Wydział Elektryczny

Introduction to Control Engineering

Experiment 2: PID Controller for the Fan Heater

Author: Gökhan Koçmarli

Advisor: dr hab. inż. Paweł Dworak

Date: 11-04-2022

1. Introduction

This document is intended to be the second report of the laboratory homeworks. The objective of the homework is to understand how a real plant can be investigated, how to get output data into directly computer using I/Os and MATLAB, how to analyze the noisy output signal and construct a model, and how to tune a PID according to the model that will have been found.

2. Experiments

After connecting the necessary cables to record the data into MATLAB, we have created a Simulink model which has two inputs and two outputs. These two inputs are fan's motor power value and temperature value in terms of zero to ten. On the other hand, there are two outputs for motor speed and temperature values by sensors.

Within the pre-defined program, we could do experiments, and record the data of them as MAT files. All the data and graphs can be found on the link given end of the document.

One may see from the Table 1, the parameters (inputs) of the plant has no meanings in the value level. I mean, it is not important that "What does 4 actually represents?". In the control engineering perspective, the important thing is to have a change in parameter, and see the response of that given change.

One can see the oscillator graphs from the Figures 1, 2, 3, and 4.

	Motor Input	Temperature Input
Exp1	4	from 5 to 3
Exp2	4	from 6 to 5
Exp3	6	from 3 to 6
Exp4	6	from 6 to 5

Table 1: Experiments that we have done to find model of the plant.

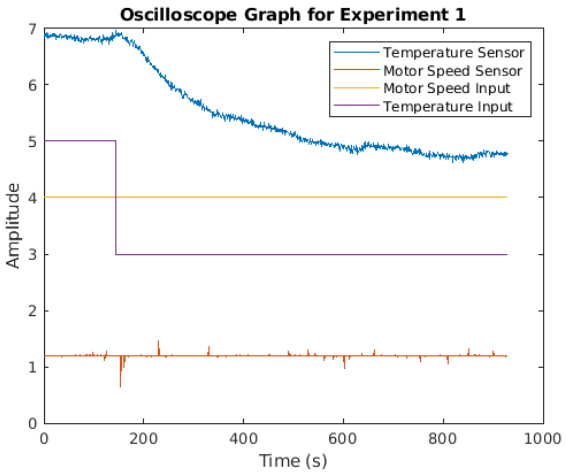

Figure 1: Oscillator Graph for the Experiment 1

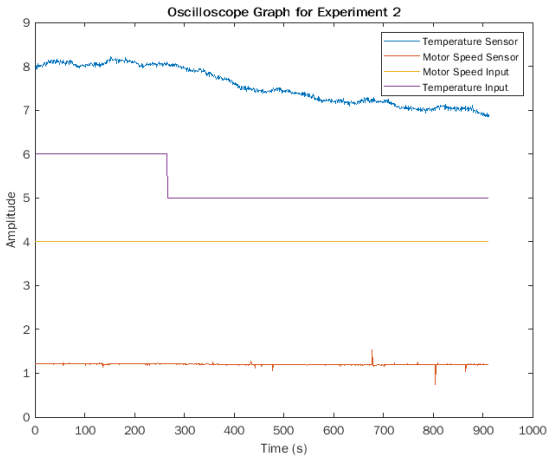

Figure 2: Oscillator Graph for the Experiment 2

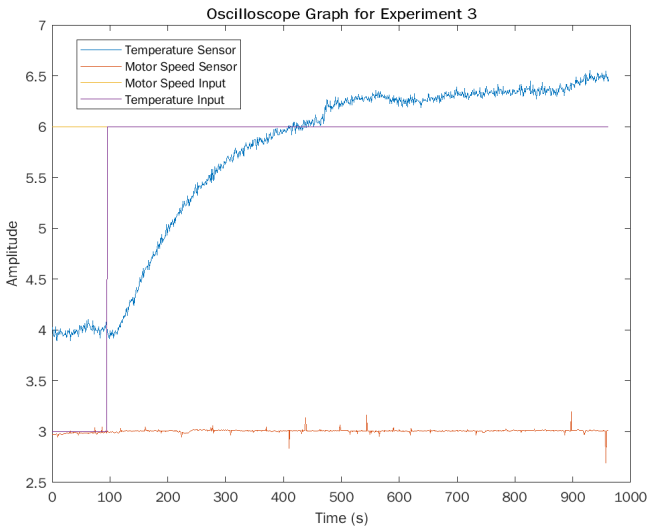

Figure 3: Oscillator Graph for the Experiment 3

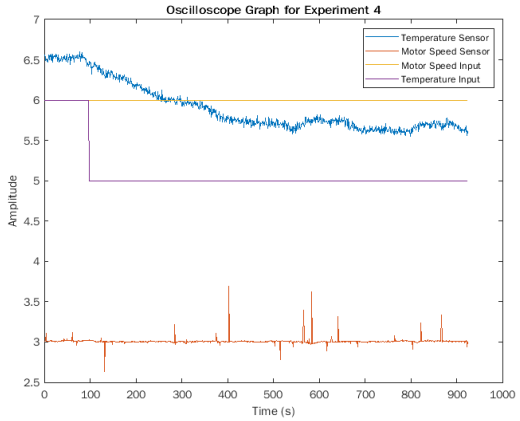

Figure 4: Oscillator Graph for the Experiment 4

On the next section, we're going to create different models for each experiment. In non-liner models, the professor suggests that, it is reasonable to have more than one model within its range of parameters.

3. Modelling with System Identification Tool

To model a step response of a plant, we can use the tool Identification Tool from System Identification Toolbox in MATLAB. In this document, we're going to use it since its algorithms much more effective comparing to our sense – and knowledge. In order to open the tool, one can call the command *systemIdentification* on MATLAB Command View.

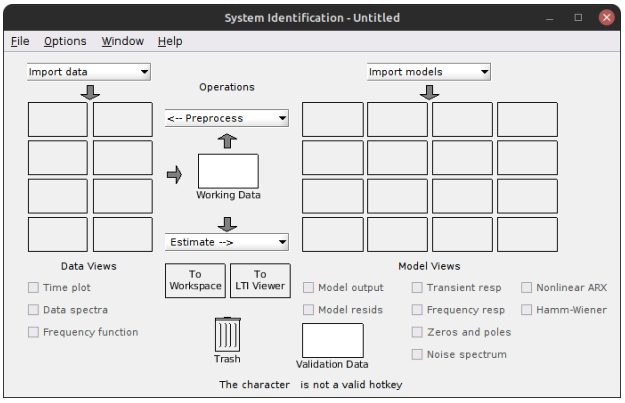

Figure 5: GUI of the System Identification App on MATLAB

We will introduce you to how to perform a modelling operation for our first experiment, however, one can do every step detailed if they want to achieve the models for other experiments.

1. Since our data is time-based, we will choose *Time Domain Data* from *Import Data* combobox.
2. In the recently shown window, it asks us *Workspace Variables* and *Data Information*. In the Workspace Variables section, the input variable is the data that we fed into the plant – for our case, it is the step function that goes from 5 to 3. On the other hand, the output variable is the sensors' data that we've collected. the Data Information section ask you details about your data. For example, data-name can be anything you want, however, sample time is crucial for our model. In the experiment, we have used

a delay of 0.2 seconds between the experiments.

3. One can verify the data with clicking *Time plot* checkbox on the window.
4. After the process of inputting the data, we are ready to process it. In the *Estimate ->* combobox, one should select *Process Models* in order to find a plant model.
5. The new window, that is opened, is a very valuable tool for us. Selecting the transfer function's details, and clicking the Estimate button, gives us the model that we want. Since we want best results, I'll try all the combinations of transfer function, and look into their achievement percentage.

3.1. Comparison of Models

I had done several estimation types for the first experiment's data. These can be seen in the unordered list below, and also the similarity of the model according to real data can be seen as a percentage.

- P1: One Pole – 87.88%
- P2: Two Pole – 81.86%
- P1Z: One Pole, One Zero – 88.68%
- P2Z: Two Pole, One Zero – 88.45%
- P1D: One Pole, One Delay – 87.32%
- P2D: Two Pole, One Delay – 87.49%
- P1I: One Pole, One Integrator – -116.9%
- P2I: Two Pole, One Integrator – -1572%
- P1DZ: One Pole, One Delay, One Zero – 88.26%
- P2DZ: Two Pole, One Delay, One Zero – 73.74%
- P1IZ: One Pole, One Integrator, One Zero – -2687%
- P2IZ: Two Pole, One Integrator, One Zero – 64.87%
- P1DI: One Pole, One Delay, One Integrator – -102.4%
- P2DI: Two Pole, One Delay, One Integrator – 75.64%
- P1DIZ: One Pole, One Delay, One Integrator, One Zero – -98.76%
- P2DIZ: Two Pole, One Delay, One Integrator, One Zero – 60.5%

One can see that the most three similarity is P1Z, P2Z, and P1DZ.

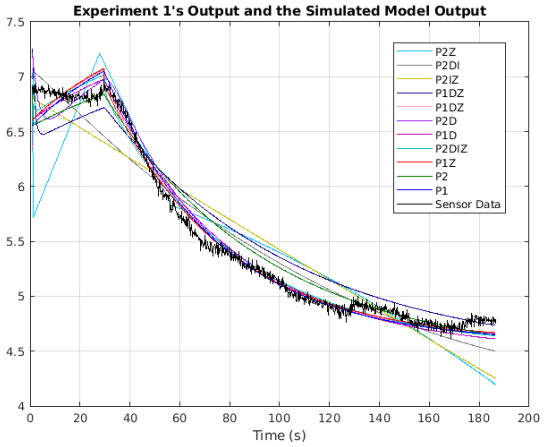

Figure 6: Estimated Model's Simulations vs Real Data for Experiment 1

Since we found an achievement nearly $\approx 88\%$, you might think that we're going to use the model of P1Z which is described in (1). One should note that the best model is not the model is very similar to our data, but the model which is reasonable for our plant. All of our experiment will tell you the best model is with zeros, maybe, however, having a zero will create a pike in the reverse direction of our desire in the first stages of the process. It is not reasonable.

$$G(s) = 1.5271 \cdot \frac{1 + 2.2184s}{1 + 47.506s} \tag{1}$$

Also, it is not enough if we found a model of P1 or P2. We need a delay coefficient due to division of zero error in the tuning operation of PIDs.

However, we can use P1 model for our controller, since it is one of the basic ones in the list. That model has an accuracy of 87.88% which is nearly 88% as the best model. You can see the mathematical model inside the Eq. (2). Notice that, it doesn't include any delay at all.

$$G(s) = \frac{1.5232}{1 + 45.891s} \tag{2}$$

4. Modelling by Trial & Error

As you can see the models that we have found for Experiment 1, none of the models are comfortable for us to create a controller for the plant. In such times, we can use the trail-and-error approach that is trying to guess the model by doing experiments on the signal response with Simulink.

Firstly, we add a transfer function with a transport delay into our Simulink design. One can give them any parameter they want. After, we have to check for the experimental output graph and our model graph for differences. One should consider the list above.

- If our model starts in different position, we should add a constant value.
- If our model ends in different position, we should change the gain parameter.
- If our model starts to change in different time, we should change the delay coefficient.
- If our model's slope is not looking like the real data, we should change the coefficient of s in the denominator.

5. Other Experiments

Let's make the same steps to the other experiments and find their best model. For each experiment, I'll introduce the best three model, and the graph for all the models and the real data. Also, you're going to see the best model in terms of mathematical expression with its accuracy rate.

After this section, we're going to discuss how should we combine the models, or should we combine them in the first hand. All the data, and the plots given on the website that you can find in the link end of the document.

This is a problem when we want to create a controller. Because nearly all methods includes delay constant on the denominator part in a division. We should add arbitrary delay if we have a non-delayed model.

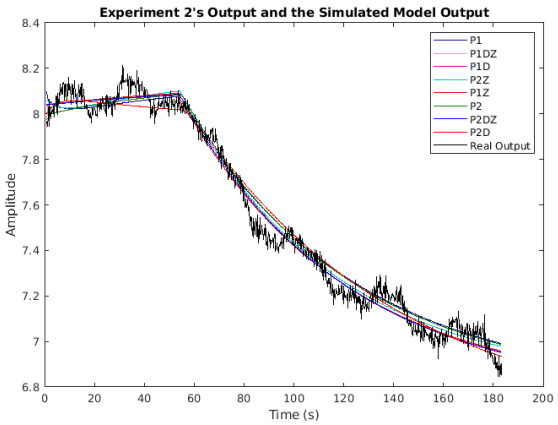

Figure 7: Estimated Model's Simulations vs Real Data for Experiment 2

For the second experiment, P1DZ, PIZ, and P1 are the best results we can find. The most accurate result is P1DZ, and the result is 87.32%. One can see the model in the (3). However, it is not an ideal model for our plant since it has zeros.

$$G(s) = 1.3542 \cdot \frac{1 + 1.0128s}{1 + 66.063} \cdot e^{-0.2s} \tag{3}$$

The model can be used has the accuracy of 87.24%, and one can see the model in Eq. (4). Due to lack of delay, it will make us some problems to create a controller.

$$G(s) = \frac{1.3536}{1 + 65.559} \tag{4}$$

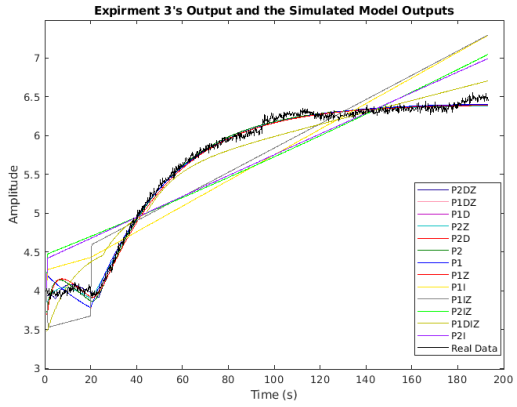

Figure 8: Estimated Model's Simulations vs Real Data for Experiment 3

For the third experiment, P2DZ, P2Z, and P2 are the best results we can find. The most accurate result is P2DZ, and the result is 94.11%. One can see the model in the (5).

$$G(s) = \frac{1.0687 \cdot (1 + 6.215s)}{(1 + 33.564s)(1 + 5.5575s)} \cdot e^{-3.5448s} \tag{5}$$

Again, the model doesn't respond to our needs. We should consider another model which is P1D with 92.69% accuracy. You can see the model in Eq. (6). We'll use it in our controller.

$$G(s) = \frac{1.0708}{1 + 34.536s} \cdot e^{-0.0124s} \tag{6}$$

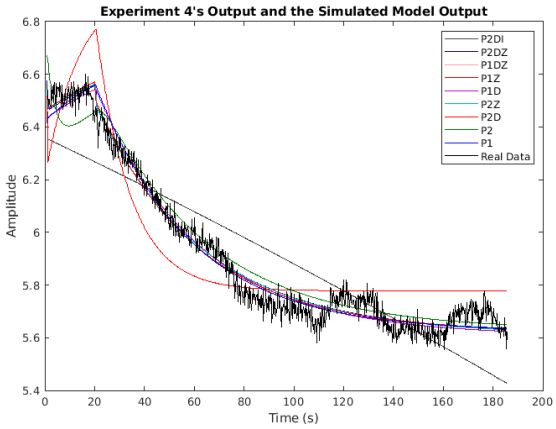

Figure 9: Estimated Model's Simulations vs Real Data for Experiment 4

For the fourth experiment, P1Z, P1DZ, and P2Z are the best results we can find. The most accurate result is P1Z, and the result is 83.28%. One can see the model in the (7).

$$G(s) = 1.1241 \cdot \frac{1 + 2.5994s}{1 + 40.751s} \tag{7}$$

Also, the model we're going to use will be P2D that has 82.53% accuracy as the best model. You can find its function in Eq. (8).

$$G(s) = \frac{1.1556}{(1 + 13.684s)(1 + 0.0042803)} \cdot e^{-0.1844} \tag{8}$$

5.1. Combining the Models

Since our experiments are in different settings, which means in different times and different initial conditions, we cannot merge them into one model with *merge()* function in MATLAB. As I asked our professor, dr. hab. Pawel Dworak, he told me that "We should create 4 different models, but you always have to specify (working point - for which motor speed (or pressure) and temperature they are valid). It's normal for nonlinear plants, and we have to deal with that in everyday practices". From the lines, we understand that creating a model for each condition is not-realistic, and will be very hard for us in this point of learning control engineering. Therefore, we're going to only choose one model from our set of models to implement a PID controller for it. The model will be from Experiment 3 because of it has includes delay coefficient which is necessary for creating a controller.

6. Controller Tuning

In this section, we're going to construct a PID controller for our model in the Eq. (8). Since a machine can only have one controller, we can only use one model for a controller. Finding parameters of PID controller can be done with some formulas which can be found in the book "Handbook of PI and PID Controller Tuning Rules" by Aidan O'Dwyer. We're also going to represent them in here.
It is a SOSPD process model. So that, we can use the rules of that teacher give us, that is published on 1935.

6.1. Calculation of Coefficients

So we can find the K_c.

$$K_c = \frac{0.95T_m}{k_0\tau_m} \tag{9}$$

$$K_c = \frac{0.95 \cdot 34.536}{1.0708 \cdot 0.0124}$$
$$K_c \approx 2470.96$$

Time to find T_i now.

$$T_i = 2.4 \cdot \tau_m \tag{10}$$

$$T_i = 2.4 \cdot 0.0124 = 0.02976$$

At last, let's find the K_d parameter of PID controller.

$$T_d = 0.4 \cdot \tau_m \tag{11}$$

$$T_d = 0.4 \cdot = 0.00496$$

6.1.1 Simulation of PID Controller

Our controller will have the formula in Eq. (12).

$$C(s) = 2470.96\left(1 + \frac{1}{0.02976s} + 0.00496s\right) \tag{12}$$

However, I found that the simulation is totally wrong due to the parameters of controller. If I use the *pidTuner* in MATLAB, it gives me the parameters of Eq. (13).

$$K_c = 1.338 \quad T_i = 0.07012 \quad T_d = 0 \tag{13}$$

You can find the results in Figures 10 and 11.

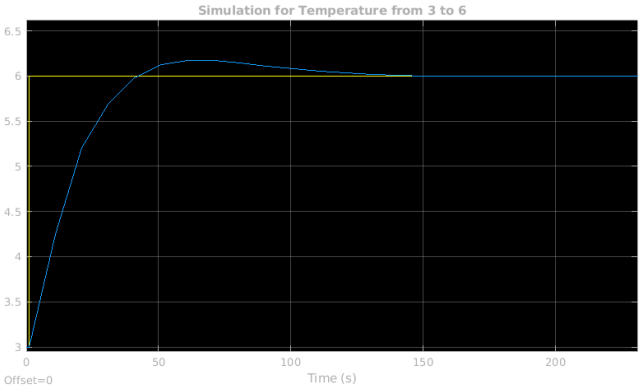

Figure 10: Temperature Input from 3 to 6

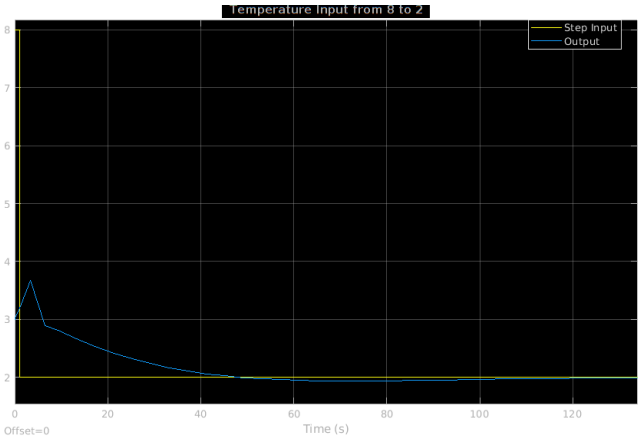

Figure 11: Temperature Input from 8 to 2

7. Conclusions

- Model is an approximation, don't make it too accurate.
- The most accurate model for a given dataset of a plant doesn't mean that is the best model for description of the plant. Before calculating our process models in any environment, we have to think about the way of working of our plant, and –not exactly, but at least– think about its model.
- Zeros in the models creates an opposite direction starting according to our desire, which is non-wanted property for our solution. Some plants may need it.
- Real-life plants mostly are non-linear objects. It means that creating a model, and a controller from that model, doesn't give us a controller for every condition in the world. In fact, it will be a controller of a minuscule of all conditions. In the industrial level, most of the time environment is the same, therefore they can use PID controllers most of the time.
- For creating a model for changing states of environment, one can use adaptive control techniques.
- System Identification Toolbox gives us the models that we want easily, and very accuratly.

The reader should note all the files and the schemes can be found online at `https://dar.vin/Intro2ContrEng-LabOne-HeaterPID`.

8. References

1. Laboratory Document of First Experiment, dr. hab. Pawel Dworak, 2022
2. Handbook of PI and PID Controller Tunning Rules, Aidan O'Dwyer, Imperial College Press, 2nd Edition