CruiseAuto Project – Milestone 4

ANSWER SHEET: Algorithm Refinement and Final Deliverables

Table of Contents

Part 1. Assignment Header

Section and Team ID: 011_03

Team Member Name	Purdue Career Account Login	Programmer Number	Detailed Description of the Work	Percent of Work
Peter Teal	pteal	Programmer 4	Completed part 4a, comparing the results of our improved algorithm to both the values we got in M3 as well as the benchmark values. Improved the initial velocity calculating function by accounting for error in the acceleration start time Wrote the introduction for the technical brief, as well as the description of the parameter identification process and helped format the technical brief Wrote the resume insert	25%
Justin Clark	clar1062	Programmer 2	This programmer worked on part 3 to identify certain parameters to target for algorithm improvement. This programmer also worked on the technical brief.	25%
Ethan Zhang	zhang5173	Programmer 1	Completed Part 4b and 4c, plotting the performance of each of the 9 tests using our algorithm and the provided boundaries, this programmer also worked on completing	25%

			parts 2 and 3 of the answer sheet. Lastly, this programmer worked on improving the smoothing function in our algorithm to predict the parameters more effectively while taking less time.	
John Soares	soaresj	Programmer 3	Completed part 5 discussing the performance of the algorithm through an overall perspective, looking at the shortcomings and strengths of the algorithm. Completed the interpretation of results in the technical brief, analyzing the reasons as to the errors of the algorithm and a conclusive statement to the client. Attempted a certain number of windows to reduce the error percentage in certain parts of the code but had no success.	25%

Part 2. Milestone 3 Feedback and Reflection

Strength: A notable strength of our algorithm is that it can smooth the data extremely well and it is somewhat accurate for calculating the parameters for each of the three car types in the benchmark data set. Another strength of our algorithm is that the code is structured and commented on very well and therefore can be easily understood and edited by other people including our clients.

Limitation: The limitations within our code were few but very avoidable. We lost points on figure management, managing the decimal places of our data, lack of metrics provided within our improvement strategies, and the lack of commenting and formatting of our code. One limitation of our algorithm is that since it has a time complexity of O(n^2), it is extremely computationally intensive and takes a long time to run, another limitation of our algorithm is that we tuned it specifically for this shape of dataset, however if used on data with differing shapes, it may not perform as well. Lastly, it had a high error for one of the benchmark tests at 30+%, meaning it didn't predict the acceleration time and the other parameters very well.

How could the feedback from M3 lead to improvements? This feedback gives a clear explanation of where we went wrong and where we went right. We plan on keeping our code easy to read and understand and keep our smoothing algorithm as it is the backbone of our model. We can also use our strength of having low error for two of the benchmark data sets to figure out how to make the third more accurate and

precise. We can use our limitations as a perfect starting point to improve our algorithm. We can also use our limitations to make sure when we submit the new algorithm we do not lack in the areas where we had previously fell short.

What concrete steps will you take to incorporate the M3 feedback to improve your algorithm?

First, we will figure out where our code has limitations and figure out how the algorithm falls short. After looking through our algorithm, we decided to change the model behind our gaussian kernel to w/7 from w/3, by tightening our filter from w/3 down to w/7, we made the filter more local and less aggressive, trading some noise reduction for much better alignment of that initial rise. And when running this, we say a much more accurate representation of the benchmark data and our algorithm had <2% error for all three tests.

Part 3. Algorithm Improvement Plan

Improvement #1

Parameter(s) Targeted: Acceleration Start Time

Describe the improvement that your team is introducing to the M3 algorithm: To improve the acceleration start time we get from our algorithm, we changed subfunction 3 to adapt better to the data. To do this, we changed the window size of which we average over values to be a larger window, this is because to improve the runtime of our program, we had to decrease the number of smoothening passes we take, so since the data is less smooth, there is more noise in it and to combat this, we have to use a larger window size to average the derivative of the data to determine when the acceleration starts, now for the Compact Hatchback, we now get an Estimated start time of 6.12s, Reference: 6.21, Error: 1.45%, for the Midsize Sedan, we get an estimated start time of 9.30s, Reference: 9.39, Error: 0.96%, and for the SUV, we get an Estimated start time of 6.79s, Reference: 6.85, Error: 0.88%. This is a significant improvement from M3, where we had an average of 3.10% error throughout the three benchmark data sets.

Why are these refinements necessary and how will they improve your algorithm?: This refinement of sub3 is necessary because this helps us more accurately determine where the acceleration start time is so we can better predict the other parameters. This is also very important because any timing error directly skews subsequent estimates of the time constant and steady state gain. By tightening our ts detection, we reduce SSE and improve the repeatability of parameter estimates when we use the larger datasets, this yields more reliable first order models that consistently match the real vehicle responses.

Improvement #2

Parameter(s) Targeted: Time Constant

Describe the improvement that your team is introducing to the M3 algorithm: To improve the time constant, in addition to the changes we made to the calculation of the acceleration start time, we changed subfunction 2, the gaussian window function. In order to improve this function, we changed the function for our gaussian kernel and made the window in which we perform the weighted average smaller, since our data is very smooth, so by making the window from 11 to 3, our function to detect the acceleration start time can detect when the accelerate start more accurately instead of being skewed. Additionally, we changed the function we use for the gaussian kernel from w/3 to w/7. Which means that the function is now much less aggressive and make the bell curve more narrow (McGuire, 2024), so the weights drop off very quickly and this really only averages the nearest 1-2 points instead of 5-6 from M3, so this trades a bit of noise reduction for much better alignment of the model's exponential fit window, which in turn improves our time constant prediction accuracy. And we can see this improvement as for the Midsize Sedan, the Estimated tau: 2.10, Reference: 1.96, Error: 7.14%, for the SUV, we get a Estimated tau: 2.95, Reference: 2.80, Error: 5.36%, and for the Compact Hatchback, we get Estimated tau: 1.59, Reference: 1.51, Error: 5.30%. This is a significant improvement from the average Time constant error from M3 of 16.4%.

Why are these refinements necessary and how will they improve your algorithm?: This refinement of Subfunction 2 is necessary because our original 11-point, wide tail Gaussian was too aggressive as it was blurring out the true exponential rise and systematically biasing our start time estimates. By shrinking the window to just 3 points and tightening σ from w/3 to w/7, we preserve the shape of the mid response region, which lowers our modified SSE and cuts down the run-to-run variability in the time start (McGuire, 2024). This change drives our average error down from 16.4% to under 6%, giving us far more accurate and repeatable time-constant predictions across all three vehicle data sets. This will also be the case if we use the larger dataset since the function now adapts to all data sets better. (McGuire, 2024)

Part 4. Algorithm Refinements Implementation and Results

4a. Refinement Results with Benchmark Data

Table 4a.1 - Results for Compact Hatchback Benchmark Data

Parameter	Benchmark Values	M3 Algorithm Values	M3 Percent Error	M4 Algorithm Values	M4 Percent Error
Acceleration start time [s]	6.21	5.86	5.64%	6.12	1.45%
Time constant [s]	1.51	1.98	31.31%	1.63	7.95%
Initial speed [m/s]	-0.09	-0.13	41.10%	-0.08	15.58%
Final speed [m/s]	25.08	24.97	0.45%	24.97	0.45%

Table 4a.2 - Results for Midsize Sedan Benchmark Data

Parameter	Benchmark Values	M3 Algorithm Values	M3 Percent Error	M4 Algorithm Values	M4 Percent Error
Acceleration start time [s]	9.39	9.17	2.34%	9.30	0.96%
Time constant [s]	1.96	2.29	16.84%	2.05	4.95%
Initial speed [m/s]	-0.22	-0.26	20.30%	-0.25	15.79%
Final speed [m/s]	24.72	24.60	0.49%	24.60	0.48%

Table 4a.3 - Results for SUV Benchmark Data

Parameter	Benchmark Values	M3 Algorithm Values	M3 Percent Error	M4 Algorithm Values	M4 Percent Error
Acceleration start time [s]	6.94	6.85	1.31%	6.79	0.88%
Time constant [s]	2.77	2.80	1.07%	2.91	3.93%
Initial speed [m/s]	0.19	0.23	21.48%	0.17	11.75%
Final speed [m/s]	24.18	24.11	0.28%	24.11	0.28%

Table 4a.4 – Results for SSE_{mod}

Vehicle	SSE_{mod} from Benchmark Parameters (Part 4a of M3)	SSE_{mod} from M3 Algorithm Parameters (Part 4b of M3)	SSE_{mod} from M4 Algorithm Parameters
Compact Hatchback	0.583	0.784	0.583
Midsize Sedan	0.508	0.578	0.509
Large SUV	0.509	0.505	0.504

4b. Refinement Results with Experimental Data

Table 4b.1 – M3 and M4 Algorithm Comparison of Experimental Data Parameters

Vehicle	Tire Type	M3 Algorithm Parameters				M4 Algorithm Parameters			
		Start time [s]	Time constant [s]	Initial speed [m/s]	Final speed [m/s]	Start time [s]	Time constant [s]	Initial speed [m/s]	Final speed [m/s]
Compact Hatchback	Summer	8.14	2.84	0.12	25.06	8.10	2.78	-0.03	25.06
Compact Hatchback	All-Season	7.58	2.18	0.11	24.92	7.76	1.87	0.00	24.92
Compact Hatchback	Winter	7.46	1.96	0.08	24.67	7.74	1.57	0.00	24.67
Midsize Sedan	Summer	8.31	2.83	0.11	24.95	8.28	2.80	-0.04	24.94
Midsize Sedan	All-Season	6.03	2.26	0.14	24.89	6.19	2.01	0.00	24.89
Midsize Sedan	Winter	7.53	1.98	0.08	24.95	7.81	1.56	0.00	24.95
Large SUV	Summer	15.93	2.67	3.52	23.64	7.65	3.77	-0.03	23.64
Large SUV	All-Season	7.25	2.59	0.14	24.03	7.27	2.46	-0.02	24.03
Large SUV	Winter	7.35	2.91	0.11	24.06	7.45	3.07	-0.02	24.06

4c. Performance Check with Experimental Parameters

Table 4c.1 – Performance Boundary Results

Vehicle	Tire Type	Within Bounds or Outside Bounds?
Compact Hatchback	Summer	Within Bounds
Compact Hatchback	All-Season	Within Bounds
Compact Hatchback	Winter	Within Bounds
Midsize Sedan	Summer	Within Bounds
Midsize Sedan	All-Season	Within Bounds
Midsize Sedan	Winter	Within Bounds
Large SUV	Summer	Within Bounds
Large SUV	All-Season	Within Bounds
Large SUV	Winter	Within Bounds

Part 5. Algorithm Performance Discussion

Do you believe your algorithm accurately reflects the true performance of the system? Why or why not?

Yes, the algorithm does accurately reflect the true performance of the system through a purely analytical lens. This has been consistently highlighted with the low error percentages regarding the parameter values given by the algorithm that surpassed no more than 15.79% in a single parameter value (Initial Speed, Midsize Sedan) and went as low as 0.45% (Final Speed, Compact Hatchback). Although the percentage error of 15.79% is not that low, within the same test (including same tire type and car) considering the values are so small, even a small deviation would cause a large increase in % error. All the other values had an error percentage of no more than 4.95%, suggesting that although there was some inaccuracy to one parameter value the holistic view towards the algorithm's accuracy is extremely positive. Thus, the accuracy of the data strengthens the fact that the algorithm does reflect the true performance of the system given the experimental data. The accuracy of the algorithm is further supported by the low SSE values, ranging from 0.583 to 0.504. Although the values are not at 0, the low values and closeness to the ideal model suggest that the given model does reflect the actual results to a good degree (Valchanov 2024)

, serving as more evidence as to the accuracy of the system and its reflection of the system's true performance. Furthermore, the performance boundary results emphasize how accurately the algorithm reflects the true performance of the system as all the values have been shown to fall within the boundaries set by the client. With all our experimental parameters fall within the given boundaries, we can see that due to the improvement of our algorithm after comparing with the benchmark dataset, we are now able to accurately model the real data.

Does your algorithm need more work that you are unable to complete because of the due date? If yes, describe what you would do. If no, justify why your analysis is complete as-is.

Although the algorithm has shown a great amount of accuracy, there are still some parts of the algorithm that could be improved still. This would mostly be associated with the inaccuracy of the initial speed values, which are consistently between 11.75% to 15.79% error in comparison to the actual benchmark data. To work on this a possible step would possibly implement a window that averages the velocity values only 0.5 seconds before the starting acceleration time. This could possibly reduce the margin of error as any difference in value will cause a large error percentage due to the small size of the values themselves. This would then be compared to the current values, and we would recalculate the percentage error to see if there was any change. Of course, the calculation for all percentage errors would have to be carried out as the influence of the initial velocity on the other parameters is also considered, and thus an overall analysis regarding the impact on the total error percentage will have to be carried out and thus a conclusion be mad upon this. This process would take a longer time to complete and thus, this is why we were not able to do so with the current time frame. Additionally, we believe that our algorithm has a few blockers, one notably being the time and space complexity, with our algorithm taking O(n^2) time. This can be very resource intensive as we move to larger datasets or need to run more smoothing passes, so one improvement we can make is choose a more efficient algorithm to smooth the data.

Does your technical brief reflect your critique of your algorithm’s performance? Remember, it is vital in engineering to accurately represent your work.

Yes, the technical brief does reflect a critique of the algorithm. The critique was done through several manners, firstly through the objective analysis of the algorithm’s runtime and percentage errors throughout M3 and M4. The comparison of data regarding both flaws in M3 and M4 was carried out and represented in the technical brief both in the introduction and in the interpretation of the results. This gave

an overall critique of the algorithm through the improvements made, and by a reference to the current results given by the algorithm such as the still alarming 15% error percentage in the initial velocity parameter. Thus, the critique regarding the algorithm's performance is made in the technical brief with the aim of acknowledging shortcomings and justifying improvements.

Part 6. Technical Brief

Use the template document provided. Upload all deliverables to Gradescope.

Part 7. Resumé Insert

MATLAB Programming Skills Acquired from ENGR 132

Tire ACC System Analysis, *Purdue University* Spring 2025

- → Programmed and tested an algorithm that effectively cleans data, eliminating outliers and errors in data readings
- → Identified key parameters from the data set such as starting velocity, ending velocity, acceleration start time, and time constant, with very low error
- → Designed a highly accurate model using the identified parameters that simulates given data sets with little error that fits within the bounds given by the client
- → Organized and explained results in an easy to digest technical brief with properly formatted figures and explanations of the processes used to reach the end results

Part 8. References

Lindeberg, T., "Discrete approximations of Gaussian smoothing and Gaussian derivatives," Journal of Mathematical Imaging and Vision, 66(5): 759–800, 2024.

Computer Vision Demonstration Website. Gaussian Filtering - Computer Vision Website Header. (n.d.). https://www.southampton.ac.uk/~msn/book/new_demo/gaussian/#:~:text=The%20Gaussian%20Smoothing%20Operator%20performs,how%20large%20the%20template%20is.

Mark Nixon & Alberto Aguado, 2002, Feature Extraction & Image Processing, Newnes

Valchanov, I. (2024, July 26). *Sum of squares: SST, SSR, SSE*. 365 Data Science. https://365datascience.com/tutorials/statistics-tutorials/sum-squares/

Nagaraj, K. (2024, December 10). Demystifying time Complexity in Programming: Understanding Efficiency analysis for optimal algorithm design | Karthikeyan Nagaraj. *Medium*. https://medium.com/infosecmatrix/demystifying-time-complexity-in-programming-understanding-efficiency-analysis-for-optimal-4a514d95fbc2

McGuire, T. (2024, November 23). On the Effectiveness of Moving Averages - Trevor McGuire - Medium. *Medium*. https://trevormcguire.medium.com/on-the-effectiveness-of-moving-averages-fb31b0c1ab6f

Part 9. MATLAB Built-in Functions

Fill out the following information **for each** MATLAB built-in function that your algorithm uses that was not explicitly taught in class. Add additional rows as needed. If you did not use any new built-in MATLAB functions, then delete the table below and write "No new function used."

Function Name & Call (include inputs/outputs)	Where in your algorithm do you use the function?	Describe the inputs needed to run the function.	Describe the outputs from the function.	Describe the theory and/or mathematics behind how the function operates.
`M4_main_011_03_Zhan5173 sprintf('Trial %d', trial));`	To plot	none	outputs to a string	No math behind this
`M4_main_011_03_Zhan5173 idx = t >= tsM4;`	To plot	none	creates a logical array of the same size as t	Using this since it is faster than using a nested loop