FOS Project
Final Milestone: Algorithm Refinement

Spring 2018

Team Member Names:	Kirthi Shankar Sivamani, Oliver Balicanta, Godfred Mantey, Spencer Dorsch
Purdue Logins:	gmantey, sdorsch, obalican, ksivaman

A. In your own words, summarize the feedback you received on project milestone M3 that could lead to improvements in your work.

Of the two feedback comments we received, the main one is about our algorithm and how four of our standard deviation values are a bit high. In general we should aim for the standard deviations values to be less than 0.035, thus improving the accuracy of our algorithm in calculating the parameter values. The other comment was on our plot for the price vs. tau data having a large range on the y-axis, making it harder to read the data.

B. Based on your feedback, what do you need to do to improve your parameter identification approaches? (Do not just reword your response to Part A. Do consider how you will incorporate your feedback into your work.)

Firstly, we will use the axis command to set the limits for the x and y axes so that the axis is shortened and the data can be more easily read. We will also brainstorm and come up with a different approach for calculating tau because our current values of tau are yielding standard deviation values way out of range. We will look at some of the ideas for tau we wrote down for milestone 2 and we will also try to come up with newer ideas in our next team meeting.

Part 1: Refinements Preview

Consult the M4 memo from FOS, Inc. for the details concerning your task. Respond to each of the prompts below in the space provided. Your goal is to introduce ***two refinements*** to your original algorithm, and these refinements must improve your solution to the FOS parameter identification problem. ***Read the rest of this document carefully*** ***before you begin your work on this milestone***.

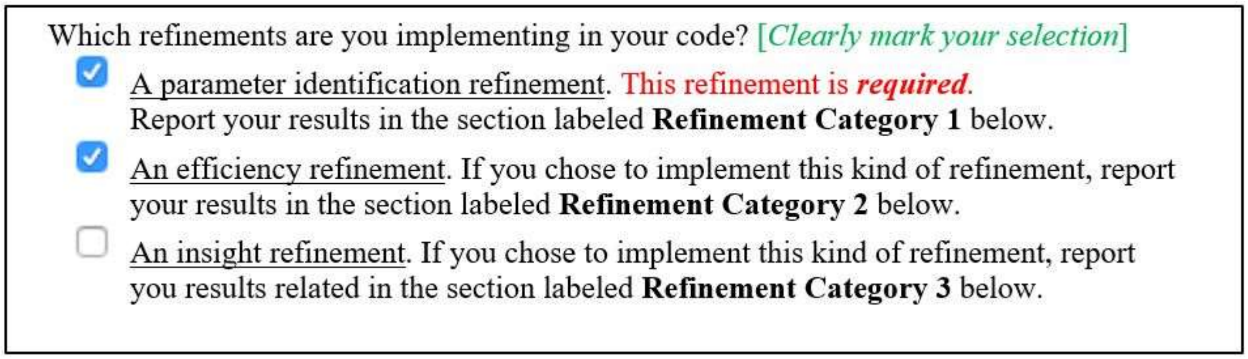
Definition of "refinement"

In this milestone, a refinement will fall into one of the following categories:

- **Refinement Category 1: Parameter Identification:** an improvement that changes the way you are doing parameter identification, and that improves your parameter identification results.
- **Refinement Category 2: Algorithm Efficiency:** an improvement that improves the efficiency of your code by (for example) removing un-needed looping structures, streamlining data handling, or otherwise reducing the execution time of your code.

- **Refinement Category 3: Algorithm Insight:** an improvement that involves analysis of your code and its limitations. For example, if you use any kind of thresholding in your code, you could determine the sensitivity of the solution to changes in that threshold parameter, and report how those changes affect your parameter identification and/or regression results.

In this milestone, you are ***REQUIRED*** to implement the parameter identification refinement (**Refinement Category 1)**. You must *also implement one of the other refinements*. You are therefore required to implement ***two*** total refinements.



Which refinements are you implementing in your code? *[Clearly mark your selection]*

[✓] A parameter identification refinement. This refinement is ***required***.
Report your results in the section labeled **Refinement Category 1** below.

[✓] An efficiency refinement. If you chose to implement this kind of refinement, report your results in the section labeled **Refinement Category 2** below.

☐ An insight refinement. If you chose to implement this kind of refinement, report you results related in the section labeled **Refinement Category 3** below.

● Refinements are given below
● Adjustment based on final values

Briefly describe, in words (not code), the nature of the refinements you will implement in your MATLAB code. Provide a brief, but thoughtful, description of your refinement, *using evidence-based rationales for why the refinement is necessary and should improve your solution*.

Refinement 1. Category 1: Parameter(s) Targeted: time constant (tau), low temperature (yl) for cooling, and high temperature (yh) for heating.

Description

The first two parameters we looked to improve for this Milestone was yl for cooling and yh for heating. We calculate both of these in the same manner for M3, by taking the mean temperature of the last 10 percent of temperature data. In our improvement for these parameters, we changed the 10 percent mean to a 2 percent mean. This means that for our Milestone 4 algorithm, we take the mean of the last 2 percent data for calculating yl in cooling data and yh in heating data.

The next parameter we improved for the M4 algorithm is time constant (tau). For Milestone 3, we calculated the temperature at tau using the formula mentioned on page 3 of the Milestone 1 project introduction pdf. We then calculated the difference of the temperature at each index of temperature vector to that of temperature at tau. We took the index at which this difference was minimum. Tau was then equal to temperature at this index minus start time. In our new method for identification of time constant, the first step is the same in which we calculate temperature at tau using the same formula from the M1 document mentioned above. We then find the number of points in the data set which have a temperature greater than temperature at

tau (for heating) or less than temperature at tau (for cooling). We find this number using the length and find functions. We then subtract this number from the number of data points to get the index of time at which temperature has reached 63.2% of its final constant value. We then subtract the start time from the time at this index to get time constant.
Rationale for Refinement YL for cooling and YH for heating: For the yh and yl parameters, we decided to plot all time history data sets on excel and visually analyze the graph to find out where the erroneous values for yh and yl are . We noticed that for FOS-4 and FOS-5, the temperature did not reach a steady state until the last 4-5% of data. Thus, by taking a mean of the last 10% of data for the calculation of the respective parameters, we were taking the mean of a non-constant temperature set, thus resulting in incorrect values for yl for cooling and yh for heating. Therefore, we decided to take the mean only of the last 2% of the data set so that we take temperature values only after steady state has been attained. Time constant (Tau) Our method for calculating tau for Milestone 4 is better than that of Milestone 3 because our Milestone 3 algorithm for tau was not robust enough to noise. For many data, the difference we were looking to minimize was found in an outlying index value for which temperature was extremely close to temperature at tau. However, our approach for finding tau for Milestone 4 is unaffected by such outliers. This improvement from M3 to M4 can be seen by the lower SSE mod values for the Milestone 4 algorithm for all time history data sets (Refer to tables 1, 2 and 3 to see SSE_{mod} values for heating data, cooling data and data for first order systems respectively).

Refinement 2. Category: __Time Efficiency ___
Description This improvement targets the run-time of the code. Our algorithm for Milestone 3 works with multiple for loops having the index vector ranging from 1 to 10240 (number of temperature values in one data set). This results in very slow processing of the processor and we often have to wait for more than 30-45 seconds to obtain the plots and the output on the command window. In improvement of our code, we get rid of 3 of the for loops from our code from Milestone 3 and only have 2 for loops remaining. We have replaced two of the for loops with while loops and 1 for loop has been vectorized.
Rationale for Refinement When we use for loops to check any condition, the algorithm checks the entire data set irrespective of whether the required value is found well before the end of the data is reached. This happens because for loops are counter controlled processes. Most of the conditions we required for the algorithm were event controlled. Thus when we changed the for loop to a while loop, we could break the loop immediately after the required value is found. This approach saved us a lot of execution time. Average value for time constant is approximately 1.5 for the heating and cooling data sets as can be seen from tables 1 and 2 in this document below. In a for loop, we were checking values till t = 10 seconds instead of breaking at t = ts. This means we were checking the vector for much more than required. The use of while loops in M4 improved the execution time by 29.84 seconds (Refer to table 5 below).

Part 2: Refinements

Resave all M3 files as **Project_M4Exec_*sss_tt*.m**, **Project_M4Algorithm_*sss_tt*.m**, and **Project_M4Regression_*sss_tt*.m** before starting to make refinements.

Refinement Category 1: Parameter Identification (***Required***)

Making all necessary refinements to your M3 algorithm in your **Project_M4Algorithm_sss_tt.m** file. Refinements must be clearly commented in your code with the text "Category 1" AND an adequate description. Then evaluate the improvement in your refined parameter identification algorithm. Use the clean and noisy calibration data from M2 and compare the parameters identified from the calibration data using the algorithm you submitted as your solution for M3 and your refined algorithm for M4. Report your results in Tables 1 and 2. Take care with units and decimal places when presenting results.

Table 1. Algorithm performance comparison to HEATING calibration parameters

	HEATING					
Parameter	M2Calibration		M3Algorithm1		M4Algorithm2	
	Clean	Noisy	Clean	Noisy	Clean	Noisy
y_L (°C)	0.00	-0.64	0.00	-0.76	0.00	-0.76
y_H (°C)	100.00	99.36	100.00	98.59	100.00	98.75
t_s (sec)	1.50	1.50	1.50	1.52	1.50	1.52
τ (sec)	0.31	1.65	0.31	1.58	0.31	1.60
SSE_{mod}	0.0000013	0.67	0.00094	0.89	0.00020	0.84
Note: Heating Actual Clean SSEmod should be 0.00 degC^2. Heating Actual Noisy SSEmod should be 0.67 degC^2.						

Table 2. Algorithm performance comparison to COOLING calibration parameters

	COOLING					
Parameter	M2Calibration		M3Algorithm1		M4Algorithm2	
	Clean	Noisy	Clean	Noisy	Clean	Noisy
y_H (°C)	100.00	98.81	98.03	98.19	100.00	98.98
y_L (°C)	0.94	-0.21	1.25	-0.80	1.04	-0.74
t_s (sec)	1.50	1.50	1.5352	1.5381	1.50	1.51
τ (sec)	1.82	1.12	1.78	1.10	1.79	1.11
SSE_{mod}	0.51	1.30	1.02	1.33	0.31	1.03
Note: Cooling Actual Clean SSEmod should be 0.51 degC^2. Cooling Actual Noisy SSEmod should be 1.30 degC^2.						

Using your M4 algorithm, analyze the 100 time histories provided by FOS, Inc. to identify the four relevant first-order system parameters (y_L, y_H, t_s, and τ) from each time history. In Table 3, copy your results from M3 for the M3 algorithm, and record your results for your M4 algorithm. Take care with units and decimal places when presenting results.

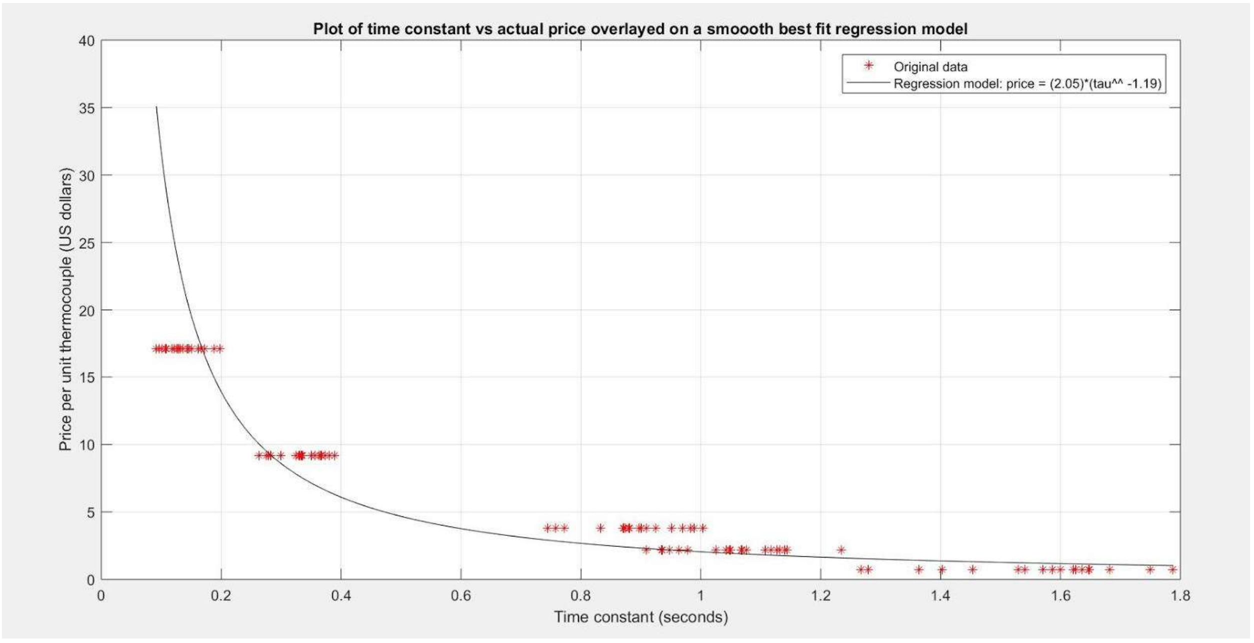
Table 3. Algorithm performance comparison for FOS designs

Model Number	**M3 Algorithm**			**M4 Algorithm**		
	τ Characteristics		**Mean SSE_{mod} ($degF^2$)**	**τ Characteristics**		**Mean SSE_{mod} ($degF^2$)**
	Mean (sec)	**Standard Deviation (sec)**		**Mean (sec)**	**Standard Deviation (sec)**	
FOS-1	0.14	0.030	2.39	0.14	0.028	0.34
FOS-2	0.34	0.038	2.46	0.34	0.028	0.34
FOS-3	0.89	0.073	2.96	0.93	0.032	0.35
FOS-4	1.05	0.087	2.83	1.10	0.031	0.35
FOS-5	1.56	0.143	2.70	1.63	0.030	0.39

As necessary, make improvements to your price versus time constant (τ) regression model in **Project_M4Regression_sss_tt.m**. Complete the price versus tau regression analysis on the 100 data sets using your M3 algorithm and your M4 algorithm. Generate a regression plot for your M3 algorithm and your M4 algorithm. Report the results of each model in Table 4.

Learning Objective (LO): 12.00 Perform linear regression
Learning Objective (LO): 13.00 Perform function discovery and data transformations
Learning Objective (LO): 07.00 Create and evaluate x-y plots suitable for technical presentation (this includes all appropriate sub-LOs)

Regression plot for Milestone 3



Regression plot for Milestone 4

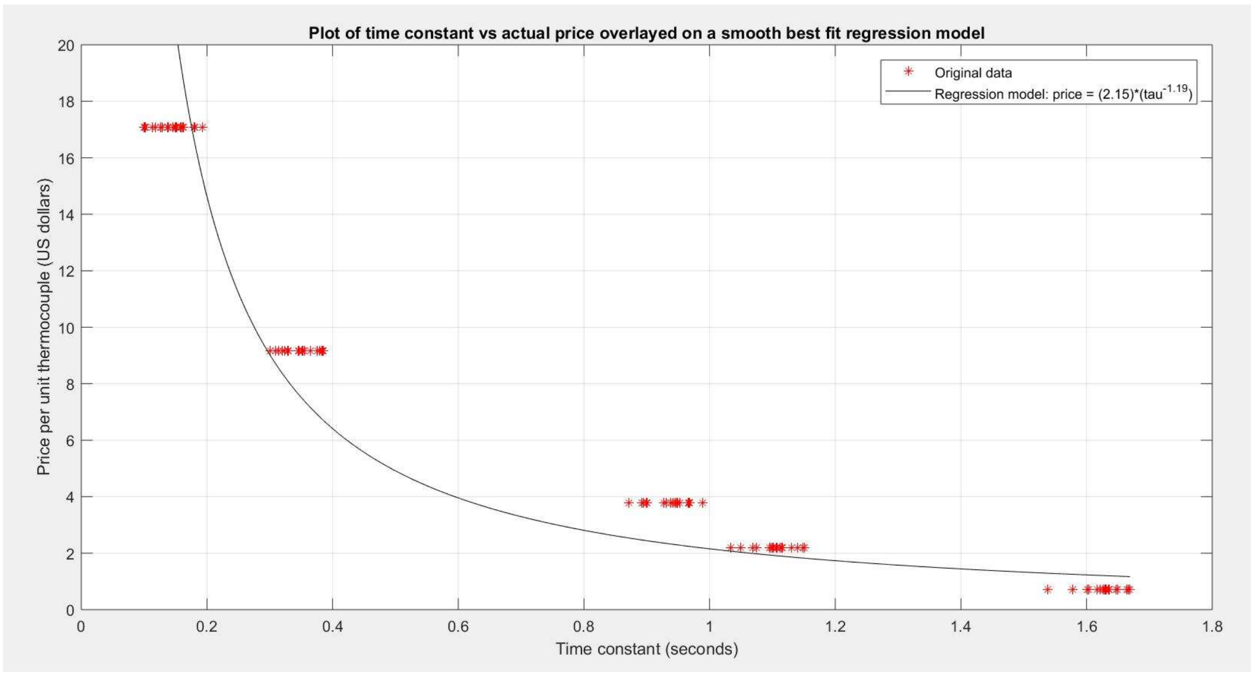


Table 4. Algorithm performance comparison for price versus time constant regression models

Regression Result	M3 Algorithm	M4 Algorithm
General Equation	price = 2.05*tau ^ (-1.19)	price = 2.15*tau ^ (-1.19)
SSE (degF^2)	2563.474	1368.415
SST (degF^2)	7687.293	7333.257
r^2	0.797	0.813

Refinement Category 2: Algorithm Efficiency

If you have refined the efficiency of your code, complete Table 5 below to show the effects of your refinements. Use the MATLAB built-in functions `tic` and `toc` to measure how long it takes your code to execute. *Efficiency refinements must be clearly commented in your code with the text Category 2 AND adequate description. Do not remove code; comment out unnecessary code and comment on the change. New code must be designated as such.*

Table 5. Efficiency measurement results.

Algorithm	**Execution Time (sec)**
M3 Algorithm	31.24
M4 Algorithm	1.40

Refinement Category 3: Algorithm Insight

If you have refined the robustness and performance of your algorithm in light of changes in a thresholding or other variable hardcoded in your algorithm, create one or more plots that illustrate the insights you have gained. The plot(s) should be suitable for technical presentation and clearly illustrate the effect of changes on the parameter identification and/or regression results. Write a paragraph that complements the plot(s). This paragraph must clearly describe changes to the thresholding or other variables hardcoded in your algorithm and the insights you gained. *The variables used in this analysis must be clearly commented in your code with the text Category 3 AND adequate description.*