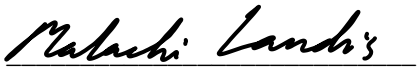
Honors Thesis Copyright Permission

This form must be signed by the student and submitted with the Capstone manuscript.

In presenting this thesis in partial fulfillment of the requirements for Honors Diploma or Certificate from The University of Alabama in Huntsville, I agree that the Library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by my advisor or, in his/her absence, by the Chair of the Department, Director of the Program, or the Dean of the Honors College. It is also understood that due recognition shall be given to me and to The University of Alabama in Huntsville in any scholarly use which may be made of any material in this thesis.

Malachi Landis
_____

Student Name (printed)


_____

Student Signature

4/30/20
_____

Date

Contents

Abstract

In this paper, a method is investigated that iteratively applies a polynomial fit to extrapolate future system outputs, makes an adjustment to a system gain, then evaluates the predicted output and the actual output at a later time whether such a change was beneficial to system performance. The goal of this method is to converge at some optimal gain during the normal operation of the system. This method is tested in MATLAB Simulink on an openly-available example system of a position-controlled motor. The gains arrived at in four separate tests are then compared to the theoretical optimal gain found in a separate analysis. The method is shown to provide poor results, and the reasons for these results are speculated upon. Possible changes to be made in future work is then discussed.

Control Systems and Tuning Gains

Control systems are critically important to modern technology. In short, they allow systems to produce a desired output in the face of disturbances to the system, oversimplified models, or unknowns about the system. Control systems typically achieve a desired output through some sort of feedback, where the current output of the system is compared to the desired output, and this error causes a change to the applied input to the system. Well designed controllers are able to use the error to reach the desired output while meeting certain design criteria, such as minimizing time to reach the desired output or power consumption.

One common controller is known as the PID, or Proportional-Integral-Derivative, controller. One reason for its commonality is its simplicity. Inside the PID controller, only three coefficients are used. The P gain is multiplied by the current error, the I gain is multiplied by the integral of the error over time, and the D gain is multiplied by the rate of change of the error. The output of the PID controller is the sum of these three terms. By tuning these three coefficients, many different systems can be controlled. While simple, it is also powerful and can achieve good results on many relatively simple systems.

Selecting the appropriate P, I, and D gains is not always trivial. If an accurate model of the system can be made, then selecting these gains may be simple. Control design methods like pole placement in the imaginary plane may be used. Not all systems are easily, or economically, modelled, and not all users of the PID controller have the skills or tools available to do this modeling. One reason for the PID controller's great success in industrial applications is its simplicity. A technician can implement a PID controller while knowing nothing about transfer functions or MATLAB, tools commonplace in controls engineering. Rather, a PID controller can be implemented in a system then tuned by hand. This is often done in relatively simple systems with good success.

Therefore, the next improvement to be made to the PID controller would be a system for adjusting the gains so as to improve performance over time. This system will be

Rotor Inertia	$3.2284 \cdot 10^{-6}\ kgm^2$
Viscous Damping	$3.5077 \cdot 10^{-6}\ Nms$
Electromotive Force Constant (K_e)	$0.0274\ V/rad/sec$
Motor Torque Constant (K_t)	$0.0274\ Nm/Amp$
Resistance	$4\ \Omega$
Inductance	$2.75 \cdot 10^{-6}\ H$

Table 1: Motor system parameters

agnostic to the details of the plant being controlled, just as the PID controller is agnostic to the plant generating an error upon which it is operating. This is therefore distinct from typical methods of adaptive control, where some model of the system is known and used to derive the methods of gain adjustment (Åström and Wittenmark 2013) A potential method for achieving this adjustment is described in the following sections. The shortcomings of the method are also discussed, as well as future work.

The example system used for testing this method comes from the University of Michigan Controls Tutorials for MATLAB and Simulink (Tilbury and Messner 1996). It is a simple motor controlled for position. The values of the physical constants are shown in Table 1. The Simulink diagram for the motor is shown in Fig. 1. All simulations were completed with sample times of 0.1ms with a unit step input at 0.1s and a total simulation time of 0.4s.

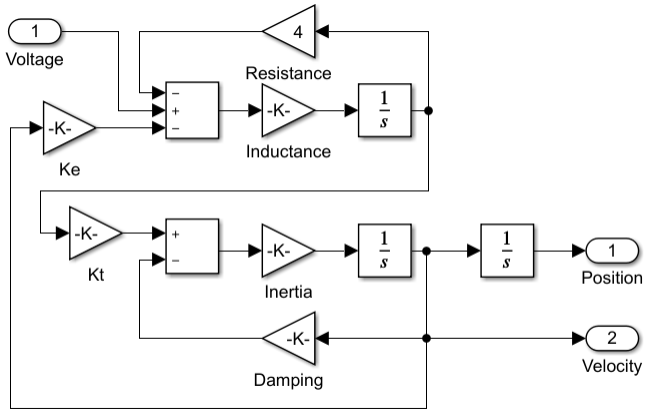


Figure 1: Simulink block diagram of motor

Concept of Operations: Predict, Adjust, Evaluate

As shown in the section title, there are three core parts to this method of gain adjustment: prediction, adjustment, and evaluation. The overall concept is this: at some time step t_n during the response y_n of the system to an input u_n, predict the input u_{n+m} and response y_{n+m} of the system after m time steps at t_{n+m}. At t_n, make some adjustment to the gains of the controller. Then, after m time steps have passed, evaluate whether the change to the gain made a positive or negative impact on the system. This evaluation is based on a score calculated using u_{n+m} and y_{n+m}. If the change was positive, make another change to the gains in the same direction, either increasing or decrease each gain, after p additional time steps. This process is repeated as long as the system is running. Qualitatively, the system attempts to predict some future performance then change the gains to see whether that actual future performance can be improved.

This method attempts to improve on a different strategy of making an adjustment to a gain, running the system for one entire cycle, then evaluating the gain adjustment based on data collected during the entire cycle. This strategy limits the rate of gain adjustment to once per cycle, which may be very slow.

Prediction

In this implementation, a polynomial of degree 5 was used to predict both the input to and output from the system 3 time steps from the current time. The MATLAB `polyfit` function was used to generate the polynomial and the `polyval` function to evaluate at the future time. Figure 2 shows the Simulink block PolyPredictor created to execute the MATLAB code. The polynomial fit prediction has two main limitations. First, 6 points are needed to construct the polynomial. These 6 points must be taken from a portion of the response without any changes in gain. Therefore, this limits the rate of gain iteration overall.

Second, an accurate prediction of the future can only be made for a small number of steps into the future. A prediction of 3 time steps ahead was found to be the maximum

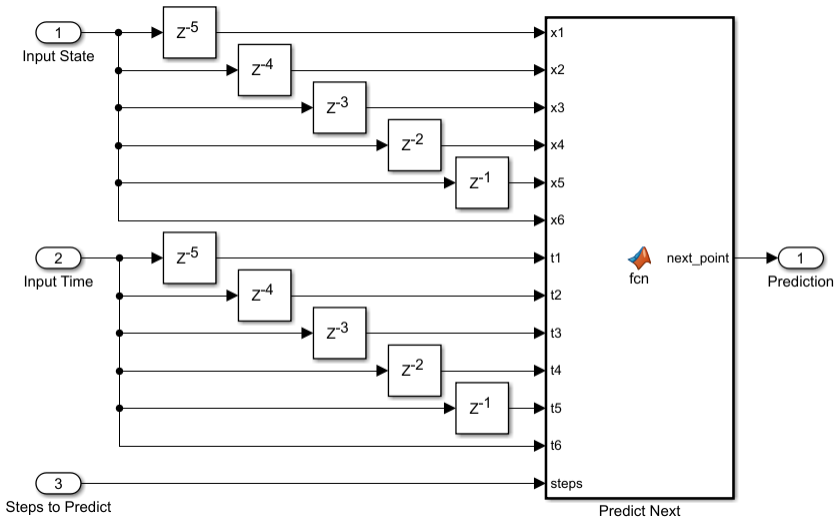


Figure 2: Simulink block diagram of PolyPredictor showing Delay blocks used to collect the last 6 data points

without noticeable discrepancies. Figure 3 shows the predicted output values compared to the actual output values when predicting 3, 4, and 5 steps ahead.

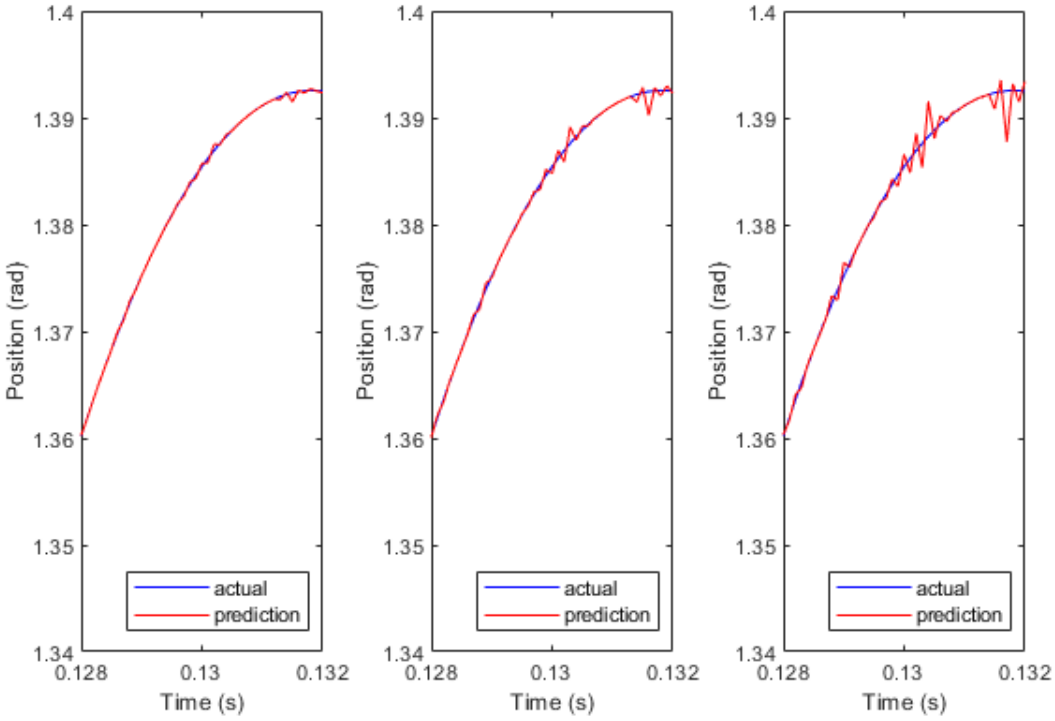


Figure 3: Comparison of 3, 4, and 5 step predictions with actual outputs

Other prediction techniques may yield more accurate predictions at greater steps into the future. AI and machine learning could be trained over time to develop a model of the system dynamics without explicitly encoding the dynamics, thereby allowing prediction farther into the future. Predicting farther into the future may be important because it would allow more time for the effects of gain adjustment to become apparent.

Adjustment

Once the future state of the system has been predicted, the gains can be adjusted. In the current method, only the proportional gain was adjusted. This adjustment occurred every 10 samples. At each adjustment period, the current gain was either incremented or decremented by the Gain Increment value. Then, in 3 time steps, the predicted and actual scores were checked and the either the adjustment was undone, the direction of iteration was reversed, or there was no change. The following simple algorithm was applied.

1. The gain is adjusted by the current gain increment.
2. In 3 time steps, the predicted and the actual score are compared.
    (a) If the absolute value of the difference between the predicted and actual score is less than a threshold, invert the iteration direction. This is done because there is not an appreciable difference between either gain. Therefore the next gain change will occur in the opposite direction.
    (b) If the difference between the predicted and actual score is positive, then continue running. This is because a lower score is considered better. If the previous change in gain produced a better result i.e. a lower score then it is assumed that the next change in the same direction will also produce a good result.
    (c) If the difference between the predicted score and the actual score is negative, then undo the previous gain change by incrementing two times in the opposite direction and inverting the iteration direction. A negative difference implies the score became greater than predicted, so the gain change had a negative effect.

It therefore needs to be reversed and two steps need to be taken as compensation.

3. This process is repeated every 10 samples.

The Simulink diagram shown in Fig. 4 shows both the increment and check blocks. A pulse generator outputs a 1 every 10 samples. Also shown are the two discrete-time integration blocks with a reset trigger linked to the increment pulse. This is used to calculate a score beginning at the moment the gain is updated.

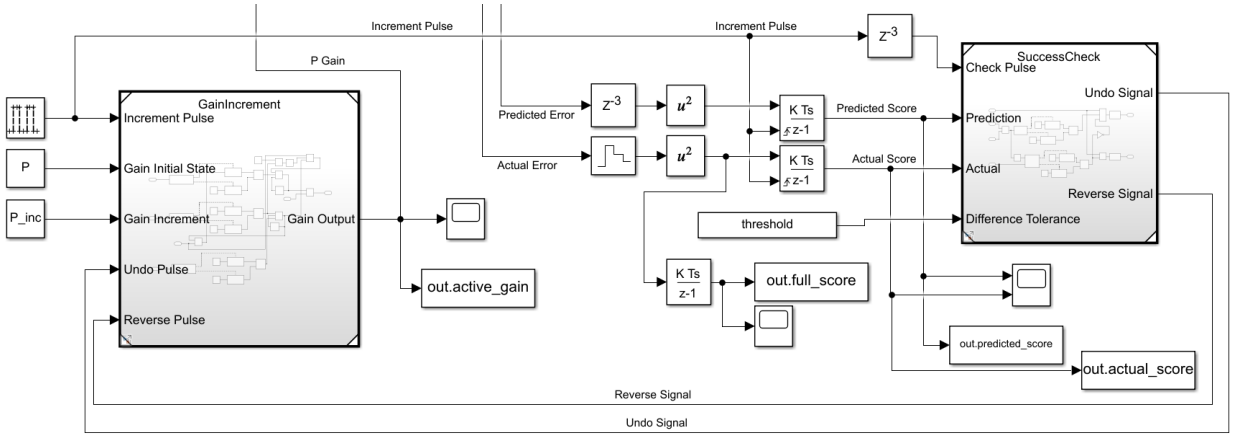


Figure 4: Simulink blocks used to iterate the gain and evaluate after 3 time steps

The CheckSuccess block is shown in Fig. 5. It uses If blocks and logic blocks to output the correct signal of do nothing, undo, or reverse.

Evaluation

Qualitatively, certain system outputs may be seen as better or worse by an engineer. This is not sufficient for the current method. An automatic, quantitative method of comparing outputs is needed to evaluate the results of different gains at rates of potentially hundreds of times per second. A score can be developed to achieve this goal. The score is defined by an algorithm and takes as input some characteristic of the system. By comparing scores generated by two sets of gains, the better set of gains can be determined.

In this implementation, the score is defined as the integral of the squared error of the system, as shown in Eq. 1. This integral is evaluated for both the prediction and the

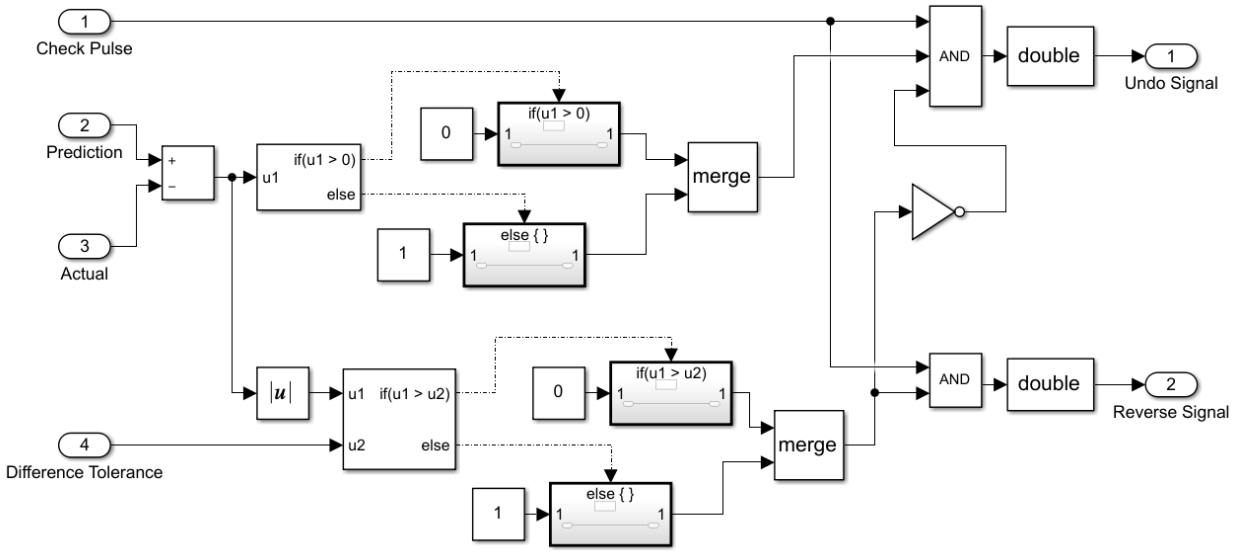


Figure 5: Simulink blocks used to evaluate the predicted and actual scores then change the iteration direction if necessary

actual result between time t_n and t_{n+m}. The lower the score, the better. To illustrate why this score was chosen, consider that the ideal response to a step input is a step output. The error between the input and output will always be zero in this case. Therefore, the integral of the squared error will also be zero. The error is squared so that both positive and negative error will cause the integral to increase.

$$S = \int_{t_n}^{t_{n+m}} e^2 \tag{1}$$

By evaluating this score between t_n and t_{n+m}, only the portion of the response that could be affected by the change in gain would be considered. This avoids any issues with poorly fit polynomials causing extreme error predictions during the first few time steps of the step input.

Example of Current Implementation

As mentioned in the Introduction, a simple motor position controller was used for testing this method. Due to the gain adjustment method being restricted to one gain, only the proportional gain was adjusted. The integral and derivative gains of the PID controller were set to zero. To determine the ideal proportional gain for the given system, a MATLAB code was created to simulate every proportional gain from 0.5 to 100 in 0.5 increments and calculate the score for each. The gain with the lowest score was defined as the optimal gain. Figure 6 shows the scores plotted against the gain that produced them. A plot of the performance of the system under the optimal gain of 33 is shown in Fig. 7. The score of this system evaluated over the entire duration is 0.009326.

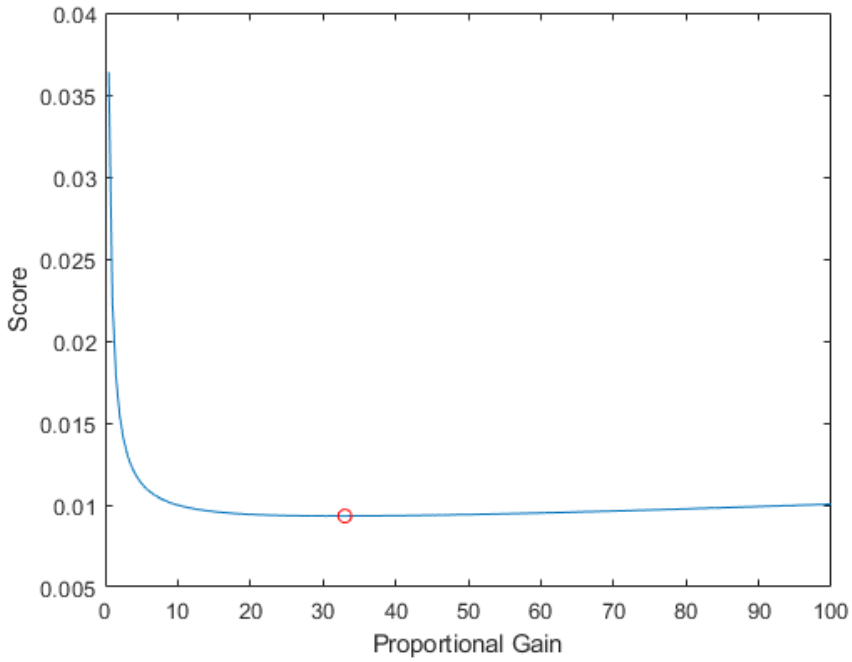


Figure 6: Score values at simulation end, lowest score at P = 33 shown in red circle

To evaluate the current method, multiple initial gains and gain increments were selected. If the method were to be successful, then all these tests ought to converge to the same gain. Table 2 shows simulations 1 through 4 and their parameters, as well as their final scores evaluated over the entire duration. This final score was compared to the the-

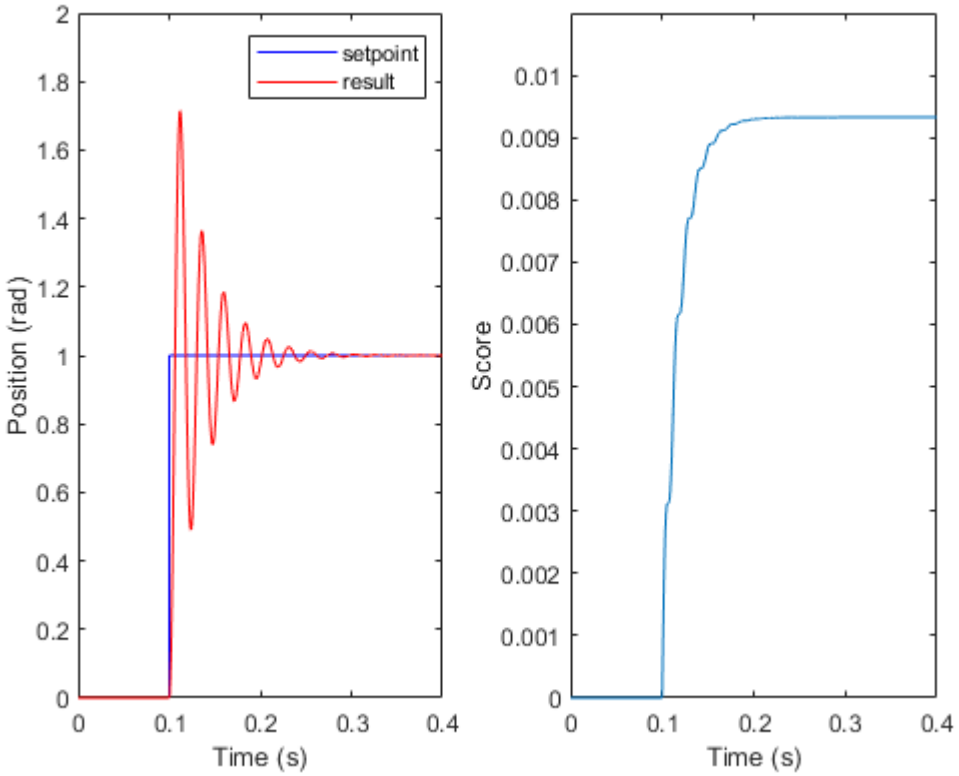


Figure 7: Response of motor with P = 33 (left) and plot of score over time (right)

oretical optimal gain's score found previously. A threshold value of $1 \cdot 10^{-11}$ was used throughout. Figures 8 through 11 show the plots of the system outputs for these simulations, as well as the gains over time.

As seen in Figs. 8 through 11, the adjustment mechanism does successfully change the gains throughout the simulation. It does not, however, seem to converge to the optimal gain of 33 that was determined previously. All of these tests resulted in a total score above that of the optimal gain. The initial gain did not seem to have an impact on the

Initial Gain	Gain Increment	Final Score	Percent above Optimal
10	0.1	0.009605	3.0
10	1	0.009817	5.3
50	0.1	0.011083	18.8
50	1	0.009568	2.6

Table 2: Parameters used to test the current method

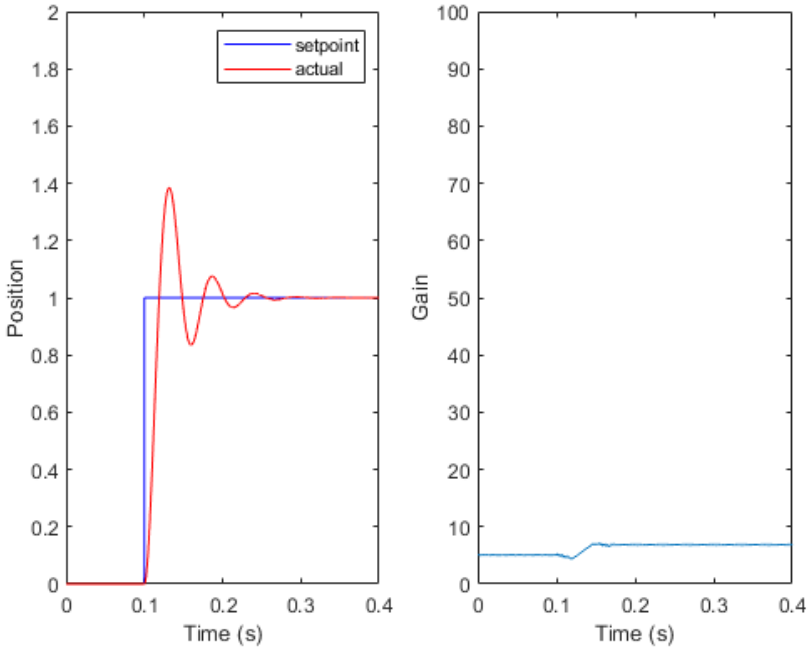


Figure 8: Initial Gain of 10, increment of 0.1

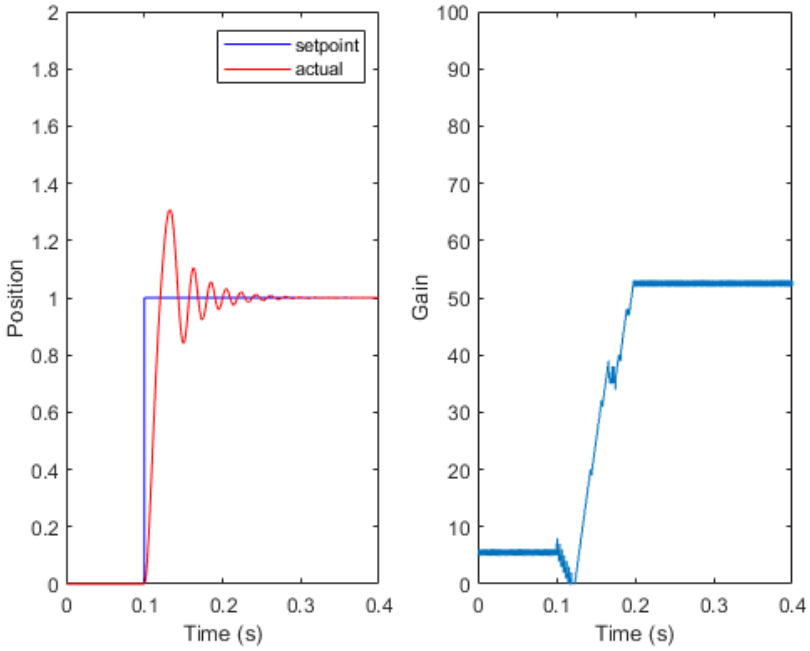


Figure 9: Initial Gain of 10, increment of 1

trajectory of the gain over time. The gain increment did have a strong impact. Overall, this method in its current implementation does not successfully change the gains over time

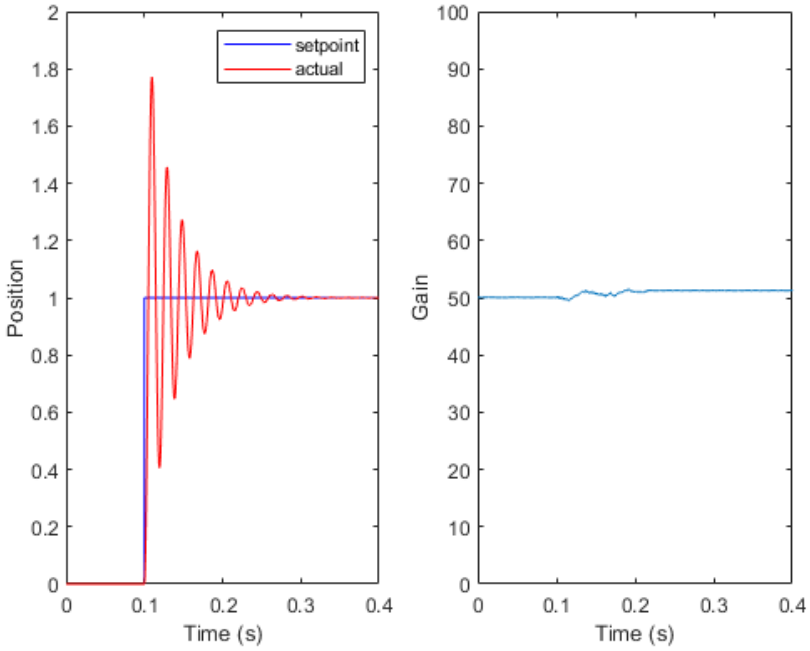


Figure 10: Initial Gain of 50, increment of 0.1

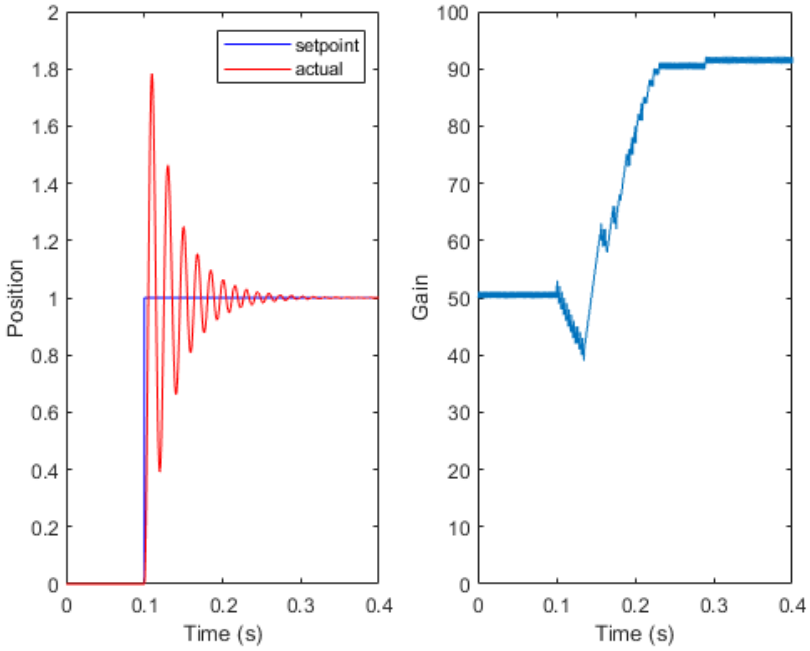


Figure 11: Initial Gain of 50, increment of 1

to reach any sort of optimum, and the potential reasons for this are discussed next.

Issues with Current Implementation

The goal of this method was to make gain adjustments rapidly during the operation of a system. This rapid adjustment leaves only 0.3ms for any change in gain to have an effect. As a result, differences between predicted and actual scores may be on the order of 10^{-11}. This leaves very little room for error in the prediction, which is almost certainly not giving predicted scores accurate to the same order of magnitude. The end result is therefore dictated mostly by the prediction algorithm and noise. This would not give useful results on any sort of real system with noisy sensors that would dominate any scoring outcome.

Choosing a correct prediction algorithm is a problem of trade-offs. Polynomial-fits of large degree may predict future outputs farther into the future, but they would also require more data points for a full fit. Low-degree polynomials could be constructed from few points, but could not successfully predict many steps into the future. Fitting of polynomials may also pose a challenge for implementing this method on real hardware in which the controller would have to calculate the fit at every iteration point. This could then further limit the number of samples per second and therefore this method's overall effectiveness.

The adjustment algorithm for changing gains does not allow for any sort of variable step size. This means that even if the algorithm itself could converge to an optimal gain in an infinite number of time steps, it may not achieve this in a very limited number of steps. A different adjustment algorithm may be able to make changes to the gains large enough to have noticeable effects on the output in only a few time steps, then make more refined changes to the gain as the optimal gain is converged upon. This algorithm is simply a first attempt at an adjustment mechanism, so it does not take into account the current error.

This method does not currently work on a full PID controller. As of yet, no algorithm has been developed to dictate how and when each of the three gains might be changed.

A better scoring algorithm must be developed. The integral of the squared error does not give any weight to oscillation, which is typically not a favored system output. Other scoring functions based on the rate of change of the error ought to be explored, as well as a way to combine multiple scores in some way. This may take the form of a weighted average of the scores.

Reference List

Åström, Karl J. and Björn Wittenmark (2013). *Adaptive Control Second Edition*. Dover Publications. ISBN: 9780486319148.

Tilbury, Dawm and Bill Messner (1996). *DC Motor Position: Simulink Modeling*. URL: https://ctms.engin.umich.edu/CTMS/index.php?example=MotorPosition&section=SimulinkModeling. (accessed: 04.29.2022).

Conclusions and Future Work

The method described above attempted to adjust the gains of a proportional gain controller during operation so as to converge on an optimal gain. It used an iterative algorithm that predicted a future score, made a change to the gain, then checked whether the score improved. In its current implementation, this method does not successfully converge an optimal gain. This is due to a variety of factors. The small step sizes result in making poor judgements on the quality of gain adjustments, judgements which are heavily effected by prediction accuracy and noise. The prediction algorithm does not predict far enough into the future to allow the effects of a gain change to become measurable. Scores generated by the integration of the squared error do not seem to correlate well with actual performance, especially with respect to oscillation.

If future work is done on this topic, several changes may be investigated. Some sort of AI or machine learning could be used to better predict future outputs. This would allow for more apparent differences between the actual and predicted scores, allowing for clearer, more robust decision making in the face of noise. A better adjustment algorithm should be developed that can handle all three gains of a PID controller, as well as make variable-size increments on these gains. This adjustment algorithm should also make its decisions using some combination of scores to better reflect truly optimal gains.

Appendix

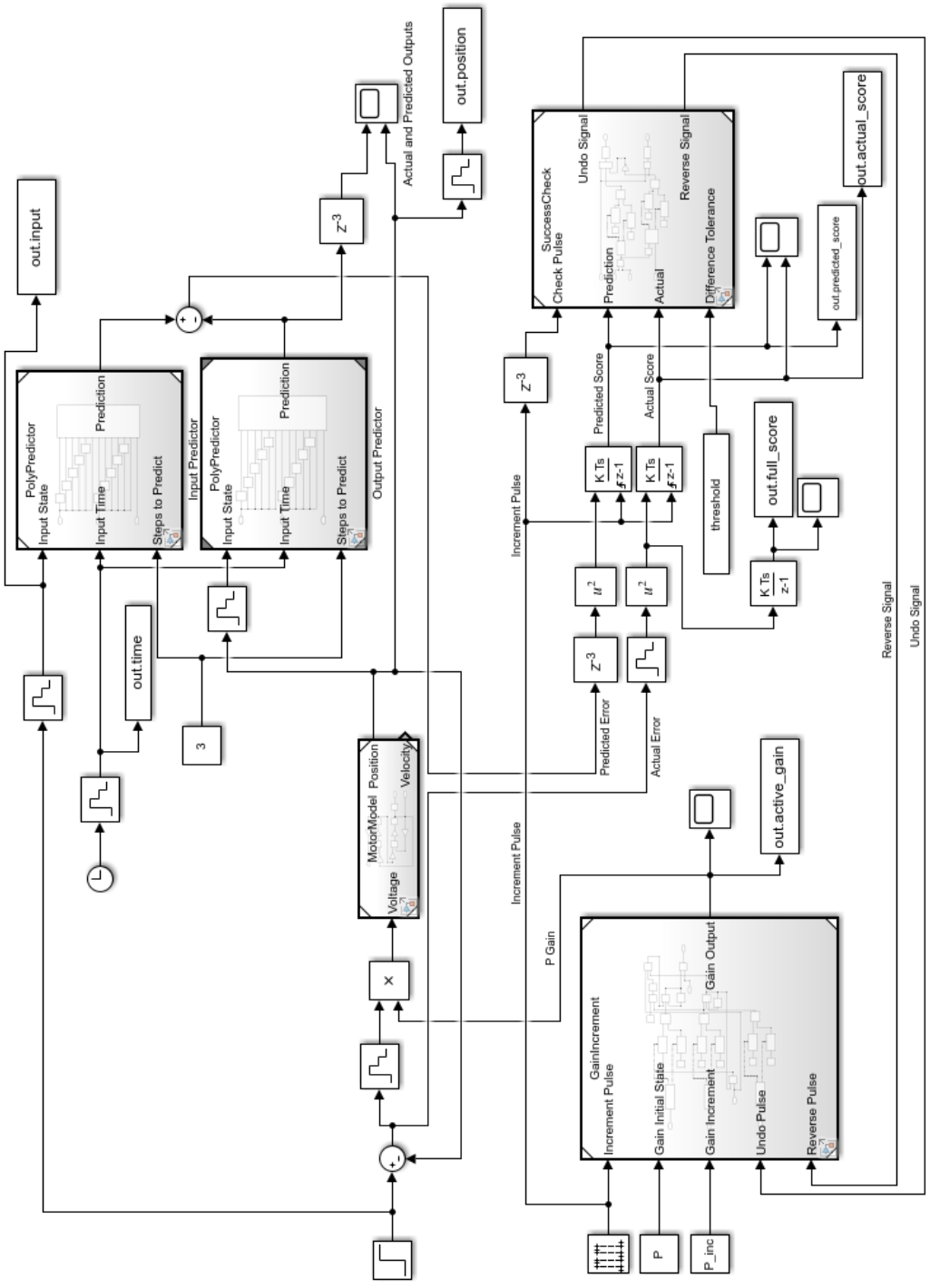


Figure 12: Full Simulink block diagram used for implementing this method