­­­To: President Frank O. Simpson

From: Team 01

RE: Data Analysis

Date: 7.30.19

**Introduction**

The client, First-Order Systems (FOS), were performing quality assurance tests on the different thermocouple designs, so they can provide precise temperature control to their client. FOS requested a MATLAB analysis algorithm that outputs parameter identification on given time histories. The client requires an analysis of the models, based on the dataset provided and graphics to visually display the data. An analysis of the margin of error must also be completed, to show the accuracy of the algorithm approach. The key criteria are the algorithm must be fully automated and determines the model parameters in an efficient manner. Another criterion when evaluating each thermocouple model is to generate a low standard deviation for each parameter amongst the 10 heating and 10 cooling experiments conducted. The constraints are contained within the data, based on the created algorithm’s consistency of accurate outputs for parameter identification. Another constraint was the time allotted to produce a functioning algorithm.

This algorithm takes an input of the data and an indication of whether it is heating or cooling thermocouple data. It then takes this data, creates a smoothed model of the data using a 4 point moving average, then calculates the parameters yL (the lowest sustained temperature before heating or after cooling), yH (the highest sustained temperature before cooling or after heating), ts (the time at which the temperature departs from its initial yL (for heating) or yH (for cooling)) and tau (time that is at the midpoint between the end of yL/yH and beginning of yH/yL respectively) for the thermocouple data.

The two biggest improvements came from improving the method of calculating yL/yH. It was necessary to improve both yL and yH as they were both used to calculate tau. yL and yH were initially calculated by finding the minimum and maximum of the data respectively. This method was no longer effective with the data given in M4 due to the increased amount of noise in the file. Thus, yL was determined by taking the mean of the temperature values from the first-time value until ts.

The second improvement involved yH it was determined by taking the mean of the temperature values from 20 points past ts to the end of the data. This method was implemented for the heating curve, and the calculations for yH and yL reverse for the cooling data. SSEmod for the cooling using the previous method was 26.5 (see table 1) compared to an SSEmod of .05 for cooling in the new parameter function.

**Procedure**

First, the data is imported, and the time and temperature data are separated into two variables, time and temp. Then, a moving mean of the data is calculated as well as a calculation of the slope between points 10 indexes apart in the temperature data. If the input to the function indicates a heating curve (indicated by a 1 for heating, 2 for cooling), ts is set to equal to the time that the maximum/minimum slope (for heating/cooling respectively) occurs. The index of that time is determined using the find() function and the variable x is set to the index of ts within the time vector. yL is then calculated by taking the mean of the temp vector from the beginning of the data set until ts for heating, or by taking the mean of the temp vector from 20 points past ts to the end of the data set for a cooling data set. yH is then calculated by taking the mean of the temp vector from the beginning of the data set until ts for cooling, or by taking the mean of the temp vector from 20 points past ts to the end of the data set for a heating data set. yTau is calculated based on its respective formula for heating or cooling. Bounds for yTau are then calculated by adding .15F to yTau for the upper bound and subtracting .15F for the lower bound of yTau. Tau for heating is then determined by taking the midpoint of the bounds and subtracting ts, and the tau for cooling is determined by subtracting ts from the lower bound of the yTau value.

**Results**

According to the calculated parameters in table 1, our algorithm for Milestone 2 was fairly accurate when calculating the heating data sets, while the tau values for both cooling data set are quite off from what we expect, so our code needs to get improved on analyzing the cooling data. And also, since the SSEmod for noisy cooling is too large, we also needed to check our SSEmod function. The SSEmod was particularly bad for the noisy cooling data compared to the other data sets, with an SSEmod of 26.5 compared to an SSEmod of .029 for the noisy heating data (see Table 1).

In Milestone 3, the parameter function increased in accuracy dramatically for the cooling data by utilizing the correct formula for yTau for cooling. The general form of equation is price = -6.4113 \* log(tau) + 2.9256. This resulted in the tau values becoming representative of the data for the cooling data. In this milestone, the SSEmod for cooling decreased from 26.5 to 5.0 from Milestone 2 to Milestone 3 (see Tables 1 and 2).

In Milestone 4 according to the calculated parameters in Table 3, our algorithm parameters were very close to the given parameters in the documents. Comparing Table 3 to the given parameters, our ts, yl, yh were only decimal points away and the tau was spot on. SSEmod for the cooling using the previous method was 5.04 (see table 2) compared to an SSEmod of .01 for cooling in the new parameter function. This function is overall most equipped to handle noisy data due to its improved ability to account for outliers in the data that could have affected the calculation of yH and yL.

**Conclusion**

The error can be characterized through the usage of the SSEmod and Standard Deviation calculations. The SSEmod improved throughout the Milestones, but has stayed relatively low, with the exception of the noisy cooling data in M2, reference Table 1. The SSEmod uses the data initial and calculated data sets, as well as the tau to find the margin of error between all of those.

The FOS thermocouples all accomplished their roles. With FOS-1 being the most efficient, and most expensive, to FOS-5 being the least efficient and least expensive. According to Figure 1 and Table 2, it shows the tau vs price graph for each test and Standard Deviation respectively. The general form of equation is price = -6.4113 \* log(tau) + 2.9256 and the R^2 value of 0.9578 which meant the accuracy of our equation was very accurate referenced from figure 1. The standard deviation is smallest for FOS-1(0.0273) and largest for FOS-5(0.3157), this indicates how tight the tau is. Following this data, we can determine that FOS-2 is the most cost efficient. The closeness of the data between FOS-1 and FOS-2 is almost negligible and the price difference is considerable therefore FOS-2 is the best design choice.

**References**

Abdulaziz Gheit. (n.d.). Retrieved from <https://www.mathworks.com/matlabcentral/answers/450562-what-is-the-best-way-to-smooth-and-compute-the-derivatives-of-noisy-data>

Dictionary.com. (n.d.). Retrieved from https://www.dictionary.com/

Eddens, S., & Shure, L. (2001). Matrix Indexing in MATLAB. Retrieved July 15, 2019, from <https://www.mathworks.com/company/newsletters/articles/matrix-indexing-in-matlab.html>

Introduction to First-Order Systems. (2014, March 28). Retrieved from <http://engineering.electrical-equipment.org/panel-building/introduction-to-first-order-systems.html>

Finch, Tony. Incremental Calculation of Weighted Mean and Variance. University of Cambridge, Feb. 2009.

‌Table 1: Results from M2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Clean Heating | Noisy Heating | Clean Cooling | Noisy Cooling |
| ts (s) | 1.5 | 1.53 | 1.5 | 1.59 |
| yL (°C) | 0 | -2.99 | 9.37 | -3.86 |
| yH (°C) | 100 | 101.34 | 100 | 102.9 |
| tau (s) | 3.01 | 1.68 | 8.19 | 3.86 |
| SSEmod | 0 | 0.029 | 0 | 26.55 |

Table 2: Results from M3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | FOS1 | FOS2 | FOS3 | FOS4 | FOS5 |
| tau (s) | 0.1556 | 0.3655 | 0.9460 | 1.1013 | 1.4928 |
| std dev | 0.0273 | 0.1104 | 0.0852 | 0.1277 | 0.3157 |
| SSEmod | 4.8963 | 5.0470 | 4.1290 | 4.3979 | 4.8644 |

Table 3: Results from M4

|  |  |  |
| --- | --- | --- |
|  | heating | cooling |
| ts | 15.7 | 7.4 |
| yl | -4.58 | 0.51 |
| yh | -3.13 | 0.88 |
| tau | 0.4 | 0.4 |
| SSEmod | 0.0573 | 0.0108 |

Figure 1: M3 Linear Regression

A close up of a map

Description automatically generated

The general form of equation is price = -6.4113 \* log(tau) + 2.9256

With R^2 value of 0.9578

Figure 2: result of M4

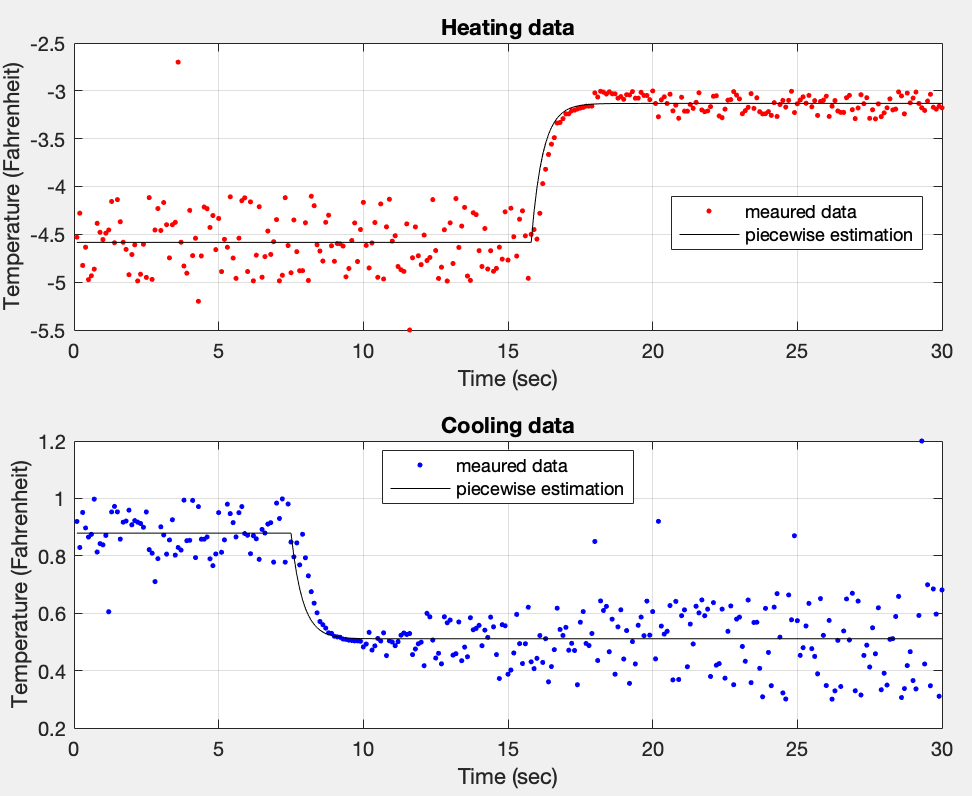
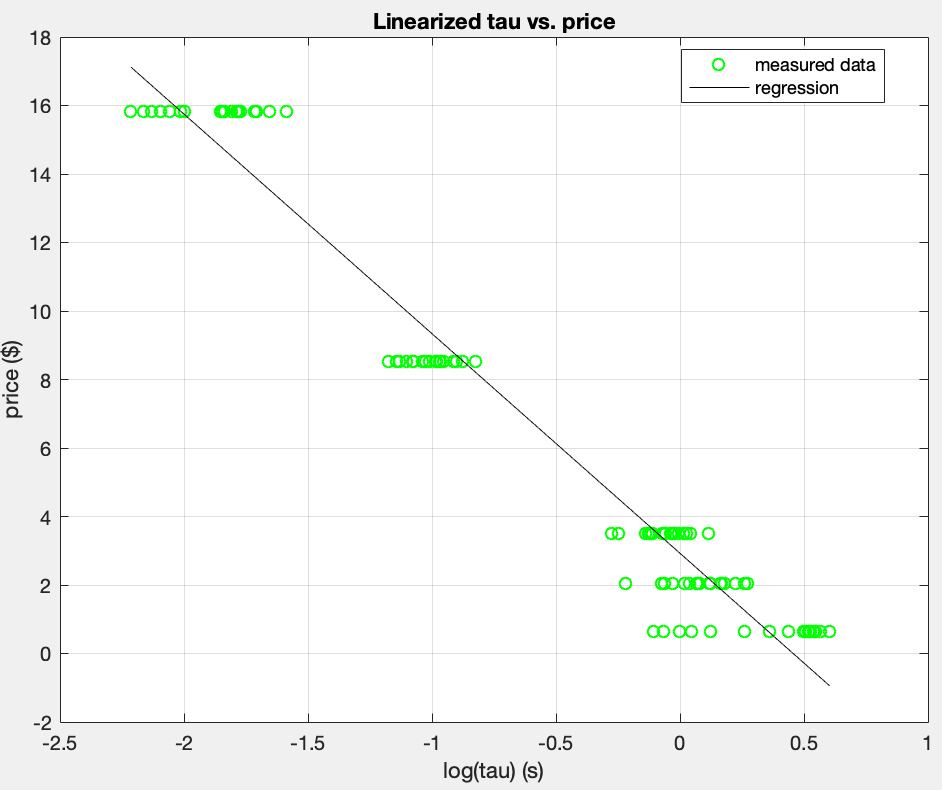


Figure 3: Linearized tau vs. price



The linearized form of equation is price = -6.4113 \* log(tau) + 2.9256